UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería en Organización Industrial

Modelling and State Estimation of Batteries

Autor:

Rodríguez Asensio, Miguel

Responsable de Intercambio en la UVa:

Fernando Tadeo Rico

Universidad de destino:

Université de Picardie Jules Verne - Amiens

Valladolid, Julio 2016.

TFG REALIZADO EN PROGRAMA DE INTERCAMBIO

TÍTULO:	Modelling and state estimation of batteries
ALUMNO:	Miguel Rodríguez Asensio
FECHA:	28/06/2016
CENTRO:	Laboratoire MIS
TUTOR:	Prof. A. El Hajjaji

Resumen

En este trabajo se ha realizado un estudio sobre el modelado y la estimación de los estados de una batería de Litio principalmente, y en la parte final del mismo sobre una batería de Plomo. Este estudio se ha desarrollado simultáneamente en MapleSim, Matlab y Simulink, utilizando el algoritmo denominado *Filtro de Kalman* (*Kalman Filter* en inglés) para estimar el estado de carga (SOC) y el estado de salud (SOH) de la batería. Este algoritmo ha sido extensamente validado a lo largo del trabajo mediante simulación, y se ha llegado a demostrar su robustez contra el ruido utilizado. Por otro lado también se ha estudiado la degradación que sufre la batería en función de la temperatura de las celdas que la componen.

Palabras clave

Filtro de Kalman, batería de Litio, estado de carga (SOC), estado de salud (SOH) y modelado baterías.

Content

1 INTRODUCTION TO BATTERIES, BATTERY MODELLING AND MAPLESIM ENVIRONMENT

1.1 ORIGINS OF THE BATTERIES

The first battery was invented by Alessandro Volta in 1800. There are two main kinds of batteries: the primary cells which, once they had been used, they cannot be recharged, and secondary cells which can be used many times thanks to their capacity of recharging when you supply an external current [1]. In Table 1 it is shown a classification with several examples of both kinds of batteries [2].

Primary Cells	**Secondary Cells**
Zinc Carbon	Sealed Lead Acid
Alkaline	Nickel Cadmium
Lithium	Nickel Metal-Hydride
	Lithium-ion
	Lithium-Polymer

Table 1.- Classification of different battery types in primary or secondary cells

For instance, silver coin or button cell batteries are lithium batteries since they are composed of lithium metal and, due to their irreversible chemical reaction, they are classified as primary cells. Also the well-known alkaline batteries, which can be easily found on store shelves, are classified into this group. The disposable nature of this kind of batteries means that there is no need of recharge control, protection circuity or fuel gauging, whereas the secondary batteries, they do need this actions in order to enhance their performance. Within this group we can found the Lithium-ion batteries, which are the ones that we are going to focus on throughout this document. This batteries are now widely used in today portable's world. At the end of this document, we are going to study the Lead-Acid batteries too, which are classified as secondary cells and are usually used in automotive applications or fixed installation due to their large size and weight.

1.2 OPERATION OF A BATTERY

A battery is a device composed by one or more electrochemical cells that converts the chemical energy contained in its active materials directly into electric energy by means of an oxidation reduction (redox) reaction, which consists in the transfer of electrons from one material to another through an electric circuit [2].

During the discharge of a battery, it operates like in Figure 1.1.a). When both terminals of the battery are connected to an external load (in this case a bulb), electrodes flow from the anode, which is oxidized, through the external load to the cathode, which is reduced due to the flow of electrodes. The electric circuit is completed in the electrolyte by the flow of anions and cations to the anode and cathode, respectively. The electrolyte is an ionic conductor that allow the transfer of charge, as ions, inside the battery between the anode and the cathode [2].

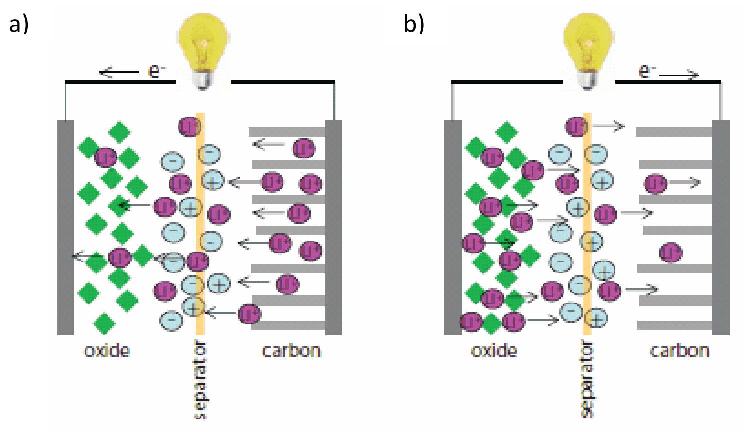


Figure 1.1.- a) Discharge of Li-Ion Battery; b) Charge of Li-Ion Battery

Source: *http://www.sigmaaldrich.com/technical-documents/articles/material-matters/ionic-liquids-for.html*

The operation inside the battery when charging is depicted in Figure 1.1.b). Now the current flow is inverted, with reduction occurring at the negative electrode and oxidation at the positive. To conclude, it is important to notice that the battery operation relies on the use of a pairs of metals that are capable of exchanging electrons.

1.3 MAIN PARAMETERS OF THE BATTERY

Each battery is characterised by the following parameters [1]:

a) Usable power (P = V*I):

It is obtained from the product of battery voltage (V) and the maximum current that it can tolerate (I). This usable power must be at least equal to the peak power so as to provide the electricity throughout all the operating range.

b) Stored Energy (KWh):

This parameter is going to determine the autonomy of the electric vehicle (EV) and the possibilities of recovering for a hybrid electric vehicle (HEV). The energy of the battery is expressed as a function of its capacity in ampere-hour (Ah) and its voltage.

c) State of charge of the battery (SOC):

One possible definition of the state of charge (SOC) could be the ratio of the remaining charge of the battery and the total charge while the battery is fully charged at the same specific standard condition. The SOC is often expressed in percentage, where 100% means fully charged and 0% means fully discharged [3].

1.4 REQUIREMENTS OF A BATTERY FOR THE ELECTRIC VEHICLE

The weight and volume of the batteries are some of the most important factors when choosing a battery. The electric vehicle (EV) must have a battery that meets the following requirements among others [1]:

a) A good mass energy (Wh/Kg): the quantity of energy stored per mass unit. It allows to define the autonomy of the battery.
b) A good power-to-weight ratio (W/Kg): The power delivered by a unit of mass of the battery.
c) A steady voltage which generate a regular performance.
d) A good autonomy.
e) A maximum battery lifespan, expressed in number of cycles (charge/discharge) that it can support. Battery lifespan is defined as the number of times that the battery can be restored until a level of energy superior to 80% of its nominal energy.
f) Less maintenance
g) Availability

1.5 BATTERY MODELLING

Research in the field of electric vehicle simulation, energy distribution and power control strategy, as well as in the estimation of batteries state of charge (SOC) and state of health (SOH) is experiencing an important increase. This growing interest in this field caused that the improvement of battery models accuracy, especially those concerning Lithium-ion batteries, has become a crucial objective.

This is the reason why in the literature there is a wide range of different approaches regarding the representation of battery behaviour using models with different degrees of complexity. Since the battery is a nonlinear system, the models usually used in electric vehicles can be classified into three different kinds:

1.5.1 ELECTROCHEMICAL MODELS

It is possible to achieve a high accuracy by using electrochemical models that aim to capture all the key behaviours of the battery. They are suitable for understanding the distributed electrochemistry reactions in the electrodes (such as, the reactions from Figure 1.2, assuming Li_yCoO_2 cathode and Li_xC_6 anode) and electrolyte. However, in order to describe the battery chemistry charge/discharge carrier mechanisms, they deploy a high number of partial differential equations (PDEs) with a large number of unknown parameters (see from equation (1.1) to (1.5)), which must be solved simultaneously with a high computational expense and a significant requirement of memory. In addition, they frequently run into over-fitting problems due to their poor model robustness under extrapolation [4], which generally precludes their use in real-time online control [5].

This kind of battery modelling tries to describe all the details of physics phenomenon that happens inside the battery. Figure 1.2 shows the anatomy of a Lithium-ion cell battery, which has four main components: the negative composite electrode connected to the negative terminal of the cell, the positive electrode connected to the positive terminal of the cell, the separator and the electrolyte.

Cathode: $Li_{1-y}CoO_2 + y\ Li^+ + y\ e^- \rightarrow LiCoO_2$

Anode: $Li_yC_6 \rightarrow C_6 + y\ Li^+ + y\ e^-$

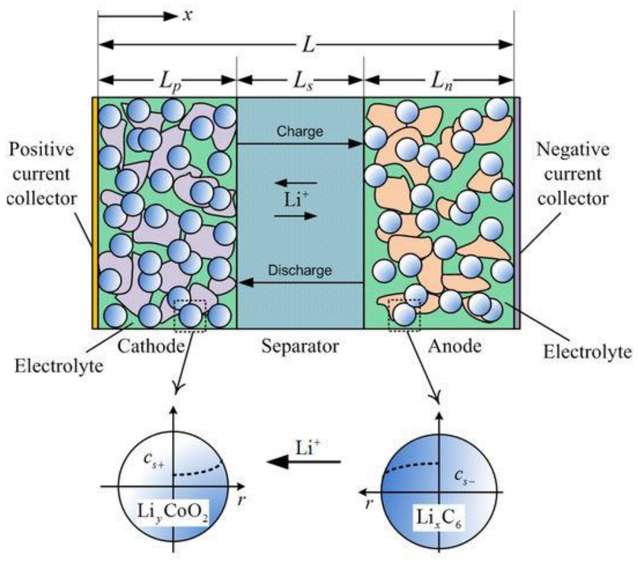


Figure 1.2.- Basic anatomy of a Lithium-ion cell

As it was stated previously in this section, the behaviour of the battery is explained by the electrochemical model with the following complex equations:

- <u>Transport in the solid phase:</u>

The partial differential equation (1.1) describes the solid phase Li^+ concentration in a single spherical active material particle in solid phase:

$$\frac{\partial c_s}{\partial t} = \frac{D_s}{r^2}\frac{\partial}{\partial x}\left(r^2 \frac{\partial c_s}{\partial r}\right) \tag{1.1}$$

Where D_s is the Li^+ diffusion coefficient in the intercalation particle of the electrodes.

- <u>Transport in electrolyte:</u>

The Li^+ concentration in the electrolyte phase changes due to the variations in the gradient diffusive flow of Li^+ ions and is described by the following PDE:

$$\epsilon \frac{\partial c_e}{\partial t} = \frac{\partial}{\partial x}\left(D_{eff}\frac{\partial c_e}{\partial x}\right) + a\,(1+t^+)\,j \tag{1.2}$$

Where ϵ is the volume fraction, D_{eff} is the Li^+ diffusion coefficient in the electrolyte, a is the specific surface area of electrode and it is equal to $\frac{3}{R_s}\left(1-\epsilon-\epsilon_f\right)$ (being ϵ_f the volume fraction of fillers and R_s the radius of intercalation of electrode), t^+ is he Li^+ transference constant in the electrolyte, and j is the wall-flux of Li^+ on the intercalation particle of electrode.

- <u>Electrical potentials:</u>

Change conservation in the solid phase of each electrode is described by Ohm's law (1.3). In the electrolyte phase, the electrical potential is described by combining Kirchhoff's law and Ohm's law (equation (1.4)).

$$\sigma_{eff}\left(\frac{\partial^2}{\partial x}\Phi_s\right) = aFj \tag{1.3}$$

$$-\sigma_{eff}\left(\frac{\partial \Phi_s}{\partial x}\right) - \kappa_{eff}\left(\frac{\partial \Phi_e}{\partial x}\right) + \frac{2\kappa_{eff}RT}{F}\,(1-t^+)\frac{\partial \ln(c_e)}{\partial x} = J \tag{1.4}$$

Where σ_{eff} is the effective electronic conductivity ($\sigma_{eff} = \sigma\left(1-\epsilon-\epsilon_{eff}\right)$, being σ the electronic conductivity in solid phase), K_{eff} is the effective ionic conductivity of the electrolyte, and J is the applied current density.

- <u>Butler-Volmer kinetics:</u>

Equation (1.5) describes the relationship between the current density, concentrations and over-potential:

$$j = k\left(c_{s,max} - c_{s,surf}\right)^{0,5}\left(c_{s,surf}\right)^{0,5}(c_e)^{0,5}\left(exp\left(0.5\frac{F\mu}{RT}\right)\right) - exp\left(-0.5\frac{F\mu}{RT}\right) \tag{1.5}$$

Where k is the reaction rate constant, $\mu = \Phi_s - \Phi_e - U_0$ is the over-potential of intercalation reaction, U_0 is the open-circuit potential for the electrode material (usually obtained from curve-fitting on experimental measurement), $c_{s,max}$ is the maximum concentration of Li^+ ions in the intercalation particles of the electrode and $c_{s,surf}$ the concentration of Li^+ ions on the surface of the intercalation particles of the electrode.

Shepherd model is one of the most widely used electrochemical models, for instance it is commonly employed for the hybrid electric vehicle (HEV) description. This model describes directly the electrochemical behaviour of the battery in terms of voltage and current [1].

$$U_k = U_0 - R_0 I_k + \frac{K_k}{z_k} \qquad (1.6)$$

Where k is a time index, U_k is the model voltage, U_0 is the open circuit voltage, R_0 is the internal ohmic resistance of the battery, K_k is the polarization resistance (expressed in ohms), I_k is the instantaneous current (amps), and z_k is the cell SOC.

1.5.2 MATHEMATICAL MODELS

In general, these models are so abstract that they cannot be used to develop a specific model, but they are still considered as a useful resource for system designers. They employ empirical equations or mathematical methods to predict the system level behaviour and system evolution, as well as its properties, such as the autonomy of a battery or its capacity [1].

1.5.3 ELECTRICAL EQUIVALENT CIRCUIT MODELS

Electrical equivalent circuit (EEC) models consist of a combination of voltage sources, resistors and capacitors. They, like the other models, try to model the battery behaviour. They are based on the reproduction of the dynamic characteristics and working principles of the battery using circuit theory. Their accuracy lies within 1-5% and their low computational intensity makes them really accurate for real-time simulation use [5].

In document [6], they carried out an experimental study which allow them to conclude that an improved Thevenin circuit model, named dual polarisation (DP) model, was the best model in terms of compromise between accuracy and computation time. Due to the great performance of this model, we decided to use it in order to develop all our study. The selected model is shown below:

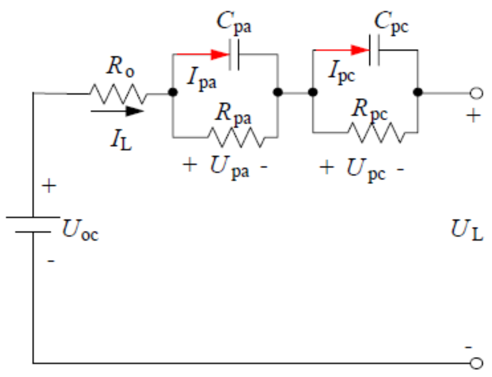


Figure 1.3.- Schematic diagram for the DP model [6]

This model allow to refine the description of polarisation characteristics of the battery and simulate the concentration polarisation and the electrochemical polarisation separately, which leads to an improved simulation at the moments of end of charge or discharge compared to the Thevenin model.

The DP model is composed of three parts [6]:

- The open-circuit voltage (U_{oc}), which is reproduced by a voltage source.
- Internal resistances such as the ohmic resistance R_0 and the polarisation resistances, which include R_{pa} to represent the effective resistance characterising electrochemical polarisation and R_{pc} to represent the effective resistance characterising concentration polarisation.
- The effective capacitances like C_{pa} and C_{pc}, which are used to describe the electrochemical polarization and the concentration polarization separately and to characterise the transient response during transfer of power to/from the battery, more concretely the network R_{pa} and C_{pa} captures the short transients (STC) and the network R_{pc} and C_{pc} captures the long ones (LTC) [5].

U_{pa} and U_{pc} are the voltages across C_{pa} and C_{pc} respectively. I_{pa} and I_{pc} are the outflow currents of C_{pa} and C_{pc} respectively. The equations that describes the electrical behaviour of this circuit will be expressed in the section 2.1.

1.6 MAPLESIM ENVIRONMENT

The most part of this work has been developed thanks to the modelling, simulation and analysis tool MapleSim, which has, among others, a specific library for the batteries. It provided us with a wide range of different batteries models. This together with the different tools to create custom components has allow us to suit our modelling and simulation needs, and therefore fulfil the goals of this study.

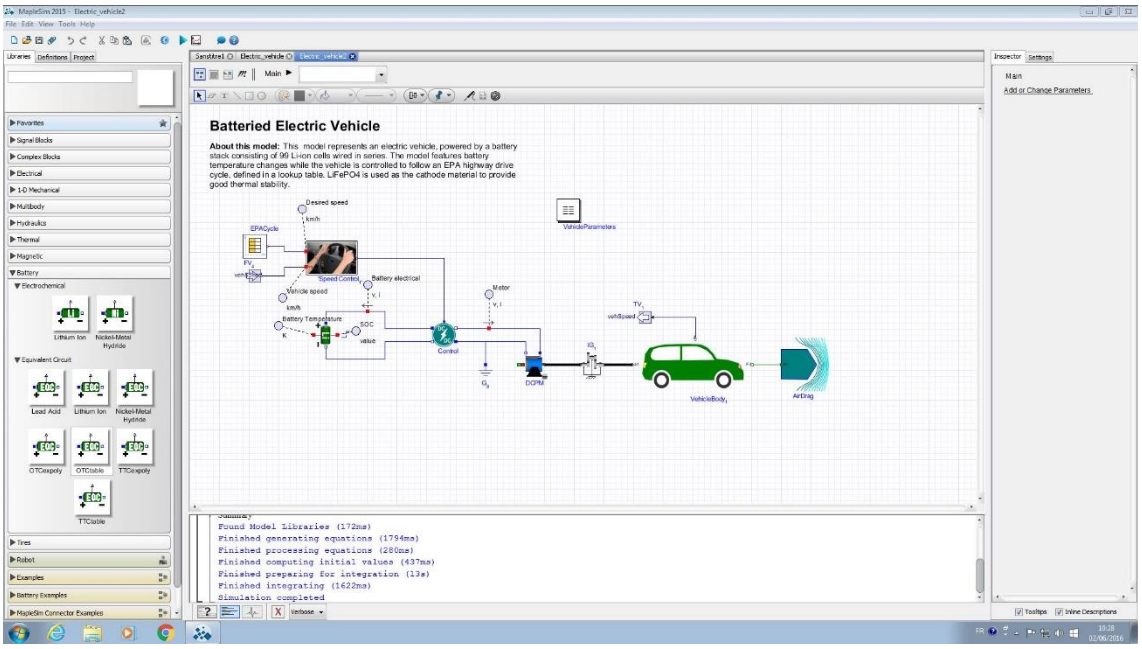
The integration of MapleSim and Maple offers more freedom in order to develop your models. Thanks to its flexibility, you are no longer restricted to built-in components or analyse. With its complete programming and analysis environment, it is possible to run simulations, customise analyses or script entirely new ones, perform optimisations, develop advanced symbolic control laws and investigate models in ways that are not possible with other tools [7]. It also allows to build component diagrams that represent physical systems in a graphical form, which is of great help for many engineers that do not find many existing simulation tools intuitive for physical modelling. Using both symbolic and numeric approaches, MapleSim automatically generates model equations from a component diagram and runs high-fidelity simulations [8].

After automatically generating these equations, MapleSim tries to simplify them with symbolic techniques that include index reduction, differential elimination, separation of independent systems, and elimination of redundant systems. The two main benefits of symbolic simplification are the following [8]:

- By symbolically resolving algebraic loops and through reducing the complexity of DAEs, symbolic simplification makes many problems, which previously were intractable, numerically solvable.

- The simplified equations are provided to the numerical solvers in a computationally efficient form. This reduces the total simulation time, in some cases, by many orders of magnitude.

Computational efficiency is particularly important for studies that requires hardware-in-the-loop (HIL) simulation, such as the implementation of a batteries in the electric vehicle, because it allows to develop higher fidelity models while the real-time performance remains accurate.



Batteried Electric Vehicle

About this model: This model represents an electric vehicle, powered by a battery stack consisting of 99 Li-Ion cells wired in series. The model features battery temperature changes while the vehicle is controlled to follow an EPA highway drive cycle, defined in a lookup table. LiFePO4 is used as the cathode material to provide good thermal stability.

```
Summary
Found Model Libraries (172ms)
Finished generating equations (1794ms)
Finished processing equations (280ms)
Finished computing initial values (437ms)
Finished preparing for integration (13s)
Finished integrating (1622ms)
Simulation completed
```

Figure 1.4- Screenshot of MapleSim, showing all the different possibilities concerning battery simulation

In Figure 1.4, we can see the appearance of MapleSim software. On the left-hand side of the screenshot, it is shown the two different approaches of battery modelling (electrochemical and EEC) that it allows, and some batteries of different nature that are already implemented in the software. As it was stated previously, during the development of this study the electrochemical and EEC models of the Lithium-ion battery were used, as well as the EEC model of the Lead-Acid battery, due to the fact that MapleSim does not include the electrochemical model of the latter.

2 MODELLING OF A LITHIUM-ION BATTERY

Firstly, we started with the comparison between the two kinds of models that MapleSim allow us to use in order to describe the performance of a Lithium-ion battery. These models are the electrochemical model and the electrical equivalent circuit (EEC) model, also known as equivalent circuit models (ECM), which has been introduced in the section 1.5.

So as to accomplish this task, we took the electrochemical model that MapleSim included in its battery library as a reference and we tried to obtain the same output with the electrical equivalent circuit (EEC) model. The equation that describe the behaviour of the EEC model in MapleSim is the following:

$$V_{batt} = N_{cell} \cdot (V_{oc} - V_{Rint} - V_{RC1} - V_{RC2} - \cdots - V_{RCn}) \qquad (\ 2.1\ )$$

Where V_{batt} is the terminal voltage, N_{cell} is the number of cells number of cells in series that compose the stack, in our study we considered a Lithium-ion battery with one cell ($N_{cell} = 1$) for almost every simulations, except in section 4.5. V_{oc} corresponds to the open circuit voltage, V_{Rint} is the voltage drop across the internal resistance (R_{int}) and V_{RCn} is the voltage drop across the n-th RC network. In this case, as we have established in section 1.5.3, we considered an electrical circuit with two RC networks, therefore its terminal voltage would be characterised by the equation (2.2):

$$V_{batt} = N_{cell} \cdot (V_{oc} - V_{Rint} - V_{RC1} - V_{RC2}) \qquad (\ 2.2\ )$$

The layout of our model is presented in *Figure 2.1*. Each subsystem contains the electrochemical model (the one on the right) or the electrical equivalent circuit model (the left one). The content of each subsystem is the same excepting the battery model used. Figure 2.2 shows the EEC model subsystem.

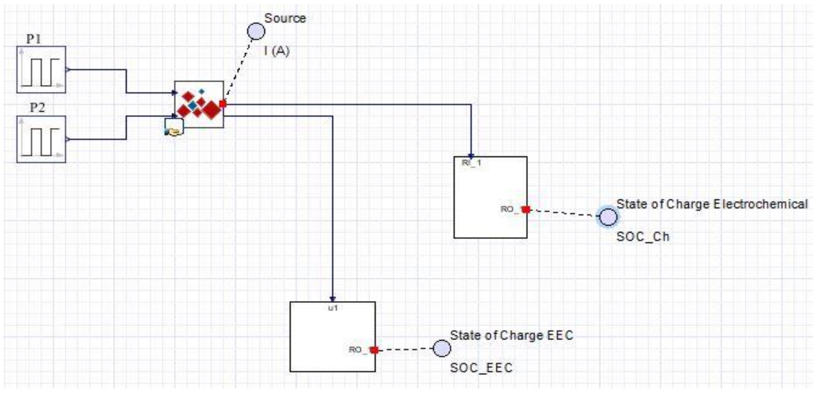


Figure 2.1.- Layout of the model developed in MapleSim

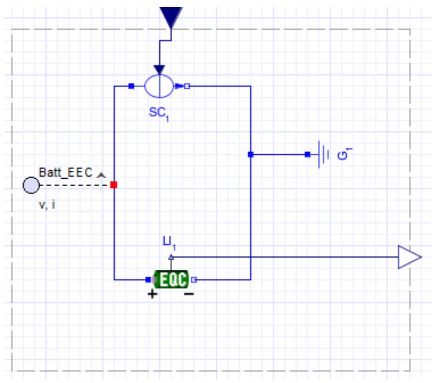
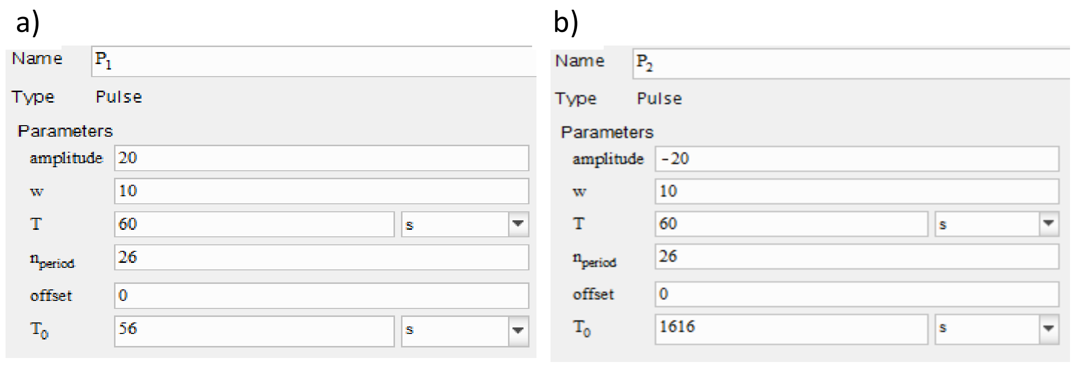


Figure 2.2.- Content of the EEC model subsystem

Each source (P_1 and P_2) was configured to work in different periods of time. In order to achieve this, the configuration of the sources is shown in Figure 2.3. As a result, the input current obtained consist of a pulse train which firstly discharge the battery and then charge it, as depicted in Figure *2.4*.



a)

Name	P_1	
Type	Pulse	
Parameters		
amplitude	20	
w	10	
T	60	s
n_{period}	26	
offset	0	
T_0	56	s

b)

Name	P_2	
Type	Pulse	
Parameters		
amplitude	-20	
w	10	
T	60	s
n_{period}	26	
offset	0	
T_0	1616	s

Figure 2.3.- Configuration of the source that: a) discharge the batteries; b) charge the batteries

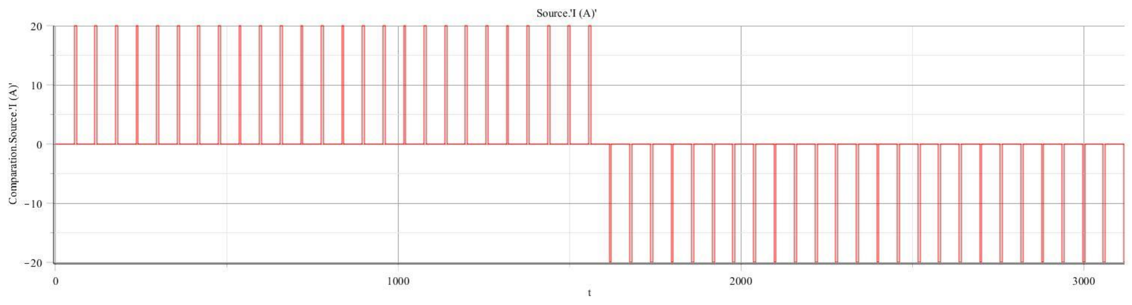


Figure 2.4.- Input current: pulse train with an amplitude of -20 A, T=60 sec. and a width of 10% of the period

During the 90% of the period the value of the current is 0 A and the remaining 10% the current is ±20A depending if we are in the discharging or charging period. Using this current as the input of our system, we obtain a comparison between the outputs of both models, electrochemical and EEC models, depicted in Figure 2.5 and Figure 2.6.

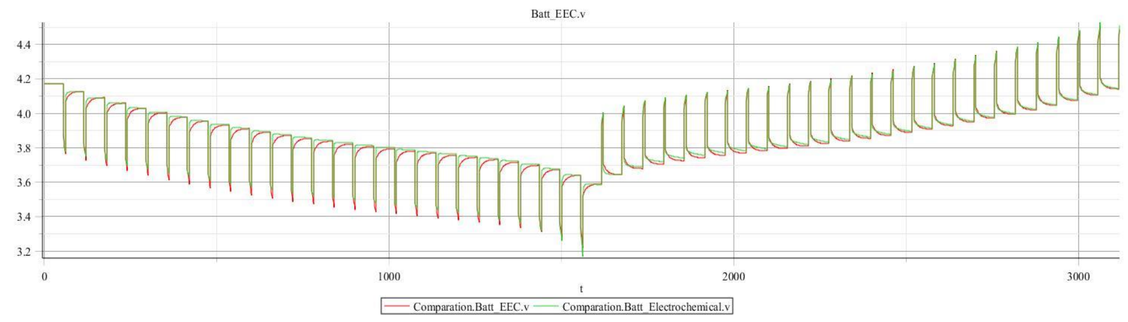


Figure 2.5.- Output: Terminal voltage of the electrochemical model (green) and the EEC model (red)

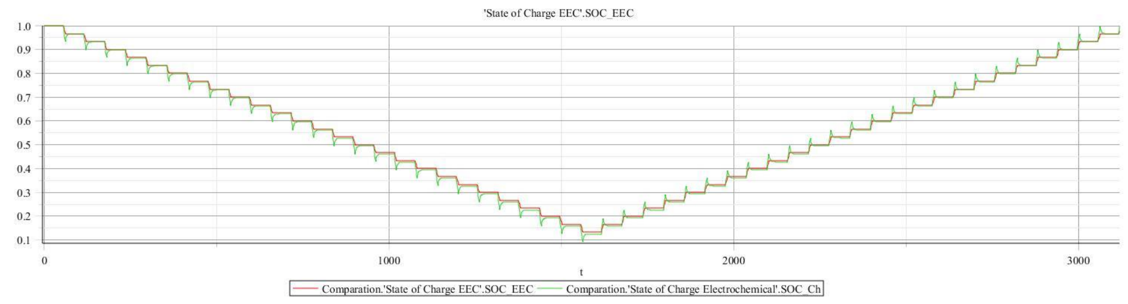


Figure 2.6.- Output: State of Charge of the electrochemical model (green) and the EEC model (red)

It can be seen that the results obtained from both models are very similar. Nevertheless, there is an estrange behaviour in the electrochemical one. For instance, concerning the *Figure 2.6*, when we study the state of charge of the battery, it is of the utmost importance to bear in mind that it depends on the current and the temperature. Therefore, if the input current is equal to zero the value of the state of charge must remain constant because these models were configured as isothermals, which means that the temperature remains constant during all the simulation. However, we can verify that the state of charge of the electrochemical model does not follow this behaviour.

On the other hand, in *Figure 2.5* it is represented the terminal voltage of both models. Here, the EEC model output is also more similar to the terminal voltage of a real battery, due to the fact that in the electrochemical model, when the current is zero the value of this output increase very abruptly and decrease a little bit later in the discharging period and vice versa when charging. Whereas for EEC model, we can clearly distinguish the contribution of the different components of the electrical circuit (*Figure 1.3*), which allow us to track the transients of the battery. When the input current becomes zero, firstly, there is a sudden increase of the output voltage due to the internal resistance, then the voltage keeps increasing but in a slower rate due to the parallel RC networks, describing the transient response. When charging it would be the other way round, firstly there is a sudden drop due to the internal resistance and then a moderate decrease due to the RC networks.

In order to make sure that the EEC model tracks better the battery behaviour we carried out another simulation with both models, but in this case the input current would be a constant current (*Figure 2.7*).

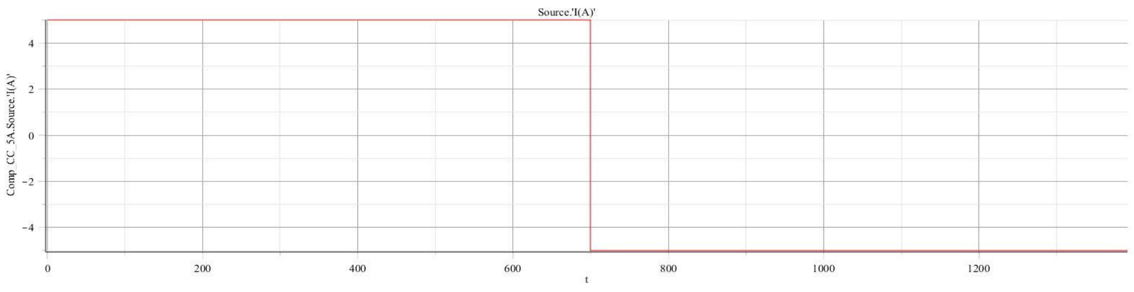


Figure 2.7.- Input current: Constant current with an amplitude of +/-5 A

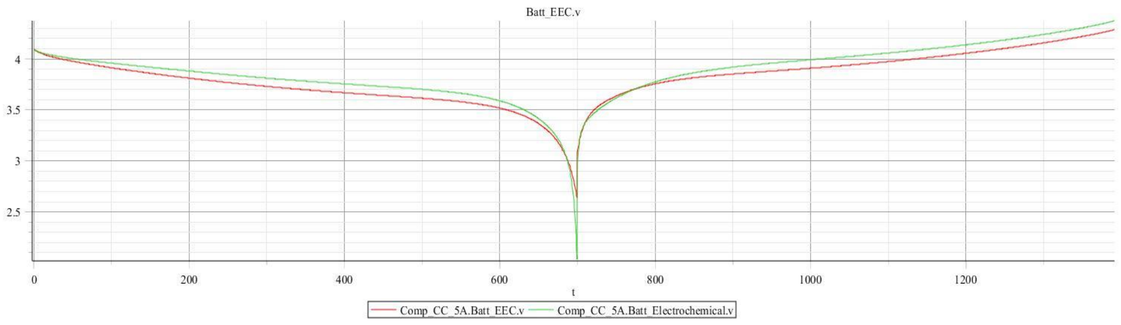


Figure 2.8.- Output: Terminal voltage of the electrochemical model (green) and the EEC model (red)

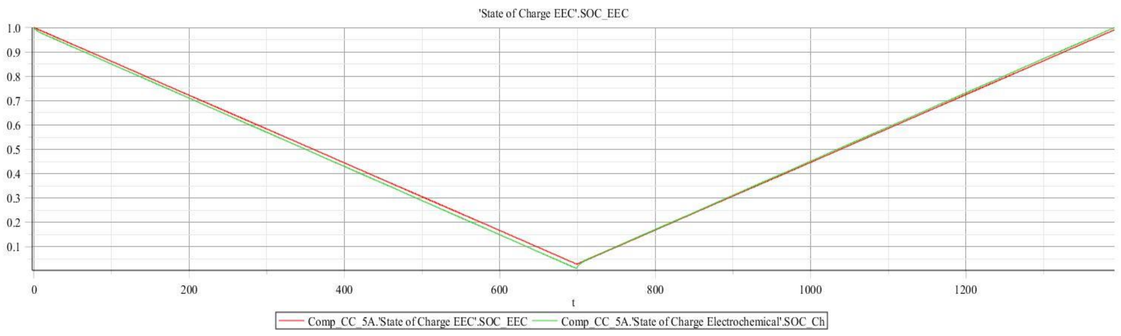


Figure 2.9.- Output: State of Charge of the electrochemical model (green) and the EEC model (red)

The behaviour of both models is almost the same for this step input with an amplitude of 5A when discharging and -5A when charging. Thus, taking everything into account we chose the EEC one as the reference battery model to carry out our study, as we verified that it was the one that describe the battery behaviour with the highest fidelity.

2.1 IMPLEMENTATION OF THE BATTERY MODEL IN MATLAB ENVIRONMENT

Once the EEC model was chosen as the reference model from which our study could be developed, we wanted to make sure that the parameters we were going to use in Matlab would allow us to have an accurate representation of the performance our battery model. The parameters used in Matlab were obtained from the EEC model that it was modelled in MapleSim. The value of each parameter depends on SOC, following the form of the equation (2.3). Where the variable "parameter" refers to the internal resistance (R_0) or any of the components of the RC networks (R_1, C_1, R_2 or C_2).

General form:

$$parameter = k_1 \cdot e^{k_2 \cdot SOC} + k_3 \qquad (2.3)$$

Particularised for each parameter:

$$R_{0,k} = 0{,}11 \cdot e^{-50 \cdot SOC_k} + 0{,}0075$$

$$R_{1,k} = 0{,}05 \cdot e^{-29 \cdot SOC_k} + 0{,}0074$$

$$R_{1,k} \cdot C_{1,k} = 3{,}5 \cdot e^{-10 \cdot SOC_k} + 10{,}5$$

$$R_{2,k} = 1 \cdot e^{-150 \cdot SOC_k} + 0{,}008$$

$$R_{2,k} \cdot C_{2,k} = -500 \cdot e^{-20 \cdot SOC_k} + 710$$

The open circuit voltage (V_{oc}) is also obtained from MapleSim and follows the same form of (2.3):

$$V_{OC_k} = 0{,}1958 * e^{1{,}332 \cdot SOC_k} + 3{,}429703601 \qquad (2.4)$$

So as to implement this model in Matlab environment it was necessary to create a linear state-space model defined by this equations:

$$\dot{x} = Ax + Bu + w \qquad (2.5)$$

$$y = Cx + Du + Hv \qquad (2.6)$$

We needed to rewrite the equations that described the electrical behaviour of the battery model into the matrix form, which fitted with the state-space formulation. This EEC model was based on the Dual Polarised circuit introduced in section 1.5.3. Our particular model is represented in Figure 2.10.

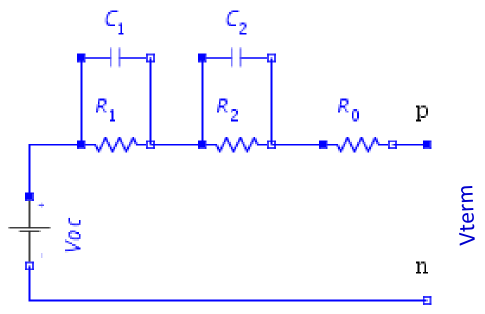


Figure 2.10.- Electrical equivalent circuit Model of the Battery

The controllable source represents the open circuit voltage (V_{oc}). The function of the rest of the components has been detailed in section 1.5.3. The electrical behaviour of the previous circuit could be described by the following system of equations:

$$\begin{cases} \dot{V}_1 = -\dfrac{V_1}{R_1(S)\cdot C_1(S)} + \dfrac{I_{batt}}{C_1(S)} \\ \dot{V}_2 = -\dfrac{V_2}{R_2(S)\cdot C_2(S)} + \dfrac{I_{batt}}{C_2(S)} \\ V_{term}(S,I) = V_{oc}(S) - V_1 - V_2 - R_0\cdot I_{batt} \end{cases} \qquad (2.7)$$

Bearing in mind that we were in a simulation environment, we chose the Coulomb counting method (2.8) in order to keep track of the SOC of our model implemented in Matlab as it assures 100% accuracy for ideal batteries working in this environment [5]:

$$S(I) = S_{init} + \frac{1}{C_{use}\cdot 3600}\cdot\int_0^t I_{batt}(t)\,dt \qquad (2.8)$$

Where S_{init} represents the SOC at the initial time t_0, I_{batt} (t) the current traversing the battery in Amperes (assumed to be positive when discharging and negative when charging) and C_{use} is the available usable capacity of the battery in Ampere-hour, which changes with the service life (in our particular case C_{use} = 1Ah and it will remain constant during most of the simulations of this study, except for the simulations concerning the state of health of the battery). However, if we take into consideration the losses that occur while charging and discharging and also during storing periods, the equation (2.8) must be slightly modified [9]:

$$S(I) = S_{init}[1 - \frac{\sigma}{24}(t - t_0)] + \frac{\eta}{C_{use} \cdot 3600} \cdot \int_0^t I_{batt}(t)\, dt \qquad (2.9)$$

Where σ is the self-discharge rate, which depends on the accumulated charge and the battery state of health (SOH). A value of 0,2% per day is recommended for this parameter; ɳ is the coulombic efficiency[1], which it is assumed to be one for discharging and less than or close to one when charging, in order to reflect the fact that only a fraction of the input energy is restored. It depends on the technology used for the battery and other variables such as the temperature, the charging/discharging current, the state of charge (SOC) and the state of health (SOH). In our particular case, as we work in the ideal conditions that characterised the simulation environment, it is assumed to be constant and equal to one in both charging and discharging process, and the self-discharge rate is going to be fixed to 0% for the same reason [10].

In order to adapt this equations to the state-space formulation we defined the following matrix based on the equations (2.7) and (2.8):

$$A = \begin{bmatrix} -\frac{1}{R_1 \cdot C_1} & 0 & 0 \\ 0 & -\frac{1}{R_2 \cdot C_2} & 0 \\ 0 & 0 & 0 \end{bmatrix} \; ; \; B = \begin{bmatrix} \frac{1}{C_1} \\ \frac{1}{C_2} \\ -\frac{1}{C_{use} \cdot 3600} \end{bmatrix} \; ; \; C = \begin{bmatrix} -1 & -1 & \frac{\partial V_{OC}}{\partial SOC} \end{bmatrix};$$

$$D = [-R_0] \; ; \; H = [1] \; ; \; x = \begin{bmatrix} V_1 \\ V_2 \\ SOC \end{bmatrix} \; ; \; u = [I_{batt}] \; ; \; y = [V_{term}] \qquad (2.10)$$

At first, we started implementing a linear model of the reference battery. Therefore, all the nonlinear expressions were linearized. Concerning the circuit parameters defined in (2.3), we supposed that their value remained constant and equal to their initial value when the battery was fully charged (SOC=1) during all the simulation. Whereas the expression of the open circuit voltage (V_{oc}) was linearized using the curve fitting tool from Matlab and an excel file with data from the battery model in MapleSim so as to fit these data with a linear curve. The linearized expression of the V_{oc} is defined as follows:

$$V_{OC_k} = 0.5552 * SOC_k + 3{,}525 \qquad (2.11)$$

[1] Coulombic efficiency: The ratio of the number of charges that enter to the battery during charging and those that can be extracted from the battery during discharging.

Once our model in Matlab was defined, we proceeded to verify if this system was able to reproduce the same behaviour as that of our battery model in MapleSim. In order to carry out this verification we introduced the same excitations for the two systems and compared their outputs. For instance, using a constant current of 1A as the input, we obtain the following outputs:

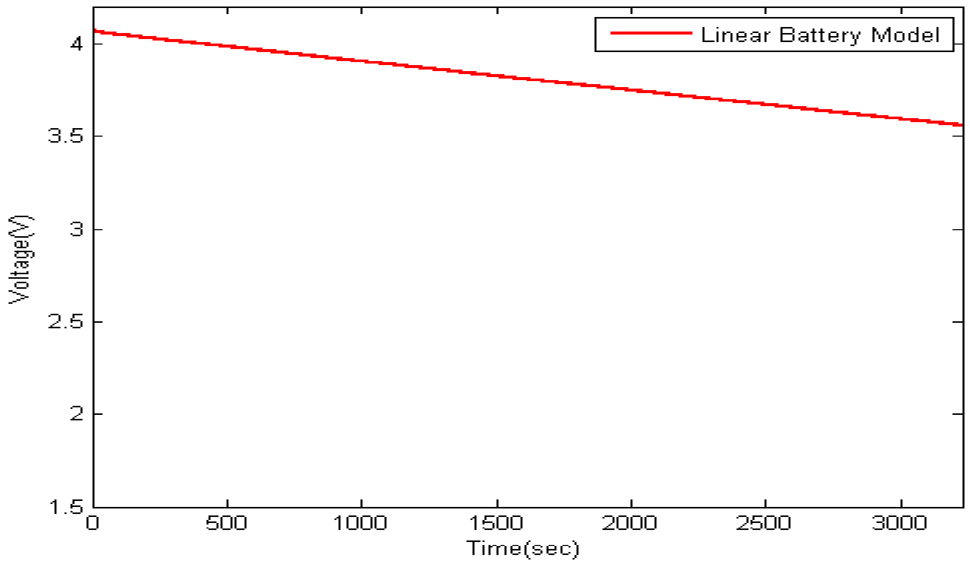


Figure 2.11.- Battery terminal voltage response in Matlab using a constant current of 1A

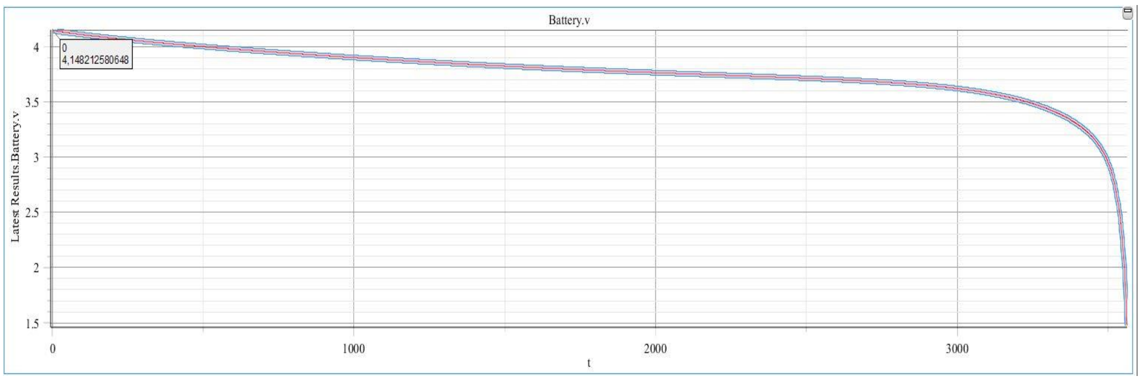


Figure 2.12.- Battery Voltage response in MapleSim using a constant current of 1A

Although at a first glace it may seem that these two systems do not have any similarities between them, we have to take into account that when we transform a model into a linear state-space model we are linearizing the model, so the system will lose all the properties related to its nonlinearities. Therefore, we have to analyse the area where the model has a linear behaviour. If we do so, we are going to find that both systems have almost the same behaviour in their flatter part.

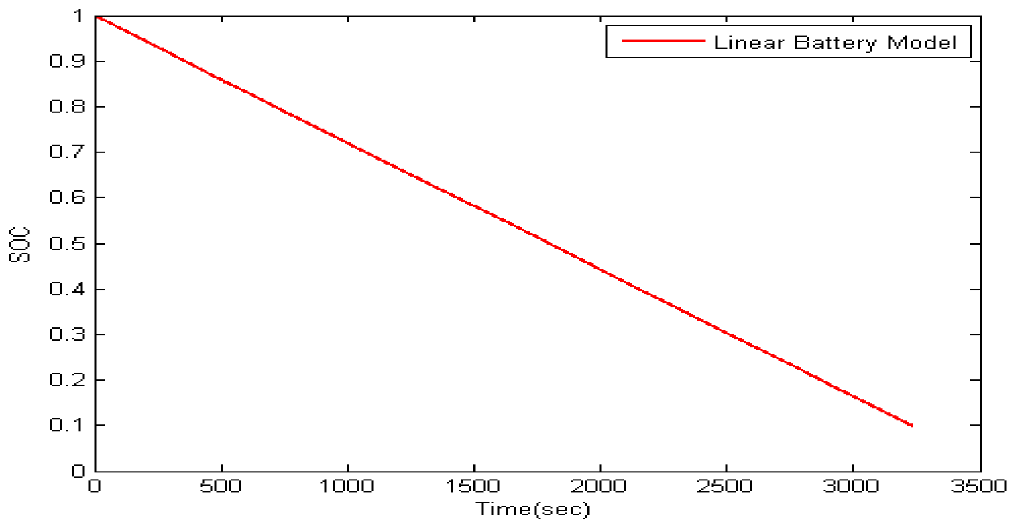


Figure 2.13.- State of charge (SOC) response in Matlab using a constant current of 1A

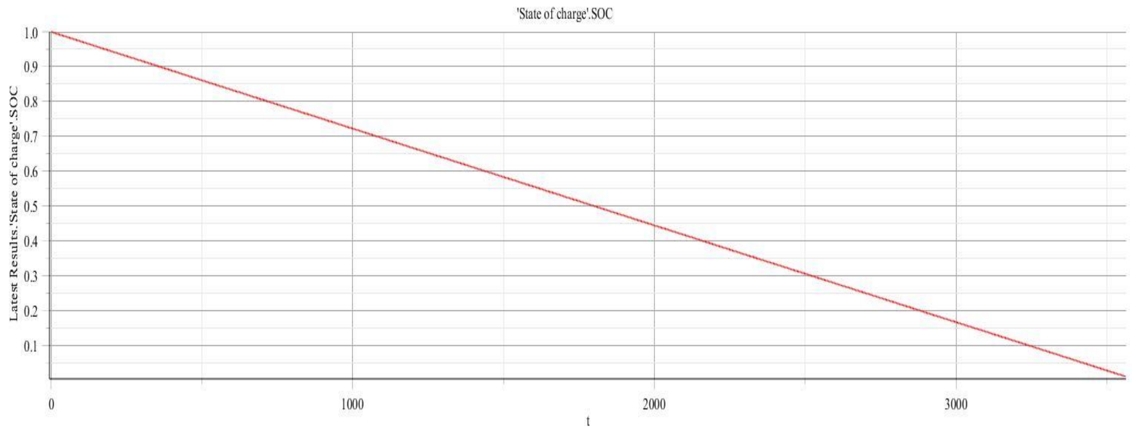


Figure 2.14.- State of charge (SOC) response in MapleSim using a constant current of 1A

It can be seen that the battery state of charge in both cases decrease at the same rate. In order to make sure that we could continue working with this model in Matlab, we introduced a new input current to the system, consisting of a square-wave pulse train with an amplitude of 10A and a period of 60 seconds during 679 seconds until the 10% of the SOC was reached, as it is shown in Figure 2.15.

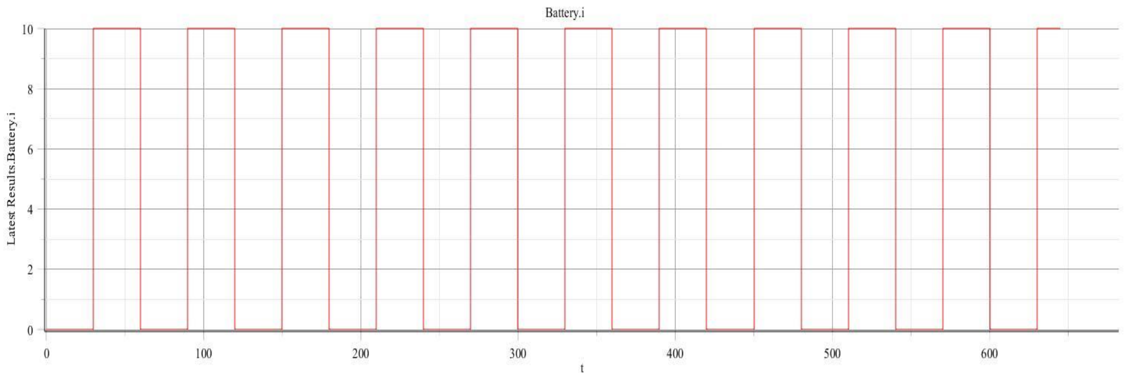


Figure 2.15.- Input current: train of square-wave pulses with an amplitude of 10A and T=60 sec

Using this current as the battery model input, we obtain the following responses, in both Matlab and MapleSim. We can verify that we achieve a great accuracy in the estimation, being the linearization of the system in Matlab the reason of the small differences that exist between both models.

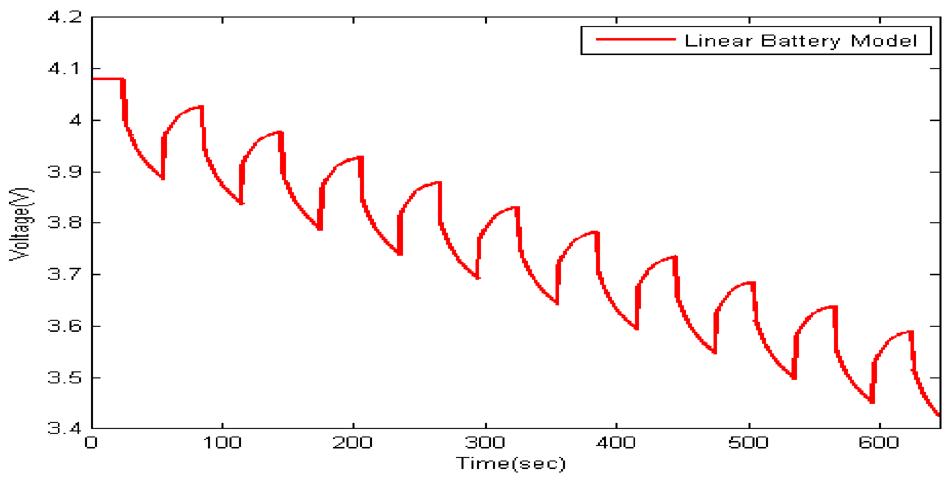


Figure 2.16.- Battery Terminal Voltage response in Matlab

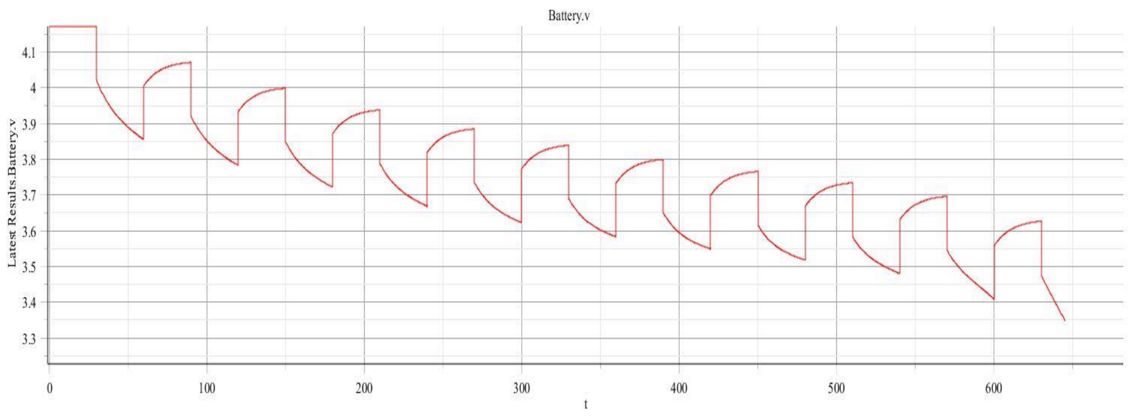


Figure 2.17.- Battery Terminal Voltage response in MapleSim

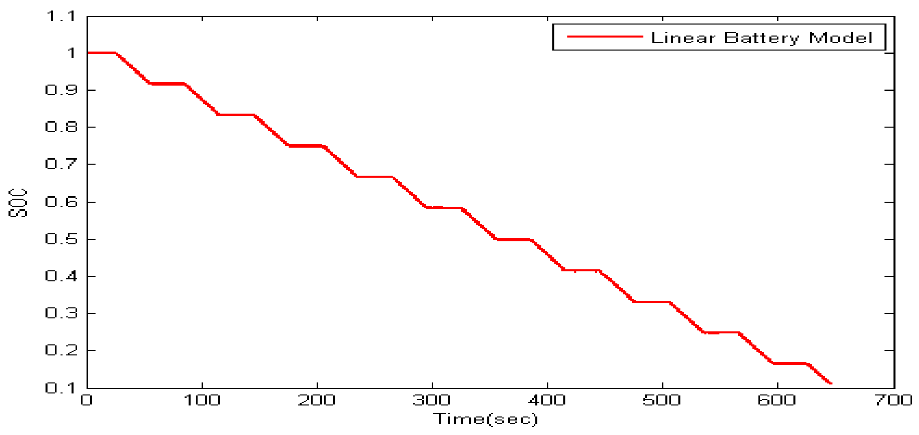


Figure 2.18.- State of charge (SOC) response in Matlab using an input current that consist of train of square-wave pulses

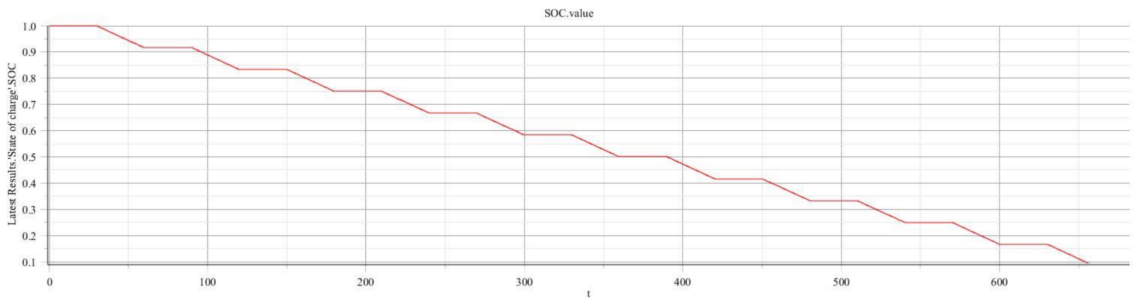


Figure 2.19.- State of charge (SOC) response in MapleSim using an input current that consist of train of square-wave pulses

From the previous graphics it is visible that this model is highly accurate if we work with a battery that has a linear behaviour. However, due to the inherent nonlinear nature of the battery behaviour, the nonlinear model was also implemented in Matlab, as it accounts for the nonlinearities. This model was developed considering that R_0, R_1, C_1, R_2, C_2 and V_{oc} depend nonlinearly on SOC, as it was defined in equations (2.3) and (2.4).

As we wanted to prove that this model accounts for the nonlinearities of the system, we used the same constant current as that which has been previously used to show that the linear system gives a good estimation of the terminal voltage only in the flatter area of the curve. Making a comparison between the outputs depicted in Figure 2.20 and Figure 2.12, we can confirm that with this model we also keep track of the nonlinearities of the battery model to some extent.

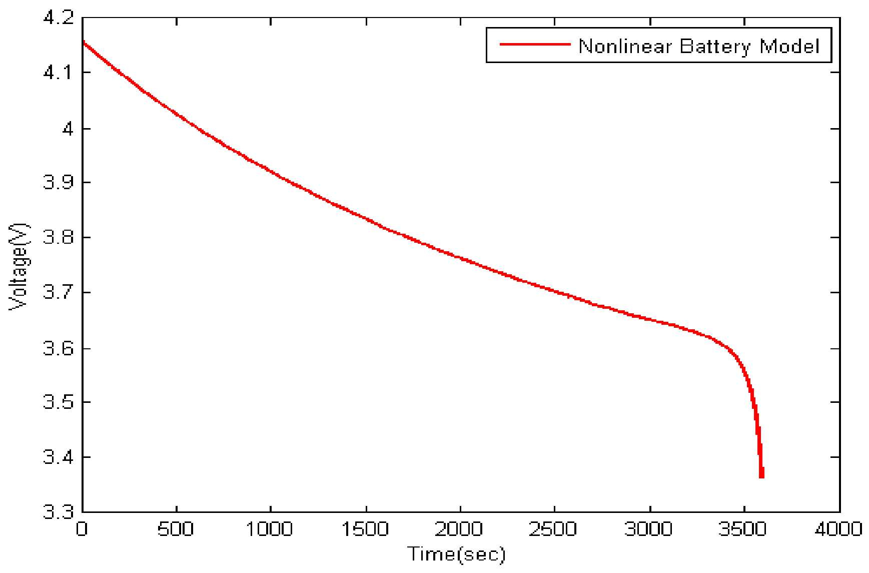


Figure 2.20.- Battery terminal voltage of the nonlinear system implemented in Matlab using a constant current of 1A as the input of the system

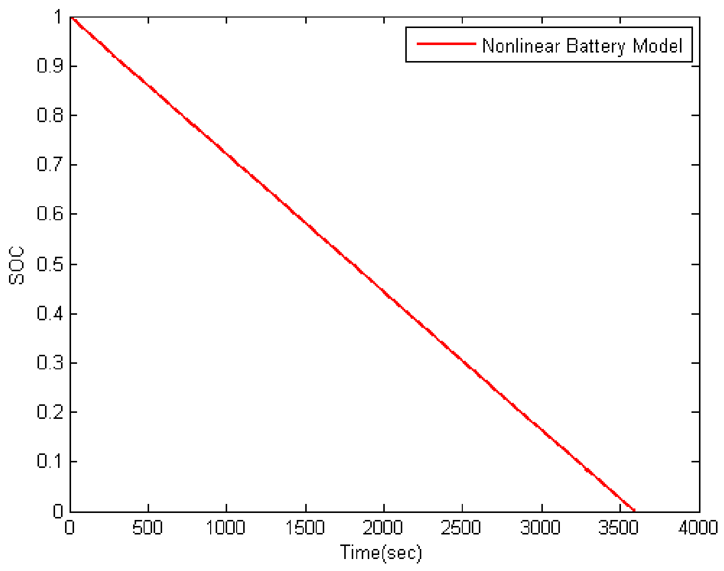


Figure 2.21.- SOC of the nonlinear system implemented in Matlab using a constant current of 1A as the input of the system

All things considered, we decided to start the simulations in Matlab with the linear system in order to develop the Simple Kalman Filter (SKF), due to the fact that it has been proved that it operates better when it works with linear systems. Later on, we would change to the nonlinear system so as to implement the Extended Kalman Filter, as we will see in chapter 3.

3 STATE OF CHARGE ESTIMATION ALGORITHM

3.1 STATE OF THE ART

State of charge estimation of commercial batteries can be done by many different methods in electrical chemistry laboratory like coulometric titration technique [3]. But this estimation is quite challenging without destruction of the battery or interruption of the battery power supply, especially the applications which require online estimations. Currently there has been intensive study on SOC estimation algorithm. Below a short review of some of them is presented [3]:

- Discharge test method

  This test could precisely find the remaining charge of the battery and then the SOC under controlled conditions, i.e., specified discharge current and ambient temperature. Its major drawback is that it is a time-consuming method and after the test the battery have no power, hence this method is not useful for the online applications of the batteries, reducing its utility to the laboratory environment only.

- Coulomb counting (Ampere-Hour integral) method

  This is the most simple and general method to obtain the battery SOC. It is characterised by the equation (2.8) that has been defined before.

  If the initial estimation of SOC is relatively precise, the results of the Coulomb counting method are quite satisfactory. Nevertheless, it has several disadvantages:

  i) It cannot get the precise initial SOC automatically, so it is highly important to have an accurate initial estimation of SOC in order to obtain a precise SOC estimation.
  ii) The Coulombic efficiency (η) can be influenced by the operation state of the battery, such as SOC, temperature, etc., which are difficult to measure and then produce cumulative effects on SOC error.
  iii) Its dependence on the precision of the current sensor that will result in cumulative effects which will influence on the precision of SOC.

  Therefore, the Coulomb counting method alone cannot meet the requirement of SOC precision.

- Open circuit voltage (OCV) method

The one to one correlation between OCV and SOC make this method an effective one to estimate SOC of Lithium-ion batteries, because it allows us to be sure that when the battery has reached balance after adequate resting, the OCV corresponds to 100% of the SOC.

This method give us a high precision SOC estimation but, on the other hand, the relaxation period requires a lot of time until the batteries reach balance. It usually take some time for them to recover from an operating state to a balanced state, this duration depends on the SOC and temperature of the battery among others. Thus, this method, if used alone, is suitable only for applications where the device is left idle, for instance, for the electric vehicles (EVs) case, this method will be useful only if they are parking rather than driving.

Moreover, careful consideration and research are needed as the OCV of some kinds of batteries depends on the charge/discharge process. For instance, the charge and discharge open circuit voltage of Lithium-ion phosphate (LFP) batteries experience the hysteresis phenomena as indicated in Figure 3.1 [11]:

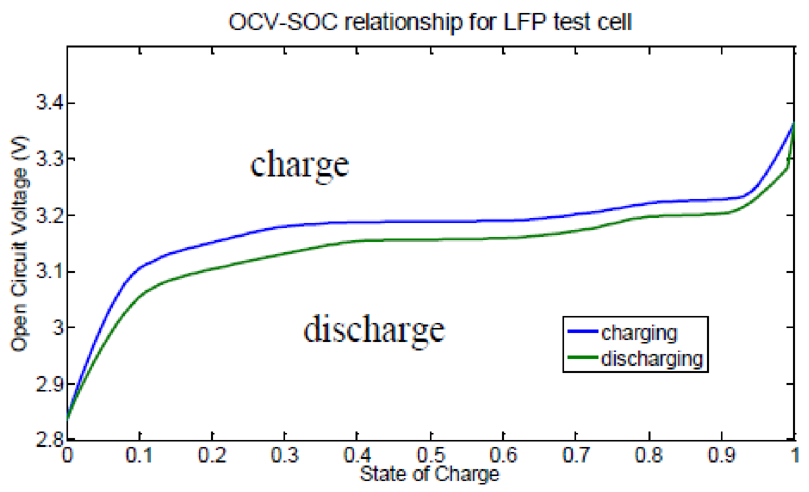


Figure 3.1.- Flat OCV-SOC curve for the LFP cell (20°C) after a 3-hour rest period [11]

Considering the hysteresis phenomena of LFP battery, it has been shown that the hysteresis is correlated with the relaxation time, with the level of hysteresis decreasing as the rest period increases. This phenomenon is depicted in the Figure 3.2 where the voltage was plotted with respect to SOC for different relaxation periods.

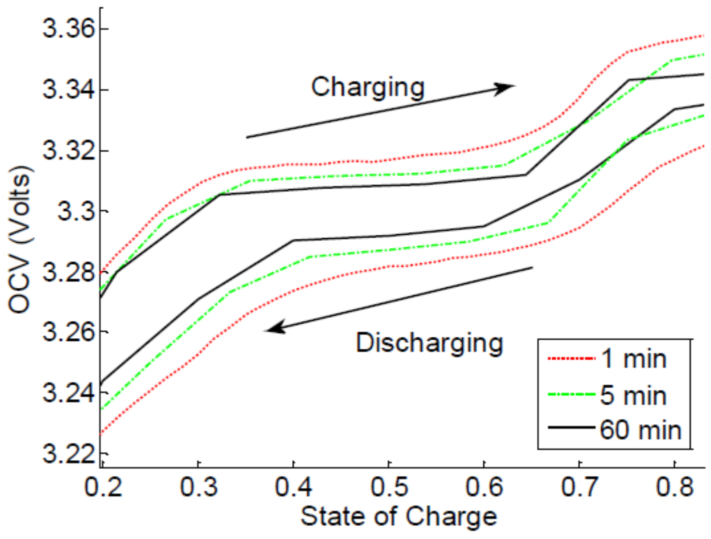


Figure 3.2.- Hysteresis decreases with the increase of the rest time in the multiple-step test conducted on the LFP cell at 20°C [11]

- Battery model-based SOC estimation method

The OCV method needs enough time resting to complete the relaxation period accurately, hence, it is not useful for online applications when the device is working, for example, while the electric vehicle is driving. In such cases, the construction of battery model in conjunction with OCV method is necessary in order to online estimate the OCV during operation. The most commonly used battery models include electrical equivalent circuit (EEC) model and electrochemical model, which have been introduced in section 1.5. Remind that the terminal voltage of the EEC model derived from the Thevenin Model, could be expressed as:

$$U_L = U_{OC} - U_{Th} - I_L R_0 \qquad (3.1)$$

Where U_L is the battery terminal voltage, the product $I_L R_0$ represent the voltage drop caused by the ohmic resistance, U_{OC} the battery OCV and U_{Th} is the voltage drop across the parallel RC networks. So it is easy to found the value of the OCV if the battery model parameters are known. The direct relation between OCV and SOC allow us to easily found the SOC by using an OCV-SOC look-up table.

For this method, the precision and complexity of battery model are very important. We have seen in the previous chapter that the desire to achieve a good compromise between accuracy and computation time, lead us to choose the EEC model composed of two RC networks among the wide variety of different battery models thanks to its accurate performance (with a mean error of 1.4%)

despite its low computational intensity. Whereas the electrochemical model, due to its high complexity, usually is only used for the battery performance analysis and battery design.

- Neural network model method

  It is based on the use of nonlinear mapping characteristics of the neural network so as to estimate the SOC. When building a model, the neural network method does not have to take into account specific details of a battery as it is suitable for all kinds of batteries. Nevertheless, it needs a great number of training sample data to train the method. Moreover, this method requires a lot of computations, which makes necessary to have powerful processing chips.

- Fuzzy logic method

  It is based on the simulation of the thinking of human beings by using the fuzzy logic on the basis of a great number of test curves, experience and reliable fuzzy logical theories. It eventually could be used to predict the SOC of the battery but it requires a deep understanding of the batteries themselves and a large number of computations.

- Integrated algorithm based on the two or more of the above methods

  Currently there are several integrated methods such as:

  - Simple correction integrated algorithm:

    It includes Ampere-Hour integrated algorithm with correction by open circuit voltage, Ampere-Hour integral method with SOC calibration after charging and so on. For batteries in pure electric vehicles:

    - The working conditions are simple: when the vehicles are moving, their batteries are mainly in a discharge state, and when the batteries are being charged in a charging station, the batteries are in a charge state. Moreover the hysteresis of the open circuit voltage is easy to estimate.
    - Thanks to the large capacities of the batteries the errors of the Ampere-Hour integral are relatively low.
    - The possibility to be fully charge is great

  All the above things considered, we can affirm that the Ampere-Hour method with initial SOC correction according to the open circuit voltage and SOC calibration after full charging could meet the precision requirement of SOC estimation of pure electric vehicles. However for batteries in hybrid electric vehicles (HEV) this method is unable to meet the requirements due to the following reasons:

- The complexity of the working conditions because when the vehicles are moving, the current is both charged and discharged so as to keep the battery SOC in a narrow range.
- Due to the small capacities of the batteries the errors of the Ampere-Hour integral are high.
- There is no opportunity of full charging when the vehicles ae parked, except for maintenance.

Therefore, for the HEB other integrated methods are needed.

- Weighted fusion algorithm

This algorithm is to add up the SOC estimated through different methods in accordance with certain weights to obtain SOC. Figure 3.3 shows the operation of this algorithm:

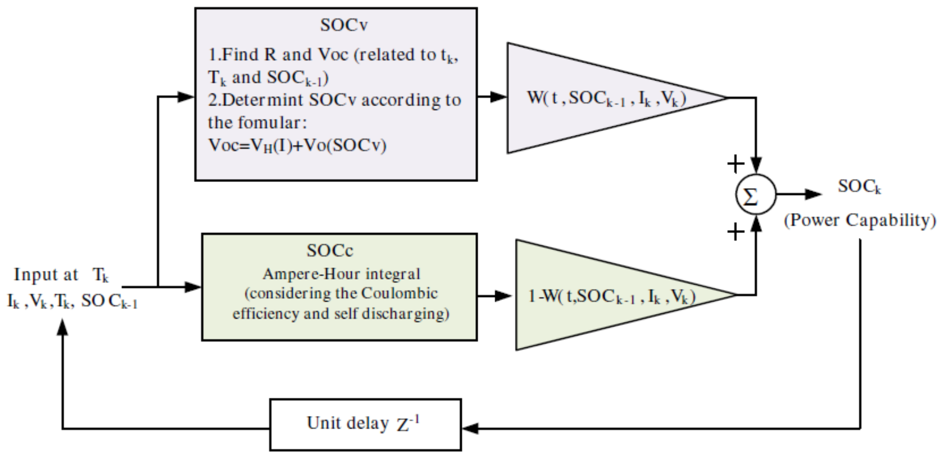

SOCv
1.Find R and Voc (related to t_k, T_k and SOC_{k-1})
2.Determint SOCv according to the fomular:
$Voc=V_H(I)+Vo(SOCv)$
$W(t, SOC_{k-1}, I_k, V_k)$
+
Σ
+
SOC_k
(Power Capability)
SOCc
Ampere-Hour integral (considering the Coulombic efficiency and self discharging)
$1-W(t, SOC_{k-1}, I_k, V_k)$
Input at T_k
I_k, V_k, T_k, SOC_{k-1}
Unit delay Z^{-1}

Figure 3.3.- weighted fusion algorithm from [3]

- Kalman filtering

Due to the impossibility of direct SOC measurement, two methods of SOC estimation are integrated as a dynamic system, in which SOC is regarded as an internal state of the system and is analysed. Furthermore, in order to take into consideration the nonlinearities of the battery, the Extended Kalman Filter (EKF) method is usually adopted. Generally, researches are conducted through systems formed by the Coulomb counting method and other battery models. In [12] it was pointed out that the meaning of EKF as a state observer lies in: when the SOC is estimated using the Coulomb counting method, the voltage of the capacitor is estimated and then the estimation values of the cell terminal voltage are obtained to act as a basis for correcting SOC; meanwhile noises and errors are taken into account, filtering gains of each step is determined in order to minimise the a posteriori error covariance, and eventually the optimal estimation of SOC is also obtained. In this way, with the combination of the Coulomb counting method and the model-based SOC estimation (Kalman Filter), which overcomes the shortcoming of cumulative errors that characterised the former, we can achieve a SOC closed-loop estimation. Moreover, since the measurement and process noise are taken into consideration, the algorithm has a strong inhibiting effect on noises, what makes it a robust method.

The Kalman filtering method used for SOC estimation relies on a reasonable battery equivalent model and a group of state equations. Therefore, as this method is highly dependent on the battery model, an accurate battery model is needed to be established so as to obtain a reliable SOC estimation. The model should not be too complex in order to save computations, but it should not be too simple either, to achieve an accurate resemblance with the battery that we are studying. Finally it is important to highlight that if the selection of the filtering gain is undesirable, the state will disperse.

- Sliding mode observer

So as to overcome the limitations of the Kalman filter method, the slip mode observer technology is used, which possesses strong robustness against the uncertainty of the model parameters and disturbance. The problem is that it is complicated to implement.

Taking everything into account, we determined that the Kalman Filter algorithm was the most suitable method to estimate the SOC of our battery due to its good compromise between accuracy and computational intensity.

3.2 INTRODUCTION TO DISCRETE SIMPLE KALMAN FILTER

Kalman filter (KF) is a well-known estimation theory introduced in 1960. The filter tries to estimate the systems state variables by providing a recursive solution through a linear optimal filtering [13].

As it was explained in section 3.1, this particular model-based state-estimation technique is proposed to estimate the state of a battery cell that are normally difficult or expensive to measure. It is widely used due to its interesting characteristics. Among other things, it can optimally (searching the minimum variance) estimate the states affected by a broadband noise contained within the system bandwidth, that cannot otherwise be filtered out using classical techniques [14].

We were based on the state space formulation defined in (2.5) and (2.6) for developing the Kalman Filter. However, in order to use this algorithm, it is necessary to consider that the battery model, of which we are trying to estimate the state vector, x_k, which includes the SOC among others, is governed by the discrete linear equation (3.2), whose output vector, y_k, is defined in (3.3).

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \tag{3.2}$$

$$y_k = Cx_k + Du_k + v_k \tag{3.3}$$

Equation (3.2) is known as the state or process equation, where $A \in R^{nxn}$ is the matrix that describes the system dynamics and relates the state at the previous time step k-1 to the current time step k when the control input u_{k-1} is equal to zero, and $B \in R^{nxp}$ relates the control input u_{k-1} to the state x_k. Whereas the equation (3.3) is called the output or measurement equation. The transformation matrix $C \in R^{mxn}$ and the feedforward matrix $D \in R^{mxp}$ relate the output y_k to the state x_k and the input u_k, respectively.

The process noise, w_k, and the measurement noise, v_k, are assumed to be random variables, independent of each other, white and with normal probability distributions [15]:

$$p(w) \sim N(0, Q) \tag{3.4}$$

$$p(v) \sim N(0, R) \tag{3.5}$$

In practice, the process noise covariance Q and the measurement noise covariance are supposed to be variable throughout time, however in this study we are going to consider them to be constants for all the simulations.

3.2.1 STATE OF CHARGE ESTIMATION BASED ON SIMPLE KALMAN FILTER ALGORITHM FOR LITHIUM-ION BATTERIES

Once it was verified that the battery model implemented in Matlab gave an accurate description of the MapleSim system behaviour, and the system of equations that governed our battery model was discretised, we continued trying to implement the Simple Kalman filter (SKF) so as to estimate the state vector x_k, which is the same as the vector "x" defined in (2.10) but discretised. The discrete SKF algorithm is defined as follows:

First of all the Kalman Filter must be initialised:

$$\hat{x}_{k-1} = E(x_0)\,;\;\; P_{k-1} = E[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^T] \qquad (3.6)$$

The state vector initialisation is obtained from the expected value that we would think the real state vector would have. The error covariance matrix P is initialised by obtaining the covariance matrix as defined in (3.6). Normally, these quantities are not precisely known, but this is not a problem due to the robustness to poor initialisation that characterises the Kaman Filter, and it will quickly converge to the true value as it runs [16].

Time update:

1. A priori state estimate update:

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1} \qquad (3.7)$$

2. Error covariance time update:

$$P_k^- = AP_{k-1}A^T + Q \qquad (3.8)$$

Measurement update:

3. Kalman gain matrix

$$K_k = P_k^- C^T (CP_k^- C^T + R)^{-1} \qquad (3.9)$$

4. Measurement residual

$$\tilde{y}_k = y_k - (C\hat{x}_k^- + Du_k) \qquad (3.10)$$

5. A posteriori state estimate update

$$\hat{x}_k = \hat{x}_k^- + K_k\tilde{y}_k \qquad (3.11)$$

6. Estimated output

$$\hat{y}_k = C\hat{x}_k + Du_k \tag{3.12}$$

7. Error covariance measurement update

$$P_k = (I - K_k \cdot C) \cdot P_k^- \tag{3.13}$$

8. Repeat steps one to seven until the simulation ends

The state of charge is tracked by using the expression defined in (2.8). Moreover, as we were working with the Simple Kalman Filter, it was necessary to use the linear expression of the open circuit voltage (V_{oc}), which was expressed in equation (2.11). Finally, in order to implement this algorithm using (3.2), the matrices defined in (2.10) must be discretised:

$$A = \begin{bmatrix} e^{\frac{-Ts}{R_{1,k}\cdot C_{1,k}}} & 0 & 0 \\ 0 & e^{\frac{-Ts}{R_{2,k}\cdot C_{2,k}}} & 0 \\ 0 & 0 & 1 \end{bmatrix}; B = \begin{bmatrix} R_{1,k}\cdot\left(1 - e^{\frac{-Ts}{R_{1,k}\cdot C_{1,k}}}\right) \\ R_{2,k}\cdot\left(1 - e^{\frac{-Ts}{R_{2,k}\cdot C_{2,k}}}\right) \\ \frac{-Ts}{C_{use,k}\cdot 3600} \end{bmatrix};$$

$$C = \begin{bmatrix} -1 & -1 & \frac{\delta V_{OC,k}}{\delta SOC_k} \end{bmatrix}; D = \begin{bmatrix} -R_{0,k} \end{bmatrix}; H = [1] \tag{3.14}$$

And the output equation (3.3) is expressed by the following equation:

$$yv_k = V_{termv,k} = V_{oc,k}(SOC_k) - V_{C1,k} - V_{C2,k} - I_{batt,k}\cdot R_{int,k} + v_k \tag{3.15}$$

Implementing this algorithm in Matlab with the parameters defined before, (2.3), we obtained the following response, which was compared to the response of the discrete linear system defined in (3.3) and (3.15):

Where: $Q = \begin{bmatrix} 10^{-6} & 0 & 0 \\ 0 & 10^{-6} & 0 \\ 0 & 0 & 10^{-6} \end{bmatrix}$ *and* $R = 1.5$

And the algorithm initialization: $\hat{x} = \begin{bmatrix} 0.1 \\ 0.5 \\ 0.9 \end{bmatrix}; P = \begin{bmatrix} 0.0009 & 0.0047 & -0.0009 \\ 0.0047 & 0.0233 & -0.0047 \\ -0.0009 & -0.0047 & 0.0009 \end{bmatrix}$

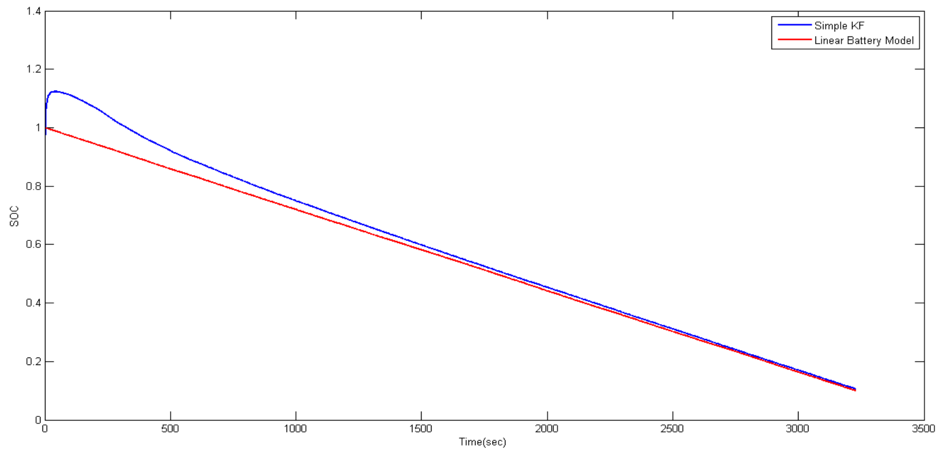


Figure 3.4.- Comparison between linear system and Simple Kalman Filter SOC estimation for an input which consists of a constant current of 1A

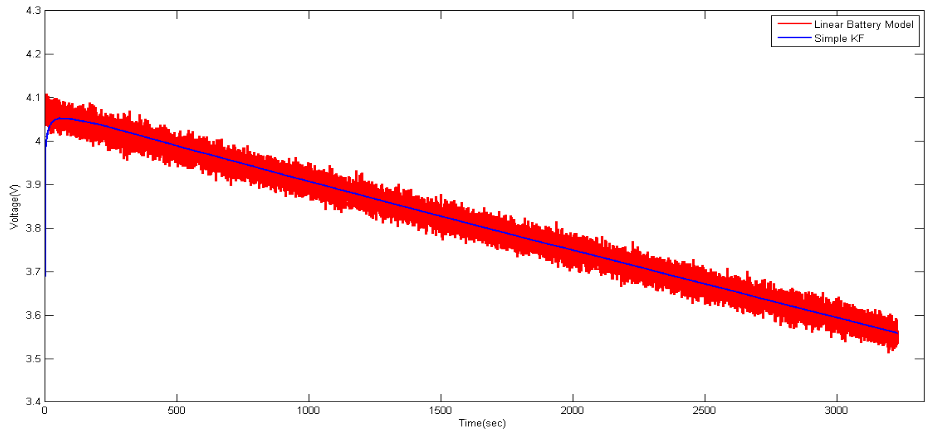


Figure 3.5.- Comparison between linear system and Simple Kalman Filter terminal voltage estimation for an input which consists of a constant current of 1A

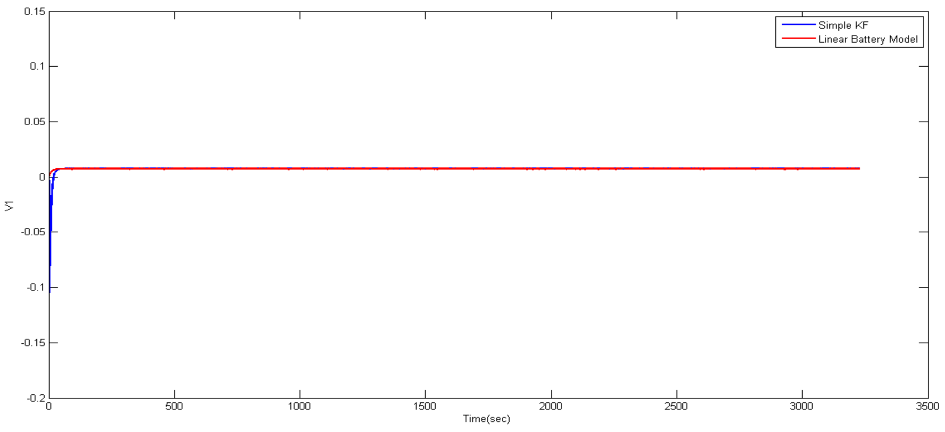


Figure 3.6.- Comparison between linear system and Simple Kalman Filter voltage drop estimation in the first RC network for an input which consists of a constant current of 1A

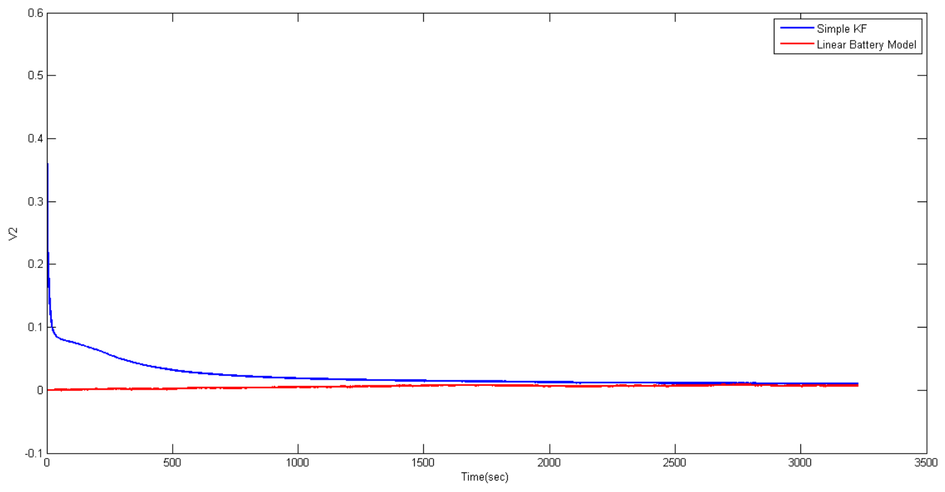


Figure 3.7.- Comparison between linear system and Simple Kalman Filter voltage drop estimation in the second RC network for an input which consists of a constant current of 1A

It is visible from the previous graphics that the Simple Kalman Filter makes a good estimation when we are working with linear systems despite the strong measurement noise (*Figure 3.5*). In a simulation environment, as we are in ideal conditions, the Coulomb counting method is supposed to give the right estimation of the SOC as we do not have errors in the measurements.

The input was changed to a square wave pulse train with an amplitude of 10A, conserving the previous initialisations. The results obtained are the following:

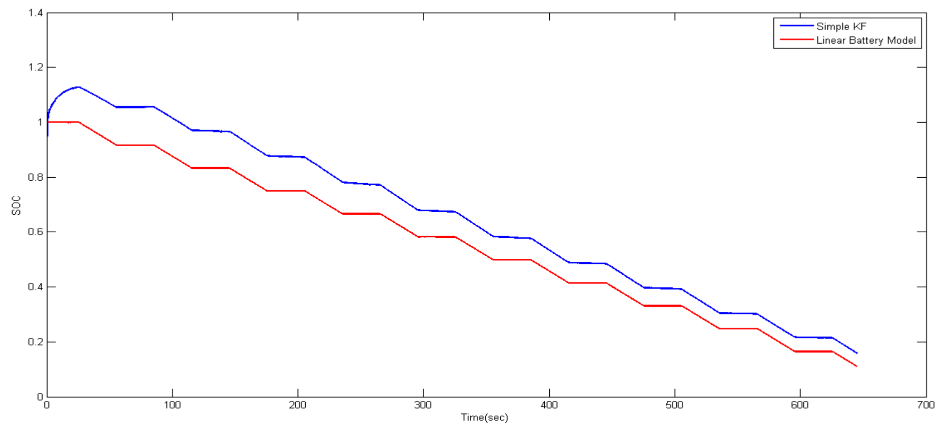


Figure 3.8.- Comparison between linear system and Simple Kalman Filter SOC estimation for an input which consists of a square wave pulse train with an amplitude of 10A

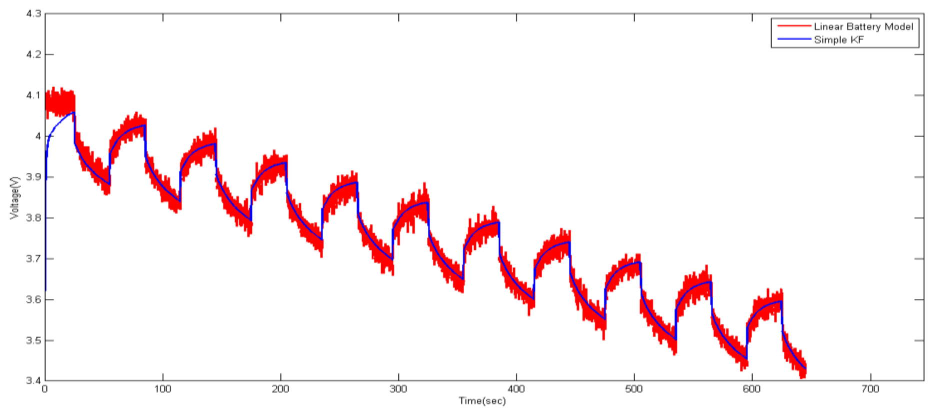


Figure 3.9.- Comparison between linear system and Simple Kalman Filter terminal voltage estimation for an input which consists of a square wave pulse train with an amplitude of 10A

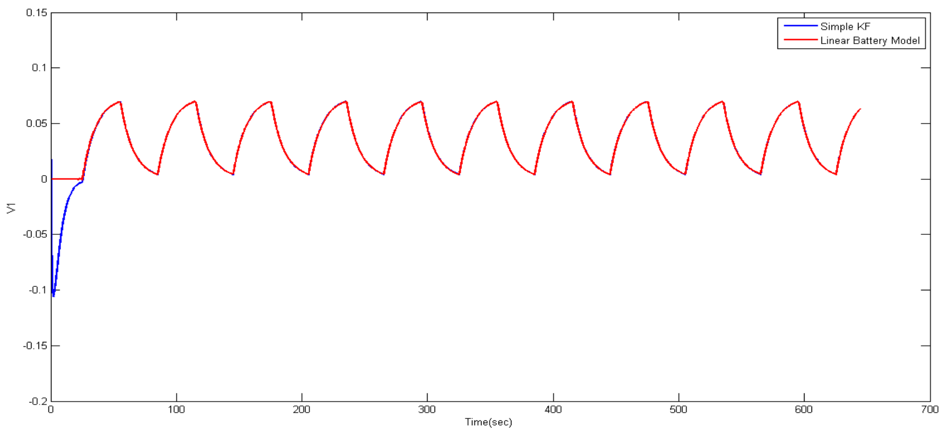


Figure 3.10.- Comparison between linear system and Simple Kalman Filter voltage drop estimation in the first RC network for an input which consists of a square wave pulse train with an amplitude of 10 A

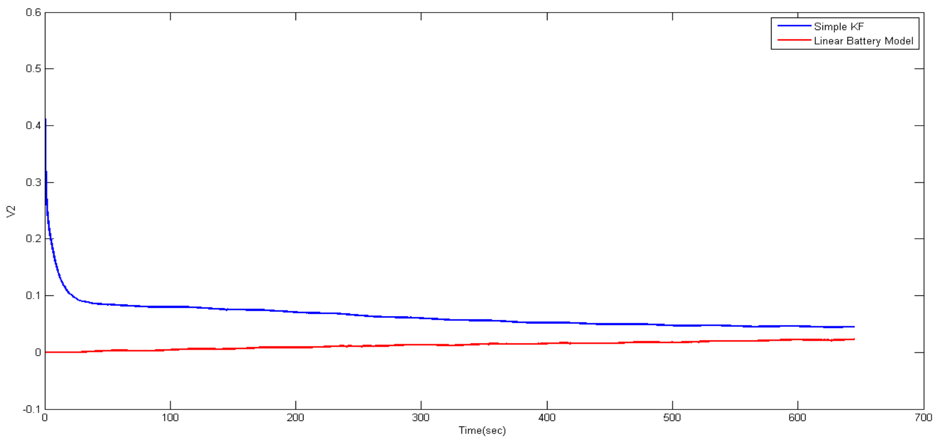


Figure 3.11.- Comparison between linear system and Simple Kalman Filter voltage drop estimation in the second RC network for an input which consists of a square wave pulse train with an amplitude of 10

We can observe that in this case, as the input current experiences abrupt changes, the SOC estimation is not as good as for the constant current. The SKF struggles due to the non-linear behaviour of the battery terminal voltage, V_{term}. This is to be expected as the algorithm is trying to minimise the error between non-linear process (V_{term}) and a linear equation by varying SOC away from its true value.

At this point, we wanted to improve this estimation, thus we proceeded with the implementation of the Extended Kalman Filter, which is able to keep track of the nonlinearities of the system. In order to do that we implemented also the nonlinear battery model, which was introduced in section 2.1. The procedure followed will be seen in the next section.

3.3 EXTENDED KALMAN FILTER

As it was said previously, in reality V_{oc}, R_1, C_1, R_2 and C_2 are nonlinearly dependant on SOC, therefore it is necessary to consider that our system (the battery model), whose state vector x_k (which includes SOC) is the aim of our estimation, is governed by the discrete nonlinear difference stochastic equation (3.16) whose measurement vector y_k is given by the equation (3.17). Where *f(.,.)* is a nonlinear state transition function and *g(.,.)* is a nonlinear measurement function.

$$x_k = f(x_{k-1}, u_{k-1}) + w_{k-1} \qquad (3.16)$$

$$y_k = g(x_k, u_k) + v_k \qquad (3.17)$$

As before, w_k and v_k are assumed to be mutually uncorrelated white Gaussian random processes, with zero mean and covariance matrices Q and R, (3.4) and (3.5), respectively. Now, at each time step $f(x_{k-1},u_{k-1})$ and $g(x_k,u_k)$ are linearised by a first-order Taylor-series expansion. We assume that *f(.,.)* and *g(.,.)* are differentiable at all operating points (x_k,u_k), as shown in (3.18) and (3.19). As we can deduce from these expressions, the matrices A and G (which replaces C) are now Jacobian matrices A_{k-1} and G_k of partial derivatives with respect to x.

$$f(x_{k-1}, u_{k-1}) \approx f(\hat{x}_{k-1}, u_{k-1}) + \underbrace{\left.\frac{\partial f(x_{k-1}, u_{k-1})}{\partial x_{k-1}}\right|_{x_{k-1}=\hat{x}_{k-1}}}_{defined\ as\ A_{k-1}} (x_{k-1} - \hat{x}_{k-1}) \qquad (3.18)$$

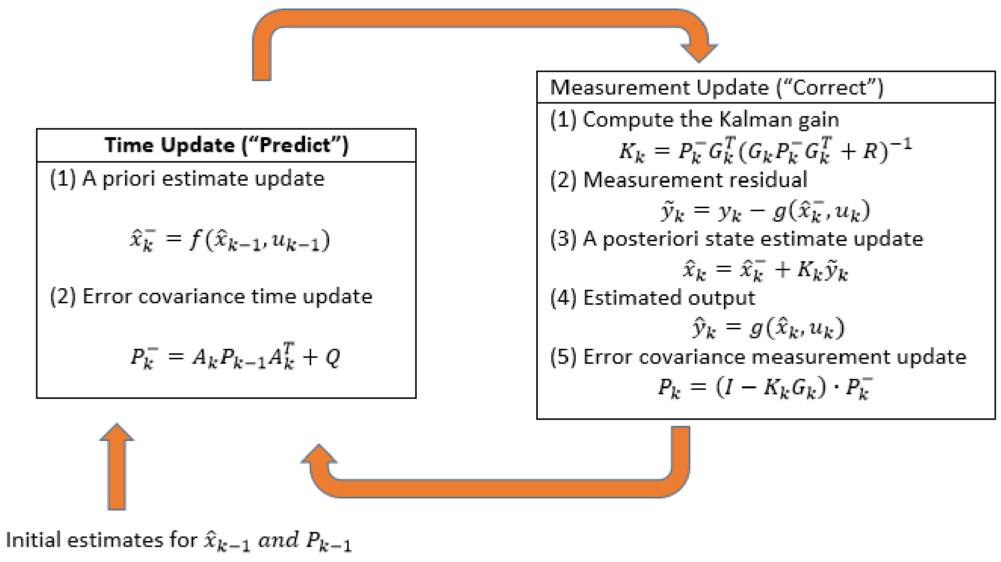
$$g(x_k, u_k) \approx g(\hat{x}_k, u_k) + \underbrace{\left.\frac{\partial g(x_k, u_k)}{\partial x_k}\right|_{x_k=\hat{x}_k}}_{defined\ as\ G_k} (x_k - \hat{x}_k) \qquad (3.19)$$

If we combine the equations (3.16) and (3.17) with (3.18) and (3.19), and we transform them into the state space formulation defined in (2.5) and *(2.6)*, we obtain the equations that describe the nonlinear system, where matrices A, B, G and D are now subscripted with k so as to highlight that they also vary with time.

$$x_k = A_{k-1} \cdot x_{k-1} + B_{k-1} \cdot u_{k-1} + w_{k-1} \qquad (\ 3.20\ )$$

$$y_k = G_k \cdot x_k + D_k \cdot u_k + v_k \qquad (\ 3.21\ )$$

The EKF algorithm remains almost the same as that of the SKF (from (3.7) to (3.13)), with only steps 1, 4 and 6 changing, as it is shown in Figure *3.12*.



Time Update ("Predict")
(1) A priori estimate update
$\hat{x}_k^- = f(\hat{x}_{k-1}, u_{k-1})$
(2) Error covariance time update
$P_k^- = A_k P_{k-1} A_k^T + Q$
Measurement Update ("Correct")
(1) Compute the Kalman gain
$K_k = P_k^- G_k^T (G_k P_k^- G_k^T + R)^{-1}$
(2) Measurement residual
$\tilde{y}_k = y_k - g(\hat{x}_k^-, u_k)$
(3) A posteriori state estimate update
$\hat{x}_k = \hat{x}_k^- + K_k \tilde{y}_k$
(4) Estimated output
$\hat{y}_k = g(\hat{x}_k, u_k)$
(5) Error covariance measurement update
$P_k = (I - K_k G_k) \cdot P_k^-$
Initial estimates for $\hat{x}_{k-1}$ and P_{k-1}

Figure 3.12.- Extended Kalman Filter algorithm

3.3.1 STATE OF CHARGE ESTIMATION BASED ON EXTENDED KALMAN FILTER ALGORITHM FOR LITHIUM-ION BATTERIES

Aiming to implement this algorithm in Matlab, we defined the nonlinear functions *f(.,.)* and *g(.,.)* as well as their respective matrices, A and G.

$$f(x_{k-1}, u_{k-1}) = A_{k-1} \cdot x_{k-1} + B_{k-1} \cdot u_{k-1} \qquad (\ 3.22\ )$$

$$g(x_k, u_k) = \underbrace{0{,}1958 * e^{1{,}332 \cdot x_{3,k}} + 3{,}429703601}_{V_{OC}\, defined\ in\ (\ 2.4\ )} - x_{1,k} - x_{2,k}$$
$$- \underbrace{\left(0.11 \cdot e^{-50 \cdot x_{3,k}} + 0.0075\right)}_{R_0 defined\ in\ (\ 2.3\ )} \cdot u_k \qquad (\ 3.23\ )$$

$$A_{k-1} = \frac{\delta f_{[i]}}{\delta x_{[j]}}(\hat{x}_{k-1}, u_{k-1}) = \begin{bmatrix} e^{\frac{-Ts}{R_{1,k-1}\cdot C_{1,k-1}}} & 0 & \frac{\delta f_1}{\delta x_3} \\ 0 & e^{\frac{-Ts}{R_{2,k-1}\cdot C_{2,k-1}}} & \frac{\delta f_2}{\delta x_3} \\ 0 & 0 & 1 \end{bmatrix};$$

(3.24)

$$G_k = \frac{\delta g_{[i]}}{\delta x_{[j]}}(\hat{x}_k, u_k) = [-1 \quad -1 \quad G(1,3)]\,;\; B_{k-1} = \begin{bmatrix} R_{1,k-1}\cdot\left(1 - e^{\frac{-Ts}{R_{1k-1}\cdot C_{1k-1}}}\right) \\ R_{2,k-1}\cdot\left(1 - e^{\frac{-Ts}{R_{2,k-1}\cdot C_{2,k-1}}}\right) \\ \frac{-Ts}{C_{use}\cdot 3600} \end{bmatrix}$$

Where:

$$G(1,3) = \frac{\delta g_{[1]}}{\delta x_{[3]}}(\hat{x}_k, u_k) = \underbrace{0.1958\cdot 1.332\cdot e^{1.332\cdot\hat{x}_3}}_{\frac{\partial V_{OC}}{\partial x_3}} - \left[\underbrace{0.11\cdot(-50)\cdot e^{-50\cdot\hat{x}_3}}_{\frac{\partial R_O}{\partial x_3}}\cdot u_k\right].$$

The matrix A defined in (3.24) was simplified due to some singularity problems experienced during the implementation of the algorithm in Matlab. Hence, we ended up using the expression of this matrix defined in (3.14), which had been employed for the Simple Kalman Filter too. The singularity problems were caused because the denominators of the terms $\frac{\delta f_1}{\delta x_3}$ and $\frac{\delta f_2}{\delta x_3}$ reached such a small values that they became infinite.

Using a constant current of 1A as the input of our system we obtain the results depicted from Figure 3.13 to Figure 3.16. The matrices Q, R, as well as the initialisations of $\hat{x}$ and P are the same as for the SKF.

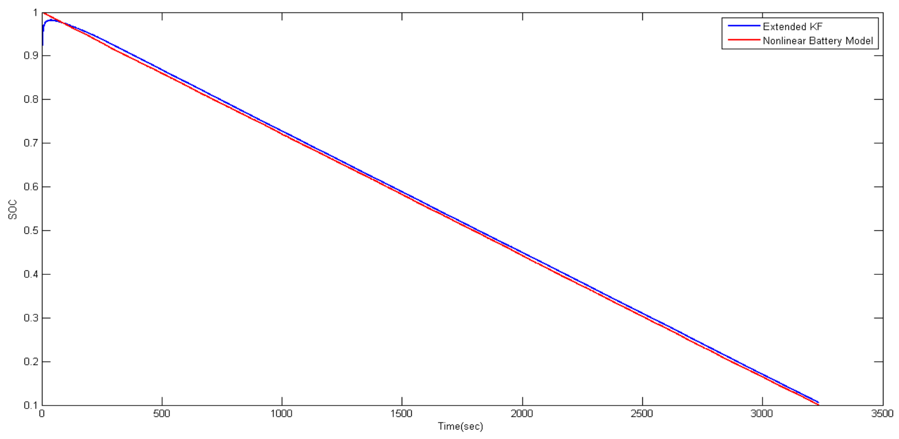


Figure 3.13.- Comparison between nonlinear system (red) and Extended Kalman Filter (blue) SOC estimation for an input which consists of a constant current of 1A

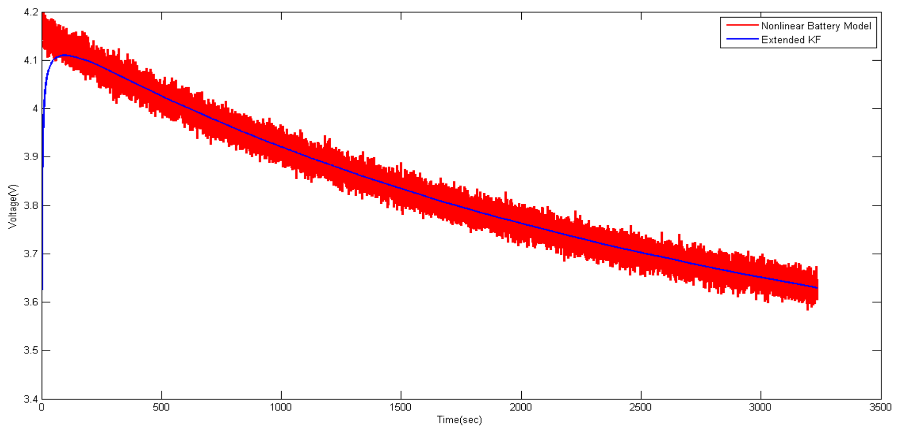


Figure 3.14.- Comparison between nonlinear system (red) and Extended Kalman Filter (blue) terminal voltage estimation for an input which consists of a constant current of 1A

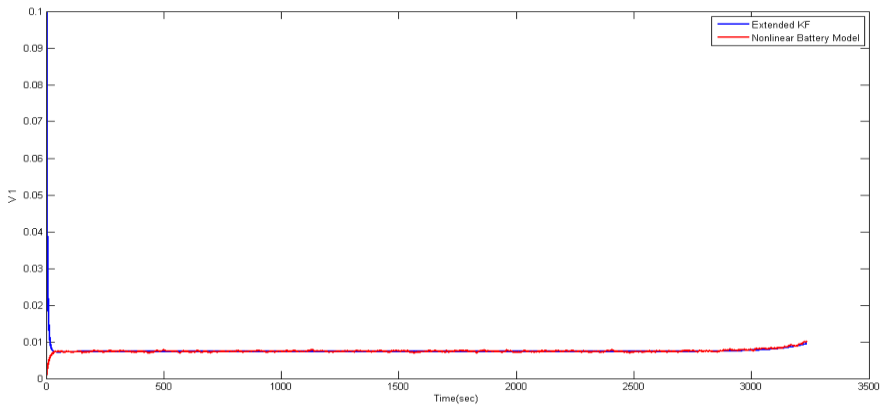


Figure 3.15.- Comparison between nonlinear system (red) and Extended Kalman Filter (blue) voltage drop estimation in the first RC network for an input which consists of a constant current of 1A

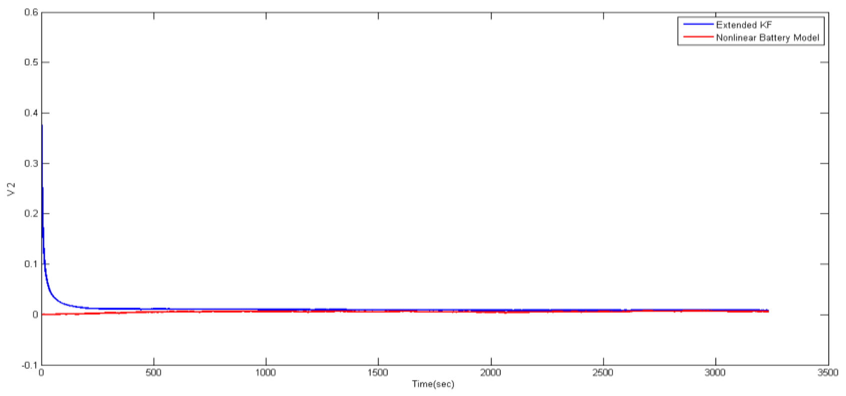


Figure 3.16.- Comparison between nonlinear system (red) and Extended Kalman Filter (blue) voltage drop estimation in the second RC network for an input which consists of a constant current of 1A

From these graphics we can affirm that, for this particular input current, this algorithm makes a good estimation. Despite the good estimation that the SKF also gave for the same input, if we compare the Figure 3.7 and Figure 3.16, which represent estimation of the voltage drop in the second RC network (V_2) using the SKF and the EKF, respectively, we can state that we achieve a faster convergence to the true vale with the latter. Therefore, for this input the estimation with the EKF has improved.

Keeping track of the errors, we obtain that the mean absolute error of the terminal voltage (V_{term}) estimation is 0,0032V and the relative error of this estimation is 0,00078067. Concerning the SOC estimation for this particular input, the value of the mean absolute error is 0,007. This values can support what we have just declared in the previous paragraph.

As SKF algorithm showed some problems of convergence with the input current which consists of a square wave pulse train of 10 A. We introduced the same input to the EKF in order to reaffirm that this algorithm provides better estimations. The initialisation matrices were the same as before.

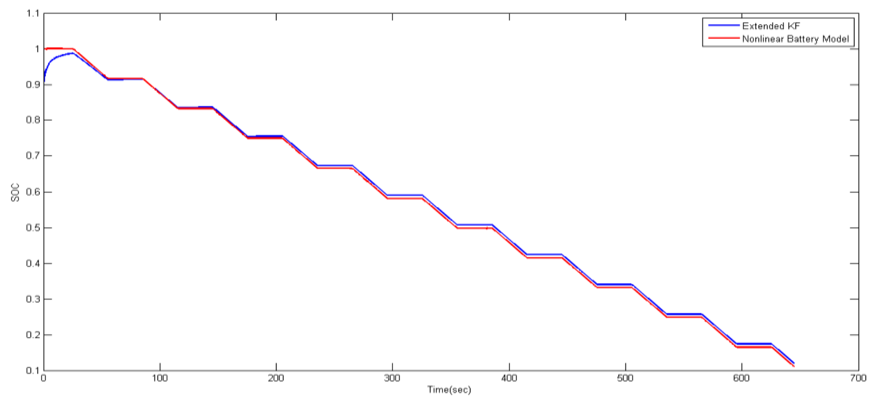


Figure 3.17.- Comparison between nonlinear system (red) and Extended Kalman Filter (blue) SOC estimation for an input which consists of a square wave pulse train with an amplitude of 10A

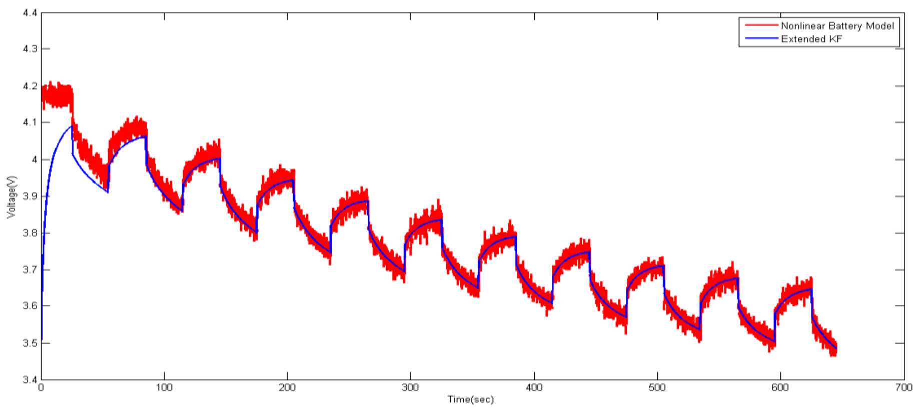


Figure 3.18.- Comparison between nonlinear system (red) and Extended Kalman Filter (blue) terminal voltage estimation for an input which consists of a square wave pulse train with an amplitude of 10A

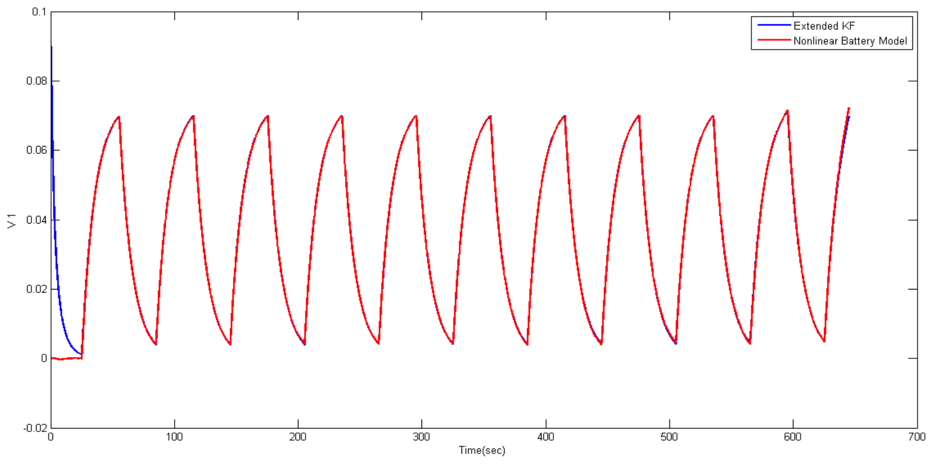


Figure 3.19.- Comparison between nonlinear system (red) and Extended Kalman Filter (blue) voltage drop estimation in the first RC network for an input which consists of a square wave pulse train with an amplitude of 10A

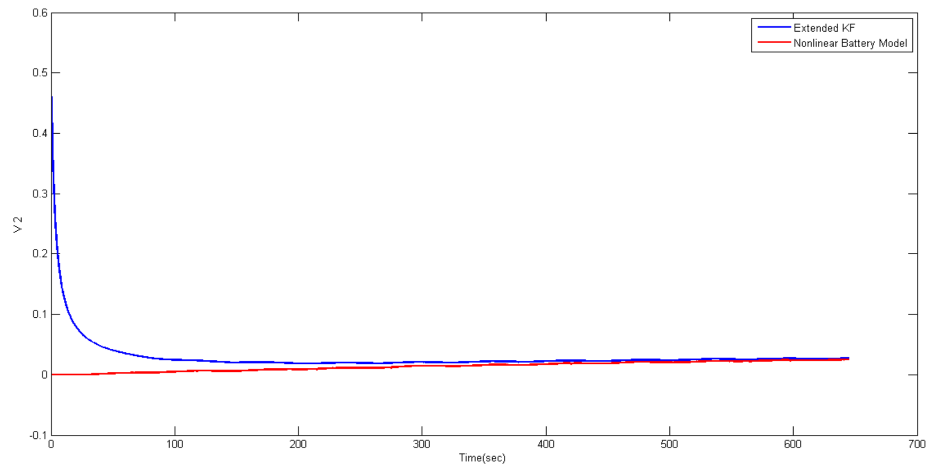


Figure 3.20.- Comparison between nonlinear system (red) and Extended Kalman Filter (blue) voltage drop estimation in the second RC network for an input which consists of a square wave pulse train with an amplitude of 10A

Keeping track of the errors, we obtain that the mean absolute error of the terminal voltage (V_{term}) estimation is 0,014706 V and the relative error of this estimation is 0,0036. Concerning the SOC estimation for this particular input, the value of the mean absolute error is 0,00812. This values are a little bit higher than those for the previous input, but anyway, with this algorithm the estimations are better compared to that of the SKF by far.

From the comparison between the SOC estimation obtained from the SKF algorithm (Figure *3.8*) and that from the EKF algorithm (Figure *3.17*), we can deduce that the SOC estimation is highly improved using the EKF algorithm. Not only does it arrive to the true value but it converges very quickly. Comparing the estimation of the voltage drop in the second RC network (V_2) it is also visible the important improvement achieved by the EKF.

Finally, in order to make sure that this algorithm makes good estimations for a wide range of different inputs, we decided to introduce an input which consists of a constant current of 50 A. As our battery was not originally designed to support such a strong current, the estimation did not have to be as perfect as for the previous cases. However, we verified that, in spite of the strong current, the estimation was quite good with a mean absolute error of the terminal voltage (V_{term}) of 0,1057V and a relative error of this estimation of 0,0316. Concerning battery SOC, it was estimated with a mean absolute error of 0,0168. These errors are really small bearing in mind that the battery is not conceived for this kinds of strong inputs.

The results that could corroborate what it was stated in the previous paragraph are depicted from *Figure 3.21* to Figure 3.24.

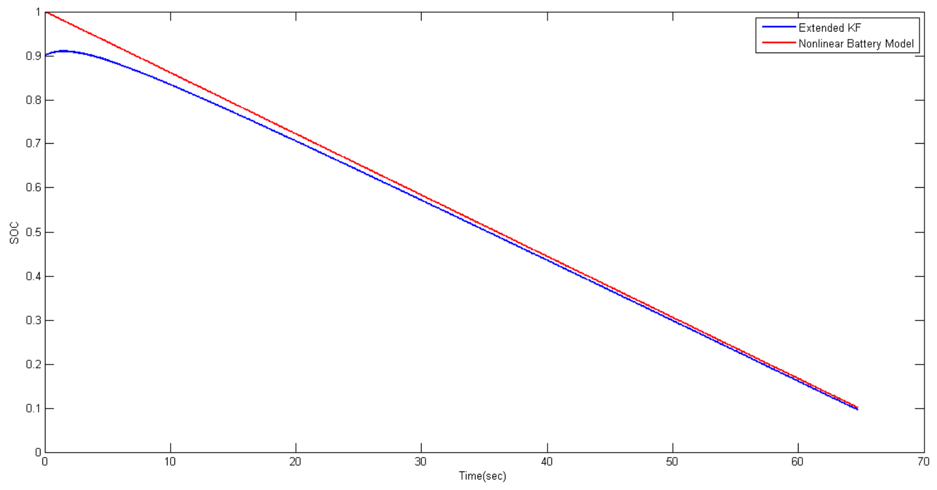


Figure 3.21.- Comparison between nonlinear system (red) and Extended Kalman Filter (blue) SOC estimation for an input which consists of a constant current of 50A

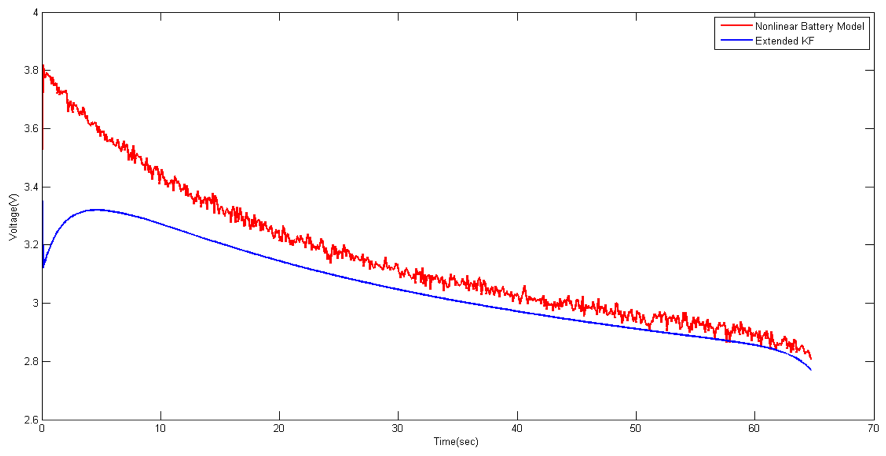


Figure 3.22.- Comparison between nonlinear system (red) and Extended Kalman Filter (blue) terminal voltage estimation for an input which consists of a constant current of 50A

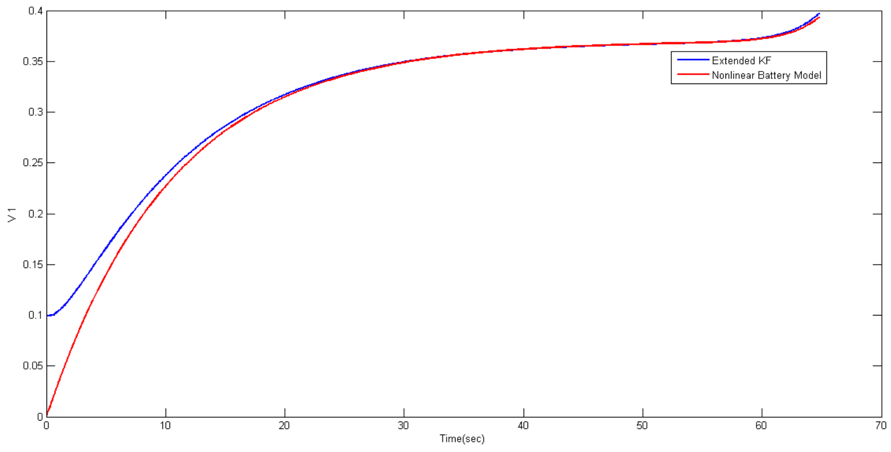


Figure 3.23.- Comparison between nonlinear system (red) and Extended Kalman Filter (blue) voltage drop estimation in the first RC network for an input which consists of a constant current of 50A

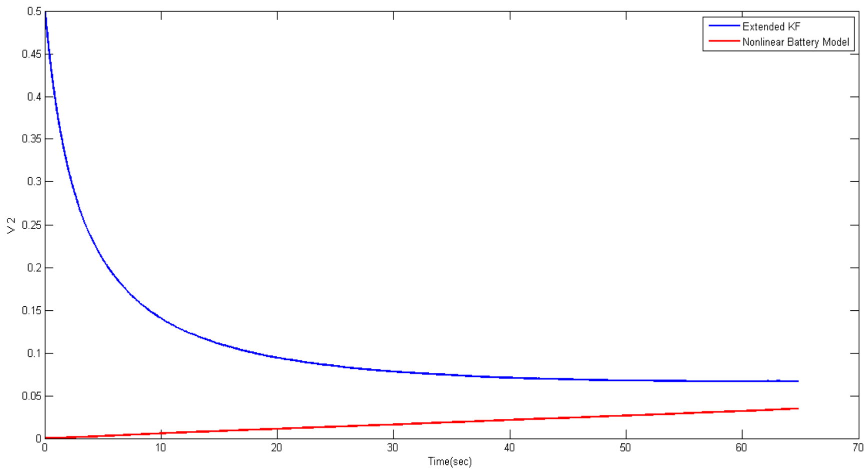


Figure 3.24.- Comparison between nonlinear system (red) and Extended Kalman Filter (blue) voltage drop estimation in the second RC network for an input which consists of a constant current of 50A

Finally, once we had confirmed that the EKF enhance all the estimations compared to the SKF, and in order to verify that the EKF algorithm also provided good estimations after several cycles of charge and discharge, we created a script in Matlab in which it was possible to define the number of cycles that our battery model and EKF would be subjected to. In this case, the number of cycles was fixed to five and the input current was the same square wave pulse train with an amplitude of 10 A and a period of 60 seconds that it was used in the simulations for the SKF and the EKF.

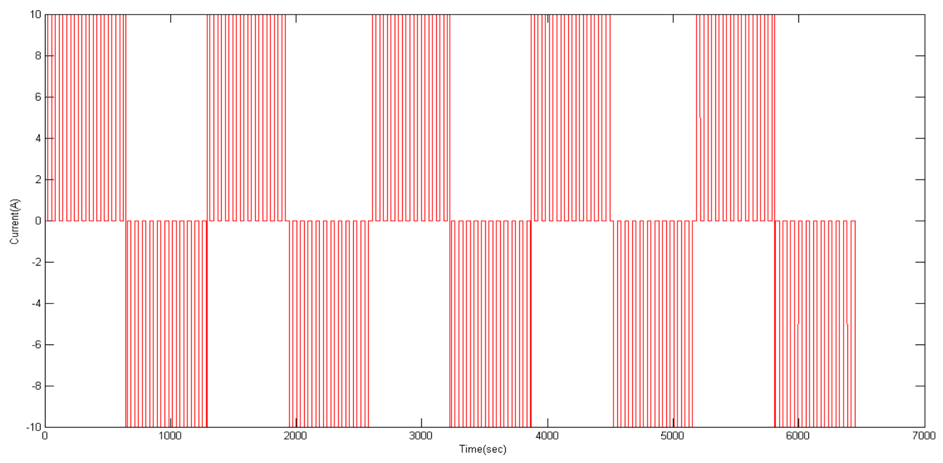


Figure 3.25.- Profile of the input current: 5 complete cycles of discharge and charge consisting of a square pulse wave with an amplitude of +/- 10A and T=60 sec

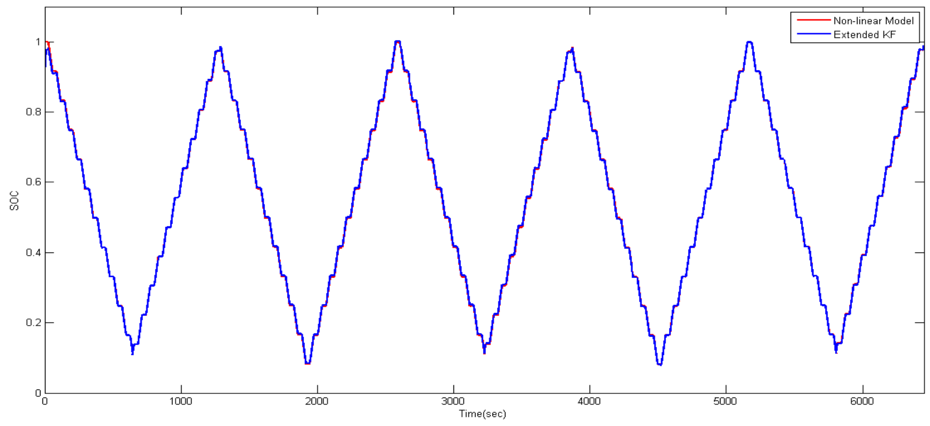


Figure 3.26.- Comparison between nonlinear system (red) and Extended Kalman Filter (blue) SOC estimation for an input which consists of a square wave pulse train with an amplitude of 10A and T=60sec

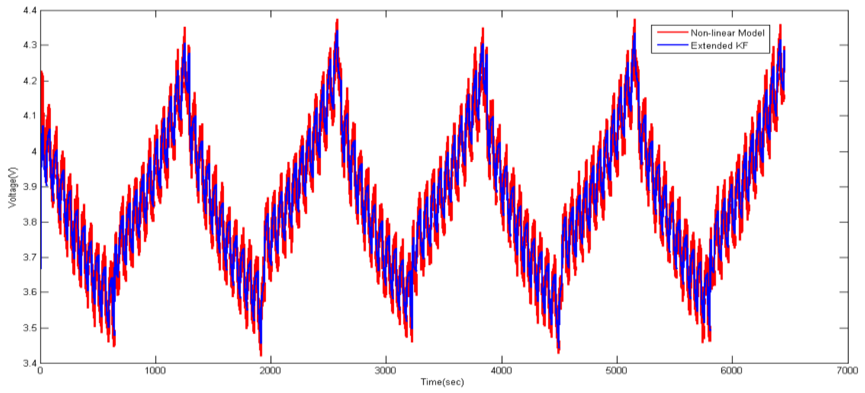


Figure 3.27.- Figure 3.18.- Comparison between nonlinear system (red) and Extended Kalman Filter (blue) terminal voltage estimation for an input which consists of a square wave pulse train with an amplitude of 10A and T=60sec

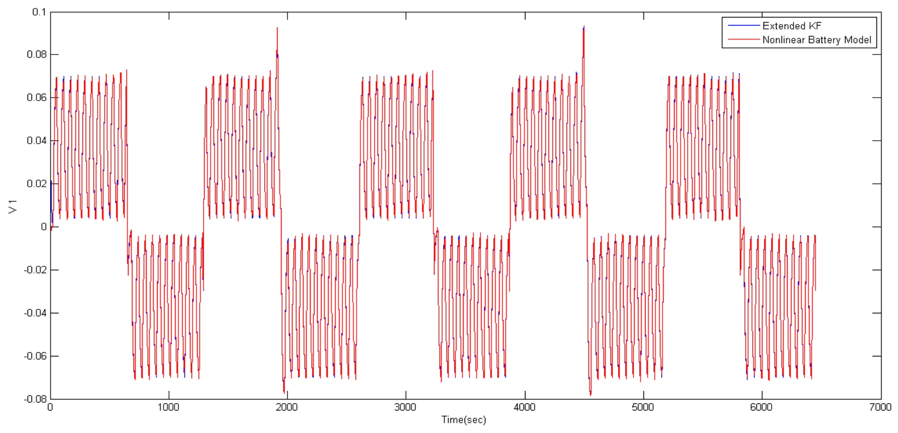


Figure 3.28.- Comparison between nonlinear system (red) and Extended Kalman Filter (blue) voltage drop estimation in the first RC network for an input which consists of a square wave pulse train with an amplitude of 10A and T=60sec

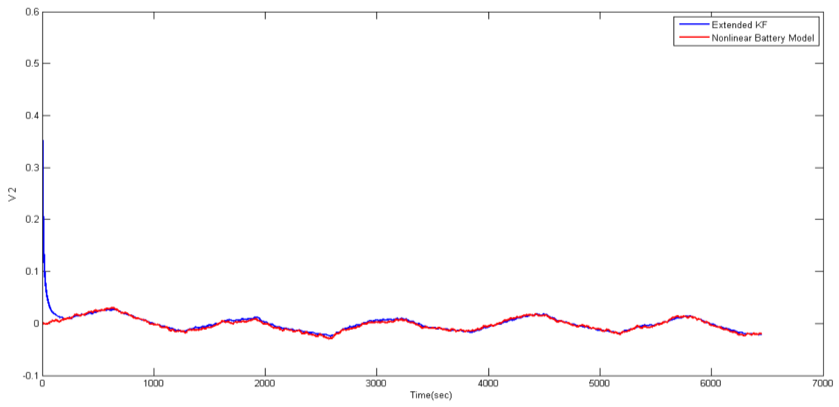


Figure 3.29.- Comparison between nonlinear system (red) and Extended Kalman Filter (blue) voltage drop estimation in the second RC network for an input which consists of a square wave pulse train with an amplitude of 10Aand T=60sec

Taking everything into account, we could confirm that our algorithm always provides a good estimation of the state of the battery. Therefore, we proceeded to implement this algorithm together with the real battery in Simulink, so as to test our EKF algorithm with real data. Our algorithm was implemented in Simulink using the block “Matlab Function” from Simulink library, as we will detail in the next section.

3.4 STATE OF CHARGE ESTIMATION BASED ON KALMAN FILTER ALGORITHM FOR LITHIUM-ION BATTERIES IN SIMULINK

The last step so as to validate our algorithm, and also to make sure that it was ready to its implementation into a battery management system (BMS), was to verify that it provided good results in real time using real data from the reference battery. In order to do this, we transformed the subsystem of our battery in MapleSim into a Simulink block, using one of the functions of the MapleSim connector which allowed to export a MapleSim model to Simulink using Simulink s-functions that is called “Simulink Component Block Generation”. Firstly, it was necessary to convert our MapleSim model workspace into a subsystem with the inputs and outputs desired for our Simulink block. This tool identifies the set of modelling components that you want to export as a block component. Since Simulink only supports data signals, properties on acausal connectors such as mechanical flanges and electrical pins, must be converted to signals using the appropriate ports [17]. The subsystem created in MapleSim is shown in *Figure 3.30* and the outcome of this transformation (*Figure 3.31*) permits us to implement our reference battery directly in Simulink (See Annex to have a further knowledge on the procedure).

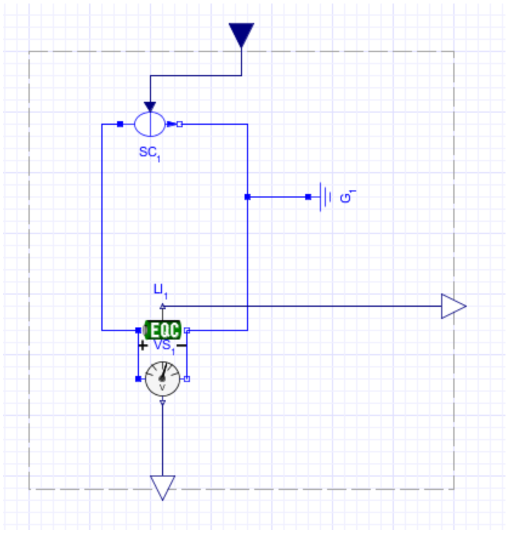


Figure 3.30.- Battery subsystem in MapleSim

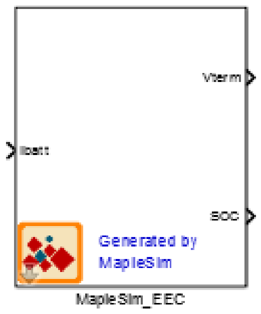


Figure 3.31.- Simulink block created from the battery in MapleSim in order to estimate the SOC

Concerning the development of the estimation algorithm, we used the block “Matlab Function” from the Simulink library, and we adapted the Matlab script to Simulink environment, i.e., we deleted the *for* loop from the script as Simulink iterates itself in each time step. We needed also to add a retard in each feedback loop in order to introduce the value of the state vector and that of the error covariance matrix at the previous time step, as well as the value of the input current at the previous time step as it was necessary for the algorithm. Once all these things were implemented correctly (Figure 3.32), we were able to carry out all the desired simulations using as input of our algorithm the real data (V_{term}) from the reference battery.

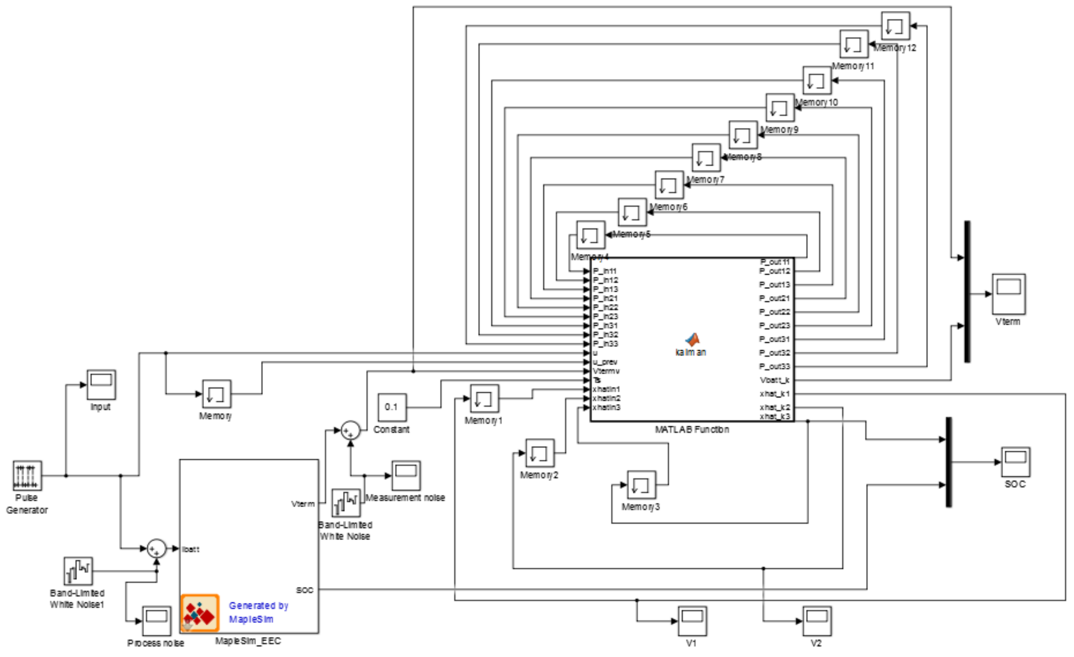


Figure 3.32.- Battery implemented together with the EKF algorithm in Simulink

We implemented both algorithms, Simple Kalman Filter (SKF) and Extended Kalman Filter (EKF), in order to study the results of both estimators. Using a sample time of 0,1 and an input which consists of a square wave pulse train with a period of 60 seconds and an amplitude of 10A, we obtain the following responses for the SKF and the EKF, respectively.

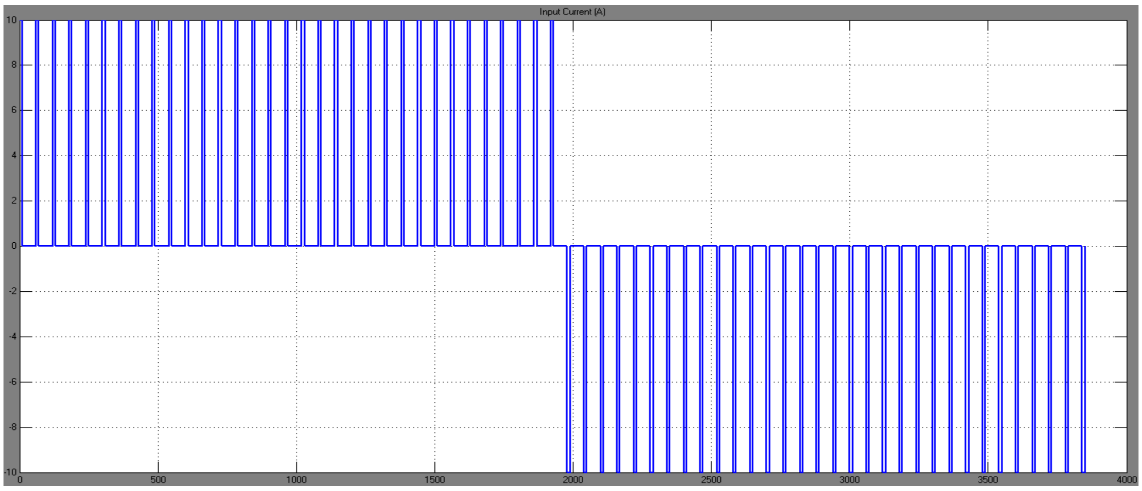


Figure 3.33.- Common input current for both algorithms: pulse train with an amplitude of 10A and a T=60 sec.

- **Simple Kalman Filter (SKF)**

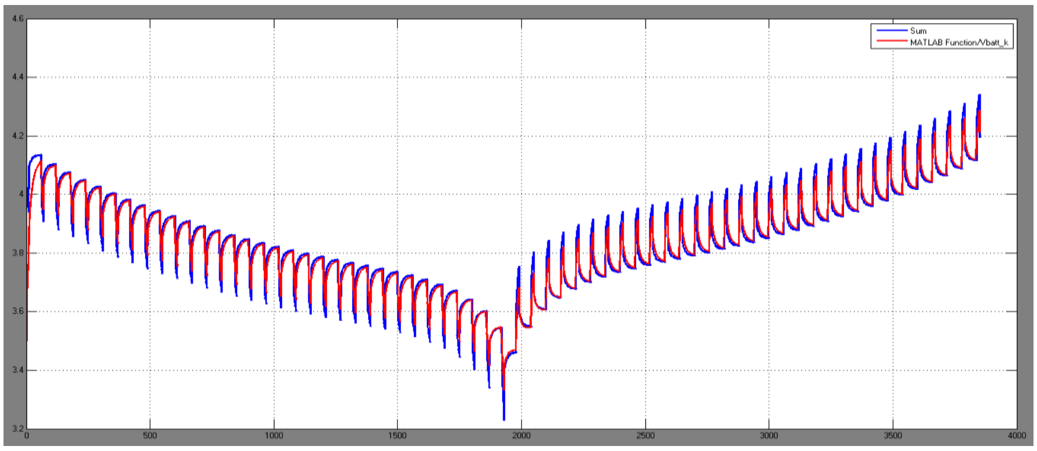


Figure 3.34.- Terminal voltage output: Comparison between real data from the reference battery (blue) and Simple Kalman Filter estimation (red)

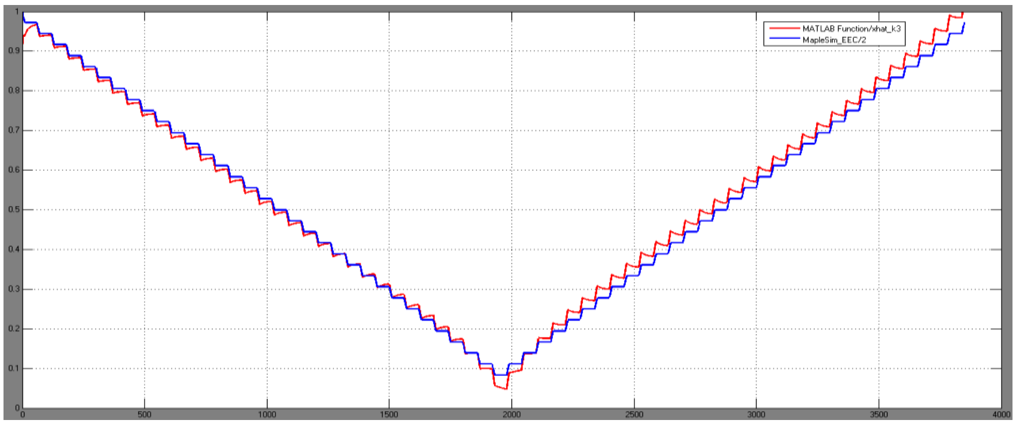


Figure 3.35.- SOC output: Comparison between real data from the reference battery (blue) and Simple Kalman Filter estimation (red)

- **Extended Kalman Filter (EKF)**

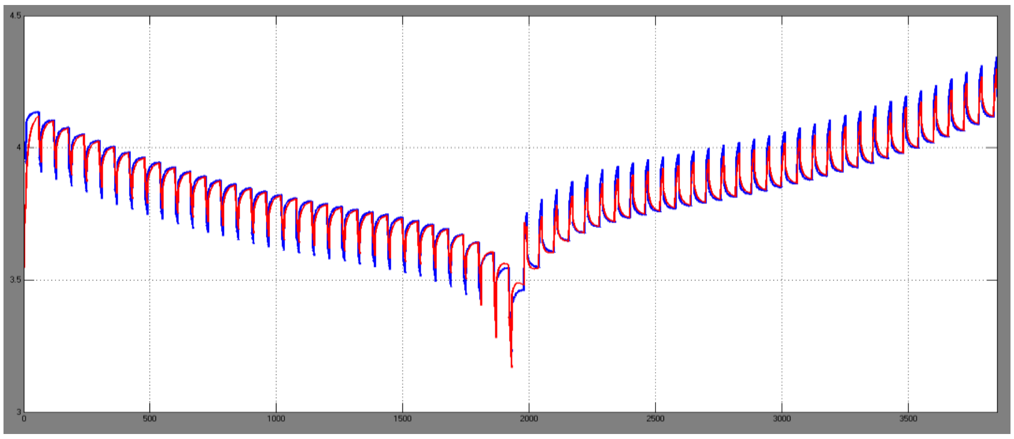


Figure 3.36.- Terminal voltage output: Comparison between real data from the reference battery (blue) and Extended Kalman Filter estimation (red)

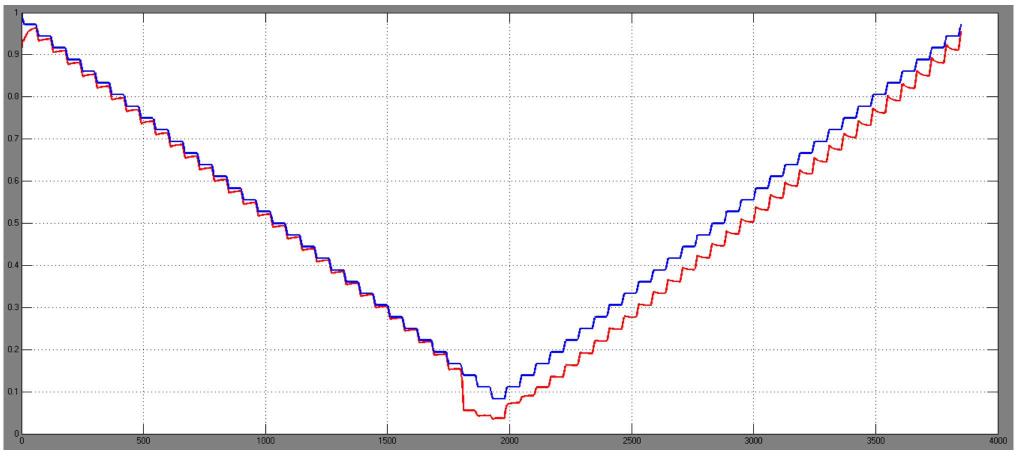


Figure 3.37.- SOC output: Comparison between real data from the reference battery (blue) and Extended Kalman Filter estimation (red)

We can observe that EKF algorithm converge really fast but when the SOC value becomes less than 20%, it starts to diverge, this can be attributed to difficulties during curve fitting as undervoltage protection distorts the voltage waveforms, making more difficult to obtain accurate estimations because it experiences abrupt changes (drops and rises) in its outputs. However, this problem of divergence is not important due to the fact that the Battery Management Systems (BMS) has a working range between 80% and 20% of SOC, this is because of the same reason from which our algorithm diverge when the battery is discharge below 20% of SOC. We also observed that this divergence was highly influenced by the sample time, because if we modified its value the convergence changed a lot. Thus, we could deduct that this problem was caused by the internal computations of Simulink and not by any problem in our algorithm.

Bearing this in mind, we decided that the discharges would last until the value of SOC was equal to 20% for both algorithms as we had seem that the SKF also experienced problems of divergence for lower values of SOC and both of them did not have any problem for values greater than 20%. As noted in *Table 2*, the EKF shows better performance than that of SKF for SOC estimation (which is our goal due to the fact that it cannot be directly measured) as it accounts for the nonlinearities in the battery behaviour.

Kalman type	SOC Estimation (%)		Terminal Voltage Estimation(V)	
	SOC error max	SOC absolute mean error	V_{term} error max	V_{term} absolute mean error
Simple	9,8 %	1,4 %	0,6580 V	0,0198 V
Extended	9,8 %	0,89%	0,6036 V	0,0198V

Table 2.- SKF and EKF estimation errors

The maximum error could seem very high but it is important to take into account that the algorithm is initialised with false values. If we measure the error when I has reached more accurate values (from t=60sec), the maximum error for the terminal voltage estimation is 0,0781V (SKF) or 0,7067V (EKF), and for the SOC estimation it is 5,56% (SKF) or 2,94% (EKF). The results from which we obtained this errors are:

- **Simple Kalman Filter (SKF)**

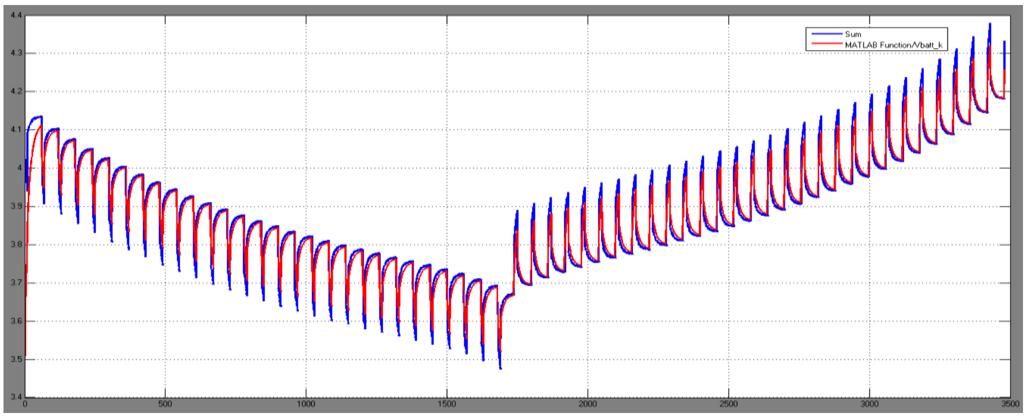


Figure 3.38.- Terminal voltage output: Comparison between real data from the reference battery (blue) and Simple Kalman Filter estimation (red)

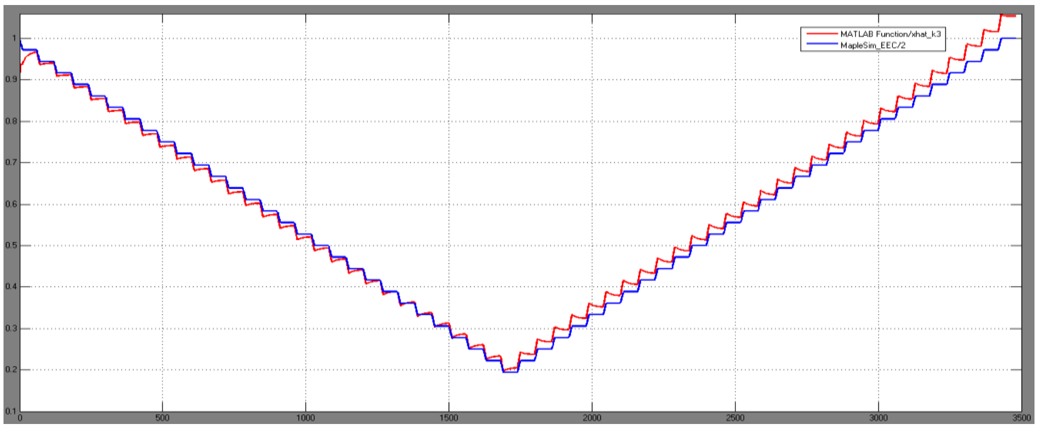


Figure 3.39.- SOC output: Comparison between real data from the reference battery (blue) and Simple Kalman Filter estimation (red)

- **Extended Kalman Filter (EKF)**

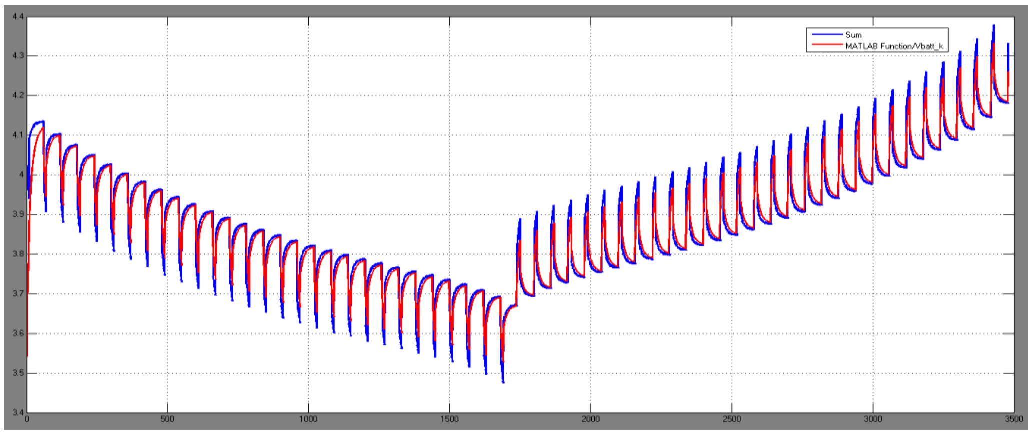


Figure 3.40.- Terminal voltage output: Comparison between real data from the reference battery (blue) and Extended Kalman Filter estimation (red)

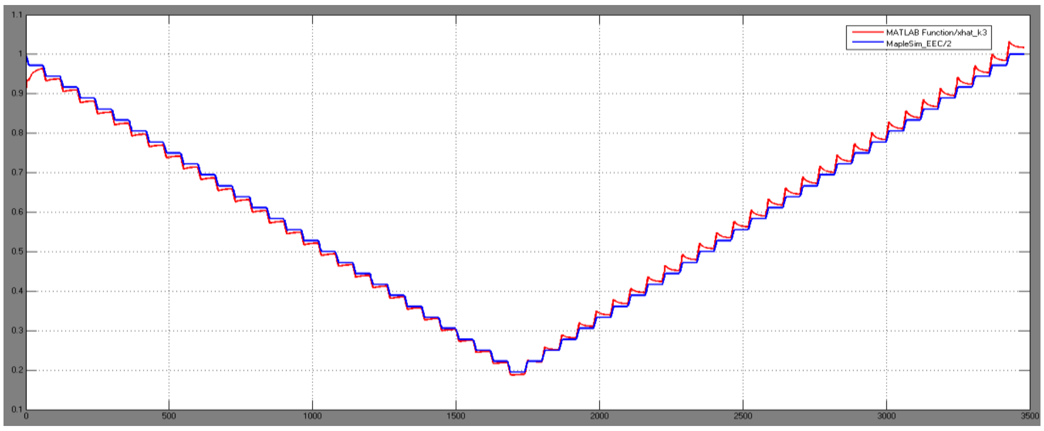


Figure 3.41.- SOC output: Comparison between real data from the reference battery (blue) and Extended Kalman Filter estimation (red)

As we have proceeded in the previous verifications of the estimation algorithms, we tested it convergence with other kind of inputs. Firstly, with a constant current of 1A and then with a constant current of 20A, in order to push our algorithm to the limit.

a) Constant current of 1A
   - **Simple Kalman Filter (SKF)**

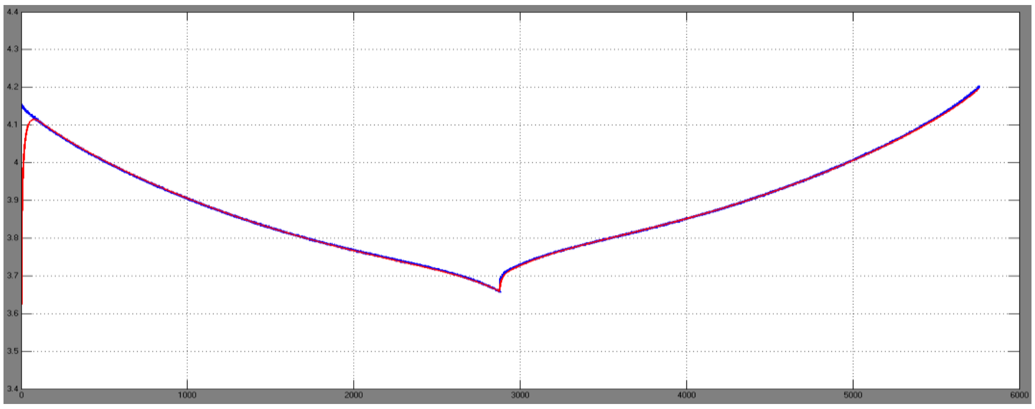


Figure 3.42.- Terminal voltage output: Comparison between real data from the reference battery (blue) and Simple Kalman Filter estimation (red)

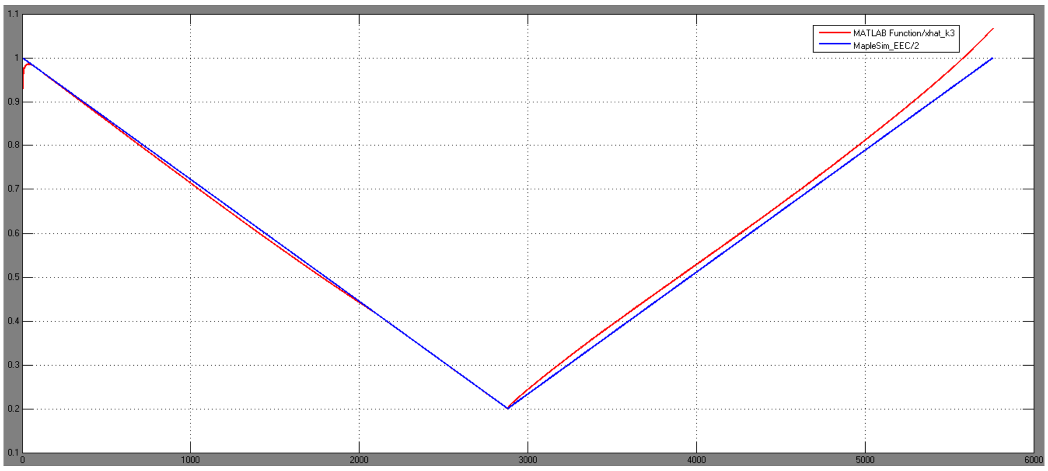


Figure 3.43.- SOC output: Comparison between real data from the reference battery (blue) and Simple Kalman Filter estimation (red)

- **Extended Kalman Filer (EKF)**

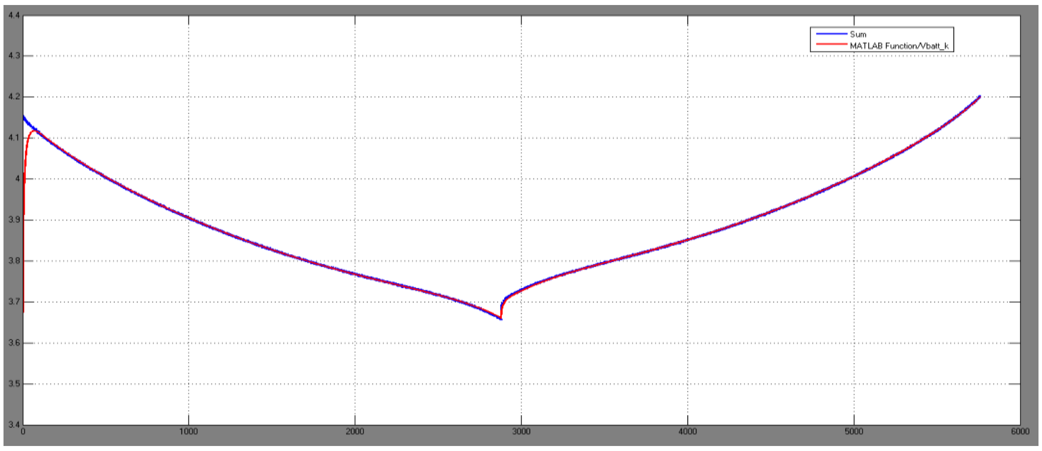


Figure 3.44.- Terminal voltage output: Comparison between real data from the reference battery (blue) and Extended Kalman Filter estimation (red)

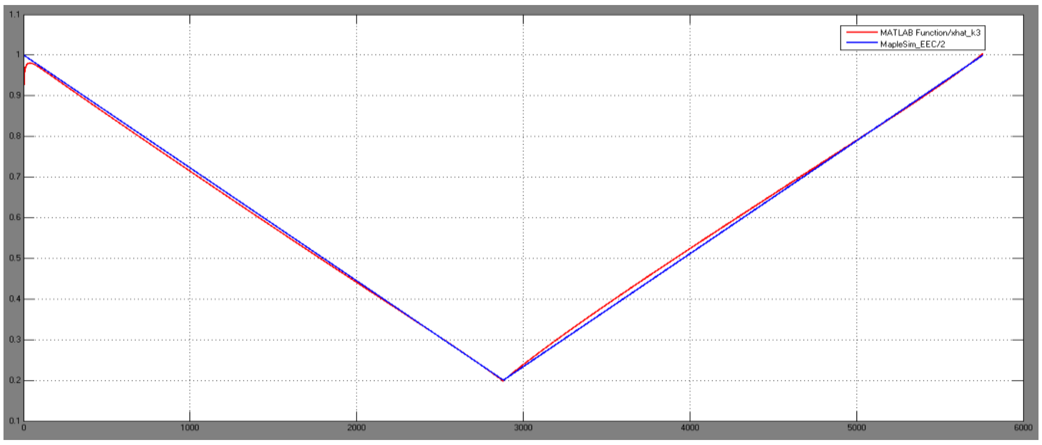


Figure 3.45.- SOC output: Comparison between real data from the reference battery (blue) and Extended Kalman Filter estimation (red)

Kalman type	SOC Estimation (%)		Terminal Voltage Estimation(V)	
	SOC error max	SOC absolute mean error	V_{term} error max	V_{term} absolute mean error
Simple	9,78%	1,37 %	0,7242 V	0,0026 V
Extended	9,79 %	0,64%	0,6696 V	0,0021 V

Table 3.- SKF and EKF estimation errors using a constant current of 1A as the input of our system

If we measure the error when it has converged despite the initial false values (from t=65 seconds), the maximum error for the terminal voltage estimation is 0,0174V (SKF) or 0,0148V (EKF), and for the SOC estimation it is 6,61% (SKF) or 1,44% (EKF). Focusing on mean errors displayed in *Table 3*, the error made in SOC estimation is highly reduced if EKF is used instead of SKF.

b) Constant current of 20A

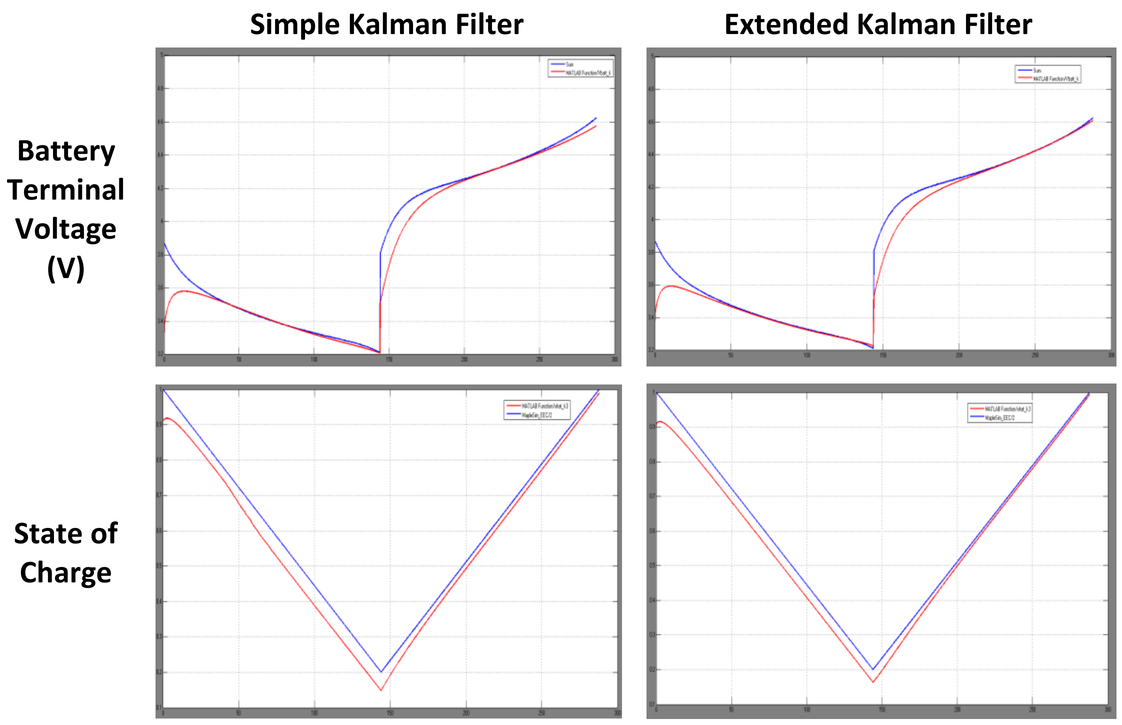


Figure 3.46.- Comparison between real data from the reference battery (blue) and Kalman Filter algorithm estimation (red) using constant current of 20A

Kalman type	SOC Estimation (%)		Terminal Voltage Estimation(V)	
	SOC error max	SOC absolute mean error	V_{term} error max	V_{term} absolute mean error
Simple	9,82%	3,56 %	0,5845 V	0,0385 V
Extended	9,83 %	2,37%	0,5302V	0,0366V

Table 4.- SKF and EKF estimation errors using as input a constant current of 20A

From *Table 4* we can deduce the same conclusion as before. Despite pushing our battery model, and therefore our algorithm, into the limit, the EKF keeps showing better performance than that of the SKF for battery SOC estimation as it reduce the mean error more than 1% which is highly important concerning the little errors both algorithms show.

Taking everything into account, we could confirm that these algorithms were ready to be implemented in applications that need real time SOC estimations, with the EKF showing better performance. From this point forward, once we were sure that the SOC estimation was accurate, we would tackle the estimation of another state of the battery that is crucial in order to develop a BMS, which is the state of health (SOH) of the battery.

4 STATE OF HEALTH ESTIMATION ALGORITHM

4.1 INTRODUCTION

State of health (SOH) measures the ability of a cell to store energy, source and sink currents, and retain charge over extended periods, relative to its initial or nominal capabilities [14]. The available charge stored is expected to fall during the lifespan of the battery due to cell usage, as the active material on the cell plates gradually degrades by mechanisms such as loss of plate active surface area, as a consequence of repeated dissolution and recrystallization, and growth of large inactive crystals within the plate structures. Such capacity loss can be deemed a loss of cell SOH. Early detection of SOH degradation would allow the battery pack to take remedial action just in time. For instance, they could include the application of conditional routines to the cell, so as to remove small sulphate crystals before they become inactive crystals, thereby restoring the cells capacity [14].

The commonly used indicators of battery SOH include battery capacity (the one we have chosen so as to measure this state of the battery), DC resistance and AC impedance. The SOH estimation methods mainly include durability model-based open-loop methods and battery model-based closed-loop methods [18]. The durability models describe the increase of solid-electrolyte interface (SEI) film resistance and battery terminal voltage, which allow the former type of methods to directly predict the changes in capacity fade and internal resistance. The latter comprises least-squares methods, Kalman filtering, other adaptive algorithms, such as fuzzy logic, to identify the battery capacity and internal resistance according to the operating data, and sample entropy method as different possibilities of SOH estimation. In Table *5* it is summarised the advantages and disadvantages of these SOH estimation methods.

Most of the above mentioned methods and those explained in section 3.1, were developed for either SOC or SOH estimation but not for both of them. The close relation between SOC and SOH was overlooked. Battery degradation has a great influence on the accuracy of SOC estimation. As battery degrades, those algorithms that only perform SOC estimation may lead to large errors. The inaccurate SOC estimations in turn may mislead the battery SOH calibration. Thus, simultaneous SOC and SOH estimations is quite beneficial. Comparing battery SOC and SOH variations, battery SOH typically change much more slowly, being necessary the use of multi-timescale state estimators. The multi-scale EKFs are used to estimate SOC and SOH, and the capacity estimation is periodically introduced in SOC update equation. However, the determination of the two time scales is heavily dependent on tuition and calibration. Moreover, this multi-timescale algorithm has a heavier computational intensity, which makes it more difficult to implement in real time applications. Hence, instead of an algorithm of this characteristics, we decided to implement an extended version of the EKF developed in section 3.3, adding two new parameters to the state vector (x): the internal resistance

(R_0) and the inverse of the available usable capacity of the battery (C_{use}), which is considered to change during the service life, contrary to what we established in section 2.1, in order to be more closer to the reality. This modified EKF algorithm will be used in the next section in order to estimate the battery SOH.

State of health (SOH) estimation			
Classification	**Method**	**Advantage**	**Disadvantage**
Durability model-based open-loop method	Durability mechanism	Comprehensive understanding	Complex, need accurate input parameters
	Durability external characteristic	Simple and easy to predict capacity fade and internal resistance increment	Based on a large number of experiments
Battery model-based close-loop method	DC resistance	Simple	Not accurate, sensitive to disturbances
	AC impedance	Accuracy	Complex
	Extended Kalman Filter (EKF)	Quite easy to implement, accurate	Sensitive to modelling accuracy
	Fuzzy logic	Accurate	Slow convergence
	Sample entropy	Simple	Need large amount of data
	Discharge voltage	Easy	Not accurate
	Adaptive control system	Online applications	Sensitive to modelling accuracy

Table 5.- Advantages and disadvantages of existing SOH estimation methods [18]

4.2 STATE OF HEALTH ESTIMATION BASED ON FIFTH-ORDER EXTENDED KALMAN FILTER USING DATA FROM MAPLESIM

The fifth-order EKF algorithm that we have developed follows the same stages than the EKF from section 3.3, which are now shown in Figure 4.1, particularised for the new estimation algorithm. The modifications were pointed out in the previous section. The enlargement of the state vector implied the redefinition of all matrices that take part in this algorithm, as it can be seen in (4.1) and (4.2).

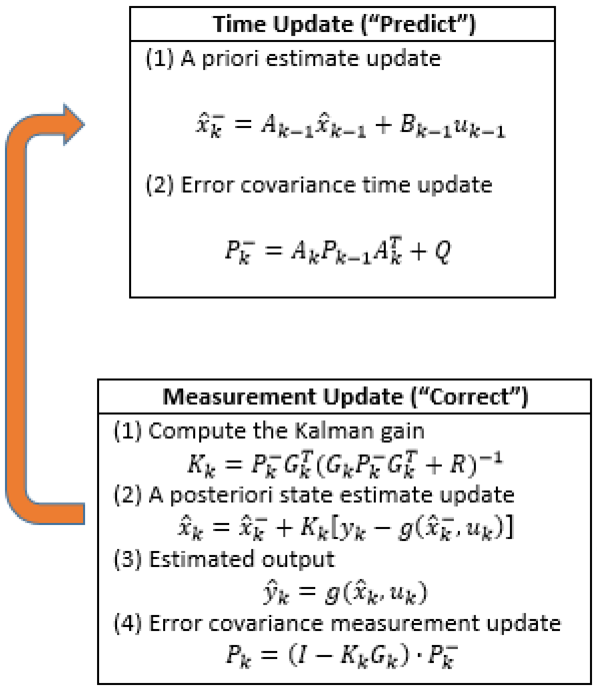

Time Update ("Predict")
(1) A priori estimate update
$\hat{x}_k^- = A_{k-1}\hat{x}_{k-1} + B_{k-1}u_{k-1}$
(2) Error covariance time update
$P_k^- = A_k P_{k-1} A_k^T + Q$
Measurement Update ("Correct")
(1) Compute the Kalman gain
$K_k = P_k^- G_k^T (G_k P_k^- G_k^T + R)^{-1}$
(2) A posteriori state estimate update
$\hat{x}_k = \hat{x}_k^- + K_k[y_k - g(\hat{x}_k^-, u_k)]$
(3) Estimated output
$\hat{y}_k = g(\hat{x}_k, u_k)$
(4) Error covariance measurement update
$P_k = (I - K_k G_k) \cdot P_k^-$

Figure 4.1.- Schematic of the fifth-order EKF estimation algorithm

Where:

$$A_k = \begin{bmatrix} e^{\frac{-Ts}{R_{1,k}\cdot C_{1,k}}} & 0 & 0 & 0 & 0 \\ 0 & e^{\frac{-Ts}{R_{2,k}\cdot C_{2,k}}} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & \frac{-T_s \cdot I_{batt,k}}{3600} \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} ;\ D_k = [0]\,;$$

$$B_k = \begin{bmatrix} R_{1,k}\cdot\left(1 - e^{\frac{-Ts}{R_{1k}\cdot C_{1k}}}\right) \\ R_{2,k}\cdot\left(1 - e^{\frac{-Ts}{R_{2,k}\cdot C_{2,k}}}\right) \\ 0 \\ 0 \\ 0 \end{bmatrix} ;\, C_k = \begin{bmatrix} -1 & -1 & \frac{\partial V_{oc,k}}{\partial x_{3,k}} & I_{batt,k} & 0 \end{bmatrix}^{2} \tag{4.1}$$

$$x_k = \begin{bmatrix} V_{1,k} \\ V_{2,k} \\ SOC_k \\ R_{0,k} \\ {}^{1}/{C_{use,k}} \end{bmatrix} \tag{4.2}$$

[2] Voc is defined in (2.4) and Ibatt is the input of our system (u = [Ibatt,k]) as it was defined in (2.10).

With this algorithm we are able not only to obtain the estimation of battery SOC but also the SOH estimation. The latter can be calculated indirectly using the fifth component of the state vector. The expression (4.3) allow us to estimate the battery SOH from the fifth state.

$$SOH = \frac{C_{use}}{C_{Nominal}} \cdot 100\% \qquad (4.3)$$

Where C_{use} is the actual capacity, which is the fifth component of the state vector and therefore it is going to change during the whole simulation, contrary to what it was established in section 2.1, and $C_{nominal}$ is the capacity established by the manufacturer in the main specifications of the battery, and in our case it is equal to the value of the capacity of the cell of the reference battery from MapleSim, that is, $C_{nominal}$ = 1 Ah.

Using in MapleSim the same current profile from the hybrid pulse test that it was used in [18], which is depicted in *Figure 4.2*, we collected some data from the reference battery and we saved it in an excel file. In this file it was saved the time in which each data was collected, the value of the current at this time, the terminal voltage of the cell, and the state of charge (SOC) and state of health (SOH) of the battery cell, so as to export all this data to Matlab environment. Then, we assigned this data to its corresponding variables in our Matlab script, where it was implemented the fifth-order EKF, in order to use them as a reference for our estimation algorithm.

As depicted in *Figure 4.3*, the cell terminal voltage estimation is highly accurate once the algorithm has found its true value, being the error since t=1222,4 seconds inferior to its mean value. Focusing on SOC estimation (*Figure 4.4*), despite the increase of the estimation error at the end of the simulation, this error is still very small (about 2,9%[3]), therefore this algorithm is still accurate to estimate the state of charge.

Concerning the SOH estimation, in Figure 4.7 it is remarkable that the SOH estimation error becomes smaller than the mean error at the end of the simulation, which means that in the end the SOH estimation converges to the real value. As a matter of fact, we can verify this phenomenon observing the Figure 4.5 and Figure 4.6.

Finally, due to the fact that MapleSim is not able to measure the voltage across the different components of the circuit that forms the EEC model, neither does it track the variation of the battery internal resistance, we decided to compare the estimation of these states with that of the nonlinear model, which does not account for the effect of the battery aging. From Figure 4.8 and the next one, it is visible that the estimation of these two states of the battery converge relatively faster to that of the nonlinear model.

[3] Considering that SOC is given in percentage, despite the fact that it is represented between 0 and 1.

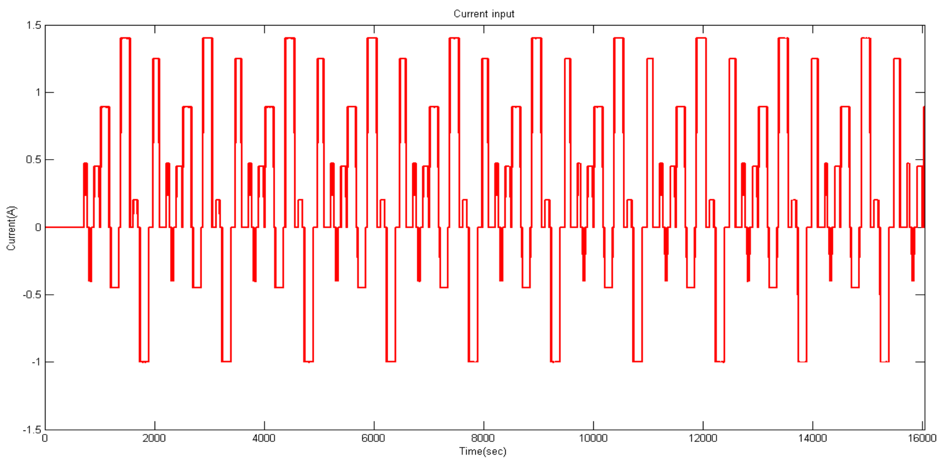


Figure 4.2.- Input: Current profile from the hybrid pulse test at 25°C

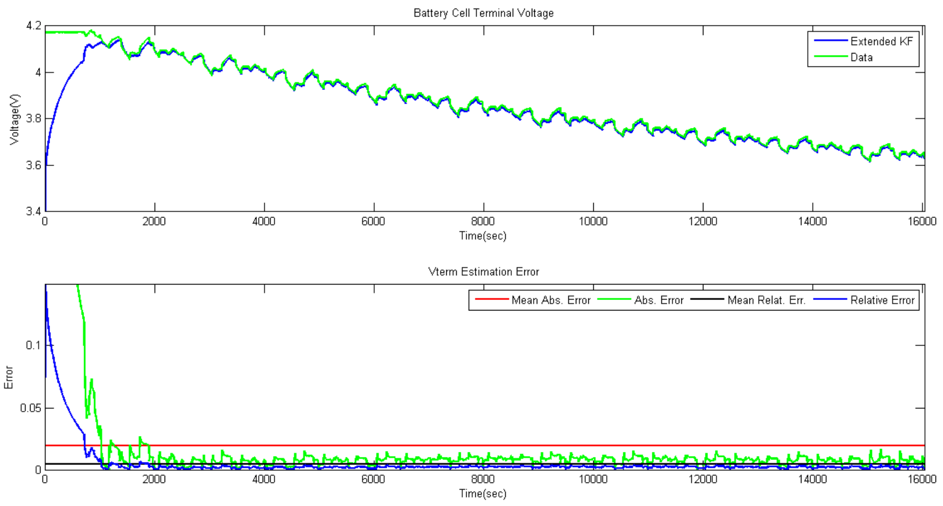


Figure 4.3.- Above: Comparison between Vterm from real data (green) and EKF estimation (blue). Below: Vterm estimation error

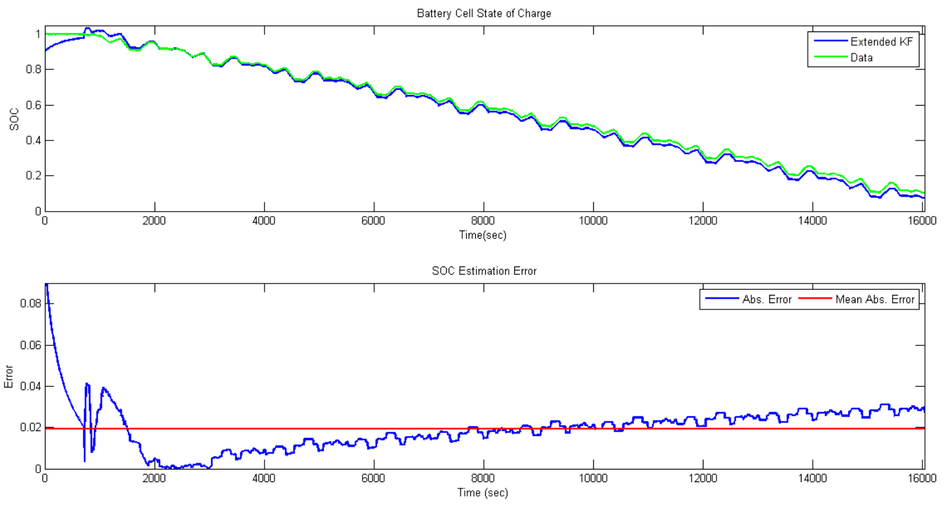


Figure 4.4.- Above: Comparison between SOC from real data (green) and EKF estimation (blue). Below: SOC estimation error

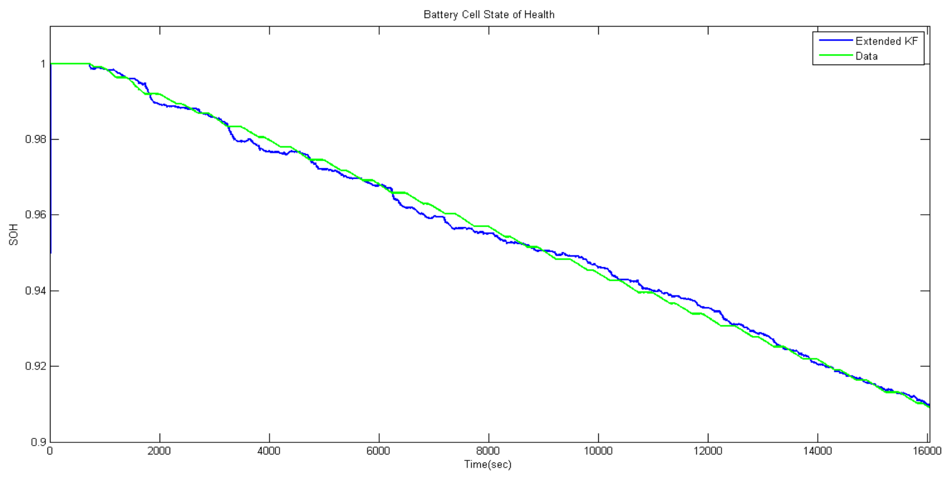


Figure 4.5.- Comparison between real SOH data from MapleSim (green) and SOH EKF estimation (blue)

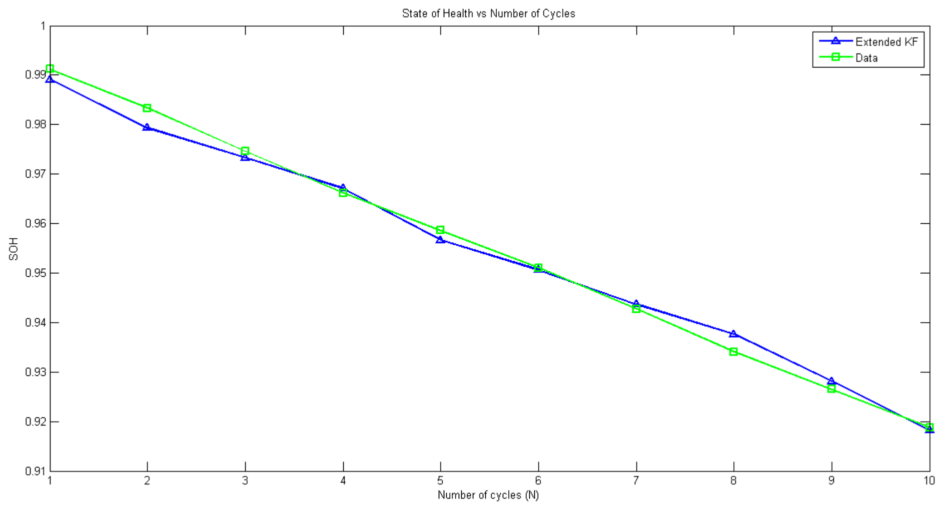


Figure 4.6.- Comparison of the SOH value in each cycle between real data from MapleSim (green) and EKF algorithm (blue)

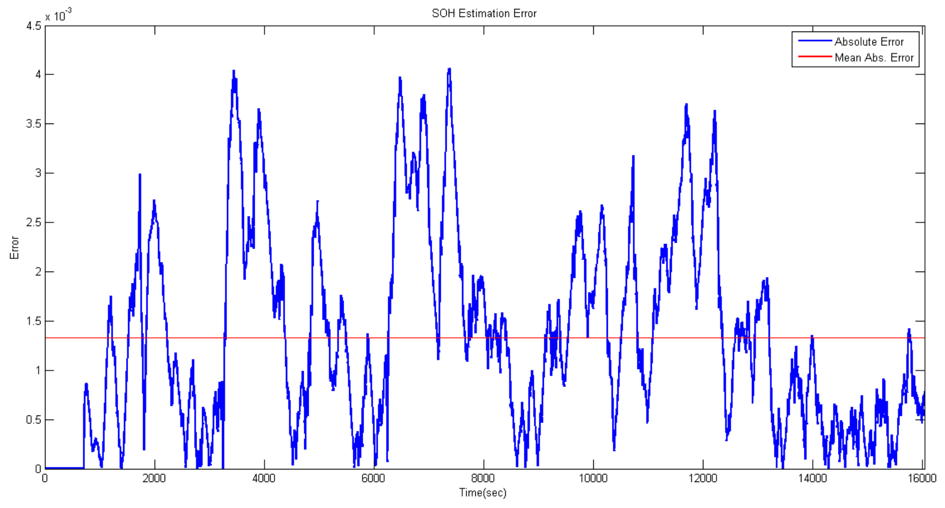


Figure 4.7.- SOH estimation error: Absolute Error (blue). Mean Error (red)

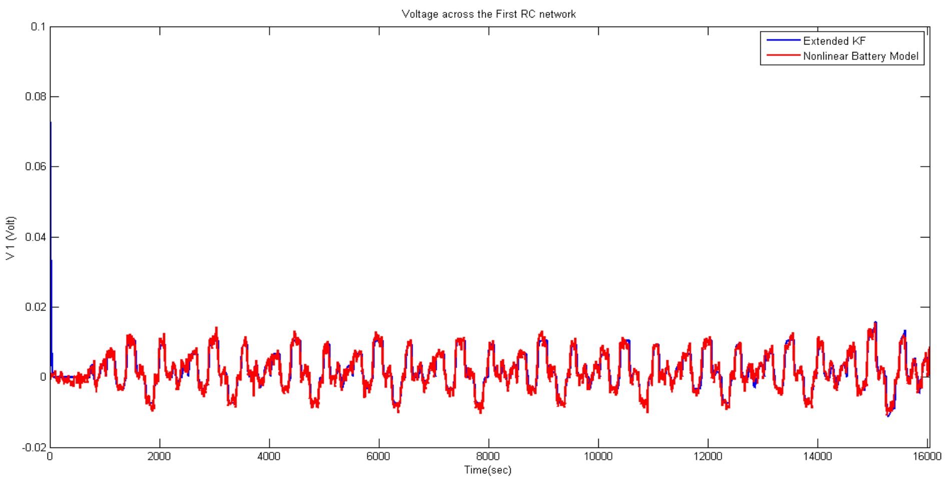


Figure 4.8.- Comparison between nonlinear system (red) and Extended Kalman Filter (blue) voltage drop estimation across the first RC network

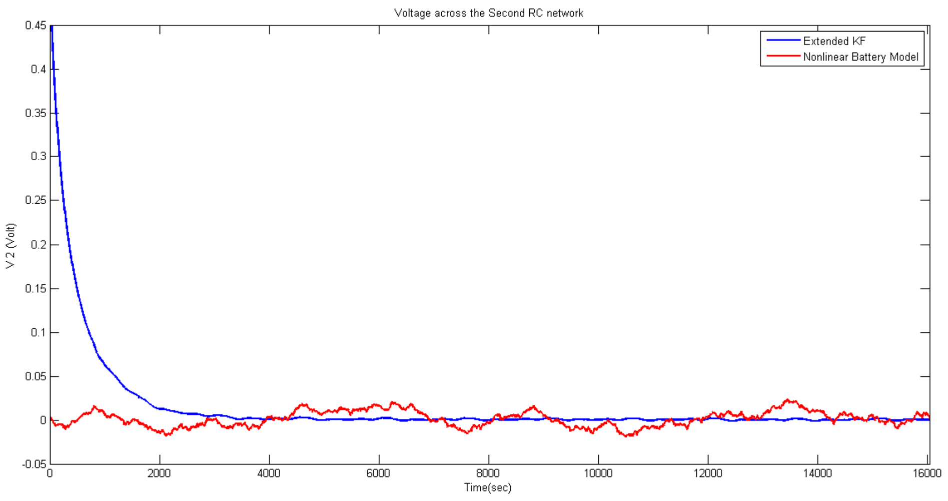


Figure 4.9.- Comparison between nonlinear system (red) and Extended Kalman Filter (blue) voltage drop estimation across the second RC network

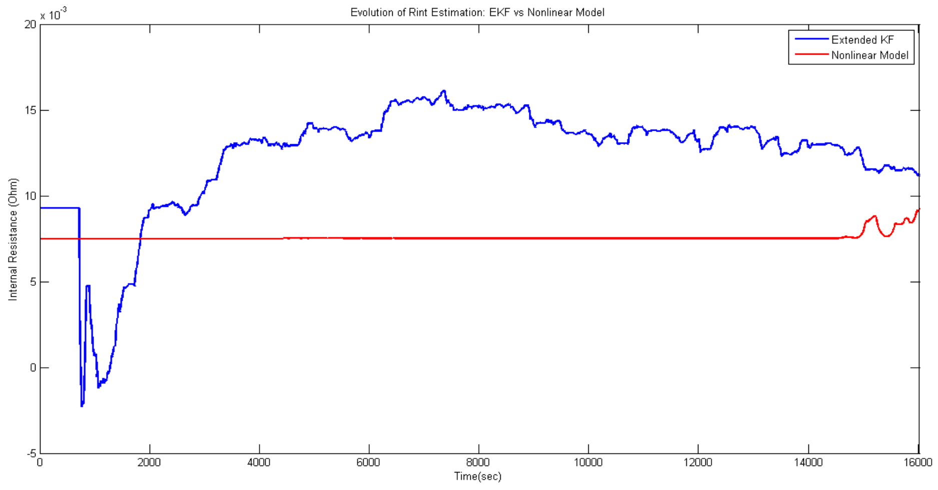


Figure 4.10.- Internal resistance estimation result from the EKF (blue) and the nonlinear model (red)

From the picture above it can be seen that in the fifth-order EKF the value of the internal resistance changes throughout the simulation, ending the simulation with a higher value of the resistance, as it was to be supposed because of the accumulation of inactive crystals that makes more difficult the flow of electrons inside the battery.

4.3 SOH AND SOC ESTIMATION USING FIFTH-ORDER EKF IN SIMULINK

Following the procedure explained in section 3.4, we obtained the Simulink block shown in Figure 4.11, which was implemented in Simulink using a model very similar to that represented in Figure 3.32, but in this case we used the fifth-order EKF instead of the original EKF, in order to simultaneously estimate the SOC and SOH. We did not achieve the co-simulation between Simulink and Matlab, because the block generated from MapleSim only works in time continuous simulations and the EKF only works with discrete time. Therefore, it was necessary to modify the initialisation of the error covariance matrix P, and that of the noise covariance matrices Q and R.

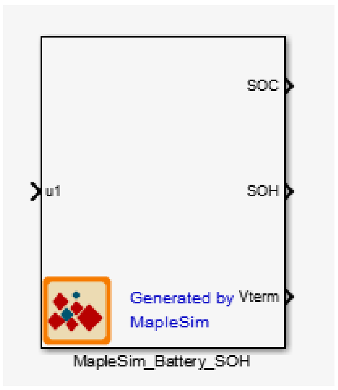


Figure 4.11.- Simulink block created from the battery in MapleSim in order to estimate the battery SOC and SOH

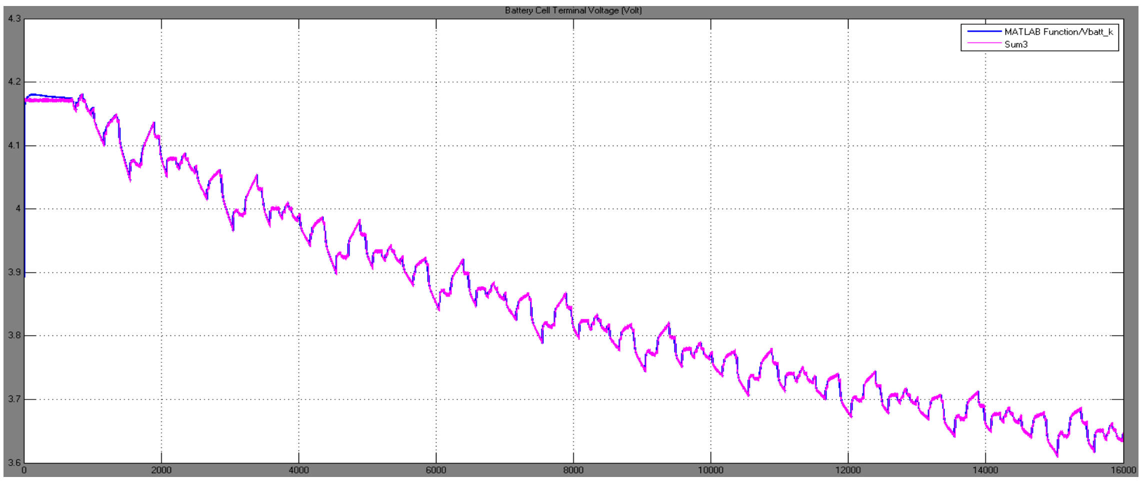


Figure 4.12.- Battery terminal voltage: Comparison between real data from the reference battery (pink) and Fifth-order EKF estimation (blue)

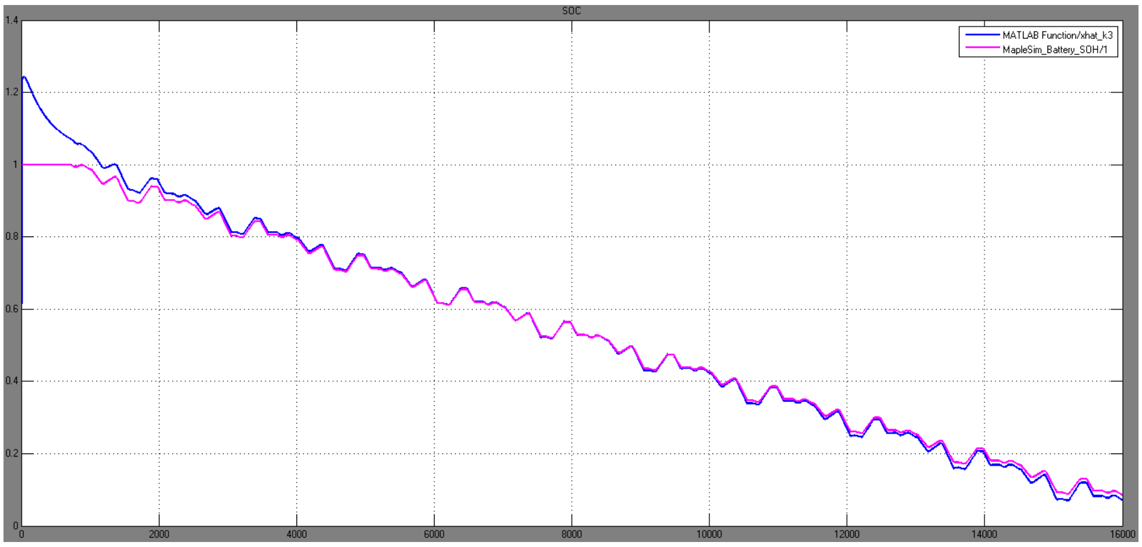


Figure 4.13.- SOC: Comparison between real data from the reference battery (blue) and Simple Kalman Filter estimation (red)

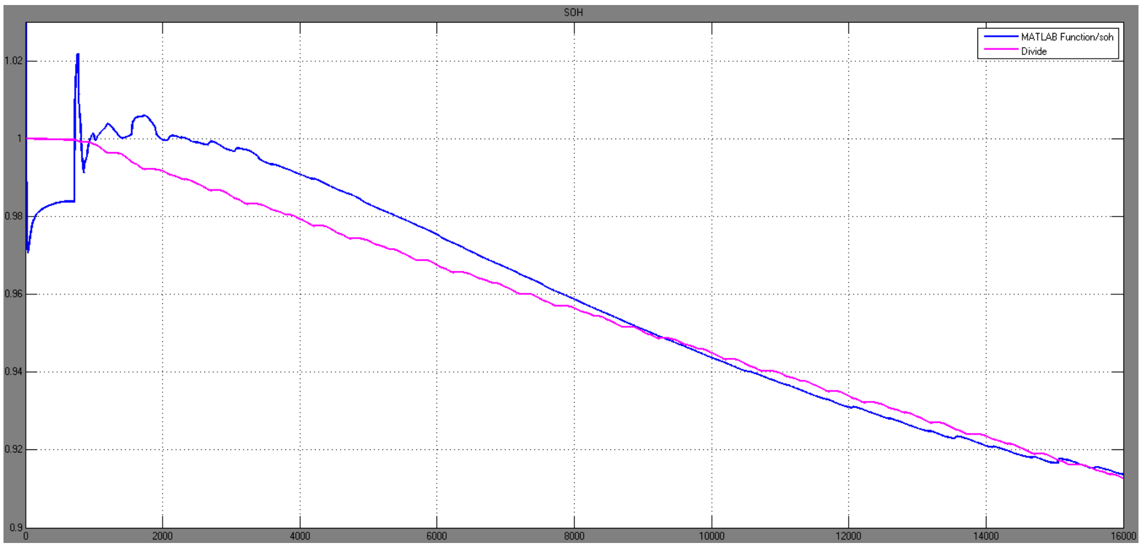


Figure 4.14.- SOH: Comparison between real data from the reference battery (blue) and Simple Kalman Filter estimation (red)

From Figure 4.12 to Figure 4.14, we can observe the estimation results obtained from the simulation in Simulink. It is visible that the convergence of the estimations to the real value is assured for all cases. In particular, the SOH estimation seems to converge slowly but once it reaches the real value it tracks the true SOH really well. Despite the fact that at a first glance it may seem that the SOC estimation error starts to increase at the end, the reality is that the error is so small that it can be neglected. In Table 6 all the estimation errors for the three variables represented above are shown.

Kalman type	SOC Estimation (%)		SOH Estimation (%)		V_{term} Estimation (V)	
	SOC error max.	SOC abs. mean err.	SOH error max.	SOH abs. mean err.	V_{term} err. max.	V_{term} abs. mean err.
5th-order EKF	24,34 %	1,54 %	7,3 %	0,56 %	0,0901 V	0,0011 V

Table 6.- Fifth-order EKF estimation errors

The maximum SOC error is extremely high due to a bad initialisation of the algorithm, in fact if we start to track the error from t=1.000 seconds, its maximum value decrease to 5,06% and if we start to track it later, this value would decrease even more. This phenomenon help us to confirm the claim stated at the beginning of the paragraph.

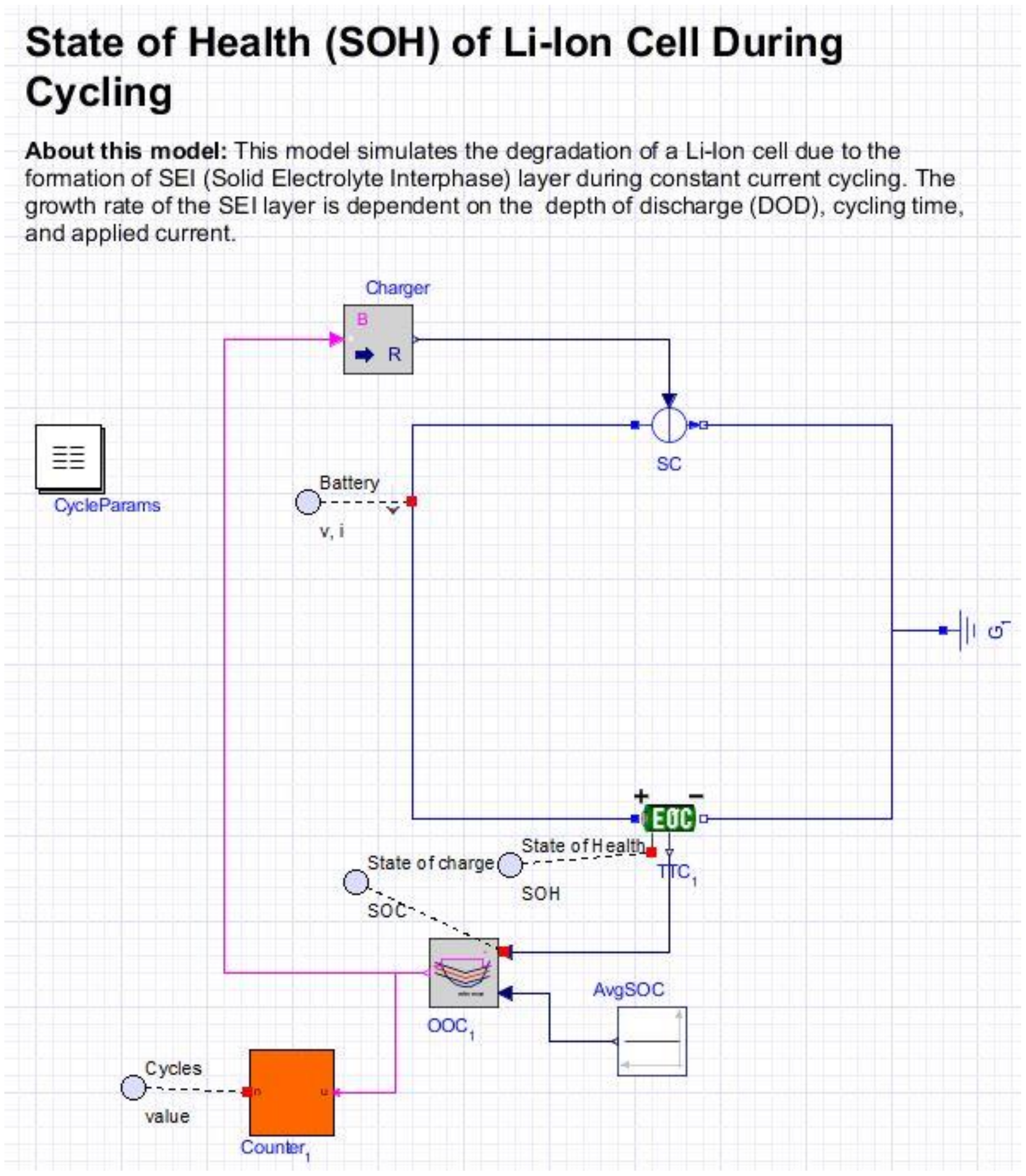
With regard to mean errors, they are small enough to consider that the algorithms provides good estimations. Despite the increase of the SOC estimation mean error compared with the error of the third-order EKF algorithm, the possibility of tracking the SOH of the battery and the reduction in the terminal voltage estimation mean error, make that the advantages of this algorithm outweigh its disadvantages. And therefore, we consider that this algorithm provides a correct simultaneous estimation of SOC and SOH.

4.4 MEASUREMENT OF THE STATE OF CHARGE AT DIFFERENT TEMPERATURES

So far, the temperature of the battery cell has been considered constant and equals to 25 °C. In this section we want to study the effect of the temperature in the degradation of the battery cell. In order to do so, the same battery has to be subjected to the same input current at the different temperatures that we want to study.

When we did the simulation, we used a constant current with an amplitude of 20 A as the input of our system in order to discharge and charge the battery cell. As battery cell aged, it started to become deteriorated, therefore it had a faster discharge. To solve this problem we decided to implement a model in MapleSim in which we could detect when the SOC was inferior to 10 % or equal to 100 % and inverse the sense of the input current. So as to achieve this we used the block "On Off Controller" (OOC_1 in *Figure 4.15*), which sets the output signal to true when the input signal, SOC, falls below the reference signal, Avg SOC, plus half of the hysteresis [19]. This Boolean signal becomes the input of a block called "Boolean to Real" ("Charger" in our MapleSim model), which converts the Boolean input to a real value preset by the parameters, *Real True* and *Real False,* -20 and 20 respectively in our case. The equation of this component is [19]:

$$y = \begin{cases} realFalse & if \quad input = false \\ realTrue & if \quad input = true \end{cases} \qquad (4.4)$$



State of Health (SOH) of Li-Ion Cell During Cycling

About this model: This model simulates the degradation of a Li-Ion cell due to the formation of SEI (Solid Electrolyte Interphase) layer during constant current cycling. The growth rate of the SEI layer is dependent on the depth of discharge (DOD), cycling time, and applied current.

Figure 4.15.- MapleSim model layout

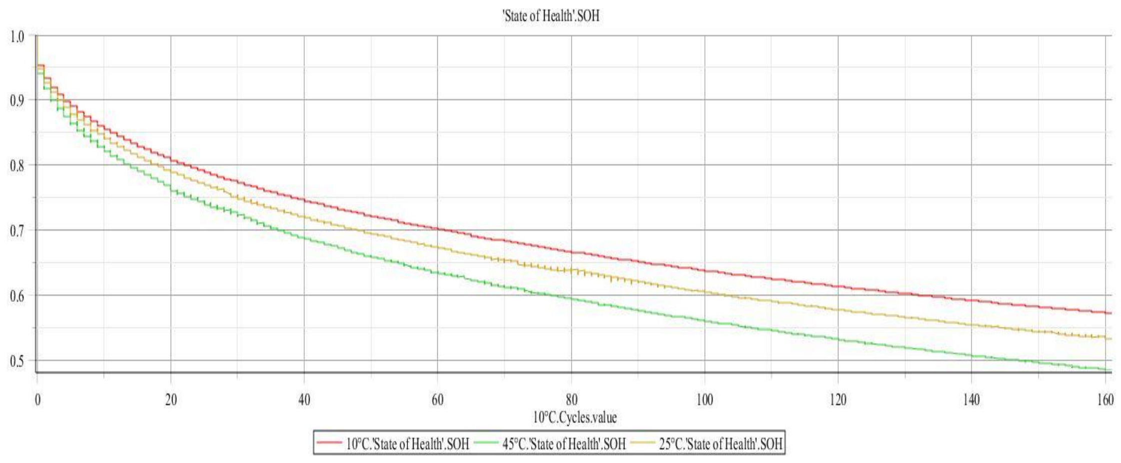


Figure 4.16.- Battery cell degradation for different temperatures. Where the y-axis represent SOH and x-axis the number of cycles

Simulation was carried out at different battery cell temperatures, 10°C, 25°C and 45°C. From *Figure 4.16* it is visible that the higher the cell temperature is, the more deterioration it will experience, and therefore the higher the decreasing of the state of health (SOH) is. Thus, it is also important to control the temperature of the battery so as to assure that the temperature does not influence a lot in the degradation of the battery and to make sure that the lifespan of the battery coincides with that of the application that it would be implemented in.

4.5 COMPARATION BETWEEN THE DEGRADATION OF ONE CELL AND A REAL BATTERY WITH EIGHT CELLS

In real life, batteries are not composed of only one cell as we have supposed during all the previous sections, in fact they usually consist of a high number of cells connected in series in order to obtain a higher power. In this section we are going to verify this improvement in battery performance using a battery pack instead of one cell. Actually, in equation (2.2) this assertion was introduced (where N_{cell} was the number of cells connected in series). Therefore, from this equation we deduce that the more cells the battery has, the higher power it will be able to supply.

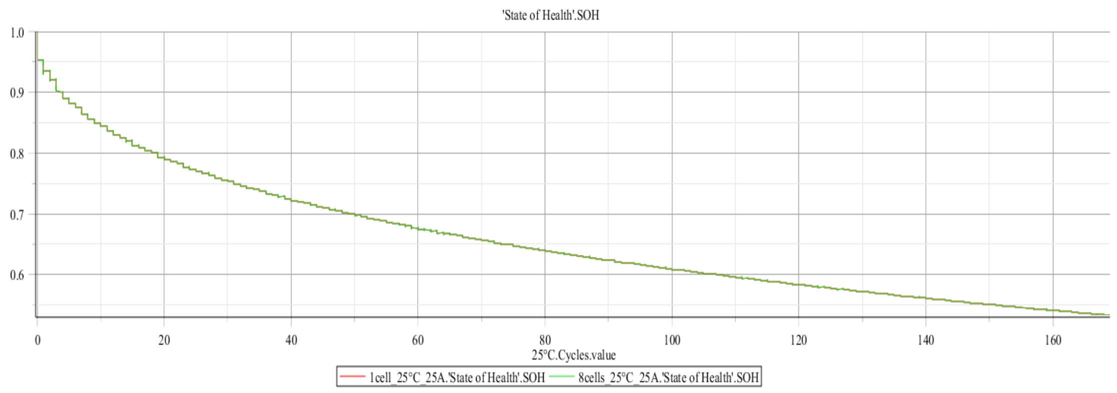


Figure 4.17.- SOH of one cell (red) and of eight cells connected in series (green)

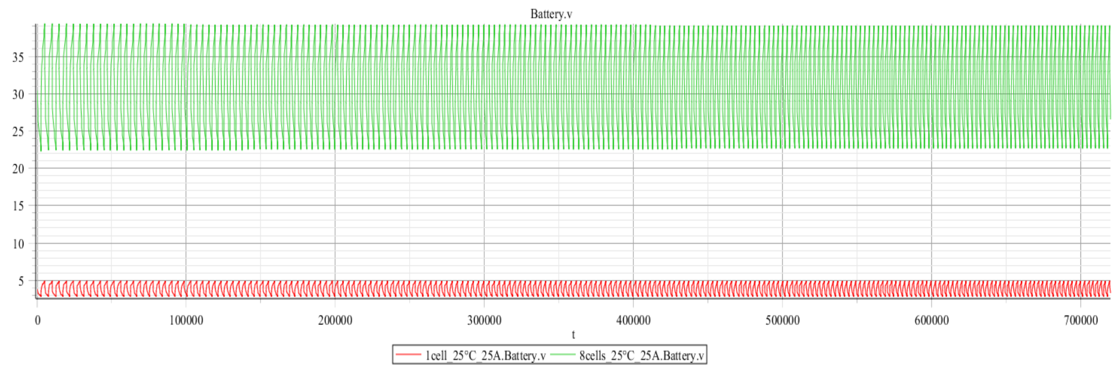


Figure 4.18.- Cell terminal voltage (red) and battery pack terminal voltage (green)

Comparing the behaviour of one cell with that of a battery pack, which consist of 8 cells, it can be seen that the degradation of both is the same (*Figure 4.17*), i.e., both experience the same SOH decrease, as it was expected bearing in mind that we are working in a simulation environment. Nevertheless, the battery pack provides a higher voltage level (*Figure 4.18*). Thus, from the power equation ($P = V \cdot I$), using the same input current, we can obtain a higher power with a battery pack than with only one cell. This is the reason why nowadays all the applications that need a battery have a battery pack.

However, as we stated in the previous paragraph, the fact that the rate of charge and discharge was the same is due to the ideal simulation environment. In real applications, the battery packs are provided with cells of slightly different characteristics, so the charging/discharging capability of weakest cell is the limiting factor. Therefore, the characteristics of this cell are important during operation in order to avoid overcharging/over discharging [13].

In Figure 4.19 it is shown the waste of capacity if we use a battery pack where it exists some variations between the two cells. This shows us more clearly the importance of an exhaustive control of the cells that compose the battery pack in order to balance their characteristics. If the balancing device is not efficient enough, the real SOC of the battery pack will be related to the real performance of this balancing device. If there is no balancing device or with dissipation, there will be some waste capacity [3].

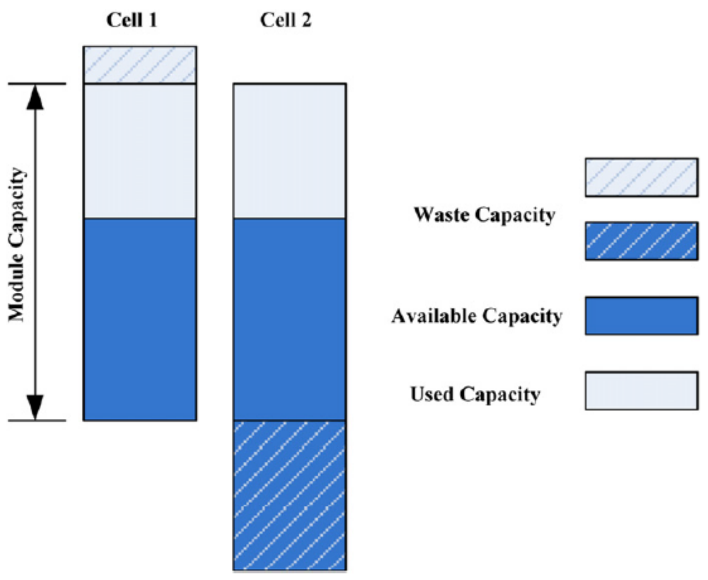


Figure 4.19.- The waste capacity and remaining capacity of a battery module (take a battery module of two cells as example) [3]

5 LEAD-ACID BATTERY MODELLING

Until now, we have considered Lithium-ion battery as the one with the best characteristics based on the different comparisons that it has been made in the literature. However, at this point, we thought that it would add some value to this report if we compared the performance of the Lithium-ion battery that it has been used for all the simulations with that of a Lead-Acid battery (which was widely used before the creation of the Lithium-ion batteries due to its good performance, despite its heavy weight) with the same specifications, that is, with the same cell resistance (R_{cell} = 0,0075 Ω), and capacity (CA = 1 Ah).

The simulation was carried out in MapleSim, using the same input current for both batteries (Figure 5.1). From the simulations results we can confirm what we said in the previous paragraph. As the batteries were configured with the same specifications their charging/discharging capability is the same (*Figure 5.2*). However, the Lithium-ion battery provides a higher voltage level than the Lead-Acid battery (*Figure 5.3*), and therefore the power that the former can supply is higher than that of the latter.

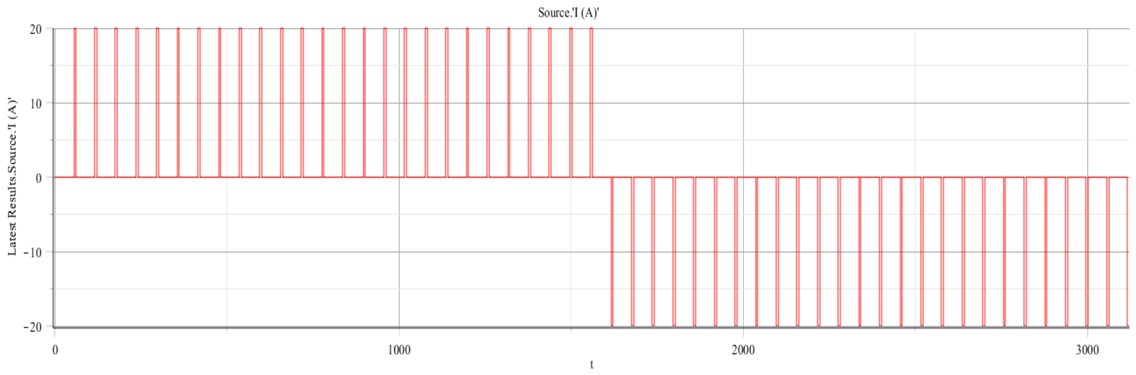


Figure 5.1.- Common input for both kinds of batteries: pulse train with an amplitude of 20A, a T=60 sec. and a width of 10% of the time period

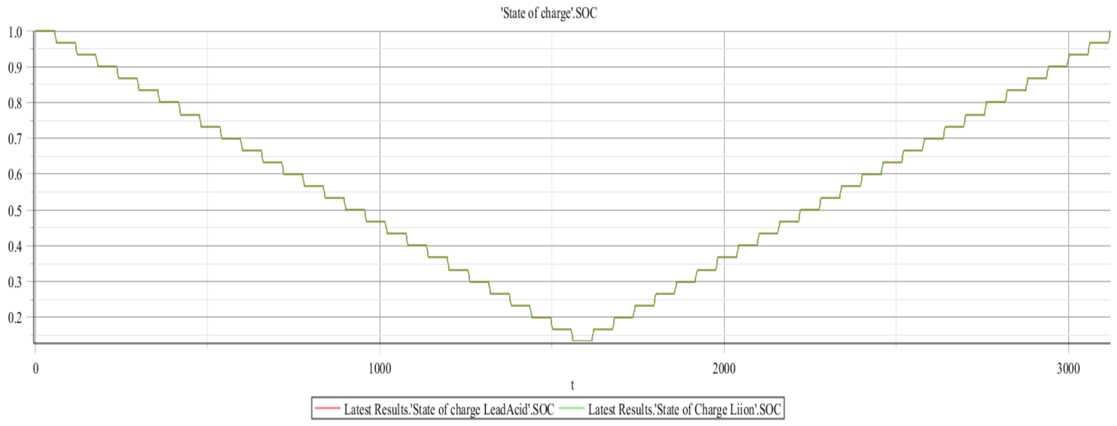


Figure 5.2.- SOC of Lead-Acid battery (red) and SOC of Lithium-ion battery (green)

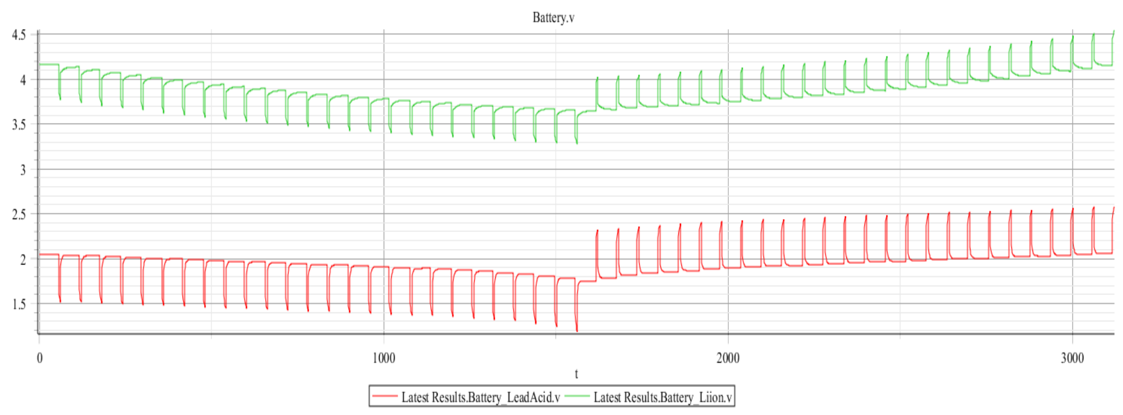


Figure 5.3.- Lead-Acid battery terminal voltage (red) and Lithium-ion battery terminal voltage (green)

Thus, from simulation results we can conclude that using a battery with the same rate of charging/discharging (i.e., the SOC of both batteries experience the same changes), if we introduce the same input, Lithium-ion battery provides higher power. Therefore it will be widely used for those applications that requires great power, such as the electric vehicle, which is growing fast this recent years.

5.1 STATE ESTIMATION FOR LEAD-ACID BATTERY

The last step of this report would be the estimation of the state of the Lead-Acid battery employed in the previous section using the third-order EKF. In order to achieve that we exported the data from MapleSim and we save it in an Excel file. Then, also from MapleSim, we obtained the expressions that define the EEC model. This model for this kind of battery is the same as the one which was depicted in Figure 2.10. However, as the battery was of different nature, it was necessary to modify the expressions that define the value of each component, which still follow the form of the equation (2.3):

$$
\begin{aligned}
R_{0,k} &= 0{,}025 \cdot e^{-24 \cdot SOC_k} + 0{,}012 \\
R_{1,k} &= 0{,}05 \cdot e^{-29 \cdot SOC_k} + 0{,}0074 \\
R_{1,k} \cdot C_{1,k} &= -3 \cdot e^{-13 \cdot SOC_k} + 3 \\
R_{2,k} &= 1 \cdot e^{-155{,}2 \cdot SOC_k} + 0{,}008 \\
R_{2,k} \cdot C_{2,k} &= -31000 \cdot e^{-88 \cdot SOC_k} + 710
\end{aligned}
\qquad (\ 5.1\ )
$$

The expression of the OCV was obtained using the curve fitting parameter tool from Matlab:

$$V_{OC,k} = 20{,}23 \cdot e^{-0{,}01499 \cdot SOC_k} - 18{,}487 \qquad (\ 5.2\ )$$

Once the redefinition was made, we assigned the data from the Excel file to their respective variables in our Matlab script. Using the current shown in *Figure 5.4* as the input of our system, the simulation results are depicted from *Figure 5.5* to *Figure 5.8*. It is visible that the state estimations are highly accurate, being the mean error of SOC estimation 0,92%[4] (Table 7). In fact, in the end the SOC estimation error starts to decrease below the value of the mean SOC estimation error. It is not possible to measure the other two states of the batter EEC model, V_1 and V_2, in MapleSim. Hence, we decided to compare its estimations with that of the nonlinear model, which describes the battery behaviour in a suitable manner.

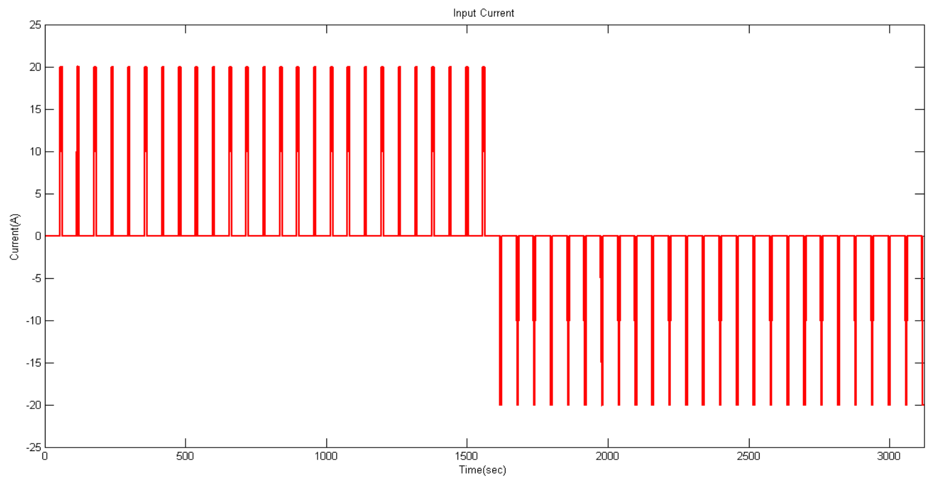


Figure 5.4.- Input current

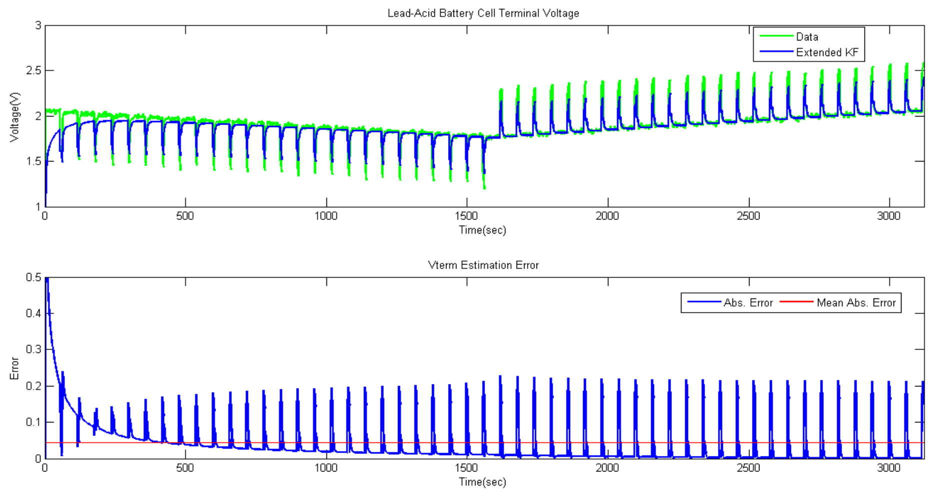


Figure 5.5.- Above: Comparison between Vterm from real data (green) and EKF estimation (blue). Below: Vterm estimation error

[4] Considering that SOC is expressed in %. Nevertheless in Figure 5.6 it is depicted as a variable that varies between 0 and 1.

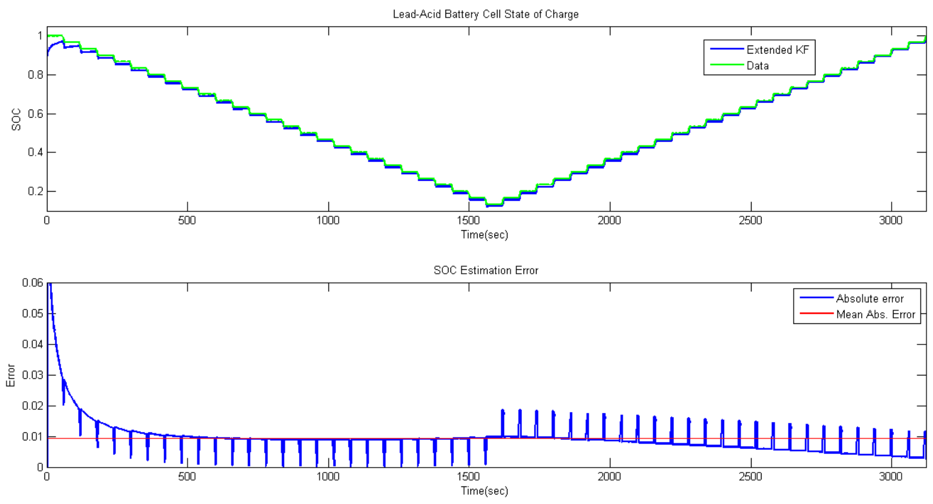


Figure 5.6.- Above: Comparison between SOC from real data (green) and EKF estimation (blue). Below: SOC estimation error

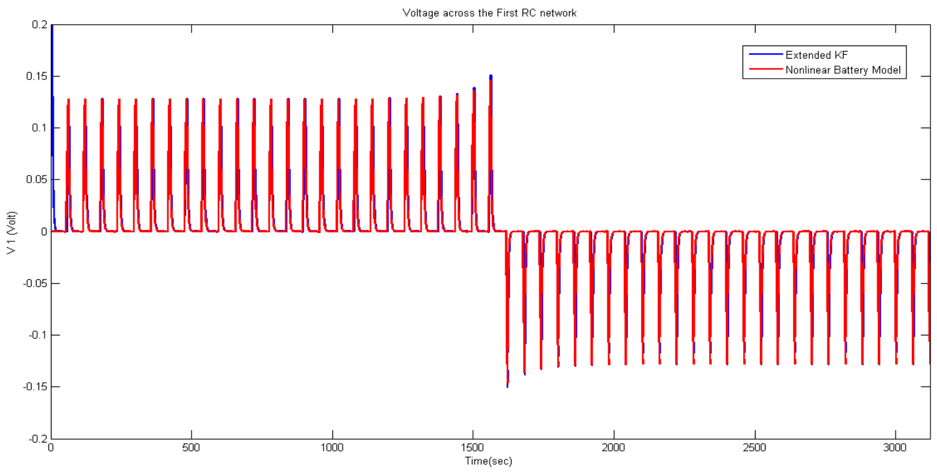


Figure 5.7.- Comparison between nonlinear battery model (red) and Extended Kalman Filter (blue) voltage drop estimation across the first RC network

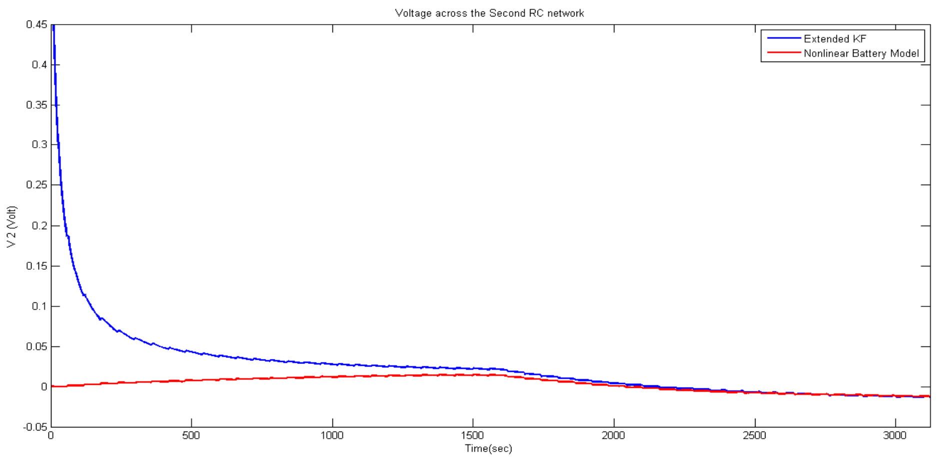


Figure 5.8.- Comparison between nonlinear battery model (red) and Extended Kalman Filter (blue) voltage drop estimation across the second RC network

Kalman type	SOC Estimation[4] (%)		Terminal Voltage Estimation(V)	
	SOC error max	SOC absolute mean error	V_{term} error max	V_{term} absolute mean error
Extended	8,67 %	0,92%	0,9412 V	0,0447 V

Table 7.- Estimation Errors

The errors made are of the same order of magnitude than that of the previous applications of the third-order EKF, therefore we can confirm that this algorithm is suitable to estimate the state of batteries from different nature and characteristics, provided that we define accurately the value of each EEC model component.

6 CONCLUSIONS

6.1 PRESENT WORK

The present work starts in a general way introducing the reader to some important concepts in the field of batteries, such as, the internal operation of batteries or the different approaches concerning battery modelling. In the next chapter firstly we chose the EEC model as the battery model of reference for our study, and then we made a co-simulation between MapleSim and Matlab. In the third chapter, a wide range of different simulations were carried out in Matlab and Simulink in order to study the convergence of the SKF and EKF, showing that the latter provided better estimations than the former for all kinds of simulations. In chapter four, we extended the number of state variables of EKF to make a simultaneous simulation of battery SOH and SOC in Matlab and Simulink. In the last chapter, a comparison between the Lithium-ion reference battery model and the Lead-Acid EEC model with the same resistance and capacity was made, showing that the Lithium-ion one was able to provide more power. Finally, also in this chapter, the EKF algorithm was employed to estimate the state of the Lead-Acid battery.

Above all, the biggest contribution of this study has been the fast convergence achieved when using the EKF so as to estimate the state of the battery. This algorithm has been verified through simulation, and from the results of this validation, we can affirm that this algorithm is ready to be implemented in a BMS in order to monitor the state of the battery pack due to its simplicity, low computational expense and robustness to measurement errors and random disturbances. Moreover, not only is it highly accurate for Lithium-ion batteries, but also for batteries of other nature, such as Lead-Acid ones.

6.2 FURTHER WORK

Due to the lack of time, there are three main aspects which I would have liked to study more deeply:

– Develop of a simultaneous SOC and SOH estimation method which converge to the true value of SOH for all kinds of inputs, without having to modify the initialisation of each variable when changing the input. Maybe an algorithm that estimates in real time the SOC and only when this estimation starts to diverge, introduce and extended version of this algorithm in order to update the parameters and SOH of the battery would be more accurate since these parameters change more slowly than the SOC.

- Pass from SOC and SOH estimation problem to the implementation of a complete BMS, which would allow me to have a more global perspective of what battery management and optimisation involves.
- Achieve the co-simulation between Matlab and Simulink for state estimation, as in this study we have struggled with the problem of working in Simulink with a time continuous block in a sample time-based model, due to the fact that Kalman Filter can only work in discrete time systems.

ANNEX- GUIDE TO USE MAPLESIM

First of all, it is necessary to create a new MapleSim model following these septs: File→New→Model. However, if you prefer to open a model that it was already created in a previous session, these are the steps: File→Open…→Choose the file where it was saved previously. Both possibilities are shown inside the red rectangle in Figure A.

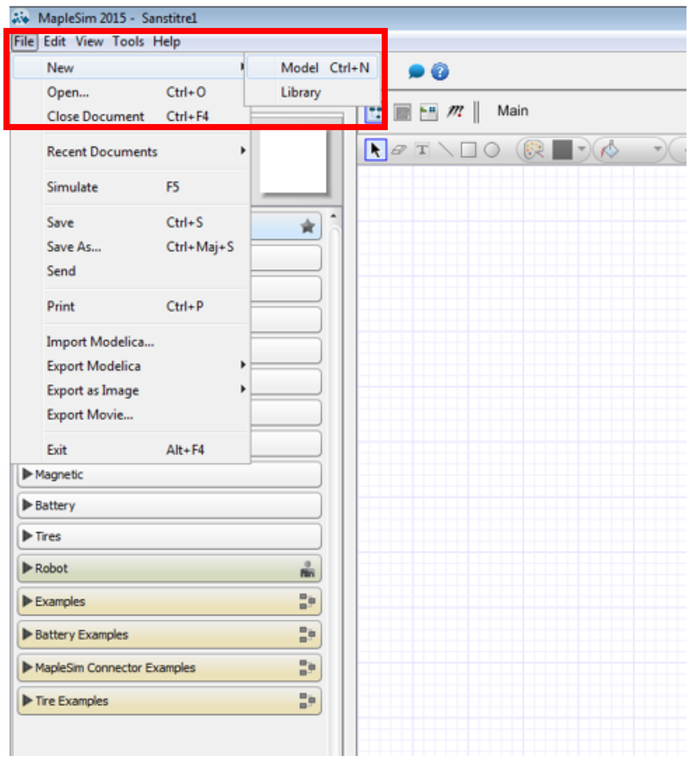
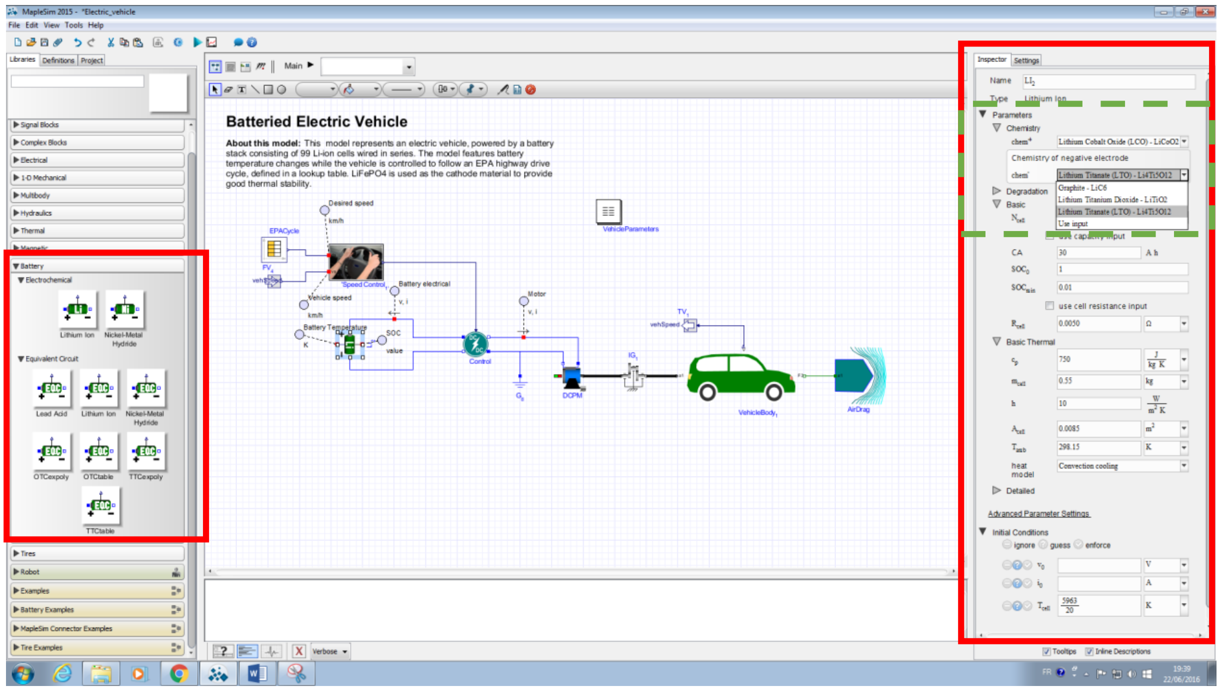


Figure A

Among the Electrochemical models it is possible to find the Lithium-ion Battery and Nickel-Metal Hydride, whereas for the EEC models we have the Lithium-ion battery, Lead-Acid, Nickel-Metal Hydride and we also have some other user-defined battery components that allow you to customise them using experimental data. All of these battery models are shown inside the red rectangle of the left-hand side in Figure B. We can use all the components listed on MapleSim libraries simply dragging them into the model.

The behaviors of the battery model are described by a small set of parameter that can be seen in Figure B inside the red rectangle of the right-hand side. For the Lithium-ion battery we can also select different kinds of cathode and anode material, for example we have fourteen different types of cathode materials and three types of anode materials (green rectangle in Figure B) for Lithium-ion chemistry.



Batteried Electric Vehicle

About this model: This model represents an electric vehicle, powered by a battery stack consisting of 99 Li-ion cells wired in series. The model features battery temperature changes while the vehicle is controlled to follow an EPA highway drive cycle, defined in a lookup table. LiFePO4 is used as the cathode material to provide good thermal stability.

Name	LI_2	
Type	Lithium Ion	
$chem^+$	Lithium Cobalt Oxide (LCO) - LiCoO2	
$chem^-$	Lithium Titanate (LTO) - Li4Ti5O12	
CA	30	A h
SOC_0	1	
SOC_{min}	0.01	
R_{cell}	0.0050	Ω
c_p	750	$\frac{J}{kg\ K}$
m_{cell}	0.55	kg
h	10	$\frac{W}{m^2\ K}$
A_{cell}	0.0085	m^2
T_{amb}	298.15	K
heat model	Convection cooling	
v_0		V
i_0		A
T_{cell}	$\frac{5963}{20}$	K

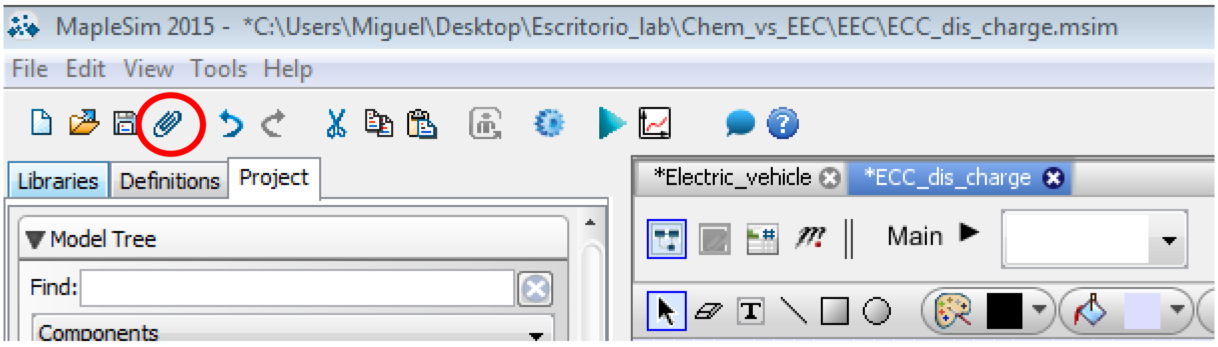
Figure B

HOW TO CREATE CUSTOM COMPONENTS

In Figure 2.1 (from section 2) we used a custom component which summed the current provided by the two sources. These sources were configured in order to work separately in different periods of time. Firstly, it worked the one that discharged the battery and then the other one which charged it.

The general process of creating a custom component for a MapleSim model consists of specifying the component equations for the custom component, component parameters and system model, specifying the port types and their values. The creation of a custom component follows these steps:

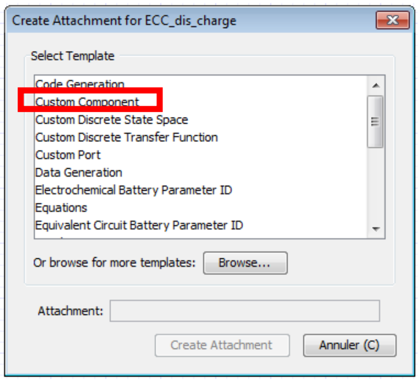
1. Click on the option "*create attachment from template*" (red circle in Figure C).



MapleSim 2015 - *C:\Users\Miguel\Desktop\Escritorio_lab\Chem_vs_EEC\EEC\ECC_dis_charge.msim

Figure C

2. Select *Custom Component*, write the name that you want to give to the attachment and then click "Create Attachment". The Maple Custom Component template is loaded.



Create Attachment for ECC_dis_charge

Select Template

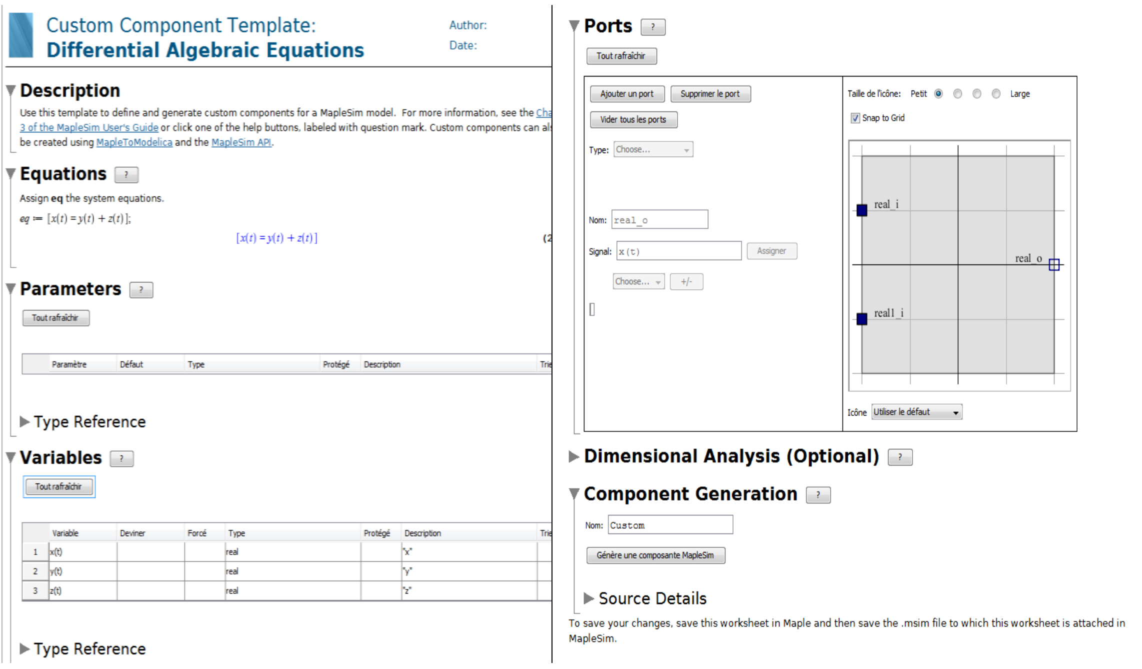
- Code Generation
- Custom Component
- Custom Discrete State Space
- Custom Discrete Transfer Function
- Custom Port
- Data Generation
- Electrochemical Battery Parameter ID
- Equations
- Equivalent Circuit Battery Parameter ID

Or browse for more templates: Browse...

Attachment:

Figure D

3. In the *Equations* section, you have to write down the equations that describe the operation that you want to give to the block. Equations, parameters and initial conditions are all entered here. In the model of Figure 2.1, as we had two inputs, y(t) and z(t) (the two sources), and we wanted to obtain an output equal to the sum of both of them, in the "Equations" section we wrote $x(t) = y(t) + z(t)$ (Figure E).



Custom Component Template: **Differential Algebraic Equations**

Author:
Date:

Description

Use this template to define and generate custom components for a MapleSim model. For more information, see the Cha 3 of the MapleSim User's Guide or click one of the help buttons, labeled with question mark. Custom components can al be created using MapleToModelica and the MapleSim API.

Equations

Assign **eq** the system equations.

$eq := [x(t) = y(t) + z(t)];$

$[x(t) = y(t) + z(t)]$ (2

Parameters

	Paramètre	Défaut	Type	Protégé	Description	Tri

▶ Type Reference

Variables

	Variable	Deviner	Forcé	Type	Protégé	Description	Tri
1	x(t)			real		"x"	
2	y(t)			real		"y"	
3	z(t)			real		"z"	

▶ Type Reference

Ports

Type: Choose...

Nom: real_o

Signal: x(t)

Taille de l'icône: Petit [✓ ●] ○ ○ ○ Large

[✓] Snap to Grid

Icône: Utiliser le défaut

▶ **Dimensional Analysis (Optional)**

Component Generation

Nom: Custom

▶ Source Details

To save your changes, save this worksheet in Maple and then save the .msim file to which this worksheet is attached in MapleSim.

Figure E

4. In the *Parameters* section, assign default values and types to model parameters. In our case, it was not necessary to define anything here because the equation did not have parameters.

5. In the *Variables* section, assign initial values and types for model parameters. The variables for our custom component were automatically defined here.
6. In the *Ports* section, add ports to the custom components by clicking Add Port ("Ajouter un port" in Figure E).
7. Provide the details for the port type, style, name and port signals.
8. In the *Component Generation* section, enter a name for the component. This will be the name shown in the *Definitions* tab in MapleSim for the custom component (Figure F).
9. Click *Generate MapleSim Component* to create your component and to bring you back into the MapleSim environment. The custom component now is going to appear in the *Definitions* tab under *Components,* as shown in Figure F.

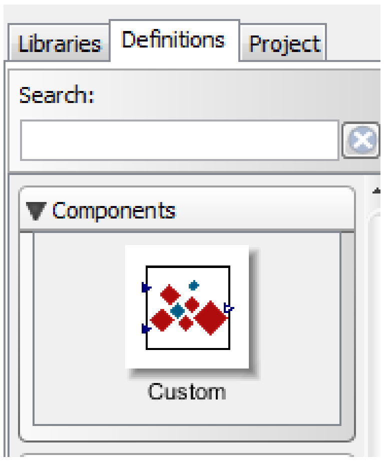


Figure F

CREATE A SUBSYSTEM

In order to create a subsystem it is necessary to select all the components that we want to group together. Then click the right bottom of the mouse and choose "*Create Subsystem*".

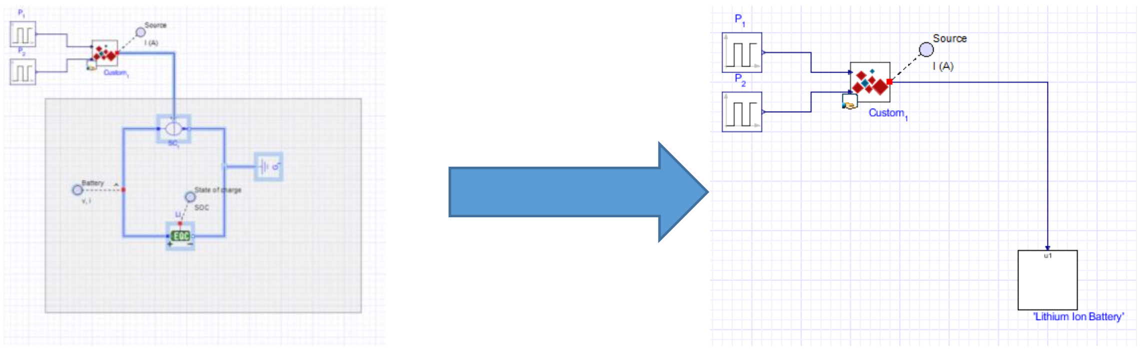


Figure G

Then if we want to continue measuring the battery states, such as SOC or terminal voltage, it is necessay to configure it as an output of the subsystem as shown in Figure 2.2.

USING THE SIMULINK COMPONENT BLOCK GENERATION TEMPLATE

In sections 3.4 and 4.3 the Simulink Component Block Generation was used in order to export the reference battery model from MapleSim to Simulink environment. In this section we are going to explain this function of MapleSim which allows to connect two of the most important software concerning battery modelling and simulation.

The MapleSim Connector provides *Simulink Component Block Generation* template in the form of a Maple worksheet for manipulating and exporting MapleSim subsystems. This template contains pre-built embedded components that allow you to generate S-function or C code from MapleSim subsystem, export the subsystem as a Simulink block, and save the source code.

The Simulink Component Block Generation consists of the following steps:

1. <u>Subsystem preparation:</u>
   The creation of the subsystem was explained in the previous section. This help MapleSim to identify the set of modelling components that you want to export as a block component.

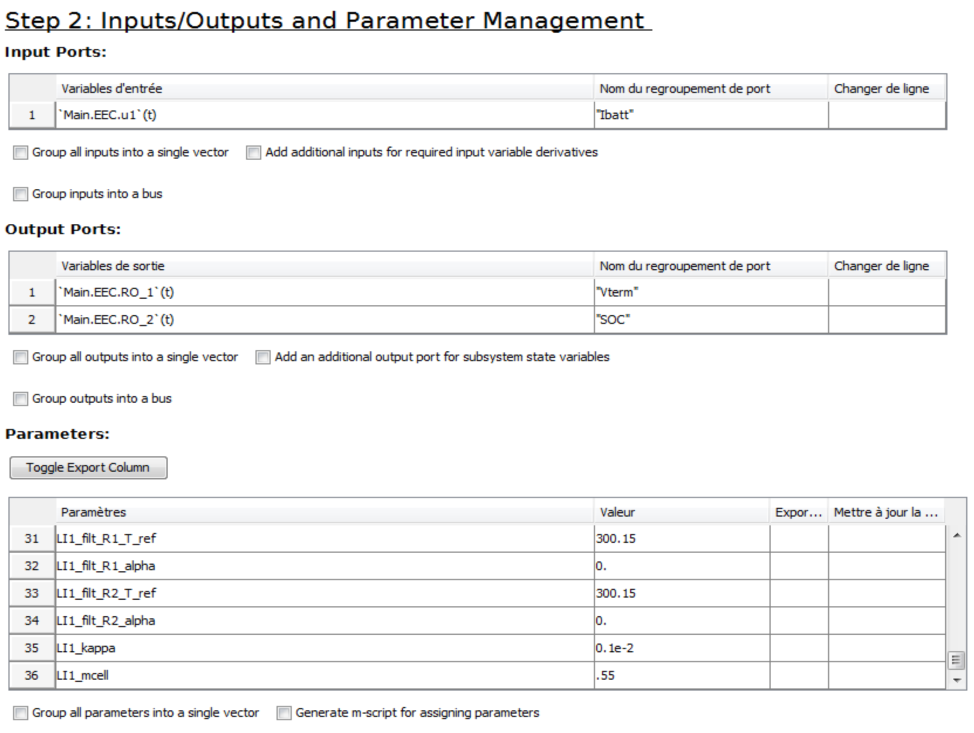
2. <u>Subsystem selection:</u>
   You can select which subsystems from your model you want to export to a Simulink block. Once a subsystem is selected, click *Load Selected Subsystem* and all the defined input and outputs ports will be loaded automatically.

3. <u>Port and Management:</u>

   MapleSim allows you to customise, define and assign parameter values to specifics ports. Subsystem components to which you assign the parameter, inherit a parameter value defined at the subsystem level. Once the subsystem is loaded you can group individual input and output variable elements into a vector array, and add additional input and output ports for customised parameter values.

   The following selections specify the input ports:

   a. If you select *Group all inputs into a single vector*, MapleSim is going to create a single vector input port for all of the input signals instead of individual ports.
   b. If you select *Add additional inputs for required input variable derivatives*, MapleSim would use calculated derivative values instead of numerical approximations



Step 2: Inputs/Outputs and Parameter Management

Input Ports:

	Variables d'entrée	Nom du regroupement de port	Changer de ligne
1	\`Main.EEC.u1\`(t)	"Ibatt"	

☐ Group all inputs into a single vector ☐ Add additional inputs for required input variable derivatives

☐ Group inputs into a bus

Output Ports:

	Variables de sortie	Nom du regroupement de port	Changer de ligne
1	\`Main.EEC.RO_1\`(t)	"Vterm"	
2	\`Main.EEC.RO_2\`(t)	"SOC"	

☐ Group all outputs into a single vector ☐ Add an additional output port for subsystem state variables

☐ Group outputs into a bus

Parameters:

Toggle Export Column

	Paramètres	Valeur	Expor...	Mettre à jour la ...
31	LI1_filt_R1_T_ref	300.15		
32	LI1_filt_R1_alpha	0.		
33	LI1_filt_R2_T_ref	300.15		
34	LI1_filt_R2_alpha	0.		
35	LI1_kappa	0.1e-2		
36	LI1_mcell	.55		

☐ Group all parameters into a single vector ☐ Generate m-script for assigning parameters

Figure H

Regarding the output ports you have the following options:

a. Select *Group all outputs into a single vector* to define outputs as an S-Function mask if you want to create custom dialog boxes and icons for your S-Function blocks. Masked dialog boxes can make it easier to specify additional parameters for S-functions.
b. Select *Add an additional output port for subsystem state variables* to add extra output ports for the state variables.

Finally, with regard to parameters options:

a. Select *Group all parameters into a single vector* to create a single parameter vector for all of the parameters in the S-function. If this option is not selected, the S-function mask will contain one parameter input box for each of the S-function parameters
b. Select *Generate m-script for assigning parameters* to generate an initialisation m-file with the parameters

Press *Toggle Export Column* to toggle selected/unselected parameters for export.

4. S-Function Options:

These settings specify the advanced options for the code generation process:

a. **Optimisation options:**

   This option specifies the degree of simplification applied to the model equation during the code generation process and eliminates redundant variables and equations in the system. It is possible to select one of the following options:

   i. *None (0):* no optimisation is performed; the default equations will be used in the generation code.

   ii. *Partial (1, 2):* removes redundant equations from the system.

   iii. *Full (3):* performs index reduction to reduce the system to an ordinary differential equation (ODE) system or a differential algebraic equation (DAE) system of index 1, and removes redundant equations.

b. **Constraint Handling Options:**

   This option is used to improve the accuracy of DAE system that has constraints. If the constraint is not satisfied, the system result may deviate from the actual solution and could lead to an increase in error at an exponential rate. These are the parameter that you can adapt to meet your specific needs:

   i. *Maximum number of projection iterations*: here it is possible to specify the maximum number of times that a projection is permitted to iterate to obtain a more accurate solution.

   ii. *Error tolerance:* to specify the desirable error tolerance to achieve after the projection.

   iii. *Apply projection during event iterations*: this option must be selected when you want to interpolate iterations to obtain a more accurate solution.

c. **Event Handling Options:**

   Use this option to improve the accuracy of DAE system with events. If the constraint is not satisfied, the system result may deviate from the actual solution and could lead to an increase in error at an exponential rate. In this section you have the following options to adapt the generated block to your requirements:

i. Set the *Maximum number of event iterations* to specify the maximum number of times that a projection is permitted to iterate to obtain a more accurate solution.

ii. Set the *Width of event hysteresis band* to specify the desirable error tolerance to achieve after the projection.

iii. Select *Optimize for use with fixed-step integrators* to optimise the event iterations as a function of hysterias bandwidth

5. <u>Generate S-Function</u>

   In this section you can provide a name and specify the location for the generated file. Moreover you can choose if you prefer to generate an S-Function without Simulink connection (*Generate S-Function (no Compile)*) or to generate an S-Function block (click *Generate and Compile S-Function*).

6. <u>View S-Function</u>

   A Matlab command window opens and the block with any of the following specified parameters is generated in Simulink:

   - Block Generation Script
   - C Code
   - Parameter Script

REFERENCES

[1] R. MKAHL, Contribution à la modélisation, au dimensionnement et à la gestion des flux énérgetiques d'un système de recharge de véhicules électriques: étude de l'interconnexion avec le reseau électrique, 2015.

[2] S. Dearborn, “Power Management in Portable Applications: Charging Lithium-Ion/Lithium-Polymer Batteries,” Microchip Technology Inc..

[3] L. L. e. al., “A review on the key issues for lithium-ion battery management in electric vehicles,” *Journal of Power Sources 226 ,* pp. 272-288, 2013.

[4] X. H. e. al., “A comparative study of equivalent circuit models for Li-ion batteries,” *Power of Sources,* no. 198, pp. 359-367, 2012.

[5] J. B. e. al., “Modelling of Lithium-ion Battery and SOC Estimation using Simple and Extended Discrete Kalman Filters for Aircraft Energy Management,” in *IECON2015*, Yokohama, 2015.

[6] H. H. e. al., “Evaluation of Lithium-Ion Battery Equivalent Circuit Models for State of Charge Estimation by an Experimental Approach,” *Energies,* pp. 582-598, 2011.

[7] T. W. J. Cao, “Multi-Domain Modeling Simulation and Applications Based on MapleSim,” 2013.

[8] MapleSoftTM, “MapleSim: Technological Superiority in Multi-Domain Physical Modeling and Simulation”.

[9] “Zhou W, et al. Battery behaviour prediction and battery working states analysis of a hybrid solar-wind power generation system. Renew Energy (2007):”.

[10] A. R. Rami Yamin, “Embedded State of Charge and State of Health Estimator based on Kalman Filter for Electric Scooter BMS”.

[11] T. H. e. al., “Simplified Extended Kalman Filter Observer for SOC Estimation of Commercial Power-Oriented LFP Lithium Battery Cells”.

[12] “C.Y. Xia, S. Zhang, H.T. Sun, Chin. J. Power Sources 31 (5) (2007) 414-417(”.

[13] S. S. e. al., “Improved extended Kalman filter or state od charge estimation of the battery pack,” *J. Power Sources,* pp. 368-376, 2014.

[14] B. B. e. al., “Nonlinear Observers for Predicting State-of-Charge and State-of-Health of Lead-Acid Batteries for Hybrid-Electric Vehicles,” *IEEE Transactions on vehicular technology,* vol. 54, no. 3, pp. 783-794, 2005.

[15] G. W. a. G. Bishop, “An Introduction to the Kalman Filter,” 2006.

[16] A. R. e. al., “SOC Estimation for Li-ion Batteries Based on Equivalent Circuit Diagrams and the Application of a Kalman Filter”.

[17] Maplesoft, Getting Started with MapleSim Connector, Canada, 2011.

[18] Y. Z. e. al., “Combined State of Charge and State of Health estimation over lithium-ion battery cell cycle lifespan for electric vehicles,” *Journal of Power Sources,* no. 273, pp. 793-803, 2015.

[19] MapleSoft TM, MapleSim Help.