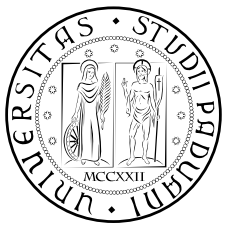


UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI TECNICA E GESTIONE DEI SISTEMI INDUSTRIALI
CORSO DI LAUREA MAGISTRALE IN INGEGNERIA MECCATRONICA

MPPT ALGORITHMS BASED ON ARTIFICIAL NEURAL NETWORKS FOR PV SYSTEM UNDER PARTIAL SHADING CONDITION

ALGORITMI MPPT PER SISTEMI FOTOVOLTAICI IN CONDIZIONE DI PARZIALE OMBREGGIATURA BASATI SU RETI NEURALI ARTIFICIALI

Relatore: Paolo Mattavelli

Correlatore : Marta Molinas

Laureando: Riccardo Salviati
1183982

ANNO ACCADEMICO: 2019/2020

SOMMARIO

All'interno di molti sistemi fotovoltaici si incontra la necessità di risolvere il problema di Maximum Power Point Tracking, in sigla MPPT. Un insieme di pannelli fotovoltaici infatti produce una potenza che varia a seconda della tensione di lavoro, della irradianza sui pannelli, e di altri parametri meno influenti come la temperatura o l'invecchiamento dei materiali che compongono il sistema considerato. Assumendo costanti le irradianze e la temperatura, la curva che descrive l'andamento della potenza generata al variare della tensione presenta un massimo globale. Nel caso in cui ci siano più pannelli fotovoltaici in serie e le irradianze sui singoli pannelli siano diverse tra di loro (situazione dovuta per esempio all'ombra di una nuvola passeggera, ad una rottura di una cella solare, o a luce riflessa da un'auto di passaggio) la curva potenza-tensione presenta anche alcuni massimi locali oltre al massimo globale. In questa situazione i pannelli fotovoltaici si dicono essere in "condizione di parziale ombreggiatura". La curva potenza-tensione è variabile nel tempo e sconosciuta, e lo scopo degli algoritmi MPPT consiste proprio nell'inseguire il punto di massima potenza globale. Esistono varie tecniche, la maggior parte delle quali[15],[5] basata sul metodo Perturba ed Osserva, in sigla P&O. Questo metodo consiste nel perturbare i pannelli (imponendo una tensione) e misurarne la potenza ottenuta. In questo modo è possibile campionare la curva potenza-tensione. In questa tesi sono stati studiati due algorirmi MPPT basati su una rete neurale artificiale di tipo Feed Forward, algoritmi ideati dagli autori degli articoli [18],[17]. Il primo algorithmo è composto solamente da una rete neurale, mentre il secondo algoritmo è un algoritmo cosiddetto ibrido: due diversi sotto-algoritmi sono combinati insieme al fine di sommare le proprietà positive di entrambi. Il

primo dei due consiste in una rete neurale, mentre il secondo è il cosiddetto algoritmo Hill Climbing. In questa tesi è stata ampliata l'analisi delle prestazioni dei due algoritmi, applicandoli in una simulazione di una porzione di un sistema fotovoltaico. È stato verificato che gli algoritmi studiati sono più efficienti di altri algoritmi nel momento in cui le irradianze sui pannelli cambiano molto velocemente, come ad esempio accade sui veicoli solari.

ABSTRACT

In many photovoltaic systems is often required to implement a Maximum Power Point tracking technique, in acronym MPPT technique. The power generated by a set of photovoltaic panels depends on the work point voltage, on the irradiances on the panels, and to a lesser extent on the temperature and on the aging of the materials of the devices considered. Assuming as constant the irradiances and the temperature, the curve that relates the power generated and the work point voltage shows a global maximum. If there are more than one panel in series and the irradiances on them are different one from each other (for example when some clouds shade the panels, or when a solar cell breaks, or when some light is reflected by a passing car) the power-voltage curve can show some local maximums besides the global one. In this situation the photovoltaic panels are under the so called "partial shading condition". The curve power-voltage is time variant and unknown: the target of the MPPT algorithms is to track the Maximum Power Point. Several techniques already exist and most of them are based on Perturbation and Observation (P&O) technique. This technique involves two actions: the Perturbation (by imposing a certain work point voltage) and Observation (measuring the power generated). Using this technique it is possible to sample the power-voltage curve. In this thesis two MPPT algorithm have been studied, both based on a Feed Forward Artificial Neural Network and both created by the authors of articles [18][17]. The first algorithm is only a Neural Network, the second one is a hybrid algorithm. When two or more sub-algorithms are combined together, the set is an hybrid algorithm. The positive properties of both of the sub-algorithms are operating in the hybrid one. The first sub-algorithm is a simle Neural Network,

the second one is a Hill Climbing algorithm. In this thesis the two algorithms are studied, and it has been extended their analysis already started in articles [18],[17]. The algorithms have been simulated on a portion of a photovoltaic system. It has been verified that both of them are more efficient than other algorithms when the irradiances on the panels are frequently changing, that can happen for example on solar vehicles.

Grazie alla mia famiglia,
che mi ha sostenuto
durante tutto il percorso.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1

INTRODUCTION

Much of photovoltaic's magic is due to its elegance and simplicity. A solar cell turns sunlight directly into electricity without fuel, moving parts, or waste products [10].

One of the most important topic in the public politic debate is the Global Warming problem, related to the creation of energy we use for our technologies. The solutions of this problem are so important that are been studied both by States, through Universities, and by private companies. Photovoltaic systems are one of the most studied and applied technology for the production of energy using renewable resources. "In fact whereas the fossil fuels laid down by solar energy over hundreds of millions of years must surely be regarded as capital, the Sun's radiation beamed at us day by day, year by year, and century by century, is effectively free income to be used or ignored as we wish"[10].

This thesis focus on the solution of a well known problem in the creation of photovoltaic energy: the tracking of the Maximum Power Point. Photovoltaic panels create the electric energy thanks to the irradiation received from the sun: generally bigger is the irradiation, and bigger is the power generated. But the power generated depends also on another parameter: the work point voltage of the panels. Bigger is this voltage and more little is the current generated by the solar cells of the panels: the consequence of this fact is that exists only one work point voltage called V_{mpp} that maximize the power generated. It is useful to know the value of this parameter: if the panels are working at that voltage then the power generated is the maxi-

mum. Working at different voltages would mean not to use the panel at their maximum potential.

In order to know the V_{mpp} a tracking algorithm is needed. The only task of this algorithm is to estimate the work point voltage that would generate the maximum power. This maximum-power-point voltage is often unknown, and depends on many quantities: the irradiances first of all, but also the temperature, or the aging of the materials that make up the panels. Then the maximum-power-point is also time-variant with these quantities. Another problem come up in the condition of partial shading: this condition occurs when the irradiance on the panels is not uniform. It can happen for example because some clouds shade a portion of the panels, or a solar cell breaks. It can happen if some leaves fall on the panels, or when some light is reflected on the panels. This situation make the estimation of the MPP more difficult: the relation between the power and the voltage in partial shading shows not only the global maximum power but also some local maximum power points. These maximum power points make the tracking of the MPP more difficult, especially while using some particular techniques of tracking like the Hill Climb algorithm.

There are several techniques that could solve this tracking problem [15], [5]. This thesis aims to study two algorithms based on a Feed Forward Artificial Neural Network, presented in articles [18], [17]. Neural Networks in fact could be useful in the solution of the MPPT problem because they are fast in the estimation of unknown parameters. Spending time in the estimation of the V_{mpp} would mean spending time without generating the maximum power, that means less efficiency. A quick estimation of the V_{mpp} on the other hand would mean that the maximum power starts to be generated early, and this increases the efficiency.

In this thesis the ANN-based algorithms are simulated, and tested in different conditions. Some characteristics of the algo-

rithm are studied: efficiency, time for the estimation of the Maximum Power Point, reaction to change of the PV system. These performance parameters are compared with the same ones of different algorithms, in order to have a useful comparison between different technologies that can guide manufacturers in the choice of the proper algorithm for their project.

The simulations are implemented using Matlab and Simulink. The simulations complete the analysis of the algorithms presented in [18], [17], using the algorithms on a simulated system and measuring the electrical parameters.

The chapters are five, including this Introduction. Chapter 2 describe in detail the problem, from the operation of the photovoltaic module to the existing solutions for the MPP tracking. Chapter 3 introduce the theory needed for the understanding of the Neural Networks. During the implementation of the algorithm some design choices are made, and they require a justification supported by the theory. In this chapter is finally described the procedure for the training of the Neural Networks. In chapter 4 the simulation procedure is presented, and the results are shown and commented. In chapter 5 the programs for the simulations are presented in detail.

2

STATE OF THE ART

2.0.1 *Model characteristics*

The solar cell is the core of the photovoltaic panels: thanks to the photovoltaic effect, electric energy is created from the power of the sunlight. The model that represents the electrical characteristics of the solar cell is illustrated in Figure 1. The solar cell is modeled as a current source with some parasitic elements in addition with a diode. The Formula for the current is

$$I = I_L - I_0[\exp(\frac{q(V - IR_s)}{AkT}) - 1] - \frac{V - IR_s}{R_{sh}} \tag{1}$$

where A is the quality factor of the diode, R_{sh} is the shunt resistance, R_s is the series resistance. I_L is the current generated thansk to the photovoltaic effect: it is proportional to the irradiation received by the panel. The proportional coefficient depends on the panel used.

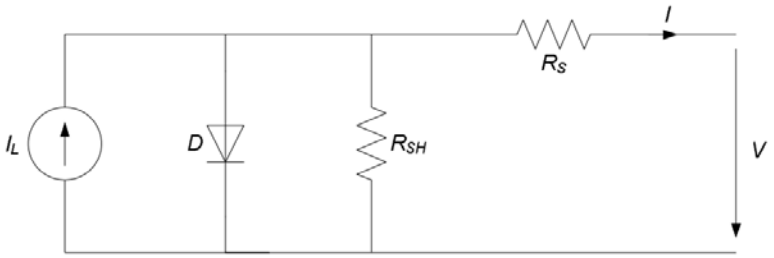


Figure 1: Electrical model. R_{sh} is the shunt resistance. R_s is the series resistance. I_L is the current created thanks to the irradiance.

The characteristics of a solar panel are usually described using the power-voltage curve, shown in Figure 2, under Standard Test Condition. In STC the irradiance is 1000 W/m^2, the temperature is 25°C. The power is calculated multiplying voltage

V and the current I that is calculated using Formula 1. In this calculation I_L is constant and proportional to the irradiation.

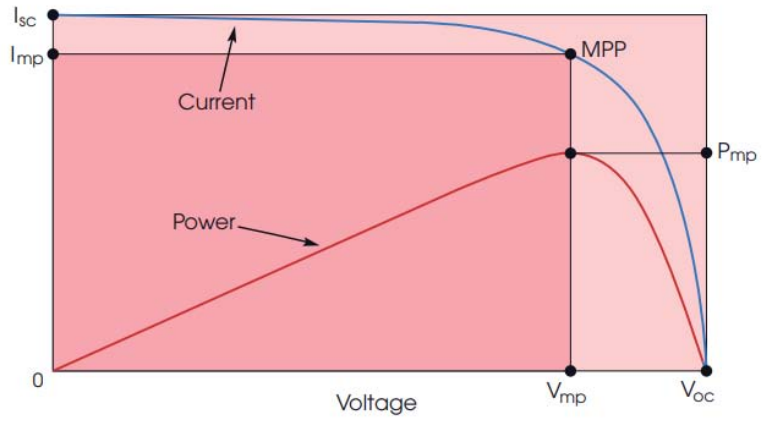


Figure 2: Example of curve power-voltage and Current-Voltage of a photovoltaic panel under standard test conditions. I_{sc} is the short-circuit current. V_{oc} is the open-circuit voltage. MPP is the Maximum Power Point.

2.0.2 *Open circuit voltage, short circuit current, maximum power point and fill factor*

There are two important points in Figure 2 : the V_{oc} (open-circuit voltage) and the I_{sc} (short-circuit current) point. Both of them are zero-Power point, but they show the maximum voltage and the maximum current that the solar panel can create. The parameter defined as $P_{mpp\_FF} = V_{oc} I_{sc}$ is used to define the **fill factor**, that is an industrial-used parameter to express the overall quality of the panel. The definition of fill factor is the following:

$$FF = \frac{V_{mpp} I_{mpp}}{V_{oc} I_{sc}} \tag{2}$$

and cannot be bigger then 1, because of the Shunt and Series Resistances shown in Figure 1. The typical FF for commercial solar cells is usually over 0.70 [10].

2.0.3 *Power-voltage curve changes*

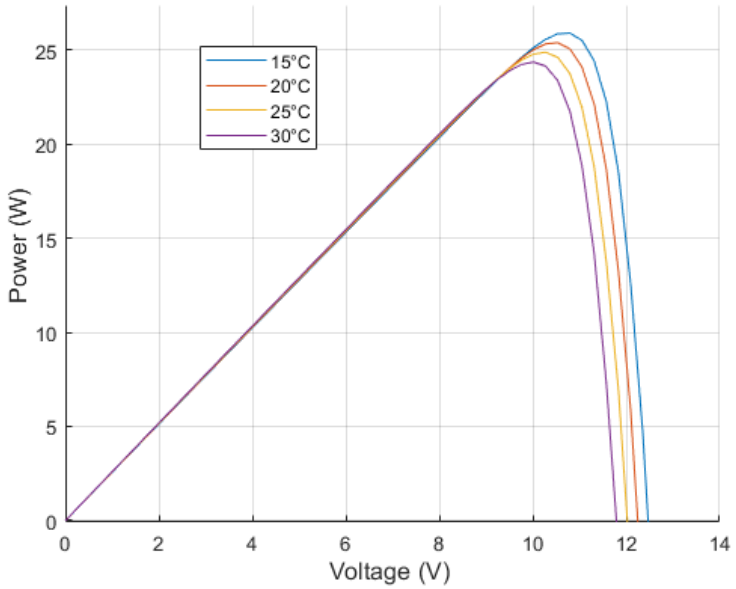


Figure 3: Three examples of the power-voltage curve of the default Simulink photovoltaic panel. The temperature is different in every curve, the irradiance is constantly 200 W/m^2.

There are three quantities that can change the power-voltage curve and cannot be controlled, that are: the temperature, the irradiation, and the aging of the materials that make up the panels.

The temperature affects negatively the power curve, since the V_{oc} depends on the temperature. One may thinks that the temperature increases the creation of current, which is true (see Formula 1) but the extracurrent created is not big enough to compensate the decrease of the open circuit voltage, that follows the linear empirical Formula 3 described in [19].

$$V_{oc}(T) = V_{oc}^{STC} + \frac{K_{v,\%}}{100}(T - 273.15) \tag{3}$$

where K_v is an empirical negative temperature coefficient. In Figure 3 are illustrated some examples of how the power-voltage curve changes related to the temperature.

The irradiance it's the primary source of energy for solar panels: bigger it is bigger is the current produced, and therefore the power. Three examples are presented in Figure 4.

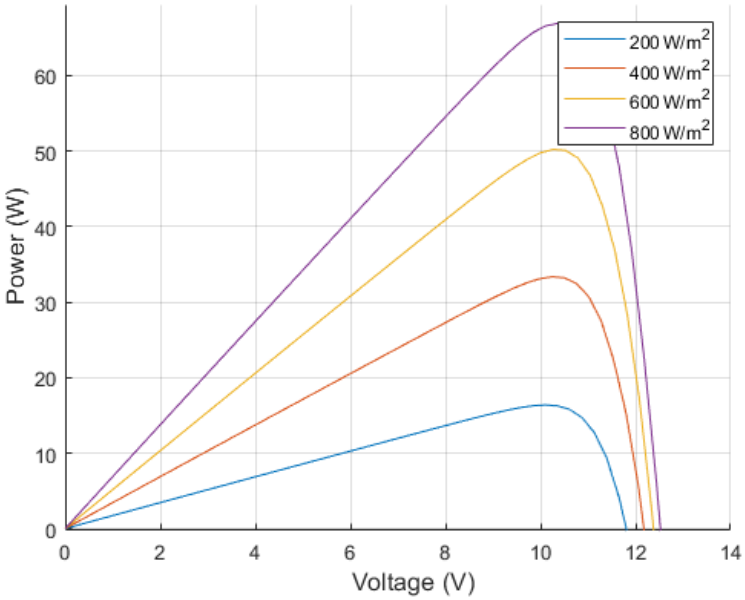


Figure 4: Four examples of the power-voltage curve of the default Simulink photovoltaic panel. The temperature is constant at 25°C, the irradiances are different in every curve.

The aging of the panels along the years decrease the maximum power P_{mpp}, and also changes the value of V_{mpp} . To understand the aging of the photovoltaic modules, the optical and electrical degradation effects are the most valuable explanation[1]. The optical degradation is due to the atmospheric elements that ruin the protection materials between the solar cells and the environment, decreasing the transmissivity (glass optical losses and encapsulating losses). The electrical one is just a slow degradation of the conducting materials, due to the interaction of the materials with air, humidity or other environmental factors. The maximum power point generally decreases with a rate of 1% per year, and the series resistance of the photovoltaic module is estimated to increase of +12.8% per 20 years of usage[1]. The effects of the aging are illustrated in Figure 5.

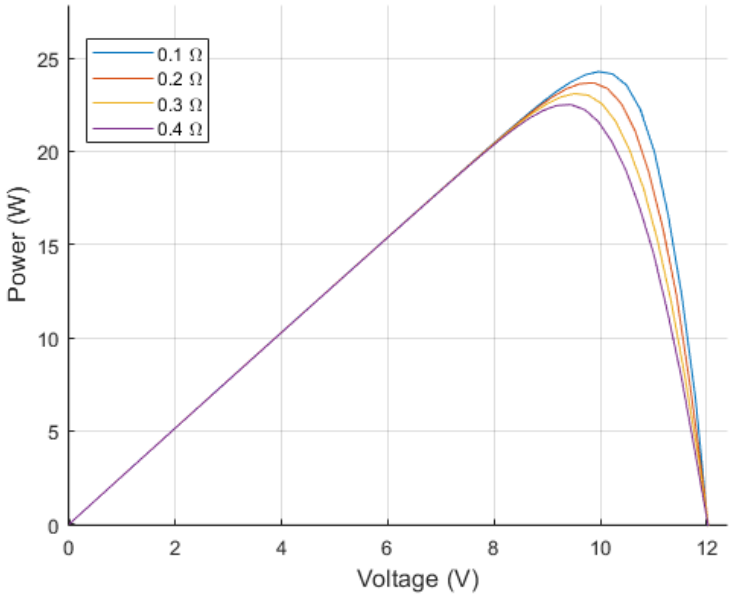


Figure 5: Four examples of the power-voltage curve of the default Simulink photovoltaic panel. The temperature is constant at 25°C, the irradiance is constant at 200 W/m^2. The aging of the materials that make up the panel is represented by the *aging resistance*, whose value is different for every curve represented.

2.0.4 *The bypass diode*

A single solar cell is of not generating enough power: it is just a high current source, working at low voltage. A photovoltaic module is the combination of multiple strings of solar cells, Figure 6.

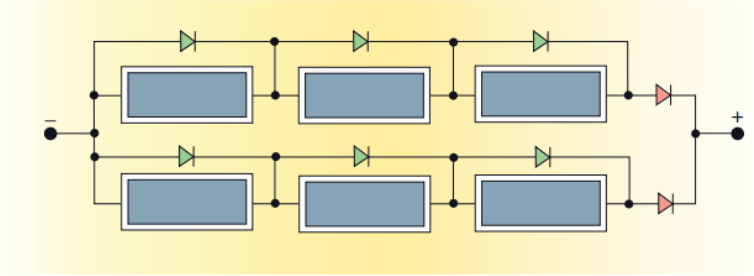


Figure 6: Photovoltaic module, where the cells are both in series and in parallel.

The solar cells can be placed in series, in parallel, or both, depending on the desired V-I output needed. There is a particular issue when there are some cells in series: when one of the cells doesn't produce the same amount of current produced by the others, that cell limits the current flowing in that branch of the circuit. This can happen when a cells is broken, or just because of different irradiation of the cells, due to shading or reflection.

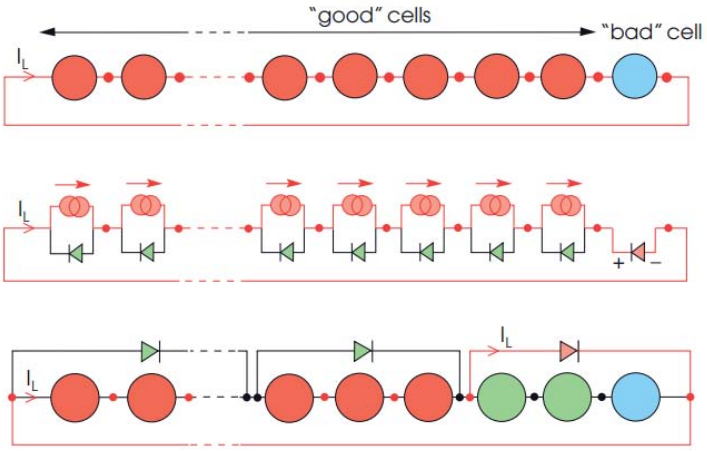
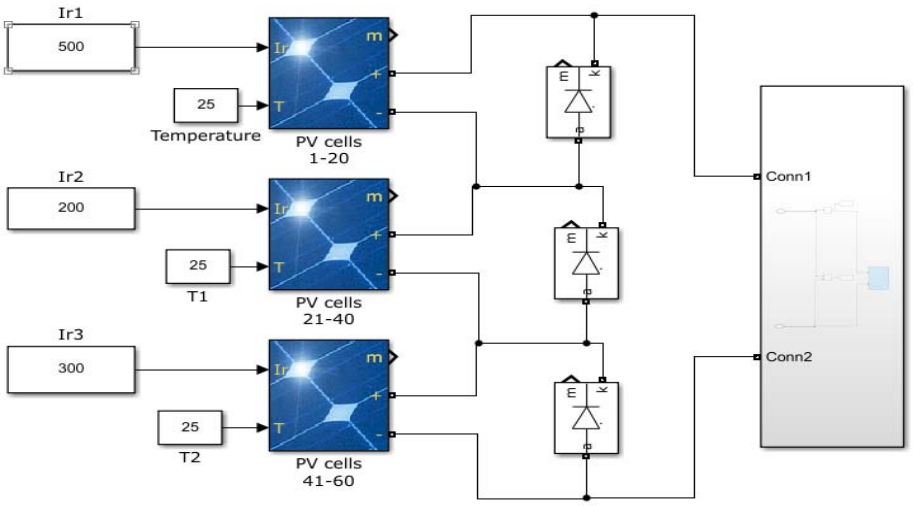
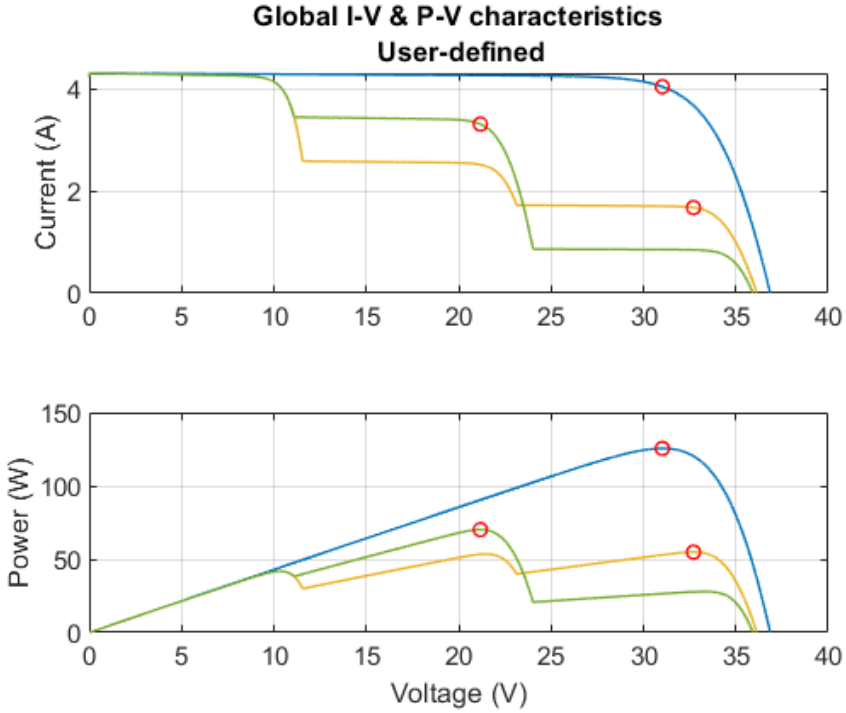


Figure 7: String of solar cells. In the first string there are not bypass diodes. The second string is the model of the first string. The third string show why the bypass diodes let the maximum current produced to flow.

In Figure 7 is illustrated the problem, and the adapt solution: a bypass diode. Thanks to the diodes in fact the current flowing will be not the minimum one but the maximum one between all the currents produced by the cells of that branch. Then the curve power-voltage will be different from the one in Figure 2. The new one is illustrated in Figure 8, and will have **many local maximums at most as many as are the bypass diodes used**.



(a) Simulink model, three photovoltaic modules in series with bypass diodes



(b) Power-Voltage curves with different irradiances

Figure 8: Three examples of power voltage curves, when the panels are irradiated differently. The blue curve represents the case where the irradiances are 500,500,500 W/m^2, the green $500, 400, 100$ W/m^2, the yellow $500, 300, 200$ W/m^2

2.1 PROBLEM STATEMENT

In chapter 2.0.4 is introduced the power-voltage curve of panels under partial shading, and is it discussed how this curve can change related to the environment in chapter 2.0.3. Now is it possible to explain the problem that this thesis aims to solve: the goal is to extract the maximum power that the panels and the environment let us to get. But it is unknown the voltage that the control should impose in order to get the maximum power, i.e. the V_{mpp}. The overall circuit in fact is the one shown in Figure 11, and some quantities need to be defined:

- output: the instant voltage V and current I created by the panels
- input in the physical system: the PWM signal, that controls the DC/DC converter. It is created by the control circuit
- input in the controller: the voltage-reference, created by the MPPT algorithm
- input in the MPPT algorithm: the instant voltage and Current, that is also the output.

Figure 9: The irradiance sensor on a panel [20].

The Control circuit, which includes the MPPT algorithm and the controller, has an important difference with the classic control circuits: it has no reference. The task of the MPPT block in fact is to find the V_{mpp}, that will be the voltage reference.

Figure 10: Photovoltaic panels under partial shading[20].

In real systems often are available more quantities measured, such as the temperature or the irradiances. The problem considered in this thesis assume not to have these measurements available, for the following reasons:

- the industrial target of this thesis is solar vehicles, or other really rapidly changing environment applications. Then for mechanical reasons is better to use the most little amount of components. In Figure 9 is it possible to observe that the irradiance sensors are quite unwieldy
- the irradiance sensors can detect the irradiance outside the panel, not on the panel, then cannot help to solve the problem of partial shading, Figure 10.

There many different topology of DC/DC converter and controller that can suit the problem considered, and many different loads as well. In Figure 11 the converter is a Boost converter, the Load a constant voltage source, and the controller a PI one. These components depend on the real applications, and don't really affect the MPPT algorithm's performances. They can affect the overall circuit performances , but the MPPT algorithm works independently from them. It will be designers' decision to choose the suitable set of controller and converter for the real application.

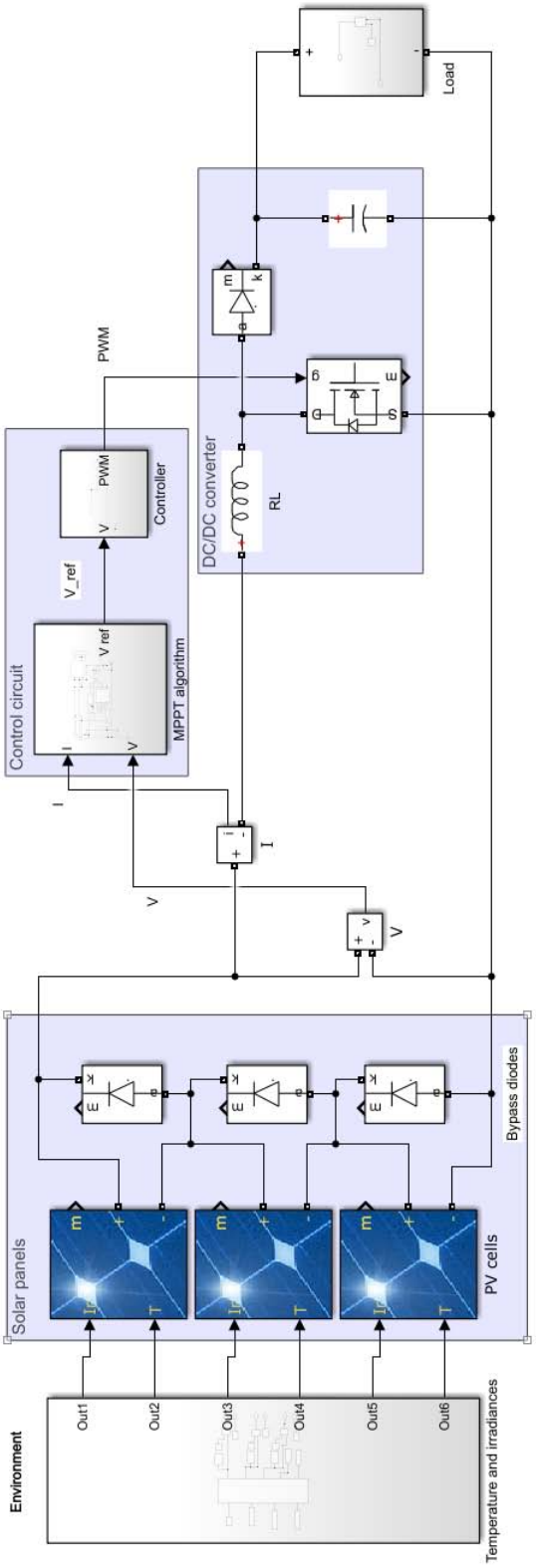


Figure 11: The physical system (photovoltaic panels, DC/DC converter, the load) and the control circuit (the MPPT algorithm and the controller) implemented in Simulink.

2.2 EXISTING SOLUTIONS

The problem of MPPT has been widely studied for more than fifteen years. The PSC problem (Partial Shading Condition) can be solved now using many different techniques, that are classified in this article [15] in the three following groups: Hard computing algorithms, Soft Computing algorithms, Hybrid algorithms. "Soft computing differs from conventional (hard) computing in that, unlike hard computing, it is tolerant of imprecision, uncertainty, partial truth, and approximation. In effect, the role model for soft computing is the human mind" [22]. Soft computing techniques include Evolutionary algorithms or algorithms based on Fuzzy Logic or Neural Networks. The hard algorithms include the Hc algorithm, the Open Circuit Voltage algorithm, and other that are listed in the following paragraphs. The hybrid algorithms are the combination of two or more of the previous ones: often there are soft computing and hard computing algorithm together. Some complete reviews among all the algorithms now existing can be found in [15] and [5].

2.2.1 *Hard computing algorithms*

These techniques compared to the soft computing algorithms are simpler, referring to both the hardware and the software. These techniques are useful expecially in hybrid algorithms, as it will be explained later in this chapter.

2.2.1.1 *Open Circuit Voltage*

The maximum power voltage is assumed to be around

$$V_{ref} = V_{oc}K$$

where K is between 0.71 and 0.80. This method shows its complete unefficiency in Figure 12, where the MPP is far away from the $V_{ref} = 37.025x0.75 = 27.76V$. This method works offline,

and requires only one measurements of V_{oc} every time-interval (decided heuristically).

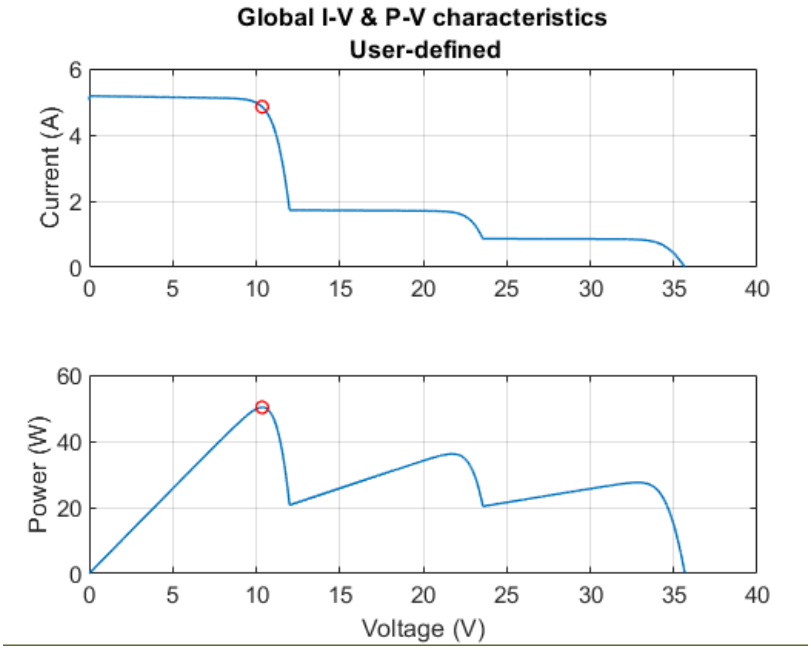


Figure 12: Photovoltaic panels under partial shading, the irradiances are [200100600] W/m^2. The maximum power point is at $V_{MPP} = 9.025V$

2.2.1.2 *Short Circuit Current*

It is the same method as explained in the chapter 2.2.1.1, but uses the current instead of the voltage:

$$I_{ref} = I_{sc}K$$

where K stays between 0.80 and 0.92 [15].

2.2.1.3 *Temperature Based Algorithm*

this method is useful to predict how the V_{MPP} changes because of the temperature [15]. It requires temperature's sensors, and just uses the Formula 4.

$$V_{MPP}(t) = V_{MPP}(T_{ref}) + K(T - T_{ref}) \tag{4}$$

2.2.1.4 *Hill Climb (HC)*

This method, based on the P&O technique presented in chapter2.2.3, is one of the most used. It explores the power-voltage curve sampling it, and track the MPP following the explored samples with the biggest power. This algorithm will be simulated in chapter 4.2, where it will be presented in detail.

2.2.1.5 *Incremental conductance (IC)*

The principle of incremental conductance (IC) method is similar to the HC algorithm. The IC algorithm is based on the P&O technique too, but the main parameter is the derivative of the current over the voltage, instead of the absolute power. This derivative is the conductance. This algorithm doesn't oscillate around the MPP as the HC does[3]. On the other hand, it is more complicated to implement it, due to the derivative calculations. When the maximum is reached the Formula 5 is satisfied.

$$0 = \frac{dI}{dV} + \frac{I}{V} \tag{5}$$

2.2.1.6 *Current sweep*

In this method the power-voltage characteristic curve is obtained using a sweep waveform for the PV array current. This scan of the characteristic curve is repeated at fixed time [11]. With this method the real MPP is obtained. On the other hand, the sweep takes certain time during which the operating point is not the MPP, which implies some loss of available power. Furthermore, the implementation complexity is high [14] due to the particular circuit needed, and also the estimation time is high. Due to the drawbacks and complexity exposed above, this MPPT method is not the best option to track the MPP continuously. However, it can be used in a hybrid algorithm, together for example with the HC algorithm.

2.2.2 *Soft Computing*

Soft Computing methods are emerged as an alternative approach to conventional techniques for partially shaded photovoltaic system because of their ability to solve the complex nonlinearity problems. Consequently, for the condition like PSC, these techniques assure faster convergence and high efficiency. Meanwhile, it should be noticed that the high cost and complexity of implementation are two main drawbacks of these methods[15].

Various optimization algorithms have been proposed in the review[5], such as Particle Swarm Optimization (PSO) which will be used in this thesis, and others like: Modified PSO, Artificial Bee Colony, Ant Colony Optimization, Simulated Annealing, Bat algorithm, Firefly algorithm, Firework algorithm, Glowworm Swarm Optimization, S-Jaya algorithm, Flower pollination, Grey Wolf Optimization, Teaching Learning Algorithm, Mine Blast Optimization, Whale Optimization Algorithm, and others.

2.2.2.1 *Fuzzy Logic Control (FLC)*

Fuzzy logic controller has wide range of applications in renewable energy applications[2]. The use of fuzzy logic controllers has been increased over the last decade because it is able to deal with imprecise inputs, doesn't need an accurate mathematical model and can handle nonlinearity.

It is not the goal of this thesis to explain in detail this complex technique. However, in the reviews [5], [14] the FLC algorithms are often the direct "competitor" of the ANN-based algorithms, that are studied in this thesis. In fact the FLC techniques are studied because of their rapidity in tracking the MPP and their effectiveness, despite their complexity. This last comment is true also for Neural Networks algorithms.

2.2.2.2 *Artificial Neural Network (ANN)*

The Artificial Neural Network are one of the most important techniques of the Artificial Intelligence. They are able to approximate complex system, learn from big data, or even forecast future events. They are useful in many theoretical research fields like function fitting, optimization or machine and deep learning. this topic is better analyzed in the next chapters.

For MPPT algorithms the Neural Networks are useful because of their rapidty in tracking the MPP: they are probably the fastest method[18]. Artificial Neural Networks are not used so often in MPPT algorithms[15], [5], but there are in particular two articles that is good to mention: articles[18], [17]. The idea of this thesis started from them and the problem to solve is the same. This thesis aims to complete the analysis of the solutions purposed by the authors of [18] and [17].

2.2.2.3 *Evolutionary Algorithms (EA)*

In artificial intelligence, an evolutionary algorithm (EA) is a generic population-based metaheuristic optimization algorithm[20]. An EA uses mechanisms inspired by biological evolution, such as reproduction, mutation, recombination, and selection. In general this class of algorithms works using a population of particles, that "explore" the unknown function in order to find the global maximum (or minimum). Inside this population there is a sort of communication between the particles, that helps to achieve the goal quicker. Evolutionary algorithms often perform well approximating solutions to all types of problems because they ideally do not make any assumption about the underlying fitness landscape[20].

In this thesis the PSO algorithm has been used, in order to have one more element of comparison for the evaluation of the performances of the ANN. PSO is one of the most common EA used in literature[5].

2.2.3 *The* P&O *methodology*

The Perturbation and Observation methodology (P&O) is used in many MPPT algorithms, especially the ones that are part of the Soft Computing techniques. It is not a MPPT algorithm, it's a technique that is used in many MPPT algorithms in order to sample the instant power-voltage curve. It consists of the following two steps:

- Perturbation: a certain voltage V_{ref} is imposed on the circuit
- Observation: the current is measured.

In this way the power-voltage curve (that is unknown) can be sampled, in other words "explored". An illustration of the process of P&O is illustrated in Figure 13, where the Perturbation and Observation is executed four times. Many algorithms are based on this technique, for example the HC, the PSO, the Neural Networks, and others.

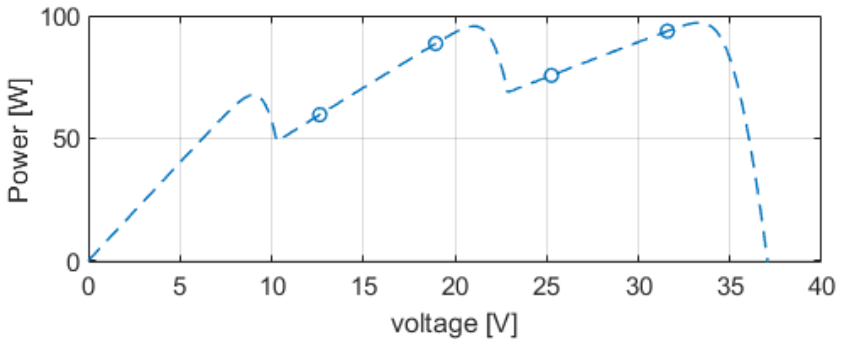


Figure 13: The Perturbation and Observation technique: some examples of sampling an unknown power-voltage curve. The samples are collected at $V_{ref} = 12.5, 19, 25, 32$ V.

3

NEURAL NETWORKS

3.1 INTRODUCTION

Artificial intelligence and neural networks are information processing paradigms inspired by the way biological neural systems process data. Artificial intelligence and cognitive modeling try to simulate some properties of biological neural networks. In the artificial intelligence field, artificial neural networks have been applied successfully to speech recognition, image analysis and adaptive control, in order to construct software agents or autonomous machines[21].

3.2 BRIEF HISTORY

The preliminary theoretical base for contemporary neural networks was independently proposed by Alexander Bain (1873) and William James(1890). In their work, both thoughts and body activity resulted from interactions among neurons within the brain. In the late 1940s psychologist Donald Hebb created a hypothesis of learning based on the mechanism of neural plasticity that is now known as Hebbian learning. Hebbian learning is considered to be a "typical" unsupervised learning rule and its later variants were early models for long term potentiation. These ideas started being applied to computational models in 1948 with Turing's B-type machines. Rosenblatt (1958) created the perceptron, an algorithm for pattern recognition based on a two-layer learning computer network using simple addition and subtraction. With mathematical notation, Rosenblatt also described circuitry "not" in the basic perceptron, such as the "exclusive-or" circuit, a circuit whose mathematical computa-

tion could not be processed until after the backpropagation algorithm was created by Werbos (1975).

Through the influence of John Hopfield (1986), who had personally convinced many researchers of the importance of the field, and the wide publication of backpropagation by Rumelhart, Hinton and Williams, the field of neural networks slowly showed signs of upswing. From this time on, the development of the field of research has almost been explosive[9]. It can no longer be itemized, but some of its results will be seen in the following[9].

3.3 STRUCTURE AND PROPERTIES

The elements that make up an ANN are illustrated in Figure 14 and have the following names[7]:

- neurons: fundamental components, they are the place where to make the calculations (see Figure 15)
- edges: connections between neurons. They are associated to a numeric value called weight.
- layers: series of neurons. When the ANN shows more then one hidden layers, the net is considered a "deep ANN", otherwise just "shallow ANN"

Every single neuron calculate the output of his own activation function, using the formula 6:

$$z = \sum_{m=1}^{M} w_m x_m + \alpha \tag{6}$$

where $x_1, x_2, ..., x_M$ are the M incoming signals, and $w_1, w_2, ..., w_M$ are the related synapses weights, while α is a value attached to every neuron called bias. The result z is then the input of the activation function, as illustrated in Figure 15. A deeper explanation of these formula can be found in this book[9].

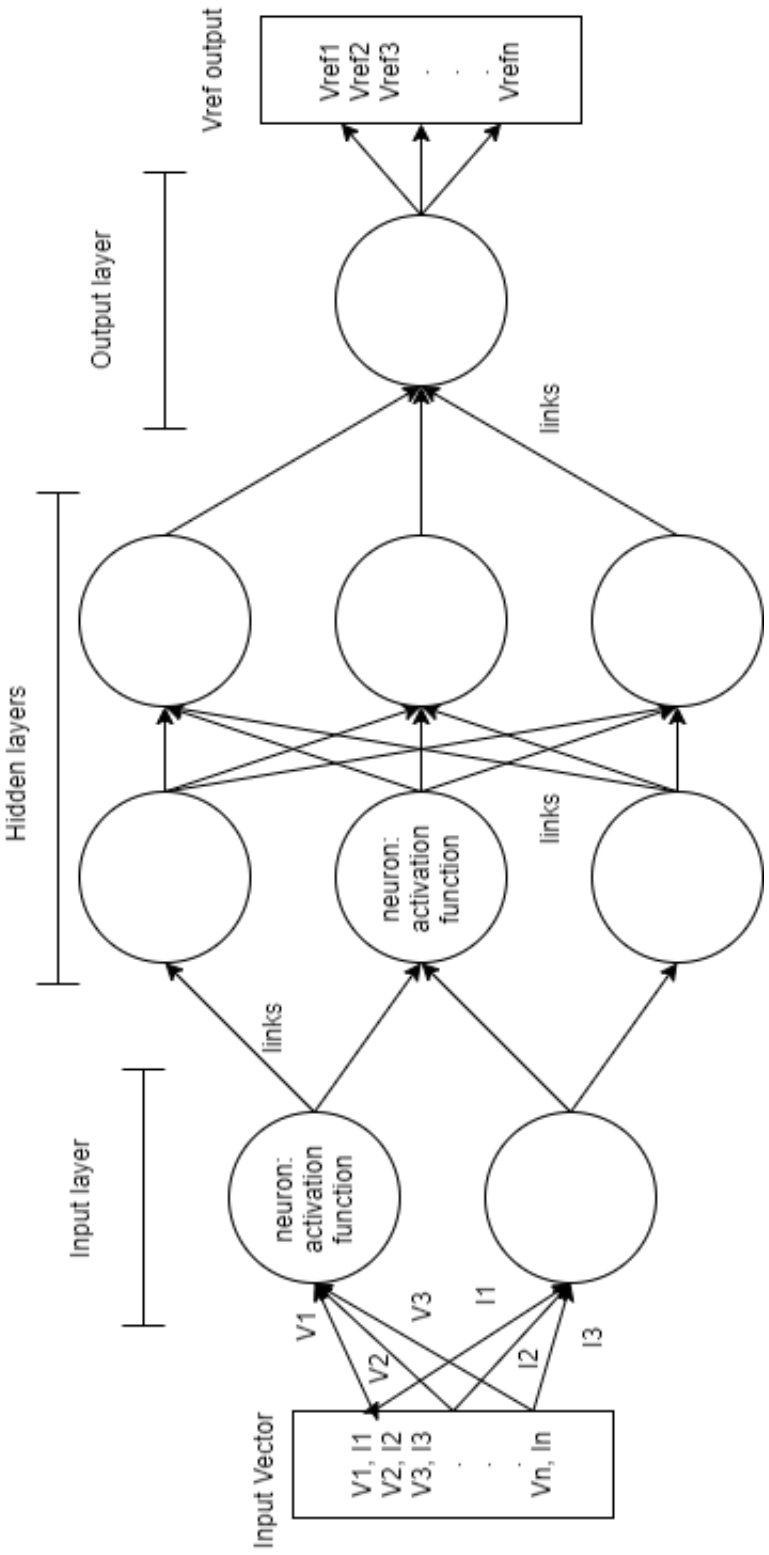


Figure 14: Structure of a simple Feedforward ANN

Two different main kinds of ANN's structure arise from the way the neurons are connected to each other: feed-forward neural network (FNN) and recurrent neural network (RNN)[18]. The Feedforward Neural Network are the most simple, and each neuron in one layer has only directed connections to the neurons of the next layer (towards the output layer). The Recurrent Neural Network is defined as the process of a neuron influencing itself by any means or by any connection.

The ANN are used for solving many different problems, including classification, prediction, filtering, optimization, pattern recognition, and function approximation (fitting). The last mentioned one is the problem of this thesis, that aims to find the nonlinear relationship between the voltage reference and the maximum power of the panel, using the data as will be explained later.

There is an important property of the Neural Networks that is good to mention: *standard multilayer feedforward networks are capable of approximating any measurable function to any desired degree of accuracy, in a very specific and satisfying sense. Has been thus established that such "mapping" networks are universal approximators. This implies that any lack of success in applications must arise from inadequate learning, insufficient numbers of hidden units or the lack of a deterministic relationship between input and target[8].* This property has been fundamental, because is it possible to assert that, referring to the thesis' problem, *always exist a more precise Neural Network that will make the MPPT algorithm more effective.* It will be just a design choice how accurate the Neural Network will be.

3.4 TRAINING

The training of a Neural Network is probably the most crucial point of the implementation of this method. During the training process all the weights of the edges and the biases of the

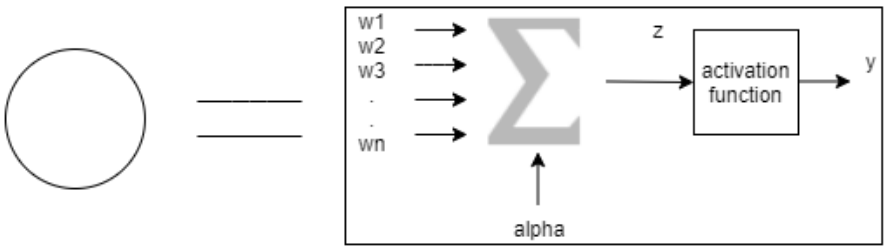


Figure 15: Structure of a single neuron

neurons are changed and adapted in order to accomplish the goal prefixed: when some weight or biases are changed, then the NN "has learned" something. There are basically three main method of learning:

- Reinforcement learning (online): the neural network try to maximize a reward function, that give a certain value to some input and output quantities
- Unsupervised learning (online): is the biologically most plausible method, but is not suitable for all problems. Only the input patterns are given; the network tries to identify similar patterns and to classify them into similar categories[9]
- Supervised Learning (offline): the training set consists of input patterns as well as their correct results in the form of the precise activation of all output neurons. Thus, for each training set that is fed into the network the output, for instance, can directly be compared with the correct solution and the network weights can be changed according to their difference. The objective is to change the weights to the effect that the network cannot only associate input and output patterns independently after the training, but can provide plausible results to unknown, similar input patterns[9].

3.4.1 *The supervised learning*

The supervised learning procedure is not always biologically plausible, but it is effective and therefore very practicable. It is the learning technique that has been been used in this thesis. The algorithm used to train the ANN in this thesis is the Levenberg-Marquardt backpropagation, that is the most used for FNN[18] because of his speed-convergence and easy-calculations properties. This algorithms is not analyzed in detail. It is presented intuitively in the following paragraphs the operation of the general back-propagation algorithm, that is useful to understand how the dataset should be in order to get a working Neural Network. In fact for the supervised learning it is needed a dataset that will be used for the training of the network. This dataset must include a sufficient and various number of input and their respective desired output. The inputs will be given to the neural network, the outputs are the results expected from the neural network.

3.4.1.1 *The backpropagation procedure*

The *backpropagation of error* learning procedure is intuitively described in this paragraph. First of all is useful to understand what does it means "to train" a neural network: it means that the edges and the biases (that are numerical values) of the neurons are increased or decreased, following the *learning procedure* used. This "strengthening" is made following the mathematical rules described by the learning procedure used, in this thesis the learning procedure is the backpropagation.

The backpropagation of error changes the weights and the biases of the neurons in order to minimize the error between the output given by the network not yet trained and the output desired. The steps of this operations are illustrated in Figure 16 and commented more in detail in the following lines. The illustrated steps of the backpropagation algorithm are:

- forward propagation: one input from the dataset is given to the neural network, that, following equation 6, returns one output called y
- error calculation: the error $y - y_e$ is calculated, where y_e is the desired output expected from the network and known from the training dataset
- backward propagation: the error is propagated backward through the net. Every neuron calculates its own contribution to the error, and changes its weights and biases using the mathematical method called "gradient descent". This method is described in [9].

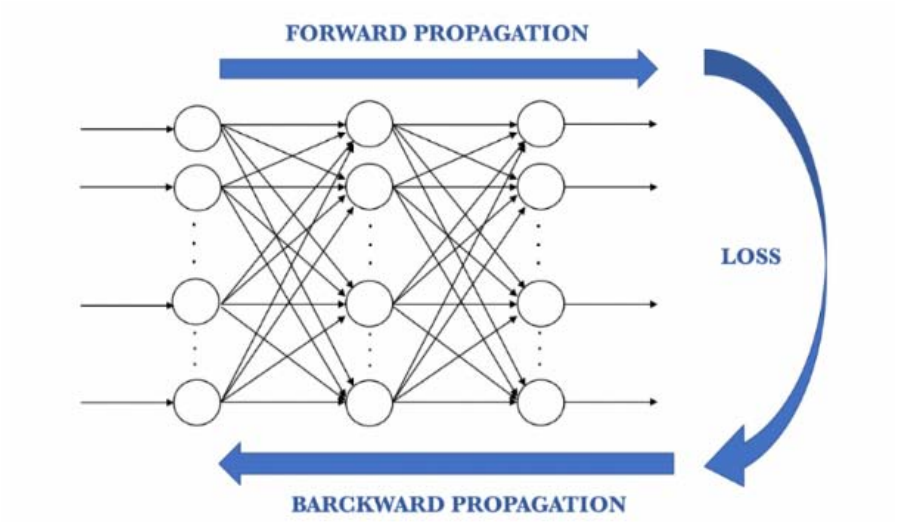


Figure 16: The three steps of the backpropagation illustrated: forwrd propagation, loss calculation (or error calculation) and backward propagation

Once the three steps are executed, the Neural Network has been "trained". The weights and the biases are changed in order to adapt the net and to reduce the error $y - y_e$. One *epoch* is the repetition of these three steps one time for every couple [input, output] of the training dataset. At every epoch the net adapt its operation in order to give the desired outputs with the desired final mean squared error (MSE). More little is the desired error, and more epochs are usually needed.

3.4.2 *Cross Validation and Overfitting*

One of the problems that occur during neural network training is called overfitting. The overfitting is detected when new data (different from the ones in the training dataset) are given as input to the network and the new error $y - y_e$ is large, despite the mean squared error previously described has been driven to a very small value. The explanation for this situation is the following: the network has memorized the training examples, but it has not learned to generalize to new situations[12]. A useful representation of the problem is illustrated in Figure 17 where there are some example of bad fitting, good fitting, and overfitting in a two-dimensional problem.

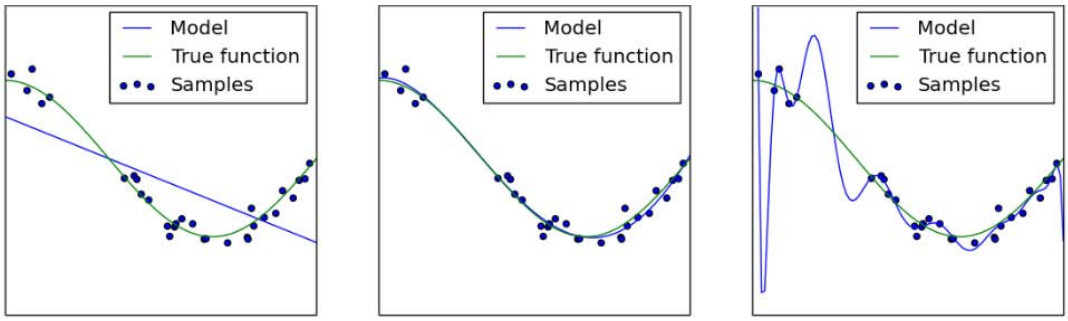


Figure 17: Example of fitting a function. In the first picture the model "saved" in the ANN is insufficiently accurate (low number of epochs), and the MSE between the model (y) and the samples (y_e) is high. In the second picture the model is good (adequate number of epochs) and the MSE is low. In the third picture the model "saved" in the ANN follows perfectly all the samples (the MSE is almost zero) but doesn't fit the true function: this caseis called overfitting.

One method for improving network generalization is to use a network that is just large enough to provide an adequate fit. The larger network you use, the more complex the functions the network can create. If you use a small enough network, it will not have enough power to overfit the data.

Unfortunately, it is difficult to know beforehand how large a network should be for a specific application. There are two other methods for improving generalization: regularization and early stopping[12]. The next sections describe these two techniques.

EARLY STOPPING In this technique the available data is divided into three subsets. The first subset is the training set, which is used for computing the gradient and updating the network weights and biases. The second subset is the validation set. The error on the validation set is monitored during the training process. The validation error normally decreases during the initial phase of training, as does the training set error. However, when the network begins to overfit the data, the error on the validation set typically begins to rise. When the validation error increases for a specified number of iterations, the training is stopped, and the weights and biases at the minimum of the validation error are returned (see Figure 18). The comparison between the validation error and the training set error is called **cross validation**.

The test set error is not used during training, but it is used to compare different models. It is also useful to plot the test set error during the training process. If the error in the test set reaches a minimum at a significantly different iteration number than the validation set error, this might indicate a poor division of the data set[12].

REGULARIZATION Another method for improving generalization is called regularization. This involves modifying the performance function, which is normally chosen to be the sum of squares of the network errors on the training set. This method has not been used in the thesis, so it will not be explained in detail.

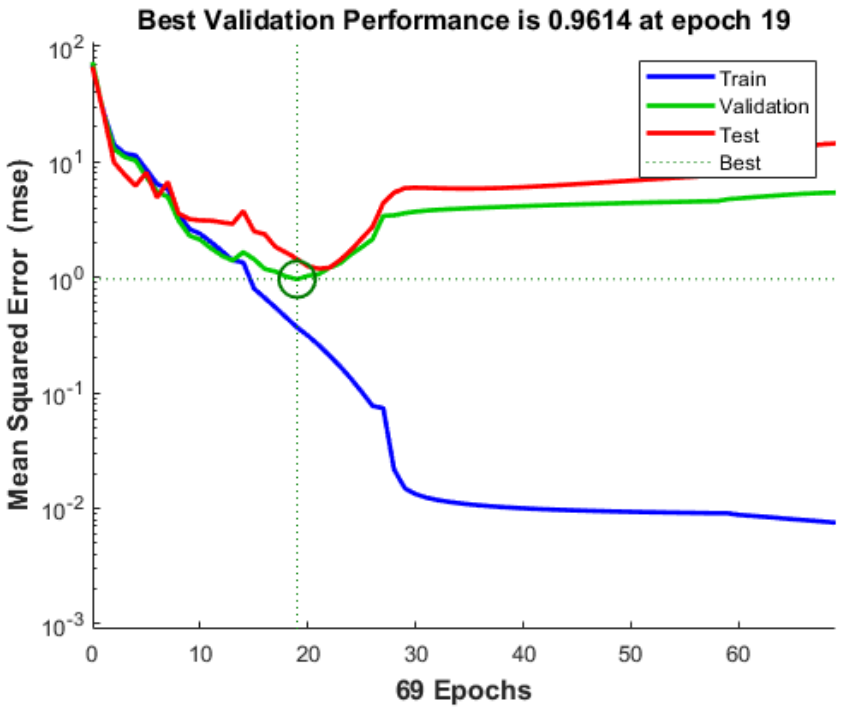


Figure 18: Example of the errors' graphs during the training of a ANN. The illustrateed errors are: Training error, validation error, and Set error. These are useful to detect and prevent the overfitting.

3.5 IMPLEMENTATION OF THE ARTIFICIAL NEURAL NETWORK

While implementing a neural network, two design problems have to be faced: the training the ANN, and the architecture choice. Because of the problem that this thesis aims to solve, a Feedforward NN is the best choice for the architecture, but there is no precise way to know how many layers and neurons and what kind of training data are needed. This two problems are addressed separately, in the following paragraphs.

3.5.1 *Training*

In this chapter it will be used a simple architecture of the ANN: three hidden layers, with respectively 18, 9, 5 neurons. This ar-

chitecture is chosen from article [17], but is not necessary the best one for this problem. This architecture is used only for separating the training problem from the architecture-choice problem, so the results obtained from different training methods can be compared and are indipendent from the architecture.

The last decision now is about what dataset to use for the training: two different datasets are available, generated as described in chapter 5. These two datasets both contains the inputs needed by the ANN and the respective desired outputs. The difference between them is the distribution of the irradiances. One dataset is generated based on a Normal distribution of irradiances, the second is based on a Uniform distribution, and they are described in chapter 4.1. The two distributions are an estimation of how the irradiances can change with time. The Normal distribution is probably closer to a real application, while the Uniform distribution is useful to test a MPPT algorithm because the wide changes of the irradiance, as can be observed in Figure 19, stress more the algorithm. To understand which one gives the best results, the four combination of training and test are simulated, as in table 1. How the simulation is performed is described in the next chapters, in this chapter only the training procedure is described indipendently.

Training distribution	Test distribution	Efficiency
Normal	Normal	0.9963
Normal	Uniform	0.9884
Uniform	Uniform	0.8574
Uniform	Normal	0.9062

Table 1: Comparison of performances between ANN trained using different dataset.

The results tell that the training using the Normal distribution is better then using the Uniform one. This fact has an in-

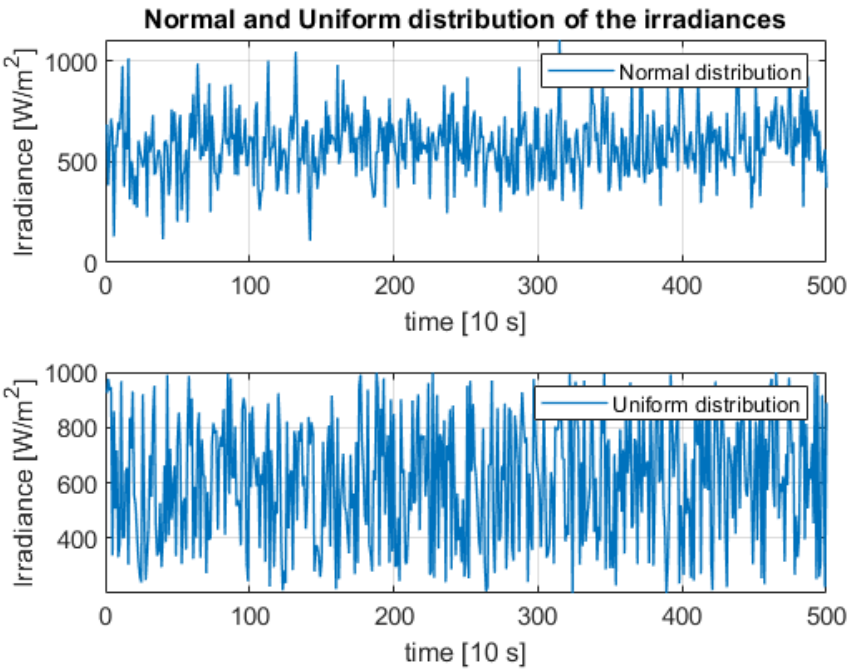


Figure 19: The two distributions of the irradiances on the first panels of the system described in chapter 5.

tuitive explanation: a neural network basically calculates the output that has the highest probability to be good. Then having a training dataset that is "compact" like the Normal distribution is better then the Uniform distribution, which spreads too much the information, with the consequence of obtaining a model not sufficiently accurate.

3.5.2 *Architecture*

The architecture design is probably the most investigateda very discussed problem when using neural networks, because there are no formulas for the design of the networks. From the literature the only thing that is clear about the final performances of the architecture of ANNs is the theorem reported in chapter 3.3, so it is known that exists always the best architecture, more precisely: there is always an architecture that makes the ANN sufficiently accurate for the goal of the problem.

To understand which architecture to use in this thesis a lot of different architectures are tested, using the same training method. In this way only the effect of the architecture are influencing the final efficiency, and in this way the best architecture can be chosen.

Without any specific reasoning the architectures will be composed by three hidden layers with the same number of neurons: this is called *rectangular configuration*, and will not change. The two parameters that will change among all the architectures are:

- NSAMPLE: the number of samples that the ANN can use during the exploration (the exploration of the algorithm is explained in detail in chapter 4.4)
- N: this is the number of neurons in every hidden layer.

In order to have an exhaustive amount of different cases, NSAMPLE will vary between 14 different values, that are in Table 2, and N will vary between 1 and 30. The total number of different architecture then is 14x30 = 420. The NSAMPLE represents the number of samples that the ANN algorithm does during the exploration: this is explained in chapter 4.4. The maximum

NSAMPLE	3	4	5	6	7	8	9	10	15	20	25	30	40	60
NEURONS	1	1	1	...										
PER	2	2	2	...										
LAYER	...	...	...	...										
	30	30	30	...										

Table 2: All the different parameters respectively for every different ANN architecture tested. The efficiency of every architecture is rapresented in Figure 20.

efficiency's values resulted from these test are very irregularly distributed, and this distribution is represented in Figure 20:

the best numerical result have to be chosen, and following Figure 20 for every NSAMPLE there is one "optimum" number of neurons per layer that is the candidate N.

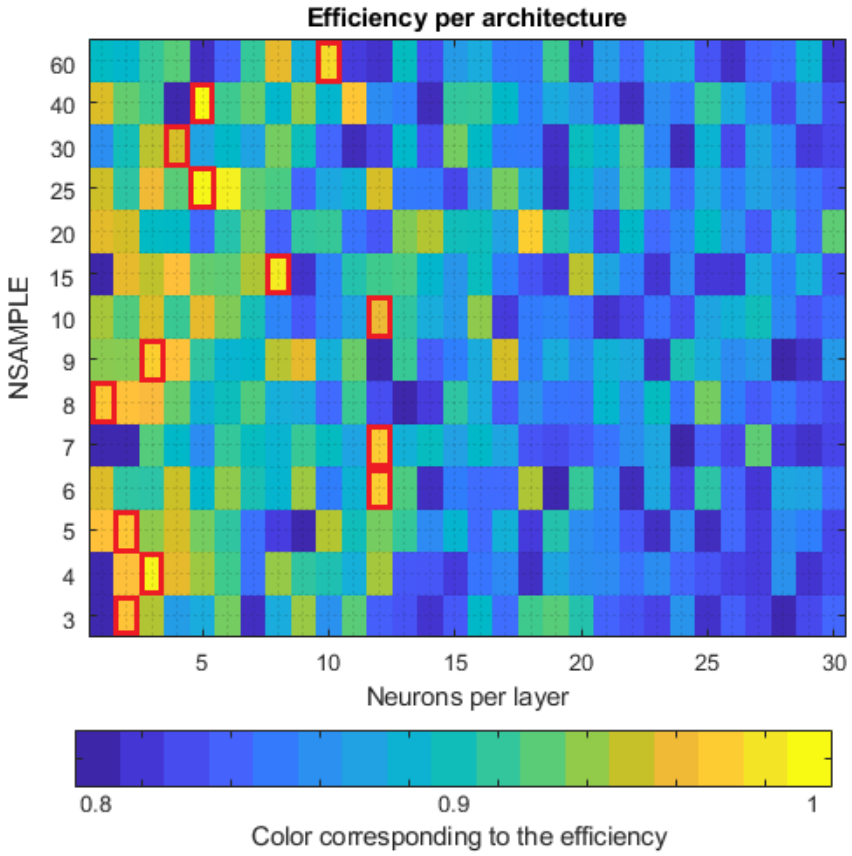


Figure 20: The distribution of the efficiencies: brighter color correspond to higher efficiency. For every combination of NSAMPLE and number of neurons per layer N there is one efficiency obtained from the simulations. The architecture represented are referring to the architectures presented in Table 2. The maximum efficiency values are highlighted by the red rectangles.

3.5.3 *Most performing ANN*

Summarizing the results from the two previous paragraphs, the best ANN are trained using the Normal irradiation dataset. The

best architectures so far found are reported in Table 3, depending on the number of sample for the exploration (the exploration of the ANN algorithm is explained in chapter 4.4).

NSAMPLE	3	4	5	6	7
HIDDEN LAYER1	2	3	2	12	12
HIDDEN LAYER2	2	3	2	12	12
HIDDEN LAYER3	2	3	2	12	12
Efficiency	0.988	0.992	0.988	0.989	0.988
NSAMPLE	8	9	10	15	20
HIDDEN LAYER1	1	3	12	8	18
HIDDEN LAYER2	1	3	12	8	18
HIDDEN LAYER3	1	3	12	8	18
Efficiency	0.988	0.989	0.985	0.992	0.988
NSAMPLE	25	30	40	60	
HIDDEN LAYER1	5	4	5	10	
HIDDEN LAYER2	5	4	5	10	
HIDDEN LAYER3	5	4	5	10	
Efficiency	0.992	0.984	0.993	0.982	

Table 3: The best architecture studied for the problem and their efficiency value obtained from the simulations. The efficiency parameter is presented in chapter 4.4.

4

SIMULATIONS AND RESULTS

In this chapter the four MPPT algorithms (Hill Climb, Hill Climb Modified, Particle Swarm Optimization, Artificial Neural Network) are presented in detail and simulated, and their performances are compared.

4.1 GENERAL TEST PROCEDURE

The idea for the test of the algorithms is the following: every algorithm is tested on the system presented in chapter 5 irradiated by 500 different combination of three irradiances, one irradiance per panel. Therefore at every combination of irradiances the power-voltage curve is different, and also the V_{mpp} and the P_{mpp}. The temperature is assumed fixed at 25°C and the panels are assumed new: thanks to these two assumptions, the power-voltage curve depends only on the irradiances.

The main parameter used for the comparison is the average efficiency, that is

$$efficiency_{average} = \frac{1}{500}\sum_{i=1}^{500} efficiency_i = \frac{1}{500}\sum_{i=1}^{500} \frac{P_{mpp\_estimated,i}}{P_{mpp\_theoric,i}}$$

The $efficiency_i$ is the efficiency obtained while the irradiances on the panels are the irradiances $(\lambda_1, \lambda_2, \lambda_3)_i$. The $P_{mpp\_estimated,i}$ is the power generated by the panels that are working at $V_{mpp\_estimated,i}$, that is the value calculated by the MPP algorithm under the irradiances combination $(\lambda_1, \lambda_2, \lambda_3)_i$. The $P_{mpp\_theoric,i}$ is known thanks to the simulations described in chapter 5, and there is one $P_{mpp\_theoric,i}$ for each combination of irradiances $(\lambda_1, \lambda_2, \lambda_3)_i$.

NOTE ON THE IRRADIANCES In order to have a complete case-study two different way of generating the random irradiances are considered. In one dataset of irradiances the random values are **uniformly** distributed around one mean value (600 W/m^2). In the second dataset the irradiances are **normally** distributed around their mean value and with a certain variance (respectively 600 W/m^2, 160 W/m^2). These numerical values are chosen referring to the studies [18] and [17]. The uniform distribution is used for the test of the algorithms. The normal distribution is used for the training of the Neural Networks, as it has been explained in chapter 3.5.1.

In the review articles [15], [5] a second parameter is considered for the comparison of the different MPPT algorithms, that is the *tracking time*, or *estimation time*. This is the time that the algorithms need in order to estimate a new $V_{mpp\_estimated}$. The faster is the estimation of the $V_{mpp\_estimated}$, the faster the $P_{mpp\_estimated}$ starts to be generated.

There are some considerations to do in order to explain the test procedure. These considerations will be explained in detail in the following chapters, now they are only presented:

- HC and HC modified: the tracking time depends on many parameters, like the voltage interval used to update the new $V_{mpp,estimated}$ and on how much "far" is the $V_{mpp,estimated}$ from the $V_{mpp,theoric}$; in general more time is used and more efficient is the algorithm, as it will be verified afterwards
- PSO: the tracking time can be set by the manufacturer, that can choose the maximum number of Perturbation and Observations the algorithm is allowed to do; in general the bigger is this number, the better is the efficiency of the algorithm, as it will be verified afterwards. It is equivalent to say that there is one tracking time value for the efficiency desired

- ANN: the tracking time can be set by the manufacturer, that can choose the maximum number of Perturbation and Observations similarly to the PSO case; but it is not necessarily true that the bigger is this number the better is the efficiency of the algorithm, as it will be shown afterwards.

After the considerations listed above it is clear that the efficiency of the first three algorithms is related to how much time the estimation process could use. If there is not sufficient time the estimation process will not complete its operation, and the estimation will be not sufficiently accurate and the efficiency obtained will be low.

Because in a Neural Network the tracking time is a parameter that can be decided by the maker, in order to compare the efficiency of the four algorithms the following procedure has been used. This procedure has been executed one time for every different combination of irradiances.

PROCEDURE Every algorithm is tested for a given time, that is called *exploration time*. The MPPT algorithm works during the exploration time, and the best estimation of V_{mpp} so far achieved is the final $V_{mpp\_estimated}$. The efficiency number $\mathfrak{i}$ is finally calculated using the $P_{mpp\_estimated}$ at the end of the exploration time.

The difference between *exploration time* and the *tracking or estimation time* is conceptual. The first one is the maximum time given to the algorithm for working, it is decided before the start of the simulation and it is not a property of the algorithm. The second one is the minimum time needed to the algorithm for getting the estimation of $V_{mpp\_estimated}$ with the desired precision, it is a property of the algorithm and it is only one value (per precision desired). Several examples of the tracking time are reported in the review [5].

4.2 THE HC AND THE HC MODIFIED

4.2.1 *The HC algorithm*

The HC algorithm is simple, and its logic is illustrated in Figure 21: the HC moves the voltage reference in one direction and then does it again if that direction increased the power obtained, otherwise the voltage reference is moved in the opposite direction.

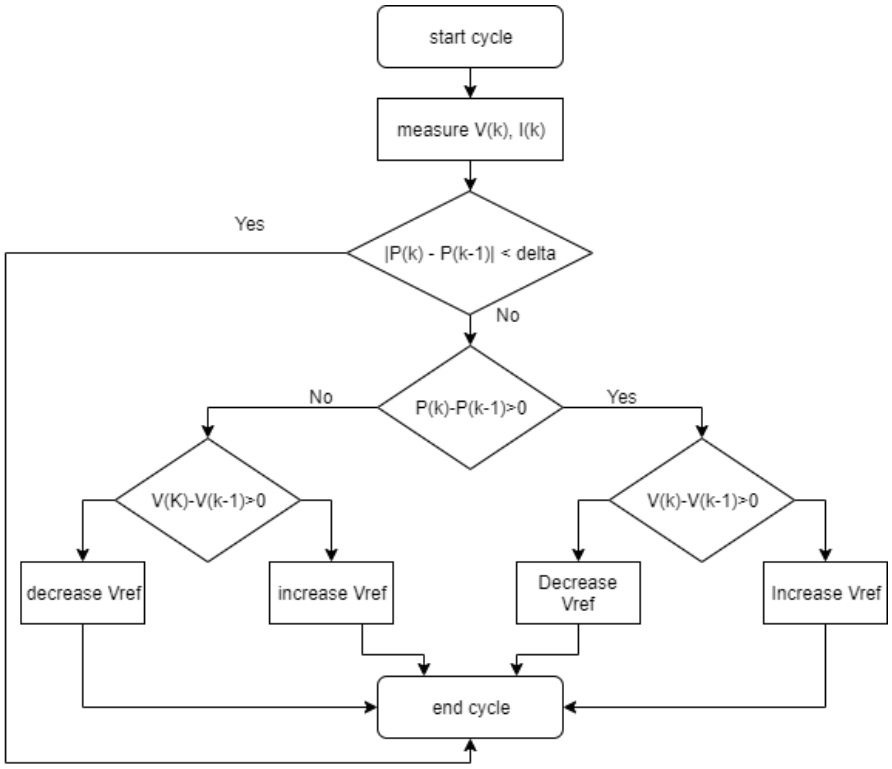


Figure 21: Hill climbing algorithm, logic flux diagram. Delta is a sufficiently little power interval.

There are two important drawbacks of this algorithm:

- when the $V_{mpp\_estimated}$ is close enough to the $V_{mpp\_theoric}$, the HC still explores around that voltage to look for better points; this exploration creates an oscillation in the power generated

- sometimes happen that the algorithm is stuck in a local maximum point, but not in the global one; the curve of the Power, as shown in chapter 2.0.4, has multiple local maximums, and only one global maximum. This event (see Figure 22) actually happens enough times to decrease the average efficiency, specially if the new $V_{mpp\_theoric}$ is far from the previous one and the change of irradiances was quick.

On the other hand, there is a really positive aspect: when a slow change in the PV curve happens, for example because of the temperature or the aging effect, then the HC algorithm follows perfectly the new global maximum.

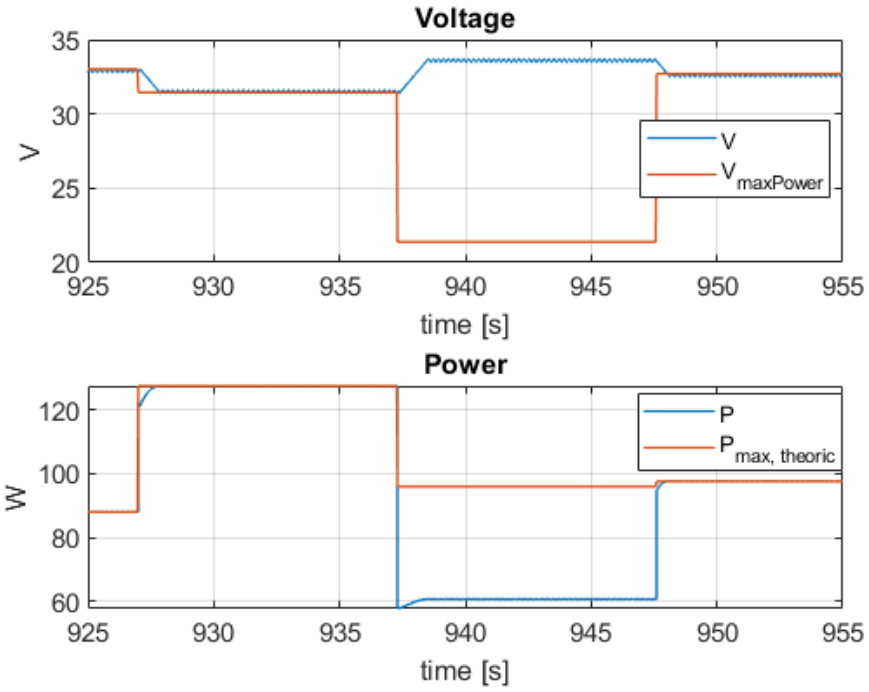


Figure 22: Hill climbing algorithm, example. At every exploration the voltage increase or decrease of 0.1 V and one Perturbation and Observation needs 0.1 s. From $time = 927$ s to $t = 937.5$ s the irradiances $\lambda_1, \lambda_2, \lambda_3 = 515; 487; 554$ W/m^2. From $time = 937.5$ s to $t = 948$ s the irradiances $\lambda_1, \lambda_2, \lambda_3 = 539; 215; 763$ W/m^2. It is illustrated the situation when the MPP found by the algorithm is not the Global MPP.

4.2.2 *The HC modified algorithm*

In this thesis two HC algorithms are simulated: the first one is the classic version of the algorithm, already presented in chapter 4.2.1. The second version is an improved[16] version of the first one. The purpose of this modified algorithm is to solve the main issue of the HC, that is described in Figure 22: the new algorithm in fact should be able to find the Global MPPT, not only local ones.

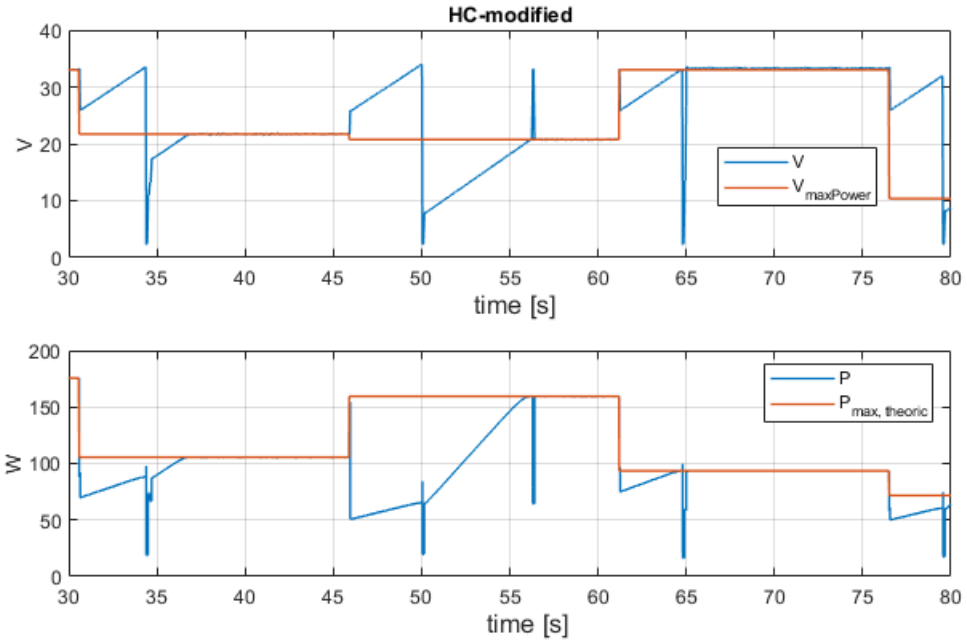


Figure 23: An example of the HC-modified algorithm working. At every exploration the voltage increase or decrease of 0.1 V and one Perturbation and Observation needs 0.1 s.

The logic of the HC-modified is the following, repeated at most as many times as many Bypass Diodes present in the circuit:

- search for a MPP around a certain V_{ref} using a simple HC
- when a local maximum (LMPP1) is found, explore some random work-point whose voltages are far from the V_{LMPP1}

- if some points with bigger power than P_{LMPP1} are found, then the algorithm will move the V_{ref} on that working-point
- another classic HC search starts, and find V_{LMPP2}.
- repeat the sequence comparing the power of LMPP2 with the new points

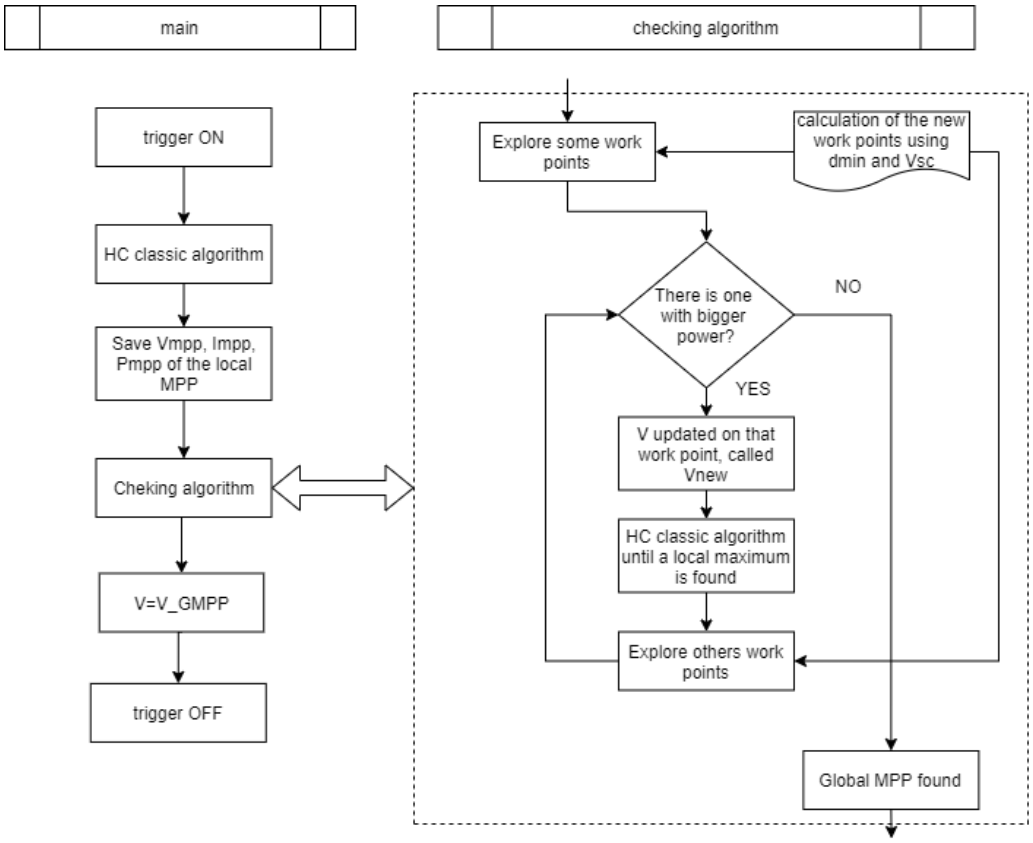


Figure 24: The logic diagram of the HC-modified algorithm. *Vmin* is the V_{sc}. *dmin* is the "distance" estimated between the power peaks in the P-V curve: *dmin*=V_{oc}/number of bypass diodes.

For a better understanding of the algorithm, two logic-flux diagram are illustrated in Figure 24. The *checking function* is not really implemented, but it is useful to help the reader to understand better the program.

4.2.3 *The results*

The two HC algorithms are tested using different exploration time. The panels are irradiated by the irradiances that are randomly generated following the uniform distribution. The average efficiency has been measured for every exploration-time used, as showed in table 4. The exploration time is measured in **samples**: every sample need 0.1 *s*, and corresponds to one Perturbation and Observation.

Exploration time [samples]	3	4	5	6	7
Efficiency of HC	0.879	0.895	0.896	0.897	0.898
Efficiency of HC modified	0.743	0.751	0.756	0.761	0.768
Exploration time [samples]	8	9	10	15	20
Efficiency of HC	0.902	0.903	0.903	0.904	0.904
Efficiency of HC modified	0.773	0.777	0.782	0.810	0.838
Exploration time [samples]	25	30	40	60	
Efficiency of HC	0.904	0.904	0.904	0.904	
Efficiency of HC modified	0.867	0.843	0.873	0.979	

Table 4: Comparison of the efficiencies between HC and HC modified, while the exploration time changes. The exploration time is measured in samples. Each sample needs 0.1 *s*, and corresponds to one Perturbation and Observation.

The results in Table 4 are illustrated in Figure 25 in order to analyze the trend of the efficiency. From Figure 25 it is possible to observe that the HC is not a good algorithm, even if the given exploration time is big. In fact ususally the HC algorithm is used in hybrid algorithms, not alone. Viceversa The HC modified could be an efficient algorithm (after 60 Perturbations and Observations the efficiency is almost 98%) but only if the exploration time is bis enough: in fact if the exploration time allows 40 or less Perturbations and Observations the efficiency is lower

than 87%. This efficiency is not good if compared to the other algorithms presented in the reviews [15],[5].

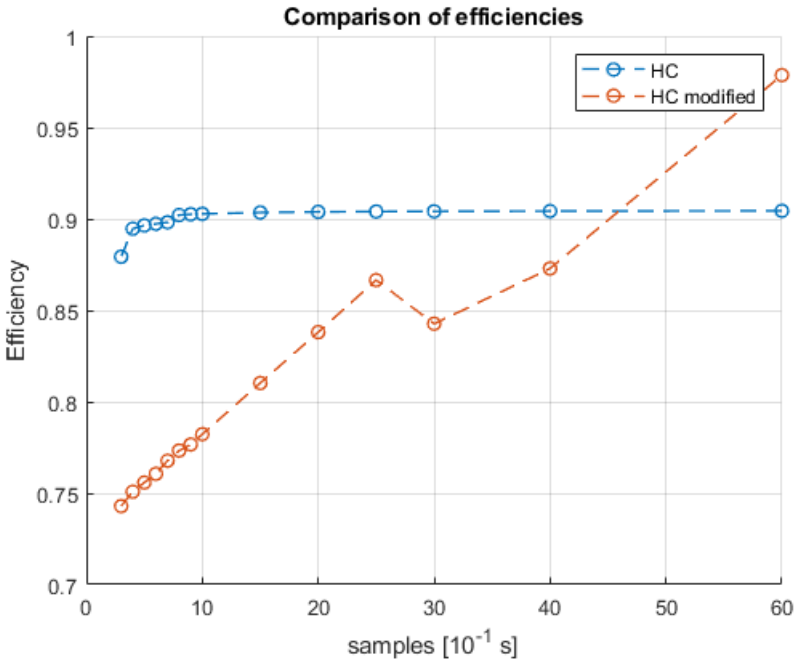


Figure 25: The trend of the efficiencies of the HC and HC modified algorithm related to the available exploration time

In Figure 26 the operation of the two HC algorithms is represented. From this figure it is possible to explain two facts that were already observed from Figure 25:

- the HC efficiency does not improve even if the exploration time is increased because the $V_{mpp\_estimated}$ often is stuck in local MPP
- the HC modified algorithm is not able to complete its exploration if the exploration time is too short (see Figure 52 in the Appendix), and the partial estimation that come out from the algorithm is not good enough to get a good efficiency; but if the exploration time is sufficient (Figure 26) then the algorithm completes the exploration and the final efficiency is good

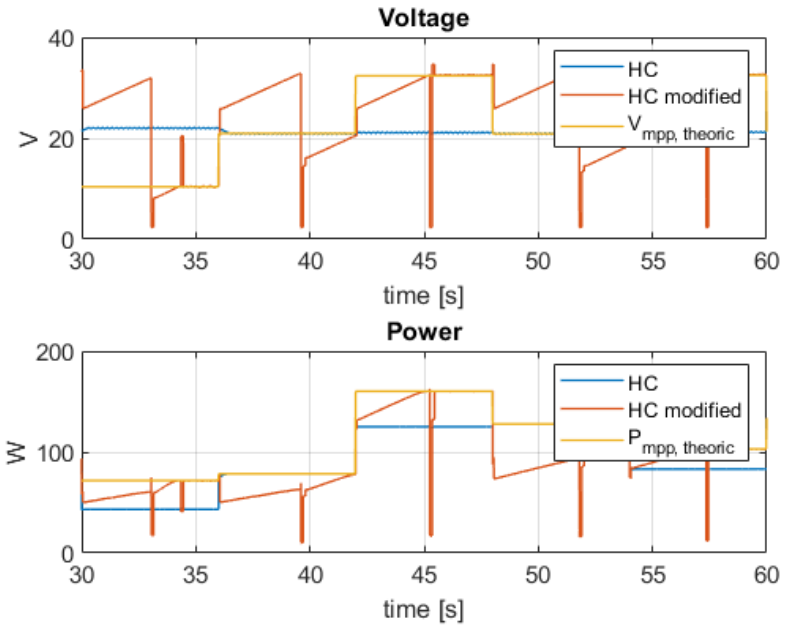


Figure 26: An example of the operation of the HC algorithms. The exploration time is 6 s. One Perturbation and Observation takes 0.1 s, then the Perturbations and Observations are 60. The average efficiency of the HC is 0.904, of the HC modified is 0.979.

4.3 THE PARTICLE SWARM OPTIMIZATION

4.3.1 *The algorithm*

PSO is a simple, intelligent optimization and a meta heuristic approach. It was proposed by Eberhard and Kennedy in 1995[4]. PSO is a type of Evolutionary Algorithm search optimization technique, originated while observing groups of birds solving the difficulties involved in optimization together. It has been invented in order to explore a N-dimensional space that is mostly unknown and to find the global optimum point of this space. In PSO there is a group of particles that explore the space: these particles communicate among the group in order to get the optimum faster.

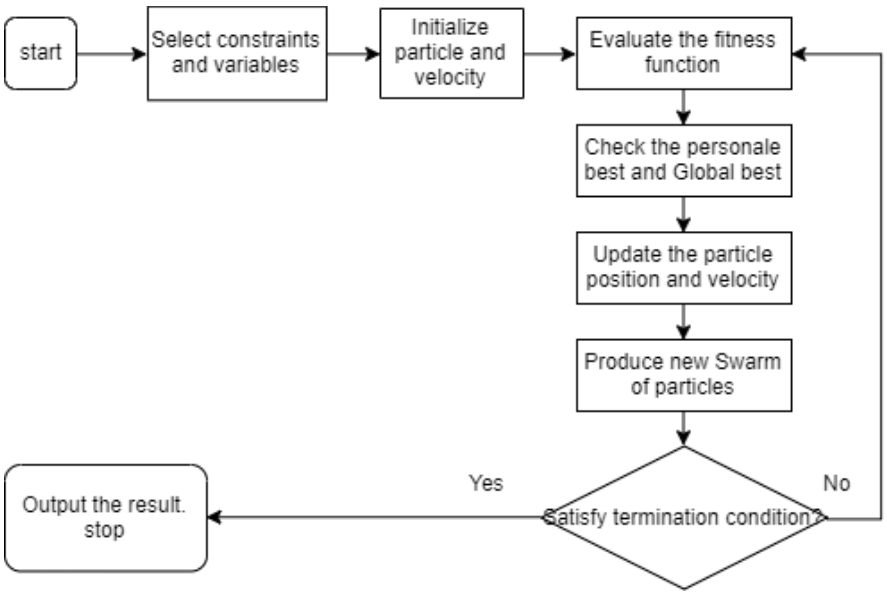


Figure 27: PSO-algorithm logic flow chart.

There are many parameters that have to be set by the programmer that can influence the overall performance of the algorithm, and they are:

- n_p: number of particles
- w_g: weighting function
- n_k: maximum number of iterations.
- w: inertia of the particles.
- c_1: cognitive coefficient
- c2: social coefficient

These parameters have been set following the suggestion of article [13] in order to get a good operation of the algorithm. A simple flowchart that represent the logic of this algorithm is in Figure 27.

It is not the goal of this thesis to analyze deeply this algorithm. It has been used just to have a useful comparison between the results of one of the most used Evolutionary algorithms and the algorithms studied.

4.3.2 *The results*

Like during the test of the HC algorithms the PSO has been tested using different exploration time. The irradiances are again randomly generated, following the same uniform distribution as in the chapter 4.2. Again the average efficiency has been measured for every exploration-time used, as showed in table 5. The exploration time is measured in **samples**: every sample needs 0.1 s, and corresponds to one Perturbation and Observation.

Exploration time [samples]	3	4	5	6	7
Efficiency of PSO	0.926	0.901	0.966	0.970	0.977
Exploration time [samples]	8	9	10	15	20
Efficiency of PSO	0.945	0.959	0.975	0.983	0.988
Exploration time [samples]	25	30	40	60	
Efficiency of PSO	0.991	0.993	0.997	0.998	

Table 5: Efficiencies of the PSO, while the exploration time change. The exploration time is measured in samples. Each sample needs 0.1 s, and corresponds to one Perturbation and Observation.

The results in table 5 are illustrated in Figure 28, where the trend of the efficiency can be observed. From the picture it is possible to observe that the efficiency improves if the exploration time increases. The efficiency is bigger than 98.9% if the exploration time allows more than 20 Perturbation and observations.

In order to have an intuitive idea of how the algorithm works, two example of the operation of the algorithm are reported in Figure 30, 29. It is possible to observe that a low number of Perturbation and Observations reduce the effectiveness of the algorithm and then the final efficiency, as reported in table 5.

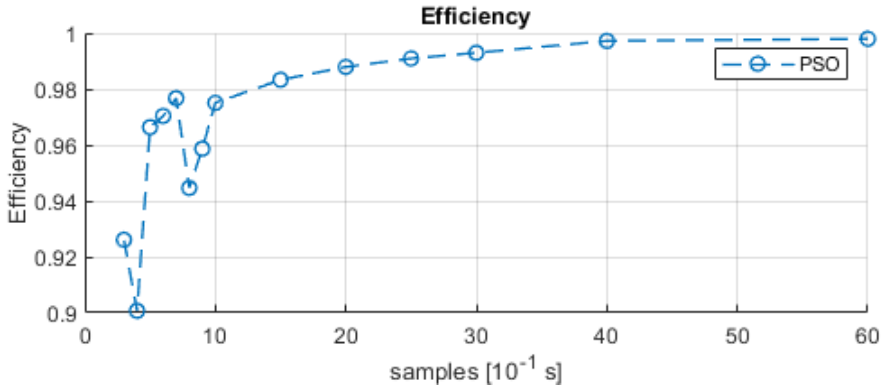


Figure 28: The trend of the efficiencies of the PSO algorithm related to the available exploration time.

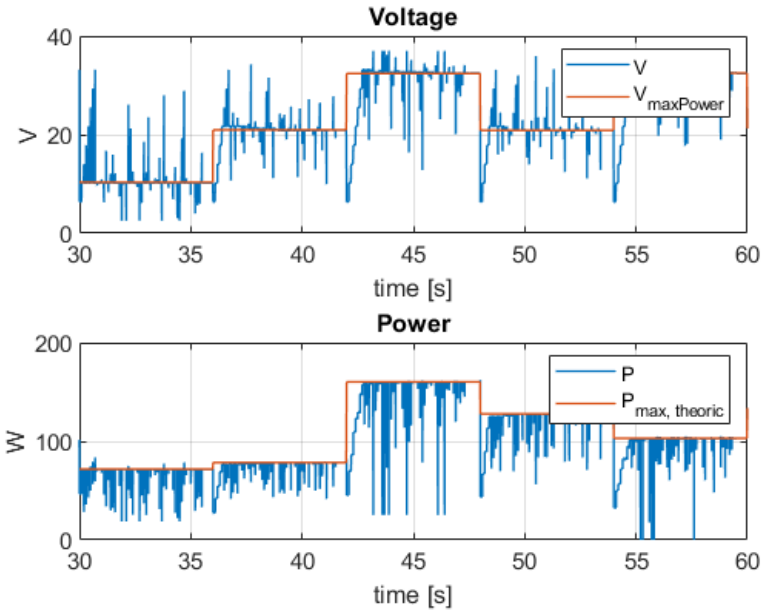


Figure 29: An example of the operation of the PSO algorithm. The exploration time is 6 s. One Perturbation and Observation takes 0.1 s, then the Perturbations and Observations are 60. The average efficiency of the PSO in this case is 0.998.

4.4 THE ANN

4.4.1 *The ANN algorithm*

An MTTP algorithm based on ANN works doing two different actions: **exploration** and **exploitation**. During the exploration

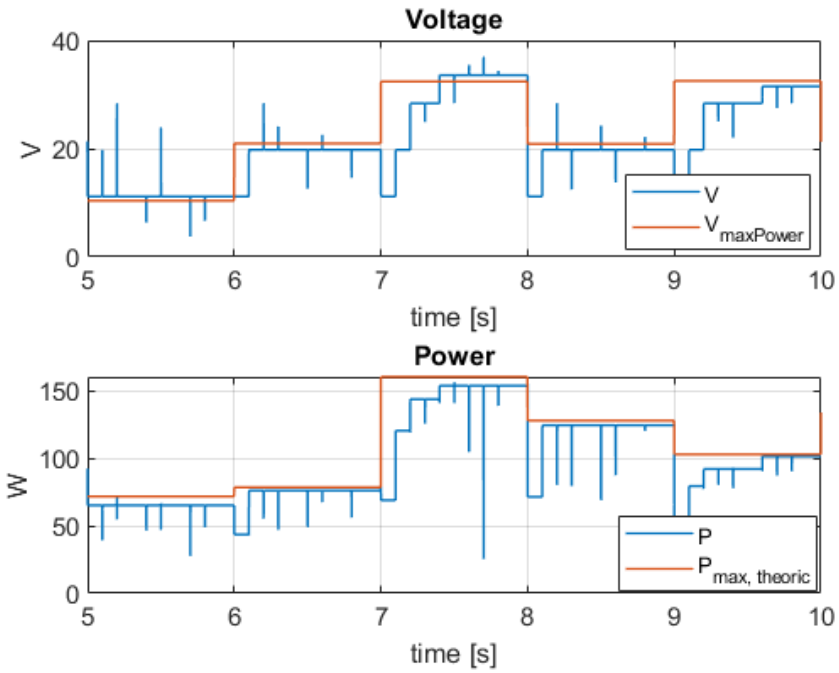


Figure 30: An example of the operation of the PSO algorithm. The exploration time is 1 s. One Perturbation and Observation takes 0.1 s, then the Perturbations and Observations are 10. The average efficiency of the ANN in this case is 0.983.

a definite number of couples (V, P), respectively voltage and power, are saved and merged in a vector X. After the collection of the samples, the completely full vector X will be the input of the ANN, that will "exploitate": the network returns as output the $V_{mpp\_estimated}$. In Figure 31 the exploitation and the exploration process are illustrated, and in Figure **??** the logic flow diagram is represented.

As in the HC modified and in the PSO simulations the exploration of the ANN starts every time the irradiances change. Then the efficiency is calculated at the end of every exploitation, and the average efficiency of the algorithm is finally the average of all the 500 different ones.

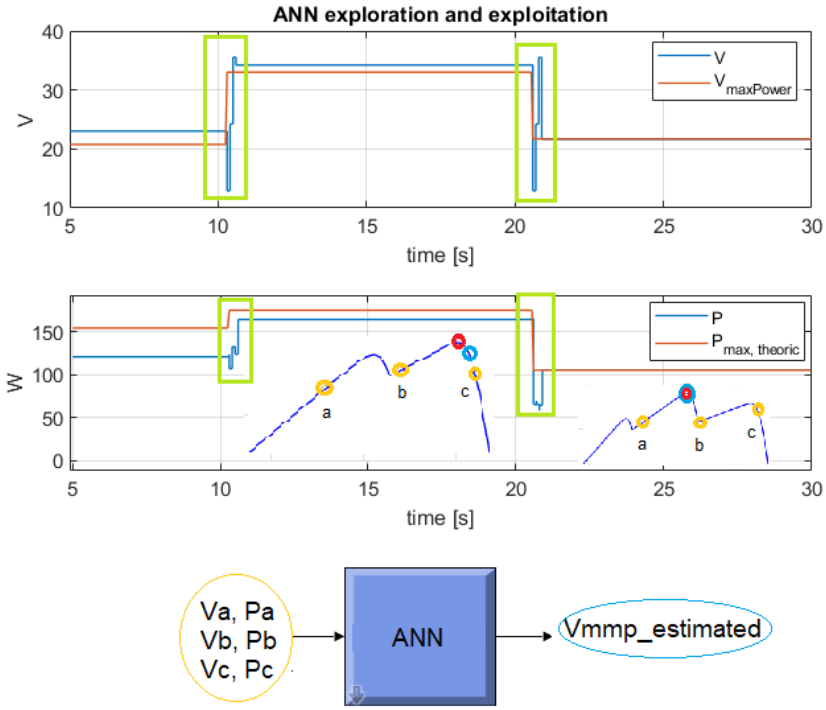


Figure 31: Operation of the ANN algorithm: exploration (green rectangles) and immediately after exploitation. In this image two different Power-Voltage curves, which refer to two different irradiances condition, are illustrated. The orange circles represent the (V, P) samples (three samples) of the exploration, the red circle the Real Global MPP, the blue circle the work-point estimated by the ANN. The output of the ANN is the $V_{mpp\_estimated}$, that is the voltage of the working point during the exploitation.

4.4.2 *The results*

The ANN has been tested using different exploration time and different irradiances, following the same methodology used for the previous algorithms. The efficiency obtained are represented in table 6 and Figure 34. Two examples of the operation of the ANN are represented in Figure 32,33, where it is possible to visualize what was clear from the numbers in table 6: even if the ANN uses few samples still the estimation of the $V_{mpp\_theoric}$ is good, and consequently the average efficiency.

Samples	3	4	5	6	7
Efficiency	0.988	0.992	0.988	0.989	0.988
Samples	8	9	10	15	20
Efficiency	0.988	0.989	0.985	0.992	0.988
Samples	25	30	40	60	
Efficiency	0.992	0.984	0.993	0.982	

Table 6: The efficiency of the ANN algorithm, while the exploration time changes. The exploration time is measured in samples. Each sample needs 0.1 s,and corresponds to one Perturbation and Observation.

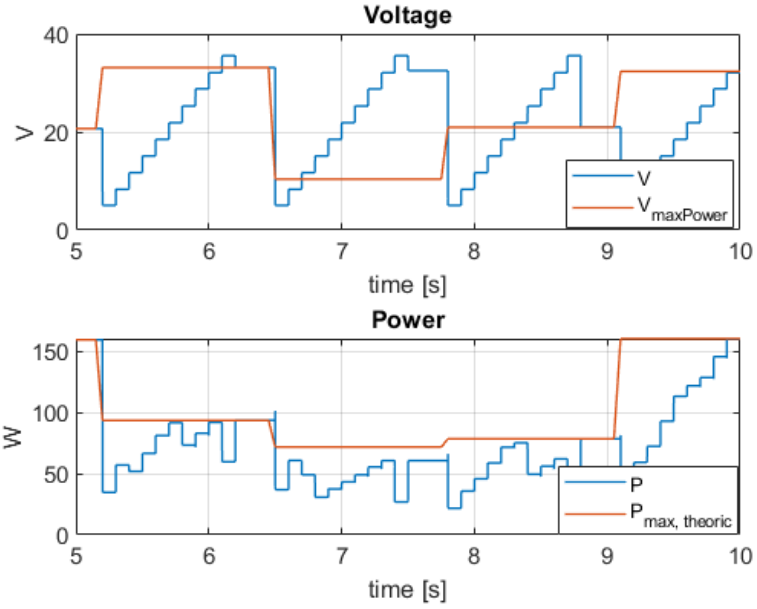


Figure 32: An example of the operation of the ANN algorithm. The exploration time is 1 s. One Perturbation and Observation takes 0.1 s, then the Perturbations and Observations are 10. The average efficiency of the ANN is 0.985

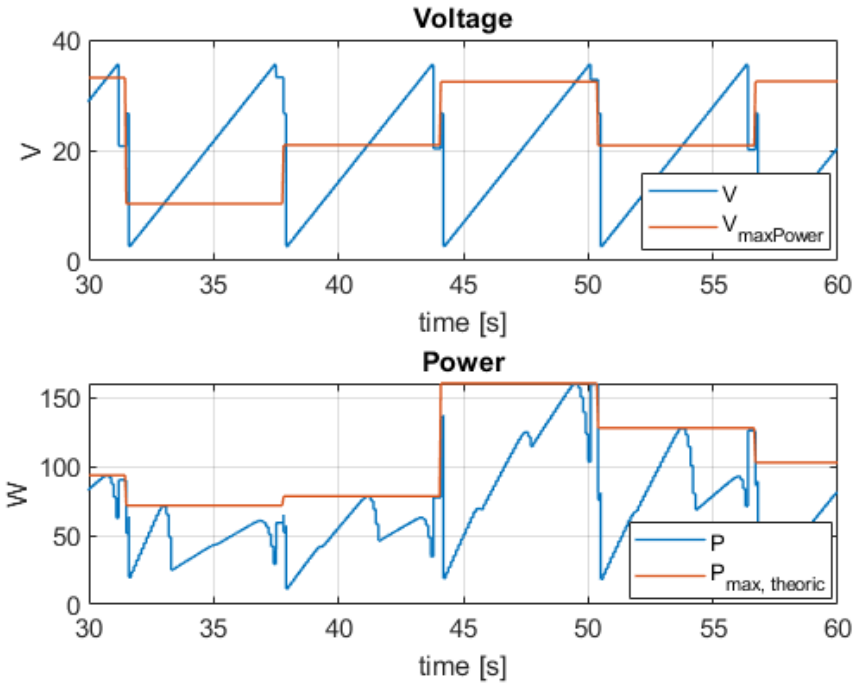


Figure 33: An example of the operation of the ANN algorithm. The exploration time is 6 s. One Perturbation and Observation takes 0.1 s, then the Perturbations and Observations are 60. The average efficiency of the ANN is 0.982.

4.4.3 *Comparison of results: ANN, HC, HC modified, PSO*

The efficiency of the ANN are compared with the efficiencies of the other algorithms, that are the HC, the HC-modified, and the PSO.

Observing Figure 34, two main facts can be observed:

- the ANN algorithm is more efficient than the others (around $98 - 99\%$) when just a few samples of exploration are available. This means that the ANN could be the best choice when a quick estimation of the MPP is needed, for example when the irradiances are rapidly changing
- PSO is the most efficient when there are more than 20 samples available. This means that if more time is available for the exploration, the PSO algorithm is preferable

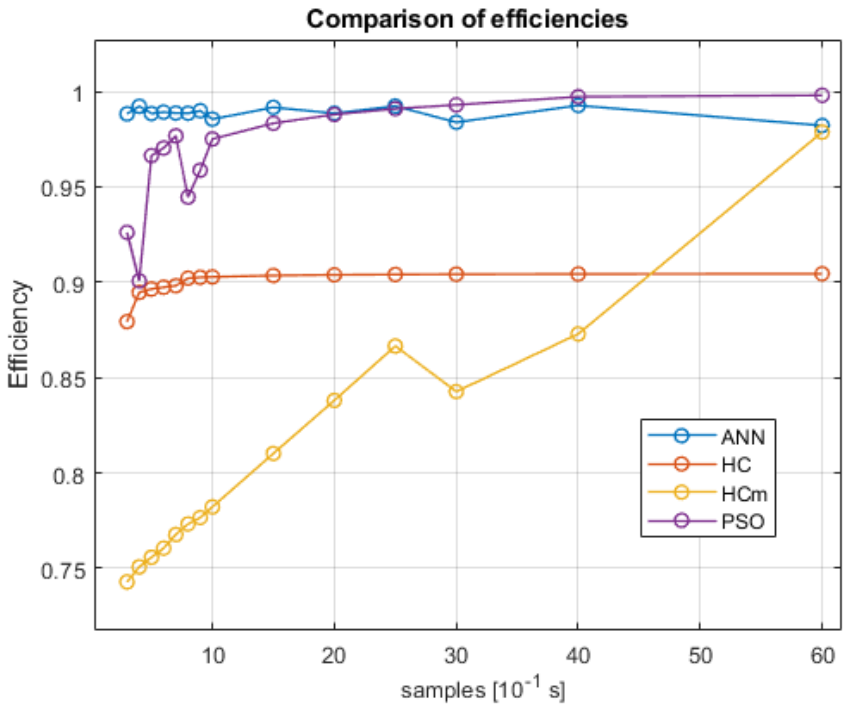


Figure 34: The algorithms' efficiencies compared. The ANN is the most efficient if the exploring time is short. When the samples for the exploration are more than 20 the PSO is preferable to the ANN.

- the HC is not efficient: too often it gets stuck in local MPP, preventing the algorithm to find the global MPP. Also if there is more time available, the efficiency doesn't improve. In fact the HC is used expecially in hybrid algorithms
- the HC-modified could be a good algorithm, but it is really slow. Only after 60 explorations the algorithm start to be competitive with the others.

The comparison so far presented consider the irradiances as the only variable that change the power-voltage curve. But in a real applications also the temperature and the aging of materials affect the P-V curve, even if less than the irradiances. This two effects are not considered in the training of the ANN, and this could decrease the efficiency of the ANN algorithm. In fact it is correct to consider that inside the Neural Network a

"model" is saved, and this model is based on the training data. As it will be described in chapter 3.5.1 the data used for the training don't consider the temperature and the aging effects, because they are assumed not measurable. But the HC can react well to little change of the power-voltage curve, because its tracking action is continuously working. So thanks to this observation an idea come up to the researchers [17]: to create a hybrid algorithm, that combines the good properties of the ANN with the properties of the HC.

4.5 THE HYBRID ANN WITH HC ALGORITHM

4.5.1 *The algorithm*

	ANN	HC
time for exploration	defined	depending on $\lvert V_{ref} - V_{mpp} \rvert$
rapidity	very rapid	generally slow
efficiency	$98 - 99\%$	depending on many factors
sensitive to bad measurements	yes	no
sensitive to system changes	yes	no
triggering strategy influence	high	always triggered
oscillations around MPP	no	yes
reaction to little changes (temperature, irradiances...)	not good	excellent
reaction to wide changes of irradiances	excellent	really bad

Table 7: Positive and negative characteristics of HC and ANN algorithm.

In the previous chapters some characteristics of the single algorithms have been analyzed. In this chapter the ANN and the HC algorithm will be combined together, in order to get an algorithm that combines the positive behavior of the two algorithms, that are showed in Table 7.

In fact while the ANN reacts well to big and quick changes of the irradiances, the HC reacts well to every little disturb (measurements, temperature...). Therefore the combination of the two, even where there are no disturbs, gets an improvements of the performances: the ANN will bring rapidly the V_{ref} close but not overlapped to the global V_{mpp}, and then the HC will move the V_{ref} closest as possible (Figure 35). This second step increase the exploration time, but it is an exploration around the global P_{mpp} so the power extracted is still almost the maximum. A drawback of this algorithm is the oscillation of the V_{ref} around the V_{gmpp}, but can be stopped or reduced by the manufacturer.

The operation of the algorithm is the following, illustrated in Figure 35:

- when a trigger variable is ON, the pure ANN start its exploration
- the pure ANN eploitates, so the V_{ref} becomes the $V_{mpp\_estimated}$ estimated by the ANN. If the ANN worked properly the $V_{mpp\_estimated}$ should be close to the Global MPP, that is the $V_{mpp\_theoric}$.
- the HC algorithm starts its exploration: the voltage reference is moved looking for the local maximum, that should be also the global maximum $V_{mpp\_theoric}$

In the hybrid algorithm the *exploration time* is divided into two different explorations: the ANN exploration, and the HC exploration. It will be a choice for the manufacturer how to distribute the available time between the two algorithms. Two examples of different choices of the ANN exploration time and of the HC exploration time are illustrated in Figure 36,37.

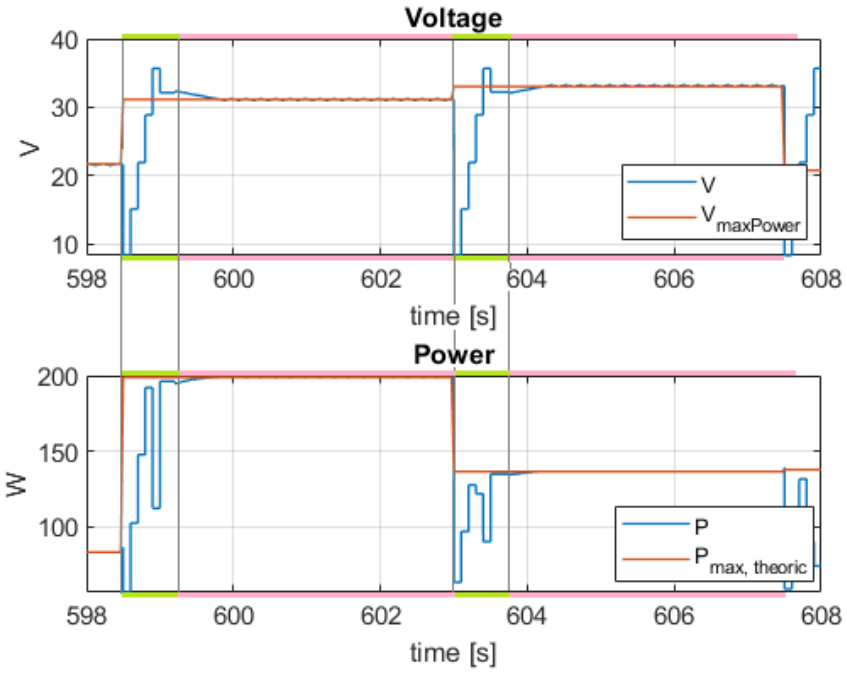


Figure 35: The Hybrid ANN algorithm operation. The green rectangles highlight the ANN exploration (5 perturbations and observations). The pink rectangle the HC exploration(100 perturbations and observations). Note the oscillations during the final exploitation, due to the exploration of the HC.

4.5.2 *The results*

The algorithm obtained is a good improvement of the ANN. One of its most valuable characteristic is that its efficiency is *at least* the efficiency of the ANN, and the HC can only improve it a little bit. In Figure 38 the hybrid algorithm is compared with the pure ANN: in the image the samples of the x-axis are only referred to the ANN exploration in both the algorithms. The hybrid algorithm then requires some extra samples for the HC exploration. In order to give to the HC exploration a sufficient number for the best results 100 extra samples are given. Therefore Figure 38 represents how the efficiency can be at most improved if the HC algorithm is added to the pure ANN.

A final observation on the hybrid algorithm: it counteracts little changes of the temperature or of the system. The good

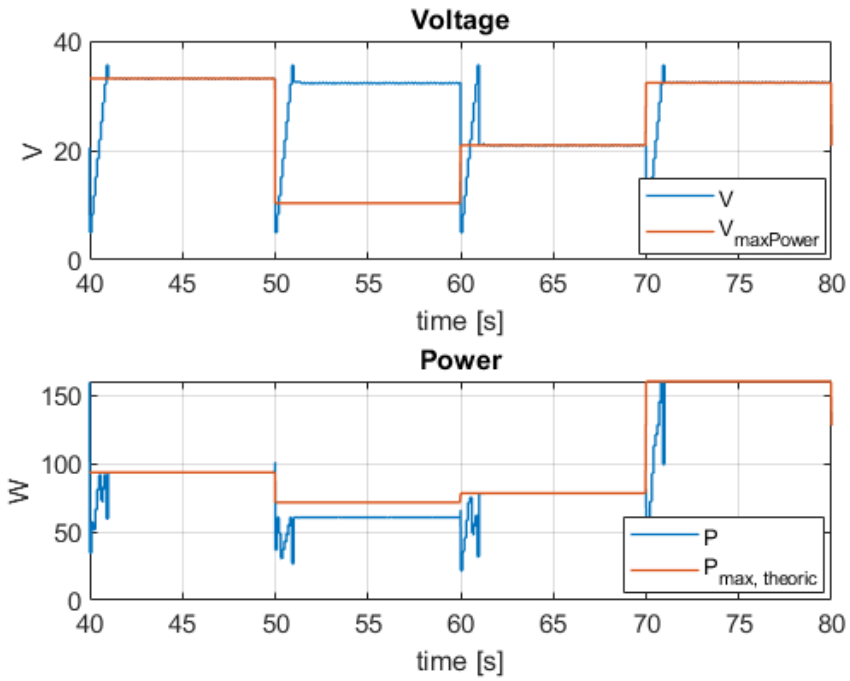


Figure 36: The Hybrid ANN algorithm operation, when the ANN exploration time is 1 s and the HC exploration time is 9 s. The HC algorithm increase or decrease its new voltage reference using steps of 0.1 V.

property of the HC, the continuous tracking of the MPP (if the change is little), is fully used.

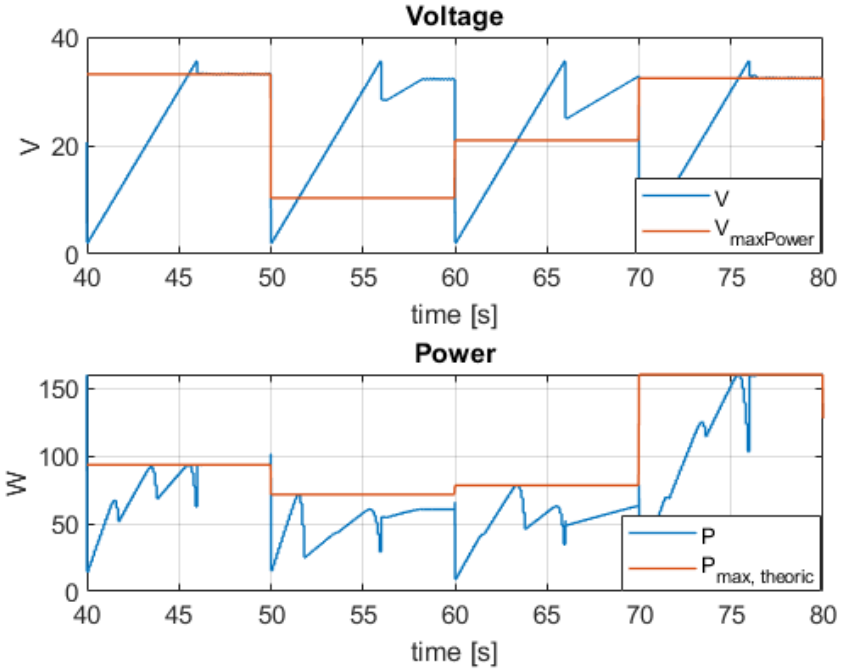


Figure 37: The Hybrid ANN algorithm operation, when the ANN exploration time is 6 s and the HC exploration time is 4 s. The HC algorithm increase or decrease its new voltage reference using steps of 0.1 V.

4.6 GENERAL CONSIDERATIONS

4.6.1 *Different Photovoltaic System*

In order to verify that the results so far obtained are correct in general, a new physical system is tested. This new system is made of five PV modules, and imitate the system used in article [6]. The solar module used in that article is the SPR-E19-320W, that is also one of the models already implemented in the Simulink library. This panels have the characteristics illustrated in the Appendix A in Figure 45,46,47, that are different enough from the modules previously used (Figure 42, 43 in Appendix A). As an example, the algorithm with 2 s of exploration time has been studied (20 samples). The results obtained from the simulations are similar to the ones obtained from the previous

Samples	3	4	5	6	7
Efficiency	0.993	0.993	0.994	0.992	0.993
Improvement	+0.5%	+0.12%	+0.55%	+0.25%	+0.47%
Samples	8	9	10	15	20
Efficiency	0.994	0.993	0.993	0.993	0.991
Improvement	+0.50%	+0.26%	+0.74%	+0.11%	+0.22%
Samples	25	30	40	60	
Efficiency	0.993	0.995	0.993	0.991	
Improvement	+0.10%	+1.08%	+0.04%	+0.92%	

Table 8: Efficiency obtained using the Hybrid ANN algorithm. The samples reported in the table are dedicated only to the ANN exploration. The HC algorithm was free to use 100 more samples for its exploration. The new efficiency is compared with the efficiency in the pure ANN, and the improvement is reported in the table.

system: using an ANN algorithm alone the maximum efficiency (97%) is obtained using the rectangular architecture with 4 neurons, [4 4 4]. The hybrid algorithm using the same ANN can get a 98.9% efficiency, which again is similar if compared to the previous system. So it is possible that the algorithm so far studied could work for different photovoltaic systems, but needs to be verified every time.

4.6.2 *Re-training online*

In a real application of the ANN algorithms there is the problem of how to create the datasets that will train the net. Often it is difficult to create a sufficiently accurate model of the irradiances, and that affect the final performances of the net. Another problem come up with time: the aging of the materials make the model initially "estimated" in the ANN not good anymore. So

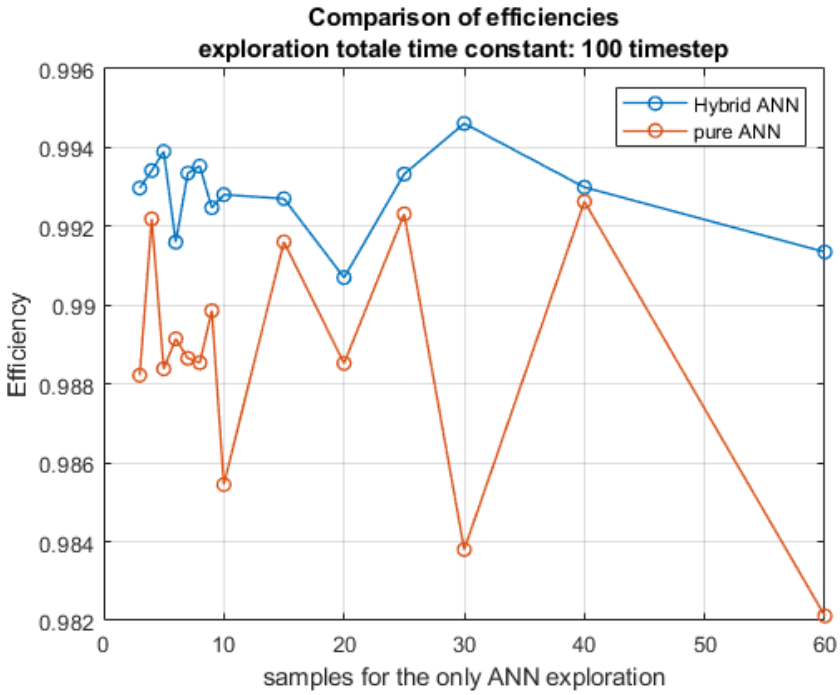


Figure 38: The Hybrid ANN efficiency compared with pure ANN. In this picture the x-axis represents the samples dedicated only to the ANN exploration. The Hybrid algorithm improves the ANN, but needs time for it, so additional 100 samples are given only to the hybrid algorithm.

a new procedure can be used in order to deal these problems: the **retraining online**.

The retraining online is based on the following property of the hybrid algorithm: it basically "corrects" the estimation of the pure ANN, thanks to the HC algorithm properties. Therefore the idea is to use this corrected $V_{mpp\_estimated}$ for training a new ANN, using a sufficient number of new data collected. A new net can be trained offline by another processor, while the first net is still working. In this way probably the new trained net could be more performing than the first one.

During the analysis of this strategy the comparison is between the efficiency of the pure ANN and of the hybrid ANN with HC algorithm. The retraining procedure follows the next steps, listed in Table 9:

- train and test the pure ANN and the hybrid algorithm, with the aging resistance (which represents the aging of the panels, as will be explained in chapter 5) $R = 0\ \Omega$. That means that the panels are new
- train and test both the algorithms using a system aged: the series resistance is increased by $R = 0.1\ \Omega$. The performances of the ANN decrease, the performances of the hybrid algorithm not significantly. While the hybrid algorithm works, it also collects data that will be used for the re-training
- using the data collected from the hybrid algorithm a new ANN is created, and tested on the same system in order to verify that the retraining action increased the efficiency. The performances of the new ANN are improved, as it is possible to observe in Figure 39.
- this procedure restart from the second step.

Four attempts to verify that the retraining method works have been executed: at every attempts the system is aged and this aging is represented by the aging resistance, whose value increases by $0.1\ \Omega$. The results are reported in Table 9 and Figure 39. This number of attempts is not sufficient to have a complete analysis of this method, but some observations can already be done.

In Figure 39 it is possible to observe an unexpected fact: even if the pure ANN improves his performances thanks to the retraining, the hybrid algorithm on the other hand decrease his efficiency. A possible explanation for this fact is that the $V_{mpp\_estimated}$ of the pure ANN (that is also the $V_{mpp\_estimated}$ by the ANN in the hybrid algorithm but before the correction of the HC algorithm) is more often far from the Global MPP but closer to other Local MPP, after every retraining. This idea would explain why the pure ANN is more efficient with retraining (because instead of estimating random work points at least

Step	Dataset used for training	Algorithm	System aging [Ω]	Efficiency	Level of efficiency
1	Initial	pure ANN	0	0.959	low
	Initial	Hybrid	0	0.997	high
	Initial	pure ANN	0.1	0.966	low
	Initial	Hybrid	0.1	0.996	high
2	First	pure ANN	0.1	0.982	medium
	First	Hybrid	0.1	0.994	high
	First	pure ANN	0.2	0.969	low
	First	Hybrid	0.2	0.994	high
3	Second	pure ANN	0.2	0.987	medium
	Second	Hybrid	0.2	0.991	high
	Second	pure ANN	0.3	0.983	medium
	Second	Hybrid	0.3	0.991	high
4	Third	pure ANN	0.3	0.986	medium
	Third	Hybrid	0.3	0.989	medium

Table 9: Procedure of test and results of the retraining, step by step. There are three lines highlighted: during that tests the new datasets for the retraining are collected, and the following networks are retrainined with these new datasets collected. In this table can be observed that the retraining improves the efficiency of the pure ANN algorithm, but decreases the efficiency of the hybrid one.

esteems local MPP) and also why the hybrid algorithm is less efficient with training.

From Figure 39 the conclusion is that is more efficient to not retrain the networks. The hybrid algorithm that uses the ANN trained only one time is able to get a high efficiency even if the system is changing with time. In article [1] it is reported that after 20 years the series resistance that represents the aging of the materials of the panels can be about 12.8% times the internal series resistance of the panels. The value of the internal series resistance of the panels used in this thesis is 0.296 Ω and then the aging resistance after 20 years of usage can be approximated at 0.058 Ω. Therefore the hybrid algorithm is probably

able to keep his efficiency also after 20 years. It has been tested using as aging resistance $R = 0.1, 0.2, 0.3, 0.4\ \Omega$ and the efficiency was always above 99%.

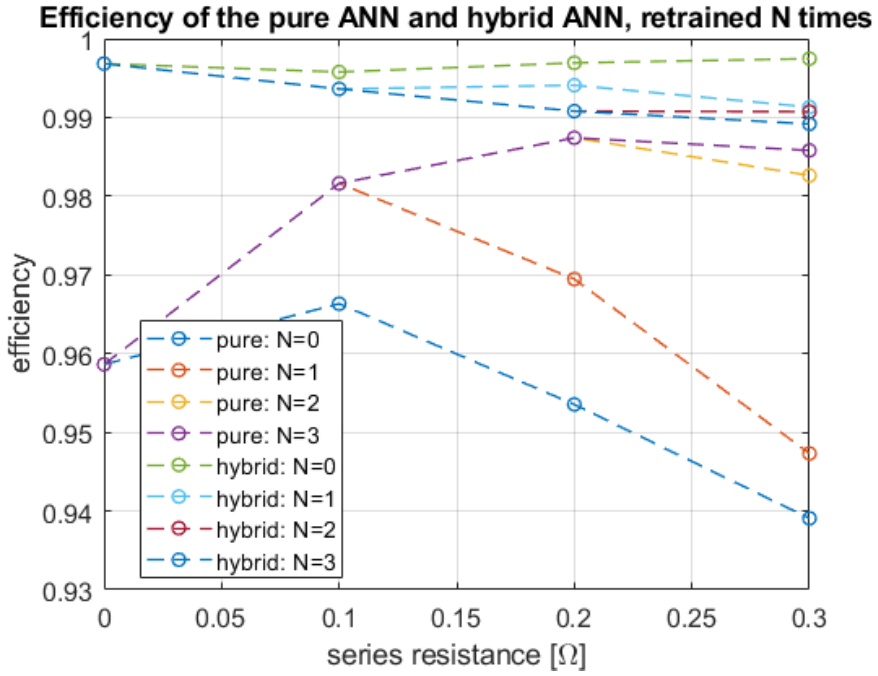


Figure 39: The trend of the efficiencies of the algorithms while time pass, and the aging of the material are affecting the power generation. The retraining process improve the efficiency of the pure ANN algorithm but decrease the one of the hybrid algorithm.

4.6.3 *Triggering Strategies*

The triggering strategy consists in the design choice about when to make every algorithm to start the exploration. This event can happen periodically, or when the controller receive some input from the environment, and this strictly depends on the real application and the manufacturer.

The triggering strategy can affect the final efficiency in a different way depending on the algorithm used. If the MPPT algorithm is the HC then basically exploration is also the exploita-

tion, so the triggering strategy doesn't affect the efficiency. If the algorithm used is the HC-modified, ANN, Hybrid ANN, or PSO then yes: during the exploration some power is lost, and in some situations it is not negligible. Comparing the two most efficient algorithms, that are the PSO and the Hybrid ANN, then it is clear that the best option is the Hbryid ANN, for two reasons:

- this algorithm can detect little changes also during the exploitation (because the HC algorithm is working), so it is not necessary to start a new exploration too often
- the exploration of the ANN is faster then the PSO, and this means that there are smaller oscillations in the output power and voltage.

5

SIMULATION PROGRAMS

In this chapter is presented the Simulink model of the physical system and the environment that is used for the simulations, that is slightly different from the model in Figure 11. In fact a too accurate model in Simulink requires too much time to be simulated (more then two hours) due to the high dynamics of some devices, but thanks to some assumptions and approximations the new model requires only from few seconds to maximum forty minutes (depending on some parameters). In this chapter are presented the programs implemented for testing the MPPT algorithms, and it is explained in detail how the calculations are executed.

5.1 THE PHYSICAL MODEL

As in Figure 11, the main components of the physical system are: the DC/DC converter, the Load, the controller, the MPPT algorithm, the photovoltaic panels, the environment. In the simplified model that have been used in the simulations every component has his needed some assumptions, as follow.

LOAD The photovoltaic panels are Direct Current Power sources, then between the panels and the load (could be the elctric grid, or different stand-alone applications) some DC/DC or DC/AC converter are usually needed. These converters from the panels point of view are a voltage source, that can control the output voltage of the panels. Therefore they are represented by a constant voltage source.

CONTROLLER The controller is not completely simulated: it is approximated using a first order system, which include also the DC/DC converter circuit. The time constant of the system is assumed to be 100 µs, that means a bandwidth of 1590 Hz. This is the highest value obtained from the simulation without choosing a too small timestep.

MPPT ALGORITHM This function block receives as input the Voltage and the Current corresponding to the last V_{ref} imposed on the circuit, and its eventual delay is included in the first order system already mentioned.

PHOTOVOLTAIC PANELS they are the default model of Simulink, their parameters are in the Appendix 42,43,44. The problem considered includes three panels in series (Figure 40).

AGING OF THE MATERIALS this effect is simply modeled as an extra series resistance for every panel that decrease the final power obtained, as you can see in Figure 40. The aging resistance model is studied and proposed in article[1].

ENVIRONMENT The environment is assumed to have constant temperature, 25°C, in order to consider only the variation of the irradiances as an environment factor. In a real application the temperature changes slowly so it is reasonable to assume it as a constant, and his effect on the efficiency of the MPPT algorithm will be studied in the chapter 4.5. The same considerations are valid for the aging effect on the panels. To have a complete case-study, two dataset of irradiances are considered: one dataset of irradiances that are **uniformly** distributed around their mean value (600 W/m^2) and one dataset whose irradiances are **normally** distributed around their mean and variance (respectively 600 W/m^2, 160 W/m^2). These numerical values are chosen referring to the studies [18] and [17]. These

two type of dataset have been used to train and also to test the Neural Networks.

In Figure 40 is illustrated the fast model used in the simulations.

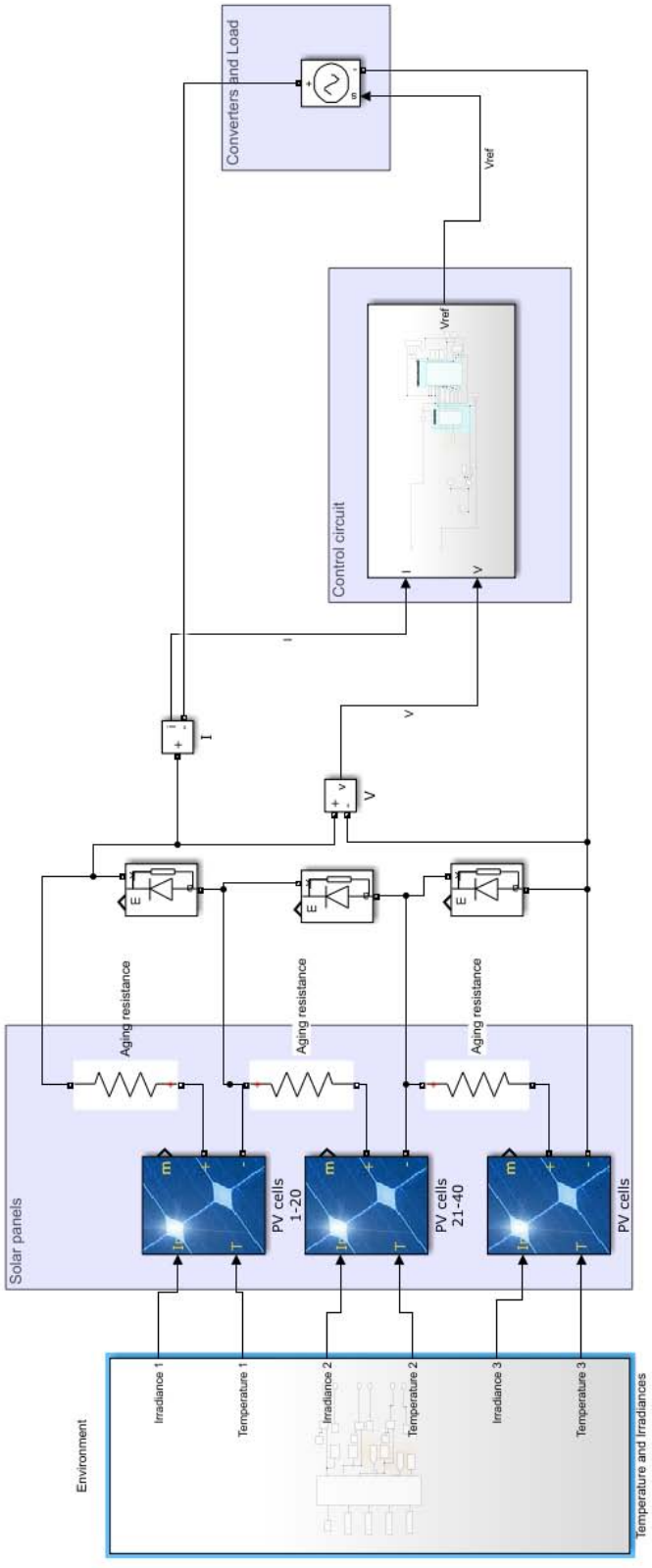


Figure 40: An illustration of the model used for the simulations

5.2 SIMULINK PROGRAMS

In this section the two main Simulink programs are presented: the
`dataset_creator`, which simulate the system and save the parameters for the training of the ANN, and the `dataset_tester`, which test the considered MPPT algorithm and calculate his performances. The MPPT alogorithms tested and their relatives code are presented in detail.

5.2.1 *Dataset creator*

In this section is described the program that will create the data for the training. These data, in the form of a two-dimension array, should include

- the power measurements corresponding to the V_{ref}, that will be the input data for the training
- the real V_{mpp} that the ANN is supposed to give as output. These are the target outputs.

This matrix should contain a sufficient amount of different power-voltge curves, to train the most general ANN. This amount, following the article [18], is decided to be 500. A full image of the Simulink program is at the end of the chapter, Figure 48.

Basically the program will do the following actions 500 times:

- create a triple of Irradiances, and use it on the panels
- scan the whole Power-Voltage curve of the panels
- sample the P-V curve, and save also the V_{mpp} and the P_{mpp}
- store these data in one row of the final matrix.

At the end the matrix will be saved in the workspace of the computer, in order to keep this data available for future utilization.

5.2.2 *The dataset matrix*

As explained in chapter 4.4, during the ANN *exploration* a definite number of samples (V, P) are collected. This number can be decided by the manufacturer, and the variable that stores it is called NSAMPLE: for example in Figure 31 you can observe that NSAMPLE $= 3$. The dataset for the training of the ANN used in Figure 31 is illustrated in Table 11.

Irradiances $[W/m^2]$			Voltage reference $V_{ref}[V]$		
686.03	238.58	651.00	12.89	24.21	35.54
384.02	1,172.54	530.63	12.89	24.21	35.54
580.14	714.36	716.06	12.89	24.22	35.54
714.76	707.44	825.45	12.89	24.22	35.54
647.02	716.30	678.22	12.89	24.22	35.54

Table 10: Some examples of the irradiances used and voltage reference used, when NSAMPLE $= 3$. These values are a little number, the real matrixes are 500 rows big.

Power measured $[W]$			V_{mpp} $[V]$
72.09	49.80	48.17	20.76
58.90	80.10	76.88	33.34
78.77	120.79	94.21	32.08
79.21	146.82	109.55	31.45
75.11	134.46	97.15	31.45

Table 11: Example of dataset training matrix obtained from the parameters in Table 10, when NSAMPLE $= 3$. The illustrated table is an portion of the real dataset training matrix, that is 500 rows long.

The $(\text{NSAMPLE} + 1)$-column of the dataset training matrix contains the Target output values, that are the real V_{mpp}. As

you can see in Table 11 in every row different values of Power and V_{mpp} are collected: that's because for every row a different triplet of irradiances are used on the photovoltaic panels, see Table 10. Two examples of the collection of two different dataset training matrixes are illustrated in Figure 41, where it can be observed that many different triplets of irradiances are tested on the circuit. The respective power-voltages curves are completely explored, then sampled (3 times in the first picture and 13 times in the second picture, because NSAMPLE is respectively 3 and 13.) and the MPP point is found.

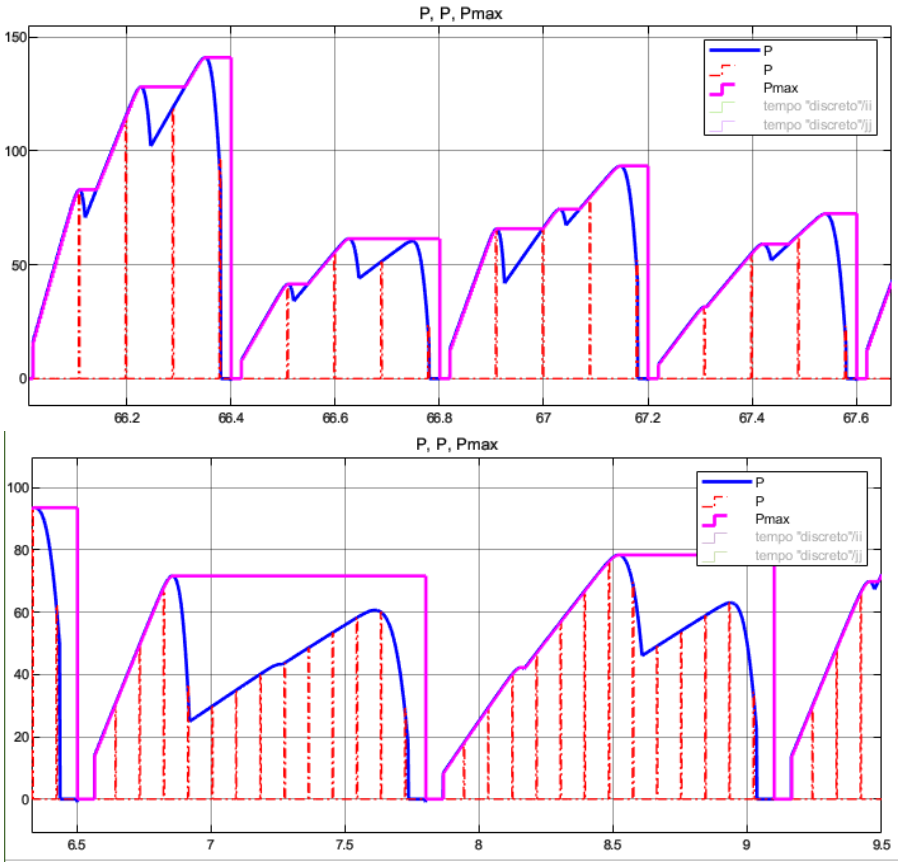


Figure 41: Illustration of the creation of two different training matrix in Simulink, respectively when NSAMPLE = 3 (first picture) or NSAMPLE = 13 (secondo picture). The respective V_{mpp} saved in matrix 11 are the ones which correspond to the MPPs highlighted by the purple line.

5.2.3 *The irradiances*

The irradiances are different for every panel, so a new triplet of irradiances creates a new Power-Voltage curve. The irradiances are generated randomly, following two different distribution. One dataset has the irradiances that are **uniformly** distributed around their mean value ($600W/m^2$) and the other one **normally** distributed around their mean and variance (respectively $600W/m^2, 160W/m^2$). These numerical values are chosen referring to the studies [18] and [17].

5.2.4 *The voltage references*

The V_{ref} used for the sampling of the power-voltage curve are chosen following a simple formula 7, where $1 \leqslant n \leqslant NSAMPLE$. Tthey are uniformly distributed between V_{sc} and V_{oc}.

$$V_{ref}(n) = \frac{0.9V_{oc}}{NSAMPLE}n + 0.05V_{oc} \tag{7}$$

A raffiguration on how the V_{ref} are distributed along the Power-Voltage curve is in Figure 41.

5.2.5 *The Simulink Functions*

The final Simulink program that creates the dataset for the training need more functions, in order to manipulate the data of the simulation and get the aimed matrix. These function blocks recall a MATLAB function, respectively reported in the Appendix B, and are:

- *duty and power manager*: this block removes some unnecessary data, prepare the data in the correct format for the next function blocks and create the desired V_{ref}, that is just a ramp from V_{sc} to V_{oc}

- *Max research*: this block look for the maximum Power along the Power-Voltage curve and save the V_{mpp} and the P_{mpp} in two variables, that will be the output
- *saving matrix*: this function save the instant voltage V corresponding to the V_{ref} of the matrix in Table 10, and store it into the final matrixes. Later stores also the results from the *Max research* function.

A complete illustration of this function inside the Simulation program is in Figure 48.

5.3 ALGORITHM-TESTER PROGRAM

The program illustrated in Figure 50 aims to calculate the performances of the chosen MPPT algorithm. The output of the program is a two-dimensional array, an example is in Table 12, where in every row (corresponding to the same triples of irradiances of Table 10) the following parameters are saved:

- V_{ref}: the steady-state V_{ref}, that is supposed to be last estimation of the V_{mpp} of the MPPT algorithm used
- P_{ref}: it is the power obtained by imposing on the circuit the V_{ref}
- $V_{max,theo}$: it is the V_{mpp} corresponding to the triple of irradiances that is used in that instant
- $P_{max,theo}$: is the P_{mpp} corresponding to the triple of irradiances that is used in that instant.

5.3.1 *The Algorithms Functions*

In Figure 50 it is possible to observe the complete Simulink program, where some function blocks that recall MATLAB functions are implemented. These are:

V_{ref}	P_{ref}	$V_{max,theo}$	$P_{max,theo}$
23.03	121.15	20.76	154.72
34.23	164.76	33.02	175.59
21.61	105.51	21.70	105.52
34.74	64.42	20.76	159.40
25.94	75.15	33.02	93.52

Table 12: Performance matrix example.

- MPPT block: the MPPT algorithm is executed in this block, the input is the only power measured and the output is the V_{ref} and the theoric V_{mpp}
- matrix creator: here the final performances matrix is filled, and the performance parameters are saved inside it.

The single block and respective programs are reported in Appendix B.

CONCLUSIONS

In this thesis four MPPT algorithms have been simulated, and a fifth hybrid algorithm have been implemented using two of the previous. All of them have been simulated, in order to complete the analysis of the performances already started in [18],[17].

The algorithms have been simulated on a portion of a photovoltaic system, where the irradiances were changing frequently. The efficiency of the algorithms have been measured, and the conclusions are the following: the ANN-based algorithms met the expectations of being faster than the other algorithms in finding the MPP. This was expected from the theory of the Neural Networks. The other algorithms simulated showed to be efficient enough only until a certain exploration time. If this time is sufficiently big and allows more than 20 perturbations and explorations, then the Particle Swarm algorithm is more efficient than Neural Network based algorithms. But if the exploration time is short, and the perturbations and explorations are only 3, or only 4, until 20, then the ANN-based algorithms are the most efficient algorithms among the studied ones. This means that these algorithms could be preferred to other MPPT algorithms in applications like solar vehicles, where the irradiances are frequently changing.

The pure ANN and the hybrid ANN-HC algorithms have been tested when the real system is slightly different from the model estimated in the Neural Network: this could be the effect of the temperature change or of the aging of the materials that make up the panels. It has been verified that the hybrid algorithm reacts well to this problem and gets again a high efficiency, while the pure ANN algorithm's efficiency decrease.

The conclusion is that when is possible it is preferable to use the hybrid ANN with HC algorithm instead of the pure ANN.

The idea for this thesis came from two articles, [18] [17]. In these articles only some parameters of the ANN algorithm have been measured, and it is not included the final efficiency of the overall circuit. In this thesis their algorithms have been applied on a little photovoltaic system in a simulation, in order to calculate the efficiency. The results have been compared to the ones of other known[15] algorithms. Now the analysis of this algorithm is more complete, and a manufacturer have more informations to understand if to prefer an ANN-based algorithm to the others. A future activity on this problem could be the implementation of the algorithm on a real application.

APPENDIX

A

DEVICES ELECTRICAL PARAMETERS

In this section the electrical characteristics of the photovoltaic panels used in simulations are listed.

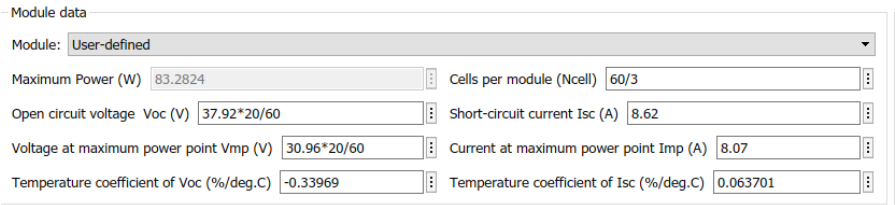


Figure 42: The parameters of the default single photovoltaic module.

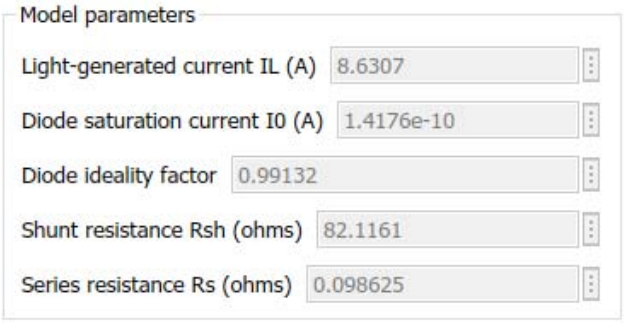


Figure 43: The parameters of the default single photovoltaic module.

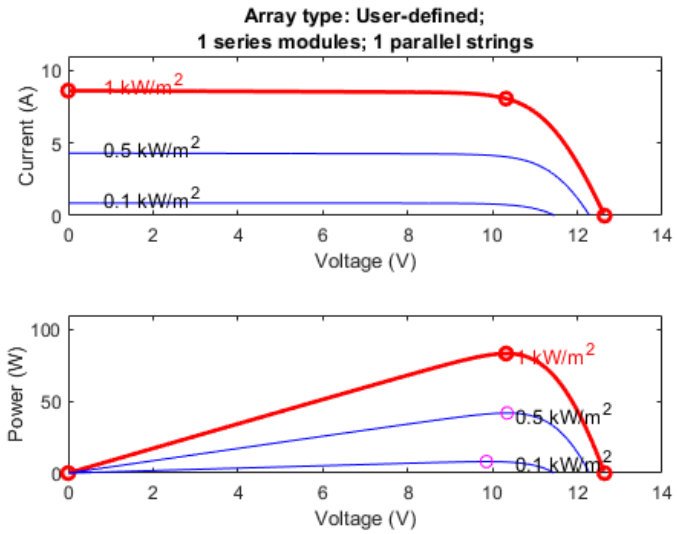


Figure 44: The characteristic power-voltage curves of the default single photovoltaic module.

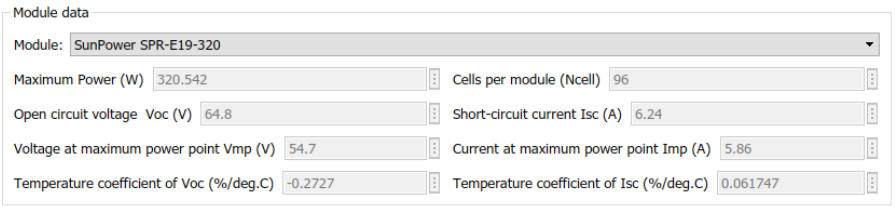


Figure 45: The parameters of the single solar module SPR-E19-320W.

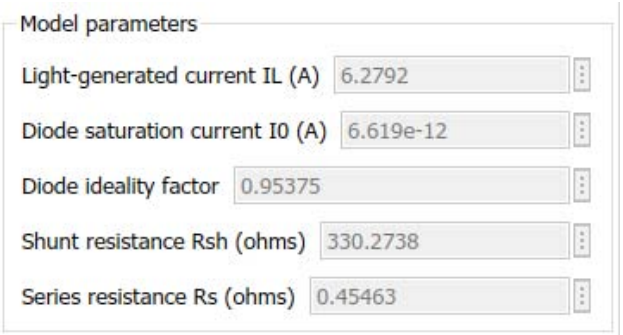


Figure 46: The parameters of the single solar module SPR-E19-320W.

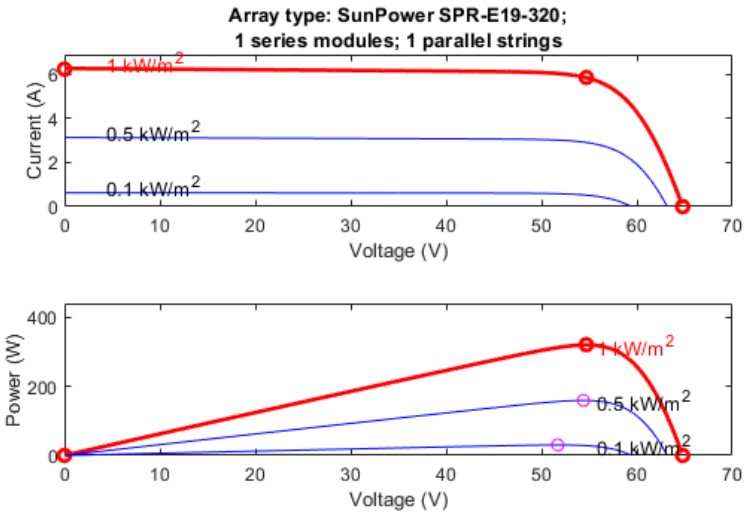


Figure 47: The characteristic power-voltage curves of the single solar module SPR-E19-320W.

B

SIMULINK AND MATLAB

In this section the complete programs used are listed.

B.1 DATASET TRAINING

B.1.1 *Duty and Power Manager*

```
function [duty, Pout, Vout, Iout] = fcn(t, contatore1,
   deltaT, NSAMPLE, P, V, I)

t1=0.05*deltaT*NSAMPLE;
t2=0.95*deltaT*NSAMPLE;

T=deltaT*NSAMPLE*(contatore1-1);
pendenzaRampa= 1/(deltaT*NSAMPLE);

 if t >= T+t1 && t < T+t2
    duty = -pendenzaRampa*(t-deltaT*NSAMPLE*contatore1);
    Pout=P;
    Vout=V;
    Iout=I;
 elseif t < t1
     p1= -pendenzaRampa*(T+t1-deltaT*NSAMPLE*contatore1);
     X=spline_4condition(t1, 0, 0, p1, 0, -pendenzaRampa);
     t0=0;
     duty =X(1) + X(2)*(t-t0) + X(3)*(t-t0)^2 + X(4)*(t-t0)
        ^3;
     Pout=0;
     Vout=0;
     Iout=0;
 elseif t>= t2+T && t < T+deltaT*NSAMPLE
```

```
    X=spline_4condition( T+deltaT*NSAMPLE,t2+T , -
        pendenzaRampa*(T+t2-deltaT*NSAMPLE*contatore1),
        0.5, -pendenzaRampa, 3);
    t0=t2+T;
    duty = X(1) + X(2)*(t-t0) + X(3)*(t-t0)^2 + X(4)*(t-t0)
        ^3;
    Pout=0;
    Vout=0;
   Iout=0;
elseif t >= T && t <= t1+T
    X=spline_4condition( t1+T, T , 0.5, -pendenzaRampa*(T+
        t1-deltaT*NSAMPLE*contatore1) , 3, -pendenzaRampa )
        ;
    t0=T;
    duty = X(1) + X(2)*(t-t0) + X(3)*(t-t0)^2 + X(4)*(t-t0)
        ^3;
    Pout=0;
    Vout=0;
    Iout=0;
else
    duty = -1;
    Pout=-1;
    Vout=-1;
    Iout=-1;
end
```

B.1.2 *Max Research*

```
function [Pmax, Vmax] = fcn (Pin, Vin,  conta1, duty)

persistent massimo;
if isempty(massimo)
    massimo=0;
end
persistent VV;
if isempty(VV)
```

```
    VV=0;
end

persistent lastconta1;
if isempty(lastconta1)
    lastconta1=1;
end

if conta1 ~= lastconta1
    lastconta1=conta1;
    massimo=0;

    VV=0;
end
if Pin > massimo
            massimo=Pin;

            VV=Vin;

end
Pmax=massimo;
Vmax=VV;
end
```

B.1.3 *Saving Matrix*

```
function [y2, Py, Vy, Iy , stop, ii, jj, dutysj]   = fcn(P,V
    , I, Pmax, duty, Vmax, NSAMPLE, NPROVE, confdataset, c1,
     soglia)

persistent i;
if isempty(i)
    i=1;
end

persistent j;
```

```
if isempty(j)
    j=1;
end
ii=i;
jj=j;
persistent datasetP;
if isempty(datasetP)
    datasetP=confdataset;
end
persistent datasetV;
if isempty(datasetV)
    datasetV=confdataset;
end
persistent datasetI;
if isempty(datasetI)
    datasetI=confdataset;
end
persistent flag;
persistent flag2;
if isempty(flag)
    flag = 0;
    flag2=0;
end
Py=datasetP;
Vy=datasetV;
Iy=datasetI;
y2=[0, Pmax];
stop=0;
dutys=zeros( NSAMPLE, 1);
for cont = 1:1:NSAMPLE
    dutys(cont)=1-(cont*(0.95-0.05)/(NSAMPLE)+0.04);
end
dutysj=dutys(j);
if c1==NPROVE+1
    stop =1;
    Py=datasetP;
    Vy=datasetV;
```

```
    Iy=datasetI;
    return;
end

if duty <= dutys(j) && Pmax > 0  && flag2 == 0
    y2=[P, Pmax];
    if i<=NPROVE

        if P > (0.001)
            datasetP(i, j)=P;
            datasetV(i, j)=V;
            datasetI(i, j)=I;
        else
            datasetP(i, j)=0;
            datasetV(i, j)=V;
            datasetI(i, j)=0;
        end

        Py=datasetP;
        Vy=datasetV;
        Iy=datasetI;
        j=j+1;
        if(j > NSAMPLE)
            j=1;
            i=i+1;
            flag =1;
            flag2 =1;
        end
    else
        Py=datasetP;
        Vy=datasetV;
        Iy=datasetI;
        stop=1;
        y2=[0, Pmax];

    end
end
```

```
if   i <= NPROVE+1 && j == 1 && i >1 && P > 0.001 && flag
    ==1
    datasetP(i-1, NSAMPLE+1)=Pmax;
    datasetV(i-1, NSAMPLE+1)=Vmax;
    datasetI(i-1, NSAMPLE+1)=Vmax;
    Py=datasetP;
    Vy=datasetV;
    Iy=datasetI;
end

if P  < 0.001
    flag =0;
elseif j > 1
    flag =1;
end
if Pmax< 0.01
    flag2=0;
end
ii=i;
jj=j;
dutysj=dutys(j);
end
```

B.2 ALGORITHM-TESTER PROGRAM

B.2.1 *ANN*

The *duty regulator and power manager* function block of Figure 51:

```
function [Vref, ctrlSelect, X, prova, VmaxP] = fcn( i,
    contatore2, deltaT, NSAMPLE, P, V_DC_load, Pmeas_init ,
    VmaxPower, simoutV)

persistent Pmeas;
if isempty(Pmeas)
    Pmeas=Pmeas_init;
```

```
end

prova=0;
Vref=0;
X=Pmeas;
VmaxP=VmaxPower(i);

voltages=zeros( NSAMPLE, 1);
for cont = 1:1:NSAMPLE
       voltages(cont)=simoutV(contatore2, cont);
end
if contatore2 <= NSAMPLE
    Vref=voltages(contatore2);
    if P>1e-3
        Pmeas(contatore2)=P;
    else
        Pmeas(contatore2)=0;
    end
    prova=0;
    ctrlSelect=1;
    X= Pmeas_init;

elseif contatore2 >=NSAMPLE+1
    X=Pmeas;
    ctrlSelect=-1;

else
    X=Pmeas;
    Vref=0.5;
    ctrlSelect=1;
    prova=-1;
end
```

B.2.2 *HC*

The code used for the HC-modified algorithm:

```
function [Vref, PEO0ENDPEO1, VN,  Pmppo] = fcn( V, I,
    counter)

persistent flag1;
persistent flag2;
persistent flag3;
persistent flag4;
persistent flag5;
persistent flag6;
persistent Vn;
persistent Vrefp;
persistent Vmpp;
persistent Pmpp;
persistent cprec;

if isempty(flag1)
    flag1 = 0;
    flag6=0;
    cprec=0;
    flag3 =0;
    flag4=0;
    flag5=0;
    Vn =0;
    flag2=0;
    Pmpp = 0;
    Vmpp=0;
    Vrefp =0;
end

if counter ~= cprec
    flag1 = 0;
    flag2 = 0;
    flag3=0;
    flag4=0;
    flag5=0;
    flag6=0;
    cprec=counter;
```

```
    Vref=Vrefp;
    PEO0ENDPEO1=-11;
    Pmppo =Pmpp;
    VN=Vn;
    Vn = 2.53;
    return;
end

dmin = 37/3;
Pn=0;
factor = 0.51;
if flag1 ==0
    Vref = 0.7*37;
    Vn = 2.35;
    PEO0ENDPEO1 =-3;
    flag1 =1;
    Pmpp =0;
    Pmppo =Pmpp;
    VN=Vn;
    return;
elseif flag2 ==0
    [Vref, flag2] = pEoSubroutine(V, I, 1, 0.1);
    PEO0ENDPEO1 =flag2-0.5;
    if flag2 ==1
        Vmpp = V;
        Pmpp = V*I;
    end
    Pmppo =Pmpp;
    VN=Vn;
    return;
elseif  Vn > Vmpp -dmin
    if flag6 ==0
        Vn=Vn+dmin;
        flag6 =1;
    end

    if Vn > 37
```

```
        if flag5 ==0
            Vref = Vmpp;
            PEO0ENDPEO1=4;
            Pmppo =Pmpp;
            VN=Vn;
            flag5=1;
            return;
        end
        local=0;
        [Vref, local] = pEoSubroutine(V, I, 1, 0.1);
        PEO0ENDPEO1=local+0.25;
        Pmppo =Pmpp;
        VN=Vn;
        return;
    else
        if flag3 ==0
            Vref = Vn;
            Vrefp=Vn;
            PEO0ENDPEO1 = 2;
            flag3 =1;
            Pmppo =Pmpp;
            VN=Vn;
            return;
        end
        Pn = V*I;
        if Pn > Pmpp && flag4 ==0
            [Vrefp, flag4] = pEoSubroutine(V, I, 1, 0.1);
            Vref=Vrefp;
            PEO0ENDPEO1 = flag4;
            if flag4 == 1
                Vmpp = V;
                Pmpp = V*I;
            end
            Pmppo =Pmpp;
            VN=Vn;

            return;
```

```
        else
            Vref = Vrefp;
            Vn = Pmpp/(I*factor);
            PEO0ENDPEO1 = 5;
            flag3=0;
            flag4 =0;
            Pmppo =Pmpp;
            VN=Vn;
            return;
        end
    end
else
    if flag3 ==0
        Vref = Vn;
        Vrefp=Vn;
        PEO0ENDPEO1 = -2;
        flag3 =1;
        Pmppo =Pmpp;
        VN=Vn;
        return;
    end
    Pn = V*I;
    if Pn > Pmpp && flag4 ==0
        [Vrefp, flag4] = pEoSubroutine(V, I, 1, 0.1);
        PEO0ENDPEO1 = flag4+0.5;
        Vref=Vrefp;
        if flag4 ==1
            Vmpp = V;
            Pmpp = V*I;
        end
        Pmppo =Pmpp;
        VN=Vn;
        return;
    else
        Vref = Vrefp;
        Vn = Pmpp/(I);
```

```
        PEO0ENDPEO1 = -5;
        flag3=0;
        flag4 =0;
        Pmppo =Pmpp;
        VN=Vn;
        return;
    end

end
end
```

B.2.3 *PSO*

It is reported here the code for the PSO-algorithm:

```
function [Vref, ctrlSelect, X, prova, VmaxP] = fcn( i,
    contatore2, noP, nVar, V, I, maxIter, Pmeas_init ,
    VmaxPower, simoutV, clock, swarmconfig)

VmaxP=VmaxPower(i);
persistent flag;
persistent flag1;
persistent flag2;
persistent flagFinal;
persistent k;
persistent t;
persistent Vrefp;
persistent Swarm;
persistent average_objective; %= zeros(1, maxIter);
persistent cgCurve;
persistent FirstP_D1;
persistent position_history;
persistent c1prec;
persistent lastTen;
persistent lasti;

dataVis=0;
```

```
ub =37;
lb = 2.53;
wMax = 0.9;
wMin = 0.2;
c1 =2;
c2 = 2;
vMax =(ub - lb) .* 0.5; % (ub - lb) .* 0.2;
vMin  = -vMax;

if isempty(flag)
    flag = 1;
    flag1 = 1;
    flag2 = 0;
    lasti=0;
    k =1;
    t=1;
    Vrefp=0;
    average_objective = zeros(1, maxIter);
    cgCurve = zeros(1, maxIter);
    FirstP_D1 = zeros(1 , maxIter);
    position_history = zeros(noP , maxIter , nVar );
    Swarm=swarmconfig;
    c1prec =i;
    flagFinal=1;
    lastTen = zeros(10, 1);

end
if  i ~= c1prec
    flag = 1;
    flag1 = 1;
    flag2 = 0;
    lasti=0;
    k =1;
    t=1;
    Vrefp=0;
    average_objective = zeros(1, maxIter);
    cgCurve = zeros(1, maxIter);
```

```
    FirstP_D1 = zeros(1 , maxIter);
    position_history = zeros(noP , maxIter , nVar );
    Swarm=swarmconfig;
    c1prec =i;
    flagFinal=1;
    lastTen = zeros(10, 1);
end

if clock ==1 && flag == 1 && flagFinal == 1

    if flag1 == 1
        flag1=0;
        ctrlSelect=1;
        X=t;

        prova = k;
        Vrefp=Swarm.Particles(k).X;
        Vref =Vrefp;
        return;
    end

    prova = k;
    flag1=1;
    ctrlSelect=2;

    currentX = Swarm.Particles(k).X;
    position_history(k , t , : ) = currentX;
    power = -V*I;

    average_objective(t) =  average_objective(t)  + power;

    if power < Swarm.Particles(k).PBEST.O
        Swarm.Particles(k).PBEST.X = currentX;
        Swarm.Particles(k).PBEST.O = power;
    end

    if power < Swarm.GBEST.O
```

```
    Swarm.GBEST.X = currentX;
    Swarm.GBEST.O = power;
end

Vrefp =Swarm.GBEST.X;
Vref =Vrefp;
X=t;
flag =0;
k=k+1;
if k > noP
    k=1;
    flag2=1;
end

if flag2 ==1
    ctrlSelect=3;
    prova = k;
    w = wMax - t .* ((wMax - wMin) / maxIter);

    FirstP_D1(t) = Swarm.Particles(1).X;
    for j = 1 : noP

        Swarm.Particles(j).V = w * Swarm.Particles(j).V
            ;
        B= + c1 * rand(1,1) * (Swarm.Particles(j).PBEST.
            X - Swarm.Particles(j).X) + c2 * rand(1,1) *
             (Swarm.GBEST.X - Swarm.Particles(j).X);
        Swarm.Particles(j).V=Swarm.Particles(j).V+B;

        index1 = find(Swarm.Particles(j).V > vMax);
        index2 = find(Swarm.Particles(j).V < vMin);

        Swarm.Particles(j).V(index1) = vMax(index1);
        Swarm.Particles(j).V(index2) = vMin(index2);

        Swarm.Particles(j).X = Swarm.Particles(j).X +
            Swarm.Particles(j).V;
```

```
        index1 = find(Swarm.Particles(j).X > ub);
        index2 = find(Swarm.Particles(j).X < lb);

        Swarm.Particles(j).X(index1) = ub(index1);
        Swarm.Particles(j).X(index2) = lb(index2);
    end

    if dataVis == 1
        outmsg = ['Iteration# ', num2str(t) , ' Swarm.
            GBEST.O = ' , num2str(Swarm.GBEST.O)];
        disp(outmsg);
    end

    cgCurve(t) = Swarm.GBEST.O;
    average_objective(t) = average_objective(t) / noP;

    t = t+1;
    flag2 = 0;
    X=t;
    Vrefp =Swarm.GBEST.X;
    Vref =Vrefp;

    lasti=lasti+1;
    if lasti>10
        somma=0;
        for i=1:1:10
            somma=somma+lastTen(i);
        end
        media=somma/10;
        if abs(media - lastTen(10))<0.1
            flagFinal =0;
            t = maxIter;
            Vrefp =Swarm.GBEST.X;
            Vref = Vrefp;
            X=t;
```

```
            end
            lasti=1;
        end
        lastTen(lasti)=Swarm.GBEST.O;
        if t > maxIter
            flagFinal =0;
            t = maxIter;
            Vrefp =Swarm.GBEST.X;
            Vref = Vrefp;
            X=t;
        end
    end

elseif clock ==0
    flag = 1;
    prova = k;
    ctrlSelect=0;
    X=t;
    Vref=Vrefp;
else
    provaan example = k-0.1;
    ctrlSelect=0;
    X=t;
    Vref=Vrefp;
end
end
```

B.2.4 *The simulink blocks*

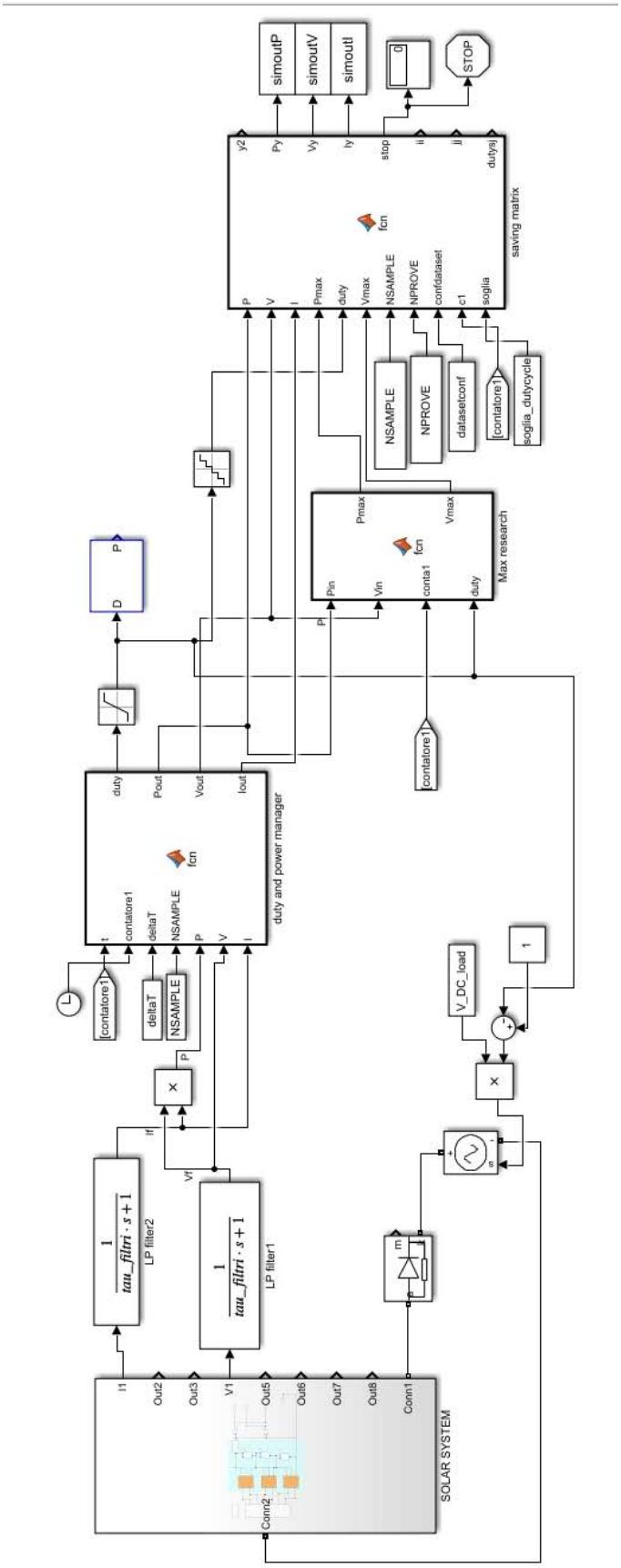


Figure 48: The complete Simulink program for the creation of the training dataset matrix. The Solar System block include the Simulink model in Figure 40.

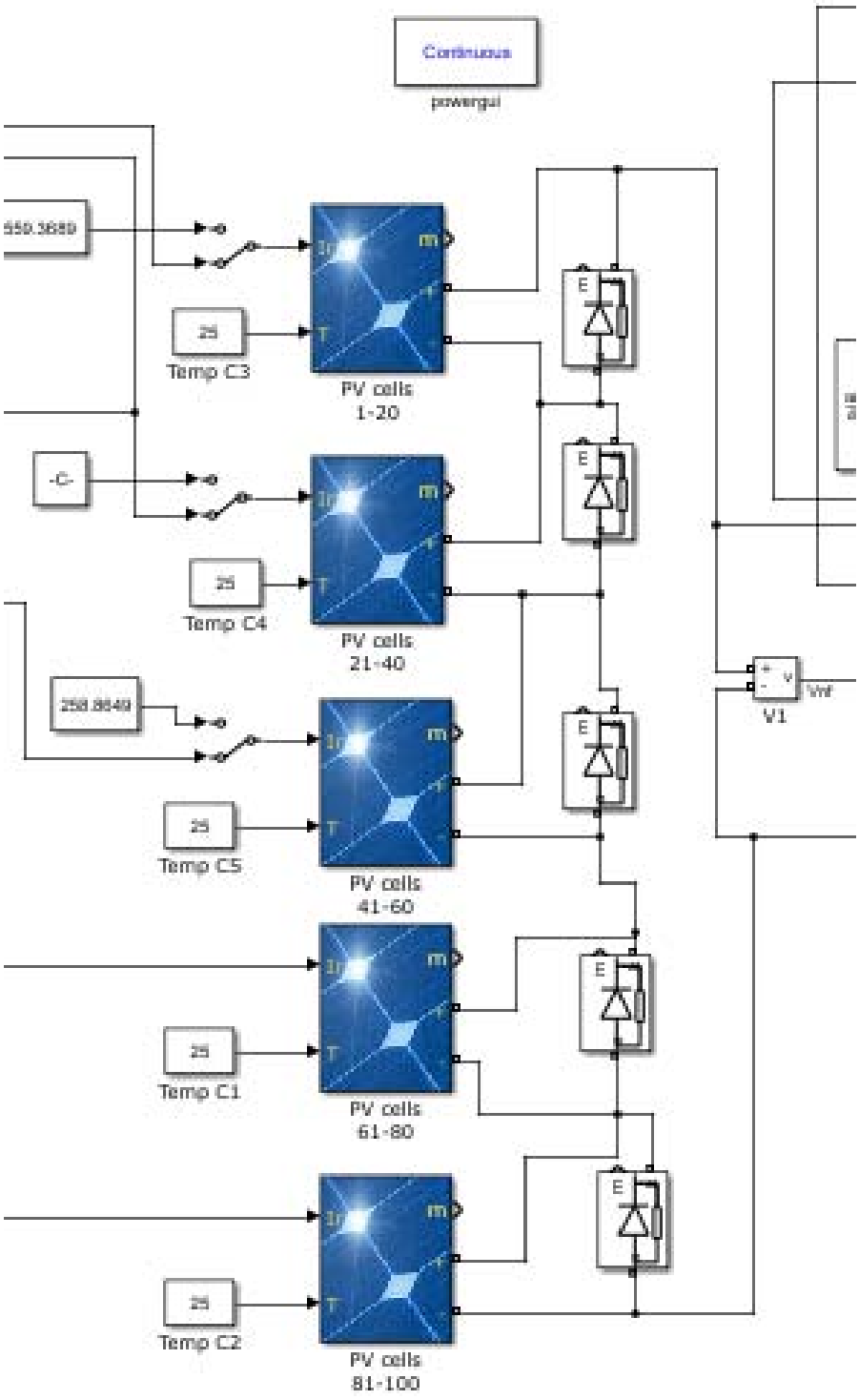


Figure 49: The new system tested, with five Photovoltaics modules SPR-E19-320W.

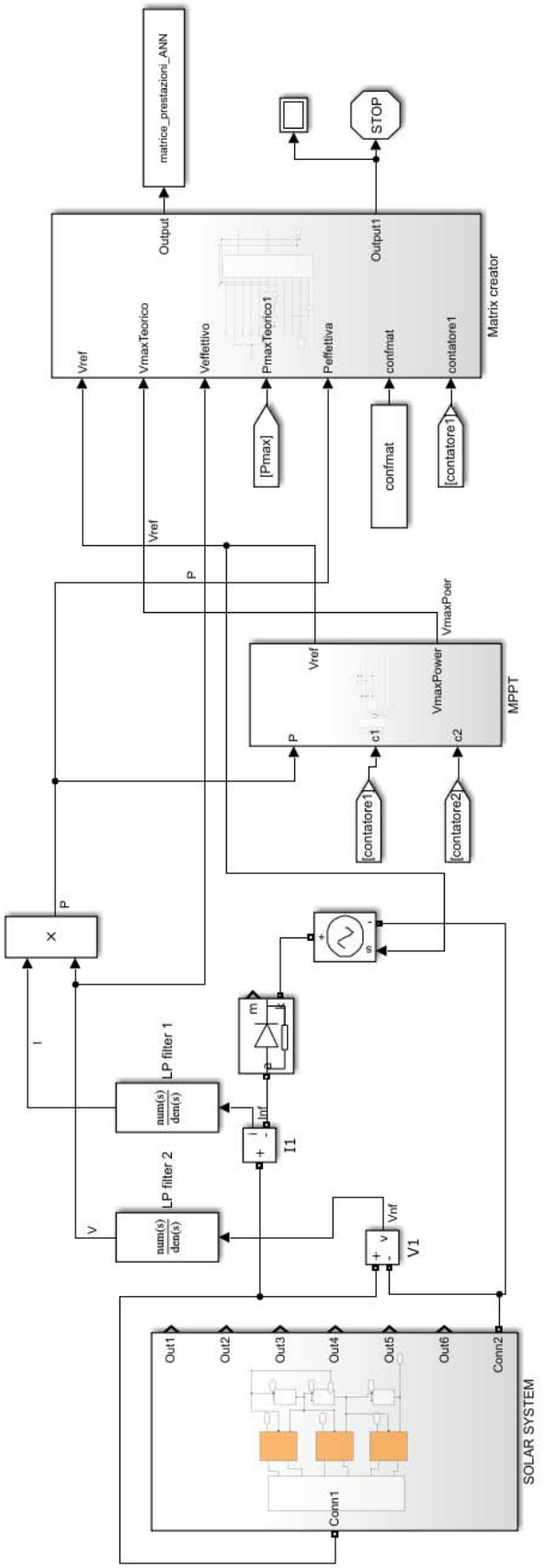


Figure 50: The complete Simulink program for the testing of the MPPT algorithm. The Solar System block include the Simulink model in Figure 40.

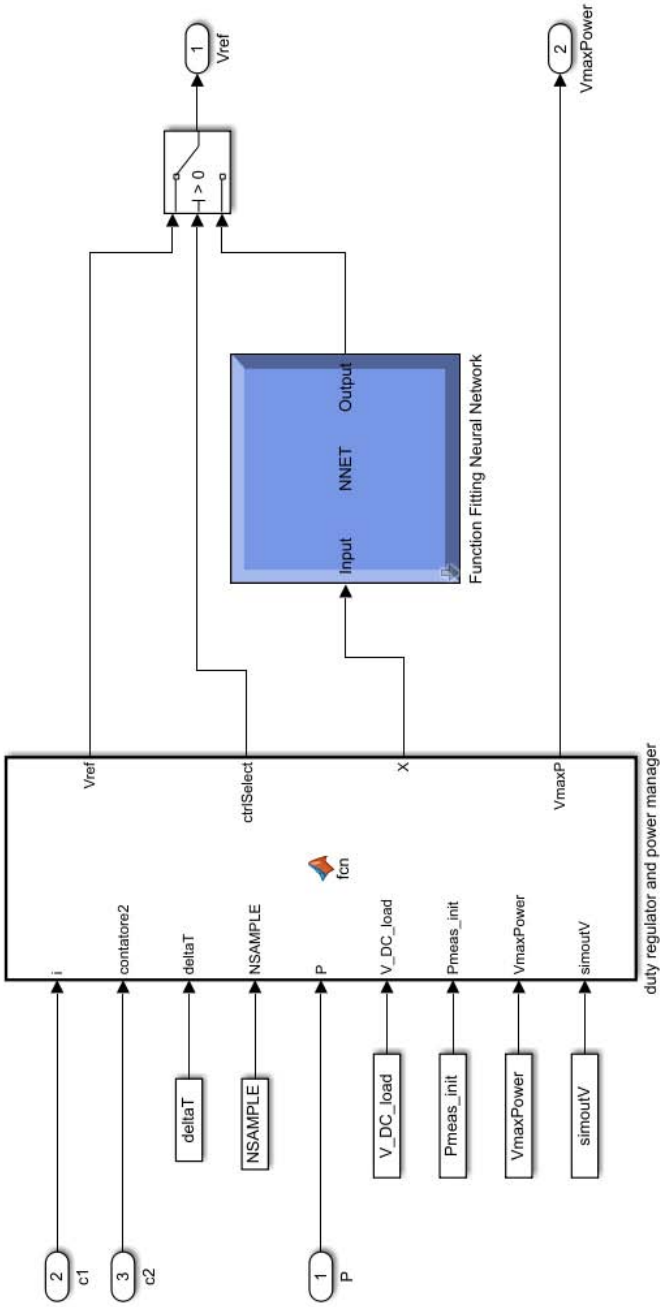


Figure 51: The MPPT block for the test of the ANN

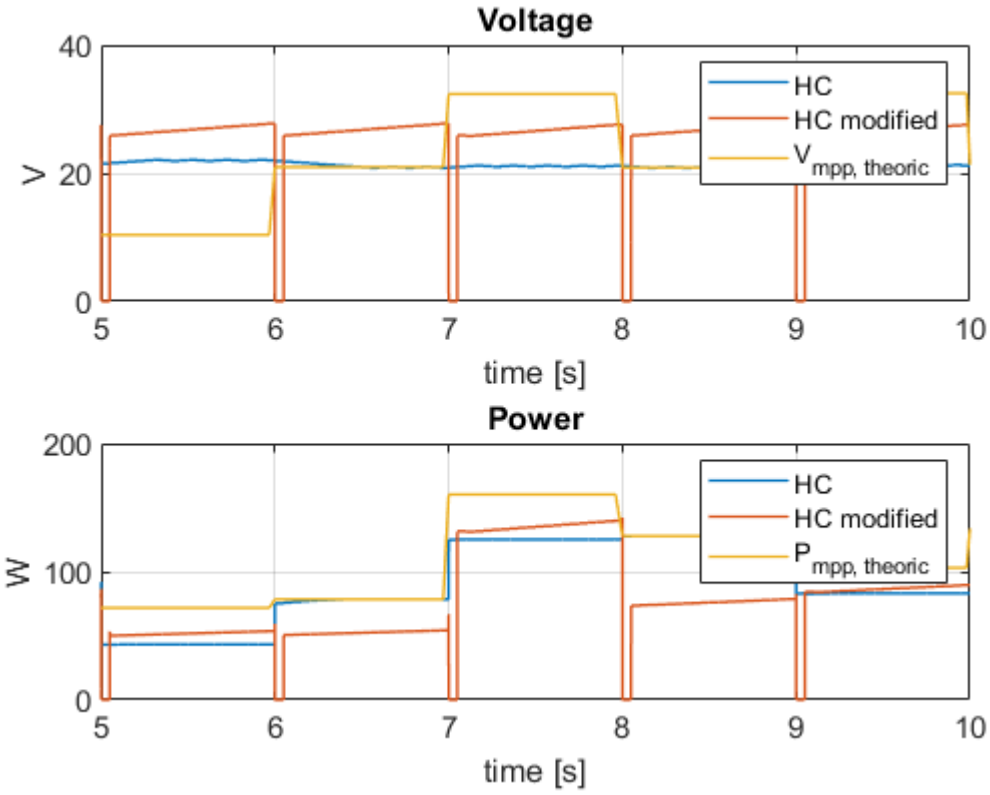


Figure 52: An example of the operation of the HC algorithms. The exploration time is 1 s. One Perturbation and Observation takes 0.1 s, then the Perturbations and Observations are 10. The average efficiency of the HC is 0.903, of the HC modified 0.782.

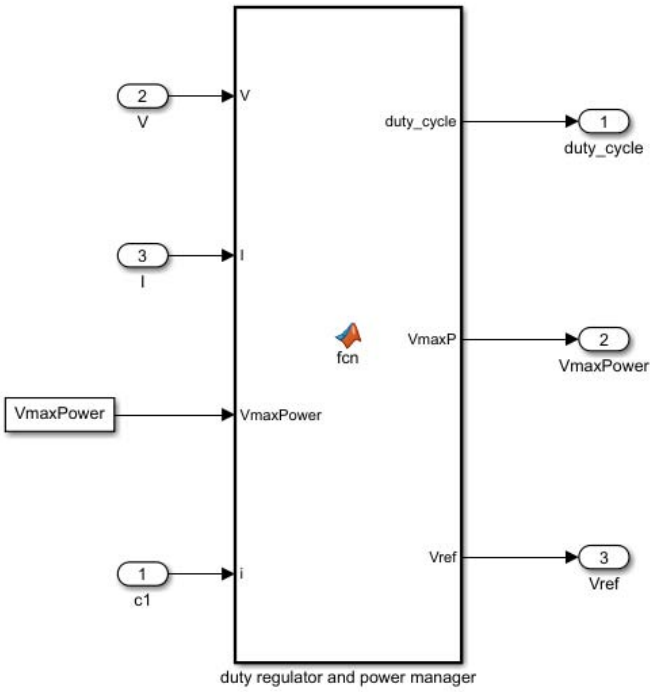


Figure 53: The MPPT block for the test of the HC and HC-modified

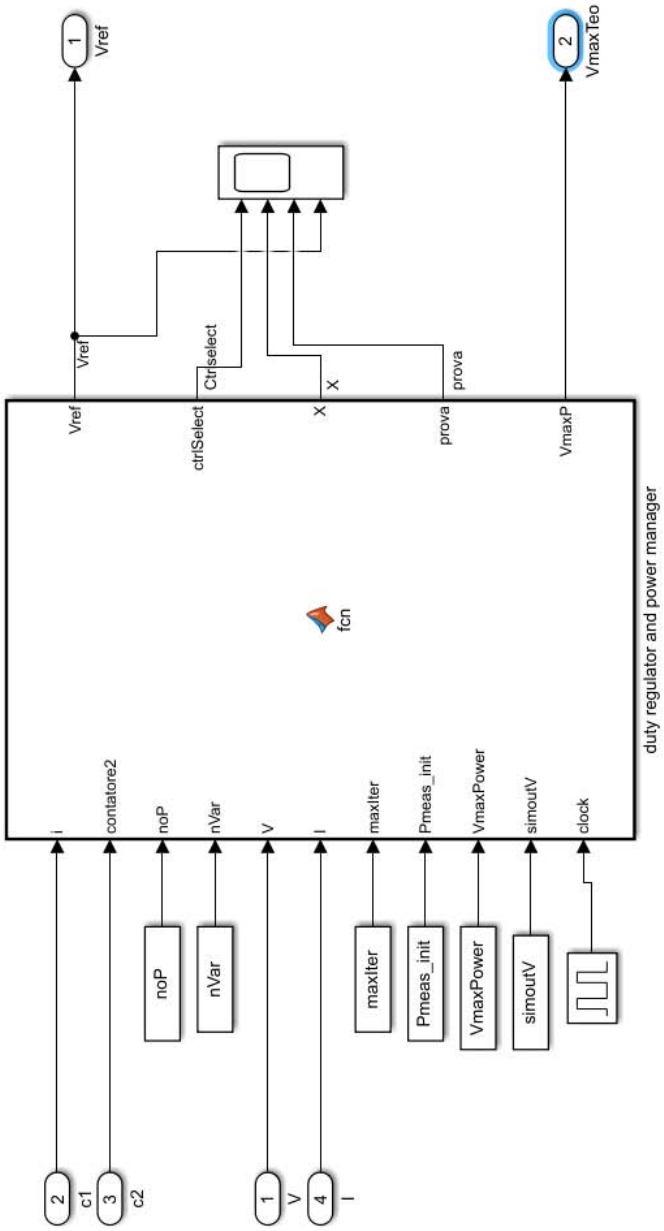


Figure 54: The MPPT block for the test of the PSO

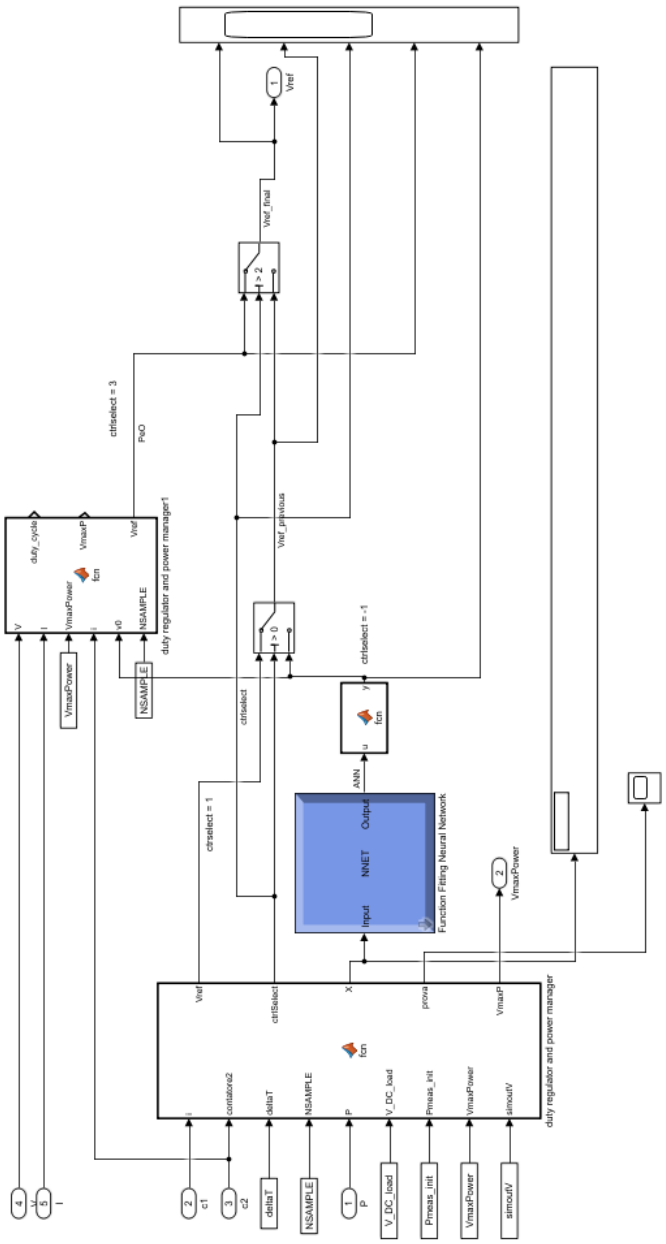


Figure 55: The MPPT block for the test of the Hybrid ANN plus HC algorithm

BIBLIOGRAPHY

[1] A. Omeiri A. Amiar A. Charki O. Riou F. Delaleux J.F. Durastanti A. Azizi, P.O. Logerais. Impact of the aging of a photovoltaic module on the performance of a grid-connected system. *Solar Energy*, (174):455–454, 2018.

[2] Khaled E. Addoweesh Abdullah M. Noman and Hussein M. Mashaly. A fuzzy logic control method for mppt of pv systems. *Sustainable Energy Technologies Center King Saud University*, October 2012.

[3] I.William Christopher and Dr.R.Ramesh. Comparative study of po and inc mppt algorithms. *American Journal of Engineering Research (AJER)*, 2(12):pp. 402–408, July 2013.

[4] Rana Noman Shafqat Ehtisham Lodhi and Kerrouche K. D. E. Application of particle swarm optimization for extracting global maximum power point in pv system under partial shadow conditions. *International Journal of Electronics and Electrical Engineering*, 5(3), 2017.

[5] Cherif Larbes Faiza Belhachat. Renewable and sustainable energy reviews. *Journal of Solar Energy Research (JSER)*, 92:513–553, 2018. URL https://doi.org/10.1016/j.rser.2018.04.094. Elsevier Ltd.

[6] Basil Qadoor Mohammed Al Shamsi Osman Abdulsalam Zahidur Rahman Hussain Attia, Mousa Mohsen. New design and implementation of a solar car of the american university of ras al khaimah: Electrical vision. *Journal of Sustainable Development of Energy, Water and Environment Systems*.

[7] Petros Ioannou and Baris Fidan. *Adaptive Control Tutorial*. SIAM, 2006. URL https://doi.org/10.1137/1.9780898718652. chapter 8.

[8] H.White K. Hornik, M. Stinchcombe. Multilayer feedforward networks are universal approximators. Neural Networks(2):359–366.

[9] D. Kriesel. *A brief introduction to Neural Networks*. 2005. URL http://www.dkriesel.com/en/science/neural_networks.

[10] Paul A. Lynn. *Electricity from Sunlight: An Introduction to Photovoltaics*. A John Wiley Sons, Ltd., Publication, 2010.

[11] M. Ermis M. Bodur. Maximum power point tracking for low power photovoltaic solar panels. *in Proc. 7th Mediterranean Electrotechnical Conference*, 2:758.

[12] Inc. MathWord. Improve shallow neural network generalization and avoid overfitting. *Matlab Site*.

[13] Seyedali Mirjalili. A simple implementation of particle swarm optimization (pso) algorithm. *MATLAB Central File Exchange*, 2020.

[14] Salman Hameed M. Saad Bin Arif Abhinandan Jain Mohammed Aslam Husain, Abu Tariq. Comparative assessment of maximum power point tracking procedures for photovoltaic systems. *KeAi Publishing*, Green Energy environment(2).

[15] D. Solyali P. Mazaheri Salehi. A review on maximum power point tracker methods and their applications. *Journal of Solar Energy Research (JSER)*, 3(2):123–133, 2018.

[16] V.Martinez Roman E. Sanz A. R. Alonso, P.Ibanez. An innovative perturb, observe and check algorithm for partially shaded pv systems. *IEEE Xplore*, 2009.

[17] G. Scelba A. Sciacca S.A. Rizzo, N. Salerno. Enhanced hybrid global mppt algorithm for pv systems operating under fast-changing partial shading conditions. *INTERNATIONAL JOURNAL of RENEWABLE ENERGY RESEARCH*, 8(1), 2018. University of Catania.

[18] Giacomo Scelba Santi Agatino Rizzo. Ann based mppt method for rapidly variable shading condition. *ResearchGate*, May 2015. University of Catania.

[19] Prof. Seppo Ovaska Student David Sanz Morales. *Maximum Power Point Tracking Algorithms for Photovoltaic Applications*. Aalto University School of Science and Technology, 2010.

[20] P. A. Vikhar. *Evolutionary algorithms: A critical review and its future prospects*. Number pp. 261–265. Jalgaon, 2016. Proceedings of the 2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC).

[21] Wikipedia. Neural network. *Wikipedia, the free enyclopedia*, 2019.

[22] L.A. Zadeh. *Soft Computing and Fuzzy Logic*, volume 11. University of Washington Libraries, 1994.