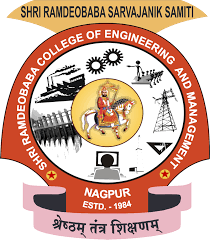
**PROJECT REPORT**

**ON**

PHOTOVOLTAIC SYSTEM ANAYSIS USING PYTHON



Session 2021-2022

Submitted by: -

1. **Mitesh Dutta ROLL NO-46**
2. **Tejas B Sawai ROLLNO-77**

**VI Semester Electrical Engineering**

**E- Circuit Design and Testing Lab (EEP376)**

Under the supervision of: -

**Dr. G. N. Goyal**

**Department of Electrical Engineering**

**Shri Ramdeobaba College of Engineering and Management, Nagpur**

*(An autonomous College of Rashtrasant Tukadoji Maharaj Nagpur University)*

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Title** | **Page Number** |
| 1 | Introduction | 2 |
| 2 | Important Libraries & Sub-Libraries used | 3 |
| 3 | Program Code | 3 |
| 4 | Conclusion | 5 |
| 5 | Applications | 6 |
| 6 | Future Scope | 6 |

**1. INTRODUCTION**

The power generation of a photovoltaic (PV) system may be documented by a capacity test that quantifies the power output of the system at set conditions, such as an irradiance of 1000 W/m2, an ambient temperature of 20°C, and a wind speed of 1 m/s. A longer test must be used to verify the system performance under a range of conditions. A year-long test samples weather and shading associated with all seasons. Shorter tests require less time, but may introduce seasonal bias, especially if the model is of inconsistent accuracy through the year (for example, if the shading is incorrectly estimated). Currently, no comprehensive standard exists for this type of test. Although the documentation of energy yield might appear to be straight forward, there are a number of subtleties that complicate the data analysis associated with variations in weather and imperfect data collection. These subtleties can complicate completion of an agreement associated with a performance guarantee, or completion of any test intended to quantify performance of a plant.

Completion of the energy yield evaluation with a very low uncertainty adds value to the project. Performance during a year may vary depending on such factors as:

• Seasonal shading issues

• Seasonal soiling

• Sensitivity to high temperatures

• Sensitivity of the model to weather – for example, if performance ratio is used as the metric, the measured performance will vary strongly with temperature

• Early system degradation

• Clipping or intentional curtailment.

Thus, there is general agreement that an energy test completed over a full year provides greater confidence that a PV system was correctly designed and installed, compared with a shorter test.

A standard for an Energy Performance Evaluation Method is challenging to write because the details of the test and test implementation depend on such things as:

• Size of the project

• Choice of the model

• Choice of the test boundary

• Responsibility for system operation and maintenance (O&M)

**2. Program Code: -**

**Main Code: -**

from tkinter import \*

from datetime import date

from datetime import datetime

today = date.today()

date = today.strftime("%B %d, %Y")

now = datetime.now()

time = now.strftime("%H:%M")

def part\_1():

root.destroy()

import part1

def part\_2():

root.destroy()

import part2

def part\_3():

root.destroy()

import part3

def part\_4():

root.destroy()

import part4

def end():

root.destroy()

root = Tk()

root.title('PV Cell Analysis Tool')

root.geometry('600x500')

logo = PhotoImage(file="icon.gif")

root.iconphoto(False,logo)

root.config(bg='black')

Label1=Label(root,text="\nWelcome to PV Cell Analysis Tool",font=('Arial Bold',18),fg='Blue',bg='Black').place(x=105,y=30)

Label2=Label(root,justify=LEFT,text=f'Time: {time}',fg='red',bg='black').place(x=0)

Label3=Label(root,justify=RIGHT,text=f'Date: {date}',fg='red',bg='black').place(x=0,y=20)

Label\_Heading=Label(root,text="To continue select one of the following options =>",font=('Arial',12),fg='springgreen',bg='black').place(x=5,y=130)

Label4=Label(root,text="1. To calculate the parameters of a PV Cell :",font=('Arial',12),fg='gainsboro',bg='black').place(x=5,y=210)

Button1=Button(root,text="Click Here!",bg='gray',command=part\_1).place(x=320,y=208)

Label5=Label(root,text="2. To display I-V curve of PV Panel :",font=('Arial',12),fg='gainsboro',bg='black').place(x=5,y=260)

Button2=Button(root,text="Click Here!",command=part\_2,bg='gray').place(x=260,y=258)

Label6=Label(root,text="3. To show TrueTracking and BackTracking of a PV cell :",font=('Arial',12),fg='gainsboro',bg='black').place(x=5,y=310)

Button3=Button(root,text="Click Here!",command=part\_3,bg='gray').place(x=405,y=308)

Label7=Label(root,text="4. To show GHI to POA graph of a PV cell :",font=('Arial',12),fg='gainsboro',bg='black').place(x=5,y=360)

Button4=Button(root,text="Click Here!",command=part\_4,bg='gray').place(x=310,y=358)

Button5=Button(root,text='Exit!',command=end,bg='gray').place(x=550,y=450)

root.mainloop()

**Calculation of P-V CELL Parameters: -**

import numpy

from tkinter import \*

from datetime import date

from datetime import datetime

today = date.today()

date = today.strftime("%B %d, %Y")

now = datetime.now()

time = now.strftime("%H:%M")

root = Tk()

root.title('Solar Parameters Calculator')

root.geometry('650x800')

logo = PhotoImage(file="icon.gif")

root.iconphoto(False,logo)

root.config(bg='black')

PVCurrent=DoubleVar()

OutputCurrent=DoubleVar()

Temprature=DoubleVar()

MaxCurrent=DoubleVar()

MaxVoltage=DoubleVar()

OCVoltage=DoubleVar()

SCCurrent=DoubleVar()

def end():

root.destroy()

exit()

def mainpage():

root.destroy()

import main

def ScCurrent():

I\_pv=float(PVCurrent.get())

I\_o=float(OutputCurrent.get())

Temp=float(Temprature.get())

V\_t =float(Temp\*1.38\*10\*\*(-23))/(1.6\*10\*\*(-19))

R\_s=float('0.1')

I\_sc=str(I\_pv - I\_o\*((2.718\*\*((I\_pv\*R\_s)/(2\*V\_t)))-1))

output=Label(root,text="Short Circuit Current = "+I\_sc+" A",font=('Arial',12),fg='white',bg='black').place(x=50,y=680)

def OcVoltage():

I\_pv=float(PVCurrent.get())

I\_o=float(OutputCurrent.get())

Temp=float(Temprature.get())

V\_t =(Temp\*1.38\*10\*\*(-23))/(1.6\*10\*\*(-19))

VarX = numpy.log((I\_pv+I\_o)/I\_o)

V\_oc=str(2\*V\_t\*VarX)

output=Label(root,text="Open Circuit voltage = "+V\_oc+' V',font=('Arial',12),fg='white',bg='black').place(x=50,y=680)

def Maxpp():

I\_m=float(MaxCurrent.get())

V\_m=float(MaxVoltage.get())

P\_m=str(I\_m\*V\_m)

output=Label(root,text="Maximum Power Point = "+P\_m+' Watts',font=('Arial',12),fg='white',bg='black').place(x=50,y=680)

def Efficiency():

I\_m=float(MaxCurrent.get())

V\_m=float(MaxVoltage.get())

E=str(((I\_m\*V\_m)/1000)\*100) #Considering input power ""P\_in"" as 1000 W/m^2

output=Label(root,text="Efficiency = "+E+' %',font=('Arial',12),fg='white',bg='black').place(x=50,y=680)

def Fillf():

I\_m=float(MaxCurrent.get())

V\_m=float(MaxVoltage.get())

I\_sc=float(SCCurrent.get())

V\_oc=float(OCVoltage.get())

F=str(float((I\_m\*V\_m)/(I\_sc\*V\_oc)))

output=Label(root,text="Fill Factor = "+F,font=('Arial',12),fg='white',bg='black').place(x=50,y=680)

Label1=Label(root,text="\nSolar Parameters Calculator",font=('Arial Bold',18),fg='blue',bg='Black').place(x=155,y=25)

Label\_Heading1=Label(root,text="Enter the following Values =>",font=('Arial',12),fg='springgreen',bg='black').place(x=20,y=110)

Label\_Heading1=Label(root,text="Chose an option from the following list to calculate the Parameter Value =>",font=('Arial',12),fg='springgreen',bg='black').place(x=20,y=390)

Label2=Label(root,justify=LEFT,text=f'Time: {time}',fg='red',bg='black').place(x=520)

Label3=Label(root,justify=RIGHT,text=f'Date: {date}',fg='red',bg='black').place(x=520,y=20)

Label6=Label(root,text="To Calculate Short Circuit Current :",font=('Arial',12),fg='gainsboro',bg='black').place(x=60,y=450)

Label7=Label(root,text="To Calculate Open Circuit Voltage :",font=('Arial',12),fg='gainsboro',bg='black').place(x=60,y=490)

Label8=Label(root,text="To Calculate Maximum Power Point :",font=('Arial',12),fg='gainsboro',bg='black').place(x=60,y=530)

Label9=Label(root,text="To Calculate Efficiency :",font=('Arial',12),fg='gainsboro',bg='black').place(x=60,y=570)

Label10=Label(root,text="To Calculate Fill Factor :",font=('Arial',12),fg='gainsboro',bg='black').place(x=60,y=610)

Label6=Label(root,text="Enter PV Current :",font=('Arial',12),fg='gainsboro',bg='black').place(x=60,y=150)

Label6=Label(root,text="Enter Output Current :",font=('Arial',12),fg='gainsboro',bg='black').place(x=60,y=180)

Label6=Label(root,text="Enter Temprature in Celcius :",font=('Arial',12),fg='gainsboro',bg='black').place(x=60,y=210)

Label6=Label(root,text="Enter Maximum Current :",font=('Arial',12),fg='gainsboro',bg='black').place(x=60,y=240)

Label6=Label(root,text="Enter Maximum Voltage :",font=('Arial',12),fg='gainsboro',bg='black').place(x=60,y=270)

Label6=Label(root,text="Enter Open Circuit Voltage :",font=('Arial',12),fg='gainsboro',bg='black').place(x=60,y=300)

Label6=Label(root,text="Enter Short Circuit Current :",font=('Arial',12),fg='gainsboro',bg='black').place(x=60,y=330)

PVCurrent1=Entry(root,textvariable=PVCurrent).place(x=400,y=149)

OutputCurrent1=Entry(root,textvariable=OutputCurrent).place(x=400,y=179)

Temprature1=Entry(root,textvariable=Temprature).place(x=400,y=209)

MaxCurrent1=Entry(root,textvariable=MaxCurrent).place(x=400,y=239)

MaxVoltage1=Entry(root,textvariable=MaxVoltage).place(x=400,y=269)

OCVoltage1=Entry(root,textvariable=OCVoltage).place(x=400,y=299)

SCCurrent1=Entry(root,textvariable=SCCurrent).place(x=400,y=329)

Button2=Button(root,text='Exit!',command=end,bg='gray').place(x=580,y=750)

Button3=Button(root,text='Return to Main Page!',command=mainpage,bg='gray').place(x=20,y=10)

Button4=Button(root,text='Click here!',command=ScCurrent,bg='gray').place(x=400,y=450)

Button5=Button(root,text='Click here!',command=OcVoltage,bg='gray').place(x=400,y=490)

Button6=Button(root,text='Click here!',command=Maxpp,bg='gray').place(x=400,y=530)

Button7=Button(root,text='Click here!',command=Efficiency,bg='gray').place(x=400,y=570)

Button8=Button(root,text='Click here!',command=Fillf,bg='gray').place(x=400,y=610)

root.mainloop()

**Output : -**

**Plotting IV Curve for different Irradiance & Temperature: -**

from tkinter import \*

from datetime import date

from datetime import datetime

from pvlib import pvsystem

import pandas as pd

import matplotlib.pyplot as plt

import os

today = date.today()

date = today.strftime("%B %d, %Y")

now = datetime.now()

time = now.strftime("%H:%M")

path='Specs.pdf'

root = Tk()

root.title('I-V Curve')

root.geometry('750x350')

logo = PhotoImage(file="icon.gif")

root.iconphoto(False,logo)

root.config(bg='black')

def end():

root.destroy()

def mainpage():

root.destroy()

import main

def pdf():

root.destroy()

os.system(path)

def run():

root.destroy()

# Module parameters for the Vikram Solar ELDORA NEO 72 SILVER SERIES:

parameters = {

'Name': 'Vikram Solar ELDORA NEO 72 SILVER SERIES',

'Date': '10/5/2009',

'T\_NOCT': 45,

'A\_c': 2.34,

'N\_s': 72,

'I\_sc\_ref': 9.22,

'V\_oc\_ref': 46.5,

'I\_mp\_ref': 8.66,

'V\_mp\_ref': 38.71,

'alpha\_sc': 0.052,

'beta\_oc': -0.310,

'a\_ref': 2.6373,

'I\_L\_ref': 5.114,

'I\_o\_ref': 8.196e-10,

'R\_s': 1.065,

'R\_sh\_ref': 381.68,

'Adjust': 8.7,

'gamma\_r': -0.476,

'PTC': 200.1,

'Technology': 'Mono-c-Si',

}

cases = [

(1000, 55),

(800, 55),

(600, 55),

(400, 25),

(400, 40),

(400, 55)

]

conditions = pd.DataFrame(cases, columns=['Eff Iradiance', 'Tcell'])

# De Soto

IL, I0, Rs, Rsh, nNsVth = pvsystem.calcparams\_desoto(

conditions['Eff Iradiance'],

conditions['Tcell'],

alpha\_sc=parameters['alpha\_sc'],

a\_ref=parameters['a\_ref'],

I\_L\_ref=parameters['I\_L\_ref'],

I\_o\_ref=parameters['I\_o\_ref'],

R\_sh\_ref=parameters['R\_sh\_ref'],

R\_s=parameters['R\_s'],

EgRef=1.121, # Band Gap

dEgdT=-0.0002677

)

# Use the output parameters from above and solve for IV curves:

curve\_info = pvsystem.singlediode(

photocurrent=IL,

saturation\_current=I0,

resistance\_series=Rs,

resistance\_shunt=Rsh,

nNsVth=nNsVth,

ivcurve\_pnts=100,

method='lambertw'

)

# For plotting the curves:

plt.figure()

for i, case in conditions.iterrows():

label = (

"$G\_{eff}$ " + f"{case['Eff Iradiance']} $W/m^2$\n"

"$T\_{cell}$ " + f"{case['Tcell']} $C$"

)

plt.plot(curve\_info['v'][i], curve\_info['i'][i], label=label)

v\_mp = curve\_info['v\_mp'][i]

i\_mp = curve\_info['i\_mp'][i]

# To mark the MPP we use:

plt.plot([v\_mp], [i\_mp], ls='', marker='o', c='k')

plt.legend(loc=(1.0, 0))

plt.xlabel('Module voltage [V]')

plt.ylabel('Module current [A]')

plt.title(parameters['Name'])

plt.show()

plt.gcf().set\_tight\_layout(True)

print(pd.DataFrame({

'i\_sc': curve\_info['i\_sc'],

'v\_oc': curve\_info['v\_oc'],

'i\_mp': curve\_info['i\_mp'],

'v\_mp': curve\_info['v\_mp'],

'p\_mp': curve\_info['p\_mp'],

}))

exit()

Label1=Label(root,text="\nI-V Curve of PV Cell",font=('Arial Bold',18),fg='blue',bg='Black').place(x=255,y=25)

Label2=Label(root,justify=LEFT,text=f'Time: {time}',fg='red',bg='black').place(x=620)

Label3=Label(root,justify=RIGHT,text=f'Date: {date}',fg='red',bg='black').place(x=620,y=20)

Label\_Heading=Label(root,text="To continue select one of the following options =>",font=('Arial',12),fg='springgreen',bg='black').place(x=20,y=110)

Label4=Label(root,text="To Display the I-V curve of 'Vikram Solar ELDORA NEO 72 SILVER SERIES' PV Panel :",font=('Arial',12),fg='gainsboro',bg='black').place(x=20,y=180)

Label5=Label(root,text="To see the specifications of Vikram Solar PV Panel :",font=('Arial',12),fg='gainsboro',bg='black').place(x=20,y=220)

Button1=Button(root,text="Click Here!",command=run,bg='gray').place(x=640,y=178)

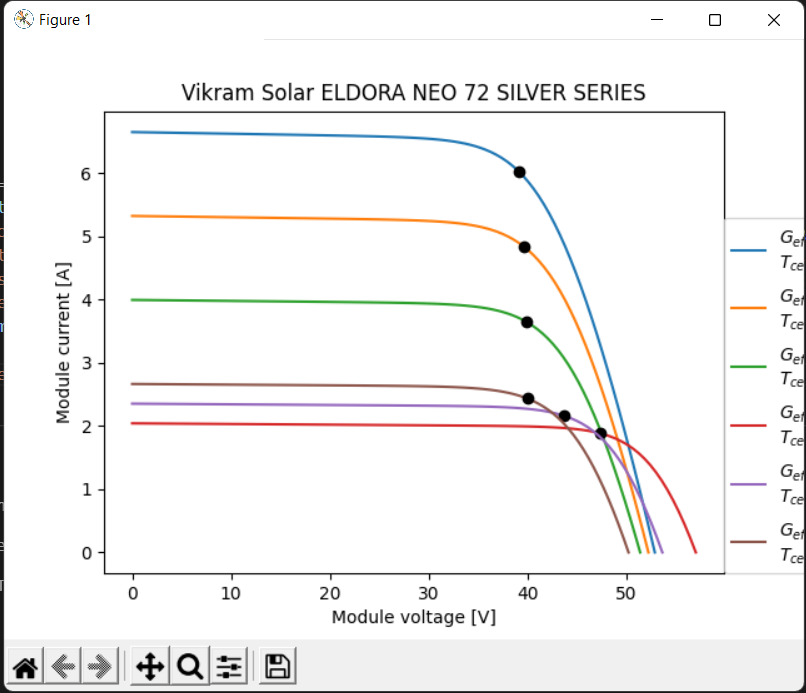
Button2=Button(root,text='Exit!',command=end,bg='gray').place(x=700,y=310)

Button3=Button(root,text='Return to Main Page!',command=mainpage,bg='gray').place(x=20,y=10)

Button5=Button(root,text="Click Here!",command=pdf,bg='gray').place(x=390,y=218)

root.mainloop()

**Output : -**



**Single-axis tracking: -**

from pvlib import solarposition, tracking

import pandas as pd

import matplotlib.pyplot as plt

from tkinter import \*

from datetime import date

from datetime import datetime

today = date.today()

date = today.strftime("%B %d, %Y")

now = datetime.now()

time = now.strftime("%H:%M")

root = Tk()

root.title('Tracking')

root.geometry('750x500')

logo = PhotoImage(file="icon.gif")

root.iconphoto(False,logo)

root.config(bg='black')

lat\_var=StringVar()

lon\_var=StringVar()

strdate=StringVar()

stpdate=StringVar()

def end():

root.destroy()

def mainpage():

root.destroy()

import main

def submit():

root.destroy()

latitude=float(lat\_var.get())

longitude=float(lon\_var.get())

Str\_date=strdate.get()

Stop\_date=stpdate.get()

tz = 'Asia/Kolkata'

lat, lon = latitude, longitude

times = pd.date\_range(Str\_date, Stop\_date, closed='left', freq='5min',

tz=tz)

solpos = solarposition.get\_solarposition(times, lat, lon)

truetracking\_angles = tracking.singleaxis(

apparent\_zenith=solpos['apparent\_zenith'],

apparent\_azimuth=solpos['azimuth'],

axis\_tilt=0,

axis\_azimuth=180,

max\_angle=90,

backtrack=False, # for true-tracking

gcr=0.5)

truetracking\_position = truetracking\_angles['tracker\_theta'].fillna(0)

truetracking\_position.plot(title='Truetracking Curve')

plt.show()

# Backtracking

fig, ax = plt.subplots()

for gcr in [0.2, 0.4, 0.6]:

backtracking\_angles = tracking.singleaxis(

apparent\_zenith=solpos['apparent\_zenith'],

apparent\_azimuth=solpos['azimuth'],

axis\_tilt=0,

axis\_azimuth=180,

max\_angle=90,

backtrack=True,

gcr=gcr)

backtracking\_position = backtracking\_angles['tracker\_theta'].fillna(0)

backtracking\_position.plot(title='Backtracking Curve',

label=f'GCR:{gcr:0.01f}',

ax=ax)

plt.legend()

plt.show()

exit()

Label\_Heading=Label(root,text="To continue enter the following values and click submit =>",font=('Arial',12),fg='springgreen',bg='black').place(x=20,y=110)

Label1=Label(root,text="\nTrueTracking And BackTracking",font=('Arial Bold',18),fg='blue',bg='Black').place(x=180,y=25)

Label2=Label(root,justify=LEFT,text=f'Time: {time}',fg='red',bg='black').place(x=620)

Label3=Label(root,justify=RIGHT,text=f'Date: {date}',fg='red',bg='black').place(x=620,y=20)

Label4=Label(root,text="Location : Asia/Calcutta",font=('Arial',12),fg='gainsboro',bg='black').place(x=40,y=180)

Label5=Label(root,text="Enter Latitude :",font=('Arial',12),fg='gainsboro',bg='black').place(x=40,y=220)

Label6=Label(root,text="Enter Longitude :",font=('Arial',12),fg='gainsboro',bg='black').place(x=40,y=260)

Label7=Label(root,text="Enter Start Date for tracking(yyyy-mm-dd):",font=('Arial',12),fg='gainsboro',bg='black').place(x=40,y=300)

Label8=Label(root,text="Enter End Date for tracking(yyyy-mm-dd):",font=('Arial',12),fg='gainsboro',bg='black').place(x=40,y=340)

Entry1=Entry(root,textvariable=lat\_var).place(x=360,y=222)

Entry2=Entry(root,textvariable=lon\_var).place(x=360,y=262)

Entry3=Entry(root,textvariable=strdate).place(x=360,y=302)

Entry4=Entry(root,textvariable=stpdate).place(x=360,y=342)

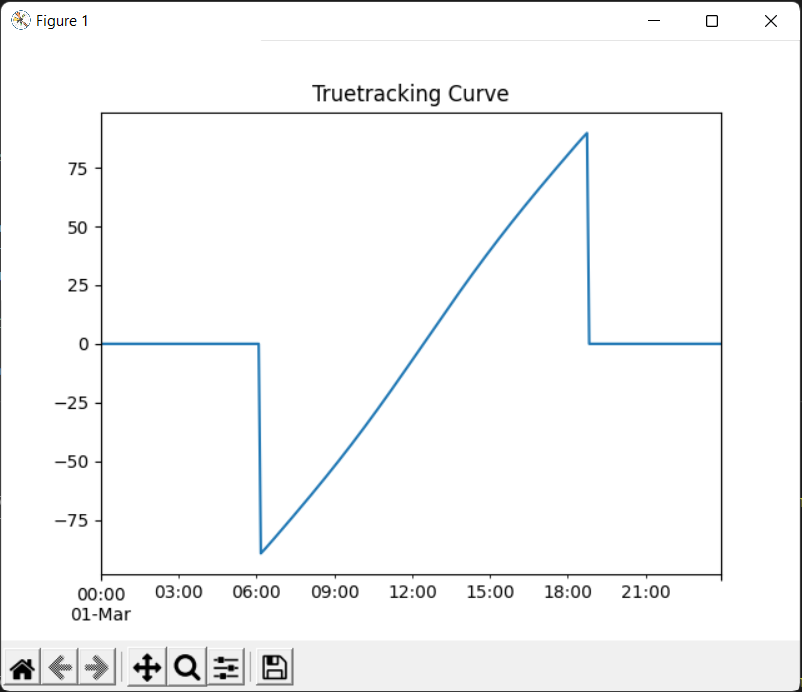
Button1=Button(root,text='Submit',command=submit,bg='gray',width=10).place(x=600,y=460)

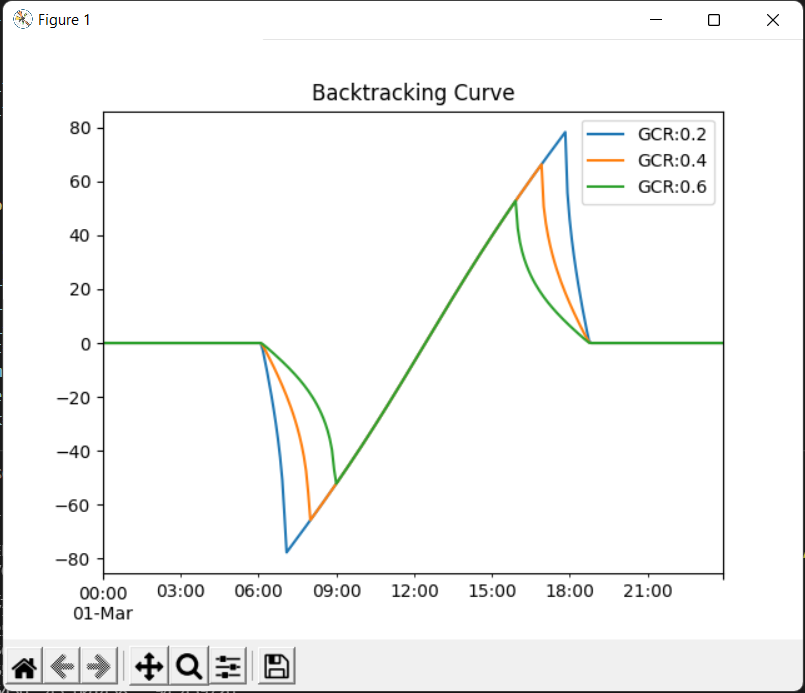
Button2=Button(root,text='Exit!',command=end,bg='gray').place(x=700,y=460)

Button3=Button(root,text='Return to Main Page!',command=mainpage,bg='gray').place(x=20,y=10)

root.mainloop()

**Output : -**





**GHI to POA Transposition: -**

from pvlib import location

from pvlib import irradiance

import pandas as pd

from matplotlib import pyplot as plt

from tkinter import \*

from datetime import date

from datetime import datetime

today = date.today()

date = today.strftime("%B %d, %Y")

now = datetime.now()

time = now.strftime("%H:%M")

root = Tk()

root.title('GHI to POA')

root.geometry('750x350')

logo = PhotoImage(file="icon.gif")

root.iconphoto(False,logo)

root.config(bg='black')

lat\_var=StringVar()

lon\_var=StringVar()

def end():

root.destroy()

exit()

def mainpage():

root.destroy()

import main

def submit():

root.destroy()

latitude=float(lat\_var.get())

longitude=float(lon\_var.get())

tz = 'Asia/Kolkata'

lat, lon = latitude, longitude

site = location.Location(lat, lon, tz=tz)

def get\_irradiance(site\_location, date, tilt, surface\_azimuth):

times = pd.date\_range(date, freq='10min', periods=6\*24,

tz=site\_location.tz)

clearsky = site\_location.get\_clearsky(times)

# Azimuth and Zenith

solar\_position = site\_location.get\_solarposition(times=times)

# GHI to POA

POA\_irradiance = irradiance.get\_total\_irradiance(

surface\_tilt=tilt,

surface\_azimuth=surface\_azimuth,

dni=clearsky['dni'],

ghi=clearsky['ghi'],

dhi=clearsky['dhi'],

solar\_zenith=solar\_position['apparent\_zenith'],

solar\_azimuth=solar\_position['azimuth'])

return pd.DataFrame({'GHI': clearsky['ghi'],

'POA': POA\_irradiance['poa\_global']})

summer\_irradiance = get\_irradiance(site, '06-20-2020', 23, 180)

winter\_irradiance = get\_irradiance(site, '12-21-2020', 23, 180)

summer\_irradiance.index = summer\_irradiance.index.strftime("%H:%M")

winter\_irradiance.index = winter\_irradiance.index.strftime("%H:%M")

fig, (ax1, ax2) = plt.subplots(1, 2, sharey=True)

summer\_irradiance['GHI'].plot(ax=ax1, label='GHI')

summer\_irradiance['POA'].plot(ax=ax1, label='POA')

winter\_irradiance['GHI'].plot(ax=ax2, label='GHI')

winter\_irradiance['POA'].plot(ax=ax2, label='POA')

ax1.set\_xlabel('Time of day (Summer)')

ax2.set\_xlabel('Time of day (Winter)')

ax1.set\_ylabel('Irradiance ($W/m^2$)')

ax1.legend()

ax2.legend()

plt.show()

exit()

Label\_Heading=Label(root,text="To continue enter the following values and click submit =>",font=('Arial',12),fg='springgreen',bg='black').place(x=20,y=110)

Label1=Label(root,text="\nGHI to POA",font=('Arial Bold',18),fg='blue',bg='Black').place(x=315,y=25)

Label2=Label(root,justify=LEFT,text=f'Time: {time}',fg='red',bg='black').place(x=620)

Label3=Label(root,justify=RIGHT,text=f'Date: {date}',fg='red',bg='black').place(x=620,y=20)

Label4=Label(root,text="Location : Asia/Calcutta",font=('Arial',12),fg='gainsboro',bg='black').place(x=40,y=180)

Label5=Label(root,text="Enter Latitude :",font=('Arial',12),fg='gainsboro',bg='black').place(x=40,y=220)

Label6=Label(root,text="Enter Longitude :",font=('Arial',12),fg='gainsboro',bg='black').place(x=40,y=260)

Entry1=Entry(root,textvariable=lat\_var).place(x=360,y=222)

Entry2=Entry(root,textvariable=lon\_var).place(x=360,y=262)

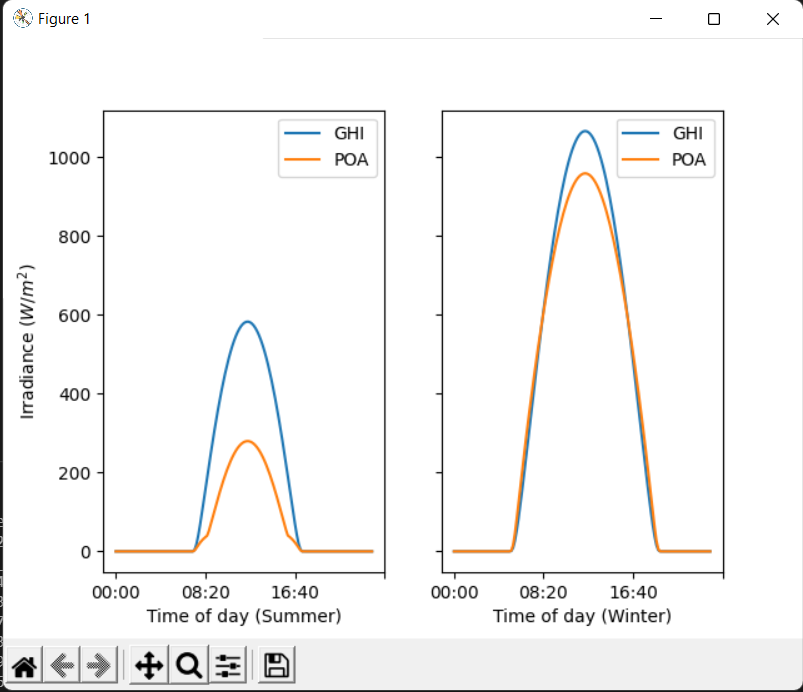
Button1=Button(root,text='Submit',command=submit,bg='gray',width=10).place(x=600,y=310)

Button2=Button(root,text='Exit!',command=end,bg='gray').place(x=700,y=310)

Button3=Button(root,text='Return to Main Page!',command=mainpage,bg='gray').place(x=20,y=10)

root.mainloop()

**Output : -**



**Important Libraries & Sub-Libraries used**

|  |  |  |
| --- | --- | --- |
| **SR NO.** | **Libraries** | **Sub-Libraries** |
| 1. | tkinter | \* |
| 2. | datetime | date, datetime |
| 3. | pvlib | pvsystem, solarposition, tracking, location, irradiance |
| 4. | matplotlib | pyplot |
| 5. | pandas |  |

**ADVANTAGES**

1. It helps to reduce the time that is utilized while calculating everything by hand.

2. More & more efficiency can be attained by using these techniques.

3. Can be used everywhere in the world irrespective of the changes in irradiance and location.

**FUTURE SCOPE**

Generation of solar energy has tremendous scope in India. The geographical location of the country stands to its benefit for generating solar energy. The reason being India is a tropical country and it receives solar radiation almost throughout the year, which amounts to 3,000 hours of sunshine. This is equal to more than 5,000 trillion kWh. Almost, all parts of India receive 4-7 kWh of solar radiation per sq meters. This is equivalent to 2,300–3,200 sunshine hours per year. States like Andhra Pradesh, Bihar, Gujarat, Haryana, Madhya Pradesh, Maharashtra, Orissa, Punjab, Rajasthan, and West Bengal have great potential for tapping solar energy due to their location. Since majority of the population live in rural areas, there is much scope for solar energy being promoted in these areas. Use of solar energy can reduce the use of firewood and dung cakes by rural household. Many large projects have been proposed in India, some of them are: i). Thar Desert of India has best solar power projects, estimated to generate 700 to 2,100 GW, ii). The Jawaharlal Nehru National Solar Mission (JNNSM) launched by the Centre is targeting 20,000 MW of solar energy power by 2022, iii). Gujarat’s pioneering solar power policy aims at 1,000 MW of solar energy generation, and Rs. 130 billion solar power plan was unveiled in July 2009, which projected to produce 20 GW of solar power by 2020. Apart from above, about 66 MW is installed for various applications in the rural area, amounting to be used in solar lanterns, street lighting systems and solar water pumps, etc. Thus, India has massive plan for Solar Energy generation that may not only fulfill the deficit of power generation but also contribute largely in Green Energy Production to help to reduce the Climatic Changes globally.