Automated Formal Verification of Stand-alone Solar Photovoltaic Systems

Alessandro Trindade[a,*], Lucas Cordeiro[b]

[a] *Federal University of Amazonas, Av. Rodrigo Octvio, 6200, Coroado I, 69077-000 Manaus-AM-Brazil*
[b] *University of Manchester, School of Computer Science, Kilburn Building, Manchester M13 9PL*

Abstract

With declining costs and increasing performance, the deployment of renewable energy systems is growing faster. In 2017, for the first time, the number of people without access to electricity dropped down below 1 billion, but trends on energy access likewise fall short of global goals. Particular attention is given to stand-alone solar photovoltaic systems in rural areas or where grid extension is unfeasible. Tools to evaluate electrification projects are available, but they are based on simulations that do not cover all aspects of the design-space. Automated verification using model checking has proven to be an effective technique to validate complex (state transition) systems. This paper marks the first application of software model checking to formally verify the design of a stand-alone solar photovoltaic system, including solar panel, charge controller, battery, inverter, and electric load. Our main focus is on the project validation to be carried out just after the system sizing, i.e., prior to buying equipment and deployment, as a safe approach to ensure the intended behavior. Five case studies were used to evaluate this proposed approach and to compare that with specialized simulation tool. Different verification tools were evaluated to compare performance and soundness among automated verifiers. The results reported by our automated verification method and by the simulation tool were com-

*Corresponding author
Email address: alessandrotrindade@ufam.edu.br (Alessandro Trindade)

pared with data collected from dwellers of the deployed cases, thereby showing the effectiveness of our approach, where specific conditions that lead to failures in a solar photovoltaic system are only detailed by the automated verification method.
Keywords: Formal verification, model checking, photovoltaic power systems.

---

1. Introduction

Lack of access to clean and affordable energy is considered a core dimension of poverty (Hussein and Leal Filho, 2012). There is a close relationship between the lack of energy and the low HDI (Human Development Index) (Coelho et al., 2015). Progress has been made worldwide; in particular, the number of people without electricity access fell below 1 billion threshold for the first time in 2017 (IEA, 2018).

In order to provide universal electricity, decentralized systems led by solar photovoltaic (PV) in off-grid and mini-grid systems will be the lowest-cost solution for three-quarters of the additional connections needed; and grid extension will be the standard especially in urban areas (IEA, 2018).

Only a niche market a few years ago, solar PV systems are now becoming a mainstream electricity provider, with an increase of approximately 50% from 2016 to 2017 in terms of new installations of PV (EPIA, 2017).

In order to simulate or evaluate a PV system there are a myriad of specialized tools available in the market, such as RETScreen, HOMER, PVWatts, SAM, and Hybrid2 (Pradhan et al., 2015; Swarnkar et al., 2016; Dobos, 2014; Blair et al., 2014; Mills and Al-Hallaj, 2004); and even general purpose simulation tools such as PSpice, and MATLAB/Simulink package (Gow and Manning, 1999; Benatiallah et al., 2017). However, those tools are based on running experiments in simulation models. Simulation has the advantage of being cheap (if compared to test in real systems) but it has the drawback of an incomplete coverage since the verification of all possible combinations and potential failures of a system is unfeasible (Clarke et al., 2018).

Formal methods based on model checking offer a great potential to obtain a more effective and faster verification in the design process (Clarke et al., 2018). Any kind of system can be specified as computer programs using mathematical logic, which constitutes the intended (correct) behavior; then, one can try to give a formal proof or otherwise establish that the program meets its specification. In this study, a mathematical model of each component of a stand-alone PV system, as panel solar, charge controller, batteries, inverter, and electrical load is used. The project requirements, as battery autonomy and power demand, besides weather conditions, as solar irradiance and ambient temperature, are inputted to the proposed tool and automatically verified during the formal process. The model checking tool reports in which conditions a system does not meet the user requirements. A key benefit to this approach is that it helps in the detection of flaws in the design phase of system development, thereby considerably improving system reliability (Akram and Niazi, 2018). The implementation of the proposed tool is carried out by means of an algorithm in language C, that is executed by three state-of-the-art model checkers to formally verifying PV designs, in order to evaluate performance and correctness: the C Bounded Model Checker (CBMC) (Kroening and Tautschnig, 2014), the Efficient SMT-based Bounded Model Checker (ESBMC) (Gadelha et al., 2018), and the Configurable Program Analysis Checker (CPAchecker).

Note that in this study, our focus is not on a novel mathematical modelling of PV systems. Instead, our novelty relies on an effective approach to perform validation of PV systems using software model checking, in a process that is intended to be done just after the system sizing, prior to buying equipment and deploying it in the field, as a safe approach to ensure the intended behavior.

Here we compare model checking with simulation tools. However how they work is completely different. On one hand, simulation depends on the choice of input variables (and their values) in order to obtain the output. On the other hand, model checking performs an exhaustive search with the goal of proving or violating a given property. In the latter case, if the property is violated (an unexpected behavior of the system), then the model checker presents the input

that causes such a violation. Therefore, we can prove the absence of system flaws using model checking.

In prior studies, the evaluation of PV systems w.r.t. user requirements were performed by simulation tools using MATLAB/Simulink (Benatiallah et al., 2017; Natsheh and Albarbar, 2012), or HOMER Pro (Lamnadi et al., 2017), where some mathematical model was adopted and the tool simulated the PV behavior over the time. Related to formal methods, in 2015, an approach for applying Monte-Carlo simulation to power system protection schemes presented limitations of incomplete coverage of all possible operating conditions (Sengupta et al., 2015). The authors proposed an automated simulation-based verification technique to verify correctness of protection settings using hybrid automata-temporal-logic framework. In 2017, a researcher suggested the application of formal methods to verify and control the behavior of computational devices interacting over a shared-smart infrastructure (Abate, 2017). The author discussed the aggregation of large populations of thermostatically-controlled loads and of PV panels, and the corresponding problems of energy management in smart buildings and smart grids. The author used approximate model checking of stochastic and hybrid models. In 2018, a verification methodology was proposed with applications to PV panels and its distributed power point tracking (Driouich et al., 2018). This approach relied on representing unpredictable behavior of the environment to cover all possible feasible scenarios. The simulation results obtained by JModelica had evident time consuming issue with almost three days of computer effort to verify the design space of one operation hour of the PV panels behavior. Another work from 2018 was the approach to modeling smart grid components using a formal specification. The authors used a state-based formal specification language named Z; they demonstrated the application to four smart grid components (Akram and Niazi, 2018). This approach is based on Petri nets.

This paper makes two main contributions. Firstly, we propose an algorithm written in language C that implements the automated verification method which formally checks the sizing and the operation of a given stand-alone PV system.

Secondly, we evaluate the verification method by comparing three state-of-the-art model checkers in five real case studies.

Outline. Section 2 gives the background about solar PV systems, design and validation of PV systems, and the mathematical modeling. Section 3 presents the automated verification technique. The methodology is presented in section 4. Section 5 is devoted to the results. Section 6 presents the conclusion and describes future work.

2. Solar Photovoltaic System

PV systems are classified into distinct types (Mohanty et al., 2016). Specifically for remote rural areas of developing countries or places where the grid extension is not feasible, the most suitable configuration is the regulated stand-alone system with battery and AC load.

2.1. Sizing and Simulation of Stand-alone Solar PV systems

The sizing and validation of a PV system can be done by hand or with the aid of tools. Here we reference the critical period (Pinho and Galdino, 2014) as an effective method to stand-alone PV sizing.

The most popular softwares are summarized at Table 1: PVWatts, SAM, HOMER, RETScreen, and Hybrid2 (Pradhan et al., 2015; Swarnkar et al., 2016; Dobos, 2014; Blair et al., 2014; Mills and Al-Hallaj, 2004). As highlights, only HOMER and Hybrid2 perform off-grid system with battery backup analysis. Hybrid2 is not supported anymore, being in disuse since that not work on Windows platforms later than Windows XP (University of Massachusetts, 2016; Pinho and Galdino, 2014). Additionally, HOMER and RETScreen include economical analysis or even optimization-sensitive analysis. However, commercial version of those tools, called RETScreen Expert and HOMER Pro, are available only for Microsoft Windows and the annual subscription typically range from US$504.00 to US$657.00. In this study, HOMER was chosen to comparative with our proposed verification approach (Section 5.3).

Table 1: Comparative coverage of reference simulation software

Characteristic	PVWatts	SAM	HOMER	RETScreen	Hybrid2
Support	X	X	X	X	
Off-grid systems			X	X	X
Hybrid systems			X	X	X
Photovoltaics	X	X	X	X	X
Batteries			X		X
Main technical (T) or economical(E)	T	T	E	E	T
Optimization			X	X	
Sensitive analysis			X	X	

2.2. Component models for a stand-alone PV system

A stand-alone PV system is illustrated in Fig.1. The PV generator is a semiconductor device that can convert solar energy into DC electricity, with high dependence on two weather variables from the site, where the system is deployed: solar irradiance G and temperature T. For night hours or rainy days, power stored in batteries can be used and it implies the presence of a charge controller (Hansen et al., 2001). The PV arrays produce DC and therefore when the PV system contains an AC load, a DC/AC conversion is required (inverter). The AC load dictates the behavior of AC electrical load from the house that will be fed by the system. In this pictured modular structure, every element produces/consumes current I and voltage V, as illustrated by their physical magnitudes, where *pv* means photovoltaic, *bat* is battery, *dc* is direct, and *ac* is alternating signals.

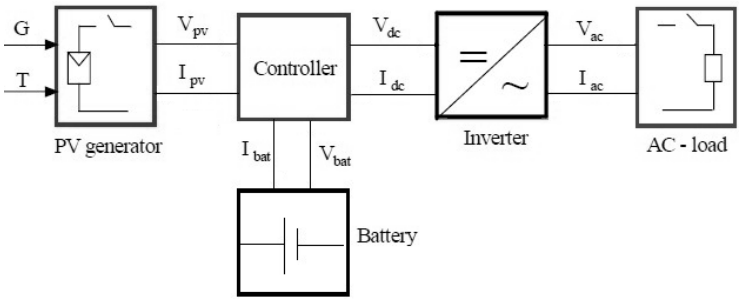

Figure 1: Block diagram for a typical stand-alone PV system, adapted from (Hansen et al., 2001).

2.3. PV System Model

Mathematical models are required in order to allow the formal specification of the PV system. At this section we will summarize the models adopted at the paper.

PV modules: a wide variety of models exists. However, the present work will rely on the simplified model of 1-diode, illustrated in Fig. 2. This model has a small error rate, between 0.03% and 4.68% from selected PV panels tested (Saloux et al., 2011).

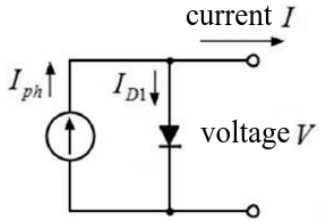

Figure 2: 1-diode equivalent PV cell/panel circuit model, adapted from (Cubas et al., 2017).

The Eq. (1) relates the output current, I, to the output voltage, V.

$$I = I_{ph} - I_{D1} = I_{ph} - I_0 \left[exp\left(\frac{V}{NaV_T} \right) \right], \tag{1}$$

where I_{ph} is the photocurrent delivered by the constant current source; I_0 is the reverse saturation current corresponding to the diode; N is the number of series-connected cells; a is the ideality or quality factor ($a = 1$ for ideal diodes and

between 1 and 2 for real diodes); V_T is the thermal voltage ($V_T = k_B T/q$); k_B is the Boltzmann constant ($1.3806503 \times 10^{-23}\, J/K$); T the temperature of cell in Kelvin; q is absolute value of the electron's charge ($-1.60217646 \times 10^{-19}\, C$).

The voltage and the current at the maximum power point tracking (MPPT), can be described by Eq. (2), and Eq. (3) (Saloux et al., 2011):

$$V_m = \frac{aNk_BT}{q} ln\left(\frac{aNk_BT}{qI_0}\frac{I_{sc}}{V_{oc}}\right). \tag{2}$$

$$I_m = I_{ph} + I_0 - \frac{aNk_BT}{q}\left(\frac{I_{sc}}{V_{oc}}\right). \tag{3}$$

However, the photo-current delivered by the constant current source (I_{ph}) or even the reverse saturation current (I_0) are not given by PV manufacturers. Therefore, Eq. (4) is used to calculate the photo-current as function of irradiance and temperature (Villalva et al., 2009):

$$I_{ph} = \frac{G}{G_{ref}}\left[I_{ph,ref} + \mu_I\left(T - T_{ref}\right)\right], \tag{4}$$

where the reference state (STC) of the cell is given by the solar irradiance $G_{ref} = 1000\, W/m^2$ and the temperature $T_{ref} = 298.15K (= 25^o\, C)$; μ_I is the short-circuit current temperature coefficient (A/K); $I_{ph,ref}$ can be approximated to the reference short-circuit current ($I_{sc,ref}$)(Villalva et al., 2009). The cell temperature (T) in degree Celsius is described by Eq. 5 (Ross, 1980):

$$T = T_{air} + \frac{NOCT - 20}{800}G, \tag{5}$$

where T_{air} is the ambient temperature, $NOCT$ is the nominal operating cell temperature (in ^{o}C), and G is the solar irradiance (W/m^2) of the place where the PV system is deployed.

Furthermore, Eq. (6) permits the saturation current (I_0) to be expressed as a function of the cell temperature as (Villalva et al., 2009)

$$I_0 = \frac{I_{sc,ref} + \mu_I(T - T_{ref})}{exp\left[\dfrac{q(V_{oc,ref} + \mu_V(T - T_{ref}))}{aNk_BT}\right] - 1}, \tag{6}$$

where $V_{oc,ref}$ is the reference open-circuit voltage and μ_V is an open-circuit voltage temperature coefficient (V/K).

Using the maximum power point current, cf. Eq. (3), and the saturation current in the reference temperature given by Eq. (6), the diode ideality factor (Saloux et al., 2011) is determined by Eq. (7):

$$a = \frac{q(V_{m,ref} - V_{oc,ref})}{N k_B T} \frac{1}{ln\left(1 - \frac{I_{m,ref}}{I_{sc,ref}}\right)}, \tag{7}$$

where V_{mref}, $V_{oc,ref}$, $I_{m,ref}$, and $I_{sc,ref}$ are key cell values obtained under both actual cell temperature and solar irradiance conditions.

Related to the **batteries**, various models have been described in the literature and the most common ones are based on lead-acid batteries (Copetti et al., 1993; Pinho and Galdino, 2014); that kind of battery has relative low cost and wide availability (Copetti et al., 1993). Here, the model adopted uses only manufacturer's data without empirical tests (Copetti et al., 1993). The discharge voltage is described by Eq. (8).

$$V_d = [2.085 - 0.12(1 - SOC)] - \frac{I}{C_{10}} \left(\frac{4}{1 + I^{1.3}} + \frac{0.27}{SOC^{1.5}} + 0.02\right) (1 - 0.007\Delta T), \tag{8}$$

where C_{10} means 10 h of rated capacity (manufacturer's data-sheet), ΔT is temperature variation ($\Delta T = T - T_{ref}$), SOC or state of charge indicates how much electric charge is stored in the cell at a given time. The depth of discharge (DOD) or the fraction of discharge, is $DOC = 1 - SOC$.

For the charging process, the parameters are described by Eq. (9) as

$$V_c = [2 + 0.16SOC] + \frac{I}{C_{10}} \left(\frac{6}{1 + I^{0.86}} + \frac{0.48}{(1 - SOC)^{1.2}} + 0.036\right) (1 - 0.025\Delta T). \tag{9}$$

The **charge controller** or controller is the responsible to manage the energy flow to PV system, batteries and loads by collecting information on the battery voltage and knowing the maximum and minimum values acceptable for

Table 2: Summary of the controller process (adapted from (Hansen et al., 2001))

Step	Constraint	Command
(1)	If $V > V_{max\_off}$ and $I_{load} < I_{pv}$	Disconnect PV array from the system
(2)	If command (1) is done and $V < V_{max\_on}$	Reconnect PV array to the system
(3)	If $V < V_{min\_off}$ and $I_{load} > I_{pv}$	Disconnect the load from the system
(4)	If command (3) is done and $V > V_{min\_on}$	Reconnect the load to the system

the battery voltage. Controllers with MPPT mechanism are the mostly used nowadays, and it maintains the PV operating at the stage of maximum power.

The steps in the modeling of the controller process are summarized in Table 2. To protect the battery against an excessive charge, the PV arrays are disconnected from the system, when the terminal voltage increases above a certain threshold V_{max_off} and when the current required by the load is less than the current delivered by the PV arrays (Hansen et al., 2001). PV arrays are connected again when the terminal voltage decreases below a certain value V_{max_on}. In order to protect the battery against excessive discharge, the load is disconnected when the terminal voltage falls below a certain threshold V_{min_off} and when the current required by the load is larger than the current delivered by the PV arrays (Hansen et al., 2001). The load is reconnected to the system, when the terminal voltage is above a certain value V_{min_on}.

The output power (P_{out}) of controller is given by Eq. (10).

$$P_{in}\eta_c = P_{out}. \tag{10}$$

Assuming that the efficiency of the controller (η_c) is a manufacturer's data, from Eq. (10) we compute Eq. (11).

$$V_{in}I_{in}\eta_c = V_{out}I_{out}, \tag{11}$$

where V_{in} is the voltage across the PV array, I_{in} is the output current of PV array, V_{out} is the DC bus voltage, and I_{out} is the output current from the converter.

The role of the **inverter** is to keep the voltage constant on the AC side, and to convert the input power P_{in} into the output power P_{out} with the best possible efficiency η_i as described by (12) (Hansen et al., 2001):

$$\eta_i = \frac{P_{out}}{P_{in}} = \frac{V_{AC} I_{AC} cos\varphi}{V_{DC} I_{DC}}, \tag{12}$$

where I_{DC} is the current required by the inverter from the DC source to be able to keep the rated voltage on the AC side, V_{DC} is the input voltage to the inverter delivered by the DC source (PV panel or battery), V_{AC} and I_{AC} are the output voltage and current, respectively, and $cos\varphi$ can be obtained from the inverter's manual.

2.4. Availability of Stand-alone PV Systems

The availability of a stand-alone PV system can be defined as the percentage of time at which a power system is capable of meeting the load requirements (Khatib and Elmenreich, 2014). The number of hours that the system is available, divided by 8,760 h, gives the annual system availability. The system availability definition depends on how critical the load application is. For critical loads, 99% is considered acceptable. While in a ordinary house electrical load, 95% is considered acceptable.

3. Automated Verification Using Model Checking

Although simulation and testing explore possible behaviors and scenarios of a given system, they leave open the question of whether unexplored trajectories may contain a flaw (Clarke et al., 2018). Formal verification conducts an exhaustive exploration of all possible behaviors; when a design is said to be "correct" by a formal verification method, it implies that all behaviors have been explored; questions regarding adequate coverage or missed behavior becomes

irrelevant (Clarke et al., 2012). Formal verification is a systematic approach that applies mathematical reasoning to obtain guarantees about the correctness of a system; one successful method in this domain is model checking (Clarke et al., 2012). Here we evaluate three state-of-the-art model checkers to formally verifying PV designs w.r.t. user requirements.

3.1. CBMC

The C Bounded Model Checker (CBMC) falsifies assertions in C programs or proves that they are safe if a completeness threshold is given (Kroening and Tautschnig, 2014). CBMC implements a bit-precise translation of a C program, annotated with assertions and with loops unrolled up to a given depth, into a logical formula. If the formula is satisfiable, then a failing execution that leads to a violated assertion exists (Kroening and Tautschnig, 2014). CBMC's verification flow can be summarized in three stages: (i) Front-end: scans, parses and type-checks C code; it converts control flow elements, such as *if* or *switch* statements, loops and jumps, into equivalent guarded *goto* statements, thus aiming to reduce verification effort; (ii) Middle-end: performs symbolic execution by eagerly unwinding loops up to a fixed bound, which can be specified by the user on a per-loop basis or globally, for all loops and finally; (iv) Back-end: supports SAT and SMT solvers to discharge verification conditions.

3.2. ESBMC

The Efficient SMT-based Bounded Model Checker (ESBMC) is a bounded and unbounded model checker for C programs (Gadelha et al., 2018), which supports the verification of LTL properties with bounded traces (Morse et al., 2015). ESBMC's verification flow can be summarized in three stages: (i) a front-end that can read and compile C code, where the system formal specification is first handled; (ii) preprocessing steps to deal with code representation, control flow and unwinding of loops, and model simplification, thereby aiming to reduce verification effort; and finally (iii) the SMT solving stage, where all constraints

and properties of the system are encoded into SMT and checked for satisfiability. ESBMC exploits the standardized input language of SMT solvers (SMT-LIB[1] logic format) to make use of a resource called *assertion stack* (Morse, 2015). This enables ESBMC, and the respective solver, to learn from previous checks, thus optimizing the search procedure and potentially eliminating a large amount of formula state space to be searched, because it solves and disregards data during the process, incrementally. This technique is called "incremental SMT" (Schrammel et al., 2017) and allows ESBMC to reduce the memory overhead, mainly when the verified system is complex and the computing platform does not have large amount of memory to deal with the entire design space state.

3.3. CPAchecker

Automatic program verification requires a choice between precision and efficiency. Historically, this trade-off was reflected in two major approaches to static verification: program analysis and model checking. In order to experiment with the trade-off, and in order to be able to set the dial between the two extreme points, Configurable Program Analysis (CPA) provides a conceptual basis for expressing different verification approaches in the same formal setting. The CPA formalism provides an interface for the definition of program analyses. Consequently, CPAchecker provides an implementation framework that allows the seamless integration of program analyses that are expressed in the CPA framework. The comparison among different approaches in the same experimental setting is intended to be easy and the experimental results are expected to be more meaningful (Beyer and Keremoglu, 2011). Related to the architecture, the central data structure is a set of control-flow automata (CFA), which consists of control-flow locations and control-flow edges. The CPA framework provides interfaces to SMT solvers and interpolation procedures (Beyer and Keremoglu, 2011).

[1] http://smtlib.cs.uiowa.edu/

4. Proposed Automated Verification Method of PV Systems

Note that the steps described here are carried out shortly after the design of the solar PV system, as soon as the equipment and its specifications are defined, i.e., before buying and deploying them, as a safe approach to ensure that the project will succeed.

The flowchart of the automated verification method is illustrated in Fig. 3.

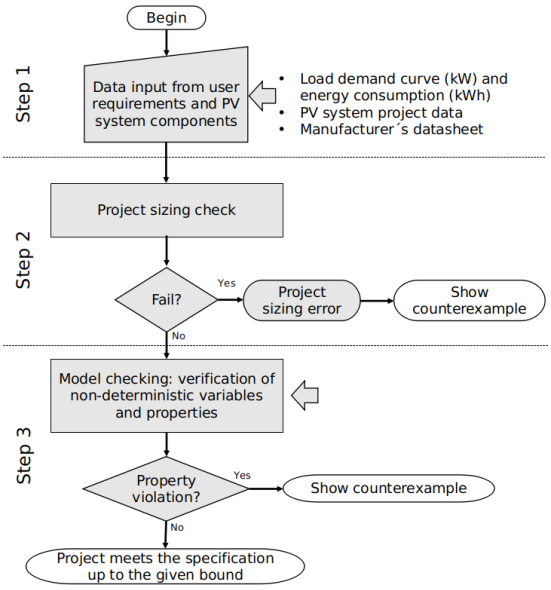

Figure 3: Flowchart of the proposed automated verification of PV systems.

In Step 1, the PV input data (e.g., load power demand and load energy consumption) and the formulae to check the sizing project, the mathematical model, the limits of the weather non-deterministic variables, are all written as an ANSI-C code (ISO, 2018). In Step 2, the sizing check of the PV system takes place: it will indicate if there is an error of sizing before to perform the automated verification of the system. This stage ensures that the system meets the standard project steps related to critical period method of sizing (Pinho and Galdino, 2014). In Step 3, weather variables (e.g., solar irradiance and ambient temperature) will be systematically explored by our verification engine based

on maximum and minimum values from the site, where the PV system will be deployed. In addition, depending on one of the desired properties of the system such as battery autonomy, energy availability, or even system power supply, our verification engine is able to indicate a failure if those properties are not met; in this particular case, it provides a diagnostic counterexample that shows in which conditions the property violation occurred.

In a nutshell, the model checker will process the ANSI-C code with constraints and properties from the PV system, and the tool will automatically verify if the PV system requirements are met. If it returns a failure (i.e., SAT), then the tool provides a counterexample, i.e., a sequence of states that leads to the property violation; this information can be used as a feedback to improve the PV system design. However, if the verification succeeds (i.e., UNSAT), there is no failure up to the bound k; therefore, the PV system will present its intended behavior up to the bound k.

Algorithm 1 describes the equivalent pseudo-code. In order to reduce the computational effort of the algorithm, every 24 h-day was considered as a time-step of 1 hour, and it was split into two parts: (a) one where it is possible to occur PV generation, during daylight, with a duration in hours depending on each site (but dependent on the sun and weather conditions); and (b) one that includes all the remaining day (without any PV generation), when the batteries are demanded to feed the house.

Lines 1 is devoted to information from the location where the PV system will be/were deployed. We use annual average minimum and maximum, related to temperature (T) and solar irradiance (G), hour by hour, from (Weatherbase, 2018), and (EnergyPlus, 2018).

Line 2 represents all the information that comes from the PV sizing and from the equipment manufacturers data: specification and data from PV, batteries, inverter and charge controller. This item includes as well information from the house's load curve.

In our study, we estimated load curves based on survey and visiting performed in June of 2017, and the the estimated load curve was converted as

vectors with 24 positions, one to each house of the day (in Watts):

- House 1: [118, 118, 118, 46, 46, 46, 95, 95, 170, 170, 296, 242, 242, 95, 95, 95, 95, 95, 342, 288, 288, 288, 288, 118];
- House 2: [136, 136, 136, 136, 136, 136, 67, 67, 184, 184, 184, 184, 184, 67, 67, 67, 67, 67, 253, 253, 253, 253, 253, 136];
- House 3: [113, 113, 113, 113, 113, 113, 67, 67, 217, 97, 97, 97, 97, 97, 97, 97, 97, 97, 263, 113, 113, 113, 113, 113];
- House 4: [207, 207, 207, 135, 135, 135, 66, 66, 161, 161, 233, 253, 248, 66, 66, 66, 66, 66, 302, 317, 322, 302, 302, 207];
- House 5: [45, 16, 16, 16, 16, 16, 0, 0, 0, 72, 72, 222, 150, 150, 0, 0, 72, 72, 814, 814, 814, 742, 742, 16].

The first automated verification is related to the sizing check (line 3), if an error is found then the algorithm stops. Then two functions, called at lines 4 and 5, are responsible for discover which hour starts the PV generation and when stops. Those functions get this information from the array inputted to the Algorithm with the solar irradiance values.

The batteries are assumed to be charged, i.e., with SOC of 100% (line 6).

The first for-loop at line 7 controls how many cycles of 24 h will be performed by the Algorithm. And the for-loop from lines 8 to 11 is responsible to discharge the battery (according the load curve) and verify the state of charge of the battery, hour-by-hour, starting at the first hour of the day after the sun goes down until the next day before the sun goes up (without PV generation). Following, at the next for-loop, from line 12 to 29, is performed the verification where there is solar irradiance and all the PV system works. The Algorithm generates information related to average temperature (T) and solar irradiance (G), hour-by-hour, using non-deterministic variables from model checker to explore all possible states and the *assume* macro to constrain the non-deterministic values using a given range (lines 15 and 16).

After that, the model from PV generator is used in the function call of line 17, to produce the voltage and current considering the states of G and T. With respect to every hour considered, the conditional *if-elseif-endif* statements from lines 18, 20, 22, 24 and 26, will imitate the charge controller work, performing the charge or discharge of batteries according to the value of different variables: if there is PV generation, the updated state of charge from batteries, the house's load and the set-up information of the PV system.

At the end of last for-loop, the state of the batteries is verified again (line 27) and the hour is adjusted to the next loop (line 28).

Nevertheless, if the verification engine does not fail, we can conclude that the PV system does not need further corrections up to the given bound k.

5. Verification and Simulation Results

5.1. Description of the Case Studies

We have performed five case studies to evaluate the proposed approach as described in Table 3. Power peak, power surge, and energy consumption were estimated based on visiting and survey applied to each house in June 2017 (before the electrification).

5.2. Objectives and Setup

We aim to answer two research questions:

RQ1 **(soundness)** Does our automated verification approach provide correct results?

RQ2 **(performance)** How do the verifiers compare to each other and to a simulation commercial tool?

All experiments were conducted on an otherwise idle Intel Xeon CPU E5-4617 (8-cores) with 2.90 GHz and 64 GB of RAM, running Ubuntu 16.04 LTS 64-bits. The setup of HOMER Pro v3.12.0: Intel Core i5-4210 (4-cores), with 1.7 GHz and 4 GB of RAM, running Windows 10. The experiments were performed with timeout of 14,400 seconds.

Algorithm 1 Model checking algorithm for stand-alone PV

```
1: declare min and max solar irradiation[24h], and temperature[24h]
2: declare case studies details : sizing and manufacturers data
3: sizing_check()
4: startPVgeneration ← findStartPVgeneration()
5: endPVgeneration ← findEndPVgeneration()
6: SOC ← 100%
7: for 1st 24h loop to Nth 24h loop do
8:     for endPVgeneration + 1 to startPVgeneration − 1 do
9:         dischargeBattery in 1h()
10:        assert(SOC ≥ SOC_min)
11:    end for
12:    for startPVgeneration to endPVgeneration do
13:        G ← nondet_uint( ) {G is non-deterministic variable}
14:        T ← nondet_uint( ) {T is non-deterministic variable}
15:        assume (Gmin ≤ G ≤ Gmax) {restricting G values}
16:        assume (Tmin ≤ T ≤ Tmax) {restricting T values}
17:        Imax, Vmax ← PVgenerationMODEL(G, T)
           {If-then-else sequence to imitate charge controller work}
18:        if (battery is empty) AND (PV is generating) then
19:            chargeBattery in 1h() {PV feed the house}
20:        else if (battery is empty) AND NOT(PV is generating) then
21:            FAIL with assert macro {Battery is empty and there is not PV generation}
22:        else if NOT(battery is empty) AND (PV is generating) then
23:            stop battery charge {PV feed the house}
24:        else if NOT(battery is empty) AND NOT(PV is generating) then
25:            dischargeBattery in 1h() {Battery feed the house}
26:        end if
27:        assert(SOC ≥ SOC_min)
28:        hour ← hour + 1
29:    end for
30: end for
31: return ( )
```

Table 3: Case studies: stand-alone solar PV systems.

Item	House 1	House 2	House 3	House 4	House 5
PV Panels	3×325 W: (3S)				4×325 W: (2S-2P)
Batteries	4×220 Ah: (2S-2P) autonomy: 48 h				4×120 Ah: (4S) autonomy: 6 h
Charge Controller	With MPPT of 150 V/35 A				
Inverter	700 W, surge: 1,600 W				1,200 W, surge: 1,600 W
Power peak (W)	342	253	263	322	814
Power surge (W)	342	722	732	896	980
Consumption (kWh/day)	3.9	3.6	2.5	4.3	4.88
GPS Coordinates	2^o44'50.0"S 60^o25'47.8"W				3^o4'20.208"S 60^o0'30.168"W
Details	Riverside indigenous community Rural Area of Manaus - Brazil				Urban house Manaus-Amazonas-Brazil

Legend: (S): Series; (P): Parallel.

Verification engine ESBMC, version v6.0.0 was used with the SMT solver Boolector version 3.0.1 (Brummayer and Biere, 2009)[2]; and an alternative ESBMC v6.0.0 was used with the SMT incremental mode[3] enabled; with SMT solver Z3 version 4.7.1 (Moura and Bjørner, 2008).

Verification engine CBMC 5.11 and MiniSat 2.2.1 were used in the comparison (Kroening and Tautschnig, 2014)[4].

Verification engine CPAchecker 1.8 was used [5], with the SMT solver MathSAT version 5.5.3 (Cimatti et al., 2013). An alternative CPAchecker configuration was tried as well, using BMC k-induction option, but without improvements of performance or soundness.

Note that the results presented here depend on the computer's processor and memory, the version of each software verifier (i.e., CBMC, ESBMC,

[2]Command-line: $ esbmc filename.c --no-bounds-check --no-pointer-check --unwind 100 --boolector

[3]Command-line: $ esbmc filename.c --no-bounds-check --no-pointer-check --unwind 100 --smt-during-symex --smt-symex-guard --z3

[4]Command-line: $ cbmc filename.c --unwind 100 --trace

[5]Command-line: $ scripts/cpa.sh -heap 64000m -stack 10240k -config config/bmc-incremental.properties -spec config/specification/sv-comp-reachability.spc filename.c

Table 4: Summary of the case-studies comparative and the automated tools.

Model Checker (SAT/UNSAT: time and message)				
Case	ESBMC 6.0.0 (Boolector 3.0.1)	ESBMC 6.0.0 (Z3 4.7.1)	CBMC 5.11 (MiniSat 2.2.1)	CPAchecker 1.8 (MathSAT 5.5.3)
House 1	Out of memory (UNKNOWN)	05 m 08 s (UNSAT)	19 m 02 s (UNSAT)	Time out (UNKNOWN)
House 2	Out of memory (UNKNOWN)	04 m 27 s (UNSAT)	18 m 59 s (UNSAT)	Time out (UNKNOWN)
House 3	Out of memory (UNKNOWN)	05 m 07 s (UNSAT)	18 m 39 s (UNSAT)	Time out (UNKNOWN)
House 4	Out of memory (UNKNOWN)	04 m 37 s (UNSAT)	18 m 36 s (UNSAT)	Time out (UNKNOWN)
House 5	Out of memory (UNKNOWN)	$\leq$ 1 sec (SAT Line 337)	$\leq$ 1 sec (SAT Line 337)	6 sec (SAT line 337)

CPAchecker), the parameters passed by the command-line, and the implemented model (software created by authors) to be solved. Additionally, any change from HOMER Pro to another simulation tool can also influence the measured time to obtain results.

5.3. Results and Discussion

Table 4 summarizes the results. The times reported in Table 4 answer RQ2. Note that an UNKNOWN result from our verification engines does not mean that a failure was found neither that the verification is successful: it indicates that the verification engine led to an *out of memory* or a *time out* situation.

The description of our results can be broken down into three parts, one for each verification engine: ESBMC, CBMC, and CPAchecker.

Related to ESBMC, we have tried two possibilities: one with Boolector and another one with Z3. The incremental option, which uses less memory, can be performed with Z3 only since ESBMC does not support the incremental mode with Boolector yet. Using ESBMC with Boolector led to an out of memory situation in all the case studies. This result was obtained in less than six minutes of execution, i.e., the 64 GB of RAM were consumed by the verification engine and the processes were killed, thus leading to an UNKNOWN result returned by ESBMC as shown in the first column of Table 4. However, running the same

version of ESBMC but using incremental solving with Z3, the experimentation returned SAT or UNSAT to all the case studies. Related to the cases that use a 700 W PV system, ESBMC could not reach an error in all the four houses and the execution time took from 04 m 27 s to 05 m 08 s. However the 1,200 W PV system (house 5) failed (SAT) in line 337 of the code, thereby indicating that the system is *incorrectly* sized; in particular, the counterexample provided by the verification engine indicated that the nominal current from the charge controller is less than the minimum current demanded by the PV system, therefore the equipment chosen is not suitable to meet the design requirements. This verification took less than 1 s to be performed, as indicated in the last line of Table 4, and it is faster than the previous analysis because ESBMC stops during the sizing check, which is in line 3 of Algorithm 1, and does not perform the rest of verification code.

Concerning the CBMC tool, similar results were obtained, but with some slower time. The experimentation returned SAT or UNSAT to all the case studies. Related to the 700 W PV systems, the tool could not reach an error in all the four houses and the execution time took from 18 m 36 s to 19 m 02 s. However the 1,200 W PV system (house 5) failed (SAT) in line 337 of the code; with the same counterexample presented by ESBMC. This verification took less than 1 s to be performed as well.

Finally, the CPAchecker tool presented some different results. Even using two different configuration possibilities, as described in Section 5.2, the verification engine presented an UNKNOWN result for all the 700 W systems. This is because the *time out* limit was reached, i.e., after 4 hours of execution the tool was unable to decide if the verification was SAT or UNSAT. However, when verifying the 1,200 W PV system, the tool presented a SAT message equal to the other engines.

In order to validate the possible flaw from house 5, we have surveyed the owner of the 1,200 W system. We identified that, in fact, the system does not meet the battery autonomy when all loads are turned on, and this was double checked with the monitoring system from the charge controller, which showed

that the maximum power or surge power were not exceeded, thus affirming RQ1; this behavior is expected since the system was purchased as an off-the-shelf solution and not as a customized design for the electrical charges of the house. The same process of validation was done to the houses 1, 2, 3 and 4, which use the 700 W PV systems: from July of 2018 to March 2019, a monthly visiting was performed to apply surveys to the dwellers and to collect data from a local monitoring system: not every month were reported some energy interruption of the PV systems. However, even when one interruption is reported in a month, this represents around 3.33% of interruption for the entire period (1/30), which indicates 96.97% of availability of the PV system and it is in accordance to what was described in Section 2.4, because the type of electrical load of the houses is not critical; this situation is considered an energy interruption, but is not considered a system flaw, further affirming RQ1.

The same five case studies were evaluated by HOMER Pro (RQ2). The simulation results showed that the project restrictions were met by four 700 W PV systems (house 1, 2, 3 and 4), without any indication of sizing error or even performance related issues. The case study that was unsuccessful during simulation was the 1,200 W (house 5); however, without any indication about the failures of this PV system (RQ2). All the simulations took less than 5 seconds (each) to be performed by HOMER Pro.

There were no divergence of results for the houses 1, 2, 3 and 4 w.r.t. our proposed approach, it is evident that the information collected from the dwellers and from the monitoring systems indicate that our approach provides the correct evaluation of the PV system, thus answering RQ2. House 5 presented flaws from all tools (automated verified or simulation); however, only automated verification approaches indicated which design error was responsible for the flaw (charge controller specification), further answering RQ2.

5.4. Threats to Validity

We have reported a favorable assessment of the proposed method. Nevertheless, we have also identified five threats to the validity of our results that can

further be assessed.

Model precision: each component of the PV system is mathematically modeled. The adoption of more complex models, or even an evaluation in a PV laboratory to validate the model could add more reliability to the results.

Time step: The run-time complexity of our proposed method is an issue; the time step of one hour can be further reduced to approximate the algorithm to the real-world scenario.

Case studies: Our case studies are performed only in one municipality. A more complete evaluation can be performed with more case studies.

Simulation Tool: Only HOMER Pro was used. The inclusion of other specialized simulation tool or even a general simulation tool that uses the same mathematical model adopted by the automated verification could change the comparative.

Temperature and Solar Irradiance Data: Information related to temperature and solar irradiance of each case study, independently of using simulation or formal verification, come from databases available online (Weatherbase, 2018; EnergyPlus, 2018). However, considering that riverside communities do not have weather stations, the data used in our study come from the closest municipality (Manaus in all case studies), where stations collect regularly those data. Therefore, the most accurate should be the use of weather stations in each location.

6. Conclusions and Future Work

We have described and evaluated an automated verification method to check whether a given PV system meets its specification using software model checking techniques. Our main focus was on the method to validate PV system projects, based on software model checking, which can cover the design-space better than simulation tools. We have considered five case studies, ranging from 253 W to 814 W; three state-of-art verification engines were considered (ESBMC, CBMC, and CPAchecker); and one specialized simulation tool (HOMER Pro). The ex-

perimental results from automated verification and simulation tools were compared with information collected from dwelles of the sites, where the real PV systems were deployed. Although the verification method proposed takes longer than simulation methods, it is able to present details that lead to failures in a PV system that is not a feature presented in commercial simulation tool. In particular, the proposed method was successful in finding sizing errors and indicating in details. Related to the verification engines comparative, the ESBMC with the Z3 solver executed in the incremental configuration presented the better performance (around four times faster than CBMC), used less RAM memory (less than 2 GB when compared to 9.2 GB of CBMC and 19.2 GB of CPAchecker), and all the results were sound because the PV owners and the monitoring system validated the possible flaws that the system could be presenting in the field. As future work, we will expand the number of case studies (preferentially in different countries), develop the code of a general purpose simulation tool to include in the comparative, and also consider other types of renewable energy and even hybrid ones to allow our method to verify typical rural electrification.

Acknowledgments

The authors would like to thank: (i) Coventry University for the case studies and for the research mobility; (ii) Newton Fund [grant number 261881580] for the research mobility support; (iii) FAPEAM - Amazonas State Foundation for Research Support [grants 009/2017 and PROTI-Pesquisa 2018], for the Virtual Machine rent (DEC-2018 to JAN-2019), for the DCTIEX II scholarship (NOV-2018), and for support the research mobility to UK; and (iv) FAS - Sustainable Amazonas Foundation for the PhD scholarship (NOV-2017 to MAR-2019).

Abate A. Verification of networks of smart energy systems over the cloud. In: Bogomolov S, Martel M, Prabhakar P, editors. Num. Soft. Verif. volume LNCS 10152; 2017. p. 1–14. doi:10.1007/978-3-319-54292-8.

Akram W, Niazi MA. A formal specification framework for smart grid com-

ponents. Complex Adaptive Systems Modeling 2018;6(1):5. doi:10.1186/s40294-018-0057-3.

Benatiallah A, Benatiallah D, Ghaitaoui T, Harrouz A, Mansouri S. Modelling and simulation of renewable energy systems in Algeria. Int J of Sc and App Inf Tech 2017;7(1):17–22.

Beyer D, Keremoglu ME. CPAchecker: A tool for configurable software verification. In: Gopalakrishnan G, Qadeer S, editors. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg; volume LNCS 6806; 2011. p. 184–90. doi:10.1007/978-3-642-22110-1_16.

Blair N, Dobos A, Freeman J, Neises T, Wagner M. System Advisor Model, SAM 2014.1.14: General Description. Technical Report; National Renewable Energy Laboratory; Colorado; 2014.

Brummayer R, Biere A. Boolector: An efficient SMT solver for bit-vectors and arrays. In: Tools and Alg. for the Const. and An. of Sys. (TACAS). volume LNCS 5505; 2009. p. 174–7. doi:10.1007/978-3-642-00768-2_16.

Cimatti A, Griggio A, Schaafsma B, Sebastiani R. The MathSAT5 SMT Solver. In: Piterman N, Smolka S, editors. Proceedings of TACAS. Springer; volume 7795 of *LNCS*; 2013. p. 93–107.

Clarke E, Klieber W, Miloš Nováček , Zuliani P. Model Checking and the State Explosion Problem; Berlin: Springer. p. 1–30. doi:10.1007/978-3-642-35746-6_1.

Clarke EM, Henzinger TA, Veith H. Introduction to model checking. In: Handbook of Model Checking. Springer; 2018. p. 1–26. doi:10.1007/978-3-319-10575-8_1.

Coelho S, Sanches-Pereira A, Tudeschini L, Escobar J, Poveda M, Coluna N, Collin A, Rovere EL, Trindade A, Pereira O. Biomass residues as electricity generation source in low HD source in regions of Brazil. In: UNESP , editor.

The XI Latin. Cong. of Elec. Gener. and Transm. CLAGTEE. 2015. p. 1–8. doi:10.13140/RG.2.1.2747.7208.

Copetti J, Lorenzo E, Chenlo F. A general battery model for PV system simulation. Prog in Photovoltaics: Res and App 1993;1(4):283–92. doi:10.1002/pip.4670010405.

Cubas J, Pindado S, Sorribes-Palmer F. Analytical calculation of photovoltaic systems maximum power point (MPP) based on the operation point. Applied Sciences 2017;7(9):870–84. doi:10.3390/app7090870.

Dobos A. PVWatts Version 5 Manual. Technical Report; National Renewable Energy Laboratory; Colorado; 2014.

Driouich Y, Parente M, Tronci E. A methodology for a complete simulation of cyber-physical energy systems. In: IEEE Work. on Envir., Energ., and Struc. Monit. Syst. (EESMS). 2018. p. 1–5. doi:10.1109/EESMS.2018.8405826.

EnergyPlus . Weather data by location. https://energyplus.net/weather-location/south_america_wmo_region_3/BRA//BRA_AM_Manaus-Gomez.Intl.AP.817300_INMET; 2018. [Acessed 09 July 2018].

EPIA . Global market outlook for photovoltaics 2017-2021. Belgium: European Phototoltaic Industry Association, 2017.

Gadelha M, Monteiro F, Morse J, Cordeiro L, Fischer B, Nicole D. ESBMC 5.0: An industrial-strength C model checker. In: 33^{rd} ACM/IEEE Int. Conf. on Aut. Soft. Engin. (ASE'18). New York, NY, USA: ACM; 2018. p. 888–91. doi:10.1145/3238147.3240481.

Gow J, Manning C. Development of a photovoltaic array model for use in power-electronics simulation studies. In: Proceedings of the 14th IEE Electric Power Applications Conference. volume 146(2); 1999. p. 193–200. doi:10.1049/ip-epa:19990116.

Hansen A, Sørensen P, Hansen L, Bindner H. Models for a stand-alone PV system. Number 1219 in Denmark. Forskningscenter Risoe. Risoe-r. Forskningscenter Risoe, 2001.

Hussein M, Leal Filho W. Analysis of energy as a precondition for improvement of living conditions and poverty reduction in sub-Saharan Africa. In: Scientific Research and Essays. volume 7(30); 2012. p. 2656–66. doi:10.5897/SRE11.929.

IEA . World Energy Outlook 2018. Paris: IEA, 2018.

ISO . ISO/IEC 9899:2018 Information Technology – Programming Languages – C. https://www.iso.org/standard/74528.html; 2018. [Acessed 14 November 2018].

Khatib T, Elmenreich W. Optimum availability of standalone photovoltaic power systems for remote housing electrification. Int Journal of Photoenergy 2014;(Article ID 475080):5 pages. doi:10.1155/2014/475080.

Kroening D, Tautschnig M. CBMC C Bounded Model Checker (competition contribution). In: Tools and Alg. for the Const. and An. of Sys. (TACAS). volume LNCS 8413; 2014. p. 389–91.

Lamnadi M, Trihi M, Boulezhar A. Study of a hybrid renewable energy system for a rural school in Tagzirt, Morocco. In: Int. Ren. and Sust. Energ. Conf. (IRSEC). 2017. p. 381–6. doi:10.1109/IRSEC.2016.7984079.

Mills A, Al-Hallaj S. Simulation of hydrogen-based hybrid systems using Hybrid2. Int J of Hydrog Energy 2004;29(10):991–9. doi:10.1016/j.ijhydene.2004.01.004.

Mohanty P, Sharma K, Gujar M, Kolhe M, Azmi A. Solar Photovoltaic System Applications; Springer International Publishing. p. 49–83. doi:10.1007/978-3-319-14663-8.

Morse J. Expressive and efficient bounded model checking of concurrent software. Ph.D. thesis; University of Southampton; 2015.

Morse J, Cordeiro LC, Nicole DA, Fischer B. Model checking LTL properties over ANSI-C programs with bounded traces. Software and System Modeling 2015;14(1):65–81. doi:10.1007/s10270-013-0366-0.

Moura LD, Bjørner N. Z3: An Efficient SMT Solver. In: Tools and Alg. for the Const. and An. of Sys. (TACAS). volume LNCS 4963; 2008. p. 337–40.

Natsheh E, Albarbar A. Solar power plant performance evaluation: simulation and experimental validation. In: J. of Physics: Conf. Ser. volume 364; 2012. p. 1–13. doi:10.1088/1742-6596/364/1/012122.

Pinho J, Galdino M. Manual de Engenharia para Sistemas Fotovoltaicos. Rio de Janeiro/RJ: CEPEL CRESESB, 2014.

Pradhan S, Singh S, Choudhury M, Dwivedy D. Study of cost analysis and emission analysis for grid connected PV systems using RETSCREEN 4 simulation software. Int J of Eng Res & Tech 2015;4(4):203–7.

Ross R. Flat-plate photovoltaic array design optimization. In: San Diego C, editor. 14th IEEE Photovoltaic Specialists Conference. 1980. p. 1126–32.

Saloux E, Teyssedou A, Sorin M. Explicit model of photovoltaic panels to determine voltages and currents at the maximum power point. Solar Energy 2011;85(5):713–22. doi:10.1016/j.solener.2010.12.022.

Schrammel P, Kroening D, Brain M, Martins R, Teige T, Bienmüller T. Incremental bounded model checking for embedded software. Formal Asp Comput 2017;29(5):911–31. doi:10.1007/s00165-017-0419-1.

Sengupta A, Mukhopadhyay S, Sinha A. Automated verification of power system protection schemesPart I: Modeling and specifications. In: IEEE Tran. on Power Del. volume 30(5); 2015. p. 2077–86. doi:10.1109/TPWRD.2014.2376571.

Swarnkar N, Gidwani L, Sharma R. An application of HOMER Pro in optimization of hybrid energy system for electrification of technical institute. In: Int. Conf. on Energ. Eff. Tech. for Sust. (ICEETS). 2016. p. 56–61. doi:10.1109/ICEETS.2016.7582899.

University of Massachusetts . Hybrid2. http://www.umass.edu/windenergy/research/topics/tools/software/hybrid2; 2016. [Accessed 1st October 2016].

Villalva M, Gazoli J, Filho E. Comprehensive approach to modeling and simulation of photovoltaic arrays. IEEE Trans on Power Elec 2009;24:1198–208. doi:10.1109/TPEL.2009.2013862.

Weatherbase . Manaus, Amazonas Travel Weather Averages. https://www.weatherbase.com/weather/weather.php3?s=23328&cityname=Manaus%2C+Amazonas%2C+Brazil; 2018. [Acessed 09 July 2018].