Arizona State University

School of Electrical, Computer and Energy Engineering

EEE 465/ 591: Photovoltaic Energy Conversion

Mini Project #1: Solar Radiation

Introduction

The four "Mini Projects" are designed to lead you toward being able to analyze and evaluate the performance of a stand-alone Photovoltaic installation and enable you to make design choices based on customer input. The Mini Projects will help you in developing your Final Project idea and you can fall back on the calculations you performed for those Mini Projects.

Mini Project 1 will help you to answer the question "What about the amount of energy we can harvest at a specific point on Earth on a given day of the year?"

The goal of this project is to become familiar with several aspects of solar radiation needed to calculate the performance of a solar cell and to calculate and design a PV system. In addition, Project 1 is intended make sure that you can set up a viable programming approach for later projects, in particular for the final project. For these reasons, Project 1 is a little different than the others, focused more on programming. The project will read in a solar radiation file, perform some calculations from a library file with the data you read in, and then plot the data in various ways.

All the programs in the mini projects are cumulative, so while the first project appears fairly straight forward – read an input file, use a library file, and different types of plots – the code will be used for all the later projects, so it is very much in your interest to comment your code and make it user friendly. You absolutely do not want to be hunting through all the details of your code 3 months from now trying to find the name of a variable to pass to a function for your final project.

Programming Approaches

Any programming language is acceptable for the projects. If you are familiar with a programming language, by all means use it. **We will be providing programming templates and basic support for Python. By some metrics, Python has become the most taught and used programming language.** There are many programming routines and support for Python. Importantly, there is substantial graphing and user interface support. Nevertheless, all the projects have been successfully done in everything from assembly language to Excel, and there are a few more detailed notes about more common programing approaches below.

All of the projects, including the final project can be done in Excel. Excel is very useful in providing a check and quick calculations. However, several of the later mini-projects involve optimization, something that is often inconvenient in Excel. Often, I have found that students using Excel end up doing the optimizations by "hand", changing values in Excel manually. This becomes increasingly cumbersome for the final project. Overall, while the Excel "calculators" are useful, I recommend that Excel is a check and a supplement rather than the full program.

Excel, in addition to the formula-based calculations that are most commonly used, also has programing in Visual Basic. Visual Basic in Excel is a fully functioning programming language, and if you are familiar with Visual Basic in Excel, then this is a perfectly acceptable route, and one that makes graphing results relatively easy and very portable.

Matlab is among the most commonly student-used programming languages. However, it is often less commonly used after graduation, as the licenses for non-students are relatively high. Nevertheless, I would estimate that historically many of the student projects are submitted in Matlab**. We will not provide**

programming templates for solar radiation for Matlab, but there are many available, and Matlab is perfectly acceptable for programming all the projects.

The decision of which programming approach is often revolving around the support for graphing and other input/output functions. This is the reason Excel is commonly used; it is fairly easy and straightforward to graphically examine the outputs. Python has extensive support for graphs and I/O, as does Matlab. In the end, it is not uncommon to spend more time on the graphing, reading files, exporting data, etc, so this should factor very heavily in your decision.

A final note is that you do not have to use the same programming language in all the projects or even portions of the projects. Historically, some students have made programming languages read output Excel files, or other combinations. I have found that there is quite a bit of fiddling in this approach to get the formats to match. It is a very flexible approach but be warned that it is typically been very “buggy” and fairly time intensive to get different parts of the code to talk to each other.

You should prepare a report in which you briefly describe your approach to the problem and include your outputs (numeric calculations, plots). In addition, submit the source code of your programs (Excel sheets, Python code).

Project Goals

The goal of the first mini-project is to read a solar radiation file for a particular location containing at least hourly solar radiation data as well as other location information, such as temperature. Your program should:

(1) Read a data file with solar radiation data in it (see later notes on solar radiation data sets).
(2) Calculate the position of the sun for every hour of the year (or whatever time period matched the solar radiation data).
(3) From (1) and (2) above and the angles of a PV module, calculate the power normal to the module for every hour of the year
(4) Integrate over a given time period to find the energy produced in a year (or month, or day, or whatever time period).

Solar Radiation Sets

Nearly every PV system calculation starts with a solar radiation data set. As such, there are several places to get the data. For the US (and also includes other countries such as India and Vietnam). The types and details of solar radiation datasets are covered more in other classes (PV Systems); here we use a dataset called TMY3 which uses ground-based measurements for 273 locations in the US. **We have already downloaded all the TMY3 data and have it in .csv files, so you don’t have query the solar radiation database**. However, depending on your final project choice, you may want to use the radiation database. European data is given at: https://ec.europa.eu/jrc/en/pvgis . They also have a TMY tool and other ways to get solar radiation data.

Project Tasks

1. Read the TMY data for a location of your choice from the given file and demonstrate that your program reads the data correctly by plotting the direct, diffuse and global radiation for every hour of the year. Make sure to label the location. **<u>In order to get credit for this part, you need to change something in the plot compared to the plot from the shell code.</u>** (3 pts)

2. Calculate the position of the sun (azimuth and elevation angle) for every hour of the year.
   a. Write down the equations for azimuth and elevation and do a hand calculation for the location of your TMY data at one hour. (2 pts)
   b. Demonstrate your code calculates the azimuth and elevation by plotting the path of the sun across the sky for one day of the year (note this is a polar plot). **<u>In order to get credit for this part, you need to change something in the plot compared to the plot from the shell code.</u>** (2 pts)

   We have given you some of the calculations in Python shell codes and library. (You do not need to use these; sometimes using someone else's program is harder than programming it yourself. However, the equations require attention to details – you need pay attention to signs, if the angles are in the correct quadrant when you take inverse trig functions and the difference between radians and degrees.)

3. In your code,
   a. Calculate the normally incident solar radiation a tilted surface for every hour of the year. You can use whatever tilt angle you like; good default values in the Northern Hemisphere are module tilt = latitude of your location, azimuth = 180. (1 pt)
   b. Calculate the total power density on your tilted surface over the year. (1 pt)
   c. Plot the solar radiation power density on your module for every hour of the year and put the total over the year in the graph heading. (1 pt)

4. Choose **<u>one</u>** of the following calculations or plots. (5 pts).
   i. Surface color map of solar radiation on a polar plot with azimuth and elevation.
   ii. "Heat map" of solar radiation vs day of year and hour of the day that is different from the provide Python shell code.
   iii. Calculate and plot the total solar radiation in each month or day of the year rather than each hour for three different tilt angles on the same plot.
   iv. Calculate and give the capacity factor for each month of the year.
   v. Calculate and plot the fraction diffuse radiation for each month of the year.
   vi. Trace the path of the sun on a polar plot (azimuth, elevation) for the same time of day for every day of the year.
   vii. Query the national solar radiation database and get data directly from it.
   viii. Number of days in a row that have < 50% of expected solar radiation.
   ix. Difference between calculated "ideal" radiation and measured radiation

Running the provided Python Code

If you do not already have Python, you will need to download it. You can download a basic version at:

https://www.python.org/ Install version 3.9.6, but other versions 3.6 or higher work as well.

If you are looking for an Integrated Development Environment (IDE), there are several options for example Spyder https://www.spyder-ide.org/. Installation can be quite involved and students in the past have used Anaconda to install the IDE of their choice. https://www.anaconda.com/

Another option is to use Jupyter Notebooks https://jupyter.org/ for you python code. This also you to combine readable text and executable code in a document. Google Colab is another option if you like Google Docs. https://colab.research.google.com/notebooks/intro.ipynb

We have noticed some rare cases in which code will work on one platform and not the other; I will keep a computer with the Spyder editor and a Linux computer with command-line Python. I will accept Jupyter files as well (they are ideal for reports, etc).

After you have installed python, you will need to download the sample files and add the library file **photovoltaic**. I recommend putting all the python code (libraries, etc) all in the same directory for simplicity.

To install the library file, go to the command prompt (cmd) and type:

```
pip install photovoltaic
```

If you don't want to install the library file, we don't need that many functions from it for this assignment.

You can also paste them into you code from the code snippet on canvas.