PHOTOVOLTAIC SYSTEM MODELLING USING PVLIB-PYTHON

T. Gurupira[1], A .J. Rix[2]
Department of Electrical and Electronic Engineering, University of Stellenbosch, Private Bag X1, Matieland, 7602, South Africa
Email: [1]tafadzwalgurupira@gmail.com; [2] rix@sun.ac.za

Abstract

In this paper, a PV system modelling software package, PVLib, is analysed and validated. This is achieved by comparing PVLib simulation results of various modelling steps with those produced by a well-known commercially available package, PVSyst. Simulation results for the same site and conditions in both software packages were compiled and compared. Analysis of the results showed coherence between PVLib and PVSyst results, thereby attesting to the credibility of PVLib simulations. This outcome has great implications for the solar industry in Southern Africa because this confirms that not only is PVLib flexible, but that it can also be used to accurately model PV systems. The development of simulation models in PVLib will serve as the foundation for future work where optimisation algorithms will be implemented to maximize yield while reducing system cost. With the help of PVLib's modelling flexibility, expansion in research and analyses as well as innovation can take place.

Keywords: PVLib, PV system, modelling, yield, simulation, module

1. Introduction

Modelling flexibility is valuable in designing, planning and predicting the yield of photovoltaic power plants. Currently there is a number of software packages that can be used to model the performance of PV Systems, for both commercial and educational purposes. Most of these simulation programs however, have limitations. For starters, users often have limited access to algorithms and assumptions used in simulation programs such as PVSyst and PVSol [1]. It is also difficult for users to simulate more complicated systems than what the authors originally allowed for. Examples of such complicated systems range from multiple array orientations, differing PV panels and inverters and varied PV system designs. Some of the packages use default input limitations such as DC to AC power ratios and they are often not easy to change. Bug fixing in such programs is also time consuming, since it has to be done by the original authors and then redistributed [1]. PVLib has the potential to bridge this gap in PV system modelling. However, we first need to determine how well PVLib fares, compared to PVSyst, which is currently regarded as an industry standard [2]. The remainder of this article 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 thereof will be given in the subsequent sections.

2. Background of PVLib

The PVLib toolbox is a standard repository for high quality PV system modelling and analysis algorithms. The toolbox has the advantage of being continuously and collaboratively developed and validated. This allows not only for better accuracy, but also for quicker bug fixing. The code is open-source so the original source code is freely available, easy to modify and redistribute. PVLib is a product of the collaborative group of PV professionals, PV Performance Modelling Collaborative (PVPMC), facilitated by Sandia Laboratories. The main objective of the group is to improve accuracy and technical rigor in performance modelling and analysis. The PVLib toolbox was first developed for MATLAB but is now also available in python. PVLib-python [3], which is the version of PVLib we used in this evaluation, is a powerful alternative to the MATLAB package. PVLib-python also plays well to the strengths of the python programming language in that it is free to access, easy to install, open source and portable across platforms. PVLib-python is also designed for collaborative development and hosted on GitHub for access and revision control, bug tracking, feature requests as well as task and source code management [4] . In addition to that, PVLib-python is supported by a comprehensive testing and validation suite, which PVLIB MATLAB does not have [5]. For the rest of the paper, the term PVLib will be used to refer specifically to PVLib-python, unless otherwise stated.

2.1 Modelling steps

The five modelling steps use the following ten PVLib modules in the toolbox and these modules are briefly described as:

1. Weather and design - collection of weather data of the desired site and details of the orientation and set up of the array.
2. DC module IV characteristics - modelling of PV module behaviour according to its IV characteristics depending on the prevailing conditions and module model.
3. DC array IV - projecting the likely behaviour of PV modules are combined into an array; paying particular attention to DC wiring losses and mismatch effects.
4. DC to AC conversion - estimating DC-AC conversion efficiency using a variety of model algorithms. DC-AC conversion allows the solar power to be tied to the grid.
5. AC system output - determining and accounting for all the energy losses on the AC side before the utility meter.

These steps are incorporated by the PVLib toolbox using modular programming and the source code is grouped into ten modules namely: tools, tmy, location, solar position, atmosphere, modelchain, tracking, irradiance, clear sky and pv system[1] [6]. This code-level and modular approach to system modelling also allows users to model and analyse each portion of the PV system performance chain.

2.2 PVLib Module Summary

The ten PVLib modules in the toolbox are briefly described as:

1. The tmy module contains modules for reading site and weather data imported in TMY2 or TMY3 format.
2. The location module contains the location class which is comprised of the name, latitude, longitude, altitude and time zone data associated with a particular geographic location.
3. The atmosphere module contains methods to calculate relative and absolute air mass and to determine pressure from altitude, or altitude from pressure.
4. The solar position module contains a variety of methods to calculate solar position, for use in the PV System simulation process.
5. The clearsky module contains a number of methods to calculate clear sky Global Horizontal Irradiance (GHI), Diffuse Horizontal Irradiance (DHI) and Direct Normal Irradiance (DNI).
6. The irradiance module contains functions for modelling Angle Of Incidence (AOI), GHI, DHI and DNI under various conditions. This module features several decomposition and transposition models that the user can make use of in addition to the two that PVSyst (Hay-Davies and Perez [7]) uses.
7. The modelchain module contains functions and cases that combine many of the modelling steps, making getting started with PVLib relatively easy and demonstrating the standard ways in which the library can be used.
8. The pvsystem module contains functions for modelling the performance and the output of PV modules and inverters.
9. The tracking module contains modules for calculating PV system yield for arrays set up in single or dual axis tracking, not fixed orientation.
10. The tools module contains modules for trigonometric, time and mathematical calculations.

The flowchart in Fig. 1. gives a general workflow of a PVLib simulation discussed in section 2.1. The PVLib modules are set up sequentially to calculate the total yield of the PV system. Modelling steps 3 and 5 are not highlighted because they are concerned with the calculation of losses. For the validation of PVLib we only focussed our attention on the modelling steps 1, 2 and 4, out of the 5.

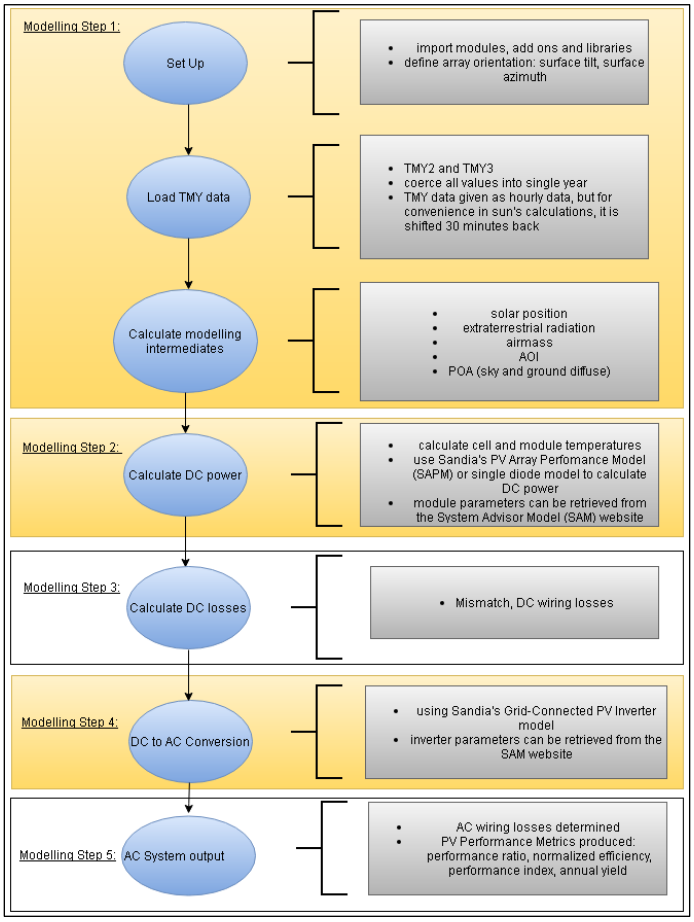

Fig. 1. PVLib workflow chart

3. Results and analyses

Simultaneous simulations were performed in PVLib and PVSyst for a location in Sandpoint, Alaska, USA, to check the credibility of PVLib. Our choice of location was influenced by the fact that we needed a location whose Typical Meteorological Year (TMY) [8] data was readily available since currently PVLib only supports weather data in TMY2 or TMY3 format [6]. The other input parameters were surface tilt, surface azimuth and albedo and their values are given in Table 1.The value of the surface azimuth in Table 1 is according to the PVLib convention given in Table 2.

In PVSyst however, if the site is in the Northern hemisphere, the plane azimuth is defined as the angle between South and the collector plane, and the system uses the antitrigonometric direction so the angle is taken as negative towards East. In the Southern Hemisphere, the plane azimuth is defined as the angle between north and the collector plane [9]. Since the site we simulated for is in the Northern hemisphere, our surface azimuth in PVSyst had to be set to 0°.

Input parameter	Value
Surface tilt	60 degrees
Surface azimuth	180 degrees
Albedo	0.2

Table 1: Parameters for the site and orientation of our PV arrays

Angle (in degrees)	Direction faced
0	North
90	East
180	South
270	West

Table 2: Surface azimuth convention

3.1 Simulation results

The results produced by each software package (PVSyst and PVLib) were compiled and then plotted on the same graph to compare the performance of the two packages. This section only features comparisons of the simulation results of some of the crucial modelling steps. Most of the results presented here are from a randomly selected summer week in Sandpoint, Alaska but some results for a randomly selected winter week are also included to show how consistent PVLib and PVSyst simulations were regardless of the season. The specific dates, the 5th of July till the 11th of July, were kept consistent for all summer comparisons so that any connections between some of the steps in the modelling chain could be quickly and easily spotted. The same was done for the winter results that were given. All the simulation results were in the form of hourly data starting from 00:00 on the chosen date and the format for the given dates is DD/MM.

3.1.1 Sun's height

The sun's height, defined as the angle between the sun's direction and the horizontal plane, was simulated and plotted as shown in Fig. 2. In PVSyst, it is calculated by the formula:

$$\begin{aligned}\mathrm{Sin(HSun)} = {} & [\mathrm{Sin(Lat)} * \mathrm{Sin(Decl)}] \\ & + [\mathrm{Cos(Lat)} * \mathrm{Cos(Decl)} \\ & * \mathrm{Cos(HA)}],\end{aligned} \tag{1}$$

where *HSun* is the sun's height, *Lat* is the observer's latitude; *Decl* is the declination angle and *HA* is the hourly angle. PVLib has a module that calculates solar angles such as the zenith and azimuth and the sun's height (angle of elevation) is given as:

$$\mathrm{HSun} = 90^0 - \mathrm{ZenSun}, \tag{2}$$

since it is complementary with the solar zenith angle (*ZenSun*) [10], [11]. The results of the sun's height simulations were consistent in both PVLib and PVSyst.

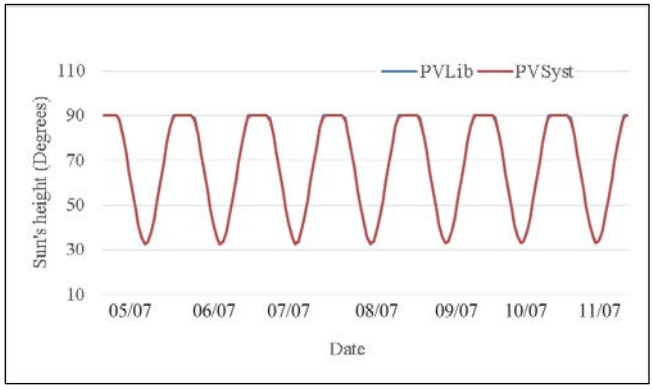

Fig. 2. Sun's height for a week in July

3.1.2 Angle of Incidence

The angle of incidence (AOI) values for the summer week were simulated in PVSyst and PVLib and the results were plotted on the same graph as shown in Fig. 3.

It was noted that the PVLib module that calculates the Angle Of Incidence (AOI), by default, includes obtuse angles and this is correct in its own right. Obtuse AOI's are however disregarded by the PVSyst AOI algorithm since PVSyst focuses

on the useful irradiance falling on the module surface. This is because an incident angle greater than 90° means the sun will be behind the surface and the surface will be shading itself [12]. As such, in PVSyst, all obtuse angles are also reduced to 90°, making the AOI range from 0° to 90°. For comparisons' sake we added an extra script in PVLib to also restrict AOI to the range 0° to 90° and the results of the new comparison are shown in Fig. 4. One should note that PVLib already uses a similar script to handle obtuse AOI's in all calculations that rely on AOI further down the modelling chain such as in the beam component calculations. The formula to calculate AOI is given as:

$$AOI = cos^{-1}\big[\cos(\theta_Z)\cos(\theta_T) + \sin(\theta_Z)\sin(\theta_T)\cos\big(\theta_A - \theta_{A,array}\big)\big], \quad (3)$$

where θ_A and θ_Z are the solar azimuth and solar zenith angles respectively and θ_T and $\theta_{A,\,array}$ are the tilt and azimuth angles of the array respectively.

The results were visibly coherent and there was a maximum percentage difference of 1% between them.

In the same manner, AOI results from a week in winter were collected from PVLib and PVSyst and plotted on the same graph, shown in Fig. 5. Fig. 5 shows that the simulation results from both software packages were consistent with each other in winter, save for during sunset and sunrise. These differences were due to the default sunset and sunrise hour settings in PVSyst [13]. The sunrise time in PVSyst on most days was noted to be later than PVLib's. The same trend was observed on most days with regards to the sunset time. PVSyst's sunset time was usually about an hour earlier than PVLib's.

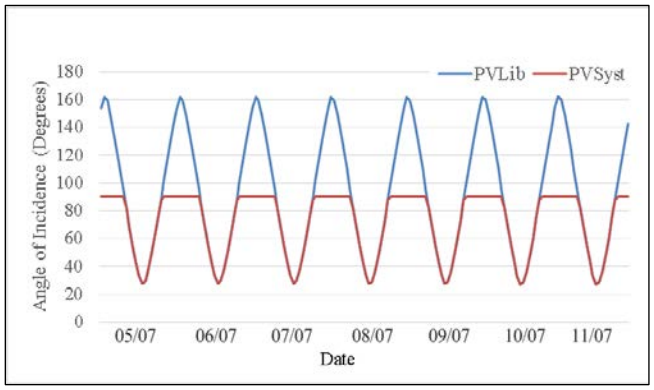

Fig. 3. Initial AOI simulation results for a week in July

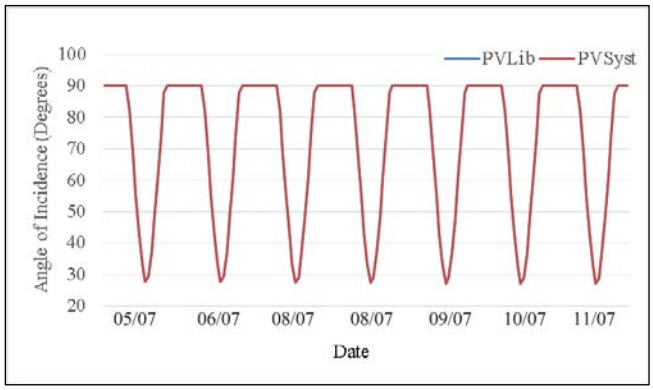

Fig. 4. AOI for a week in July (summer)

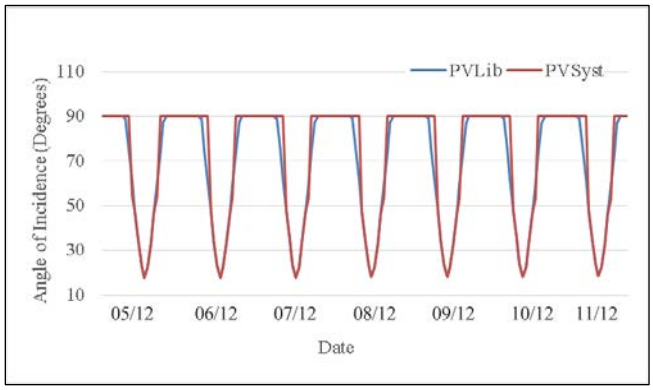

Fig. 5. AOI for a week in December (winter)

3.1.3 Plane of Array Irradiance (POA)

The plane of array (POA) irradiance was simulated on both PVSyst and PVLib platforms. The beam, sky diffuse and global POA components yielded by PVLib and PVSyst were pitted against each other and the results are summarised in this subsection.

Fig. 6 and Fig. 7 show simulation results of the Beam irradiance as given by PVSyst and PVLib in summer and winter respectively. In both seasons, as can be deduced from the two plots, the performance of PVLib was just as consistent with PVSyst's.

To determine the Sky Diffuse Irradiance, the Hay-Davies Transposition Model was used in both software packages [7] [14]. The formulation for the sky diffuse irradiation according to the Hay and Davies model is:

$$E_d = \mathrm{DHI} * [A_i \cos(\mathrm{AOI}) + (1 - A_i)\tfrac{1+cos\theta_T}{2}], \quad (4)$$

where *DHI* is the diffuse horizontal irradiance, *AOI* is the angle of incidence, and θ_T is the tilt angle of the array.

Fig. 8 and Fig. 9 show the comparison of PVLib and PVSyst results of the Sky Diffuse irradiance simulations for weeks in summer and winter respectively. The deduction from Fig. 8 and Fig. 9 are that the simulations in the two software packages are consistent with each other.

The POA Global irradiance was also simulated in PVSyst and PVLib for a week in summer and a week in winter as shown by Fig. 10 and Fig. 11 respectively.The plots of the POA Global results for both PVLib and PVSyst in summer and winter showed noticeable coherence.

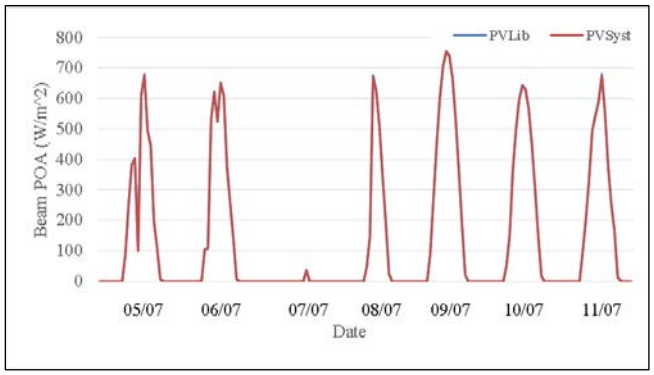

Fig. 6. POA Beam irradiance for a week in July (summer)

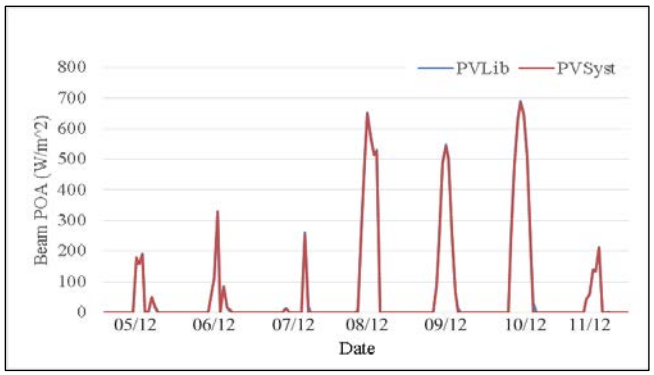

Fig. 7. POA Beam irradiance for a week in December (winter)

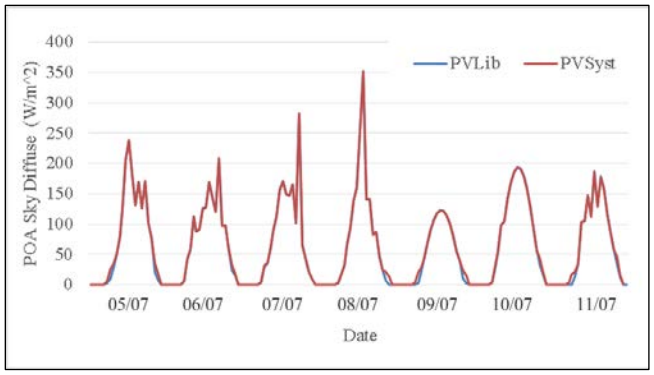

Fig. 8. Sky Diffuse POA irradiance for a week in July (summer)

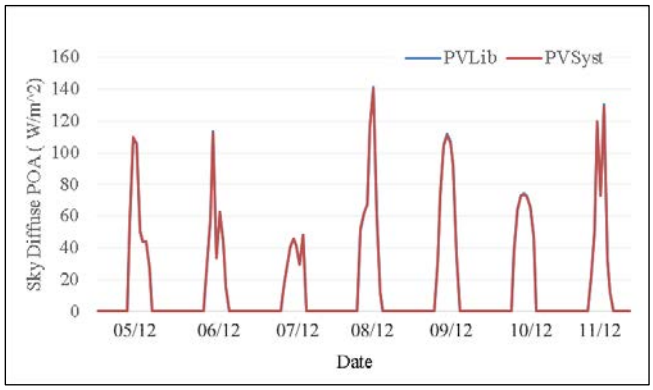

Fig. 9. Sky Diffuse POA Irradiance for a week in December (winter)

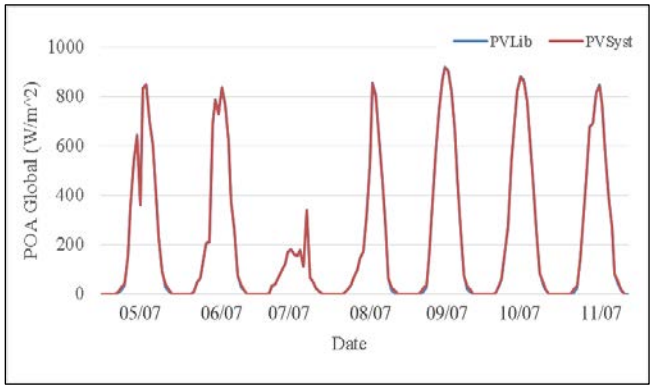

Fig. 10. POA Global Irradiance for a week in July (summer)

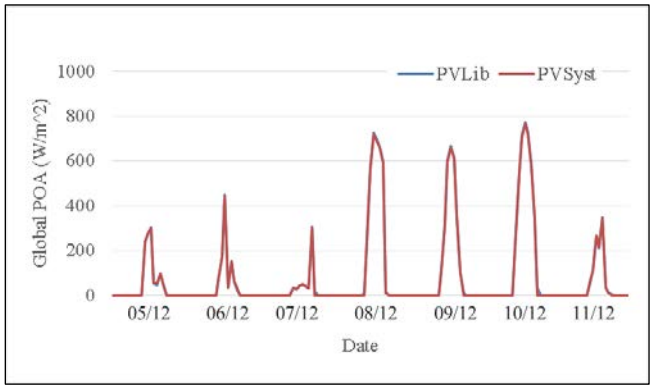

Fig. 11. POA Global Irradiance for a week in December (winter)

3.1.4 Array voltage and current

The current and voltage characteristics of the module strings were modelled in both PVLib and PVSyst and the results compared to each other. Both software packages use single diode equivalent circuit models to define the entire I-V curve of a cell, module or array as a function of environmental variables such as irradiance, temperature and spectral content

[15] [16]. PVLib uses the De Soto Single Diode Model [17] and PVSyst uses the PVSyst module model [18]. Fig. 12 and Fig. 13 show the summer and winter array voltage plots comparing the PVLib and PVSyst results.

The results show that the two software packages operate coherently for the most part, in both summer and winter. The notable differences on the voltage plots are due to the fact that PVSyst estimates its voltage curve based on the running time of the module or array [13], which in this case happens to start and end at different times as PVLib. The overlap, however, does not affect the yield, because current will not yet be flowing in the arrays so the will not be any DC power output.

The array currents from both packages were also compared to each other for both summer and winter weeks. Only the summer results are given, and they are shown in Fig. 14. Both packages performed coherently in summer and winter, with PVLib showing slightly higher peak values in both seasons. The difference in peak values can be attributed to the different single diode models used. The PVLib model tends to produce slightly more optimistic peak current values consistently.

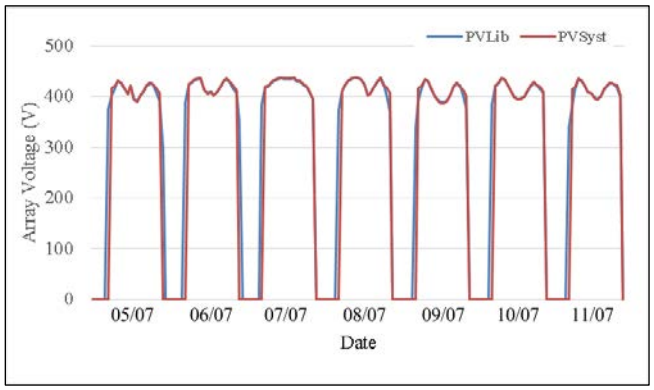

Fig. 12. Array Voltage for a week in July (summer)

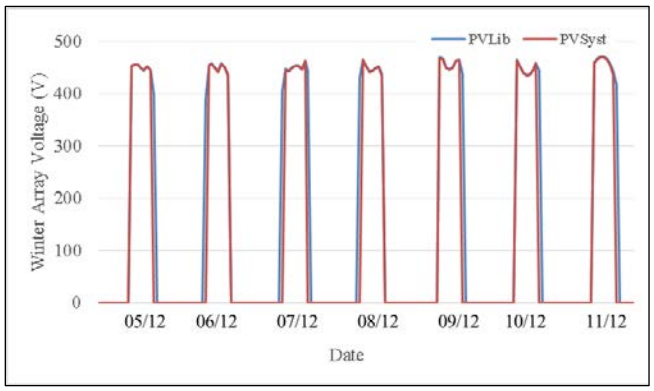

Fig. 13. Array Voltage for a week in December (winter)

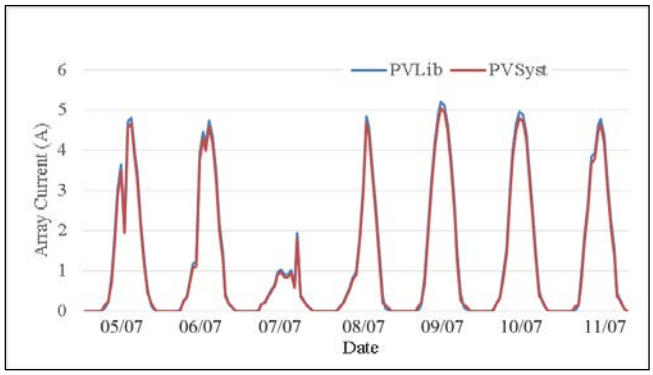

Fig. 14. Array Current for a week in July (summer)

3.1.5 Power

A single string with 11 SunPower 230W modules, in series, was used to simulate DC power output in PVLib and PVSyst and a comparison was drawn from the results. The choice of the PV module model was influenced by availability of its data in the PV module databases used by PVLib and PVSyst and also rankings from a solar panel comparison [19]. Using the single diode models mentioned in section 3.1.4, the direct current (DC) power for a week in summer was simulated in both PVLib and PVSyst. The results were plotted against each other to see how the software packages fared with each other in predicting the yield. PVLib also has a point-value method it uses to predict module and array I-V characteristics and DC power. The point-value model, Sandia PV Array Performance Model (SAPM) [20], was also used to simulate DC power output and its results were plotted together with the single diode model DC results.

The two software packages also use two different models to approximate the temperature of the PV cells. PVSyst uses the Fairman model [21] while PVLib uses the Sandia Cell (SAPM Temp) model [22]. The Fairman model is given by the formula:

$$T_m = \mathrm{T}_a + \frac{E_{POA}}{U_0+U_1+WS}, \tag{5}$$

where T_m is the module temperature, T_a is the ambient temperature, E_{POA} is the irradiance incident on the plane of the array or module U_0 is the constant heat transfer component and U_1 is the convective heat transfer component and WS is the wind speed. The SAPM temperature model is of the form:

$$\mathrm{T}_c = T_m + \frac{E_{POA}}{E_0} * \Delta T, \tag{6}$$

where T_c is the cell temperature, E_0 is the reference irradiance (1000W/m^2), and T_m, the module temperature, is given by:

$$T_m = E_{POA} * (e^{a+b*WS}) + T_a, \quad (7)$$

where a and b are parameters that depend on the module construction and materials, as well as the mounting configuration of the module.

Taking advantage of the fact that PVLib-python is open-source, we added a script that uses the Fairman model to calculate cell temperatures to our PVLib-python library the same way PVSyst calculates its cell temperatures. Fig. 15 shows results of the DC power simulations generated by the different DC power models, using different cell temperature models in the two software packages. All the PVLib results in the plot are indexed using, first the model used for the I-V characteristics, and then the temperature model used to determine the cell temperature. For the I-V model, SD denotes single diode, and SAPM denotes the Sandia PV Array Performance Model. As for the temperature models, SAPM is the Sandia cell temperature model and Fairman model is as is.

The results, as can be deduced from the plots, are very similar. The PVLib SD SAPM Temp, PVLib SAPM SAPM Temp and PVLib SD Fairman Temp plots show average percentage differences of 8, 7 and 5% respectively, compared to PVSyst results. A plot of one summer day's results, given in Fig. 16, shows these differences more visibly. From Fig. 16, it is evident that PVLib results are more optimistic compared to PVSyst. This can be attributed to the higher peak values as seen in the array current plots.

An SMA America Sunny Boy 3800TL US 22_208V inverter was used to simulate alternating current (AC) power output in PVLib and PVSyst and a comparison was drawn from the results. This particular inverter was chosen because it is available in the inverter databases used by PVLib and PVSyst and the inverter itself was compatible with the SunPower modules. Fig. 17 shows the comparison of the AC output simulations. Simulation results of the energy at the inverter output (representing the AC Power) in Fig. 17 show that the PVLib results did not follow the same trend as the DC values. The PVLib SD SAPM Temp, PVLib SAPM SAPM Temp and PVLib SD Fairman Temp AC results gave average percentage differences of 6, 8 and 10% compared to PVSyst AC power results. These are large differences so future work will include getting these results to match, using the same models.

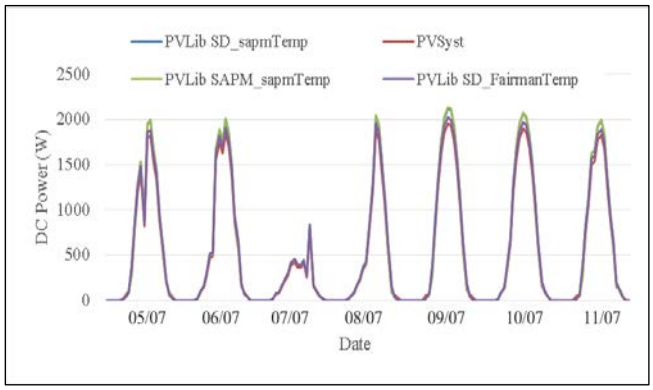

Fig. 15. DC Power Yield for a week in July (summer)

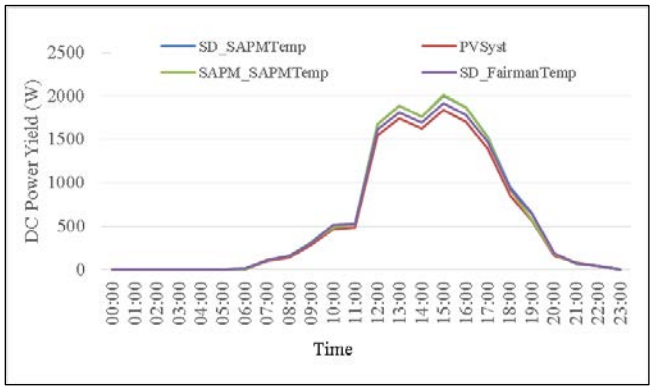

Fig. 16. DC Power Yield for 6 July

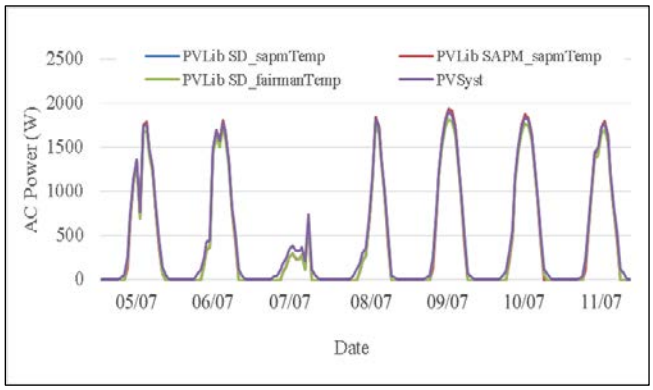

Fig. 17. AC Power Yield for a week in July (summer)

4. Conclusion

Although not to the same degree of accuracy as PVSyst, the results produced by PVLib showed consistency. This validates the credibility of PVLib and its competence for use in modelling PV System performance. Furthermore, some of the advantages of using PVLib were experienced first-hand during the process of simulation and analysis. An example is PVLib's code-level and modular approach to PV modelling that proved

to be particularly useful in the analysis of intermediate steps and their results. The open-source nature of PVLib also played a significant role in the evaluation of the results themselves as it allowed flexibility in the DC and AC yield modelling as well as in the presentation of AOI. Going forward, we will adapt PVLib for formats of weather data used widely in Southern Africa since currently it only supports the TMY format. For us, use of PVLib will serve as the foundation for future work where optimisation algorithms will be implemented to maximize yield while considering other constraints on system design. Accurately modelling PV system yield in PVLib will help lower financial risk and reduce financing costs and in turn ultimately lower electricity costs.

References

[1] S. Ransome, "PV PERFORMANCE MODELLING WITH PVPMC/PVLIB," in *2016 PVSAT12 Liverpool*, 2016.

[2] Tom Lombardo, "HelioScope: An Integrated Photovoltaic Design Tool," *Electronic Design*, 2014.

[3] R. W. Andrews, J. S. Stein, C. Hansen, and D. Riley, "Introduction to the open source PV LIB for python Photovoltaic system modelling package," *2014 IEEE 40th Photovolt. Spec. Conf. PVSC 2014*, pp. 170–174, 2014.

[4] W. F. Holmgren, R. W. Andrews, A. T. Lorenzo, and J. S. Stein, "PVLIB Python 2015," *2015 IEEE 42nd Photovolt. Spec. Conf. PVSC 2015*, 2015.

[5] "Comparison with PVLIB MATLAB — pvlib-python 0.3.3+26.gf281b0f.dirty documentation." [Online]. Available: http://pvlib-python.readthedocs.io/en/latest/comparison_pvlib_matlab.html.

[6] "PVLib-python Modules." [Online]. Available: http://pvlib-python.readthedocs.io/en/latest/modules.html.

[7] P. G. Loutzenhiser, H. Manz, C. Felsmann, P. A. Strachan, T. Frank, and G. M. Maxwell, "Empirical validation of models to compute solar irradiance on inclined surfaces for building energy simulation," *Sol. Energy*, vol. 81, no. 2, pp. 254–267, 2007.

[8] S. Wilcox and W. Marion, "Users manual for TMY3 data sets," *Renew. Energy*, no. May, p. 51, 2008.

[9] PVSyst, "Glossary > Plane azimuth." [Online]. Available: http://files.pvsyst.com/help/plane_azimuth.htm. [Accessed: 16-Apr-2016].

[10] ITACA.org, "Part 3: Calculating Solar Angles | ITACA." [Online]. Available: http://www.itacanet.org/the-sun-as-a-source-of-energy/part-3-calculating-solar-angles/.

[11] Sandia National Labratories, "PV Performance Modeling Collaborative | Sun Position." [Online]. Available: https://pvpmc.sandia.gov/modeling-steps/1-weather-design-inputs/sun-position/.

[12] itacanet.org, "Angle of incidence." [Online]. Available: http://www.itacanet.org/the-sun-as-a-source-of-energy/part-3-calculating-solar-angles/#3.5.-Angle-Of-Incidence.

[13] PVSyst, "sun-shields shadings." [Online]. Available: http://files.pvsyst.com/help/index.html?sunshields_shading.htm. [Accessed: 16-Jul-2016].

[14] Sandia National Laboratories, "PV Performance Modeling Collaborative | Hay and Davies Sky Diffuse Model." [Online]. Available: https://pvpmc.sandia.gov/modeling-steps/1-weather-design-inputs/plane-of-array-poa-irradiance/calculating-poa-irradiance/poa-sky-diffuse/hay-sky-diffuse-model/.

[15] Lindholm, J. G. Fossum, and E. L. Burgess, "Application of the superposition principle to solar-cell analysis," *IEEE Trans. Electron Devices*, vol. 26, 1979.

[16] Sandia National Laboratories, "PV Performance Modeling Collaborative | Single Diode Equivalent Circuit Models." [Online]. Available: https://pvpmc.sandia.gov/modeling-steps/2-dc-module-iv/diode-equivalent-circuit-models/. [Accessed: 05-May-2016].

[17] W. De Soto, S. A. Klein, and W. A. Beckman, "Improvement and validation of a model for photovoltaic array performance," *Sol. Energy*, vol. 80, no. 1, pp. 78–88, 2006.

[18] Sandia National Laboratories, "PV Performance Modeling Collaborative | PVsyst Module Model." [Online]. Available: https://pvpmc.sandia.gov/modeling-steps/2-dc-module-iv/diode-equivalent-circuit-models/pvsyst-module-model/.

[19] SRoeCo Solar, "Solar Panel Comparison Chart." [Online]. Available: http://sroeco.com/solar/learn-solar/solar-panel-comparison/. [Accessed: 08-Apr-2016].

[20] Sandia National Laboratories, "PV Performance Modeling Collaborative | Point-value models." [Online]. Available: https://pvpmc.sandia.gov/modeling-steps/2-dc-module-iv/point-value-models/. [Accessed: 05-May-2016].

[21] D. Faiman, "Assessing the Outdoor Operating Temperature of Photovoltaic Modules," *Prog. Photovolt Res. Appl.*, vol. 15, no. February 2013, pp. 659–676, 2007.

[22] Sandia National Laboratories, "PV Performance Modeling Collaborative | Sandia Module Temperature Model." [Online]. Available: https://pvpmc.sandia.gov/modeling-steps/2-dc-module-iv/module-temperature/sandia-module-temperature-model/. [Accessed: 03-May-2016].