ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

THESIS PRESENTED TO
ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF THE
MASTERS IN ELECTRICAL ENGINEERING
M.Sc.A

BY
Nicolas KAMEL

OPTIMIZATION OF VERY LOW TIME STEP FPGA-BASED SIMULATIONS USING A FIXED ADMITTANCE MATRIX APPROACH

MONTREAL, “DECEMBER 19, 2014”

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS:

Mr. Handy Fortin-Blanchette, PhD, ing., Thesis Supervisor
Département de génie électrique de l'École de technologie supérieure

Mr. Kamal Al-Haddad, PhD, ing., President of the Board of Examiners
Département de génie électrique de l'École de technologie supérieure

Mr. Olivier Tremblay, MscA, ing, External Examiner
Institut de recherche d'Hydro-Québec

Mr. Jean Bélanger, External Examiner
OPAL-RT technologies

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND THE PUBLIC

ON "DECEMBER 11, 2014"

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

ACKNOWLEDGEMENTS

First, I would like to thank Handy Fortin-Blanchette, my research supervisor, for challenging my assumptions, encouraging my ideas, and sharing with me his wisdom as a researcher. I would also like to express my gratitude to Jean Bélanger, the CEO of Opal-RT Technologies for giving me the opportunity to conduct my research in such an innovative company. I extend my thanks to every else at OPAL-RT who assisted me in my research, in particular Amine Yamane and Luc-André Grégoire. I also thank NSERC and FQRNT for their financial support. Finally, I thank my parents for their support and for instilling in me a sense of work ethic, without which this work would not have been possible.

OPTIMIZATION OF VERY LOW TIME STEP FPGA-BASED SIMULATIONS USING A FIXED ADMITTANCE MATRIX APPROACH

Nicolas KAMEL

ABSTRACT

This research is dedicated to the study of the fixed admittance matrix approach presented by Pejovic and Maksimovic (1994). More specifically, this work has three objectives. First, to study and quantify the effects of this method on simulation accuracy. Second, to use this knowledge to develop a method to tune the G_s parameter without relying on a trial and error process. An algorithm is proposed to automatically optimize the G_s parameter which has been validated on three topologies: a two level inverter, a three level NPC inverter, and a direct matrix converter. Third, to explore possible solutions that conserve the advantages of a fixed admittance matrix approach (small time step, low memory consumption) while mitigating its drawbacks (loss of simulation accuracy). A method is proposed which offers the accuracy of a variable admittance matrix approach, but with significantly less memory consumption. This method has been validated offline, however, it remains to be seen if it is a viable candidate for real time implementation and more research must be done.

Keywords: Real time simulation, fixed admittance matrix, power electronics, FPGA

OPTIMISATION D'UNE SIMULATION FPGA À PETIT PAS DE CALCUL EN UTILISANT UNE MATRICE D'ADMITTANCE FIXE

Nicolas KAMEL

RÉSUMÉ

Cette recherche est dédiée à l'étude de la méthode présentée par Pejovic and Maksimovic (1994). Plus précisément, ce travail a trois objectifs. Premièrement, d'étudier et de quantifier les effets de cette méthode sur la précision des simulations. Deuxièmement, à partir de cette information, de développer une méthode pour optimiser le paramètre G_s sans avoir recours à l'essaie et l'erreur. Un algorithme est donc proposé pour optimiser d'une manière automatique le paramètre G_s. Cette méthode a été validée sur trois topologies différentes, soient un onduleur deux niveaux, un onduleur trois niveaux NPC et un convertisseur matricielle directe. Troisièmement, d'explorer des solutions possibles qui conservent les avantages d'utiliser une matrice d'admittance fixe (petit pas de calcul, faible consommation de mémoire) tout en réduisant ses inconvénients (simulations moins précises). Une méthode est proposée qui offre la précision d'une matrice d'admittance variable, mais avec un requis de mémoire beaucoup plus faible. Cette méthode à été validée en temps différé, mais il reste à déterminer si elle peut être implémenter sur FPGA avec un pas de calcul assez petit.

Mot-clés : Simulation en temps réel, matrice d'admittance fixe, électronique de puissance, FPGA

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABREVIATIONS

CPU	Central processing unit
ETS	École de Technologie Supérieure
FPGA	Field-programmable gate array
GPU	Graphics processing unit
KCL	Kirchoff's current law
KVL	Kirchoff's voltage law
MNA	Modified nodal analysis
SPS	SimPowerSystems

INTRODUCTION

The real time simulation of power electronics is particularly challenging due to the low time steps that must be reached in order to accurately simulate high frequency phenomena. In order to achieve such time steps, computationally expensive operations such as matrix inversion must be avoided. The fixed admittance matrix method proposed by Pejovic and Maksimovic (1994) (hereby referred to simply as the "Pejovic method") eliminates these expensive operations by modeling a switch as a capacitor when it is OFF and an inductor when it is ON. Moreover, the algorithm's structure allows it to be hardware accelerated with an FPGA implementation. It is therefore a promising simulation technique for the real time simulation of converters operating at frequencies of 20kHz and higher and it has been successfully used in commercial applications. Unfortunately, this method has two main drawbacks which limit its adoption in industry. First, the value of a parameter named G_s must be chosen when performing simulations. The G_s parameter controls the value of the capacitance and inductance of the modeled switch. It therefore has a critical effect on simulation accuracy, and unfortunately its optimal value depends on a multitude of variables such as the converter topology, the nature of the load, the value of the input source, etc. Currently, this parameter must be tuned manually through trial and error which is a time consuming and sometimes inaccurate process. The second drawback of the Pejovic method is that even for an optimal value of the parameter, the simulation accuracy may not be acceptable. This work is dedicated to the study of the Pejovic method, with the goal of mitigating the drawbacks of the method and increasing its commercial potential. The first chapter of this thesis is dedicated to a literature review of the various methods used for the real time simulation of power electronics. In the second chapter, the Pejovic method is explained in detail and the effect of the G_s parameter on simulation accuracy following a commutation is studied. In the third chapter, the effect of the G_s parameter on the output load current and commutation losses is examined. In the fourth chapter, a method to automatically optimize the G_s parameter is derived and validated for three case studies. Finally, in the fifth chapter, a hybrid fixed-variable admittance matrix method is proposed. This method lessens the memory required when precomputing and storing all the possible matrix inverses of a variable admit-

tance matrix. However, the minimum time step achievable with this method on FPGA has not yet been assessed and this a potential path for future researchers.

CHAPTER 1

LITERATURE REVIEW

1.1 Fundamentals of circuit simulation

The simulation of electrical circuits is fundamental to the areas of electronic design automation, power systems, and power electronics where the behavior of circuits is tested and analyzed prior to manufacturing. (Najm, 2010, p.3) explains how this behavior can be quantified for the use of circuit simulators: "The behavior of a circuit is captured by a set of equations that are formulated by combining the element equations and Kirchoff's Current and Voltage Laws (KCL and KVL). In general, this results in a set of simultaneous non-linear first-order differential equations." The first step is to form this system of equations using a systematic method such as Modified Nodal Analysis (MNA). Then, this system of equations is discretized using a numerical integration method such as Backward-Euler at a certain time step. As Belanger *et al.* (2010) explains, this system is solved at every time step with the output states of a given time step becoming the input states of the next time step. The required length of the time step is determined by the bandwidth of the system to be simulated. If the simulator is to accurately replicate high frequency phenomena, a low time step must be chosen. This poses a problem for real-time simulation, where the time required to solve the network equations must be less than or equal to the simulation time step. Furthermore, in hardware-in-the-loop (HIL) simulations in which a physical controller is connected to the simulator, not only must be the network equations be solved at every step, but the simulated output must be sent to the controller and the controller input must be sent to the simulator. Therefore, the time it takes to solve the network equations in addition to the I/O latency between the simulator and the controller must be less than or equal to the simulation time step.

1.2 CPU and GPU based simulation

A commonly followed guideline for power electronics established by Gole *et al.* (1997) is to choose a time step equal to 1% of the commutation period. Since high frequency converters are becoming more and more common, this poses a challenge. If a converter is to be simulated with a switching frequency of 70 kHz, then a time step of about 143 ns is required. According to Blanchette *et al.* (2012), modern CPUs can only achieve time steps around 5μs. Although CPUs have high clock frequencies, their parallelism is limited, and it takes a significant amount of time for the simulated output to be sent from the CPU to the controller.

GPUs are becoming popular in the area of scientific computing due to their massively parallel architecture. Jalili-Marandi and Dinavahi (2009) used GPUs to accelerate the simulation of large power systems. They reported speed-ups of up to 344.8 over a CPU implementation. However, it is important to note that despite the speed-up, it did not reach real-time performance, and according to Lustig and Martonosi (2013) a main disadvantage of the GPU is the overhead when transferring data to and from it.

1.3 FPGA based simulation

Another massively parallel device that has become popular in the world of real time simulation is the FPGA. FPGAs have a very high degree of parallelism that allows certain types of operations such as matrix multiplication to be significantly accelerated. This principle is explained

below. Consider the following matrix-vector multiplication:

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} & A_{15} & A_{16} & A_{17} & A_{18} \\ A_{21} & A_{22} & A_{23} & A_{24} & A_{25} & A_{26} & A_{27} & A_{28} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \\ b_8 \end{bmatrix} \tag{1.1}$$

On an FPGA, the dot product between the first row of the matrix A with the vector b and the dot product between the second row of the matrix A with the vector b can be computed in parallel. The calculation of the product between the first row of the matrix A and the vector b is illustrated in figure 1.1. First, each entry of the first row of the matrix must be multiplied with the corresponding entry of the vector. On an FPGA, these eight multiplications can be done in parallel, a speed up of eight over a sequential implementation. It is important to note however, that the speed-up potential of the FPGA is limited by the fact that the results of these multiplications must be summed together. Since an addition operation can only take two inputs, multiple sequential layers of addition are required (in this example, 3), which increases the calculation time. That being said, the speed-up offered by FPGAs for matrix vector multiplication is still significant. In addition, the latency between the FPGA and the physical I/Os is very low.

Unfortunately, solving a system of equations is a more complex process than a matrix vector multiplication, and even on an FPGA, the time it requires often exceeds the allowed time step. Indeed, for a 5x5 matrix, Dohi *et al.* (2012) reported latencies of 3.05 and 8.69 microseconds for solving a system of five equations using Gauss-Jordan elimination and Cramer's rule, respectively. Mahapatra *et al.* (2012) reported performing a 4x4 matrix inversion in 290ns, a relatively small time step. However, in order the solve the system of equations, the inverse

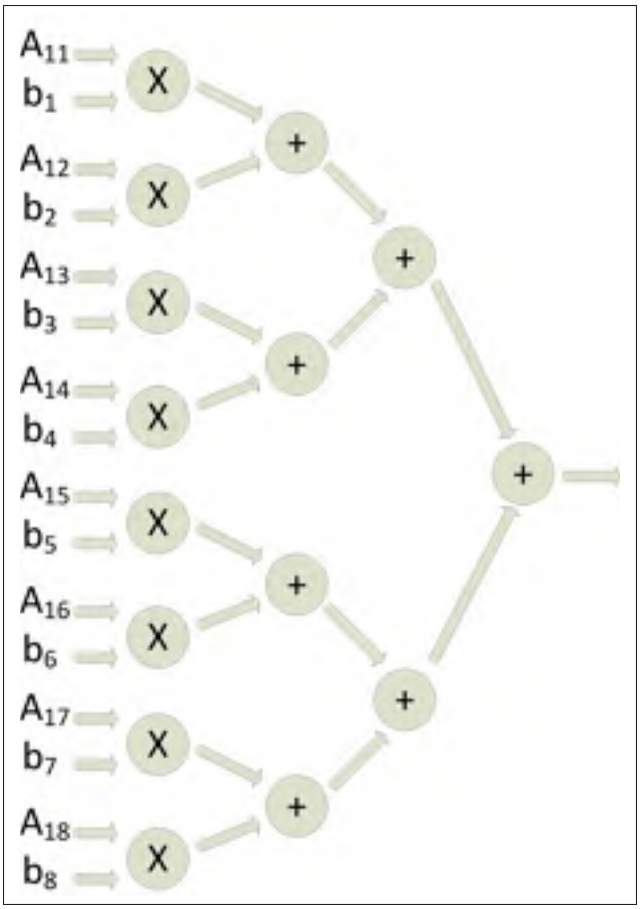

Figure 1.1 Matrix vector multiplication implemented on an FPGA

would have to be multiplied by the input vector which would increase the time step. Furthermore, 4x4 and 5x5 matrices are small by power electronics standards. For example, the system matrix of the boost converter formed by Pejovic and Maksimovic (1994) (which contains only two switches) is 7x7. Therefore, solving the system matrix at every step is not feasible for high frequency power electronics, regardless of whether a CPU, GPU, or FPGA is used.

1.4 Precomputing the inverses of a variable admittance matrix

A method to quickly solve the network equations is to precompute the inverse of the system matrix before the start of the simulation as explained by Bachir *et al.* (2010). Therefore, the outputs can be obtained at every step by simply multiplying the input vector by the inverse matrix, a task that can be accomplished very quickly on an FPGA as was previously explained.

However, the system matrix of a power converter changes every time a commutation occurs. The approach of precomputing the inverse can still be used as long as the inverses of all possible system matrices are precomputed and stored in the FPGA's memory. For a power converter with n switches, there are 2^n possible system matrices. As the number of switches increases, the amount of memory required on the FPGA grows exponentially. Blanchette *et al.* (2012) reported that this approach is only viable for converters limited to 6 or 7 switches. Although external DDR3 memory can be added to the FPGA, the time required to read from the memory is too long for real time simulation. Due to these constraints, other techniques have been proposed to simulate high frequency converters with over seven switches in real-time.

1.5 Incorporating a priori knowledge

One such method is to incorporate a priori knowledge about the converter to be simulated into the real time simulator. Indeed, the complexity of the converter to simulate can be significantly reduced if assumptions are made about its modes of operation. This has been used for the simulation of MMCs. Gregoire *et al.* (2011) determined a priori the conditions that would result in a cell of the MMC being ON, OFF, or in high-impedance mode. Then, functions were elaborated that determined the current through the cell's inductor and the voltage across its output capacitor based upon the state of the cell. By doing this, 60 cells composed of two IGBT/diodes each were simulated on an FPGA with a time step of 250ns. For this large number of switches, such a low time step would not be achievable if the system of equations were formed and had to be solved at each step. However, the above method is not generalizable to other topologies.

Myaing and Dinavahi (2011) simulated a three level voltage source inverter on an FPGA with a very low time step. The possible switching combinations of the converter were determined a priori, excluding faulted states. For example, for the four IGBTs in one arm, only five valid states were considered, instead of the theoretically possible sixteen. At every time step of the simulation, based on the polarity of the output current and the current switching state of the converter, the output voltage and input current were calculated. Using this approach a

time step of 12.5ns was reported. Despite this impressive time step, this method suffers from the same drawback of the MMC method: lack of generality. If, for example, one wishes to simulate a faulted state of the converter, then this model is no longer valid. Furthermore, for more complicated topologies, establishing the flow of converter switching states becomes an elaborate undertaking. For a general purpose solver, another approach must be used.

1.6 Special switch models

One approach is to use switch models that allow the system matrix to contain certain characteristics. Blanchette *et al.* (2012) modeled each switch as a resistance in parallel with a capacitance. When the switch is ON, the value of the resistance is low, and when the switch is OFF, the value of the resistance is high; therefore the admittance matrix is variable. For converters with a low number of switches, the capacitance is chosen to match the physical parasitic capacitance of the switch and the circuit is simulated by precomputing the inverses of the variable admittance matrix. When this no longer becomes feasible for higher number of switches, the system equations are solved using an iterative approach. Gauss-Seidel was used, which, according to the authors, converged in two iterations and they reported a time step 75ns for a boost converter. However, in order to guarantee convergence, the switch's parasitic capacitance had to be set to a value that ensured that the system matrix was diagonally dominant. This can affect the simulation accuracy since large capacitances may be required to ensure diagonal dominance if the time step is large.

Another switch model proposed by Pejovic and Maksimovic (1994) consisted of an inductor when the switch was ON and a capacitor when the switch was OFF. This representation allows the system matrix to remain constant, regardless of the switch states. This method will be explained by detail in the next chapter.

1.7 Conclusion

It can be seen that the real time simulation of high frequency power converters is limited by numerous constraints. Indeed, the limited memory on board FPGAs prevents precomputing the inverses of converters with more than a few number of switches. The computational complexity of solving a system of equations results in unacceptably high time steps. Making use of a priori knowledge results in simulators that cannot be easily extend to different cases. Finally, as will be explored at length in the subsequent chapters, using special switch models reduces simulation accuracy.

CHAPTER 2

EFFECT OF G_S PARAMETER ON SIMULATION ACCURACY FOLLOWING A COMMUTATION

In this chapter, the nature of the G_s parameter will be explained. Then, the theoretical commutation of a two level inverter will be compared and contrasted to its commutation when it is simulated with the Pejovic method, and it the effect of the G_s parameter during commutation will be examined.

2.1 Explanation of the Pejovic method

The method presented by Pejovic and Maksimovic (1994) is popular in the field of real time simulation because low time steps can be achieved due to the fact that the admittance matrix is constant for all switch combinations. In this section, this method will be presented in detail.

2.1.1 Switch model

Pejovic and Maksimovic (1994) guaranteed a constant system matrix by modeling each switch as a conductance (denoted as G_s) in parallel with a current source. This method benefits from the fact that the discrete companion models of the inductor and the capacitor are both represented as a conductance in parallel with a current source. The only difference between the two components is how the value of the current source is determined. With the Pejovic method, a switch is modeled as an inductor when ON (L_{ON}) and a capacitor when OFF (C_{OFF}). This is illustrated in figure 2.1. On the left side are the switch representations in continuous time, and on the right side are the discrete companion models. It can also be seen that the value of the L_{ON} and C_{OFF} depend on the value of G_s.

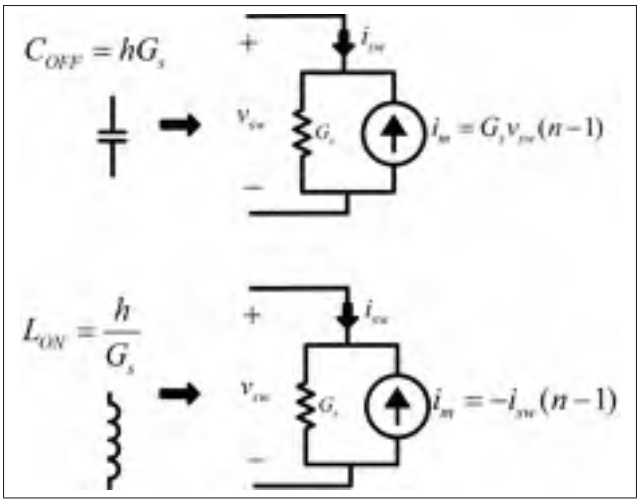

Figure 2.1 The Pejovic switch model. The continuous time representations of the switches when OFF and ON are shown on the left, and their equivalent discrete representations are shown on the right

2.1.2 Calculating the value of the memory current source

If Backward-Euler numerical differentiation is used, when the switch is OFF, the current source $i_m(n)$ is equal to $G_s v_{sw}(n-1)$ and when the switch is ON, $i_m(n)$ is equal to $-i_{sw}(n-1)$. The benefit of modeling switches in such a manner is that only the value of G_s appears in the system matrix; the value of the current source appears in the input vector. If a constant value of G_s is chosen, the matrix is constant for all possible switch combinations and the inverse of the admittance matrix needs to be computed only once at the beginning of the simulation.

2.1.3 Switch update rules

In order to calculate the value of the memory current source, the states of the switches must be known. The conditions under which a switch will turn ON or OFF are explained below.

2.1.3.1 Turn ON

An ideal switch turns ON the same step that its gate turns ON. A diode turns ON the step after the voltage across it becomes positive. With the Pejovic method, a pair consisting of a

switch and an antiparellel diode can be modelled as one component. This component turns ON the same step that its gate turns ON or, if the gate is OFF, the step after the voltage across it becomes negative.

2.1.3.2 Turn OFF

An ideal switch turns OFF the same step that its gate turns OFF. A diode turns OFF the step after the current through it becomes negative. If its gate is OFF, a switch/diode pair turns OFF the step after the current through the pair becomes positive.

2.1.4 Drawbacks of the model

In order to maximize the simulation accuracy, the values of L_{ON} and C_{OFF} should be kept as small as possible. From figure 2.1 it can be seen that the only way to do so is to reduce the size of the time step h. However, the minimum time step is limited by the speed at which the FPGA can multiply the precomputed inverse of the admittance matrix by the input vector. Despite being able to perform each dot product operation in parallel, for a given dot product, a series of sequential additions is required which limits the minimum possible time step as shown in figure 1.1. Belanger *et al.* (2013) simulated several converters on an FPGA using the Pejovic method with time steps in the order of 100ns. With a time step of 100ns, and a G_s set to 1, L_{ON} is equal to 100nH and C_{OFF} is equal to 100nF, values which are not negligible. The other parameter that the determines the size of the parasites is the switch conductance, G_s. C_{OFF} is directly proportional to G_s and L_{ON} is inversely proportional to G_s, so by varying the value of G_s, the value of one parasite can be reduced, at the cost in an increase of the magnitude of the other parasite. The value of G_s therefore has an important effect on simulation accuracy, and it can be very time consuming empirically choosing its optimal value for a given simulation. This is the motivation for developing an automatic method to optimize the G_s parameter.

2.1.5 Pejovic's error analysis

Pejovic and Maksimovic (1994) analyzed the commutation of two ideal switches from ON-OFF to OFF-ON. This situation is illustrated in figure 2.2.

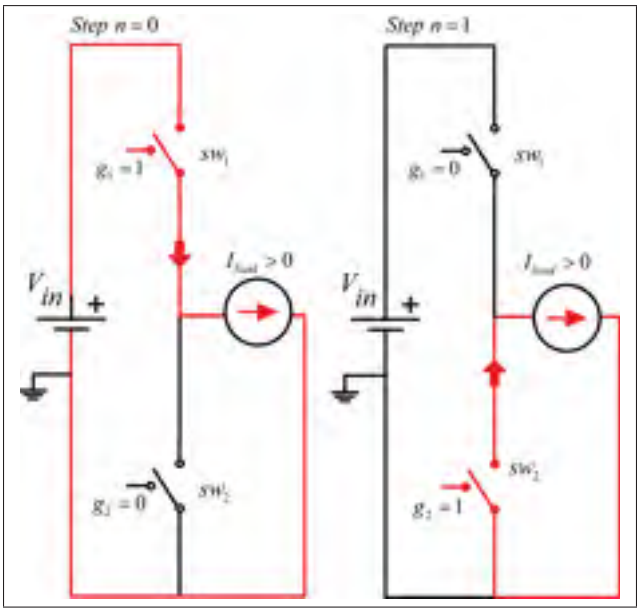

Figure 2.2 Pejovic's analysis of the behaviour of the commutation of one arm

When this scenario was simulated using the Pejovic method, Pejovic and Maksimovic (1994) observed that there were errors on the switches' voltage and current due to the parasitic L_{ON} and C_{OFF} and they quantified the error as follows:

$$e_{v_{sw_2}}(n) = V_{in}(f(n) - f(n-1)) - \frac{I_{load}}{G_s f(n)} \tag{2.1}$$

$$e_{i_{sw_1}}(n) = I_{load}(f(n) - f(n-1)) + G_s V_{in} f(n) \tag{2.2}$$

where

$$f(n) = \frac{sin(\frac{n\pi}{4})}{2^{\frac{n}{2}}} \tag{2.3}$$

They observed that these errors eventually decayed towards zero and that the instantaneous current error was directly proportional to the value of G_s while the instantaneous voltage error was inversely proportional to G_s. Based on their experiments with DC-DC converters, they determined that the optimal value of G_s was equal to $\frac{I_{load}}{V_{in}}$.

As can be seen, this analysis was performed for a circuit consisting of only ideal switches whose states changed simultaneously. Recall from section 2.1.3 that the state of an ideal switch changes during the same time step that its gate changes. However, most power electronic circuits consist of diodes and diode/switch pairs instead of ideal switches. This complicates the analysis since the states of the diodes and diode/switch pairs depend on the currents and voltages at the previous step. In other words, even if the gates of the two switch/diode pairs are changed simultaneously, the states of the switch/diode pairs will not necessarily change at the same time. In section 2.3, the original analysis of Pejovic and Maksimovic (1994) will be augmented by taking this effect into account. Beforehand, however, the theoretical behavior of a power electronic circuit undergoing a commutation will be examined. The analysis will be performed on one arm of the two level inverter shown in figure 2.3. The purpose of this analysis is to highlight the errors that the Pejovic method introduces when a commutation occurs. Quantifying these errors with respect to the G_s parameter is an essential step for developing an algorithm to automatically find the G_s value that minimizes the simulation error.

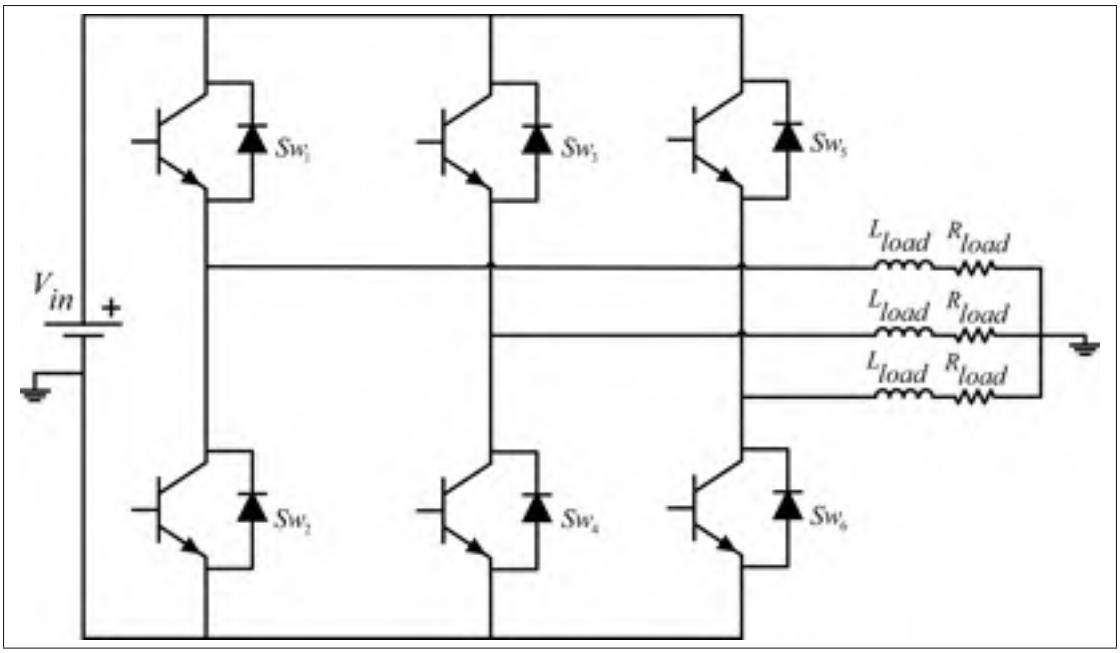

Figure 2.3 Schematic of a two level inverter. Note that the AC side of the converter is grounded. This will cause the load current to always be positive

2.2 Theoretical commutation process

In this section, it will be assumed that no dead time is used and that the gates of the two switches in the arm change simultaneously. The effect of dead time will be covered in section 2.3.4.2. It will also be assumed the load current varies slowly with respect to the duration of the commutation. Hence, the inductive load current is approximated by a constant current source I_{load} as shown in figure 2.4. The symbols used in the following subsections are also identified on figure 2.4.

2.2.1 sw_1 turn OFF and sw_2 turn ON process

At some time step $n = 0$, the following conditions are assumed: the gate of the top switch sw_1 is ON ($g_1 = 1$), the gate of the bottom switch sw_2 is OFF ($g_2 = 0$) and the load current I_{load} is positive. Since $g_1 = 1$ and $I_{load} > 0$, I_{load} must flow through sw_1. Since $g_2 = 0$, no current flows through sw_2 or d_2. If at the next step, $n = 1$, the gate signals are changed such that $g_1 = 0$

and $g_2 = 1$, no current flows through sw_1 or d_1 and I_{load} must flow through d_2 (d_2 turns ON). This process is shown in figure 2.4.

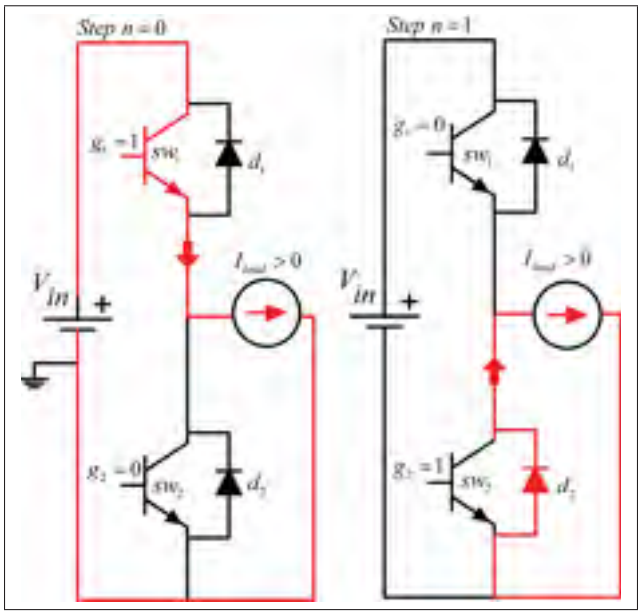

Figure 2.4 Theoretical commutation process from ON-OFF to OFF-ON

2.2.2 sw_1 turn ON and sw_2 turn OFF process

At some time step $n = 0$, the following conditions are assumed: $g_1 = 0$, $g_2 = 1$ and $I_{load} > 0$. Since $g_1 = 0$, no current flows through $sw1$ or d_1. Since $g_2 = 1$ and $I_{load} > 0$, I_{load} flows through d_2. If at the next step, $n = 1$, the gate signals are changed such that $g_1 = 1$ and $g_2 = 0$, I_{load} flows through sw_1 and no current flows through sw_2 or d_2 (d_2 turns OFF). This process is shown in figure 2.5.

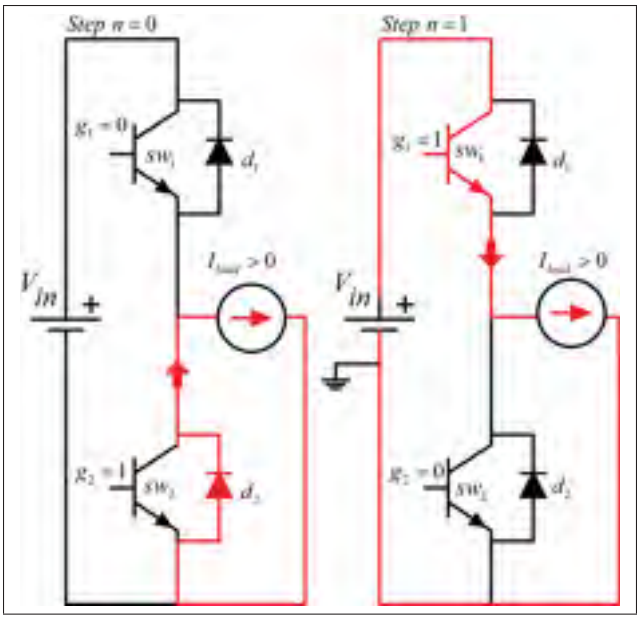

Figure 2.5 Theoretical commutation process from OFF-ON to ON-OFF

2.3 Commutation process using Pejovic method

It can be observed that in the theoretical case, when a commutation occurs the switches and diodes changed states the moment the gate signals changed. In the Pejovic method, that is not the case. It will be shown that there are situations in which the switch/diode pairs update several steps after the gate signals change.

2.3.1 Notation conventions

Throughout this work, the following conventions will be used: (t_{k_x}) marks the moments throughout a simulation when an OFF-ON to ON-OFF commutation begins; (t_{k_y}) marks the moments throughout a simulation when an ON-OFF to OFF-ON commutation begins; (t_{k_i}) corresponds to the moments when a switch turns ON; (t_{k_j}) corresponds to the moments when a switch turns OFF.

2.3.2 Equations for the switch voltage and current

The inverter arm modeled using the Pejovic method is presented in figure 2.6. Each switch/-diode pair is replaced by a current source in parallel with a resistance. Recall from section 2.1.3 that in order to update the state of a switch/diode pair the pair's current and voltage at the previous time step are required. Thus, in order to analyze the commutation process of a two-level inverter, the equations for the switch/diode pair's current and voltage will be formulated.

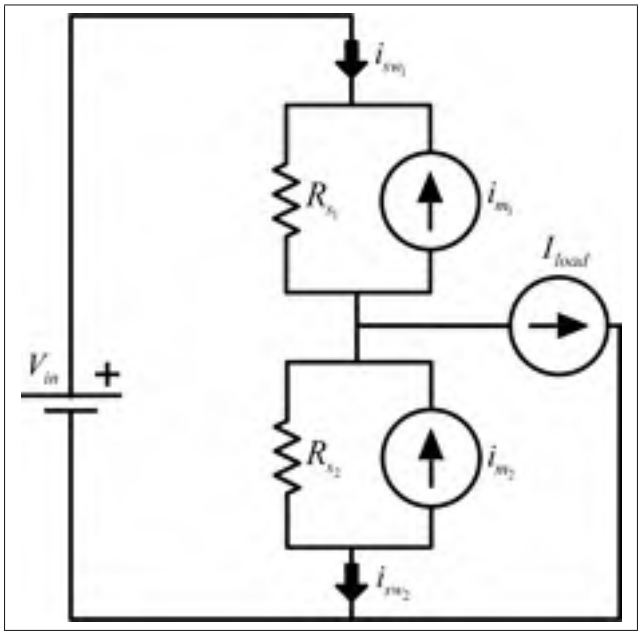

Figure 2.6 Equivalent circuit of one arm of a two-level inverter modeled using the Pejovic method. Note that R_s is simply $\frac{1}{G_s}$

First, KCL is applied at the node where the three current sources meet.

$$i_{R_{s1}}(n) = i_{m_1}(n) + i_{R_{s2}}(n) - i_{m_2}(n) + i_{load}(n) \tag{2.4}$$

where i_{m_1} and i_{m_2} are the values of the current sources (memory elements) of the upper and lower switches, respectively, $i_{R_{s1}}$ and $i_{R_{s2}}$ are the values of the current flowing through the parallel resistance of the equivalent model of the upper and lower switch/diode pairs, respectively. $i_{load}(n)$ is the value of the inductive load current.

Then KVL is applied around the loop composed of V_{in}, R_{s_1}, and R_{s_2}.

$$V_{in} = R_{s_1} i_{R_{s1}}(n) + R_{s_2} i_{R_{s2}}(n) \tag{2.5}$$

where $R_{s_1} i_{R_{s1}}(n)$ is the voltage across the top switch/diode pair and $R_{s_2} i_{R_{s2}}$(n) is the voltage across the bottom switch/diode pair.

Substituting equation (2.4) into equation (2.5), and rearranging yields:

$$i_{R_{s2}}(n) = \frac{V_{in} - R_{s_1}(i_{m_1}(n) - i_{m_2}(n) + i_{load}(n))}{R_{s_1} + R_{s_2}} \tag{2.6}$$

Substituting equation (2.6) into equation (2.4), and simplifying yields:

$$i_{R_{s1}}(n) = \frac{V_{in} + R_{s_2}(i_{mem_1}(n) - i_{m_2}(n) + i_{load}(n))}{R_{s_1} + R_{s_2}} \tag{2.7}$$

The current flowing through switch/diode pairs can be expressed as

$$i_{sw_1}(n) = i_{R_{s1}}(n) - i_{m_1}(n) \tag{2.8}$$

$$i_{sw_2}(n) = i_{R_{s2}}(n) - i_{m_2}(n) \tag{2.9}$$

Substituting (2.6) into (2.9) yields:

$$i_{sw_2}(n) = \frac{V_{in} - R_{s_1}(i_{m_1}(n) - i_{m_2}(n) + i_{load}(n))}{R_{s_1} + R_{s_2}} - i_{m_2}(n) \tag{2.10}$$

Similarly, substituting (2.7) into (2.8) yields:

$$i_{sw_1}(n) = \frac{V_{in} + R_{s_2}(i_{m_1}(n) - i_{m_2}(n) + i_{load}(n))}{R_{s_1} + R_{s_2}} - i_{m_1}(n) \tag{2.11}$$

The expressions for the voltages across the switch/diode pairs are

$$v_{sw_1}(n) = R_{s_1}(i_{sw_1}(n) + i_{m_1}(n)) = R_{s_1} \frac{V_{in} + R_{s_2}(i_{m_1}(n) - i_{m_2}(n) + i_{load}(n))}{R_{s_1} + R_{s_2}} \tag{2.12}$$

$$v_{sw_2}(n) = R_{s_2}(i_{sw_2}(n) + i_{m_2}(n)) = R_{s_2}\frac{V_{in} - R_{s_1}(i_{m_1}(n) - i_{m_2}(n) + i_{load}(n))}{R_{s_1} + R_{s_2}} \tag{2.13}$$

Equations 2.10 to 2.13 represent the voltages and currents of the two switches at a given time step in terms of the values of the current memory sources $i_{m_1}(n)$ and $i_{m_2}(n)$. This representation is important since the way in which the memory current sources are calculated depends on the state of the switches. In the next subsection these expressions will be used to gain a deeper understanding into the behaviour of the switches voltages and currents following a commutation and how the G_s parameter affects the accuracy of these quantities.

2.3.3 sw_1 turn ON and sw_2 turn OFF process

In this subsection, the behavior of the inverter arm will be analyzed for the case when the top switch turns ON and the bottom switch turns OFF.

2.3.3.1 Interval 1: sw_1 is ON and sw_2 is ON

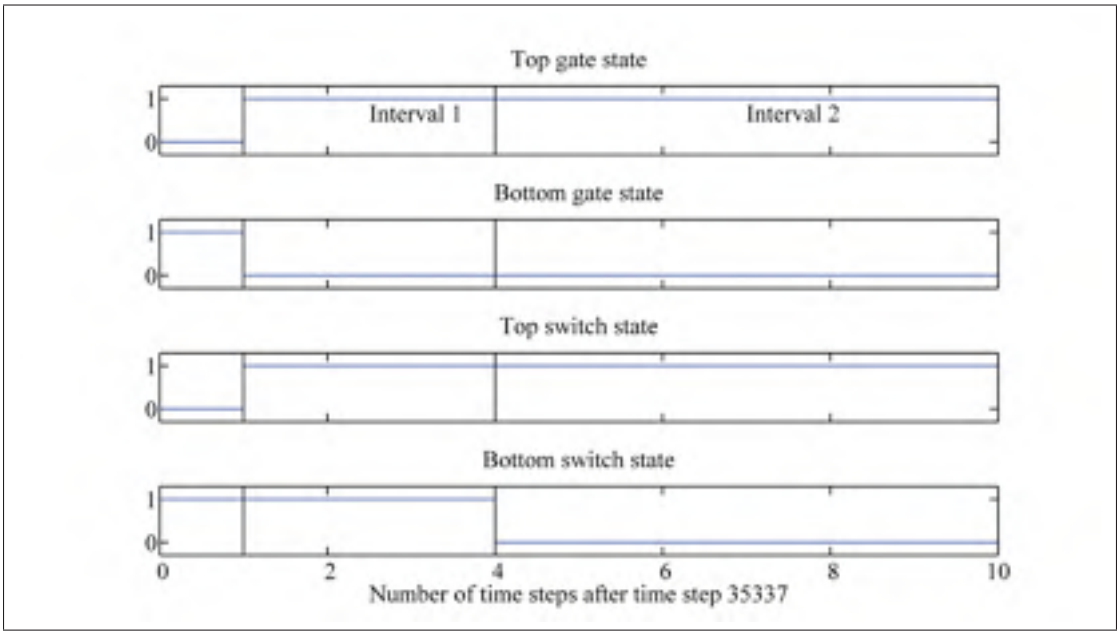

Figure 2.7 Gate and switch states during a commutation from OFF-ON to ON-OFF using the Pejovic method

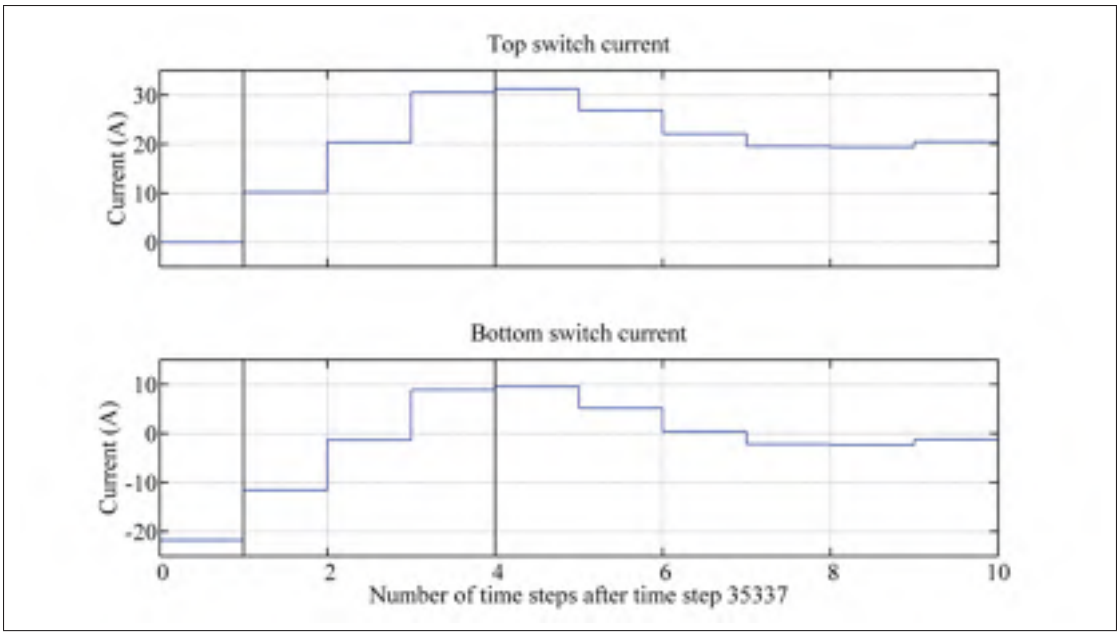

Figure 2.8 Switch currents during a commutation
from OFF-ON to ON-OFF using the Pejovic method

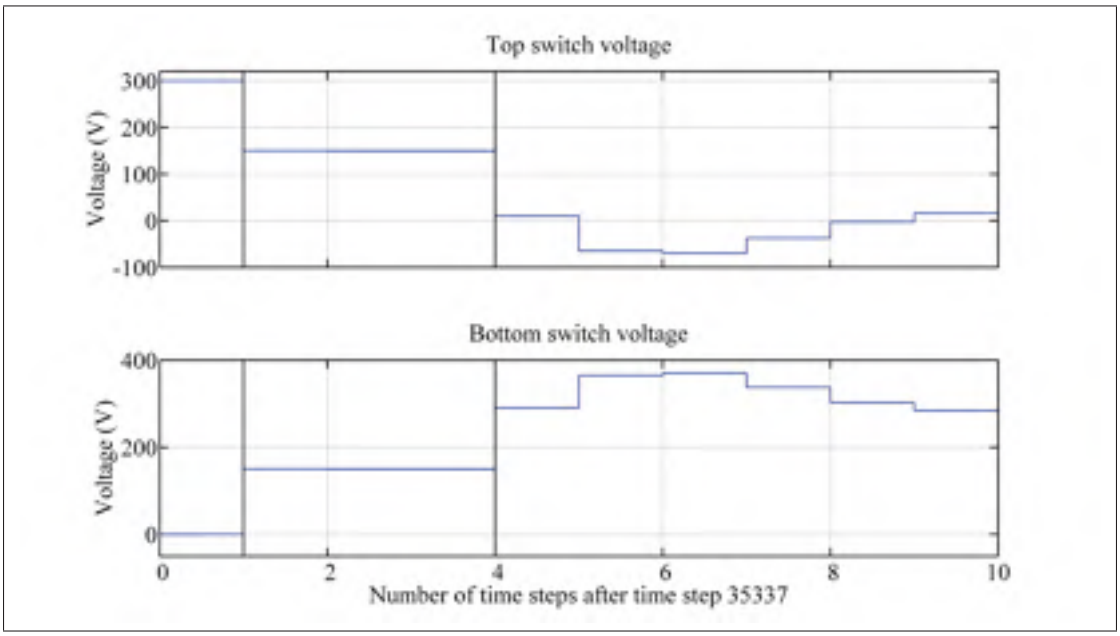

Figure 2.9 Switch voltages during a commutation
from OFF-ON to ON-OFF using the Pejovic method

The behavior of the circuit during a sample OFF-ON to ON-OFF commutation is illustrated in figures 2.7, 2.8, and 2.9. It is assumed that the value of $i_{load}(n)$ varies little during the commutation from OFF-ON to ON-OFF. Therefore, $i_{load}(n)$ is approximated by current source $I_{load}(t_{k_x})$ whose value is constant during the commutation beginning at moment t_{k_x}. At some time step $n = 0$, it is assumed that $g_1 = 0$ and the top switch/diode pair is OFF ($sw_1 = 0$). Also, $g_2 = 1$ and the bottom diode/switch pair is ON ($sw_2 = 1$), conducting a negative current with the same magnitude as $I_{load}(t_{k_x})$. This is shown on the left of side of figure 2.10. At the next step, $n = 1$, the gate signals are changed such that $g_1 = 1$ and $g_2 = 0$ causing sw_1 to turn ON.

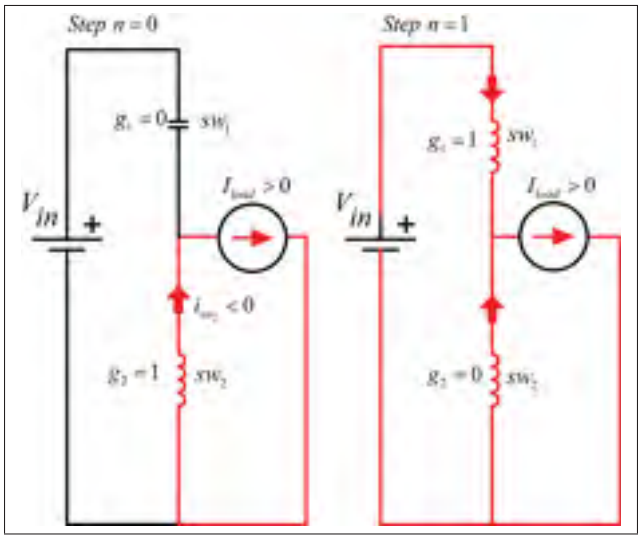

Figure 2.10 Continuous time representation of the transition from OFF-ON to the ON-ON intermediate interval

Since at the previous time step, i_{sw_2} was negative, sw_2 remains ON. Therefore, both the top and bottom switches are ON, and since each ON switch is modeled as an inductor, the continuous time representation of the circuit consists of two inductors in series as shown in the right hand side of figure 2.10. Referring back to the discretized circuit shown in figure 2.6, during this interval, $i_{m_1}(n) = -i_{sw_1}(n-1)$ and $i_{m_2}(n) = -i_{sw_2}(n-1)$. Substituting the values of $i_{m_1}(n)$ and $i_{m_2}(n)$ into (2.10) and simplifying yields:

$$i_{sw_2}(n) = \frac{V_{in}}{R_{s_1} + R_{s_2}} + i_{sw_2}(n-1) \tag{2.14}$$

Equations (2.12) and (2.13) reduce to

$$v_{sw_1}(n) = R_{s_1}\frac{V_{in}}{R_{s_1}+R_{s_2}} \tag{2.15}$$

$$v_{sw_2}(n) = R_{s_2}\frac{V_{in}}{R_{s_1}+R_{s_2}} \tag{2.16}$$

This is simply a voltage divider, and if the two switch resistances are the same (meaning that the same value of G_s is used for each switch), then the two switch voltages will be equal to half of the input voltage. These equations are valid as long as both switches in the arm are ON. Equation (2.14) reveals that every time step during interval 1 (when both switches are ON), i_{sw_2} will increase by a certain value. This can be expressed as:

$$\Delta i_{sw_2} = \frac{V_{in}}{R_{s_1}+R_{s_2}} \tag{2.17}$$

Recall that R_{s_1} and R_{s_2} are simply $\frac{1}{R_{s_1}}$ and $\frac{1}{R_{s_2}}$ respectively. If the same value of G_s is used for both switches, then 2.17 reduces to:

$$\Delta i_{sw_2} = 0.5V_{in}G_s \tag{2.18}$$

Eventually, i_{sw_2} will become positive (in the example shown in figures 2.7,2.8, and 2.9, this occurs at the 3$^{\text{rd}}$ time step). Since g_2 is OFF, the step after i_{sw_2} becomes positive, sw_2 turns OFF (in the example case, the 4$^{\text{th}}$ step) and the circuit enters the second interval which will be described in the next section. This behaviour is different from the theoretical case in two ways. First, instead of sw_2 immediately turning OFF and conducting no current when g_2 turns OFF, with the Pejovic method, sw_2 remains ON and the value of i_{sw_2} will increase over a certain number of time steps. Second, before sw_2 turns OFF, i_{sw_2} will become positive for one time step. This does not occur in the theoretical case because when $g_2 = 0$, $i_{sw_2} = 0$ and i_{d_2} can only conduct negative current. In other words, with the Pejovic method, i_{sw_2} will overshoot the zero value of current. The number of steps during which both switches remain ON is equal to the number of steps it takes i_{sw_2} to become positive. Recalling that the value of i_{sw_2} at the beginning

of the commutation was $-I_{load}(t_{k_x})$, the duration of the first interval can be expressed as:

$$\text{ceil}(\frac{I_{load}(t_{k_x})}{\Delta i_{sw_2}}) \tag{2.19}$$

where ceil indicates to round up the expression inside the parentheses. The value by which i_{sw_2} increased during the ON-ON interval is equal to $\text{ceil}(\frac{I_{load}(t_{k_x})}{\Delta i_{sw_2}})\Delta i_{sw_2}$. The value of the positive overshoot of i_{sw_2} is determined by simply subtracting the value of the load current from the previous expression. Therefore, if the last step of interval 1 is denoted as int^1_{end}, then the value of the current error at int^1_{end} is given by:

$$e_i(int^1_{end}) = \text{ceil}(\frac{I_{load}(t_{k_x})}{\Delta i_{sw_2}})\Delta i_{sw_2} - I_{load}(t_{k_x}) \tag{2.20}$$

From equation 2.18 it can be seen that as G_s increases, Δi_{sw_2} increases as well. In addition, the duration of the first interval (the number of steps where sw_1 and sw_2 are ON simultaneously) will decrease, but the overshoot at the end of this interval will increase.

2.3.3.2 Interval 2: sw_1 is ON and sw_2 is OFF

The step after i_{sw_2} becomes positive, sw_2 turns OFF and the second interval begins. Throughout the second interval, sw_1 is ON and sw_2 is OFF. Although the analysis of the first interval was performed directly on the Pejovic model of the circuit presented in figure 2.6, for the second interval it will be more straightforward to first consider the continuous time representation of the circuit and then analyze the effect of backward-Euler discretization. The continuous time representation of the circuit during the second interval is shown in figure 2.11.

The equations describing the behavior of this circuit can be written in the state-space form:

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{2.21}$$

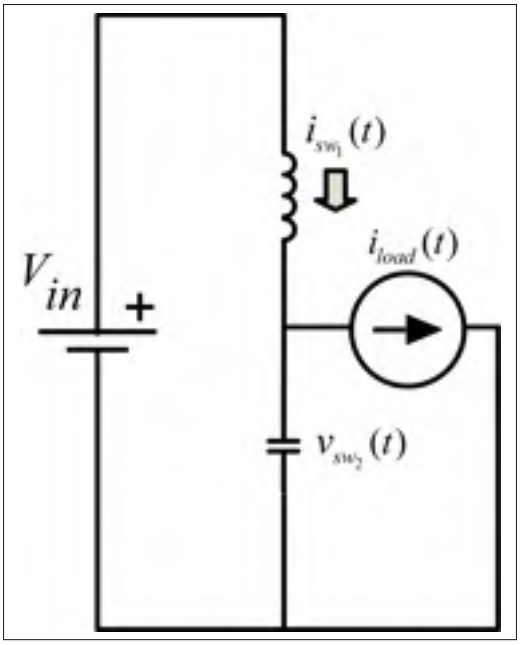

Figure 2.11 Continuous time representation of the circuit for the second interval

This representation will be useful in quantifying the oscillating behavior of the circuit during the second interval. For the inverter arm, equation 2.21 is equivalent to

$$\begin{bmatrix} \frac{di_{sw_1}(t)}{dt} \\ \frac{dv_{sw_2}(t)}{dt} \end{bmatrix} = \begin{bmatrix} 0 & -\frac{1}{L} \\ \frac{1}{C} & 0 \end{bmatrix} \begin{bmatrix} i_{sw_1}(t) \\ v_{sw_2}(t) \end{bmatrix} + \begin{bmatrix} 0 & \frac{1}{L} \\ -\frac{1}{C} & 0 \end{bmatrix} \begin{bmatrix} i_{load}(t) \\ V_{in} \end{bmatrix} \tag{2.22}$$

The next step is to obtain the equivalent discrete time state-space representation of the circuit. As explained by (Cellier and Kofman, 2006, p.38), the backward-Euler numerical integration algorithm can be written as

$$x_{n+1} = x_n + h\dot{x}(n+1) \tag{2.23}$$

Substituting 2.21 into 2.23, the following expression is obtained:

$$x_{n+1} = x_n + hAx_{n+1} + hBu \tag{2.24}$$

This can be rewritten as:

$$x_{n+1} = (I - hA)^{-1}x_n + h(I - hA)^{-1}Bu \tag{2.25}$$

or

$$x_{n+1} = A_d x_n + B_d u \tag{2.26}$$

where

$$A_d = (I - hA)^{-1} \tag{2.27}$$

and

$$B_d = h(I - hA)^{-1} B \tag{2.28}$$

Substituting the A matrix from 2.22 into 2.27 yields:

$$A_d = \frac{1}{1 + \frac{h^2}{LC}} \begin{bmatrix} 1 & -\frac{h}{L} \\ \frac{h}{C} & 1 \end{bmatrix} \tag{2.29}$$

Recall that with the Pejovic method $C = hG_s$ and $L = \frac{h}{G_s}$. Equation 2.29 can therefore be rewritten as:

$$A_d = \frac{1}{2} \begin{bmatrix} 1 & -G_s \\ \frac{1}{G_s} & 1 \end{bmatrix} \tag{2.30}$$

or:

$$A_d = \frac{1}{\sqrt{2}} \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{G_s}{\sqrt{2}} \\ \frac{1}{\sqrt{2}G_s} & \frac{1}{\sqrt{2}} \end{bmatrix} \tag{2.31}$$

This can be rewritten as

$$A_d = \frac{1}{\sqrt{2}} \begin{bmatrix} \cos\alpha & -G_s \sin\alpha \\ \frac{1}{G_s}\sin\alpha & \cos\alpha \end{bmatrix} \tag{2.32}$$

where α is 45 degrees. Similarly, it can be shown that:

$$B_d = \frac{1}{\sqrt{2}} \begin{bmatrix} \sin\alpha & G_s \cos\alpha \\ -\frac{1}{G_s}\cos\alpha & \sin\alpha \end{bmatrix} \tag{2.33}$$

Inserting 2.32 and 2.33 into 2.26

$$\begin{bmatrix} i_{sw_1}(n+1) \\ v_{sw_2}(n+1) \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} \cos\alpha & -G_s \sin\alpha \\ \frac{1}{G_s}\sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} i_{sw_1}(n) \\ v_{sw_2}(n) \end{bmatrix} + \begin{bmatrix} = \frac{1}{\sqrt{2}} - \sin\alpha & G_s\cos\alpha \\ \frac{1}{G_s}\cos\alpha & \sin\alpha \end{bmatrix} \begin{bmatrix} i_{load} \\ V_{in} \end{bmatrix} \tag{2.34}$$

Equation 2.34 describes how the values of the switches' voltage and current change from one step to another. The switches' voltage and currents k steps after the circuit has entered the second interval are represented by the following equations:

$$i_{sw_1}(k) = I_{load}(t_{k_x}) + 2^{-\frac{k}{2}}((i_{sw_1}(0) - I_{load}(t_{k_x}))\cos(k\alpha) - G_s(v_{sw_2}(0) - V_{in})\sin(k\alpha)) \tag{2.35}$$

$$v_{sw_2}(k) = V_{in} - 2^{-\frac{k}{2}}(\frac{(-i_{sw_1}(0) + I_{load}(t_{k_x}))}{G_s}\sin(k\alpha) + (-v_{sw_2}(0) + V_{in})\cos(k\alpha)) \tag{2.36}$$

where $i_{sw_1}(0)$ and $v_{sw_2}(0)$ are the initial values of the L_{ON} and C_{OFF}, respectively, during the second interval. The initial value of the L_{ON} during the second interval interval is equal to the current through sw_1 at the end of the first interval. Similarly, the initial value of the C_{OFF} during the second interval is equal to the voltage across sw_2 at the end of the first interval. This can be expressed as:

$$i_{sw_1}(0) = i_{sw_1}(int^1_{end}) \tag{2.37}$$

and:

$$v_{sw_2}(0) = v_{sw_2}(int^1_{end}) \tag{2.38}$$

The correct value of i_{sw_1} is $I_{load}(t_{k_x})$ and the correct value of v_{sw_2} is V_{in}. Equations 2.35 and 2.36 show that i_{sw_1} is equal to $I_{load}(t_{k_x})$ plus an erroneous oscillatory term and v_{sw_2} is equal to V_{in} minus an erroneous oscillatory term:

$$i_{sw_1}(k) = \underbrace{I_{load}(t_{k_x})}_{\text{correct value of } i_{sw_1}} + \underbrace{2^{-\frac{k}{2}}}_{\text{damping factor}} \underbrace{((i_{sw_1}(0) - I_{load}(t_{k_x}))\cos(k\alpha) - G_s(v_{sw_2}(0) - V_{in})\sin(k\alpha))}_{\text{oscillatory error term}}$$

$$v_{sw_2}(k) = \underbrace{V_{in}}_{\text{correct value of } v_{sw_2}} - \underbrace{2^{-\frac{k}{2}}}_{\text{damping factor}} \underbrace{\left(\frac{(-i_{sw_1}(0)+I_{load}(t_{k_x}))}{G_s}\sin(k\alpha)+(-v_{sw_2}(0)+V_{in})\cos(k\alpha)\right)}_{\text{oscillatory error term}}$$

The magnitude of the error on i_{sw_1} is directly proportional to G_s and the magnitude of the error on v_{sw_2} is inversely proportional to G_s. Given enough time these oscillations will be damped to zero. Indeed, at every step, they are damped by $\sqrt{2}$. This damping is purely numeric since in theory an LC circuit should oscillate forever. If the initial conditions $i_{sw_1}(0)$ and $v_{sw_2}(0)$ are set to zero, then equations 2.35 and 2.36 reduce to equations 2.1 and 2.2 formulated by Pejovic and Maksimovic (1994). This is logical since they did not consider the L_{ON}-L_{ON} interval.

2.3.4 sw_1 turn OFF and sw_2 turn ON process

In this subsection, the behavior of the inverter arm will be analyzed for the case when the top switch turns OFF and the bottom switch turns ON.

2.3.4.1 Interval 1: sw_1 is OFF and sw_2 is ON

It is assumed that the value of $i_{load}(n)$ varies little during the commutation from ON-OFF to OFF-ON. Therefore, $i_{load}(n)$ is approximated by current source $I_{load}(t_{k_y})$ whose value is constant during the commutation beginning at moment t_{k_y}. At some time step $n = 0$, $g_1 = 1$ and $g_2 = 0$. sw_2 is OFF, with $v_{sw_2} = V_{in}$. sw_1 is ON with $i_{sw_1} = I_{load}(t_{k_y})$. At the next step, $n = 1$, the gate signals are changed such that $g_1 = 0$ and $g_2 = 1$. From the update rules described in 2.1.3, since $g_2 = 1$, the bottom switch/diode pair turns ON. In addition, since $g_1 = 0$ and $i_{sw_1}(n-1) > 0$, sw_1 turns OFF. The circuit is therefore represented by a C_{OFF} in series with an L_{ON}. The behavior is described by the following equations::

$$i_{sw_2}(k) = I_{load}(t_{k_y}) + 2^{-\frac{k}{2}}((i_{sw_2}(0) - I_{load}(t_{k_y}))\cos(k\alpha) - G_s(v_{sw_1}(0) - V_{in})\sin(k\alpha)) \quad (2.39)$$

$$v_{sw_1}(k) = V_{in} - 2^{-\frac{k}{2}}\left(\frac{(-i_{sw_2}(0) + I_{load}(t_{k_y}))}{G_s}\sin(k\alpha) + (-v_{sw_1}(0) + V_{in})\cos(k\alpha)\right) \qquad (2.40)$$

In this case, the initial values $i_{sw_2}(0)$ and $v_{sw_1}(0)$ are set to zero since there is no intermediate interval.

2.3.4.2 Effect of dead time

Recall, that for the transition from $g_1 = 0$, $g_2 = 1$ to $g_1 = 1$, $g_2 = 0$, there was an intermediate interval where both switch/diode pairs were ON. This interval is purely an artifact of the simulation algorithm. It can also be observed that for the $g_1 = 1$, $g_2 = 0$ to $g_1 = 0$, $g_2 = 1$ transition, there is no such intermediate interval. While this is true for the case with no dead time, if a dead time is added, an artificial intermediate interval will present itself.

2.3.4.2.1 Interval 1: sw_1 is OFF and sw_2 is OFF

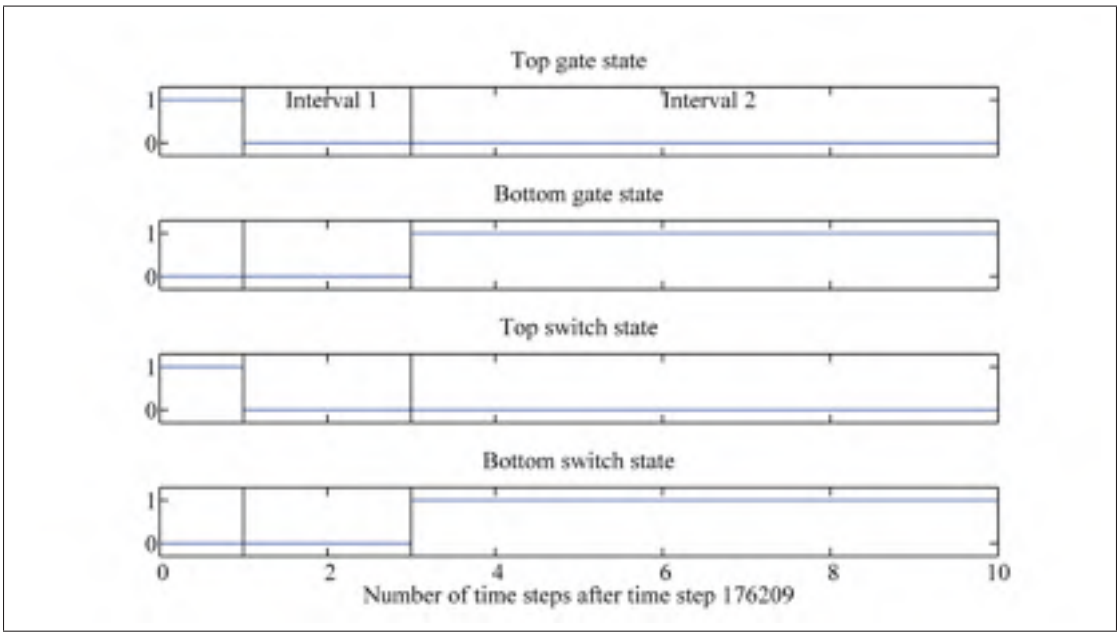

Figure 2.12 Gate and switch states during a commutation from ON-OFF to OFF-ON using the Pejovic method

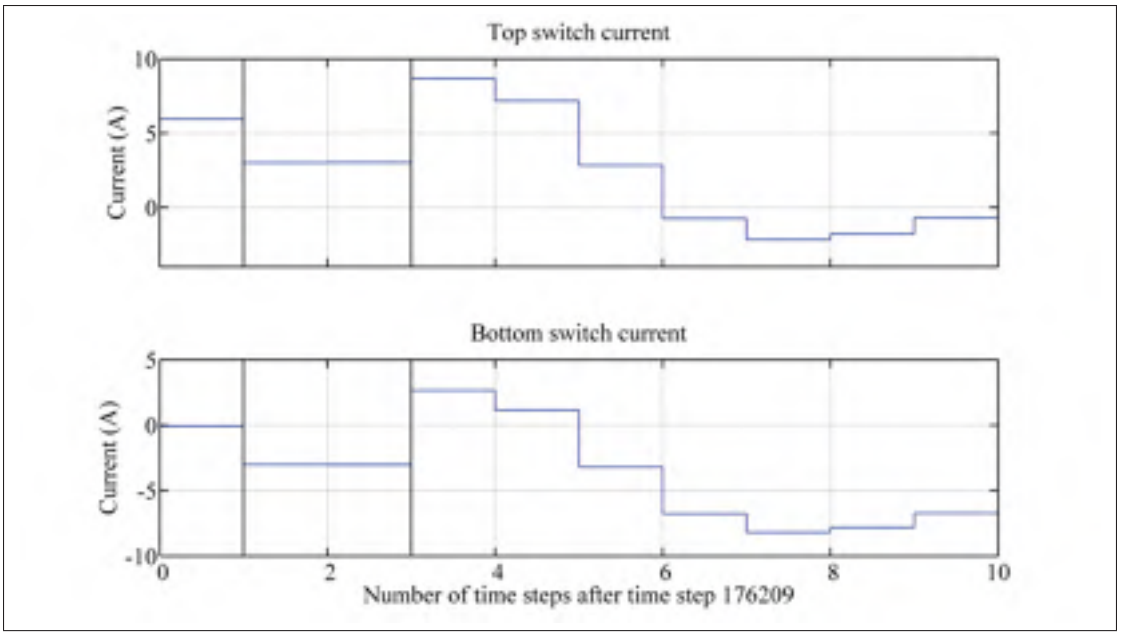

Figure 2.13 Switch currents during a commutation from ON-OFF to OFF-ON using the Pejovic method

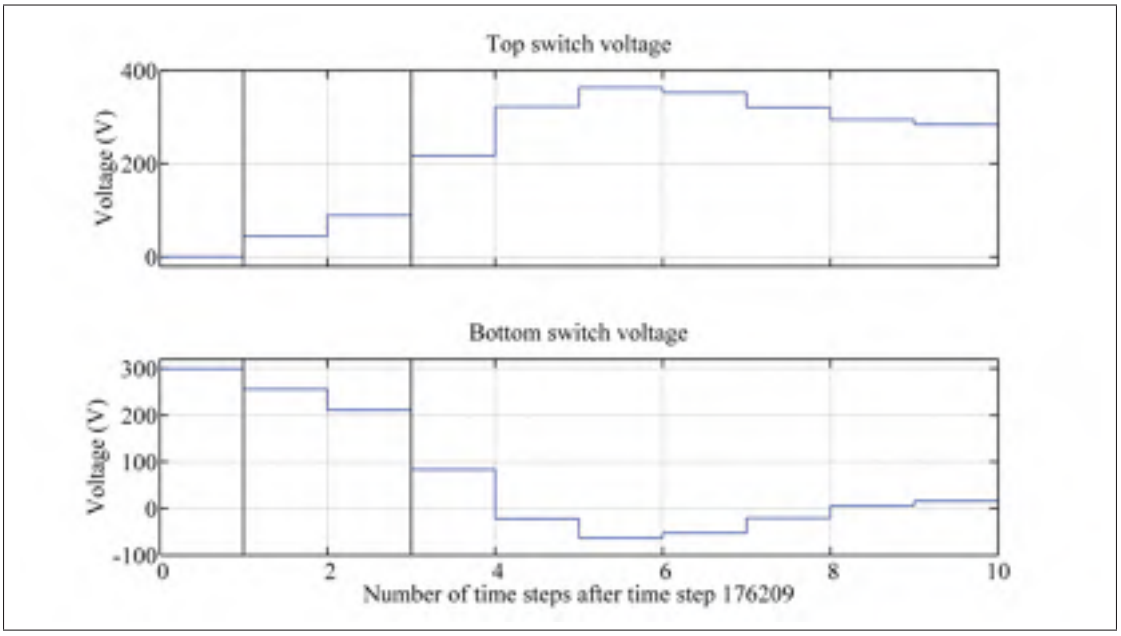

Figure 2.14 Switch voltages during a commutation from ON-OFF to OFF-ON using the Pejovic method

The behavior of the circuit during a sample ON-OFF to OFF-ON transition with deadtime is illustrated in figures 2.12, 2.13, and 2.14. It is assumed that at some time step $n = 0$, $g_1 = 1$ and $g_2 = 0$. As in the case with no dead time, sw_2 is OFF with $v_{sw_2} = Vin$ and sw_1 is ON with $i_{sw_1} = I_{load}(t_{k_y})$. At the next step, $n = 1$, the gate signals are changed such that $g_1 = 0$ and $g_2 = 0$. In theory, when both the gates turn OFF, the bottom diode will turn ON so that the inductive load current is not interrupted. However, with the Pejovic method, this is not the case. As in the case with no dead time, sw_1 turns OFF. However, instead of turning ON, sw_2 will remain OFF as well. This is due to the fact that at $n = 0$, the voltage across sw_2 was positive. The continuous time representation of the circuit therefore consists of two C_{OFF} in series. As explained in subsection 2.1.2, $i_{m_1}(n) = \frac{v_{sw_1}(n-1)}{R_{s_1}}$ and $i_{m_2}(n) = \frac{v_{sw_2}(n-1)}{R_{s_2}}$. By inserting these into equations (2.12) and (2.13), and simplifying, the following expressions are obtained:.

$$v_{sw_1}(n) = v_{sw_1}(n-1) + \Delta v_{sw} \tag{2.41}$$

$$v_{sw_2}(n) = v_{sw_2}(n-1) - \Delta v_{sw} \tag{2.42}$$

Where

$$\Delta v_{sw} = \frac{R_{s_1} R_{s_2}}{R_{s_1} + R_{s_2}} I_{load}(t_{k_y}) \tag{2.43}$$

is the rate of change of the switch voltage. If the values of R_{s_1} and R_{s_2} are the same, then:

$$\Delta v_{sw} = \frac{I_{load}(t_{k_y})}{2G_s} \tag{2.44}$$

It can be seen that the value Δv_{sw} will vary through the simulation as the value of $I_{load}(t_{k_y})$ changes. It is also interesting to note that Δv_{sw} is inversely proportional to G_s. Recall that during the first interval of the sw_1 turn ON and sw_2 turn OFF process, Δi_{sw} is directly proportional to G_s. This is consistent with the pattern so far: increasing G_s improves some cases, but causes a deterioration in others. The switch currents are

$$i_{sw_1}(n) = \frac{R_{s_2}}{R_{s_1} + R_{s_2}} I_{load}(t_{k_y}) \tag{2.45}$$

$$i_{sw_2}(n) = -\frac{R_{s_1}}{R_{s_1}+R_{s_2}} I_{load}(t_{k_y}) \tag{2.46}$$

If the values of R_{s_1} and R_{s_2} are the same, then the current through both switches is of the same magnitude and equal to half of the load the current. Unlike the L_{ON}-L_{ON} interval, there will be an overshoot at the end of the C_{OFF}-C_{OFF} interval only if v_{sw_2} becomes negative before the deadtime elapsed. If the deadtime elapses first, then there is no overshoot at the end of this phase. If the last step of the first interval is denoted as int^1_{end}, then the value of the current error at int^1_{end} (if the dead time hasn't elapsed) is given by:

$$e_v(int^1_{end}) = \text{ceil}(\frac{V_{in}}{\Delta V_{sw}})\Delta V_{sw_2} - V_{in} \tag{2.47}$$

where: ceil($\frac{V_{in}}{\Delta V_{sw}}$) is the number of steps where both switches are OFF.

sw_2 will turn ON and the circuit will enter the second interval the step after v_{sw_2} becomes negative or the step when the dead time period elapses (when $g_2 = 1$), whichever comes first. In the example shown in figures 2.12, 2.13, and 2.14, the deadtime elapses by the 3$^{\text{rd}}$ step which is before v_{sw_2} becomes negative. Therefore, on the 3$^{\text{rd}}$ step, sw_2 turns ON.

2.3.4.2.2 Interval 2: sw_1 is OFF and sw_2 is ON

The behaviour of the circuit during the second interval is described by equations 2.39 and 2.40, with the initial values $i_{sw_2}(0)$ and $v_{sw_1}(0)$ set to their values at the end of the first interval.

2.4 Conclusion

As seen in the above analysis, the Pejovic method introduces artificial intervals (L_{ON}-L_{ON} and C_{OFF}-C_{OFF}). At the end of these intervals, there will be an error on the switch waveforms due to the switch/diode pairs requiring values of the previous time step in order to update. It was shown that the value of G_s has an effect on this error. For the sw_1 turn ON and sw_2 turn OFF transition, the error on the switch current at the end of the L_{ON}-L_{ON} interval is directly proportional to G_s. For the sw_1 turn OFF and sw_2 turn ON transition, the error on the switch

voltage at the end of the C_{OFF}-C_{OFF} interval is inversely proportional to G_s (if the deadtime hasn't elapsed yet). It is therefore clear that a compromise must be made. It was also shown that the Pejovic method introduces oscillations during the second interval. During this interval, the voltage oscillations is inversely proportional to G_s, while the magnitude of the current oscillations is directly proportional to G_s. Once again, a compromise must be when selecting the appropriate value of G_s. In the next chapter, the effect of the G_s parameter on the output current and the commutation losses will be examined.

CHAPTER 3

EFFECT OF G_S PARAMETER ON FUNDAMENTAL COMPONENT OF OUTPUT CURRENT AND ON COMMUTATION LOSSES

In the previous chapter, the error introduced by the Pejovic method during commutations was analyzed. In this chapter, the error introduced by the Pejovic method on the fundamental component of the output current (i_{load_1}) and on the commutation losses will be analysed.

3.1 Effect of G_s parameter on i_{load_1}

The effect of G_s on i_{load_1} is clearly illustrated by examining extreme values of G_s. For very low values of G_s, one arm will always be composed of a large inductance ($L_{ON} = \frac{h}{G_s}$) and small capacitance ($C_{OFF} = hG_s$). The very small C_{OFF} will charge very quickly with respect to the switching frequency and the very large L_{ON} will charge very slowly, meaning that both components can be approximated as open circuits. This approximation is illustrated in figure 3.1.

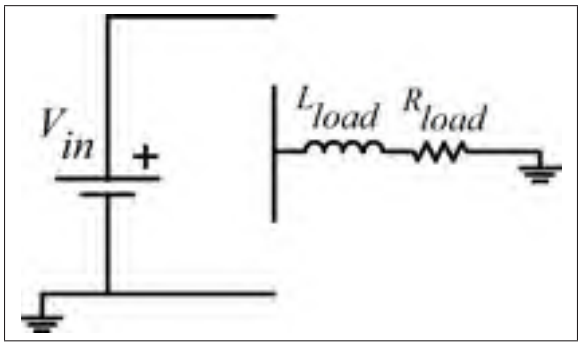

Figure 3.1 Approximate circuit representation for a 2 level inverter with a very low value of G_s

It can be seen that there is an open circuit between V_{in} and the load, and the load current will therefore be zero. This is clearly different from an actual inverter were the load current is sinusoidal and its fundamental component assumes a certain non-zero value. For very large

values of G_s, one arm will always be composed of a small inductance and large capacitance. The very large C_{OFF} will charge very slowly with respect to the switching frequency and the very small L_{ON} will charge very quickly, meaning that both components can be approximated as short circuits circuits. This approximation is illustrated in figure 3.2.

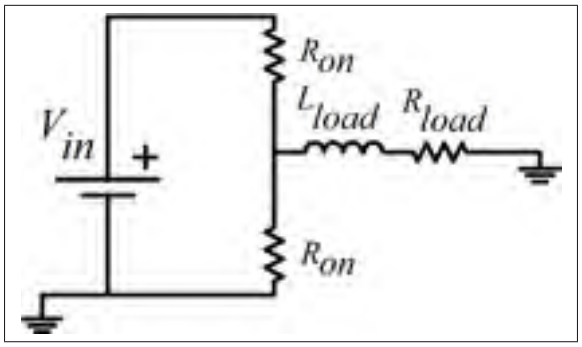

Figure 3.2 Approximate circuit representation for a 2 level inverter with a very high value of G_s

It can be seen that the input voltage source is essentially shorted (the only resistance across it will be two R_{ON}, the very small ON resistances of the switches). Hence, sw_1 and sw_2 will conduct a large amount of current and the input power drawn from the DC voltage source will be very high. Since the value of R_{ON} is negligible compared to the load impedance, the voltage across the load will be approximately $\frac{V_{in}}{2}$. Therefore the output current will have the same form of the current in a series RL circuit driven by a DC voltage source. This means that as time goes on, it will approach a DC value which is once again significantly different from the case of the theoretical inverter. This situation is illustrated in figure 3.3.

Once again, a compromise must be made. Selecting too high of a value of G_s will cause the input power drawn to be enormous. Selecting too low of a value of G_s will cause the output current and power to be too low.

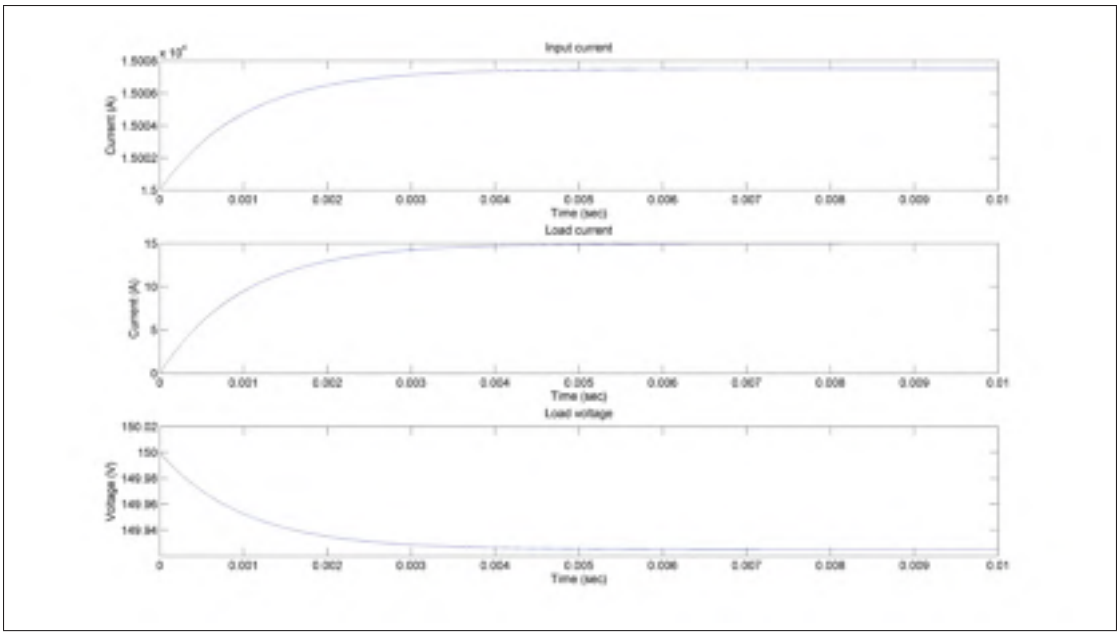

Figure 3.3 Waveforms for a 2 level inverter with a very high value of G_s

3.2 Effect of G_s parameter on commutation losses

Whenever a commutation occurs during a simulation using the Pejovic method, energy is lost. This processes is illustrated in figure 3.4. It is assumed that at some step n the top switch is OFF (and modeled as a capacitor) and the bottom switch is ON (and modeled as an inductor). It is also assumed that the top switch has been OFF long enough for the oscillations across it to have been damped out.

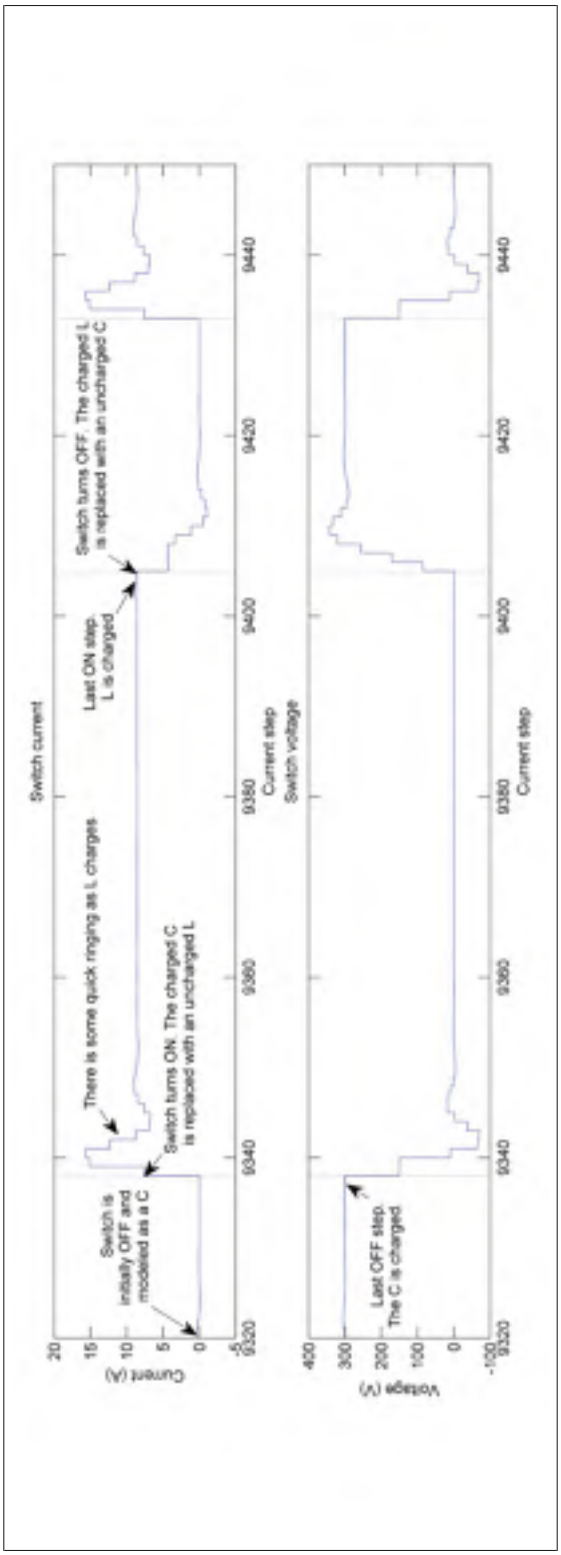

Figure 3.4 Visualization of the commutation losses introduced by the Pejovic method

Hence, $i_{sw_1}(n) = 0$ and $v_{sw_1}(n) = V_{in}$. The C_{OFF} is therefore storing an energy equal to $0.5Cv^2_{sw_1}(n)$. [1] Concretely, this stored energy manifests itself in the form of the memory current source which has a value equal to $i_{m_1}(n) = G_s V_{sw_1}(n-1)$. At $n+1$, suppose the gates change such that $g_1 = 1$ and $g_2 = 0$. As explained previously, the top switch turns ON so that it is now modelled as an inductor. The value of its current source memory element is equal to $i_{m_1}(n) = -i_{sw_1}(n-1)$. However, since the current at the previous time step was zero, the magnitude of the current source is zero, meaning that it contains no charge. It can therefore be seen that a charged C_{OFF} is replaced by an uncharged L_{ON}, meaning that the energy stored in the C_{OFF} at step n is lost. Similarly, during turn OFF, a charged L_{ON} is replaced by a C_{OFF} with an initial memory value of $i_{m_1}(n) = G_s v_{sw_1}(n-1)$. As shown in section 2.2.4, $v_{sw_1}(n-1) = 0$. Therefore, the C_{OFF} is uncharged. [2] Therefore energy is lost every time a commutation occurs. The total losses for a given switch during turn ON can be represented by:

$$\sum_{i}^{I} 0.5CV^2_{sw}(t_{k_{i-1}}) \tag{3.1}$$

where t_{k_i} are the turn ON moments of the switch during the course of the simulation. Similarly, the total losses during turn OFF are:

$$\sum_{j}^{J} 0.5LI^2_{sw}(t_{k_{j-1}}) \tag{3.2}$$

where t_{k_j} are the turn OFF moments of the switch during the course of the simulation. As was shown in the previous section, the values of i_{sw} and v_{sw} are affected by the value of G_s. The

[1] This equation is valid for continuous time (it is derived by integrating the power of the capacitor with respect to time). Since in discrete time integration is approximate, the energy stored in a capacitor or inductor is not necessarily $0.5Cv(t)^2$ or $0.5Li(t)^2$, but empirically this discrepancy did not affect the optimization results.

[2] It should be noted that the losses of the bottom switches are different from those of the top ones. As was explained in section 2.2.3, the bottom switch only turns OFF after a $L_{ON} - L_{ON}$ intermediate interval during which the voltage across it is $\frac{V_{in}}{2}$ and not the zero volts across the top switch prior to OFF. Therefore, during turn OFF, unlike the case of the top L_{ON} which was replaced by an uncharged C_{OFF}, the bottom L_{ON} is replaced by a C_{OFF} that contains some initial charge ($i_{m_2} = G_s \frac{V_{in}}{2}$). Hence, the losses for the bottom switches are lower than the losses of the top switches. For the purpose of this work, this discrepancy will be ignored.

total energy lost in the switch during the simulation can be expressed as a function of G_s

$$E_{loss}(G_s) = \sum_{i}^{I} 0.5hG_s v_{sw}^2(G_s, t_{k_{i-1}}) + \sum_{j}^{J} 0.5\frac{h}{G_s} i_{sw}^2(G_s, t_{k_{j-1}}) \tag{3.3}$$

As mentioned previously, this equation is approximate due to the effect of discretization on the energy calculation. The extent of this approximation was studied on the two level inverter with the parameter values listed in table 4.2 with a G_s of 0.0505. The true energy loss was determined using the following equation:

$$E_{loss} = \sum_{n}^{N} i_{sw}(n) v_{sw}(n) h \tag{3.4}$$

where h is the size of the simulation time step and N is the number of steps during the simulation. The energy loss calculated using this equation was 68.29J compared to a loss of only 22.96J calculated by equation 3.3. While this is a significant difference between the two calculation methods, they are still within an order of magnitude of each other, and, as will be shown in the final chapter, the approximation is sufficiently adequate for the optimization algorithm to provide optimal values of G_s. It is also important to point out that the turn ON losses are directly proportional to G_s and that the turn OFF losses are inversely proportional to G_s. Again, it is seen that a compromise must be made when choosing a value of G_s.

3.3 Conclusion

In this chapter, the effect of the G_s parameter on the output current was examined. It was found that selecting too high or too low of a G_s value completely alters the behavior of the circuit. The effect of the G_s parameter on the commutation losses was also examined. It was shown that selecting too high or too low of a G_s value results in unacceptable losses. Therefore a compromise must be made. This is consistent with the analysis of chapter 2, which showed that too high or too low of a G_s value results in inaccurate switch voltage or current waveforms following a commutation. This effect of the G_s on multiple aspects of the circuits behaviour explains why choosing an appropriate value of G_s through trial and error is so difficult. In the

next chapter, a method to automatically find the optimal value of G_s is proposed and validated for three case studies.

CHAPTER 4

OPTIMIZATION METHOD AND VALIDATION

In this chapter, a method to automatically optimize the value of the G_s parameter is proposed and its is performance assessed for three case studies.

4.1 Optimization method

In the previous chapter, the following equation was derived for the commutation losses accrued during the simulation:

$$E_{loss}(G_s) = \sum_{i}^{I} 0.5hG_s v_{sw}^2(G_s, t_{k_{i-1}}) + \sum_{j}^{J} 0.5\frac{h}{G_s} i_{sw}^2(G_s, t_{k_{j-1}}) \tag{4.1}$$

In order to simplify the optimization problem, it will be assumed that near the optimal value of G_s i_{sw} and v_{sw} are independent of G_s. Therefore, equation 4.1 can be rewritten as:

$$E_{loss}(G_s) = 0.5hG_s V_{SQ} + 0.5\frac{h}{G_s} I_{SQ} \tag{4.2}$$

where

$$I_{SQ} = \sum_{j}^{J} I_{sw}^2(t_{k_{j-1}}) \tag{4.3}$$

$$V_{SQ} = \sum_{i}^{I} V_{sw}^2(t_{k_{i-1}}) \tag{4.4}$$

An expression for the value of G_s that minimizes this loss can be obtained by evaluating the following derivative:

$$\frac{dE_{loss}(G_s)}{dG_s} = 0 \tag{4.5}$$

Evaluating this expression yields:

$$0.5hV_{SQ} - 0.5\frac{h}{G_s^2} I_{SQ} \tag{4.6}$$

Rearranging and simplifying yields:

$$G_s = \sqrt{\frac{I_{SQ}}{V_{SQ}}} \tag{4.7}$$

4.1.1 Implementation of the algorithm

From equation 4.7, it can be seen that in order to find the optimal G_s, the values of I_{SQ} and V_{SQ} must be determined. The values will be determined by running a reference SPS simulation and measuring the switch voltages and currents at the moments $t_{k_{j-1}}$ and $t_{k_{i-1}}$. It is important to note the basis for this algorithm is the assumption of hard switching, that is that at the moments before the commutation ($t_{k_{j-1}}$ and $t_{k_{i-1}}$) the switch voltage/current is non-zero. If that is not the case, then the proposed algorithm is not applicable and further investigation for this case is needed. For example, in the case of a boost converter operating in discontinuous conduction mode (DCM), the diode will turn OFF at a zero crossing of current. In order words, when the L_{ON} is replaced by the C_{OFF}, no energy is lost. Therefore, I_{SQ} will be equal to zero, and the G_s calculated by the script will be zero. Another way of seeing the situation is that the algorithm chooses a G_s that makes the L_{ON} and C_{OFF} losses equal to each other. However, since in this case the L_{ON} losses are zero, the algorithm will calculate a G_s of zero.

4.1.2 Analytical expression for the case of the two level inverter

For certain cases, the optimal G_s can be calculated directly from an analytical expression without having to use the proposed optimization algorithm. The grounded two level inverter shown in figure 4.1 that has been studied in this work is such a case. For this converter, in theory, the switch current is equal to the load current when it is ON and the switch voltage is equal to the input voltage when it is OFF. By assuming that the load current varies slowly from one step to another, I_{SQ} can be rewritten as

$$I_{SQ} = \sum_{j}^{J} I_{load}^2(t_{k_j}) \tag{4.8}$$

and V_{SQ} can be rewritten as

$$V_{SQ} = \sum_{i}^{I} V_{in}^2 \tag{4.9}$$

The number of turn ON and turn OFF moments are equal (or there will be a difference of at most 1 between them), so I = J. Therefore, G_s can be expressed as:

$$G_s = \sqrt{\sum_{j}^{J} \frac{I_{load}^2(t_{k_j})}{V_{in}^2}} \tag{4.10}$$

This can be rewritten as

$$G_s = \frac{I_{load_{RMS}}(t_{k_j})}{V_{in}} \tag{4.11}$$

If the switching frequency is much higher than the load current frequency, then equation 4.11 can be rewritten as

$$G_s = \frac{I_{load_{RMS}}}{V_{in}} \tag{4.12}$$

It is also crucial to highlight the fact that the optimal value of G_s depends on the ratio between the load current and the input voltage. This ratio is altered by the value the load resistance, which means that as the load changes, the optimal value of G_s changes as well. This is especially problematic for the case of motors connected to the converter, since in this case the load is dynamic. Indeed, a major limitation of the Pejovic method is that the optimal value of G_s is only valid for a specific operating point.

4.1.3 Relationship between the proposed algorithm and simulation accuracy following a commutation

The proposed algorithm is designed to minimize the commutation losses. In this subsection, it will be shown that it also gives reasonable G_s values with respect to maximizing the simulation accuracy following a commutation. In the second chapter, it was shown that when a commutation occurred there was an intermediate interval where the two switches were either both ON or both OFF, followed by an interval where one switch was ON and the other was OFF. It was also shown that at the end of the intermediate interval there was an error on the switch voltage

and current. During the second interval, there were oscillations whose magnitudes depended on the values of $-i_{sw_0} + I_{load}(t_k)$ and $-v_{sw_0} + V_{in}$ where i_{sw_0} and v_{sw_0} are equal to the switch current and voltage, respectively, at the end of the first interval. If there is no error at the end of the first interval, then $i_{sw_0} = I_{load}(t_k)$ and $v_{sw_0} = V_{in}$ and the magnitude of the oscillations is zero. In other words, if the error at the end of the intermediate interval is kept small, then the error during the second interval will be small as well. The maximum possible voltage error is ΔV_{sw} which is equal to $\frac{I_{load}(t_{k_y})}{2G_s}$. The maximum possible current error is ΔI_{sw} which is equal to $0.5 G_s V_{in}$. These voltage and current errors are normalized with respect to the input voltage and load current, respectively. The normalized errors are $\frac{I_{load}(t_{k_y})}{2V_{in}G_s}$. and $\frac{G_s V_{in}}{2I_{load}(t_{k_x})}$. These errors are present during every commutation.

$$e_i = \sum_{x}^{X} \frac{G_s V_{in}}{2I_{load}(t_{k_x})} \tag{4.13}$$

$$e_v = \sum_{y}^{Y} \frac{I_{load}(t_{k_y})}{2G_s V_{in}} \tag{4.14}$$

The total error can be minimized by setting these errors equal to each other. Doing so and simplifying, as well as assuming that the load current varies little between the OFF-ON to ON-OFF and ON-OFF to OFF-ON transitions (meaning that $I_{load}(t_{k_x}) = I_{load}(t_{k_y})$) the following expression is obtained:

$$G_s = \sum_{x}^{X} \frac{I_{load}(t_{k_x})}{V_{in}} \tag{4.15}$$

This expression is equivalent to

$$G_s = \frac{I_{load_{AVG}}(t_{k_x})}{V_{in}}. \tag{4.16}$$

This is similar to expression obtained by minimizing the losses which is restated here:

$$G_s = \frac{I_{load_{RMS}}(t_{k_j})}{V_{in}} \tag{4.17}$$

Assuming that the load varies little between t_{k_j} and t_{k_x}, then the only difference between the two expressions is the difference between $I_{load_{RMS}}(t_{k_j})$ and $I_{load_{AVG}}(t_{k_j})$. For a pure sinusoidal load current with a DC offset it can be shown that these values are similar. Therefore, for the

specific case of the grounded two level inverter, the G_s that minimizes the commutation losses is close the G_s that minimizes the current and voltage errors during commutations.

4.2 Optimization methods validation

In this section, the performance of the automatic optimization method developed in section 4.1 will be examined for three topologies (two level inverter, three level NPC inverter, direct matrix converter). For each topology, the optimal G_s was determined automatically by performing a benchmark SPS simulation and using a script to calculate I_{SQ}, V_{SQ}, and the optimal value of G_s. In order to assess the validity of the G_s value obtained using the automatic method, the error on I_{load_1} and commutation losses with the optimal G_s were compared to those for a range of G_s values. For a range of G_s values between 0.00001 and 1000, a simulation was performed. For each G_s value, the commutation losses and the I_{load_1} error were measured. These results were then traced as a function of G_s. The I_{load_1} error and commutation losses (as a percentage of SPS output power) were added together and also traced as a function of G_s.

4.2.1 Summary of results

Table 4.1 Summary of results at optimal Gs

	Two level	**NPC**	**Matrix**
Optimal G_s (S)	0.0505	0.0188	0.1905
Losses	326.8 W	324.2 W	256.7 W
Losses as a % of SPS output power	4.740 %	33.78 %	5.674
% error of i_{load_1}	-3.20 %	-5.80 %	- 1.75 %

4.2.2 Two level inverter

The circuit shown in figure 2.3 was tested. It is restated in figure 4.1 for convenience.

After performing a benchmark SPS simulation and running the automatic optimization algorithm, the following values were calculated: $I_{SQ} = 1.3792 \times 10^6 A^2$. $V_{SQ} = 5.4005 \times 10^8 V^2$, $G_s = 0.0505S$. The optimal G_s obtained automatically is compared to the simulation results

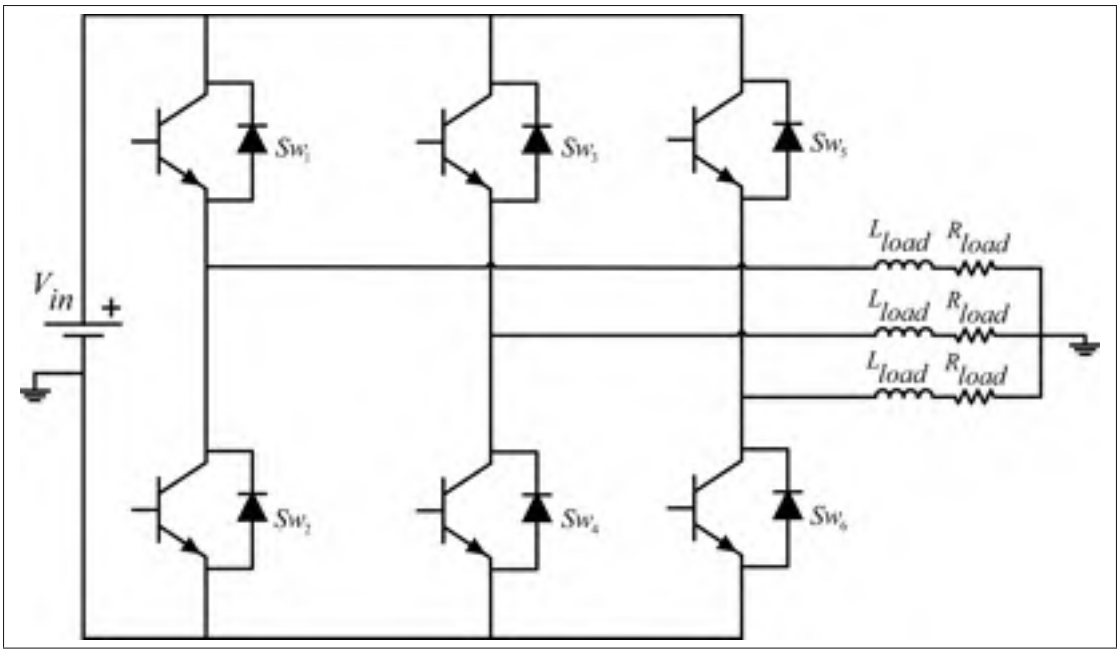

Figure 4.1 Schematic of two level inverter

Table 4.2 Two level inverter parameter values

Parameters	**Values**
f_{sw}	50 kHz
V_{in}	300 V
f_{ref}	50 Hz
R_{on}	1 mΩ
deadtime	500 ns
modulation index	0.40
R_{load}	10 Ω
L_{load}	38.1 mH
T_s	210 ns

for various G_s values shown in figures 4.2, 4.3, and 4.4. From figure 4.2 it can be seen that for the small values of G_s towards the left of the graph, the losses are small and that there is a large error on i_{load_1}. As explained in section 3.1, this is due to the fact that as the value of G_s gets smaller, the circuit starts to resemble an open circuit. For large values of G_s, the losses are very large. It also appears that the error on the fundamental current approaches zero as the G_s gets larger. However, this is deceiving since for even larger values of G_s (e.g. around 100000),

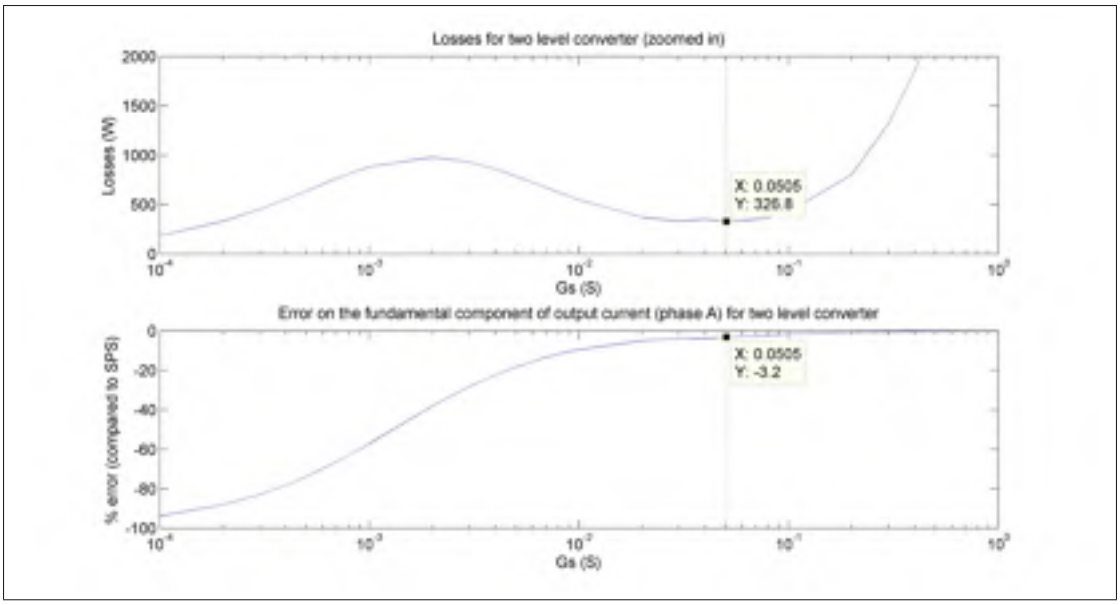

Figure 4.2 Losses and error on i_{load_1} for different Gs values for a two level inverter

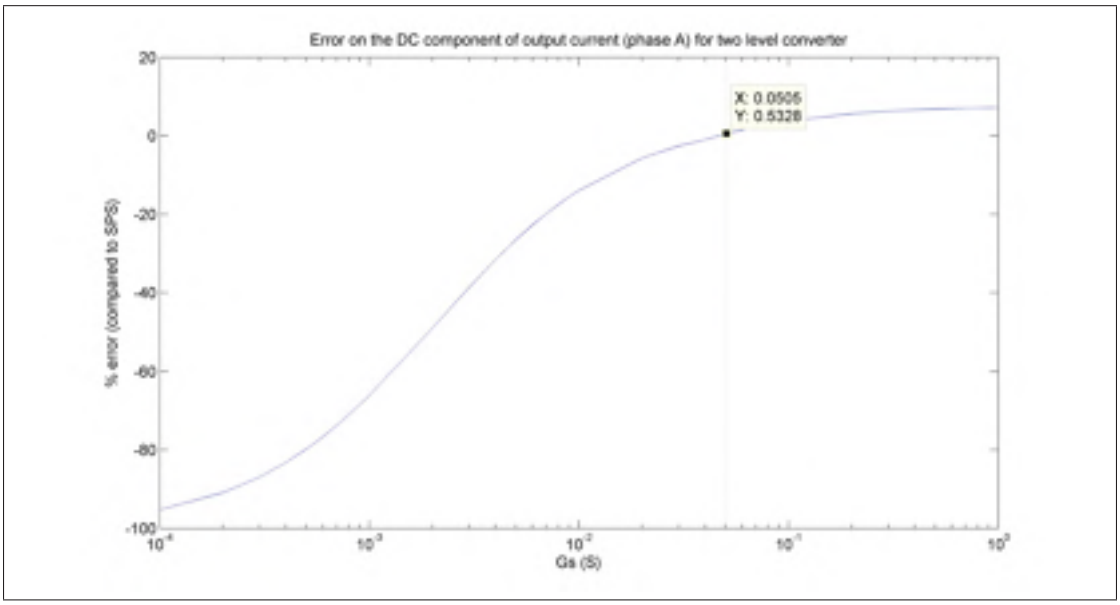

Figure 4.3 Error on the DC component of the output current for different Gs values for a two level inverter

the error on i_{load_1} will be large. Indeed, for very large values of G_s, the output of the circuit will resemble that of an RL circuit. It's just that the values of G_s shown on the graph didn't reach high enough for that effect to begin to become noticeable. However, the increase of

losses with increasing G_s is noticeable. Theoretically, this losses will increase without bound as G_s. In practice, they will be limited by the R_{ON} on the switches to some very large value. It can also be seen that there is a local minimum of losses which occurs very near the value of G_s determined by the automatic optimization algorithm. The graph can be interpreted in the following manner. Near the optimal value of G_s, the assumption that G_s has a negligible effect on the values of the switch current and voltage holds true. Therefore, the commutation losses increase as the chosen G_s moves away from the optimal G_s. However, the further away the chosen G_s is from the optimal value of G_s, the less this assumption is valid, and the effects described in section 3.1 begin to dominate.

In summary, for very low values of G_s the losses are very small, for very high values of G_s the losses are very large and for values between these two extremes, there is a local minimum of losses. The G_s that produces this local minimum of losses also produces an accurate output current waveform (3.2% error).

By adding the (absolute value of) the percent error on i_{load_1} with the percent loss, a concave curve with a global minimum is once again obtained as seen in figure 4.4. The G_s calculated using the automatic optimization algorithm occurs very near this minimum.

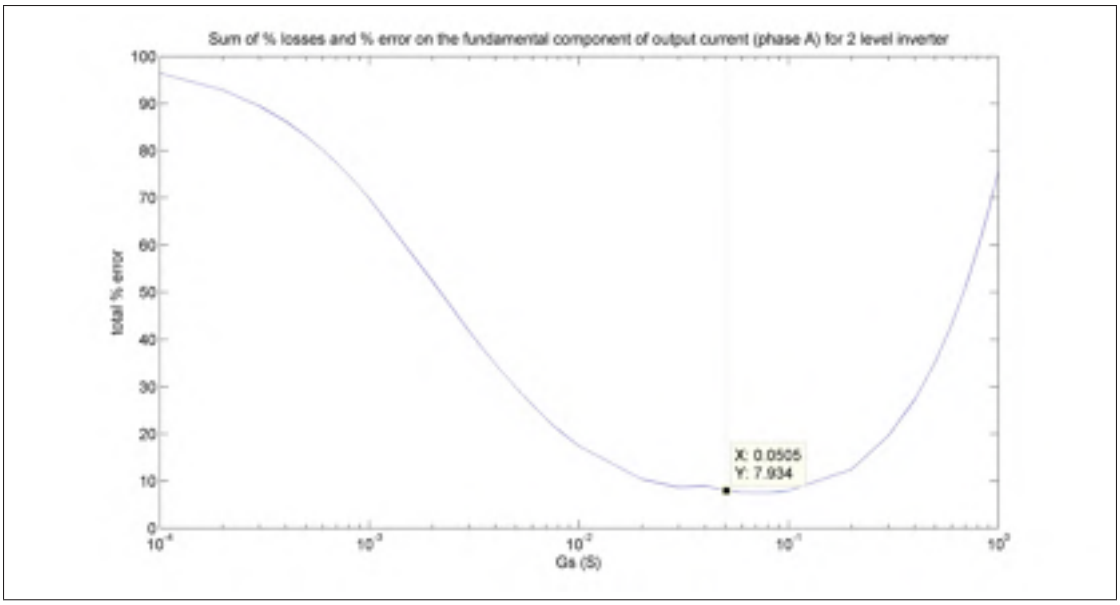

Figure 4.4 Sum of % losses and % error on the fundamental output current for different Gs values for a two level inverter

4.2.2.1 Example analytical calculation of G_s

In section 4.1 it was shown that an analytical expression can be used to calculate the value of G_s for a two level inverter. This expression is restated here for convenience

$$G_s = \frac{I_{load_{RMS}}}{V_{in}} \tag{4.18}$$

In this section, this expression will be validated by comparing it the output of the optimization algorithm. For the two level inverter in question, when the SPS baseline simulation was run, the measured load current was 15.162 A RMS and the input voltage was 300V. By inserting these values into equation 4.18, a G_s of 0.05054 is obtained, which is essentially identical to the G_s found using the optimization algorithm.

4.2.2.2 Waveforms at optimal G_s

The waveforms for the simulation with the optimal G_s are presented in figures 4.5 to 4.9. Figure 4.5 shows that the load current with the Pejovic simulation is very close to its SPS counterpart.

Figures 4.8 and 4.9 show the switch current and voltage waveforms. It can be seen that there is an overshoot every time a commutation occurs and that the percentage overshoot in the voltage and current waveforms are close to each other. This is expected as explained in subsection 4.1.3. The inductive load filters out this high frequency noise, explaining the accuracy of the output current waveform. Similar observations can be made for the waveforms for the following two cases.

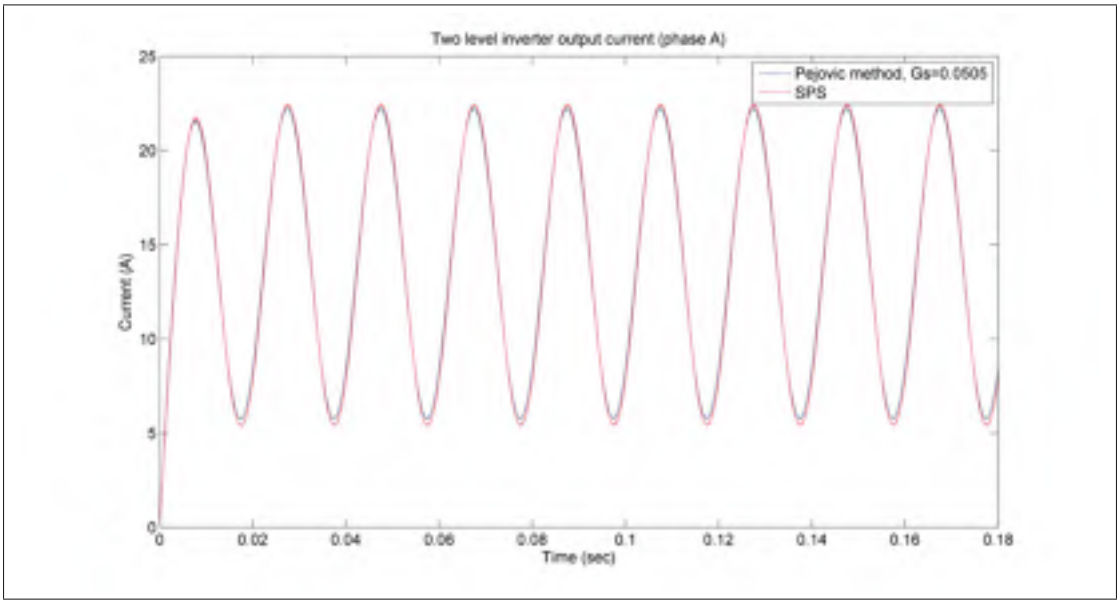

Figure 4.5 I_{out} waveform

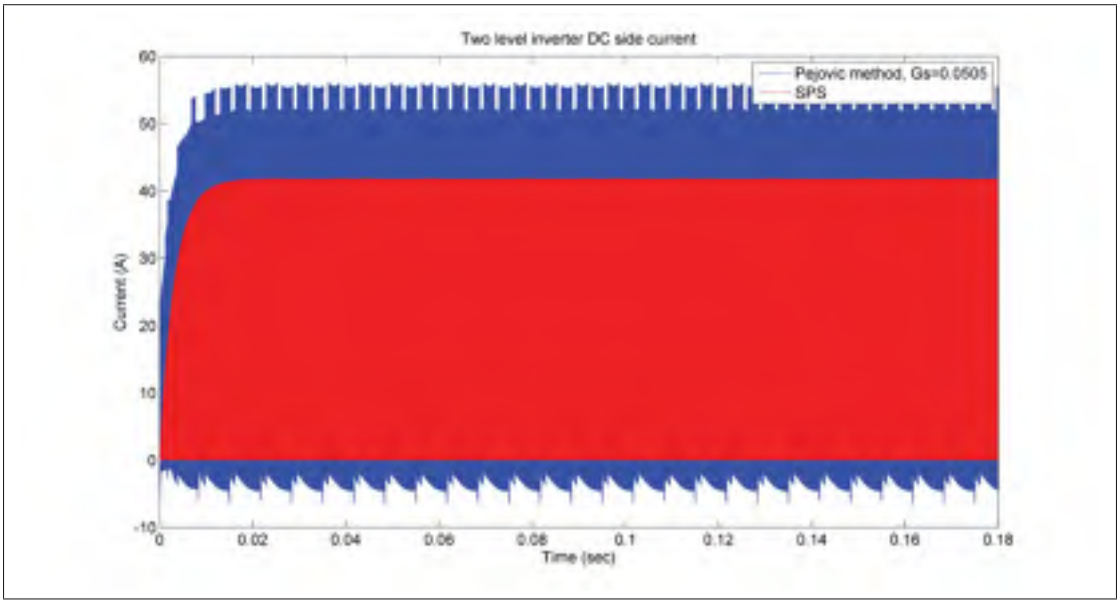

Figure 4.6 DC side current waveform

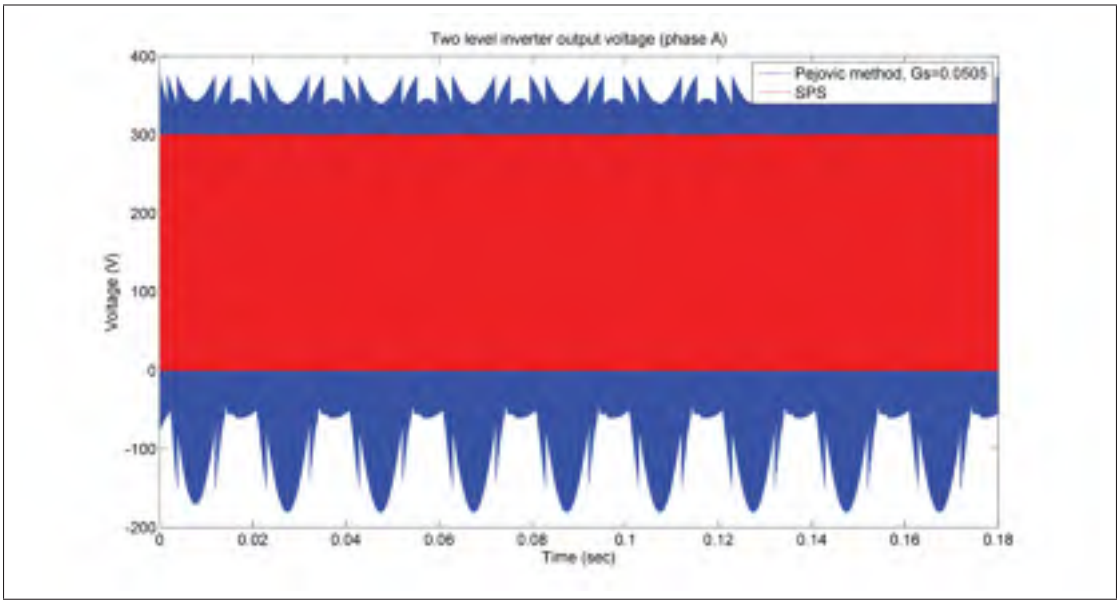

Figure 4.7 V_{out} waveform

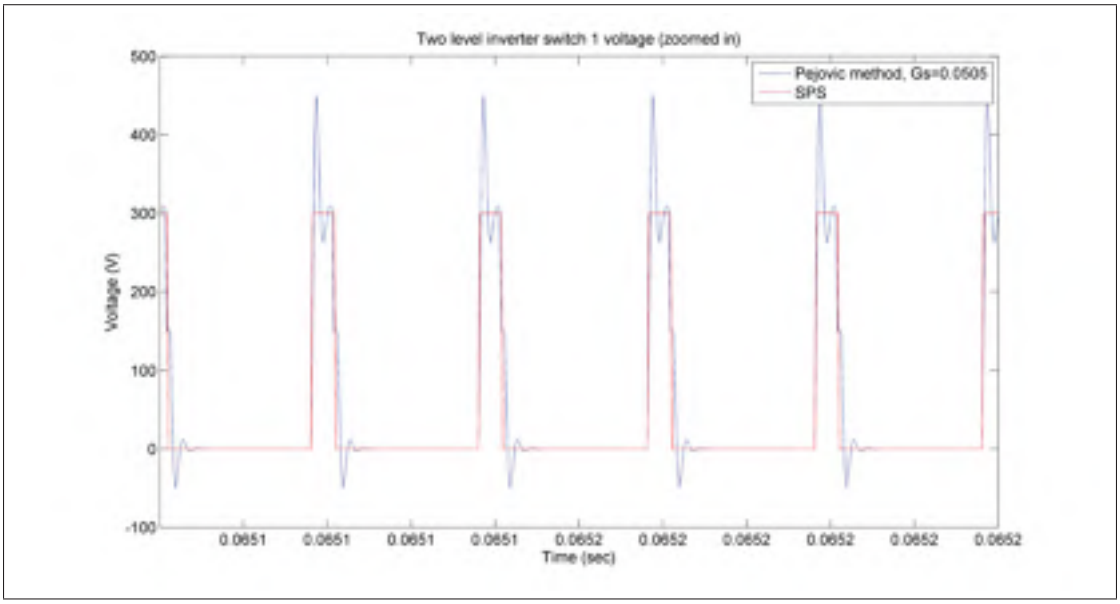

Figure 4.8 v_{sw_1} waveform

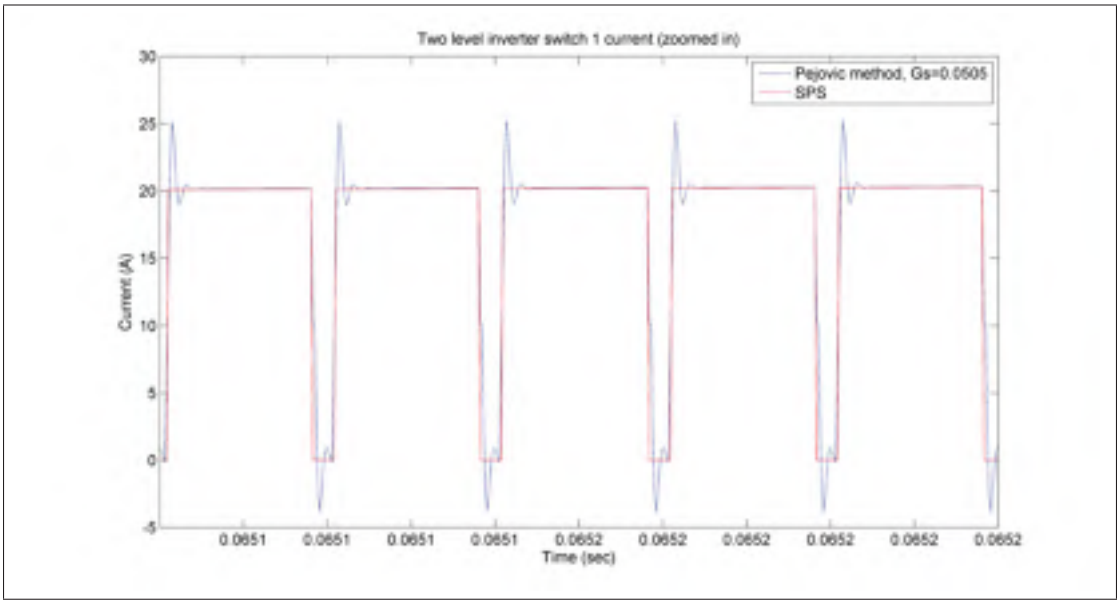

Figure 4.9 i_{sw_1} waveform

4.2.3 Three level NPC inverter

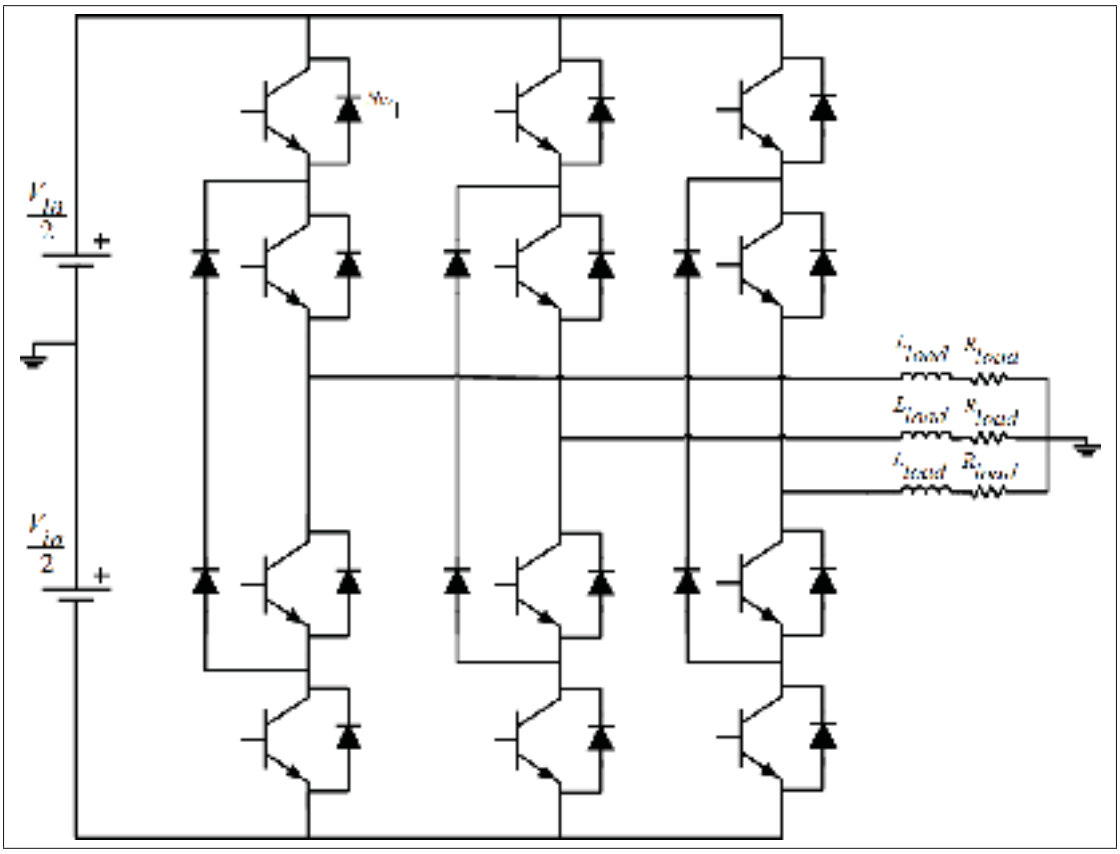

Figure 4.10 Schematic of three level NPC inverter

Table 4.3 Three level NPC inverter parameter values

Parameters	Values
f_{sw}	50 kHz
V_{in}	600 V
f_{ref}	50 Hz
R_{on}	1 mΩ
deadtime	500 ns
modulation index	0.40
R_{load}	10 Ω
L_{load}	38.1 mH
T_s	420 ns

The next converter the automatic optimization algorithm was validated on was the three level NPC inverter shown in figure 4.10. After performing a benchmark SPS simulation and running the automatic optimization algorithm, the following values were calculated: $I_{SQ} = 9.0210 \times 10^6 A^2$. $V_{SQ} = 2.5651 \times 10^8 V^2$, $G_s = 0.0188S$. The optimal G_s obtained automatically is compared to the simulation results for various G_s values shown in figures 4.11 and 4.12. In the case of the three level NPC inverter, like in the case of the two level inverter, the losses approach zero for a small values of G_s and increase significantly for higher values of G_s. However, unlike the two level inverter, there is no local minimum of losses as depicted in figure 4.11. However, by adding the (absolute value of) the percent error on i_{load_1} with the percent loss, a concave curve with a global minimum is once again obtained as shown in figure 4.12. The G_s calculated using the automatic optimization algorithm once again occurs very near this minimum.

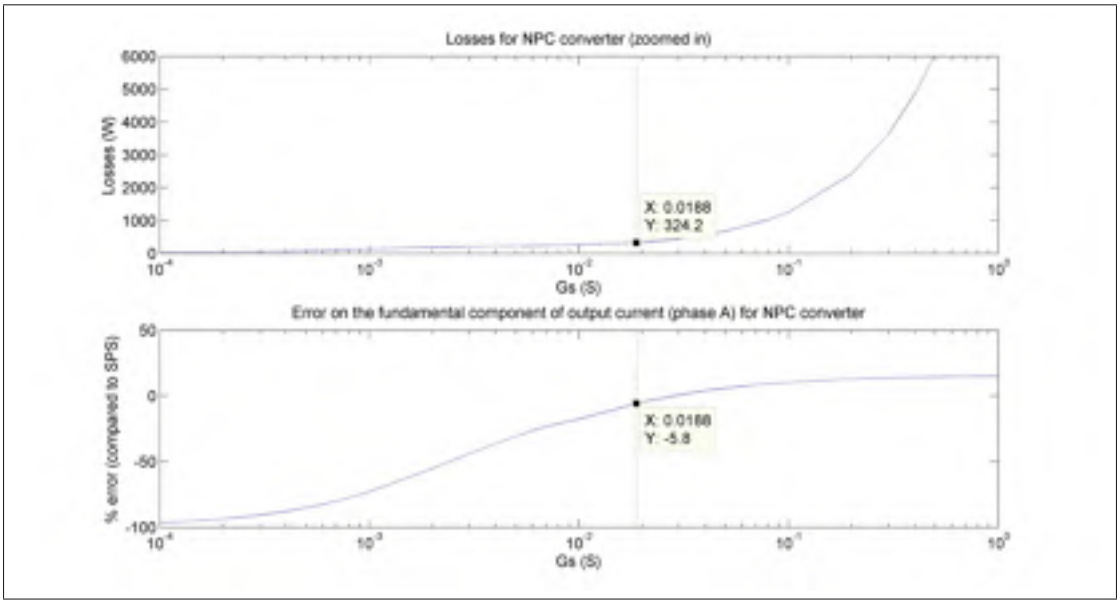

Figure 4.11 Losses and error on the fundamental current for different Gs values for a three level NPC inverter

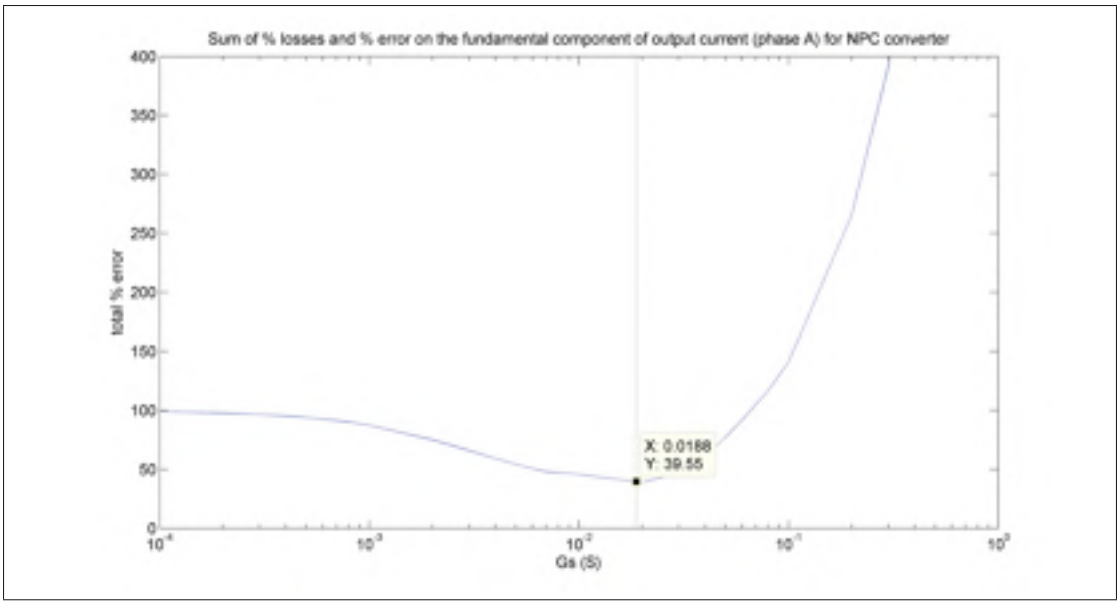

Figure 4.12 Sum of % losses and % error on the fundamental output current for different G_s values for a three level NPC inverter

4.2.3.1 Waveforms at optimal Gs

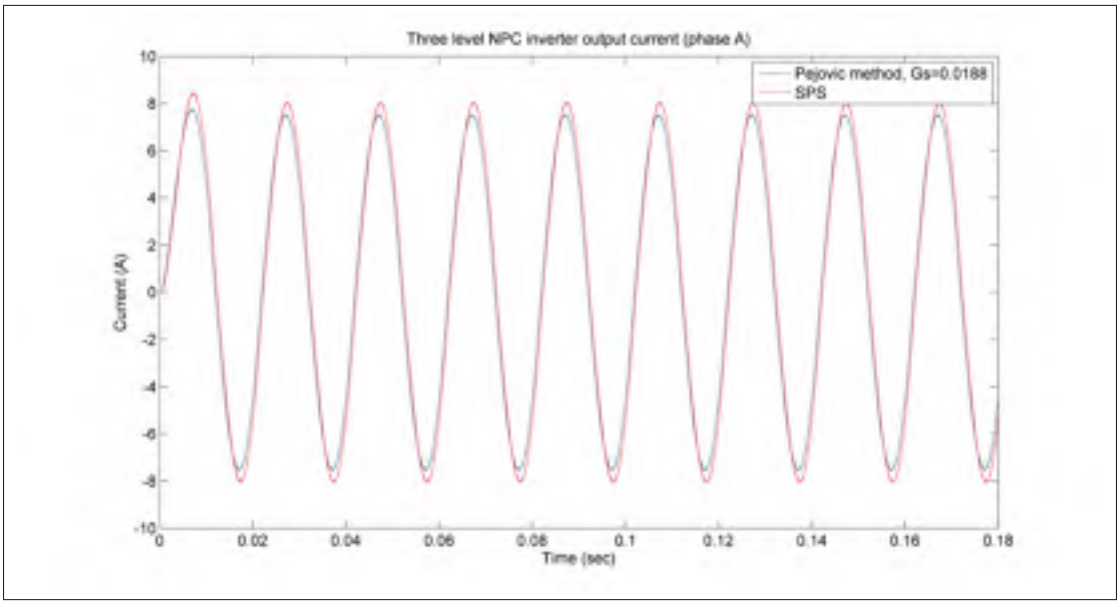

Figure 4.13 Iout waveform

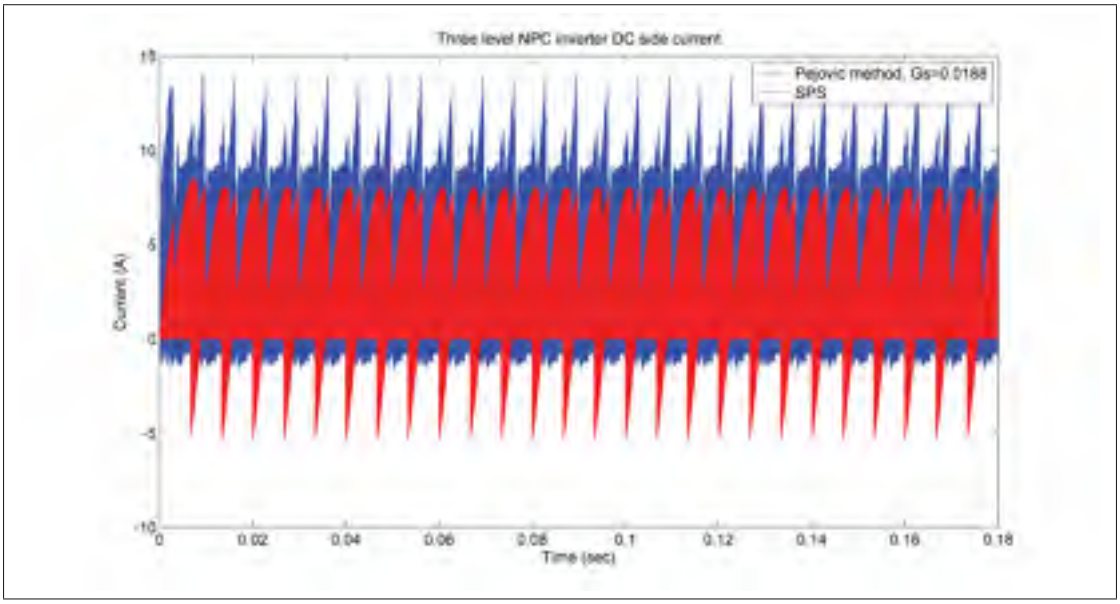

Figure 4.14 DC side current waveform

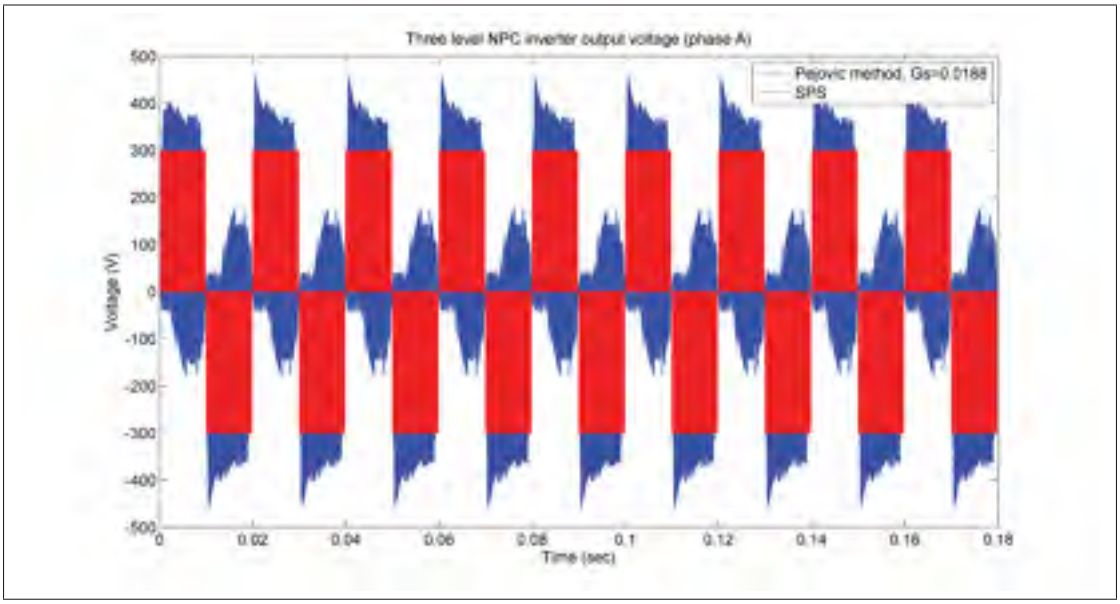

Figure 4.15 Vout waveform

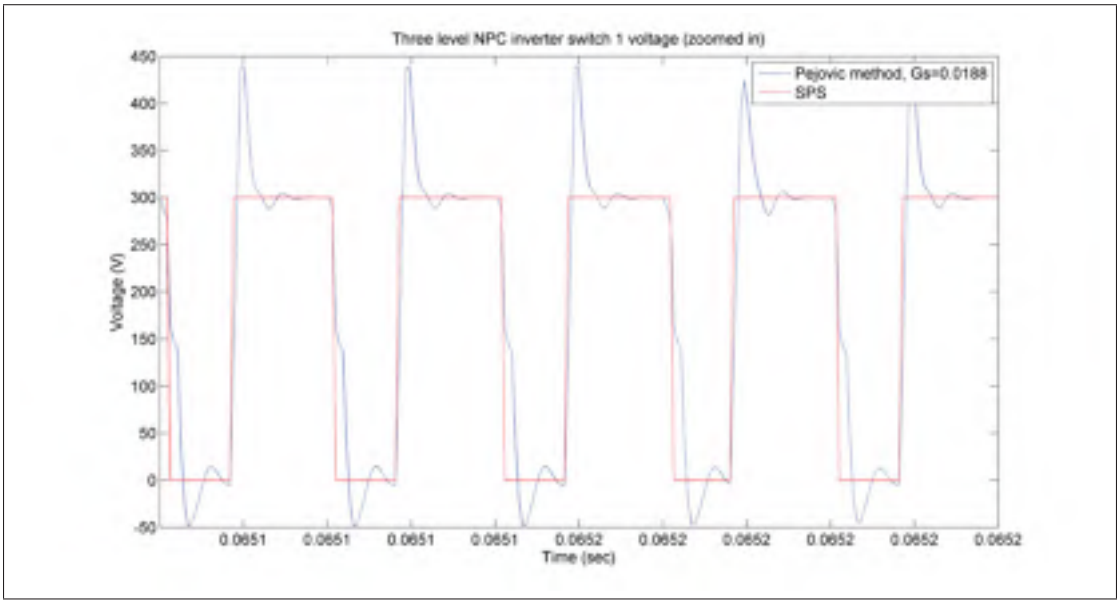

Figure 4.16 v_{sw_1} waveform

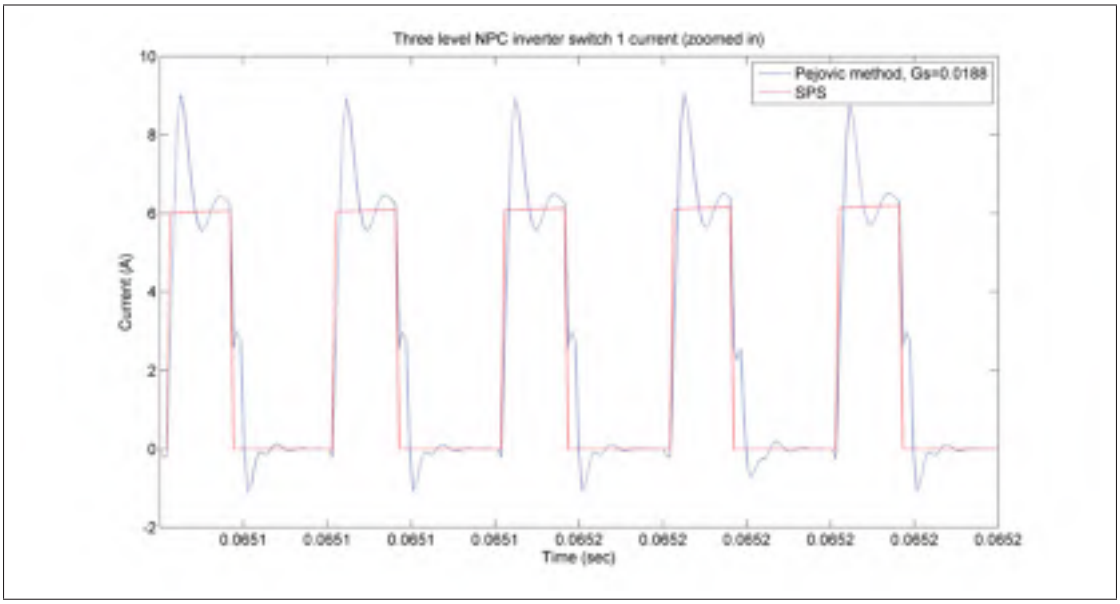

Figure 4.17 i_{sw_1} waveform

4.2.4 Direct matrix converter

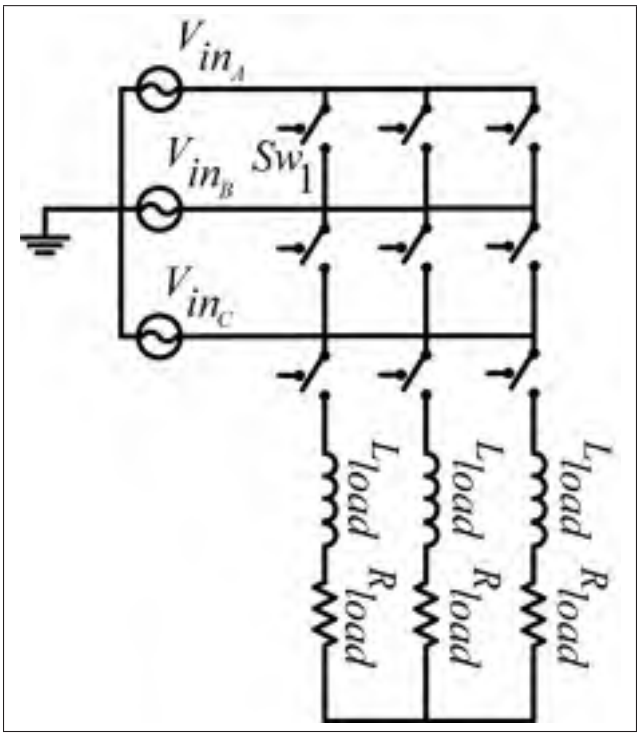

Figure 4.18 Schematic of matrix converter

Table 4.4 Direct matrix converter parameter values

Parameters	Values
f_{sw}	14 kHz
$V_{in\phi}$	100 V_{pk}
f_{grid}	50 Hz
f_{ref}	150 Hz
R_{on}	1 mΩ
deadtime	500 ns
modulation index	0.86
R_{load}	2 Ω
L_{load}	1 mH
T_s	210 ns

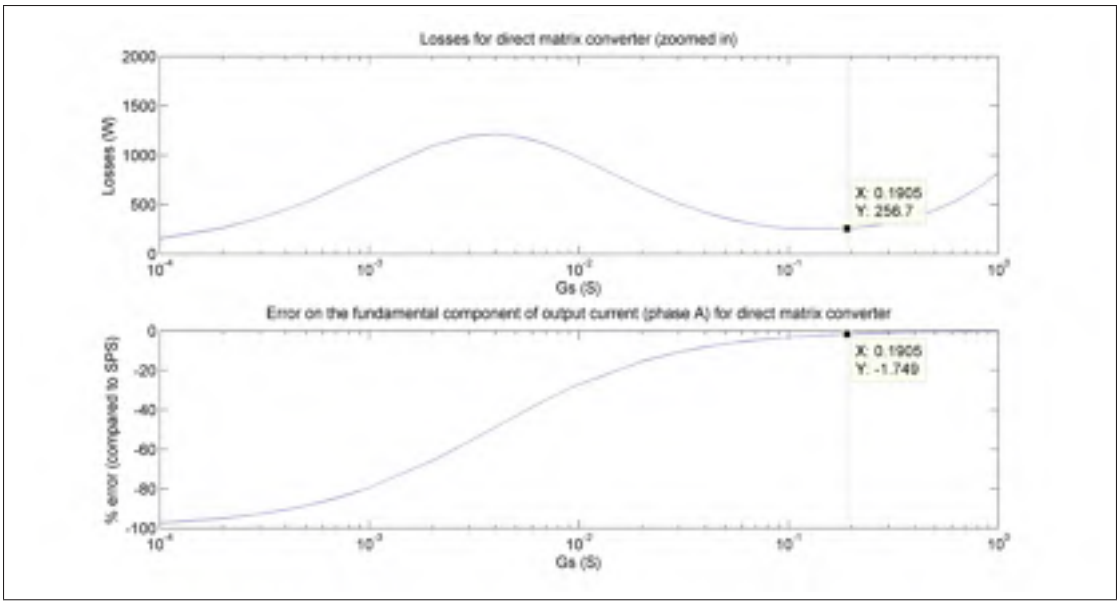

Figure 4.19 Losses and error on the fundamental current for different Gs values for a direct matrix converter

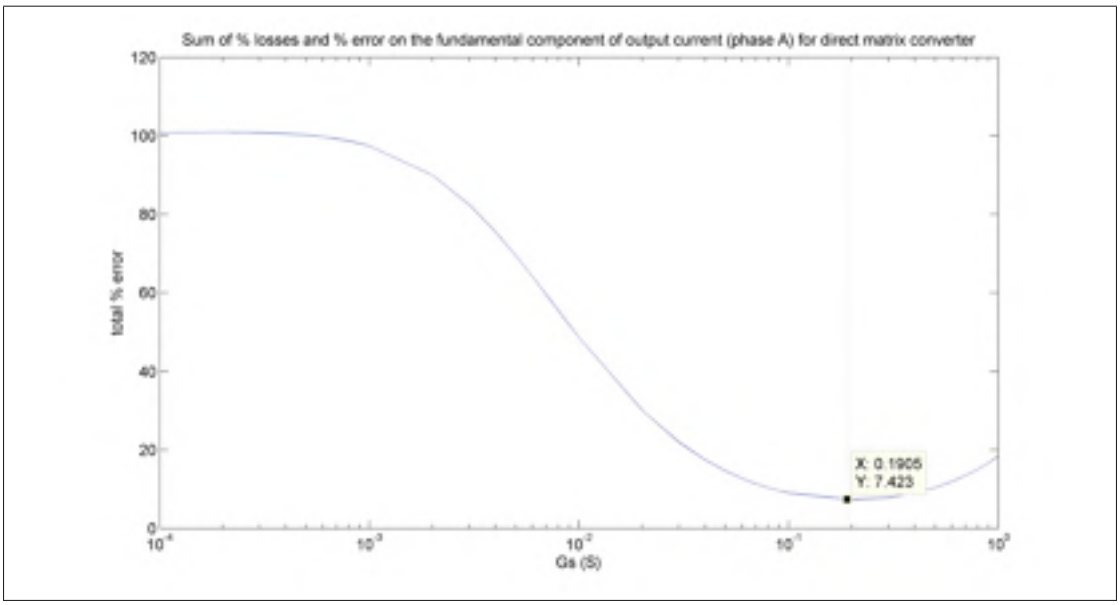

Figure 4.20 Sum of % losses and % error on the fundamental output current for different Gs values for a direct matrix converter

The final converter the automatic optimization algorithm was validated on was the direct matrix converter shown in figure 4.18. After performing a benchmark SPS simulation and running the

automatic optimization algorithm, the following values were calculated: $I_{SQ} = 1.2085 \times 10^6 A^2$. $V_{SQ} = 3.3285 \times 10^7 V^2$, $G_s = 0.1905S$. The optimal G_s obtained automatically is compared to the simulation results for various G_s values shown in figures 4.19 and 4.20.

For the case of the direct matrix converter, the general shape of the loss and current error curves are very similar to those of the two level inverter. For low values of G_s, the losses are small, for large values of G_s, the losses are large and in between there is a local minimum. By adding the (absolute value of) the percent error on i_{load_1} with the percent loss, a concave curve with a global minimum is obtained as shown in figure 4.20. The G_s calculated using the automatic optimization algorithm once again occurs very near this minimum.

It is encouraging to note that even though the rigorous analysis of the optimization method was performed for a two level inverter, the performance of the algorithm appears to generalize well to other converter topologies.

4.2.4.1 Waveforms at optimal Gs

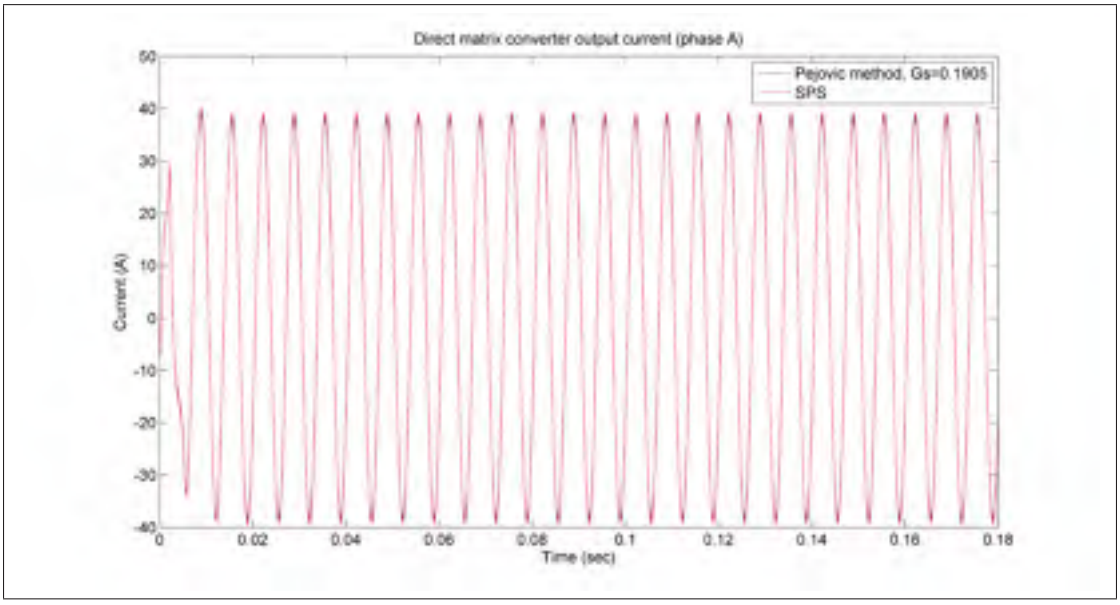

Figure 4.21 I_{out} waveform

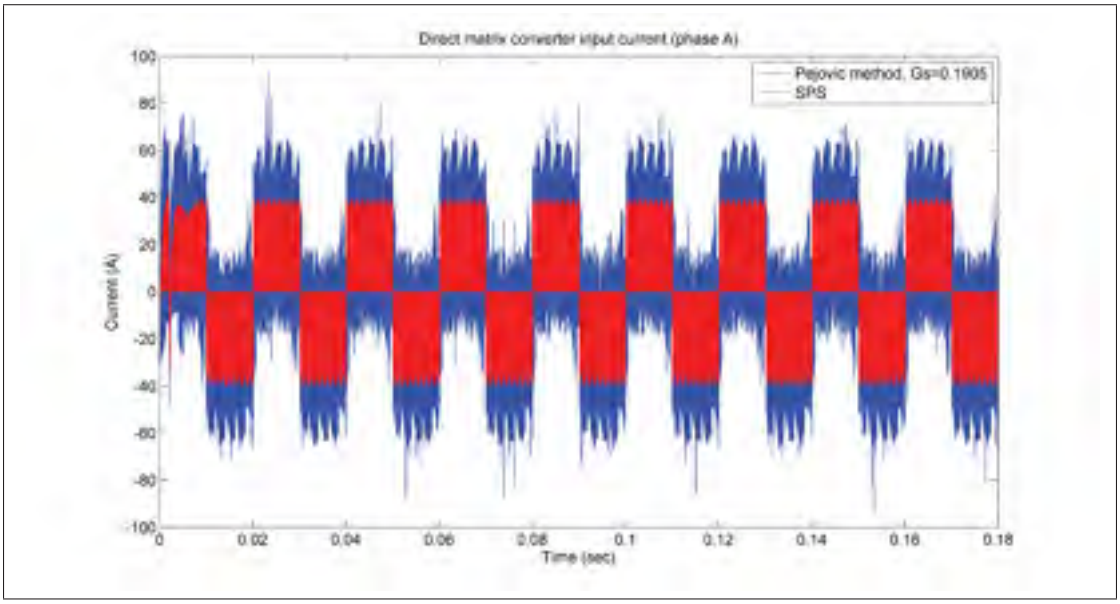

Figure 4.22 I_{in} waveform

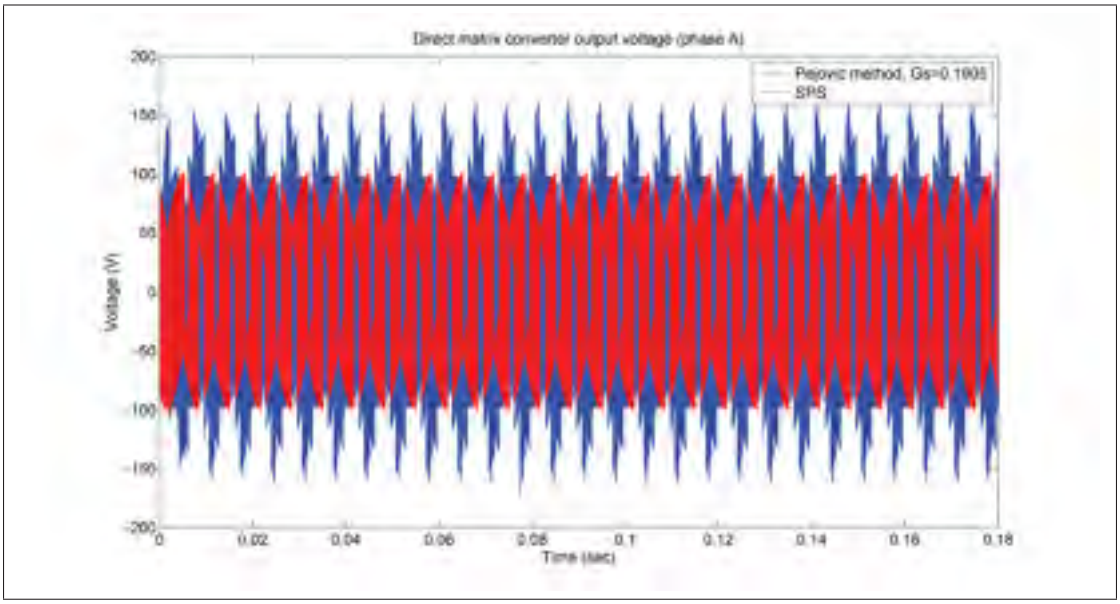

Figure 4.23 V_{out} waveform

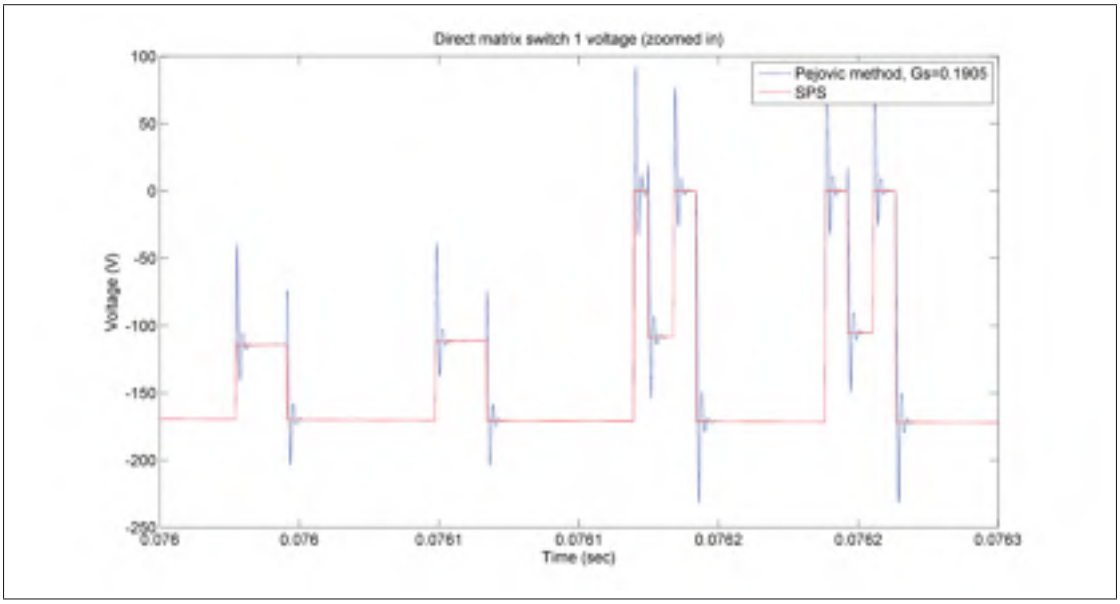

Figure 4.24 v_{sw_1} waveform

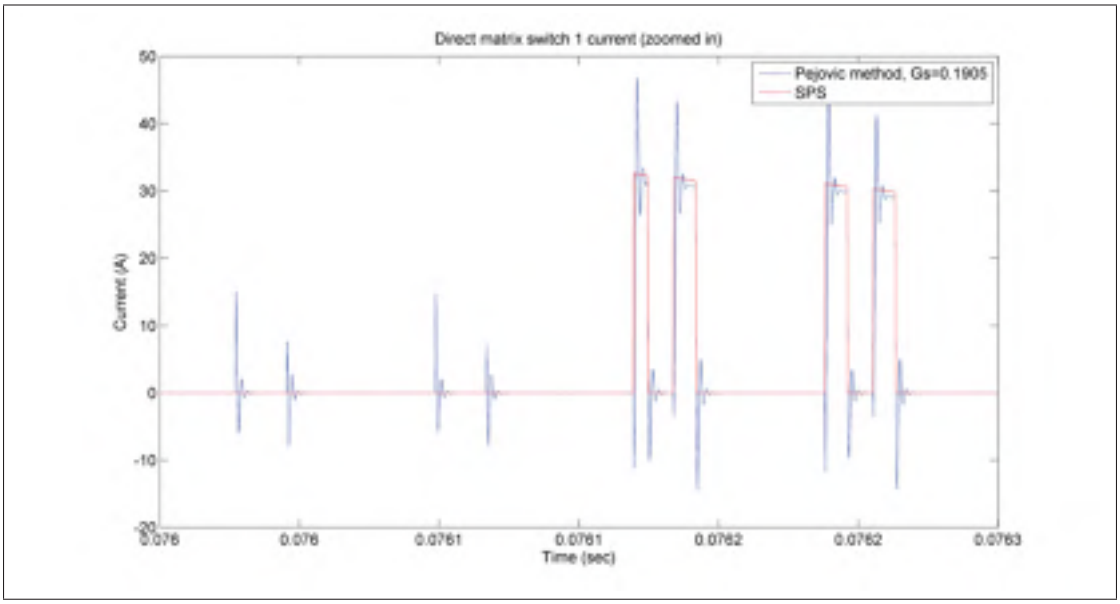

Figure 4.25 i_{sw_1} waveform

4.3 Conclusion

In this chapter, a method to automatically find the optimal value of the G_s parameter was derived. It is based on minimizing the commutation losses, and it was also shown that it is effective at minimizing the switch voltage and current errors following a commutation. In the case of the two level inverter, it was shown that the proposed algorithm reduces to an analytical expression. Finally, the proposed algorithm was validated for three topologies since the G_s it calculated was very close to the G_s that minimized the sum of the commutation losses and output current error.

CHAPTER 5

HYBRID FIXED-VARIABLE ADMITTANCE MATRIX METHOD

As the results in the previous chapter indicate, the proposed algorithm is effective in determining the optimal G_s for a given topology. However, even with an optimal G_s there are still some major issues with the fixed admittance matrix approach. Overshoots persist, artificial losses remain present, increasing with the switching frequency, and the optimal G_s varies with the operating point of the converter. Therefore, depending on the application, the fixed admittance matrix method may not produce accurate enough simulations and other research paths should be explored.

In this chapter, a method is proposed to reduce the amount of storage required when precomputing the inverses of a variable admittance matrix approach. This method has been validated offline for a boost and an indirect matrix converter. However, it remains to be seen if the proposed method can be implemented on an FPGA with a small enough time step and further research is required.

5.1 The Sherman-Morrison-Woodbury identity

Recall, that precomputing the inverses of a variable admittance matrix is not feasible for larger converter due to the limited on-board FPGA. Indeed, for an indirect matrix converter (having 12 switches) modelled with the method proposed by Blanchette *et al.* (2012), the size of the system matrix is 27x27. Therefore $2^{12} = 4096$ matrices of size 27x27 must be stored which is prohibitively large. That being said, whenever a commutation occurs, the entire matrix does not change. Indeed, if c switches change, then only c columns change as well. The Sherman-Morrison-Woodbury identity explained by (Gentle, 1998, p.110) can be very useful in this situation. The identity states that if some change UV^T is added to the original matrix A, then the inverse of this new matrix can be obtained as follows:

$$(A^{'})^{-1} = (A + UV^T)^{-1} = A^{-1} - A^{-1}U(I + V^T A^{-1}U)^{-1}V^T A^{-1} \tag{5.1}$$

This can be rewritten as:

$$(A^{'})^{-1} = (A + UV^T)^{-1} = A^{-1} - A^{-1}UR^{-1}V^TA^{-1} \tag{5.2}$$

where:

$$R = (I + V^TA^{-1}U) \tag{5.3}$$

Therefore, by precomputing the inverse of the original matrix A, if only c columns change with respect to A matrix during the simulation, then instead of inverting a matrix of dimension A, the matrix R with a dimension of only c can be inverted instead. Sudha *et al.* (1993) used this identity to limit the maximum dimension of the matrix to invert at each time step to the number switches in the converter. However, real time inversion, even if the dimension of the matrix to invert is reduced, is not feasible.

5.2 Explanation of the proposed method

In the method proposed in this chapter, online inversion is avoided by inverting the different possible R^{-1} matrices offline and storing them on the FPGA. Since the dimension of R^{-1} is significantly smaller than A^{-1} this results in a large reduction in required space. This will be analyzed in depth further on. All that needs to be done online is to use the appropriate R^{-1} matrix to reconstruct the required A^{-1} matrix corresponding to the current switch state. This is illustrated in the following example: Consider some initial matrix A, whose inverse is precomputed:

$$A = \begin{bmatrix} 1 & 11 & 17 & 12 \\ 2 & 3 & -4 & -3 \\ 6 & 6 & 7 & 23 \\ 12 & 34 & 2 & 0 \end{bmatrix} \tag{5.4}$$

Now consider that some change UV^T is applied to A such that

$$A^{'} = A + UV^T = \begin{bmatrix} 1 & 11 & 17 & 14 \\ 2 & 4 & -4 & -1 \\ 6 & 6 & 7 & 23 \\ 12 & 34 & 2 & 0 \end{bmatrix} \tag{5.5}$$

Columns 2 and 4 have changed. In order to determine the inverse of $A^{'}$, the U and V matrices must first be formed. This can be done using the following method. For each column j of $A^{'}$, if column j of $A^{'}$ is different from column j of A, then insert the difference between the two columns in the U matrix and add a column of zeros to V with a non zero entry at row j. In this example,

$$U = \begin{bmatrix} 0 & 2 \\ 1 & 2 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{5.6}$$

$$V = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \tag{5.7}$$

The next step is to calculate R^{-1} by obtaining R using 5.3 and inverting it.

$$R^{-1} = \begin{bmatrix} 0.5913 & -1.0699 \\ -0.3157 & 0.2132 \end{bmatrix} \tag{5.8}$$

As expected, it is only a 2x2 matrix since only two columns are different between A and $A^{'}$ The previous steps can be performed offline for all switch state possibilities and all possible R^{-1} matrices can be stored in memory. During the course of the simulation, whenever a commutation occurs, U and V must be formed using the method described previously, and the

appropriate R^{-1} matrix must be fetched from memory. Then the $(A^{'})^{-1}$ matrix corresponding to the current switch state can be reconstructed using equation 5.2. It remains to be seen whether these steps can be performed in a timely fashion on FPGA. If so, this method is very promising since it requires much less memory than the traditional approach of precomputing all the inverses of the variable admittance matrix.

5.3 Required memory calculation

If the inverse of the A matrix when all switches are OFF (denoted as A_0^{-1}) and the inverse of the A matrix when all switches are ON (denoted as A_1^{-1}) are precomputed and stored, then the dimension of the R^{-1} matrices for all other switch states is significantly reduced. The dimensions of the matrices to precompute and store and shown in table 5.1.

The maximum dimension of the matrices that must be precomputed (excluding A_0^{-1} and A_1^{-1}) is floor($\frac{N}{2}$). This is due to the fact that as the number of columns that change with respect to A_0^{-1} increases, the number of columns that change with respect to A_1^{-1} decreases. The required dimension of the matrix to store is equal to the minimum of those two numbers. For each matrix dimension from 1 to floor($\frac{N}{2}$) there will be a certain number of matrices that must be precomputed. The number of matrices to precomputed for a given dimension d is equal to:

$$2\binom{N}{d} \tag{5.9}$$

where $\binom{N}{d}$ is the binomial coefficient which can be rewritten as:

$$\frac{N!}{d!(N-d)!} \tag{5.10}$$

Equation 5.9 can be understood by considering an example circuit with 8 switches. There are $\binom{8}{1}$ combinations of switch states with only switch ON and $\binom{8}{1}$ combinations of switch states with only switch OFF. There are therefore $2\binom{8}{1}$ matrices with only one switch ON or OFF. Since A_0^{-1} and A_1^{-1} are precomputed, there are $2\binom{8}{1}$ 1x1 matrices to precompute. Similarly, there are $2\binom{8}{2}$ 2x2 matrices and $2\binom{8}{3}$ 3x3 matrices to precompute. However, there are only $\binom{8}{4}$

Table 5.1 Precomputed matrix dimensions

Switch states				Size of matrix to precompute	Comments
sw_1	sw_2	sw_3	sw_4		
0	0	0	0	dim(A)	The inverse of the A matrix with all switches OFF (A_0^{-1}) is precomputed in its entirety.
0	0	0	1	1	Only one column changed with respect to A_0^{-1}
0	0	1	0	1	
0	0	1	1	2	
0	1	0	0	1	
0	1	0	1	2	
0	1	1	0	2	
0	1	1	1	1	In this case three columns changed with respect to A_0^{-1} but only one changed with respect to A_1^{-1}
1	0	0	0	1	
1	0	0	1	2	
1	0	1	0	2	
1	0	1	1	1	
1	1	0	0	2	
1	1	0	1	1	
1	1	1	0	1	
1	1	1	1	dim(A)	The inverse of the A matrix with all switches ON (A_1^{-1}) is precomputed in its entirety.

4x4 matrices, not $2\binom{8}{4}$. This is due to the fact the set of combinations where 4 switches are ON is identical to the set of combinations where 4 switches are OFF since there are 8 switches in total. More formally, equation 5.9 is valid for dimensions of 1 to floor($\frac{N}{2}$) if N is odd, but only for dimensions of 1 to $\frac{N}{2}-1$ if N is even. In this case, the number of matrices of dimension $\frac{N}{2}$ is

$$\binom{N}{\frac{N}{2}}. \tag{5.11}$$

The number of matrix entries that must be stored is equal to:

$$2(\dim(A))^2 + \sum_{d}^{\text{floor}(\frac{N}{2})} 2d^2 \binom{N}{d} \tag{5.12}$$

if N is odd or:

$$2(\dim(A))^2 + \sum_{d}^{\frac{N}{2}-1} 2d^2 \binom{N}{d} + (\frac{N}{2})^2 \binom{N}{\frac{N}{2}} \tag{5.13}$$

if N is even. The space required to store the precomputed matrices is obtained by simply multiplying the number of entries obtained using 5.12 or 5.13 by the number of bits required to store one entry.

An indirect matrix converter formed using the method proposed by Blanchette *et al.* (2012) has twelve switches and a system matrix of dimension 27. It is assumed that each matrix entry occupies 32 bits. By using the proposed hybrid fixed-variable admittance matrix approach, only 378,696 bytes of storage are required, compared to 11,943,936 bytes using the traditional precomputed approach. The amount of space required is reduced by a significant 96.8%. Therefore, a possible research path is to assess whether this method can be implemented on an FPGA with a small time step.

CONCLUSION

In this work, the accuracy of the fixed admittance matrix approach was studied in detail. The analysis performed by Pejovic and Maksimovic (1994) was augmented and the transient voltage and current errors following commutation was quantified in greater detail. In addition, the artificial energy losses due to the method were quantified. This analysis was used to develop an algorithm to automatically find the optimal value of G_s. For the cases studied (two level inverter, three level NPC inverter, direct matrix converter), it was found that the algorithm calculated the optimal G_s with respect the switching losses and the output current error without having to resort to a lengthy trial and error process. There is, however, still much work to done to improve the real time simulation of power converters. A method was proposed to reduced the space required to store the inverses of a variable admittance matrix method. However, more research is required to determine whether it is a viable solution for real time simulation. Other possible research paths could be to attempt to improve the Pejovic method or to explore new hardware.

BIBLIOGRAPHY

Bachir, Tarek-Ould, Jean-Pierre David, Christian Dufour, and Jean Belanger. Nov 2010. "Effective FPGA-based electric motor modeling with floating-point cores". In *IECON 2010 - 36th Annual Conference on IEEE Industrial Electronics Society*. p. 829-834.

Belanger, J., P. Venne, and J.-N. Paquini. July 2010. "The What, Where and Why of Real-Time Simulation". In *PES IEEE, general meeting*. p. 37-49.

Belanger, J., A. Yamane, A. Yen, S. Cense, and P.-Y. Robert. Nov 2013. "Validation of eHS FPGA reconfigurable low-latency electric and power electronic circuit solver". In *Industrial Electronics Society, IECON 2013 - 39th Annual Conference of the IEEE*. p. 5418-5423.

Blanchette, Handy-Fortin, Tarek Ould-Bachir, and Jean-Pierre Davidi. 2012. "A State-Space Modeling Approach for the FPGA-based Real-Time Simulation of High Switching Frequency Power Converters". *IEEE Transactions on Industrial Electronics*, vol. 59, n° 12, p. 4555-4567.

Cellier, François E. and Ernesto Kofman, 4 2006. *Continuous System Simulation*. ed. 2006. Springer.

Dohi, K., Y. Hatanaka, K. Negi, Y. Shibata, and K. Oguri. Aug 2012. "Deep-pipelined FPGA implementation of ellipse estimation for eye tracking". In *Field Programmable Logic and Applications (FPL), 2012 22nd International Conference on*. p. 458-463.

Gentle, James E., 8 1998. *Numerical Linear Algebra for Applications in Statistics (Statistics and Computing)*. ed. 1998. Springer.

Gole, A.M., A. Keri, C. Kwankpa, E.W. Gunther, H.W. Dommel, I. Hassan, J.R. Marti, J.A. Martinez, K.G. Fehrle, L. Tang, M.F. McGranaghan, O.B. Nayak, P.F. Ribeiro, R. Iravani, and R. Lasseter. Jan 1997. "Guidelines for modeling power electronics in electric power engineering applications". *Power Delivery, IEEE Transactions on*, vol. 12, n° 1, p. 505-514.

Gregoire, Luc-Andre, Jean Belanger, and Wei Li. June 2011. "FPGA-based real-time simulation of multilevel modular converter HVDC systems". In *ELECTRIMACS 2011*. p. 1119-1125.

Jalili-Marandi, Vahid and Venkata Dinavahi. July 2009. "Large-scale transient stability simulation on graphics processing units". In *Power Energy Society General Meeting, 2009. PES '09. IEEE*. p. 1-6.

Lustig, Daniel and Margaret Martonosi. Feb 2013. "Reducing GPU offload latency via fine-grained CPU-GPU synchronization". In *High Performance Computer Architecture (HPCA2013), 2013 IEEE 19th International Symposium on*. p. 354-365.

Mahapatra, C., S. Mahboob, V.C.M. Leung, and T. Stouraitis. Dec 2012. " Fast Inverse Square Root Based Matrix Inverse for MIMO-LTE Systems ". In *Control Engineering and Communication Technology (ICCECT), 2012 International Conference on*. p. 321-324.

Myaing, Aung and V. Dinavahi. Jan 2011. " FPGA-Based Real-Time Emulation of Power Electronic Systems With Detailed Representation of Device Characteristics ". *Industrial Electronics, IEEE Transactions on*, vol. 58, n° 1, p. 358-368.

Najm, Farid N., 2 2010. *Circuit Simulation*. ed. 1. Wiley-IEEE Press.

Pejovic, Predrag and Dragan Maksimovic. 1994. " A method for fast time-domain simulation of networks with switches ". *IEEE Transactions on Power Electronics*, vol. 9, n° 4, p. 449-456.

Sudha, S. A, A Chandrasekaran, and V. Rajagopalan. Mar 1993. " New approach to switch modelling in the analysis of power electronic systems ". *Electric Power Applications, IEE Proceedings B*, vol. 140, n° 2, p. 115-123.