MASTER THESIS

TITLE: Determinació de la càrrega de bateries i condensadors mitjançant mesures d’impedància

MASTER DEGREE: Master in Science in Telecommunication Engineering & Management

AUTHOR: Kevin Sangüesa Cremades

DIRECTOR: Ángel Cuadras Tomás

DATE: May 15th 2015
Títol: Determinació de la càrrega de bateries i condensadors mitjançant mesures d'impedància

Autor: Kevin Sangüesa Cremades

Director: Ángel Cuadras Tomás

Data: May 15 th 2015

Resum

L’objectiu d’aquest PFC es desenvolupar un sistema per determinar la impedància de bateries i condensadors a partir de les mesures de tensió i corrent en l’espai temporal. La impedància és d’interès per poder-la relacionar amb l’ estat de càrrega i enveliment de les bateries.

Es proposa transformar els senyals de tensió i corrent a l’espai freqüencial i determinar així la impedància per a diferents circuits prova.

Farem servir diferents senyals d’excitació amb un denominador comú que és utilitzar senyals harmònics de tal manera que puguem escombrar tot l’espectre de la impedància. Analitzarem la transformació a l’espai fasorial, amb especial èmfasi en els errors deguts a l’enfinestrat, aliasing, freqüència de mostreig i els introduïts deguts a l’escenari experimental.

Els resultats experimentals es validaran amb els corresponents resultats de la impedància obtinguts amb un programa de simulació de circuits juntament amb la simulació teòrica dels circuits proba per tal de corroborar l’eficàcia i limitacions d’aquest mètode.

Amb aquest projecte podem simplificar l’estudi d’impedàncies basats en analitzadors d’impedàncies a sistemes generadors de senyals harmònics (quadrats, triangulars, trapezoïdals...).
Overview

The objective of this PFC is to develop a system to determine the impedance of capacitors and batteries from voltage and current measurements in the time domain. The impedance is of interest since this is related to the state of charge and aging of the batteries.

It is proposed to transform the voltage and current signals to the frequency domain and thus determine the impedance for different test circuits.

We will use different excitation signals with a common characteristic that is using harmonic signals so that we can compute the whole impedance spectrum. We will analyse the transformation to the phasor domain, with special emphasis on errors due to windowing, aliasing, sampling frequency and the ones introduced due to the experimental stage.

The experimental results are validated with the corresponding impedance results obtained with a circuit simulation program together with the theoretical simulation of the test circuits to prove the effectiveness and limitations of this method.

With this project we can simplify the study of impedances based on impedance analysers to harmonic signal generators systems (square, triangular, trapezoidal ...).
Este Proyecto Final de Carrera
está dedicado a Don Rafael Rodríguez
admirable profesor que sabe reconocer
y valorar el trabajo y el esfuerzo de sus alumnos
# TABLE OF CONTENTS

INTRODUCTION .................................................................................................................. 1

CHAPTER 1. BACKGROUND ............................................................................................... 2

1.1. Introduction to the frequency Transforms ................................................................. 2
   1.1.1. The Fourier Transform ......................................................................................... 2
   1.1.2. The Laplace Transform ....................................................................................... 2
   1.1.3. The Fourier Transform vs. the Laplace Transform ............................................... 3
   1.1.4. The DFT and DTFT ............................................................................................ 5
   1.1.5. The Fast Fourier Transform (FFT) ...................................................................... 7

1.2. Capacitors ...................................................................................................................... 9
   1.2.1. Approaches of equivalent circuit for capacitors .................................................. 9

CHAPTER 2. THEORETICAL APPROACH ............................................................................ 11

2.1. Harmonic Signal .......................................................................................................... 11

2.2. Fourier Series ............................................................................................................... 11
   2.2.1. Existence of the Fourier Series: Dirichlet's Condition ....................................... 12
   2.2.2. Computing the Fourier Series of f(t) ................................................................. 12
   2.2.3. The Gibbs phenomenon .................................................................................... 16

2.3. Numerical Computation errors of the Fourier Transform ......................................... 17
   2.3.1. Aliasing and Leakage in Numerical Computations ............................................ 18
   2.3.1.1. Leakage .......................................................................................................... 26
   2.3.1.2. The 'art' of Windowing .................................................................................. 27
   2.3.1.3. Other uses of Windowing ............................................................................. 31

CHAPTER 3. MATERIALS & METHODS ............................................................................. 32

3.1. Circuit Characterization .............................................................................................. 32
   3.1.1. Circuit 1 ............................................................................................................. 32
   3.1.2. Circuit 2 ............................................................................................................ 33
   3.1.3. Circuit 3 ............................................................................................................ 35

3.2. Matlab Process ............................................................................................................ 37
   3.2.1. Computed Impedance with Matlab Data .......................................................... 37
   3.2.2. Computed Impedance with TopSpice Data ......................................................... 40
   3.2.3. Error Calculation ............................................................................................. 44
   3.2.3.1. Error Calculation in case of Leakage ............................................................. 46

3.3. Laboratory set up ........................................................................................................ 48

CHAPTER 4. RESULTS ......................................................................................................... 53

4.1. Input Signals ................................................................................................................ 53

4.2. Experimental Results ................................................................................................ 54
   4.2.1. Results for Circuit 1 .......................................................................................... 56
   4.2.2. Results for Circuit 2 .......................................................................................... 59
   4.2.3. Results for Circuit 3 .......................................................................................... 62
INTRODUCTION

One of the current problems for batteries and capacitors is their degradation during their lifetime. Portable electronic devices operate with electrical energy stored in batteries, as in mobile phones, music players, and all kind of movable electronic equipment’s. The lifetime of batteries is reduced due to its usage, thus translating in longer time to charge them and shorter to replace.

The aim of this PFC is to design a method able to compute the impedance of a battery through the excitation signal harmonics. To achieve this, we use an equivalent capacitor/battery electrical circuit (three different circuits are used) to calculate how the impedance of a healthy battery changes during its lifetime. In addition we will consider the limitations of the method due to the data processing together with the limitations in the experimental cases.

In this PFC, a Matlab program has been done to compute the Fast Fourier Transform (digital Fourier Transform) of the time-domain signals of voltage and current to obtain the impedance frequency response of three different circuits. We have also used TopSpice software for circuit simulation, Maple for writing the mathematical equations and functions.

Errors obtained from the difference between theoretical, TopSpice and experimental results have been evaluated. Also, guides to improve impedance accuracy and resolution are given, as a function of input frequency and amplitude module as well as the sampling frequency.

This PFC is split up in four main chapters plus one annexes:

Chapter 1: Includes the description of the basic concepts like the different techniques to analyse the spectra domain and the characteristics of capacitors.

Chapter 2: Study of the limitations and numerical computation errors due to the digital process carried out in Matlab using as example the pulse signal. There will be explained the link of the pulse signal spectra with its Fourier Series.

Chapter 3: In this chapter it is exposed the layout and characteristics of the three circuits that will be used to compute their impedances in chapter 4. It is explained the whole digital process developed in Matlab that computes the impedance and calculate the errors introduced by this process for two different cases. At last it is explained the laboratory set up used for measuring the experimental signals of V and I used to compute the impedance in chapter 4.

Chapter 4: The representation of all the impedance figures and the errors obtained for the different excitation signals and circuits in order to compare and explain the results obtained between TopSpice and experimental cases.

Annexes: All the Matlab scripts that have been used to study, compute and represent the figures and impedances of this PFC as well as the explanations and images of how work the TopSpice software.
CHAPTER 1. BACKGROUND

In this chapter, the background to perform signal transformation from time-domain to frequency-domain is presented. Also, a brief introduction to capacitors is also introduced.

1.1. Introduction to the frequency Transforms

We present here different available transforms that go from the mathematical transform expression of a continuous time signal into its continuous frequency domain up to the computed transformation of a discrete time signal into its discrete frequency domain in a software tool.

1.1.1. The Fourier Transform

The Fourier transform is a tool that allows to (see Eq. 1.1) decompose a function of time $f(t)$ into the frequencies that make it up as a continuous sum of exponentials of the form $e^{-j\omega t}$, whose frequencies are restricted to the imaginary axis in the complex plane ($s = j\omega$). The Fourier transform of a time function itself is a complex-valued function of frequency, whose absolute value represents the amount of that frequency present in the original function, and whose complex argument is the phase offset of the basic sinusoid in that frequency. The Fourier transform is called the frequency domain representation of the original signal. The term Fourier transform refers to both the frequency domain representation and the mathematical operation that associates the frequency domain representation to a time function. Such representation is quite valuable in the analysis and signal processing. (See reference [1] pages 361-362).

The Fourier transform is not limited to functions of time, but in order to have a unified language, the domain of the original function is commonly referred to as the time domain. For many functions of practical interest one can define an operation that reverses this: the inverse Fourier transformation, also called Fourier synthesis, of a frequency domain representation combines the contributions of all the different frequencies to recover the original function of time.

\[
F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t} \, dt
\]  

(1.1)

1.1.2. The Laplace Transform

The Laplace transform transforms a function of time into a function of complex frequency where $s = \sigma + j\omega$, in which the sigma ‘$\sigma$’ parameter represents the attenuation and the $j\omega$ the imaginary frequency. The LT (Laplace transform) is
related to the FT (Fourier transform), but whereas the FT expresses a function or signal as a superposition of sinusoids, the Laplace transform (described by Eq. 1.2) expresses a function as a superposition of moments.

\[ F(s) = \int_0^\infty f(t) e^{-st} dt \]  

(1.2)

The LT provides an alternative functionality that simplifies the process of analysing the behaviour of the system or synthesizing a new system based on a set of specifications. So, for example, Laplace transformation from the time domain to the frequency domains transforms differential equations into algebraic equations and multiplication into convolution (also applicable to the FT). See reference [1] pages 361-362.

The LT unlike the FT, there exists even for input growing exponentials. Second, the Fourier transform cannot be used easily to analyse unstable or even marginally stable systems. The basic reason for both these difficulties is that for some signals, such as \( e^{at}u(t) \) \((a > 0)\), ordinary sinusoids or exponentials of the form \( e^{jwt} \) are incapable of synthesizing exponentially growing signals.

### 1.1.3. The Fourier Transform vs. the Laplace Transform

The Fourier transform is a part of the Laplace transform, while the FT is only evaluated when \( e^{-jwt} \) the LT is evaluated for \( e^{-(\sigma+jwt)} \) so if \( \sigma = 0 \) we get the particular case of the LT, the FT. The sigma gives information of the transient part this is why the LT is especially used in control engineering, where transient part is important. On the other hand for spectral representation of a signal we will use the Fourier Transform since for our purpose of representing the frequency domain of a signal it is enough and much easier to work with a transform of the kind of \( e^{-jwt} \) in addition there is a function implemented in Matlab called Fast Fourier Transform that already does this computation. See reference [1] pages 361, 362 and 370 for more information about the connections of FT with the LT.

Table 1.1 describes the advantages and disadvantages of both Transforms and the cases when it is better to use the FT or LT. In this work, we focus on FT instead of LT because we achieve the same results with less complexity since what we want to see is the frequency spectrum of stable time-domain signals where it is not needed the information of the transient part ‘\( \sigma \)’.

<table>
<thead>
<tr>
<th>Table 1.1. Comparisons of FT and LT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Fourier Transform</strong></td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>Frequency spaces transform.</td>
</tr>
<tr>
<td>----------------------------</td>
</tr>
<tr>
<td>Useful if you want to see the frequency spectrum of a signal.</td>
</tr>
<tr>
<td>Only work for functions that converge to zero at inf and stable signals (non-exponential). Translate ( f(t) ) to ( w ) domain (( s = jw )).</td>
</tr>
<tr>
<td>Does not have all the info in this case only frequency and amplitude.</td>
</tr>
<tr>
<td>Used for frequency-domain analysis and for signals with constant mean signals.</td>
</tr>
<tr>
<td>Easiest for frequency domain. There exist the FFT function and not a Fast Laplace transform.</td>
</tr>
<tr>
<td>Only works on signals where Dirichlet's condition is satisfied.</td>
</tr>
<tr>
<td>Used on signal analysis, processing, etc.</td>
</tr>
<tr>
<td>If you care about the future also (-inf to inf).</td>
</tr>
<tr>
<td>Steady state behaviour of a dynamical</td>
</tr>
</tbody>
</table>
1.1.4. The DFT and DTFT

The Discrete Fourier Transform is a particular case of the Fourier transform for sequences of finite length where the respective spectra are evaluated in a specific frequency, thus obtaining a discrete or ‘sampled’ spectra. It is important not to confuse the DFT with the DTFT (Discrete Time Fourier Transform). With the DTFT the Fourier transform is calculated in the interval $-\infty < n < \infty$ so for sequences than can be of infinite length, while for the DFT its calculated over the temporal interval $0 < n < N - 1$, being $N$ the length of the sequence of finite duration, be or not periodical. See references [1] pages 348-350 and [2] pages 337, 382-383 for further understanding about the DFT periodicity.

The DTFT (see Eqs. 1.3) is the “exact” transform of a sequence, defined in the continuous values inside the interval $\Omega = -\pi$ to $\Omega = \pi$ obtained through analytical techniques, while the DFT is a “mutilation” of the DTFT.

$$X(e^{j\Omega}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j2\pi n}$$

or

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n} \quad (1.3)$$

The DFT (see Eq. 1.4) is extremely important in the field of frequency (spectrum) analysis because it takes a discrete signal in the time domain and transforms that signal into its discrete frequency domain representation repeated periodically. Without a discrete-time to discrete-frequency transform we would not be able to compute the Fourier transform with a microprocessor or DSP based system. The numerical calculation of the DFT implemented via software is the efficient algorithm called Fast Fourier Transform (FFT). It is the speed and discrete nature of the FFT that allows us to analyse a signal’s spectrum with Matlab or in real-time on a Spectrum Analyser.

$$F(\omega) = \sum_{n=0}^{N-1} f(nT) e^{-j\omega nT} \quad (1.4)$$

The DFT is not the same that the DTFT. Both start with a discrete-time signal, but the DFT produces a discrete frequency domain representation while the DTFT is continuous in the frequency domain. These two transforms have much in common, however. It is therefore helpful to have a basic understanding of the properties of the DTFT. See references [2] pages 340-341 and [15] for further information about the DFT and its link with the DTFT.
Periodicity: The DTFT, $X(e^{j\Omega})$, is periodic. One period extends from $\Omega = -\pi$ to $\Omega = \pi$. Taking advantage of this redundancy, the DFT is only defined in the region between 0 and $f_s$.

Symmetry: When the region between 0 and $f_s$ is examined, it can be seen that there is even symmetry around the centre point, $0.5 f_s$, the Nyquist frequency. This symmetry adds redundant information. Fig. 1.8 shows the DFT (implemented with Matlab's FFT function) of a cosine with a frequency one-tenth the sampling frequency (the sampling frequency must be at least twice the highest frequency component). Note that the data between $0.5 f_s$ and $f_s$ is a mirror image of the data between 0 and $0.5 f_s$. See reference [15] page 2.
1.1.5. The Fast Fourier Transform (FFT)

FFT or Fast Fourier Transform is an effective tool for computing the DFT of a signal. The FFT is a faster version of the Discrete Fourier Transform (DFT). The FFT utilizes some clever algorithms to do the same thing as the DTF, but in much less time.

The number of computations required in performing the DFT was dramatically reduced by an algorithm developed by Tukey and Cooley in 1965. This algorithm known as the Fast Fourier Transform (FFT), reduces the number of computations from something of the order of $N^2$ to $N \log N$. To compute one sample of the DFT we require $N$ complex multiplications and $N - 1$ complex additions. To compute $N$ such values (DFT for $n = 0, 1, ..., N - 1$), we require a total of $N^2$ complex multiplications and $N(N - 1)$ complex additions. For a large number of $N$, these computations can be prohibitively time-consuming, even for high-speed computer. See references [1] page 352-356 to view the improvement of the FFT algorithm computations.

Although there are many variations of the original Tukey-Cooley algorithm, there can be grouped into two basic types: decimation-in-time and decimation-in-frequency. The algorithm is simplified if we choose $N$ to be a power of 2, although such a choice is not essential and not needed.

The decimation in time algorithm: if the input signal has $N = 8$ samples/points the $N$ point data sequence is divided into two $N/2$ point sequences consisting of even and odd numbered samples respectively, the Fig. 1.3 shows the resulting samples of the FFT using the decimation in time algorithm. See references [2] pages 367-376 and [15] pages 3-6 for a deeper understanding about the FFT mechanics.
The decimation in frequency algorithm is similar to the decimation in time algorithm. The only difference is that instead of dividing the $N$ samples into two sequences of even and odd numbered samples, we divide $N$ into two sequences formed by the first $N/2$ and the last $N/2$ digits, proceeding in the same way until a single-point of the DFT is reached in the $\log_2 N$ steps. The total number of computations in this algorithm is the same as that in the decimation-in-time algorithm. The Eq. 1.5 shows the algorithm in Matlab used to compute the FFT.
As we will see in the next section in order to compute the FFT correctly and do not have leakage, the time waveform must be captured for exactly an integer number of cycles then the assumption of periodicity in time matches the intended waveform of infinite extent.

1.2. Capacitors

A conventional capacitor is formed by a couple of conductive surfaces separated by a dielectric material that stands a potential difference and acquires a determinate electrical charge, positive in one plate and negative in the other one. The stored energy in one of the plates is proportional to the potential difference between them.

Depending on the material between plates higher voltages may be achieved. This implies higher energy densities. Their capacity is measured in farads.

![CAPACITOR](image)

**Fig. 1.4** Performance of a common capacitor

The common capacitor (see Fig. 1.4) store static electricity by building up opposite charges on two metal plates (blue and red) separated by an insulating material called a dielectric (orange).

1.2.1. Approaches of equivalent circuit for capacitors

In a real capacitor, the dielectric losses, modelled by a resistor $R_c$, must be taken into account. Its equivalent circuit or model is shown in Fig. 1.5 although depending on the type of capacitor it can be used more complex patterns than the ones in Fig. 1.5 for more accurate analysis.
According to Fig. 1.5 the equivalent series resistor (ESR) and the equivalent parallel resistor (EPR) have influence on capacitors charge and discharge behaviour. The ESR is due to the electrode, electrolyte and contact resistances and dissipates power in internal heating when being charged or discharged. Although the ESR achieves small values, in the order of few ohms, it impacts to the energy efficiency and to the power density of the capacitor.

On the other hand, the EPR (also called leakage resistor) achieves higher values than ESR, usually above kilohms, and determines the leakage current in the capacitor. The EPR is responsible for the capacitor self-discharge time. Its value must be as high as possible to limit the leakage current. Finally, the capacity \( C \) accounts for the energy that can be stored in a capacitor.

From this behaviour, the function of the simplified equivalent circuit can be inferred:

- The equivalent series resistor \( (R_s) \) offers a resistance to the current flow \( (I) \). Depending on the \( R_s \) value, the capacitor \( (C) \) will charge faster or slower. As high \( R_s \) is, the more time to charge the capacitor. This may prevent fast power transfer to loads.

- Once the capacitor is charged and there is no current injection, the self-discharging will start through the \( R_p \). So, higher values of \( R_p \) will make the capacitor \( (C) \) to have low self-discharge times.
CHAPTER 2. THEORETICAL APPROACH

In this chapter, we will describe the methodology to achieve impedance using harmonics. We will focus on one of the signals used to compute the different circuit impedances in the next chapters. We will also study the errors obtained from the numerical computation of this signal. Results and conclusions of this study can be extrapolated to the other signals used.

2.1. Harmonic Signal

In order to study the whole spectrum of circuit impedance using one signal we will need a signal that is composed of harmonics. The pulse signal is an example of a harmonic signal but there are also much more examples like the triangular and trapezoidal signals. These signals will also be used to the study the circuit impedance on the last chapter of this PFC together with the pulse. In section 2.2 we are going to study and explain the spectrum of the pulse signal thought its link with the Fourier Series.

![Fig. 2.1 Pulse Signal f(t)](image)

2.2. Fourier Series

Signals are vectors that can be represented as a sum of their components in a variety of ways. A large number of orthogonal signal sets that can be used as basis signals for generalized Fourier series exist. Most well-known signal sets are trigonometric (sinusoid) functions but there are other as exponential functions, Bessel functions, Chebyschev polynomials. In this section we consider trigonometric sets for representing and explaining Fig. 2.1.

A Trigonometric Fourier series indicates that a periodic signal \( f(t) \) can be expressed as a sum of sine and cosine waves plus a continuous value (DC). The coefficients of this series are \( a_0, a_n \) and \( b_n \).
Determinació de la càrrega de bateries i condensadors mitjançant mesures d’impedància

\[ f(t) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty} \left[ a_n \cos(n\omega_0 t) + b_n \sin(n\omega_0 t) \right] \] \hspace{1cm} (2.1)

A sinusoid of frequency \( n\omega_0 \) is called the \( n \)th harmonic of the sinusoid of frequency \( \omega_0 \) when \( n \) is an integer. In this set the sinusoid of frequency \( \omega_0 \) is called the fundamental, it serves as an anchor of which all the remaining terms are harmonics.

2.2.1. Existence of the Fourier Series: Dirichlet’s Condition

There are two basic conditions for the existence of the Fourier series for any signal.

1. For the series to exist, the coefficients \( a_0, a_n \) and \( b_n \) must be finite. From Eqs. (2.3), (2.4) and (2.5), it follows that the existence of these coefficients is guaranteed if \( f(t) \) is absolutely integrable over one period that is:

\[ \int_{-T}^{T} |f(t)| dt < \infty \] \hspace{1cm} (2.2)

This condition is known as the weak Dirichlet condition. If a function \( f(t) \) satisfies the weak Dirichlet conditions, the existence of a Fourier series is guaranteed, but the series may not converge at every point. For example, if a function \( f(t) \) is infinite at some point, then obviously the series representing the function will be no convergent at that point. Similarly, if a function has an infinite number of maxima and minima components of frequencies approaching to infinity also does not converge. Consequently, the coefficients in the series at higher frequencies do not decay rapidly, so that the series will not converge rapidly or uniformly.

2. For a convergent Fourier Series, in addition to condition (2.2), it is required that the function \( f(t) \) has only a finite number of maxima and minima in one period, and only a finite number of finite discontinuities in one period. These two conditions are known as the strong Dirichlet conditions. For example, any periodic waveform that can be generated in a laboratory satisfies strong Dirichlet conditions, and hence possesses a convergent Fourier series. Thus, a physical possibility of a periodic waveform is a valid and sufficient condition for the existence of a convergent series.

2.2.2. Computing the Fourier Series of \( f(t) \)

The amplitude spectrum indicates the amount (amplitudes) of various frequency components of \( f(t) \). If \( f(t) \) is a smooth function, its variations are less rapid. Synthesis of such function requires predominantly lower-frequency sinusoids
and relatively small amounts of rapidly varying (higher frequency) sinusoids. The amplitude spectrum of such a function would decay swiftly with frequency. To synthesize such a function we require fewer changes, such as jump discontinuities, contains rapid variations and its synthesis requires relatively large amount of high-frequency components. The amplitude spectrum of such a signal would decay slowly with frequency, and to synthesize such a function, we require many terms in its Fourier series for a good approximation, the pulse signal $f(t)$ (see Fig. 2.1) is a discontinuous function with finite jump discontinuities, and therefore its amplitude spectrum decays rather slowly as $1/n$ (see Eq. 2.5). On the other hand, the triangular pulse periodic signal is smoother because it is continuous function (no jump discontinuities). Its spectrum decays rapidly with frequency as $1/n^2$.

Fig. 2.1 shows the pulse signal $f(t)$ and its approximation by a truncated trigonometric Fourier series that includes only the first $n$ harmonics (also called coefficients) for $n = 3$ and 11 is represented in the Figures 2.2 and 2.3. The plot in time of the truncated Fourier series approximates closely to the function $f(t)$ as $n$ increases (Fig. 2.4), and we expect that the series will converge exactly to $f(t)$ as $n \to \infty$. (See Matlab scrip in annex A.1 for computing Fig. 2.2, 2.3 and 2.4)

![Fig. 2.2 Fourier series taking 3 coefficients](image1)

![Fig. 2.3 Fourier series taking 11 coefficients](image2)
Thus, the signal on Fig. 2.1 can be represented on time and frequency domains as a Fourier series. Depending on the number of coefficients taken on the Fourier series the time shape will resemble more to a real pulse signal in addition the spectrum amplitude will resemble more to the corresponding Fourier series hence we will see more information of the spectrum of the real pulse signal although the Fourier series (on time and frequency) will be never exactly the same as a pure pulse signal (on the time and frequency domain) by the simple fact that we can’t compute and infinite number of harmonics.

As seen in the previous figures the Fourier series resembles more to a real pulse signal when taking more coefficients.

The Fourier series coefficients for the signal of the Fig. 2.1 are calculated as follows:

\[
a_n = \frac{2}{T} \left[ \int_0^{T/2} f(t) \cos(n\omega_0 t) \, dt \right] = \frac{2}{T} \left[ \int_0^{T/2} \frac{1}{2} dt + \int_0^{T/2} \cos(2\pi T) \, dt \right] = \frac{2}{T} \left[ \frac{T}{2} \right] = \frac{T}{2} = \frac{2}{2} = 1
\] (2.3)

\[
a_n = \frac{2}{T} \left[ \int_0^{T/2} \sin(n\omega_0 t) \, dt \right] = \frac{2}{T} \left[ \frac{\sin(n\omega_0 T/2)}{n\omega_0} \right] = \frac{2}{T} \left[ \frac{\sin \left( \frac{n \cdot 2\pi T}{2} \right)}{n\omega_0} \right] = \frac{1}{n\pi} \left[ \sin(n\pi) \right] = 0
\] (2.4)

\[
b_n = \frac{2}{T} \left[ \int_0^{T/2} \cos(n\omega_0 t) \, dt \right] = \frac{2}{T} \left[ \frac{\cos(n\omega_0 T/2)}{n\omega_0} \right] = \frac{2}{T} \left[ \frac{\cos \left( \frac{n \cdot 2\pi T}{2} \right)}{n\omega_0} \right] + \frac{\cos \left( \frac{n \cdot 2\pi 0}{2} \right)}{n\omega_0} = \frac{1}{n\pi} \left[ \cos(n\pi) \right] = \frac{1}{n\pi} \left[ (-1)^n - 1 \right]
\] (2.5)

From \( T/2 \) to \( T \) the value of coefficients \( a_n \) and \( b_n \) are 0 because the \( f(t) \) value on this time is 0 V this is why it is not included on the equations 2.4 and 2.5. Also there are coefficients of \( b_n \) only for odd values of \( n \) otherwise they are 0.
The final Fourier series for the pulse signal in Fig. 2.1 is:

\[
 f(t) = \frac{1}{2} + \frac{2}{\pi} \left[ \sin(\omega_0 t) + \frac{1}{3} \sin(3 \omega_0 t) + \frac{1}{5} \sin(5 \omega_0 t) + \ldots + \frac{1}{2n-1} \sin((2n-1) \omega_0 t) \right] \tag{2.6}
\]

The Eq. (2.6) has the following shape in the spectrum domain for the first 5 harmonics (being the first harmonic frequency of 1 Hz since \( T = 1 \text{s} \)). (Matlab script of Fig. 2.5 in annex A.2).

![Fig. 2.5 Spectrum of the firsts 11 coefficients of the Fourier series of Fig. 2.1](image)

The table 2.1 show the amplitude values for the DC, odd and even frequencies of the corresponding Fourier series for the signal of the Fig. 2.1. (See reference [7] to view the Fourier Series coefficients of other signals like the triangular one).

**Table 2.1.** Coefficient values represented on the figure 2.5

<table>
<thead>
<tr>
<th>Coefficients</th>
<th>Fourier Series</th>
<th>Coefficients</th>
<th>Fourier Series</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a_0 ) (0 Hz)</td>
<td>( \frac{1}{2} = 0.5 )</td>
<td>( b_6 ) (6 Hz)</td>
<td>0</td>
</tr>
<tr>
<td>( b_1 ) (1 Hz)</td>
<td>( \frac{2}{\pi} = 0.6366 )</td>
<td>( b_7 ) (7 Hz)</td>
<td>( \frac{2}{7\pi} = 0.0910 )</td>
</tr>
<tr>
<td>( b_2 ) (2 Hz)</td>
<td>0</td>
<td>( b_8 ) (8 Hz)</td>
<td>0</td>
</tr>
<tr>
<td>( b_3 ) (3 Hz)</td>
<td>( \frac{2}{3\pi} = 0.2122 )</td>
<td>( b_9 ) (9 Hz)</td>
<td>( \frac{2}{9\pi} = 0.0707 )</td>
</tr>
</tbody>
</table>
\[
\begin{array}{|c|c|c|}
\hline
b_4 (4 \text{ Hz}) & 0 & b_{10} (11 \text{ Hz}) & 0 \\
\hline
b_5 (5 \text{ Hz}) & \frac{2}{5\pi} = 0.1273 & \\
\hline
\end{array}
\]

## 2.2.3. The Gibbs phenomenon

There is one curious case called: the Gibbs phenomenon that involves both the fact that Fourier sums of the different oscillations (harmonics) overshoot (or ‘rings’) at a jump discontinuity (Fig. 2.6), and that this overshoot does not die out as the frequency increases. Even though this error (9% best case when \( n \to \infty \)) yet the total energy error \( \to 0 \) when \( n \to \infty \). (For more information about Gibbs phenomenon see references [7], [9] and especially [1] pages 204 and 205). Gibbs phenomenon is present only when there is a jump discontinuity in \( f(t) \).

![Fig. 2.6 Gibbs phenomenon is visible especially when the number of harmonic is large (n>100)](image)

In order to prove that the total energy error tends to 0 when the number of coefficients 'n' grows its explained by the comparison of the errors obtained with the Fast Fourier Transform of the pulse signal with its Fourier Series and Fourier Transform at the same sampling rate and input frequency.
Fig. 2.7 shows the error percentage due to the difference between the coefficient values of the theoretical Fourier transform and the coefficients values of the Fast Fourier Transform of the pulse signal.

![Error vs Frequency Graph](image)

**Fig. 2.7** Error between real (Fourier Transform) and computed values (FFT), $f_s = 10$ KHz

(Matlab script of Fig. 2.7 is in annex A.11.)

We can prove that theory says when $n \to \infty$ which can be translate when the sampling frequency is sufficiently higher than the input signal frequency then the difference in the errors of graphics of Fig. 2.7 and Fig. 2.15 (Fig. 2.15 is seen in section 2.3.1 and it is shown the error due to the FFT of the pulsed signal versus the theoretical Fourier Transform values) is equal to 0.

Anyways this approach to the Fourier Series and the Gibbs phenomenon is only a matter of mathematically proven the fact that total energy errors tends to 0 as sampling frequency increases. There is a point where the sampling frequency is sufficiently high that error between both ways of computing the spectra of the pulse signal is negligible (equals to 0) but in this PFC it is not seen since our input signal is sampled and transformed to the frequency domain by the FFT and not by using the Fourier Series.

### 2.3. Numerical computation errors of the Fourier Transform

Numerical computation of the Fourier transform of $f(t)$ requires sample values of $f(t)$ because a digital computer can work only with discrete data (sequence of numbers). Moreover, a computer can compute $F(\omega)$ only at some discrete values of $\omega$ [samples of $F(\omega)$]. We therefore need to relate sample of $F(\omega)$ to samples of $f(t)$. This task can be accomplished by using the sampling theorem, where:
Also, the sampling interval \( T = 1/f_s \). Therefore:

\[
T \leq \frac{1}{2B}
\]  

(2.8)

Thus as long as the sampling frequency \( f_s \) is greater than twice the signal bandwidth \( B \) (in hertz) the computed \( \tilde{F}(\omega) \) will consist of nonoverlapping repetitions of \( F(\omega) \).

(Fig. 2.8 is an example described on the book (see reference [1]), we will later investigate this example to a computed case with Matlab and see its connections).

We begin with a timelimited signal \( f(t) \) (Fig. 2.8a, it is called timelimited since it is a finite signal in the time domain) and its spectrum \( F(\omega) \) (Fig. 2.8b). Since \( f(t) \) is timelimited, \( F(\omega) \) is non-bandlimited (a signal that's finite in time theoretically is infinite in the frequency domain). For convenience, we shall show all spectra as functions of the frequency variable \( f \) (in Hertzs) rather than \( \omega \). According to the sampling theorem, the computed spectrum \( \tilde{F}(\omega) \) of the sampled signal \( \tilde{f}(t) \) consists \( F(\omega) \) repeating every \( f_s \) Hz where \( f_s = 1/T \). This is depicted in the Fig. 2.8c and 2.8d.

2.3.1. Aliasing and Leakage in Numerical Computation

The aliasing is an effect that causes different signals to become indistinguishable (or aliases of one another) when sampled. In signal processing a signal is aliased when the sampling frequency is lower than twice the bandwidth of the signal (Eq. 2.7) making the sampled signal to take wrong amplitude values in the frequency spectrum. On the other hand, the greater is the \( f_s \) the lower is the aliasing effect of the computed signal amplitudes in the frequency spectrum (as we shall see below).

Figure 2.8d shows the presence of aliasing in the samples of the spectrum \( F(\omega) \). This aliasing error can be reduced as much as desire by increasing the sampling frequency \( f_s \) (decreasing the sampling interval \( T = 1/f_s \)). The aliasing can never be eliminated for timelimited \( f(t) \) because its spectrum \( F(\omega) \) is non-bandlimited. Imagine that we had started out with a signal having a bandlimited spectrum \( (\omega) \); there would be no aliasing in the spectrum in Fig. 2.8d. Unfortunately such a signal is non-timelimited and its repetition in time would result in a signal overlapping of its replicas (aliasing in the time domain). In this case we shall have to contend with errors in signal samples. In other words, in computing the direct or inverse Fourier transform numerically, we can reduce
the error as much as we wish, but the **error can never be eliminated**. This fact is true of numerical computation of direct and inverse Fourier Transforms, regardless of the method used. For example, if we determine the Fourier transform by direct integration numerically, using:

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t} dt$$  \hspace{1cm} (2.9)

There will be an error because the interval of integration $\Delta t$ can never be made zero (that's why increasing sampling frequency reduces this $\Delta t$ thus reducing aliasing error). Similar remarks, in addition to the problem explained in section 2.2.2 due to the fact that some signals present finite jump discontinuities as the square shape signal, apply to numeral computation of the inverse transform. Therefore, we should always keep in mind the nature of this error in our results, once the pulse signal is sampled it is not going to match the Fourier series computed in Fig. 2.5. In our discussion (Fig 2.8), we assumed $f(t)$ to be a timelimited signal. If $f(t)$ is not timelimited, we would need to timelimit it because numerical computations can work only with finite data.

As said previously, it is going to take a computed example with Matlab data in order to determine the error and compare to its theoretical value. The time-signal used is a unipolar square signal of period $T = 1 \, \text{s}$, 50% of duty cycle and 1 V of pulsed value.

The continuous Voltage-Time signal $f(t)$ is represented in Fig. 2.9a with its absolute spectra module $F(\omega)$ in (b). (see annex A.4 for the Matlab script that computes the DTFT of the Voltage-Time signal used). On the other hand we have represented the equivalent discrete signals of Fig. 2.9 in Fig. 2.10.
If $f(t)$ (Fig. 2.9a) is sampled at a rate higher than the minimum given by the Nyquist/sampling theorem, in this case at a rate of 50 Hz, the sampled function $\tilde{f}(t)$ (Fig. 2.10a) and its computed spectra with the FFT tool is as shown in Fig. 2.10b. (Matlab script to compute Fig. 2.10b in annex A.5).

If we compare the theoretical DTFT Fig. 2.9b of $f(t)$ with the numerical computation algorithm (FFT) Fig. 2.10b, errors between amplitudes of these both ways to calculate its spectrum must exist as stated above (also see Fig. 2.8d).

Higher sampling frequencies will let us see a wider spectrum thus viewing how the values of the FFT at a high frequencies tends to 0 see Fig 2.11 where it is seen the spectrum of the pulse signal sampled with a sampling frequency of 200 Hz.
In Fig. 2.11 it is represented the frequency values that forms the pulse signal, values in even frequencies are 0 as theory states (see table 2.1 Fourier series coefficients of the pulse signal).

As we see in Fig.2.12 the difference of amplitudes between both of the Fourier calculations (with DTFT and FFT) are as similar that we could say there is no difference between the mathematical true value of the spectra Fig.2.9b and the computed numerically in a software Fig.2.10b.
**Fig. 2.12** Continuous spectrum in frequency domain (DTFT) vs sampled spectrum (FFT)

To the naked eye it seems like there is no error between the amplitudes of both signals but it is not true, the error must exist even if it is really small. In that case, we are going to proceed to make a zoom in the high frequency domain (where the effect of the computed aliasing should be visible in the different amplitude values) Fig. 2.13. (Matlab script to compute Fig. 2.12 in annex A.6).

![Continuous spectrum in frequency domain (DTFT) vs sampled spectrum (FFT)](image)

**Fig. 2.13** Continuous spectrum in frequency domain (DTFT) versus sampled version (FFT)

Now, it is visible the difference between a real plot (in blue) and the numerical computed transform (in red). We can say that the samples of FFT are not exactly the same as the ones in the continuous blue from the DTFT. This is a small error that will be studied in the chapter 2.3 in order to see its impact on impedance determination.

There is another way to measure this error, between real and computed/simulated values. Real amplitude values of such signal are represented by the Fourier series coefficients (see section 2.1). If we study the amplitudes of the Fourier series of the same pulse signal and we contrast its amplitude with the amplitude values given by the FFT, we may see that amplitudes on same frequencies take different values.
Taking a look at Fig. 2.14(a) which shows the firsts 11 amplitude terms (computed with the FFT at a sampling frequency of 10 KHz) and the firsts 11 coefficients given by the Fourier series in Fig. 2.5, the amplitudes are similar but comparing these values with the table 2.2 and the table 2.1 the amplitudes roughly takes the same values.

Table 2.2. Firsts 11 amplitude values (samples) of the FFT of \( f(t) \)

<table>
<thead>
<tr>
<th>Frequency</th>
<th>FFT</th>
<th>Frequency</th>
<th>FFT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 Hz</td>
<td>0.5000</td>
<td>6 Hz</td>
<td>4.233e-17</td>
</tr>
<tr>
<td>1 Hz</td>
<td>0.636619</td>
<td>7 Hz</td>
<td>0.090945</td>
</tr>
<tr>
<td>2 Hz</td>
<td>3.867e-18</td>
<td>8 Hz</td>
<td>7.641e-17</td>
</tr>
<tr>
<td>3 Hz</td>
<td>0.212206</td>
<td>9 Hz</td>
<td>0.070735</td>
</tr>
<tr>
<td>4 Hz</td>
<td>3.843e-17</td>
<td>10 Hz</td>
<td>9.481e-17</td>
</tr>
<tr>
<td>5 Hz</td>
<td>0.127324</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
So far, we see in table 2.2 there is a small error in even amplitudes where the transform should be 0. In Fig. 2.15 the error between real and computed values is calculated. The error of the amplitudes increases as the frequency, does; this is due to the effect of the aliasing explained in section 2.3.1. In order to reduce the aliasing effect introduced by the FFT we can increase the sampling frequency: this reduces the amplitude error.

In Fig. 2.16 this error decreases with a sampling frequency ten times higher (100 KHz) than the sampling frequency used in Fig. 2.15, so it has proven that increasing the sampling frequency reduces the error of aliasing. On future computations we will have results closer to the theoretical ones by decreasing the time interval between samples on the time domain (increasing sampling frequency).

The Matlab script that calculates the error of Fig. 2.15 and Fig. 2.16 is in annex A.7.

Fig. 2.15 Error between real (Fourier series) and computed values (FFT), $f_s = 10 \text{ KHz}$
The formula used to calculate these errors in Fig. 2.15 and 2.16:

\[
Error(\%) = \frac{\text{Fourier Serie Value}(V) - \text{FFT Value}(V)}{\text{Fourier Serie Value}(V)} \times 100
\] (2.10)

Table 2.3. Firsts 11 amplitude values (samples) of the FFT of \( f(t) \) with sampling frequency of 100 KHz

<table>
<thead>
<tr>
<th>Frequency</th>
<th>FFT</th>
<th>Frequency</th>
<th>FFT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 Hz</td>
<td>0.5000</td>
<td>6 Hz</td>
<td>0</td>
</tr>
<tr>
<td>1 Hz</td>
<td>0.636619</td>
<td>7 Hz</td>
<td>0.090945</td>
</tr>
<tr>
<td>2 Hz</td>
<td>0</td>
<td>8 Hz</td>
<td>0</td>
</tr>
<tr>
<td>3 Hz</td>
<td>0.212206</td>
<td>9 Hz</td>
<td>0.070735</td>
</tr>
<tr>
<td>4 Hz</td>
<td>0</td>
<td>10 Hz</td>
<td>0</td>
</tr>
<tr>
<td>5 Hz</td>
<td>0.127324</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
If we take a look on table 2.3, the even frequencies are now 0 V as the Fourier series, and values of odd frequencies are closer to its Fourier series than the ones on table 2.2, thanks to increasing the sampling frequency.

When increasing the sampling frequency we do not only solve the problem of aliasing but also the leakage that appear on even frequencies. For this last case the amplitude values on even frequencies are 0 V while with \( f_s = 10 \) KHz the even frequencies have small amplitudes due to the leakage of nearby frequencies.

2.3.1.1. Leakage

Leakage is the smearing of energy from true frequencies of the signal into adjacent frequencies. Leakage also causes the amplitude representation of the signal to be less than true amplitude of the signal. This effect occurs when the FFT is computed from a block of data that is not periodic.

The leakage produced by the FFT itself is redundant when using an integer number of periods and can be solved by increasing the sampling frequency, but there are other cases where it takes important values.

For example, FFT treats Fig. 2.10a as a periodic signal but Fig 2.17 as an aperiodic signal since the data recorded in Fig. 2.17 has a no integral number of cycles. This assumption is violated and spectral leakage occurs.

![Fig. 2.17 Non integer number of cycles of \( f(t) \)](image)

In our case, for simulated (theoretical and TopSpice data) and experimental cases we will always try to compute the FFT with an integral number of cycles (especially one cycle in order not to consume many resources). On the other
hand, we will study when it is not exactly recording one period (Fig. 2.17) and how to minimize the problem of leakage when it occurs.

Fig. 2.17 would be the case where we will have leakage. If we take a look at the spectra on table 2.4 the amplitude values in odd frequencies measure values different to 0 due to the spectral leakage (red colour) while the amplitudes on even frequencies decreases if we compare to samples of table 2.3.

Table 2.4. Firsts 11 amplitude values (samples) of the FFT of \( f(t) \) with sampling frequency of 10 KHz

<table>
<thead>
<tr>
<th>Sample</th>
<th>FFT</th>
<th>Sample</th>
<th>FFT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0,50055</td>
<td>7</td>
<td>0,0011</td>
</tr>
<tr>
<td>2</td>
<td>0,636618</td>
<td>8</td>
<td>0,090939</td>
</tr>
<tr>
<td>3</td>
<td>0,0011</td>
<td>9</td>
<td>0,0011</td>
</tr>
<tr>
<td>4</td>
<td>0,212203</td>
<td>10</td>
<td>0,070727</td>
</tr>
<tr>
<td>5</td>
<td>0,0011</td>
<td>11</td>
<td>0,0011</td>
</tr>
<tr>
<td>6</td>
<td>0,127319</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

So the FFT introduces leakage and it also produces aliasing, as it is seen this aliasing is more visible at higher frequencies where the error between real and computed amplitudes is higher (see Fig. 2.15, 2.16 and 2.8d).

2.3.1.2. The ‘art’ of Windowing

Windowing is a technique that is applied to correct leakage. It consists in shaping a window that is exactly zero at the beginning and at the end of the data block and has some special shape in between. This function is then multiplied with the time data block forcing the signal to be periodic thus if a sequence of data is windowed, the output data have the number of samples of the window used.

For the case above in Fig. 2.17 and table 2.4, the way to solve the leakage is by windowing the time domain signal. An example of this can be set but multiplying this signal by a certain window although this will change the resultant spectra domain and we couldn’t determinate the exact effect of windowing by looking at the spectra of the resultant signal. In order to see the improvement of windowing when leakage occurs refer to the chapter 3.2.3.1.
Table 2.5. Some of the most used windows and their qualities

<table>
<thead>
<tr>
<th>Window</th>
<th>Best for these Signal Types</th>
<th>Frequency Resolution</th>
<th>Spectral Leakage</th>
<th>Amplitude Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barlett</td>
<td>Random</td>
<td>Good</td>
<td>Fair</td>
<td>Fair</td>
</tr>
<tr>
<td>Blackman</td>
<td>Random or mixed</td>
<td>Poor</td>
<td>Best</td>
<td>Good</td>
</tr>
<tr>
<td>Flat top</td>
<td>Sinusoids</td>
<td>Poor</td>
<td>Good</td>
<td>Best</td>
</tr>
<tr>
<td>Hanning</td>
<td>Random</td>
<td>Good</td>
<td>Good</td>
<td>Fair</td>
</tr>
<tr>
<td>Hamming</td>
<td>Random</td>
<td>Good</td>
<td>Fair</td>
<td>Fair</td>
</tr>
<tr>
<td>Kaiser-Bessel</td>
<td>Random</td>
<td>Fair</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>None (boxcar)</td>
<td>Transient &amp; Synchronous Sampling</td>
<td>Best</td>
<td>Poor</td>
<td>Poor</td>
</tr>
<tr>
<td>Tukey</td>
<td>Random</td>
<td>Good</td>
<td>Poor</td>
<td>Poor</td>
</tr>
<tr>
<td>Welch</td>
<td>Random</td>
<td>Good</td>
<td>Good</td>
<td>Fair</td>
</tr>
</tbody>
</table>

The most intuitive window is the rectangular one. We can say that any function in the time domain is multiplied by a rectangular window (see Fig. 2.18a).

When selecting the right window (looking at his spectrum) we will look at their two main properties:

- **Main lobe**: The greater the time length of a window the narrower the main lobe of its Fourier Transform, and higher the frequency resolution (nearby frequencies will be distinguished easily).

- **Attenuation of side lobes**: The smaller the attenuation of side lobes respect to the main lobe (typical case of rectangular windows, for the other hand they present a narrower main lobe) can produce the effect of leakage, so for interest in amplitude accuracy, we will chose windows with high attenuation of side lobes respect to the main lobe.

Fig. 2.18 shows a set of different windows, each of them with pros and cons depending on what is being looked for, if frequency resolution or amplitude accuracy. A window that gives the best frequency resolution and at the same time the best amplitude accuracy does not exist, as we can see the windowing have a trade off of prioritizing one of the two aspects.
CHAPTER 2: Theoretical Approach

(a) Rectangular

(b) Blackman

(c) Flat top
Determinació de la càrrega de bateries i condensadors mintjançant mesures d’impedància

(d) Hanning

(e) Hamming

(f) Kaiser-Bessel
Fig. 2.18 Time and Frequency Domain Window Shapes

More windows function can be found in [5]. The above images have been computed in Matlab with the function: \texttt{wvtool(kaiser(L),7)}, where \( L \) is the length of the windows. In all cases a length of 64 samples was considered and the "7" factor is a special case for this window that gives the possibility of playing with the parameter that affects the side lobe attenuation of the Fourier transform of the window (Its default values is 0.5 which spectrum and time shape resembles to a rectangular window).


2.3.1.3. Other uses of windowing

If a block of data has an infinite length it is not possible to compute the DFT since DFT is only defined for a finite block of data. On the other hand, if a block of data is finite but very large, it is possible that the compute of the FFT entails problems when extracting results in real-time. In these cases, to obtain the DFT we have to deal with subsequences of computable lengths. This data truncation causes error because of spectral spreading (smearing) and leakage. The leakage also causes aliasing. Leakage can be reduced by using tapered window for signal truncation. But this choice increase spectral spreading or smearing. The spectral spreading can be reduced by selecting the window with the properties that satisfies better the results we expect to obtain.
CHAPTER 3. MATERIAL AND METHODS

As mentioned above in the overview and introduction, three different equivalent circuits to model capacitors and batteries have been chosen. In the first section we study the theoretical impedance of such circuits. In the second part we propose a method implemented with Matlab to compute the impedance and calculate the errors due to the numerical computation for one of the circuits. In last part of this chapter the experimental lab setup to obtain the experimental voltage and current data from the different circuits is presented.

3.1. Circuit Characterization

Three different circuits are presented with the software tool TopSpice, their respective impedances and its impedance formula. Each of these circuits grows in complexity by adding more capacitor and resistor cells.

3.1.1. Circuit 1

The first circuit is formed with a resistor of 1 KΩ and a capacitor of 1 mF in series.

![Fig. 3.1 Circuit 1 schematic](image)

The impedance of this circuit is described by

\[ Z_1 = R_1 + \frac{1}{j\omega C_1} \]  

(3.1)

and is represented in Fig. 3.2. As we can see the spectrum shape starts behaving as a capacitor until this approaches to the continuous resistor value. The cut-off frequency (which is of 0.1591 Hz, see Eq. 3.2) acts as a transition between both circuit behaviours.
\[ f_1 = \frac{1}{2 \pi C_1 (R_1 + R_2)} = 0.1446 \text{ Hz} \] (3.2)

![Circuit 1 impedance](image)

**Fig. 3.2** Circuit 1 impedance

### 3.1.2. Circuit 2

The second circuit is formed with a series resistor of 100 $\Omega$ and a parallel resistor of 1 K$\Omega$ with a capacitor of 1 mF. Its impedance is given by Eq. 3.3 and represented in Fig. 3.4.

\[ Z_2 = R_1 + \frac{1}{\frac{1}{C_1} - \frac{1}{j \omega C_1 R_2}} \] (3.3)
As we can see the spectrum shape starts as a continuous resistor value formed by the sum of $R_1 + R_2$ then it behaves as a capacitor at the cut-off frequency $f_1$ (which is of 0.1446 Hz, see Eq. 3.4) until the cut-off frequency $f_2$, from $f_2$ onwards the impedance approaches to the continuous resistor value of $R_1$.

\begin{equation}
  f_1 = \frac{1}{2 \pi C_1 (R_1 + R_2)} = 0.1446 \text{ Hz}
\end{equation}

\begin{equation}
  f_2 = \frac{1}{2 \pi C_1 R_1} = 1.591 \text{ Hz}
\end{equation}
3.1.3. Circuit 3

The third circuit is formed with a series resistor of 100 Ω plus two parallel RC cells as depicted in Fig. 3.5. The Eq. 3.5 describes the formula of impedance for the circuit 3, which is represented in Fig. 3.6.

\[
Z_2 = R_1 + \frac{1}{j\omega + \frac{1}{C_1R_2}} + \frac{1}{j\omega + \frac{1}{C_2R_3}} \tag{3.5}
\]

The Eq. 3.6 calculates the equivalent capacitance of \( C_1 + C_2 \) which determines the impedance slope between \( f_3 \) and \( f_4 \) regions in Fig. 3.6.

\[
C_{12} = \frac{C_1C_2}{C_1 + C_2} = 90.90 \mu F \tag{3.6}
\]
As we can see the spectrum shape starts as a continuous resistor value formed by the sum of $R_1 + R_2 + R_3$ then at $f_1$ it immediately starts behaving as a capacitor of 1 mF until $f_2$, in the transition from $f_2$ up to $f_3$ there can be found the contribution of $R_1 + R_2$ punctually, at $f_3$ it behaves as a capacitor of value of $C_1 + C_2$ (see Eq. 3.6) up to $f_4$, from $f_4$ onwards the impedance approaches to the continuous resistor value of $R_1$. Eq. 3.7 determines the frequency values where impedance change its behaviour.

\[
f_1 = \frac{1}{2\pi C_1 (R_1 + R_2 + R_3)} = 2.61 \cdot 10^{-2} \text{Hz}
\]

\[
f_2 = \frac{1}{2\pi C_1 (R_1 + R_2)} = 0.1446 \text{Hz}
\]

\[
f_3 = \frac{1}{2\pi C_{12} (R_1 + R_2)} = 1.5917 \text{Hz}
\]

\[
f_4 = \frac{1}{2\pi C_{12} R_1} = 17.508 \text{Hz}
\]
3.2. Matlab Process

In this chapter, we explain the process of computing the impedance with the Matlab script developed using the Eq. 3.8. The computed impedance is done for the circuit 1 using the pulse signal.

\[
Z(\omega) = \frac{V(\omega)}{I(\omega)}
\]  

(3.8)

This chapter is split up in three sections. In the first one we will compute the impedance using the voltage and current signals created directly in Matlab. In the second section we will compute the impedance with the data of the voltage and current signals previously extracted from TopSpice (to view the process of extracting the data of time signals from TopSpice go to annex B). Finally, we will compare the errors resulting from both of these procedures.

3.2.1. Computed Impedance with Matlab Data

The process steps to compute the impedance for this case is shown in the following diagram block:
The Matlab script to perform these calculations is given in Annex A.8. In the part 1, the impedance for the circuit 1 is computed. The voltage source (pulse signal) is built with the input frequency \( f_0 = 10 \text{ mHz} \) sampled at the rate of 1 KHz \( (f_s) \), the time between samples are given by the sampling interval \( 1/f_s \). The corresponding input current ‘i’ of the circuit 1 is computed with the known voltage source and the elements that form this circuit \((R_1 \text{ and } C_1)\).

In part 2, the FFT of both signals by the command in Matlab ‘fft’ is computed. The FFT yields the spectrum information of these signals between the frequencies 0 and \( f_s \) where the spectrum from 0 Hz to \( f_s/2 \) is a mirror of \( f_s/2 \) to \( f_s \), that’s why we will only represent the first part (from 0 Hz up to \( f_s/2 \) Hz). For this representation, we will have to choose half of the points of the FFT of both signals and divide the amplitudes of the resulting transformation by the total number of samples of the ‘fft’ (see: AmplitudeV=FFThalfV/length(FTV)). This division is done since the ‘fft’ computes the value of one harmonic (point/sample) over the total number of points of the sampled signals \( v[n] \) and \( i[n] \) so we got to divide each amplitude (harmonic) by the total number of samples. After normalizing the amplitudes of both signals, we will have to multiply by 2 these amplitudes since we will only represent half of the spectrum (remember from 0 Hz up to \( f_s/2 \), but we will exclude the DC component and the Nyquist component (in case that this one exist) of being multiplied by 2. The frequency axis is calculated up to \( f_s/2 \). The number of the FFT points corresponds to the number of samples of the input signal (see Fig. 1.3).

It is inferred that if we choose a higher sampling frequency, not only the resulting values of the FFT improve but also a larger part of the spectrum is computed. In this case, as we see in Fig. 3.2 the values of the impedance is the same for frequencies higher than 1 Hz. Thus taking one more decade of spectrum is already enough to be able to see the whole spectrum despite of such spectrum is halved in two sections and we will only see up to \( f_s/2 \), which is 500 Hz.

The impedance ‘\( Z_{in} \)’ is computed by dividing the values of amplitudes of the FFT of the voltage source by the current ones. We then, take the absolute values of the FFT of both signals, since the FFT yields the amplitudes on its complex form.
Then these values of voltage and intensity in the frequency domain are saved in new variables called `absAmplitudeVPRO` and `absAmplitudeIFPRO` where only the amplitudes in odd frequencies (also for the amplitude in frequency 0) will be saved. As we know from the previous chapter the Fourier series of the pulse signal have harmonics in the DC and the odd frequencies hence the corresponding current wave will have information only in such frequencies. This is done to avoid dividing values on even frequencies since the corresponding values of impedance for such frequencies will be of the order of 0/0 and we will have infinite unreal amplitudes of $Z_{in}$ for these frequencies.

Finally, $Z_{in}$ is now normalized for the values where the Fourier Series exists. We can compute this in $dB$ with the command: $ZinPROdB=20*log10(ZinPRO)$ in order to represent the $Z_{in}$ in $dB$ vs frequency in Hz with logarithmical scale (command: `semilogx`).

The representation of the computed impedance is done against the theoretical one given by the Eq. 3.1.

To evaluate the part 3 of this script, the errors resulting from the computation of this impedance is calculated against the values of the theoretical impedance, in addition it is calculated the standard deviation of such error.

Fig. 3.8 and 3.9 show the impedance simulated versus the theoretical one, for different sampling frequencies (10 Hz and 100 Hz).
If we compare both figures, we can see how the computed impedance approaches to the theoretical one as we increase the sampling frequency thus translating in a reduction of the error in $Z_{in}$ (the errors resulting from this computation are shown in the section 3.2.3 of this chapter).

### 3.2.2. Computed Impedance with TopSpice Data

To compute the impedance for data obtained from simulations we perform a four-step process, as illustrated in Fig. 3.10.

1. **Read and load the data of voltage and current signals from TopSpice files**
2. **Interpolate the Voltage and Current time signals with a certain sampling frequency**
3. **Transform the interpolated time signals into its frequency domain (FFT) and obtain the impedance $Z(\omega)$ (Eq. 3.8)**
4. **Calculation of errors due to the numerical computation of $Z(\omega)$**

![Diagram block](image-url)
In this case the circuit 1 is created in TopSpice with the corresponding parts in addition of the input pulse signal. (See annex B to view TopSpice process).

In the part 1 of the script in annex A.9, first we load the data of both signals (voltage and current) extracted from the TopSpice, as shown in Fig. 3.11a and b.

The samples from TopSpice of $v(t)$ and $i(t)$ are not spaced with the same interval (see annex B Fig. B.6 time samples). As we can see in both graphics of Fig. 3.11 there are many amplitude values relatively closer when time is equal to 0 and $T/2$ seconds so interpolation step is needed to make both signals be x-spaced.
In the part 2, the time vector ‘t’ is created for one period of the signal thus taking as first time sample of the vector the first time sample given by the time data extracted from TopSpice ‘tv(1)’ until the period of the signal which is read by the last time sample ‘period=tv(end)’ of the time vector from TopSpice in addition from this period is deducted one sample ‘-(1/fs)’ else we would have leakage in the spectrum since that sample at the end of the time vector corresponds to one sample of the next cycle (second period). The samples of the time vector are x-spaced by the sampling interval 1/fs.

The next step is to compute interpolation of the voltage and current signals by the command: ‘interp1’ the voltage signal is constructed over the time vector ‘t’ using the time and voltage data from TopSpice thus resulting in a signal where all samples are spaced by the same time interval which depends of the sampling frequency.

The interpolation step is needed in order to relate the number of samples of the voltage and current signal with the sampling frequency besides of having samples spaced with the same interval that yield in the frequency spectrum frequencies related to the Fourier series of the input signal. When the sampling frequency increases we decrease the Δt making the time signal to have its amplitude value closer to the real one thus translating in a reduction of the error (as explained in chapter 2.3.1).

The resulting signals from interpolation are depicted in Fig. 3.12a and b.
In comparison with Fig. 3.11, graphics a and b of Fig. 3.12 do not have greater quantity of amplitude values (as it is visible) when time is equal to 0 and $T/2$ seconds.

The explanation of the computation of the fft, $Z_{in}$ and errors of this simulation is the same as the ones explained in section 3.2.2 (part 2 and 3).

The Fig. 3.13 and 3.14 shows the impedance simulated versus the theoretical one, for different sampling frequencies (10 Hz and 100 Hz).
If we compare both figures, we can see how the simulated impedance approaches to the theoretical one as we increase the sampling frequency thus translating in a reduction of the error in $Z_{\text{in}}$ (the errors resulting from this computation are shown in the section 3.2.3 of this chapter).

### 3.2.3. Error Calculation

In this section, the error between the theoretical impedance given by the Eq. 3.1 and the impedances simulated in the chapters 3.2.1 and 3.2.2 are going to be exposed.

The error for $Z_{\text{in}}$ that is calculated with the following formula:

$$\text{Impedance Error (\%)} = \left| \frac{\text{Theoretical Impedance (dB)} - \text{Simulated Impedance (dB)}}{\text{Theoretical Impedance (dB)}} \right| \times 100 \tag{3.9}$$

Once the error of the impedance is calculated on each frequency, we are going to compute the average value of this error ($AE$) with the Eq. 3.10.
Another tool that lets us measure the quality of the error obtained with Eq. 3.9 is the Standard Deviation (SD) parameter shown in Eq. 3.11. This expression is used to calculate the equidistance of the impedance errors obtained with Eq. 3.9. Lower values of SD means that error values are more equidistant hence translating in a more trusted result as SD of the error approximates to 0. (See reference [16] related to Standard Deviation).

\[
Average\ Error\ (%) = \frac{1}{N} \sum_{n=1}^{N} \text{Impedance\ Error}_n(\%)
\]  

(3.10)

\[
Standard\ Deviation\ Error\ (%) = 100 \cdot \sqrt{\frac{1}{N} \sum_{n=1}^{N} (\text{IE}_n - \text{AE})^2}
\]  

(3.11)

The variable ‘IE’ corresponds to the Impedance Error in each frequency (see Eq. 3.9) while the variable ‘AE’ corresponds to the Average Error in Eq. 3.10.

These three last formulas are computed in the Matlab scripts part 4 (see annex A.8 i.e.).

In the table 2.1, is shown the AE and SD for the two different cases at two different sampling frequencies.

**Table 2.1 Values of errors for Circuit 1**

<table>
<thead>
<tr>
<th>Error in % Calculated with the Computation of Matlab Data</th>
<th>$f_s = 10\ Hz$</th>
<th>$f_s = 100\ Hz$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AE</td>
<td>0.71</td>
<td>0.071</td>
</tr>
<tr>
<td>SD</td>
<td>0.019</td>
<td>0.0019</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Error in % Calculated with the Computation of TopSpice Data</th>
<th>$f_s = 10\ Hz$</th>
<th>$f_s = 100\ Hz$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AE</td>
<td>0.72</td>
<td>0.069</td>
</tr>
<tr>
<td>SD</td>
<td>0.02</td>
<td>0.0057</td>
</tr>
</tbody>
</table>

The conclusions that we can extract by looking at the table 2.1 are that best results (AE and SD) are obtained when the impedance is computed from the theoretical expressions of $v(t)$ and $i(t)$.

On the other hand the AE error with TopSpice data (for $f_s = 100\ Hz$) simulation is smaller than the Matlab one but its SD is worse. The Matlab case yields the
closest results to the theoretical since with the TopSpice method we are loading samples of data that have been previously sampled in other software (TopSpice) thus introducing small errors in amplitude accuracy and time resolution. Experimental results seen in chapter 4 will be closer to the results obtained in TopSpice case than in the Matlab one.

3.2.3.1. **Error Calculation in case of Leakage**

In this section we evaluate the error due to leakage as introduced in sections 2.3.1.1. and 2.3.1.2. The effect of leakage occurs when the sampled signal $V[n]$ has a non-integer number of cycles like in this case (see Fig. 2.17) thus being the corresponding current signal $I[n]$ also with the same non-integer number of cycles.

The resulting simulated impedance of the circuit 1 computed with the corresponding $V[n]$ and $I[n]$ signals is shown in Fig. 3.15.

![Fig. 3.15 Circuit 1 impedance, effect of leakage](image)

As we see in Fig. 3.15 the effect of leakage makes the simulated impedance to take serious wrong values. In order to solve this problem, the $V[n]$ and $I[n]$ signals are multiplied in the time (discrete) domain by the window Kaiser-Bessel (see annex A.8 part 1) with the commands: $V=V.*kaiser(length(V),7)'$ for the voltage signal and $I=I.*kaiser(length(I),7)'$ for the current signal.

The window Kaiser-Bessel offers one of the greater leakage factor (0%) and relative side lobe attenuation (-66dB) thus yielding one of the best amplitude
accuracy (review section 2.4.1.2.). See Fig. 3.16 of the impedance of circuit 1 where the “leaking” signals have been windowed thus translating in an improvement of the impedance errors depicted in the table 3.2 as well as a visual enhancement of the impedance.

### Table 3.2 Error comparison between leakage and corrected leakage (windowing)

<table>
<thead>
<tr>
<th></th>
<th>$f_s = 100$ Hz</th>
</tr>
</thead>
<tbody>
<tr>
<td>AE</td>
<td>0.071</td>
</tr>
<tr>
<td>SD</td>
<td>0.0019</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$f_s = 100$ Hz</th>
</tr>
</thead>
<tbody>
<tr>
<td>AE</td>
<td>10.86</td>
</tr>
<tr>
<td>SD</td>
<td>13.36</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$f_s = 100$ Hz</th>
</tr>
</thead>
<tbody>
<tr>
<td>AE</td>
<td>0.16</td>
</tr>
<tr>
<td>SD</td>
<td>0.071</td>
</tr>
</tbody>
</table>

The $AE$ and $SD$ errors from the “windowed” impedance highly improve from the respective values given by the “leaking” case thus highlighting the proven and correct use of windowing.

The window Flat Top (Fig 2.18c) introduces greater relative side lobe attenuation than the Kaiser-Bessel although Flat Top window has a wider main lobe (worse frequency resolution). This translates that amplitudes in frequencies closer to the starting frequency (0 Hz) takes worse amplitude values due to the
frequency resolution mismatch between the frequencies of interest and the closest lobules to the main lobe.

As we see in Fig. 3.17 when correcting the effect of leakage with the Flat Top window, the impedance value is not as good as the Kaiser window for low frequencies. On the other hand for high frequencies we can see that the Flat Top window approaches more to the theoretical impedance although in both cases (Kaiser and Flat Top windows) the error is almost negligible in this range so it is considered that the Kaiser-Bessel approaches more to the theoretical impedance since for low frequencies this yields better results (see simulated impedance of Fig. 3.16 and 3.17 at low frequencies).

As we explained in section 2.3.1.2 there is not an exact window that is the best, but it is an art of understanding which window should be applied in order to approach more to the results we expect to obtain.

3.3. Laboratory set up

In this section, the laboratory components that we have used to measure and extract the data information of the voltage and current signals injected in the three circuits using different input signals are explained.

We have built the 3 different circuits in a protoboard (see one of the circuits built in Fig. 3.18).
The equipment shown in Fig. 3.19 is a Biologic VSP that uses different probes to inject the voltage signal and measure the corresponding current signal of the circuit built in the protoboard.
The VSP is controlled automatically with the software EC-Lab V10.40. We can build the 3 different input voltage signals (square, triangular and trapezoidal) here and we can select how we want that these be, for example the sampling interval (maximum of 1 ms), the maximum voltage, a ramp voltage, etc...

For example in Fig. 3.20, we depict the triangular signal built in blue colour and the corresponding measured current of the circuit in red for the circuit 3. Each input signal has been created for each circuit, selecting its appropriate range of measurements like current amplitude range.

Once the current of the circuit is measured, we can export in a txt file the voltage-time and current-time data measured by this device in order to load it from Matlab and compute the experimental impedance.
For the sake comparison, we also measured the impedance by impedance spectroscopy, as illustrated in Fig. 3.21. We can select the starting and ending frequency and the number of points per decade that we want to have. The program uses sinusoids of different frequencies inside the selected range in order to measure the circuit impedance.

We have used the impedance spectroscopy tool of VSP to compare this impedance measurement with the computed impedance from the experimental data of V and I extracted with this device.
Fig. 3.21 EC-Lab Software Impedance measurement
CHAPTER 4. RESULTS

In the first part of this chapter the three excitation signals used to measure the impedance of the different circuits are presented. In the second part we evaluate the computed impedance of circuits with the experimental data and TopSpice data. Finally, conclusions will be extracted after contrasting the results obtained.

4.1. Input Signals

The presentation of the method used in this work has been done with the pulse signal, from which the evaluation of FFT and the impedance had been done so far. In this chapter we will generalize the method to three signals used to evaluate the impedance of the circuits proposed. In the Fig. 4.1 we see their time domain shape. The reason to use these signals is because they are common switching on and off batteries and supercaps.

![Square Wave](image)

![Triangular Wave](image)

![Trapezoidal Wave](image)

**Fig. 4.1 Square, triangular and trapezoidal Excitation signals**

As mentioned on previous chapter, different harmonics signals like the proposed above have been used. These signals are easily built in a simulation software by switching on and off a continuous voltage value. The characteristic of the triangular signal in comparison to the pulse signal is that it has not a finite
jump discontinuity (review section 2.2.2 to see which impact has a jump discontinuity of \( f(t) \) in its spectra domain) so the spectra decays faster as frequency increases (see spectra, in red, of triangular signal in Fig. 4.2). On the other hand the trapezoidal signal is a more realistic case of the pulse signal since its fall and rise time do not switch automatically from 0 V to 10 V. We can see in Fig. 4.2, green colour, that spectra of the trapezoidal signal decays slower than spectra of triangular signal but is closer the lobules amplitude of the pulse signal due to the fall and rise time are not long.

The corresponding modulus of each signal is shown in Fig. 4.2.

![Fig. 4.2 Excitation signals spectra](image)

The amplitude of the spectra module of the excitation signals will be determinant when calculating the errors of the impedance computation in the next sections.

### 4.2. Experimental Results

In this section we will compute the impedance with the data obtained from the experimental case and we will contrast this impedance with the impedance obtained with the data of TopSpice.
The diagram block in Fig. 4.3 shows the steps used in Matlab to compute the impedance using the data of EC-Lab.

Fig. 4.3 Diagram block

Several properties of the input signal have been chosen in order to measure experimentally the voltage and current signals for the correct impedance computation in the different circuits:

1- The input frequency \( f_\text{in} \) of the excitation signals is 10 mHz. This frequency has been chosen due to the limitation of having a maximum sampling frequency given by the device VSP (of 1 KHz). As the relation of \( f_\text{s}/f_\text{in} \) increases the value of the computed amplitudes get closer to the theoretical ones in addition the input frequency acts as the minimum frequency that we can compute and visualize in the impedance spectra.

2- An amplitude of 10 V for all signals. The problem of selecting a lower amplitude for example of 1 V, gives worse results when computing the impedance since the voltage measured by probes of the system add small oscillation amplitude errors on the measured voltage (see Fig. 4.4) current signals (see Fig. 4.5). The impact of this low amplitude oscillation errors are lowered by selecting a higher amplitude for example of 10 V thus minimizing this error (since signal-to-noise ratio ‘SNR’ improves) that may has been introduced by the probes of the device, the minimum resolution that the device can yield, some parasitic effect in the protoboard/components, thermal noise, etc…

3- Each input signal starts at 0 V during 1 second, this has been selected because when we introduce, for example, a pulse signal the transition of the corresponding current signal during the interval where it reaches the pick current of 10 V/1KΩ = 10 mA is made in an infinitesimal period and the software is not capable of reaching this pick value unless we do this procedure. So it was seen that selecting a previous period of 0 V helped the amplitude of the current signal to take a closer value to the theoretical one (10 mA).
4.2.1. Results for Circuit 1

In the following impedance figures we will see in (a): the theoretical impedance (green) analytically obtained from the mathematical equations, the experimental impedance computed with the data recorded in the software EC-
Lab (blue) and the impedance spectroscopy (red) obtained with the analyser. For cases (b): the theoretical impedance (green) and the simulated impedance computed with the data recorded in the circuit simulation program ‘TopSpice’ (blue).

Fig. 4.6 Impedance with Pulse signal

Fig. 4.7 Impedance with Triangular signal

Fig. 4.8 Impedance with Trapezoidal signal
Experimental impedance computed using the pulse excitation in Fig. 4.6a shows a good behaviour profile up to around 70-80 Hz. At this point, errors mentioned above that lead us to determine the properties of the input signal, distort the impedance spectra, so that results are not valuable from 80 Hz.

When selecting a triangular or a trapezoidal excitation (Fig. 4.7a and 4.8a) its good behaviour profile decays to the 2 Hz. In these cases, it is added to the errors introduced by the experimental scenario, the spectra shape of these signals. If we take a look on the Fig. 4.2 we see that the module of the pulse signal is greater than the other two signals at high frequencies thus having greater values of modulus improves the accuracy of impedance calculations while for the triangular and trapezoidal their modules decay faster as the frequency increases translating in a shorter impedance measurable bandwidth of 2 Hz (see table 4.1 for the measurements of these errors).

The limitation of computing a good spectra profile of impedance is due to the sampling frequency mainly. The maximum sampling frequency that VSP can work with is 1 KHz so the maximum visible spectra is limited to \( f_s/2 \). Since this scenario introduces additional errors as mentioned in section 4.2 2, the quality of the impedance is reduced hence decreasing the impedance bandwidth to 70-80 Hz and 2 Hz for the others.

On their TopSpice cases (Fig. 4.6b, 4.7b and 4.8b) the corresponding simulated impedances yield better results since errors due to the experimental scenario are not introduced thus having a good behaviour profile of impedance up to \( f_s/2 \) for pulse excitation and up to 10 Hz for triangular and trapezoidal ones.

In the case of trapezoidal signal, its module is greater than the pulse one for low frequencies although the error introduced by the digital process is very small at low frequencies so in this range the difference of impedance values using these two signals is negligible.

**Table 4.1** Comparison of errors between real case (experimental) and simulated case (TopSpice)

<table>
<thead>
<tr>
<th>Error Calculation up to the frequency 2 Hz with a sampling frequency of 1 KHz for both cases experimental and TopSpice</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Experimental (%)</td>
<td>TopSpice (%)</td>
</tr>
<tr>
<td>Pulse</td>
<td>A E</td>
<td>0.096</td>
</tr>
<tr>
<td></td>
<td>S D</td>
<td>0.15</td>
</tr>
<tr>
<td>Triangular</td>
<td>A E</td>
<td>0.19</td>
</tr>
<tr>
<td></td>
<td>S D</td>
<td>0.16</td>
</tr>
<tr>
<td>Trapezoidal</td>
<td>A E</td>
<td>0.16</td>
</tr>
<tr>
<td></td>
<td>S D</td>
<td>0.14</td>
</tr>
</tbody>
</table>

In table 4.1 it is calculated the Average Error and its SD of the computed impedances for the different excitation signals and for experimental and TopSpice cases. As we could expect the pulse signal yields better results than
the triangular and trapezoidal ones due to its greater modulus (see Fig. 4.2). Experimental results are worse than TopSpice ones due to the additions of errors introduced as explained in section 4.2.2. However, they remain below 0.2% in any case, which means that the method is of interest.

4.2.2. Results for Circuit 2

In addition of the figures exposed above, in this section it is introduced a new case ‘a.1’ that shows the experimental impedance computed with an input frequency of 2 Hz (for triangular and trapezoidal excitations).

![Fig. 4.9 Impedance with Pulse signal](image)
Determinació de la càrrega de bateries i condensadors mintjant mesures d’impedància

Fig. 4.10 Impedance with Triangular signal

(a)

(b)

Fig. 4.11 Impedance with Trapezoidal signal

Similar explanations as in section 4.2.1 can be used to explain the figures and results of the impedance measurements for circuit 2 shown in the figures above. On one hand the limitation of having a maximum sampling frequency of 1 KHz
and the addition of the experimental scenario errors make the computable bandwidth to be decreased at the same frequencies as in 4.2.1 figures for same excitation signals. On the other hand the impedance of circuit 2 shows an impedance behaviour that goes further than the limitation frequency of 2 Hz given by the triangular and trapezoidal excitations.

In order to reach further spectra of impedance measurements with a good quality, it has been used a new case ‘a.1’ where the input frequency has been increased up to 2 Hz, this helps to improve the impedance measurements in the range where it was impossible to have a good behaviour profiles of impedance. If we take a look at Fig. 4.10a the quality of impedance is limited at 2 Hz, but in Fig. 4.10a.1 the impedance yields a good behaviour profile up to the 80 Hz. This method can be used to provide a bit more than 1 decade of quality for the impedance measurement.

### Table 4.2 Comparison of errors between real case (experimental) and simulated case (TopSpice)

<table>
<thead>
<tr>
<th>Input Frequency of 10 mHz</th>
<th>Experimental(%)</th>
<th>TopSpice(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Up to 2 Hz</td>
<td>Up to 80 Hz</td>
</tr>
<tr>
<td>Pulse</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AE</td>
<td>0,08 (up to 80 Hz)</td>
<td>0,105</td>
</tr>
<tr>
<td>SD</td>
<td>0,11 (up to 80 Hz)</td>
<td>0,0058</td>
</tr>
<tr>
<td>Triangular</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AE</td>
<td>0,36</td>
<td>0,067</td>
</tr>
<tr>
<td>SD</td>
<td>0,26</td>
<td>0,042</td>
</tr>
<tr>
<td>Trapezoidal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AE</td>
<td>0,34</td>
<td>0,0702</td>
</tr>
<tr>
<td>SD</td>
<td>0,19</td>
<td>0,045</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input Frequency of 2 Hz</th>
<th>From 2 Hz up to 80 Hz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triangular</td>
<td></td>
</tr>
<tr>
<td>AE</td>
<td>0,19</td>
</tr>
<tr>
<td>SD</td>
<td>0,23</td>
</tr>
<tr>
<td>Trapezoidal</td>
<td></td>
</tr>
<tr>
<td>AE</td>
<td>0,37</td>
</tr>
<tr>
<td>SD</td>
<td>0,74</td>
</tr>
</tbody>
</table>

In table 4.2 it is calculated the Average Error and its SD of the computed impedances for the different excitation signals and for experimental and TopSpice cases of circuit 2.

Quality of experimental impedance is under the quality of the simulated impedance with TopSpice even though its impedance spectra reproduces the theoretical spectra of impedance until some point (frequency). As we could expect the pulse signal yields better results than the triangular and trapezoidal ones due to its greater modulus. Errors remain below 0.4% up to the frequency of 2 Hz and the same percentage still remains when measuring the impedance errors up to the frequency of 80 Hz using an input frequency of 2 Hz.
4.2.3. Results for Circuit 3

Similar explanations as in section 4.2.1 and 4.2.2 can be used explain the figures and results of the impedance computation for circuit 3.

![Fig. 4.12 Impedance with Pulse signal](image-url)
In this section, it has also been introduced the case a.1 for the pulse excitation (Fig. 4.12a.1) since there is an impedance behaviour of interest over the 50 Hz.

**Fig. 4.13** Impedance with Triangular signal

**Fig. 4.14** Impedance with Trapezoidal signal
The 50 Hz is the frequency limit for having a good behaviour profile of impedance so it has been selected a second input frequency of 10 Hz for computing the impedance of circuit 3 using the pulse excitation. The use of an input frequency of 10 Hz yields good impedance measurements up to a frequency of 100 Hz.

Table 4.3 Comparison of errors between real case (experimental) and simulated case (TopSpice)

<table>
<thead>
<tr>
<th>Input Frequency of 10 mHz</th>
<th>Experimental(%)</th>
<th>TopSpice(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Up to 2 Hz</td>
<td>Up to 100 Hz</td>
</tr>
<tr>
<td><strong>Pulse</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>AE</em></td>
<td>0.19 (up to 10 Hz)</td>
<td>1.11</td>
</tr>
<tr>
<td><em>SD</em></td>
<td>0.17 (up to 10 Hz)</td>
<td>0.12</td>
</tr>
<tr>
<td><strong>Triangular</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>AE</em></td>
<td>0.17</td>
<td>0.44</td>
</tr>
<tr>
<td><em>SD</em></td>
<td>0.31</td>
<td>0.33</td>
</tr>
<tr>
<td><strong>Trapezoidal</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>AE</em></td>
<td>0.26</td>
<td>0.45</td>
</tr>
<tr>
<td><em>SD</em></td>
<td>0.24</td>
<td>0.35</td>
</tr>
<tr>
<td><strong>Input Frequency of:</strong></td>
<td>From 2 Hz up to 100 Hz</td>
<td></td>
</tr>
<tr>
<td><strong>Pulse</strong> (10 Hz)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>AE</em></td>
<td>0.16 (From 10 Hz)</td>
<td></td>
</tr>
<tr>
<td><em>SD</em></td>
<td>0.07 (From 10 Hz)</td>
<td></td>
</tr>
<tr>
<td><strong>Triangular</strong> (2 Hz)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>AE</em></td>
<td>0.94</td>
<td></td>
</tr>
<tr>
<td><em>SD</em></td>
<td>0.88</td>
<td></td>
</tr>
<tr>
<td><strong>Trapezoidal</strong> (2 Hz)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>AE</em></td>
<td>0.56</td>
<td></td>
</tr>
<tr>
<td><em>SD</em></td>
<td>1.04</td>
<td></td>
</tr>
</tbody>
</table>

In table 4.3 it is calculated the Average Error and its SD of the computed impedances for the different excitation signals and for experimental and TopSpice cases of circuit 3.

Quality of experimental impedance is under the quality of the simulated impedance with TopSpice even though its impedance spectra reproduces the theoretical spectra of impedance until some point (frequency). As we could expect the pulse signal yields better results than the triangular and trapezoidal ones due to its greater modulus. Errors remain below 0.3% up to the frequency of 2 Hz when using a triangular and trapezoidal excitation signal with an input frequency of 10 mHz. When it is used a pulse excitation of 10 mHz the errors remains below 0.2% up to a frequency of 10 Hz.

Errors results are still under the 1% when using an input frequency of 2 Hz and measuring up to the 100 Hz, for pulse excitation errors still remain below the 0.2% from the 10 up to the 100 Hz.

As shown in all the figures of this chapter the impedance curves in red are measured by the system device with the method explained in chapter 3.3 even though it is done with most known implemented method where numerical
computation errors should not be presented as it is not used harmonic signals. The impedance obtained in red yields worst results for a larger range of bandwidth thus only offering good results mainly at high frequencies. On the other hand the method carried out in this PFC yields worst results at high frequencies where errors of numerical computation are more predominant in addition of errors explained in 4.2. 2. In contrast, if sampling frequency could be selected as high as we desire (sampling frequency of 1 KHz is quite small frequency) or at least some decades higher, experimental impedances would yield less error mainly at high frequencies and would let us measure larger impedance bandwidths.
CONCLUSIONS

The method of impedance computation using harmonic excitation signals (pulse, triangular and trapezoidal) is presented together with the whole process of digital signal processing. Previous research of the method has been performed with use of pulse signal for a theoretical case (section 3.2.1) in order to study the limitations of the method and the side effects introduced by the numerical computation.

At first, the excitation signals mentioned in the last paragraph were tested in a circuit simulation software (TopSpice) for consequently export this case to an experimental one. Together with the use of different excitation signals that were compared by taking into account their spectral module and the corresponding results obtained from the impedance calculation errors, performed research shows usefulness of pulse excitation since it guarantees lowest object’s spectrum calculation errors for all analysed circuits compared to triangular and trapezoidal excitations due to the jump discontinuity introduced by the pulse excitation that lowers the lobule amplitude decaying factor (greater module of signal spectra as frequency increases than the triangular and trapezoidal ones).

It is also studied and proven the usefulness of a digital processing technique named ‘windowing’ that corrects the leakage error using a determined window when the excitation signals are not periodic.

Practical experiments in measuring system led to the conclusion that it is justified to decrease the range of \( f_{min} \) and \( f_{max} \) in order to increase accuracy impedance spectrum calculations as seen in Fig. (a.1) of chapter 4 were the input frequency of the different excitation signals were changed in order to measure different ranges of the spectrum. This method also would yield good results when measuring high impedance bandwidths if we could use higher sampling frequencies thus selecting the appropriate sampling and input frequencies for the range of frequencies that we want to measure (from a minimum frequency up to a maximum one).

So finally, we can say, that the best approach is to use a pulse signal with conditions presented in section 4.2.2, this makes the impedance computation errors be below the 0.2% for the 3 different scenarios together with the selection of a sampling frequency as large as possible.

Regarding my personal evaluation about this PFC I would like remark several aspects that have provided me a great gratification about the knowledge carried out; I have understood and applied more deeply the Fourier transform, Discrete Time Fourier Transform, Discrete Fourier Transform and Fast Fourier Transform algorithm not only numerically computed but also its understanding over paper, the use of differential equations in order to demonstrate the current and voltage transient curves calculation for the different circuits and excitation signals, the Fourier series applied to a pulse signal and to any other wave that satisfied the condition of existing the corresponding Fourier series and its connection with Fourier Transform and spectra calculation, the different
changes in the excitation signal properties due to experimental errors introduced. Also the review and improvement of circuit simulation with software TopSpice and a greater knowledge about Matlab functions for digital signal processing and other mathematical usages. As last point, I have also used and learnt for first time how to work with a Biologic VSP in order to carry out the experimental labour seen in the chapters 3 and 4 of this PFC.
REFERENCES


[6] Geometric wave Fourier analysis http://hyperphysics.phy-astr.gsu.edu/hbase/audio/geowv.html#c1


[14] FFT & Matlab example http://www.public.iastate.edu/~e_m.350/FFT%205.pdf
[15] FFT & Other transforms  

[16] FFT Matlab implemented algorithm, decimation in time and frequency  

[17] Matlab interpolation methods  

[18] Matlab FFT  
http://www.mathworks.es/help/techdoc/ref/fft.html
ANNEXES

ANNEX A. MATLAB SCRIPTS

1. Square Signal built with its Trigonometric Fourier Series

```matlab
clear;
t=0:0.001:2;

%SEÑAL CUADRADA TRIGONOMETRICA
xx=0;
for n=0:99 %number of coefficients
x=((2/pi)*(1/(2*n+1))*((sin(2*pi*(2*n+1)*t))));
xx=xx+x;
end
y=xx+0.5;
plot(t,y)
xlabel 'Time (s)'
ylabel '|Amplitude (V)|'
```

2. Square Signal Fourier Series spectra coefficients

```matlab
zz=2;
for z=0
    SerieFourierV(z+1)=0.5;
    Frec(z+1)=0;
end
for z=1:(n-1)
    SerieFourierV(zz)=(1/(z*pi))*((-1)^(z-1)+1);
    Frec(zz)=z;
    zz=zz+1;
end
stem(Frec,SerieFourierV)
xlabel 'Frecuency (Hz)'
ylabel '|Amplitude (V)|'
```

3. Square Signal with Fourier Series - Gibbs phenomenon

```matlab
clear;
t=0:0.001:1;
y0=0.5+ 0.5*square(2*pi*t);

%SEÑAL CUADRADA TRIGONOMETRICA
xx=0;
for n=0:2
x=((2/pi)*(1/(2*n+1))*((sin(2*pi*(2*n+1)*t))));
xx=xx+x;
end
y1=xx+0.5;
x=0;
```
xx=0;
for n=0:10
  x=((2/pi)*((1/(2*n+1))*((sin(2*pi*(2*n+1)*t)))));
  xx=xx+x;
end
y2=xx+0.5;
x=0;
xx=0;
for n=0:99
  x=((2/pi)*((1/(2*n+1))*((sin(2*pi*(2*n+1)*t)))));
  xx=xx+x;
end
y3=xx+0.5;

hline0=plot(t,y0,'m', 'linewidht',2);
hold on
hline1=plot(t,y1,'r');
hline2=plot(t,y2,'g');
hline3=plot(t,y3,'b');
hold off

h=[hline0;hline1;hline2;hline3];
xlabel 'Time (s)'
ylabel 'Amplitude (V)'
legend('original f','n=3','n=11','n=100')

4. DTFT of Square Signal

fo=1;
T=1/fo;
f=-10:0.1:10;
y=(exp(-j*f*T/2).*{sin(pi*f*T/2)})./(pi*f);
plot(f,abs(y))
xlabel 'Frequency (Hz)'
ylabel '|F(w)|'

5. Bilateral FFT of Square Signal

clear;

fo=1;
fs=50;

xi=0:1/fs:(1/fo)-(1/fs);
yiv=0.5+ 0.5*square(2*pi*fo*xi);
stem(xi,yiv,'.')
ylim([-0.2 1.3])
xlim([0 1/fo])
xlabel 'Time (s)'
ylabel 'Amplitude (V)'

FTv=fft(yiv);
powerV=FTv/length(xi);
powerV(2:end)=powerV(2:end);
Frecuencia=[0:length(FTv)-1]*{fs/length(FTv)};
abspowerV=abs(powerV);
6. Amplitude mismatch between DTFT and FFT

clear; % clear all variables

fs=100;
fo=1;
xi=0:1/fs:(1/fo)-(1/fs);
yiv=0.5+ 0.5*square(2*pi*fo*xi);

ZP=pow2(nextpow2(length(xi))+1);
FTv=fft(yiv,ZP);
powerV=FTv/length(xi);
Frecuencia=[0:length(FTv)-1]*(fs/length(FTv));
abspowerV=abs(powerV);

fo=1;
T=1/fo;
f=-10:1/fs:20;
y=(exp(-j*f*T/2).*(sin(pi*f*T/2)))./(pi*f);
figure

hStem=stem(Frecuencia(1:52),abspowerV(1:52),'r.')
hold on
hLine=plot(f(1001:end),abs(y(1001:end)))
hold off
h=[hStem;hLine];
xlabel('Frequency (Hz)')
ylabel 'Amplitude (V)'
legend('FFT','DTFT')

7. Error between Fourier Series and FFT amplitudes of V and I

In case we want to compare the amplitude of the Fourier series of V and I to the amplitude of V and I computed with the FFT for the Matlab case, TopSpice or experimental case this script should be added right after the following part is finished:

%%% 3. INPUT IMPEDANCE  %%%

%%% 3.1 Serie de Fourier V&I  %%%

zz=2;
for z=0
    SFV(z+1)=0.5; %Fourier series of V
    ESFV(z+1)=abs((absAmplitudeVPRO(z+1)-SFV(z+1))/SFV(z+1))*100;
    % Error in Fourier Series of V
    SFI(z+1)=0; % Fourier series of I
ESFI(z+1)=abs((absAmplitudeIPRO(z+1)-SFI(z+1))/SFI(z+1))*100;
% Error in Fourier Series of I
for z=1:2:(n-1)
    SFV(zz)=(1/(z*pi))*(((1)^z-1)+1);  % Fourier series of V
    ESFV(zz)=abs((absAmplitudeVFRO(zz)-SFV(zz))/SFV(zz))*100;
% Error in Fourier Series of V
    Impedancia(zz)=abs(R1+(1/(j*2*pi*1*C1*z))); % Circuit 1 Impedance
    SFI(zz)=SFV(zz)/Impedancia(zz);
% Fourier series of I
    ESFI(zz)=abs((absAmplitudeIPRO(zz)-SFI(zz))/SFI(zz))*100;
% Error in Fourier Series of I
    zz=zz+1;
end

8. Impedance Computation of Circuit 1 with Matlab theoretical V and I Signals for Square excitation

clear; % clear all variables
clf; % clear all figures

R=1000; % Resistor
C=1e-3; % Capacitor
tau=R*C; % Tau of the Circuit 1
fo=10e-3; % Input Signal Frequency
fs=100; % Sampling Frequency

%%%% 1. V(t) & I(t) %%%

% Exactly one cycle
V=0.5+ 0.5*square(2*pi*fo*t); % Pulse signal Input Signal (V)
%V=V.*kaiser(length(V),7)'; % Windowing of V(t)

% Circuit Current (A)
I1=(exp(-ti/tau))/R;
I2=-(exp(-ti/tau))/R;
I=[I1,I2];
%I=I.*kaiser(length(I),7)'; % Windowing of I(t)

%%%% 2. FAST FOURIER TRANSFORM %%%

FTV=fft(V); % FFT Voltage
FTI=fft(I); % FFT Current
half=ceil((length(FTV)+1)/2);
FFThalfV=FTV(1:half);
FFThalfI=FTI(1:half);
AmplitudeV=FFThalfV/length(FTV); % Amplitud Normalizada en V
AmplitudeI=FFThalfI/length(FTI); % Amplitud Normalizada en A

% No esta elevado al cuadrado porque es amplitud y no potencia
% Impar length(FTV) excluye nyquist point
% Las componentes en DC y la de nyquist, si existen, son únicas y no
% deberian multiplicarse x 2
if rem(length(FTV),2) % Si length(FTV) es par, rem devuelve 0
    AmplitudeV(2:end)=AmplitudeV(2:end)*2; % Si rem devuelve 1
    AmplitudeI(2:end)=AmplitudeI(2:end)*2;
else % no multiplica por dos el punto de nyquist si es que existe
    AmplitudeV(2:end)=AmplitudeV(2:end)*2;
    AmplitudeI(2:end)=AmplitudeI(2:end)*2;
end

end
AmplitudeV(2:end-1)=AmplitudeV(2:end-1)*2; % Si rem devuelve 0
AmplitudeI(2:end-1)=AmplitudeI(2:end-1)*2;
end
Frequency=[0:half-1]*(fs/length(FTV));
absAmplitudeV=abs(AmplitudeV);
absAmplitudeI=abs(AmplitudeI);

%%% 3. INPUT IMPEDANCE %%%%

Zin=absAmplitudeV./absAmplitudeI;
ZinIMAG=imag(AmplitudeV./AmplitudeI);
ZinREAL=real(AmplitudeV./AmplitudeI);
n=length(Zin);
for z=1 % DC Value
    ZinPROIMAG(z)=ZinIMAG(z);
    ZinPROREAL(z)=ZinREAL(z);
    ZinPRO(z)=Zin(z);
    FrequencyPRO(z)=Frequency(z);
    absAmplitudeVPRO(z)=absAmplitudeV(z);
    absAmplitudeI PRO(z)=absAmplitudeI(z);
end
% PRO variables are Normalized for DC and Odd Frequencies (Square Signal case)

ZinPROdB=20*log10(ZinPRO);

Simulated Impedance
Impedancia=20*log10(abs(R+(1./(j*2*pi*C*FrequencyPRO))));

Theoretical Impedance
semilogx(FrequencyPRO,ZinPROdB,'-.',FrequencyPRO,Impedancia)
xlim([fo fs])
ylim([30 90])
xlabel ('Frequency (Hz)'
ylabel ('|Module (dB)|'
legend('Simulated','Theoretical')
%title('P1,Square,fs=1KHz,fin=10mHz')

%%% 4. ERRORS Calculation %%%

ErrorZin=(abs(Impedancia-ZinPROdB)./Impedancia);
AverageError=sum(ErrorZin(2:end))/(length(ErrorZin)-1);
AverageErrorPer=100*AverageError;
StandardDesPer=100*sqrt(sum(((ErrorZin(2:end)-
  AverageError).^2))/(length(ErrorZin)-1));
9. Impedance Computation of Circuit 1 with TopSpice data of V and I signals for Square excitation

```matlab
% clear all variables
% clear all figures

R1=1000;
C1=1000e-6;

load Top1Vsquare.txt; % Loads the txt file of the
Source Voltage from Topspice
tvv=Top1Vsquare(:,1); % Time samples of V
vv=Top1Vsquare(:,2); % Amplitude (Volts)
load Top1Isquare.txt; % Loads the txt file of the
Input Current from Topspice
tii=Top1Isquare(:,1); % Time samples of I
ii=Top1Isquare(:,2); % Amplitude (Ampers)
plot (tii,ii)

[a,b,c]=unique(tvv); % Reads the time samples
k=1; % if time samples are equal
while (length(b)>=k) % then save the first one loaded
  tv(k)=tvv(b(k)); % corresponding V and I data
  v(k)=vv(b(k)); % This must be done because
  k=k+1; % the interpolation function can't
take
end % two same values of time
[a,b,c]=unique(tii);
k=1;
while (length(b)>=k)
ti(k)=tii(b(k));
i(k)=ii(b(k));
k=k+1;
end %END

period=tv(end); % Signal period
fs=1000; % Sampling Frequency

%%% 1.INTERPOLATION of V(t) & I(t) %%%
t=v(1):1/fs:period-(1/fs); % Exactly one cycle
V=interp1(tv,v,t); % Interpolation of the Pulse
signal Input Signal (V)
I=interp1(ti,i,t); % Interpolación of the Input
Current (A)

%%% 2.FAST FOURIER TRANSFORM %%%
FTV=fft(V); % FFT Voltage
FTI=fft(I); % FFT Current
half=ceil((length(FTV)+1)/2);
FFThalfV=FTV(1:half);
FFThalfI=FTI(1:half);
AmplitudeV=FFThalfV/length(FTV); % Amplitud Normalizada en V
AmplitudeI=FFThalfI/length(FTI); % Amplitud Normalizada en A
```
Determinació de la càrrega de bateries i condensadors mintjançant mesures d’impedància

% No esta elevado al cuadrado porque es amplitud y no potencia
% Impar length(FTV) excluye nyquist point
% Las componentes en DC y la de nyquist, si existen, son únicas y no
% deberían multiplicarse x 2
if rem(length(FTV),2) % Si length(FTV) es par, rem devuelve 0
    AmplitudeV(2:end)=AmplitudeV(2:end)*2; % Si rem devuelve 1
else % no multiplica por dos el punto de nyquist si es que existe
    AmplitudeV(2:end-1)=AmplitudeV(2:end-1)*2; % Si rem devuelve 0
    AmplitudeI(2:end-1)=AmplitudeI(2:end-1)*2;
end
Frequency=[0:half-1]*(fs/length(FTV));
absAmplitudeV=abs(AmplitudeV);
absAmplitudeI=abs(AmplitudeI);

%%% 3.INPUT IMPEDANCE %%%
Zin=absAmplitudeV./absAmplitudeI;
ZinIMAG=imag(AmplitudeV./AmplitudeI);
ZinREAL=real(AmplitudeV./AmplitudeI);
n=length(Zin);
for z=1 % DC Value
    ZinPROIMAG(z)=ZinIMAG(z);
    ZinPROREAL(z)=ZinREAL(z);
    ZinPRO(z)=Zin(z);
    FrequencyPRO(z)=Frequency(z);
    absAmplitudeVPRO(z)=absAmplitudeV(z);
    absAmplitudeIPRO(z)=absAmplitudeI(z);
end
for z=2:2:n % Odd Frequencies
    ZinPROIMAG((z/2)+1)=ZinIMAG(z);
    ZinPROREAL((z/2)+1)=ZinREAL(z);
    ZinPRO((z/2)+1)=Zin(z);
    FrequencyPRO((z/2)+1)=Frequency(z);
    absAmplitudeVPRO((z/2)+1)=absAmplitudeV(z);
    absAmplitudeIPRO((z/2)+1)=absAmplitudeI(z);
end
%PRO variables are Normalized for DC and Odd Frequencies (Square Signal case)
ZinPROdB=20*log10(ZinPRO);
Simulated Impedance=20*log10(abs(R1+(1./(j*2*pi*C1*FrequencyPRO))));
Theoretical Impedance
semilogx(FrequencyPRO,ZinPROdB,’-.’,FrequencyPRO,Impedancia)
xlim([fo fs])
ylim([min(Impedancia)-5] (max(Impedancia)+5))
xlabel 'Frecuency (Hz)'
ylabel '|Module (dB)|'
legend('Simulated','Theoretical')
%title('P1,Square,fs=1 KHz,fin=10 mHz')

%%% 4.ERRORS Calculation %%%
Frec=201; % Selects up to which frequency sample we want to calculate
% the AE and SD
ErrorZin=(abs(Impedancia-ZinPROdB)./Impedancia);
AverageError=sum(ErrorZin(2:Frec))/(Frec-1);
AverageErrorPer=100*AverageError;
StandardDesPer=100*sqrt(sum(((ErrorZin(2:Frec)-
AverageError).^2))/(Frec-1));

When computing Impedance with triangular excitation signal change Part ‘3’ by the following text:

%%% 3. INPUT IMPEDANCE  %%%
Zin=abs(absAmplitudeV)./abs(absAmplitudeI);
ZinIMAG=imag(absAmplitudeV./absAmplitudeI);
ZinREAL=real(absAmplitudeV./absAmplitudeI);
nn=length(Zin);
for z=1
    ZinPROIMAG(z)=ZinIMAG(z);
    ZinPROREAL(z)=ZinREAL(z);
    ZinPRO(z)=Zin(z);
    FrequencyPRO(z)=Frequency(z);
    abspowerVPRO(z)=absAmplitudeV(z);
    abspowerIPRO(z)=absAmplitudeI(z);
end
n=0;
while ((4+4*n)<=nn)
    for z=(2+4*n):(4+4*n)
        ZinPROIMAG(z-n)=ZinIMAG(z);
        ZinPROREAL(z-n)=ZinREAL(z);
        ZinPRO(z-n)=Zin(z);
        FrequencyPRO(z-n)=Frequency(z);
        abspowerVPRO(z-n)=absAmplitudeV(z);
        abspowerIPRO(z-n)=absAmplitudeI(z);
    end
    n=n+1;
end

When computing Impedance with Trapezoidal excitation signal change Part ‘3’ by the following text:

%%% 3. INPUT IMPEDANCE  %%%
Zin=abs(absAmplitudeV)./abs(absAmplitudeI);
ZinIMAG=imag(absAmplitudeV./absAmplitudeI);
ZinREAL=real(absAmplitudeV./absAmplitudeI);
nn=length(Zin);
for z=1
    ZinPROIMAG(z)=ZinIMAG(z);
    ZinPROREAL(z)=ZinREAL(z);
    ZinPRO(z)=Zin(z);
    FrequencyPRO(z)=Frequency(z);
end
n=0;
while ((5+5*n)<=nn)
    for z=(2+5*n):(5+5*n)
        ZinPROIMAG(z-n)=ZinIMAG(z);
        ZinPROREAL(z-n)=ZinREAL(z);
        ZinPRO(z-n)=Zin(z);
        FrequencyPRO(z-n)=Frequency(z);
    end
    n=n+1;
end
When computing Impedance of circuit 2 change impedance formula and components values by:

\[
R_1=100; \\
R_2=1000; \\
C_1=1000e^{-6}; \\
\text{Impedancia}=20*\log_{10}(\text{abs}(R_1+(1/C_1)./((j*2*pi*1*\text{FrequencyPRO})+(1/(C_1*R_2)))); \quad \% \text{ Theoretical Impedance}
\]

When computing Impedance of circuit 3 change impedance formula and components values by:

\[
R_1=100; \\
R_2=1000; \\
C_1=100e^{-6}; \\
R_3=5e3; \\
C_2=10*C_1; \\
\text{Impedancia}=20*\log_{10}(\text{abs}(1e2+(1/C_1)./((j*2*pi*1*\text{FrequencyPRO})+(1/(C_1*R_2)))+(1/C_2)./((j*2*pi*1*\text{FrequencyPRO})+(1/(C_2*R_3)))); \quad \% \text{Theoretical Impedance}
\]

10. Impedance Computation of Circuit 1 with Experimental data of V and I signals for Square excitation

\[
\text{clear; \quad \% \text{clear all variables}} \\
\text{clf; \quad \% \text{clear all figures}} \\
R_1=1000; \\
C_1=1000e^{-6}; \\
\text{load P1Trap1mF.txt}; \quad \% \text{Loads the txt file of V and I} \\
\text{tvv=P1Trap1mF(:,1);} \\
\text{vv=P1Trap1mF(:,2);} \\
\text{ii=P1Trap1mF(:,3)*1e^{-3};} \\
\text{load P1Impedance1mF.txt;} \quad \% \text{Loads the impedance measured with software EC-Lab} \\
\text{fre=P1Impedance1mF(:,1);} \\
\text{Re=P1Impedance1mF(:,2);} \\
\text{Im=P1Impedance1mF(:,3);} \\
\text{ZinExp=20*log_{10}((Re+\text{abs(Im))});} \\
\text{[a,b,c]=unique(tvv);} \quad \% \text{Reads the time samples} \\
k=1; \quad \% \text{if time samples are equal} \\
\text{while (length(b)>=k)} \quad \% \text{then save the first one loaded and its} \\
\text{tv(k)=tvv(b(k));} \quad \% \text{corresponding V and I data} \\
\text{v(k)=vv(b(k));} \quad \% \text{This must be done because} \\
\text{i(k)=ii(b(k));} \]
k=k+1; % the interpolation function can't take end % two same values of time

period=tv(end); % Signal period
fs=1000; % Sampling Frequency

%%% 2. FAST FOURIER TRANSFORM %%%

FTV=fft(v(1001:end)); % FFT Voltage
FTI=fft(i(1001:end)); % FFT Current
half=ceil((length(FTV)+1)/2);
FFThalfV=FTV(1:half);
FFThalfI=FTI(1:half);
AmplitudeV=FFThalfV/length(FTV); % Amplitude Normalized en V
AmplitudeI=FFThalfI/length(FTI); % Amplitude Normalized en A
% No esta elevado al cuadrado porque es amplitud y no potencia
% Impar length(FTV) excluye nyquist point
% Las componentes en DC y la de nyquist, si existen, son únicas y no % deberian multiplicarse x 2
if rem(length(FTV),2) % Si length(FTV) es par, rem devuelve 0
    AmplitudeV(2:end)=AmplitudeV(2:end)*2; % Si rem devuelve 1
    AmplitudeI(2:end)=AmplitudeI(2:end)*2;
else % no multiplica por dos el punto de nyquist si es que existe
    AmplitudeV(2:end-1)=AmplitudeV(2:end-1)*2; % Si rem devuelve 0
    AmplitudeI(2:end-1)=AmplitudeI(2:end-1)*2;
end
Frequency=[0:half-1]*(fs/length(FTV));
absAmplitudeV=abs(AmplitudeV);
absAmplitudeI=abs(AmplitudeI);

%%% 3. INPUT IMPEDANCE %%%

Zin=absAmplitudeV./absAmplitudeI;
ZinIMAG=imag(AmplitudeV./AmplitudeI);
ZinREAL=real(AmplitudeV./AmplitudeI);
n=length(Zin);
for z=1 % DC Value
    ZinPROIMAG(z)=ZinIMAG(z);
    ZinPROREAL(z)=ZinREAL(z);
    ZinPRO(z)=Zin(z);
    FrequencyPRO(z)=Frequency(z);
    absAmplitudeVPRO(z)=absAmplitudeV(z);
    absAmplitudeIPRO(z)=absAmplitudeI(z);
for z=2:2:n % Odd Frequencies
    ZinPROIMAG((z/2)+1)=ZinIMAG(z);
    ZinPROREAL((z/2)+1)=ZinREAL(z);
    ZinPRO((z/2)+1)=Zin(z);
    FrequencyPRO((z/2)+1)=Frequency(z);
    absAmplitudeVPRO((z/2)+1)=absAmplitudeV(z);
    absAmplitudeIPRO((z/2)+1)=absAmplitudeI(z);
end

%PRO variables are Normalized for DC and Odd Frequencies (Square Signal
%case)
Determinació de la càrrega de bateries i condensadors mitjançant mesures d’impedància

\[ Z_{\text{inPROdB}} = 20 \log_{10}(Z_{\text{inPRO}}) \]  

% Simulated Impedance
\[ \text{Impedancia} = 20 \log_{10}(\text{abs}(R1 + (1/(j*2*\pi*C1*FrequencyPRO)))) \]  

% Theoretical Impedance
\[ \text{semilogx}([\text{FrequencyPRO}, Z_{\text{inPROdB}}]', 'r'), \]
\[ \% \text{semilogx}([\text{FrequencyPRO}, \text{Impedancia}], \text{grid on}) \]
\[ \% \text{xlim}([1/\text{period} \ fs]) \]
\[ \% \text{ylim}([\text{min(Impedancia)} - 5 \ (\text{max(Impedancia)} + 5)]) \]
\[ \% \text{xlabel} \ '\text{Frecuency (Hz)' \}
\[ \% \text{ylabel} \ '|\text{Module (dB)}\}' \]
\[ \% \text{legend(}'\text{Experimental}', '\text{Theoretical}', '\text{Spectroscopy}') \]

%%%% 4. ERRORS Calculation %%%

Frec=201;  % Selects up to which frequency sample we want to calculate the AE and SD
ErrorZin=(abs(Impedancia-ZinPROdB)./Impedancia);
AverageError=sum(ErrorZin(2:Frec))/(Frec-1);
AverageErrorPer=100*AverageError;
StandardDesPer=100*sqrt(sum((ErrorZin(2:Frec)-AverageError).^2))/(Frec-1));

Same changes must be done as in annex A.9 when using Triangular and Trapezoidal excitations and when computing impedances of circuit 2 and 3.

11. Error between Fourier Transform and FFT

clear;  % clear all variables
clf;    % clear all figures

R=1000;  % Resistance
C=1e-3;  % Capacitance
\[ \tau = R \cdot C; \]  % Tau of the Circuit 1
fo=1;    % Input Signal Frequency
fs=10000;  % Sampling Frequency

T=1/fo;
f=fo:2:(fs/2)-1;
y=2*abs((exp(-j*f*T/2).*\((\sin(\pi*f*T/2))\)/(\pi*f)));

%%% 1. V(t) & I(t) %%%

t=0:1/fs:(1/fo)-(1/fs);  % Exactly one cycle
V=0.5+ 0.5*\text{square}(2*pi*fo*t);  % Square Wave Input Signal (V)

\[ \text{ti} = 0:1/\text{fs}:(1/\text{fo})/2+(4/\text{fs}); \]
\[ \% \text{Circuit Current (A)} \]
\[ \text{I1} = (\exp(-\text{ti}/\tau))/R; \]
\[ \% \text{Circuit Current (A)} \]
\[ \text{I2} = -\exp(-\text{ti}/\tau)/R; \]
\[ \% \text{Circuit Current (A)} \]
\[ \text{I} = [\text{I1, I2}]; \]
%% 2.FAST FOURIER TRANSFORM  

% FFT Voltage
FTV=fft(V);
% FFT Current
FTI=fft(I);

half=ceil((length(FTV)+1)/2);
FFThalfV=FTV(1:half);
FFThalfI=FTI(1:half);

AmplitudeV=FFThalfV/length(FTV);  % Amplitud Normalizada en V
AmplitudeI=FFThalfI/length(FTI);  % Amplitud Normalizada en A

if rem(length(FTV),2)  % Si length(FTV) es par, rem devuelve 0
    AmplitudeV(2:end)=AmplitudeV(2:end)*2;  % Si rem devuelve 1
    AmplitudeI(2:end)=AmplitudeI(2:end)*2;
else  % no multiplica por dos el punto de nyquist si es que existe
    AmplitudeV(2:end-1)=AmplitudeV(2:end-1)*2;  % Si rem devuelve 0
    AmplitudeI(2:end-1)=AmplitudeI(2:end-1)*2;
end

Frequency=[0:half-1]*(fs/length(FTV));
absAmplitudeV=abs(AmplitudeV);
absAmplitudeI=abs(AmplitudeI);

%%% 3.INPUT IMPEDANCE  

Zin=absAmplitudeV./absAmplitudeI;
ZinIMAG=imag(AmplitudeV./AmplitudeI);
ZinREAL=real(AmplitudeV./AmplitudeI);
n=length(Zin);

for z=1  % DC Value
    ZinPROIMAG(z)=ZinIMAG(z);
    ZinPROREAL(z)=ZinREAL(z);
    FrequencyPRO(z)=Frequency(z);
    absAmplitudeVPRO(z)=absAmplitudeV(z);
    absAmplitudeIPRO(z)=absAmplitudeI(z);
end

% PRO variables are Normalized for DC and Odd Frequencies (Square Signal case)

%%% 3.1 Serie de Fourier V&I  

zz=1;
while (zz<length(absAmplitudeVPRO))
    ESFV(zz)=abs((absAmplitudeVPRO(zz+1)-y(zz))/y(zz));
    zz=zz+1;
end

semilogx(FrequencyPRO(2:end),ESFV,'.');
xlabel 'Frecuency (Hz)'
ylabel '|Error (%)|'
ANNEX B. TOPSPICE SIMULATION

On the next figures are shown step by step how to extract and save in a txt file the voltage and current data from TopSpice.

Step 1

First of all, we design the circuit (in this case circuit 1) with the corresponding voltage source which is a pulse signal of 1 second period, 50% duty cycle with 1 Volt of high amplitude and 0 volts as low amplitude.

Fig. B.1 Circuit 1 and Voltage source design
Step 2

In order to see the transient graphics of both signal, we chose the conditions of stop time and step ceiling, in this case we will only see one period so we choose stop time as 1 second in addition step ceiling lets us sample the signal as this time interval, for example 1 millisecond although the program detects when there are fastest transitions and this automatically decreases the sampling interval thus gathering more information and saving more samples in the txt file when this happens.

![Fig. B.2 Selecting the graphical characteristics](image-url)
Step 3

In the images below we see the graphical representation of the source voltage signal plus the corresponding current signal of the circuit. As we can see the current waveform has some fastest transitions at the start time and the half of time.

![Graphical representation of voltage and current signals](image)

**Fig. B.3** Representation of Voltage and Current Signals
Step 4

Below it is shown the steps used to save the graph data of the signals chosen in a txt file.

![Fig. B.4 Saving the Data in a txt File](image)

Each of the signals is saved in a different txt file, one for voltage and the other for current as depicted in Fig. B.5.

![Fig. B.5 Txt files where the signals are saved](image)
If we open the txt files in each is saved the samples of time in the first column and in the second column the samples of the volts for the first file (P1Vsn1H) and amperes for the second file (P1Isn1H).