COMPUTERS AND INFORMATICS
2757-8259 2021, Volume 1 Issue 2
Research Article

Power loss calculation of Photovoltaics using Python

Kerry Sado
University of Duhok, Duhok, Iraq, kerry.sado@uod.ac
Ismail Ali
University of Duhok, Duhok, Iraq, ismail@uod.ac
Lokman Hadi
University of Duhok, Duhok, Iraq, lokman.hadi@uod.ac
Shivan Sado
University of Nebraska-Lincoln, Lincoln, NE, USA, ssado2@unl.edu

Submitted: *14.06.2021*
Accepted: *31.08.2021*
Published: *30.12.2021*

Abstract:

The issue of electrical power losses in solar Photovoltaic modules during partial shade is discussed in this study. PV modules work more efficiently with maximum power if solar irradiation reaches the entire surface of the panel without any parts been shaded. A PV system modeling software package, PVLib, is used to calculate the power loss of PV modules from partial shading using Python coding. In-order to calculate the power loss of modules from partial shading, the current and voltage (IV) curves for individual solar photovoltaic cells should be calculated, by solving the single diode equation IV curve. Parameters of an available PV module are used. Standard Test Condition (STC) temperature and parameters are used in the calculations. Different Python libraries and packages are used and imported in this report. Brief introduction to the background of each library is introduced, steps of adding an external package to Python are also mentioned.

Keywords: *PV cell, Python, Renewable energy, Solar panel*

Cite this paper as: Sado, K, Ali, I, Hadi, L, & Sado, S. Power loss calculation of Photovoltaics using Python, *Computers and Informatics*, 2021, *1*(2), 74-82.

1. INTRODUCTION

Although the PV cell is the primary power generation unit, since module-level parameters are much more accessible and it greatly decreases the computational scope of the simulation, PV modeling is mostly performed for convenience at the module level. Module-level simulations, though, are too coarse to be able to simulate results such as mismatch of cell to cell or partial shading. This example measures IV curves at cell level and combines them to recreate the IV curve at module level. It uses this approach to find the maximum power under various shading and irradiance conditions. Flexibility of modeling is useful in designing, organizing, the performance of photovoltaic panels and predicting them. Currently, A range of software packages are available that can be used to model PV Systems' performance, for both commercial and educational purposes. However, most of these simulation programs have limitations. For starters, users often have limited access to algorithms and assumptions used in simulation programs such as PVSyst and PVSol [1]. Users often find it difficult to simulate more sophisticated systems than what the developers originally allowed. Examples of such complex systems range from multiple array orientations, differing PV panels and inverters and varied PV system designs. Some packages use default input limitations such as DC to AC power ratios and they are often not easy to modify. Bug fixing is often time-consuming in those systems, as it has to be performed by the innovative authors and then rearranged. PVLib has the ability to bridge this gap in PV system modelling. The rest of this report is organized as follows: the following section gives account of the theoretical framework of PVLib. The analyses of the results of the simulations and the conclusions will be given in the following pages.

2. HISTORY OF PVLIB

PVLib toolbox is a common repository for algorithms for high-quality modeling and study of PV structures. The toolbox has the privilege of being developed and tested continually and collaboratively. This not only makes for better accuracy, but also for easier bug fixing. The code is open-source; therefore, users can freely view, easily modify and redistribute the original source code. pvlib python is developed on GitHub by contributors from academia, national laboratories, and private industry [2]. In this examples different Python libraries are imported, libraries like Panda, NumPy, SciPy and Matplotlib. A brief introduction to each library will be given in the following:

2.1. Panda

The panda package is the most powerful platform accessible to today's Python-based data scientists and analysts. All of the focus can be paid to powerful machine learning and stylish visualization methods, but pandas are the foundation of most data projects. Pandas is built on top of the NumPy package, which means that Pandas uses or reproduces a lot of the NumPy structure [3]. Panda data is also used to feed SciPy statistical analysis, Matplotlib plotting functions, and Scikit-learn machine learning algorithms.

2.2. Numpy

NumPy is a package for Python. It stands for 'Python Numeric'. It is a library consisting of objects in a multidimensional array and a set of array processing routines [4]. Using NumPy, following operations can be performed by developers:

- Mathematical and/or logical operations on arrays.
- Fourier transforms and routines for shape manipulation.

- Operations related to linear algebra. NumPy has built in functions for linear algebra and random number generation.

2.3. Scipy

SciPy is a series of mathematical algorithms based on Python's NumPy extension and convenience functions [5]. By providing the user with high-level commands and classes to manipulate and visualize data, it adds important power to the interactive Python session. With SciPy, an interactive Python session becomes a data-processing and system-prototyping environment rivaling systems, such as MATLAB, IDL, Octave, R-Lab, and SciLab.

2.4. Matplotlib

It is one of the most popular Python packages used for data visualization. Matplotlib provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonotTkinter [6]. It can be used in Python and IPython shells, Jupyter notebook and web application servers also.

3. SINGLE DIODE EQUATION

In this section the solutions of single diode equation used in pvlib-python to generate an IV curve of a PV module will be reviewed.

pvlib-python supports two ways to solve the single diode equation:

1. Lambert W-Function
2. Bishop's Algorithm

3.1. Lambert W-Function

According to Lambert algorithm, the single diode model equation can be given as [7]:

$$I = I_L - I_0\left(\exp\left(\frac{V + IR_s}{nNsV_{th}}\right) - 1\right) - \frac{V + IR_s}{R_{sh}} \quad (1)$$

Lambert W-function is the inverse of the function

$$f(w) = w\exp(w)$$

Defining the following parameter, z, is necessary to transform the single diode equation into a form that can be expressed as a Lambert W-function

$$z = \frac{R_s I_0}{nNsV_{th}\left(1 + \frac{R_s}{R_{sh}}\right)}\exp\left(\frac{R_s(I_L + I_0) + V}{nNsV_{th}\left(1 + \frac{R_s}{R_{sh}}\right)}\right) \quad (2)$$

where,

I_L: Is the light current (A)

I_D: reverse saturation current of the diode (A)
R_s: series resistance (Ω)
R_{sh}: shunt resistance (Ω)
n : ideality factor of the diode (unitless)

Then Lambert W-function, W(z) can be used to solve the current.

$$I = \frac{I_L + I_0 - \frac{V}{R_{ah}}}{1 + \frac{R_s}{R_{sh}}} - \frac{nNsV_{th}}{R_s} W(z) \tag{3}$$

3.2. Bishop's Algorithm

This algorithm uses an explicit solution that finds points on the IV curve by first solving for V_d where V_d is the diode voltage V_d=V+I∗Rs. Then the voltage is backed out from V_d. Points with specific voltage [8][9], such as open circuit, can be given by:

$$V_{oc,\text{est}} = nNsV_{th} \log\left(\frac{I_L}{I_0} + 1\right) \tag{4}$$

It`s known that V_d=0 corresponds to a voltage less than zero, and we can also show that when V_d=$V_{oc,\text{est}}$, the resulting current is also negative, meaning that the corresponding voltage must be in the 4th quadrant and therefore greater than the open circuit voltage (proof below) [10]. Therefore, the entire forward-bias 1st quadrant IV-curve is bounded because V_{oc}<$V_{oc,\text{est}}$.

$$\begin{gathered}
I = I_L - I_0\left(\exp\left(\frac{V_{oc,\text{est}}}{nNsV_{\text{th}}}\right) - 1\right) - \frac{V_{oc,\text{est}}}{R_{sh}} \\
I = I_L - I_0\left(\exp\left(\frac{nNsV_{th}\log\left(\frac{I_L}{I_0} + 1\right)}{nNsV_{th}}\right) - 1\right) - \frac{nNsV_{th}\log\left(\frac{I_L}{I_0} + 1\right)}{R_{sh}} \\
I = I_L - I_0\left(\exp\left(\log\left(\frac{I_L}{I_0} + 1\right)\right) - 1\right) - \frac{nNsV_{th}\log\left(\frac{I_L}{I_0} + 1\right)}{R_{sh}} \\
I = I_L - I_L - \frac{nNsV_{th}\log\left(\frac{I_L}{I_0} + 1\right)}{R_{sh}} \\
I = I_L - I_0\left(\frac{I_L}{I_0}\right) - \frac{nNsV_{th}\log\left(\frac{I_L}{I_0} + 1\right)}{R_{sh}} \\
I = -\frac{nNsV_{th}\log\left(\frac{I_L}{I_0} + 1\right)}{R_{sh}}
\end{gathered} \tag{5}$$

Bishop's method to calculate points on the IV curve with V ranging from the reverse breakdown voltage to open circuit voltage python codes are described below:

```
kwargs = {
      'breakdown_factor': parameters['breakdown_factor'],
      'breakdown_exp': parameters['breakdown_exp'],
      'breakdown_voltage': parameters['breakdown_voltage'],
   }
   v_oc = singlediode.bishop88_v_from_i(
```

```
        0.0, *sde_args, **kwargs
    )
    vd = np.linspace(0.99*kwargs['breakdown_voltage'], v_oc, ivcurve_pnts)
    ivcurve_i, ivcurve_v, _ = singlediode.bishop88(vd, *sde_args, **kwargs)
    return pd.DataFrame({
        'i': ivcurve_i,
        'v': ivcurve_v,

    })
```

4. CALCULATING A MODULE'S IV CURVES

Multiple methods are used to calculate the electrical parameters for an IV curve of a solar PV cell at a certain irradiance and temperature by using the module's base characteristics at standard test conditions (STC) conditions. Afterwards those electrical parameters are used to calculate the module's IV curve by solving the single-diode equation using the Lambert W method.

The single-diode equation is an equivalent electrical circuit (see Fig. 1) model of a PV cell with five electrical parameters that depend on the operating conditions [11, 12].

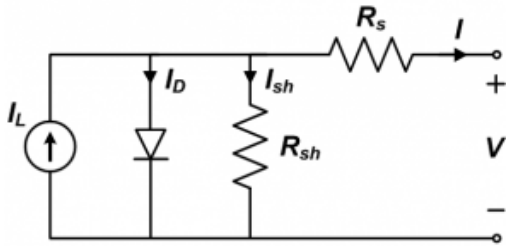

Figure 1. Equivalent circuit for a single solar cell [13].

To calculate the five electrical parameters of a single diode model we will use pvlib.pvsystem.calcparams_desoto() which is imported from Pvlib package, then we use pvlib.pvsystem.singlediode() to generate the IV curves. Modules reference parameters used i.e., the parameters at STC, theses parameters are:

Parameters:

- effective_irradiance (numeric) – The irradiance (W/m2) that is converted to photocurrent.
- temp_cell (numeric) – The average temperature of a cells within a module, measured in C.
- alpha_sc (float) – The temperature coefficient of short-circuit current of the module in units of A/C.
- a_ref (float) – The product of the normal diode ideality factor (n, unitless), quantity of cells in series (Ns), and cell thermal voltage at STC, in units of V.
- I_L_ref (float) – The light-generated current (or photocurrent) at STC, measured in amperes.
- I_o_ref (float) – The dark or diodes reverse saturation current at STC, measured in amperes.
- R_sh_ref (float) – The value of shunt resistance at STC, in ohms.
- R_s (float) – The value of series resistance of the cell at STC, in ohms.
- EgRef (float) – Energy bandgap at reference temperature in units of eV. 1.121 Ev.
- dEgdT (float) – Temperature dependence of the energy bandgap at stc in units of 1/K.
- irrad_ref (float (optional, default=1000)) – Reference solar irradiance in W/m2.

- temp_ref (float (optional, default at STC=25)) – Reference temperature of the cell measured in C.
- Then the function returns Tuple of the following results:
- photocurrent (numeric) – Light-generated current measured in amperes
- saturation_current (numeric) – Diode saturation curent in amperes
- resistance_series (float) – Series resistance in ohms
- resistance_shunt (numeric) – Shunt resistance in ohms
- nNsVth (numeric) – The product of the usual diode ideality factor (n, unitless), number of cells in series (Ns), and cell thermal voltage at specified effective irradiance and cell temperature.

Parameters for a PV module available in the lab are used, Python codes for adjusting the reference parameters according to STC are used. The IV curve is shown in Fig. 2 below:

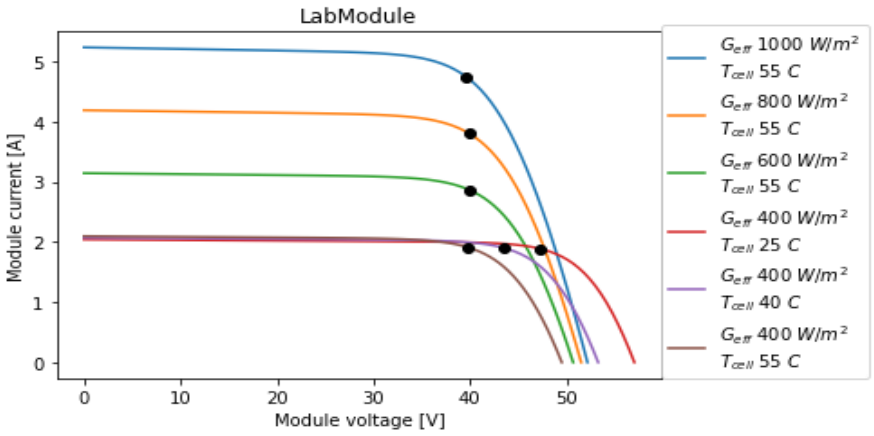

Figure 2. SD IV curve.

The above figure shows the effect of both Irradiance and cell temperature on the current and voltage of the cell, the dotted points at the knee of the curve are the values of current and voltage at maximum power point (MPP).

5. PARTIAL MODULE SHADING

For simplicity PV modelling is often done at the module level because it reduces the computational scope of the simulation. However, these types of simulations are poor to be able to module cell to cell partial shading. In this report cell-level IV curves are calculated and then combined to reconstruct the module-level IV curves. The module-level IV curves are used to find the maximum power of each cells combined under different shading and irradiance conditions. During the simulation we assume that shading only applies to beam irradiance (i.e., the same amount of diffuse irradiance is received by all cells) and the temperature of a cell is uniform and not affected by cell-level irradiance variation. The same parameters of the PV cell used previously in this report are assumed for calculating the shading effects on the module. First, IV curves for individual cells are calculated, the process is as follows:

- A single-diode model with given set of cell parameters at STC is used to calculate the single diode equation parameters for the cell at the operating conditions. De Soto model is used via pvlib.pvsystem.calcparams_desoto().

- In order to calculate reverse bias characteristic in addition to the forward characteristic of a single diode module Bishop '88 method (pvlib.singlediode.bishop88()) is used from Pvlib, this method This gives us a set of (V, I) points on the cell's IV curve.
- After calculating cell-level IV curves, then we compare a fully-irradiated cell's curve to a shaded cell's curve. One important points should be noted that shading typically does not reduce a cell's irradiation to zero – tree shading and row-to-row shading block the beam portion of irradiance but leave the diffuse portion largely intact. In this simulation plot, we assume that the shaded cell receives 350W/m2 as the amount of irradiance.

Fig. 3 shows the IV curve for both forward and reverse biased cell with Full-sun (i.e., Irradiation is 1000W/m2) cell and for shaded cell (i.e., Irradiation is 350W/m2).

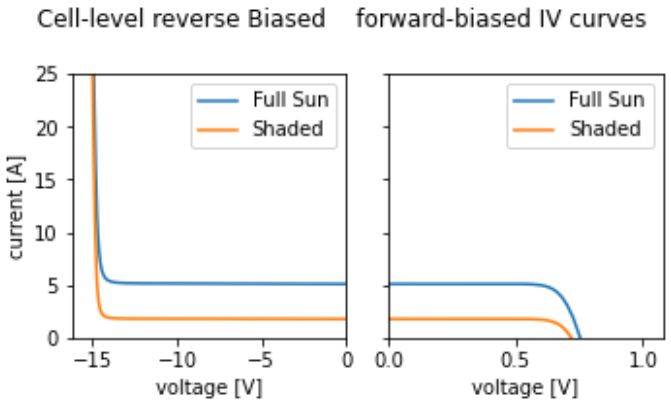

Figure 3. IV curve of Full-Sun and Shaded cell.

The above figure shows that a cell's current decreases roughly in proportion to the reduction in irradiance from shading, but voltage changes much less, comparing to Figure 2, the voltage changes proportionally with the temperature of the cell. It concluded from the above figure that the effect of shading is shifting the I-V current down to lower currents rather than change the curve's shape. I-V curves from Cell-Level simulation are combined to create a module I-V curve, cells in each substring are added in series, and the substrings are also connected in series and have a parallel bypass diode to protect from reverse biased voltages. Each module has 72 cells connected in series and normal crystalline silicon cell reaches only ~0.6V in forward bias, for convenience we will only assume that there are three types of cells (fully irradiated, fully shaded, and partially shaded), also it is assumed that the rest of the cells will behave identically within each of the three mentioned cell types. For simulation purpose we assume that the bottom 10% of the module is shaded. Assuming 12 cells per column, that means one row of cells is fully shaded and another row is partially shaded. Even though only 10% of the module is shaded, the maximum power is decreased by roughly 80%. Without using bypass diodes, operating the shaded module at the same current as the fully irradiated module would create a reverse-bias voltage of several hundred volts. Yet, the diodes prevent the reverse voltage from exceeding 1.5V (three diodes at 0.5V each). Module-Level IV curves are shown in Fig. 4.

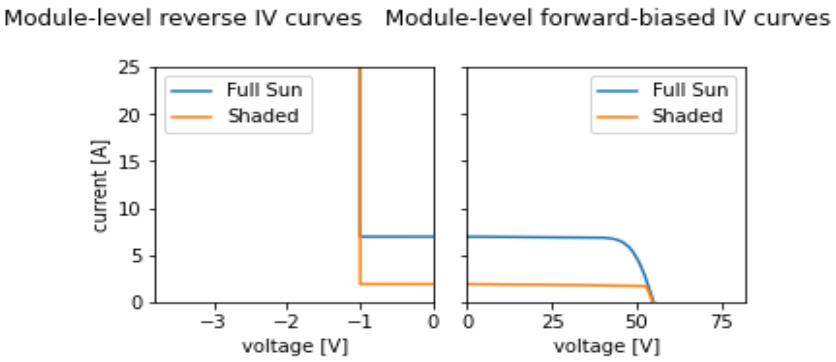

Figure 4. Module-Level IV curves.

6. CONCLUSION

In this article, a PV system modeling software package, PVLib, is used to calculate the power loss of PV modules from partial shading. In order to calculate the power loss of modules from partial shading, the IV curves for individual cells are calculated, by solving the single diode IV curve, then combined to create the IV curve of module-based system. It is concluded that using Python to investigate the performance of Photovoltaic cells is efficient in terms of time consuming when comparing with other simulation programs. When the algorithm is used for larger models, the simulation takes additional time and processor usage, however it is faster than other methods. Additionally, it is concluded that the algorithm performs efficiently in terms of time and processor usage when comparing to other algorithms and simulation software's, specially comparing to MATLAB.

REFERENCES

[1] Ransome , S., Stein, J., Holmgren, W., Sutterlueti, J., Pvpmc, T., & Performance, P. V. PV Performance Modelling with Pvpmc/Pvlıb introduction to Pvpmc/Pvlib why do we need the Pvpmc? Introduction To Python Pvlib - Open Source Toolbox Contributing to Pvlib Pvpmc Website Contents : PV Performance Modelling Steps Running Pvlib Python Scri, 2–5.

[2] Holmgren, W. F., Hansen, C. W., & Mikofski, M. A. Pvlib Python: a Python package for modeling Solar Energy Systems. *J. Open Source Software* 2018: *3*(29), 884. doi: 10.21105/joss.00884.

[3] McKinney, W. & Team, P. D. Pandas - Powerful Python data analysis toolkit. 1625, 2015.

[4] N. Community, NumPy User Guide 1.19. 214, 2020.

[5] SciPy Community, SciPy Reference Guide 0.7. 1229, 2013.

[6] Wall, L. et al. About the Tutorial Copyright & Disclaimer. 2, 2015. doi: 10.1017/CBO9781107415324.004.

[7] Corless, R. M., Gonnet, G. H., Hare, D. E. G., Jeffrey, D. J., & Knuth, D. E. On the Lambert W function. *Adv. Comput. Math.* 1996*: 5*(1), 329–359. doi: 10.1007/bf02124750.

[8] Bishop, J. W. Computer simulation of the effects of electrical mismatches in photovoltaic cell interconnection circuits. *Sol. Cells* 1988: *25*(1), 73–89. doi: 10.1016/0379-6787(88)90059-2.

[9] Izadian, A., Pourtaherian, A., & Motahari, S. Basic model and governing equation of solar cells used in power and control applications. 2012 IEEE Energy Conversion Congress and Exposition (ECCE), 2012, 1483-1488. doi: 10.1109/ECCE.2012.6342639.

[10] Hansen, C. Parameter Estimation for Single Diode Models of Photovoltaic Modules. 2015.

[11] Sado, K. A., Hassan, L. H., & Moghavvemi, M. Design of a PV-powered DC water pump system for irrigation: a case study. 2018 53rd International Universities Power Engineering Conference (UPEC), 2018, 1–6. doi: 10.1109/UPEC.2018.8542072.
[12] Sado, K. A. & Hassan, L. H. Modeling and Implementation of a PV-powered DC Water Pump System for Irrigation in Duhok City. *Acad. J. Nawroz Univ*ersity 2018: *7*(4). doi: 10.25007/ajnu.v7n4a294.
[13] Tamrakar, V., Gupta, S. C., & Sawle, Y. Single-Diode Pv Cell Modeling and Study of Characteristics of Single and Two-Diode Equivalent Circuit. *Electr. Electron. Eng. An Int. J.* 2015: 4, 13–24. doi: 10.14810/elelij.2015.4302.