Contents

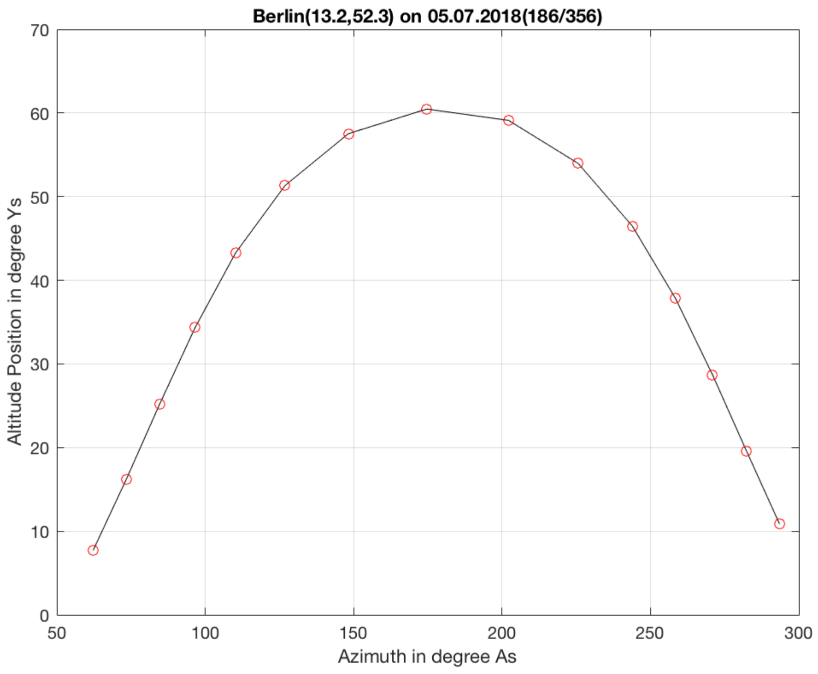
Exercise(1):

```
function [Am,Ys,SAz] = sundata(long,lati,DOY,LT,TZ)
%**************************************************************************
%This function returns the Air mass, Sun elevation angle, Sun azimuth angle given the
%longitude, latitude, day number of year, local time and time zone
%**************************************************************************
% day angle in degree
J =360*DOY/365;
% radian = degree*pi/180
deta = 0.3948-23.2559*cos((J+9.1)*pi/180)-0.3915*cos((2*J+5.4)*pi/180)-
0.176*cos((3*J+26)*pi/180);
% time equation in second
TEQ =
0.0066+7.3525*cos((J+85.9)*pi/180)+9.9359*cos((2*J+108.9)*pi/180)+0.3387*cos((3*J+10
5.2)*pi/180);
% True local time in second
TLT = (LT-TZ)*3600 + 4* long*60+TEQ*60;
w = (12 - TLT/3600)*15;% Hour angle in degree every hour
% Sun Elevation
Ys=asin(cos(w*pi/180)*cos(lati*pi/180)*cos(deta*pi/180)+sin(lati*pi/180)*sin(deta*pi/180);
% Air Mass
Am = 1./sin(Ys);
% '-' for TLT<=12;'+' for TLT>12
logik = zeros(1,length(TLT));% '-' for TLT<=12;'+' for TLT>12
logik = TLT <= 12*3600;
factor = 1-2*logik;
% Sun Azimuth: transform radian to degree with(degree = radian*180/pi)
SAz = 180 + factor.*acos( (sin(Ys)*sin(lati*pi/180)-sin(deta*pi/180)) ./
(cos(Ys)*cos(lati*pi/180)) )*180/pi;
Ys =   Ys*180/pi;
end
```

Then:

```
%************************************************************************
%test function [Am,Ys,SAz] = sundata(long,lati,DOY,LT,TZ)
%berlin's longitude=13.2;latitude=52.3;time zone=2
%05.07.2018 is the 186th day of the year
%************************************************************************
clc;
clear;
[Am,Ys,SAz] = sundata(13.2,52.3,186,[6:20],2)
plot(SAz,Ys,'color','k');
hold on;
plot(SAz,Ys,'ro')
title('Berlin(13.2,52.3) on 05.07.2018(186/356)');
xlabel('Azimuth in degree As');
ylabel('Altitude Position in degree Ys');
grid on;
```

the output relation between Azimuth and elevation is as shown in figure(1).



Figure(1): Azimuth and Altitude of the sun for Berlin on 05.07.2018

Exercise(2):

```
function [I] = PVmod(Volt,Irrad,Temp)
%************************************************************************
%this function determines the current of the described PV module given
%the operating voltage, irradiance and temperature
%Volt is in Volt, Irrad is in w/m^2, Temp is in K
%************************************************************************
%simplify the variable's name
V=Volt;
S=Irrad;
T=Temp;
%**********************Constants according to the datasheet********************
Isc1=5.5;                          % Short circuit current at STC
Voc1=44.8;                         % Open circuit voltage at STC
S1=1000;                           % Standard irradiance
K=1.381e-23;                       % Boltzmann constant
q=1.60e-19;                        % Change of an electron
Vg=1.12;                           % Band gap Voltage of the pn junction
n=72;                              % Number of cells in series
T1=25+273;                         % Temperature of STC in   Kelvin scale
K0=0.065/100;                      % Short circuit current temp. coefficient
%***********************Choose matched Rs,Rsh and y************************
Rs=0.002;                          % Cell series resistance
Rsh=300;                           % Cell shunt resistance
y=1.6;                             % Quality factor
%**************************Model of PV Module***************************
IL=Isc1*(S/S1)*(1+K0*(T-T1));
IL1=Isc1;
Is1=IL1/(exp(q*Voc1/n/(y*K*T1))-1);
Is= Is1*((T/T1)^(3/y))*(exp(((-q*Vg)/(y*K))*(1/T-1/T1)));
```

```
%*********************Iterate to calculate I using Newton's method***************
I=zeros(1,length(V));
i=0;
while i<=10
F=IL-I-Is.*(exp(q.*(V./n+I.*Rs)./(y*K*T))-1)-(V./n+I.*Rs)./Rsh;
% dF=gradient(F);
dF=-1-Is.*Rs*q./(y*K*T).*exp(q.*(V./n+I.*Rs)./(y*K*T))-Rs/Rsh;
I=I-(F)./(dF);
i=i+1;
end
%*******************************************************************
end
```

Then:

```
%*******************************************************************
%                Plotting V-I and V-P characteristics in the different Temperatures
%                            Figure(1) draws the V-I characteristics
%                            Figure(2) draws the V-P characteristics
%*******************************************************************
%Given range of Voltage
V=(0:0.01:50);
%In this case, the Irradiance is fixed
S = 1000;
%=====================Temperature is 0 Celsius degree=====================
T = 0+273;%T is the Temperature
I=PVmod(V,S,T);%Using the PVmod to calculate the output current
P=(V.*I);%Power is equal to Current multiplies Voltage
figure(1);
plot(V,I,'k','LineStyle','-','Linewidth',1.2);
axis([0 50 0 6]);
hold on;grid on;
figure(2);
plot(V,P,'k','LineStyle','-','Linewidth',1.2);
```

```
axis([0 50 0 220]);
hold on;grid on;
%=====================Temperature is 25 Celsius degree====================
figure(1);
T = 25+273;
I=PVmod(V,S,T);
P=(V.*I);
plot(V,I,'k','LineStyle','--','Linewidth',1.2);
figure(2);
plot(V,P,'k','LineStyle','--','Linewidth',1.2);
%=====================Temperature is 50 Celsius degree====================
T = 50+273;
I=PVmod(V,S,T);
P=(V.*I);
figure(1);
plot(V,I,'k','LineStyle','-.','Linewidth',1.2);
figure(2);
plot(V,P,'k','LineStyle','-.','Linewidth',1.2);
%==========================Set the axis for figure========================
figure(1);
xlabel('\fontsize{16}Module Voltage [V]');
ylabel('\fontsize{16}Module Current [A]');
legend('T=0','T=25','T=50');
title('V-I characteristics ','fontsize',14);
figure(2);
xlabel('\fontsize{16}Module Voltage [V]');
ylabel('\fontsize{16}Module Power [W]');
legend('T=0','T=25','T=50');
title('V-P characteristics ','fontsize',14);
```

```
%*************************************************************************
%          Plotting V-I and V-P characteristics in the different Irradiances
%                              Figure(3) draws the V-I characteristics
%                              Figure(4) draws the V-P characteristics
%*************************************************************************
%Given range of Voltage
V=(0:0.01:50);
%In this case, the Temperature is fixed
T = 25+273;%T is the Temperature
%=======================Irradiances is 0.5 kw/m^2=======================
S = 500;
I=PVmod(V,S,T);%Using the PVmod to calculate the output current
P=(V.*I);%Power is equal to Current multiplicates Voltage
figure(3);
plot(V,I,'k','LineStyle','-','Linewidth',1.2);
axis([0 50 0 9]);
hold on;grid on;
figure(4);
plot(V,P,'k','LineStyle','-','Linewidth',1.2);
axis([0 50 0 300]);
hold on;grid on;
%=======================Irradiances is 1 kw/m^2=========================
figure(3);
S = 1000;
I=PVmod(V,S,T);
P=(V.*I);
plot(V,I,'k','LineStyle','--','Linewidth',1.2);
figure(4);
plot(V,P,'k','LineStyle','--','Linewidth',1.2);
%=======================Irradiances is 1.5 kw/m^2=======================
S = 1500;
I=PVmod(V,S,T);
P=(V.*I);
```

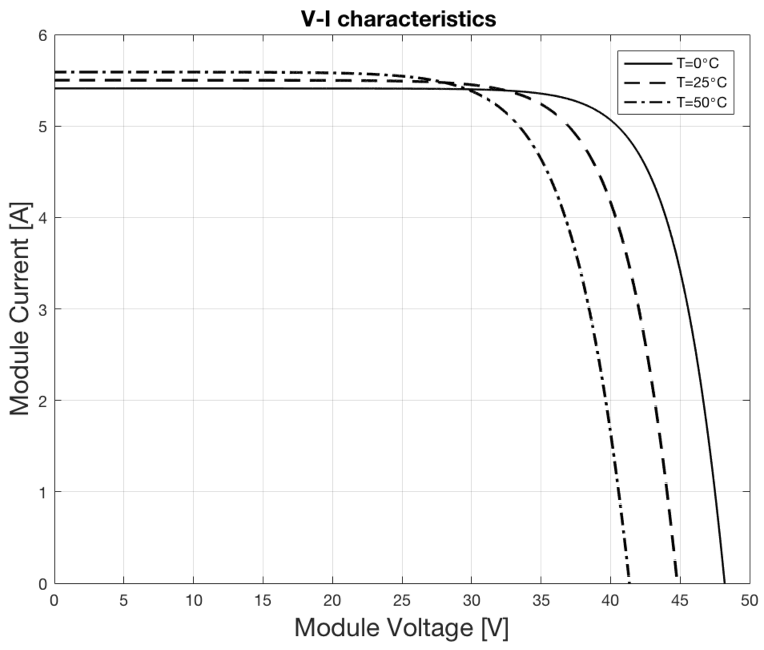
```
figure(3);
plot(V,I,'k','LineStyle','-.','Linewidth',1.2);
figure(4);
plot(V,P,'k','LineStyle','-.','Linewidth',1.2);
%========================Set the axis for figure===========================
figure(3);
xlabel('\fontsize{16}Module Voltage [V]');
ylabel('\fontsize{16}Module Current [A]');
legend('S=0.5 kw/m^2','S=1 kw/m^2','S=1.5 kw/m^2');
title('V-I characteristics ','fontsize',14);
figure(4);
xlabel('\fontsize{16}Module Voltage [V]');
ylabel('\fontsize{16}Module Power [W]');
legend('S=0.5 kw/m^2','S=1 kw/m^2','S=1.5 kw/m^2');
title('V-P characteristics ','fontsize',14);
```

i. The effects of increasing and decreasing the temperature for voltage-current characteristics of the module is as shown in figure(1).

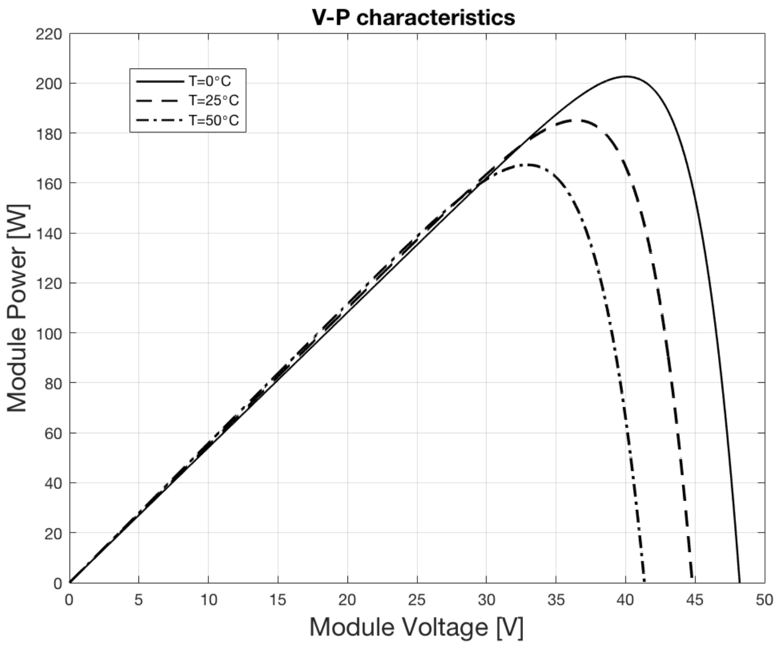
ii. The effects of increasing and decreasing the temperature for voltage-power characteristics of the module is as shown in figure(2).

iii. The effects of increasing and decreasing the irradiance for voltage-current characteristics of the module is as shown in figure(3).

iv. The effects of increasing and decreasing the irradiance for voltage-power characteristics of the module is as shown in figure(4).



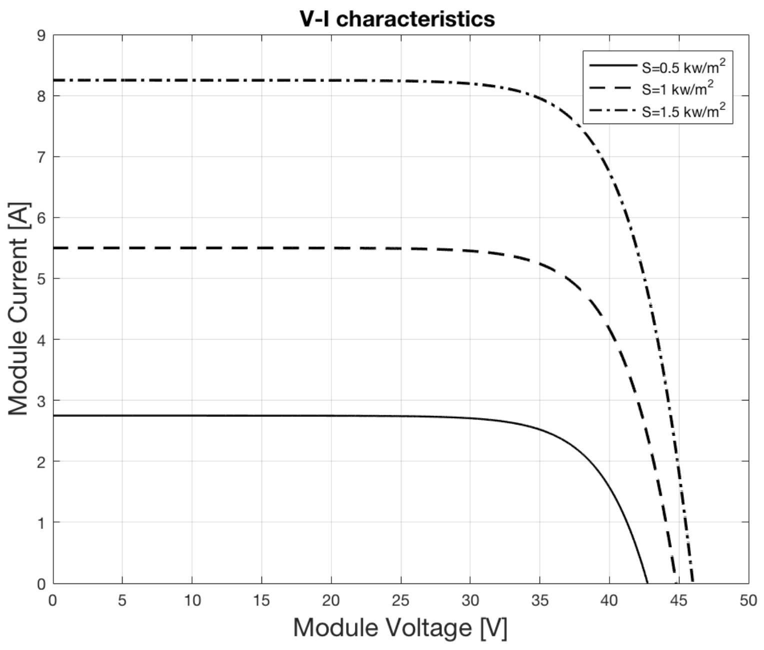
Figure(1):The voltage-current characteristic of PV at different Temperature.



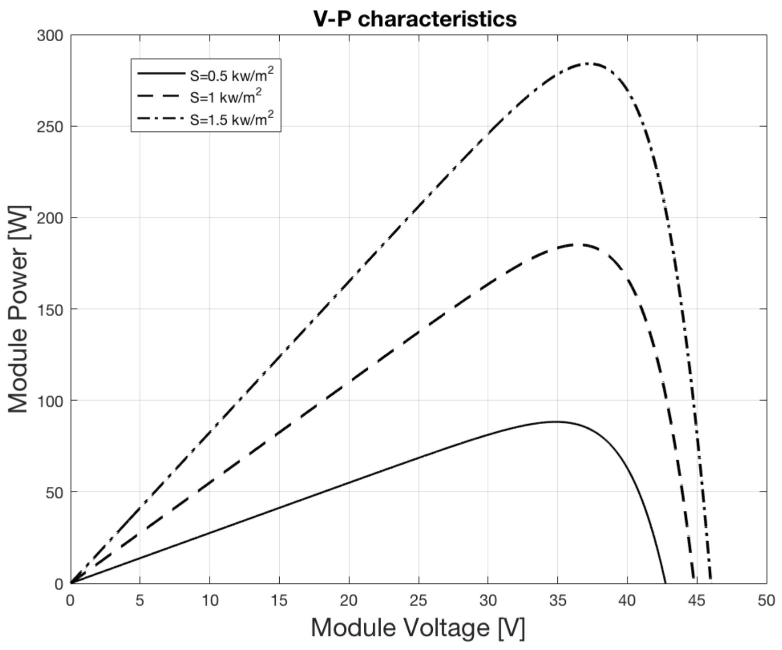
Figure(2): The voltage-power characteristic of PV at different Temperature.

Conclusion:

From the graphs we can see that with the increase of the temperature, the open circuit voltage[Voc] is dropping, the short circuit current[Isc] is slowly increasing, and the power at max power point[Pmpp] is dropping.



Figure(3):The voltage-current characteristic of PV at different Irradiance.



Figure(4):The voltage-power characteristic of PV at different Irradiance.

Conclusion:

With the increase of Irradiance, the current increases significantly and the voltage makes almost no changes, furthermore the Pmpp increases very significantly.

Exercise(3):

```
%*************************************************************************
%                    calculate FF and eta max through given sectional area and STC
%*************************************************************************
%Given STC
V = [0:0.01:50];
Irrad = 1000;
Temp = 25+273;
%using PVmod to calculate current and power
I = PVmod(V,Irrad,Temp);
P=(V.*I);
%Pmax is the max value, maxarg is the max value's index
[Pmax,maxarg] = max(P);
%find Vmpp at Pmax
Vmpp=V(maxarg);
%find Impp at Pmax
Impp=I(maxarg);

%find short circuit current when the voltage is zero
%use V<1e-3 to approach be equal to zero
Vzero = find(V<=1e-3);
Isc = I(Vzero(1));
%find open circuit voltage when the current is zero
%use I<1e-3 to approach be equal to zero
Izero = find(I<=1e-3);
Voc = V(Izero(1));

%L is the Length, W is the width, both are in meter
L=1593/1000;
w=790/1000;
A=L*w; %A is the area and it is in m^2
S=1000; %S is the standard irradiance
```

```
%calculate the fill factor and the efficiency
eff=Vmpp*Impp/(A*S);
FF=Vmpp*Impp/(Voc*Isc);
disp([eff,FF]);
```

Answer:

The efficiency(η_{max}) = 0.1471

The fill factor(FF) = 0.7512

Exercise(4):

```
%*************************************************************************
% This function determines the pure resistive load to be interfaced to the
% module keeping it working at the maximum power point MPP for any
% given irradiance and temperature
%*************************************************************************
function  [Rm] = Rmpp(Irrad,Temp)
%calculate power through the function PVmod
V=(0:0.01:50);
I=PVmod(V,Irrad,Temp);
P=(V.*I);
%Pmax is the max value, maxarg is the max value's index
[Pmax,maxarg] = max(P);
%find Vmpp at Pmax
Vmax=V(maxarg);
%find Impp at Pmax
Imax=I(maxarg);
%return the resistance value using Ohm's law
Rm=Vmax/Imax;
end
```

Then:

```
%test the function Rmpp(Irrad,Temp) given different Temperature and Irradiance
Irrad_list=[200,200,600,600,1000,1000];
Temp_list =[0,25,0,25,0,25]+273;
%the times of experiments
times = length(Irrad_list);
%the list contains all 6 experiments' results
Rm_list = zeros(1,times);
for i = 1:times;
    Rm_list(i) = Rmpp(Irrad_list(i),Temp_list(i));
end
```

The results is shown as in table(1):

	(1)	(2)	(3)	(4)	(5)	(6)
Irradiance(W/m^2)	200	200	600	600	1000	1000
Temperature(^oC)	0	25	0	25	0	25
Resistive Load(Ω)	36.254	32.3163	12.8612	11.6117	7.9136	7.1782

Table(1):Resistive load for given irradiance and temperature

Exercise(5):

```
%*****************************************************************
%This function determines the power loss compared to the case of using ideal MPPT for
% any values of the resistive loads.
%*****************************************************************
function [powerLoss] = PLoss(Irrad,Temp,R)
%calculate Power through PVmod
V=(0:0.01:50);
```

```
I=PVmod(V,Irrad,Temp);
P=(V.*I);
%find maximal Power
Pmax=max(P);

% Get corresponding voltage and current for the given R,
% assume that the intersection point is the point with mininal difference
R_all=V./I;
[Rmin,argmin] = min(abs(R_all-R));
Vmin = V(argmin);
Imin = I(argmin);

%calculate percentage power difference
Pmin = Vmin*Imin;
powerLoss = (Pmax-Pmin)/Pmax;
end
```

Then:

```
%*****************************************************************
% Calculate Power loss at STC [Irrad = 1000w/m^2, Temp = 25'c]
% under different resistive load
%*****************************************************************
Irrad = 1000; Temp = 25+273;
% given the test
R_list = [0,3,7,13,100];
times = length(R_list);
% this list contains all results of 6 experiments under STC
PLoss_list_STC = zeros(1,times);
for i = 1:times;
    PLoss_list_STC(i) = PLoss(Irrad,Temp,R_list(i));
end
```

```
disp(PLoss_list_STC);
%************************************************************************
% Calculate Power loss at NOCT [Irrad = 800w/m^2, Temp = 20'c]
% under different resistive load
%************************************************************************
Irrad = 800;
Temp = 20+273;
% this list contains all results of 6 experiments under NOCT
PLoss_list_NOCT = zeros(1,times);
for i = 1:times;
    PLoss_list_NOCT(i) = PLoss(Irrad,Temp,R_list(i));
end
disp(PLoss_list_NOCT);
```

The results is shown as in table(1):

Load(Ω)	0	3	7	13	100
Power loss(%) - STC	100.00	50.98	0.11	27.63	89.15
Power loss(%) - NOCT	100.00	61.26	11.18	14.29	86.81

Table(1):Power loss for the given resistive loads

Exercise(6):

```
%****************************************************************************
% This function determines the next iterator voltage value for tracking maximum
% power point based on the Hill Climbing Algorithm
%****************************************************************************
function [Vnew] = HClimb( Irrad,Temp,Vold,Vcur,V_step )
%Calculate old and current powers through calling 'PVmod'
%Here V and P are scalar
Iold=PVmod(Vold,Irrad,Temp);
```

```
Pold=Vold.*Iold;
Icur=PVmod(Vcur,Irrad,Temp);
Pcur=Vcur.*Icur;
%*********************************************************************
%                       Hill Climbing(Perturb&Obeserve) Table
%                        Perturbation    PowerChange      Next
%         case 1:       Positive          Positive          Positive
%         case 2:       Positive          Negative          Negative
%         case 3:       Negative          Positive          Negative
%         case 4:       Negative          Negative          Positive
%*********************************************************************
%========================Positive Perturbation========================
if Vcur-Vold>0
    if Pcur-Pold>0%Positive PowerChange
        %case one:
        Vnew=Vcur+V_step;
    else
        %case two:
        Vnew=Vcur-V_step;
    end
%========================Negative Perturbation========================
else
    if Pcur-Pold>0%Positive PowerChange
        %case three:
        Vnew=Vcur-V_step;
    else
        %case four:
        Vnew=Vcur+V_step;
    end
end
end
```

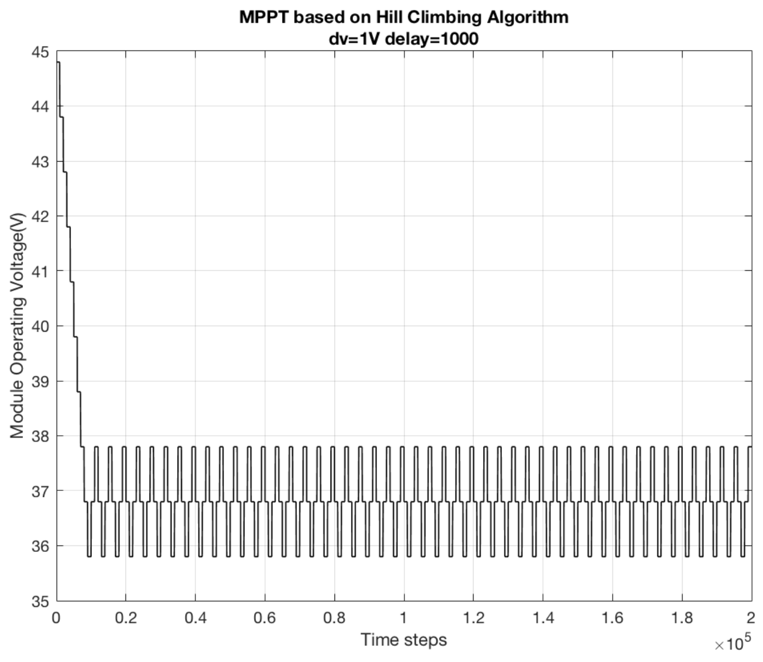
```
%***********************************************************************
% This function can perform maximum power point tracking MPPT through pulse
% width modulation of a buck-boost converter based on the Hill Climbing
% Algorithm assuming a fixed step for voltage change of 1 volt. Converter
% output voltage is kept fixed at 30V.
%***********************************************************************
function [D_delay,Vnew_delay] = PVHC( Irrad,Temp,V_step )
%=========================Define Constants===========================
% fixed converter output voltage: 30V
Vo=30;
% starting Vi from Voc[Open Circuit Voltage]:44.8V
Voc=44.8;
% iterations for Hill Climbing: 200 steps
iterations=200;
% when Vi changes, Duty cycle needs time to settle down,
% so the system settling time "delay": 1000
delay=1000;
%====================================================================
% initialize for the iterator, Vi starts from Voc
Vold = 44.8;
Vcur=Vold-V_step;
% Vi contains all iterative transient states
Vnew = zeros(1,iterations);
Vnew(1)=Vold; Vnew(2)=Vcur;
% Di contains all iterative transient states
D = zeros(1,iterations);
% Pulse width modulation:   buck-boost Vo=D*Vi/(1-D) => D=Vo/(Vi+Vo)
D(1) = Vo/(Vo+Vnew(1));
D(2) = Vo/(Vo+Vnew(2));
% iterator of HClimb
for i=3:iterations
    Vnew(i)=HClimb(Irrad,Temp,Vold,Vcur,V_step);
    Vold=Vcur;
    Vcur=Vnew(i);
    %D=Vo/(Vi+Vo)
    D(i)=Vo/(Vo+Vnew(i));
end
% after every step's changing D need a delay to settle down
% here D_delay means settled Duty cycle
D_delay = zeros(1,iterations*delay);
Vnew_delay = zeros(1,iterations*delay);
% among the every step repeat the value 1000 times
% to simulate the reality condition
 for i=1:iterations
        Vnew_delay((i-1)*1000+1:i*1000)=Vnew(i);
        D_delay((i-1)*1000+1:i*1000)=D(i);
 end
end
```

Then:

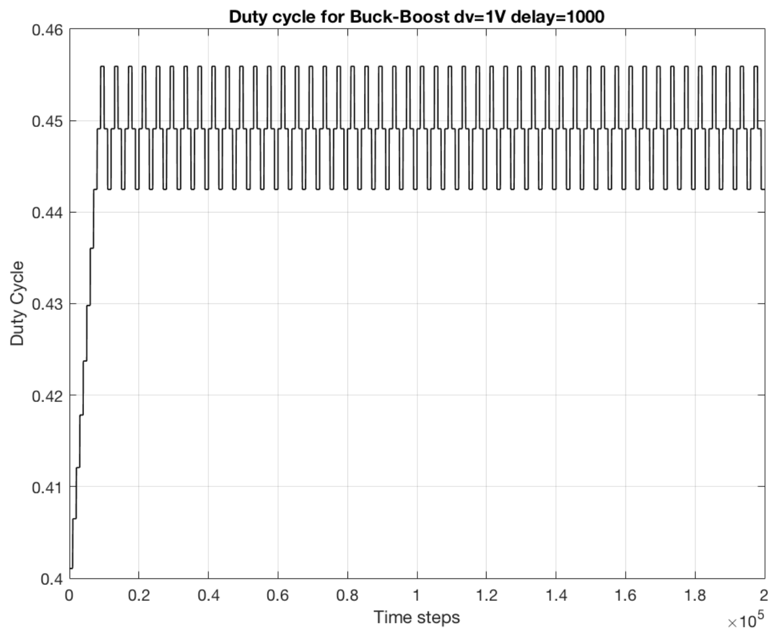
```
% test function PVHC and plot D and Vnew
% step for voltage change: 1V
V_step = 1;
Irrad=1000;
Temp=25+273;
% output the V and D
[D_delay,Vnew_delay] = PVHC( Irrad,Temp,V_step );
% plot the Module Operating Voltage
figure(1);
plot(Vnew_delay,'k','Linewidth',0.8);
xlabel('Time steps')
ylabel('Module Operating Voltage(V)')
title({'MPPT based on Hill Climbing Algorithm','dv=1V delay=1000'});
grid on;
% plot the Duty Cycle
```

```
figure(2);
plot(D_delay,'k','Linewidth',0.8);
xlabel('Time steps')
ylabel('Duty Cycle')
title('Duty cycle for Buck-Boost dv=1V delay=1000');
grid on;
```

The performance of operating voltage is shown as fig.(1), and duty cycle is shown as fig.(2)



Figure(1):operating voltage(V) based on Hill Climbing MPPT

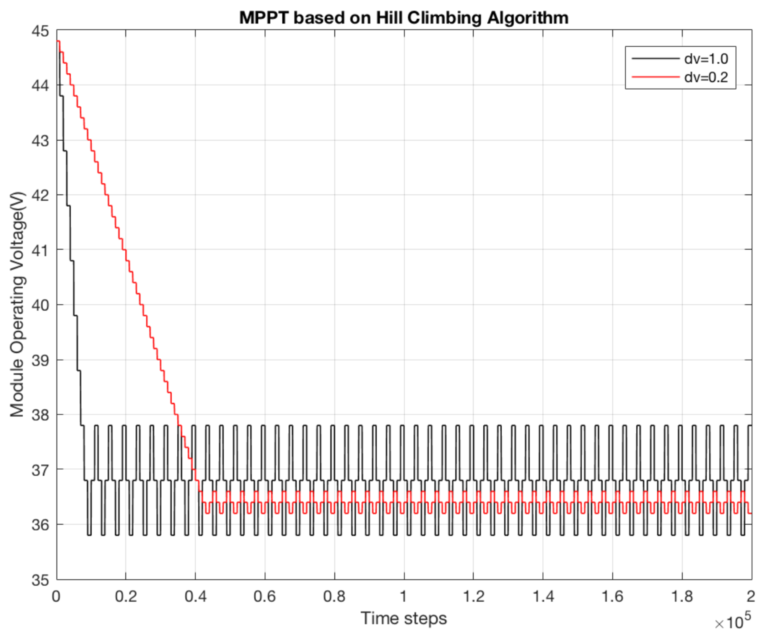


Figure(2):Duty cycle based on Hill Climbing MPPT

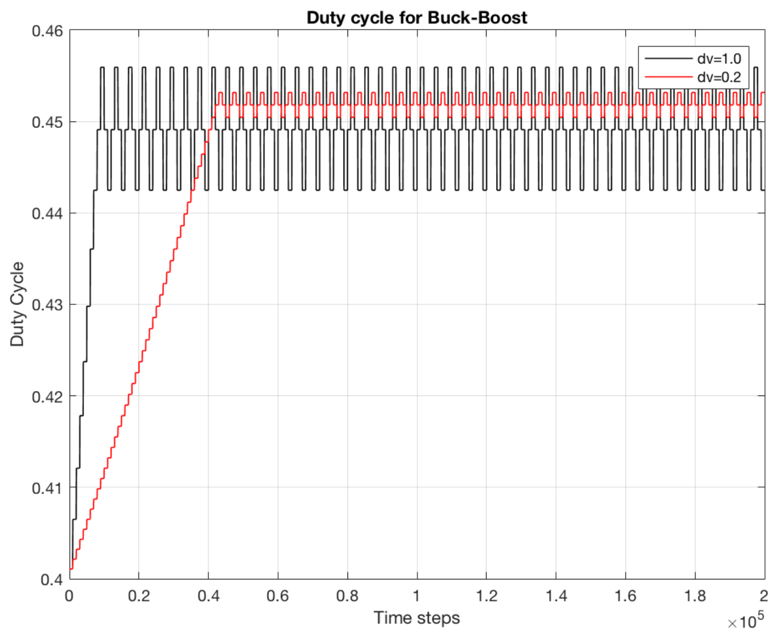
Exercise(7):

```
% test function PVHC with different voltage step and plot D and Vnew
% step for voltage change: 1.0 and 0.2
Irrad=1000;
Temp=25+273;
%*************************step for voltage change: 1.0************************
V_step = 1;
%output the V and D
[D_delay_1,Vnew_delay_1] = PVHC( Irrad,Temp,V_step );
%*************************step for voltage change: 0.2************************
V_step = 0.2;
%output the V and D
[D_delay_2,Vnew_delay_2] = PVHC( Irrad,Temp,V_step );
%*********************plot the Module Operating Voltage**********************
figure(1);
plot(Vnew_delay_1,'k','Linewidth',0.8);
hold on;
plot(Vnew_delay_2,'r','Linewidth',0.8);
xlabel('Time steps')
ylabel('Module Operating Voltage(V)')
title('MPPT based on Hill Climbing Algorithm');
legend('dv=1.0','dv=0.2');
grid on;
%****************************plot the Duty Cycle****************************
figure(2);
plot(D_delay_1,'k','Linewidth',0.8);
hold on;
plot(D_delay_2,'r','Linewidth',0.8);
xlabel('Time steps')
ylabel('Duty Cycle')
title('Duty cycle for Buck-Boost');
legend('dv=1.0','dv=0.2'); grid on;
```

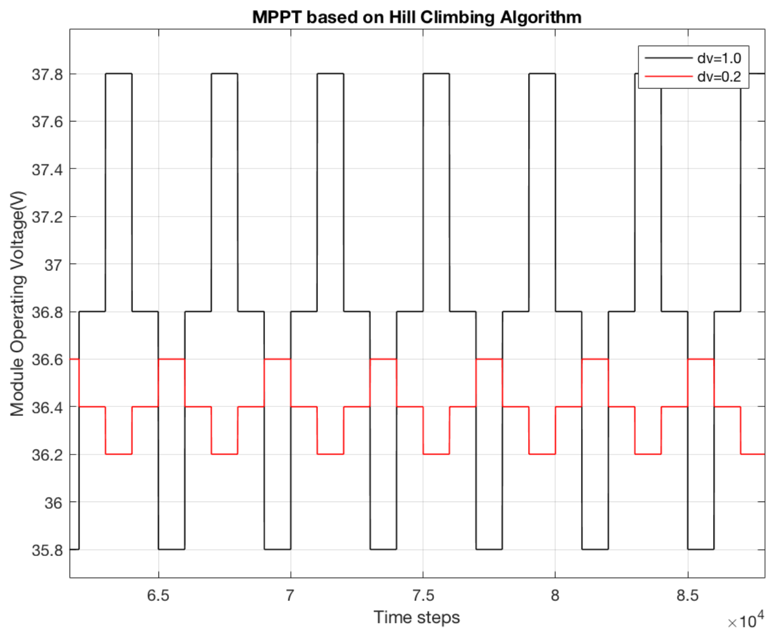
The comparison of performance of operating voltage is shown as figure(1), and duty cycle is shown as figure(2). The comparison of steady state performance of operating voltage is shown as figure(3), and duty cycle is shown as figure(4).



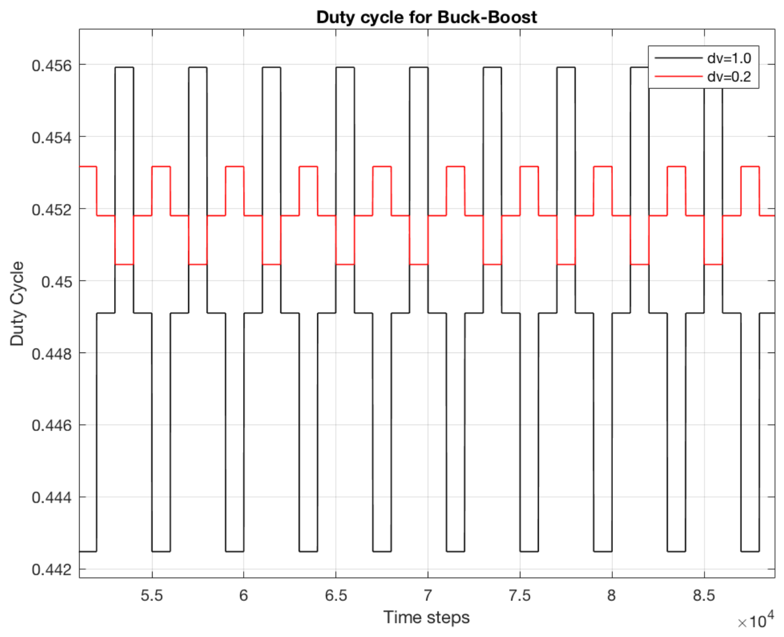
Figure(1): Operating voltage Performance of both Hill Climbing MPPT



Figure(2): Operating duty cycle of both Hill Climbing MPPT



Figure(3):Steady state performance of operating voltage of both Hill Climbing



Figure(4):Steady state performance of duty cycle of both Hill Climbing

Conclusion:

When step is equal to 1.0, the response is quickly stabilized, it takes only about 10 steps, and the voltage is oscillating at 36.8 and the duty cycle is at 0.449.

When step is equal to 0.2, the response is slower stabilized, it takes about 40 steps, and the voltage is oscillating at 36.4 and the duty cycle is at 0.452.

In a word, “the process is repeated periodically until the MPP is reached. The system then oscillates about the MPP. The oscillation can be minimized by reducing the perturbation step size. However, a smaller perturbation size slows down the MPPT.”[1] The MPP will be quickly fund with big perturbation step size, but the accuracy is lower than the former.

Exercise(8):

```
%************************************************************************
% This function can perform maximum power point tracking MPPT
% based on the Adaptive Hill Climbing Algorithm
%************************************************************************\
function [Vnew] = AHClimb( Irrad,Temp,Vold,Vcur,C )
% Calculate old and current powers through calling 'PVmod'
% Here V and P are scalar
Iold=PVmod(Vold,Irrad,Temp);
Pold=Vold.*Iold;
Icur=PVmod(Vcur,Irrad,Temp);
Pcur=Vcur.*Icur;
% difference of Power means Pcur-Pold
dP = Pcur - Pold;
% difference of Voltage means Vcur-Vold
dV = Vcur - Vold;
% step size = C*dP/dV
if dV~=0
    V_step = C*(dP/dV);
else %if dV=0, directly set step=0
    V_step = 0;
end
Vnew=Vcur+V_step;
end
```

```
%**************************************************************************
% This function can perform maximum power point tracking MPPT through pulse
% width modulation of a buck-boost converter based on the Adaptive Hill Climbing
% Algorithm. Converter output voltage is kept fixed at 30V.
%**************************************************************************
function [D_delay,Vnew_delay] = PVAHC( Irrad,Temp,C )
%==========================Define Constants===========================
%fixed converter output voltage: 30V
Vo=30;
%starting Vi from Voc[Open Circuit Voltage]:44.8V
Voc=44.8;
% iterations for adaptive Hill Climbing: 300 steps
iterations=300;
%wenn Vi changes, Duty cycle needs time to settle down,
%so the system settling time "delay": 1000
delay=1000;
%==========================Define Constants===========================
%initialize for the iterator, Vi starts from 0
Vold = 0;
Vcur= C;
%Vi contains all iterative transient states
Vnew = zeros(1,iterations);
Vnew(1)=Vold; Vnew(2)=Vcur;
%Di contains all iterative transient states
D = zeros(1,iterations);
% Pulse width modulation: buck-boost Vo=D*Vi/(1-D) => D=Vo/(Vi+Vo)
D(1) = Vo/(Vo+Vnew(1));
D(2) = Vo/(Vo+Vnew(2));
%iterator of HClimb
for i=3:iterations
    Vnew(i)=AHClimb(Irrad,Temp,Vold,Vcur,C);
    Vold=Vcur;
    Vcur=Vnew(i);
    %D=Vo/(Vi+Vo)
    D(i)=Vo/(Vo+Vnew(i));
end
% after every step's changing D need a delay to settle down
% here D_delay means settled Duty cycle
D_delay = zeros(1,iterations*delay);
Vnew_delay = zeros(1,iterations*delay);
% among the every step repeat the value 1000 times
% to simulate the reality condition
 for i=1:iterations
        Vnew_delay((i-1)*1000+1:i*1000)=Vnew(i);
        D_delay((i-1)*1000+1:i*1000)=D(i);
 end
end
```

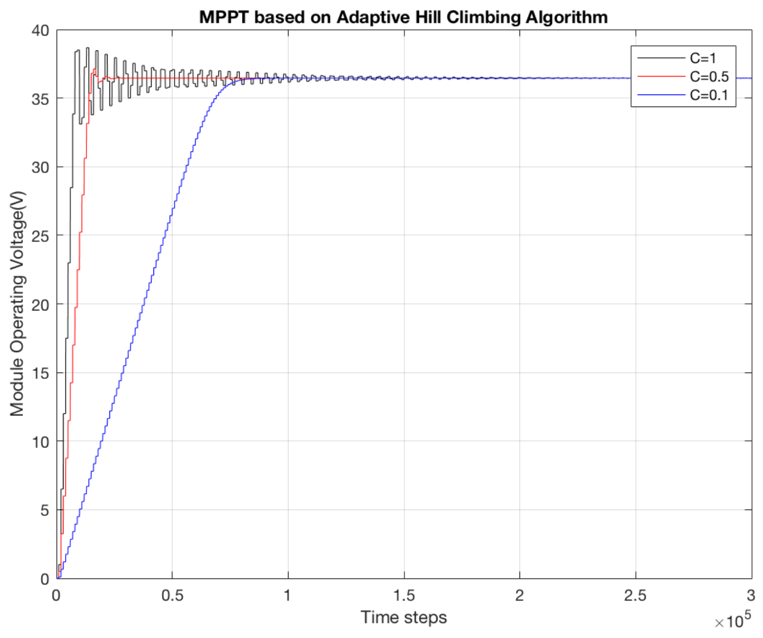
Then:

```
%**********************************************************************
% test function PVHC with different factor and plot D and Vnew
% different factor C: 0.1, 0.5 and 1.0
%**********************************************************************
Irrad=1000;
Temp=25+273;
%===========================factor C : 1.0===========================
C = 1;
%output the V and D
[D_delay_1,Vnew_delay_1] = PVAHC( Irrad,Temp,C );
%===========================factor C : 0.5===========================
C = 0.5;
%output the V and D
[D_delay_2,Vnew_delay_2] = PVAHC( Irrad,Temp,C );
%===========================factor C : 0.1===========================
C = 0.1;
```

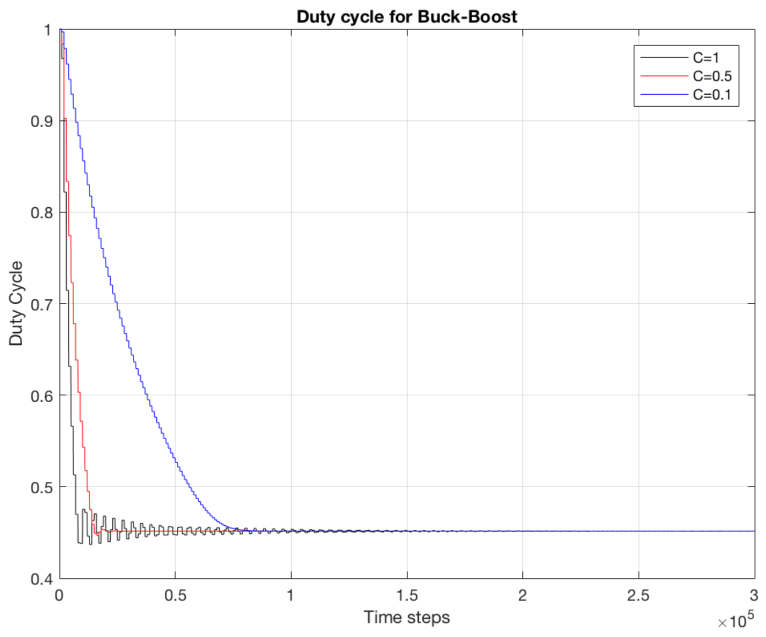
```
%output the V and D
[D_delay_3,Vnew_delay_3] = PVAHC( Irrad,Temp,C );
%*********************plot the Module Operating Voltage**********************
figure(1);
plot(Vnew_delay_1,'k');
hold on;
plot(Vnew_delay_2,'r');
plot(Vnew_delay_3,'b');
xlabel('Time steps');
ylabel('Module Operating Voltage(V)');
title('MPPT based on Adaptive Hill Climbing Algorithm');
legend('C=1','C=0.5','C=0.1');
grid on;
%*********************plot the Module Operating Voltage**********************
figure(2);
plot(D_delay_1,'k');
hold on;
plot(D_delay_2,'r');
plot(D_delay_3,'b');
xlabel('Time steps');
ylabel('Duty Cycle');
title('Duty cycle for Buck-Boost');
legend('C=1','C=0.5','C=0.1');
grid on;
```

The comparison of performance of operating voltage is shown as figure(1), and duty cycle is shown as figure(2).

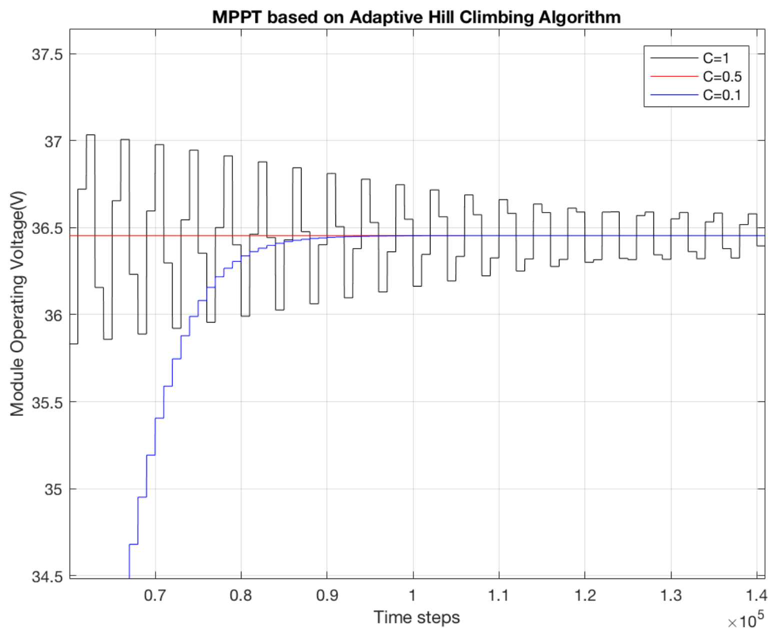
The comparison of steady state performance of operating voltage is shown as figure(3), and duty cycle is shown as figure(4).



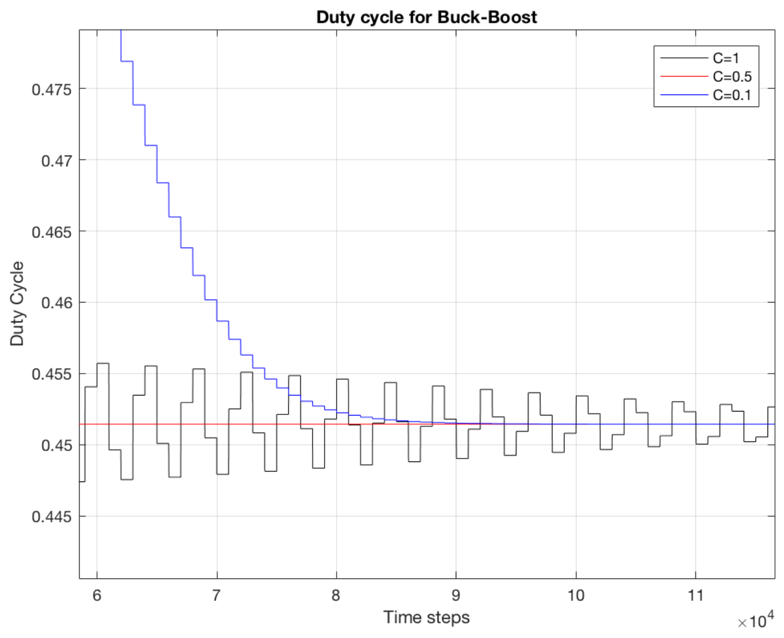
Figure(1): Operating voltage Performance of Adaptive Hill Climbing MPPT



Figure(2): Operating duty cycle of Adaptive Hill Climbing MPPT



Figure(3):Steady state performance of operating voltage of adaptive Hill Climbing



Figure(4):Steady state performance of duty cycle of adaptive Hill Climbing

Conclusion:

Compare with the Hill Climbing algorithm, the adaptive Hill Climbing algorithm has a variable perturbation size that gets smaller towards the MPP and the oscillation is smaller and smaller with time steps as shown in fig.(3) and fig.(4).

The adaptive Hill Climbing has a fast tracking at the first stage, which is far from the max power point, and a finer tracking at second state, which is near by the MPP.

The factor C decides the large of step size, it effects the tracking speed and accuracy. We can see from fig.(1) and fig.(2) that when C is equal to 1 the system is fast stabilized, however it has strong oscillation. When C is 0.1, the system has no oscillation, but it takes long time steps to go to steady state. When C takes a median value, which is 0.5, the system has slight oscillation, and it is stable quickly.

REFERENCES:

[1] Trishan Esram, Patrick L. Chapman, “Comparison of Photovoltaic Array Maximum Power Point Tracking Techniques”, IEEE TRANSACTIONS ON ENERGY CONVERSION, VOL. 22, NO. 2, JUNE 2007, PP 440