REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Ecole Nationale Polytechnique
Département Electronique
Centre de Développement des Energies Renouvelables

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

End of studies project
To obtain the diploma of State Engineer in Electronics

Theme:

Modeling and analysis of the state of charge of batteries for photovoltaic use

Authors

MZIR Mahdi BELABED Youcef

• Mr. Adnane Mourad	ENP	President
• Ms. Deglai Aicha	CDER	Supervisor
• Mr. Haddadi Mourad	ENP	Co-supervisor
• M . Cherif LARBES	ENP	Examiner

ENP 2021

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Ecole Nationale Polytechnique
Département Electronique
Centre de Développement des Energies Renouvelables

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

End of studies project
To obtain the diploma of State Engineer in Electronics

Theme:

Modeling and analysis of the state of charge of batteries for photovoltaic use

Authors

MZIR Mahdi BELABED Youcef

• Mr. Adnane Mourad	ENP	President
• Ms. Deglai Aicha	CDER	Supervisor
• Mr. Haddadi Mourad	ENP	Co-supervisor
• M . Cherif LARBES	ENP	Examiner

ENP 2021

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Ecole Nationale Polytechnique
Département Electronique
Centre de Développement des Energies Renouvelables

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

Projet de fin d'Etudes
Pour l'obtention du diplôme d'Ingénieur d'Etat en Electronique

---

Thème:

Modélisation et analyse de l'état de charge de batteries à usage photovoltaïque

Auteurs

MZIR Mahdi BELABED Youcef

• Mr. Adnane Mourad	ENP	Président
• Mme. Deglai Aicha	CDER	Promoteur
• Mr. Haddadi Mourad	ENP	Co-promoteur
• M . Cherif LARBES	ENP	Examinateur

ENP 2021

ملخص

تتمحور هذه الأطروحة حول تقدير حالة الشحن في بطاريات تخزين الطاقة الشمسية. نقدم النتائج التي توصلنا إليها بعد تكييف نماذج متعددة لتقدير الشحنة مع حالتنا ثم تحليل النتائج لكل منها و مقارنتها لاستنتاج النموذج الأفضل. نقوم بذلك من لتقنيتين مختلفتين للبطاريات تستخدمان في إنتاج الطاقة الكهروضوئية وتخزينها: بطاريات أيون الليثيوم وبطاريات الرصاص الحمضية. نقوم بتطوير دائرة كهربائية مكافئة لبطارية أيون الليثيوم الخاصة بنا والتي نضيف إليها مرشح الحالة. نحن نطور ونطبق نماذج مختلفة للتعلم الآلي لتقدير حالة الشحن. نحن نتحقق من صحة النتائج التي توصلنا إليها من خلال دورتين مستقلتين.

الكلمات المفتاحية: حالة الشحن ، البطارية ، الليثيوم ، حمض الرصاص ، الطاقة الشمسية ، التقدير.

RÉSUMÉ

Cette thèse porte sur l'estimation de l'état de charge des batteries de stockage d'énergie solaire. Nous présentons nos résultats après avoir adapté plusieurs modèles d'estimation de l'état de charge à notre problème, en analysant les résultats pour chacun et en les comparant pour en déduire le meilleur modèle. Nous le faisons pour deux technologies de batteries différentes qui sont utilisées dans la production et le stockage d'énergie photovoltaïque : les batteries lithium-ion et plomb-acide. Nous développons un circuit équivalent pour notre batterie Lithium-ion auquel nous ajoutons un filtre d'état. Nous développons et appliquons différents modèles d'apprentissage automatique pour estimer l'état de charge. Nous validons nos résultats avec deux cycles autonomes.

Mots Clé : Etat de charge, Batterie, Lithium, Plomb-acide, Energie Solaire, Estimation.

ABSTRACT

This thesis revolves around state of charge estimation in solar energy storage batteries. We present our findings after adapting multiple state of charge estimation models to our problem, analyzing the results for each and comparing them to deduce the better model. We do this for two different battery technologies which are used in photovoltaic energy production and storage: the lithium-ion and lead-acid batteries. We develop an equivalent circuit for our Lithium-ion battery to which we add a state filter. We develop and apply different machine learning models to estimate state of charge. We validate our findings with two standalone cycles.

Key words: State of charge, Battery, Lithium,Lead-acid,Solar Energy,Estimation.

"Je dédie ce travail à ceux dont les mots d'encouragement, soutient et inquiétude sonnent des plus profondes places de ma mémoire: mes chers parents Rezki et Soraya. Aucun succès ne pourrais prendre forme sans eux, aucun accmplissement ne pourra rembourser leur grâce."

∼ Mahdi

Je dédie ce travail à mon père Belabed Nacereddine qui m'a soutenu jusqu'à la toute fin.

اللهم أغفر له و ارحمه و أفسح له في قبره و أدخله في جنة النعيم

-Belabed Youcef

Acknowledgments

On the very outset of this thesis, we give our humblest thanks and praises to Allah almighty for his showers of blessings and benediction throughout our years in school.

We would also like to communicate our truest love, gratitude, and appreciation for our respective Parents, brothers, sisters, and families as a whole for the colossal effort and immense dedication and trust they put into getting us where we are. Our happiness, successes, achievements and especially this work would be nowhere within the realm of possibility without them.

On a similar note, a debt of gratitude is owed to our supervisor Ms. Aicha Degla, Doctorate Researcher at the Renewable Energy Development Center CDER, for her continued support, her ceaseless encouragement and her unwavering will to assist us with her intellectual advice in every step of this work. We also forward our thanks to our co-supervisor Mr. Mourad Haddadi for his contributions.

We would like to thank the members of the Jury for the attention they gave to our work and the interesting proposals they supplied us with.

We are grateful to every teacher who contributed in paving the way for our Studying Journey.

Last but not least, we present our sincerest thanks to our friends for their priceless company and serious encouragements.

Table of Contents

List of Tables

List of Figures

NOMENCLATURE

δX	Differential of the variable X
$\frac{\partial}{\partial x}$	Partial derivative with respect to x
$\frac{d}{dx}$	Derivative with respect ro x
$\hat{x}$	Éstimate for the variable x
$\hat{x}^-$	Á posteriori estimate
$\int_a^b$	Definite integral
$\langle .,. \rangle$	Dot product
$\mathscr{R}^n$	n-dimensional real vector space
ρ (.,.)	Distance measure
$\sum$	Discrete sum
$\widetilde{x}$	Approximate value for x
A^T	Transpose of A
A^{-1}	Inverse of A
$X \sim N(m, \sigma^2)$	X is a normal random variable with 'm' mean and 'σ^2' variance
E [.]	Expected value
I	Current
T	Temperature
V	Voltage

Acronyms

ANN Aritificial Neural Networks.

BMS Battery Management System.

BNN Backpropagation Neural Networks.

CC Constant Current.

CDER Renewable Energy Development Center.

CV Constant Voltage.

DC Direct Current.

ECM Equivalent Circuit Model.

EKF Extended Kalman Filter.

GPU Graphical Processing Unit.

KF Kalman Filter.

KNN K-Nearest Neighbors.

MBE Mean Bias Error.

MCC Modified Coulomb Counting.

NN Neural Networks.

OCV Open Circuit Voltage.

ODE Ordinary Differential Equation.

PV Photovoltaic.

RBF Radial Basis Function.

RE Relative Error.

ref Reference.

RMSE Root Mean Square Error.

SOC State Of Charge.

SV Support Vectors.

SVM Support Vector Machines.

SVR Support Vector Regressor.

UKF Unscented Kalman Filter.

UPC Uninterruptible Power System.

ZIR Zero Input Response.

ZSR Zero State Response.

CHAPTER 1

INTRODUCTION

1.1 Context and motivations

The sun is humongous ball of hydrogen and helium that's about 152.04 million km away from earth. Its surface is constantly exploding, burning, reforming and exploding again. And most importantly, emitting energy to the far reaches of the solar system, that which we call "sunlight". Only a minuscule fraction of it reaches the surface of the earth, which amount to 430 quintillion Joules/hour. That's more than 410 quintillion consumed yearly by humans on earth. An endless, clean and renewable source of energy is being transmitted to earth since the dawn of time, and finally we are reaching a point we can seize it. [25]

To seize any form of energy means converting it to electricity and storing it. Turning sunlight into an electric current is done through photovoltaic conversion. PV conversion imitates the natural process of photosynthesis using a semiconductor (mostly Silicium) that capture the photons, thereby creating an electric imbalance in the cell which generates a current. Photovoltaic cells basic mechanism hasn't changed since their invention in 1954 although enhancements are being made up to this day like the addition of perovskite crystals for increased efficiency[11]. We can also see an evolution in how they are used, like concentrated solar plants that redirect sunlight to maximize light capture thereby increasing yield and efficiency. Even with these advancements, we could ask ourselves: why is solar irradiation still a secondary source of energy? The answer is that energy storage remains a bottleneck for its advancement.

Solar energy cannot be directly deployed on the electric grid because of the inconsistency of solar ray intensity. It must first be stored electrochemically i.e. in rechargeable batteries. Batteries come in all shapes and sizes, and each must be handled well to not degrade. Some types of batteries are vulnerable to overcharging, others must never be fully discharged or else they'll stop being operational, so on and so forth. To avoid these mistakes and keep the battery healthy, we must know the SoC (State of Charge).

Most mobile phones and all kinds of devices represent the state of charge as an emptying tank. If a battery is analogous to a water tank, then the SoC is how full the tank is relative to its capacity. SoC is a percentage which represents the level of charge of a battery.

The saying "knowing is half the battle" holds true here since it is crucial to know the state of charge at any given time when employing a battery powered electrical device, and doubly so for solar power storage. A regulator is essential to a photovoltaic system, since the energy output of the solar panels is unpredictable. And this electric regulator must receive a good estimate of the SoC to achieve this. This thesis' goal is to compare different methods of estimating SoC and applying them to different solar battery technologies to evaluate their viability.

1.2 Objectives

Through this thesis, we:

- Sleuth the available state of charge estimation literature and select suitable ones for our purposes.
- Obtain different state of charge estimation results from different selected methods.
- Perform an error analysis and compare different state of charge estimation methods in the context of solar energy storage.
- Compare two different battery technologies used for the purposes of solar energy storage.

1.3 Organisation of the thesis

We start off by going over the state of the art in terms of SoC estimation and solar battery technology in chapter 2. After a brief introduction to the concept of SoC estimation, we go over the types of batteries used in photovoltaic storage nowadays. Further in, we showcase SoC estimation methods from old to new but always relevant to current day battery management systems.

In chapter 3, we present the selected SoC estimation methods. First are the Ampere-Hour integral methods. They use the inherent relationship between the current and charge of a battery to continuously estimate the state of charge. We present the basic version and the improved version. Then comes the state filter known as the Kalman filter. It is a prediction-correction estimator that uses a feedback control. It makes use of the fact that the battery changes with the SoC and that the SoC is dependent on its previous value. To use the Kalman filter for battery SoC estimation, we first have to simulate the battery using an equivalent Thevenin model, meaning a generator, a resistor, and RC cells connected in series. This model is only valid for the Lithium-ion battery. We use this equivalent circuit to calculate the suiting Kalman filter parameters. We extend the kalman filter with linearisation to account for the battery's non-linearity. We then use data driven learning models. First, a BNN (Backpropagation Neural Network) is taught how to estimate the SoC with the voltage, current and temperature as features. For less implementation complexity, we also used two statistical algorithms often utilized in machine learning i.e. SVM and KNN which rely more on pure mathematical calculus and operate by solving a specific optimization problem.

Chapter 4 is dedicated to our work methodology and implementation. To start with, we explain the data acquisition methodology. We take care to showcase both batteries' test bench characteristics in doing so. Afterwards, we explain and showcase the SoC methods software implementation.

Chapter 5 concerns our resulting data, analysis and our conclusions. With our measurements and results in order, we proceed to analyze the results of our SoC estimations. To do so we perform quantitative analysis and qualitative analysis in both batteries' cases and for each SoC estimation method. After a thorough examination of our results, we validate the best SoC estimation method with a standalone PV cycle.

CHAPTER 2

STATE OF THE ART

2.1 Introduction

This chapter's goal is to present an overview of battery state of charge estimation research. First, we will explain what state of charge is and why it needs to be estimated. Then we will showcase the battery technologies used in our thesis work and the differences in how they function, namely lithium-ion batteries and lead-acid batteries. Lastly, we will have a rundown of the oft-used state of charge estimation algorithms, from those that use direct physical measurements to learning algorithms and mathematical models. This section will refer to the latest research in all kinds of battery applications from different fields to gain a broad perspective on the state of charge estimation world of research.

2.2 State of Charge Estimation

The state of charge is expressed as a percentage that indicates how much electric charge is left to consume in a battery. A fully charged battery's SoC equals 100%, while a fully discharged battery has a SoC of 0%. It cannot be measured directly, so it must be estimated using an online or offline method. Offline estimation means first collecting the data then using it to find model parameters which will estimate SoC. Online methods estimate the model as the process of charge or discharge is happening. Offline methods like ampere hour counting are more accurate but they are drawn out, costly and overall

impractical. That is why researchers have put a tremendous effort into developing online SoC estimation methods

SoC means different things in different contexts. In electric vehicles for example, SoC is equivalent to fuel gauge. It must take into account the drivers' interpretation of its value. While in a long-term energy storage context, SoC is primarily used to maintain the battery for a long time and formulate a control strategy. These different needs make the world of SoC estimation a vast one, with hundreds of papers on the topic.

State of charge estimation is a field as old as rechargeable batteries themselves, therefore we can find all kinds of methods to model and observe it in all types of batteries. However, depending on the type of battery and its use the accuracy of the estimation methods may be completely different. Since the advent of solar energy exploitation is a relatively recent field of study, there is yet to be a consensus for optimal storage methods. Fortunately, a consensus has come about in the vast and long-lived world of energy storage as to which methods are generally viable for a new battery application. It is by judiciously testing these tried-and-true methods on new cases that a suitable approach can be surmised and the understanding of its application can be expanded. Before an experimental methodology can be established, a thorough understanding of the case is needed. Meaning the type of battery, and the use for which the battery is intended.

2.3 Electrochemical Batteries

2.3.1 Definition

Electrochemical accumulators receive electrical energy and store it as potential chemical energy which can then be converted to an electrical current. A photovoltaic battery consists of several accumulators connected in series and/or in parallel to reach the desired equivalent voltage. Batteries function thanks to the oxidation and reduction of the electrodes, which manifest electrically in the charge and discharge of the battery. The electrodes are submerged in a conducive solution to allow current flow. The batteries are therefore categorized according to the materials of the cathodes and the solution. The ox/red reactions change the electrodes' composition, making them degrade over time. This change reduces the battery's efficiency, which is tantamount to its degradation. In order to preserve batteries for more extended periods, batteries with particular chemical compositions have been developed. These batteries can be recharged to restore the electrodes. Lithium-ion and lead-acid batteries are two examples of rechargeable

batteries.

2.3.2 Lithium-ion Batteries

Lithium-ion batteries are composed of a lithium-alloy metal oxide anode and a graphite cathode. Between the two electrodes is an electrolyte separator in the middle which permits the travel of ions. A symbiotic relationship between the two cathodes grants this type of battery a long lifespan. This composition is illustrated in figure 2.1[38]:

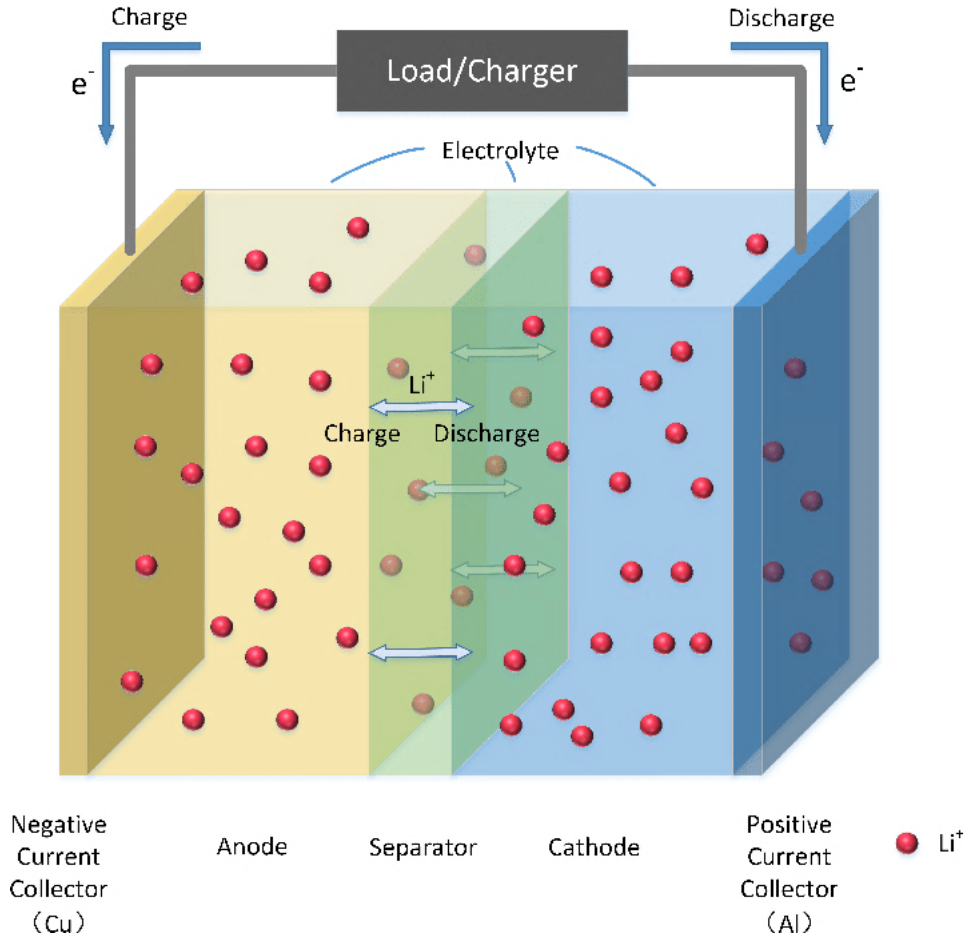

Figure 2.1: Lithium-ion battery Diagram.

Lithium-ion batteries are used in a wide array of fields and quickly replacing older technology like lead-acid. Portable electronics, electric vehicles, power backup are some of the many areas which employ lithium-ion batteries. In solar energy storage lithium-ion batteries tend to be more expensive due to the high voltages required for the task.

Figure 2.2 is a graph which shows the evolution of the current and voltage during a Lithium-ion battery charging cycle:

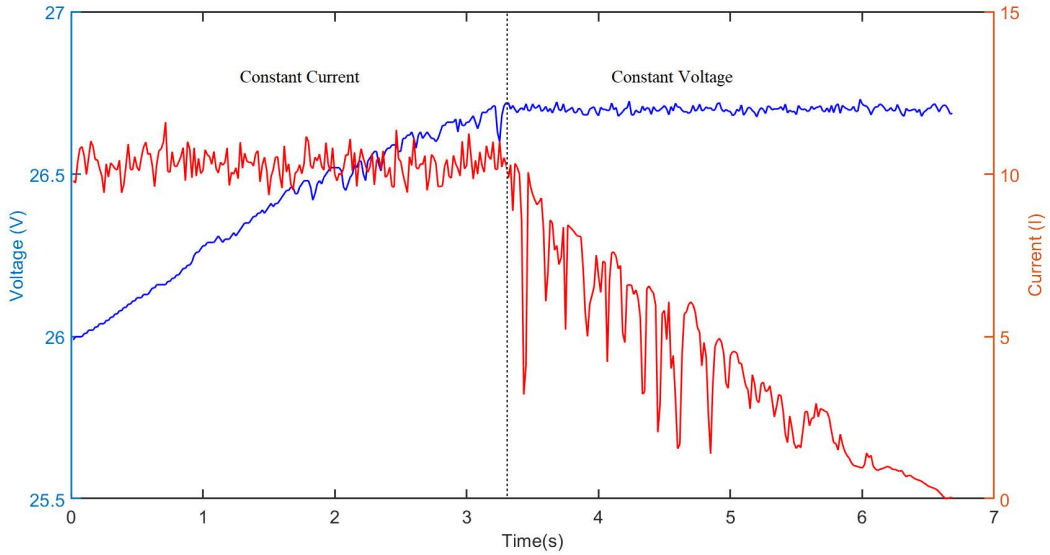

Figure 2.2: Lithium-ion battery charging stages.

A Lithium-ion battery is charged in two stages: a constant current stage (CC) followed by a constant voltage stage (CV). In the constant current stage, the voltage increases linearly until the maximum charge voltage is reached. It keeps this value while the current decreases until a certain point when the battery is fully charged. The lithium-ion can be discharged even below 30% SoC with little to no damage. The charging speed is also about four times faster than lead-acid batteries. Innovations are a constant in Lithium-ion batteries due to their prevalence, and they usually need SoC knowledge to function.

2.3.3 Lead-acid Batteries

Fig.2.3 is a diagram of a lead acid battery [21] :

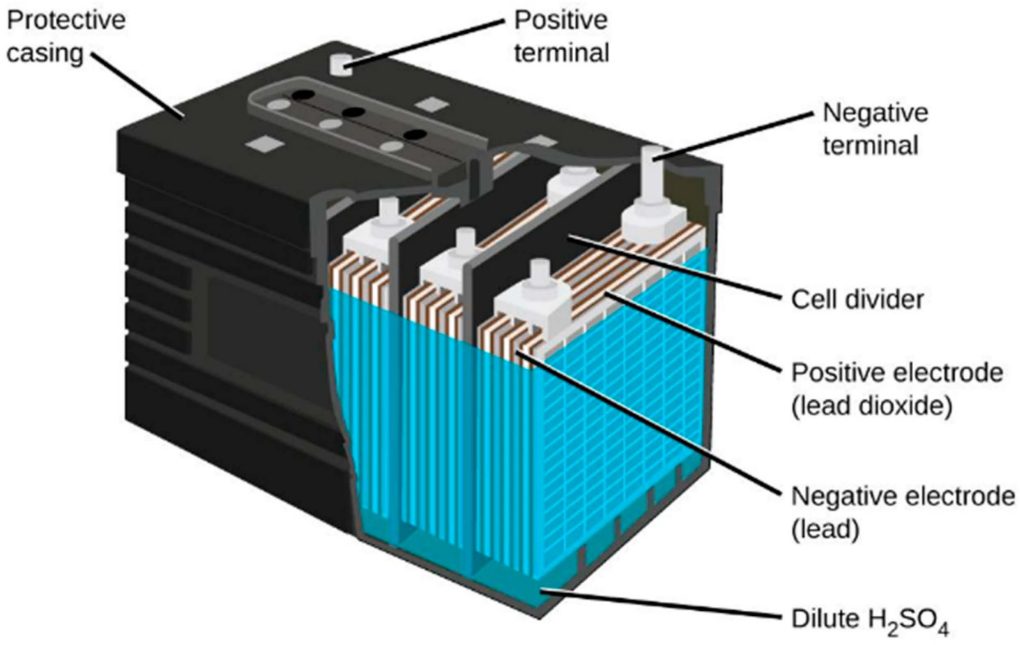

Figure 2.3: Lead-acid battery Diagram

The lead-acid battery accumulator comprises lead electrodes submerged in an acidic solution, usually sulfuric acid at around 1.3 density. The electrodes are composed of parallel plates alternating between positive (Lead Dioxide) and negative (Lead) plates as shown in figure 2.3. This distribution increases the battery capacity. These batteries have been used for almost two centuries, they are made from low-cost materials which makes them abundant in the market.

Lead-acid batteries are adapted to many different applications. The majority are used in automobiles as starting, lighting and ignition power supply. Backup and emergency power supply is another application for its ease of transport. They are also used as propulsion batteries for smaller battery powered vehicles like electric scooters. Solar energy storage primarily uses lead-acid batteries.

Figure 2.4 represents the terminal voltage and current of a lead-acid battery during charging.

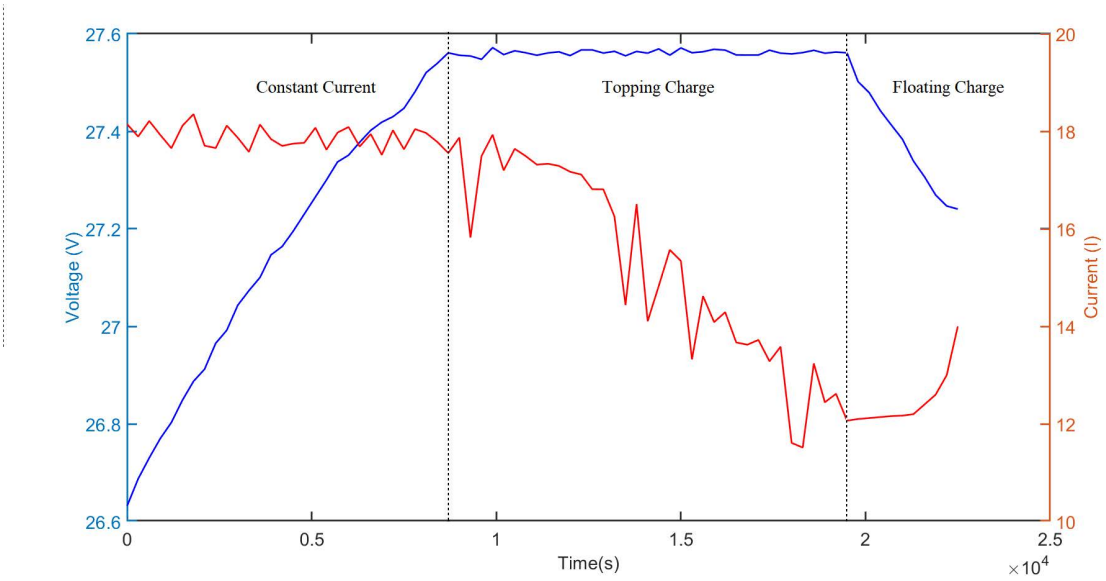

Figure 2.4: Lead-acid battery charging stages

There are three stages to charging in this method: first is the constant current charging phase. Aptly named, as can be seen. the current is constant while the voltage steadily increases. When the SoC reaches about 70% the "topping charge" stage begins. the battery continues charging at a lower current. This stage takes longer but is necessary to keep the battery healthy long-term, and it finishes the 30% charging left to reach saturation. The battery reaches full charge when the current is at a low level. The final stage is the "float charge" which keeps the battery at full charge through compensating for the self-discharge of the battery by reducing the voltage.

2.4 SoC Estimation Methods

2.4.1 Book-Keeping Approach

Book-keeping is a term employed in financial matters. Just like an actuary keeps track of money flow to infer the state of a business, an engineer can keep track of the flow of current to infer the state of charge. Book-keeping in our context means to keep track of the current of charge/discharge to deduce the SoC since the electric charge has direct a relation to the current which is depicted in equation 2.1:

$$q = \int i(t)\mathrm{d}t \tag{2.1}$$

Seeing as this expression is an unshakable physical law, there is only really one valid book-keeping method: the Ampere-hour integral method or more commonly known as Coulomb Counting. This is by far the most commonly used SoC estimation method, it can be found on devices all around you like mobile phones or various home appliances. However that declaration comes with an asterisk since it is used together with other, more case-specific methods. That is because it has drawbacks like the accumulation of errors. Indeed, as is the case with its financial book keeping, one error in the books will affect all the subsequent calculation. Improved Coulomb Counting algorithms have been proposed [14] but it is mostly used as a reference to adequately estimate the state of charge.

2.4.2 Direct Measurement

It is self-evident that the SoC will have a physical, measurable effect on the battery. So, the first approach towards SoC estimation was to track the evolution of the SoC with respect to a physical property. The SoC can be deduced by measuring said property. The physical property can be the battery's:

- Open Circuit Voltage: It has a proportional relationship with the SoC and is easily measured. The relationship is linear in the case of the lead-acid battery, and it is not for a Lithium-ion battery. Nevertheless, measuring the OCV at any given point allows us to infer the corresponding SoC. However, many factors disallow this simple method from being viable including the non-negligible amount of time for the OCV to reach its steady state when discharging, the considerable effect temperature has on the relationship, the low-variance phases in the SoC-

OCV evolution... This method is a starting point from which more sophisticated algorithms can sprout.

- Terminal Voltage : This method relies on the proportional relationship between the SoC and the electromotive force of the battery, which translates to a linear relationship between the SoC and the load voltage. The abrupt drop at the end of the battery discharge induces significant errors in estimation. [30]

- Internal Impedance: It is a complex quantity that links the battery voltage with its current. Its relation to the SoC can only be adequately represented by measuring it at different frequencies and by applying different current values. The relationship has also been found to be affected by temperature. Considering that all these factors have raised the cost of implementing an online impedance-based approach, it remains a well-performing and promising SoC estimation tool. Using a parameter identification method in conjunction with this method can reduce the errors down to 0.5%. [37]

2.4.3 Mathematical Models

The ideal that SoC estimation strives for is a simple mathematical algorithm like Coulomb Counting but with added robustness against errors. To achieve this an effort has been made to develop mathematical models that emulate the battery while outputting a good SoC estimation.

Some models like the Shepherd model [27] or the Unnewehr model [26] use an explicit expression of the battery's terminal voltage as a function of SoC. The Nernst model [12] is the most accurate of these. The Nernst equation 2.2 writes the terminal voltage as a function of not only the SoC but also current of the battery :

$$V_t(SoC) = OCV_{max} - R_0 i - \frac{a}{SoC} - bSoC + c.ln(SoC) + d.ln(1 - SoC) \tag{2.2}$$

Where OCV_{max} is the open circuit voltage when the battery is fully charged, R_0 is the internal resistance of the battery, and a, b, c and d are factors to be determined through a statistical regression algorithm using our measured data. This approach does not take into account the hysteresis effect. A more detailed $V_t(SoC)$ function can be established via regression techniques.

2.4.4 State Filters

State filters are an essential tool of control systems, and a control system needs an accurate estimate of whatever informs its actions. Voltage regulators being control systems, state filters are useful for estimating SoC. To calculate its current state, the state estimator uses previous estimates as an input in its calculations. This auto-regressive model approach allows considerable improvement of state estimation and tighter bounds for the values of the state. Of course, in our case the state in question is the SoC.

A recent example of a such filter is the H-infinity filter which uses weighted vector norms to minimize the maximum possible value of its error, for this reason it is also called the minmax filter. [5]

The particle filter is an analog of the Monte-Carlo method for Markov process, which earned it the name of Sequential Monte Carlo method. It estimates the posterior distribution of hidden states using the measurement process. It is a probabilistic method of interest with promising results. [8]

Despite being one of the oldest examples of state filters, the highly adaptable Kalman filter invented in 1960 [19] is still the most widely used in SoC estimation. It uses prior and current state estimations to extrapolate hidden states of the system. It even updates its own factors according to the changing output state. The Kalman filter uses a linear function of the current state depending on the previous state. In the case of Battery SoC estimation, the factors of the equation are matrixes. The elements of these matrixes are calculated via the component values of an equivalent Thevenin model.
The following chart in Fig.2.5 represents the Kalman family of filters:

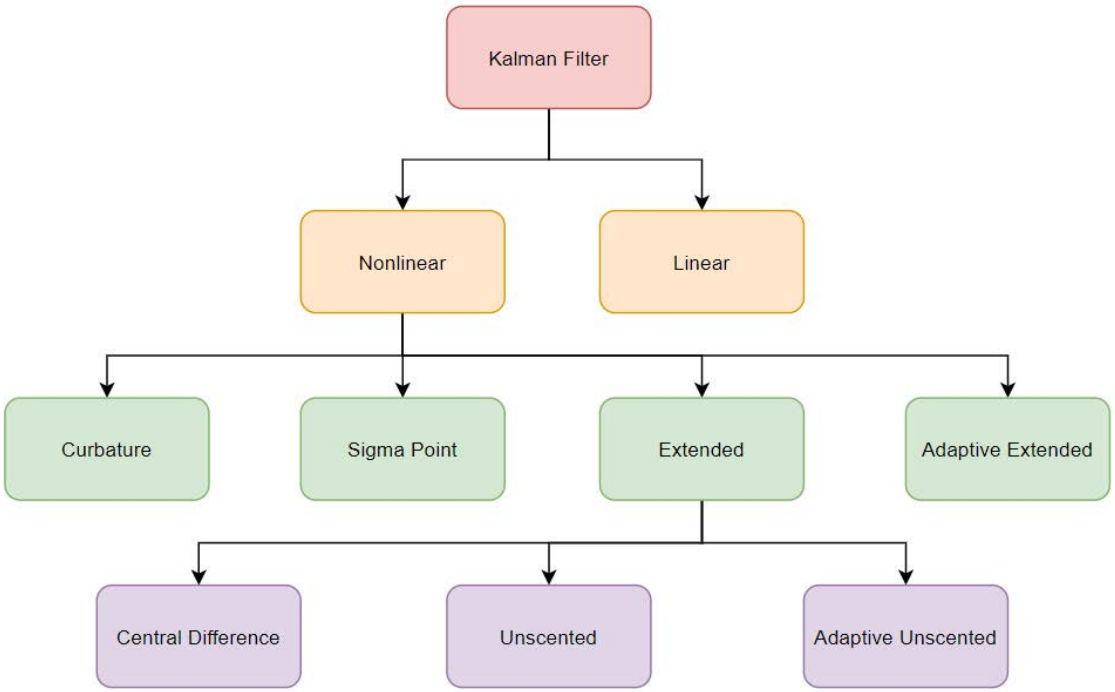

Figure 2.5: Kalman Filters Chart

Many improvements have been made upon the Kalman filter (2.5). The Extended Kalman Filter (EKF)[29] takes into account non-linear system model, while the Unscented Kalman Filter (UKF) [18] uses an unscented transformation to correct the model when the system is so highly non-linear that the EKF performs poorly.

2.4.5 Machine Learning Algorithms

They are learning-based data-driven models. Compared to the previous categories, this one is only in its infancy. That is why it has enjoyed a high level of enthusiasm and fast development.

. These methods exploit the Coulomb Counting results to train a model to predict the SoC value using parameters such as voltage and temperature. Amongst them are:

- Artificial Neural Networks: Since batteries are highly non-linear systems, mathematical models are hardly enough to simulate charge and discharge processes despite the scientific community's best efforts. ANNs are naturally non-linear which makes them suitable for the task. Backpropagation Neural Networks and

Recursive Neural Networks have both been used for SoC estimation with results surpassing the Coulomb Counting method in accuracy. Different types of ANNs like have also been used, like fuzzy neural networks [34] and like for all SoC estimation methods, each type of ANN has its strengths and weaknesses. [17]

- Support Vector Machines: SVMs construct hyperplanes that cut through the data which are close, under a set distance, to the regression line. Points that are on the edge of the tolerated distance are called support points which ensure that our predictions aren't far from the real SoC value. [39]
- Genetic Algorithm: Developed in the 1960s by John Holland, genetic algorithms try to mimic natural selection and biological evolution to solve optimization problems. Mutations, crossover and various biological phenomena are included in the structure of the algorithm, then a randomization process takes place according to the "survival of the fittest" principle. After each iteration a new "generation" is born and the process can be repeated to improve the optimization. They've been used in for SoC estimation and found to have good dynamic performance and robustness [7].

Other methods such as K-Nearest Neighbors [32], extreme learning machines [15] and deterministic observers [16] have been used for battery SoC estimation with varying levels of success.

2.4.6 Hybrid Methods

Different SoC estimation methods yield different results of varying accuracy, with each method having its own pros and cons. By combining different approaches when possible, researchers are able to discover symbiotic relationships that compensate for the fault of each one. [20, 22]

2.5 Conclusion

Throughout this chapter, we've showcased the state of the art regarding battery SoC estimation. After a brief overview of SoC estimation in general, we delved into the relevant lithium technologies for solar energy storage, namely lead-acid natteries and lithium-ion batteries. Showing our current understanding and briefly explaining the latest in their applications and charging methods. Then we've gone over the current state of SoC estimation, by categorizing and briefly describing the methods with which SoC is estimated to this day. Some on the cutting edge of development e.g. Genetic Algorithms, and some tried and true methods that are still being expanded upon to this day e.g. Kalman filtering.

CHAPTER 3

DESCRIPTION OF SELECTED MODELS FOR SOC ESTIMATION

3.1 Introduction

This chapter aims to pose the State of Charge Estimation problem for lithium-ion and Lead-acid batteries. We give a rundown of the five chosen models and later used by us in practice for that purpose. We attempt to explain the theory behind each modelling to varying degrees of complexity while also elucidating the model's approach to the problematic.

Energy storage systems are widely used in photovoltaic systems (PV), power grids, and electric vehicles. In PVs specifically, a storage system -usually comprised of several battery cells connected in series or parallel- is necessary for safeguarding and ensuring the operation at night. However, these batteries usually need an additional management system to monitor their state, and make sure they do not function outside their safe operating area. To that end, the battery management system (BMS) performs numerous calculations and computes several variables related to the battery state. One significant value is the battery State of Charge (SOC), representing the actual capacity expressed as a percentage of the fully charged capacity. Unfortunately, it is amply delicate to have a precise calculation of the SoC since it depends on parameters essentially tied to the chemical state of the battery, which are very hard -if not impossible- to measure and trace its evolution. To remedy that, we will attempt to select models and algorithms meant to

estimate the SoC, using practically measurable parameters (Temperature, Voltage and Current...etc.)

As discussed in the previous chapter, many SoC estimation models have been conceived, each having its own advantages and disadvantages. In our case, we had to select a set of models that would be compatible with the measurements carried out in the experimental set up, and the two battery technologies used. In other words, we had to choose models that would use measures related to the external brightness, the battery temperature voltage and current of the battery, the load and the source (PV). The models settled for were:

1. The Coulomb-counting method (for both Lithium-ion and Lead-acid batteries).
2. Modified Coulomb Counting (Lithium-ion / Lead-acid).
3. Extended Kalman filter (Lithium-ion).
4. Neural Networks (Lithium-ion / Lead-acid).
5. Machine Learning algorithms (Lithium-ion / Lead-acid):
    a) Support Vector Machines
    b) K-Nearest neighbors

3.2 Coulomb-counting

The Coulomb counting method measures the discharging current of a battery and integrates it over time in order to estimate SoC [35], as given by:

$$SoC(t) = SoC(t_0) - \int_{t_0}^{t} \frac{\eta.\tau.\ I(\tau)}{Q_n} \tag{3.1}$$

η : Efficinecy during charge and discharge operations.
Q_n : Nominal Capacity

In case of discrete data separated by a sampling interval Δt, the integration becomes the cumulative summation of the previous states with the present fraction of the current on nominal capacity:

$$SoC_{k+1} = SoC_k - \frac{\eta.\Delta t.I_k}{Q_n} \tag{3.2}$$

Coulomb-counting presents many disadvantages, the major one being its dependency

on the Battery's efficiency (η) and nominal capacity (Qn) which -unlike their nature in the equation- are not constants, and are actually influenced by the temperature and current direction. Despite that, it is going to serve as a standard reference which we attempt to approximate using the models that will follow.

Using the Coulomb counting estimation of the SoC as a reference is a common practice in the field of State of Charge estimation, this is due to its simplicity of implementation and the accuracy it offers without needing any data related to the chemistry of the battery. However, its use is often limited to serving as a reference to tune other estimation models in the testing phase (on PC), rather than being practically used in battery management systems. Since the latter often have computing units with minimal accuracy, the errors would accumulate with each step of the integration, leading to very erroneous results.

3.3 Modified Coulomb-counting

As previously explained, the Coulomb Counting's dependence on a constant value of the battery's efficiency η (a parameter that varies in theory) is quite inconvenient. In order to make the model more useful, η is assumed to be 1 for discharge and 98% for charge instead of having one constant value for all scenarios. For what follows, this variant of the Coulomb Counting will be referred to as the Modified Coulomb Counting (MCC).

3.4 Extended Kalman Filter

3.4.1 The Original Formulation of The Kalman Filter

3.4.1.1 Classical Kalman Filter

What follows is a description of the original formulation of the filte as explained in [36]. The Kalman filter addresses the general problem of estimating the state $x \in \mathscr{R}^n$ of a discrete-time controlled process that is governed by the linear stochastic difference equation 3.3a with a measurement $z \in \mathscr{R}^n$ related to x_k as in 3.3b. These two equations are called "the State Space" equations.

$$x_k = Ax_{k-1} + B\ u_k + w_{k-1} \tag{3.3a}$$

$$z_k = Hx_k + v_k \tag{3.3b}$$

w_k and v_k are random independent variables that represent the process noise and measurement noise (both assumed to be of white nature), Their probability distributions are considered normal:

$$\begin{cases} p(w) \sim N(0,Q) \\ p(v) \sim N(0,R) \end{cases} \tag{3.4}$$

- Q is the process noise covariance and R is the measurement noise covariance.
- The subscript indicates the index of the sample in the time series, thus 'k' refers to an actual state, whereas 'k-1' indicates a previous state. $u \in \mathscr{R}^l$ is the optional control input.
- In practice, the matrices A, B, H, Q and R can vary with each step, but for now we will assume they remain constant.

The Computational Origins of the Filter

PS: From this point onward the superscript '$^-$' will indicate that the estimate is in its "à priori" stage, which means that it is expected to undergo a correction later. If there is no superscript '$^-$', the estimate is in its corrected (or "à posteriori") form. Let $\hat{x}_k^- \in \mathscr{R}^n$ be the à priori estimate at step k, and $\hat{x}_k \in \mathscr{R}^n$ the à posteriori estimate. Their respective errors are defined as:

$$e_k^- = x_k - \hat{x}_k^- \tag{3.5a}$$

$$e_k = x_k - \hat{x}_k \tag{3.5b}$$

And their covariances are respectively:

$$P_k^- = E[e_k^- \; e_k^{-T}] \tag{3.6a}$$

$$P_k = E[e_k \; e_k^T] \tag{3.6b}$$

The goal of the Kalman filter is to create an equation in which the à posteriori estimate $\hat{x}_k^-$ is computed as a linear combination of the à priori estimate $\hat{x}_k^-$ and the measurement residual (the residual is the difference between the actual measurement z_k and the ideal measurement prediction Hx^k). In the implementations of the Kalman filter, this is usually referred to as 'The correction step':

$$\hat{x}_k = \hat{x}_k^- + K(z_k - H\hat{x}_k^-) \tag{3.7}$$

K (n x m) is the gain or blending factor (often called the Kalman gain). The purpose of such a matrix is to create a corrected version of the estimate ($\hat{x}_k$) that is the closest possible to the real state x_k based on the à priori state. i.e. it means to minimize the à posteriori error covariance e_k as much as possible.

Analytically, to accomplish this minimization we substitute equation 3.7 into the definition for e_k 3.5b and then substitute that into the covariance definition 3.6b. We then compute the corresponding expected value ($E[e_k e_k^T]$), derive it with respect to K and solve the null equation for K, we obtain:

$$K_k = \frac{P_k^- H^T}{\left(H P_k^- H^T + R\right)} \tag{3.8}$$

Notice that:

$$\begin{cases} \lim\limits_{R_k \to 0} \; K_k = H^{-1} \\ \lim\limits_{P_k^- \to 0} \; K_k = 0 \end{cases} \tag{3.9}$$

The Probabilistic Origins of the Filter

The Kalman filter conserves the first and second moments of the state probability distribution, which means that the à posteriori state estimate is equal to the mean (the first moment) of the state distribution and the à posteriori estimate error covariance equation 3.6b reflects the variance of the state distribution (the second non-central moment):

$$\begin{cases} E\left[x_k\right] = \hat{x}_k \\ E\left[e_k \; e_k^T\right] = E\left[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T\right] = P_k \end{cases} \tag{3.10}$$

The state distribution conditioned on all previous measurements z_k ($p(x_k|z_k)$) is considered normal if the conditions of equations 3.4 are satisfied:

$$p\left(x_k | \; z_k\right) \sim N\left(E\left[x_k\right], E\left[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T\right]\right) = N(\hat{x}_k, \; P_k) \tag{3.11}$$

The Kalman Filter Algorithm

The Kalman filter is a prediction-correction estimator that uses a feedback control. We can group the equations governing the filter into two categories:

- The time update equations (or predictor equations) to project forward in time (compute the à priori estimates)
- The measurement update equations (or correction equations) to serve as feedback (compute a corrected à posteriori estimate from the à priori one).

Time update/ Prediction equations:

$$\begin{cases} \hat{x}_k^- = A\hat{x}_{k-1} + B\ u_k \\ P_k^- = A\ P_{k-1}A^T + Q \end{cases} \tag{3.12}$$

Measurement update/ Correction equations:

$$\begin{cases} K_k = P_k^- H^T \left(HP_k^- H^T + R\right)^{-1} \\ \hat{x}_k = \hat{x}_k^- + K\left(z_k - H\hat{x}_k^-\right) \\ P_k = (I - K_k H)P_k^- \end{cases} \tag{3.13}$$

3.4.1.2 Extended Kalman Filter

The extended Kalman Filter is used when the process to be estimated and (or) the measurement relationship to the process is non-linear. A Kalman filter that linearizes about the current mean and covariance is referred to as an extended Kalman filter or EKF.

In a similar fashion to how a Taylor series works, we can linearize the estimation around the current estimate using the partial derivatives.

The process x and the measurement z are represented by non-linear stochastic difference equations.

$$x_k = f(x_{k-1},\ u_k,\ w_k) \tag{3.14a}$$

$$z_k = h(x_k, v_k) \tag{3.14b}$$

Since the values of w_k and v_k are unknown in practice, we will define the approximate values of the state and measurement vectors ($\widetilde{x}_k$ and $\widetilde{z}_k resp.$) which do not take the noise terms into consideration as:

$$\widetilde{x}_k = f(\hat{x}_{k-1},\ u_k, 0) \tag{3.15a}$$

$$\widetilde{z}_k = h(\widetilde{x}_k, 0) \tag{3.15b}$$

The Computational Origins of the Filter

First, we write new governing equations that linearize an estimate of $\widetilde{x}_k$ and $\widetilde{z}_k$:

$$x_k \approx \widetilde{x}_k + A(x_{k-1} - \hat{x}_{k-1}) + W\ w_{k-1} \tag{3.16a}$$

$$z_k \approx \widetilde{z}_k + H(x_k - \hat{x}_k) + V\ v_{k-1} \tag{3.16b}$$

Where :

- $\widetilde{x}_k$ and $\widetilde{z}_k$ are the approximate state and measurement vectors from equation 3.15a and equation 3.15b
- A and W are the Jacobian matrices of partial derivatives of f with respect to x and w respectively:

$$A_{[i,j]} = \frac{\partial f_{[i]}}{\partial x_{[j]}}(\hat{x}_{k-1},\ u_k, 0),\ \ W_{[i,j]} = \frac{\partial f_{[i]}}{\partial W_{[j]}}(\hat{x}_{k-1},\ u_k, 0)$$

- H and V are the Jacobian matrices of partial derivatives of h with respect to x and v respectively:

$$H_{[i,j]} = \frac{\partial h_{[i]}}{\partial x_{[j]}}(\hat{x}_{k-1},\ u_k, 0),\ \ V_{[i,j]} = \frac{\partial h_{[i]}}{\partial v_{[j]}}(\hat{x}_{k-1},\ u_k, 0)$$

  For simplicity in notation we will not use the subscript k with A , W , H , and V for now.

  Let the prediction error and the measurement residual be:

$$\widetilde{e}_{x_k} = x_k - \widetilde{x}_k \tag{3.17a}$$

$$\widetilde{e}_{z_k} = z_k - \widetilde{z}_k \tag{3.17b}$$

In practice we do not have access to x_k but we do have access to z_k. Using equations 3.17a and 3.17b , the governing equations for the error process can be expressed as:

$$\widetilde{e}_{x_k} \approx\ A(x_{k-1} - \hat{x}_{k-1}) + \varepsilon_k \tag{3.18a}$$

$$\widetilde{e}_{z_k} = H\widetilde{e}_{x_k} + \eta_k \tag{3.18b}$$

The random variables having approximately the following probability distributions

$$\begin{cases} p(\widetilde{e}_{x_k}) \sim N(0, E[\widetilde{e}_{x_k}, \widetilde{e}_{x_k}^T]) \\ p(\varepsilon_k) \sim N(0, WQ_kW^T) \\ p(\eta_k) \sim N(0, VR_kV^T) \end{cases} \tag{3.19}$$

Where ϵ_k and η_k represent new independent random variables having zero mean and covariance matrices WQW^T and VQV^T.
Equation 3.18a and equation 3.18b are linear, closely resembling the difference and measurement equations 3.3a and 3.3b from the discrete Kalman filter: This suggests the idea of using $\tilde{e}_{z_k}$ and a second Kalman filter to estimate $\tilde{e}_{x_k}$ the same way we estimated x_k using z_k in the regular Kalman algorithm. We can then obtain the à posteriori state estimates for the original non-linear process using the estimate $\hat{e}_k$:

$$\hat{x}_k = \tilde{x}_k + \hat{e}_k \tag{3.20}$$

The Kalman Filter equation to estimate $\hat{e}_k$ is:

$$\hat{e}_k = K_k \tilde{e}_{z_k} \tag{3.21}$$

Substituting equation 3.21 into 3.20 and then projecting that into equation 3.17b we obtain the measurement update (correction) equation:

$$\begin{aligned} \hat{x}_k &= \tilde{x}_k + K_k \tilde{e}_{z_k} \\ &= \tilde{x}_k + K_k (z_k - \tilde{z}_k) \end{aligned} \tag{3.22}$$

Extended Kalman Filter Algorithm

We will revert to using the superscript '-' notation and will refer by it to the approximates ($\tilde{x}_k$, $\tilde{z}_k$) that do not take noise into consideration (and must therefore be corrected in the measurement update step). Moreover, we will add the subscript k to the Jacobians A , W , H , and V , to indicate that they change with every time step.
Time update/ Prediction equations:

$$\begin{cases} K_k = P_k^- H^T \left(H P_k^- H^T + R \right)^{-1} \\ P_k^- = (I - K_k H) P_k^- \end{cases} \tag{3.23}$$

Measurement update/ Correction equations:

$$\begin{cases} K_k = P_k^- H_k^T \left(H_k P_k^- H_k^T + V_k R_k V_k^T \right)^{-1} \\ \hat{x}_k = \hat{x}_k^- + K_k (z_k - h(\hat{x}_k^-, 0)) \\ P_k = (I - K_k H_k) P_k^- \end{cases} \tag{3.24}$$

3.4.2 Formulation of The Kalman Filter for Battery SoC Estimation

3.4.2.1 Battery modeling

To build the Kalman filter, first we need to model the battery with an equivalent circuit model (ECM) in order to establish the state space equations governing the battery. According to [10], The typical ECM for a Lithium-ion battery consists of:

- A voltage source, in this case the OCV (Open Circuit Voltage)
- A resistor in series R_0 acts as an internal resistance for the battery and represents contact resistance among the parts such as the electrode material, diaphragm resistance and electrolyte.
- 'n' RC networks to reflect the dynamic characteristics such as the diffusion effect and the polarization effect of the battery.

The model is based on the Thevenin's theorem, where the entire circuit is replaced by a voltage source (Thevenin voltage, which is the OCV), a resistor (Thevenin resistance) and an impedance z represented by the RC cells.
P.S.: The number of RC networks determines the order of the model, choosing a higher order model will indeed make the ECM more physically accurate, but will make the determination of its parameters much more complex and unnecessarily lengthen the simulations [10]. For our application, a first order Thevenin ECM is going to be used, as shown in figure 3.1

3.4.2.2 Establishing the state-space equations for the equivalent circuit

Using Kirchhoff's relations:

$$
\begin{aligned}
&V_1 = R_1 i_r = R_1 (I_L - i_c) \\
&C_1.V_1 = q \\
&\implies C_1.\frac{dV_1}{dt} = \frac{dq}{dx} = i_c \\
&\implies V_1 = R_1 \left(I_L - C_1.\frac{dV_1}{dt} \right)
\end{aligned}
$$

Finally:

$$
\frac{dV_1}{dt} = \frac{I_L}{C_1} - \frac{V_1}{R_1.C_1} \tag{3.25}
$$

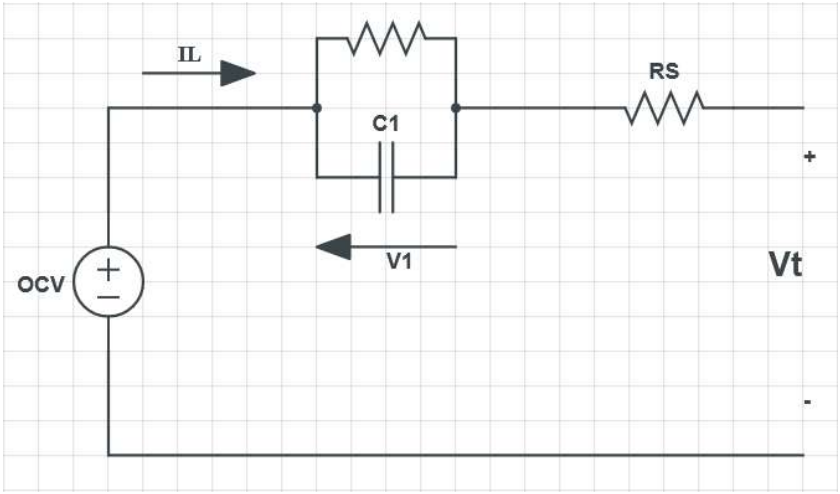

Figure 3.1: Thevenin Model Circuit

On the other hand:

$$V_t = OCV(SoC) - I_L.R_0 - V_1 \tag{3.26}$$

PS: OCV is dependent on the SoC ($OCV = h(SoC)$) where h is a non-linear function.

Let 'X' be the state vector of the system, 'u' the input vector and 'z' the output vector. X is set to be [V1, SoC], the current (IL) is the input 'u' and the terminal voltage Vt is the output 'y'.

$$\begin{cases} u = I_L \\ z = V_t \\ X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} V_1 \\ SoC \end{bmatrix} \end{cases}$$

The solution to equation 3.25 is the combination of the Zero-Input Response (ZIR) and the Zero-State Response (ZSR) [24]:
The ZIR can be determined by setting the inputI_L to '0':

$$\frac{dV_1}{dt} = -\frac{V_1}{R_1.C_1}$$

By integrating over an interval of 0 to t:

$$\int \frac{dV_1}{V_1} = -\frac{1}{R_1.C_1} \int d\ t$$
$$\Longrightarrow V_1(t) = Ke^{-\frac{t}{R_1.C_1}}$$

With the initial condition being: $V_1(t) = V_1(0)$

$$V_1^{ZIR}(t) = V_1(0)e^{-\frac{t}{R_1.C_1}} \tag{3.27}$$

For the ZSR, the input is considered a constant I_L and the initial state value is considered zero (V1(0) =0), the equation becomes a first order ODE with a non-null right side, it is solved by combining the homogeneous solution and a particular solution: For the particular solution, it is clear that setting V1 as $R_1 I_L$ satisfies the equation:

$$V_{1,p}(t) = R_1 I_L$$

The homogeneous solution is found in a similar fashion to the ZIR, i.e.:

$$V_{1,h}(t) = K\prime\, e^{-\frac{t}{R_1.C_1}}$$

Finally, the ZSR solution:

$$V_1(t) = V_{1,p}(t) + V_{1,h}(t) = R_1 I_L + K\prime\, e^{-\frac{t}{R_1.C_1}}$$

All that remains is to find the value of the constant K', which is done by using the ZSR condition:

$$\begin{aligned} & V_1(t=0) = 0 \\ & \implies R_1 I_L + K' e^{-\frac{0}{R_1.C_1}} = 0 \\ & \implies R_1 I_L + K' = 0 \\ & \implies K' = -R_1 I_L \end{aligned}$$

$$V_1^{ZSR}(t) = R_1 I_L (1 - e^{-\frac{t}{R_1.C_1}}) \tag{3.28}$$

Discretizing the ZIR and the ZSR respectively with a sampling interval of Ts gives:

$$\begin{cases} V_{1,k+1}^{ZIR} = V_{1,k} e^{-\frac{T_s}{R_1.C_1}} \\ V_{1,k+1}^{ZSR} = R_1 I_{L,k} (1 - e^{-\frac{Ts}{R_1.C_1}}) \end{cases} \tag{3.29}$$

The full voltage response is:

$$V_{1,k+1} = V_{1,k} e^{-\frac{T_s}{R_1.C_1}} + R_1 I_{L,k} (1 - e^{-\frac{T_s}{R_1.C_1}}) \tag{3.30}$$

We can combine equation 3.2 and equation 3.30 in matrixial form to constitute the state space model of the equivalent circuit :

$$\begin{bmatrix} V_{1,\ k+1} \\ SoC_{k+1} \end{bmatrix} = \begin{bmatrix} e^{-T_s/R_1C_1} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} V_{1,\ k} \\ SoC_k \end{bmatrix} + \begin{bmatrix} R_1(1 - e^{-T_s/R_1C_1}) \\ -\eta.T_s/Q_n \end{bmatrix} I_{\ k} + w_{\ k} \tag{3.31}$$

$$X_{k+1} = \begin{bmatrix} e^{-T_s/R_1C_1} & 0 \\ 0 & 1 \end{bmatrix} X_k + \begin{bmatrix} R_1(1-e^{-T_s/R_1C_1}) \\ -\eta.T_s/Q_n \end{bmatrix} u_k + w_k \tag{3.32}$$

Let A and B be the two matrices:

$$A = \begin{bmatrix} e^{-\frac{T_s}{R_1C_1}} & 0 \\ 0 & 1 \end{bmatrix} \quad , \quad B = \begin{bmatrix} R_1\left(1-e^{-\frac{T_s}{R_1C_1}}\right) \\ -\eta.\frac{T_s}{Q_n} \end{bmatrix}$$

Thus:

$$X_{k+1} = A.X_k + B.u_k + w_k \tag{3.33a}$$

$$V_t = OCV(SoC) - I_L.R_0 - V_1 \tag{3.33b}$$

Let H be the global non-linear function that links the state elements (SoC, V1) and the input I_L to the output Vt. For the sake of computational simplicity, we will choose to keep the noise as a separate term from the non-linear function:

$$X_{k+1} = A.X_k + B.u_k + w_k \tag{3.34a}$$

$$y = H(X_k,\ u_k) + v_k \tag{3.34b}$$

PS: There are three noticeable differences between the obtained state space equations and the ones used in the original formulation of the extended Kalman Filter:

- **Unlike the original formulation, the first state space equation (equation 3.34a) is indeed linear, and only the measurement equation (equation 3.34b)) has a non-linear function in it. This is by no means an anomaly; the original formulation is a broad and general one, and it adapts differently to different cases.**
- **The measurement/output (in this case Vt) also depends on the input u (i.e I_L). Once we get to the linearization, this will not be a problem since the Jacobian Matrix with respect to 'u' doesn't affect the Algorithm.**
- **As mentioned before, the measurement noise v_k is not included in the non linear function H for the sake of simplicity**

Equation 3.34b is linearized using first order Taylor series:

$$y = \frac{\partial H}{\partial X}\delta X + \frac{\partial H}{\partial u}\delta u$$

We set C and D to be the two matrices:

$$\begin{cases} C = \dfrac{\partial H}{\partial X} = \begin{bmatrix} \frac{\partial H}{\partial V_1} \\ \frac{\partial H}{\partial SoC} \end{bmatrix} = \begin{bmatrix} -1 \\ \frac{\partial OCV}{\partial SoC} \end{bmatrix} \\ D = \dfrac{\partial H}{\partial u} = -R_0 \end{cases}$$

Therefore:

$$y = C.\delta\ X + D.\delta\ u$$

By discretizing the previous equation, we obtain:

$$y_{k+1} = C.X_k + D.u_k + v_k$$

Ultimately, the state-space equations for the system are:

$$\begin{cases} X_{k+1} = A.X_k + B.u_{\ k} + w_{\ k} \\ y_{k+1} = C.X_k + D.u_k + v_k \end{cases} \tag{3.35}$$

$$\begin{cases} A = \begin{bmatrix} e^{-\frac{T_s}{R_1 C_1}} & 0 \\ 0 & 1 \end{bmatrix}, B = \begin{bmatrix} R_1\left(1 - e^{-\frac{T_s}{R_1 C_1}}\right) \\ -\eta.\frac{T_s}{Q_n} \end{bmatrix} \\ C = \begin{bmatrix} -1 \\ \frac{\partial OCV}{\partial SoC} \end{bmatrix}, = [-R_0] \end{cases} \tag{3.36}$$

The steps of the Extended Kalman Filter can thus be summarized as follows [4]:

1. Initialization:
$$\begin{cases} \hat{X}_0 = X_0 \\ \hat{P}_0 = P_0 \end{cases}$$
2. Estimate next state based on equation 3.33a:
$$\hat{X}^-_{k+1} = A.\hat{X}^-_k + B.u_{\ k} + w_{\ k}$$
3. Compute error covariance:
$$P^-_{k+1} = A.P^-_k.A^T + QP^-_{k+1} = A.P^-_k.A^T + Q$$
4. Compute the Kalman gain:
$$K_{k+1} = P^-_{k+1}.\ C^T.[C.Pk - .CT + R]^{-1}$$

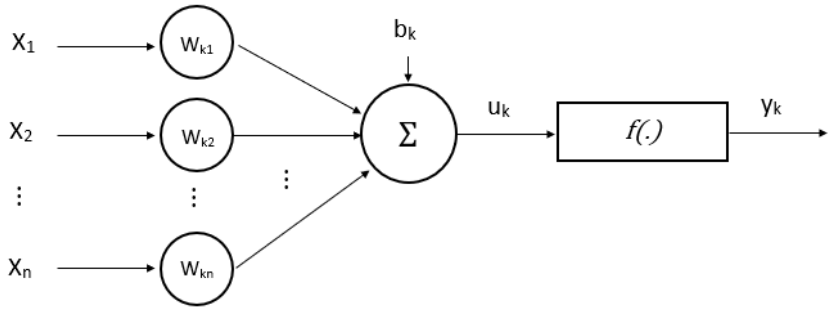

Figure 3.2: Deep Neural Networks' Structure Diagram

5. Correction of the state estimation:

$$\hat{X}_{k+1} = \hat{X}^{-}_{k+1} + K_{k+1}[y_k - C.\hat{X}^{-}_{k+1}]$$

6. Correction of the error covariance:

$$P_k = (I - K_k C)P^{-}_k$$

7. Loop back to step 2 for the next sample.

3.5 Artificial Neural-Networks

Artificial Neural Networks are a computational model composed of artificial neurons (called Perceptrons), linked by weights that can be updated depending on the performance parameters that evaluate the accuracy of the present prediction [1]. The basic neural network is composed of three main layers, the first being the inputs layer, the middle one is called the hidden layer (due to it being unrelated to the outside) and the last layer produces outputs. A model of a basic neural network is displayed in 3.2

More complex neural networks contain more than a single hidden layer, and are the basis of deep learning.

- $X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ is the input vector.

- $W_k = \begin{bmatrix} Wk_1 \\ Wk_2 \\ \vdots \\ Wk_n \end{bmatrix}$ is the weights vector.
- b_k is the bias term.
- u_k is the linear combination of X, W and b: $u_k = W.X + b$
- f(.) is the activation function (usually nonlinear).
- y_k is the output vector.

The input vector consists of recorded/measured variables called 'features', the latter are passed to the hidden layers where they get multiplied by weights and summed with the bias. At the end of the network the linear combination is passed through an activation function producing an output (or prediction). A back propagation algorithm is used in which the error of the current prediction (i.e., the difference between the current output and the desired output) will influence the weights, the latter get updated to accommodate the lack of accuracy and produce better predictions for the next step (Epoch).

After the network had been trained and the weights have been fixed, the test phase takes place: A portion of the data never seen before by the network is passed through it to produce predictions, only this time there is no back-propagation or updating the weights, because the test is meant to evaluate our pre-trained model. For that purpose, a number of error metrics are taken into record to give an idea of the model's ability to work on new unseen data.

In our case, the input vector will contain the variables measured in the experimental phase, i.e., the temperature, the discharge/charge current and the voltage. Whereas the desired output is going to be the coulomb-counting state of charge. As for the choice of training and testing data, several strategies will be discussed in the methods section.

3.6 Machine Learning algorithms

Machine learning is a broad term that can signify any algorithm that learns and improves upon supplied data, the term encompasses even Artificial Neural Networks and Deep Learning. However, in this section, by 'machine learning algorithms' we refer to unique learning algorithms that deal with tabular data, and use computational statistics.

It is worth noting that most of these algorithms are usually used for classification problems rather than regression (Classification is the type of learning in which the model attempts to predict a category (i.e., A discrete value), whereas regression is the process of predicting a continuous variable). Consequentially, we had to select algorithms that had a variant compatible with regression problems to comply with the nature of our objective consisting of predicting the continuous value of SoC. The two algorithms settled for were:

3.6.1 Support Vector Machines (Support Vector Regressor)

Let $X = [x_1 \quad x_2 \ \ldots \quad x_n{}^T]$ be the training data and $y = [y_1 \quad y_2 \quad \ldots \quad y_n]^T$ the target outputs. ϵ-SV regression attempts to find the prediction function f (x) that has at most a deviation of ϵ from the true values y_i, but must also be as flat as possible. In this section we summarize the SVM regression's principle described in Tutorial [33]

Linear Functions case
Linear functions follow the model:

$$f(x) = \langle w, x \rangle + b \tag{3.37}$$

where $\langle ., . \rangle$ is the dot product. In the basic linear case, the problem translates to finding a linear margin (f(x)) that is exactly 'ϵ ' away from the regression line (the line which passes by most of the observation points). This means that the points that are precisely deviated by ϵ from the regression line are Support points on which the delimiter function f(x)) leans, thus the name 'Support Vectors'. This way we ensure that predictions given by the linear function f(x) are only different by a small amount of ϵ to the true target values.

Plus, we want f(x) to be as flat as possible so we must minimize the norm of the weight's matrix w; thus, the formulation of the optimization problem:

$$\begin{aligned} &minimize \ \ \frac{1}{2}||w||^2 \\ &Subject\ to \begin{cases} y_i - \langle w, x \rangle - b \leq \epsilon \\ \langle w, x \rangle + b - y_i \leq \epsilon \end{cases} \end{aligned} \tag{3.38}$$

The latter is referred as a 'convex optimization problem'

In order to increase the feasibility of the convex optimization problem, we may want to tolerate some extra errors by introducing 'slack variables' (ξ_i, ξ_i^*) to the problem.

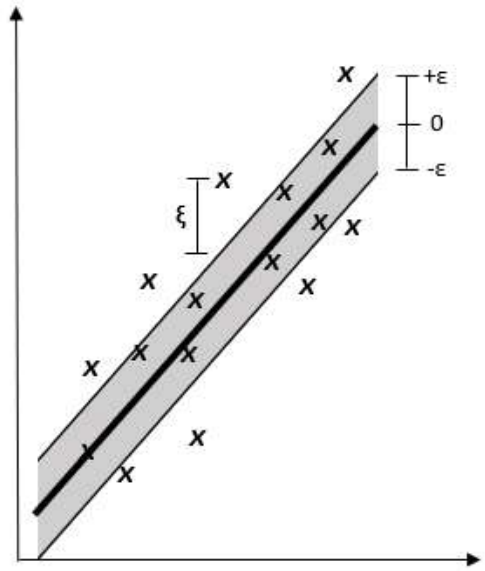

Figure 3.3: Support Vectors and slack variables for SVR

However, this slack interval is going to cost us in terms of the flatness of f(x). This trade-off is determined by the coefficient C, which translates how much we intend to tolerate deviations greater than ϵ (high C for low tolerance and inversely):

$$\begin{aligned} &minimize \ \frac{1}{2}||w||^2 + C.\sum_{i=1}^{l}(\xi_i + \xi_i^*) \\ &Subject\ to \begin{cases} y_i\langle w,x\rangle - b \leq \epsilon + \xi_i \\ \langle w,x\rangle + b - y_i \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases} \end{aligned} \tag{3.39}$$

The observations on and outside the $epsilon$ band are "support vectors". The slack value is illustrated for one of the observations in figure 3.3. The solution to the optimization problem depends only on this restricted set of observations (the support vectors).

Non Linear Functions

Elevating and adapting the linear situation to non-linear will necessitate two steps :

1. **Dual Formulation of the problem:**
   A Lagrange function is constructed from the objective function and the constraints, by introducing a dual set of variables.

$$L = \frac{\mathbf{1}}{\mathbf{2}}||w||^{\mathbf{2}} + C.\sum_{i=\mathbf{1}}^{l}\left(\xi_i + \xi_i^*\right) - \sum_{i=\mathbf{1}}^{l}\left(\eta_i.\xi_i + \eta_i^*\ .\ \xi_i^*\right) - \sum_{i=1}^{l}\alpha_i(\epsilon + \xi_i - y_i + \langle w,x\rangle + b) - \sum_{i=1}^{l}\alpha_i^*(\epsilon + \xi_i + y_i + \langle w,x\rangle - b)$$

η_i , η_i^* , α_i , α_i^* re dual Lagrange multipliers and they have to satisfy positivity constraints, i.e., η_i , η_i^* , α_i , $\alpha_i^* \geq 0$

By applying the saddle point condition which states that the partial derivatives of L with respect to the primal variables w, b, ξ_i^*,ξ_i^*,($\frac{\partial L}{\partial x}$, $\frac{\partial L}{\partial x}$, $\frac{\partial L}{\partial x}$) have to be nullified for optimality, we obtain:

$$w = \sum_{i=1}^{l} \left(\alpha_i + \alpha_i^*\right) x_i \tag{3.40}$$

and therefore

$$f(x) = \sum_{i=1}^{l} \left(\alpha_i + \alpha_i^*\right) \langle x_i, x \rangle + b \tag{3.41}$$

The latter result is called the Support Vector expansion, i.e.. The weights 'w' can be completely described as a linear combination of the training patterns x_i. Even when evaluating f(x) we need not compute w explicitly [13].

2. **Kernel introduction:**

Now since we reformulated the problem so that the weights (features) are described as a linear combination of the inputs, the obvious idea to adapt the problem to non-linearity is to map the inputs by a non-linear function ϕ, this way 'w' itself becomes a linear combination of non-linear elements, making the whole function f(x) non-linear. The mapped inputs $\phi(x_i)$ constitute a new features space:

$$w = \sum_{i=1}^{l} \left(\alpha_i - \alpha_i^*\right) . \phi(x_i) \tag{3.42}$$

In f(x) (equation 3.37), this translates to the dot product between $\phi(x_i)$ and x. This operation is represented as a kernel function k(xi,x) :

$$f(x) = \sum_{i=1}^{l} \left(\alpha_i - \alpha_i^*\right) . k(x_i, x) + b \tag{3.43}$$

In summary:

As figure 3.4 illustrates, the input samples are mapped into a feature space by a map $\phi(x_i)$ .Then dot products with x are computed. This corresponds to evaluating kernel functions k(xi , x). Finally, the dot products are added up using the weights $\left(\alpha_i - \alpha_i^*\right)$. This, plus the constant term b yields the final prediction output. The process described here is very similar to regression in a neural network, with the difference, that in the SV case the weights in the input layer are a subset of the training patterns.

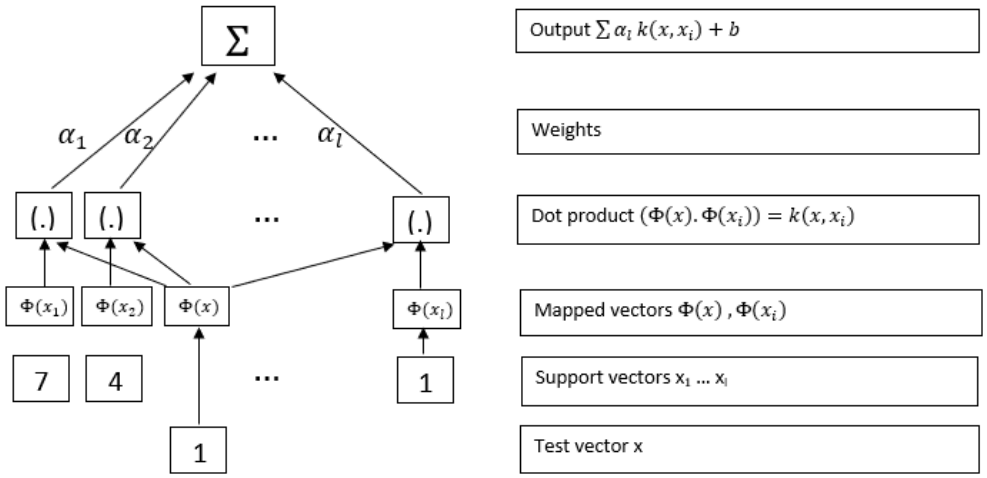

Figure 3.4: Summary of Support Vector Regression

ν-SVM

In regular SVR, it is quite delicate to determine ϵ à priori. A handy solution to this problem came in the form of the ϵ-support vector regression (ϵ-SVR) variant of the algorithm. In ϵ-SVR, ϵ is indirectly determined by the parameter ϵ , which represents the upper limit of the percentage of error points (points outside the ϵ-tube) relatively to all the points [3]. In other words, it decides how many support vectors we want to consider with respect to the entirety of the points. However, unlike ϵ , ν does affect the slack interval and must therefore be considered as a trade-off in the objective function. The formulation of the problem becomes:

$$\begin{aligned} &minimize \ \ \frac{1}{2}||w||^2 + C.(\nu\xi + \frac{1}{l}\sum_{i=1}^{l}(\xi_i + \xi_i^*)) \\ &Subject\ to \begin{cases} y_i - \langle w, x\rangle - b \leq \epsilon + \xi_i \\ \langle w, x\rangle + b - y_i \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases} \end{aligned} \tag{3.44}$$

3.6.2 K-nearest neighbors Regressor

KNN in regression is very similar to its classification counterpart. If $X = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}^T$ is the training data and $y = \begin{bmatrix} y_1 & y_2 & \dots & y_n \end{bmatrix}^T$ are the target outputs, the KNN classifier will compute the distances between the current test point x (for which we want to predict

the output) and every training point $x_i,\ i \in 1,2,..,n$ and sort them :

$$\rho(x,x_1) \leq \rho(x,x_2) \leq \ldots \leq \rho(x,x_n)$$

Next, 'k' training points that have the closest distance to x will be chosen. Among the labels of the k chosen points, the most frequent one is going to be selected as a final prediction. This is called a vote process and it involves finding the mode among the k.

$$\hat{y} = mode(y_1, y_2, \ \ldots, y_k) \tag{3.45}$$

In regression however, this last step is different. Since the target outputs are not specific numbered categories, but are instead distinct real numbers, the notion of voting doesn't hold up. In order to decide on a prediction among the k chosen values, the KNN regressor will compute the mean value of the k labels [6]:

$$\hat{y} = \frac{1}{k}\sum_{1}^{k} y_i \tag{3.46}$$

There is no strict rule that determines the number of neighbors k that gives the best results. In practice it is left as a parameter for the user to tune up in order to improve the predictions. Nonetheless, a few important points must be taken into consideration when setting the value of k.

Choosing a very small k will result in overfitting, whereas making k very big will lead to selecting training points further than and more dissimilar to x, leading to highly erroneous results. This is a case of Bias-Variance Tradeoff:

- if we choose a smaller k (for example k=1), we obtain a more flexible regressor (i.e depending on the test point x, the one very closest neighbor to it will always determine the output), thus the regressor is said to have a high variance and low balance.

- Inversely, if k is very big (k=n for example), regardless of what the test point x is, the output is always going to be the mean value of all the training labels. This is a situation of low variance and high bias [6]. Consequentially, we have to choose a k that is neither too small nor too big, and it is up to the user to find this sweet spot of k through trial and error.

3.7 Conclusion

In this chapter, we have seen how five different models approach the SoC estimation problem in different manners. We also gave a theoretical description of each modelling to showcase the contrast in complexity between them. On top of that, the section provides enough information related to the algorithms for most of them to be implemented with relative ease.

CHAPTER 4

METHODOLOGY AND IMPLEMENTATION

4.1 Introduction

This chapter is divided into 2 sections. The first section details the experimental methodology with which the data was acquired. We explain the components of the experimental set-up and the procedures followed for each battery. The second section details the implementation of each SoC method. The software used and the computer code will be showcased to explain how we've gone from theory to practice and estimated the state of charge.

4.2 Methodology

4.2.1 General Methodology

Using the same photovoltaic array, charge and discharge processes have been performed on both Lithium-ion and lead-acid batteries. Fig. 4.1 shows and overview of the measurement chain to record voltage, current and temperature data during these processes. The Renewable Energy Development Center (CDER) supported these experiments.

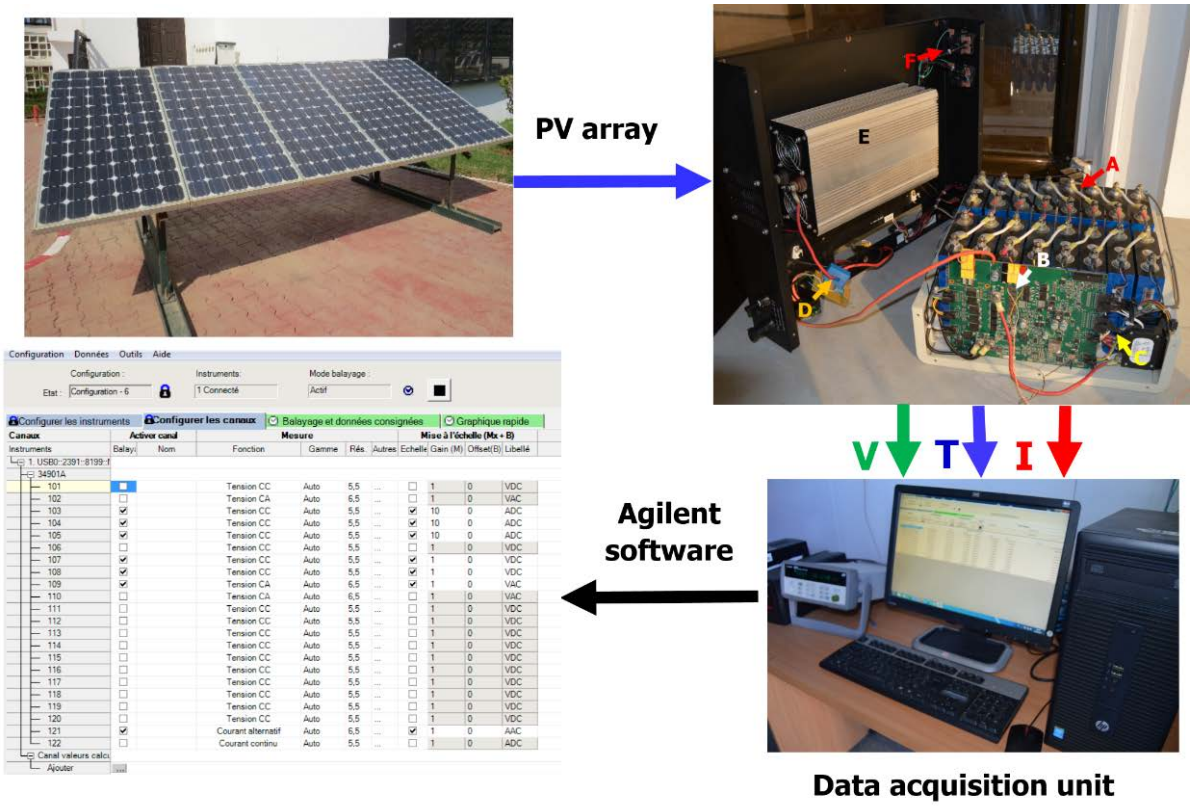

Figure 4.1: Experimental test bench

4.2.2 Lithium-ion Battery

The battery comprises 16 lithium-ion accumulators with an output of 3V DC. The sixteen cells are divided into two eight-cell series batteries connected for an equivalent 24V DC lithium-ion PV battery with a capacity of 1280Wh and a nominal capacity of 53.33 Ah. Parallel to the battery is a battery regulator that controls the charge and discharge. Speaking of, the charge and discharge processes are done separately. The battery is charged ,while disconnected from any load, thanks to a PV array which consists of 5 solar panels connected in series of 150 Wp nominal power each, giving an overall nominal power of 750 Wp. The nominal voltage is of course 24V while the maximum current is 25A. The charging process starts with the battery fully discharged (SoC=0%), and ends when it is fully charged (SoC=100%). During discharge, the battery is disconnected from the PV array and connected in parallel to a DC/AC inverter to guard it. The battery discharges its current into 825W AC lamps. The discharging process starts with the battery fully charged (SoC=100%), and ends when it is fully discharged (SoC=0%).

Data is acquired via voltage and current sensors connected to both the battery inputs

and outputs as well as the photovoltaic generator,inverter and utility. The battery's temperature is measured with a type k thermocouple. All data is recorded in the data logging system with a 1 minute step. Fig. 4.2 shows the synoptic diagram of the measurement chain for the lithium-ion battery.

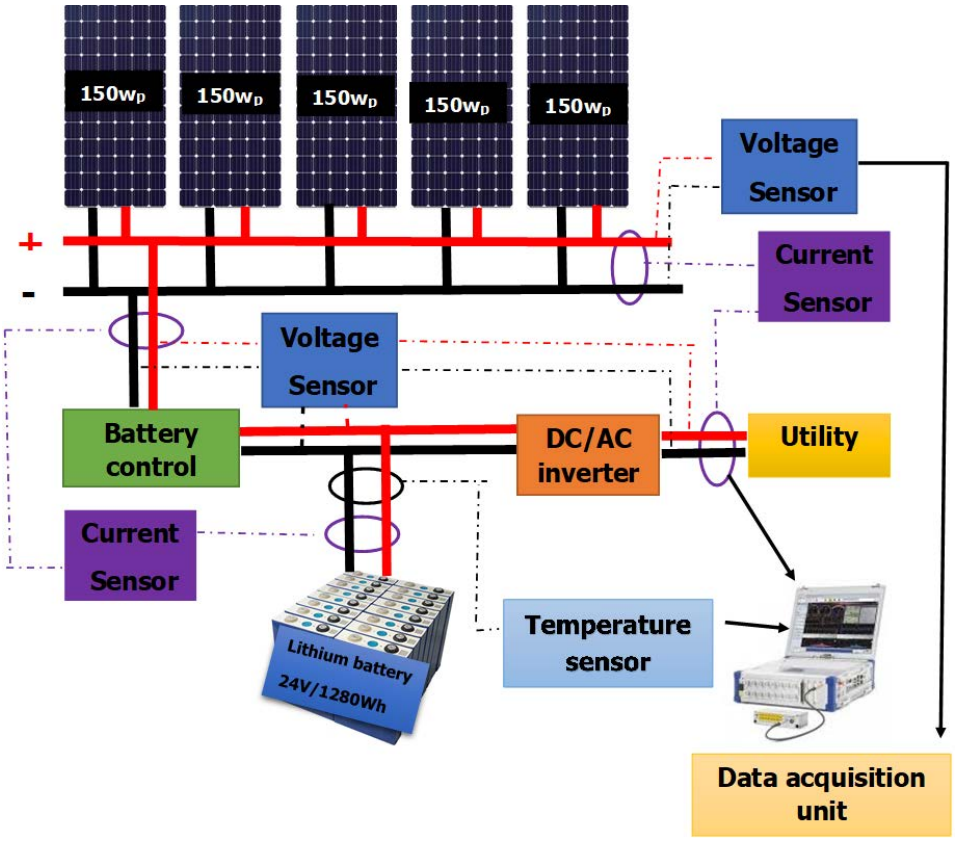

Figure 4.2: Lithium-ion battery experimental test bench schematic diagram.

4.2.3 Lead-Acid Battery

Two 250 Ah/12 VDC sealed gel lead-acid batteries we connected in series to obtain a 24 VDC lead-acid battery. The same 750 Wp PV array supplies this battery, connected through a TS 45A battery charge controller used to protect the device from overcharging and a BG 60A charge controller which prevents depth discharge. The controllers keep the SoC in the 30%-90% range The charging current is supplied by the PV array while the load comprises 20 220 W DC lamps. Charging can only begin when the battery is considered fully discharged i.e. (SoC=30%) and the load is disconnected. Discharging can

only begin when the battery is considered fully charged (SoC above 85%) and the PV array is disconnected. Data is recorded via the same measurement chain for the Lithium-ion battery. Only since lead-acid battery takes longer to charge and discharge, the time step used is five minutes. Fig. 4.3[9] shows the synoptic diagram of the measurement chain for the lead-acid battery.

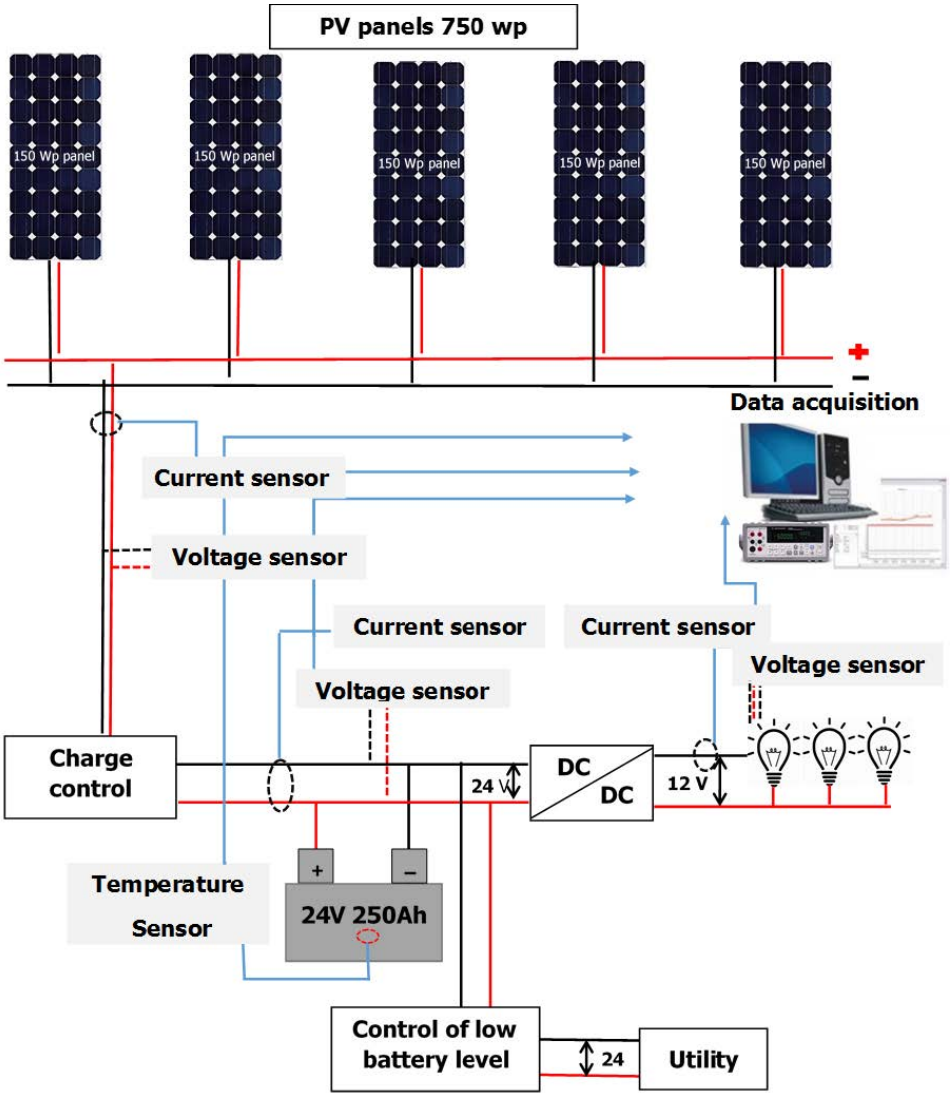

Figure 4.3: Lead-acid battery experimental test bench schematic diagram.

4.3 Implementation

4.3.1 Coulomb Counting and Modified Coulomb Counting

We can program the models simply by implementing the iterative calculus provided by the discrete version of the Coulomb Counting equation (equation 3.2). For every step, we take the SoC of the previous step and subtract from it the product of the present steps' current, sampling time and efficiency divided by the nominal capacity.

Such a calculation is easily implementable on Excel, which works perfectly for us since the data is gathered in an excel file.

Reminder that one of the Coulomb-Counting's weak points is the fact that the SoC needs to be initialized for any charge or discharge. As explained before, the batteries are assumed to be completely discharged at the beginning of the charging phase, and completely charged at the beginning of discharge, therefore the SoC will be initialized to 0% for the first scenario and 100% for the second.

4.3.2 Extended Kalman Filter

The steps of the Extended Kalman Filter algorithm have already been established in the previous chapter; we only need to implement them correctly using a programming language. Any language that can perform basic mathematical computation and matrix calculus can be used (thus C, Matlab, Python, etc are all candidates). In our case, the algorithm was written in python, on a Jupyter platform.

Note that in the C matrix (3.36), the partial derivative $\frac{\partial OCV}{\partial SoC}$ is taken as the slope of the graph $OCV = f(SoC_{ref})$. OCV being the measured voltage in absence of a load, and SoC_ref being the Coulomb-Counting SoC. Figure 4.4 presents the piece of code responsible for executing the Extended Kalman Algorithm.

4.3.3 Support Vector Machines and K-Nearest Neighbors

The optimal tool for using machine learning algorithms is the Sickit-Learn Library. Scikit-Learn is a Python module pre-configured with a multitude of state-of-the-art Machine learning algorithms in the form of a simplified high-level language [28]. The package can be used by non-practitioners, and no intrinsic knowledge of the algorithms' theoretical aspect is needed. Most of them can be called as functions and manipulated in the span of a few lines.

```
Vc0=0
tho=R1*C1

###### 1-Initialization #######
# State variables
X = np.array([[Vc0],[1]])  # State vector: [[V1],[SOC]]
U = np.array([0.0]) # Input ([iL])
# Error and noise convariance matrices
P = np.diag([1,1])
Q = np.diag([1,1])
R = np.array([0.5])
# State-space equation matrices
A = np.array([[0.9,0],[0,1]])
B = np.array([[-0.001,0],[0,0]])
C = np.array([[-1,deriv]])
D = np.array([-R0])
# Initializations for later use
V1 = np.zeros(time.size) # Predicted capacitor voltage
V1[0] = Vc0
soc = np.zeros(time.size)
soc[0] = 100
K = np.array([[0.0],[0.0]])
Error = np.array([0])
i = 1
while i < time.size:
    U[0][0] = battery_current[i-1]
    sampl_time = time[i]-time[i-1]
    A[0][0] = math.exp(-sampl_time/tho)
    B[0][0] = R1*(1 - math.exp(-sampl_time/tho))
    B[1][0] = -sampl_time/(capacity*3600)

##### 2- A priori state estimation #####
    X = np.dot(A,X) + np.dot(B,U)

##### 3- Compute error covariance #####
    P = np.dot(A,np.dot(P,np.transpose(A))) + Q
    V1[i] = ocv[int(X[1][0]*1000)] - X[0][0] + np.dot(D,U)
    Error = battery_voltage[i] - V1[i]

##### 4- Compute Kalman gain #####
    K_denom = float(np.dot(C,np.dot(P,np.transpose(C))) + R)
    K = np.dot(P,np.transpose(C))/K_denom
##### 5- Update/correct state estimate #####
    X = X + Error*K

##### 6- Update/correct error covariance #####
    P = np.dot((np.diag([1,1]))-np.dot(K,C)),P)

    soc[i] = X[1][0]*100
    i = i+1
```

Figure 4.4: Implementation of the Extended Kalman Filter

The Jupyter notebook is a web application that serves as a platform allowing us to call libraries necessary for machine/deep learning and utilize them using python. However, unlike machine learning algorithms, deep learning is quite demanding in terms of resources and GPU. Luckily, the google colab application -which is a hosted Jupyter notebook service- provides us with free access to remote computing resources including GPUs while having a near-identical interface to that of a regular Jupyter notebook. We used Google colab for implementing both machine learning and deep learning models.

In implementing machine/deep learning models, we must first decide on which features we are going to consider. In our case, we want to use the physical measurement recorded through the experiments (i.e. Current, Voltage and Temperature) as learning features. The target variable (or label) is the SoC.

The second most important part is deciding how the data is going to be split for training and testing. Usually, the data is randomly shuffled before being divided into two portions: p% for training and (1-p) % for testing (p > 1-p because more data should go into training the model). However, proceeding this way means that only the test portion is telling of the model's performance, because the training portion has already been seen, and is eventually going to be recognized to perfection since it was previously used for learning. Usually, this is not an issue, but for our application, we want to be able to compare the predictions of these two models (SVM and KNN) with those of the analytical ones (Coulomb-Counting, MCC and Kalman), it is consequentially preferable to have a prediction for every single sample of the data.

To achieve this, we perform a k-fold cross validation fitting for our model. K-fold cross validation is a process in which the model is applied on the data 'k' different times (called 'k' folds), but in each fold the train-test splitting is done differently. Figure 4.5 ([31]) illustrates the concept with a 5-fold cross validation example. In every split, the test fold is different than it is in the other splits, we can exploit that to record predictions for every sample of the data-set by concatenating the test folds of all splits in one vector (think of it as stacking the blue portions in the figure together), this way all the data is swept through without any of it being part of a train set.

The cross_val_predict and the KFold functions provided by the sklearn.model_selection package in sickit learn allows us to do this automatically, combining them produces a vector containing predictions that are part of test sets corresponding to different splits in a 5-fold cross validation.

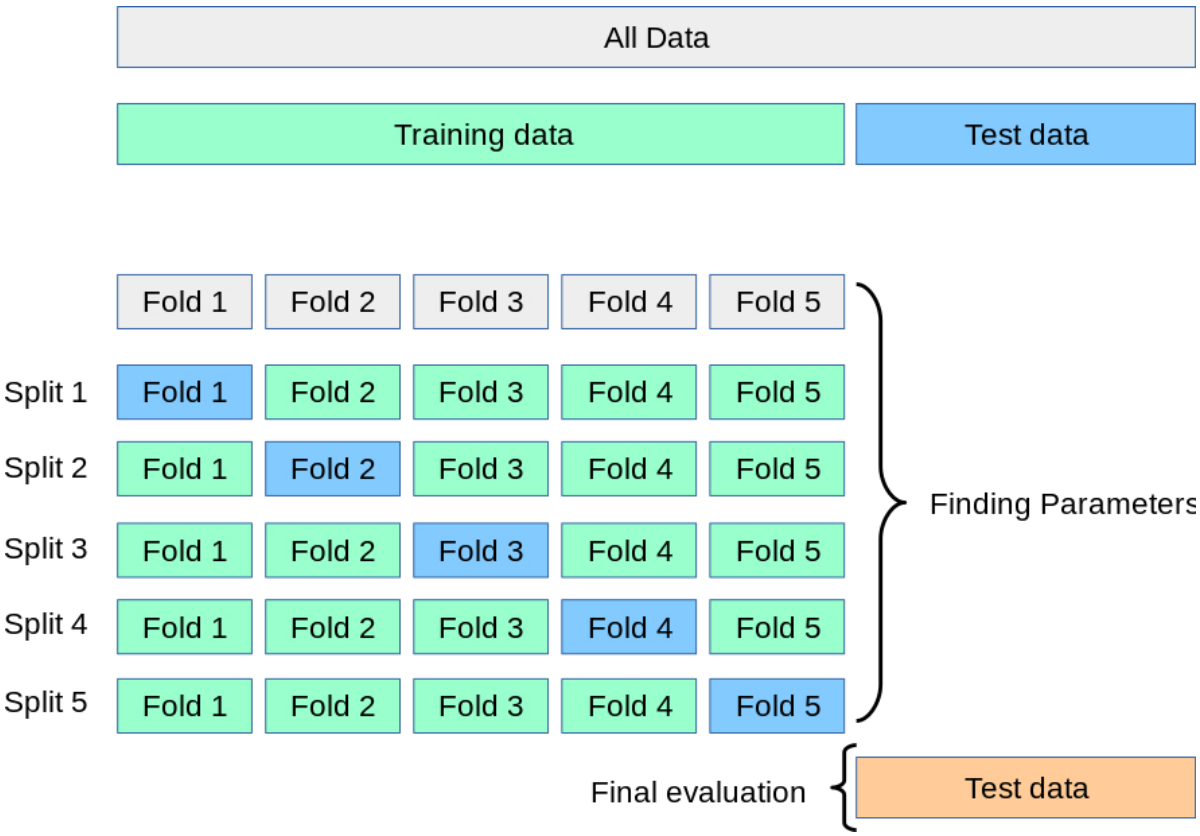

Figure 4.5: Illustration of the cross validation process

We used the ν-SVM variant of the SVM algorithm with an RBF kernel, a value of 1.0 for nu, 0.125 for gamma and 1.0 for C. Figure 4.6a shows the piece of code corresponding to the SVM implementation. For KNN, we used the default parameters, fig 4.6b shows the code corresponding to its implementation.

4.3.4 Neural Networks

Training Feed-Forward Neural networks necessitates another type of library, in our case we used Pytorch. PyTorch is an optimized tensor library for deep learning using GPUs and CPUs (https://pytorch.org/docs/stable/index.html). The procedure of implementing elementary Neural networks is usually always the same: We first define the network (by setting up the number of hidden layers, the number of inputs/outputs and the activation functions), select a loss metric and an optimizer (for back-propagation), fit the model on the training data while improving upon the errors and finally fitting the trained model on the test data to get the predictions ad evaluate them.

We use the same approach while also applying k-fold cross validation on Neural

```
from sklearn.svm import NuSVR
from sklearn.model_selection import cross_val_predict
from sklearn.model_selection import KFold

cv=KFold(n_splits=5, shuffle=True)
rgs=NuSVR(nu=1.0, C=1.0, kernel='rbf',gamma=0.125)

cv_pred=cross_val_predict(rgs,
                X_d,
                y_d,
                cv=cv)
```

(a) Nu-SVR regressor

```
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import cross_val_predict
from sklearn.model_selection import KFold

cv=KFold(n_splits=5, shuffle=True)
rgs=KNeighborsRegressor()
cv_pred=cross_val_score(rgs,
                X_d,
                y_d,
                cv=5)
```

(b) KNN regressor

Figure 4.6: Implementation of machine learning algorithms

Networks which is less simple than it is on KNN and SVM. This time we have to employ the 'KFold' function to produce 'k' different splits (configurations), in each one, the data is shuffled and then split in a different manner. KFold takes record of original order of the data (before shuffling) by creating arrays containing the original indices, these will be necessary to reset the predictions to their original order later on.

Fig.4.7 shows the feed-forward network set up. Fig.4.8 shows the preamble in which we set the error-criterion, the optimizer and initialized some variables. The learning phase is showcased in Fig.4.9.

Finally, the test data from all folds are gathered in one vector and then put back into

```
import torch.nn.functional as F
class Regressor(nn.Module):
    def __init__(self):
        super().__init__()
        self.fc1 = nn.Linear(3, 9)
        self.fc2 = nn.Linear(9, 9)
        self.fc3 = nn.Linear(9, 1)

        # Dropout module with 0.2 drop probability
        self.dropout = nn.Dropout(p=0.2)

    def forward(self, x):
        # make sure input tensor is flattened
        x = x.view(x.shape[0], -1)

        # Now with dropout
        x = self.dropout(torch.tanh(self.fc1(x)))
        x = self.dropout(torch.tanh(self.fc2(x)))
        # output so no dropout here
        x = torch.sigmoid(self.fc3(x))

        return x
```

Figure 4.7: Creation of the neural network structure

```
model = Regressor()
criterion = nn.MSELoss()
optimizer = optim.SGD(model.parameters(), lr=0.002)

#Number of folds
k= 5
#Nuumber of Epochs
epochs = 100
#Initializations for later use
kfold_pred, test_ids = np.array([]), np.array([]), np.array([])
#
```

Figure 4.8: Selection of the optimizer and the error criterion

```
# The learning process strts here
for e in range(epochs):
    running_loss = 0
    train_target,train_preds=[], []
    test_target,test_preds=[], []
    train_time=[]
    loss=0
    for feature, target,index in trainloader:
        target=target.view(-1,1)
        optimizer.zero_grad()
        preds = model(feature.float())
        loss=criterion(preds, target.float())
        loss.backward()
        optimizer.step()
        train_target.append(target.item())
        train_preds.append(preds.item())
        train_time.append(index.item())

    else:
        test_loss = 0
        test_time=[]
        with torch.no_grad():
            for i,data in enumerate(testloader,0):
                feature, target,index=data
                target=target.view(-1,1)
                optimizer.zero_grad()
                preds = model(feature.float())
                test_target.append(target.item())
                test_preds.append(preds.item())
                test_time.append(index.item())
    #
```

Figure 4.9: Fitting the model on the training data and testing on the test-set

their original order using the indices array and a custom function we created, this is illustrated in figure 4.10

```
#For every fold, we record the predicted data, as well as their original indexes
kfold_pred=np.append(kfold_pred,test_preds)
test_ids=np.append(test_ids,test_id)

# We use a function we created to sort the predictions in the same order they were in before the shuffling
kfold_pred=sort(kfold_pred, test_ids)
```

Figure 4.10: Merging and saving the test predictions in addition to their respective indices

4.4 Conclusion

In this chapter, we reviewed the methodology that was followed to achieve a desirable data acquisition and processing for our project. First, we showcased the experimental test-bench used to perform and record the different measurements for both battery technologies, highlighting the differences and similarities between the procedures. Following that, we showcased the software side of things i.e. the implementation of the aforementioned SoC estimation models. For each model we explained the logic behind the coding and provided some snippets to support that.

CHAPTER 5

RESULTS AND DISCUSSION

5.1 Introduction

In this chapter, we will finally showcase the obtained results from each of the SoC estimation methods used and analyse them. Past this introduction, this chapter is divided into three sections: analysis for lithium-ion battery SoC estimation, analysis for Lead-acid battery SoC estimation and a conclusion. Each analysis is performed on the charge and discharge phases separately. Our analysis is thorough . One is a quantitative analysis which consists of calculating the cumulative frequencies for each estimated SoC and comparing them to the reference SoC, it gives a distribution of "closeness' to the ideal SoC. The other is a qualitative analysis. It includes calculating the RMSE (Root Mean Square Erreur) and the MBE (Mean Bias Error) which indicate the overall accuracy of the model. Using the RMSE, correlation and the standard deviation of each estimation we draw a Taylor diagram to further illustrate the degree of correspondence between the models and the reference SoC. After a thorough analysis and informed interpretation of our indicators we come to a decision as to which SoC is most suited for the battery. We proceed to validate the model on standalone PV cycles. These cycles include both charge and discharge phases continuously. One where the consumption remains constant, and the other where the consumption evolves inversely to the solar irradiance. This validation tells us whether the model is viable in a real future potential application. We repeat this process for the other battery technology and conclude the chapter.

5.2 Lithium Battery

5.2.1 Charge

The following results concern the charging mode of the lithium-ion battery. Notice that the charging period of the battery takes about 14 hours. Over the first 9 hours of charging, corresponding to the first charging phase, the voltage gradually increases with irradiation. During that time, the battery temperature grows with the charging current. After 14 hours, the charging time maintains the battery voltage relatively constant by reducing the charging current to its minimum value of about 300 mA.
The battery charging profile corresponding to the lithium battery has conceded over two phases: constant current and constant voltage. The battery charge controller integrated through the storage unit case study displays the battery charging profile. As shown in Fig.5.1.

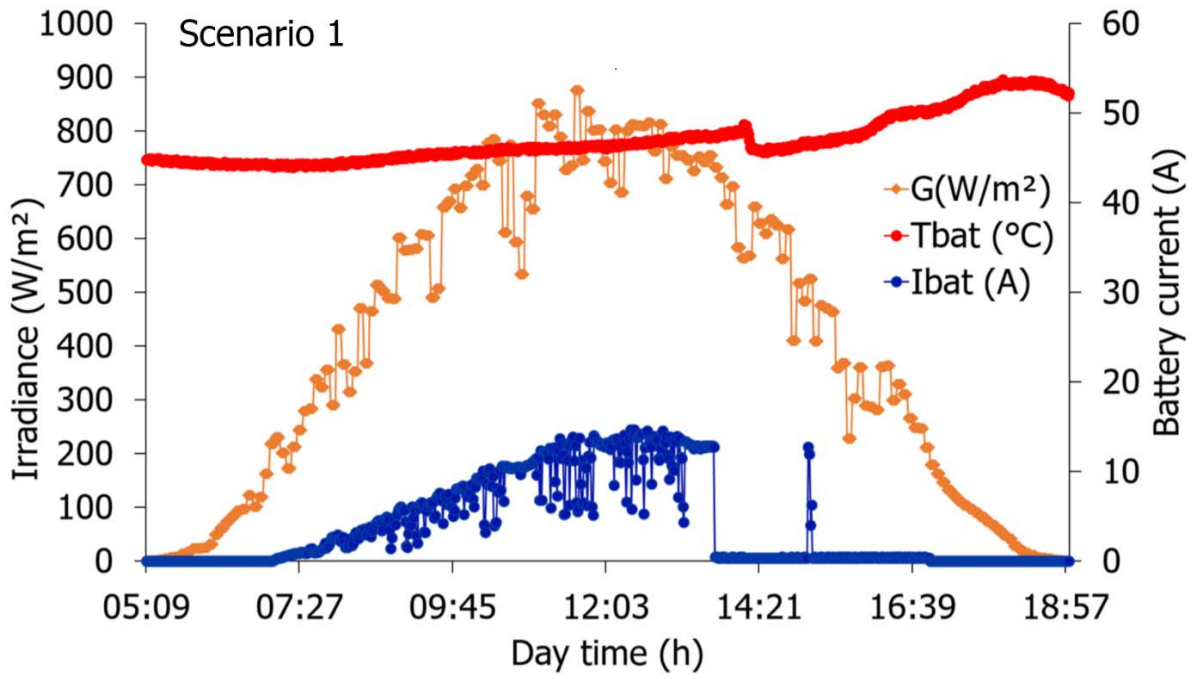

Figure 5.1: Lithium battery charging profile.

The following Fig.5.2 depicts the evolution of the five battery SoC estimation methods considering the charging mode of the lithium battery. It is observed that most of the models have identical convergence, except for the EKF model, which is considered far from being taken into account during this phase.

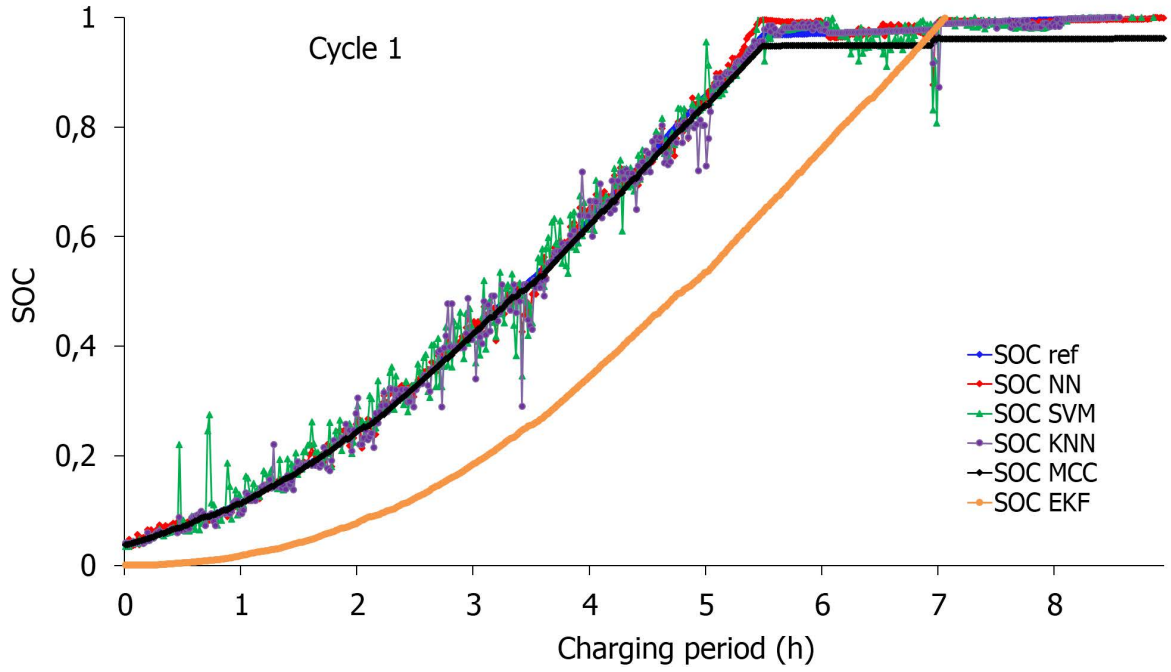

Figure 5.2: SoC modelling for lithium battery during charge mode.

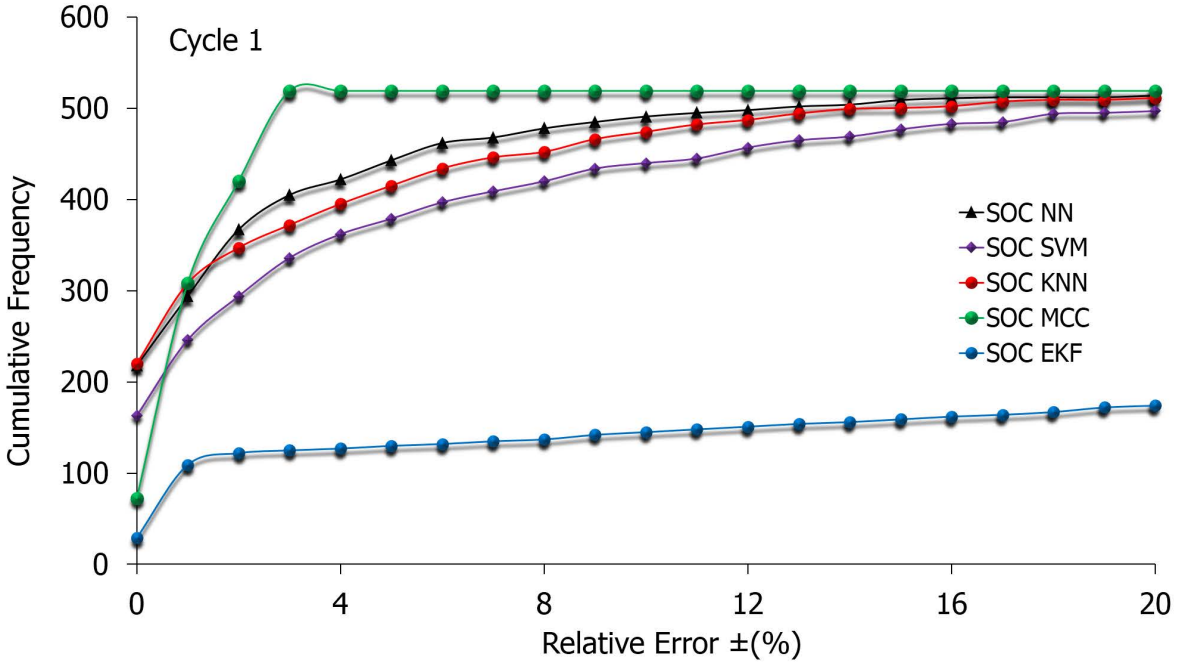

Figure 5.3: Cumulative Frequency Vs relative error for lithium battery during charge mode..

Table 5.1 presents numerical results of these cumulative frequencies achieved for RE (%) within a range of 0 to ± 27%; during the charging mode period.

RE (± %)	0	3	6	9	12	15	18	21	24	27
SoC_{NN}	41.85	78.03	88.70	93.12	95.61	97.72	98.30	98.68	100	100
SoC_{SVM}	31.29	64.51	67.22	83.32	87.74	91.58	94.84	96	97.53	100
SoC_{KNN}	42.24	71.42	83.32	89.47	93.50	96	97.72	98.68	99.07	100
SoC_{MCC}	13.82	100	100	100	100	100	100	100	100	100
SoC_{EKF}	5.56	24	25.34	27.26	28.99	30.52	32.06	33.98	79.86	100

Table 5.1: Comparative results of cumulative frequency distribution Vs. The relative error for charge mode of lithium battery during cycle 1

It is noted that around 42% of the SoC_{KNN} has a RE of ±0%, whereas the SoC_{MCC} has achieved all its data with RE less or equal to 3%, followed by the Neural network model SoC, which acquired all its data with RE equal to 24%. Finally, 100% of the data are recorded for the Kalman, SVM and KNN models, respectively, for 27% of error.

To enhance a practical selection for the lithium battery model operating SoC model in the PV system. another analysis is provided. Table 5.2 offers further comparisons between the five SoC models considering four cycles of charging mode based on statistical errors.

	Cycle 1		Cycle 2		Cycle 3		Cycle 4	
	MBE	RMSE	MBE	RMSE	MBE	RMSE	MBE	RMSE
SoC_{NN}	$-6.28{\cdot}10^{-6}$	0.00013	$2.07{\cdot}10^{-5}$	0.00046	$1.68{\cdot}10^{-5}$	0.00019	$6.38{\cdot}10^{-6}$	$5.45{\cdot}10^{-5}$
SoC_{SVM}	$-5.88{\cdot}10^{-6}$	0.00013	$3.25{\cdot}10^{-6}$	0.000072	$-1.46{\cdot}10^{-6}$	$1.72{\cdot}10^{-5}$	$4.4\cdot10^{-5}$	$3.84\cdot10^{-5}$
SoC_{KNN}	$-8.64{\cdot}10^{-6}$	0.00019	$5.97{\cdot}10^{-8}$	0.000001	0	0	0	0
SoC_{MCC}	$-3.74{\cdot}10^{-7}$	$8.54{\cdot}10^{-6}$	$-1.8{\cdot}10^{-6}$	0.00004	-0.00011	0.0013	-0.00017	0.0014
SoC_{EKF}	$7.01\cdot10^{-5}$	0.0015	$6.59\cdot10^{-5}$	030014	$1.68\ 10^{-5}$	0.00019	0.0015	0.013

Table 5.2: Statistical errors of lithium-ion battery during four cycles of charge mode

From table 5.2, the battery SoC_{KNN} displays very low values of MBE and RMSE during the four charging cycles compared with the remaining models to achieving $-8.64.10^{-6}$ of MBE and -0.00019 of RMSE during the first cycle. Afterwards, during the second cycle, the battery SoC_{KNN} reached $5.97\cdot10^{-8}$ of MBE and -0.000001 of RMSE. In addition, zero MBE and RMSE were realized during cycle 3 and cycle 4.

Fig.5.4 depicts the Taylor diagram standard error for lithium battery during cycle 1 charge mode. We observe that SVM, KNN and NN have more or less the same standard deviation as that of the reference (about 0.28) with varying correlations and RSME's; NN has the highest correlation (~0.996) and the lowest RMSE (~0.025), followed by KNN with a lesser correlation and higher RMSE (0.992 and 0.04 respectively) and SVM (0.99 RMSE and 0.05 correlation). On the other hand, Kalman and MCC have a lower standard deviation than the latter three (about 0.275) but a very high correlation nearing the value of 1 and an RMSE close to zero.

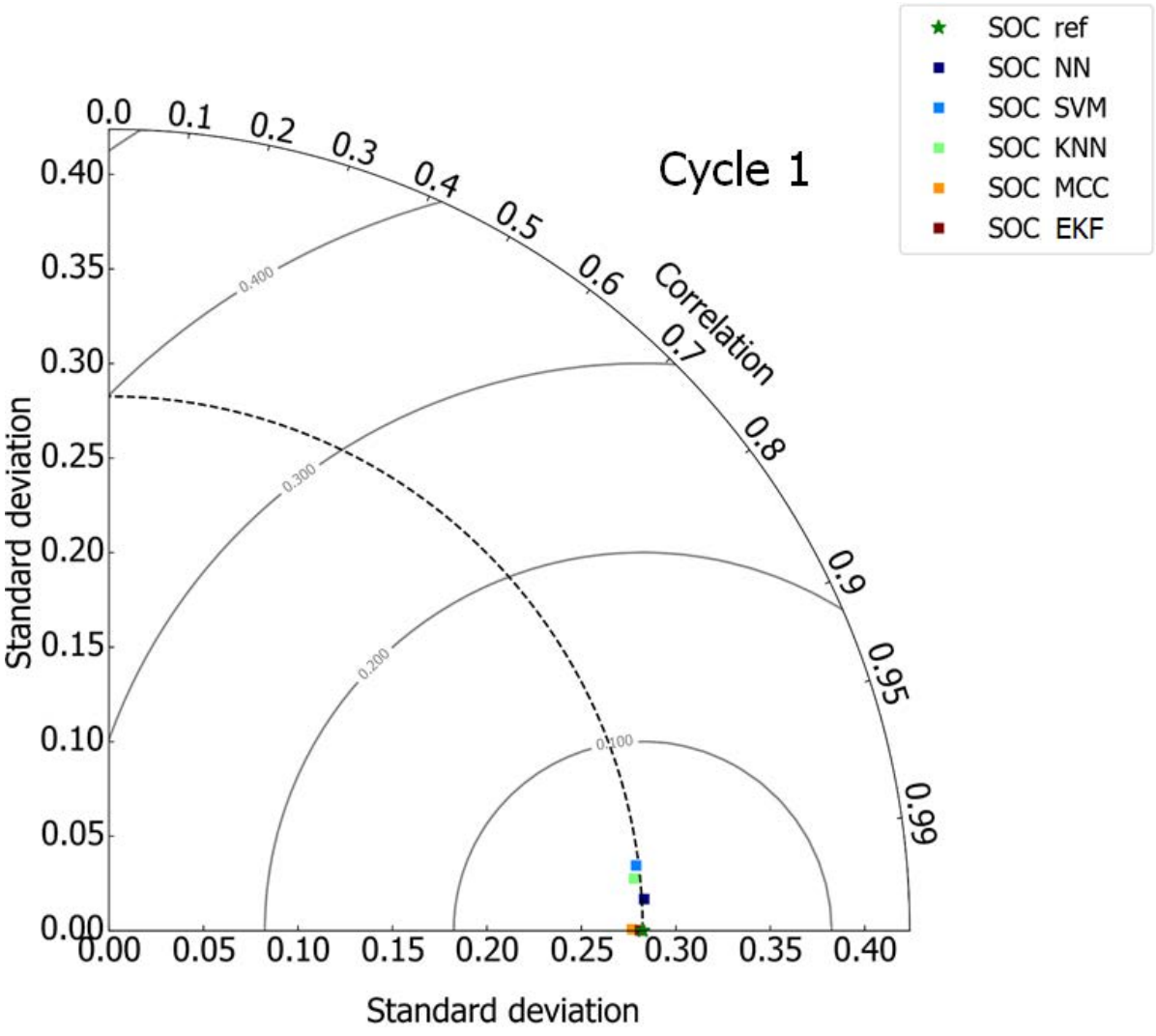

Figure 5.4: Taylor diagram standard error for lithium battery during charge mode..

The previously obtained results suggest that the KNN model is the most suitable for the lithium charging mode. To better highlight the battery SoC_{KNN} model, another quantitative analysis is provided for the lithium charging mode, considering four battery cycles. As shown in Fig.5.5, the maximum data is within zero error for the four charging cycles.

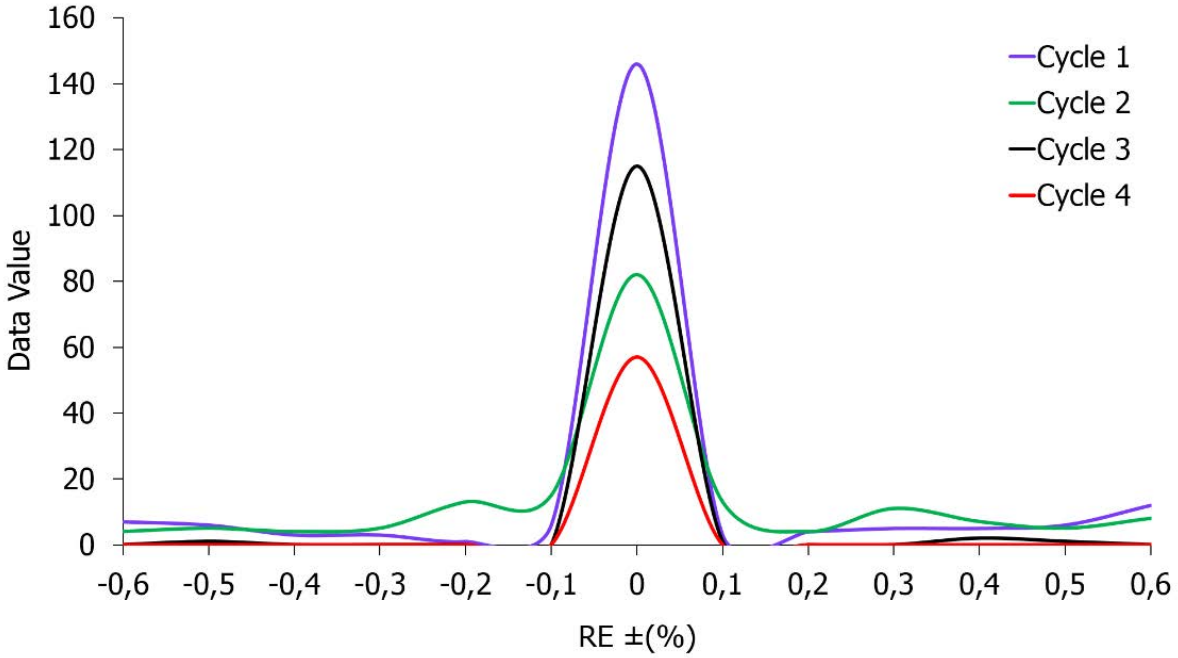

Figure 5.5: Data quantity Vs relative error based on SoC_{KNN} for lithium battery during 04 cycles of charge mode

5.2.2 Discharge

Fig.5.6 shows a comparison of five SoC models during cycle 1 of the discharge mode. All the simulated models present a good similarity to SoC_{ref} except for the SoC_{SVM} model since it displays some fluctuations.

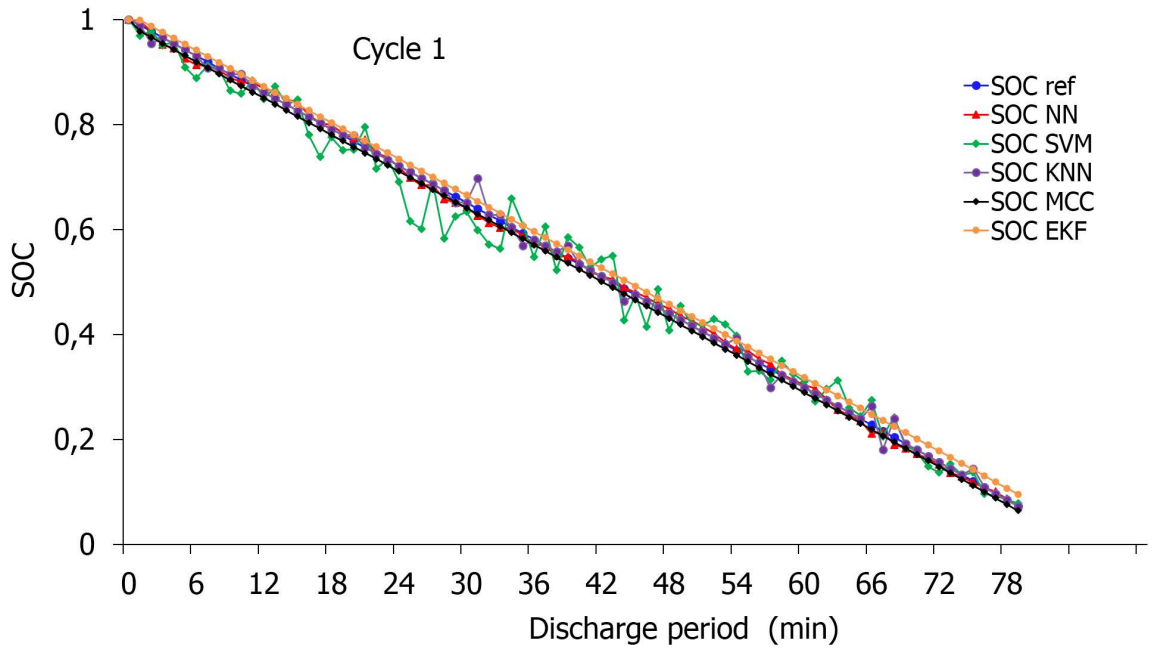

Figure 5.6: SoC modelling for lithium battery during discharge mode

As depicted in Fig. 5.6 the battery was considered as initially fully charged and not connected to the PV array; the battery provides only an external utility of 825 W to ensure the discharge process. The SoC during this mode is a linear function, which decreases when the battery supplies the utility over time.

Fig.5.7 provides the cumulative frequency via the relative error percentage RE (%) during discharge mode for lithium battery SoC evolution. Table 5.3 presents numerical results of these cumulative frequencies achieved for RE (%) within a range of 0 to ±20%; during the first cycle of discharging mode.

RE (± %)	0	2	4	6	8	10	12	14	16	18	20
SoC_{NN}	42.5	90	96.25	98.75	100	100	100	100	100	100	100
SoC_{SVM}	15	33.75	52.5	66.25	73.75	82.5	91.25	96.25	96.25	100	100
SoC_{KNN}	80	87.5	91.25	92.5	92.5	95	95	95	97.5	98.75	100
SoC_{MCC}	1.25	75	88.75	93.75	97.5	97.5	97.5	100	100	100	100
SoC_{EKF}	1.25	52.25	68.75	77.5	83.75	87.5	90	92.5	93.75	95	100

Table 5.3: Comparative results of cumulative frequency distribution Vs. The relative error for discharge mode of lithium battery during cycle 1.

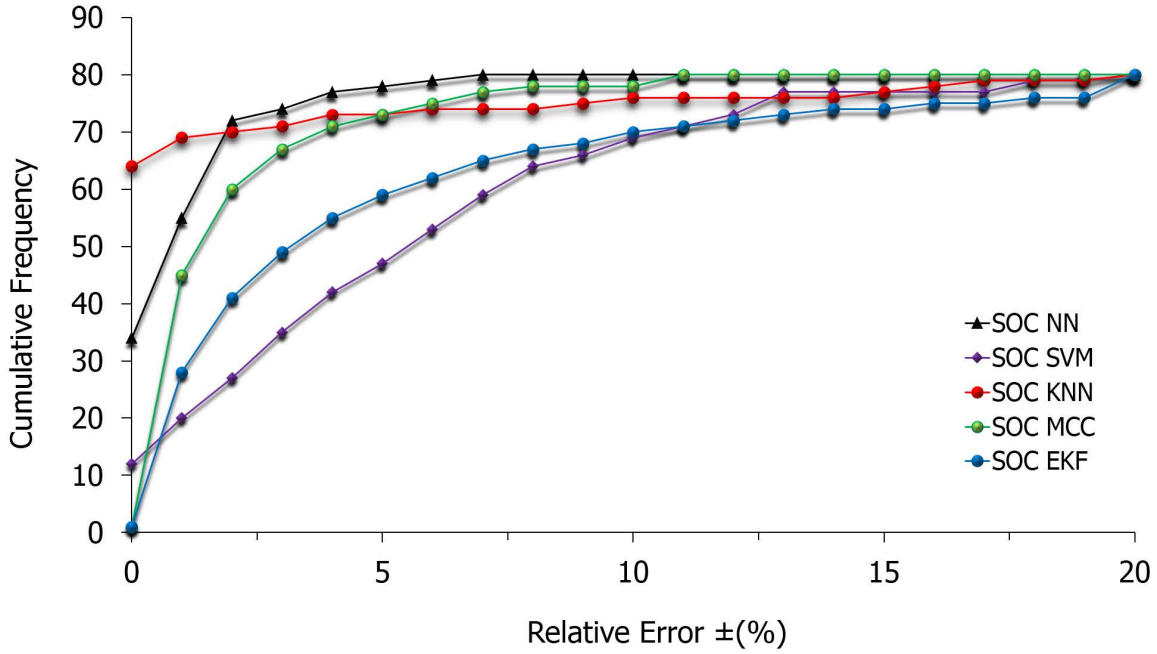

Figure 5.7: Cumulative Frequency Vs relative error for lithium battery during discharge mode.

It is noted that around 80% of the SoC_{KNN} has a RE of ±0%, whereas the SoC_{MCC} and SoC_{EKF} have only 1.25 % of data for the same error followed by SoC_{NN} and SoC_{SVM} respectively, SoC_{NN} achieved all its data with RE less or equal to 8%, followed by the SoC_{SVM} and SoC_{MCC}, which acquired all its data with RE equal to 18%. Finally, 100% of the data are recorded for the Kalman, and KNN models, respectively, for 27% of error.

Fig.5.8 depicts Taylor diagram standard error for lithium battery during discharge mode related to cycle 1. We observe that NN and KNN have the same standard deviation as the reference (0.275) and a very low RMSE of 0.01 (High correlation). Kalman and MCC are the opposite, they present a slightly lower standard deviation but their correlation to the reference is higher (nearing the value of 1 with an RMSE close to 0). SVM on the other hand has a much lower correlation to the reference (about 0.99 with a RMSE of about 0.045) compared to the four others as well as a lower standard deviation of 0.25.

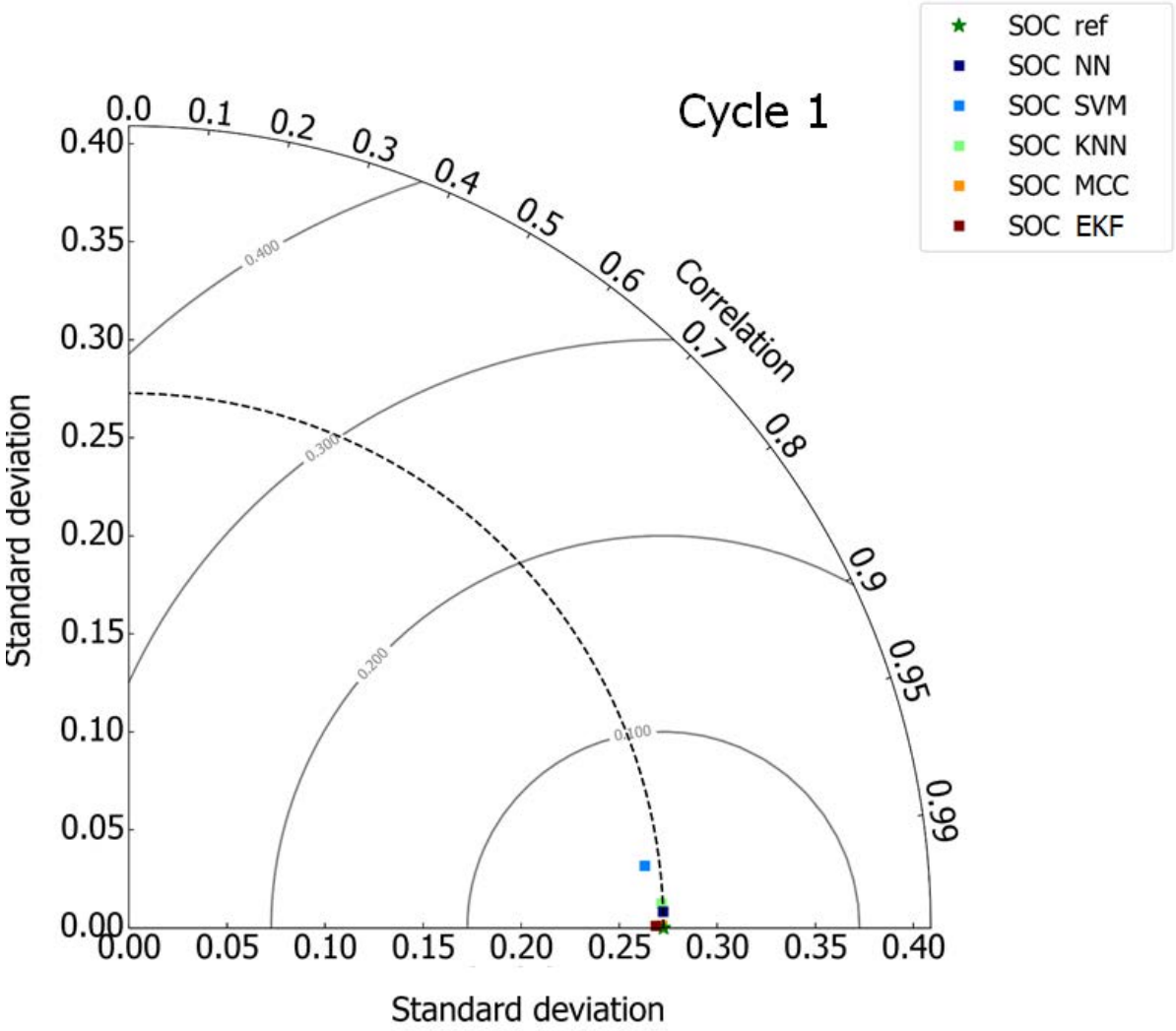

Figure 5.8: Taylor diagram standard error for lithium battery during discharge mode.

To make a proper choice for the lithium battery operating SoC model in the PV system another statistical analysis is provided based on MBE and RMSE errors as shown in Table 5.4, describing further comparisons between the five SoC models considering four cycles of discharging mode.

	Cycle 1		Cycle 2		Cycle 3		Cycle 4	
	MBE	RMSE	MBE	RMSE	MBE	RMSE	MBE	RMSE
SoC_{NN}	$1.8{\cdot}10^{-8}$	$1.61{\cdot}10^{-7}$	$-1.58{\cdot}10^{-6}$	$1.3{\cdot}10^{-5}$	$6.59{\cdot}10^{-6}$	0.00012	$-1.42{\cdot}10^{-6}$	$3.65{\cdot}10^{-5}$
SoC_{SVM}	$1.3{\cdot}10^{-7}$	$1.16{\cdot}10^{-6}$	$1.35{\cdot}10^{-7}$	$1.1{\cdot}10^{-6}$	$2.35{\cdot}10^{-7}$	$4.54{\cdot}10^{-6}$	$-5.81{\cdot}10^{-7}$	$1.49{\cdot}10^{-5}$
SoC_{KNN}	0	0	$29{\cdot}10^{-5}$	0.0026	$6.59{\cdot}10^{-6}$	0.00012	0	0
SoC_{MCC}	$1.46{\cdot}10^{-6}$	$1.3{\cdot}10^{-5}$	$1.52{\cdot}10^{-6}$	$1.3{\cdot}10^{-5}$	$6.47{\cdot}10^{-6}$	0.00012	$5.62{\cdot}10^{-7}$	$1.44{\cdot}10^{-5}$
SoC_{EKF}	0	0	0	0	0	0	0	0

Table 5.4: Statistical errors of lithium-ion battery during four cycles of discharge mode.

Based on table 5.4, the battery SoC_{EKF} displays very low values of MBE and RMSE during all four cycles compared with the remaining models to achieving 0 error of MBE and RMSE, respectively. Afterwards, the battery SoC_{KNN} reached 0 error of MBE and RMSE. During cycle two and cycle four, respectively.

The previously obtained results suggest that the Kalman model is more suited for lithium-ion discharging mode, followed closely by the KNN model. However, considering the charging mode as well the KNN model is more suited for lithium-ion SoC estimation overall.

To better highlight the battery SoC_{KNN} model, another quantitative analysis is provided for the lithium discharging mode, considering four battery cycles. As shown in Fig. 5.9, the maximum data is within zero error for the four charging cycles.

Based on the aforementioned analysis, the battery SoC_{KNN} model is more suited for lithium battery manufacturers achieving less than $2.6{\cdot}^{10-3}(\%)$ of MBE and RMSE errors.

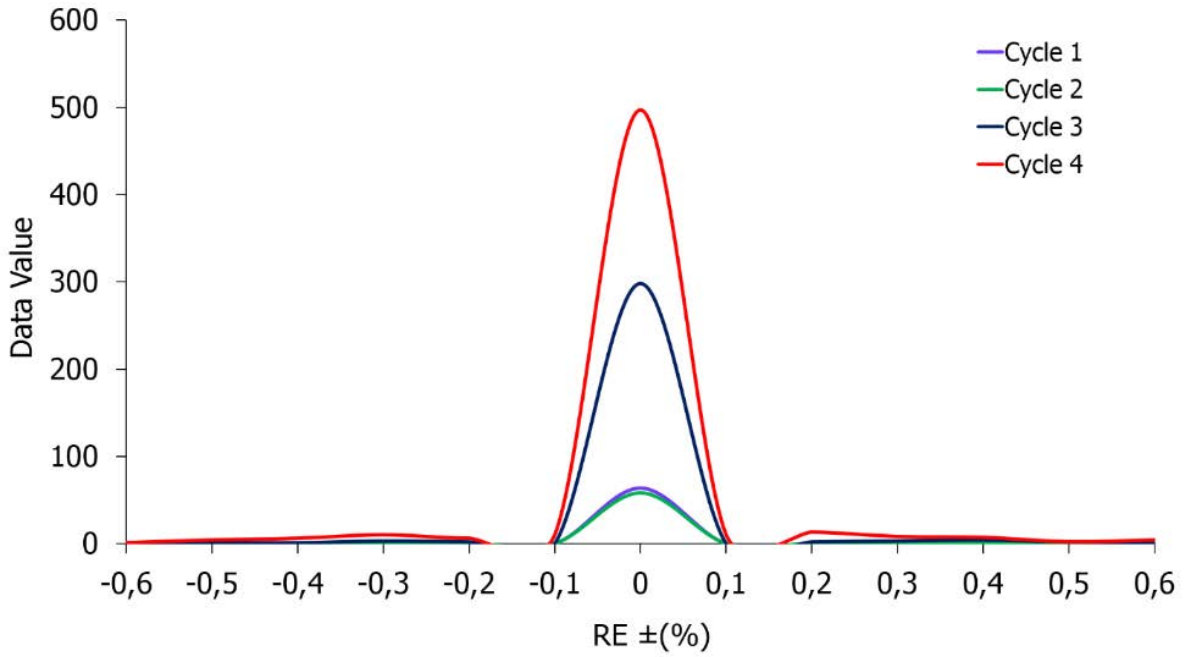

Figure 5.9: Data quantity Vs relative error based on SoC_{KNN} for lithium battery during 04 cycles of charge mode.

5.2.3 Validation

This section objective consists of validating the updated SoC_{KNN} model in a real operating stand-alone PV system. The system consists of a PV array with a nominal power of 750 Wp, a lithium-ion battery incorporated through the off-grid system with DC/AC inverter and AC load. The model validation involved two different scenarios: the first one depicts the system behaviour over two days in Fig. 5.10. We started the simulation with a completely discharged battery, whereas for the second scenario, the model was tested over three working days with a fully charged battery. Fig. 5.12 displays two typical days of operation of the PV system under study, during which the PV current produced from the daily solar irradiation I_{pv}, the charging battery current I_{bat}, and the load consumption I_{load} are reported, Fig.5.11 and Fig.5.13 show the battery SoC_{KNN} evolution compared to the SoC_{ref}. A good fit between both estimation is shown during charge and discharge modes.

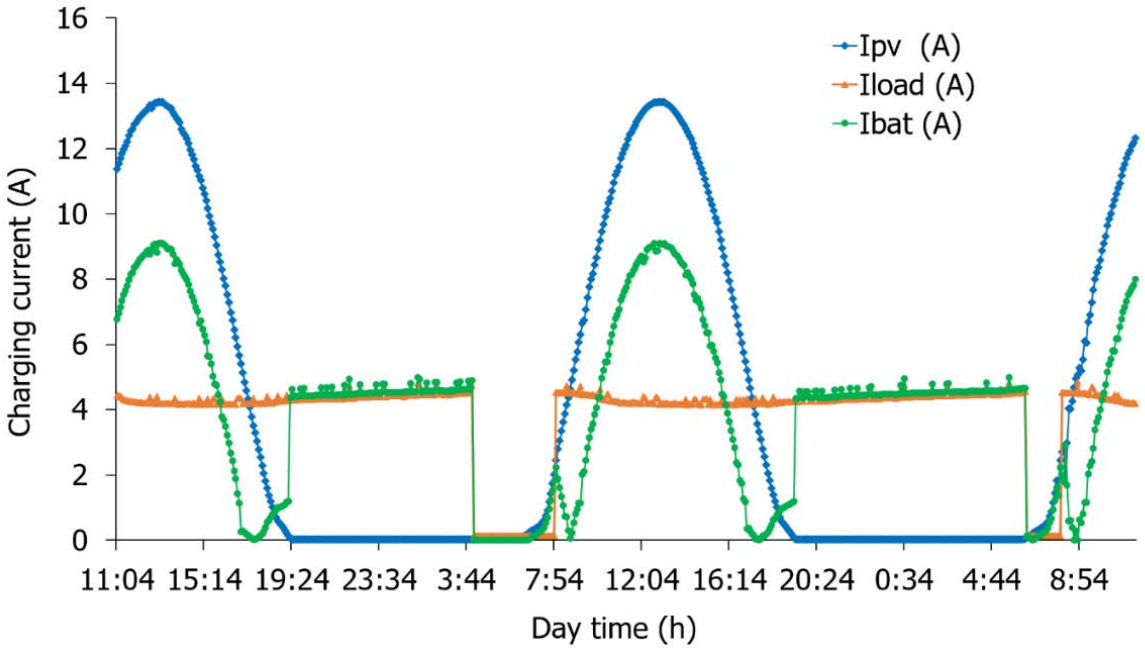

Figure 5.10: Battery current profile evolution with PV and load current in standealone PV system for scenario 1.

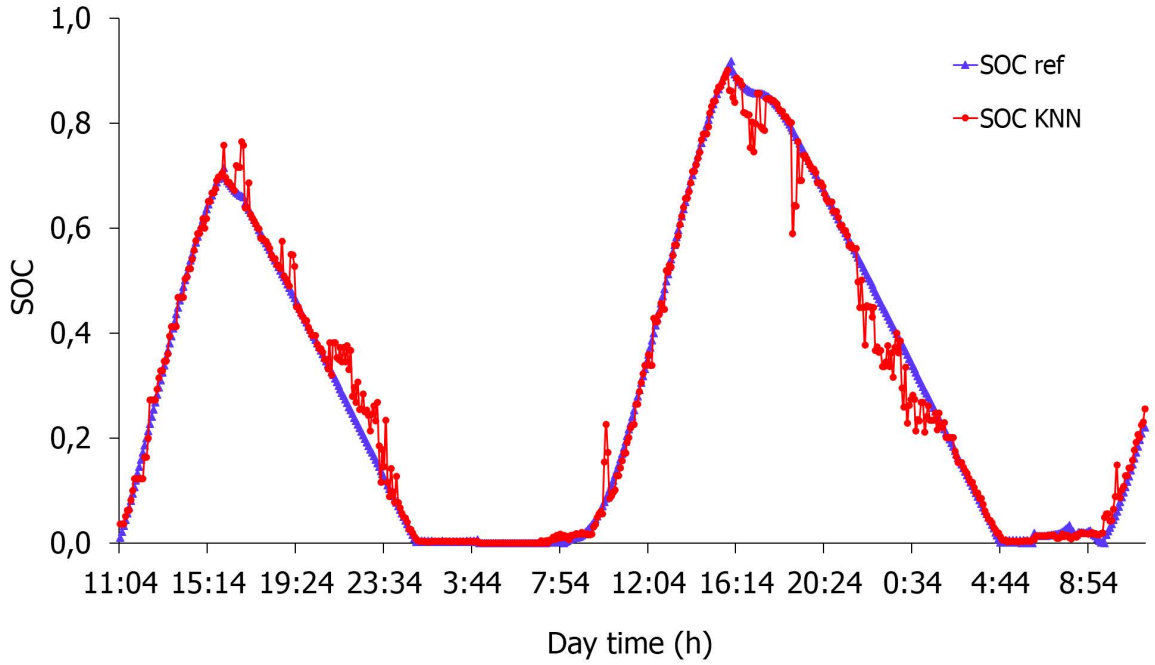

Figure 5.11: Simulation of battery SoC for scenario 1.

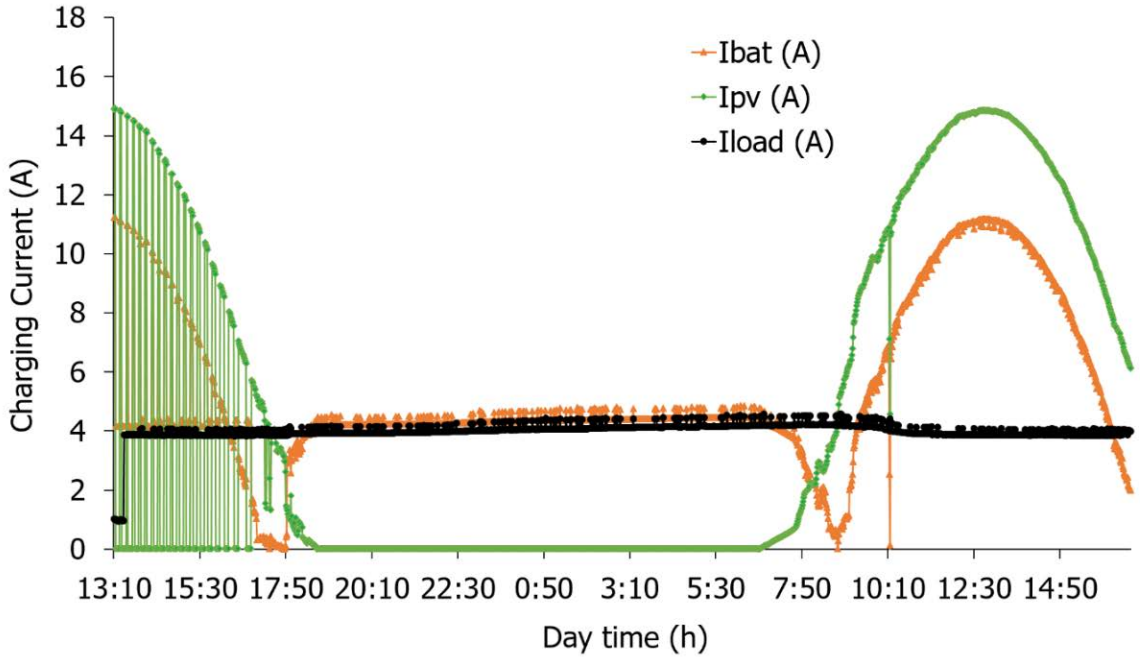

Figure 5.12: Battery current profile evolution with PV and load current in standalone PV system for scenario 2.

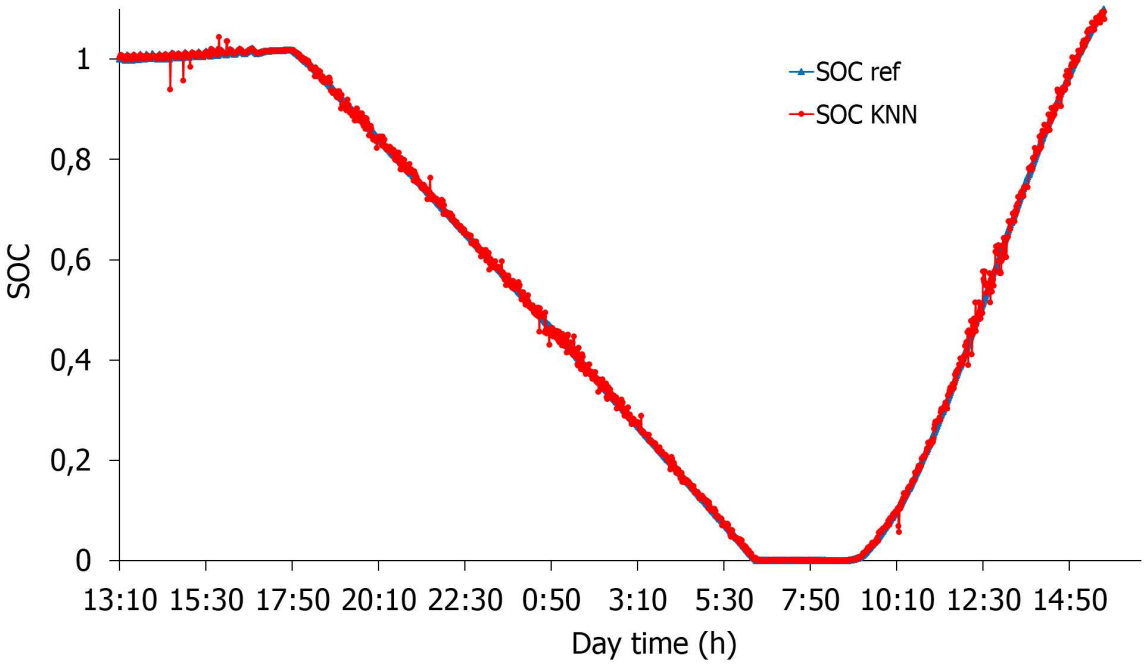

Figure 5.13: Simulation of battery SoC for scenario 2.

5.3 Lead-acid Batteries

5.3.1 Charge

The charging period of the battery spans four days. We notice that during the first seven hours of charging, the voltage increases gradually with irradiation, this is called the boosting phase. Note also that the battery temperature increases with the charging current. After four days, the latter hits its minimum value of zero after which the battery maintains a relatively constant voltage signaling the beginning of the floating phase.

The battery charging profile corresponding to the lead-acid battery was conducted in two phases: Boost and Floating. The battery charge controller integrated through the storage unit case study displays the battery charging profile. As shown in Fig. 5.14.

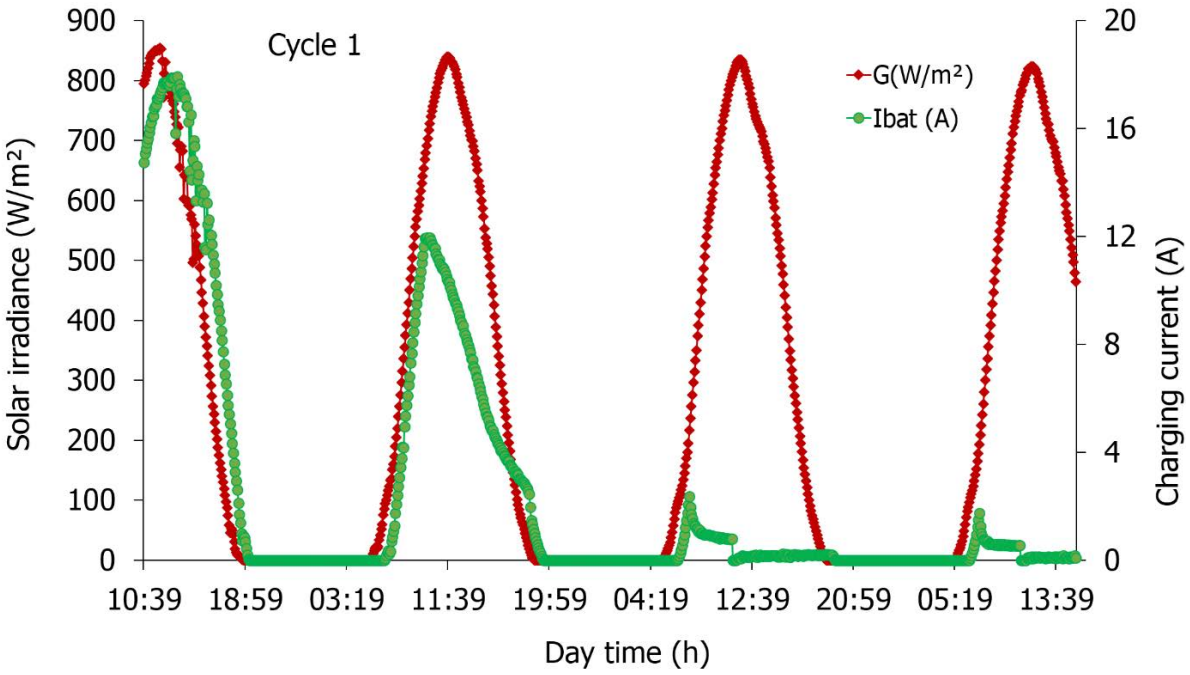

Figure 5.14: Lead-acid battery charging profile.

Fig. 5.15 shows the evolution of the four battery SoC models considering the charging mode of the lead-acid battery. It is observed that most of the models have close convergences to the reference.

Fig.5.16 provides the cumulative frequency via the relative error percentage RE (%) during charge mode for lead-acid battery SoC. Table 5.5 presents the cumulative frequencies achieved for RE within a range of 0 to ± 8%; during the charging mode.

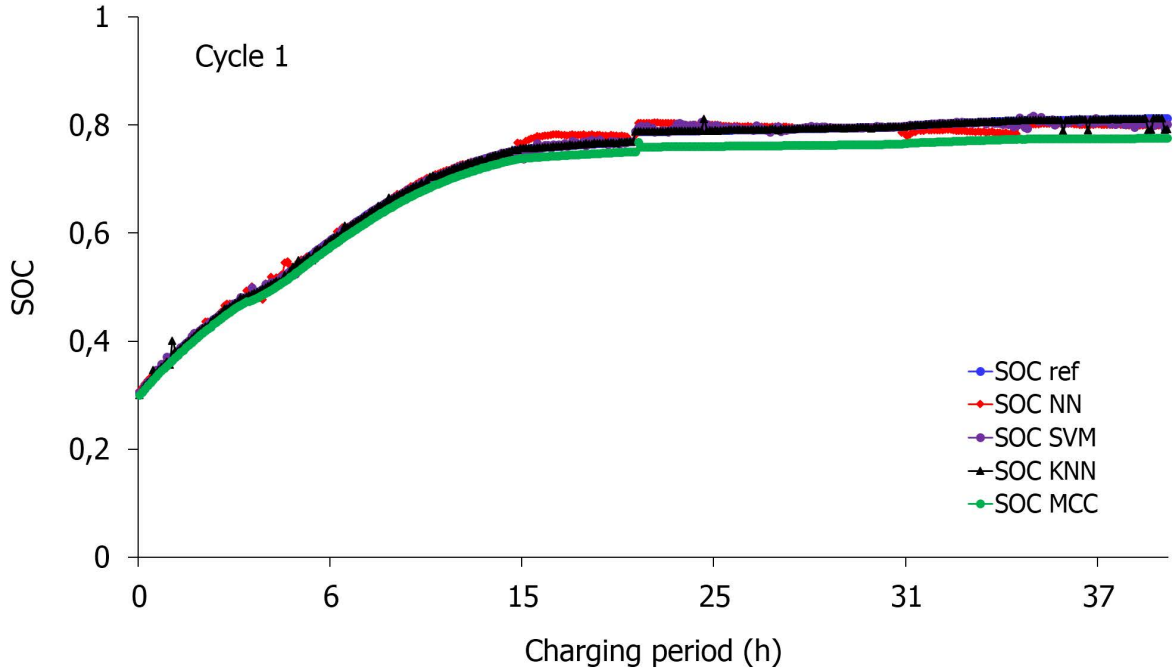

Figure 5.15: SoC modelling for lead-acid battery during charge mode

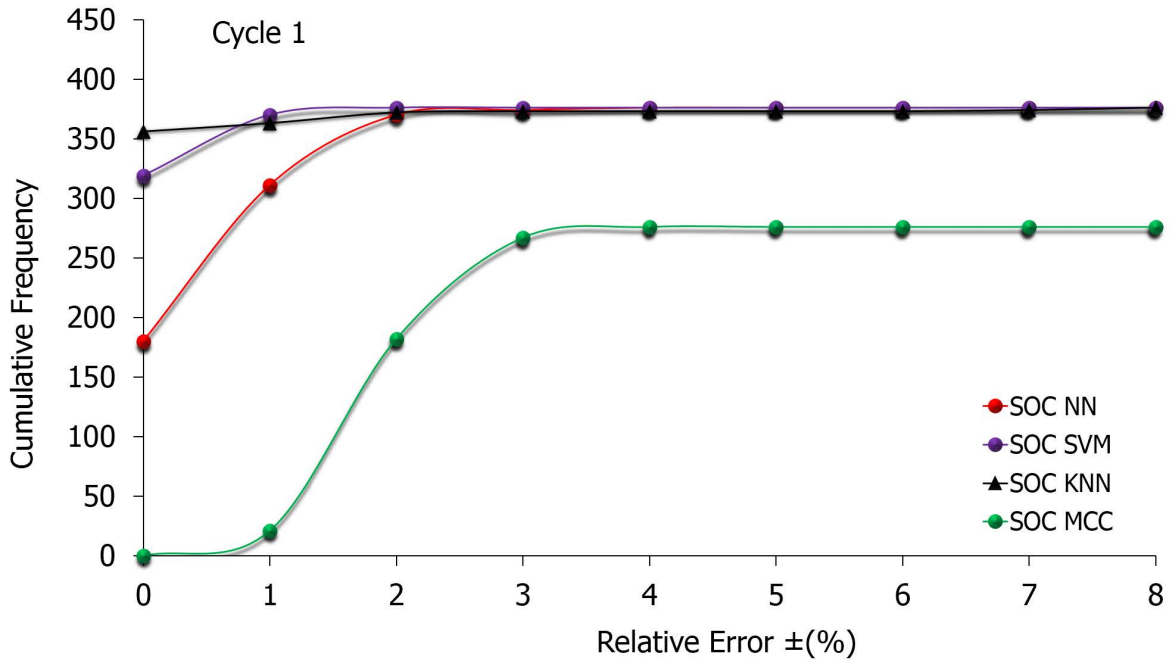

Figure 5.16: Cumulative Frequency Vs relative error for lead-acid during charge mode.

RE (± %)	0	1	2	3	4	5	6	7	8
SoC_{NN}	46.8	80.86	96.2	97.24	100	100	100	100	100
SoC_{SVM}	82.94	96.20	100	100	100	100	100	100	100
SoC_{KNN}	92.56	94.38	96.72	96.98	97.24	100	100	100	100
SoC_{MCC}	0	5.46	47.32	69.42	100	100	100	100	100

Table 5.5: Comparative results of cumulative frequency distribution Vs. The relative error for charge mode of lead-acid battery during cycle 1.

To enhance a practical selection for the lead-acid battery model operating SoC model in the PV system. Another analysis is provided. In Table 5.6, further comparisons between the five SoC models considering four cycles of charging mode will be made based on statistical errors.

	Cycle 1		Cycle 2		Cycle 3		Cycle 4	
	MBE	RMSE	MBE	RMSE	MBE	RMSE	MBE	RMSE
SoC_{NN}	-5.33 $\cdot 10^{-6}$	0.0001	0	0	-4.68 $\cdot 10^{-5}$	0.001	-2.94 $\cdot 10^{-6}$	0.000049
SoC_{SVM}	3.2 $\cdot 10^{-6}$	0.0063	8.34 $\cdot 10^{-8}$	0.000002	-2.72 $\cdot 10^{-7}$	0.000064	-2.56 $\cdot 10^{-6}$	0.000043
SoC_{KNN}	1.56 $\cdot 10^{-5}$	0.0003	-2.64 $\cdot 10^{-5}$	0.00038	-6.01 $\cdot 10^{-5}$	0.0014	-1.84 $\cdot 10^{-6}$	0.000031
SoC_{MCC}	1.43 $\cdot 10^{-5}$	0.0002	-2.62 $\cdot 10^{-5}$	0.00037	-6.04 $\cdot 10^{-6}$	0.00014	-1.75 $\cdot 10^{-6}$	0.000029

Table 5.6: Statistical errors of lead-acid battery during four cycles of charge mode

From table 5.6, the battery SoC_{NN} displays very low values of MBE and RMSE during the four charging cycles compared with the remaining models to achieving -5.33.10-6 of MBE and 0.0001 of RMSE respectively during the first cycle. During the second cycle, SoC_{NN} reaches zero MBE and zero RMSE. In addition, -4.68 $\cdot 10^{-5}$ MBE and -2.94$\cdot 10^{-5}$ RMSE were realized during cycle 3 and -4.68$\cdot 10^{-5}$ MBE, -2.94$\cdot 10^{-6}$ RMSE during cycle 4.

We can notice that around 92% of the SoC_{KNN} has a RE of ±0% but it only achieves a perfect score on its data at 5% RE, whereas the SoC_{SVM} has achieved all its data with RE less or equal to 2%, followed by the MCC, which acquired all its data with RE equal to 4%. NN on the other hand has 46.8% of the predictions at 0% RE and achieves perfect score in under 3% RE.

Fig.5.17 depicts the Taylor diagram standard error for lead-acid battery during charge mode related to cycle 1. We notice that KNN, SVM, NN have more or less the same standard deviation as the reference (0.175), however, of the three models SVM has the highest correlation (0.997) with an RMSE of 0.005 followed by NN (0.992 correlation and 0.01 RMSE) and finally KNN (0.97 correlation and 0.027 RMSE). MCC presents a different behavior, it has a lower standard deviation compared to the previous three (0.12) and an RMSE comparable to that of NN but with a slightly higher correlation to the reference (that still doesn't surpass that of SVM).

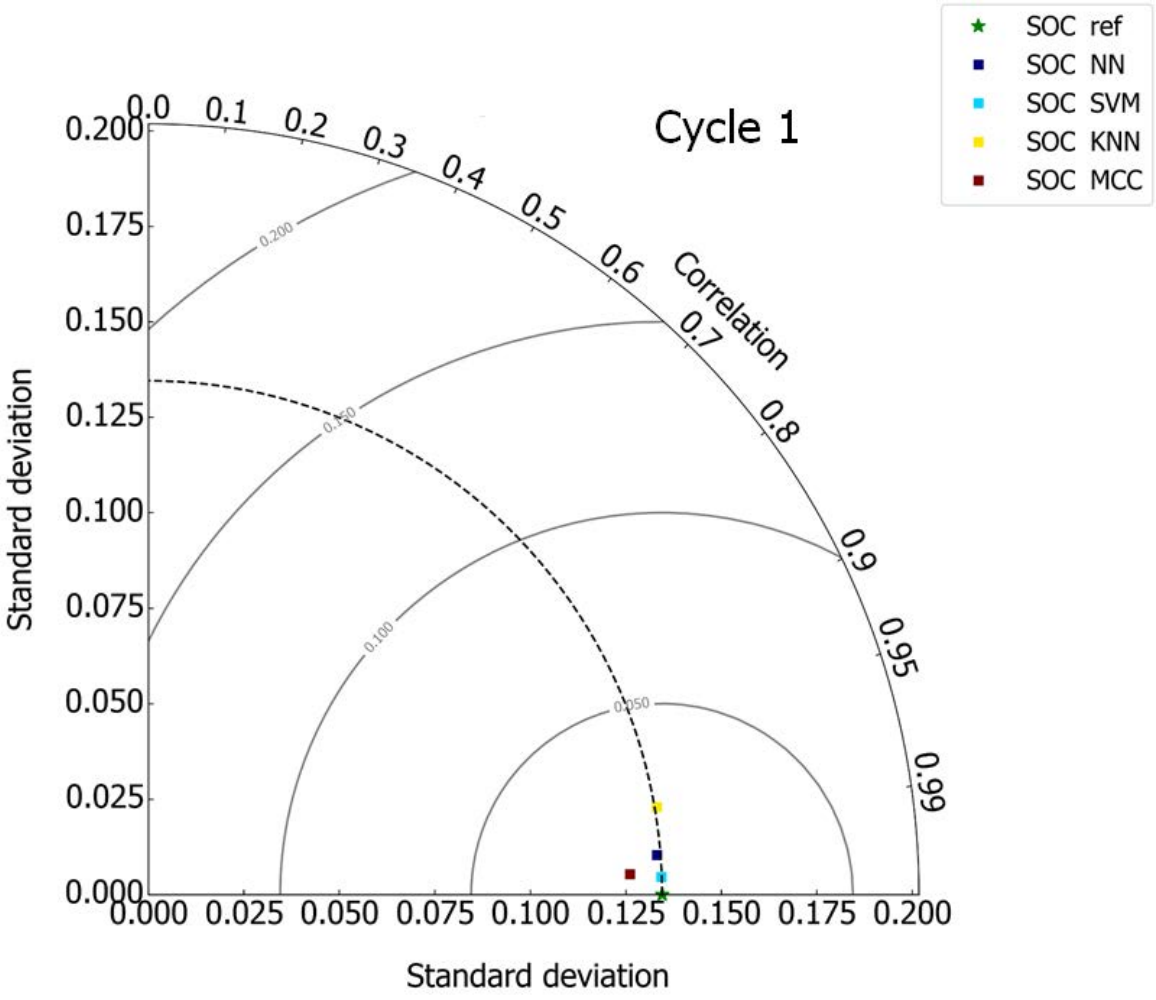

Figure 5.17: Taylor diagram standard error for lead-acid during charge mode

The previously obtained results suggest that the battery SoC_{NN} is more adequate for the lead-acid charging mode. To better highlight the battery SoC_{NN} model, another quantitative analysis is provided for the lead-acid charging mode, considering four battery cycles. As shown in Fig.5, notice that the majority of the data is within zero error for the four charging cycles.

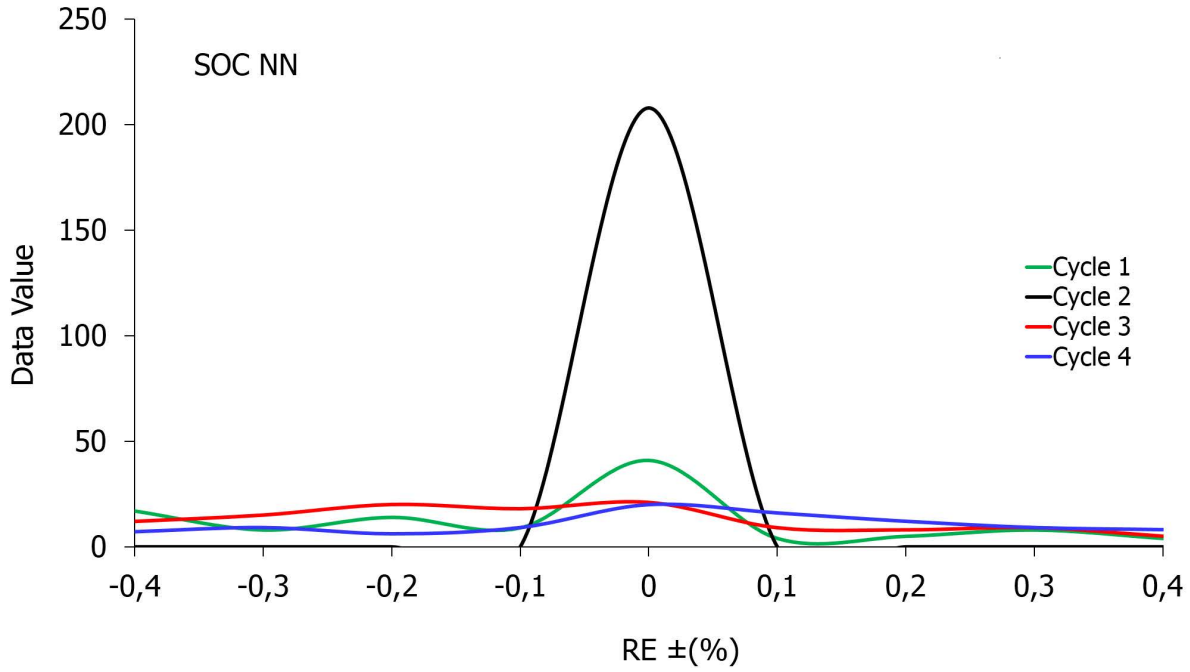

Figure 5.18: Data quantity Vs relative error based on SoC_{NN} for leadacid battery during 04 cycles of charge mode

5.3.2 Discharge

As shown in Fig.5.19, the battery was initially fully charged (with 100% SoC) and disconnected from the PV array; the battery supplies an external load of only 220 W to ensure the discharge process. The SoC during this mode is a linear function and decreases as the battery supplies the utility over time.

Fig.5.19 show a comparison of four SoC models during cycle 1 of the discharge mode. All the simulated models present a good compatibility with SoC_{ref}, the SoC_{SVM} model shows some fluctuations at first but they disappear after a while.

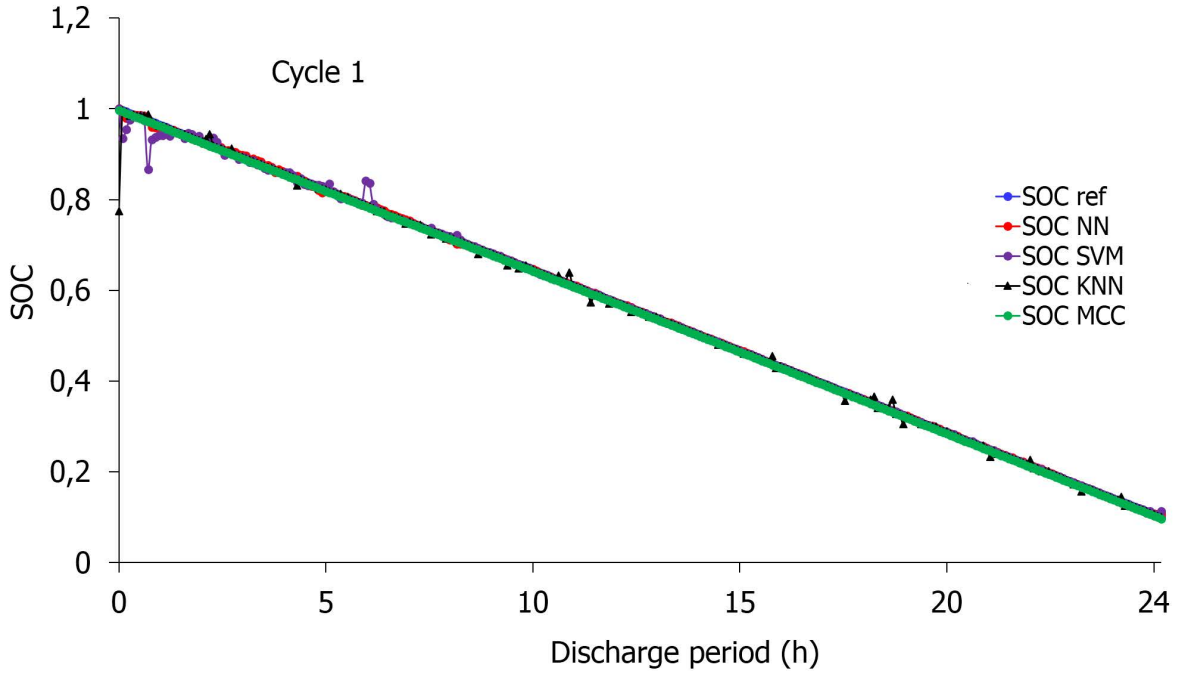

Figure 5.19: SoC modelling for lead-acid battery during discharge mode

Fig.5.20 illustrates the cumulative frequency via the RE (%) during the discharge mode for lead-acid battery. Table 5.7 presents numerical results of these cumulative frequencies achieved for RE (%) within a range of 0 to ± 11%; during the first cycle of discharging mode.

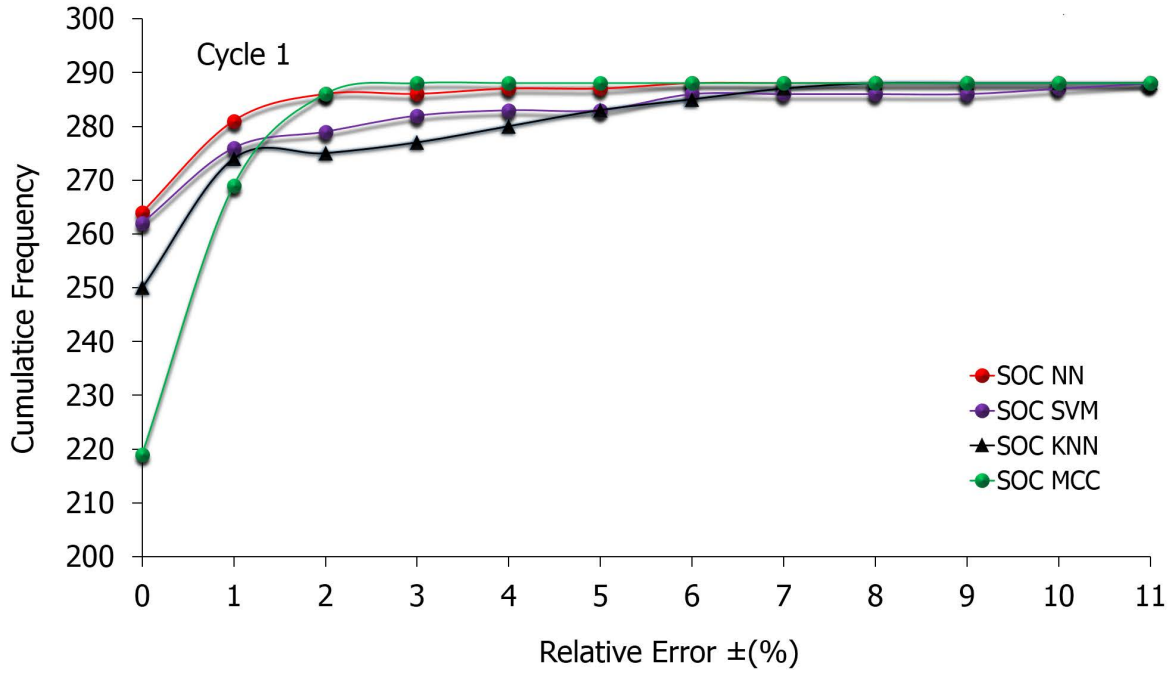

Figure 5.20: Cumulative Frequency Vs relative error for lead-acide battery during discharge mode.

RE (± %)	0	1	2	3	4	5	6	7	8	9	10	11
SoC_{NN}	89.76	95.54	97.34	97.24	97.58	97.58	100	100	100	100	100	100
SoC_{SVM}	89.09	93.84	94.86	95.88	96.22	96.22	97.24	97.24	97.24	97.24	97.58	100
SoC_{KNN}	85	93.16	93.5	94.18	95.2	96.22	96.9	97.58	100	100	100	100
SoC_{MCC}	74.46	91.46	97.24	100	100	100	100	100	100	100	100	100

Table 5.7: Comparative results of cumulative frequency distribution Vs. The relative error for discharge mode of lead acid battery during cycle 1

It is noted that around 89.76% of the SoC_{NN} has a RE of ±0%, whereas the SoC_{MCC} has only 74.46% of data for the same error, followed by SoC_{SVM} (89.09) and SoC_{KNN} (85) respectively, SoC_{MCC} achieved all its data with RE less or equal to 3%, followed by the SoC_{NN} at 6% RE then SoC_{KNN} at 8% and finally SoC_{SVM} which acquired all its data with RE equal to 11%.

Fig.5.21 depicts the Taylor diagram standard error for lead-acid battery during discharge mode related to cycle 1. We observe that all models have a standard deviation equal to that of the reference (0.26). They only vary in terms of RMSE and correlation; MCC and NN have a high correlation to the target point approaching the ideal value of 1 (and an RMSE approaching 0), SVM and KNN however seem to be located a bit further away but close to each other, with a correlation of 0.998 and an RMSE of 0.015 for SVM as well as a correlation of 0.997 for KNN (with 0.02 RMSE).

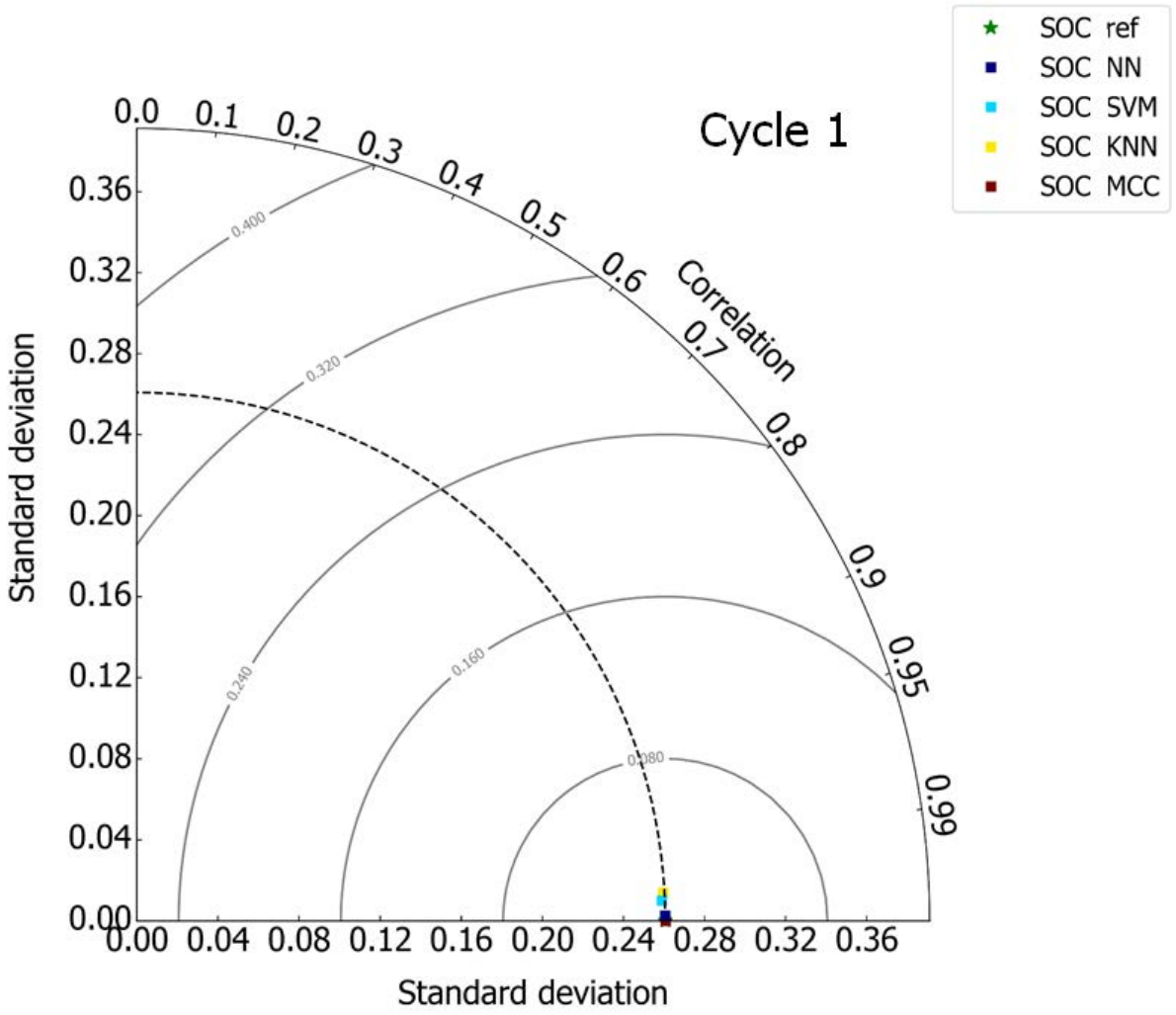

Figure 5.21: Taylor diagram standard error for lead-acid battery during discharge mode

To improve a proper choice for the lead-acid battery model operating SoC model in the PV system. another statistical analysis is provided based on MBE and RMSE errors as shown in Table 5.8, describing further comparisons between the five SoC models considering four cycles discharging mode.

	Cycle 1		Cycle 2		Cycle 3		Cycle 4	
	MBE	RMSE	MBE	RMSE	MBE	RMSE	MBE	RMSE
SoC_{NN}	$1.08 \cdot 10^{-8}$	0.00013	$-6.26 \cdot 10^{-7}$	0.00029	$-2.71 \cdot 10^{-7}$	0.000108	$-5.75 \cdot 10^{-7}$	0.00012
SoC_{SVM}	$-7.97 \cdot 10^{-7}$	0.013	$2.05 \cdot 10^{-7}$	0.000095	$-4.33 \cdot 10^{-5}$	0.0017	$-1.95 \cdot 10^{-7}$	0.000043
SoC_{KNN}	$7.80 \cdot 10^{-4}$	0.013	0	0	$-4.01 \cdot 10^{-7}$	0.00016	$-3.15 \cdot 10^{-6}$	0.0007
SoC_{MCC}	$1.00 \cdot 10^{-5}$	0.00017	$1.10 \cdot 10^{-6}$	0.00051	$1.97 \cdot 10^{-7}$	0.000078	$3.57 \cdot 10^{-8}$	0.000008

Table 5.8: Statistical errors of lead-acid battery during four cycles of charge mode.

Based on table 4, the SoC_{NN} has the lowest MBE and RMSE for cycles 1 ($1.08 \cdot 10^{-8}$ and $-6.26 \cdot 10^{-7}$ respectively), the second lowest for cycle 3 and the third lowest values for the two remaining cycles. SoC_{SVM} has the second lowest values for cycles 1,2 and 4 but displays very a high MBE and RMSE during cycle 3. Moreover, the SoC_{KNN} reaches 0 error of MBE and RMSE during cycle 2 but shows high values for the remaining cycles. MCC on the other hand has very low error values for Cycle 3 and 4 but displays higher values in the other two cycles.

The previously obtained results suggest that the battery SoC_{NN} is more adequate for the lead-acid charging mode. To better highlight the battery SoC_{NN} model, another quantitative analysis is provided for the lead-acid discharging mode, considering four battery cycles. As shown in Fig.5.22, the majority of the data is centered around zero error for the four charging cycles.

Based on the aforementioned analysis, the battery SoC_{NN} model is more suited for lead-acid battery manufacturers achieving less than $2.6 \cdot 10^{-3}(\%)$ of MBE and RMSE errors.

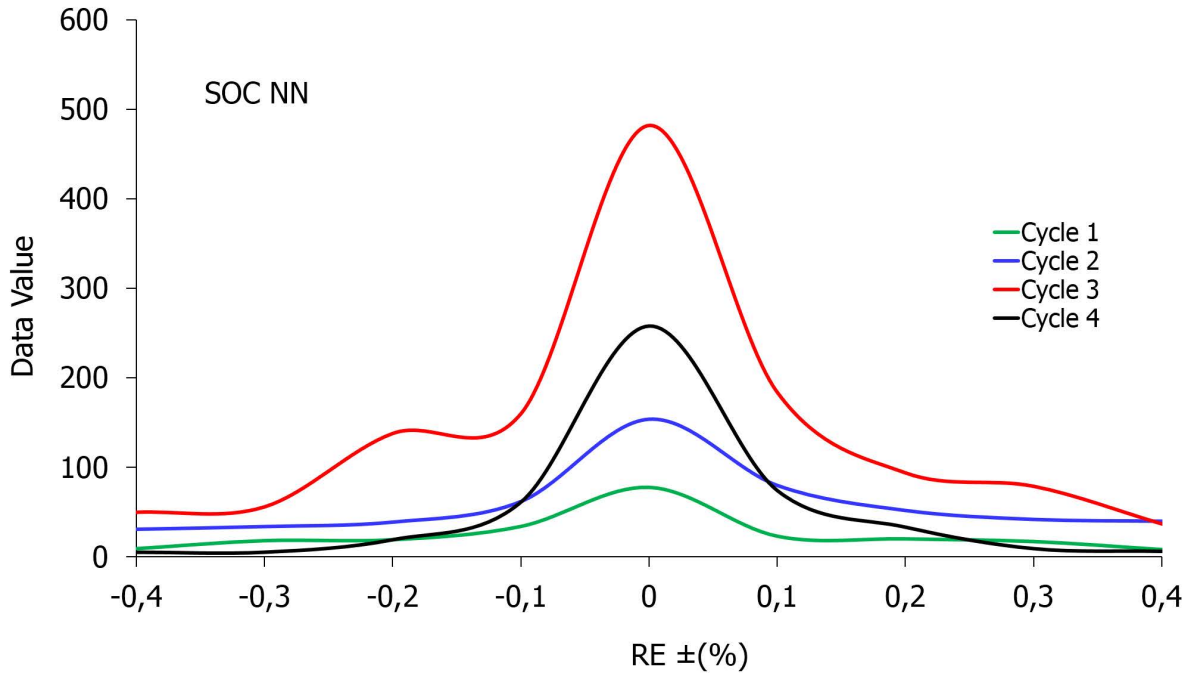

Figure 5.22: Data quantity Vs relative error based on SoC_{NN} for leadacid battery during 04 cycles of charge mode.

5.3.3 Validation

This section's objective consists of validating the updated SoC_{NN} model in a real operating stand-alone PV system. The system consists of a PV array with a nominal power of 750 Wp, a lead-acid battery and DC load. The model validation involved two different scenarios: the first one depicts an unfavorable scenario recorded in the span of one day as displayed in Fig. 5.23 in which the consumption evolves inversely with the solar irradiace (the simulation was started with a completely discharged battery). The second scenario depicts a linear profile, which means the consumption I_{load} remains constant when the irradiance varies , the model was tested along two working days with a fully charged battery (with an SoC of 65%).

Fig.5.25 displays two typical periods of operation of the PV system under study, during which the PV current produced from the daily solar irradiation I_{pv}, the charging battery current I_{bat}, and the load consumption I_{load} are reported, Fig.5.24 and Fig.5.26 show the battery SoC_{NN} evolution compared to the SoC_{ref}. A good similarity between the SoC_{NN} and the reference SoC is observed during both charge and discharge modes.

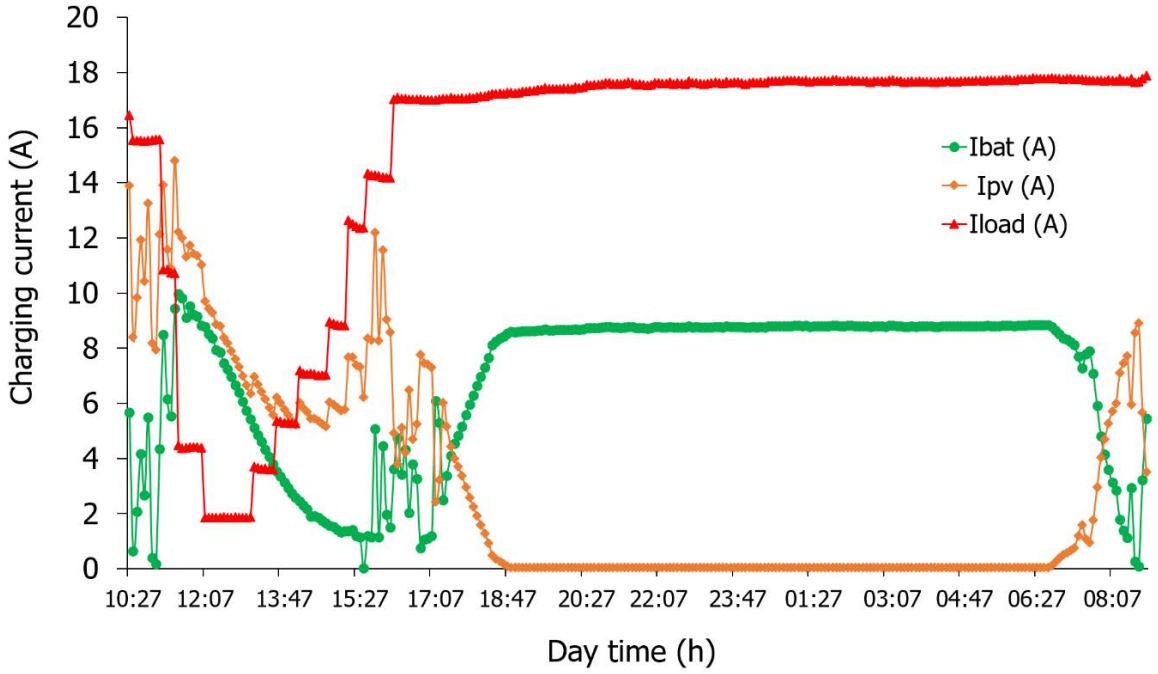

Figure 5.23: Battery current profile evolution with PV and load current in standalone PV system for scenario 1.

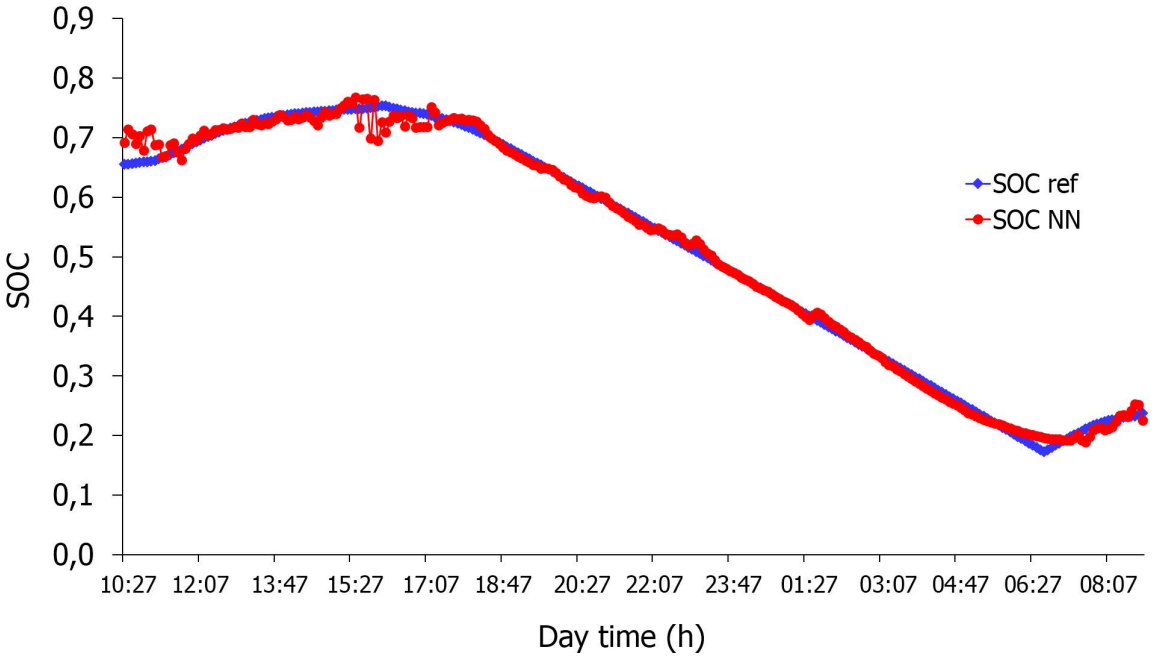

Figure 5.24: Simulation of battery SoC for scenario 1.

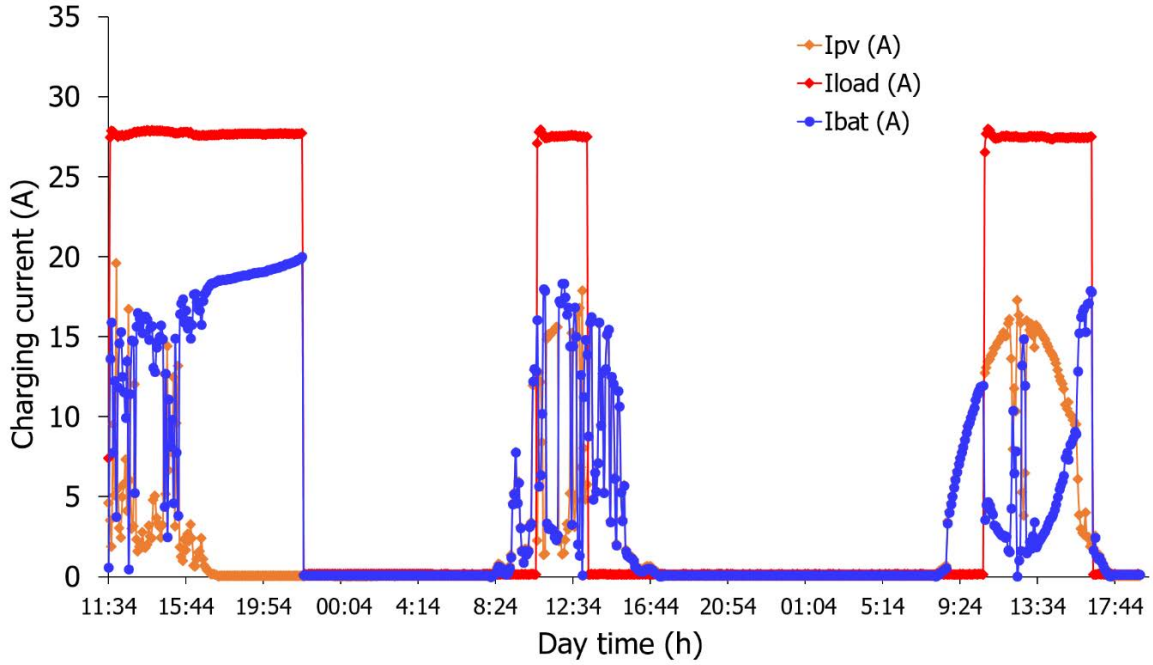

Figure 5.25: Battery current profile evolution with PV and load current in standalone PV system for scenario 2.

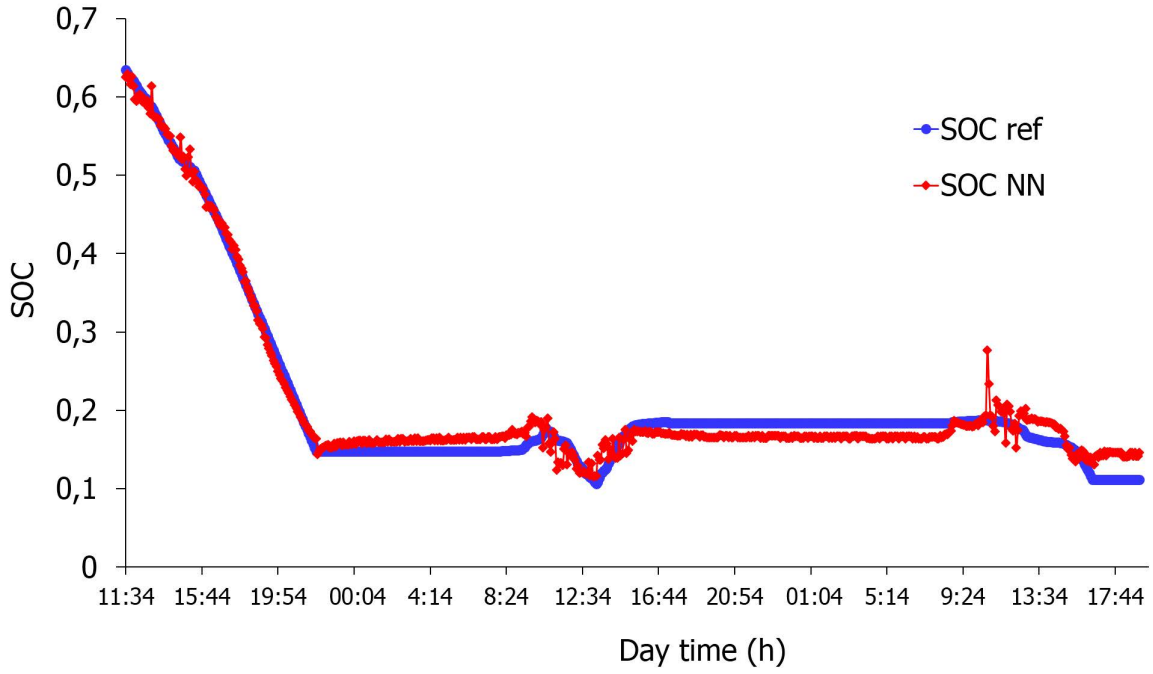

Figure 5.26: Simulation of battery SoC for scenario 2

5.4 Conclusion

In this chapter we've displayed the collected data, analysed the different SoC estimation methods quantitatively and qualitatively, selected the optimal one and validated it in different standalone PV cycles. We've gone through this process for a lead-acid battery and lithium-ion battery. We've concluded that the K-Nearest Neigbours method is best for the lithium-ion battery while the neural network model is best for the lead-acid battery.

CHAPTER 6

CONCLUSION

In this thesis, we touched upon a variety of elements related to the state of charge estimation problem. In chapter 2, we exposed a myriad of SoC estimation methods, specifying the research that went into the field, as well as some recent innovations and novel approaches to it.

The algorithm used in commercial battery management systems is Coulomb-Counting. The model in itself is considered as a highly trusted method for SOC estimation but presents a huge disadvantage in practice; it uses an integral (or cumulative summation) in its formulation. This means that the model is only recommended when the arithmetic units have sufficient accuracy and storing capability to prevent the errors from piling up, which is often not the case for commercial BMS's. The improvement we achieved concerned the use of the Coulomb-Counting model as a mere reference to build, train and optimize other algorithms that would finally produce results as trust-worthy as those of the ideal Coulomb-Counting, while being more practically implementable into Commercial BMS's. In chapter 3, we selected a handful of those models that seemed the most coherent with our objectives and endeavored in elucidating the theory behind each one of them. The end goal being to deduce an appropriate way to implement them.

In chapter 4, we demonstrated our modus operandi exposing the experimental test-bench used to measure and record the different measurements for the two battery technologies. Afterwards, we fleshed out the strategy behind the implementation of the five SoC algorithms, as well as the platforms and programming languages used.

The subsequent chapter 5 describes a detailed analysis we performed on the different

results produced by the models. First, we represented the estimations in the form of regular graphs, which showed a highly convergent behavior towards the reference from most algorithms. This warranted the need of a more sophisticated way to distinguish between their performances. We resorted to statistical metrics like the cumulative frequencies of the relative errors along with the Mean Bias Error (MBE) and the Root Mean Square Error (RMSE). To solidify our study even further, we also utilized the Taylor Diagram, which is a graphical representation of three statistical metrics at once, namely the Correlation, the RMSE and the standard deviation.

Taking all of that into account, we established that the optimal model for state of charge estimation in lithium-Ion batteries is the K-Nearest Neighbors Regressor. The model displayed a high predictive accuracy, an optimal standard deviation and the majority of its predictions had relative errors close to 0%. KNN is inherently simple as an algorithm (arguably the simplest model of the five) because it relies solely on a distance calculus, combined with a comparison and a mean calculation, all of which necessitate little computing power on hardware level.

For lead-acid batteries, the favored model was the Neural Network algorithm on both charge and discharge modes. This time however, accuracy varied strongly amongst the models throughout the 4 cycles. The reason we arbitrated in favor of NN was because of its overall consistency (the other models would perform well on a couple of cycles but show very poor estimations on the others).

To close off this enquiry, it was necessary to test the selected models on a validation dataset. Two autonomous cycles portraying several charge-discharge phases were used for the lithium-ion battery, the KNN model displayed high accuracy estimations for both of them, proof of its strong predictive ability. In the case of lead-acid battery, we started by validating on an unfavorable cycle presenting an irregular scenario in which the current was forced to evolve inversely with the irradiance. Nevertheless, the NN model still managed to provide highly accurate estimations. Subsequently, we validated the model on a regular linear cycle which produced very precise predictions too.

In terms of real-life implementation of the two models and its feasibility, we must reiterate that the KNN model (selected for lithium-Ion battery) is more practical to realize on a regular Battery Management System, or a UPS (Uninterruptible Power System) in general, due to the simplicity of its formulation as well as the limited number of calculus it needs to perform. For the Neural Network algorithm however, more effort and material might be needed because of its complexity and heavy weight. In [23] and idea was proposed to implement Back Propagation neural networks by connecting a

pc (capable of running the algorithm) to the battery management system through an ethernet cable. Other works ([2]) have studied the use of Field Programmable Gate Arrays (FPGA) in implementing said model.

In conclusion, this thesis presents a study of different models applied in estimating the State of Charge for two different technologies of batteries. A comparative analysis between the different estimations revealed the algorithms capable of achieving state of the art predictions for each type. The retained models (namely K-Nearest Neighbors for lithium-Ion Batteries and Neural Networks for Lead acid) have been tested for accuracy on autonomous scenarios involving multiple charging/discharging phases. It was concluded hat the two selected methods satisfied the statistic requirements of performance and were deemed ready to be forwarded into the real-life implementation discussion.

BIBLIOGRAPHY

[1] G. C. S. ALMEIDA, A. C. Z. D. SOUZA, AND P. F. RIBEIRO, *A Neural Network Application for a Lithium-Ion Battery Pack State-of-Charge Estimator with Enhanced Accuracy*, (2020).

[2] F. BARONTI, R. RONCELLA, R. SALETTI, AND W. ZAMBONI, *Fpga implementation of the mix algorithm for state-of-charge estimation of lithium-ion batteries*, in IECON 2014-40th annual conference of the IEEE industrial electronics society, IEEE, 2014, pp. 5641–5646.

[3] D. BASAK, S. PAL, D. CH, AND R. PATRANABIS, *Support vector regression*, (2007), pp. 203–224.

[4] N. BELAZI, *Estimation de l'état de charge (SOC) et de l'état de santé (SOH) d'une batterie Lithium-Ion utilisée pour les voitures électriques*, PhD thesis, École de technologie supérieure, 2012.

[5] T. CAI, Y. LIU, Z. HE, M. GAO, AND J. LIU, *SOC estimation of Lithium-ion battery based on an Extended H-infinity filter*, in 2019 IEEE 17th International Conference on Industrial Informatics (INDIN), Helsinki, Finland, July 2019, IEEE, pp. 1700–1705.

[6] G. H. CHEN AND D. SHAH, *Explaining the success of nearest neighbor methods in prediction*, Foundations and Trends in Machine Learning, 10 (2018), pp. 337–588.

[7] L. CHEN, Z. WANG, Z. LU, J. LI, B. JI, H. WEI, AND H. PAN, *A Novel State-of-Charge Estimation Method of Lithium-Ion Batteries Combining the Grey Model and Genetic Algorithms*, IEEE Transactions on Power Electronics, 33 (2018), pp. 8797–8807.

[8] Z. CHEN, H. SUN, G. DONG, J. WEI, AND J. WU, *Particle filter-based state-of-charge estimation and remaining-dischargeable-time prediction method for lithium-ion batteries*, Journal of Power Sources, 414 (2019), pp. 158–166.

[9] A. DEGLA, M. CHIKH, A. CHOUDER, F. BOUCHAFAA, AND A. TAALLAH, *Update battery model for photovoltaic application based on comparative analysis and parameter identification of lead‚Äìacid battery models behaviour*, IET Renewable Power Generation, 12 (2018), pp. 484–493.

[10] X. DING, D. ZHANG, J. CHENG, B. WANG, AND P. C. K. LUK, *An improved Thevenin model of lithium-ion battery with high accuracy for electric vehicles*, Applied Energy, 254 (2019), p. 113615.

[11] I. ERMANOVA, N. YAGHOOBI NIA, E. LAMANNA, E. DI BARTOLOMEO, E. KOLESNIKOV, L. LUCHNIKOV, AND A. DI CARLO, *Crystal Engineering Approach for Fabrication of Inverted Perovskite Solar Cell in Ambient Conditions*, Energies, 14 (2021), p. 1751.

[12] H. FANG, X. ZHAO, Y. WANG, Z. SAHINOGLU, T. WADA, S. HARA, AND R. A. DE CALLAFON, *State-of-charge estimation for batteries: A multi-model approach*, in 2014 American Control Conference, Portland, OR, USA, June 2014, IEEE, pp. 2779–2785.

[13] N. GUENTHER AND M. SCHONLAU, *Support vector machines*, Stata Journal, 16 (2016), pp. 917–937.

[14] L. HE AND D. GUO, *An Improved Coulomb Counting Approach Based on Numerical Iteration for SOC Estimation With Real-Time Error Correction Ability*, IEEE Access, 7 (2019), pp. 74274–74282.

[15] M. S. HOSSAIN LIPU, M. A. HANNAN, A. HUSSAIN, M. H. SAAD, A. AYOB, AND M. N. UDDIN, *Extreme Learning Machine Model for State-of-Charge Estimation of Lithium-Ion Battery Using Gravitational Search Algorithm*, IEEE Transactions on Industry Applications, 55 (2019), pp. 4225–4234.

[16] X. HU, F. SUN, AND Y. ZOU, *Estimation of State of Charge of a Lithium-Ion Battery Pack for Electric Vehicles Using an Adaptive Luenberger Observer*, Energies, 3 (2010), pp. 1586–1603.

[17] M. ISMAIL, R. DLYMA, A. ELRAKAYBI, R. AHMED, AND S. HABIBI, *Battery state of charge estimation using an Artificial Neural Network*, in 2017 IEEE Transportation Electrification Conference and Expo (ITEC), Chicago, IL, USA, June 2017, IEEE, pp. 342–349.

[18] S. J. JULIER AND J. K. UHLMANN, *A New Extension of the Kalman Filter to Nonlinear Systems*, p. 12.

[19] R. E. KALMAN, *A new approach to linear filtering and prediction problems.*, (1960).

[20] J. KIM AND B. H. CHO, *State-of-Charge Estimation and State-of-Health Prediction of a Li-Ion Degraded Battery Based on an EKF Combined With a Per-Unit System*, IEEE Transactions on Vehicular Technology, 60 (2011), pp. 4249–4260.

[21] M. C. KOCER, C. CENGIZ, M. GEZER, D. GUNES, M. A. CINAR, B. ALBOYACI, AND A. ONEN, *Assessment of Battery Storage Technologies for a Turkish Power Network*, Sustainability, 11 (2019), p. 3669.

[22] X. LAI, S. WANG, L. HE, L. ZHOU, AND Y. ZHENG, *A hybrid state-of-charge estimation method based on credible increment for electric vehicle applications with large sensor and model errors*, Journal of Energy Storage, 27 (2020), p. 101106.

[23] S. LI, S. LI, H. ZHAO, AND Y. AN, *Design and implementation of state-of-charge estimation based on back-propagation neural network for smart uninterruptible power system*, International Journal of Distributed Sensor Networks, 15 (2019).

[24] W. LI, Y. YANG, D. WANG, AND S. YIN, *The multi-innovation extended Kalman filter algorithm for battery SOC estimation*, Ionics, 26 (2020), pp. 6145–6156.

[25] R. E. LOPEZ, *Nearest Star: The Surprising Science of Our Sun* ***Nearest Star: The Surprising Science of Our Sun*** *Leon Golub and Jay M. Pasachoff , Harvard U. Press, Cambridge, Mass., 2001. $29.95 (304 pp.). ISBN 0-674-00467-1*, Physics Today, 54 (2001), pp. 59–60.

[26] M. J. MANWELL J, *Extension of the kinetic battery model for wind/hybrid power systems.*, (1994).

[27] S. MOORE AND M. ESHANI, *An Empirically Based Electrosource Horizon Lead-Acid Battery Model*, Feb. 1996, p. 960448.

[28] F. PEDREGOSA, G. VAROQUAUX, A. GRAMFORT, V. MICHEL, B. THIRION, O. GRISEL, M. BLONDEL, P. PRETTENHOFER, R. WEISS, V. DUBOURG, J. VANDERPLAS, A. PASSOS, D. COURNAPEAU, M. BRUCHER, M. PERROT, AND É. DUCHESNAY, *Scikit-learn: Machine learning in Python*, Journal of Machine Learning Research, 12 (2011), pp. 2825–2830.

[29] B. RZEPKA, S. BISCHOF, AND T. BLANK, *Implementing an Extended Kalman Filter for SoC Estimation of a Li-Ion Battery with Hysteresis: A Step-by-Step Guide*, Energies, 14 (2021), p. 3733.

[30] S. SATO AND A. KAWAMURA, *A new estimation method of state of charge using terminal voltage and internal resistance for lead acid battery*, in Proceedings of the Power Conversion Conference-Osaka 2002 (Cat. No.02TH8579), vol. 2, Osaka, Japan, 2002, IEEE, pp. 565–570.

[31] SICKIT-LEARN, *Cross-validation: evaluating estimator performance [Online]. Available:*.
`https://scikit-learn.org/stable/modules/cross_validation.html`.

[32] M. S. SIDHU, D. RONANKI, AND S. WILLIAMSON, *Hybrid State of Charge Estimation Approach for Lithium-ion Batteries using k-Nearest Neighbour and Gaussian Filter-based Error Cancellation*, in 2019 IEEE 28th International Symposium on Industrial Electronics (ISIE), Vancouver, BC, Canada, June 2019, IEEE, pp. 1506–1511.

[33] A. J. SMOLA AND B. SCHÖLKOPF, *A tutorial on support vector regression*, Statistics and computing, 14 (2004), pp. 199–222.

[34] S. SONG, Z. WEI, H. XIA, M. CEN, AND C. CAI, *State-of-charge (SOC) estimation using T-S Fuzzy Neural Network for Lithium Iron Phosphate Battery*, in 2018 26th International Conference on Systems Engineering (ICSEng), Sydney, Australia, Dec. 2018, IEEE, pp. 1–5.

[35] A. STEFANOPOULOU AND Y. KIM, *System-level management of rechargeable lithium-ion batteries*, 2015.

[36] G. WELCH, G. BISHOP, ET AL., *An introduction to the kalman filter*, (1995).

[37] Q. YANG, J. XU, B. CAO, AND X. LI, *A simplified fractional order impedance model and parameter identification method for lithium-ion batteries*, PLOS ONE, 12 (2017), p. e0172424.

[38] J. ZHANG, L. ZHANG, F. SUN, AND Z. WANG, *An Overview on Thermal Safety Issues of Lithium-ion Batteries for Electric Vehicle Application*, IEEE Access, 6 (2018), pp. 23848–23863.

[39] L. ZHANG, K. LI, D. DU, C. ZHU, AND M. ZHENG, *A sparse least squares support vector machine used for SOC estimation of Li-ion Batteries*, IFAC-PapersOnLine, 52 (2019), pp. 256–261.

Technical data Isofoton IS- 150/24

Electrical data

NOMINAL OUTPUT PMPP : 150W
MAX .POWER TOLERANCE : +/-5[%]
MAX .VOLTAGE SYSTEM : 34,6V
NOMINAL VOLTAGE UMPP : 24V
NOMINAL CURRENT IMPP : 4,35 AMPS
OPEN CIRCUIT VOLTAGE V_{oc} : 43,2V
SHORT CIRCUIT I_{sc} : 4,45 AMPS
TEMPERATURE COEFFICIENT OF SHORT CIRCUIT VOLTAGE : KA
TEMPERATURE COEFFICIENT OF SHORT CIRCUIT CURRENT : KA
TEMPERATURE COEFFICIENT OUTPUT : KA
CELL CONVERSION EFFECIENCY: KA
MODULE CONVERSION EFFICIENCY:KA
CERTIFICATE : IEC 61215 SCHUTZKLASSE II CE-KONFORMITAT

Dimensions

LENGTH : 1224 MM / 48,19
DEPTH : 39,5 M /1,56
WIDTH : 1047M/41,22
WEIGHT : 17KG/37,5
TYPE OF OUTPUT TERMINAL : J-BOX MC CABLES

Cells

NUMBER OF CELLS PER MODULE : 72
CELL TECHNOLOGY : MONO CRYSTAL
CELL FORM : KA
CELL SIZE : 125 ×125[MM]
CELL CONTACTING :KA

Figure 1: Lithium-ion battery technical data.

Technical data Isofoton IS- 150/24

Electrical data

NOMINAL OUTPUT PMPP : 150W
MAX .POWER TOLERANCE : +/-5[%]
MAX .VOLTAGE SYSTEM : 34,6V
NOMINAL VOLTAGE UMPP : 24V
NOMINAL CURRENT IMPP : 4,35 AMPS
OPEN CIRCUIT VOLTAGE V_{oc} : 43,2V
SHORT CIRCUIT I_{sc} : 4,45 AMPS
TEMPERATURE COEFFICIENT OF SHORT CIRCUIT VOLTAGE : KA
TEMPERATURE COEFFICIENT OF SHORT CIRCUIT CURRENT : KA
TEMPERATURE COEFFICIENT OUTPUT : KA
CELL CONVERSION EFFECIENCY: KA
MODULE CONVERSION EFFICIENCY:KA
CERTIFICATE : IEC 61215 SCHUTZKLASSE II CE-KONFORMITAT

Dimensions

LENGTH : 1224 MM / 48,19
DEPTH : 39,5 M /1,56
WIDTH : 1047M/41,22
WEIGHT : 17KG/37,5
TYPE OF OUTPUT TERMINAL : J-BOX MC CABLES

Cells

NUMBER OF CELLS PER MODULE : 72
CELL TECHNOLOGY : MONO CRYSTAL
CELL FORM : KA
CELL SIZE : 125 ×125[MM]
CELL CONTACTING :KA

Figure 2: Lead-acid battery technical data.