Design and control of an energy storage system for voltage flicker caused by clouds passing over photovoltaic systems

Matlab scripts

Lynrick Wix

Delft University of Technology

Design and control of an energy storage for voltage flicker caused by clouds passing over photovoltaic systems

Matlab scripts

by

Lynrick Wix

Student number 4534654

Project duration 14 February — November 9 2023

Thesis committee:			
	Prof. Dr. Ir Pavol Bauer	Delft University of Technology	Chairman
	Dr. Ir. Gautham Ram Chandra Mouli	Delft University of Technology	Supervisor
	Dr. Ir. Arjan van Voorden	Stedin	Supervisor
	Dr. Ir. Milos Cvetkovic	Delft University of Technology	External Member

External supervisor:		
	Ing. Henk Fidder	Stedin

STEDIN.NET

TU Delft Delft University of Technology

1

Simulation scripts

1.1. Voltage control simulation Matlab script

```
%% Load the data
clear all
worst_day_pv = readmatrix('C:\Users\lynri\Documents\MATLAB\Stedin Thesis 2023\
    worst_day_sensor249.csv');
opts = detectImportOptions('C:\Users\lynri\Documents\MATLAB\Stedin Thesis 2023\
    worst_day_sensor249.csv');
opts.SelectedVariableNames = {'Var2'};
time_data = readtable('C:\Users\lynri\Documents\MATLAB\Stedin Thesis 2023\
    worst_day_sensor249.csv',opts);
time_string = table2array(time_data);
power_rating = 3480;
efficiency_inverter = 0.94;
sensitivity = 0.00106737202547458;
% sensitivity = 0.001523221949563;
% sensitivity = 0.00202728792318498;
% sensitivity = 0.00350877192982469; %Sensitivity of E-bike charging station
%%
test_data = worst_day_pv(:,1);
sinus_MA_voltage_output_day = [];
sinus_MA_voltage_output = [];
supercapacitor_power_power_simpleMA = [];
window = [12];
for i = 1:length(window)
    sinus_MA_voltage_output_day(1:window(i),i) = test_data(1:window(i));
    for j = window(i)+1:length(test_data)
        sinus_MA_voltage_output_day(j,i) = sum(test_data(j-window(i):j-1))/(
            window(i));
    end
end
sinus_MA_voltage_output = sinus_MA_voltage_output_day;
delta_V = [];
supercapacitor_power_PCC = [];
for i = 1:length(sinus_MA_voltage_output)
    delta_V(i) = sinus_MA_voltage_output(i) - worst_day_pv(i,1);
    supercapacitor_power_PCC(i) = delta_V(i)/sensitivity;
end
% supercapacitor_power_MA = [];
for j = 1:length(supercapacitor_power_PCC)
    if supercapacitor_power_PCC(j) < 0
        supercapacitor_power_MA(j) = supercapacitor_power_PCC(j)*
            efficiency_inverter;
    elseif supercapacitor_power_PCC(j) >= 0
```

```
        supercapacitor_power_MA(j) = supercapacitor_power_PCC(j)/
            efficiency_inverter;
    end
end
simple_layer_power = supercapacitor_power_MA;
simple_layer_energy = [];
for i = 1:length(simple_layer_power)
    if i == 1
            energy_used = 0;
            simple_layer_energy(i) = 0;
    else
            energy_used = (simple_layer_power(i) + simple_layer_power(i-1))
                *2*0.5;
            simple_layer_energy(i) = simple_layer_energy(i-1) - (energy_used)
                /3600;
    end
end
%% Filter to only when PV is operating
% indexes_pv_operation = find(time_string > '2017-08-04 05:00:00.00' &
    time_string < '2017-08-04 22:00:00.00');
test_data = worst_day_pv(:,1);
sinus_MA_voltage_output_day = [];
sinus_MA_voltage_output = [];
supercapacitor_power_power_simpleMA = [];
window = [12];
for i = 1:length(window)
    sinus_MA_voltage_output_day(1:window(i),i) = test_data(1:window(i));
    weighting_multiplier = [];
    weighting_multiplier = linspace(1,window(i),window(i));
    weighting_fraction = [];
    for k = 1:window(i)
        weighting_fraction(k) = cos((1/window(i))*k*pi/2)^2;
    end
    weighting_fraction = flip(weighting_fraction);
%     for j = indexes_pv_operation(1):indexes_pv_operation(end)
    for j = window(i)+1:length(test_data)
        sinus_MA_voltage_output_day(j,i) = sum(test_data(j-window(i):j-1).*
            weighting_fraction')/sum(weighting_fraction);
    end
%     sinus_MA_voltage_output_day = sinus_MA_voltage_output_day(
    indexes_pv_operation,i);
end
sinus_MA_voltage_output = sinus_MA_voltage_output_day;
delta_V = [];
supercapacitor_power_PCC = [];
for i = 1:length(sinus_MA_voltage_output)
    delta_V(i) = sinus_MA_voltage_output(i) - worst_day_pv(i,1);
    supercapacitor_power_PCC(i) = delta_V(i)/sensitivity;
end
for j = 1:length(supercapacitor_power_PCC)
    if supercapacitor_power_PCC(j) < 0
        supercapacitor_power_simpleMA(j) = supercapacitor_power_PCC(j)*
            efficiency_inverter;
    elseif supercapacitor_power_PCC(j) >= 0
        supercapacitor_power_simpleMA(j) = supercapacitor_power_PCC(j)/
            efficiency_inverter;
    end
end
first_layer_power = supercapacitor_power_simpleMA;
first_layer_energy = [];
for i = 1:length(first_layer_power)
```

```
    if i == 1
            energy_used = 0;
            first_layer_energy(i) = 0;
    else
            energy_used = (first_layer_power(i) + first_layer_power(i-1))*2*0.5;
            first_layer_energy(i) = first_layer_energy(i-1) - (energy_used)/3600;
    end
end
% sinus_MA_voltage_output(indexes_pv_operation) = sinus_MA_voltage_output_day;
%%
delta_v_data = [];
difference_samplerate = [];
ramprate_power = [];
ramprate_voltage = [];
difference_power_worst_day = [];

for i = 1:length(worst_day_pv(:,1))-1
    delta_v_data(i) = worst_day_pv(i+1,1) - worst_day_pv(i,1);
    difference_samplerate(i) = pyrunfile('sample_rate.py','d',x = char(
       time_string(i)), y = char(time_string(i+1)));
    ramprate_voltage(i) = delta_v_data(i);
end
visible_flicker_curve = [1 40/80 18/80 20/80 23/80 0.36 30/80 38/80 43/80 50/80
   55/80 62/80 73/80 95/80 123/80 170/80 200/80];
annoying_flicker_curve = [115/80 55/80 35/80 45/80 55/80 68/80 76/80 93/80 110/80
    135/80 160/80 184/80 207/80 255/80 310/80 403/80 480/80];
visible_flicker_curve_polygon = [0 visible_flicker_curve 0 0];
annoying_flicker_curve_polygon = [0 annoying_flicker_curve 0 0];
time_dips_seconds = [0.05 0.1 0.2 0.3 1 2 3 6 12 30 60 120 180 360 720 1800
   3600];
time_dips_seconds_polygon = [0.025 time_dips_seconds 4000 0.025];
difference_voltage_percentage_control = ramprate_voltage*100/230;
idx_after = inpolygon(difference_samplerate,abs(
   difference_voltage_percentage_control),time_dips_seconds_polygon,
   visible_flicker_curve_polygon);
ida_after = inpolygon(difference_samplerate,abs(
   difference_voltage_percentage_control),time_dips_seconds_polygon,
   annoying_flicker_curve_polygon);

find_indexes_pos_MA = find(~idx_after == 1);
find_indexes_neg_MA = find(~ida_after == 1);
actual_indexes_pos_MA = find_indexes_pos_MA+1;
actual_indexes_neg_MA = find_indexes_neg_MA+1;

%% Calculating delta V to then calculate how much power should the supercapacitor
    deliver
indexes_pv_operation = find(time_string > '2017-08-04 05:00:00.00' & time_string
   < '2017-08-04 22:00:00.00');
test_data = worst_day_pv(:,1);
sinus_MA_voltage_output_day = [];
sinus_MA_voltage_output = [];
supercapacitor_power_power_simpleMA = [];
window = [12];
for i = 1:length(window)
    %sinus_MA_voltage_output_day(1:window(i),i) = test_data(1:window(i));
    weighting_multiplier = [];
    weighting_multiplier = linspace(1,window(i),window(i));
    weighting_fraction = [];
    for k = 1:window(i)
        weighting_fraction(k) = cos((1/window(i))*k*pi/2)^2;
    end
```

```
    weighting_fraction = flip(weighting_fraction);
    for j = indexes_pv_operation(1):indexes_pv_operation(end)
%     for j = window(i)+1:length(test_data)
        sinus_MA_voltage_output_day(j,i) = sum(test_data(j-window(i):j-1).*
           weighting_fraction')/sum(weighting_fraction);
    end
    sinus_MA_voltage_output_day = sinus_MA_voltage_output_day(
       indexes_pv_operation,i);
end
% for i = 1:length(window)
%     %sinus_MA_voltage_output_day(1:window(i),i) = test_data(1:window(i));
% %     weighting_multiplier = [];
% %     weighting_multiplier = linspace(1,window(i),window(i));
% %     weighting_fraction = [];
% %     for k = 1:window(i)
% %         weighting_fraction(k) = cos((1/window(i))*k*pi/2)^2;
% %     end
% %     weighting_fraction = flip(weighting_fraction);
% %     for j = indexes_pv_operation(1):indexes_pv_operation(end)
%     for j = window(i)+1:length(test_data)
%         sinus_MA_voltage_output_day(j,i) = sum(test_data(j-window(i):j-1))/(
   window(i));
%     end
% %     sinus_MA_voltage_output_day = sinus_MA_voltage_output_day(
   indexes_pv_operation,i);
% end
% sinus_MA_voltage_output = sinus_MA_voltage_output_day;
sinus_MA_voltage_output = test_data;
sinus_MA_voltage_output(indexes_pv_operation) = sinus_MA_voltage_output_day;
delta_V = [];
supercapacitor_power_PCC = [];
for i = 1:length(sinus_MA_voltage_output)
    delta_V(i) = sinus_MA_voltage_output(i) - worst_day_pv(i,1);
    supercapacitor_power_PCC(i) = delta_V(i)/sensitivity;
end
for j = 1:length(supercapacitor_power_PCC)
    if supercapacitor_power_PCC(j) < 0
        supercapacitor_power_simpleMA(j) = supercapacitor_power_PCC(j)*
           efficiency_inverter;
    elseif supercapacitor_power_PCC(j) >= 0
        supercapacitor_power_simpleMA(j) = supercapacitor_power_PCC(j)/
           efficiency_inverter;
    end
end
second_layer_power = supercapacitor_power_simpleMA;
second_layer_energy = [];
for i = 1:length(second_layer_power)
    if i == 1
            energy_used = 0;
            second_layer_energy(i) = 0;
        else
            energy_used = (second_layer_power(i) + second_layer_power(i-1))
               *2*0.5;
            second_layer_energy(i) = second_layer_energy(i-1) - (energy_used)
               /3600;
    end
end


soc_range_simpleMA_SOC = zeros(length(supercapacitor_power_simpleMA),1);
soc_range_simpleMA_SOC(1) = 0;
```

```
total_energy_used_supercapacitor_simpleMA = max(cumtrapz(
   supercapacitor_power_simpleMA))*2;
soc_range_simpleMA = cumtrapz(supercapacitor_power_simpleMA)*2/(3600);
supercapacitor_power_PCC_SOC = [];
supercapacitor_power_simpleMA_SOC = [];
% third_layer_energy = [];
% fourth_layer_energy = [];

for i = 1:length(supercapacitor_power_simpleMA)
    Wh_range_acceptable = [-5 5];
%     correction_factor_slope_c = 0/20;
%     correction_factor_slope_d = 0.1/20;
    correction_factor_slope_c = 0.01;
    correction_factor_slope_d = 0.01;
    % SoC Deadzone Curve implementation
    if supercapacitor_power_simpleMA(i) > 0
        if soc_range_simpleMA_SOC(i) > Wh_range_acceptable(1) &&
           soc_range_simpleMA_SOC(i) < Wh_range_acceptable(end)
            supercapacitor_power_PCC_SOC(i) = supercapacitor_power_PCC(i);
            supercapacitor_power_simpleMA_SOC(i) = supercapacitor_power_simpleMA(
               i);
        elseif soc_range_simpleMA_SOC(i) < Wh_range_acceptable(1)
            delta_soc = soc_range_simpleMA_SOC(i) - Wh_range_acceptable(1);
            supercapacitor_power_PCC_SOC(i) = supercapacitor_power_PCC(i)*(1+abs(
               delta_soc)*correction_factor_slope_d);
            supercapacitor_power_simpleMA_SOC(i) = supercapacitor_power_simpleMA(
               i)*(1-abs(delta_soc)*correction_factor_slope_d);
        elseif soc_range_simpleMA_SOC(i) > Wh_range_acceptable(end)
            supercapacitor_power_PCC_SOC(i) = supercapacitor_power_PCC(i);
            supercapacitor_power_simpleMA_SOC(i) = supercapacitor_power_simpleMA(
               i);
        end
    else
        if soc_range_simpleMA_SOC(i) > Wh_range_acceptable(1) &&
           soc_range_simpleMA_SOC(i) < Wh_range_acceptable(end)
            supercapacitor_power_PCC_SOC(i) = supercapacitor_power_PCC(i);
            supercapacitor_power_simpleMA_SOC(i) = supercapacitor_power_simpleMA(
               i);
        elseif soc_range_simpleMA_SOC(i) < Wh_range_acceptable(1)
            supercapacitor_power_PCC_SOC(i) = supercapacitor_power_PCC(i);
            supercapacitor_power_simpleMA_SOC(i) = supercapacitor_power_simpleMA(
               i);
        elseif soc_range_simpleMA_SOC(i) > Wh_range_acceptable(end)
            delta_soc = soc_range_simpleMA_SOC(i) - Wh_range_acceptable(end);
            supercapacitor_power_PCC_SOC(i) = supercapacitor_power_PCC(i)*(1-abs(
               delta_soc)*correction_factor_slope_c);
            supercapacitor_power_simpleMA_SOC(i) = supercapacitor_power_simpleMA(
               i)*(1-abs(delta_soc)*correction_factor_slope_c);
        end
    end
%     third_layer_power(i) = supercapacitor_power_simpleMA_SOC(i);
%     if i == 1
%             energy_used = 0;
%             third_layer_energy(i) = 0;
%     else
%             energy_used = (third_layer_power(i) + third_layer_power(i-1))
   *2*0.5;
%             third_layer_energy(i) = third_layer_energy(i-1) - (energy_used)
   /3600;
%     end
    %Idle Charging
```

```
    if (supercapacitor_power_simpleMA_SOC(i) < 50 &&
       supercapacitor_power_simpleMA_SOC(i) > -50)
        if soc_range_simpleMA_SOC(i) > 5
            supercapacitor_power_simpleMA_SOC(i) = 50;
%             supercapacitor_power_simpleMA_SOC(i) =
   supercapacitor_power_simpleMA_SOC(i);
        elseif soc_range_simpleMA_SOC(i) < -5
            supercapacitor_power_simpleMA_SOC(i) = -100;
%             supercapacitor_power_simpleMA_SOC(i) =
   supercapacitor_power_simpleMA_SOC(i);
        end
    else
        supercapacitor_power_simpleMA_SOC(i) = supercapacitor_power_simpleMA_SOC(
           i);
    end
%     if time_string(i) < '2017-08-04 05:00:00.00' | time_string(i) > '2017-08-04
    19:00:00.00'
%         supercapacitor_power_PCC_SOC(i) = 0;
%         supercapacitor_power_simpleMA_SOC(i) = 0;
%     end
%     supercapacitor_power_simpleMA_SOC(indexes_pv_operation) =
   supercapacitor_power_simpleMA_SOC_extra;
%     supercapacitor_power_simpleMA_SOC = supercapacitor_power_simpleMA;
    if i == 1
        energy_used = 0;
        soc_range_simpleMA_SOC(i) = 0;
    else
        energy_used = (supercapacitor_power_simpleMA_SOC(i) +
           supercapacitor_power_simpleMA_SOC(i-1))*2*0.5;
        soc_range_simpleMA_SOC(i+1) = soc_range_simpleMA_SOC(i) - (energy_used)
           /3600;
    end

    if supercapacitor_power_simpleMA_SOC(i) < 0
        supercapacitor_power_PCC_SOC(i) = supercapacitor_power_simpleMA_SOC(i)/
           efficiency_inverter;
    elseif supercapacitor_power_simpleMA_SOC(i) >= 0
        supercapacitor_power_PCC_SOC(i) = supercapacitor_power_simpleMA_SOC(i)*
           efficiency_inverter;
    end
end

% energy_supercapacitor_Wh_simpleMA = (total_energy_used_suwpercapacitor_simpleMA
   - min(cumtrapz(supercapacitor_power_simpleMA))*2)/(60*60);
energy_supercapacitor_Wh_simpleMA_SOC = (max(soc_range_simpleMA_SOC)- min(
   soc_range_simpleMA_SOC));
% soc_range_simpleMA_SOC = cumtrapz(supercapacitor_power_simpleMA_SOC)*2/(3600);
maximum_charge_power_simpleMA = min(supercapacitor_power_simpleMA);
maximum_discharge_power_simpleMA = max(supercapacitor_power_simpleMA);
maximum_charge_power_simpleMA_SOC = min(supercapacitor_power_simpleMA_SOC);
maximum_discharge_power_simpleMA_SOC = max(supercapacitor_power_simpleMA_SOC);
supercapacitor_power_lag = [];
supercapacitor_power_lag(1) = 0;
for j = 2:length(worst_day_pv(:,1))
    supercapacitor_power_lag(j) = supercapacitor_power_PCC_SOC(1,j-1);
end
new_voltage = [];
for k = 1:length(window)
    for j = 1:length(worst_day_pv(:,1))
        new_voltage(k,j) = worst_day_pv(j,1) + sensitivity*
           supercapacitor_power_PCC_SOC(j);
```

```
%           new_voltage(k,j) = worst_day_pv(j,1) + sensitivity*
    supercapacitor_power_lag(j);
     end
end
new_voltage = new_voltage';
delta_v_data_SOC =[];
ramprate_voltage_SOC = [];
difference_samplerate_SOC = [];
for i = 1:length(new_voltage)-1
     delta_v_data_SOC(i) = new_voltage(i+1) - new_voltage(i);
     difference_samplerate_SOC(i) = pyrunfile('sample_rate.py','d',x = char(
        time_string(i)), y = char(time_string(i+1)));
     ramprate_voltage_SOC(i) = delta_v_data_SOC(i);
end
%% Making plot to see all the voltage violations with SOC control
find_indexes_ramprate_SOC = [];
violation_voltageramp_vcontrol_SOC = [];
find_indexes_ramprate_SOC = find(abs(ramprate_voltage_SOC) > 230*0.003);
violation_voltageramp_vcontrol_SOC = ramprate_voltage_SOC(
    find_indexes_ramprate_SOC);
figure
yyaxis right
plot(time_string(2:end),ramprate_voltage_SOC,'Color', 'b');
hold on
stem(time_string(~idx_after),ramprate_voltage_SOC(~idx_after), 'LineWidth', 1, '
    LineStyle','-','MarkerFaceColor','red','MarkerEdgeColor','red', 'Color','m');
ylabel('Voltage ramprate (V/s)')
hold on
yyaxis left
plot(time_string(2:end),pv_power_MA_extended_SOC(2:end),'Color',[73/255 76/255
    83/255])
ylabel('Power (W)')
title({'Voltage ramprate and power output at PCC with ESS implementation','
    Voltage Control with window of 6 elements'})
legend('Calculated power at PCC','Voltage fluctuation each second throughout the
    day','Voltage fluctuation exceeding the visible flicker threshold')
xlabel('Time')
grid on
%% Plotting all violations at PCC
find_indexes_ramprate = find(abs(ramprate_voltage) > 230*0.003);
violation_voltageramp_old = ramprate_voltage(find_indexes_ramprate);
figure
yyaxis right
plot(time_string(2:end),ramprate_voltage,'Color', 'b');
hold on
stem(time_string(~idx_before),ramprate_voltage(~idx_before), 'LineWidth', 1, '
    LineStyle','-','MarkerFaceColor','red','MarkerEdgeColor','red', 'Color','m');
ylabel('Voltage ramprate (V/s)')
hold on
yyaxis left
plot(time_string(2:end),worst_day_pv(2:end,2),'Color',[73/255 76/255 83/255])
ylabel('Power (W)')
title({'Voltage ramprate and power output at PCC without ESS implementation'})
legend('Power at PCC','Voltage fluctuation each second throughout the day','
    Voltage fluctuation exceeding the visible flicker threshold')
xlabel('Timestamps')
grid on
%%
full_range_SOC = [-50 -30 Wh_range_acceptable 30 50];
correction_values = [correction_factor_slope*(full_range_SOC(1)-
    Wh_range_acceptable(1)) correction_factor_slope*(full_range_SOC(2)-
```

```
    Wh_range_acceptable(1)) 0 0 correction_factor_slope*(full_range_SOC(5)-
    Wh_range_acceptable(2)) correction_factor_slope*(full_range_SOC(end)-
    Wh_range_acceptable(2))];
figure
plot(full_range_SOC,correction_values, 'Color',[51/255 206/255 224/255], '
    LineWidth', 2,'MarkerSize', 30, 'Marker', '.')
% hold on
grid on
% plot(time_string,soc_range_simpleMA_SOC, 'Color',[255/255 92/255 57/255])
ylabel('Correction value')
xlabel('Energy used from ESS (Wh)')
formatspec = "SOC correction deadzone curve, correction factor slope = %0.3f";
Title = compose(formatspec, correction_factor_slope);
title(Title)
%%
figure
plot(time_string,soc_range_simpleMA, 'Color',[51/255 206/255 224/255])
hold on
plot(time_string,soc_range_simpleMA_SOC, 'Color',[255/255 92/255 57/255])
formatspec = "Energy usage of supercapacitor using SOC control vs SOC control
    with idling charging";
Title = compose(formatspec);
title(Title)
%%
aruba_blue = [63/255 144/255 223/255];
aruba_red = [239/255 48/255 62/255];
aruba_yellow = [255/255 210/255 0/255];
royal_orange = [255/255 154/255 0/255];
stedin_grey = [77/255 77/255 77/255];
figure
plot(worst_day_pv(:,1),'LineWidth', 2,'Color',stedin_grey,'Marker','.','LineStyle
    ','-','MarkerSize',17)
hold on
plot(sinus_MA_voltage_output,'LineWidth',2,'Color',aruba_blue,'Marker','.','
    LineStyle','-','MarkerSize',17)
grid on
xlim([27670 27690])
xticklabels(0:5:50)
title({'Calculated moving average with customized weights voltage at PCC versus
    measured voltage at PCC','Window size = 6 elements'})
xlabel('Datapoint')
ylabel('Voltage (V)')
legend('Measured voltage at PCC','Voltage at PCC using moving average with
    customized weights','Location','Best')
%%
power_pcc = [];
for i = 1:length(supercapacitor_power_PCC_SOC)
    power_pcc(i) = worst_day_pv(i,2) + supercapacitor_power_PCC_SOC(i);
end
count = 0;
for j = 2:length(power_pcc)
    difference_power_check(j) = abs(power_pcc(j)-power_pcc(j-1))/2;
    if difference_power_check(j) > 800
        count = count + 1;
    end
end
%%

figure
plot(time_string,soc_range_simpleMA_SOC(2:end),'LineWidth',1,'Color',stedin_grey)
    ;
```

```
hold on
yline(5,'Color',aruba_red,'LineWidth',3,'Label',{'Upper ESS', 'energy threshold'
    }, 'LabelHorizontalAlignment','right')
hold on
yline(-5,'Color',aruba_red,'LineWidth',3,'Label',{'Lower ESS', 'energy threshold'
    }, 'LabelHorizontalAlignment','left','LabelVerticalAlignment','bottom')
xlabel('Time')
ylabel('Energy (Wh)')
grid on
ax = gca;
ax.GridAlpha = 0.4;
ax.GridLineStyle = "--";
ax.MinorGridLineStyle = "-";
ax.XColor = [0 0 0];
ax.YColor = [0 0 0];
ax.GridColor = [0 0 0];
title({'ESS energy usage throughout the day','Idle charge power = -100W','Idle
    discharge power = 50W'})
ylim([-15 10])
%%
f = figure;
f.Position = [50 50 850 500];
plot(time_string,simple_layer_energy,'LineWidth',1,'Color',stedin_grey);
hold on
plot(time_string,first_layer_energy,'LineWidth',1,'Color',aruba_blue);
hold on
plot(time_string,second_layer_energy,'LineWidth',1,'Color',royal_orange);
hold on
plot(time_string,third_layer_energy,'LineWidth',1,'Color',aruba_yellow);
hold on
plot(time_string,soc_range_simpleMA_SOC(2:end),'LineWidth',1,'Color',aruba_red);
xlabel('Time')
ylabel('Energy (Wh)')
grid on
ax = gca;
ax.GridAlpha = 0.4;
ax.GridLineStyle = "--";
ax.MinorGridLineStyle = "-";
ax.XColor = [0 0 0];
ax.YColor = [0 0 0];
ax.GridColor = [0 0 0];
legend_simple_cmp = 'Simple moving average, energy usage = %0.2f Wh';
legend_simple = compose(legend_simple_cmp,max(simple_layer_energy)-min(
    simple_layer_energy));
legend_first_cmp = 'Moving average with optimised weights, energy usage = %0.2f
    Wh';
legend_first = compose(legend_first_cmp,max(first_layer_energy)-min(
    first_layer_energy));
legend_second_cmp = 'Window of operation check, energy usage = %0.2f Wh';
legend_second = compose(legend_second_cmp,max(second_layer_energy)-min(
    second_layer_energy));
legend_third_cmp = 'SoC power reduction curve, energy usage = %0.2f Wh';
legend_third = compose(legend_third_cmp,max(third_layer_energy)-min(
    third_layer_energy));
legend_fourth_cmp = 'Idle charge and discharge, energy usage = %0.2f Wh';
legend_fourth = compose(legend_fourth_cmp,max(soc_range_simpleMA_SOC)-min(
    soc_range_simpleMA_SOC));
legend({string(legend_simple),string(legend_first),string(legend_second),string(
    legend_third),string(legend_fourth)})
title({'ESS energy usage throughout the day with each control layer applied','
    Voltage Control'})
```

```
% ylim([-15 10])
%% Plot Battery, PV POWER and AC voltage measurements
aruba_blue = [63/255 144/255 223/255];
aruba_red = [239/255 48/255 62/255];
aruba_yellow = [255/255 210/255 0/255];
royal_orange = [255/255 154/255 0/255];
stedin_grey = [77/255 77/255 77/255];
aruba_green = [132/255,194/255,37/255];

find_time = find(time_string > '2017-08-04 11:58:00.000' & time_string < '
    2017-08-04 12:04:00.000');

fig2 = figure;
fig2.Position = [50 50 950 600];
set(fig2,'defaultAxesColorOrder',[[0 0 0]; [0 0 0]]);
battery_power_zero = zeros(1,length(worst_day_pv(:,2)));
yyaxis left
ph = gobjects(1);
% ph(1) = plot(time_string,worst_day_pv(:,2),'LineWidth',2,'Color',aruba_blue);
% hold on
ph(1) = plot(time_string,battery_power_zero,'LineWidth',2,'Color',aruba_green,'
    LineStyle','-');
ylim([-2000 6000])
ylabel('Power (W)')
yyaxis right
hold on
ph(2) = plot(time_string,worst_day_pv(:,1),'LineWidth',2,'Color',stedin_grey);
hold on
for j = 1:length(find_indexes_pos_MA)
    ph(j+2) = plot(time_string([find_indexes_pos_MA(j), actual_indexes_pos_MA(j)
        ]),worst_day_pv([find_indexes_pos_MA(j), actual_indexes_pos_MA(j)],1),  '
        Marker', '.', 'MarkerSize', 20,'Color',royal_orange, 'LineWidth', 2,'
        LineStyle','-');
    hold on
end
for j = 1:length(find_indexes_neg_MA)
    ph(j+length(find_indexes_pos_MA)+2) = plot(time_string([find_indexes_neg_MA(j
        ), actual_indexes_neg_MA(j)]),worst_day_pv([find_indexes_neg_MA(j),
        actual_indexes_neg_MA(j)],1), 'Marker', '.', 'MarkerSize', 25, 'Color',
        aruba_red, 'LineWidth', 3,'LineStyle','-');
    hold on
end
if length(find_indexes_pos_MA) > 0
    if length(find_indexes_neg_MA) > 0
        legend(ph([1,2,3,length(find_indexes_neg_MA)+length(find_indexes_pos_MA)
            +2]),'ESS Power','Voltage at PCC','Measured Visible Flicker Moment','
            Measured Annoying Flicker Moment','Location', 'northeast');
    else
        legend(ph([1,2,3]),'ESS Power','Voltage at PCC','Measured Visible Flicker
             Moment','Location', 'Best');
    end
else
    if length(find_indexes_neg_MA) > 0
        legend(ph([1,2,3]),'ESS Power','Voltage at PCC','Measured Annoying
            Flicker Moment','Location', 'Best');
    else
        legend(ph([1,2]),'ESS Power','Voltage at PCC','Location', 'Best');
    end
end
ylim([210 245])
ylabel('Voltage (V)')
```

```
grid on
ax = gca;
ax.GridAlpha = 0.4;
ax.GridLineStyle = "-";
ax.XColor = [0 0 0];
ax.GridColor = [0 0 0];
% title({string(first_line);setup;string(control_line);string(second_line)})
xlabel('Time')
xlim([time_string(find_time(1)) time_string(find_time(end))])
%% Plot Battery, PV POWER and AC voltage measurements
aruba_blue = [63/255 144/255 223/255];
aruba_red = [239/255 48/255 62/255];
aruba_yellow = [255/255 210/255 0/255];
royal_orange = [255/255 154/255 0/255];
stedin_grey = [77/255 77/255 77/255];
aruba_green = [132/255,194/255,37/255];

find_time = find(time_string > '2017-08-04 11:58:00.000' & time_string < '
    2017-08-04 12:04:00.000');

fig2 = figure;
fig2.Position = [50 50 950 600];
set(fig2,'defaultAxesColorOrder',[[0 0 0]; [0 0 0]]);
battery_power_zero = zeros(1,length(worst_day_pv(:,2)));
yyaxis left
ph = gobjects(1);
% ph(1) = plot(time_string,worst_day_pv(:,2),'LineWidth',2,'Color',aruba_blue);
% hold on
ph(1) = plot(time_string,supercapacitor_power_PCC_SOC,'LineWidth',2,'Color',
    aruba_green,'LineStyle','-');
ylim([-2000 6000])
ylabel('Power (W)')
yyaxis right
hold on
ph(2) = plot(time_string,new_voltage,'LineWidth',2,'Color',stedin_grey);
hold on
% for j = 1:length(find_indexes_pos_MA)
%     ph(j+3) = plot(time_string([find_indexes_pos_MA(j), actual_indexes_pos_MA(j
    )]),new_voltage([find_indexes_pos_MA(j), actual_indexes_pos_MA(j)]),  'Marker
    ', '.', 'MarkerSize', 20,'Color',royal_orange, 'LineWidth', 2,'LineStyle','-')
    ;
%     hold on
% end
% for j = 1:length(find_indexes_neg_MA)
%     ph(j+length(find_indexes_pos_MA)+3) = plot(time_string([find_indexes_neg_MA
    (j), actual_indexes_neg_MA(j)]),new_voltage([find_indexes_neg_MA(j),
    actual_indexes_neg_MA(j)]), 'Marker', '.', 'MarkerSize', 25, 'Color',aruba_red
    , 'LineWidth', 3,'LineStyle','-');
%     hold on
% end
% if length(find_indexes_pos_MA) > 0
%     if length(find_indexes_neg_MA) > 0
%         legend(ph([1,2,3,4,length(find_indexes_neg_MA)+length(
    find_indexes_pos_MA)+3]),'PV Power','ESS Power','Voltage at PCC','Measured
    Visible Flicker Moment','Measured Annoying Flicker Moment','Location', 'Best')
    ;
%     else
%         legend(ph([1,2,3,4]),'PV Power','ESS Power','Voltage at PCC','Measured
    Visible Flicker Moment','Location', 'Best');
%     end
% else
```

```
%     if length(find_indexes_neg_MA) > 0
%         legend(ph([1,2,3,4]),'PV Power','ESS Power','Voltage at PCC','Measured
    Annoying Flicker Moment','Location', 'Best');
%     else
%         legend(ph([1,2,3]),'PV Power','ESS Power','Voltage at PCC','Location',
    'Best');
%     end
% end
legend(ph([1,2]),'ESS Power','Voltage at PCC','Location', 'Northeast');
ylim([210 245])
ylabel('Voltage (V)')
grid on
ax = gca;
ax.GridAlpha = 0.4;
ax.GridLineStyle = "-";
ax.XColor = [0 0 0];
ax.GridColor = [0 0 0];
% title({string(first_line);setup;string(control_line);string(second_line)})
xlabel('Time')
xlim([time_string(find_time(1)) time_string(find_time(end))])
```

1.2. Power control simulation Matlab script

```
%% Load the data
clear all
worst_day_pv = readmatrix('C:\Users\lynri\Documents\MATLAB\Stedin Thesis 2023\
    worst_day_sensor249.csv');
opts = detectImportOptions('C:\Users\lynri\Documents\MATLAB\Stedin Thesis 2023\
    worst_day_sensor249.csv');
opts.SelectedVariableNames = {'Var2'};
time_data = readtable('C:\Users\lynri\Documents\MATLAB\Stedin Thesis 2023\
    worst_day_sensor249.csv',opts);
time_string = table2array(time_data);
power_rating = 3480;
efficiency_inverter = 0.94;
sensitivity = 0.00106737202547458;
% sensitivity = 0.001523221949563;
% sensitivity = 0.00202728792318498;
%%
test_data = worst_day_pv(:,2);
sinus_MA_voltage_output_day = [];
sinus_MA_voltage_output = [];
supercapacitor_power_power_simpleMA = [];
window = [30];
for i = 1:length(window)
    sinus_MA_voltage_output_day(1:window(i),i) = test_data(1:window(i));
    for j = window(i)+1:length(test_data)
        sinus_MA_voltage_output_day(j,i) = sum(test_data(j-window(i):j-1))/(
            window(i));
    end
end
supercapacitor_power_simpleMA_grid = sinus_MA_voltage_output_day;
supercapacitor_power_PCC = [];
for i = 1:length(supercapacitor_power_simpleMA_grid)
    supercapacitor_power_PCC(i) = supercapacitor_power_simpleMA_grid(i) -
        worst_day_pv(i,2);
end
% supercapacitor_power_MA = [];
for j = 1:length(supercapacitor_power_PCC)
    if supercapacitor_power_PCC(j) < 0
        supercapacitor_power_MA(j) = supercapacitor_power_PCC(j)*
```

```
            efficiency_inverter;
    elseif supercapacitor_power_PCC(j) >= 0
        supercapacitor_power_MA(j) = supercapacitor_power_PCC(j)/
            efficiency_inverter;
    end
end
simple_layer_power = supercapacitor_power_MA;
simple_layer_energy = [];
for i = 1:length(simple_layer_power)
    if i == 1
            energy_used = 0;
            simple_layer_energy(i) = 0;
    else
            energy_used = (simple_layer_power(i) + simple_layer_power(i-1))
                *2*0.5;
            simple_layer_energy(i) = simple_layer_energy(i-1) - (energy_used)
                /3600;
    end
end
%%
test_data = worst_day_pv(:,2);
sinus_MA_voltage_output_day = [];
sinus_MA_voltage_output = [];
supercapacitor_power_power_simpleMA = [];
sinus_MA_pv_output = [];
window = [30];
for i = 1:length(window)
    sinus_MA_pv_output(1:window(i),i) = test_data(1:window(i));
    weighting_multiplier = [];
    weighting_multiplier = linspace(1,window(i),window(i));
    weighting_fraction = [];
    for k = 1:window(i)
        weighting_fraction(k) = cos((1/window(i))*k*pi/2)^2;
    end
    weighting_fraction = flip(weighting_fraction);
    for j = window(i)+1:length(test_data)
        sinus_MA_pv_output(j,i) = sum(test_data(j-window(i):j-1).*
            weighting_fraction')/sum(weighting_fraction);
    end
end
supercapacitor_power_MA_grid = sinus_MA_pv_output;
supercapacitor_power_PCC = [];
for i = 1:length(supercapacitor_power_simpleMA_grid)
    supercapacitor_power_PCC(i) = supercapacitor_power_simpleMA_grid(i) -
        worst_day_pv(i,2);
end
% supercapacitor_power_MA = [];
for j = 1:length(supercapacitor_power_PCC)
    if supercapacitor_power_PCC(j) < 0
        supercapacitor_power_MA(j) = supercapacitor_power_PCC(j)*
            efficiency_inverter;
    elseif supercapacitor_power_PCC(j) >= 0
        supercapacitor_power_MA(j) = supercapacitor_power_PCC(j)/
            efficiency_inverter;
    end
end
first_layer_power = supercapacitor_power_MA;
first_layer_energy = [];
for i = 1:length(first_layer_power)
    if i == 1
            energy_used = 0;
```

```
            first_layer_energy(i) = 0;
    else
            energy_used = (first_layer_power(i) + first_layer_power(i-1))*2*0.5;
            first_layer_energy(i) = first_layer_energy(i-1) - (energy_used)/3600;
    end
end
%%
indexes_pv_operation = find(time_string > '2017-08-04 05:00:00.00' & time_string
   < '2017-08-04 22:00:00.00');
test_data = worst_day_pv(:,2);
sinus_MA_power_output_day = [];

delta_v_data = [];
difference_samplerate = [];
ramprate_power = [];
ramprate_voltage = [];
difference_power_worst_day = [];

for i = 1:length(worst_day_pv(:,2))-1
    delta_v_data(i) = worst_day_pv(i+1,1) - worst_day_pv(i,1);
    difference_samplerate(i) = pyrunfile('sample_rate.py','d',x = char(
       time_string(i)), y = char(time_string(i+1)));
    ramprate_voltage(i) = delta_v_data(i);
end
visible_flicker_curve = [1 40/80 18/80 20/80 23/80 0.36 30/80 38/80 43/80 50/80
   55/80 62/80 73/80 95/80 123/80 170/80 200/80];
annoying_flicker_curve = [115/80 55/80 35/80 45/80 55/80 68/80 76/80 93/80 110/80
    135/80 160/80 184/80 207/80 255/80 310/80 403/80 480/80];
visible_flicker_curve_polygon = [0 visible_flicker_curve 0 0];
annoying_flicker_curve_polygon = [0 annoying_flicker_curve 0 0];
time_dips_seconds = [0.05 0.1 0.2 0.3 1 2 3 6 12 30 60 120 180 360 720 1800
   3600];
time_dips_seconds_polygon = [0.025 time_dips_seconds 4000 0.025];
difference_voltage_percentage_control = ramprate_voltage*100/230;
idx_after = inpolygon(difference_samplerate,abs(
   difference_voltage_percentage_control),time_dips_seconds_polygon,
   visible_flicker_curve_polygon);
ida_after = inpolygon(difference_samplerate,abs(
   difference_voltage_percentage_control),time_dips_seconds_polygon,
   annoying_flicker_curve_polygon);

find_indexes_pos_MA = find(~idx_after == 1);
find_indexes_neg_MA = find(~ida_after == 1);
actual_indexes_pos_MA = find_indexes_pos_MA+1;
actual_indexes_neg_MA = find_indexes_neg_MA+1;


supercapacitor_power_simpleMA_grid = [];
window = [30];
for i = 1:length(window)
    sinus_MA_pv_output(1:window(i),i) = test_data(1:window(i));
    weighting_multiplier = [];
    weighting_multiplier = linspace(1,window(i),window(i));
    weighting_fraction = [];
    for k = 1:window(i)
        weighting_fraction(k) = cos((1/window(i))*k*pi/2)^2;
    end
    weighting_fraction = flip(weighting_fraction);
    for j = indexes_pv_operation(1):indexes_pv_operation(end)
        sinus_MA_pv_output(j,i) = sum(test_data(j-window(i):j-1).*
           weighting_fraction')/sum(weighting_fraction);
```

```
    end
    sinus_MA_power_output_day = sinus_MA_pv_output(indexes_pv_operation,i);
end
supercapacitor_power_simpleMA_grid = test_data;
supercapacitor_power_simpleMA_grid(indexes_pv_operation) =
   sinus_MA_power_output_day;


supercapacitor_power_PCC = [];
for i = 1:length(supercapacitor_power_simpleMA_grid)
    supercapacitor_power_PCC(i) = supercapacitor_power_simpleMA_grid(i) -
       worst_day_pv(i,2);
end
for j = 1:length(supercapacitor_power_PCC)
    if supercapacitor_power_PCC(j) < 0
        supercapacitor_power_simpleMA(j) = supercapacitor_power_PCC(j)*
           efficiency_inverter;
    elseif supercapacitor_power_PCC(j) >= 0
        supercapacitor_power_simpleMA(j) = supercapacitor_power_PCC(j)/
           efficiency_inverter;
    end
end

second_layer_power = supercapacitor_power_simpleMA;
second_layer_energy = [];
for i = 1:length(second_layer_power)
    if i == 1
            energy_used = 0;
            second_layer_energy(i) = 0;
    else
            energy_used = (second_layer_power(i) + second_layer_power(i-1))
               *2*0.5;
            second_layer_energy(i) = second_layer_energy(i-1) - (energy_used)
               /3600;
    end
end
soc_range_simpleMA = zeros(length(supercapacitor_power_simpleMA),1);
soc_range_simpleMA(1) = 0;
supercapacitor_power_PCC_SOC = [];
for i = 1:length(supercapacitor_power_PCC)
    % SoC Deadzone Curve implementation
    Wh_range_acceptable = [-5 5];
%     correction_factor_slope_c = 0/20;
%     correction_factor_slope_d = 0/20;
    correction_factor_slope_c = 0.01;
    correction_factor_slope_d = 0.01;
    if supercapacitor_power_simpleMA(i) > 0
        if soc_range_simpleMA(i) > Wh_range_acceptable(1) && soc_range_simpleMA(i
           ) < Wh_range_acceptable(end)
            supercapacitor_power_PCC_SOC(i) = supercapacitor_power_PCC(i);
            supercapacitor_power_simpleMA_SOC(i) = supercapacitor_power_simpleMA(
               i);
        elseif soc_range_simpleMA(i) < Wh_range_acceptable(1)
            delta_soc = soc_range_simpleMA(i) - Wh_range_acceptable(1);
            supercapacitor_power_PCC_SOC(i) = supercapacitor_power_PCC(i)*(1+abs(
               delta_soc)*correction_factor_slope_d);
            supercapacitor_power_simpleMA_SOC(i) = supercapacitor_power_simpleMA(
               i)*(1-abs(delta_soc)*correction_factor_slope_d);
        elseif soc_range_simpleMA(i) > Wh_range_acceptable(end)
            supercapacitor_power_PCC_SOC(i) = supercapacitor_power_PCC(i);
            supercapacitor_power_simpleMA_SOC(i) = supercapacitor_power_simpleMA(
```

```
                i);
        end
    else
        if soc_range_simpleMA(i) > Wh_range_acceptable(1) && soc_range_simpleMA(i
            ) < Wh_range_acceptable(end)
            supercapacitor_power_PCC_SOC(i) = supercapacitor_power_PCC(i);
            supercapacitor_power_simpleMA_SOC(i) = supercapacitor_power_simpleMA(
                i);
        elseif soc_range_simpleMA(i) < Wh_range_acceptable(1)
            supercapacitor_power_PCC_SOC(i) = supercapacitor_power_PCC(i);
            supercapacitor_power_simpleMA_SOC(i) = supercapacitor_power_simpleMA(
                i);
        elseif soc_range_simpleMA(i) > Wh_range_acceptable(end)
            delta_soc = soc_range_simpleMA(i) - Wh_range_acceptable(end);
            supercapacitor_power_PCC_SOC(i) = supercapacitor_power_PCC(i)*(1-abs(
                delta_soc)*correction_factor_slope_c);
            supercapacitor_power_simpleMA_SOC(i) = supercapacitor_power_simpleMA(
                i)*(1-abs(delta_soc)*correction_factor_slope_c);
        end
    end
%     third_layer_power(i) = supercapacitor_power_simpleMA_SOC(i);
%     if i == 1
%             energy_used = 0;
%             third_layer_energy(i) = 0;
%     else
%             energy_used = (third_layer_power(i) + third_layer_power(i-1))
    *2*0.5;
%             third_layer_energy(i) = third_layer_energy(i-1) - (energy_used)
    /3600;
%     end

%     Idle Charging
    if (supercapacitor_power_simpleMA_SOC(i) < 50 &&
        supercapacitor_power_simpleMA_SOC(i) > -50)
        if soc_range_simpleMA(i) > 5
            supercapacitor_power_simpleMA_SOC(i) = 50;
%             supercapacitor_power_simpleMA_SOC(i) =
    supercapacitor_power_simpleMA_SOC(i);
        elseif soc_range_simpleMA(i) < -5
            supercapacitor_power_simpleMA_SOC(i) = -100;
%             supercapacitor_power_simpleMA_SOC(i) =
    supercapacitor_power_simpleMA_SOC(i);
        else
            supercapacitor_power_simpleMA_SOC(i) =
                supercapacitor_power_simpleMA_SOC(i);
        end
    else
        supercapacitor_power_simpleMA_SOC(i) = supercapacitor_power_simpleMA_SOC(
            i);
    end

%     if time_string(i) < '2017-08-04 05:00:00.00' | time_string(i) > '2017-08-04
    19:00:00.00'
%         supercapacitor_power_PCC_SOC(i) = 0;
%         supercapacitor_power_simpleMA_SOC(i) = 0;
%     end
%     supercapacitor_power_simpleMA_SOC(i) = supercapacitor_power_simpleMA(i);
    if i == 1
        soc_range_simpleMA(i) = 0;
    else
        energy_used = (supercapacitor_power_simpleMA_SOC(i) +
```

```
            supercapacitor_power_simpleMA_SOC(i-1))*2*0.5;
        soc_range_simpleMA(i+1) = soc_range_simpleMA(i) - (energy_used)/3600;
    end

    if supercapacitor_power_simpleMA_SOC(i) < 0
        supercapacitor_power_PCC_SOC(i) = supercapacitor_power_simpleMA_SOC(i)/
            efficiency_inverter;
    elseif supercapacitor_power_simpleMA_SOC(i) >= 0
        supercapacitor_power_PCC_SOC(i) = supercapacitor_power_simpleMA_SOC(i)*
            efficiency_inverter;
    end
end

energy_supercapacitor_Wh_simpleMA_SOC = (max(soc_range_simpleMA)- min(
    soc_range_simpleMA));
maximum_charge_power_simpleMA = min(supercapacitor_power_simpleMA);
maximum_discharge_power_simpleMA = max(supercapacitor_power_simpleMA);
maximum_charge_power_simpleMA_SOC = min(supercapacitor_power_simpleMA_SOC);
maximum_discharge_power_simpleMA_SOC = max(supercapacitor_power_simpleMA_SOC);
%%
four_layer = struct;
four_layer.energy = energy_supercapacitor_Wh_simpleMA_SOC;
four_layer.soc_range = soc_range_simpleMA;
four_layer.max_charge = maximum_charge_power_simpleMA_SOC;
four_layer.max_discharge = maximum_discharge_power_simpleMA_SOC;
%%
power_control = struct;
power_control.one = one_layer;
power_control.two = two_layer;
power_control.three = three_layer;
power_control.four = four_layer;
save("power_control_all_layers","power_control");

%%
full_range_SOC = [-50 -30 Wh_range_acceptable 30 50];
correction_values = [correction_factor_slope*(full_range_SOC(1)-
    Wh_range_acceptable(1)) correction_factor_slope*(full_range_SOC(2)-
    Wh_range_acceptable(1)) 0 0 correction_factor_slope*(full_range_SOC(5)-
    Wh_range_acceptable(2)) correction_factor_slope*(full_range_SOC(end)-
    Wh_range_acceptable(2))];
figure
plot(full_range_SOC,correction_values, 'Color',[51/255 206/255 224/255], '
    LineWidth', 2,'MarkerSize', 30, 'Marker', '.')
% hold on
grid on
% plot(time_string,soc_range_simpleMA_SOC, 'Color',[255/255 92/255 57/255])
ylabel('Correction value')
xlabel('Energy used from ESS (Wh)')
formatspec = "SOC correction deadzone curve, correction factor slope = %0.3f";
Title = compose(formatspec, correction_factor_slope);
title(Title)
%%
figure
plot(time_string,soc_range_simpleMA, 'Color',[51/255 206/255 224/255])
hold on
plot(time_string,soc_range_simpleMA_SOC, 'Color',[255/255 92/255 57/255])
formatspec = "Energy usage of supercapacitor using SOC control vs SOC control
    with idling charging";
Title = compose(formatspec);
title(Title)
%% Calculate the new voltage
```

```
new_voltage = [];
pv_power = worst_day_pv(:,2);
pv_power_extended_SOC = [];
supercapacitor_power_lag = [];
supercapacitor_power_lag(1) = 0;
for j = 2:length(worst_day_pv(:,1))
    supercapacitor_power_lag(j) = supercapacitor_power_PCC_SOC(1,j-1);
end
for k = 1:length(window)
    for j = 1:length(worst_day_pv(:,1))
        new_voltage(k,j) = worst_day_pv(j,1) + sensitivity*
           supercapacitor_power_PCC_SOC(k,j);
        pv_power_extended_SOC(k,j) = pv_power(j) + supercapacitor_power_PCC_SOC(k
           ,j);
%         new_voltage(k,j) = worst_day_pv(j,1) + sensitivity*
   supercapacitor_power_lag(k,j);
%         pv_power_extended_SOC(k,j) = pv_power(j) + supercapacitor_power_lag(k,j
   );
    end
end
new_voltage = new_voltage';
difference_samplerate_SOC = [];
for k = 1:length(window)
    for j = 2:length(worst_day_pv(:,1))
        difference_samplerate_SOC(j) = pyrunfile('sample_rate.py','d',x = char(
           time_string(j)), y = char(time_string(j+1)));
        delta_v_data_SOC(j) = new_voltage(j+1) - new_voltage(j);
        ramprate_voltage_SOC(j) = delta_v_data_SOC(j);
    end
end
% visible_flicker_curve = [1 40/80 18/80 20/80 23/80 0.36 30/80 38/80 43/80 50/80
    55/80 62/80 73/80 95/80 123/80 170/80 200/80];
% annoying_flicker_curve = [115/80 55/80 35/80 45/80 55/80 68/80 76/80 93/80
   110/80 135/80 160/80 184/80 207/80 255/80 310/80 403/80 480/80];
% visible_flicker_curve_polygon = [0 visible_flicker_curve 0 0];
% annoying_flicker_curve_polygon = [0 annoying_flicker_curve 0 0];
% time_dips_seconds = [0.05 0.1 0.2 0.3 1 2 3 6 12 30 60 120 180 360 720 1800
   3600];
% time_dips_seconds_polygon = [0.025 time_dips_seconds 4000 0.025];
% difference_voltage_percentage = ramprate_voltage_SOC*100/230;
% idx_after = inpolygon(difference_samplerate,abs(difference_voltage_percentage),
   time_dips_seconds_polygon, visible_flicker_curve_polygon);
% ida_after = inpolygon(difference_samplerate,abs(difference_voltage_percentage),
   time_dips_seconds_polygon, annoying_flicker_curve_polygon);
%
% find_indexes_pos_MA = find(~idx_after == 1);
% find_indexes_neg_MA = find(~ida_after == 1);
% actual_indexes_pos_MA = find_indexes_pos_MA+1;
% actual_indexes_neg_MA = find_indexes_neg_MA+1;

%% Making plot to see all the voltage violations with SOC control
find_indexes_ramprate_SOC = [];
violation_voltageramp_vcontrol_SOC = [];
find_indexes_ramprate_SOC = find(abs(ramprate_voltage_SOC) > 230*0.003);
violation_voltageramp_vcontrol_SOC = ramprate_voltage_SOC(
   find_indexes_ramprate_SOC);
figure
yyaxis right
plot(time_string(2:end),ramprate_voltage_SOC,'Color', 'b');
hold on
stem(time_string(~idx_after),ramprate_voltage_SOC(~idx_after), 'LineWidth', 1, '
```

```
    LineStyle','-','MarkerFaceColor','red','MarkerEdgeColor','red', 'Color','m');
ylabel('Voltage ramprate (V/s)')
hold on
yyaxis left
plot(time_string(2:end),pv_power_extended_SOC(2:end),'Color',[73/255 76/255
    83/255])
ylabel('Power (W)')
title({'Voltage ramprate and power output at PCC with ESS implementation','Power
    Control with window of 20 elements'})
legend('Calculated power at PCC','Voltage fluctuation each second throughout the
    day','Voltage fluctuation exceeding the visible flicker threshold')
xlabel('Time')
grid on
%% Plotting all violations at PCC
find_indexes_ramprate = find(abs(ramprate_voltage) > 230*0.003);
violation_voltageramp_old = ramprate_voltage(find_indexes_ramprate);
figure
yyaxis right
plot(time_string(2:end),ramprate_voltage,'Color', 'b');
hold on
stem(time_string(~idx_before),ramprate_voltage(~idx_before), 'LineWidth', 1, '
    LineStyle','-','MarkerFaceColor','red','MarkerEdgeColor','red', 'Color','m');
ylabel('Voltage ramprate (V/s)')
hold on
yyaxis left
plot(time_string(2:end),worst_day_pv(2:end,2),'Color',[73/255 76/255 83/255])
ylabel('Power (W)')
title({'Voltage ramprate and power output at PCC without ESS implementation'})
legend('Power at PCC','Voltage fluctuation each second throughout the day','
    Voltage fluctuation exceeding the visible flicker threshold')
xlabel('Timestamps')
grid on
%%
power_pcc = [];
for i = 1:length(supercapacitor_power_PCC_SOC)
    power_pcc(i) = worst_day_pv(i,2) + supercapacitor_power_PCC_SOC(i);
end
count = 0;
for j = 2:length(power_pcc)
    difference_power_check(j) = abs(power_pcc(j)-power_pcc(j-1))/2;
    if difference_power_check(j) > 400
        count = count + 1;
    end
end
%%
find_sunrise = find(time_string > '2017-08-04 05:00:00.00');
sunrise = find_sunrise(1);
find_sunset = find(time_string > '2017-08-04 22:00:00.00');
sunset = find_sunset(1);
%%
aruba_blue = [63/255 144/255 223/255];
aruba_red = [239/255 48/255 62/255];
aruba_yellow = [255/255 210/255 0/255];
royal_orange = [255/255 154/255 0/255];
stedin_grey = [77/255 77/255 77/255];
figure
plot(test_data,'LineWidth', 2,'Color',stedin_grey,'Marker','.','LineStyle','-','
    MarkerSize',17)
hold on
plot(supercapacitor_power_simpleMA_grid,'LineWidth',2,'Color',aruba_blue,'Marker'
    ,'.','LineStyle','-','MarkerSize',17)
```

```
grid on
xlim([27350 27400])
xticklabels(0:5:50)
title({'Calculated moving average with customized weights power at PCC versus
    power output of PV system','Window size = 30 elements'})
xlabel('Datapoint')
ylabel('Power (W)')
legend('Power output of PV system','Power at PCC using moving average with
    customized weights','Location','Best')
%%
figure
% a = area([time_string(sunrise), time_string(sunrise), time_string(sunset),
    time_string(sunset)], [-100 4000 4000 -100]);
a.FaceColor = [255/255 154/255 0/255];
a.FaceAlpha = 0.3;
hold on
plot(time_string,worst_day_pv(:,2),'LineWidth',1,'Color',[77/255 77/255 77/255])
% hold on
% xline(time_string(sunrise),'Color',[255/255 154/255 0/255],'LineWidth',3,'Label
    ','Lower bound time for summer operating window', 'LabelHorizontalAlignment','
    left');
% hold on
% xline(time_string(sunset), 'Color',[255/255 154/255 0/255],'LineWidth',3,'Label
    ','Upper bound time for summer operating window', 'LabelHorizontalAlignment','
    right');
grid on
% xlabel('Time')
% ylabel('Power (W)')
titlestring = compose('Output power from PV system on %d %s %d with window of
    operation',day(time_string(end)),string(month(time_string(end),'name')),year(
    time_string(end)));
% title(titlestring)
ax = gca;
ax.GridAlpha = 0.4;
ax.GridLineStyle = ":";
ax.MinorGridLineStyle = "--";
ax.XColor = [0 0 0];
ax.YColor = [0 0 0];
ax.GridColor = [0 0 0];
set(gca,'XColor', 'none','YColor','none')
set(gca, 'color', 'none');
ylim([0 3500])
%%
%%
f = figure;
f.Position = [50 50 950 600];
% a = area([time_string(sunrise), time_string(sunrise), time_string(sunset),
    time_string(sunset)], [-100 4000 4000 -100]);
% a.FaceColor = [255/255 154/255 0/255];
% a.FaceAlpha = 0.3;
% hold on
plot(summer_data{3,42},summer_data{1,42},'LineWidth',1,'Color',[77/255 77/255
    77/255])
% hold on
% xline(time_string(sunrise),'Color',[255/255 154/255 0/255],'LineWidth',3,'Label
    ','Lower bound time for summer operating window', 'LabelHorizontalAlignment','
    left');
% hold on
% xline(time_string(sunset), 'Color',[255/255 154/255 0/255],'LineWidth',3,'Label
    ','Upper bound time for summer operating window', 'LabelHorizontalAlignment','
    right');
```

```
grid on
xlabel('Time')
ylabel('Power (W)')
% titlestring = compose('Output power from PV system on %d %s %d',day(spring_data
    {3,67}(end)),string(month(spring_data{3,67}(end),'name')),year(spring_data
    {3,42}(end)));
% title(titlestring)
ax = gca;
ax.GridAlpha = 0.4;
ax.GridLineStyle = "--";
ax.MinorGridLineStyle = "-";
ax.XColor = [0 0 0];
ax.YColor = [0 0 0];
ax.GridColor = [0 0 0];
ylim([0 4000])
% xlim([datenum('2017-08-04 00:00:00.00') datenum('2017-08-05 00:00:00.00')])
%%
figure
plot(time_string,soc_range_simpleMA(2:end),'LineWidth',1,'Color',stedin_grey);
hold on
yline(5,'Color',aruba_red,'LineWidth',3,'Label','Upper ESS energy threshold', '
    LabelHorizontalAlignment','right')
hold on
yline(-5,'Color',aruba_red,'LineWidth',3,'Label','Lower ESS energy threshold', '
    LabelHorizontalAlignment','left','LabelVerticalAlignment','bottom')
xlabel('Time')
ylabel('Energy (Wh)')
grid on
ax = gca;
ax.GridAlpha = 0.4;
ax.GridLineStyle = "--";
ax.MinorGridLineStyle = "-";
ax.XColor = [0 0 0];
ax.YColor = [0 0 0];
ax.GridColor = [0 0 0];
title({'ESS energy usage throughout the day','Idle charge power = -100W','Idle
    discharge power = 50W'})
%%
fig2 = figure;
fig2.Position = [50 50 800 450];
set(fig2,'defaultAxesColorOrder',[[0 0 0]; [0 0 0]]);
plot(time_string,worst_day_pv(:,2),'LineWidth',2,'Color',aruba_blue);
ylabel('Power (W)')
hold on
yyaxis right
plot(time_string,worst_day_pv(:,1),'LineWidth',2,'Color',stedin_grey);
hold on
% stem(time_string(~ida_before),worst_day_pv(~ida_before,1),'LineWidth',1,'Color
    ',aruba_red);
% hold on

% yline(-5,'Color',aruba_red,'LineWidth',3,'Label','Lower ESS energy threshold',
    'LabelHorizontalAlignment','left','LabelVerticalAlignment','bottom')
xlabel('Time')
ylabel('Voltage (V)')
ylim([230 250])
find_xlim_low = find(time_string > '2017-08-04 10:31:00.00');
find_xlim_high = find(time_string > '2017-08-04 10:32:30.00');
xlim([time_string(find_xlim_low(1)),time_string(find_xlim_high(1))])
grid on
ax = gca;
```

```
ax.GridAlpha = 0.4;
ax.GridLineStyle = "--";
ax.MinorGridLineStyle = "-";
ax.XColor = [0 0 0];
ax.YColor = [0 0 0];
ax.GridColor = [0 0 0];
legend('PV system power','Voltage at PCC','Location','northwest')
titlestring = compose('Output power from PV system and voltage at PCC on %d %s %d
    ',day(time_string(end)),string(month(time_string(end),'name')),year(
    time_string(end)));
title(titlestring)

%%
fig2 = figure;
fig2.Position = [50 50 500 400];
ph = gobjects(1);
% set(fig2,'defaultAxesColorOrder',[[0 0 0]; [0 0 0]]);
ph(1) = plot(time_string,worst_day_pv(:,2),'Color',stedin_grey,'LineWidth',1);
ylim([0 4700])
ylabel('Power (W)')
hold on
yyaxis right
ph(2) = plot(time_string,worst_day_pv(:,1),'LineWidth',1,'Color',aruba_blue);
voltage_meas = worst_day_pv(:,1)
% hold on
% stem(time_string(~ida_before),worst_day_pv(~ida_before,1),'LineWidth',1,'Color
    ',aruba_red);
% hold on
for j = 1:length(find_indexes_pos_MA)
    ph(j+3) = plot(time_string([find_indexes_pos_MA(j), actual_indexes_pos_MA(j)
        ]),voltage_meas([find_indexes_pos_MA(j), actual_indexes_pos_MA(j)]),  '
        Marker', '.', 'MarkerSize', 10,'Color',royal_orange, 'LineWidth', 1,'
        LineStyle','-');
    hold on
end
for j = 1:length(find_indexes_neg_MA)
    ph(j+length(find_indexes_pos_MA)+3) = plot(time_string([find_indexes_neg_MA(j
        ), actual_indexes_neg_MA(j)]),voltage_meas([find_indexes_neg_MA(j),
        actual_indexes_neg_MA(j)]), 'Marker', '.', 'MarkerSize', 10, 'Color',
        aruba_red, 'LineWidth', 1,'LineStyle','-');
    hold on
end
ylim([210 240])
grid on
ax = gca;
ax.GridAlpha = 0.4;
ax.GridLineStyle = "--";
ax.MinorGridLineStyle = "-";
ax.XColor = [0 0 0];
ax.YColor = [0 0 0];
ax.GridColor = [0 0 0];
ylabel('Voltage (V)')
% legend(ph([1,2,3,4,length(find_indexes_neg_MA)+length(find_indexes_pos_MA)+3])
    ,'Measured PV Power','Measured BESS Power','Measured Voltage at PCC','Measured
     Visible Flicker Moment','Measured Annoying Flicker Moment','Location', 'Best
    ');
% legend('PV system power','Voltage at PCC','Location','southeast')

% yline(-5,'Color',aruba_red,'LineWidth',3,'Label','Lower ESS energy threshold',
    'LabelHorizontalAlignment','left','LabelVerticalAlignment','bottom')
%%
```

```
fig2 = figure;
fig2.Position = [50 50 800 450];
set(fig2,'defaultAxesColorOrder',[[0 0 0]; [0 0 0]]);
plot(time_string,worst_day_pv(:,2),'LineWidth',2,'Color',stedin_grey);
ylabel('Power (W)')
hold on
plot(time_string,supercapacitor_power_MA_grid,'LineWidth',2,'Color',aruba_blue,'
    LineStyle','-');
hold on
plot(time_string,supercapacitor_power_simpleMA_grid,'LineWidth',2,'Color',
    aruba_blue,'LineStyle',':');
% stem(time_string(~ida_before),worst_day_pv(~ida_before,1),'LineWidth',1,'Color
    ',aruba_red);
% hold on

% yline(-5,'Color',aruba_red,'LineWidth',3,'Label','Lower ESS energy threshold',
    'LabelHorizontalAlignment','left','LabelVerticalAlignment','bottom')
xlabel('Time')
% find_xlim_low = find(time_string > '2017-08-04 10:31:00.00');
% find_xlim_high = find(time_string > '2017-08-04 10:32:30.00');
% xlim([time_string(find_xlim_low(1)),time_string(find_xlim_high(1))])
grid on
ax = gca;
ax.GridAlpha = 0.4;
ax.GridLineStyle = "--";
ax.MinorGridLineStyle = "-";
ax.XColor = [0 0 0];
ax.YColor = [0 0 0];
ax.GridColor = [0 0 0];
legend('PV system power','Voltage at PCC','Location','northwest')
titlestring = compose('Output power from PV system with two types of moving
    averages on %d %s %d',day(time_string(end)),string(month(time_string(end),'
    name')),year(time_string(end)));
title(titlestring)
ylim([1600 2200])
find_xlim_low = find(time_string > '2017-08-04 15:33:00.00');
find_xlim_high = find(time_string > '2017-08-04 15:37:00.00');
xlim([time_string(find_xlim_low(1)),time_string(find_xlim_high(1))])
grid on
ax = gca;
ax.GridAlpha = 0.4;
ax.GridLineStyle = "--";
ax.MinorGridLineStyle = "-";
ax.XColor = [0 0 0];
ax.YColor = [0 0 0];
ax.GridColor = [0 0 0];
legend('PV system power','Moving average with optimised weight distribution
    output','Simple moving average output','Location','southeast')
%%
f = figure;
f.Position = [50 50 850 500];
plot(time_string,simple_layer_energy,'LineWidth',1,'Color',stedin_grey);
hold on
plot(time_string,first_layer_energy,'LineWidth',2,'Color',aruba_blue);
hold on
plot(time_string,second_layer_energy,'LineWidth',1,'Color',royal_orange, '
    LineStyle','-.');
hold on
plot(time_string,third_layer_energy,'LineWidth',1,'Color',aruba_yellow);
hold on
plot(time_string,soc_range_simpleMA(2:end),'LineWidth',1,'Color',aruba_red);
```

```
xlabel('Time')
ylabel('Energy (Wh)')
grid on
ax = gca;
ax.GridAlpha = 0.4;
ax.GridLineStyle = "--";
ax.MinorGridLineStyle = "-";
ax.XColor = [0 0 0];
ax.YColor = [0 0 0];
ax.GridColor = [0 0 0];
legend_simple_cmp = 'Simple moving average, energy usage = %0.2f Wh';
legend_simple = compose(legend_simple_cmp,max(simple_layer_energy)-min(
    simple_layer_energy));
legend_first_cmp = 'Moving average with optimised weights, energy usage = %0.2f
    Wh';
legend_first = compose(legend_first_cmp,max(first_layer_energy)-min(
    first_layer_energy));
legend_second_cmp = 'Window of operation check, energy usage = %0.2f Wh';
legend_second = compose(legend_second_cmp,max(second_layer_energy)-min(
    second_layer_energy));
legend_third_cmp = 'SoC power reduction curve, energy usage = %0.2f Wh';
legend_third = compose(legend_third_cmp,max(third_layer_energy)-min(
    third_layer_energy));
legend_fourth_cmp = 'Idle charge and discharge, energy usage = %0.2f Wh';
legend_fourth = compose(legend_fourth_cmp,max(soc_range_simpleMA)-min(
    soc_range_simpleMA));
legend({string(legend_simple),string(legend_first),string(legend_second),string(
    legend_third),string(legend_fourth)},'Location','southwest')
title({'ESS energy usage throughout the day with each control layer applied','
    Power Control'})
%%
figure
time_new = [];
set_power_new = [];
time_new(1) = 0;
set_power_new(1) = 400;
for i = 1:length(time)
     time_new(i+1) = time(i);
     set_power_new(i+1) = set_power(i);
end
plot(time,-set_power,'LineWidth',2,'Color',aruba_red);
hold on
plot(time,battery_power,'LineWidth',2,'Color',stedin_grey);
hold on
plot([time(1) time(1)],[0 -set_power_new(1)],'LineWidth',2,'Color',aruba_red)
xlabel('Time (s)')
ylabel('Power (W)')
ylim([-800 800])
grid on
ax = gca;
ax.GridAlpha = 0.4;
ax.GridLineStyle = "--";
ax.MinorGridLineStyle = "-";
ax.XColor = [0 0 0];
ax.YColor = [0 0 0];
ax.GridColor = [0 0 0];
legend('Power setpoint sent','Measured battery power at inverter output','
    Location','Best')
% titlestring = compose('Output power from PV system and voltage at PCC on %d %s
    %d',day(time_string(end)),string(month(time_string(end),'name')),year(
    time_string(end)));
```

```
% title(titlestring)
title({'Power setpoint versus measured power from inverter','-400 to 400W power
    setpoint','AC connected ESS experimental setup'})
%%
fig2 = figure;
fig2.Position = [50 50 800 600];
% time_new = [];
% set_power_new = [];
% time_new(1) = 0;
% set_power_new(1) = 400;
% for i = 1:length(time)
%     time_new(i+1) = time(i);
%     set_power_new(i+1) = set_power(i);
% end
plot(worst_day_pv(25090:25122,2),'LineWidth',2,'Color',stedin_grey,'LineStyle','-
    ','Marker','.','MarkerSize',30);
hold on
plot(sinus_MA_voltage_output_day(25090:25122),'LineWidth',2,'Color',aruba_blue,'
    LineStyle','none','Marker','.','MarkerSize',30);
hold on
% plot([time(1) time(1)],[0 -set_power_new(1)],'LineWidth',2,'Color',aruba_red)
xlabel('Datapoint')
ylabel('Power (W)')
ylim([2000 3500])
xlim([0 35])
grid on
ax = gca;
ax.GridAlpha = 0.4;
ax.GridLineStyle = "--";
ax.MinorGridLineStyle = "-";
ax.XColor = [0 0 0];
ax.YColor = [0 0 0];
ax.GridColor = [0 0 0];
legend('Measured PV power','Predicted output power','Location','southwest')
% titlestring = compose('Output power from PV system and voltage at PCC on %d %s
    %d',day(time_string(end)),string(month(time_string(end),'name')),year(
    time_string(end)));
% title(titlestring)
% title({'Power setpoint versus measured power from inverter','-400 to 400W power
     setpoint','AC connected ESS experimental setup'})
%% Plot Battery, PV POWER and AC voltage measurements
aruba_blue = [63/255 144/255 223/255];
aruba_red = [239/255 48/255 62/255];
aruba_yellow = [255/255 210/255 0/255];
royal_orange = [255/255 154/255 0/255];
stedin_grey = [77/255 77/255 77/255];
aruba_green = [132/255,194/255,37/255];

find_time = find(time_string > '2017-08-04 11:58:00.000' & time_string < '
    2017-08-04 12:04:00.000');

fig2 = figure;
fig2.Position = [50 50 950 600];
set(fig2,'defaultAxesColorOrder',[[0 0 0]; [0 0 0]]);
battery_power_zero = zeros(1,length(worst_day_pv(:,2)));
yyaxis left
ph = gobjects(1);
ph(1) = plot(time_string,worst_day_pv(:,2),'LineWidth',2,'Color',aruba_blue);
hold on
ph(2) = plot(time_string,battery_power_zero,'LineWidth',2,'Color',aruba_green,'
    LineStyle','-');
```

```
ylim([-2000 6000])
ylabel('Power (W)')
yyaxis right
hold on
ph(3) = plot(time_string,worst_day_pv(:,1),'LineWidth',2,'Color',stedin_grey);
hold on
for j = 1:length(find_indexes_pos_MA)
    ph(j+3) = plot(time_string([find_indexes_pos_MA(j), actual_indexes_pos_MA(j)
        ]),worst_day_pv([find_indexes_pos_MA(j), actual_indexes_pos_MA(j)],1),  '
        Marker', '.', 'MarkerSize', 20,'Color',royal_orange, 'LineWidth', 2,'
        LineStyle','-');
    hold on
end
for j = 1:length(find_indexes_neg_MA)
    ph(j+length(find_indexes_pos_MA)+3) = plot(time_string([find_indexes_neg_MA(j
        ), actual_indexes_neg_MA(j)]),worst_day_pv([find_indexes_neg_MA(j),
        actual_indexes_neg_MA(j)],1), 'Marker', '.', 'MarkerSize', 25, 'Color',
        aruba_red, 'LineWidth', 3,'LineStyle','-');
    hold on
end
if length(find_indexes_pos_MA) > 0
    if length(find_indexes_neg_MA) > 0
        legend(ph([1,2,3,4,length(find_indexes_neg_MA)+length(find_indexes_pos_MA
            )+3]),'PV Power','ESS Power','Voltage at PCC','Measured Visible
            Flicker Moment','Measured Annoying Flicker Moment','Location', '
            northeast');
    else
        legend(ph([1,2,3,4]),'PV Power','ESS Power','Voltage at PCC','Measured
            Visible Flicker Moment','Location', 'Best');
    end
else
    if length(find_indexes_neg_MA) > 0
        legend(ph([1,2,3,4]),'PV Power','ESS Power','Voltage at PCC','Measured
            Annoying Flicker Moment','Location', 'Best');
    else
        legend(ph([1,2,3]),'PV Power','ESS Power','Voltage at PCC','Location', '
            Best');
    end
end
ylim([210 245])
ylabel('Voltage (V)')
grid on
ax = gca;
ax.GridAlpha = 0.4;
ax.GridLineStyle = "-";
ax.XColor = [0 0 0];
ax.GridColor = [0 0 0];
% title({string(first_line);setup;string(control_line);string(second_line)})
xlabel('Time')
xlim([time_string(find_time(1)) time_string(find_time(end))])
%% Plot Battery, PV POWER and AC voltage measurements
aruba_blue = [63/255 144/255 223/255];
aruba_red = [239/255 48/255 62/255];
aruba_yellow = [255/255 210/255 0/255];
royal_orange = [255/255 154/255 0/255];
stedin_grey = [77/255 77/255 77/255];
aruba_green = [132/255,194/255,37/255];

find_time = find(time_string > '2017-08-04 11:58:00.000' & time_string < '
    2017-08-04 12:04:00.000');
```

```
fig2 = figure;
fig2.Position = [50 50 950 600];
set(fig2,'defaultAxesColorOrder',[[0 0 0]; [0 0 0]]);
battery_power_zero = zeros(1,length(worst_day_pv(:,2)));
yyaxis left
ph = gobjects(1);
ph(1) = plot(time_string,worst_day_pv(:,2),'LineWidth',2,'Color',aruba_blue);
hold on
ph(2) = plot(time_string,supercapacitor_power_PCC_SOC,'LineWidth',2,'Color',
    aruba_green,'LineStyle','-');
ylim([-2000 6000])
ylabel('Power (W)')
yyaxis right
hold on
ph(3) = plot(time_string,new_voltage,'LineWidth',2,'Color',stedin_grey);
hold on
% for j = 1:length(find_indexes_pos_MA)
%     ph(j+3) = plot(time_string([find_indexes_pos_MA(j), actual_indexes_pos_MA(j
    )]),new_voltage([find_indexes_pos_MA(j), actual_indexes_pos_MA(j)]),  'Marker
    ', '.', 'MarkerSize', 20,'Color',royal_orange, 'LineWidth', 2,'LineStyle','-')
    ;
%     hold on
% end
% for j = 1:length(find_indexes_neg_MA)
%     ph(j+length(find_indexes_pos_MA)+3) = plot(time_string([find_indexes_neg_MA
    (j), actual_indexes_neg_MA(j)]),new_voltage([find_indexes_neg_MA(j),
    actual_indexes_neg_MA(j)]), 'Marker', '.', 'MarkerSize', 25, 'Color',aruba_red
    , 'LineWidth', 3,'LineStyle','-');
%     hold on
% end
% if length(find_indexes_pos_MA) > 0
%     if length(find_indexes_neg_MA) > 0
%         legend(ph([1,2,3,4,length(find_indexes_neg_MA)+length(
    find_indexes_pos_MA)+3]),'PV Power','ESS Power','Voltage at PCC','Measured
    Visible Flicker Moment','Measured Annoying Flicker Moment','Location', 'Best')
    ;
%     else
%         legend(ph([1,2,3,4]),'PV Power','ESS Power','Voltage at PCC','Measured
    Visible Flicker Moment','Location', 'Best');
%     end
% else
%     if length(find_indexes_neg_MA) > 0
%         legend(ph([1,2,3,4]),'PV Power','ESS Power','Voltage at PCC','Measured
    Annoying Flicker Moment','Location', 'Best');
%     else
%         legend(ph([1,2,3]),'PV Power','ESS Power','Voltage at PCC','Location',
    'Best');
%     end
% end
legend(ph([1,2,3]),'PV Power','ESS Power','Voltage at PCC','Location', 'Northeast
    ');
ylim([210 245])
ylabel('Voltage (V)')
grid on
ax = gca;
ax.GridAlpha = 0.4;
ax.GridLineStyle = "-";
ax.XColor = [0 0 0];
ax.GridColor = [0 0 0];
% title({string(first_line);setup;string(control_line);string(second_line)})
xlabel('Time')
```

```
xlim([time_string(find_time(1)) time_string(find_time(end))])
```

1.3. Full year power control simulation Matlab script

```
%% Load in the power and voltage values
clear all
pv_data = readmatrix('Stedin Thesis 2023/sensorid_249.csv');
power_rating = 3480;
efficiency_inverter = 0.94;
sensitivity = 0.00106737202547458;
%% Load in the timestamp values
opts = detectImportOptions('Stedin Thesis 2023/sensorid_249.csv');
opts.SelectedVariableNames = {'Time'};
time_data = readtable('Stedin Thesis 2023/sensorid_249.csv',opts);
time_string = table2array(time_data);
%% Algorithm to sort each day
indexes = find(contains(string(time_string),"00:00:0") == 1);
difference_indexes = [];
for i = 1:length(indexes)-1
    difference_indexes(i) = indexes(i+1) - indexes(i);
end
indexes_days = find(difference_indexes > 10);
whole_day_indexes = difference_indexes(indexes_days);
days_in_between = [];
power_day_data = {};
lower_bound = [];
upper_bound = [];
for i = 1:length(whole_day_indexes)-1
    bound(1) = indexes(2);
    bound(i+1) = bound(i) + whole_day_indexes(i);
    daytoday_data{1,i} = pv_data(bound(i):bound(i+1),3)';
    daytoday_data{2,i} = pv_data(bound(i):bound(i+1),7)';
    daytoday_data{3,i} = time_string(bound(i):bound(i+1));
end
%% Filter per season
find_indexes_winter = [];
find_indexes_spring = [];
find_indexes_summer = [];
find_indexes_fall = [];
for i = 1:321
    indexes_winter = find(daytoday_data{3,i}(1) < "2017-03-20" | daytoday_data{3,
       i}(1) > "2017-12-21");
    if isempty(indexes_winter) == logical(0)
        find_indexes_winter = [find_indexes_winter; i];
    else
        find_indexes_winter = find_indexes_winter;
    end
    indexes_spring = find(daytoday_data{3,i}(1) >= "2017-03-20" & daytoday_data
       {3,i}(1) < "2017-06-21");
    if isempty(indexes_spring) == logical(0)
        find_indexes_spring = [find_indexes_spring; i];
    else
        find_indexes_spring = find_indexes_spring;
    end
    indexes_summer = find(daytoday_data{3,i}(1) >= "2017-06-21" & daytoday_data
       {3,i}(1) < "2017-09-22");
    if isempty(indexes_summer) == logical(0)
        find_indexes_summer = [find_indexes_summer; i];
    else
        find_indexes_summer = find_indexes_summer;
    end
```

```
    indexes_fall = find(daytoday_data{3,i}(1) >= "2017-09-22" & daytoday_data{3,i
        }(1) < "2017-12-21");
    if isempty(indexes_fall) == logical(0)
        find_indexes_fall = [find_indexes_fall; i];
    else
        find_indexes_fall = find_indexes_fall;
    end
end
time_winter = {'08:00:00.00','17:00:00.00'};
time_spring = {'06:00:00.00', '20:00:00.00'};
time_summer = {'05:00:00.00', '22:00:00.00'};
time_fall = {'08:00:00.00','19:00:00.00'};
sum_year = length(find_indexes_winter) + length(find_indexes_spring) + length(
    find_indexes_summer) + length(find_indexes_fall);
winter_data = {};
for k = 1:length(find_indexes_winter)
    winter_data{1,k} = daytoday_data{1,find_indexes_winter(k)};
    winter_data{2,k} = daytoday_data{2,find_indexes_winter(k)};
    winter_data{3,k} = daytoday_data{3,find_indexes_winter(k)};
end
spring_data = {};
for k = 1:length(find_indexes_spring)
    spring_data{1,k} = daytoday_data{1,find_indexes_spring(k)};
    spring_data{2,k} = daytoday_data{2,find_indexes_spring(k)};
    spring_data{3,k} = daytoday_data{3,find_indexes_spring(k)};
end
summer_data = {};
for k = 1:length(find_indexes_summer)
    summer_data{1,k} = daytoday_data{1,find_indexes_summer(k)};
    summer_data{2,k} = daytoday_data{2,find_indexes_summer(k)};
    summer_data{3,k} = daytoday_data{3,find_indexes_summer(k)};
end
fall_data = {};
for k = 1:length(find_indexes_fall)
    fall_data{1,k} = daytoday_data{1,find_indexes_fall(k)};
    fall_data{2,k} = daytoday_data{2,find_indexes_fall(k)};
    fall_data{3,k} = daytoday_data{3,find_indexes_fall(k)};
end
%% Filter to only when PV is operating and apply ASCWMA to the window of
    operation
indexes_pv_operation = {};
sinus_MA_voltage_output = {};
use_data = {};
use_data = summer_data;
actual_voltage = {};
for l = 1:length(use_data)
    indexes_pv_operation{l} = find(timeofday(use_data{3,l}) > time_summer{1} &
        timeofday(use_data{3,l}) < time_summer{2});
    test_data = use_data{1,l};
    sinus_MA_power_output_day = [];
    window = [30];
    for i = 1:length(window)
    %sinus_MA_voltage_output_day(1:window(i),i) = test_data(1:window(i));
        weighting_multiplier = [];
        weighting_multiplier = linspace(1,window(i),window(i));
        weighting_fraction = [];
        for k = 1:window(i)
            weighting_fraction(k) = cos((1/window(i))*k*pi/2)^2;
        end
        weighting_fraction = flip(weighting_fraction);
        for j = indexes_pv_operation{l}(1):indexes_pv_operation{l}(end)
```

```
            sinus_MA_power_output_day(j,i) = sum(test_data(j-window(i):j-1).*
                weighting_fraction)/sum(weighting_fraction);
        end
        sinus_MA_power_output_day = sinus_MA_power_output_day(
            indexes_pv_operation{l},i);
    end
    sinus_MA_power_output{l} = test_data;
    sinus_MA_power_output{l}(indexes_pv_operation{l}) = sinus_MA_power_output_day
        ;
    daytoday_data{4,l} = sinus_MA_power_output;
end
%% SOC deration and Idle charge applied to each day of the year.
% use_data = {};
% use_data = fall_data;
delta_V = {};
supercapacitor_power_PCC = {};
supercapacitor_power_simpleMA = {};
soc_range_simpleMA_SOC = {};
supercapacitor_power_PCC = {};
supercapacitor_power_PCC_SOC = {};
supercapacitor_power_simpleMA_SOC = {};
energy_supercapacitor_Wh_simpleMA_SOC = [];
maximum_charge_power_simpleMA_SOC = [];
maximum_discharge_power_simpleMA_SOC = [];
for k = 1:length(use_data)
    for i = 1:length(sinus_MA_power_output{k})
        supercapacitor_power_PCC{k}(i) = sinus_MA_power_output{k}(i) - use_data
            {1,k}(i);
    end
    for j = 1:length(supercapacitor_power_PCC{k})
        if supercapacitor_power_PCC{k}(j) < 0
            supercapacitor_power_simpleMA{k}(j) = supercapacitor_power_PCC{k}(j)*
                efficiency_inverter;
        elseif supercapacitor_power_PCC{k}(j) >= 0
            supercapacitor_power_simpleMA{k}(j) = supercapacitor_power_PCC{k}(j)/
                efficiency_inverter;
        end
    end
    soc_range_simpleMA_SOC{k} = zeros(length(supercapacitor_power_simpleMA{k}),1)
        ;
    soc_range_simpleMA_SOC{k}(1) = 0;
    % total_energy_used_supercapacitor_simpleMA = max(cumtrapz(
        supercapacitor_power_simpleMA))*2;
    % soc_range_simpleMA = cumtrapz(supercapacitor_power_simpleMA)*2/(3600);

    for i = 1:length(supercapacitor_power_simpleMA{k})
        Wh_range_acceptable = [-5 5];
        correction_factor_slope_c = 0.01;
        correction_factor_slope_d = 0.01;
        % SoC Deadzone Curve implementation
        if supercapacitor_power_simpleMA{k}(i) > 0
            if soc_range_simpleMA_SOC{k}(i) > Wh_range_acceptable(1) &&
                soc_range_simpleMA_SOC{k}(i) < Wh_range_acceptable(end)
                supercapacitor_power_PCC_SOC{k}(i) = supercapacitor_power_PCC{k}(
                    i);
                supercapacitor_power_simpleMA_SOC{k}(i) =
                    supercapacitor_power_simpleMA{k}(i);
            elseif soc_range_simpleMA_SOC{k}(i) < Wh_range_acceptable(1)
                delta_soc = soc_range_simpleMA_SOC{k}(i) - Wh_range_acceptable(1)
                    ;
                supercapacitor_power_PCC_SOC{k}(i) = supercapacitor_power_PCC{k}(
```

```
            i)*(1+abs(delta_soc)*correction_factor_slope_d);
        supercapacitor_power_simpleMA_SOC{k}(i) =
            supercapacitor_power_simpleMA{k}(i)*(1-abs(delta_soc)*
            correction_factor_slope_d);
    elseif soc_range_simpleMA_SOC{k}(i) > Wh_range_acceptable(end)
        supercapacitor_power_PCC_SOC{k}(i) = supercapacitor_power_PCC{k}(
            i);
        supercapacitor_power_simpleMA_SOC{k}(i) =
            supercapacitor_power_simpleMA{k}(i);
    end
else
    if soc_range_simpleMA_SOC{k}(i) > Wh_range_acceptable(1) &&
       soc_range_simpleMA_SOC{k}(i) < Wh_range_acceptable(end)
        supercapacitor_power_PCC_SOC{k}(i) = supercapacitor_power_PCC{k}(
            i);
        supercapacitor_power_simpleMA_SOC{k}(i) =
            supercapacitor_power_simpleMA{k}(i);
    elseif soc_range_simpleMA_SOC{k}(i) < Wh_range_acceptable(1)
        supercapacitor_power_PCC_SOC{k}(i) = supercapacitor_power_PCC{k}(
            i);
        supercapacitor_power_simpleMA_SOC{k}(i) =
            supercapacitor_power_simpleMA{k}(i);
    elseif soc_range_simpleMA_SOC{k}(i) > Wh_range_acceptable(end)
        delta_soc = soc_range_simpleMA_SOC{k}(i) - Wh_range_acceptable(
            end);
        supercapacitor_power_PCC_SOC{k}(i) = supercapacitor_power_PCC{k}(
            i)*(1-abs(delta_soc)*correction_factor_slope_c);
        supercapacitor_power_simpleMA_SOC{k}(i) =
            supercapacitor_power_simpleMA{k}(i)*(1-abs(delta_soc)*
            correction_factor_slope_c);
    end
end
% Idle Charging
if (supercapacitor_power_simpleMA_SOC{k}(i) < 50 &&
   supercapacitor_power_simpleMA_SOC{k}(i) > -50)
    if soc_range_simpleMA_SOC{k}(i) > 5
        supercapacitor_power_simpleMA_SOC{k}(i) = 50;
    elseif soc_range_simpleMA_SOC{k}(i) < -5
        supercapacitor_power_simpleMA_SOC{k}(i) = -100;
    end
else
    supercapacitor_power_simpleMA_SOC{k}(i) =
       supercapacitor_power_simpleMA_SOC{k}(i);
end

if i == 1
    energy_used = 0;
    soc_range_simpleMA_SOC{k}(i) = 0;
else
    energy_used = (supercapacitor_power_simpleMA_SOC{k}(i) +
       supercapacitor_power_simpleMA_SOC{k}(i-1))*2*0.5;
    soc_range_simpleMA_SOC{k}(i+1) = soc_range_simpleMA_SOC{k}(i) - (
       energy_used)/3600;
end

if supercapacitor_power_simpleMA_SOC{k}(i) < 0
    supercapacitor_power_PCC_SOC{k}(i) =
       supercapacitor_power_simpleMA_SOC{k}(i)/efficiency_inverter;
elseif supercapacitor_power_simpleMA_SOC{k}(i) >= 0
    supercapacitor_power_PCC_SOC{k}(i) =
       supercapacitor_power_simpleMA_SOC{k}(i)*efficiency_inverter;
```

```
        end
%         for j = 1:length(supercapacitor_power_PCC_SOC{k})
%             actual_voltage{k}(j) = use_data{2,k}(j) +
    supercapacitor_power_PCC_SOC{k}(j)*sensitivity;
%         end
    end

    % energy_supercapacitor_Wh_simpleMA = (
       total_energy_used_suwpercapacitor_simpleMA- min(cumtrapz(
       supercapacitor_power_simpleMA))*2)/(60*60);
    energy_supercapacitor_Wh_simpleMA_SOC(k) = (max(soc_range_simpleMA_SOC{k})-
       min(soc_range_simpleMA_SOC{k}));
    % soc_range_simpleMA_SOC = cumtrapz(supercapacitor_power_simpleMA_SOC)
       *2/(3600);
    maximum_charge_power_simpleMA(k) = min(supercapacitor_power_simpleMA{k});
    maximum_discharge_power_simpleMA(k) = max(supercapacitor_power_simpleMA{k});
    maximum_charge_power_simpleMA_SOC(k) = min(supercapacitor_power_simpleMA_SOC{
       k});
    maximum_discharge_power_simpleMA_SOC(k) = max(
       supercapacitor_power_simpleMA_SOC{k});
end

%%
actual_voltage = {};
for i = 1:length(supercapacitor_power_PCC_SOC)
    for j = 1:length(supercapacitor_power_PCC_SOC{i})
        actual_voltage{i}(j) = use_data{2,i}(j) + supercapacitor_power_PCC_SOC{i
           }(j)*sensitivity;
    end
end

%%
aruba_blue = [63/255 144/255 223/255];
aruba_red = [239/255 48/255 62/255];
aruba_yellow = [255/255 210/255 0/255];
royal_orange = [255/255 154/255 0/255];
average_consumption = mean(energy_supercapacitor_Wh_simpleMA_SOC);
figure
stem(energy_supercapacitor_Wh_simpleMA_SOC, 'LineStyle','-','Color', aruba_blue,
   'MarkerFaceColor',aruba_yellow,'MarkerEdgeColor',royal_orange, 'LineWidth', 1)
hold on
yline(average_consumption,'Color',aruba_red,'LineWidth',3)
ylabel('Energy usage (Wh)')
xlabel('Number of days')
title_second_cmp = 'Season period : %s - %s';
title_second_line = compose(title_second_cmp,"2017-12-21","2017-03-20");
title_third_line = 'Power Control';
title({'Energy usage of voltage control during the winter period',string(
   title_second_line),string(title_third_line)})
linestr = 'Average energy usage of ESS during the season, %0.2f Wh';
line_compose = compose(linestr,average_consumption);
legend('Energy usage of ESS for each day in the season',string(line_compose),'
   Location','south')
%%

for i = 1:length(actual_voltage)
    for j = 1:length(actual_voltage{i})-1
        delta_v_data{i}(j) = actual_voltage{i}(j+1) - actual_voltage{i}(j);
        delta_v_data_before{i}(j) = use_data{2,i}(j+1) - use_data{2,i}(j);
        ramprate_voltage{i}(j) = delta_v_data{i}(j)/2;
        ramprate_voltage_before{i}(j) = delta_v_data_before{i}(j)/2;
```

```
    end
    indexes_visible_curve{i} = find(100*abs(ramprate_voltage{i})/230 > 0.36);
    indexes_visible_curve_before{i} = find(100*abs(ramprate_voltage_before{i})
       /230 > 0.36);
    amount_visible_flicker(i) = length(indexes_visible_curve{i});
    amount_visible_flicker_before(i) = length(indexes_visible_curve_before{i});
end
%%
figure
stem(amount_visible_flicker_before)
hold on
stem(amount_visible_flicker)
```

1.4. Full year voltage control simulation Matlab script

```
%% Load in the power and voltage values
clear all
pv_data = readmatrix('Stedin Thesis 2023/sensorid_249.csv');
power_rating = 3480;
efficiency_inverter = 0.94;
% % sensitivity = 0.00106737202547458;
%% Load in the timestamp values
opts = detectImportOptions('Stedin Thesis 2023/sensorid_249.csv');
opts.SelectedVariableNames = {'Time'};
time_data = readtable('Stedin Thesis 2023/sensorid_249.csv',opts);
time_string = table2array(time_data);
%% Algorithm to sort each day
indexes = find(contains(string(time_string),"00:00:0") == 1);
difference_indexes = [];
for i = 1:length(indexes)-1
    difference_indexes(i) = indexes(i+1) - indexes(i);
end
indexes_days = find(difference_indexes > 10);
whole_day_indexes = difference_indexes(indexes_days);
days_in_between = [];
power_day_data = {};
lower_bound = [];
upper_bound = [];
for i = 1:length(whole_day_indexes)-1
    bound(1) = indexes(2);
    bound(i+1) = bound(i) + whole_day_indexes(i);
    daytoday_data{1,i} = pv_data(bound(i):bound(i+1),3)';
    daytoday_data{2,i} = pv_data(bound(i):bound(i+1),7)';
    daytoday_data{3,i} = time_string(bound(i):bound(i+1));
end
%% Filter per season
find_indexes_winter = [];
find_indexes_spring = [];
find_indexes_summer = [];
find_indexes_fall = [];
for i = 1:321
    indexes_winter = find(daytoday_data{3,i}(1) < "2017-03-20" | daytoday_data{3,
       i}(1) > "2017-12-21");
    if isempty(indexes_winter) == logical(0)
        find_indexes_winter = [find_indexes_winter; i];
    else
        find_indexes_winter = find_indexes_winter;
    end
    indexes_spring = find(daytoday_data{3,i}(1) >= "2017-03-20" & daytoday_data
       {3,i}(1) < "2017-06-21");
    if isempty(indexes_spring) == logical(0)
```

```
        find_indexes_spring = [find_indexes_spring; i];
    else
        find_indexes_spring = find_indexes_spring;
    end
    indexes_summer = find(daytoday_data{3,i}(1) >= "2017-06-21" & daytoday_data
       {3,i}(1) < "2017-09-22");
    if isempty(indexes_summer) == logical(0)
        find_indexes_summer = [find_indexes_summer; i];
    else
        find_indexes_summer = find_indexes_summer;
    end
    indexes_fall = find(daytoday_data{3,i}(1) >= "2017-09-22" & daytoday_data{3,i
       }(1) < "2017-12-21");
    if isempty(indexes_fall) == logical(0)
        find_indexes_fall = [find_indexes_fall; i];
    else
        find_indexes_fall = find_indexes_fall;
    end
end
time_winter = {'08:00:00.00','17:00:00.00'};
time_spring = {'06:00:00.00', '20:00:00.00'};
time_summer = {'05:00:00.00', '22:00:00.00'};
time_fall = {'08:00:00.00','19:00:00.00'};
sum_year = length(find_indexes_winter) + length(find_indexes_spring) + length(
   find_indexes_summer) + length(find_indexes_fall);
winter_data = {};
for k = 1:length(find_indexes_winter)
    winter_data{1,k} = daytoday_data{1,find_indexes_winter(k)};
    winter_data{2,k} = daytoday_data{2,find_indexes_winter(k)};
    winter_data{3,k} = daytoday_data{3,find_indexes_winter(k)};
end
spring_data = {};
for k = 1:length(find_indexes_spring)
    spring_data{1,k} = daytoday_data{1,find_indexes_spring(k)};
    spring_data{2,k} = daytoday_data{2,find_indexes_spring(k)};
    spring_data{3,k} = daytoday_data{3,find_indexes_spring(k)};
end
summer_data = {};
for k = 1:length(find_indexes_summer)
    summer_data{1,k} = daytoday_data{1,find_indexes_summer(k)};
    summer_data{2,k} = daytoday_data{2,find_indexes_summer(k)};
    summer_data{3,k} = daytoday_data{3,find_indexes_summer(k)};
end
fall_data = {};
for k = 1:length(find_indexes_fall)
    fall_data{1,k} = daytoday_data{1,find_indexes_fall(k)};
    fall_data{2,k} = daytoday_data{2,find_indexes_fall(k)};
    fall_data{3,k} = daytoday_data{3,find_indexes_fall(k)};
end
%% Filter to only when PV is operating and apply ASCWMA to the window of
   operation
indexes_pv_operation = {};
sinus_MA_voltage_output = {};
use_data = {};
use_data = fall_data;
actual_voltage = {};
for l = 1:length(use_data)
    indexes_pv_operation{l} = find(timeofday(use_data{3,l}) > time_fall{1} &
       timeofday(use_data{3,l}) < time_fall{2});
    test_data = use_data{2,l};
    sinus_MA_voltage_output_day = [];
```

```
    window = [12];
    for i = 1:length(window)
    %sinus_MA_voltage_output_day(1:window(i),i) = test_data(1:window(i));
        weighting_multiplier = [];
        weighting_multiplier = linspace(1,window(i),window(i));
        weighting_fraction = [];
        for k = 1:window(i)
            weighting_fraction(k) = cos((1/window(i))*k*pi/2)^2;
        end
        weighting_fraction = flip(weighting_fraction);
        for j = indexes_pv_operation{l}(1):indexes_pv_operation{l}(end)
            sinus_MA_voltage_output_day(j,i) = sum(test_data(j-window(i):j-1).*
               weighting_fraction)/sum(weighting_fraction);
        end
        sinus_MA_voltage_output_day = sinus_MA_voltage_output_day(
           indexes_pv_operation{l},i);
    end
    sinus_MA_voltage_output{l} = test_data;
    sinus_MA_voltage_output{l}(indexes_pv_operation{l}) =
       sinus_MA_voltage_output_day;
    daytoday_data{4,l} = sinus_MA_voltage_output;
end
% %% SOC deration and Idle charge applied to each day of the year.
% use_data = {};
% use_data = fall_data;
delta_V = {};
supercapacitor_power_PCC = {};
supercapacitor_power_simpleMA = {};
soc_range_simpleMA_SOC = {};
supercapacitor_power_PCC = {};
supercapacitor_power_PCC_SOC = {};
supercapacitor_power_simpleMA_SOC = {};
for k = 1:length(use_data)
    for i = 1:length(sinus_MA_voltage_output{k})
        delta_V{k}(i) = sinus_MA_voltage_output{k}(i) - use_data{2,k}(i);
        supercapacitor_power_PCC{k}(i) = delta_V{k}(i)/sensitivity;
    end
    for j = 1:length(supercapacitor_power_PCC{k})
        if supercapacitor_power_PCC{k}(j) < 0
            supercapacitor_power_simpleMA{k}(j) = supercapacitor_power_PCC{k}(j)*
               efficiency_inverter;
        elseif supercapacitor_power_PCC{k}(j) >= 0
            supercapacitor_power_simpleMA{k}(j) = supercapacitor_power_PCC{k}(j)/
               efficiency_inverter;
        end
    end
    soc_range_simpleMA_SOC{k} = zeros(length(supercapacitor_power_simpleMA{k}),1)
       ;
    soc_range_simpleMA_SOC{k}(1) = 0;
    % total_energy_used_supercapacitor_simpleMA = max(cumtrapz(
       supercapacitor_power_simpleMA))*2;
    % soc_range_simpleMA = cumtrapz(supercapacitor_power_simpleMA)*2/(3600);

    for i = 1:length(supercapacitor_power_simpleMA{k})
        Wh_range_acceptable = [-5 5];
        correction_factor_slope_c = 0.01;
        correction_factor_slope_d = 0.01;
        % SoC Deadzone Curve implementation
        if supercapacitor_power_simpleMA{k}(i) > 0
            if soc_range_simpleMA_SOC{k}(i) > Wh_range_acceptable(1) &&
               soc_range_simpleMA_SOC{k}(i) < Wh_range_acceptable(end)
```

```
            supercapacitor_power_PCC_SOC{k}(i) = supercapacitor_power_PCC{k}(
                i);
            supercapacitor_power_simpleMA_SOC{k}(i) =
                supercapacitor_power_simpleMA{k}(i);
        elseif soc_range_simpleMA_SOC{k}(i) < Wh_range_acceptable(1)
            delta_soc = soc_range_simpleMA_SOC{k}(i) - Wh_range_acceptable(1)
                ;
            supercapacitor_power_PCC_SOC{k}(i) = supercapacitor_power_PCC{k}(
                i)*(1+abs(delta_soc)*correction_factor_slope_d);
            supercapacitor_power_simpleMA_SOC{k}(i) =
                supercapacitor_power_simpleMA{k}(i)*(1-abs(delta_soc)*
                correction_factor_slope_d);
        elseif soc_range_simpleMA_SOC{k}(i) > Wh_range_acceptable(end)
            supercapacitor_power_PCC_SOC{k}(i) = supercapacitor_power_PCC{k}(
                i);
            supercapacitor_power_simpleMA_SOC{k}(i) =
                supercapacitor_power_simpleMA{k}(i);
        end
    else
        if soc_range_simpleMA_SOC{k}(i) > Wh_range_acceptable(1) &&
            soc_range_simpleMA_SOC{k}(i) < Wh_range_acceptable(end)
            supercapacitor_power_PCC_SOC{k}(i) = supercapacitor_power_PCC{k}(
                i);
            supercapacitor_power_simpleMA_SOC{k}(i) =
                supercapacitor_power_simpleMA{k}(i);
        elseif soc_range_simpleMA_SOC{k}(i) < Wh_range_acceptable(1)
            supercapacitor_power_PCC_SOC{k}(i) = supercapacitor_power_PCC{k}(
                i);
            supercapacitor_power_simpleMA_SOC{k}(i) =
                supercapacitor_power_simpleMA{k}(i);
        elseif soc_range_simpleMA_SOC{k}(i) > Wh_range_acceptable(end)
            delta_soc = soc_range_simpleMA_SOC{k}(i) - Wh_range_acceptable(
                end);
            supercapacitor_power_PCC_SOC{k}(i) = supercapacitor_power_PCC{k}(
                i)*(1-abs(delta_soc)*correction_factor_slope_c);
            supercapacitor_power_simpleMA_SOC{k}(i) =
                supercapacitor_power_simpleMA{k}(i)*(1-abs(delta_soc)*
                correction_factor_slope_c);
        end
    end
    % Idle Charging
    if (supercapacitor_power_simpleMA_SOC{k}(i) < 50 &&
        supercapacitor_power_simpleMA_SOC{k}(i) > -50)
        if soc_range_simpleMA_SOC{k}(i) > 5
            supercapacitor_power_simpleMA_SOC{k}(i) = 50;
        elseif soc_range_simpleMA_SOC{k}(i) < -5
            supercapacitor_power_simpleMA_SOC{k}(i) = -100;
        end
    else
        supercapacitor_power_simpleMA_SOC{k}(i) =
            supercapacitor_power_simpleMA_SOC{k}(i);
    end

    if i == 1
        energy_used = 0;
        soc_range_simpleMA_SOC{k}(i) = 0;
    else
        energy_used = (supercapacitor_power_simpleMA_SOC{k}(i) +
            supercapacitor_power_simpleMA_SOC{k}(i-1))*2*0.5;
        soc_range_simpleMA_SOC{k}(i+1) = soc_range_simpleMA_SOC{k}(i) - (
            energy_used)/3600;
```

```
        end

        if supercapacitor_power_simpleMA_SOC{k}(i) < 0
            supercapacitor_power_PCC_SOC{k}(i) =
               supercapacitor_power_simpleMA_SOC{k}(i)/efficiency_inverter;
        elseif supercapacitor_power_simpleMA_SOC{k}(i) >= 0
            supercapacitor_power_PCC_SOC{k}(i) =
               supercapacitor_power_simpleMA_SOC{k}(i)*efficiency_inverter;
        end
%         for j = 1:length(supercapacitor_power_PCC_SOC{k})
%             actual_voltage{k}(j) = use_data{2,k}(j) +
    supercapacitor_power_PCC_SOC{k}(j)*sensitivity;
%         end
    end

    % energy_supercapacitor_Wh_simpleMA = (
       total_energy_used_suwpercapacitor_simpleMA - min(cumtrapz(
       supercapacitor_power_simpleMA))*2)/(60*60);
    energy_supercapacitor_Wh_simpleMA_SOC(k) = (max(soc_range_simpleMA_SOC{k})-
       min(soc_range_simpleMA_SOC{k}));
    % soc_range_simpleMA_SOC = cumtrapz(supercapacitor_power_simpleMA_SOC)
       *2/(3600);
    maximum_charge_power_simpleMA(k) = min(supercapacitor_power_simpleMA{k});
    maximum_discharge_power_simpleMA(k) = max(supercapacitor_power_simpleMA{k});
    maximum_charge_power_simpleMA_SOC(k) = min(supercapacitor_power_simpleMA_SOC{
       k});
    maximum_discharge_power_simpleMA_SOC(k) = max(
       supercapacitor_power_simpleMA_SOC{k});
end

%%
actual_voltage = {};
for i = 1:length(supercapacitor_power_PCC_SOC)
    for j = 1:length(supercapacitor_power_PCC_SOC{i})
        actual_voltage{i}(j) = use_data{2,i}(j) + supercapacitor_power_PCC_SOC{i
           }(j)*sensitivity;
    end
end

%%
aruba_blue = [63/255 144/255 223/255];
aruba_red = [239/255 48/255 62/255];
aruba_yellow = [255/255 210/255 0/255];
royal_orange = [255/255 154/255 0/255];
average_consumption = mean(energy_supercapacitor_Wh_simpleMA_SOC);
figure
stem(energy_supercapacitor_Wh_simpleMA_SOC, 'LineStyle','-','Color', aruba_blue,
   'MarkerFaceColor',aruba_yellow,'MarkerEdgeColor',royal_orange, 'LineWidth', 1)
hold on
yline(average_consumption,'Color',aruba_red,'LineWidth',3)
ylabel('Energy usage (Wh)')
xlabel('Number of days')
title_second_cmp = 'Season period : %s - %s';
title_second_line = compose(title_second_cmp,"2017-12-21","2017-03-20");
title_third_line = 'Voltage Control';
title({'Energy usage of voltage control during the winter period',string(
   title_second_line),string(title_third_line)})
linestr = 'Average energy usage of ESS during the season, %0.2f Wh';
line_compose = compose(linestr,average_consumption);
legend('Energy usage of ESS for each day in the season',string(line_compose),'
   Location','south')
```

```
%%

for i = 1:length(actual_voltage)
    for j = 1:length(actual_voltage{i})-1
        delta_v_data{i}(j) = actual_voltage{i}(j+1) - actual_voltage{i}(j);
        delta_v_data_before{i}(j) = use_data{2,i}(j+1) - use_data{2,i}(j);
        ramprate_voltage{i}(j) = delta_v_data{i}(j)/2;
        ramprate_voltage_before{i}(j) = delta_v_data_before{i}(j)/2;
    end
    indexes_visible_curve{i} = find(100*abs(ramprate_voltage{i})/230 > 0.36);
    indexes_visible_curve_before{i} = find(100*abs(ramprate_voltage_before{i})
       /230 > 0.36);
    amount_visible_flicker(i) = length(indexes_visible_curve{i});
    amount_visible_flicker_before(i) = length(indexes_visible_curve_before{i});
end
%%
figure
stem(amount_visible_flicker_before)
hold on
stem(amount_visible_flicker)
```

1.5. Voltage flicker curve mapping Matlab script

```
visible_flicker_curve = [1 40/80 18/80 20/80 23/80 0.36 30/80 38/80 43/80 50/80
   55/80 62/80 73/80 95/80 123/80 170/80 200/80];
annoying_flicker_curve = [115/80 55/80 35/80 45/80 55/80 68/80 76/80 93/80 110/80
    135/80 160/80 184/80 207/80 255/80 310/80 403/80 480/80];
visible_flicker_curve_polygon = [0 visible_flicker_curve 0 0];
annoying_flicker_curve_polygon = [0 annoying_flicker_curve 0 0];
time_dips_seconds = [0.05 0.1 0.2 0.3 1 2 3 6 12 30 60 120 180 360 720 1800
   3600];
time_dips_seconds_polygon = [0.025 time_dips_seconds 4000 0.025];
difference_voltage_percentage = ramprate_voltage*100/230;
difference_voltage_percentage_control = ramprate_voltage_SOC*100/230;
idx_before = inpolygon(difference_samplerate,abs(difference_voltage_percentage),
   time_dips_seconds_polygon, visible_flicker_curve_polygon);
ida_before = inpolygon(difference_samplerate,abs(difference_voltage_percentage),
   time_dips_seconds_polygon, annoying_flicker_curve_polygon);
idx_after = inpolygon(difference_samplerate_SOC,abs(
   difference_voltage_percentage_control),time_dips_seconds_polygon,
   visible_flicker_curve_polygon);
ida_after = inpolygon(difference_samplerate_SOC,abs(
   difference_voltage_percentage_control),time_dips_seconds_polygon,
   annoying_flicker_curve_polygon);
aruba_blue = [63/255 144/255 223/255];
aruba_red = [239/255 48/255 62/255];
aruba_yellow = [255/255 210/255 0/255];
royal_orange = [255/255 154/255 0/255];
stedin_grey = [77/255 77/255 77/255];
aruba_green = [132/255,194/255,37/255];
first_line_cmp = 'Voltage fluctuation relative to visible and annoying flicker
   curve';
first_line = compose(first_line_cmp);
control_line_cmp = 'Control Mode : %s';
control_line = compose(control_line_cmp,string(data.("Control Mode")(1)));
second_line_cmp = 'Testing time : %s - %s';
second_line = compose(second_line_cmp,time_string(1),time_string(end));
%% Divide the voltage fluctuations between inside and outside the operation
   window
indexes_pv_operation = [];
indexes_no_operation = [];
```

```
indexes_pv_operation = find(time_string(2:end) > '2017-08-04 05:00:00.000' &
    time_string(2:end) < '2017-08-04 22:00:00.000');
indexes_no_operation = find(time_string(2:end) < '2017-08-04 05:00:00.000' |
    time_string(2:end) > '2017-08-04 22:00:00.000');

indexes_before_operation_logical = false(length(difference_voltage_percentage),1)
    ;
indexes_before_no_operation_logical = false(length(difference_voltage_percentage)
    ,1);
annoying_indexes_before_operation_logical = false(length(
    difference_voltage_percentage),1);
annoying_indexes_before_no_operation_logical = false(length(
    difference_voltage_percentage),1);

indexes_after_operation_logical = false(length(
    difference_voltage_percentage_control),1);
indexes_after_no_operation_logical = false(length(
    difference_voltage_percentage_control),1);
annoying_indexes_after_operation_logical = false(length(
    difference_voltage_percentage_control),1);
annoying_indexes_after_no_operation_logical = false(length(
    difference_voltage_percentage_control),1);

indexes_before_operation_logical(indexes_no_operation) = logical(1);
indexes_before_no_operation_logical(indexes_pv_operation) = logical(1);
annoying_indexes_before_operation_logical(indexes_no_operation) = logical(1);
annoying_indexes_before_no_operation_logical(indexes_pv_operation) = logical(1);

indexes_after_operation_logical(indexes_no_operation) = logical(1);
indexes_after_no_operation_logical(indexes_pv_operation) = logical(1);
annoying_indexes_after_operation_logical(indexes_no_operation) = logical(1);
annoying_indexes_after_no_operation_logical(indexes_pv_operation) = logical(1);

indexes_before_operation = or(indexes_before_operation_logical,idx_before');
indexes_before_no_operation = or(indexes_before_no_operation_logical,idx_before')
    ;
annoying_indexes_before_operation = or(annoying_indexes_before_operation_logical,
    ida_before');
annoying_indexes_before_no_operation = or(
    annoying_indexes_before_no_operation_logical,ida_before');

indexes_after_operation = or(indexes_after_operation_logical,idx_after');
indexes_after_no_operation = or(indexes_after_no_operation_logical,idx_after');
annoying_indexes_after_operation = or(annoying_indexes_after_operation_logical,
    ida_after');
annoying_indexes_after_no_operation = or(
    annoying_indexes_after_no_operation_logical,ida_after');

% for j = 1:length(difference_voltage_percentage)
%     if time_string(j+1) < '2017-08-04 05:00:00.00' | time_string(j+1) >
    '2017-08-04 19:00:00.00'
%         indexes_before_operation_logical(j) = logical(1);
%     else
%         indexes_before_operation)logical(j) = idx_before(j);
%     end
% end
% for j = 1:length(difference_voltage_percentage)
%     if time_string(j+1) > '2017-08-04 05:00:00.00' & time_string(j+1) <
    '2017-08-04 19:00:00.00'
%         indexes_before_no_operation(j) = logical(1);
%     else
```

```
%         indexes_before_no_operation(j) = idx_before(j);
%     end
% end
%
% difference_voltage_operation = difference_voltage_percentage(
    indexes_pv_operation(1:end-1));
% difference_voltage_no_operation = difference_voltage_percentage(
    indexes_no_operation(1:end-1));
% difference_voltage_operation_control = difference_voltage_percentage_control(
    indexes_pv_operation(1:end-1));
% difference_voltage_no_operation_control = difference_voltage_percentage_control
    (indexes_no_operation);
%% Before ESS operation
fig2 = figure;
fig2.Position = [50 50 700 400];
ph = gobjects(1);
ph(1) = semilogx(time_dips_seconds,visible_flicker_curve,'LineWidth',2,'Color',
    royal_orange);
hold on
ph(2) = semilogx(time_dips_seconds,annoying_flicker_curve,'LineWidth',2,'Color',
    aruba_red);
set(gca, 'XDir','reverse');
grid on
hold on
ph(3) = scatter(difference_samplerate,abs(difference_voltage_percentage),'
    MarkerFaceColor',aruba_blue,'MarkerEdgeColor',aruba_blue-0.2, 'LineWidth',0.5)
    ;
hold on
ph(4) = scatter(difference_samplerate(~indexes_before_operation),abs(
    difference_voltage_percentage(~indexes_before_operation)),'MarkerFaceColor',
    royal_orange,'MarkerEdgeColor',aruba_green, 'LineWidth',0.5);
hold on
ph(5) = scatter(difference_samplerate(~indexes_before_no_operation),abs(
    difference_voltage_percentage(~indexes_before_no_operation)),60,'
    MarkerFaceColor',royal_orange,'MarkerEdgeColor',stedin_grey, 'LineWidth',1,'
    Marker','square');
% ph(5) = scatter(difference_samplerate(~indexes_before_no_operation),abs(
    difference_voltage_percentage(~indexes_before_no_operation)),'MarkerFaceColor
    ',royal_orange,'MarkerEdgeColor',royal_orange/1.25, 'LineWidth',0.5);
hold on
ph(6) = scatter(difference_samplerate(~annoying_indexes_before_operation),abs(
    difference_voltage_percentage(~annoying_indexes_before_operation)),'
    MarkerFaceColor',aruba_red,'MarkerEdgeColor',aruba_green, 'LineWidth',0.5);
hold on
ph(7) = scatter(difference_samplerate(~annoying_indexes_before_no_operation),abs(
    difference_voltage_percentage(~annoying_indexes_before_no_operation)),60,'
    MarkerFaceColor',aruba_red,'MarkerEdgeColor',stedin_grey, 'LineWidth',1,'
    Marker','square');
% ph(7) = scatter(difference_samplerate(~annoying_indexes_before_no_operation),
    abs(difference_voltage_percentage(~annoying_indexes_before_no_operation)),60,'
    MarkerFaceColor',aruba_red,'MarkerEdgeColor',aruba_red/1.25, 'LineWidth',0.5);
% scatter(difference_samplerate(~annoying_indexes_before_no_operation),abs(
    difference_voltage_percentage(~annoying_indexes_before_no_operation)),'
    MarkerFaceColor',aruba_red,'MarkerEdgeColor',aruba_red, 'LineWidth',0.5);
xlim([0.1 10])
ylim([0 2])
ylabel('Voltage dip (%)')
xlabel('Time between dips')
ax = gca;
ax.GridAlpha = 0.4;
ax.GridLineStyle = "-";
```

```
ax.MinorGridLineStyle = "-";
ax.XColor = [0 0 0];
ax.YColor = [0 0 0];
ax.GridColor = [0 0 0];
first_line = 'Voltage flicker curve';

third_line_cmp = '%d visible flicker(s) and %d annoying flicker(s) within the
    operating window';
third_line = compose(third_line_cmp,length(difference_samplerate_SOC(~
    indexes_before_operation))-length(difference_samplerate_SOC(~
    annoying_indexes_before_operation)),length(difference_samplerate_SOC(~
    annoying_indexes_before_operation)));
fourth_line_cmp = '%d visible flicker(s) and %d annoying flicker(s) outside the
    operating window';
fourth_line = compose(fourth_line_cmp,length(difference_samplerate_SOC(~
    indexes_before_no_operation))-length(difference_samplerate_SOC(~
    annoying_indexes_before_no_operation)),length(difference_samplerate_SOC(~
    annoying_indexes_before_no_operation)));

title({string(first_line);string(third_line);string(fourth_line)})
legend('Visible flicker threshold curve','Annoying flicker threshold curve','Non-
    visible flicker','Visible flicker within the operating window','Visible
    flicker outside of the operating window','Annoying flicker within the
    operating window', 'Annoying flicker outside of the operating window','
    Location','northeast')
% legend(ph([1,2,3,4,6]),'Visible flicker threshold curve','Annoying flicker
    threshold curve','Non-visible flicker','Visible flicker','Annoying flicker','
    Location', 'northeast');
% legend('Visible flicker threshold curve','Annoying flicker threshold curve','
    Non-visible flicker','Visible flicker','Annoying flicker')
%%
figure;
semilogx(time_dips_seconds,visible_flicker_curve,'LineWidth',2,'Color',
    royal_orange);
hold on
semilogx(time_dips_seconds,annoying_flicker_curve,'LineWidth',2,'Color',
    aruba_red);
set(gca, 'XDir','reverse');
grid on
hold on
scatter(difference_samplerate,abs(difference_voltage_percentage),'MarkerFaceColor
    ',aruba_blue,'MarkerEdgeColor',aruba_blue-0.2, 'LineWidth',0.5);
hold on
scatter(difference_samplerate(~indexes_before_operation),abs(
    difference_voltage_percentage(~indexes_before_operation)),'MarkerFaceColor',
    royal_orange,'MarkerEdgeColor',royal_orange, 'LineWidth',0.5);
% hold on
% scatter(difference_samplerate(~indexes_before_no_operation),abs(
    difference_voltage_percentage(~indexes_before_no_operation)),60,'
    MarkerFaceColor',royal_orange,'MarkerEdgeColor',stedin_grey, 'LineWidth',1,'
    Marker','square');
hold on
scatter(difference_samplerate(~annoying_indexes_before_operation),abs(
    difference_voltage_percentage(~annoying_indexes_before_operation)),'
    MarkerFaceColor',aruba_red,'MarkerEdgeColor',aruba_red, 'LineWidth',0.5);
hold on
% scatter(difference_samplerate(~annoying_indexes_before_no_operation),abs(
    difference_voltage_percentage(~annoying_indexes_before_no_operation)),60,'
    MarkerFaceColor',aruba_red,'MarkerEdgeColor',stedin_grey, 'LineWidth',1,'
    Marker','square');
scatter(difference_samplerate(~annoying_indexes_before_no_operation),abs(
```

```
    difference_voltage_percentage(~annoying_indexes_before_no_operation)),'
    MarkerFaceColor',aruba_red,'MarkerEdgeColor',aruba_red, 'LineWidth',0.5);
xlim([0.1 1])
ylim([0 1])
ylabel('Voltage dip (%)')
xlabel('Time between dips')
ax = gca;
ax.GridAlpha = 0.4;
ax.GridLineStyle = "-";
ax.MinorGridLineStyle = "-";
ax.XColor = [0 0 0];
ax.YColor = [0 0 0];
ax.GridColor = [0 0 0];

title({'Voltage fluctuation relative to visible and annoying flicker curve' ;'PV
    system example'})
% legend('Visible flicker threshold curve','Annoying flicker threshold curve','
    Non-visible flicker','Visible flicker within the operating window','Visible
    flicker outside of the operating window','Annoying flicker within the
    operating window', 'Annoying flicker outside of the operating window')
legend('Visible flicker threshold curve','Annoying flicker threshold curve','Non-
    visible flicker','Visible flicker','Annoying flicker')
%% After ESS operation
fig2 = figure;
fig2.Position = [50 50 700 400];
semilogx(time_dips_seconds,visible_flicker_curve,'LineWidth',2,'Color',
    royal_orange);
hold on
semilogx(time_dips_seconds,annoying_flicker_curve,'LineWidth',2,'Color',
    aruba_red);
set(gca, 'XDir','reverse');
grid on
hold on
scatter(difference_samplerate_SOC,abs(difference_voltage_percentage_control),'
    MarkerFaceColor',aruba_blue,'MarkerEdgeColor',aruba_blue-0.1, 'LineWidth',0.5)
    ;
hold on
scatter(difference_samplerate_SOC(~indexes_after_operation),abs(
    difference_voltage_percentage_control(~indexes_after_operation)),'
    MarkerFaceColor',royal_orange,'MarkerEdgeColor',aruba_green, 'LineWidth',0.5);
hold on
scatter(difference_samplerate_SOC(~indexes_before_no_operation),abs(
    difference_voltage_percentage_control(~indexes_before_no_operation)),60,'
    MarkerFaceColor',royal_orange,'MarkerEdgeColor',stedin_grey, 'LineWidth',1,'
    Marker','square');
hold on
scatter(difference_samplerate_SOC(~annoying_indexes_after_operation),abs(
    difference_voltage_percentage_control(~annoying_indexes_after_operation)),'
    MarkerFaceColor',aruba_red,'MarkerEdgeColor',aruba_green, 'LineWidth',0.5);
hold on
scatter(difference_samplerate_SOC(~annoying_indexes_before_no_operation),abs(
    difference_voltage_percentage_control(~annoying_indexes_before_no_operation))
    ,60,'MarkerFaceColor',aruba_red,'MarkerEdgeColor',stedin_grey, 'LineWidth',2,'
    Marker','square');
hold on
% scatter(difference_samplerate_SOC(632),abs(
    difference_voltage_percentage_control(632)),60,'MarkerFaceColor',aruba_blue,'
    MarkerEdgeColor',[0 0 0], 'LineWidth',2,'Marker','square');
xlim([0.1 10])
ylim([0 2])
ylabel('Voltage dip (%)')
```

```
xlabel('Time between dips')
first_line = 'Voltage flicker curve';
third_line_cmp = '%d visible flicker(s) and %d annoying flicker(s) within the
    operating window';
third_line = compose(third_line_cmp,length(difference_samplerate_SOC(~
    indexes_after_operation))-length(difference_samplerate_SOC(~
    annoying_indexes_after_operation)),length(difference_samplerate_SOC(~
    annoying_indexes_after_operation)));
fourth_line_cmp = '%d visible flicker(s) and %d annoying flicker(s) outside the
    operating window';
fourth_line = compose(fourth_line_cmp,length(difference_samplerate_SOC(~
    indexes_before_no_operation))-length(difference_samplerate_SOC(~
    annoying_indexes_before_no_operation)),length(difference_samplerate_SOC(~
    annoying_indexes_before_no_operation)));

ax = gca;
ax.GridAlpha = 0.4;
ax.GridLineStyle = "-";
ax.MinorGridLineStyle = "-";
ax.XColor = [0 0 0];
ax.YColor = [0 0 0];
ax.GridColor = [0 0 0];

title({string(first_line);string(third_line);string(fourth_line)})
legend('Visible flicker threshold curve','Annoying flicker threshold curve','Non-
    visible flicker','Visible flicker within the operating window','Visible
    flicker outside of the operating window','Annoying flicker within the
    operating window', 'Annoying flicker outside of the operating window','
    Location', 'northeast')
filename_cmp = 'Voltage_flicker_%s_%s_%s_%s_%s_%s_%d';
filename = compose(filename_cmp,string(data.("Control Mode")(1)),string(year(
    time_string(1))),string(month(time_string(1))),string(day(time_string(1))),
    string(hour(time_string(1))),string(minute(time_string(1))),floor(second(
    time_string(1))))
%%
saveas(figvoltage_flicker,string(filename),'tiff')
%%
date_extended = time_string(2:end);
%%
figure
yyaxis right
plot(date_extended,ramprate_voltage_SOC,'Color', 'b');
hold on
stem(date_extended(find(idx_after == 0)),ramprate_voltage_SOC(~idx_after), '
    LineWidth', 1, 'LineStyle','-','MarkerFaceColor',[255/255 153/255 51/255],'
    MarkerEdgeColor',[0 0 0], 'Color',[255/255 153/255 51/255]);
hold on
stem(date_extended(find(ida_after == 0)),ramprate_voltage_SOC(~ida_after), '
    LineWidth', 1, 'LineStyle','-','MarkerFaceColor',[255/255 0 0],'
    MarkerEdgeColor',[0 0 0], 'Color',[255/255 0 0]);
ylabel('Voltage ramprate (V/s)')
hold on
yyaxis left
plot(time_string,pv_power_MA_extended_SOC,'Color',[73/255 76/255 83/255])
ylabel('Power (W)')
title('Voltage ramprate and power output at PCC with supercapacitor
    implementation')
xlabel('Timestamps')
%%
figure
ax = gca;
```

```
yyaxis right
plot(date_extended,ramprate_voltage,'Color', 'b');
hold on
stem(date_extended(find(idx_before == 0)),ramprate_voltage(~idx_before), '
    LineWidth', 2, 'LineStyle','-','MarkerFaceColor',[255/255 153/255 51/255],'
    MarkerEdgeColor','k', 'Color',[255/255 153/255 51/255]);
hold on
stem(date_extended(find(ida_before == 0)),ramprate_voltage(~ida_before), '
    LineWidth', 2, 'LineStyle','-','MarkerFaceColor',[255/255 0 0],'
    MarkerEdgeColor','k', 'Color',[255/255 0 0]);
ylabel('Voltage ramprate (V/s)')
hold on
yyaxis left
plot(time_string,worst_day_pv(:,2),'Color',[73/255 76/255 83/255])
ylabel('Power (W)')
title('Voltage ramprate and power output at PCC without supercapacitor
    implementation')
xlabel('Timestamps')
%%
indexes_visible = [];
indexes_annoying = [];
indexes_visible_control = [];
indexes_annoying_control = [];
indexes_visible = find(idx_before == 0);
indexes_annoying = find(~idx_before & ~ida_before);
indexes_visible_control = find(idx_after == 0);
indexes_annoying_control = find(~idx_after & ~ida_after);
indexes_final = unique([indexes_visible indexes_visible_control]);

indexes_final_annoying = [];
for j = 1:length(indexes_visible)
    indexes_final_visible(j) = find(indexes_final == indexes_visible(j));
end
for i = 1:length(indexes_annoying)
    indexes_final_annoying(i) = find(indexes_final == indexes_annoying(i));
end
indexes_annoying_control = find(~idx_after & ~ida_after);
indexes_final_visible_control = [];
indexes_final_annoying_control = [];
for k = 1:length(indexes_annoying_control)
    indexes_final_annoying_control(k) = find(indexes_final ==
        indexes_annoying_control(k));
end
for j = 1:length(indexes_visible_control)
    indexes_final_visible_control(j) = find(indexes_final ==
        indexes_visible_control(j));
end
% indexes_final = unique([indexes_visible indexes_final_visible_control]);
%%
indexes_before = or(~idx_before,~ida_before);
figure
histogram(abs(difference_voltage_percentage(indexes_before)),100,'FaceColor', [0
    128/255 255/255])
hold on
histogram(abs(difference_voltage_percentage_control(indexes_before)),100, '
    FaceColor', [0 215/255 0])
%%
figure
Ax1 = axes;
% Ax2 = axes('Position',get(gca,'Position'),'Visible','Off');
p1 = stem(difference_voltage_percentage(indexes_final), 'LineWidth', 1, '
```

```
    LineStyle','-','MarkerFaceColor',[0 128/255 255/255],'MarkerEdgeColor',[0
    128/255 255/255], 'Color',[0 128/255 255/255],'MarkerSize',5);
hold on
p2 = stem(indexes_final_visible, difference_voltage_percentage(~idx_before), '
    LineWidth', 1, 'LineStyle','-','MarkerFaceColor',[255/255 153/255 51/255],'
    MarkerEdgeColor',[255/255 153/255 51/255], 'Color',[0 128/255 255/255],'
    MarkerSize',5);
hold on
p3 = stem(indexes_final_annoying,difference_voltage_percentage(~ida_before), '
    LineWidth', 1, 'LineStyle','-','MarkerFaceColor',[255/255 0 0],'
    MarkerEdgeColor',[255/255 0 0], 'Color',[0 128/255 255/255],'MarkerSize',5);
hold on
p4 = stem(difference_voltage_percentage_control(indexes_final), 'LineWidth', 1, '
    LineStyle','-','MarkerFaceColor',[0 215/255 0],'MarkerEdgeColor',[0 215/255
    0], 'Color',[0 215/255 0],'MarkerSize',5);
hold on
p5 = stem(indexes_final_visible_control,difference_voltage_percentage_control(~
    idx_after), 'LineWidth', 1, 'LineStyle','-','MarkerFaceColor',[255/255 153/255
     51/255],'MarkerEdgeColor',[255/255 153/255 51/255], 'Color',[0 215/255 0],'
    MarkerSize',5);
hold on
p6 = stem(indexes_final_annoying_control,difference_voltage_percentage_control(~
    ida_after), 'LineWidth', 1, 'LineStyle','-','MarkerFaceColor',[255/255 0 0],'
    MarkerEdgeColor',[255/255 0 0], 'Color',[0 215/255 0],'MarkerSize',5);
hold off
% Ax2 = copyobj(Ax1, gcf);
lh1 = legend(Ax1,[p1 p2 p3 p4 p5 p6],'Voltage fluctuations before ESS operation',
    'Visible flicker before ESS operation','Annoying flicker before ESS operation'
    , 'Voltage fluctuations after ESS operation','Visible flicker after ESS
    operation','Annoying flicker after ESS operation');
lh1.Location = 'NorthEast';
% lh2 = legend(Ax2,[p3 p4 p5],'Fixed flicker after ESS operation','Unfixed
    visible flicker after ESS operation','Unfixed annoying flicker after ESS
    operation');
% lh2.Location = 'SouthWestOutside';
title('Voltage fluctuations before and after ESS operation')
ylabel('Voltage fluctuation percentage (%)')
xlabel('Number of instances')
%%
indexes_voltage_ramp = 1:1:length(difference_voltage_percentage);
figure
plot(indexes_voltage_ramp,difference_voltage_percentage)
hold on
stem(indexes_voltage_ramp(~annoying_indexes_before_no_operation),
    difference_voltage_percentage(~annoying_indexes_before_no_operation),'Color',
    aruba_yellow)
hold on
stem(indexes_voltage_ramp(~annoying_indexes_after_no_operation),
    difference_voltage_percentage(~annoying_indexes_after_no_operation),'Color',
    aruba_red)
```

2

Experimental setup scripts

2.1. GUI Matlab code for DC connected ESS experimental setup

```
classdef test_setup_thesis_EbikeChargingStation < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                          matlab.ui.Figure
        TargetESSoutputpower1LayerTextAreaLabel_17  matlab.ui.control.Label
        TargetESSoutputpower1LayerTextAreaLabel_16  matlab.ui.control.Label
        TargetESSoutputpower1LayerTextAreaLabel_15  matlab.ui.control.Label
        TargetESSoutputpower1LayerTextAreaLabel_14  matlab.ui.control.Label
        TargetESSoutputpower1LayerTextAreaLabel_13  matlab.ui.control.Label
        TargetESSoutputpower1LayerTextAreaLabel_12  matlab.ui.control.Label
        TargetESSoutputpower1LayerTextAreaLabel_11  matlab.ui.control.Label
        TargetESSoutputpower1LayerTextAreaLabel_10  matlab.ui.control.Label
        TargetESSoutputpower1LayerTextAreaLabel_9  matlab.ui.control.Label
        TargetESSoutputpower1LayerTextAreaLabel_8  matlab.ui.control.Label
        TargetESSoutputpower1LayerTextAreaLabel_7  matlab.ui.control.Label
        TargetESSoutputpower1LayerTextAreaLabel_6  matlab.ui.control.Label
        DesignedandprogrammedbyLynrickWixLabel  matlab.ui.control.Label
        MeasuredBatteryPowerTextArea    matlab.ui.control.TextArea
        MeasuredBatteryPowerLabel       matlab.ui.control.Label
        TargetESSoutputpower1LayerTextAreaLabel_5  matlab.ui.control.Label
        TargetESSoutputpower1LayerTextAreaLabel_4  matlab.ui.control.Label
        TargetESSoutputpower1LayerTextAreaLabel_3  matlab.ui.control.Label
        TargetESSoutputpower1LayerTextAreaLabel_2  matlab.ui.control.Label
        TargetESSoutputpower4LayerTextArea  matlab.ui.control.TextArea
        TargetESSoutputpower4LayerTextAreaLabel  matlab.ui.control.Label
        TargetESSoutputpower3LayerTextArea  matlab.ui.control.TextArea
        TargetESSoutputpower3LayerTextAreaLabel  matlab.ui.control.Label
        TargetESSoutputpower2LayerTextArea  matlab.ui.control.TextArea
        TargetESSoutputpower2LayerTextAreaLabel  matlab.ui.control.Label
        TargetESSoutputpower1LayerTextArea  matlab.ui.control.TextArea
        TargetESSoutputpower1LayerTextAreaLabel  matlab.ui.control.Label
        MeasuredBatterySOCTextArea      matlab.ui.control.TextArea
        MeasuredBatterySOCLabel         matlab.ui.control.Label
        PowerControlElementsforWindowEditField  matlab.ui.control.
           NumericEditField
        PowerControlElementsforWindowEditFieldLabel  matlab.ui.control.Label
        VoltageControlElementsforWindowEditField  matlab.ui.control.
           NumericEditField
        VoltageControlElementsforWindowEditFieldLabel  matlab.ui.control.Label
        MeasuredVoltageFluctuationEditField  matlab.ui.control.EditField
        MeasuredVoltageFluctuationEditFieldLabel  matlab.ui.control.Label
```

```
    Image3                          matlab.ui.control.Image
    Image2                          matlab.ui.control.Image
    ExportdatatoCSVfileButton       matlab.ui.control.Button
    BatterySOCTextArea              matlab.ui.control.TextArea
    BatterySOCTextAreaLabel         matlab.ui.control.Label
    PoweratPCCTextArea              matlab.ui.control.TextArea
    PoweratPCCTextAreaLabel         matlab.ui.control.Label
    PauseSwitch                     matlab.ui.control.ToggleSwitch
    PauseSwitchLabel                matlab.ui.control.Label
    DropDown                        matlab.ui.control.DropDown
    DropDownLabel                   matlab.ui.control.Label
    TargetESSoutputpowerTextArea    matlab.ui.control.TextArea
    TargetESSoutputpowerTextAreaLabel  matlab.ui.control.Label
    TargetoutputvoltageTextArea     matlab.ui.control.TextArea
    TargetoutputvoltageLabel        matlab.ui.control.Label
    MeasuredPVoutputpowerTextArea   matlab.ui.control.TextArea
    MeasuredPVoutputpowerLabel      matlab.ui.control.Label
    StatusofESSInverterLamp_2       matlab.ui.control.Lamp
    StatusofSolarInverterLamp       matlab.ui.control.Lamp
    MeasuredVoltageTextArea         matlab.ui.control.TextArea
    MeasuredVoltageTextAreaLabel    matlab.ui.control.Label
    ReadwriteregistersButton        matlab.ui.control.Button
    InitializeConnectionsButton     matlab.ui.control.Button
    Image                           matlab.ui.control.Image
    UIAxes4                         matlab.ui.control.UIAxes
    UIAxes                          matlab.ui.control.UIAxes
    UIAxes2                         matlab.ui.control.UIAxes
    UIAxes3                         matlab.ui.control.UIAxes
end


properties (Access = public)
        modbusserver;
        modbusport;
        modbus_device_1;
        registertype = 'holdingregs';
        startadress = 38;
        count = 1;
        voltage;
        pv_data;
        worst_day_pv;
        row_temp;
        sensitivity;
        systemswitch;
        switch_value;
        turnonread = "On";
        error = 0;
        error_ESS_inverter;
        error_PV_inverter;
        controlmode = "Read Values Only";
        permission_to_control_voltage = 1;
        permission_to_control_power = 1;
        windowsize_voltage;
        windowsize_power;
        soc_battery;
        energy_ATEPS;
        soc = [];
        energy_Wh = [];
        ess_power = [];
        time_day = {};
        data_export = {};
```

```
        delay_messages = 1.67;
        correction_factor_discharging_power;
        correction_factor_charging_power;
        correction_factor_discharging_voltage;
        correction_factor_charging_voltage;
        soc_lower_limit_voltage;
        soc_upper_limit_voltage;
        soc_lower_limit_power;
        soc_upper_limit_power;
        idle_charge_power_ctrl;
        idle_discharge_power_ctrl;
        idle_charge_voltage_ctrl;
        idle_discharge_voltage_ctrl;
        soc_deadzone_lower_limit_voltage;
        soc_deadzone_upper_limit_voltage;
        soc_deadzone_lower_limit_power;
        soc_deadzone_upper_limit_power;
        power_threshold_idle_voltage;
        power_threshold_idle_power;
        ESSinverter;
        PVmeter;
        Gridmeter;
        config;
        idle_charge_voltage_factor;
        idle_discharge_voltage_factor;
        idle_charge_power_factor;
        idle_discharge_power_factor;
        nr_cycles;
end


% Callbacks that handle component events
methods (Access = private)

    % Button pushed function: InitializeConnectionsButton
    function InitializeConnectionsButtonPushed(app, event)
        app.ESSinverter = struct;
        app.PVmeter = struct;
        app.Gridmeter = struct;
        cla(app.UIAxes)
        cla(app.UIAxes2)
        cla(app.UIAxes3)
        %Initialize all connections to the modbus devices and
        %initializing all parameters
        app.config = readcell('C:\Users\ti\Desktop\Lynrick Thesis - Matlab
            Files\config_ebike.csv');
        %Initializing the connections with the devices
        app.ESSinverter.connection_type = app.config{1,2};
        app.ESSinverter.ip_address = app.config{2,2};
        app.ESSinverter.port = app.config{3,2};
        app.PVmeter.connection_type = app.config{4,2};
        app.PVmeter.com_port = app.config{5,2};
        app.Gridmeter.connection_type = app.config{6,2};
        app.Gridmeter.com_port = app.config{7,2};
        %Setting in the parameters from the config file
        app.sensitivity = app.config{8,2};
        app.energy_ATEPS = app.config{9,2};
        app.soc_battery = app.config{10,2};
        app.correction_factor_discharging_voltage = app.config{12,2};
        app.correction_factor_charging_voltage = app.config{13,2};
        app.soc_lower_limit_voltage = app.config{14,2};
```

```
app.soc_upper_limit_voltage = app.config{15,2};
app.windowsize_voltage = app.config{16,2};

app.idle_charge_voltage_ctrl = app.config{17,2};
app.idle_discharge_voltage_ctrl = app.config{18,2};
app.idle_charge_voltage_factor = app.config{17,4};
app.idle_discharge_voltage_factor = app.config{18,4};

app.soc_deadzone_lower_limit_voltage = app.config{19,2};
app.soc_deadzone_upper_limit_voltage = app.config{20,2};
app.power_threshold_idle_voltage = app.config{21,2};
app.correction_factor_discharging_power = app.config{23,2};
app.correction_factor_charging_power = app.config{24,2};
app.soc_lower_limit_power = app.config{25,2};
app.soc_upper_limit_power = app.config{26,2};
app.windowsize_power = app.config{27,2};

app.idle_charge_power_ctrl = app.config{28,2};
app.idle_discharge_power_ctrl = app.config{29,2};
app.idle_charge_power_factor = app.config{28,4};
app.idle_discharge_power_factor = app.config{29,4};

app.soc_deadzone_lower_limit_power = app.config{30,2};
app.soc_deadzone_upper_limit_power = app.config{31,2};
app.power_threshold_idle_power = app.config{32,2};

app.VoltageControlElementsforWindowEditField.Value = app.config
   {16,2};
app.PowerControlElementsforWindowEditField.Value = app.config{27,2};

%Reading in all of the addresses from the modbus devices
app.ESSinverter.set_power.address = app.config{34,2};
app.ESSinverter.set_power.type = app.config{34,3};
app.ESSinverter.set_power.scalefactor = app.config{34,4};
app.ESSinverter.set_power.unitid = app.config{34,5};
app.ESSinverter.set_power.register_type = app.config{34,6};

app.ESSinverter.error_check.address = app.config{35,2};
app.ESSinverter.error_check.type = app.config{35,3};
app.ESSinverter.error_check.scalefactor = app.config{35,4};
app.ESSinverter.error_check.unitid = app.config{35,5};
app.ESSinverter.error_check.register_type = app.config{35,6};

app.ESSinverter.output_voltage.address = app.config{36,2};
app.ESSinverter.output_voltage.type = app.config{36,3};
app.ESSinverter.output_voltage.scalefactor = app.config{36,4};
app.ESSinverter.output_voltage.unitid = app.config{36,5};
app.ESSinverter.output_voltage.register_type = app.config{36,6};

app.ESSinverter.output_current.address = app.config{37,2};
app.ESSinverter.output_current.type = app.config{37,3};
app.ESSinverter.output_current.scalefactor = app.config{37,4};
app.ESSinverter.output_current.unitid = app.config{37,5};
app.ESSinverter.output_current.register_type = app.config{37,6};

app.ESSinverter.output_power.address = app.config{38,2};
app.ESSinverter.output_power.type = app.config{38,3};
app.ESSinverter.output_power.scalefactor = app.config{38,4};
app.ESSinverter.output_power.unitid = app.config{38,5};
app.ESSinverter.output_power.register_type = app.config{38,6};
```

```
            app.PVmeter.output_power.address = app.config{40,2};
            app.PVmeter.output_power.type = app.config{40,3};
            app.PVmeter.output_power.scalefactor = app.config{40,4};
            app.PVmeter.output_power.unitid = app.config{40,5};
            app.PVmeter.output_power.register_type = app.config{40,6};

            app.PVmeter.error_check.address = app.config{41,2};
            app.PVmeter.error_check.type = app.config{41,3};
            app.PVmeter.error_check.scalefactor = app.config{41,4};
            app.PVmeter.error_check.unitid = app.config{41,5};
            app.PVmeter.error_check.register_type = app.config{41,6};

            app.Gridmeter.output_power.address = app.config{44,2};
            app.Gridmeter.output_power.type = app.config{44,3};
            app.Gridmeter.output_power.scalefactor = app.config{44,4};
            app.Gridmeter.output_power.unitid = app.config{44,5};
            app.Gridmeter.output_power.register_type = app.config{44,6};

            app.Gridmeter.output_voltage.address = app.config{43,2};
            app.Gridmeter.output_voltage.type = app.config{43,3};
            app.Gridmeter.output_voltage.scalefactor = app.config{43,4};
            app.Gridmeter.output_voltage.unitid = app.config{43,5};
            app.Gridmeter.output_voltage.register_type = app.config{43,6};

            app.ESSinverter.device = modbus(app.ESSinverter.connection_type,app.
                ESSinverter.ip_address,app.ESSinverter.port);
            msgbox("Initialized all connection to devices!");
            app.error_ESS_inverter = read(app.ESSinverter.device,app.ESSinverter.
                error_check.register_type,app.ESSinverter.error_check.address+1,1,
                app.ESSinverter.error_check.unitid);
            app.error_PV_inverter = read(app.ESSinverter.device,app.PVmeter.
                error_check.register_type,app.PVmeter.error_check.address+1,1,app.
                PVmeter.error_check.unitid);
            app.soc = [];
            app.ess_power(1) = 0;
            soc_axes_voltage = [0 20 app.soc_deadzone_lower_limit_voltage app.
                soc_deadzone_upper_limit_voltage 80 100];
%             correction_values_charging = [app.correction_factor_charging*(0-app
    .soc_lower_limit) app.correction_factor_charging*(20-app.soc_lower_limit) 0 0
    app.correction_factor_charging*(80-app.soc_upper_limit) app.
    correction_factor_charging*(100-app.soc_upper_limit)];
            correction_values_charging_voltage = [0 0 0 0 app.
                correction_factor_charging_voltage*(80-app.
                soc_deadzone_upper_limit_voltage) app.
                correction_factor_charging_voltage*(100-app.
                soc_deadzone_upper_limit_voltage)];
%             correction_values_discharging = [app.correction_factor_discharging
    *(0-app.soc_lower_limit) app.correction_factor_discharging*(20-app.
    soc_lower_limit) 0 0 app.correction_factor_discharging*(80-app.soc_upper_limit
    ) app.correction_factor_discharging*(100-app.soc_upper_limit)];
            correction_values_discharging_voltage = [app.
                correction_factor_discharging_voltage*(0-app.
                soc_deadzone_lower_limit_voltage) app.
                correction_factor_discharging_voltage*(20-app.
                soc_deadzone_lower_limit_voltage) 0 0 0 0 ];
            hold(app.UIAxes,'on')
            plot(app.UIAxes,soc_axes_voltage,abs(
                correction_values_charging_voltage)*100, 'Color', [255/255,
                209/255, 0],'LineWidth',2);
            plot(app.UIAxes,soc_axes_voltage,abs(
                correction_values_discharging_voltage)*100, 'Color', [51/255,
```

```
            153/255, 255/255],'LineWidth',2);
        hold(app.UIAxes,'off')
        app.UIAxes.XLim = [app.soc_lower_limit_voltage-2.5 app.
            soc_upper_limit_voltage+2.5];
        app.UIAxes.YLim = [-1 5];
        legend(app.UIAxes, 'Charge','Discharge')
        strTitleSoCDeadzone = "Charge = %0.3f and Discharge = %0.3f";
        titleSoCDeadzone = compose(strTitleSoCDeadzone, app.
            correction_factor_charging_voltage, app.
            correction_factor_discharging_voltage);
        app.UIAxes.Subtitle.String = titleSoCDeadzone;

        soc_axes_power = [0 20 app.soc_deadzone_lower_limit_power app.
            soc_deadzone_upper_limit_power 80 100];
%         correction_values_charging = [app.correction_factor_charging*(0-app
    .soc_lower_limit) app.correction_factor_charging*(20-app.soc_lower_limit) 0 0
    app.correction_factor_charging*(80-app.soc_upper_limit) app.
    correction_factor_charging*(100-app.soc_upper_limit)];
        correction_values_charging_power = [0 0 0 0 app.
            correction_factor_charging_power*(80-app.
            soc_deadzone_upper_limit_power) app.
            correction_factor_charging_power*(100-app.
            soc_deadzone_upper_limit_power)];
%         correction_values_discharging = [app.correction_factor_discharging
    *(0-app.soc_lower_limit) app.correction_factor_discharging*(20-app.
    soc_lower_limit) 0 0 app.correction_factor_discharging*(80-app.soc_upper_limit
    ) app.correction_factor_discharging*(100-app.soc_upper_limit)];
        correction_values_discharging_power = [app.
            correction_factor_discharging_power*(0-app.
            soc_deadzone_lower_limit_power) app.
            correction_factor_discharging_power*(20-app.
            soc_deadzone_lower_limit_power) 0 0 0 0 ];
        hold(app.UIAxes4,'on')
        plot(app.UIAxes4,soc_axes_power,abs(correction_values_charging_power)
            *100, 'Color', [255/255, 209/255, 0],'LineWidth',2);
        plot(app.UIAxes4,soc_axes_power,abs(
            correction_values_discharging_power)*100, 'Color', [51/255,
            153/255, 255/255],'LineWidth',2);
        hold(app.UIAxes4,'off')
        app.UIAxes4.XLim = [app.soc_lower_limit_power-2.5 app.
            soc_upper_limit_power+2.5];
        app.UIAxes4.YLim = [-1 5];
        legend(app.UIAxes, 'Charge','Discharge')
        strTitleSoCDeadzone = "Charge = %0.3f and Discharge = %0.3f";
        titleSoCDeadzone = compose(strTitleSoCDeadzone, app.
            correction_factor_charging_power, app.
            correction_factor_discharging_power);
        app.UIAxes4.Subtitle.String = titleSoCDeadzone;

        hold(app.UIAxes2,'on')
        area(app.UIAxes2, [0 0 app.soc_lower_limit_voltage app.
            soc_lower_limit_voltage], [0 app.power_threshold_idle_voltage app.
            power_threshold_idle_voltage 0], 'FaceColor', [0/255 0/255
            204/255])
        area(app.UIAxes2, [0 0 app.soc_lower_limit_voltage app.
            soc_lower_limit_voltage], [0 -app.power_threshold_idle_voltage -
            app.power_threshold_idle_voltage 0], 'FaceColor', [0/255 0/255
            204/255])
        area(app.UIAxes2, [app.soc_upper_limit_voltage app.
            soc_upper_limit_voltage 100 100], [0 app.
            power_threshold_idle_voltage app.power_threshold_idle_voltage 0],
```

```
            'FaceColor', [0/255 0/255 204/255])
        area(app.UIAxes2, [app.soc_upper_limit_voltage app.
            soc_upper_limit_voltage 100 100], [0 -app.
            power_threshold_idle_voltage -app.power_threshold_idle_voltage 0],
             'FaceColor', [0/255 0/255 204/255])
%             area(app.UIAxes2, [app.soc_upper_limit 100], [-20 20], 'FaceColor',
    [255/255 92/255 57/255])
        hold(app.UIAxes2,'off')
        app.UIAxes2.XLim = [app.soc_lower_limit_voltage-0.5 app.
            soc_upper_limit_voltage+0.5];
        app.UIAxes2.YLim = [-100 100];
        legend(app.UIAxes2,'Window of Operation')
        strTitleIdleVoltage = "Charge = %0.1f and Discharge = %0.1f";
        titleIdleVoltage = compose(strTitleIdleVoltage, app.
            idle_charge_voltage_factor, app.idle_discharge_voltage_factor);
        app.UIAxes2.Title.String = 'Idle Charge/Discharge Window of Operation
             - Voltage Control';
        app.UIAxes2.Subtitle.String = titleIdleVoltage;
        hold(app.UIAxes3,'on')
        area(app.UIAxes3, [0 0 app.soc_lower_limit_power app.
            soc_lower_limit_power], [0 app.power_threshold_idle_power app.
            power_threshold_idle_power 0], 'FaceColor', [127/255 0/255
            255/255])
        area(app.UIAxes3, [0 0 app.soc_lower_limit_power app.
            soc_lower_limit_power], [0 -app.power_threshold_idle_power -app.
            power_threshold_idle_power 0], 'FaceColor', [127/255 0/255
            255/255])
        area(app.UIAxes3, [app.soc_upper_limit_power app.
            soc_upper_limit_power 100 100], [0 app.power_threshold_idle_power
            app.power_threshold_idle_power 0], 'FaceColor', [127/255 0/255
            255/255])
        area(app.UIAxes3, [app.soc_upper_limit_power app.
            soc_upper_limit_power 100 100], [0 -app.power_threshold_idle_power
             -app.power_threshold_idle_power 0], 'FaceColor', [127/255 0/255
            255/255])
%             area(app.UIAxes2, [app.soc_upper_limit 100], [-20 20], 'FaceColor',
    [255/255 92/255 57/255])
        hold(app.UIAxes3,'off')
        app.UIAxes3.XLim = [app.soc_lower_limit_power-0.5 app.
            soc_upper_limit_power+0.5];
        app.UIAxes3.YLim = [-100 100];
        legend(app.UIAxes3,'Window of Operation')
        strTitleIdlePower = "Charge = %0.1f and Discharge = %0.1f";
        titleIdlePower = compose(strTitleIdlePower, app.
            idle_charge_power_factor, app.idle_discharge_power_factor);
        app.UIAxes3.Title.String = 'Idle Charge/Discharge Window of Operation
             - Power Control';
        app.UIAxes3.Subtitle.String = titleIdlePower;
        %First check if there are any errors to the inverters
        if app.error_ESS_inverter == 0
            app.StatusofESSInverterLamp_2.Color = [0/255 255/255 0/255];
        elseif app.error_ESS_inverter > 0
            app.StatusofESSInverterLamp_2.Color = [255/255 92/255 57/255];
            formatError = "ESS inverter experienced error : %d";
            errormessage = compose(formatError,app.error_ESS_inverter);
            msgbox(errormessage);
        end

        if app.error_PV_inverter == 0
            app.StatusofSolarInverterLamp.Color = [0/255 255/255 0/255];
        elseif app.error_PV_inverter == 1
```

```
            app.StatusofSolarInverterLamp.Color = [255/255 92/255 57/255];
        end
        exportapp(app.UIFigure,'appcontent_DC_setup.pdf')
    end

    % Button pushed function: ReadwriteregistersButton
    function ReadwriteregistersButtonPushed(app, event)
        %Initialize all of the used variables
        app.nr_cycles = 1;
        i = app.nr_cycles;
        data = zeros(app.windowsize_voltage,2);
        data_array = data(:,1)+230;
        data_array_voltage = cat(2,data_array,data);
        data = zeros(app.windowsize_power,2);
        data_array = data(:,1)+230;
        data_array_power = cat(2,data_array,data);
        sinus_MA_voltage_output_day = [];
        sinus_MA_power_output_day = [];
        voltage_array = [];
        %Live plot the ESS power
        h = animatedline('Color',[255/255, 209/255, 0],'LineWidth',2);
        title('ESS Power')

        % window_size
        while(app.error == 0)
            %State of charge calculation of the BESS
            tic;
            if i == 1
                app.soc(i) = app.soc_battery;
                app.energy_Wh(i) = app.energy_ATEPS*app.soc_battery/100;
                app.ess_power(i) = 0;
            else
                app.ess_power(i) = battery_power;
                energy_used(i) = (app.ess_power(i) + app.ess_power(i-1))*
                   elapsed_time*0.5/3600;
                app.soc(i) = app.soc(i-1) - (energy_used(i)/app.energy_ATEPS)
                   *100;
                app.energy_Wh(i) = app.energy_Wh(i-1) - energy_used(i);
            end
            % if i == 1
            %     app.soc(i) = app.soc_battery;
            %     app.energy_Wh(i) = app.energy_ATEPS*app.soc_battery/100;
            %     app.ess_power(i) = 0;
            % end
            %Check if there are any errors to the inverter and or it
            %for the general error checking
            app.error_ESS_inverter = read(app.ESSinverter.device,app.
               ESSinverter.error_check.register_type,app.ESSinverter.
               error_check.address+1,1,app.ESSinverter.error_check.unitid);
            if app.error_ESS_inverter > 0
                error_ESS_inverter_binary = 1;
            else
                error_ESS_inverter_binary = 0;
            end
            app.error_PV_inverter = read(app.ESSinverter.device,app.PVmeter.
               error_check.register_type,app.PVmeter.error_check.address+1,1,
               app.PVmeter.error_check.unitid);
            if app.error_PV_inverter > 0
                error_PV_inverter_binary = 1;
            else
                error_PV_inverter_binary = 0;
```

```
                end
%                 app.error_PV_inverter = read(app.modbus_device_1,'coils
    ',11,1,1);
                app.error  = or(app.error_PV_inverter,error_ESS_inverter_binary);
                axis([i-30,i,-1500,1500])
                xlabel('Datapoint');
                ylabel('Power (W)')
                if string(app.turnonread) == "On"

                    % write(app.modbus_device_1,'holdingregs',app.startadress,app
                        .pv_data(i),1,'single')
                    % app.voltage = read(app.modbus_device_1,'holdingregs',app.
                        startadress,app.count,'single');
                    % voltage_array(i) = app.voltage;
                    % plot(voltage_array(1:i),'Color',[255/255, 209/255, 0])
                    % drawnow
                    % i = i+1;
                    % xlim([i-50 i])
                    % pause(0.10)

                    if (string(app.controlmode) == "Voltage Control")
                        app.permission_to_control_power = 1;
                        %Read values first = read(app.ESSinverter.device,app.
                            ESSinverter.error_check.register_type,app.ESSinverter.
                            error_check.address+1,1,app.ESSinverter.error_check.
                            unitid);
                        pv_power = read(app.ESSinverter.device,app.PVmeter.
                            output_power.register_type,app.PVmeter.output_power.
                            address+1,1,app.PVmeter.output_power.unitid,app.
                            PVmeter.output_power.type)/app.PVmeter.output_power.
                            scalefactor;
                        grid_voltage = read(app.ESSinverter.device,app.Gridmeter.
                            output_voltage.register_type,app.Gridmeter.
                            output_voltage.address+1,1,app.Gridmeter.
                            output_voltage.unitid,app.Gridmeter.output_voltage.
                            type)/app.Gridmeter.output_voltage.scalefactor;
%                         ESS_set_output_power = read(app.modbus_device_1,"
    inputregs",843,1,100,'int16')/1;
                        battery_power = -read(app.ESSinverter.device, "inputregs
                            ", 843, 1, 100, 'int16')/1;
                        grid_power = -read(app.ESSinverter.device,app.Gridmeter.
                            output_power.register_type,app.Gridmeter.output_power.
                            address+1,1,app.Gridmeter.output_power.unitid,app.
                            Gridmeter.output_power.type)/app.Gridmeter.
                            output_power.scalefactor;
                        ac_load_power = read(app.ESSinverter.device,"inputregs
                            ",818,1,100,'uint16')/1;
                        %First layer of control - Ascending Cosinusoidal
                        %Weighted Moving Average
                        app.DropDown.FontColor = [0/255 0/255 204/255];
                        app.DropDown.FontWeight = 'bold';
                        weighting_multiplier = [];
                        weighting_multiplier = linspace(1,app.windowsize_voltage,
                            app.windowsize_voltage);
                        weighting_fraction = [];
                        for k = 1:app.windowsize_voltage
                            weighting_fraction(k) = cos((1/app.windowsize_voltage
                                )*k*pi/2)^2;
                        end
                        weighting_fraction = flip(weighting_fraction);
                        sinus_MA_voltage_output_day(i) = sum(data_array_voltage
```

```
    (:,1).*weighting_fraction')/sum(weighting_fraction);
if app.permission_to_control_voltage < app.
    windowsize_voltage+1
     output_voltage(i) = grid_voltage;
else
     output_voltage(i) = sinus_MA_voltage_output_day(i);
end
delta_v = output_voltage(i) - grid_voltage;
ESS_set_output_power_first = delta_v/app.sensitivity;
%Second layer of control - SOC deadzone control
if ESS_set_output_power_first > 0
     if app.soc(i) > app.soc_deadzone_lower_limit_voltage
        && app.soc(i) < app.
        soc_deadzone_upper_limit_voltage
         ESS_set_output_power_second =
            ESS_set_output_power_first;
     elseif app.soc(i) < app.
        soc_deadzone_lower_limit_voltage
         delta_soc = app.soc(i) - app.
            soc_deadzone_lower_limit_voltage;
         ESS_set_output_power_second =
            ESS_set_output_power_first*(1-abs(delta_soc)*
            app.correction_factor_discharging_voltage);
     elseif app.soc(i) > app.
        soc_deadzone_upper_limit_voltage
         ESS_set_output_power_second =
            ESS_set_output_power_first;
     end
else
     if app.soc(i) > app.soc_deadzone_lower_limit_voltage
        && app.soc(i) < app.
        soc_deadzone_upper_limit_voltage
         ESS_set_output_power_second =
            ESS_set_output_power_first;
     elseif app.soc(i) < app.
        soc_deadzone_lower_limit_voltage
         ESS_set_output_power_second =
            ESS_set_output_power_first;
     elseif app.soc(i) > app.
        soc_deadzone_upper_limit_voltage
         delta_soc = app.soc(i) - app.
            soc_deadzone_upper_limit_voltage;
         ESS_set_output_power_second =
            ESS_set_output_power_first*(1-abs(delta_soc)*
            app.correction_factor_charging_voltage);
     end
end
%Third layer of control - Idle Charge/Discharge
if (ESS_set_output_power_second < app.
    power_threshold_idle_voltage &&
    ESS_set_output_power_second > -app.
    power_threshold_idle_voltage)
     if app.soc(i) > app.soc_upper_limit_power
         % ESS_set_output_power_third = app.
            idle_discharge_power_ctrl;
         ESS_set_output_power_third_ideal = app.
            idle_discharge_voltage_ctrl;
         ESS_set_output_power_third = app.
            idle_discharge_voltage_ctrl;
         % if ESS_set_output_power_third < app.
            idle_discharge_voltage_ctrl
```

```
            %     ESS_set_output_power_third = app.
                idle_discharge_voltage_ctrl;
            % end
        elseif app.soc(i) < app.soc_lower_limit_power
            ESS_set_output_power_third_ideal = app.
                idle_charge_voltage_ctrl;
            ESS_set_output_power_third = app.
                idle_charge_voltage_ctrl;
            % if ESS_set_output_power_third < app.
                idle_charge_voltage_ctrl
            %     ESS_set_output_power_third = app.
                idle_charge_voltage_ctrl;
            % end
        else
            ESS_set_output_power_third_ideal =
                ESS_set_output_power_second;
            ESS_set_output_power_third =
                ESS_set_output_power_second;
            % if pv_power < 100
            %     ESS_set_output_power_third = -100;
            % end
        end
    else
        ESS_set_output_power_third_ideal =
            ESS_set_output_power_second;
        ESS_set_output_power_third =
            ESS_set_output_power_second;
    end
    %Checking if window is filled with new values to
    %proceed
    if app.permission_to_control_voltage < app.
        windowsize_voltage+1
        ESS_set_output_power_pre_prep = 0;%-0.1667*pv_power
            ;%0.4*pv_power;
        ESS_set_output_power_third_ideal = 0;
            % if pv_power < 100
            %     ESS_set_output_power_third = -100;
            %     ESS_set_output_power_third_ideal = 0;
            % end
    else
        ESS_set_output_power_pre_prep =
            ESS_set_output_power_third;
        ESS_set_output_power_final_ideal =
            ESS_set_output_power_third_ideal;
    end
    %Window of operation control layer
    now_data = datetime('now','Format','d-MMM-y HH:mm:ss.SSS'
        );
    if hour(now_data) > 5 && hour(now_data) < 22
        ESS_set_output_power_pre =
            ESS_set_output_power_pre_prep;
        ESS_set_output_power_final_ideal =
            ESS_set_output_power_third_ideal;
    else
        ESS_set_output_power_pre = 0;
        ESS_set_output_power_final_ideal = 0;
    end
    ESS_set_output_power_floor = floor(
        ESS_set_output_power_pre);
    % Hard coding to not overload the inverter and
    % battery
```

```
                    if ESS_set_output_power_floor > 500
                        ESS_set_output_power = 500;
                        ESS_set_output_power_final_ideal = 500;
                    elseif ESS_set_output_power_floor < -1000
                        ESS_set_output_power = -1000;
                        ESS_set_output_power_final_ideal = -1000;
                    else
                        ESS_set_output_power = ESS_set_output_power_floor;
                        ESS_set_output_power_final_ideal = 0;
                    end
                    ESS_set_output_power_pre = ESS_set_output_power;
                    ESS_set_output_power = -floor(ESS_set_output_power_pre +
                       pv_power + -0.1667*pv_power);
                    %Change color of the text field to highlight which
                    %control mode is used at the moment
                    app.TargetESSoutputpowerTextArea.Value = num2str(-
                       ESS_set_output_power);
                    app.TargetESSoutputpowerTextArea.FontColor = [0/255 0/255
                        204/255];
                    app.TargetESSoutputpowerTextArea.FontWeight = 'bold';
                    app.MeasuredVoltageTextArea.FontColor = [0/255 0/255
                       204/255];
                    app.MeasuredVoltageTextArea.FontWeight = 'bold';
                    app.MeasuredPVoutputpowerTextArea.FontColor = [0/255
                       0/255 0/255];
                    app.MeasuredPVoutputpowerTextArea.FontWeight = 'normal';
                    app.row_temp = [grid_voltage pv_power output_voltage(i)];
                    app.permission_to_control_voltage = app.
                       permission_to_control_voltage + 1;
                elseif (string(app.controlmode) == "Power Control")
                    app.permission_to_control_voltage = 1;
                    %Read values first = read(app.ESSinverter.device,app.
                       ESSinverter.error_check.register_type,app.ESSinverter.
                       error_check.address+1,1,app.ESSinverter.error_check.
                       unitid);
                    pv_power = read(app.ESSinverter.device,app.PVmeter.
                       output_power.register_type,app.PVmeter.output_power.
                       address+1,1,app.PVmeter.output_power.unitid,app.
                       PVmeter.output_power.type)/app.PVmeter.output_power.
                       scalefactor;
                    grid_voltage = read(app.ESSinverter.device,app.Gridmeter.
                       output_voltage.register_type,app.Gridmeter.
                       output_voltage.address+1,1,app.Gridmeter.
                       output_voltage.unitid,app.Gridmeter.output_voltage.
                       type)/app.Gridmeter.output_voltage.scalefactor;
%                     ESS_set_output_power = read(app.modbus_device_1,"
    inputregs",843,1,100,'int16')/1;
                    battery_power = -read(app.ESSinverter.device, "inputregs
                       ", 843, 1, 100, 'int16')/1;
                    grid_power = -read(app.ESSinverter.device,app.Gridmeter.
                       output_power.register_type,app.Gridmeter.output_power.
                       address+1,1,app.Gridmeter.output_power.unitid,app.
                       Gridmeter.output_power.type)/app.Gridmeter.
                       output_power.scalefactor;
                    ac_load_power = read(app.ESSinverter.device,"inputregs
                       ",818,1,100,'uint16')/1;
                    %First layer of control - Ascending Cosinusoidal
                    %Weighted Moving Average
                    app.DropDown.FontColor = [127/255 0/255 255/255];
                    app.DropDown.FontWeight = 'bold';
                    weighting_multiplier = [];
```

```
weighting_multiplier = linspace(1,app.windowsize_power,
   app.windowsize_power);
weighting_fraction = [];
for k = 1:app.windowsize_power
    weighting_fraction(k) = cos((1/app.windowsize_power)*
       k*pi/2)^2;
end
weighting_fraction = flip(weighting_fraction);
sinus_MA_power_output_day(i) = sum(data_array_power(:,2)
   .*weighting_fraction')/sum(weighting_fraction);
if app.permission_to_control_power < app.windowsize_power
   +1
    output_power(i) = pv_power;
else
    output_power(i) = sinus_MA_power_output_day(i);
end
ESS_set_output_power_first = output_power(i) - pv_power;
%Second layer of control - SOC deadzone control
if ESS_set_output_power_first > 0
    if app.soc(i) > app.soc_deadzone_lower_limit_power &&
        app.soc(i) < app.soc_deadzone_upper_limit_power
        ESS_set_output_power_second =
           ESS_set_output_power_first;
    elseif app.soc(i) < app.
       soc_deadzone_lower_limit_power
        delta_soc = app.soc(i) - app.
           soc_deadzone_lower_limit_power;
        ESS_set_output_power_second =
           ESS_set_output_power_first*(1-abs(delta_soc)*
           app.correction_factor_discharging_power);
    elseif app.soc(i) > app.
       soc_deadzone_upper_limit_power
        ESS_set_output_power_second =
           ESS_set_output_power_first;
    end
else
    if app.soc(i) > app.soc_deadzone_lower_limit_power &&
        app.soc(i) < app.soc_deadzone_upper_limit_power
        ESS_set_output_power_second =
           ESS_set_output_power_first;
    elseif app.soc(i) < app.
       soc_deadzone_lower_limit_power
        ESS_set_output_power_second =
           ESS_set_output_power_first;
    elseif app.soc(i) > app.
       soc_deadzone_upper_limit_power
        delta_soc = app.soc(i) - app.
           soc_deadzone_upper_limit_power;
        ESS_set_output_power_second =
           ESS_set_output_power_first*(1-abs(delta_soc)*
           app.correction_factor_charging_power);
    end
end
%Third layer of control - Idle Charge/Discharge
if (ESS_set_output_power_second < app.
   power_threshold_idle_power &&
   ESS_set_output_power_second > -app.
   power_threshold_idle_power)
    if app.soc(i) > app.soc_upper_limit_power
        ESS_set_output_power_third_ideal = app.
           idle_discharge_power_ctrl;
```

```
            ESS_set_output_power_third = app.
                idle_discharge_power_ctrl;
            % if ESS_set_output_power_third < app.
                idle_discharge_power_ctrl
            %     ESS_set_output_power_third = app.
                idle_discharge_power_ctrl;
            % end
        elseif app.soc(i) < app.soc_lower_limit_power
            ESS_set_output_power_third_ideal = app.
                idle_charge_power_ctrl;
            ESS_set_output_power_third = app.
                idle_charge_power_ctrl;
            % if ESS_set_output_power_third < app.
                idle_charge_power_ctrl
            %     ESS_set_output_power_third = app.
                idle_charge_power_ctrl;
            % end
        else
            ESS_set_output_power_third_ideal =
                ESS_set_output_power_second;
            ESS_set_output_power_third =
                ESS_set_output_power_second;
            % if pv_power < 100
            %     ESS_set_output_power_third = -100;
            % end
        end
    else
        ESS_set_output_power_third_ideal =
            ESS_set_output_power_second;
        ESS_set_output_power_third =
            ESS_set_output_power_second;
        % if pv_power < 100
        %     ESS_set_output_power_third = -100;
        % end
    end
    %Checking if window is filled with new values to
    %proceed
    if app.permission_to_control_power < app.windowsize_power
        +1
        ESS_set_output_power_pre_prep = 0;
        ESS_set_output_power_third_ideal = 0;
            % if pv_power < 100
            %     ESS_set_output_power_third = -100;
            %     ESS_set_output_power_third_ideal = 0;
            % end
    else
        ESS_set_output_power_pre_prep =
            ESS_set_output_power_third;
        ESS_set_output_power_third_ideal =
            ESS_set_output_power_third;

    end
    %Window of operation control layer
    now_data = datetime('now','Format','d-MMM-y HH:mm:ss.SSS'
        );
    if hour(now_data) > 5 && hour(now_data) < 22
        ESS_set_output_power_pre =
            ESS_set_output_power_pre_prep;
        ESS_set_output_power_final_ideal =
            ESS_set_output_power_third_ideal;
        % if pv_power < 100
```

```
                        %     ESS_set_output_power_third = -100;
                        %     ESS_set_output_power_final_ideal =
                            ESS_set_output_power_third_ideal;
                        % end
                    else
                        ESS_set_output_power_pre = 0;
                        ESS_set_output_power_final_ideal = 0;
                    end
                    ESS_set_output_power = -floor(ESS_set_output_power_pre +
                        pv_power + -0.1667*pv_power);
%                     set_voltage_power = grid_voltage + app.sensitivity*
    ESS_set_output_power;
                    %Change color of the text field to highlight which
                    %control mode is used at the moment
                    app.TargetESSoutputpowerTextArea.Value = num2str(-
                        ESS_set_output_power);
                    app.row_temp = [grid_voltage pv_power grid_voltage];
                    app.MeasuredVoltageTextArea.FontColor = [0/255 0/255
                        0/255];
                    app.MeasuredVoltageTextArea.FontWeight = 'normal';
                    app.TargetESSoutputpowerTextArea.FontColor = [127/255
                        0/255 255/255];
                    app.TargetESSoutputpowerTextArea.FontWeight = 'bold';
                    app.MeasuredPVoutputpowerTextArea.FontColor = [127/255
                        0/255 255/255];
                    app.MeasuredPVoutputpowerTextArea.FontWeight = 'bold';
                    app.permission_to_control_power = app.
                        permission_to_control_power + 1;
                elseif (string(app.controlmode) == "Read Values Only")
                    now_data = datetime('now','Format','d-MMM-y HH:mm:ss.SSS'
                        );
                    app.permission_to_control_voltage = 1;
                    app.permission_to_control_power = 1;
                    % Read the values first
                    pv_power = read(app.ESSinverter.device,app.PVmeter.
                        output_power.register_type,app.PVmeter.output_power.
                        address+1,1,app.PVmeter.output_power.unitid,app.
                        PVmeter.output_power.type)/app.PVmeter.output_power.
                        scalefactor;
                    grid_voltage = read(app.ESSinverter.device,app.Gridmeter.
                        output_voltage.register_type,app.Gridmeter.
                        output_voltage.address+1,1,app.Gridmeter.
                        output_voltage.unitid,app.Gridmeter.output_voltage.
                        type)/app.Gridmeter.output_voltage.scalefactor;
                    % ESS_set_output_power = read(app.modbus_device_1,"
                        inputregs",843,1,100,'int16')/1;
                    battery_power = -read(app.ESSinverter.device, "inputregs
                        ", 843, 1, 100, 'int16')/1;
                    grid_power = -read(app.ESSinverter.device,app.Gridmeter.
                        output_power.register_type,app.Gridmeter.output_power.
                        address+1,1,app.Gridmeter.output_power.unitid,app.
                        Gridmeter.output_power.type)/app.Gridmeter.
                        output_power.scalefactor;
                    ac_load_power = read(app.ESSinverter.device,"inputregs
                        ",818,1,100,'uint16')/1;
                    % Write to ESS inverter to not pump any power
                    % write(app.ESSinverter.device,app.ESSinverter.set_power.
                        register_type,app.ESSinverter.set_power.address+1,
                        floor(pv_power),app.ESSinverter.set_power.unitid,app.
                        ESSinverter.set_power.type);
                    app.DropDown.FontColor = [0/255 0/255 0/255];
```

```
        app.DropDown.FontWeight = 'normal';
        app.TargetESSoutputpowerTextArea.FontColor = [0/255 0/255
            0/255];
        app.TargetESSoutputpowerTextArea.FontWeight = 'normal';
        app.MeasuredVoltageTextArea.FontColor = [0/255 0/255
           0/255];
        app.MeasuredVoltageTextArea.FontWeight = 'normal';
        app.MeasuredPVoutputpowerTextArea.FontColor = [0/255
           0/255 0/255];
        app.MeasuredPVoutputpowerTextArea.FontWeight = 'normal';
        % ESS_set_output_power = -floor(pv_power);
        ESS_set_output_power = -floor(0.1833*pv_power);
        ESS_set_output_power_first = ESS_set_output_power;
        ESS_set_output_power_second = ESS_set_output_power;
        ESS_set_output_power_third = ESS_set_output_power;
        ESS_set_output_power_pre = ESS_set_output_power;
        ESS_set_output_power_final_ideal = ESS_set_output_power;
        app.TargetESSoutputpowerTextArea.Value = num2str(-
           ESS_set_output_power);
        app.row_temp = [grid_voltage pv_power grid_voltage];
    end
    %Update MA array for both Voltage and Power control
    temp_row_voltage = [app.row_temp; data_array_voltage];
    data_array_voltage = temp_row_voltage(1:app.
       windowsize_voltage,:);
    temp_row_power = [app.row_temp; data_array_power];
    data_array_power = temp_row_power(1:app.windowsize_power,:);
    %Measure the voltage fluctuations and output it on the
    %text field.
    battery_soc_victron = read(app.ESSinverter.device, "inputregs
       ", 844, 1, 100, 'uint16');
    app.MeasuredBatterySOCTextArea.Value = num2str(
       battery_soc_victron);
    app.TargetESSoutputpower1LayerTextArea.Value = num2str(
       ESS_set_output_power_first);
    app.TargetESSoutputpower2LayerTextArea.Value = num2str(
       ESS_set_output_power_second);
    app.TargetESSoutputpower3LayerTextArea.Value = num2str(
       ESS_set_output_power_third);
    app.TargetESSoutputpower4LayerTextArea.Value = num2str(
       ESS_set_output_power_pre);
    app.MeasuredBatteryPowerTextArea.Value = num2str(
       battery_power);
    %Write to the ESS the power command = read(app.ESSinverter.
       device,app.ESSinverter.error_check.register_type,app.
       ESSinverter.error_check.address+1,1,app.ESSinverter.
       error_check.unitid);
    if string(app.controlmode) == "Read Values Only"
        if hour(now_data) > 5 && hour(now_data) < 22
            ESS_set_output_power_actual = 0;
        end
    else
        if hour(now_data) > 5 && hour(now_data) < 22
            write(app.ESSinverter.device,app.ESSinverter.
               set_power.register_type,app.ESSinverter.set_power.
               address+1,ESS_set_output_power,app.ESSinverter.
               set_power.unitid,app.ESSinverter.set_power.type);
            ESS_set_output_power_actual = read(app.ESSinverter.
               device,app.ESSinverter.set_power.register_type,app
               .ESSinverter.set_power.address+1,1,app.ESSinverter
               .set_power.unitid,app.ESSinverter.set_power.type);
```

```
                    else
                        ESS_set_output_power_actual = 0;
                        % write(app.ESSinverter.device,app.ESSinverter.
                            set_power.register_type,app.ESSinverter.set_power.
                            address+1,ESS_set_output_power,app.ESSinverter.
                            set_power.unitid,app.ESSinverter.set_power.type);
                    end
                end
                pause(app.delay_messages);
                elapsed_time = toc;
                grid_voltage_new = read(app.ESSinverter.device,app.Gridmeter.
                    output_voltage.register_type,app.Gridmeter.output_voltage.
                    address+1,1,app.Gridmeter.output_voltage.unitid,app.
                    Gridmeter.output_voltage.type)/app.Gridmeter.
                    output_voltage.scalefactor;
                grid_power_new = -read(app.ESSinverter.device,app.Gridmeter.
                    output_power.register_type,app.Gridmeter.output_power.
                    address+1,1,app.Gridmeter.output_power.unitid,app.
                    Gridmeter.output_power.type)/app.Gridmeter.output_power.
                    scalefactor;
                ac_load_power_new = read(app.ESSinverter.device,"inputregs
                    ",818,1,100,'uint16')/1;
                battery_power_new = -read(app.ESSinverter.device, "inputregs
                    ", 843, 1, 100, 'int16')/1;
                pv_power_new = read(app.ESSinverter.device,app.PVmeter.
                    output_power.register_type,app.PVmeter.output_power.
                    address+1,1,app.PVmeter.output_power.unitid,app.PVmeter.
                    output_power.type)/app.PVmeter.output_power.scalefactor;
                %Output values to the GUI
                app.MeasuredVoltageTextArea.Value = num2str(grid_voltage_new)
                    ;
                app.MeasuredPVoutputpowerTextArea.Value = num2str(pv_power);
                app.TargetoutputvoltageTextArea.Value = num2str(app.row_temp
                    (3));
                app.BatterySOCTextArea.Value = num2str(app.soc(i));
%                 app.PoweratPCCTextArea.Value = num2str(app.row_temp(2) +
    ESS_set_output_power);
                app.PoweratPCCTextArea.Value = num2str(grid_power_new+
                    ac_load_power_new);
                app.StatusofSolarInverterLamp.Color = [0/255 255/255 0/255];
                app.StatusofESSInverterLamp_2.Color = [0/255 255/255 0/255];
                voltage_array(i) = grid_voltage_new;
                % if i == 1
                %     app.soc(i) = app.soc_battery;
                %     app.energy_Wh(i) = app.energy_ATEPS*app.soc_battery
                    /100;
                %     app.ess_power(i) = 0;
                % else
                %     app.ess_power(i) = battery_power;
                %     energy_used(i) = (app.ess_power(i) + app.ess_power(i-1)
                    )*elapsed_time*0.5/3600;
                %     app.soc(i) = app.soc(i-1) - (energy_used(i)/app.
                    energy_ATEPS)*100;
                %     app.energy_Wh(i) = app.energy_Wh(i-1) - energy_used(i);
                % end
                if i == 1
                    app.MeasuredVoltageFluctuationEditField.Value = num2str
                        (0);
                    app.MeasuredVoltageFluctuationEditField.FontColor =
                        [0/255 0/255 0/255];
                    app.MeasuredVoltageFluctuationEditField.FontWeight = '
```

```
                        normal';
                    else
                        difference_voltage = abs((voltage_array(i) -
                            voltage_array(i-1)));
                        app.MeasuredVoltageFluctuationEditField.Value = num2str(
                            difference_voltage);
                        if difference_voltage > 0.0084*230
                            app.MeasuredVoltageFluctuationEditField.FontColor =
                                [255/255 92/255 57/255];
                            app.MeasuredVoltageFluctuationEditField.FontWeight =
                                'bold';
                        elseif difference_voltage > 0.0036*230
                            app.MeasuredVoltageFluctuationEditField.FontColor =
                                [255/255 153/255 51/255];
                            app.MeasuredVoltageFluctuationEditField.FontWeight =
                                'normal';
                        else
                            app.MeasuredVoltageFluctuationEditField.FontColor =
                                [0/255 0/255 0/255];
                            app.MeasuredVoltageFluctuationEditField.FontWeight =
                                'normal';
                        end
                    end
                    %Build data to be exported
                    app.data_export{i,1} = datetime('now','Format','d-MMM-y HH:mm
                        :ss.SSS');
                    app.data_export{i,2} = grid_voltage_new;
                    app.data_export{i,3} = app.row_temp(3);
                    app.data_export{i,4} = -ESS_set_output_power_actual;
                    app.data_export{i,5} = -ESS_set_output_power;
                    app.data_export{i,6} = battery_power_new;
                    app.data_export{i,7} = pv_power_new;
%                     app.data_export{i,6} = app.row_temp(2) +
    ESS_set_output_power;
                    app.data_export{i,8} = grid_power_new +ac_load_power_new ;
                    app.data_export{i,9} = app.soc(i);
                    app.data_export{i,10} = app.energy_Wh(i);
                    app.data_export{i,11} = app.controlmode;
                    app.data_export{i,12} = ESS_set_output_power_first;
                    app.data_export{i,13} = ESS_set_output_power_second;
                    app.data_export{i,14} = ESS_set_output_power_third;
                    app.data_export{i,15} = ESS_set_output_power_pre;
                    app.data_export{i,16} = ESS_set_output_power_final_ideal;
                    i = i + 1;
                    app.nr_cycles = app.nr_cycles + 1;
                    % drawnow;
                    % plot(app.UIAxes,ESS_set_output_power)
                    addpoints(h,i,battery_power_new);
                    grid on;
                    drawnow;
                elseif string(app.turnonread) == "On" & app.error == 1
                    %Write to ESS inverter a power command for ZERO power
                    %and reset the MA.
                    app.permission_to_control_voltage = 1;
                    app.permission_to_control_power = 1;
%                     write(app.modbus_device_1,'holdingregs',app.startadress
    ,0,1,'single')
                    %Color the lamps to RED if there is an error with the
                    %inverters
                    if app.error_ESS_inverter == 0
                        app.StatusofESSInverterLamp_2.Color = [0/255 255/255
```

```
                        0/255];
                elseif app.error_ESS_inverter > 0
                    app.StatusofESSInverterLamp_2.Color = [255/255 92/255
                        57/255];
                    formatError = "ESS inverter experienced error : %d";
                    errormessage = compose(formatError,app.error_ESS_inverter
                        );
                    msgbox(errormessage);
                end

                if app.error_PV_inverter == 0
                    app.StatusofSolarInverterLamp.Color = [0/255 255/255
                        0/255];
                elseif app.error_PV_inverter == 1
                    app.StatusofSolarInverterLamp.Color = [255/255 92/255
                        57/255];
                end

                pause(app.delay_messages);
            elseif string(app.turnonread) == "Off" & app.error == 1
                %Write to ESS inverter a power command for ZERO power
                %and reset the MA.
                app.permission_to_control_voltage = 1;
                app.permission_to_control_power = 1;
%                 write(app.modbus_device_1,'holdingregs',app.startadress
    ,0,1,'single')
                %Color the lamps to RED if there is an error with the
                %inverters
                if app.error_ESS_inverter == 0
                    app.StatusofESSInverterLamp_2.Color = [0/255 255/255
                        0/255];
                elseif app.error_ESS_inverter > 0
                    app.StatusofESSInverterLamp_2.Color = [255/255 92/255
                        57/255];
                    formatError = "ESS inverter experienced error : %d";
                    errormessage = compose(formatError,app.error_ESS_inverter
                        );
                    msgbox(errormessage);
                end

                if app.error_PV_inverter == 0
                    app.StatusofSolarInverterLamp.Color = [0/255 255/255
                        0/255];
                elseif app.error_PV_inverter == 1
                    app.StatusofSolarInverterLamp.Color = [255/255 92/255
                        57/255];
                end

                pause(app.delay_messages);
            else
                %When paused, reset MA and pause writing to exported
                %data
                app.permission_to_control_voltage = 1;
                app.permission_to_control_power = 1;
%                 write(app.modbus_device_1,'holdingregs',app.startadress
    ,0,1,'single')
                app.StatusofESSInverterLamp_2.Color = [0/255 255/255 0/255];
                app.data_export{i,1} = datetime('now','Format','d-MMM-y HH:mm
                    :ss.SSS');
                app.data_export{i,2} = grid_voltage_new;
                app.data_export{i,3} = app.row_temp(3);
```

```
            app.data_export{i,4} = 0;
            app.data_export{i,5} = 0;
            app.data_export{i,6} = battery_power_new;
            app.data_export{i,7} = pv_power_new;
            app.data_export{i,8} = app.row_temp(2) + 0;
            app.data_export{i,9} = app.soc(i);
            app.data_export{i,10} = app.energy_Wh(i);
            app.data_export{i,11} = 'Pause';
            app.data_export{i,12} = ESS_set_output_power_first;
            app.data_export{i,13} = ESS_set_output_power_second;
            app.data_export{i,14} = ESS_set_output_power_third;
            app.data_export{i,15} = ESS_set_output_power_pre;
            app.data_export{i,16} = ESS_set_output_power_final_ideal;
            pause(app.delay_messages);
        end
    end
    %Again if there is an error, change the color of the lamp to
    %RED
    if app.error_ESS_inverter == 0
        app.StatusofESSInverterLamp_2.Color = [0/255 255/255 0/255];
    elseif app.error_ESS_inverter > 0
        app.StatusofESSInverterLamp_2.Color = [255/255 92/255 57/255];
        formatError = "ESS inverter experienced error : %d";
        errormessage = compose(formatError,app.error_ESS_inverter);
        msgbox(errormessage);
    end

    if app.error_PV_inverter == 0
        app.StatusofSolarInverterLamp.Color = [0/255 255/255 0/255];
    elseif app.error_PV_inverter == 1
        app.StatusofSolarInverterLamp.Color = [255/255 92/255 57/255];
    end
end

% Callback function
function IPofModbusDeviceEditFieldValueChanged(app, event)
    app.modbusserver = app.IPofModbusDeviceEditField.Value;
end

% Callback function
function PortnumberofModbusDeviceEditFieldValueChanged(app, event)
    app.modbusport = app.PortnumberofModbusDeviceEditField.Value;
end

% Value changed function: MeasuredVoltageTextArea
function MeasuredVoltageTextAreaValueChanged(app, event)
    % value = app.MeasuredVoltageTextArea.Value;

end

% Value changing function: MeasuredVoltageTextArea
function MeasuredVoltageTextAreaValueChanging(app, event)
    changingValue = event.Value;

end

% Value changed function: TargetoutputvoltageTextArea
function TargetoutputvoltageTextAreaValueChanged(app, event)
    % value = app.TargetoutputvoltageTextArea.Value;

end
```

```
% Value changed function: MeasuredPVoutputpowerTextArea
function MeasuredPVoutputpowerTextAreaValueChanged(app, event)
    % value = app.MeasuredPVoutputpowerTextArea.Value;

end

% Value changed function: TargetESSoutputpowerTextArea
function TargetESSoutputpowerTextAreaValueChanged(app, event)
    % value = app.TargetESSoutputpowerTextArea.Value;
    %
end

% Value changed function: DropDown
function DropDownValueChanged(app, event)
    app.controlmode = app.DropDown.Value;
    k = 1;
end

% Value changed function: PauseSwitch
function PauseSwitchValueChanged(app, event)
    app.turnonread = app.PauseSwitch.Value;

end

% Value changed function: PoweratPCCTextArea
function PoweratPCCTextAreaValueChanged(app, event)
    % value = app.PoweratPCCTextArea.Value;

end

% Value changed function: BatterySOCTextArea
function BatterySOCTextAreaValueChanged(app, event)
    % value = app.BatterySOCTextArea.Value;
    %
end

% Button pushed function: ExportdatatoCSVfileButton
function ExportdatatoCSVfileButtonPushed(app, event)
    test_data = struct;
    data_length = length(app.data_export);
    year_data = year(app.data_export{data_length,1});
    month_data = month(app.data_export{data_length,1});
    day_data = day(app.data_export{data_length,1});
    hour_data = hour(app.data_export{data_length,1});
    minute_data = minute(app.data_export{data_length,1});
    second_data =  floor(second(app.data_export{data_length,1}));
    formatspec = "C:/Users/ti/Desktop/Lynrick Thesis - Matlab Files/data_
       %d_%d_%d_%d%d%d.mat";
    filename = compose(formatspec,year_data,month_data,day_data,hour_data
       ,minute_data,second_data);
    varNames = ["Time","Measured Voltage (V)","Set Voltage (V)", "Actual
       Set Power (W)","Sent Set Power (W)", "BESS Measured Power", "PV
       Output Power (W)", "PCC Power (W)", "BESS SOC (%)", "BESS Energy (
       Wh)", "Control Mode","ESS Setpoint One Layer","ESS Setpoint Two
       Layer","ESS Setpoint Three Layer","ESS Setpoint Fourth Layer","ESS
        Setpoint Final Ideal"];
    data_table = cell2table(app.data_export,"VariableNames",varNames);
    config_table = cell2table(app.config,"VariableNames",["Date of Test",
        string(app.data_export{data_length,1}),"Bye","Have a great time
       ","Lynrick","Wix"]);
```

```
            test_data.data = data_table;
            test_data.configuration = config_table;
            save(filename,"-struct","test_data");
%             writetable(data_table,filename,'Sheet','Data');
%             writetable(config_table,filename,'Sheet', 'Configuration');
            msgbox("Succesfully exported the measured data!");
        end

        % Value changed function: MeasuredVoltageFluctuationEditField
        function MeasuredVoltageFluctuationEditFieldValueChanged(app, event)
%             value = app.MeasuredVoltageFluctuationEditField.Value;
%
        end

        % Button down function: UIAxes
        function UIAxesButtonDown(app, event)

        end

        % Value changed function: VoltageControlElementsforWindowEditField
        function VoltageControlElementsforWindowEditFieldValueChanged(app, event)
            % value = app.VoltageControlElementsforWindowEditField.Value;

        end

        % Value changed function: PowerControlElementsforWindowEditField
        function PowerControlElementsforWindowEditFieldValueChanged(app, event)
            % value = app.PowerControlElementsforWindowEditField.Value;

        end

        % Value changed function: MeasuredBatterySOCTextArea
        function MeasuredBatterySOCTextAreaValueChanged(app, event)
            % value = app.MeasuredBatterySOCTextArea.Value;

        end

        % Value changed function: TargetESSoutputpower1LayerTextArea
        function TargetESSoutputpower1LayerTextAreaValueChanged(app, event)
            % value = app.TargetESSoutputpower1LayerTextArea.Value;
            %
        end

        % Value changed function: TargetESSoutputpower2LayerTextArea
        function TargetESSoutputpower2LayerTextAreaValueChanged(app, event)
            % value = app.TargetESSoutputpower2LayerTextArea.Value;
            %
        end

        % Value changed function: TargetESSoutputpower3LayerTextArea
        function TargetESSoutputpower3LayerTextAreaValueChanged(app, event)
            % value = app.TargetESSoutputpower3LayerTextArea.Value;
            %
        end

        % Value changed function: TargetESSoutputpower4LayerTextArea
        function TargetESSoutputpower4LayerTextAreaValueChanged(app, event)
            % value = app.TargetESSoutputpower4LayerTextArea.Value;
            %
        end
```

```
        % Value changed function: MeasuredBatteryPowerTextArea
        function MeasuredBatteryPowerTextAreaValueChanged(app, event)
            % value = app.MeasuredBatteryPowerTextArea.Value;

        end

        % Callback function
        function CLEARVARIABLESButtonPushed(app, event)
            % clear app.data_export
            % clear app.ESSinverter
            % clear app.Gridmeter
            % clear app.PVmeter
            % clear app.config
            % app.time_day = {};
            % app.ess_power = [];
            % app.energy_Wh = [];
            % app.permission_to_control_power = 1;
            % app.permission_to_control_voltage = 1;
            % app.nr_cycles = 1;
            % msgbox("Succesfully clear data and variables!");
        end
    end

    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Get the file path for locating images
            pathToMLAPP = fileparts(mfilename('fullpath'));

            % Create UIFigure and hide until all components are created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.AutoResizeChildren = 'off';
            app.UIFigure.Color = [0.8 0.8 0.8];
            app.UIFigure.Position = [200 -100 1337 805];
            app.UIFigure.Name = 'MATLAB App';

            % Create UIAxes3
            app.UIAxes3 = uiaxes(app.UIFigure);
            title(app.UIAxes3, 'Idle Charge/Discharge - Power Control')
            xlabel(app.UIAxes3, 'SoC of Battery (%)')
            ylabel(app.UIAxes3, 'Power from ESS (W)')
            zlabel(app.UIAxes3, 'Z')
            app.UIAxes3.XGrid = 'on';
            app.UIAxes3.YGrid = 'on';
            app.UIAxes3.Position = [863 14 357 206];

            % Create UIAxes2
            app.UIAxes2 = uiaxes(app.UIFigure);
            title(app.UIAxes2, 'Idle Charge/Discharge - Voltage Control')
            xlabel(app.UIAxes2, 'SoC of Battery (%)')
            ylabel(app.UIAxes2, 'Power from ESS (W)')
            zlabel(app.UIAxes2, 'Z')
            app.UIAxes2.XGrid = 'on';
            app.UIAxes2.YGrid = 'on';
            app.UIAxes2.Position = [434 14 365 205];

            % Create UIAxes
            app.UIAxes = uiaxes(app.UIFigure);
```

```
title(app.UIAxes, 'SoC Power Reduction Curve - Voltage Control')
xlabel(app.UIAxes, 'SoC of Battery (%)')
ylabel(app.UIAxes, 'Reduction in Power (%)')
zlabel(app.UIAxes, 'Z')
app.UIAxes.XGrid = 'on';
app.UIAxes.YGrid = 'on';
app.UIAxes.ButtonDownFcn = createCallbackFcn(app, @UIAxesButtonDown,
    true);
app.UIAxes.Position = [24 14 366 206];

% Create UIAxes4
app.UIAxes4 = uiaxes(app.UIFigure);
title(app.UIAxes4, 'SoC Power Reduction Curve - Power Control')
xlabel(app.UIAxes4, 'SoC of Battery (%)')
ylabel(app.UIAxes4, 'Reduction in Power (%)')
zlabel(app.UIAxes4, 'Z')
app.UIAxes4.XGrid = 'on';
app.UIAxes4.YGrid = 'on';
app.UIAxes4.Position = [25 229 366 206];

% Create Image
app.Image = uiimage(app.UIFigure);
app.Image.Position = [442 223 733 487];
app.Image.ImageSource = fullfile(pathToMLAPP, 'GUI_DC_connection.
    drawio.png');

% Create InitializeConnectionsButton
app.InitializeConnectionsButton = uibutton(app.UIFigure, 'push');
app.InitializeConnectionsButton.ButtonPushedFcn = createCallbackFcn(
    app, @InitializeConnectionsButtonPushed, true);
app.InitializeConnectionsButton.Position = [21 748 276 37];
app.InitializeConnectionsButton.Text = 'Initialize Connections';

% Create ReadwriteregistersButton
app.ReadwriteregistersButton = uibutton(app.UIFigure, 'push');
app.ReadwriteregistersButton.ButtonPushedFcn = createCallbackFcn(app,
     @ReadwriteregistersButtonPushed, true);
app.ReadwriteregistersButton.Position = [21 611 276 32];
app.ReadwriteregistersButton.Text = 'Read/write registers';

% Create MeasuredVoltageTextAreaLabel
app.MeasuredVoltageTextAreaLabel = uilabel(app.UIFigure);
app.MeasuredVoltageTextAreaLabel.HorizontalAlignment = 'right';
app.MeasuredVoltageTextAreaLabel.Position = [786 568 102 22];
app.MeasuredVoltageTextAreaLabel.Text = 'Measured Voltage';

% Create MeasuredVoltageTextArea
app.MeasuredVoltageTextArea = uitextarea(app.UIFigure);
app.MeasuredVoltageTextArea.ValueChangedFcn = createCallbackFcn(app,
    @MeasuredVoltageTextAreaValueChanged, true);
app.MeasuredVoltageTextArea.ValueChangingFcn = createCallbackFcn(app,
     @MeasuredVoltageTextAreaValueChanging, true);
app.MeasuredVoltageTextArea.HorizontalAlignment = 'center';
app.MeasuredVoltageTextArea.Position = [893 561 56 37];

% Create StatusofSolarInverterLamp
app.StatusofSolarInverterLamp = uilamp(app.UIFigure);
app.StatusofSolarInverterLamp.Position = [652 668 34 34];
app.StatusofSolarInverterLamp.Color = [1 1 1];

% Create StatusofESSInverterLamp_2
```

```
app.StatusofESSInverterLamp_2 = uilamp(app.UIFigure);
app.StatusofESSInverterLamp_2.Position = [854 655 28 28];
app.StatusofESSInverterLamp_2.Color = [1 1 1];

% Create MeasuredPVoutputpowerLabel
app.MeasuredPVoutputpowerLabel = uilabel(app.UIFigure);
app.MeasuredPVoutputpowerLabel.HorizontalAlignment = 'right';
app.MeasuredPVoutputpowerLabel.Position = [505 536 81 30];
app.MeasuredPVoutputpowerLabel.Text = {'Measured PV '; 'output power'
    };

% Create MeasuredPVoutputpowerTextArea
app.MeasuredPVoutputpowerTextArea = uitextarea(app.UIFigure);
app.MeasuredPVoutputpowerTextArea.ValueChangedFcn = createCallbackFcn
    (app, @MeasuredPVoutputpowerTextAreaValueChanged, true);
app.MeasuredPVoutputpowerTextArea.HorizontalAlignment = 'center';
app.MeasuredPVoutputpowerTextArea.Position = [596 533 106 37];

% Create TargetoutputvoltageLabel
app.TargetoutputvoltageLabel = uilabel(app.UIFigure);
app.TargetoutputvoltageLabel.HorizontalAlignment = 'right';
app.TargetoutputvoltageLabel.Position = [973 564 75 30];
app.TargetoutputvoltageLabel.Text = {'Target output'; ' voltage'};

% Create TargetoutputvoltageTextArea
app.TargetoutputvoltageTextArea = uitextarea(app.UIFigure);
app.TargetoutputvoltageTextArea.ValueChangedFcn = createCallbackFcn(
    app, @TargetoutputvoltageTextAreaValueChanged, true);
app.TargetoutputvoltageTextArea.HorizontalAlignment = 'center';
app.TargetoutputvoltageTextArea.Position = [1055 561 59 37];

% Create TargetESSoutputpowerTextAreaLabel
app.TargetESSoutputpowerTextAreaLabel = uilabel(app.UIFigure);
app.TargetESSoutputpowerTextAreaLabel.HorizontalAlignment = 'right';
app.TargetESSoutputpowerTextAreaLabel.Position = [741 682 138 22];
app.TargetESSoutputpowerTextAreaLabel.Text = 'Target ESS output power
    ';

% Create TargetESSoutputpowerTextArea
app.TargetESSoutputpowerTextArea = uitextarea(app.UIFigure);
app.TargetESSoutputpowerTextArea.ValueChangedFcn = createCallbackFcn(
    app, @TargetESSoutputpowerTextAreaValueChanged, true);
app.TargetESSoutputpowerTextArea.HorizontalAlignment = 'center';
app.TargetESSoutputpowerTextArea.Position = [891 667 59 37];

% Create DropDownLabel
app.DropDownLabel = uilabel(app.UIFigure);
app.DropDownLabel.HorizontalAlignment = 'right';
app.DropDownLabel.Position = [35 688 65 22];
app.DropDownLabel.Text = 'Drop Down';

% Create DropDown
app.DropDown = uidropdown(app.UIFigure);
app.DropDown.Items = {'Read Values Only', 'Power Control', 'Voltage
    Control'};
app.DropDown.ValueChangedFcn = createCallbackFcn(app,
    @DropDownValueChanged, true);
app.DropDown.Placeholder = 'Read Values Only';
app.DropDown.Position = [115 673 167 51];
app.DropDown.Value = 'Read Values Only';
```

```
% Create PauseSwitchLabel
app.PauseSwitchLabel = uilabel(app.UIFigure);
app.PauseSwitchLabel.HorizontalAlignment = 'center';
app.PauseSwitchLabel.Position = [381 584 39 22];
app.PauseSwitchLabel.Text = 'Pause';

% Create PauseSwitch
app.PauseSwitch = uiswitch(app.UIFigure, 'toggle');
app.PauseSwitch.ValueChangedFcn = createCallbackFcn(app,
    @PauseSwitchValueChanged, true);
app.PauseSwitch.Position = [310 544 45 101];
app.PauseSwitch.Value = 'On';

% Create PoweratPCCTextAreaLabel
app.PoweratPCCTextAreaLabel = uilabel(app.UIFigure);
app.PoweratPCCTextAreaLabel.HorizontalAlignment = 'right';
app.PoweratPCCTextAreaLabel.Position = [807 528 81 22];
app.PoweratPCCTextAreaLabel.Text = 'Power at PCC';

% Create PoweratPCCTextArea
app.PoweratPCCTextArea = uitextarea(app.UIFigure);
app.PoweratPCCTextArea.ValueChangedFcn = createCallbackFcn(app,
    @PoweratPCCTextAreaValueChanged, true);
app.PoweratPCCTextArea.HorizontalAlignment = 'center';
app.PoweratPCCTextArea.Position = [892 521 57 37];

% Create BatterySOCTextAreaLabel
app.BatterySOCTextAreaLabel = uilabel(app.UIFigure);
app.BatterySOCTextAreaLabel.HorizontalAlignment = 'right';
app.BatterySOCTextAreaLabel.Position = [642 276 72 22];
app.BatterySOCTextAreaLabel.Text = 'Battery SOC';

% Create BatterySOCTextArea
app.BatterySOCTextArea = uitextarea(app.UIFigure);
app.BatterySOCTextArea.ValueChangedFcn = createCallbackFcn(app,
    @BatterySOCTextAreaValueChanged, true);
app.BatterySOCTextArea.HorizontalAlignment = 'center';
app.BatterySOCTextArea.Position = [727 269 81 37];

% Create ExportdatatoCSVfileButton
app.ExportdatatoCSVfileButton = uibutton(app.UIFigure, 'push');
app.ExportdatatoCSVfileButton.ButtonPushedFcn = createCallbackFcn(app
    , @ExportdatatoCSVfileButtonPushed, true);
app.ExportdatatoCSVfileButton.Position = [24 525 271 66];
app.ExportdatatoCSVfileButton.Text = 'Export data to CSV file';

% Create Image2
app.Image2 = uiimage(app.UIFigure);
app.Image2.Position = [419 718 223 66];
app.Image2.ImageSource = fullfile(pathToMLAPP, 'Stedin_logo.jpg');

% Create Image3
app.Image3 = uiimage(app.UIFigure);
app.Image3.Position = [933 719 309 66];
app.Image3.ImageSource = fullfile(pathToMLAPP, 'TU_delft_logo.jpg');

% Create MeasuredVoltageFluctuationEditFieldLabel
app.MeasuredVoltageFluctuationEditFieldLabel = uilabel(app.UIFigure);
app.MeasuredVoltageFluctuationEditFieldLabel.HorizontalAlignment = '
    right';
app.MeasuredVoltageFluctuationEditFieldLabel.FontSize = 16;
```

```
app.MeasuredVoltageFluctuationEditFieldLabel.FontWeight = 'bold';
app.MeasuredVoltageFluctuationEditFieldLabel.Position = [701 781 233
    22];
app.MeasuredVoltageFluctuationEditFieldLabel.Text = 'Measured Voltage
     Fluctuation';

% Create MeasuredVoltageFluctuationEditField
app.MeasuredVoltageFluctuationEditField = uieditfield(app.UIFigure, '
    text');
app.MeasuredVoltageFluctuationEditField.ValueChangedFcn =
    createCallbackFcn(app,
    @MeasuredVoltageFluctuationEditFieldValueChanged, true);
app.MeasuredVoltageFluctuationEditField.HorizontalAlignment = 'center
    ';
app.MeasuredVoltageFluctuationEditField.Position = [685 718 264 48];

% Create VoltageControlElementsforWindowEditFieldLabel
app.VoltageControlElementsforWindowEditFieldLabel = uilabel(app.
    UIFigure);
app.VoltageControlElementsforWindowEditFieldLabel.HorizontalAlignment
     = 'center';
app.VoltageControlElementsforWindowEditFieldLabel.FontColor = [0 0
    0.8];
app.VoltageControlElementsforWindowEditFieldLabel.Position = [25 462
    132 30];
app.VoltageControlElementsforWindowEditFieldLabel.Text = {'Voltage
    Control'; '# Elements for Window'};

% Create VoltageControlElementsforWindowEditField
app.VoltageControlElementsforWindowEditField = uieditfield(app.
    UIFigure, 'numeric');
app.VoltageControlElementsforWindowEditField.ValueChangedFcn =
    createCallbackFcn(app,
    @VoltageControlElementsforWindowEditFieldValueChanged, true);
app.VoltageControlElementsforWindowEditField.HorizontalAlignment = '
    center';
app.VoltageControlElementsforWindowEditField.FontColor = [0 0 0.8];
app.VoltageControlElementsforWindowEditField.Position = [165 453 44
    48];

% Create PowerControlElementsforWindowEditFieldLabel
app.PowerControlElementsforWindowEditFieldLabel = uilabel(app.
    UIFigure);
app.PowerControlElementsforWindowEditFieldLabel.HorizontalAlignment =
     'center';
app.PowerControlElementsforWindowEditFieldLabel.FontColor = [0.502 0
    1];
app.PowerControlElementsforWindowEditFieldLabel.Position = [228 462
    132 30];
app.PowerControlElementsforWindowEditFieldLabel.Text = {'Power
    Control'; '# Elements for Window'};

% Create PowerControlElementsforWindowEditField
app.PowerControlElementsforWindowEditField = uieditfield(app.UIFigure
    , 'numeric');
app.PowerControlElementsforWindowEditField.ValueChangedFcn =
    createCallbackFcn(app,
    @PowerControlElementsforWindowEditFieldValueChanged, true);
app.PowerControlElementsforWindowEditField.HorizontalAlignment = '
    center';
app.PowerControlElementsforWindowEditField.FontColor = [0.502 0 1];
```

```
app.PowerControlElementsforWindowEditField.Position = [368 453 44
    48];

% Create MeasuredBatterySOCLabel
app.MeasuredBatterySOCLabel = uilabel(app.UIFigure);
app.MeasuredBatterySOCLabel.HorizontalAlignment = 'right';
app.MeasuredBatterySOCLabel.Position = [639 232 76 30];
app.MeasuredBatterySOCLabel.Text = {'Measured'; ' Battery SOC'};

% Create MeasuredBatterySOCTextArea
app.MeasuredBatterySOCTextArea = uitextarea(app.UIFigure);
app.MeasuredBatterySOCTextArea.ValueChangedFcn = createCallbackFcn(
    app, @MeasuredBatterySOCTextAreaValueChanged, true);
app.MeasuredBatterySOCTextArea.HorizontalAlignment = 'center';
app.MeasuredBatterySOCTextArea.Position = [728 229 81 37];

% Create TargetESSoutputpower1LayerTextAreaLabel
app.TargetESSoutputpower1LayerTextAreaLabel = uilabel(app.UIFigure);
app.TargetESSoutputpower1LayerTextAreaLabel.HorizontalAlignment = '
    right';
app.TargetESSoutputpower1LayerTextAreaLabel.Position = [846 463 208
    22];
app.TargetESSoutputpower1LayerTextAreaLabel.Text = 'Target ESS output
     power - 1 Layer';

% Create TargetESSoutputpower1LayerTextArea
app.TargetESSoutputpower1LayerTextArea = uitextarea(app.UIFigure);
app.TargetESSoutputpower1LayerTextArea.ValueChangedFcn =
    createCallbackFcn(app,
    @TargetESSoutputpower1LayerTextAreaValueChanged, true);
app.TargetESSoutputpower1LayerTextArea.HorizontalAlignment = 'center'
    ;
app.TargetESSoutputpower1LayerTextArea.Position = [1063 448 59 37];

% Create TargetESSoutputpower2LayerTextAreaLabel
app.TargetESSoutputpower2LayerTextAreaLabel = uilabel(app.UIFigure);
app.TargetESSoutputpower2LayerTextAreaLabel.HorizontalAlignment = '
    right';
app.TargetESSoutputpower2LayerTextAreaLabel.Position = [847 421 208
    22];
app.TargetESSoutputpower2LayerTextAreaLabel.Text = 'Target ESS output
     power - 2 Layer';

% Create TargetESSoutputpower2LayerTextArea
app.TargetESSoutputpower2LayerTextArea = uitextarea(app.UIFigure);
app.TargetESSoutputpower2LayerTextArea.ValueChangedFcn =
    createCallbackFcn(app,
    @TargetESSoutputpower2LayerTextAreaValueChanged, true);
app.TargetESSoutputpower2LayerTextArea.HorizontalAlignment = 'center'
    ;
app.TargetESSoutputpower2LayerTextArea.Position = [1063 407 59 37];

% Create TargetESSoutputpower3LayerTextAreaLabel
app.TargetESSoutputpower3LayerTextAreaLabel = uilabel(app.UIFigure);
app.TargetESSoutputpower3LayerTextAreaLabel.HorizontalAlignment = '
    right';
app.TargetESSoutputpower3LayerTextAreaLabel.Position = [847 379 208
    22];
app.TargetESSoutputpower3LayerTextAreaLabel.Text = 'Target ESS output
     power - 3 Layer';
```

```
% Create TargetESSoutputpower3LayerTextArea
app.TargetESSoutputpower3LayerTextArea = uitextarea(app.UIFigure);
app.TargetESSoutputpower3LayerTextArea.ValueChangedFcn =
    createCallbackFcn(app,
    @TargetESSoutputpower3LayerTextAreaValueChanged, true);
app.TargetESSoutputpower3LayerTextArea.HorizontalAlignment = 'center'
    ;
app.TargetESSoutputpower3LayerTextArea.Position = [1063 364 59 37];

% Create TargetESSoutputpower4LayerTextAreaLabel
app.TargetESSoutputpower4LayerTextAreaLabel = uilabel(app.UIFigure);
app.TargetESSoutputpower4LayerTextAreaLabel.HorizontalAlignment = '
    right';
app.TargetESSoutputpower4LayerTextAreaLabel.Position = [846 337 208
    22];
app.TargetESSoutputpower4LayerTextAreaLabel.Text = 'Target ESS output
     power - 4 Layer';

% Create TargetESSoutputpower4LayerTextArea
app.TargetESSoutputpower4LayerTextArea = uitextarea(app.UIFigure);
app.TargetESSoutputpower4LayerTextArea.ValueChangedFcn =
    createCallbackFcn(app,
    @TargetESSoutputpower4LayerTextAreaValueChanged, true);
app.TargetESSoutputpower4LayerTextArea.HorizontalAlignment = 'center'
    ;
app.TargetESSoutputpower4LayerTextArea.Position = [1061 322 61 37];

% Create TargetESSoutputpower1LayerTextAreaLabel_2
app.TargetESSoutputpower1LayerTextAreaLabel_2 = uilabel(app.UIFigure)
    ;
app.TargetESSoutputpower1LayerTextAreaLabel_2.HorizontalAlignment = '
    right';
app.TargetESSoutputpower1LayerTextAreaLabel_2.Position = [849 442 208
     22];
app.TargetESSoutputpower1LayerTextAreaLabel_2.Text = 'Moving Average'
    ;

% Create TargetESSoutputpower1LayerTextAreaLabel_3
app.TargetESSoutputpower1LayerTextAreaLabel_3 = uilabel(app.UIFigure)
    ;
app.TargetESSoutputpower1LayerTextAreaLabel_3.HorizontalAlignment = '
    right';
app.TargetESSoutputpower1LayerTextAreaLabel_3.Position = [848 400 208
     22];
app.TargetESSoutputpower1LayerTextAreaLabel_3.Text = 'SoC Deadzone
    Reduction Curve';

% Create TargetESSoutputpower1LayerTextAreaLabel_4
app.TargetESSoutputpower1LayerTextAreaLabel_4 = uilabel(app.UIFigure)
    ;
app.TargetESSoutputpower1LayerTextAreaLabel_4.HorizontalAlignment = '
    right';
app.TargetESSoutputpower1LayerTextAreaLabel_4.Position = [849 358 208
     22];
app.TargetESSoutputpower1LayerTextAreaLabel_4.Text = 'Idle Charge/
    Discharge';

% Create TargetESSoutputpower1LayerTextAreaLabel_5
app.TargetESSoutputpower1LayerTextAreaLabel_5 = uilabel(app.UIFigure)
    ;
app.TargetESSoutputpower1LayerTextAreaLabel_5.HorizontalAlignment = '
```

```
    right';
app.TargetESSoutputpower1LayerTextAreaLabel_5.Position = [831 309 225
     30];
app.TargetESSoutputpower1LayerTextAreaLabel_5.Text = {'Filled up
    window check hard coded limit, '; 'and Window of operating check'
    };

% Create MeasuredBatteryPowerLabel
app.MeasuredBatteryPowerLabel = uilabel(app.UIFigure);
app.MeasuredBatteryPowerLabel.HorizontalAlignment = 'right';
app.MeasuredBatteryPowerLabel.Position = [629 308 84 30];
app.MeasuredBatteryPowerLabel.Text = {'Measured'; ' Battery Power'};

% Create MeasuredBatteryPowerTextArea
app.MeasuredBatteryPowerTextArea = uitextarea(app.UIFigure);
app.MeasuredBatteryPowerTextArea.ValueChangedFcn = createCallbackFcn(
    app, @MeasuredBatteryPowerTextAreaValueChanged, true);
app.MeasuredBatteryPowerTextArea.HorizontalAlignment = 'center';
app.MeasuredBatteryPowerTextArea.FontColor = [1 0.8196 0];
app.MeasuredBatteryPowerTextArea.Position = [727 309 81 37];

% Create DesignedandprogrammedbyLynrickWixLabel
app.DesignedandprogrammedbyLynrickWixLabel = uilabel(app.UIFigure);
app.DesignedandprogrammedbyLynrickWixLabel.FontWeight = 'bold';
app.DesignedandprogrammedbyLynrickWixLabel.Position = [477 226 96
    72];
app.DesignedandprogrammedbyLynrickWixLabel.Text = {'Designed and'; '
    programmed by'; 'Lynrick Wix'};

% Create TargetESSoutputpower1LayerTextAreaLabel_6
app.TargetESSoutputpower1LayerTextAreaLabel_6 = uilabel(app.UIFigure)
    ;
app.TargetESSoutputpower1LayerTextAreaLabel_6.HorizontalAlignment = '
    right';
app.TargetESSoutputpower1LayerTextAreaLabel_6.Position = [809 316 15
    22];
app.TargetESSoutputpower1LayerTextAreaLabel_6.Text = 'W';

% Create TargetESSoutputpower1LayerTextAreaLabel_7
app.TargetESSoutputpower1LayerTextAreaLabel_7 = uilabel(app.UIFigure)
    ;
app.TargetESSoutputpower1LayerTextAreaLabel_7.HorizontalAlignment = '
    right';
app.TargetESSoutputpower1LayerTextAreaLabel_7.Position = [1126 453 15
     22];
app.TargetESSoutputpower1LayerTextAreaLabel_7.Text = 'W';

% Create TargetESSoutputpower1LayerTextAreaLabel_8
app.TargetESSoutputpower1LayerTextAreaLabel_8 = uilabel(app.UIFigure)
    ;
app.TargetESSoutputpower1LayerTextAreaLabel_8.HorizontalAlignment = '
    right';
app.TargetESSoutputpower1LayerTextAreaLabel_8.Position = [1126 414 15
     22];
app.TargetESSoutputpower1LayerTextAreaLabel_8.Text = 'W';

% Create TargetESSoutputpower1LayerTextAreaLabel_9
app.TargetESSoutputpower1LayerTextAreaLabel_9 = uilabel(app.UIFigure)
    ;
app.TargetESSoutputpower1LayerTextAreaLabel_9.HorizontalAlignment = '
    right';
```

```
app.TargetESSoutputpower1LayerTextAreaLabel_9.Position = [1126 371 15
    22];
app.TargetESSoutputpower1LayerTextAreaLabel_9.Text = 'W';

% Create TargetESSoutputpower1LayerTextAreaLabel_10
app.TargetESSoutputpower1LayerTextAreaLabel_10 = uilabel(app.UIFigure
    );
app.TargetESSoutputpower1LayerTextAreaLabel_10.HorizontalAlignment =
    'right';
app.TargetESSoutputpower1LayerTextAreaLabel_10.Position = [1126 329
    15 22];
app.TargetESSoutputpower1LayerTextAreaLabel_10.Text = 'W';

% Create TargetESSoutputpower1LayerTextAreaLabel_11
app.TargetESSoutputpower1LayerTextAreaLabel_11 = uilabel(app.UIFigure
    );
app.TargetESSoutputpower1LayerTextAreaLabel_11.HorizontalAlignment =
    'right';
app.TargetESSoutputpower1LayerTextAreaLabel_11.Position = [702 540 15
    22];
app.TargetESSoutputpower1LayerTextAreaLabel_11.Text = 'W';

% Create TargetESSoutputpower1LayerTextAreaLabel_12
app.TargetESSoutputpower1LayerTextAreaLabel_12 = uilabel(app.UIFigure
    );
app.TargetESSoutputpower1LayerTextAreaLabel_12.HorizontalAlignment =
    'right';
app.TargetESSoutputpower1LayerTextAreaLabel_12.Position = [939 569 25
    22];
app.TargetESSoutputpower1LayerTextAreaLabel_12.Text = 'V';

% Create TargetESSoutputpower1LayerTextAreaLabel_13
app.TargetESSoutputpower1LayerTextAreaLabel_13 = uilabel(app.UIFigure
    );
app.TargetESSoutputpower1LayerTextAreaLabel_13.HorizontalAlignment =
    'right';
app.TargetESSoutputpower1LayerTextAreaLabel_13.Position = [1102 569
    25 22];
app.TargetESSoutputpower1LayerTextAreaLabel_13.Text = 'V';

% Create TargetESSoutputpower1LayerTextAreaLabel_14
app.TargetESSoutputpower1LayerTextAreaLabel_14 = uilabel(app.UIFigure
    );
app.TargetESSoutputpower1LayerTextAreaLabel_14.HorizontalAlignment =
    'right';
app.TargetESSoutputpower1LayerTextAreaLabel_14.Position = [949 528 15
    22];
app.TargetESSoutputpower1LayerTextAreaLabel_14.Text = 'W';

% Create TargetESSoutputpower1LayerTextAreaLabel_15
app.TargetESSoutputpower1LayerTextAreaLabel_15 = uilabel(app.UIFigure
    );
app.TargetESSoutputpower1LayerTextAreaLabel_15.HorizontalAlignment =
    'right';
app.TargetESSoutputpower1LayerTextAreaLabel_15.Position = [800 276 25
    22];
app.TargetESSoutputpower1LayerTextAreaLabel_15.Text = '%';

% Create TargetESSoutputpower1LayerTextAreaLabel_16
app.TargetESSoutputpower1LayerTextAreaLabel_16 = uilabel(app.UIFigure
    );
```

```
            app.TargetESSoutputpower1LayerTextAreaLabel_16.HorizontalAlignment =
                'right';
            app.TargetESSoutputpower1LayerTextAreaLabel_16.Position = [800 236 25
                22];
            app.TargetESSoutputpower1LayerTextAreaLabel_16.Text = '%';

            % Create TargetESSoutputpower1LayerTextAreaLabel_17
            app.TargetESSoutputpower1LayerTextAreaLabel_17 = uilabel(app.UIFigure
                );
            app.TargetESSoutputpower1LayerTextAreaLabel_17.HorizontalAlignment =
                'right';
            app.TargetESSoutputpower1LayerTextAreaLabel_17.Position = [949 674 15
                22];
            app.TargetESSoutputpower1LayerTextAreaLabel_17.Text = 'W';

            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = test_setup_thesis_EbikeChargingStation

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```

2.2. GUI Matlab code for AC connected ESS experimental setup

```
classdef test_setup_thesis_ACSetup_ESPLab < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                          matlab.ui.Figure
        TargetESSoutputpower1LayerTextAreaLabel_18  matlab.ui.control.Label
        AverageCellVoltageTextArea        matlab.ui.control.TextArea
        AverageCellVoltageLabel           matlab.ui.control.Label
        TargetESSoutputpower1LayerTextAreaLabel_17  matlab.ui.control.Label
        TargetESSoutputpower1LayerTextAreaLabel_16  matlab.ui.control.Label
        TargetESSoutputpower1LayerTextAreaLabel_15  matlab.ui.control.Label
        TargetESSoutputpower1LayerTextAreaLabel_14  matlab.ui.control.Label
        TargetESSoutputpower1LayerTextAreaLabel_13  matlab.ui.control.Label
```

```
TargetESSoutputpower1LayerTextAreaLabel_12  matlab.ui.control.Label
TargetESSoutputpower1LayerTextAreaLabel_11  matlab.ui.control.Label
TargetESSoutputpower1LayerTextAreaLabel_10  matlab.ui.control.Label
TargetESSoutputpower1LayerTextAreaLabel_9  matlab.ui.control.Label
TargetESSoutputpower1LayerTextAreaLabel_8  matlab.ui.control.Label
TargetESSoutputpower1LayerTextAreaLabel_7  matlab.ui.control.Label
TargetESSoutputpower1LayerTextAreaLabel_6  matlab.ui.control.Label
StatusofBattery                 matlab.ui.control.Lamp
StatusofESSInverter             matlab.ui.control.Lamp
DesignedandprogrammedbyLynrickWixLabel  matlab.ui.control.Label
MeasuredBatteryPowerTextArea    matlab.ui.control.TextArea
MeasuredBatteryPowerLabel       matlab.ui.control.Label
TargetESSoutputpower1LayerTextAreaLabel_5  matlab.ui.control.Label
TargetESSoutputpower1LayerTextAreaLabel_4  matlab.ui.control.Label
TargetESSoutputpower1LayerTextAreaLabel_3  matlab.ui.control.Label
TargetESSoutputpower1LayerTextAreaLabel_2  matlab.ui.control.Label
TargetESSoutputpower4LayerTextArea  matlab.ui.control.TextArea
TargetESSoutputpower4LayerTextAreaLabel  matlab.ui.control.Label
TargetESSoutputpower3LayerTextArea  matlab.ui.control.TextArea
TargetESSoutputpower3LayerTextAreaLabel  matlab.ui.control.Label
TargetESSoutputpower2LayerTextArea  matlab.ui.control.TextArea
TargetESSoutputpower2LayerTextAreaLabel  matlab.ui.control.Label
TargetESSoutputpower1LayerTextArea  matlab.ui.control.TextArea
TargetESSoutputpower1LayerTextAreaLabel  matlab.ui.control.Label
MeasuredBatterySOCTextArea      matlab.ui.control.TextArea
MeasuredBatterySOCLabel         matlab.ui.control.Label
PowerControlElementsforWindowEditField  matlab.ui.control.
   NumericEditField
PowerControlElementsforWindowEditFieldLabel  matlab.ui.control.Label
VoltageControlElementsforWindowEditField  matlab.ui.control.
   NumericEditField
VoltageControlElementsforWindowEditFieldLabel  matlab.ui.control.Label
MeasuredVoltageFluctuationEditField  matlab.ui.control.EditField
MeasuredVoltageFluctuationEditFieldLabel  matlab.ui.control.Label
Image3                          matlab.ui.control.Image
Image2                          matlab.ui.control.Image
ExportdatatoCSVfileButton       matlab.ui.control.Button
BatterySOCTextArea              matlab.ui.control.TextArea
BatterySOCTextAreaLabel         matlab.ui.control.Label
PoweratPCCTextArea              matlab.ui.control.TextArea
PoweratPCCTextAreaLabel         matlab.ui.control.Label
PauseSwitch                     matlab.ui.control.ToggleSwitch
PauseSwitchLabel                matlab.ui.control.Label
DropDown                        matlab.ui.control.DropDown
DropDownLabel                   matlab.ui.control.Label
TargetESSoutputpowerTextArea    matlab.ui.control.TextArea
TargetESSoutputpowerLabel       matlab.ui.control.Label
TargetoutputvoltageTextArea     matlab.ui.control.TextArea
TargetoutputvoltageLabel        matlab.ui.control.Label
MeasuredPVoutputpowerTextArea   matlab.ui.control.TextArea
MeasuredPVoutputpowerLabel      matlab.ui.control.Label
StatusofGridmeter               matlab.ui.control.Lamp
StatusofPVmeter                 matlab.ui.control.Lamp
MeasuredVoltageTextArea         matlab.ui.control.TextArea
MeasuredVoltageTextAreaLabel    matlab.ui.control.Label
ReadwriteregistersButton        matlab.ui.control.Button
InitializeConnectionsButton     matlab.ui.control.Button
Image                           matlab.ui.control.Image
UIAxes4                         matlab.ui.control.UIAxes
UIAxes                          matlab.ui.control.UIAxes
UIAxes2                         matlab.ui.control.UIAxes
```

```
    UIAxes3                         matlab.ui.control.UIAxes
end


properties (Access = public)
        modbusserver;
        modbusport;
        modbus_device_1;
        registertype = 'holdingregs';
        startadress = 38;
        count = 1;
        voltage;
        pv_data;
        worst_day_pv;
        row_temp;
        sensitivity;
        systemswitch;
        switch_value;
        turnonread = "On";
        error = 0;
        error_ESS_inverter;
        error_PV_meter;
        error_Battery;
        controlmode = "Read Values Only";
        permission_to_control_voltage = 1;
        permission_to_control_power = 1;
        windowsize_voltage;
        windowsize_power;
        soc_battery;
        energy_ATEPS;
        soc;
        energy_Wh = [];
        ess_power = [];
        time_day = {};
        data_export = {};
        delay_messages = 1.67;
        correction_factor_discharging_power;
        correction_factor_charging_power;
        correction_factor_discharging_voltage;
        correction_factor_charging_voltage;
        soc_lower_limit_voltage;
        soc_upper_limit_voltage;
        soc_lower_limit_power;
        soc_upper_limit_power;
        idle_charge_power_ctrl;
        idle_discharge_power_ctrl;
        idle_charge_voltage_ctrl;
        idle_discharge_voltage_ctrl;
        soc_deadzone_lower_limit_voltage;
        soc_deadzone_upper_limit_voltage;
        soc_deadzone_lower_limit_power;
        soc_deadzone_upper_limit_power;
        power_threshold_idle_voltage;
        power_threshold_idle_power;
        ESSinverter;
        PVmeter;
        Gridmeter;
        Battery;
        config;
        idle_charge_voltage_factor;
        idle_discharge_voltage_factor;
```

```
        idle_charge_power_factor;
        idle_discharge_power_factor;
        nr_cycles;
        error_Grid_meter;
end


% Callbacks that handle component events
methods (Access = private)

    % Button pushed function: InitializeConnectionsButton
    function InitializeConnectionsButtonPushed(app, event)
        app.ESSinverter = struct;
        app.PVmeter = struct;
        app.Gridmeter = struct;
        cla(app.UIAxes)
        cla(app.UIAxes2)
        cla(app.UIAxes3)
        %Initialize all connections to the modbus devices and
        %initializing all parameters
        app.config = readcell('C:\Users\lynri\OneDrive\Documents\MATLAB\
            config_acsetup.csv');
        %Initializing the connections with the devices
        app.ESSinverter.connection_type = app.config{1,2};
        app.ESSinverter.ip_address = app.config{2,2};
        app.ESSinverter.port = app.config{3,2};
        app.PVmeter.connection_type = app.config{4,2};
        app.PVmeter.com_port = app.config{5,2};
        app.Gridmeter.connection_type = app.config{6,2};
        app.Gridmeter.com_port = app.config{7,2};
        %Setting in the parameters from the config file
        app.sensitivity = app.config{8,2};
        app.energy_ATEPS = app.config{9,2};
        app.soc_battery = app.config{10,2};
        app.correction_factor_discharging_voltage = app.config{12,2};
        app.correction_factor_charging_voltage = app.config{13,2};
        app.soc_lower_limit_voltage = app.config{14,2};
        app.soc_upper_limit_voltage = app.config{15,2};
        app.windowsize_voltage = app.config{16,2};
        app.idle_charge_voltage_ctrl = app.config{17,2};
        app.idle_discharge_voltage_ctrl = app.config{18,2};
        app.soc_deadzone_lower_limit_voltage = app.config{19,2};
        app.soc_deadzone_upper_limit_voltage = app.config{20,2};
        app.power_threshold_idle_voltage = app.config{21,2};
        app.correction_factor_discharging_power = app.config{23,2};
        app.correction_factor_charging_power = app.config{24,2};
        app.soc_lower_limit_power = app.config{25,2};
        app.soc_upper_limit_power = app.config{26,2};
        app.windowsize_power = app.config{27,2};
        app.idle_charge_power_ctrl = app.config{28,2};
        app.idle_discharge_power_ctrl = app.config{29,2};
        app.soc_deadzone_lower_limit_power = app.config{30,2};
        app.soc_deadzone_upper_limit_power = app.config{31,2};
        app.power_threshold_idle_power = app.config{32,2};

        app.VoltageControlElementsforWindowEditField.Value = app.config
            {16,2};
        app.PowerControlElementsforWindowEditField.Value = app.config{27,2};

        %Reading in all of the addresses from the modbus devices
        app.ESSinverter.set_power.address = app.config{34,2};
```

```
app.ESSinverter.set_power.type = app.config{34,3};
app.ESSinverter.set_power.scalefactor = app.config{34,4};
app.ESSinverter.set_power.unitid = app.config{34,5};
app.ESSinverter.set_power.register_type = app.config{34,6};

app.ESSinverter.error_check.address = app.config{35,2};
app.ESSinverter.error_check.type = app.config{35,3};
app.ESSinverter.error_check.scalefactor = app.config{35,4};
app.ESSinverter.error_check.unitid = app.config{35,5};
app.ESSinverter.error_check.register_type = app.config{35,6};

app.ESSinverter.output_voltage.address = app.config{36,2};
app.ESSinverter.output_voltage.type = app.config{36,3};
app.ESSinverter.output_voltage.scalefactor = app.config{36,4};
app.ESSinverter.output_voltage.unitid = app.config{36,5};
app.ESSinverter.output_voltage.register_type = app.config{36,6};

app.ESSinverter.output_current.address = app.config{37,2};
app.ESSinverter.output_current.type = app.config{37,3};
app.ESSinverter.output_current.scalefactor = app.config{37,4};
app.ESSinverter.output_current.unitid = app.config{37,5};
app.ESSinverter.output_current.register_type = app.config{37,6};

app.ESSinverter.output_power.address = app.config{38,2};
app.ESSinverter.output_power.type = app.config{38,3};
app.ESSinverter.output_power.scalefactor = app.config{38,4};
app.ESSinverter.output_power.unitid = app.config{38,5};
app.ESSinverter.output_power.register_type = app.config{38,6};

app.PVmeter.output_power.address = app.config{40,2};
app.PVmeter.output_power.type = app.config{40,3};
app.PVmeter.output_power.scalefactor = app.config{40,4};
app.PVmeter.output_power.unitid = app.config{40,5};
app.PVmeter.output_power.register_type = app.config{40,6};

app.PVmeter.error_check.address = app.config{41,2};
app.PVmeter.error_check.type = app.config{41,3};
app.PVmeter.error_check.scalefactor = app.config{41,4};
app.PVmeter.error_check.unitid = app.config{41,5};
app.PVmeter.error_check.register_type = app.config{41,6};

app.Gridmeter.output_power.address = app.config{44,2};
app.Gridmeter.output_power.type = app.config{44,3};
app.Gridmeter.output_power.scalefactor = app.config{44,4};
app.Gridmeter.output_power.unitid = app.config{44,5};
app.Gridmeter.output_power.register_type = app.config{44,6};

app.Gridmeter.output_voltage.address = app.config{43,2};
app.Gridmeter.output_voltage.type = app.config{43,3};
app.Gridmeter.output_voltage.scalefactor = app.config{43,4};
app.Gridmeter.output_voltage.unitid = app.config{43,5};
app.Gridmeter.output_voltage.register_type = app.config{43,6};

app.Gridmeter.error_check.address = app.config{45,2};
app.Gridmeter.error_check.type = app.config{45,3};
app.Gridmeter.error_check.scalefactor = app.config{45,4};
app.Gridmeter.error_check.unitid = app.config{45,5};
app.Gridmeter.error_check.register_type = app.config{45,6};

app.Battery.voltage.address = app.config{47,2};
app.Battery.voltage.type = app.config{47,3};
```

```
            app.Battery.voltage.scalefactor = app.config{47,4};
            app.Battery.voltage.unitid = app.config{47,5};
            app.Battery.voltage.register_type = app.config{47,6};

            app.Battery.current.address = app.config{48,2};
            app.Battery.current.type = app.config{48,3};
            app.Battery.current.scalefactor = app.config{48,4};
            app.Battery.current.unitid = app.config{48,5};
            app.Battery.current.register_type = app.config{48,6};

            app.Battery.soc.address = app.config{49,2};
            app.Battery.soc.type = app.config{49,3};
            app.Battery.soc.scalefactor = app.config{49,4};
            app.Battery.soc.unitid = app.config{49,5};
            app.Battery.soc.register_type = app.config{49,6};

            app.ESSinverter.device = modbus(app.ESSinverter.connection_type,app.
               ESSinverter.ip_address,app.ESSinverter.port);
            app.PVmeter.device = modbus(app.PVmeter.connection_type,app.PVmeter.
               com_port);
            app.Gridmeter.device = modbus(app.Gridmeter.connection_type,app.
               Gridmeter.com_port);
            msgbox("Initialized all connection to devices!");
            app.error_ESS_inverter = read(app.ESSinverter.device,app.ESSinverter.
               error_check.register_type,app.ESSinverter.error_check.address+1,1,
               app.ESSinverter.error_check.unitid);
            app.error_PV_meter = read(app.PVmeter.device,app.PVmeter.error_check.
               register_type,app.PVmeter.error_check.address+1,1,app.PVmeter.
               error_check.unitid);
            app.error_Grid_meter = read(app.Gridmeter.device,app.Gridmeter.
               error_check.register_type,app.Gridmeter.error_check.address+1,1,
               app.Gridmeter.error_check.unitid);
            battery_voltage = read(app.ESSinverter.device, app.Battery.voltage.
               register_type, app.Battery.voltage.address+1, 1, app.Battery.
               voltage.unitid, app.Battery.voltage.type)/app.Battery.voltage.
               scalefactor;
            battery_soc = read(app.ESSinverter.device, app.Battery.soc.
               register_type, app.Battery.soc.address+1, 1, app.Battery.soc.
               unitid, app.Battery.soc.type)/app.Battery.soc.scalefactor;

%             app.error_ESS_inverter = 0;
%             app.error_PV_inverter = 0;
            app.soc = [];
            app.ess_power(1) = 0;
            soc_axes_voltage = [0 20 app.soc_lower_limit_voltage app.
               soc_upper_limit_voltage 80 100];
%             correction_values_charging = [app.correction_factor_charging*(0-app
   .soc_lower_limit) app.correction_factor_charging*(20-app.soc_lower_limit) 0 0
   app.correction_factor_charging*(80-app.soc_upper_limit) app.
   correction_factor_charging*(100-app.soc_upper_limit)];
            correction_values_charging_voltage = [0 0 0 0 app.
               correction_factor_charging_voltage*(80-app.
               soc_deadzone_upper_limit_voltage) app.
               correction_factor_charging_voltage*(100-app.
               soc_deadzone_upper_limit_voltage)];
%             correction_values_discharging = [app.correction_factor_discharging
   *(0-app.soc_lower_limit) app.correction_factor_discharging*(20-app.
   soc_lower_limit) 0 0 app.correction_factor_discharging*(80-app.soc_upper_limit
   ) app.correction_factor_discharging*(100-app.soc_upper_limit)];
            correction_values_discharging_voltage = [app.
               correction_factor_discharging_voltage*(0-app.
```

```
            soc_deadzone_lower_limit_voltage) app.
            correction_factor_discharging_voltage*(20-app.
            soc_deadzone_lower_limit_voltage) 0 0 0 0 ];
        hold(app.UIAxes,'on')
        plot(app.UIAxes,soc_axes_voltage,abs(
            correction_values_charging_voltage)*100, 'Color', [255/255,
            209/255, 0],'LineWidth',2);
        plot(app.UIAxes,soc_axes_voltage,abs(
            correction_values_discharging_voltage)*100, 'Color', [51/255,
            153/255, 255/255],'LineWidth',2);
        hold(app.UIAxes,'off')
        app.UIAxes.XLim = [app.soc_lower_limit_voltage-2.5 app.
            soc_upper_limit_voltage+2.5];
        app.UIAxes.YLim = [-1 5];
        legend(app.UIAxes, 'Charge','Discharge')
        strTitleSoCDeadzone = "Charge = %0.3f and Discharge = %0.3f";
        titleSoCDeadzone = compose(strTitleSoCDeadzone, app.
            correction_factor_charging_voltage, app.
            correction_factor_discharging_voltage);
        app.UIAxes.Subtitle.String = titleSoCDeadzone;

        soc_axes_power = [0 20 app.soc_lower_limit_power app.
            soc_upper_limit_power 80 100];
%         correction_values_charging = [app.correction_factor_charging*(0-app
    .soc_lower_limit) app.correction_factor_charging*(20-app.soc_lower_limit) 0 0
    app.correction_factor_charging*(80-app.soc_upper_limit) app.
    correction_factor_charging*(100-app.soc_upper_limit)];
        correction_values_charging_power = [0 0 0 0 app.
            correction_factor_charging_power*(80-app.
            soc_deadzone_upper_limit_power) app.
            correction_factor_charging_power*(100-app.
            soc_deadzone_upper_limit_power)];
%         correction_values_discharging = [app.correction_factor_discharging
    *(0-app.soc_lower_limit) app.correction_factor_discharging*(20-app.
    soc_lower_limit) 0 0 app.correction_factor_discharging*(80-app.soc_upper_limit
    ) app.correction_factor_discharging*(100-app.soc_upper_limit)];
        correction_values_discharging_power = [app.
            correction_factor_discharging_power*(0-app.
            soc_deadzone_lower_limit_power) app.
            correction_factor_discharging_power*(20-app.
            soc_deadzone_lower_limit_power) 0 0 0 0 ];
        hold(app.UIAxes4,'on')
        plot(app.UIAxes4,soc_axes_power,abs(correction_values_charging_power)
            *100, 'Color', [255/255, 209/255, 0],'LineWidth',2);
        plot(app.UIAxes4,soc_axes_power,abs(
            correction_values_discharging_power)*100, 'Color', [51/255,
            153/255, 255/255],'LineWidth',2);
        hold(app.UIAxes4,'off')
        app.UIAxes4.XLim = [app.soc_lower_limit_power-2.5 app.
            soc_upper_limit_power+2.5];
        app.UIAxes4.YLim = [-1 5];
        legend(app.UIAxes, 'Charge','Discharge')
        strTitleSoCDeadzone = "Charge = %0.3f and Discharge = %0.3f";
        titleSoCDeadzone = compose(strTitleSoCDeadzone, app.
            correction_factor_charging_power, app.
            correction_factor_discharging_power);
        app.UIAxes4.Subtitle.String = titleSoCDeadzone;

        hold(app.UIAxes2,'on')
        area(app.UIAxes2, [0 0 app.soc_lower_limit_voltage app.
            soc_lower_limit_voltage], [0 app.power_threshold_idle_voltage app.
```

```
            power_threshold_idle_voltage 0], 'FaceColor', [0/255 0/255
            204/255])
        area(app.UIAxes2, [0 0 app.soc_lower_limit_voltage app.
            soc_lower_limit_voltage], [0 -app.power_threshold_idle_voltage -
            app.power_threshold_idle_voltage 0], 'FaceColor', [0/255 0/255
            204/255])
        area(app.UIAxes2, [app.soc_upper_limit_voltage app.
            soc_upper_limit_voltage 100 100], [0 app.
            power_threshold_idle_voltage app.power_threshold_idle_voltage 0],
            'FaceColor', [0/255 0/255 204/255])
        area(app.UIAxes2, [app.soc_upper_limit_voltage app.
            soc_upper_limit_voltage 100 100], [0 -app.
            power_threshold_idle_voltage -app.power_threshold_idle_voltage 0],
             'FaceColor', [0/255 0/255 204/255])
%             area(app.UIAxes2, [app.soc_upper_limit 100], [-20 20], 'FaceColor',
    [255/255 92/255 57/255])
        hold(app.UIAxes2,'off')
        app.UIAxes2.XLim = [app.soc_lower_limit_voltage-3 app.
            soc_upper_limit_voltage+3];
        app.UIAxes2.YLim = [-100 100];
        legend(app.UIAxes2,'Window of Operation')
        strTitleIdleVoltage = "Charge = %0.1f and Discharge = %0.1f";
        titleIdleVoltage = compose(strTitleIdleVoltage, app.
            idle_charge_voltage_ctrl, app.idle_discharge_voltage_ctrl);
        app.UIAxes2.Title.String = 'Idle Charge/Discharge Window of Operation
             - Voltage Control';
        app.UIAxes2.Subtitle.String = titleIdleVoltage;
        hold(app.UIAxes3,'on')
        area(app.UIAxes3, [0 0 app.soc_lower_limit_power app.
            soc_lower_limit_power], [0 app.power_threshold_idle_power app.
            power_threshold_idle_power 0], 'FaceColor', [127/255 0/255
            255/255])
        area(app.UIAxes3, [0 0 app.soc_lower_limit_power app.
            soc_lower_limit_power], [0 -app.power_threshold_idle_power -app.
            power_threshold_idle_power 0], 'FaceColor', [127/255 0/255
            255/255])
        area(app.UIAxes3, [app.soc_upper_limit_power app.
            soc_upper_limit_power 100 100], [0 app.power_threshold_idle_power
            app.power_threshold_idle_power 0], 'FaceColor', [127/255 0/255
            255/255])
        area(app.UIAxes3, [app.soc_upper_limit_power app.
            soc_upper_limit_power 100 100], [0 -app.power_threshold_idle_power
             -app.power_threshold_idle_power 0], 'FaceColor', [127/255 0/255
            255/255])
%             area(app.UIAxes2, [app.soc_upper_limit 100], [-20 20], 'FaceColor',
    [255/255 92/255 57/255])
        hold(app.UIAxes3,'off')
        app.UIAxes3.XLim = [app.soc_lower_limit_power-3 app.
            soc_upper_limit_power+3];
        app.UIAxes3.YLim = [-100 100];
        legend(app.UIAxes3,'Window of Operation')
        strTitleIdlePower = "Charge = %0.1f and Discharge = %0.1f";
        titleIdlePower = compose(strTitleIdlePower, app.
            idle_charge_power_ctrl, app.idle_discharge_power_ctrl);
        app.UIAxes3.Title.String = 'Idle Charge/Discharge Window of Operation
             - Power Control';
        app.UIAxes3.Subtitle.String = titleIdlePower;
        %First check if there are any errors to the inverters
        if battery_voltage > 57.4 || battery_voltage < 42
            app.error_Battery = 1;
        else
```

```
        if battery_soc > 90 || battery_soc < 10
            app.error_Battery = 1;
        else
            app.error_Battery = 0;
        end
    end

    if app.error_Battery == 0
        app.StatusofBattery.Color = [0/255 255/255 0/255];
    elseif app.error_Battery > 0
        app.StatusofBattery.Color = [255/255 92/255 57/255];
    end

    if app.error_ESS_inverter == 0
        app.StatusofESSInverter.Color = [0/255 255/255 0/255];
    elseif app.error_ESS_inverter > 0
        app.StatusofESSInverter.Color = [255/255 92/255 57/255];
        formatError = "ESS inverter experienced error : %d";
        errormessage = compose(formatError,app.error_ESS_inverter);
        msgbox(errormessage);
    end

    if app.error_PV_meter == 0
        app.StatusofPVmeter.Color = [0/255 255/255 0/255];
    elseif app.error_PV_meter > 0
        app.StatusofPVmeter.Color = [255/255 92/255 57/255];
    end

    if app.error_Grid_meter == 0
        app.StatusofGridmeter.Color = [0/255 255/255 0/255];
    elseif app.error_Grid_meter > 0
        app.StatusofGridmeter.Color = [255/255 92/255 57/255];
    end
    exportapp(app.UIFigure,'appcontent_ACsetup.pdf')
end

% Button pushed function: ReadwriteregistersButton
function ReadwriteregistersButtonPushed(app, event)
    %Initialize all of the used variables
    app.nr_cycles = 1;
    i = app.nr_cycles;
    data = zeros(app.windowsize_voltage,2);
    data_array = data(:,1)+230;
    data_array_voltage = cat(2,data_array,data);
    data = zeros(app.windowsize_power,2);
    data_array = data(:,1)+230;
    data_array_power = cat(2,data_array,data);
    sinus_MA_voltage_output_day = [];
    sinus_MA_power_output_day = [];
    voltage_array = [];
    %Live plot the ESS power
    h = animatedline('Color',[255/255, 209/255, 0],'LineWidth',2);
    title('ESS Power')
    app.error = 0;
    % window_size
    while(app.error == 0)
        %State of charge calculation of the BESS
        tic;
        if i == 1
            app.soc(i) = app.soc_battery;
            app.energy_Wh(i) = app.energy_ATEPS*app.soc_battery/100;
```

```
                    app.ess_power(i) = 0;
                else
                    app.ess_power(i) = battery_power_dc;
                    energy_used(i) = (app.ess_power(i) + app.ess_power(i-1))*
                        elapsed_time*0.5/3600;
                    app.soc(i) = app.soc(i-1) - (energy_used(i)/app.energy_ATEPS)
                        *100;
                    app.energy_Wh(i) = app.energy_Wh(i-1) - energy_used(i);
                end
                %Check if there are any errors to the inverter and or it
                %for the general error checking
                app.error_ESS_inverter = read(app.ESSinverter.device,app.
                    ESSinverter.error_check.register_type,app.ESSinverter.
                    error_check.address+1,1,app.ESSinverter.error_check.unitid);
                if app.error_ESS_inverter > 0
                    error_ESS_inverter_binary = 1;
                else
                    error_ESS_inverter_binary = 0;
                end
                app.error_PV_meter = read(app.PVmeter.device,app.PVmeter.
                    error_check.register_type,app.PVmeter.error_check.address+1,1,
                    app.PVmeter.error_check.unitid);
                if app.error_PV_meter > 0
                    error_PV_inverter_binary = 1;
                else
                    error_PV_inverter_binary = 0;
                end
                battery_voltage = read(app.ESSinverter.device, app.Battery.
                    voltage.register_type, app.Battery.voltage.address+1, 1, app.
                    Battery.voltage.unitid, app.Battery.voltage.type)/app.Battery.
                    voltage.scalefactor;
                battery_soc = read(app.ESSinverter.device, app.Battery.soc.
                    register_type, app.Battery.soc.address+1, 1, app.Battery.soc.
                    unitid, app.Battery.soc.type)/app.Battery.soc.scalefactor;
                if battery_voltage > 57.4 || battery_voltage < 42
                    app.error_Battery = 1;
                else
                    if battery_soc > 90 || battery_soc < 10
                        app.error_Battery = 1;
                    else
                        app.error_Battery = 0;
                    end
                end

                if app.error_Battery == 0
                    app.StatusofBattery.Color = [0/255 255/255 0/255];
                elseif app.error_Battery > 0
                    app.StatusofBattery.Color = [255/255 92/255 57/255];
                end
                app.error_Grid_meter = read(app.Gridmeter.device,app.Gridmeter.
                    error_check.register_type,app.Gridmeter.error_check.address
                    +1,1,app.Gridmeter.error_check.unitid);
                if app.error_Grid_meter == 0
                    app.StatusofGridmeter.Color = [0/255 255/255 0/255];
                elseif app.error_Grid_meter > 0
                    app.StatusofGridmeter.Color = [255/255 92/255 57/255];
                end

%                 app.error_PV_inverter = read(app.modbus_device_1,'coils
    ',11,1,1);
                error_pre_pre = or(error_PV_inverter_binary,app.error_Grid_meter)
```

```
    ;
error_pre = or(error_pre_pre,app.error_Battery);
app.error  = or(app.error_PV_meter,error_pre);
axis([i-30,i,-1500,1500])
xlabel('Datapoint');
ylabel('Power (W)')
if string(app.turnonread) == "On"

    % write(app.modbus_device_1,'holdingregs',app.startadress,app
       .pv_data(i),1,'single')
    % app.voltage = read(app.modbus_device_1,'holdingregs',app.
       startadress,app.count,'single');
    % voltage_array(i) = app.voltage;
    % plot(voltage_array(1:i),'Color',[255/255, 209/255, 0])
    % drawnow
    % i = i+1;
    % xlim([i-50 i])
    % pause(0.10)

    if (string(app.controlmode) == "Voltage Control")
        app.permission_to_control_power = 1;
        %Read values first = read(app.ESSinverter.device,app.
           ESSinverter.error_check.register_type,app.ESSinverter.
           error_check.address+1,1,app.ESSinverter.error_check.
           unitid);
        pv_power = 190 + read(app.PVmeter.device,app.PVmeter.
           output_power.register_type,app.PVmeter.output_power.
           address+1,1,app.PVmeter.output_power.unitid,app.
           PVmeter.output_power.type)/app.PVmeter.output_power.
           scalefactor;
        grid_voltage = read(app.Gridmeter.device,app.Gridmeter.
           output_voltage.register_type,app.Gridmeter.
           output_voltage.address+1,1,app.Gridmeter.
           output_voltage.unitid,app.Gridmeter.output_voltage.
           type)/app.Gridmeter.output_voltage.scalefactor;
        % ESS_set_output_power = read(app.modbus_device_1,"
           inputregs",843,1,100,'int16')/1;
        % battery_power = -read(app.ESSinverter.device, app.
           ESSinverter.output_power.register_type, app.
           ESSinverter.output_power.address+1, 1, app.ESSinverter
           .output_power.unitid, app.ESSinverter.output_power.
           type)/app.ESSinverter.output_power.scalefactor;

        battery_voltage = read(app.ESSinverter.device, app.
           Battery.voltage.register_type, app.Battery.voltage.
           address+1, 1, app.Battery.voltage.unitid, app.Battery.
           voltage.type)/app.Battery.voltage.scalefactor;
        battery_current = read(app.ESSinverter.device, app.
           Battery.current.register_type, app.Battery.current.
           address+1, 1, app.Battery.current.unitid, app.Battery.
           current.type)/app.Battery.current.scalefactor;
        grid_power = 190 + read(app.Gridmeter.device,app.
           Gridmeter.output_power.register_type,app.Gridmeter.
           output_power.address+1,1,app.Gridmeter.output_power.
           unitid,app.Gridmeter.output_power.type)/app.Gridmeter.
           output_power.scalefactor;
        battery_soc = read(app.ESSinverter.device, app.Battery.
           soc.register_type, app.Battery.soc.address+1, 1, app.
           Battery.soc.unitid, app.Battery.soc.type)/app.Battery.
           soc.scalefactor;
        battery_power = grid_power - pv_power;
```

```
battery_power_dc = -read(app.ESSinverter.device, app.
    ESSinverter.output_power.register_type, app.
    ESSinverter.output_power.address+1, 1, app.ESSinverter
    .output_power.unitid, app.ESSinverter.output_power.
    type)/app.ESSinverter.output_power.scalefactor;
%First layer of control - Ascending Cosinusoidal
%Weighted Moving Average
app.DropDown.FontColor = [0/255 0/255 204/255];
app.DropDown.FontWeight = 'bold';
weighting_multiplier = [];
weighting_multiplier = linspace(1,app.windowsize_voltage,
    app.windowsize_voltage);
weighting_fraction = [];
for k = 1:app.windowsize_voltage
    weighting_fraction(k) = cos((1/app.windowsize_voltage
        )*k*pi/2)^2;
end
weighting_fraction = flip(weighting_fraction);
sinus_MA_voltage_output_day(i) = sum(data_array_voltage
    (:,1).*weighting_fraction')/sum(weighting_fraction);
if app.permission_to_control_voltage < app.
    windowsize_voltage+1
    output_voltage(i) = grid_voltage;
else
    output_voltage(i) = sinus_MA_voltage_output_day(i);
end
grid_voltage_pre = data_array_voltage(1,1);
delta_v = output_voltage(i) - grid_voltage;
ESS_set_output_power_first = delta_v/app.sensitivity;
%Second layer of control - SOC deadzone control
if ESS_set_output_power_first > 0
    if app.soc(i) > app.soc_deadzone_lower_limit_voltage
        && app.soc(i) < app.
        soc_deadzone_upper_limit_voltage
        ESS_set_output_power_second =
            ESS_set_output_power_first;
    elseif app.soc(i) < app.
        soc_deadzone_lower_limit_voltage
        delta_soc = app.soc(i) - app.
            soc_deadzone_lower_limit_voltage;
        ESS_set_output_power_second =
            ESS_set_output_power_first*(1-abs(delta_soc)*
            app.correction_factor_discharging_voltage);
    elseif app.soc(i) > app.
        soc_deadzone_upper_limit_voltage
        ESS_set_output_power_second =
            ESS_set_output_power_first;
    end
else
    if app.soc(i) > app.soc_deadzone_lower_limit_voltage
        && app.soc(i) < app.
        soc_deadzone_upper_limit_voltage
        ESS_set_output_power_second =
            ESS_set_output_power_first;
    elseif app.soc(i) < app.
        soc_deadzone_lower_limit_voltage
        ESS_set_output_power_second =
            ESS_set_output_power_first;
    elseif app.soc(i) > app.
        soc_deadzone_upper_limit_voltage
        delta_soc = app.soc(i) - app.
```

```
                        soc_deadzone_upper_limit_voltage;
                    ESS_set_output_power_second =
                        ESS_set_output_power_first*(1-abs(delta_soc)*
                        app.correction_factor_charging_voltage);
                end
            end
            %Third layer of control - Idle Charge/Discharge
            if (ESS_set_output_power_second < app.
               power_threshold_idle_voltage &&
               ESS_set_output_power_second > -app.
               power_threshold_idle_voltage)
                if app.soc(i) > app.soc_upper_limit_power
                    ESS_set_output_power_third = app.
                        idle_discharge_power_ctrl;
                elseif app.soc(i) < app.soc_lower_limit_power
                    ESS_set_output_power_third = app.
                        idle_charge_power_ctrl;
                else
                    ESS_set_output_power_third =
                        ESS_set_output_power_second;
                end
            else
                ESS_set_output_power_third =
                   ESS_set_output_power_second;
            end
            %Checking if window is filled with new values to
            %proceed
            if app.permission_to_control_voltage < app.
               windowsize_voltage+1
                ESS_set_output_power_pre = 0;
            else
                ESS_set_output_power_pre = ESS_set_output_power_third
                    ;
            end
%                 ESS_set_output_power = floor(ESS_set_output_power_pre);
            % Hard coding to not overload the inverter and
            %Hard coded limit for now
            if ESS_set_output_power_pre > 1500
                ESS_set_output_power_pre = 1500;
            elseif ESS_set_output_power_pre < -1250
                ESS_set_output_power_pre = -1250;
            else
                ESS_set_output_power_pre = ESS_set_output_power_pre;
            end
            ESS_set_output_power = ESS_set_output_power_pre;
            ESS_set_output_power = -floor(ESS_set_output_power);
            %Change color of the text field to highlight which
            %control mode is used at the moment
            app.TargetESSoutputpowerTextArea.FontColor = [0/255 0/255
                204/255];
            app.TargetESSoutputpowerTextArea.FontWeight = 'bold';
            app.MeasuredVoltageTextArea.FontColor = [0/255 0/255
               204/255];
            app.MeasuredVoltageTextArea.FontWeight = 'bold';
            app.MeasuredPVoutputpowerTextArea.FontColor = [0/255
               0/255 0/255];
            app.MeasuredPVoutputpowerTextArea.FontWeight = 'normal';
            app.row_temp = [grid_voltage pv_power output_voltage(i)];
            app.permission_to_control_voltage = app.
               permission_to_control_voltage + 1;
        elseif (string(app.controlmode) == "Power Control")
```

```
app.permission_to_control_voltage = 1;
%Read values first = read(app.ESSinverter.device,app.
    ESSinverter.error_check.register_type,app.ESSinverter.
    error_check.address+1,1,app.ESSinverter.error_check.
    unitid);
pv_power = 190 + read(app.PVmeter.device,app.PVmeter.
    output_power.register_type,app.PVmeter.output_power.
    address+1,1,app.PVmeter.output_power.unitid,app.
    PVmeter.output_power.type)/app.PVmeter.output_power.
    scalefactor;
grid_voltage = read(app.Gridmeter.device,app.Gridmeter.
    output_voltage.register_type,app.Gridmeter.
    output_voltage.address+1,1,app.Gridmeter.
    output_voltage.unitid,app.Gridmeter.output_voltage.
    type)/app.Gridmeter.output_voltage.scalefactor;
% ESS_set_output_power = read(app.modbus_device_1,"
    inputregs",843,1,100,'int16')/1;
% battery_power = -read(app.ESSinverter.device, app.
    ESSinverter.output_power.register_type, app.
    ESSinverter.output_power.address+1, 1, app.ESSinverter
    .output_power.unitid, app.ESSinverter.output_power.
    type)/app.ESSinverter.output_power.scalefactor;
battery_voltage = read(app.ESSinverter.device, app.
    Battery.voltage.register_type, app.Battery.voltage.
    address+1, 1, app.Battery.voltage.unitid, app.Battery.
    voltage.type)/app.Battery.voltage.scalefactor;
battery_current = read(app.ESSinverter.device, app.
    Battery.current.register_type, app.Battery.current.
    address+1, 1, app.Battery.current.unitid, app.Battery.
    current.type)/app.Battery.current.scalefactor;
grid_power = 190 + read(app.Gridmeter.device,app.
    Gridmeter.output_power.register_type,app.Gridmeter.
    output_power.address+1,1,app.Gridmeter.output_power.
    unitid,app.Gridmeter.output_power.type)/app.Gridmeter.
    output_power.scalefactor;
battery_soc = read(app.ESSinverter.device, app.Battery.
    soc.register_type, app.Battery.soc.address+1, 1, app.
    Battery.soc.unitid, app.Battery.soc.type)/app.Battery.
    soc.scalefactor;
battery_power = grid_power - pv_power;
battery_power_dc = -read(app.ESSinverter.device, app.
    ESSinverter.output_power.register_type, app.
    ESSinverter.output_power.address+1, 1, app.ESSinverter
    .output_power.unitid, app.ESSinverter.output_power.
    type)/app.ESSinverter.output_power.scalefactor;
%First layer of control - Ascending Cosinusoidal
%Weighted Moving Average
app.DropDown.FontColor = [127/255 0/255 255/255];
app.DropDown.FontWeight = 'bold';
weighting_multiplier = [];
weighting_multiplier = linspace(1,app.windowsize_power,
    app.windowsize_power);
weighting_fraction = [];
for k = 1:app.windowsize_power
     weighting_fraction(k) = cos((1/app.windowsize_power)*
        k*pi/2)^2;
end
weighting_fraction = flip(weighting_fraction);
sinus_MA_power_output_day(i) = sum(data_array_power(:,2)
    .*weighting_fraction')/sum(weighting_fraction);
if app.permission_to_control_power < app.windowsize_power
```

```
                        +1
                        output_power(i) = pv_power;
                    else
                        output_power(i) = sinus_MA_power_output_day(i);
                    end
                    pv_power_pre = data_array_power(1,2);
                    ESS_set_output_power_first = output_power(i) - pv_power;
                    %Second layer of control - SOC deadzone control
                    if ESS_set_output_power_first > 0
                        if app.soc(i) > app.soc_deadzone_lower_limit_power &&
                            app.soc(i) < app.soc_deadzone_upper_limit_power
                            ESS_set_output_power_second =
                                ESS_set_output_power_first;
                        elseif app.soc(i) < app.
                            soc_deadzone_lower_limit_power
                            delta_soc = app.soc(i) - app.
                                soc_deadzone_lower_limit_power;
                            ESS_set_output_power_second =
                                ESS_set_output_power_first*(1-abs(delta_soc)*
                                app.correction_factor_discharging_power);
                        elseif app.soc(i) > app.
                            soc_deadzone_upper_limit_power
                            ESS_set_output_power_second =
                                ESS_set_output_power_first;
                        end
                    else
                        if app.soc(i) > app.soc_deadzone_lower_limit_power &&
                            app.soc(i) < app.soc_deadzone_upper_limit_power
                            ESS_set_output_power_second =
                                ESS_set_output_power_first;
                        elseif app.soc(i) < app.
                            soc_deadzone_lower_limit_power
                            ESS_set_output_power_second =
                                ESS_set_output_power_first;
                        elseif app.soc(i) > app.
                            soc_deadzone_upper_limit_power
                            delta_soc = app.soc(i) - app.
                                soc_deadzone_upper_limit_power;
                            ESS_set_output_power_second =
                                ESS_set_output_power_first*(1-abs(delta_soc)*
                                app.correction_factor_charging_power);
                        end
                    end
                    %Third layer of control - Idle Charge/Discharge
                    if (ESS_set_output_power_second < app.
                        power_threshold_idle_power &&
                        ESS_set_output_power_second > -app.
                        power_threshold_idle_voltage)
                        if app.soc(i) > app.soc_upper_limit_power
                            ESS_set_output_power_third = app.
                                idle_discharge_power_ctrl;
%                             ESS_set_output_power_third = app.
    idle_discharge_power_factor*pv_power;
                        elseif app.soc(i) < app.soc_lower_limit_power
                            ESS_set_output_power_third = app.
                                idle_charge_power_ctrl;
                        else
                            ESS_set_output_power_third =
                                ESS_set_output_power_second;
                        end
                    else
```

```
                        ESS_set_output_power_third =
                            ESS_set_output_power_second;
                    end
                    %Checking if window is filled with new values to
                    %proceed
                    if app.permission_to_control_power < app.windowsize_power
                        +1
                        ESS_set_output_power_pre = 0;
                    else
                        ESS_set_output_power_pre = ESS_set_output_power_third
                            ;
                    end
                    %Hard coded limit for now
                    if ESS_set_output_power_pre > 1500
                        ESS_set_output_power_pre = 1500;
                    elseif ESS_set_output_power_pre < -1250
                        ESS_set_output_power_pre = -1250;
                    else
                        ESS_set_output_power_pre = ESS_set_output_power_pre;
                    end
                    ESS_set_output_power = ESS_set_output_power_pre;
                    ESS_set_output_power = -floor(ESS_set_output_power);
%                     set_voltage_power = grid_voltage + app.sensitivity*
    ESS_set_output_power;
                    %Change color of the text field to highlight which
                    %control mode is used at the moment
                    app.row_temp = [grid_voltage pv_power grid_voltage];
                    app.MeasuredVoltageTextArea.FontColor = [0/255 0/255
                        0/255];
                    app.MeasuredVoltageTextArea.FontWeight = 'normal';
                    app.TargetESSoutputpowerTextArea.FontColor = [127/255
                        0/255 255/255];
                    app.TargetESSoutputpowerTextArea.FontWeight = 'bold';
                    app.MeasuredPVoutputpowerTextArea.FontColor = [127/255
                        0/255 255/255];
                    app.MeasuredPVoutputpowerTextArea.FontWeight = 'bold';
                    app.permission_to_control_power = app.
                        permission_to_control_power + 1;
                elseif (string(app.controlmode) == "Read Values Only")
                    app.permission_to_control_voltage = 1;
                    app.permission_to_control_power = 1;
                    % Read the values first
                    pv_power = 190 + read(app.PVmeter.device,app.PVmeter.
                        output_power.register_type,app.PVmeter.output_power.
                        address+1,1,app.PVmeter.output_power.unitid,app.
                        PVmeter.output_power.type)/app.PVmeter.output_power.
                        scalefactor;
                    grid_voltage = read(app.Gridmeter.device,app.Gridmeter.
                        output_voltage.register_type,app.Gridmeter.
                        output_voltage.address+1,1,app.Gridmeter.
                        output_voltage.unitid,app.Gridmeter.output_voltage.
                        type)/app.Gridmeter.output_voltage.scalefactor;
                    % ESS_set_output_power = read(app.modbus_device_1,"
                        inputregs",843,1,100,'int16')/1;
                    % battery_power = -read(app.ESSinverter.device, app.
                        ESSinverter.output_power.register_type, app.
                        ESSinverter.output_power.address+1, 1, app.ESSinverter
                        .output_power.unitid, app.ESSinverter.output_power.
                        type)/app.ESSinverter.output_power.scalefactor;
                    battery_voltage = read(app.ESSinverter.device, app.
                        Battery.voltage.register_type, app.Battery.voltage.
```

```
                    address+1, 1, app.Battery.voltage.unitid, app.Battery.
                    voltage.type)/app.Battery.voltage.scalefactor;
                battery_current = read(app.ESSinverter.device, app.
                    Battery.current.register_type, app.Battery.current.
                    address+1, 1, app.Battery.current.unitid, app.Battery.
                    current.type)/app.Battery.current.scalefactor;
                grid_power = 190 + read(app.Gridmeter.device,app.
                    Gridmeter.output_power.register_type,app.Gridmeter.
                    output_power.address+1,1,app.Gridmeter.output_power.
                    unitid,app.Gridmeter.output_power.type)/app.Gridmeter.
                    output_power.scalefactor;
                battery_power = grid_power - pv_power;
                battery_soc = read(app.ESSinverter.device, app.Battery.
                    soc.register_type, app.Battery.soc.address+1, 1, app.
                    Battery.soc.unitid, app.Battery.soc.type)/app.Battery.
                    soc.scalefactor;
                battery_power_dc = -read(app.ESSinverter.device, app.
                    ESSinverter.output_power.register_type, app.
                    ESSinverter.output_power.address+1, 1, app.ESSinverter
                    .output_power.unitid, app.ESSinverter.output_power.
                    type)/app.ESSinverter.output_power.scalefactor;
                %                           ac_load_power = read(app.
                    ESSinverter.device,"inputregs",818,1,100,'uint16')/1;
                % Write to ESS inverter to not pump any power
                % write(app.ESSinverter.device,app.ESSinverter.set_power.
                    register_type,app.ESSinverter.set_power.address+1,
                    floor(pv_power),app.ESSinverter.set_power.unitid,app.
                    ESSinverter.set_power.type);
                app.DropDown.FontColor = [0/255 0/255 0/255];
                app.DropDown.FontWeight = 'normal';
                app.TargetESSoutputpowerTextArea.FontColor = [0/255 0/255
                     0/255];
                app.TargetESSoutputpowerTextArea.FontWeight = 'normal';
                app.MeasuredVoltageTextArea.FontColor = [0/255 0/255
                    0/255];
                app.MeasuredVoltageTextArea.FontWeight = 'normal';
                app.MeasuredPVoutputpowerTextArea.FontColor = [0/255
                    0/255 0/255];
                app.MeasuredPVoutputpowerTextArea.FontWeight = 'normal';
                ESS_set_output_power = -floor(0);
                ESS_set_output_power_first = ESS_set_output_power;
                ESS_set_output_power_second = ESS_set_output_power;
                ESS_set_output_power_third = ESS_set_output_power;
                ESS_set_output_power_pre = ESS_set_output_power;
%                 battery_power = 0;
                app.row_temp = [grid_voltage pv_power grid_voltage];
            end
            %Update MA array for both Voltage and Power control
            temp_row_voltage = [app.row_temp; data_array_voltage];
            data_array_voltage = temp_row_voltage(1:app.
                windowsize_voltage,:);
            temp_row_power = [app.row_temp; data_array_power];
            data_array_power = temp_row_power(1:app.windowsize_power,:);
            %Measure the voltage fluctuations and output it on the
            %text field.
%             battery_soc_victron = read(app.ESSinverter.device, "
    inputregs", 844, 1, 100, 'uint16');
            % app.MeasuredBatteryPowerTextArea.Value = num2str(
                battery_power);
            cell_voltage_avg = battery_voltage/14;
            %Write to the ESS the power command = read(app.ESSinverter.
```

```
    device,app.ESSinverter.error_check.register_type,app.
    ESSinverter.error_check.address+1,1,app.ESSinverter.
    error_check.unitid);

% if battery_voltage < 42 || battery_voltage > 57.4
%     if string(app.controlmode) == "Read Values Only"
%         ESS_set_output_power = 0;
%         write(app.ESSinverter.device,app.ESSinverter.
    set_power.register_type,app.ESSinverter.set_power.address
    +1,0,app.ESSinverter.set_power.unitid,app.ESSinverter.
    set_power.type);
%         ESS_set_output_power_actual = 0;
%     else
%         ESS_set_output_power = 0;
%         write(app.ESSinverter.device,app.ESSinverter.
    set_power.register_type,app.ESSinverter.set_power.address
    +1,0,app.ESSinverter.set_power.unitid,app.ESSinverter.
    set_power.type);
%         ESS_set_output_power_actual = 0;
%     end
%
% else
%     if string(app.controlmode) == "Read Values Only"
%         ESS_set_output_power = 0;
%         write(app.ESSinverter.device,app.ESSinverter.
    set_power.register_type,app.ESSinverter.set_power.address
    +1,0,app.ESSinverter.set_power.unitid,app.ESSinverter.
    set_power.type);
%         ESS_set_output_power_actual = 0;
%     else
%         % ESS_set_output_power_actual = 0;
%         write(app.ESSinverter.device,app.ESSinverter.
    set_power.register_type,app.ESSinverter.set_power.address
    +1,ESS_set_output_power,app.ESSinverter.set_power.unitid,
    app.ESSinverter.set_power.type);
%         ESS_set_output_power_actual = ESS_set_output_power;
%     end
% end
now_data = datetime('now','Format','d-MMM-y HH:mm:ss.SSS');
if string(app.controlmode) == "Read Values Only" || (
    battery_voltage < 42 || battery_voltage > 57.4)
    if hour(now_data) > 5 && hour(now_data) < 22
        ESS_set_output_power_actual = 0;
    end
else
    if hour(now_data) > 5 && hour(now_data) < 22
        write(app.ESSinverter.device,app.ESSinverter.
            set_power.register_type,app.ESSinverter.set_power.
            address+1,ESS_set_output_power,app.ESSinverter.
            set_power.unitid,app.ESSinverter.set_power.type);
        ESS_set_output_power_actual = read(app.ESSinverter.
            device,app.ESSinverter.set_power.register_type,app
            .ESSinverter.set_power.address+1,1,app.ESSinverter
            .set_power.unitid,app.ESSinverter.set_power.type);
    else
        ESS_set_output_power_actual = 0;
    end
end
%Delay to ensure writing is happening
pause(app.delay_messages);
elapsed_time = toc;
```

```
                    %Output values to the GUI
                    % app.MeasuredVoltageTextArea.Value = num2str(app.row_temp(1)
                        );
                    app.MeasuredPVoutputpowerTextArea.Value = num2str(app.
                        row_temp(2));
                    app.TargetoutputvoltageTextArea.Value = num2str(app.row_temp
                        (3));
                    app.BatterySOCTextArea.Value = num2str(app.soc(i));
                    app.AverageCellVoltageTextArea.Value = num2str(
                        cell_voltage_avg);
                    app.TargetESSoutputpowerTextArea.Value = num2str(-
                        ESS_set_output_power);
                    app.MeasuredBatterySOCTextArea.Value = num2str(battery_soc);
                    app.TargetESSoutputpower1LayerTextArea.Value = num2str(
                        ESS_set_output_power_first);
                    app.TargetESSoutputpower2LayerTextArea.Value = num2str(
                        ESS_set_output_power_second);
                    app.TargetESSoutputpower3LayerTextArea.Value = num2str(
                        ESS_set_output_power_third);
                    app.TargetESSoutputpower4LayerTextArea.Value = num2str(
                        ESS_set_output_power_pre);
%                     app.PoweratPCCTextArea.Value = num2str(app.row_temp(2) +
    ESS_set_output_power);
                    % app.PoweratPCCTextArea.Value = num2str(grid_power);
                    app.StatusofPVmeter.Color = [0/255 255/255 0/255];
                    app.StatusofGridmeter.Color = [0/255 255/255 0/255];
                    grid_voltage_new = read(app.Gridmeter.device,app.Gridmeter.
                        output_voltage.register_type,app.Gridmeter.output_voltage.
                        address+1,1,app.Gridmeter.output_voltage.unitid,app.
                        Gridmeter.output_voltage.type)/app.Gridmeter.
                        output_voltage.scalefactor;
                    grid_power_new = 190 + read(app.Gridmeter.device,app.
                        Gridmeter.output_power.register_type,app.Gridmeter.
                        output_power.address+1,1,app.Gridmeter.output_power.unitid
                        ,app.Gridmeter.output_power.type)/app.Gridmeter.
                        output_power.scalefactor;
                    battery_power_new = grid_power_new - pv_power;
                    pv_power_new = 190 + read(app.PVmeter.device,app.PVmeter.
                        output_power.register_type,app.PVmeter.output_power.
                        address+1,1,app.PVmeter.output_power.unitid,app.PVmeter.
                        output_power.type)/app.PVmeter.output_power.scalefactor;
                    app.MeasuredVoltageTextArea.Value = num2str(grid_voltage_new)
                        ;
                    app.PoweratPCCTextArea.Value = num2str(grid_power_new);
                    app.MeasuredBatteryPowerTextArea.Value = num2str(
                        battery_power_new);
                    voltage_array(i) = grid_voltage_new;
                    if i == 1
                        app.MeasuredVoltageFluctuationEditField.Value = num2str
                            (0);
                        app.MeasuredVoltageFluctuationEditField.FontColor =
                            [0/255 0/255 0/255];
                        app.MeasuredVoltageFluctuationEditField.FontWeight = '
                            normal';
                    else
                        difference_voltage = abs((voltage_array(i) -
                            voltage_array(i-1)));
                        app.MeasuredVoltageFluctuationEditField.Value = num2str(
                            difference_voltage);
                        if difference_voltage > 0.0084*230
                            app.MeasuredVoltageFluctuationEditField.FontColor =
```

```
                        [255/255 92/255 57/255];
                    app.MeasuredVoltageFluctuationEditField.FontWeight =
                        'bold';
                elseif difference_voltage > 0.0036*230
                    app.MeasuredVoltageFluctuationEditField.FontColor =
                        [255/255 153/255 51/255];
                    app.MeasuredVoltageFluctuationEditField.FontWeight =
                        'normal';
                else
                    app.MeasuredVoltageFluctuationEditField.FontColor =
                        [0/255 0/255 0/255];
                    app.MeasuredVoltageFluctuationEditField.FontWeight =
                        'normal';
                end
            end
            %Build data to be exported
            app.data_export{i,1} = datetime('now','Format','d-MMM-y HH:mm
                :ss.SSS');
            app.data_export{i,2} = grid_voltage_new;
            app.data_export{i,3} = app.row_temp(3);
            app.data_export{i,4} = -ESS_set_output_power_actual;
            app.data_export{i,5} = -ESS_set_output_power;
            app.data_export{i,6} = battery_power_new;
            app.data_export{i,7} = pv_power_new;
%             app.data_export{i,6} = app.row_temp(2) +
    ESS_set_output_power;
            app.data_export{i,8} = grid_power_new;
            app.data_export{i,9} = app.soc(i);
            app.data_export{i,10} = app.energy_Wh(i);
            app.data_export{i,11} = app.controlmode;
            app.data_export{i,12} = ESS_set_output_power_first;
            app.data_export{i,13} = ESS_set_output_power_second;
            app.data_export{i,14} = ESS_set_output_power_third;
            app.data_export{i,15} = ESS_set_output_power_pre;
            app.data_export{i,16} = battery_voltage;
            app.data_export{i,17} = cell_voltage_avg;
            app.data_export{i,18} = battery_current;
            app.data_export{i,19} = battery_soc;
            app.data_export{i,20} = battery_power_dc;
            i = i + 1;
            app.nr_cycles = app.nr_cycles + 1;
            % drawnow;
            % plot(app.UIAxes,ESS_set_output_power)
            addpoints(h,i,battery_power_new);
            grid on;
            drawnow;
        elseif string(app.turnonread) == "On" & app.error == 1
            %Write to ESS inverter a power command for ZERO power
            %and reset the MA.
            app.permission_to_control_voltage = 1;
            app.permission_to_control_power = 1;
%             write(app.modbus_device_1,'holdingregs',app.startadress
    ,0,1,'single')
            %Color the lamps to RED if there is an error with the
            %inverters
            if app.error_ESS_inverter == 0
                app.StatusofGridmeter.Color = [0/255 255/255 0/255];
            elseif app.error_ESS_inverter > 0
                app.StatusofGridmeter.Color = [255/255 92/255 57/255];
                formatError = "ESS inverter experienced error : %d";
                errormessage = compose(formatError,app.error_ESS_inverter
```

```
                        );
                    msgbox(errormessage);
                end

                if app.error_PV_meter == 0
                    app.StatusofPVmeter.Color = [0/255 255/255 0/255];
                elseif app.error_PV_meter == 1
                    app.StatusofPVmeter.Color = [255/255 92/255 57/255];
                end

                pause(app.delay_messages);
            elseif string(app.turnonread) == "Off" & app.error == 1
                %Write to ESS inverter a power command for ZERO power
                %and reset the MA.
                app.permission_to_control_voltage = 1;
                app.permission_to_control_power = 1;
%                 write(app.modbus_device_1,'holdingregs',app.startadress
    ,0,1,'single')
                %Color the lamps to RED if there is an error with the
                %inverters
                if app.error_ESS_inverter == 0
                    app.StatusofGridmeter.Color = [0/255 255/255 0/255];
                elseif app.error_ESS_inverter > 0
                    app.StatusofGridmeter.Color = [255/255 92/255 57/255];
                    formatError = "ESS inverter experienced error : %d";
                    errormessage = compose(formatError,app.error_ESS_inverter
                        );
                    msgbox(errormessage);
                end

                if app.error_PV_meter == 0
                    app.StatusofPVmeter.Color = [0/255 255/255 0/255];
                elseif app.error_PV_meter == 1
                    app.StatusofPVmeter.Color = [255/255 92/255 57/255];
                end

                pause(app.delay_messages);
            else
                %When paused, reset MA and pause writing to exported
                %data
                app.permission_to_control_voltage = 1;
                app.permission_to_control_power = 1;
%                 write(app.modbus_device_1,'holdingregs',app.startadress
    ,0,1,'single')
                app.StatusofGridmeter.Color = [0/255 255/255 0/255];
                app.data_export{i,1} = datetime('now','Format','d-MMM-y HH:mm
                    :ss.SSS');
                app.data_export{i,2} = app.row_temp(1);
                app.data_export{i,3} = app.row_temp(3);
                app.data_export{i,4} = 0;
                app.data_export{i,5} = 0;
                app.data_export{i,6} = battery_power;
                app.data_export{i,7} = pv_power_new;
                app.data_export{i,8} = grid_power_new;
                app.data_export{i,9} = app.soc(i);
                app.data_export{i,10} = app.energy_Wh(i);
                app.data_export{i,11} = 'Pause';
                app.data_export{i,12} = ESS_set_output_power_first;
                app.data_export{i,13} = ESS_set_output_power_second;
                app.data_export{i,14} = ESS_set_output_power_third;
                app.data_export{i,15} = ESS_set_output_power_pre;
```

```
            app.data_export{i,16} = battery_voltage;
            app.data_export{i,17} = cell_voltage_avg;
            app.data_export{i,18} = battery_current;
            app.data_export{i,19} = battery_soc;
            app.data_export{i,20} = battery_power_dc;
            pause(app.delay_messages);
        end
    end
    %Again if there is an error, change the color of the lamp to
    %RED
    if app.error_ESS_inverter == 0
        app.StatusofGridmeter.Color = [0/255 255/255 0/255];
    elseif app.error_ESS_inverter > 0
        app.StatusofGridmeter.Color = [255/255 92/255 57/255];
        formatError = "ESS inverter experienced error : %d";
        errormessage = compose(formatError,app.error_ESS_inverter);
        msgbox(errormessage);
    end

    if app.error_PV_meter == 0
        app.StatusofPVmeter.Color = [0/255 255/255 0/255];
    elseif app.error_PV_meter == 1
        app.StatusofPVmeter.Color = [255/255 92/255 57/255];
    end
end

% Callback function
function IPofModbusDeviceEditFieldValueChanged(app, event)
    app.modbusserver = app.IPofModbusDeviceEditField.Value;
end

% Callback function
function PortnumberofModbusDeviceEditFieldValueChanged(app, event)
    app.modbusport = app.PortnumberofModbusDeviceEditField.Value;
end

% Value changed function: MeasuredVoltageTextArea
function MeasuredVoltageTextAreaValueChanged(app, event)
    % value = app.MeasuredVoltageTextArea.Value;

end

% Value changing function: MeasuredVoltageTextArea
function MeasuredVoltageTextAreaValueChanging(app, event)
    changingValue = event.Value;

end

% Value changed function: TargetoutputvoltageTextArea
function TargetoutputvoltageTextAreaValueChanged(app, event)
    % value = app.TargetoutputvoltageTextArea.Value;

end

% Value changed function: MeasuredPVoutputpowerTextArea
function MeasuredPVoutputpowerTextAreaValueChanged(app, event)
    % value = app.MeasuredPVoutputpowerTextArea.Value;

end

% Value changed function: TargetESSoutputpowerTextArea
```

```
        function TargetESSoutputpowerTextAreaValueChanged(app, event)
            % value = app.TargetESSoutputpowerTextArea.Value;
            %
        end

        % Value changed function: DropDown
        function DropDownValueChanged(app, event)
            app.controlmode = app.DropDown.Value;
            k = 1;
        end

        % Value changed function: PauseSwitch
        function PauseSwitchValueChanged(app, event)
            app.turnonread = app.PauseSwitch.Value;

        end

        % Value changed function: PoweratPCCTextArea
        function PoweratPCCTextAreaValueChanged(app, event)
            % value = app.PoweratPCCTextArea.Value;

        end

        % Value changed function: BatterySOCTextArea
        function BatterySOCTextAreaValueChanged(app, event)
            % value = app.BatterySOCTextArea.Value;
            %
        end

        % Button pushed function: ExportdatatoCSVfileButton
        function ExportdatatoCSVfileButtonPushed(app, event)
            test_data = struct;
            data_length = length(app.data_export);
            year_data = year(app.data_export{data_length,1});
            month_data = month(app.data_export{data_length,1});
            day_data = day(app.data_export{data_length,1});
            hour_data = hour(app.data_export{data_length,1});
            minute_data = minute(app.data_export{data_length,1});
            second_data =  floor(second(app.data_export{data_length,1}));
            formatspec = "C:/Users/lynri/OneDrive/Documents/MATLAB/ThesisData/
               data_%d_%d_%d_%d%d%d.mat";
            filename = compose(formatspec,year_data,month_data,day_data,hour_data
               ,minute_data,second_data);
            varNames = ["Time","Measured Voltage (V)","Set Voltage (V)", "Actual
               Set Power (W)","Sent Set Power (W)", "BESS Measured Power", "PV
               Output Power (W)", "PCC Power (W)", "BESS SOC (%)", "BESS Energy (
               Wh)", "Control Mode","ESS Setpoint One Layer","ESS Setpoint Two
               Layer","ESS Setpoint Three Layer","ESS Setpoint Fourth Layer","
               Battery Pack Voltage (V)","Average Cell Voltage (V)","Battery Pack
                Current (A)","Measured Battery Pack SoC (%)","Battery Power DC (W
               )"];
            data_table = cell2table(app.data_export,"VariableNames",varNames);
            config_table = cell2table(app.config,"VariableNames",["Date of Test",
                string(app.data_export{data_length,1}),"Bye","Have a great time
               ","Lynrick","Wix"]);
            test_data.data = data_table;
            test_data.configuration = config_table;
            save(filename,"-struct","test_data");
%             writetable(data_table,filename,'Sheet','Data');
%             writetable(config_table,filename,'Sheet', 'Configuration');
            msgbox("Succesfully exported the measured data!");
```

```
        end

        % Value changed function: MeasuredVoltageFluctuationEditField
        function MeasuredVoltageFluctuationEditFieldValueChanged(app, event)
%             value = app.MeasuredVoltageFluctuationEditField.Value;
%
        end

        % Button down function: UIAxes
        function UIAxesButtonDown(app, event)

        end

        % Value changed function: VoltageControlElementsforWindowEditField
        function VoltageControlElementsforWindowEditFieldValueChanged(app, event)
            % value = app.VoltageControlElementsforWindowEditField.Value;

        end

        % Value changed function: PowerControlElementsforWindowEditField
        function PowerControlElementsforWindowEditFieldValueChanged(app, event)
            % value = app.PowerControlElementsforWindowEditField.Value;

        end

        % Value changed function: MeasuredBatterySOCTextArea
        function MeasuredBatterySOCTextAreaValueChanged(app, event)
            % value = app.MeasuredBatterySOCTextArea.Value;

        end

        % Value changed function: TargetESSoutputpower1LayerTextArea
        function TargetESSoutputpower1LayerTextAreaValueChanged(app, event)
            % value = app.TargetESSoutputpower1LayerTextArea.Value;
            %
        end

        % Value changed function: TargetESSoutputpower2LayerTextArea
        function TargetESSoutputpower2LayerTextAreaValueChanged(app, event)
            % value = app.TargetESSoutputpower2LayerTextArea.Value;
            %
        end

        % Value changed function: TargetESSoutputpower3LayerTextArea
        function TargetESSoutputpower3LayerTextAreaValueChanged(app, event)
            % value = app.TargetESSoutputpower3LayerTextArea.Value;
            %
        end

        % Value changed function: TargetESSoutputpower4LayerTextArea
        function TargetESSoutputpower4LayerTextAreaValueChanged(app, event)
            % value = app.TargetESSoutputpower4LayerTextArea.Value;
            %
        end

        % Value changed function: MeasuredBatteryPowerTextArea
        function MeasuredBatteryPowerTextAreaValueChanged(app, event)
            % value = app.MeasuredBatteryPowerTextArea.Value;

        end
```

```
        % Callback function
        function CLEARVARIABLESButtonPushed(app, event)
            % clear app.data_export
            % clear app.ESSinverter
            % clear app.Gridmeter
            % clear app.PVmeter
            % clear app.config
            % app.time_day = {};
            % app.ess_power = [];
            % app.energy_Wh = [];
            % app.permission_to_control_power = 1;
            % app.permission_to_control_voltage = 1;
            % app.nr_cycles = 1;
            % msgbox("Succesfully clear data and variables!");
        end

        % Value changed function: AverageCellVoltageTextArea
        function AverageCellVoltageTextAreaValueChanged(app, event)
            % value = app.AverageCellVoltageTextArea.Value;

        end
    end

    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Get the file path for locating images
            pathToMLAPP = fileparts(mfilename('fullpath'));

            % Create UIFigure and hide until all components are created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.AutoResizeChildren = 'off';
            app.UIFigure.Color = [0.8 0.8 0.8];
            app.UIFigure.Position = [200 -100 1337 805];
            app.UIFigure.Name = 'MATLAB App';

            % Create UIAxes3
            app.UIAxes3 = uiaxes(app.UIFigure);
            title(app.UIAxes3, 'Idle Charge/Discharge - Power Control')
            xlabel(app.UIAxes3, 'SoC of Battery (%)')
            ylabel(app.UIAxes3, 'Power from ESS (W)')
            zlabel(app.UIAxes3, 'Z')
            app.UIAxes3.YGrid = 'on';
            app.UIAxes3.Position = [863 14 357 206];

            % Create UIAxes2
            app.UIAxes2 = uiaxes(app.UIFigure);
            title(app.UIAxes2, 'Idle Charge/Discharge - Voltage Control')
            xlabel(app.UIAxes2, 'SoC of Battery (%)')
            ylabel(app.UIAxes2, 'Power from ESS (W)')
            zlabel(app.UIAxes2, 'Z')
            app.UIAxes2.YGrid = 'on';
            app.UIAxes2.Position = [434 14 365 205];

            % Create UIAxes
            app.UIAxes = uiaxes(app.UIFigure);
            title(app.UIAxes, 'SoC Power Reduction Curve - Voltage Control')
            xlabel(app.UIAxes, 'SoC of Battery (%)')
```

```
ylabel(app.UIAxes, 'Reduction in Power (%)')
zlabel(app.UIAxes, 'Z')
app.UIAxes.YGrid = 'on';
app.UIAxes.ButtonDownFcn = createCallbackFcn(app, @UIAxesButtonDown,
    true);
app.UIAxes.Position = [24 14 366 206];

% Create UIAxes4
app.UIAxes4 = uiaxes(app.UIFigure);
title(app.UIAxes4, 'SoC Power Reduction Curve - Power Control')
xlabel(app.UIAxes4, 'SoC of Battery (%)')
ylabel(app.UIAxes4, 'Reduction in Power (%)')
zlabel(app.UIAxes4, 'Z')
app.UIAxes4.YGrid = 'on';
app.UIAxes4.Position = [25 229 366 206];

% Create Image
app.Image = uiimage(app.UIFigure);
app.Image.Position = [442 223 764 487];
app.Image.ImageSource = 'acsetup_gui.png';

% Create InitializeConnectionsButton
app.InitializeConnectionsButton = uibutton(app.UIFigure, 'push');
app.InitializeConnectionsButton.ButtonPushedFcn = createCallbackFcn(
    app, @InitializeConnectionsButtonPushed, true);
app.InitializeConnectionsButton.Position = [21 748 276 37];
app.InitializeConnectionsButton.Text = 'Initialize Connections';

% Create ReadwriteregistersButton
app.ReadwriteregistersButton = uibutton(app.UIFigure, 'push');
app.ReadwriteregistersButton.ButtonPushedFcn = createCallbackFcn(app,
     @ReadwriteregistersButtonPushed, true);
app.ReadwriteregistersButton.Position = [21 611 276 32];
app.ReadwriteregistersButton.Text = 'Read/write registers';

% Create MeasuredVoltageTextAreaLabel
app.MeasuredVoltageTextAreaLabel = uilabel(app.UIFigure);
app.MeasuredVoltageTextAreaLabel.HorizontalAlignment = 'right';
app.MeasuredVoltageTextAreaLabel.Position = [821 574 102 22];
app.MeasuredVoltageTextAreaLabel.Text = 'Measured Voltage';

% Create MeasuredVoltageTextArea
app.MeasuredVoltageTextArea = uitextarea(app.UIFigure);
app.MeasuredVoltageTextArea.ValueChangedFcn = createCallbackFcn(app,
    @MeasuredVoltageTextAreaValueChanged, true);
app.MeasuredVoltageTextArea.ValueChangingFcn = createCallbackFcn(app,
     @MeasuredVoltageTextAreaValueChanging, true);
app.MeasuredVoltageTextArea.HorizontalAlignment = 'center';
app.MeasuredVoltageTextArea.Position = [926 561 71 37];

% Create StatusofPVmeter
app.StatusofPVmeter = uilamp(app.UIFigure);
app.StatusofPVmeter.Position = [699 665 34 34];
app.StatusofPVmeter.Color = [1 1 1];

% Create StatusofGridmeter
app.StatusofGridmeter = uilamp(app.UIFigure);
app.StatusofGridmeter.Position = [917 665 34 34];
app.StatusofGridmeter.Color = [1 1 1];

% Create MeasuredPVoutputpowerLabel
```

```
app.MeasuredPVoutputpowerLabel = uilabel(app.UIFigure);
app.MeasuredPVoutputpowerLabel.HorizontalAlignment = 'right';
app.MeasuredPVoutputpowerLabel.Position = [505 536 81 30];
app.MeasuredPVoutputpowerLabel.Text = {'Measured PV '; 'output power'
    };

% Create MeasuredPVoutputpowerTextArea
app.MeasuredPVoutputpowerTextArea = uitextarea(app.UIFigure);
app.MeasuredPVoutputpowerTextArea.ValueChangedFcn = createCallbackFcn
    (app, @MeasuredPVoutputpowerTextAreaValueChanged, true);
app.MeasuredPVoutputpowerTextArea.HorizontalAlignment = 'center';
app.MeasuredPVoutputpowerTextArea.Position = [596 533 89 37];

% Create TargetoutputvoltageLabel
app.TargetoutputvoltageLabel = uilabel(app.UIFigure);
app.TargetoutputvoltageLabel.HorizontalAlignment = 'right';
app.TargetoutputvoltageLabel.Position = [1018 561 75 30];
app.TargetoutputvoltageLabel.Text = {'Target output'; ' voltage'};

% Create TargetoutputvoltageTextArea
app.TargetoutputvoltageTextArea = uitextarea(app.UIFigure);
app.TargetoutputvoltageTextArea.ValueChangedFcn = createCallbackFcn(
    app, @TargetoutputvoltageTextAreaValueChanged, true);
app.TargetoutputvoltageTextArea.HorizontalAlignment = 'center';
app.TargetoutputvoltageTextArea.Position = [1097 559 72 37];

% Create TargetESSoutputpowerLabel
app.TargetESSoutputpowerLabel = uilabel(app.UIFigure);
app.TargetESSoutputpowerLabel.HorizontalAlignment = 'right';
app.TargetESSoutputpowerLabel.Position = [565 420 138 29];
app.TargetESSoutputpowerLabel.Text = {'Target ESS '; 'output power'};

% Create TargetESSoutputpowerTextArea
app.TargetESSoutputpowerTextArea = uitextarea(app.UIFigure);
app.TargetESSoutputpowerTextArea.ValueChangedFcn = createCallbackFcn(
    app, @TargetESSoutputpowerTextAreaValueChanged, true);
app.TargetESSoutputpowerTextArea.HorizontalAlignment = 'center';
app.TargetESSoutputpowerTextArea.Position = [711 417 58 37];

% Create DropDownLabel
app.DropDownLabel = uilabel(app.UIFigure);
app.DropDownLabel.HorizontalAlignment = 'right';
app.DropDownLabel.Position = [35 688 65 22];
app.DropDownLabel.Text = 'Drop Down';

% Create DropDown
app.DropDown = uidropdown(app.UIFigure);
app.DropDown.Items = {'Read Values Only', 'Power Control', 'Voltage
    Control'};
app.DropDown.ValueChangedFcn = createCallbackFcn(app,
    @DropDownValueChanged, true);
app.DropDown.Placeholder = 'Read Values Only';
app.DropDown.Position = [115 673 167 51];
app.DropDown.Value = 'Read Values Only';

% Create PauseSwitchLabel
app.PauseSwitchLabel = uilabel(app.UIFigure);
app.PauseSwitchLabel.HorizontalAlignment = 'center';
app.PauseSwitchLabel.Position = [381 584 39 22];
app.PauseSwitchLabel.Text = 'Pause';
```

```
% Create PauseSwitch
app.PauseSwitch = uiswitch(app.UIFigure, 'toggle');
app.PauseSwitch.ValueChangedFcn = createCallbackFcn(app,
    @PauseSwitchValueChanged, true);
app.PauseSwitch.Position = [310 544 45 101];
app.PauseSwitch.Value = 'On';

% Create PoweratPCCTextAreaLabel
app.PoweratPCCTextAreaLabel = uilabel(app.UIFigure);
app.PoweratPCCTextAreaLabel.HorizontalAlignment = 'right';
app.PoweratPCCTextAreaLabel.Position = [842 533 81 22];
app.PoweratPCCTextAreaLabel.Text = 'Power at PCC';

% Create PoweratPCCTextArea
app.PoweratPCCTextArea = uitextarea(app.UIFigure);
app.PoweratPCCTextArea.ValueChangedFcn = createCallbackFcn(app,
    @PoweratPCCTextAreaValueChanged, true);
app.PoweratPCCTextArea.HorizontalAlignment = 'center';
app.PoweratPCCTextArea.Position = [926 521 71 37];

% Create BatterySOCTextAreaLabel
app.BatterySOCTextAreaLabel = uilabel(app.UIFigure);
app.BatterySOCTextAreaLabel.HorizontalAlignment = 'right';
app.BatterySOCTextAreaLabel.Position = [823 241 72 22];
app.BatterySOCTextAreaLabel.Text = 'Battery SOC';

% Create BatterySOCTextArea
app.BatterySOCTextArea = uitextarea(app.UIFigure);
app.BatterySOCTextArea.ValueChangedFcn = createCallbackFcn(app,
    @BatterySOCTextAreaValueChanged, true);
app.BatterySOCTextArea.HorizontalAlignment = 'center';
app.BatterySOCTextArea.Position = [905 234 58 37];

% Create ExportdatatoCSVfileButton
app.ExportdatatoCSVfileButton = uibutton(app.UIFigure, 'push');
app.ExportdatatoCSVfileButton.ButtonPushedFcn = createCallbackFcn(app
    , @ExportdatatoCSVfileButtonPushed, true);
app.ExportdatatoCSVfileButton.Position = [24 525 271 66];
app.ExportdatatoCSVfileButton.Text = 'Export data to CSV file';

% Create Image2
app.Image2 = uiimage(app.UIFigure);
app.Image2.Position = [419 718 223 66];
app.Image2.ImageSource = fullfile(pathToMLAPP, 'Stedin_logo.jpg');

% Create Image3
app.Image3 = uiimage(app.UIFigure);
app.Image3.Position = [933 719 309 66];
app.Image3.ImageSource = fullfile(pathToMLAPP, 'TU_delft_logo.jpg');

% Create MeasuredVoltageFluctuationEditFieldLabel
app.MeasuredVoltageFluctuationEditFieldLabel = uilabel(app.UIFigure);
app.MeasuredVoltageFluctuationEditFieldLabel.HorizontalAlignment = '
    right';
app.MeasuredVoltageFluctuationEditFieldLabel.FontSize = 16;
app.MeasuredVoltageFluctuationEditFieldLabel.FontWeight = 'bold';
app.MeasuredVoltageFluctuationEditFieldLabel.Position = [701 781 233
    22];
app.MeasuredVoltageFluctuationEditFieldLabel.Text = 'Measured Voltage
     Fluctuation';
```

```
% Create MeasuredVoltageFluctuationEditField
app.MeasuredVoltageFluctuationEditField = uieditfield(app.UIFigure, '
    text');
app.MeasuredVoltageFluctuationEditField.ValueChangedFcn =
    createCallbackFcn(app,
    @MeasuredVoltageFluctuationEditFieldValueChanged, true);
app.MeasuredVoltageFluctuationEditField.HorizontalAlignment = 'center
    ';
app.MeasuredVoltageFluctuationEditField.Position = [685 718 264 48];

% Create VoltageControlElementsforWindowEditFieldLabel
app.VoltageControlElementsforWindowEditFieldLabel = uilabel(app.
    UIFigure);
app.VoltageControlElementsforWindowEditFieldLabel.HorizontalAlignment
     = 'center';
app.VoltageControlElementsforWindowEditFieldLabel.FontColor = [0 0
    0.8];
app.VoltageControlElementsforWindowEditFieldLabel.Position = [25 462
    132 30];
app.VoltageControlElementsforWindowEditFieldLabel.Text = {'Voltage
    Control'; '# Elements for Window'};

% Create VoltageControlElementsforWindowEditField
app.VoltageControlElementsforWindowEditField = uieditfield(app.
    UIFigure, 'numeric');
app.VoltageControlElementsforWindowEditField.ValueChangedFcn =
    createCallbackFcn(app,
    @VoltageControlElementsforWindowEditFieldValueChanged, true);
app.VoltageControlElementsforWindowEditField.HorizontalAlignment = '
    center';
app.VoltageControlElementsforWindowEditField.FontColor = [0 0 0.8];
app.VoltageControlElementsforWindowEditField.Position = [165 453 44
    48];

% Create PowerControlElementsforWindowEditFieldLabel
app.PowerControlElementsforWindowEditFieldLabel = uilabel(app.
    UIFigure);
app.PowerControlElementsforWindowEditFieldLabel.HorizontalAlignment =
     'center';
app.PowerControlElementsforWindowEditFieldLabel.FontColor = [0.502 0
    1];
app.PowerControlElementsforWindowEditFieldLabel.Position = [228 462
    132 30];
app.PowerControlElementsforWindowEditFieldLabel.Text = {'Power
    Control'; '# Elements for Window'};

% Create PowerControlElementsforWindowEditField
app.PowerControlElementsforWindowEditField = uieditfield(app.UIFigure
    , 'numeric');
app.PowerControlElementsforWindowEditField.ValueChangedFcn =
    createCallbackFcn(app,
    @PowerControlElementsforWindowEditFieldValueChanged, true);
app.PowerControlElementsforWindowEditField.HorizontalAlignment = '
    center';
app.PowerControlElementsforWindowEditField.FontColor = [0.502 0 1];
app.PowerControlElementsforWindowEditField.Position = [368 453 44
    48];

% Create MeasuredBatterySOCLabel
app.MeasuredBatterySOCLabel = uilabel(app.UIFigure);
app.MeasuredBatterySOCLabel.HorizontalAlignment = 'right';
```

```
app.MeasuredBatterySOCLabel.Position = [641 237 76 30];
app.MeasuredBatterySOCLabel.Text = {'Measured'; ' Battery SOC'};

% Create MeasuredBatterySOCTextArea
app.MeasuredBatterySOCTextArea = uitextarea(app.UIFigure);
app.MeasuredBatterySOCTextArea.ValueChangedFcn = createCallbackFcn(
    app, @MeasuredBatterySOCTextAreaValueChanged, true);
app.MeasuredBatterySOCTextArea.HorizontalAlignment = 'center';
app.MeasuredBatterySOCTextArea.Position = [730 234 61 37];

% Create TargetESSoutputpower1LayerTextAreaLabel
app.TargetESSoutputpower1LayerTextAreaLabel = uilabel(app.UIFigure);
app.TargetESSoutputpower1LayerTextAreaLabel.HorizontalAlignment = '
    right';
app.TargetESSoutputpower1LayerTextAreaLabel.Position = [904 463 208
    22];
app.TargetESSoutputpower1LayerTextAreaLabel.Text = 'Target ESS output
     power - 1 Layer';

% Create TargetESSoutputpower1LayerTextArea
app.TargetESSoutputpower1LayerTextArea = uitextarea(app.UIFigure);
app.TargetESSoutputpower1LayerTextArea.ValueChangedFcn =
    createCallbackFcn(app,
    @TargetESSoutputpower1LayerTextAreaValueChanged, true);
app.TargetESSoutputpower1LayerTextArea.HorizontalAlignment = 'center'
    ;
app.TargetESSoutputpower1LayerTextArea.Position = [1121 448 59 37];

% Create TargetESSoutputpower2LayerTextAreaLabel
app.TargetESSoutputpower2LayerTextAreaLabel = uilabel(app.UIFigure);
app.TargetESSoutputpower2LayerTextAreaLabel.HorizontalAlignment = '
    right';
app.TargetESSoutputpower2LayerTextAreaLabel.Position = [905 420 208
    22];
app.TargetESSoutputpower2LayerTextAreaLabel.Text = 'Target ESS output
     power - 2 Layer';

% Create TargetESSoutputpower2LayerTextArea
app.TargetESSoutputpower2LayerTextArea = uitextarea(app.UIFigure);
app.TargetESSoutputpower2LayerTextArea.ValueChangedFcn =
    createCallbackFcn(app,
    @TargetESSoutputpower2LayerTextAreaValueChanged, true);
app.TargetESSoutputpower2LayerTextArea.HorizontalAlignment = 'center'
    ;
app.TargetESSoutputpower2LayerTextArea.Position = [1121 406 59 37];

% Create TargetESSoutputpower3LayerTextAreaLabel
app.TargetESSoutputpower3LayerTextAreaLabel = uilabel(app.UIFigure);
app.TargetESSoutputpower3LayerTextAreaLabel.HorizontalAlignment = '
    right';
app.TargetESSoutputpower3LayerTextAreaLabel.Position = [905 379 208
    22];
app.TargetESSoutputpower3LayerTextAreaLabel.Text = 'Target ESS output
     power - 3 Layer';

% Create TargetESSoutputpower3LayerTextArea
app.TargetESSoutputpower3LayerTextArea = uitextarea(app.UIFigure);
app.TargetESSoutputpower3LayerTextArea.ValueChangedFcn =
    createCallbackFcn(app,
    @TargetESSoutputpower3LayerTextAreaValueChanged, true);
app.TargetESSoutputpower3LayerTextArea.HorizontalAlignment = 'center'
```

```
    ;
app.TargetESSoutputpower3LayerTextArea.Position = [1121 364 59 37];

% Create TargetESSoutputpower4LayerTextAreaLabel
app.TargetESSoutputpower4LayerTextAreaLabel = uilabel(app.UIFigure);
app.TargetESSoutputpower4LayerTextAreaLabel.HorizontalAlignment = '
    right';
app.TargetESSoutputpower4LayerTextAreaLabel.Position = [906 336 208
    22];
app.TargetESSoutputpower4LayerTextAreaLabel.Text = 'Target ESS output
     power - 4 Layer';

% Create TargetESSoutputpower4LayerTextArea
app.TargetESSoutputpower4LayerTextArea = uitextarea(app.UIFigure);
app.TargetESSoutputpower4LayerTextArea.ValueChangedFcn =
    createCallbackFcn(app,
    @TargetESSoutputpower4LayerTextAreaValueChanged, true);
app.TargetESSoutputpower4LayerTextArea.HorizontalAlignment = 'center'
    ;
app.TargetESSoutputpower4LayerTextArea.Position = [1121 321 59 37];

% Create TargetESSoutputpower1LayerTextAreaLabel_2
app.TargetESSoutputpower1LayerTextAreaLabel_2 = uilabel(app.UIFigure)
    ;
app.TargetESSoutputpower1LayerTextAreaLabel_2.HorizontalAlignment = '
    right';
app.TargetESSoutputpower1LayerTextAreaLabel_2.Position = [905 443 208
     22];
app.TargetESSoutputpower1LayerTextAreaLabel_2.Text = 'Moving Average'
    ;

% Create TargetESSoutputpower1LayerTextAreaLabel_3
app.TargetESSoutputpower1LayerTextAreaLabel_3 = uilabel(app.UIFigure)
    ;
app.TargetESSoutputpower1LayerTextAreaLabel_3.HorizontalAlignment = '
    right';
app.TargetESSoutputpower1LayerTextAreaLabel_3.Position = [906 400 208
     22];
app.TargetESSoutputpower1LayerTextAreaLabel_3.Text = 'SoC Deadzone
    Reduction Curve';

% Create TargetESSoutputpower1LayerTextAreaLabel_4
app.TargetESSoutputpower1LayerTextAreaLabel_4 = uilabel(app.UIFigure)
    ;
app.TargetESSoutputpower1LayerTextAreaLabel_4.HorizontalAlignment = '
    right';
app.TargetESSoutputpower1LayerTextAreaLabel_4.Position = [906 358 208
     22];
app.TargetESSoutputpower1LayerTextAreaLabel_4.Text = 'Idle Charge/
    Discharge';

% Create TargetESSoutputpower1LayerTextAreaLabel_5
app.TargetESSoutputpower1LayerTextAreaLabel_5 = uilabel(app.UIFigure)
    ;
app.TargetESSoutputpower1LayerTextAreaLabel_5.HorizontalAlignment = '
    right';
app.TargetESSoutputpower1LayerTextAreaLabel_5.Position = [985 293 128
     44];
app.TargetESSoutputpower1LayerTextAreaLabel_5.Text = {'Window of
    Operation, '; 'Filled up window check'; 'and hard coded limit'};
```

```
% Create MeasuredBatteryPowerLabel
app.MeasuredBatteryPowerLabel = uilabel(app.UIFigure);
app.MeasuredBatteryPowerLabel.HorizontalAlignment = 'right';
app.MeasuredBatteryPowerLabel.Position = [596 469 107 30];
app.MeasuredBatteryPowerLabel.Text = {'Measured'; ' Battery Power'};

% Create MeasuredBatteryPowerTextArea
app.MeasuredBatteryPowerTextArea = uitextarea(app.UIFigure);
app.MeasuredBatteryPowerTextArea.ValueChangedFcn = createCallbackFcn(
    app, @MeasuredBatteryPowerTextAreaValueChanged, true);
app.MeasuredBatteryPowerTextArea.HorizontalAlignment = 'center';
app.MeasuredBatteryPowerTextArea.FontColor = [1 0.8196 0];
app.MeasuredBatteryPowerTextArea.Position = [710 462 59 37];

% Create DesignedandprogrammedbyLynrickWixLabel
app.DesignedandprogrammedbyLynrickWixLabel = uilabel(app.UIFigure);
app.DesignedandprogrammedbyLynrickWixLabel.FontWeight = 'bold';
app.DesignedandprogrammedbyLynrickWixLabel.Position = [468 227 96
    72];
app.DesignedandprogrammedbyLynrickWixLabel.Text = {'Designed and'; '
    programmed by'; 'Lynrick Wix'};

% Create StatusofESSInverter
app.StatusofESSInverter = uilamp(app.UIFigure);
app.StatusofESSInverter.Position = [852 410 34 34];
app.StatusofESSInverter.Color = [1 1 1];

% Create StatusofBattery
app.StatusofBattery = uilamp(app.UIFigure);
app.StatusofBattery.Position = [669 316 34 34];
app.StatusofBattery.Color = [1 1 1];

% Create TargetESSoutputpower1LayerTextAreaLabel_6
app.TargetESSoutputpower1LayerTextAreaLabel_6 = uilabel(app.UIFigure)
    ;
app.TargetESSoutputpower1LayerTextAreaLabel_6.HorizontalAlignment = '
    right';
app.TargetESSoutputpower1LayerTextAreaLabel_6.Position = [769 469 14
    22];
app.TargetESSoutputpower1LayerTextAreaLabel_6.Text = 'W';

% Create TargetESSoutputpower1LayerTextAreaLabel_7
app.TargetESSoutputpower1LayerTextAreaLabel_7 = uilabel(app.UIFigure)
    ;
app.TargetESSoutputpower1LayerTextAreaLabel_7.HorizontalAlignment = '
    right';
app.TargetESSoutputpower1LayerTextAreaLabel_7.Position = [769 424 14
    22];
app.TargetESSoutputpower1LayerTextAreaLabel_7.Text = 'W';

% Create TargetESSoutputpower1LayerTextAreaLabel_8
app.TargetESSoutputpower1LayerTextAreaLabel_8 = uilabel(app.UIFigure)
    ;
app.TargetESSoutputpower1LayerTextAreaLabel_8.HorizontalAlignment = '
    right';
app.TargetESSoutputpower1LayerTextAreaLabel_8.Position = [792 241 14
    22];
app.TargetESSoutputpower1LayerTextAreaLabel_8.Text = '%';

% Create TargetESSoutputpower1LayerTextAreaLabel_9
app.TargetESSoutputpower1LayerTextAreaLabel_9 = uilabel(app.UIFigure)
```

```
    ;
app.TargetESSoutputpower1LayerTextAreaLabel_9.HorizontalAlignment = '
    right';
app.TargetESSoutputpower1LayerTextAreaLabel_9.Position = [964 241 15
    22];
app.TargetESSoutputpower1LayerTextAreaLabel_9.Text = '%';

% Create TargetESSoutputpower1LayerTextAreaLabel_10
app.TargetESSoutputpower1LayerTextAreaLabel_10 = uilabel(app.UIFigure
    );
app.TargetESSoutputpower1LayerTextAreaLabel_10.HorizontalAlignment =
    'right';
app.TargetESSoutputpower1LayerTextAreaLabel_10.Position = [1183 453
    14 22];
app.TargetESSoutputpower1LayerTextAreaLabel_10.Text = 'W';

% Create TargetESSoutputpower1LayerTextAreaLabel_11
app.TargetESSoutputpower1LayerTextAreaLabel_11 = uilabel(app.UIFigure
    );
app.TargetESSoutputpower1LayerTextAreaLabel_11.HorizontalAlignment =
    'right';
app.TargetESSoutputpower1LayerTextAreaLabel_11.Position = [1183 414
    14 22];
app.TargetESSoutputpower1LayerTextAreaLabel_11.Text = 'W';

% Create TargetESSoutputpower1LayerTextAreaLabel_12
app.TargetESSoutputpower1LayerTextAreaLabel_12 = uilabel(app.UIFigure
    );
app.TargetESSoutputpower1LayerTextAreaLabel_12.HorizontalAlignment =
    'right';
app.TargetESSoutputpower1LayerTextAreaLabel_12.Position = [1183 371
    14 22];
app.TargetESSoutputpower1LayerTextAreaLabel_12.Text = 'W';

% Create TargetESSoutputpower1LayerTextAreaLabel_13
app.TargetESSoutputpower1LayerTextAreaLabel_13 = uilabel(app.UIFigure
    );
app.TargetESSoutputpower1LayerTextAreaLabel_13.HorizontalAlignment =
    'right';
app.TargetESSoutputpower1LayerTextAreaLabel_13.Position = [1183 328
    14 22];
app.TargetESSoutputpower1LayerTextAreaLabel_13.Text = 'W';

% Create TargetESSoutputpower1LayerTextAreaLabel_14
app.TargetESSoutputpower1LayerTextAreaLabel_14 = uilabel(app.UIFigure
    );
app.TargetESSoutputpower1LayerTextAreaLabel_14.HorizontalAlignment =
    'right';
app.TargetESSoutputpower1LayerTextAreaLabel_14.Position = [689 540 14
     22];
app.TargetESSoutputpower1LayerTextAreaLabel_14.Text = 'W';

% Create TargetESSoutputpower1LayerTextAreaLabel_15
app.TargetESSoutputpower1LayerTextAreaLabel_15 = uilabel(app.UIFigure
    );
app.TargetESSoutputpower1LayerTextAreaLabel_15.HorizontalAlignment =
    'right';
app.TargetESSoutputpower1LayerTextAreaLabel_15.Position = [997 528 14
     22];
app.TargetESSoutputpower1LayerTextAreaLabel_15.Text = 'W';
```

```
            % Create TargetESSoutputpower1LayerTextAreaLabel_16
            app.TargetESSoutputpower1LayerTextAreaLabel_16 = uilabel(app.UIFigure
                );
            app.TargetESSoutputpower1LayerTextAreaLabel_16.HorizontalAlignment =
                'right';
            app.TargetESSoutputpower1LayerTextAreaLabel_16.Position = [999 567 11
                 22];
            app.TargetESSoutputpower1LayerTextAreaLabel_16.Text = 'V';

            % Create TargetESSoutputpower1LayerTextAreaLabel_17
            app.TargetESSoutputpower1LayerTextAreaLabel_17 = uilabel(app.UIFigure
                );
            app.TargetESSoutputpower1LayerTextAreaLabel_17.HorizontalAlignment =
                'right';
            app.TargetESSoutputpower1LayerTextAreaLabel_17.Position = [1169 567
                11 22];
            app.TargetESSoutputpower1LayerTextAreaLabel_17.Text = 'V';

            % Create AverageCellVoltageLabel
            app.AverageCellVoltageLabel = uilabel(app.UIFigure);
            app.AverageCellVoltageLabel.HorizontalAlignment = 'right';
            app.AverageCellVoltageLabel.Position = [565 375 138 30];
            app.AverageCellVoltageLabel.Text = {'Average '; 'Cell Voltage'};

            % Create AverageCellVoltageTextArea
            app.AverageCellVoltageTextArea = uitextarea(app.UIFigure);
            app.AverageCellVoltageTextArea.ValueChangedFcn = createCallbackFcn(
                app, @AverageCellVoltageTextAreaValueChanged, true);
            app.AverageCellVoltageTextArea.HorizontalAlignment = 'center';
            app.AverageCellVoltageTextArea.Position = [711 372 58 37];

            % Create TargetESSoutputpower1LayerTextAreaLabel_18
            app.TargetESSoutputpower1LayerTextAreaLabel_18 = uilabel(app.UIFigure
                );
            app.TargetESSoutputpower1LayerTextAreaLabel_18.HorizontalAlignment =
                'right';
            app.TargetESSoutputpower1LayerTextAreaLabel_18.Position = [758 379 25
                 22];
            app.TargetESSoutputpower1LayerTextAreaLabel_18.Text = 'V';

            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = test_setup_thesis_ACSetup_ESPLab

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            if nargout == 0
                clear app
            end
        end
```

```
        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```

2.3. Matlab code for data post processing

```
%%
find_power_control = find(data.("Control Mode") == "Power Control" | data.("
    Control Mode") == "Pause");
data_power_control = data(find_power_control,:);
find_voltage_control = find(data.("Control Mode") == "Voltage Control" | data.("
    Control Mode") == "Pause");
data_voltage_control = data(find_voltage_control,:);
find_reading_values = find(data.("Control Mode") ==  "Read Values Only" | data.("
    Control Mode") == "Pause");
data_reading_values = data(find_reading_values,:);
time_string = data.Time;

aruba_blue = [63/255 144/255 223/255];
aruba_red = [239/255 48/255 62/255];
aruba_yellow = [255/255 210/255 0/255];
royal_orange = [255/255 154/255 0/255];
stedin_grey = [77/255 77/255 77/255];
aruba_green = [132/255,194/255,37/255];

delta_v_data = [];
difference_samplerate_SOC = [];
ramprate_PV_power = [];
ramprate_voltage_SOC = [];
difference_power_worst_day = [];
ramprate_PV = [];
delta_pv_p_data = [];
delta_pv_data = [];

% time_string = string(data(:,1));
% voltage_meas = cell2mat(data(:,2));
% pv_power_meas = cell2mat(data(:,7));

for i = 1:length(data.("Measured Voltage (V)"))-1
    delta_v_data(i) = data.("Measured Voltage (V)")(i+1) - data.("Measured
        Voltage (V)")(i);
%       delta_v_data(i) = voltage_meas(i+1) - voltage_meas(i);
%       delta_pv_data(i) = pv_power_meas(i+1) - pv_power_meas(i);
    delta_pv_p_data(i) = data.('PCC Power (W)')(i+1) - data.('PCC Power (W)')(i);
    delta_pv_data(i) = data.('PV Output Power (W)')(i+1) - data.('PV Output Power
        (W)')(i);
%     delta_v_data(i) = voltage_meas(i+1) - voltage_meas(i);
%     delta_pv_data(i) = pv_power_meas(i+1) - pv_power_meas(i);

    difference_samplerate_SOC(i) = pyrunfile('sample_rate_seconds.py','d',x =
        char(time_string(i)), y = char(time_string(i+1)));
    ramprate_voltage_SOC(i) = delta_v_data(i);
    ramprate_PV_power(i) = delta_pv_p_data(i);
    ramprate_PV(i) = delta_pv_data(i);
end
```

```
visible_flicker_curve = [1 40/80 18/80 20/80 23/80 0.36 30/80 38/80 43/80 50/80
    55/80 62/80 73/80 95/80 123/80 170/80 200/80];
annoying_flicker_curve = [115/80 55/80 35/80 45/80 55/80 68/80 76/80 93/80 110/80
     135/80 160/80 184/80 207/80 255/80 310/80 403/80 480/80];
visible_flicker_curve_polygon = [0 visible_flicker_curve 0 0];
annoying_flicker_curve_polygon = [0 annoying_flicker_curve 0 0];
time_dips_seconds = [0.05 0.1 0.2 0.3 1 2 3 6 12 30 60 120 180 360 720 1800
    3600];
time_dips_seconds_polygon = [0.025 time_dips_seconds 4000 0.025];
difference_voltage_percentage_control = ramprate_voltage_SOC*100/230;
idx_after = inpolygon(difference_samplerate_SOC,abs(
    difference_voltage_percentage_control),time_dips_seconds_polygon,
    visible_flicker_curve_polygon);
ida_after = inpolygon(difference_samplerate_SOC,abs(
    difference_voltage_percentage_control),time_dips_seconds_polygon,
    annoying_flicker_curve_polygon);

find_indexes_pos_MA = find(~idx_after == 1);
find_indexes_neg_MA = find(~ida_after == 1);
actual_indexes_pos_MA = find_indexes_pos_MA+1;
actual_indexes_neg_MA = find_indexes_neg_MA+1;

%% Plot Battery, PV VOLTAGE and AC voltage measurements
find_pv_flux = find(abs(ramprate_PV) > 300);
find_pv_flux_before = find_pv_flux + 1;
first_line_cmp = 'Battery power, PV power and PCC voltage measurements';
if string(configuration{2,2}) == "169.254.1.2"
    setup = "AC connected ESS experimental setup";
elseif string(configuration{2,2}) == "192.168.11.3"
    setup = "DC connected ESS experimental setup";
end
first_line = compose(first_line_cmp);
control_line_cmp = 'Control Mode : %s';
control_line = compose(control_line_cmp,string(data.("Control Mode")(1)));
second_line_cmp = 'Testing time : %s - %s';
second_line = compose(second_line_cmp,time_string(1),time_string(end));
filename_cmp = 'Bat_power_Measured_voltage_%s_%s_%s_%s_%s_%s_%d';
filename = compose(filename_cmp,string(data.("Control Mode")(1)),string(year(
    time_string(1))),string(month(time_string(1))),string(day(time_string(1))),
    string(hour(time_string(1))),string(minute(time_string(1))),floor(second(
    time_string(1))))
fig2 = figure;
% fig2.Position = [50 50 800 450];
set(fig2,'defaultAxesColorOrder',[[0 0 0]; [0 0 0]]);
% yyaxis left
% ylabel('Power (W)')
ph = gobjects(1);
ph(1) = plot(data.Time(1:end),data.("PV Output Power (W)")(1:end),'LineWidth',2,'
    Color',aruba_blue);
hold on
ph(2) = plot(data.Time,data.("BESS Measured Power"),'LineWidth',2,'Color',
    aruba_green);
hold on
ylabel('Power (W)')
yyaxis right
ph(3) = plot(data.Time,data.("Measured Voltage (V)"),'LineWidth',2,'Color',
    stedin_grey);
hold on
for j = 1:length(find_indexes_pos_MA)
    ph(j+3) = plot(time_string([find_indexes_pos_MA(j), actual_indexes_pos_MA(j)
        ]),data.("Measured Voltage (V)")([find_indexes_pos_MA(j),
```

```
        actual_indexes_pos_MA(j)]),  'Marker', '.', 'MarkerSize', 20,'Color',
        royal_orange, 'LineWidth', 2,'LineStyle','-');
    hold on
end
for j = 1:length(find_indexes_neg_MA)
    ph(j+length(find_indexes_pos_MA)+3) = plot(time_string([find_indexes_neg_MA(j
        ), actual_indexes_neg_MA(j)]),data.("Measured Voltage (V)")([
        find_indexes_neg_MA(j), actual_indexes_neg_MA(j)]), 'Marker', '.', '
        MarkerSize', 25, 'Color',aruba_red, 'LineWidth', 3,'LineStyle','-');
    hold on
end
% for j = 1:length(find_pv_flux)
%     xline(time_string(find_pv_flux(j)), 'Color',aruba_yellow, 'LineWidth', 2,'
    LineStyle','-','Alpha',0.6);
%     hold on
%     xline(time_string(find_pv_flux_before(j)), 'Color',aruba_yellow, 'LineWidth
    ', 2,'LineStyle','-','Alpha',0.6);
%     hold on
% end
ylabel('Voltage (V)')
ylim([220 240])
if length(find_indexes_pos_MA) > 0
    if length(find_indexes_neg_MA) > 0
        legend(ph([1,2,3,4,length(find_indexes_neg_MA)+length(find_indexes_pos_MA
            )+3]),'Measured PV Power','Measured BESS Power','Measured Voltage at
            PCC','Measured Visible Flicker Moment','Measured Annoying Flicker
            Moment','Location', 'Best');
    else
        legend(ph([1,2,3,4]),'Measured PV Power','Measured BESS Power','Measured
            Voltage at PCC','Measured Visible Flicker Moment','Location', 'Best');
    end
else
    if length(find_indexes_neg_MA) > 0
        legend(ph([1,2,3,4]),'Measured PV Power','Measured BESS Power','Measured
            Voltage at PCC','Measured Annoying Flicker Moment','Location', 'Best')
            ;
    else
        legend(ph([1,2,3]),'Measured PV Power','Measured BESS Power','Measured
            Voltage at PCC','Location', 'Best');
    end
end
grid on
ax = gca;
ax.GridAlpha = 0.4;
ax.GridLineStyle = "-";
ax.XColor = [0 0 0];
ax.GridColor = [0 0 0];
% title({string(first_line);setup;string(control_line);string(second_line)})
xlabel('Time')
%% Plot Battery, PV POWER and AC voltage measurements
find_pv_flux = find(abs(ramprate_PV) > 300);
find_pv_flux_before = find_pv_flux + 1;
first_line_cmp = 'Battery power, PV power and PCC power measurements';
if string(configuration{2,2}) == "169.254.1.2"
    setup = "AC connected ESS experimental setup";
elseif string(configuration{2,2}) == "192.168.11.3"
    setup = "DC connected ESS experimental setup";
end
first_line = compose(first_line_cmp);
control_line_cmp = 'Control Mode : %s';
control_line = compose(control_line_cmp,string(data.("Control Mode")(1)));
```

```
second_line_cmp = 'Testing time : %s - %s';
second_line = compose(second_line_cmp,time_string(1),time_string(end));
filename_cmp = 'Bat_power_Measured_voltage_%s_%s_%s_%s_%s_%s_%d';
filename = compose(filename_cmp,string(data.("Control Mode")(1)),string(year(
    time_string(1))),string(month(time_string(1))),string(day(time_string(1))),
    string(hour(time_string(1))),string(minute(time_string(1))),floor(second(
    time_string(1))))
fig2 = figure;
fig2.Position = [50 50 800 450];
set(fig2,'defaultAxesColorOrder',[[0 0 0]; [0 0 0]]);
% yyaxis left
% ylabel('Power (W)')
ph = gobjects(1);
ph(1) = plot(data.Time(1:end),data.("PV Output Power (W)")(1:end),'LineWidth',2,'
    Color',aruba_blue);
hold on
ph(2) = plot(data.Time,data.("BESS Measured Power"),'LineWidth',2,'Color',
    aruba_green);
hold on
ylabel('Power (W)')
ph(3) = plot(data.Time,data.("PCC Power (W)"),'LineWidth',2,'Color',stedin_grey);
hold on
for j = 1:length(find_indexes_pos_MA)
    ph(j+3) = plot(time_string([find_indexes_pos_MA(j), actual_indexes_pos_MA(j)
        ]),data.("PCC Power (W)")([find_indexes_pos_MA(j), actual_indexes_pos_MA(j
        )]),  'Marker', '.', 'MarkerSize', 20,'Color',royal_orange, 'LineWidth',
        2,'LineStyle','-');
    hold on
end
for j = 1:length(find_indexes_neg_MA)
    ph(j+length(find_indexes_pos_MA)+3) = plot(time_string([find_indexes_neg_MA(j
        ), actual_indexes_neg_MA(j)]),data.("PCC Power (W)")([find_indexes_neg_MA(
        j), actual_indexes_neg_MA(j)]), 'Marker', '.', 'MarkerSize', 25, 'Color',
        aruba_red, 'LineWidth', 3,'LineStyle','-');
    hold on
end
for j = 1:length(find_pv_flux)
    xline(time_string(find_pv_flux(j)), 'Color',aruba_yellow, 'LineWidth', 2,'
        LineStyle','-','Alpha',0.6);
    hold on
    xline(time_string(find_pv_flux_before(j)), 'Color',aruba_yellow, 'LineWidth',
         2,'LineStyle','-','Alpha',0.6);
    hold on
end
if length(find_indexes_pos_MA) > 0
    if length(find_indexes_neg_MA) > 0
        legend(ph([1,2,3,4,length(find_indexes_neg_MA)+length(find_indexes_pos_MA
            )+3]),'Measured PV Power','Measured BESS Power','Measured Power at PCC
            ','Measured Visible Flicker Moment','Measured Annoying Flicker Moment'
            ,'Location', 'Best');
    else
        legend(ph([1,2,3,4]),'Measured PV Power','Measured BESS Power','Measured
            Power at PCC','Measured Visible Flicker Moment','Location', 'Best');
    end
else
    if length(find_indexes_neg_MA) > 0
        legend(ph([1,2,3,4]),'Measured PV Power','Measured BESS Power','Measured
            Power at PCC','Measured Annoying Flicker Moment','Location', 'Best');
    else
        legend(ph([1,2,3]),'Measured PV Power','Measured BESS Power','Measured
            Power at PCC','Location', 'Best');
```

```
    end
end
grid on
ax = gca;
ax.GridAlpha = 0.4;
ax.GridLineStyle = "-";
ax.XColor = [0 0 0];
ax.GridColor = [0 0 0];
title({string(first_line);setup;string(control_line);string(second_line)})
xlabel('Time')
%% Plot Battery, PV POWER and AC voltage measurements
find_pv_flux = find(abs(ramprate_PV) > 300);
find_pv_flux_before = find_pv_flux + 1;
first_line_cmp = 'Battery power, PV power and PCC power measurements';
if string(configuration{2,2}) == "169.254.1.2"
    setup = "AC connected ESS experimental setup";
elseif string(configuration{2,2}) == "192.168.11.3"
    setup = "DC connected ESS experimental setup";
end
first_line = compose(first_line_cmp);
control_line_cmp = 'Control Mode : %s';
control_line = compose(control_line_cmp,string(data.("Control Mode")(1)));
second_line_cmp = 'Testing time : %s - %s';
second_line = compose(second_line_cmp,time_string(1),time_string(end));
filename_cmp = 'Bat_power_Measured_voltage_%s_%s_%s_%s_%s_%s_%d';
filename = compose(filename_cmp,string(data.("Control Mode")(1)),string(year(
   time_string(1))),string(month(time_string(1))),string(day(time_string(1))),
   string(hour(time_string(1))),string(minute(time_string(1))),floor(second(
   time_string(1))))
fig2 = figure;
fig2.Position = [50 50 800 450];
set(fig2,'defaultAxesColorOrder',[[0 0 0]; [0 0 0]]);
% yyaxis left
% ylabel('Power (W)')
ph = gobjects(1);
ph(1) = plot(data.Time(1:end),data.("PV Output Power (W)")(1:end),'LineWidth',2,'
   Color',aruba_blue);
hold on
ph(2) = plot(data.Time,data.("Battery Power DC (W)"),'LineWidth',2,'Color',
   aruba_green);
hold on
ylabel('Power (W)')
if string(configuration{2,2}) == "169.254.1.2"
    power_setpoint = data.("ESS Setpoint Fourth Layer")(2:end);
    power_setpoint(length(data.("ESS Setpoint Fourth Layer"))) = 0;
    ph(3) = plot(time_string, power_setpoint,"LineWidth",2,'Color',stedin_grey);
elseif string(configuration{2,2}) == "192.168.11.3"
    ph(3) = plot(data.Time,data.("ESS Setpoint Fourth Layer")+data.("PV Output
       Power (W)")-0.1667*data.("PV Output Power (W)"),'LineWidth',2,'Color',
       stedin_grey);
end
hold on
for j = 1:length(find_indexes_pos_MA)
    ph(j+3) = plot(time_string([find_indexes_pos_MA(j), actual_indexes_pos_MA(j)
       ]),data.("PCC Power (W)")([find_indexes_pos_MA(j), actual_indexes_pos_MA(j
       )]),  'Marker', '.', 'MarkerSize', 20,'Color',royal_orange, 'LineWidth',
       2,'LineStyle','-');
    hold on
end
for j = 1:length(find_indexes_neg_MA)
    ph(j+length(find_indexes_pos_MA)+3) = plot(time_string([find_indexes_neg_MA(j
```

```
        ), actual_indexes_neg_MA(j)]),data.("PCC Power (W)")([find_indexes_neg_MA(
        j), actual_indexes_neg_MA(j)]), 'Marker', '.', 'MarkerSize', 25, 'Color',
        aruba_red, 'LineWidth', 3,'LineStyle','-');
    hold on
end
for j = 1:length(find_pv_flux)
    xline(time_string(find_pv_flux(j)), 'Color',aruba_yellow, 'LineWidth', 2,'
       LineStyle','-','Alpha',0.6);
    hold on
    xline(time_string(find_pv_flux_before(j)), 'Color',aruba_yellow, 'LineWidth',
        2,'LineStyle','-','Alpha',0.6);
    hold on
end
if length(find_indexes_pos_MA) > 0
    if length(find_indexes_neg_MA) > 0
        legend(ph([1,2,3,4,length(find_indexes_neg_MA)+length(find_indexes_pos_MA
           )+3]),'Measured PV Power','Measured BESS Power','Power setpoint sent',
           'Measured Visible Flicker Moment','Measured Annoying Flicker Moment','
           Location', 'Best');
    else
        legend(ph([1,2,3,4]),'Measured PV Power','Measured BESS Power','Power
           setpoint sent','Measured Visible Flicker Moment','Location', 'Best');
    end
else
    if length(find_indexes_neg_MA) > 0
        legend(ph([1,2,3,4]),'Measured PV Power','Measured BESS Power','Power
           setpoint sent','Measured Annoying Flicker Moment','Location', 'Best');
    else
        legend(ph([1,2,3]),'Measured PV Power','Measured BESS Power','Power
           setpoint sent','Location', 'Best');
    end
end
grid on
ax = gca;
ax.GridAlpha = 0.4;
ax.GridLineStyle = "-";
ax.XColor = [0 0 0];
ax.GridColor = [0 0 0];
title({string(first_line);setup;string(control_line);string(second_line)})
xlabel('Time')
%%
indexes_pv_operation = [];
indexes_no_operation = [];
indexes_pv_operation = find(hour(time_string(2:end)) >= 5 & hour(time_string(2:
   end)) <= 22);
indexes_no_operation = find(hour(time_string(2:end)) < 5 | hour(time_string(2:end
   )) > 22);

indexes_after_operation_logical = [];
indexes_after_no_operation_logical = [];
annoying_indexes_after_operation_logical = [];
annoying_indexes_after_no_operation_logical = [];

indexes_after_operation_logical = false(length(
   difference_voltage_percentage_control),1);
indexes_after_no_operation_logical = false(length(
   difference_voltage_percentage_control),1);
annoying_indexes_after_operation_logical = false(length(
   difference_voltage_percentage_control),1);
annoying_indexes_after_no_operation_logical = false(length(
   difference_voltage_percentage_control),1);
```

```
indexes_after_operation_logical(indexes_no_operation) = logical(1);
indexes_after_no_operation_logical(indexes_pv_operation) = logical(1);
annoying_indexes_after_operation_logical(indexes_no_operation) = logical(1);
annoying_indexes_after_no_operation_logical(indexes_pv_operation) = logical(1);

indexes_after_operation = [];
indexes_after_no_operation = [];
annoying_indexes_after_operation = [];
annoying_indexes_after_no_operation = [];

indexes_after_operation = or(indexes_after_operation_logical,idx_after');
indexes_after_no_operation = or(indexes_after_no_operation_logical,idx_after');
annoying_indexes_after_operation = or(annoying_indexes_after_operation_logical,
    ida_after');
annoying_indexes_after_no_operation = or(
    annoying_indexes_after_no_operation_logical,ida_after');

if string(configuration{2,2}) == "169.254.1.2"
     setup = "AC connected ESS experimental setup";
     power_range = data.("Sent Set Power (W)");
elseif string(configuration{2,2}) == "192.168.11.3"
     power_range = data.("Sent Set Power (W)")-data.("PV Output Power (W)")
         -(-0.1667*data.("PV Output Power (W)"));
     setup = "DC connected ESS experimental setup";
end
figvoltage_flicker = figure;
set(figvoltage_flicker,'defaultAxesColorOrder',[[0 0 0]; [0 0 0]]);
semilogx(time_dips_seconds,visible_flicker_curve,'LineWidth',3,'Color',
    royal_orange);
hold on
semilogx(time_dips_seconds,annoying_flicker_curve,'LineWidth',3,'Color',
    aruba_red);
set(gca, 'XDir','reverse');
grid on
hold on
scatter(difference_samplerate_SOC,abs(difference_voltage_percentage_control),'
    MarkerFaceColor',aruba_blue,'MarkerEdgeColor',aruba_blue-0.2, 'LineWidth',0.5)
    ;
hold on
scatter(difference_samplerate_SOC(~indexes_after_operation),abs(
    difference_voltage_percentage_control(~indexes_after_operation)),'
    MarkerFaceColor',royal_orange,'MarkerEdgeColor',aruba_green, 'LineWidth',0.5);
hold on
scatter(difference_samplerate_SOC(~indexes_after_no_operation),abs(
    difference_voltage_percentage_control(~indexes_after_no_operation)),60,'
    MarkerFaceColor',royal_orange,'MarkerEdgeColor',stedin_grey, 'LineWidth',2,'
    Marker','square');
hold on
scatter(difference_samplerate_SOC(~annoying_indexes_after_operation),abs(
    difference_voltage_percentage_control(~annoying_indexes_after_operation)),'
    MarkerFaceColor',aruba_red,'MarkerEdgeColor',aruba_green, 'LineWidth',0.5);
hold on
scatter(difference_samplerate_SOC(~annoying_indexes_after_no_operation),abs(
    difference_voltage_percentage_control(~annoying_indexes_after_no_operation))
    ,60,'MarkerFaceColor',aruba_red,'MarkerEdgeColor',stedin_grey, 'LineWidth',2,'
    Marker','square');
hold on
% scatter(difference_samplerate_SOC(632),abs(
    difference_voltage_percentage_control(632)),60,'MarkerFaceColor',aruba_blue,'
    MarkerEdgeColor',[0 0 0], 'LineWidth',2,'Marker','square');
```

```
xlim([0.1 10])
ylim([0 3])
ylabel('Voltage dip (%)')
xlabel('Time between dips')
ax = gca;
ax.GridAlpha = 0.4;
ax.GridLineStyle = "--";
ax.XColor = [0 0 0];
ax.YColor = [0 0 0];
ax.GridColor = [0 0 0];
first_line_cmp = 'Voltage flicker curve, %s';
first_line = compose(first_line_cmp,setup)
third_line_cmp = '%d visible flicker(s) and %d annoying flicker(s) within the
    operating window';
third_line = compose(third_line_cmp,length(difference_samplerate_SOC(~
    indexes_after_operation))-length(difference_samplerate_SOC(~
    annoying_indexes_after_operation)),length(difference_samplerate_SOC(~
    annoying_indexes_after_operation)));
fourth_line_cmp = '%d visible flicker(s) and %d annoying flicker(s) outside the
    operating window';
fourth_line = compose(fourth_line_cmp,length(difference_samplerate_SOC(~
    indexes_after_no_operation))-length(difference_samplerate_SOC(~
    annoying_indexes_after_no_operation)),length(difference_samplerate_SOC(~
    annoying_indexes_after_no_operation)));
title({string(first_line);string(control_line);string(second_line);string(
    third_line);string(fourth_line)})
legend('Visible flicker threshold curve','Annoying flicker threshold curve','Non-
    visible flicker','Visible flicker within the operating window','Visible
    flicker outside of the operating window','Annoying flicker within the
    operating window', 'Annoying flicker outside of the operating window','
    Location', 'Best')
filename_cmp = 'Voltage_flicker_%s_%s_%s_%s_%s_%s_%d';
filename = compose(filename_cmp,string(data.("Control Mode")(1)),string(year(
    time_string(1))),string(month(time_string(1))),string(day(time_string(1))),
    string(hour(time_string(1))),string(minute(time_string(1))),floor(second(
    time_string(1))))
%%
first_line_cmp = 'Set powerpoint(s) vs Actual set powerpoint vs Measured battery
    power';
first_line = compose(first_line_cmp);
control_line_cmp = 'Control Mode : %s';
control_line = compose(control_line_cmp,string(data.("Control Mode")(1)));
second_line_cmp = 'Testing time : %s - %s';
second_line = compose(second_line_cmp,string(time_string(1)),string(time_string(
    end)));
third_line_cmp  = ['SoC during test : %.4f%% - %.4f%%'];
third_line = compose(third_line_cmp,data.("BESS SOC (%)")(1),data.("BESS SOC (%)
    ")(end));
filename_cmp = 'PCC_power_Measured_voltage_%s_%s_%s_%s_%s_%s_%d';
filename = compose(filename_cmp,string(data.("Control Mode")(1)),string(year(
    time_string(1))),string(month(time_string(1))),string(day(time_string(1))),
    string(hour(time_string(1))),string(minute(time_string(1))),floor(second(
    time_string(1))))
fig4 = figure;
set(fig4,'defaultAxesColorOrder',[0 0 0]);
plot(time_string, data.("ESS Setpoint One Layer"),"LineWidth",2,'Color',
    aruba_blue);
hold on
plot(time_string, data.("ESS Setpoint Two Layer"),"LineWidth",2,'Color',
    royal_orange);
hold on
```

```
plot(time_string, data.("ESS Setpoint Three Layer"),"LineWidth",2,'Color',
    aruba_yellow);
hold on
plot(time_string, data.("ESS Setpoint Fourth Layer"),"LineWidth",2,'Color',
    aruba_red);
% hold on
% plot(time_string, data.("Actual Set Power (W)")-data.("PV Output Power (W)")
    -(-0.1667)*data.("PV Output Power (W)"),"LineWidth",2,'Color',aruba_green);
hold on
first_line_cmp = 'Battery power, PV power and PCC voltage measurements from %s';
if string(configuration{2,2}) == "169.254.1.2"
    plot(time_string, data.("Battery Power DC (W)"),"LineWidth",2,'Color',
        stedin_grey);
elseif string(configuration{2,2}) == "192.168.11.3"
    plot(time_string, data.("BESS Measured Power"),"LineWidth",2,'Color',
        stedin_grey);
end
hold on
ylabel('Power (W)')
yyaxis right
ylabel('SoC (%)')
xlabel('Time')
hold on
plot(time_string, data.("BESS SOC (%)"),"LineWidth",2,'Color',aruba_green)
ax = gca;
ax.GridAlpha = 0.4;
ax.GridLineStyle = "--";
ax.XColor = [0 0 0];
ax.YColor = [0 0 0];
ax.GridColor = [0 0 0];
title({string(first_line);string(control_line);string(second_line);string(
    third_line)})
legend('ESS Setpoint One Layer Control','ESS Setpoint Two Layer Control','ESS
    Setpoint Three Layer Control', 'ESS Setpoint Fourth Layer Control','BESS
    Measured Power','BESS SoC')
grid on
%%
soc_range_simpleMA = [];
power_range = [];
% power_range = data.("Actual Set Power (W)")(eight_aug)-data.("PV Output Power (
    W)")(eight_aug);
if string(configuration{2,2}) == "169.254.1.2"
    setup = "AC connected ESS experimental setup";
    power_range = data.("Sent Set Power (W)");
elseif string(configuration{2,2}) == "192.168.11.3"
    power_range = data.("Sent Set Power (W)")-data.("PV Output Power (W)")
        -(-0.1667*data.("PV Output Power (W)"));
    setup = "DC connected ESS experimental setup";
end
for i = 1:length(power_range)
    if i == 1
        soc_range_simpleMA(i) = 0;
    else
        energy_used = (power_range(i) + power_range(i-1))*2*0.5/3600;
        soc_range_simpleMA(i) = soc_range_simpleMA(i-1) - energy_used;
    end
end
first_line_cmp = 'Measured energy usage and SoC from %s';
first_line = compose(first_line_cmp,string(setup));
control_line_cmp = 'Control Mode : %s';
control_line = compose(control_line_cmp,string(data.("Control Mode")(1)));
```

```
second_line_cmp = 'Testing time : %s - %s';
second_line = compose(second_line_cmp,string(time_string(1)),string(time_string(
    end)));
third_line_cmp  = ['SoC range during test : %.4f%% - %.4f%%'];
third_line = compose(third_line_cmp,max(data.("BESS SOC (%)")),min(data.("BESS
    SOC (%)")));
fourth_line_cmp = 'Energy usage : %.2f Wh with power to energy ratio : %.2f';
energy_usage = max(data.("BESS Energy (Wh)")) - min(data.("BESS Energy (Wh)"));
max_power = max(abs(data.("BESS Measured Power")));
power_energy_ratio = max_power/energy_usage;
% fourth_line = compose(fourth_line_cmp,energy_usage,power_energy_ratio);
filename_cmp = 'PCC_power_Measured_voltage_%s_%s_%s_%s_%s_%s_%d';
filename = compose(filename_cmp,string(data.("Control Mode")(1)),string(year(
    time_string(1))),string(month(time_string(1))),string(day(time_string(1))),
    string(hour(time_string(1))),string(minute(time_string(1))),floor(second(
    time_string(1))))
fig4 = figure;
fig4.Position = [50 50 600 450];
set(fig4,'defaultAxesColorOrder',[0 0 0]);
ylabel('SoC (%)')
xlabel('Time')
hold on
plot(time_string, data.("BESS SOC (%)"),"LineWidth",2,'Color',aruba_green)
find_less_energy = find(time_string > '23-Aug-2023 12:00:00.000' & time_string <
    '23-Aug-2023 20:00:00.000');
energy_window = data.("BESS Energy (Wh)")(find_less_energy); %- data.("BESS
    Energy (Wh)")(find_less_energy(1));
soc_new = data.("BESS SOC (%)")(find_less_energy);
new_energy_usage = 0.5*11040 - min(energy_window);
new_power_to_energy_ratio = max_power/new_energy_usage;
% hold on
% xline(time_string(find_less_energy(1)),'Color',aruba_red,'LineWidth',2)
% hold on
% xline(time_string(find_less_energy(end)),'Color',aruba_red,'LineWidth',2)
% hold on
% plot(time_string, soc_range_simpleMA,"LineWidth",2,'Color',stedin_grey)
ax = gca;
ax.GridAlpha = 0.4;
ax.GridLineStyle = "--";
ax.XColor = [0 0 0];
ax.YColor = [0 0 0];
ax.GridColor = [0 0 0];
% fourth_line = compose(fourth_line_cmp,new_energy_usage,
    new_power_to_energy_ratio);
% third_line = compose(third_line_cmp,max(soc_new),min(soc_new));
fourth_line = compose(fourth_line_cmp,energy_usage,power_energy_ratio);
title({string(first_line);string(control_line);string(second_line);string(
    third_line);string(fourth_line)})
% legend('Calculated BESS SoC','Location','Best')
grid on
%%
find_less_energy = find(hour(time_string) > 11 & hour(time_string) < 22);
energy_window = data.("BESS Energy (Wh)")(find_less_energy) - data.("BESS Energy
    (Wh)")(find_less_energy(1));
energy_usage = 0 - min(energy_window);
new_power_to_energy_ratio = max_power/energy_usage;
%%
saveas(fig1,string(filename),'tiff')
%%
saveas(fig2,string(filename),'tiff')
%%
```

```
% first_line_cmp = 'Battery, PCC and PV power measurements from %s';
% if string(configuration{2,2}) == "169.254.1.2"
%     setup = "ESP lab AC test setup";
% elseif string(configuration{2,2}) == "192.168.11.3"
%     setup = "E-bike charging station";
% end
% first_line = compose(first_line_cmp,setup);
% control_line_cmp = 'Control Mode : %s';
% control_line = compose(control_line_cmp,string(data.("Control Mode")(1)));
% second_line_cmp = 'Testing time : %s - %s';
% second_line = compose(second_line_cmp,time_string(1),time_string(end));
% filename_cmp = 'Bat_power_PV_power_%s_%s_%s_%s_%s_%s_%d';
% ph = gobjects(1);
% filename = compose(filename_cmp,string(data.("Control Mode")(1)),string(year(
    time_string(1))),string(month(time_string(1))),string(day(time_string(1))),
    string(hour(time_string(1))),string(minute(time_string(1))),floor(second(
    time_string(1))))
% fig3 = figure;
% ph(1) = plot(data.Time(1:end),data.("PCC Power (W)")(1:end),'LineWidth',2,'
    Color',stedin_grey);
% hold on
% ph(2) = plot(data.Time,data.("BESS Measured Power"),'LineWidth',2,'Color',
    aruba_green);
% hold on
% ph(3) = plot(data.Time,data.("PV Output Power (W)"),'LineWidth',2,'Color',
    aruba_blue);
% hold on
% for j = 1:length(find_indexes_pos_MA)
%     ph(j+3) = plot(time_string([find_indexes_pos_MA(j), actual_indexes_pos_MA(j
    )]),data.("PCC Power (W)")([find_indexes_pos_MA(j), actual_indexes_pos_MA(j)])
    , 'Marker', '.', 'MarkerSize', 20,'Color',royal_orange, 'LineWidth', 2);
%     hold on
% end
% Legend{length(find_indexes_pos_MA)+1,1} = 'Measured Annoying Flicker Moment';
% for j = 1:length(find_indexes_neg_MA)
%     ph(j+length(find_indexes_pos_MA)+3) = plot(time_string([find_indexes_neg_MA
    (j), actual_indexes_neg_MA(j)]),data.("PCC Power (W)")([find_indexes_neg_MA(j)
    , actual_indexes_neg_MA(j)]), 'Marker', '.', 'MarkerSize', 25, 'Color',
    aruba_red, 'LineWidth', 3);
%     hold on
% end
% legend(ph([1,2,3,4,length(find_indexes_pos_MA)+4]),'Measured PCC Power','
    Measured BESS Power','PV Output Power','Measured Visible Flicker Moment','
    Measured Annoying Flicker Moment','Location', 'Best');
% ylabel('Power (W)')
% grid on
% title({string(first_line);string(control_line);string(second_line)})
% xlabel('Time')
% % ylim([-100 2000])
%%
% first_line_cmp = 'PCC power measurements from %s';
% if string(configuration{2,2}) == "169.254.1.2"
%     setup = "ESP lab AC test setup";
% elseif string(configuration{2,2}) == "192.168.11.3"
%     setup = "E-bike charging station";
% end
% first_line = compose(first_line_cmp,setup);
% control_line_cmp = 'Control Mode : %s';
% control_line = compose(control_line_cmp,string(data.("Control Mode")(1)));
% second_line_cmp = 'Testing time : %s - %s';
% second_line = compose(second_line_cmp,time_string(1),time_string(end));
```

```
% filename_cmp = 'Bat_power_PV_power_%s_%s_%s_%s_%s_%s_%d';
% filename = compose(filename_cmp,string(data.("Control Mode")(1)),string(year(
    time_string(1))),string(month(time_string(1))),string(day(time_string(1))),
    string(hour(time_string(1))),string(minute(time_string(1))),floor(second(
    time_string(1))));
% fig3 = figure;
% ph(1) = plot(data.Time(1:end),data.("PCC Power (W)")(1:end),'LineWidth',2,'
    Color',stedin_grey);
% hold on
% for j = 1:length(find_indexes_pos_MA)
%     ph(j+1) = plot(time_string([find_indexes_pos_MA(j), actual_indexes_pos_MA(j
    )]),data.("PCC Power (W)")([find_indexes_pos_MA(j), actual_indexes_pos_MA(j)])
    , 'Marker', '.', 'MarkerSize', 20,'Color',royal_orange, 'LineWidth', 2);
%     hold on
% end
% Legend{length(find_indexes_pos_MA)+1,1} = 'Measured Annoying Flicker Moment';
% for j = 1:length(find_indexes_neg_MA)
%     ph(j+length(find_indexes_pos_MA)+1) = plot(time_string([find_indexes_neg_MA
    (j), actual_indexes_neg_MA(j)]),data.("PCC Power (W)")([find_indexes_neg_MA(j)
    , actual_indexes_neg_MA(j)]), 'Marker', '.', 'MarkerSize', 25, 'Color',
    aruba_red, 'LineWidth', 3);
%     hold on
% end
% legend(ph([1,2,length(find_indexes_pos_MA)+2]),'Measured PCC Power','Measured
    Visible Flicker Moment','Measured Annoying Flicker Moment','Location', 'Best')
% ylabel('Power (W)')
% grid on
% title({string(first_line);string(control_line);string(second_line)})
% xlabel('Time')
%%
time_delta = data.Time(2:end);
fig3 = figure;
plot(time_delta,ramprate_PV,'LineWidth',2,'Color',aruba_yellow);
hold on
plot(time_delta,ramprate_PV_power,'LineWidth',2,'Color',stedin_grey)
hold on
stem(time_delta(~idx_after)',delta_pv_p_data(~idx_after),'LineWidth',2,'Color',
    royal_orange)
hold on
stem(time_delta(~ida_after)',delta_pv_p_data(~ida_after),'LineWidth',2,'Color',
    aruba_red)
% hold on
% plot(data.Time(1:end),data.("PCC Power (W)")(1:end),'LineWidth',2)
ylabel('Power (W)')
grid on
% title({string(first_line);string(control_line);string(second_line)})
legend('Ramprate PV power','Ramprate PCC power','Instances of visible flicker','
    Instances of annoying flicker','Location', 'Best')
xlabel('Time')
indexes_delta_pv = find(abs(ramprate_PV) > 800);
%%
saveas(fig3,string(filename),'tiff')
%%
%%
figure;
yyaxis left
plot(time_string(2:end),ramprate_PV_power)
hold on
yyaxis right
plot(time_string(2:end),ramprate_voltage_SOC)
ylim([-2 2])
```

```
%%
figure
plot(time_string,data.("ESS Setpoint Fourth Layer"))
hold on
plot(time_string,data.('BESS Measured Power'))
legend('Setpoint','Measured')
%%
losses = [];
for i = 1:length(data.("BESS Measured Power"))
    losses(i) = (data.("BESS Measured Power")(i)+data.("PV Output Power (W)")(i))
       -data.("PCC Power (W)")(i);
end
%%
figure;
plot(time_string, data.("BESS Measured Power")+data.("PV Output Power (W)"));
% hold on
% plot(time_string, data.("PCC Power (W)"));
hold on
plot(time_string, data.("PCC Power (W)"));
legend('ESS Setpoint Fourth Layer Control + PV Power','PV Power','PCC Power')
grid on
%%
eight_aug = find(time_string(2:end) > '8-Aug-2023 00:00:00.000' & time_string(2:
   end) < '8-Aug-2023 23:00:00.000');
%%
soc_range_simpleMA = [];
power_range = [];
% power_range = data.("Actual Set Power (W)")(eight_aug)-data.("PV Output Power (
   W)")(eight_aug);
power_range = data.("BESS Measured Power");
for i = 1:length(power_range)
    if i == 1
        soc_range_simpleMA(i) = 0;
    else
        energy_used = (power_range(i) + power_range(i-1))*2*0.5/3600;
        soc_range_simpleMA(i) = soc_range_simpleMA(i-1) - energy_used;
    end
end

%%
figure;
plot(time_string, data.("BESS Energy (Wh)")-data.("BESS Energy (Wh)")(1),'Color',
   aruba_blue,'LineWidth',2);
hold on
plot(time_string,soc_range_simpleMA,'Color',stedin_grey,'LineWidth',2)
legend('Measured','Theoretical')
%%
window = [12];
datos = data.("Set Voltage (V)");
for i = 1:length(window)
%     sinus_MA_pv_output(1:window(i),i) = data.("PV Output Power (W)")(1:window(i
   ));
    weighting_multiplier = [];
    weighting_multiplier = linspace(1,window(i),window(i));
    weighting_fraction = [];
    for k = 1:window(i)
        weighting_fraction(k) = cos((1/window(i))*k*pi/2)^2;
    end
    weighting_fraction = flip(weighting_fraction);
    for j = window(i)+1:length(datos)
        sinus_MA_pv_output(j,i) = sum(datos(j-window(i):j-1).*weighting_fraction
```

```
            ')/sum(weighting_fraction);
    end
%     sinus_MA_power_output_day = sinus_MA_pv_output;
end
%%
figure
plot(sinus_MA_pv_output-data.("PV Output Power (W)"))
hold on
plot(data.("ESS Setpoint Fourth Layer"))
%%
figure
plot(sinus_MA_pv_output)
hold on
plot(data.("Set Voltage (V)"))
%%
figure
yyaxis left
plot(data.("PV Output Power (W)"))
hold on
yyaxis right
plot(data.("Measured Voltage (V)"))

%% Plot Battery, PV VOLTAGE and AC voltage measurements
find_pv_flux = find(abs(ramprate_PV) > 300);
find_pv_flux_before = find_pv_flux + 1;
first_line_cmp = 'Battery power, PV power and PCC voltage measurements';
% if string(configuration{2,2}) == "169.254.1.2"
%     setup = "AC connected ESS experimental setup";
% elseif string(configuration{2,2}) == "192.168.11.3"
%     setup = "DC connected ESS experimental setup";
% end
setup = "AC connected ESS experimental setup";
first_line = compose(first_line_cmp);
control_line_cmp = 'Control Mode : %s';
control_line = compose(control_line_cmp,"Reading Values Only");
second_line_cmp = 'Testing time : %s - %s';
second_line = compose(second_line_cmp,time_string(1),time_string(end));
filename_cmp = 'Bat_power_Measured_voltage_%s_%s_%s_%s_%s_%s_%d';
filename = compose(filename_cmp,"Reading Values Only",string(year(time_string(1))
   ),string(month(time_string(1))),string(day(time_string(1))),string(hour(
   time_string(1))),string(minute(time_string(1))),floor(second(time_string(1))))
fig2 = figure;
fig2.Position = [50 50 800 450];
set(fig2,'defaultAxesColorOrder',[[0 0 0]; [0 0 0]]);
time_s = datetime(time_string);
% yyaxis left
% ylabel('Power (W)')
ph = gobjects(1);
ph(1) = plot(time_s,pv_power_meas,'LineWidth',2,'Color',aruba_blue);
hold on
ph(2) = plot(time_s,battery_power,'LineWidth',2,'Color',aruba_green);
hold on
ylabel('Power (W)')
yyaxis right
ph(3) = plot(time_s,voltage_meas,'LineWidth',2,'Color',stedin_grey);
hold on
for j = 1:length(find_indexes_pos_MA)
    ph(j+3) = plot(time_s([find_indexes_pos_MA(j), actual_indexes_pos_MA(j)]),
       voltage_meas([find_indexes_pos_MA(j), actual_indexes_pos_MA(j)]),  'Marker
       ', '.', 'MarkerSize', 20,'Color',royal_orange, 'LineWidth', 2,'LineStyle',
       '-');
```

```
    hold on
end
for j = 1:length(find_indexes_neg_MA)
    ph(j+length(find_indexes_pos_MA)+3) = plot(time_s([find_indexes_neg_MA(j),
        actual_indexes_neg_MA(j)]),voltage_meas([find_indexes_neg_MA(j),
        actual_indexes_neg_MA(j)]), 'Marker', '.', 'MarkerSize', 25, 'Color',
        aruba_red, 'LineWidth', 3,'LineStyle','-');
    hold on
end
% for j = 1:length(find_pv_flux)
%     xline(time_string(find_pv_flux(j)), 'Color',aruba_yellow, 'LineWidth', 2,'
    LineStyle','-','Alpha',0.6);
%     hold on
%     xline(time_string(find_pv_flux_before(j)), 'Color',aruba_yellow, 'LineWidth
    ', 2,'LineStyle','-','Alpha',0.6);
%     hold on
% end
ylabel('Voltage (V)')
ylim([230 240])
if length(find_indexes_pos_MA) > 0
    if length(find_indexes_neg_MA) > 0
        legend(ph([1,2,3,4,length(find_indexes_neg_MA)+length(find_indexes_pos_MA
            )+3]),'Measured PV Power','Measured BESS Power','Measured Voltage at
            PCC','Measured Visible Flicker Moment','Measured Annoying Flicker
            Moment','Location', 'Best');
    else
        legend(ph([1,2,3,4]),'Measured PV Power','Measured BESS Power','Measured
            Voltage at PCC','Measured Visible Flicker Moment','Location', 'Best');
    end
else
    if length(find_indexes_neg_MA) > 0
        legend(ph([1,2,3,4]),'Measured PV Power','Measured BESS Power','Measured
            Voltage at PCC','Measured Annoying Flicker Moment','Location', 'Best')
            ;
    else
        legend(ph([1,2,3]),'Measured PV Power','Measured BESS Power','Measured
            Voltage at PCC','Location', 'Best');
    end
end
grid on
ax = gca;
ax.GridAlpha = 0.4;
ax.GridLineStyle = "-";
ax.XColor = [0 0 0];
ax.GridColor = [0 0 0];
title({string(first_line);setup;string(control_line);string(second_line)})
xlabel('Time')
```