École Polytechnique Fédérale de Lausanne

Bachelor Semester Project

# Numerical simulation of power systems for real-time simulation applications

*Author:*
Marc Mitjans

*Professor:*
Prof. Mario Paolone

*Supervisor:*
Reza Razzaghi

*A report submitted in fulfilment of the requirements for the degree in Engineering in Industrial Technologies*

*in the*

Escola Tècnica Superior d'Enginyeria Industrial de Barcelona, UPC

July 2014

ETSEIB

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

# *Abstract*

Distributed Electrical Systems Laboratory

Engineering in Industrial Technologies

**Numerical simulation of power systems for real-time simulation applications**

by Marc MITJANS

This report concerns to the study of FPGA-based electromagnetic transient simulations of power systems. Along its content, the discretization of electric circuits concerning power systems will be treated for RLC circuits and transmission lines. Our goal is to be able to define a program that, by reading a text file with the information of a certain network, computes the nodal admittance matrix for the Fixed Admittance Matrix Nodal Method.

Several examples will be shown with the comparison between the values obtained from the EMTP-RV simulator software and the ones obtained from our solver in order to validate the model.

Finally, the nodal admittance matrix (NAM) will be used in the code programmed for the CompactRIO and its FPGA for real-time simulations.

# Contents

# List of Figures

# Chapter 1

# Introduction

Electromagnetic transient simulations in power systems are widely used in many power applications. The so-called real-time simulators allow an exact imitation on a smaller scale of the transient and steady-state behavior of the real power system, which will allow to directly control and protect the system. As it is explained in [5] and [6], despite the advantages of the usage of real-time simulators in convential processors, it has also certain drawbacks that have to be taken into account:

1. The real-time simulation time step is lower-bounded due to the limited power of the simulation hardware.
2. The degree of complexity of real power system might be too high to be simulated with exact precision, and thus the accuracy of the results can be affected.

In order to overcome to a great extent the mentioned problems, a new hardware simulator has been introduced, the Field Programmable Gate Array (FPGA). Unlike general processors, the FPGA architecture allows parallel execution of instructions, overcoming the limitation that resides in the general processing units (series execution of information). This great advantage, among others, will allow a considerable decrease in the minimum simulation time step to orders of one microsecond or even of hundreds of nanoseconds.

The study presented in this report will be focused on FPGA-based real-time simulation. The main objective is to develop a code in LabVIEW that, by reading a file which contains all the information regarding a certain electric network, is able to compute the nodal admittance matrix for the Fixed Admittance Matrix Nodal Method, to be able to use it in a FPGA hardware for real-time simulations.

Chapter 2

# Numerical Solvers and Discretized Models

For the study of the behavior of a certain electric network, not only its design has to be specified, but it is also needed to define a proper method to compute the numerical solution for the network. In order to do so, there are two numerical solvers that accomplish with that purpose: the *state-space representation* and the *Modified Nodal Analysis (MNA)*.

The first approach is based on solving directly the several differential equations that govern the behavior of the network. This method can be described by the following equation:

$$\dot{x} = A_k x + B_k u \tag{2.1}$$

Where $x$ is the vector of states, $u$ the vector of control inputs, and $A_k$ and $B_k$ the matrices that set their relation. Even though this method gives very exact results, the formation of both matrices and the following resolution can be quite computationally expensive. The *Modified Nodal Analysis* approach, however, overcomes some of these difficulties. This method can be represented as follows:

$$[A][x_n] = [b_n] \tag{2.2}$$

Where $[A]$ corresponds to the fixed matrix of a discretized model of the network elements, called the *Nodal Admittance Matrix (NAM)*, and $[b_n]$ to the current history terms related to each element and the independent sources. This solver computes for every time step the new values for the unknown vector $[x_n]$, which may correspond to the node voltages and some of the currents in the circuit, and updates at every iteration the history terms in $[b_n]$.

Regarding the advantages and disadvantages of both integration methods, in this study we will focus on the MNA.

## 2.1 Integration Methods

In order to discretize the differential equations for the network elements, among all the techniques proposed in [2] the two methods in [8] stand out: the Backward Euler and the Trapezoidal techniques are described in this section.

Let the purple line in *Figure 2.1* be the real curve of a certain node voltage of a circuit between times $(t - \triangle t)$ and $t$, where $\triangle t$ corresponds to the time step used in the simulation:

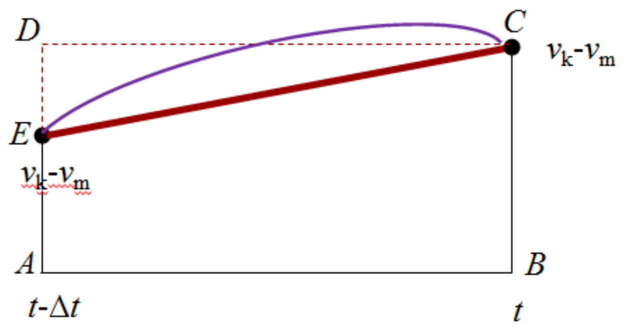


Figure 2.1: Backward Euler vs Trapezoidal.

The Trapezoidal integration method corresponds to the computation of the area inside the quadrilateral $ABCE$. This is represented by following equation:

$$\int_{t-\triangle t}^{t} [v_k(t) - v_m(t)]dt = \frac{\triangle t}{2}\{[v_k(t) - v_m(t)] + [v_k(t-\triangle t) - v_m(t-\triangle t)]\} \tag{2.3}$$

On the other hand, the Backward Euler method computes the area inside the $ABCD$ quadrilateral:

$$\int_{t-\triangle t}^{t} [v_k(t) - v_m(t)]dt = \triangle t \cdot [v_k(t) - v_m(t)] \tag{2.4}$$

As it can be seen from *Figure 2.1*, the Trapezoidal method usually has a better resemblance to the exact integral than the Backward Euler, but the computation is also more costly than the second one ((2.3) *versus* (2.4)). Also, the trapezoidal method applied for switches can produce an undesired underdamping. For these reasons, the discrete models of the lumped elements will be based on the Backward Euler method.

## 2.2 Discrete models for lumped elements

In this section, the discretization of the several lumped elements that are considered will be treated, in order to both go from continuous to discrete time and to linearize the voltage-current relation. At first, the resistor, the inductor, the capacitor and voltage sources will be considered. Then, the model used for discretizing an ideal switch will be described. Finally, the last part will be dedicated to modelling single and multiconductor transmission lines.

### 2.2.1 RLC models

For an RLC model, recall the relation between the voltage drop across the terminals of the element and the current going through it:

1. Resistor: $v_R = i_R R$
2. Inductor: $v_I = L\dfrac{di_I}{dt}$
3. Capacitor: $i_C = C\dfrac{dv_C}{dt}$

These relations, however, correspond to a continuous time model. Thus, in [8] the discretized models of an RLC element can be represented by an equivalent resistor $R_{eq}$ and a current source $I_{hist}$ in parallel, as depicted in *Figure 2.2*.

The equivalent resistance and the history term for the independent current source vary for every one of the three elements. Their corresponding values are presented in *Table 2.1*, where $\triangle V_C$ corresponds to $[v_k(t-\triangle t) - v_m(t-\triangle t)]$, and $i_L^n$ to the current through the inductor at the previous state.

In order to model the voltage sources, only the voltage drop across its terminals will be taken into account. Thus, a direct source and an alternating voltage source will be modeled similarly.

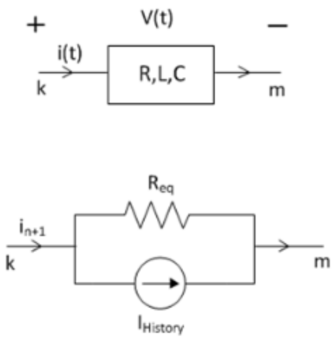


FIGURE 2.2: Discretized model of an RLC element.

| Element | $R_{eq}$ | $I_hist$ |
|---|---|---|
| Resistor | $R$ | - |
| Inductor | $\frac{L}{\triangle t}$ | $i_L^n$ |
| Capacitor | $\frac{\triangle t}{C}$ | $-\frac{C}{\triangle t} \cdot (\triangle V_C)$ |

TABLE 2.1: Backward Euler models for RLC elements.

### 2.2.2 Switch model

Even though the ideal switch representation is often used to easily describe the behavior of a switch element, this model would require $2^n$ different nodal admittance matrices (where $n$ is the number of switches that are present in the network), one for each combination of *on/off* states. In this study, the approach to model real switches proposed by Dr. Predrag Pejovic, from the University of Belgrad, will be applied. This technique will allow us to model the switches with just one single and fixed matrix. As it is described in [3], a real switch can be approximated by the circuit in *Figure 2.3*:

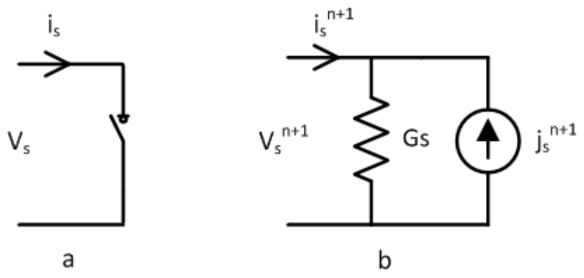


FIGURE 2.3: Discrete-time representation of a real switch.

This representation can be obtained by adding, in parallel, an ideal switch and an inductance (in series) with an ideal switch and a capacitor (also in series). Then, the value of the equivalent resistance $G_s$ corresponds to both $\dfrac{C_s}{\triangle t}$ and $\dfrac{\triangle t}{L_s}$. The value of the independent source $J_s^{n+1}$ is defined as follows:

$$\begin{aligned} J_s^{n+1} &= -i_s^n && \text{for the 'on' state} \\ J_s^{n+1} &= G_s V_s^n && \text{for the 'off' state} \end{aligned} \tag{2.5}$$

Where $-i_s^n$ corresponds to the current going through the switch across the negative terminal at the previous state $(t - \triangle t)$; and $V_s{}^n$, to the voltage drop across its terminals.

This model will allow us to fix the nodal admittance matrix during the whole simulation. Otherwise, one matrix for every single combination of all switches would be needed, and this cannot be done in FPGA.

### 2.2.3 Transmission lines - Bergeron model

Concerning the case of single and multiconductor transmission lines, [1] and [4] propose a very simple and commonly used method to model a good representation of a constant transmission line, the so-called Bergeron model.

#### 2.2.3.1 Single transmission lines

As it is described in [4], the losses along the transmission line can be considered by splitting the line in four parts and adding three lumped resistors, two at the terminals and one at the center. This model can be simplified by adding half of the middle resistance to each one of the terminals. Then, the equivalent Bergeron model of a single transmission line is the one depicted at *Figure 2.4*. This model works under the assumption that $Z_C >> R/4$.

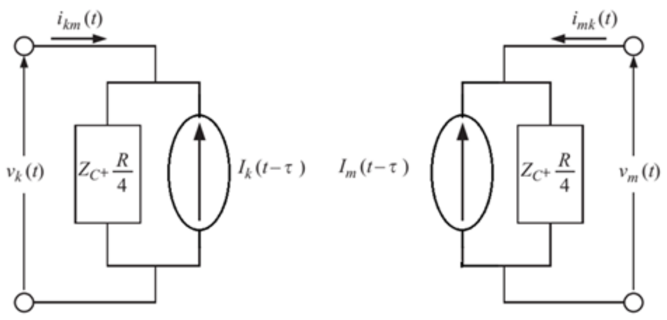


FIGURE 2.4: Equivalent two-port network for line with lumped losses.

The equations governing the history term in the independent current sources are the following:

$$\begin{aligned} I'_k(t-\tau) = & \frac{Z_C}{(Z_C+R/4)^2}[V_m(t-\tau)+(Z_C-R/4)I_{mk}(t-\tau)]+ \\ & +\frac{R/4}{(Z_C+R/4)^2}[V_k(t-\tau)+(Z_C-R/4)I_km(t-\tau)] \end{aligned} \tag{2.6}$$

$$\begin{aligned} I'_m(t-\tau) = & \frac{Z_C}{(Z_C+R/4)^2}[V_k(t-\tau)+(Z_C-R/4)I_{km}(t-\tau)]+ \\ & +\frac{R/4}{(Z_C+R/4)^2}[V_m(t-\tau)+(Z_C-R/4)I_mk(t-\tau)] \end{aligned} \tag{2.7}$$

Where $R$ corresponds to the resistance modelling the total loss along the line, $tau$ to the travelling time and $Z_C$ to the characteristic impedance of the line. $I_{km}$ and $I_{mk}$ are the currents entering the positive and negative terminals of the line respectively.

#### 2.2.3.2 Multiconductor transmission lines

In the case of multiconductor transmission lines, where multiple lines are coupled among them, an additional step must be taken in order to decouple them an proceed with the approach mentioned in the above section for every independent line.

The equations governing the behavior of the wave propagations of voltages and currents are described as follows:

$$-\left[\frac{dv_p}{dx}\right] = [Z'_p][i_p] \tag{2.8}$$

$$-\left[\frac{di_p}{dx}\right] = [Y'_p][v_p] \tag{2.9}$$

Where $v_p$ and $i_p$ are the phase voltage and current vectors, and $[Z'_p]$ and $[Y'_p]$ the longitudinal impedances and transvere addmitances respectively.

These equations can be transformed using the eigenvalues of the matrices to obtain the final relations that will be used throughout the rest of the report:

$$[v_p] = [T_v][v_m] \tag{2.10}$$

$$[i_p] = [T_i][i_m] \tag{2.11}$$

Where $[T_v]$ and $[T_i]$ are the so-called *transformation matrices*, and $v_m$ and $i_m$ correspond to the voltages and currents at the terminals of the transmission lines (modal values), while $v_p$ and $i_p$ again correspond to the phase voltages and currents. This method allows us to decouple a multiconductor transmission line, and linearly and independently obtain the values for all the variables used to model transmission line.

# Chapter 3

# The Modified Nodal Analysis LabVIEW Solver

Taking into account all the models mentioned in Chapter 2, a network solver has been created by means of the software LabVIEW which, in rough outlines, computes the nodal admittance matrix (NAM) for a given electric circuit, the vector of history terms $[b_n]$ described at the beginning of Chapter 2 and solves the equation $[x_n] = [A]^{-1}[b_n]$ for every iteration, allowing us to obtain the transient and steady-state values for all the unknown variables.

In order to run correctly the program, the *netlist* file of the circuit must be provided. This file can be obtained as an output file from the EMTP-RV software.

This chapter will cover all the details concerning the code. Regarding the section for the nodal admittance matrix itself, the first part will mainly be referred to RLC circuits with switches, and the second part will include the single and multiconductor transmission lines.

The main code, named *Simulation*, needs several funtions (VIs in the LabVIEW language) to be able to run properly. The main structure of the algorithm is the following: The *Data* VI is in charge of gathering all the needed information for the construction of the main code. Then, another VI named *Values* gathers the output of the *Data* and computes all the needed values for the construction of the NAM. Finally, this matrix is sent to every iteration in the *Simulation* VI, where the right-hand side equation is continuously computed. *Figure 3.1* shows a schematic of the actual structure of the main program.

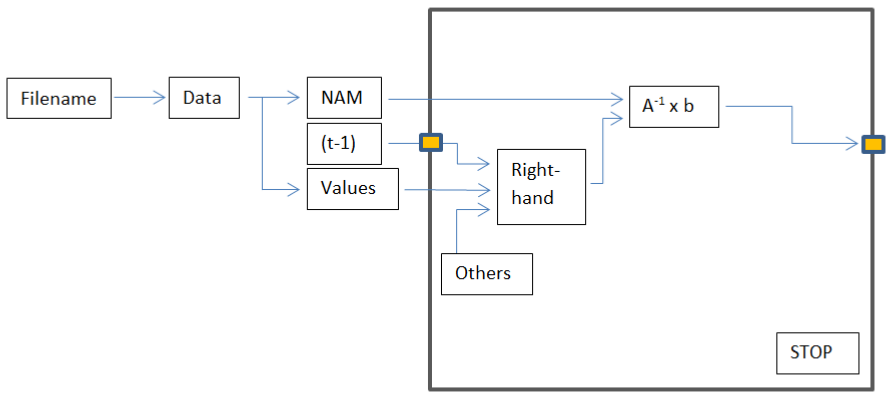


FIGURE 3.1: All the information about the circuit is obtained from the *.net* file, and is processed to calculate the unknown values of the variables at every iteration.

## 3.1 The gathering of data

When a certain circuit is design in the EMTP-RV environment, a *.net* file is created with the name of the EMTP-RV design. This file contains all the information about the circuit. For every element, two lines are written: The first one contains the information about the type of element and the node pins between which it is connected; the second one contains the specific value of the element in question (for R,L or C).

The *Data* VI was built to read that *.net* file and arrange all the needed data in a two-dimensional array, so that all this information can be easily accessed any time it is needed. It is important to mention that certain specifications have to be met, as the code reads one line at a time and compares the characters with some predefined ones. Then, the following assumptions have to be made:

1. The nodes have to be named with the letter $V$, followed by the number of the node $i$, $\forall i = 1...N$, where $N$ is the number of nodes to be considered.

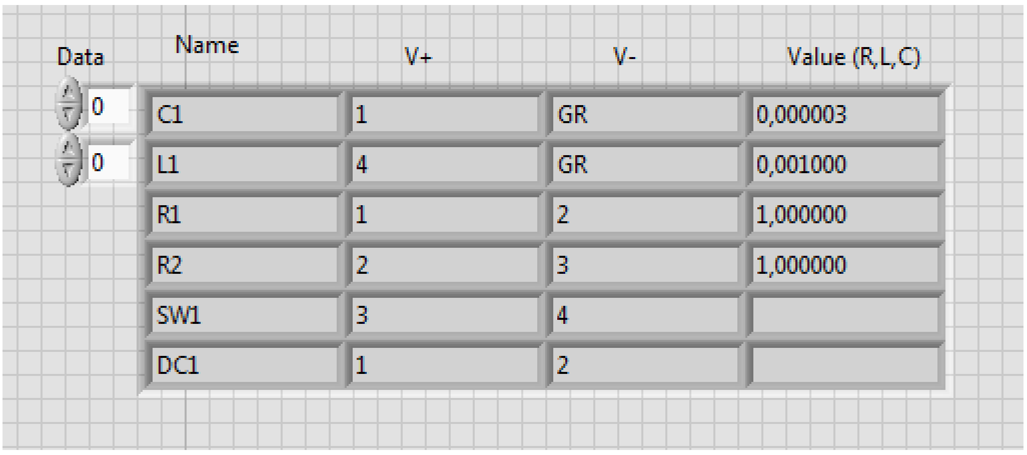
2. The resistors, inductors, capacitors, switches and voltage sources have to be named with the letter which indicates the type of the element (R, L, C, SW, DC or AC) followed by its corresponding number $i$, defined in the same way as in the previous point.

3. The numbers for each element must have a consecutive order *inside* their element type (e.g. R1, R2, L1, C1, C2, C3).

RLC elements are identified by the keyword *_RLC* at the beginning of their corresponding line. This indicates the VI that the following line will have to be properly scanned. Then, it obtains the name of the element and the names of the nodes and stores them in a 1D array. Next, the code jumps to the following line, and reads and stores the value of the element at the end of that 1D array. Finally, this array is appended to a previously created 2D array, which contains all the elements that had already been checked.

Switches and DC and AC sources are differentiated by the keywords *_SW0*, *_Vp* and *_Vsine* respectively. The procedure to obtain the data is the same as for RLC elements, but in these cases no special value will be read (as the several $G_s$ and the values for the voltage sources will be set manually).

Finally, the columns of the final 2D array that outputs the *Data* VI correspond to the following information: Name, $V^+$, $V^-$ and Value. An example is depicted in the *Figure* 3.2. This VI also outputs other 2D arrays of information, but that are formed from this general array.



| Name | V+ | V- | Value (R,L,C) |
|---|---|---|---|
| C1 | 1 | GR | 0,000003 |
| L1 | 4 | GR | 0,001000 |
| R1 | 1 | 2 | 1,000000 |
| R2 | 2 | 3 | 1,000000 |
| SW1 | 3 | 4 | |
| DC1 | 1 | 2 | |

FIGURE 3.2: Among other information, the *Data* VI outputs a 2D array of the information concerning all the elements in the circuit.

## 3.2 The nodal admittance matrix for RLC circuits

### 3.2.1 Values

One of the destinations of the main 2D array that is output from *Data* is the *Values* VI. This function uses the information of all the elements to compute some values that will be needed in the construction of the *NAM*, so that they have to be computed just once, gaining some computational speed. Some of these values might be the number of nodes of the circuit, the number of additional currents that are defined or even an array containing all the information about the elements with an additional current associated.

This list of values will not only be used for the construction of the NAM, but also in the main code *Simulation*.

### 3.2.2 The NAM

The building of this square matrix is mainly the center of the Solver. It is a $n \times n$ matrix, where $n$ corresponds to the number of unknown variables that will be computed at each iteration.

The total number of variables will be composed by all the node voltages and some additional currents that have to be defined. Every one of this currents will be associated only to an inductor, switch or voltage source (either DC or AC), and will be set as entering to the positive terminal of the element. This matrix is computed generally based on the KCL equations (and KVL for voltage sources), and can be devided in three main parts. In order to build the matrix, each one of them has to follow its respective rule:

1. The top-left part: It corresponds to the portion that covers the rows and columns corresponding to the unknown potential of the nodes. For the elements in the diagonal, each one is equal to the sum of conductances (just for resistors and capacitors) connected to that node. The off diagonal elements correspond to the negative conductances of the elements connected between nodes $i$ and $j$, which are the nodes corresponding to the row and column respectively.
2. The top-right part: This part covers the last group of columns (corresponding to the additional currents), but the same rows as before. In this case, for every current (column), a $+1$ number has to be placed in the row that corresponds to the voltage from which that current is *"leaving"*, and a $-1$ to the voltage that it is arriving to.
3. The bottom part: This one covers all the columns, and the rows referred to the additional currents. For this part no special rules are defined, and it has to be obtained by applying KCL to every one of the corresponding elements. It has to be taken into account that the $I_{history}$ term will always be in the right-hand side equation, as it corresponds to the independent current source.

In order to build the NAM, the code accesses every element of the matrix (filled with zeros at the beginning) and applies the criteria defined above. Then, at every position it gets access to all the elements in the *Data* array to perform the requiered operations for each element.

For an RLC circuit, this function VI outputs the global nodal admittance matrix and a vector with the names of the unknown variables (voltages and currents).

### 3.2.3 The right-hand side of the equation

Once the NAM is computed, it is sent to the main loop that computes the values for the unknown variables for every iteration. In order to do so, the right-hand side of the equation must be built from all the history terms (these terms can be obtained using the so-called shift register in LabVIEW). Together with these values, the *Right Hand* function needs also the following information: The *Data* array from the *Values* VI, an array referring to the additional currents, an array corresponding to the $G_s$ values for every switch (in the right order), the number of nodes and currents, a list with the values for the voltage sources, and finally a boolean list which indicates which switches are *on* and *off* at the current state. This last boolean list is previously computed in a for loop. It needs as input a list with the closing and opening times for every switch (the switches are initially open).

The *Right Hand* VI checks for every element in the column vector whether a capacitor is connected to that voltage or whether that row corresponds to an inductor, a switch or a source, and performs the requiered computations to obtain the history term for the element in question. In the case of switches, $J_s^{n+1}$ will be chosen accordingly to its state (on/off). For the sources, their value will be obtained from the input array *Sources*.

Once the product $[A]^{-1} \cdot [b_n]$ is computed to obtain $[x_n]$, the array is sent to the next iteration by means of the shift register, and the value that is desired to observe is appended to another array, that later will be used by a *Waveform Graph* to display the evolution of that variable through time.

## 3.3 The nodal admittance matrix for transmission lines

The integration of transmission lines in the circuit varies considerably the structure of both the main program and how the nodal admittance matrix is obtained. Despite this, the theory behind RLC ciruits from the above section still holds. However, in these cases the matrix will be increased with new additional elements. The structure of the code will also change, as depicts *Figure 3.3*.

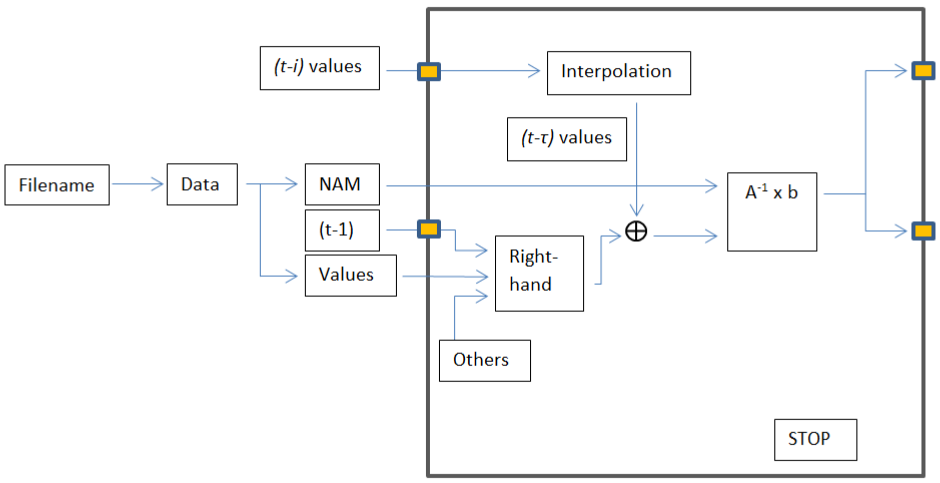


FIGURE 3.3: New blocks that compute the interpolated values have to be added inside the main loop.

### 3.3.1 The NAM

As it has been mentioned in Chapter 2, every decoupled transmission line is split in two parts, one corresponding to each terminal. The code is built in such a way that it considers every part of every decoupled transmission line as independent from the rest. This method allows arranging information more easily. Each transmission line involves 8 variables to be taken into account: four voltages and four currents. However, from these four voltages, two of them correspond directly to already considered nodes of the circuit (phase voltages). That means that for every transmission line, six new variables will have to be included in the matrix.

Horizontally displayed, the exact order of the variables in the matrix will be the following (voltages above and currents below):

$$\{V_1, .., V_M, V^d_{k1}, V^d_{m1}, .., V^d_{kN_1}, V^d_{mN_1}, V^c_{k1}, V^c_{m1}, .., V^c_{kN_2}, V^c_{mN_2}, \cdots$$

$$\cdots, I_L, I_{SW}, I_{DC}, I_{AC}, I_{p1}, .., I_{p(N_1+N_2)}, I^d_{k1}, I^d_{m1}, .., I^d_{kN_1}, I^d_{mN_1}, I^c_{k1}, I^c_{m1}, .., I^c_{kN_2}, I^c_{mN_2}\}$$

In the notation used, $M$ corresponds to the amount of nodes considered, $N_1$ corresponds to the number of decoupled transmission lines (with the $d$ superscript) and $N_2$ to the total number of coupled transmission lines (with the $c$ superscript). The additional currents $I_p$ are the phase currents, defined as entering to the phase voltages. Inside the matrix, the rows corresponding to them will define the relationship in (2.10). On the other hand, the $I_k$ and $I_m$ currents will define the relationship in (2.11). Recall that, for already decoupled transmission lines, the transformation matrices in (2.10) and (2.11) correspond to the identity matrix.

All the new voltages that have been added to the top-left part of the matrix will include in their diagonal elements the value of $1/Z_C$, and zero everywhere else.

### 3.3.2 The right-hand side equation

With the addition of transmission lines, not only the admittance matrix suffered important modifications, but the definition of the right-hand side equation has also to be updated. While all the other terms will remain the same, in in the rows that go from $V^d_{k1}$ to $V^c_{mN_2}$ the corresponding history terms will be added. Nevertheless, the computation of these terms has several subtleties with respect to the other elements.

#### 3.3.2.1 Interpolation method

If we recall the history terms for the transmission lines seen in Chapter 2, they require a specific value obtained $(t - \tau)$ steps before. Also, the time step used in the simulation will hardly ever be a divisor of the propagation time of the wave. In order to overcome this (*a priori*) setback, an interpolation between the values at the instants $(t - P)$ and $(t - P - 1)$ will have to be computed. The interpolation factors, $a_0$ and $a_1$, are defined as follows:

$$
\begin{aligned}
&\frac{\tau}{\triangle t} = P + \frac{\epsilon_1}{\triangle t} \\
&\epsilon_2 = \triangle t - \epsilon_1 \\
&a_0 = \frac{\epsilon_2}{\triangle t};\ a_1 = \frac{\epsilon_1}{\triangle t}
\end{aligned}
\tag{3.1}
$$

And so, the value of $I'(t - \tau)$ will be obtained from $a_0 \cdot I'(t - P) + a_1 \cdot I'(t - P - 1)$. In order to speed up the computation time, these calculations have been converted to matrix multiplications. For that purpose, a matrix $[K]$ has been created from the coefficients in (2.6) and (2.7) for every transmission line, and $a_0$ and $a_1$ have been converted to diagonal matrices, covering all the interpolation values for all the decoupled matrices together. Then, the resulting values at $(t - \tau)$ will be calculated as follows:

$$\vec{I}'(t - \tau) = [a_0][K]\vec{I}(t - P) + [a_1][K]\vec{I}(t - P - 1) \tag{3.2}$$

Where the vector $\vec{I}(t-n)$ corresponds to the values from $I_{k1}^{d}$ to $I_{mN_2}^{c}$ at time $(t-n)$.

The values of $\vec{I}(t-P)$ for every decoupled transmission line, however, have to be stored for the future access. This problem can be solved by creating a so-called *circular buffer*. It is a 2D array, empty at the beginning, and at every iteration the values of the desired currents, $\vec{I}(t)$, get stored at the column with index 0, and the column corresponding to the values of $(t - P_{max} - 1)$ is deleted, where $P_{max}$ corresponds to the maximum value of $P$ among all the transmission lines. Then, in order to compute the interpolation, the desired values $\vec{I}(t-P)$ and $\vec{I}(t-P-1)$ can be easily obtained.

Finally, the vector of $\vec{I}'(t-\tau)$ is inserted to the right-hand part that was previously computed for RLC cases (in the columns corresponding to the $V_k$, $V_m$ variables), and the computation of the variables is carried through by solving, again, $[A]^{-1}[b_n]$.

# Chapter 4

# Simulation Results

In order to test and validate the program described in Chapter 2, several simulation examples have been executed, two for RLC circuits and one for coupled transmission lines. In this chapter, the opbtained results will be compared by means of MATLAB to the ones computed by EMTP-RV. For all the examples, a simulation time step of $1\mu s$ has been considered. It is important to also set the integration method as Backward Euler in the EMTP-RV environment, to get a better resemblance between the results.

It is important to mention that, as the code hasn't been optimally simplified, the execution time for the simulation is greater than the time spent by EMTP-RV. This difference of time is also increased in those networks where transmission lines are included.

## 4.1 Example 1: A three phase inverter

The circuit depicted in *Figure 4.1* shows the schematic of this example. Every pair of switches is synchronized to avoid floating points, so that they open and close alternately. In between the ground and the nodes between the switches, three inductors in series with three resistors have been connected.

From the schematic in *Figure 4.1* it is easy to determine the number of variables, which will be the sum of the seven voltages and the ten currents. Once the code is executed, the *Data* array that is output is the one in *Figure 4.2*:

Where $GR$ refers to the ground reference, and the values in the fourth column are expressed in their natural units ($\Omega$, $H$ and $F$). Then, the NAM that is obtained is depicted in *Figure 4.3*.

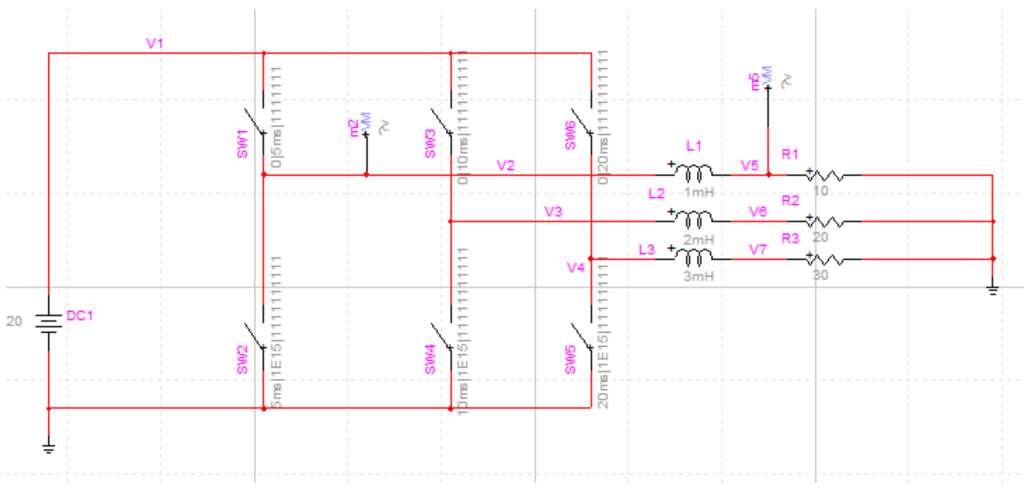
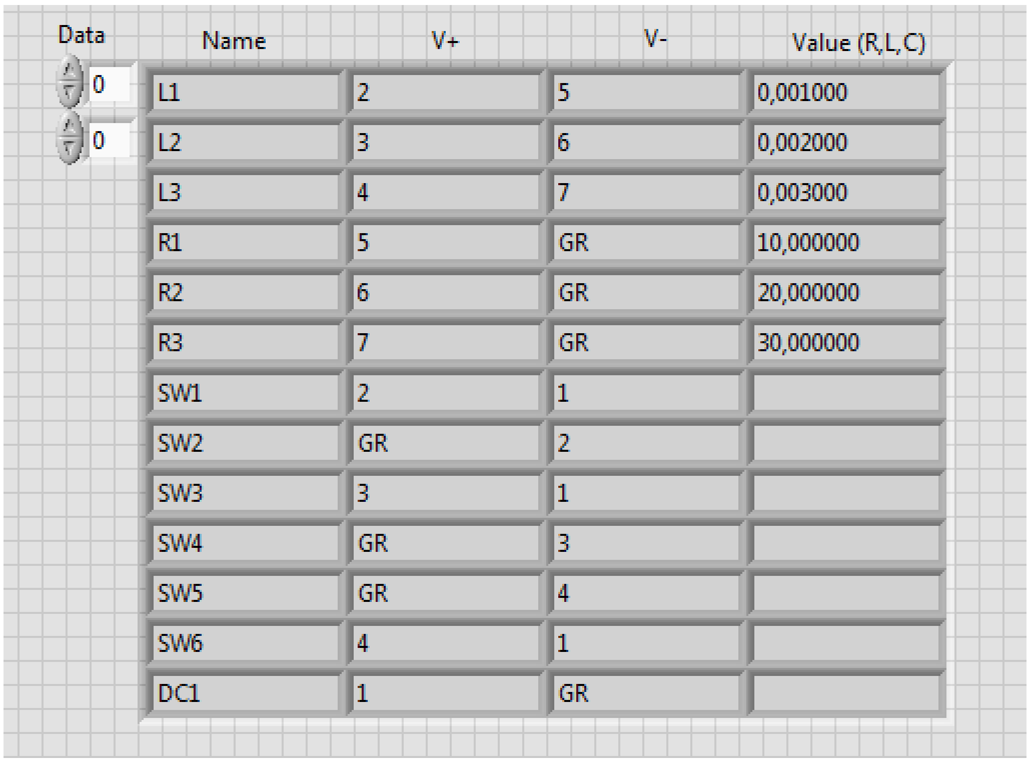


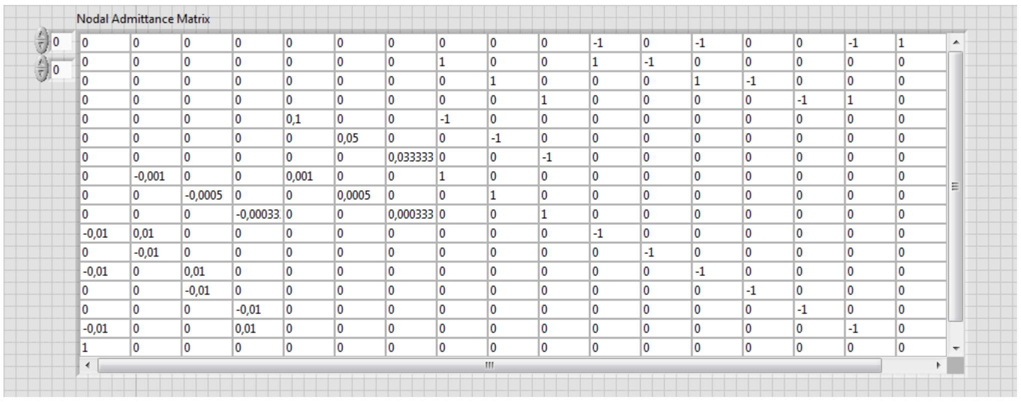
FIGURE 4.1: The schematic of a 3 phase inverter. 10 additional currents have to be defined, one for each switch, inductor and DC source.



Data
0
0

| Name | V+ | V- | Value (R,L,C) |
|---|---|---|---|
| L1 | 2 | 5 | 0,001000 |
| L2 | 3 | 6 | 0,002000 |
| L3 | 4 | 7 | 0,003000 |
| R1 | 5 | GR | 10,000000 |
| R2 | 6 | GR | 20,000000 |
| R3 | 7 | GR | 30,000000 |
| SW1 | 2 | 1 | |
| SW2 | GR | 2 | |
| SW3 | 3 | 1 | |
| SW4 | GR | 3 | |
| SW5 | GR | 4 | |
| SW6 | 4 | 1 | |
| DC1 | 1 | GR | |

FIGURE 4.2: The output Data for the 3 phase inverter example obtained from the *.net* file.


Nodal Admittance Matrix

0
0

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | -1 | 0 | 0 | -1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | -1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | -1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | -1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0,1 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0,05 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0,033333 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | -0,001 | 0 | 0 | 0,001 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | -0,0005 | 0 | 0 | 0,0005 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | -0,00033 | 0 | 0 | 0,000333 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| -0,01 | 0,01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | -0,01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 |
| -0,01 | 0 | 0,01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 |
| 0 | 0 | -0,01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 |
| 0 | 0 | 0 | -0,01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 |
| -0,01 | 0 | 0 | 0,01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

FIGURE 4.3: The nodal admittance matrix is computed from the 2D array of information obtained from the *Data* function.

Until the seventh row and column, the resistors' and capacitors' conductances are introduced following the criteria mentioned in Chapter 3. Thus, if we take a look at *Figure 4.1* we can see that indeed only $V_5$, $V_6$ and $V_7$ are connected to a resistor (their conductances are in the diagonal of the top-left part). For the top-right part, the currents entering and leaving the nodes are represented with $-1$ and $+1$ respectively (e.g. for $SW3$, its current is leaving $V_3$ and entering $V_1$, which corresponds to the eleventh column). Finally, regarding the KCL equations at the bottom part, the ones for the inductors, switches and sources are, respectively, of the form:

$$i_{hist} = \frac{\triangle t}{L}[v_L^- - v_L^+] + i_L \tag{4.1}$$

$$i_{hist} = G_s[v_s^+ - v_s^-] - i_s = J_s^{n+1} \tag{4.2}$$

$$E_{AC/DC} = v^+ - v^- \tag{4.3}$$

If we pick, for instance, $L1$, we can see that a 1 is assigned to the diagonal, which represents the current $i_L$ and the terms of $\triangle t/L$ $(= 0,01)$ are placed, with the respective positive and negative signs, at the columns corresponding to the voltages $V_5$ and $V_2$.

As an example, the evolution of the voltage $V_2$ will be compared. After running the simulation with a simulation time $t_{max}$ of $32ms$, the curve that corresponds to both voltages is obtained. The comparison between the results of our code are depicted in *Figure 4.4*.

As it can be noticed, both curves don't follow exactly the same path, and even a huge negative spike appears when the switch closes. If we zoom in *Figure 4.4*, a small difference can also be appreciated between both curves.

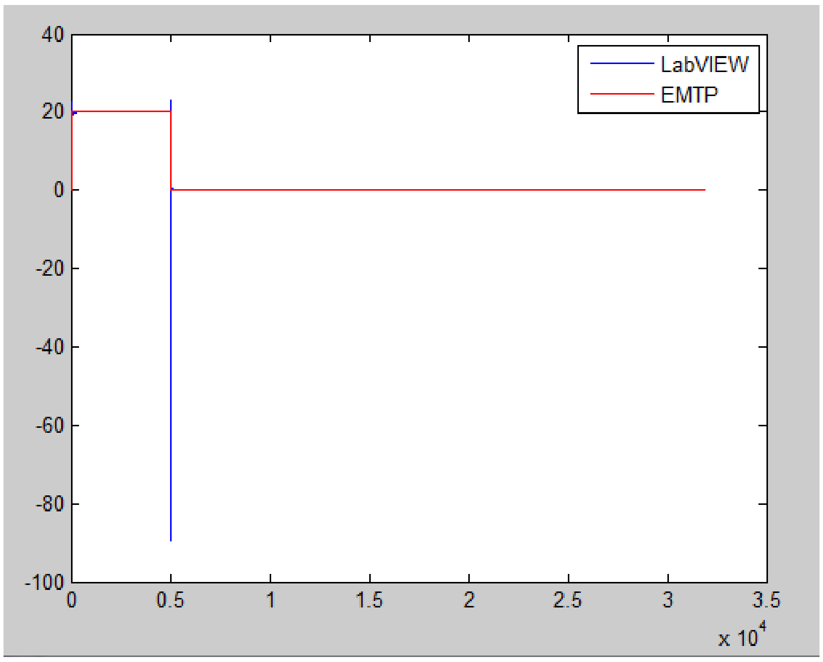


FIGURE 4.4: Evolution through time of voltage $V_2$.

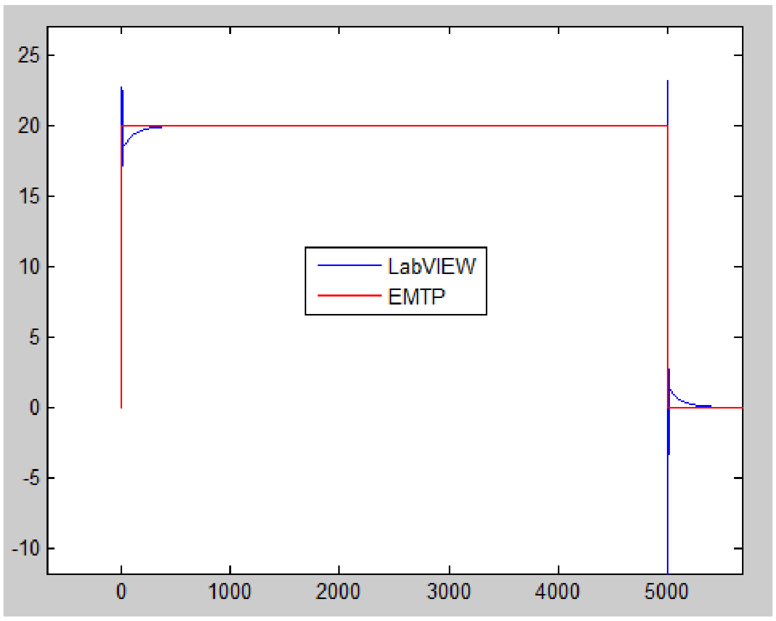


FIGURE 4.5: A zoomed in image for $V_2$.

These divergence between the two methods can be explained by means of the model of the switch. While in this study we have discretized a real switch using the Pejovic model, EMTP-RV uses ideal switches. This different methods have different effects to the transient behavior when they either close or open.

In order to remove this divergence as much as possible, another circuit has been used as example with the Pejovic model added in place of the ideal switch (see Chapter 2, Section 2.2.2).

## 4.2 Example 2: RLC circuit with Pejovic model for switches

In this case, the circuit that has been chosen contains two voltage sources, a DC source of $10V$ and an AC source with an amplitude of $230V$, a frequency of $60Hz$ and a phase shift of $0\ rad$. The circuit is shown in *Figure 4.6*.

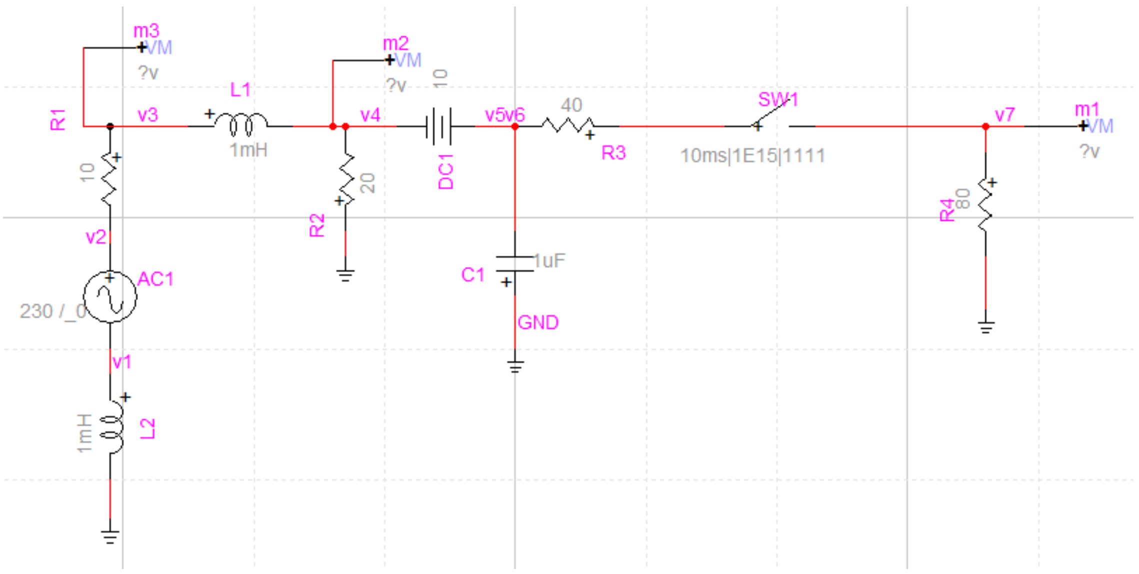


FIGURE 4.6: The schematic of the second RLC circuit used as example.

However, in this case $SW1$ has been replaced by the Pejovic model (*Figure 4.7*) in another EMTP-RV file, in order to compare both results.

Again, in this case the two graphics for $V_1$ will be compared to finally validate the code for, at least, RLC circuits. After running a simulation of $32ms$, the results are displayed in *Figure 4.8*.

As it can be seen in the zoomed in image (*Figure4.9*), the red dots (cooresponding to the curve obtained from the EMTP-RV software) and the blue dots (corresponding to the values obtained with our code) match perfectly well. Even though that to be able to

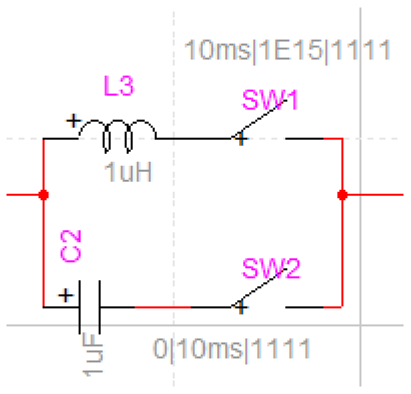


FIGURE 4.7: The Pejovic model for a switch.

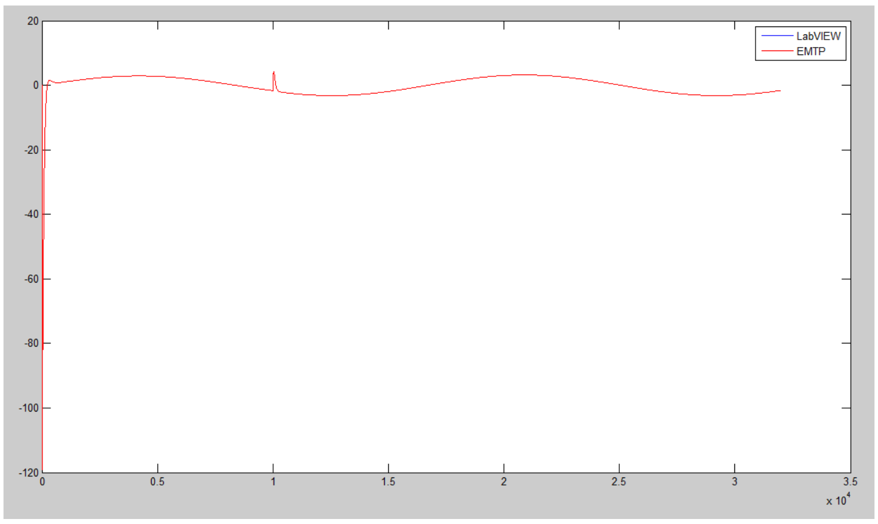


FIGURE 4.8: Voltage 1 for the second RLC circuit.

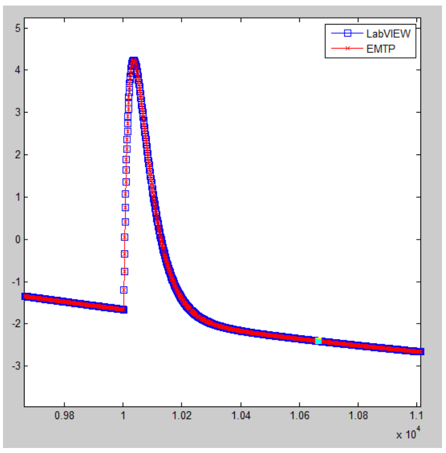


FIGURE 4.9: Voltage 1 zoomed in for the second RLC circuit.

completely certify that the code is perfectly valid for all kinds of RLC circuits it should be tested with every one of them, this verification allows us to convincely affirm that the code can be used with any other RLC circuit with satisfactory results.

## 4.3 Example 3: Transmission lines simulation

In the case where transmission lines are included in the network, it is not be possible to verify the model as it has been done in the previous section, where all the point values match between the two simulators, but instead they will just be appoximately close. This is caused by the integration method used in EMTP-RV. Even though the Backward Euler is selected, the software doesn't use the method exactly as we have defined it, and adds extra mid-points in between the values to compute a non-linear interpolation, and definitively to obtain different values. However, if the integration method selected is the Trapezoidal one, the results resemble more to the ones obtained in our program using Backward Euler. The study of this behavior will be treated in further studies.

The circuit example that has been used to test the code is composed by 10 groups of 2 coupled transmission lines each, 10 DC sources of 100000 volts, 10 resistors and 1 switch between nodes 4 and 5 with $G_s = 0,1$; all of them connected to a total number of 30 nodes. Taking into account Chapter 3, Section 2.3, this will result in a total number of 161 variables, which corresponds to a NAM of $161 \times 161$ elements. In this model, no more switches have been added due to the divergence between the Pejovic model and the ideal switch, as this divergence would be high enough to distort too much the comparison.

*Figure 4.10* depicts the schematic of the circuit. Each one of the five groups is devided into two groups of two coupled transmission lines. For this study, the voltage $V_4$ will be used to compare the results.

### 4.3.1 EMTP-RV Backward Euler

The results obtained applying Backward Euler as the integration method in both simulators can be compared in *Figure 4.11* and in *Figure 4.12*. Even though in the first one it might seem that both curves are well aligned, the zoomed in figure shows that they are relatively far from representing the same values.

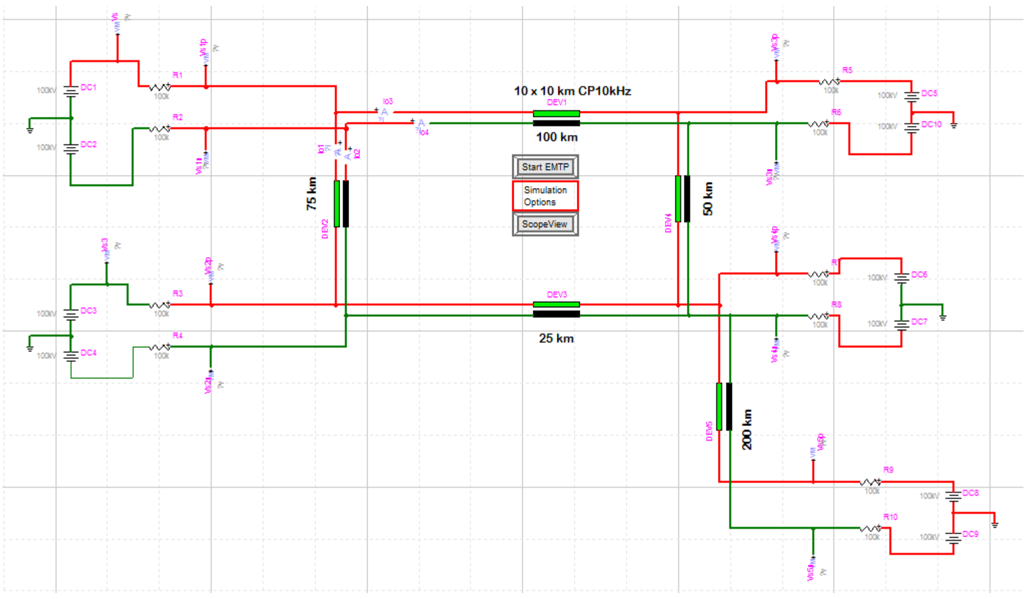


FIGURE 4.10: MTDC circuit for transmission lines.

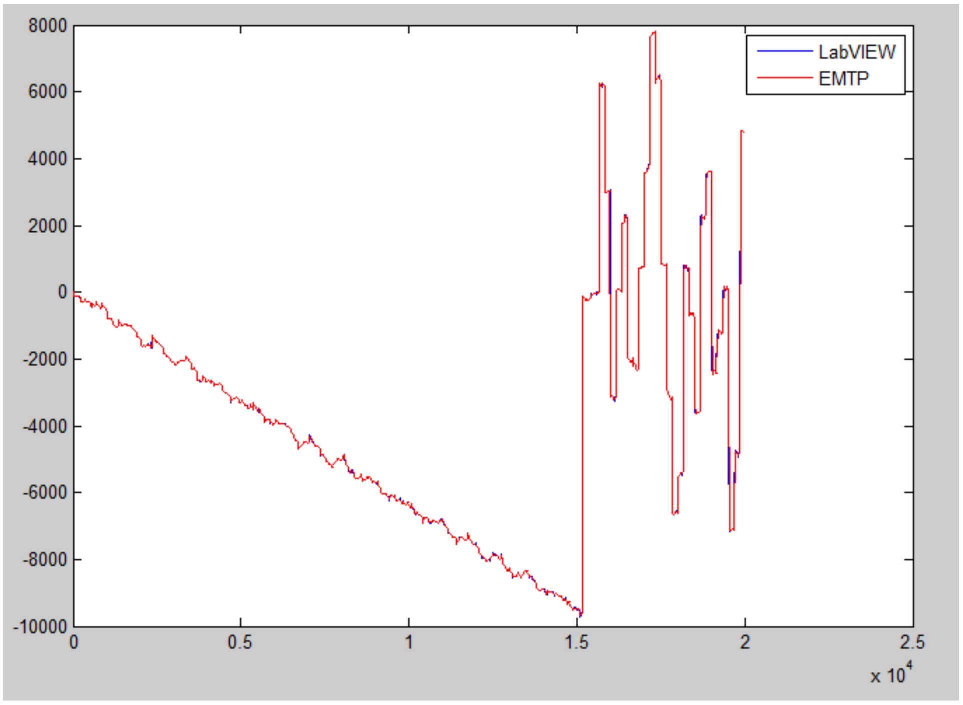


FIGURE 4.11: The transient of the voltage $V_4$ using Backward Euler in both simulators.

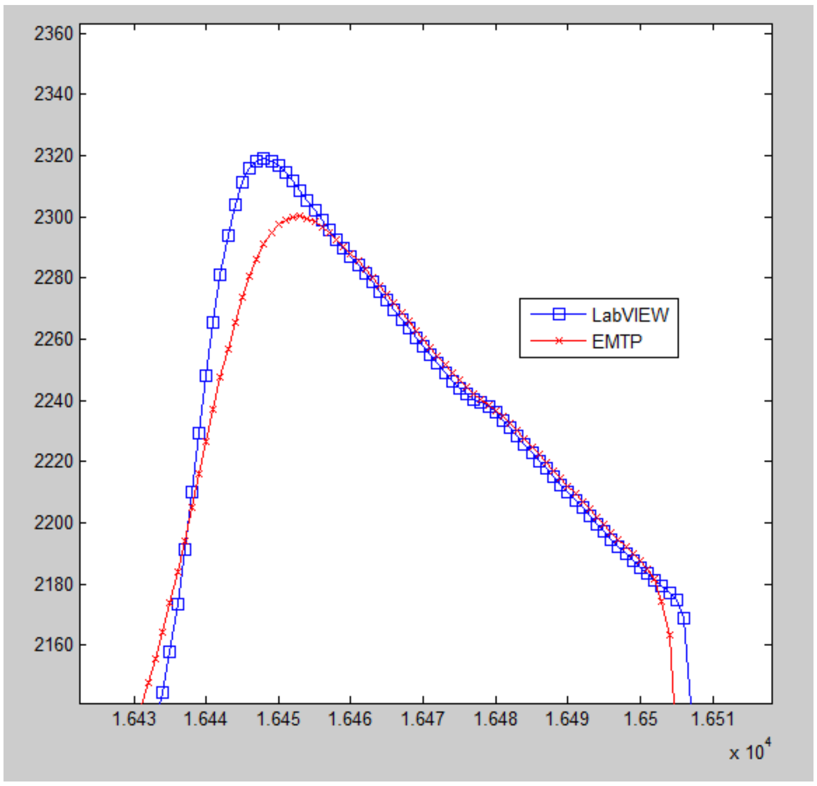


FIGURE 4.12: A *zoomed in* section of the transient of the voltage $V_4$ using Backward Euler in both simulators. it is also possible to see the double amount of values obtained.

### 4.3.2 EMTP-RV Trapezoidal

When the Trapezoidal method is selected, however, even though the values still don't match perfectly well, an enormous improvement with respect to the Backward Euler method has been obtained. Again, the comparison can be observed in both *Figure 4.13* and *Figure 4.14*. In this case, a full-length simulation of $100ms$ has been run.

In contrast with the results from the RLC circuit simulation, the ones obtained from this example cannot validate with exact certainty the simulator program with the same level of confidence due to the difference between the interpolation methods.

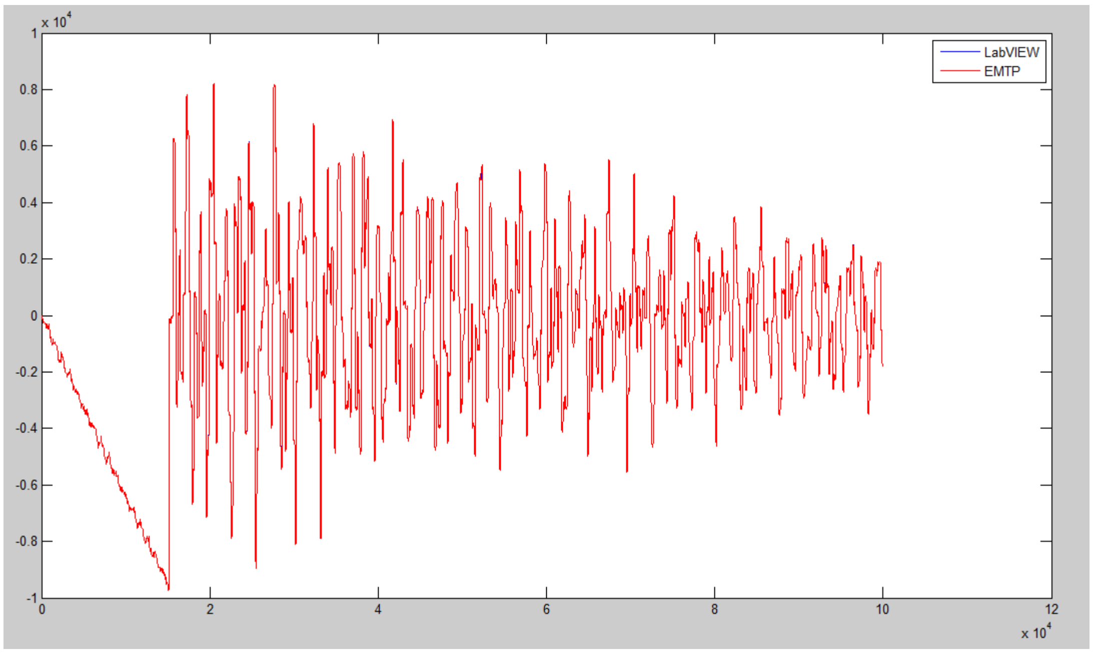


FIGURE 4.13: The transient of the voltage $V_4$ using the Trapezoidal method in the EMTP-RV software with a $100ms$ simulation.

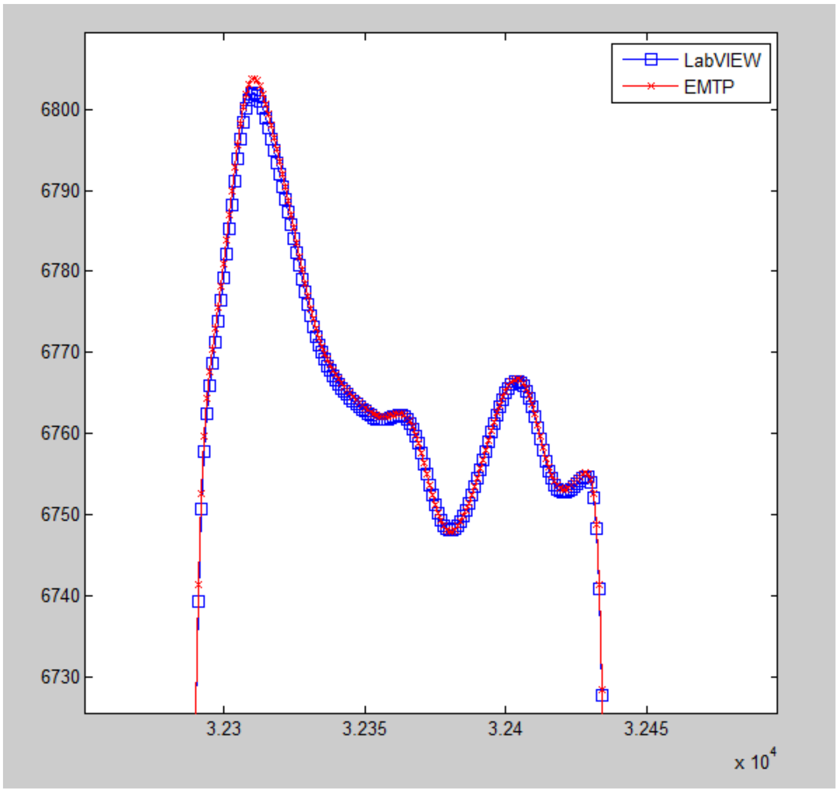


FIGURE 4.14: A *zoomed in* section of the transient of the voltage $V_4$ using the Trapezoidal method in the EMTP-RV environment.

Chapter 5

# Real-time simulation in FPGA

Once the code has been validated, it can be transferred to the FPGA target, but not before applying certain changes to the main code to make it executable. As it has been stated before, an FPGA target is a very powerful hardware that allows extremely high-speed performances in real-time by allowing parallel executions, but as a tradeoff it is very inflexible with respect to the programming code, and the amount of resources available (in the form of look-up tables (LTU's), memory blocks or DSP blocks (to perform mathematical operations)) is very limited. Thus, the new simulation code must be highly optimized in order to both speed it up and save as many resources as possible.

The FPGA module is one of the components of the controller CompactRIO, from National Instruments. It is a controller containig several pieces of hardware used for several industrial applications. For this project, the microcontroller and the FPGA module will be used. The first one will allow us to set the environment for the FPGA execution; that is, it will compute the NAM (using the code described in Chapter 3) and all the other variables that might be needed in the simulation (such as information about the elements of the circuit, gathered in the 2D array called Data), and convert and send them to the FPGA. This step can be done with the use of "Host to target - DMA FIFO", a way to store data in the microcontroller and send it to the FPGA. It follows the theory of a queue - first in first out. Then, the FPGA module will use all this information to compute the requiered values at each iteration, which will correspond to real-time values.

In order to define the code for the FPGA module, four limitations have been taken into account: First, this module doesn't allow working with strings. This directly affects the Data array, so the information regarding the name of the components must be converted to numbers. We must recall that the elements' names are defined as the letter corresponding to each element type (C, L, SW...) followed by an increasing number (1, 2, 3...). However, for the needed functions for the simulation only the element type is requiered. Thus, each element type has been converted to an integer (C - 1, L - 2,...). The next limitation is the use of arrays. The FPGA module only allows working with 1D arrays, so all the 2D arrays have to be split. This can be easily overcome with the use of clusters, which allow gathering information in just one block. Next, the representation of all numbers must be converted from double (DBL) to fixed-point (FXP). This is a direct consequence of the limited amount of resources in the FPGA, as it doesn't allow dynamic memory allocation, and everything must be pre-allocated beforehand. This also affects the size of all the arrays, as they have to be predefined and fixed as well.

In FPGA programming, a special type of loops can be used, called single-cycle timed loops. They allow computing what is inside in just 1 *tick*, which corresponds to $1/40\mu s$.

For the matrix multiplication, an externally pre-defined multiplier has been used, which gets as inputs the NAM resized in one row and the right-hand vector, and outputs the result of the multiplication.[1]

In the next two sections, the programs used for both RLC circuits and circuits containing transmission lines will be generally explained. Similarly to what was done in Chapter 4, the results obtained with the real-time simulation will be compared to the ones obtained by means of our simulator (Chapter 3).

## 5.1 Simulation code for RLC circuits

For the simulation of RLC circuits, the CompactRIO NI 9068 made by National Instruments has been used. A fixed-point configuration of 32 bits of word length and 10 bits of integer part has been defined for the inverse of the NAM, and 5 bits for the integer part for all the other values (right-hand vector and output vector (variables)). This allows a precision of $10^{-9}$, but limits the values to $\pm 16$. Also, to generalize the code the size of the arrays has been set to 30, so that any circuit with less than 30 elements per element type can be analyzed. The main difference with the offline simulation code in Chapter 3 is the computation of the right-hand vector. In order to exploit the capacity of parallel executions, the right-hand vector for each element type is formed independently, and then they are added all together. This method increases the speed of the code, but increases the amount of resources used. In the example shown in *Figure 5.1*, with all these specifications, a time step of $800ns$ (32 *ticks*) could be reached.

[1]This multiplier is not included in the files that come together with this report, as it can only be used in LabVIEW 2014.

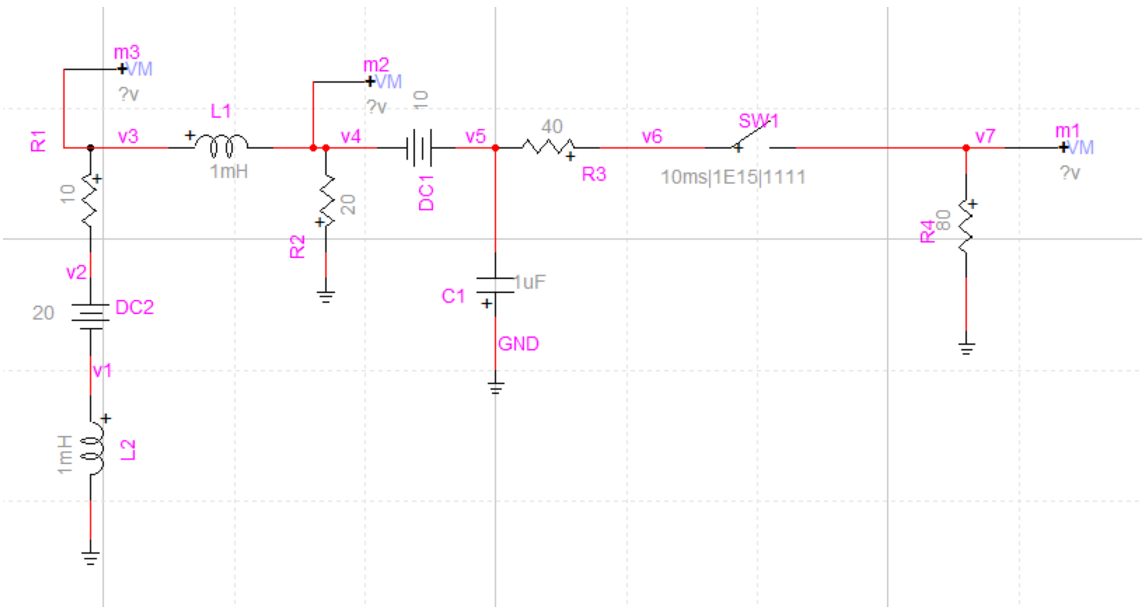


FIGURE 5.1: The RLC circuit used as an example in the CompactRIO NI 9068.

Due to the high precision chosen for the values and the lack of any other operation rather than the matrix multiplication and the formation of the right-hand side of the equation, the results obtained with both an ordinary PC and the FPGA are extremely close, as depict *Figures 5.2* and *5.3* for the voltage $V_4$.

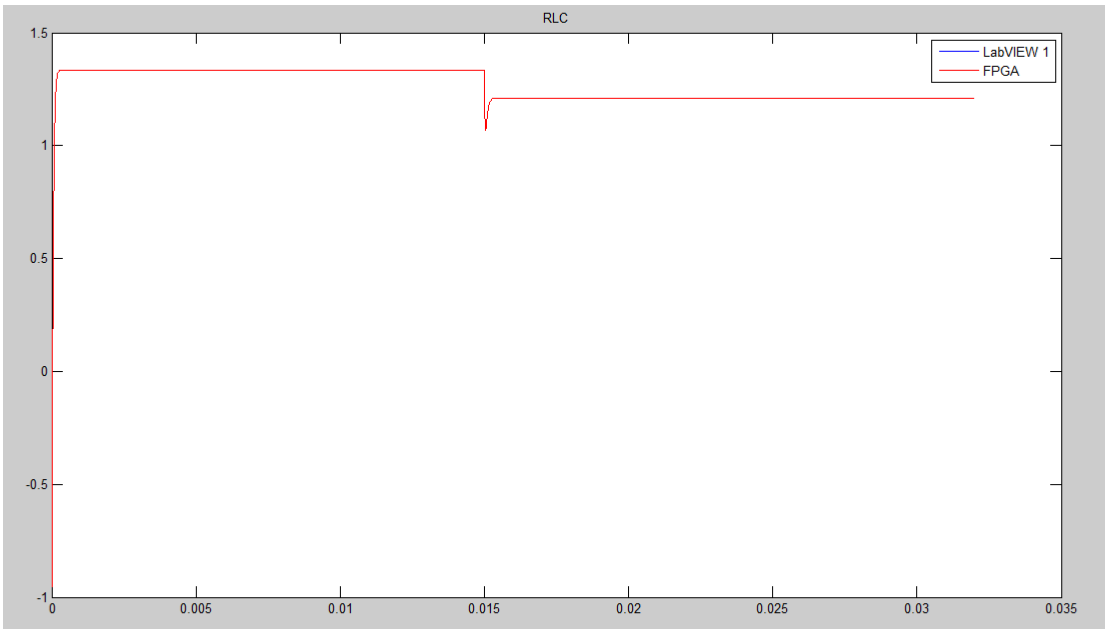


FIGURE 5.2: The waveform of $V_4$ for the RLC circuit in *Figure 5.1*.

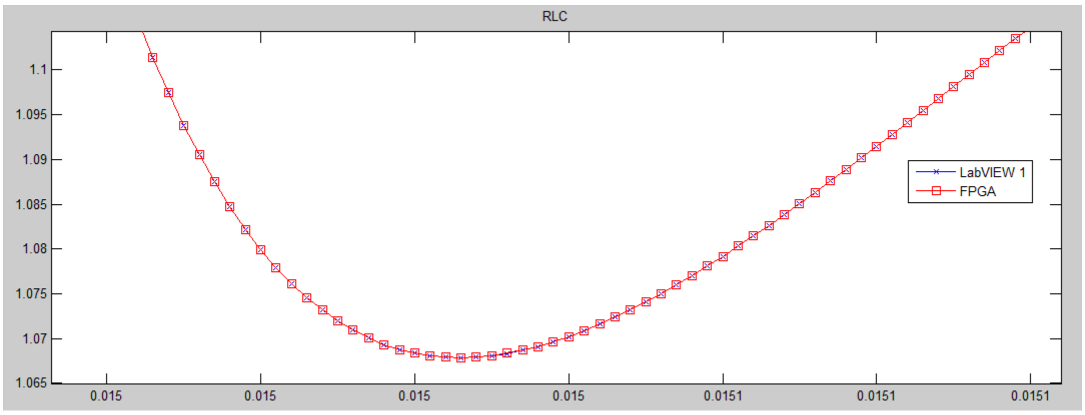


FIGURE 5.3: A zoomed in version of $V_4$.

## 5.2 Simulation code for transmission lines

The main idea for the general code which includes all the elements that have been taken into account along this study is the same as for the previous case. However, the inclusion of transmission lines drastically increases the amount of resources (mainly LTU's), together with the decrease of speed of the code, due to both the increase in the number of variables and the circular buffers together with the interpolation method to obtain the values for the right-hand vector (see Chapter 3 for more details).

At this point, the amount of information needed to build the code is too big to be processed by any existing hardware. However, the main goal for this code is to be able to simulate the behavior of a multi-terminal DC network (MTDC), introduced in rough outlines in Chapter 3, *Figure 4.10* (it is a special and rather new network configuration built specifically for transmission lines, which basically improves the power transmission). Hence, a more specific code has been defined for this type of networks. It considerably reduces the variety of circuits that can be simulated, but allows us to simulate all kinds of MTDC-like networks. The final code does not fit inside the cRIO used for RLC circuits, and a new controller had to be used in this case, named CompactRIO NI 9033. [2]

The main difference between general circuits and MTDC-like networks is the absence of AC sources, capacitors and inductors. Consequently, the right-hand vector can be formed taking into account only resistors, switches and DC sources, which reduces the amount of resources consumed in this function by half. However, this reduction of resources alone is not enough, and another simplification has to be carried out. As the right-hand vector includes lots of zeros, some columns of the $NAM^{-1}$ and rows of the $[b_n]$

[2]CompactRIO NI 9033 is a new controller that is in its beta version, and hasn't been officially released yet.

vector can be removed without changing the output of the multipication (just rescaling the size of the vector). These columns correspond to the ones referring to the node voltages, as there are no capacitors in this network (and therefore their corresponding values in the $[b_n]$ vector will always be 0), and to the ones referring to the values of $I_p$, $I_k$ and $I_m$ (recall that these rows were used to set the relationship between phase and modal variables, which corresponds to a matrix multiplication without any independent term). This shrinking will reduce the number of columns in the NAM by $n_v + 2 \cdot TL$, and the same number for the number of rows in the right-hand vector. This reduction allows shrinking the $161 \times 161$ matrix to a $161 \times 51$ one. Also, only one value for $G_s$ and one value for the DC sources are used for all of them.

Concerning the interpolation method defined in Chapter 2, as the matrix multiplication consumes a considerable amount of resources, this interpolation had to be carried out inside a *for loop*, element by element. Recall that the interpolation was of the form:

$$I'(t-\tau) = f(x_i) \cdot a_0 + f(x_{i-1}) \cdot a_1 \tag{5.1}$$

Where $f(x_i) = \vec{k} \cdot \vec{x}_i$. As this operation is time-invariant, $f(x_{i-1}) = f^{i-1}(x)$. This relationship allows us to compute just once the values of $\vec{I}'(t-P)$, and use these values to both compute $\vec{I}'(t-\tau)$ and send them to the next iteration as $\vec{I}'(t-P-1)$, which allows us to save half of the DSP blocks used in this function. These circular buffers are used in the form of “target-scoped” FIFO’s, which corresponds to the type of FIFO used only inside the FPGA module.

Finally, the last modification with respect to the RLC case is the change in the data type of the fixed-point configuration. In this case, in order to save as many resources as possible, for each different array a specific fixed-point configuration has been chosen for it, so that it matches the range and precision of their values without wasting any memory. Table 5.1 shows the specific configuration for each element.

Despite the high precision shown in the last column of Table 5.1, as happened in Chapter 4, the results between our PC simulator and the real-time simulation are not as close as the ones for the previous example. In this case, the discrepancy doesn’t come from the difference between the interpolation methods, but from the multiple operations that must be carried out along the simulation (both the general matrix multiplication and the formation of the right-hand side equation, together with the storage of the voltages and currents corresponding to the already decoupled transmission lines in the circular buffers, and the interpolation to form the values $I'(t-\tau)$ for the $[b_n]$ vector), together with the different fixed-point configurations selected for each array of variables.

| Array | Word length | Integer part | max | min | precision |
|---|---|---|---|---|---|
| $NAM^{-1}$ | 32 | 11 | 1024 | -1024 | 4,77E-7 |
| Right-hand | 32 | 8 | 128 | -128 | 5,96E-8 |
| Variables | 32 | 9 | 256 | -256 | 1,19E-7 |
| *I'(t-P)* | 26 | 4 | 8 | -8 | 2,38E-7 |
| $G_s$ values | 26 | 2 | 4 | 0 | 5,96E-8 |
| Sources | 10 | 7 | 127,875 | 0 | 0,125 |
| VCB | 23 | 9 | 256 | -256 | 6,10E-5 |
| CCB | 26 | 2 | 2 | -2 | 5,96E-8 |
| Coefficient 1 | 16 | 0 | 1 | 0 | 1,53E-5 |
| Coefficient 2 | 12 | -6 | 0,015621 | 0 | 3,81E-6 |
| Coefficient 3 | 13 | -13 | 0,000122 | 0 | 1,49E-8 |
| Coefficient 4 | 16 | -6 | 0,015625 | 0 | 2,38E-7 |
| $a_0$ | 16 | 0 | 1 | 0 | 1,53E-5 |
| $a_1$ | 16 | 0 | 1 | 0 | 1,53E-5 |

Table 5.1: Word length and integer part are expressed in bits for each of the arrays used in the simulation code. VCB and CCB correspond to both the voltage and current circular buffers respectively. The four coefficients correspond to the ones in equations 2.6 and 2.7, used in the interpolation together with $a_0$ and $a_1$. The negative values increase the precision, but limit the maximum and minimum reachable values. The elements with a minimum value of 0 have unsigned values.

With all these specifications, a time step of $5\mu s$ (200 *ticks*) could be reached. As in Chapter 4, in this report the comparison will be done with the waveforms of voltage $V_4$, depicted in *Figure 5.4*, for the general waveform (32ms), and *Figure 5.5*, for a zoomed in version. Because of the great number of variables, not all of them have been checked. However, for those that were compared in this study, a maximum relative error of $1,53\%$ was obtained, which is small enough to allow validating the results. In *Figure 5.4*, the maximum relative error is of $0,93\%$.

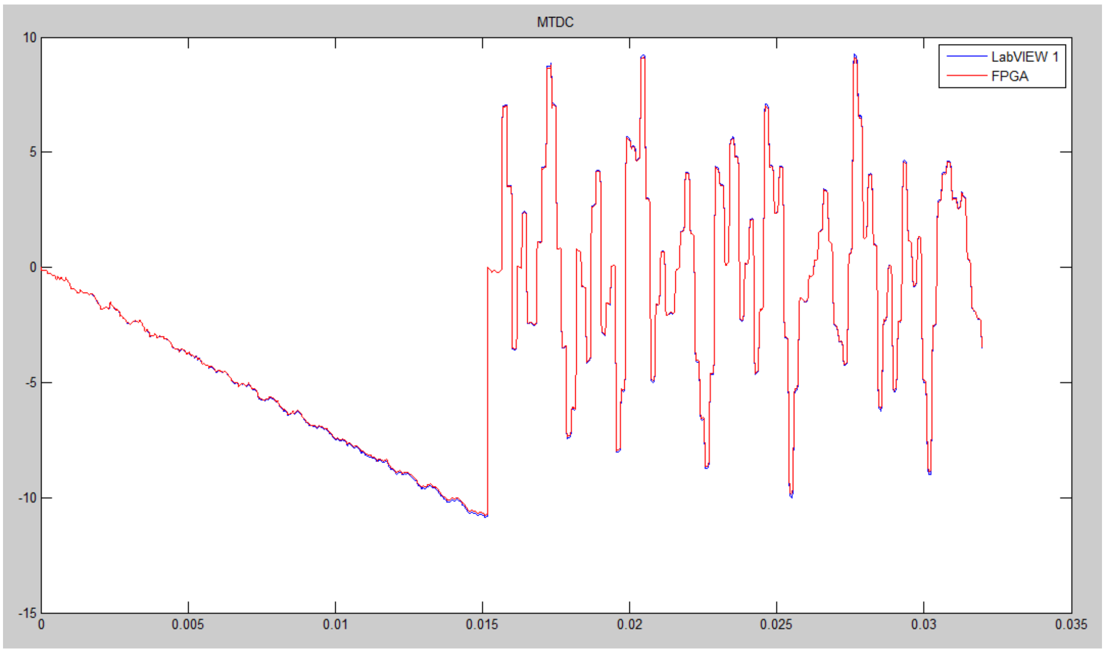


FIGURE 5.4: The waveform of $V_4$ for the MTDC circuit in *Figure 4.10*.

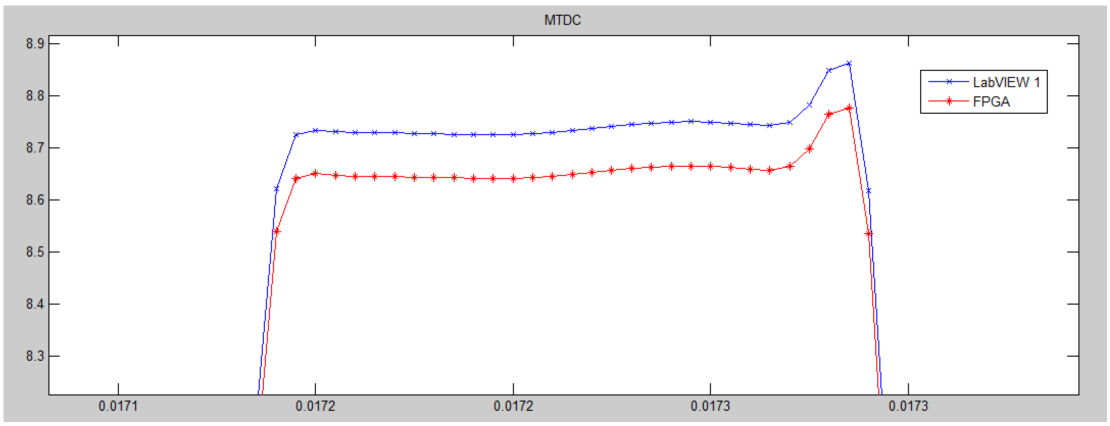


FIGURE 5.5: A zoomed in version of $V_4$.

# Chapter 6

# Conclusive remarks and future work

This study has been very useful to allow appreciating for oneself the state-of-the-art of real-time simulation of power systems and to deepen in the transient circuit analysis. More precisely, the identification of discretized RLC circuits has been carried out by means of the Backward Euler integration technique, together with modelling real switches for fast simulation of networks with integrated switches. In particular, the Pejovic model has been considered in this study in order to compute by means of LabVIEW the nodal admittance matrix for the *Fixed Admittance Matrix Nodal Method (FAMNM)* which will allow faster FPGA-based real-time simulations. Several examples have been designed in EMTP-RV software and run in both simulators (EMTP-RV and the one coded using LabVIEW). It must be said that regarding the infinite number of electric circuits that can be built, the code can never be completely validated by direct comparison. Nevertheless, the comparison for simple RLC circuits allowed us to validate the code with a great level of confidence.

Right after, RLC circuits with included coupled and decoupled transmission lines have also been taken into account. Concretely, the Bergeron method for modelling transmission lines has been used. In the case of transmission lines, even though the code for the nodal admittance matrix could not be validated with the same confidence as previously (as the values obtained from both the EMTP-RV software and the code from LabVIEW don't match as exactly as in simple RLC circuits), the results showed that very likely the Backward Euler technique used in both the code from LabVIEW and EMTP-RV are not processed equally. This conclusion extracted from that comparison will be used in future works, in order to find the exact integration technique used by EMTP-RV and analysze its validation.

After the validation of the general code, the final step of the project included the design of an FPGA-based LabVIEW code for real-time simulation based on all the previously designed programs. Concretely, time steps of 800$ns$ and 5$\mu s$ could be reached for both circuits with and without transmission lines respectively. This considerable difference between time steps is obtained due to the greater amount of operations that are needed to compute the transient of the variables for transmission lines, which increase the amount of used resources and therefore decrease the speed of the code. This new FPGA code will be used in future works for several analysis of certain power systems, which will allow us to obtain much more accurate and fast results.

Right now we are working on the publication of a scientific journal paper on power systems for the journal *IEEE Transactions on Industrial Electronics* together with the Distributed Electrical Systems Laboratory (DESL), from EPFL, where the results obtained with the FPGA code through this study will be analyzed.

# Bibliography

[1] Hermann W. Dommel. Digital computer solution of electromagnetic transients in single and multiphase networks. *IEEE Summer Power Meeting, Chicago, Ill., June 23-28, 1968*, June 1968.

[2] Charles A. Thompson. A study of numerical integration techniques for use in the companion circuit methods of transient circuit analysis. January 1992.

[3] Pedrag Pejovic and Dragan Maksimovic. A method for fast time-domain simulation of networks with switches. February 1993.

[4] Neville Watson and Jos Arrillaga. Power systems electromagnetic transients simulation. *The Institution of Engineering and Technology, London, United Kingdom*, 2003.

[5] Mahmoud Matar and Reza Iravani. Fpga implementation of power electronic converter model for real-time simulation of electromagnetic transients. *Natural Sciences and Engineering Research Council of Canada (NSERC)*, December 2009.

[6] Mario Paolone Reza Razzaghi and F. Rachidi. A general purpose fpga-based real-time simulator for power systems applications.

[7] Yuan Chen and Venkata Dinavahi. Fpga-based real-time emtp. *Natural Science and Engineering Research Council of Canada (NSERC)*, May 2008.

[8] Reza Razzaghi and Prof. Mario Paolone. Real-time simulation of electrical circuits using embedded hardware platforms.