Universidad Zaragoza
1542

Proyecto Fin de Carrera

Aplicación didáctica para el estudio de topologías y técnicas de control clásicas dedicadas a la conversión DC-AC en electrónica de potencia

Autor

Diego Salas Forns

Director

D. Francisco José Pérez Cebolla

ANEXOS
(2/2)

Escuela de Ingeniería y Arquitectura

Septiembre 2014

ANNEX I

WORKING PRINCIPLE OF INVERTER CIRCUITS. STUDY OF CLASSICAL TOPOLOGIES AND CONTROL TECHNIQUES DEDICATED TO DC-AC TRANSFORMATION IN POWER ELECTRONIC.

Author

Diego Salas Forns

Director

D. Francisco José Pérez Cebolla

INDEX

INDEX OF TABLES

INDEX OF FIGURES

1 INTRODUCTION

This project consists of the study and simulation of converter circuits (DC/AC) more usual, called Inverter Circuits.

The objective of this project is to create an educational application focused for new engineering students who begin to study the world of inverters.

It will create simulations of all usual inverter circuits in two computer programs:

- Simulink (MATLAB)
- PSpice

After being created these simulations in which students can access them and, changing the parameters he wants, to observe and study the behavior of every inverter, it's to create a GUI (such as a computer program) where the student can change parameters and observe the outputs (both numerical and graphical) of each inverter circuit.

Mainly they can study and observe the following concepts:

- Different control techniques (which can modify the parameters).
- Modify the input parameters (input voltage, simulation time, frequency circuit and load values).
- The outputs of the load. The waveforms of the voltages and currents in the load.
- The numerical values of voltage and current, including harmonic distortion (THD).

This project will be developed as follows:

First we will briefly describe the principles of working of the inverters and the most important theoretical concepts for students to remember them.

After a detailed study of each invertir that has been chosen for this project is done.

Later it will develop the operation of the "MATLAB" program and the simulation tool "Simulink". Then it will develop in a graphic way all inverter circuits simulated, as well as more representative graphs of each of them.

Then it will develop as well as in the previous paragraph but with the other simulation program "PSpice".

And finally, before exposing which are the findings of this project, explains how, and in which programming code has been programmed the aforementioned GUI. Including all the programming code used, explaining each command that is what it means and why it has been used.

2 WORKING PRINCIPLE OF INVERTERS

To start with the study of different kind of inverters fed by voltage source (DC) it starts with the simplest of them, the single-phase half-bridge inverter with square wave modulation.

Later it develops in more detail the working of this type of inverter, but now it will provide us the basis for the proper understanding of all the control methods that will be studied after.

2.1 WORKING PRINCIPLE OF A SIMPLE INVERTER

The following figure (Figure 1) shows this inverter. Must know that in this type of inverter, the number of switches (power semiconductors we consider ideal switches) are two, and arrows indicate the only possible direction of current flow. These divide the voltage of the DC source into two parts by capacitors [2].

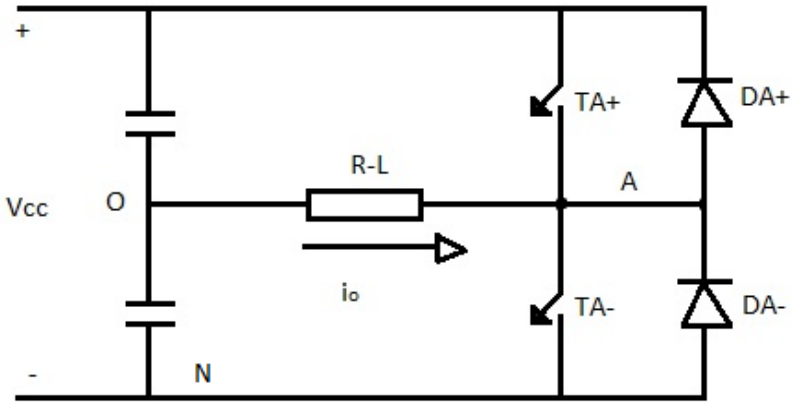


Figure 1 Single-Phase Half-Bridge Inverter.

If: T_{A+} is ON, during T/2 (180º) → $V_{A0} = \frac{Vcc}{2}$

T_{A-} is ON, during the other T/2 (180º) → $V_{A0} = -\frac{Vcc}{2}$

The following figure (Figure 2) shows the waveforms of the voltages on the ideal switches, as well as the output load:

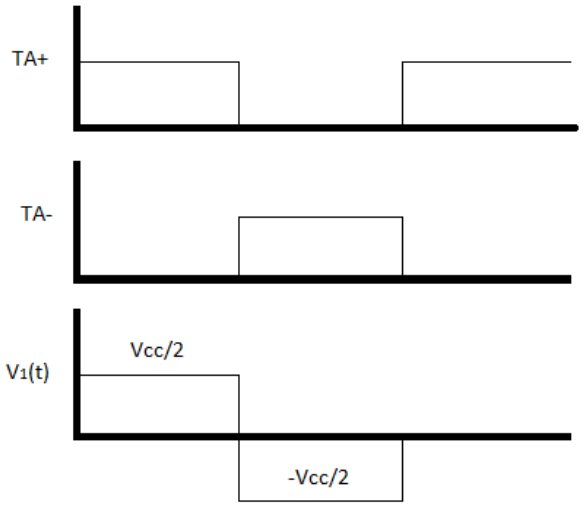


Figure 2 Waveforms of the transistors of Single-Phase Half-Bridge Inverter.

T_{A+} and T_{A-} don't have to be working at the same time, in order not to cause a short circuit. Therefore, for real applications, it will be necessary to apply a time-out or "blanking time" between: connecting the first switch and the second switch off.

If the load is only resistive:

- Diodes positioned in antiparallel with the switches never drive, because when the voltage applied to the load is positive, the current is positive, and when the voltage is negative, the current is negative. Always being the voltage and current in phase.

If the load is ohmic-inductive:

- Diodes positioned in antiparallel with the switches would drive because the voltage and current would be out of phase each other.

The semiconductor has to withstand a reverse voltage equal to Vcc. The voltage source inverter is needed for this to be three jacks or be symmetrical.

2.2 IMPORTANT THEORETICAL CONCEPTS

The following concepts are well known, but we will remember the equations for each of them [3-4]:

RMS Voltage (effective value):

$$V_{A0} = \sqrt{\frac{2}{T}} \sqrt{\int_0^{\frac{T}{2}} \frac{V_{CC}^{\ 2}}{4} dt} = \frac{V_{CC}}{2} \qquad (2.1)$$

Amplitude of the fundamental component:

$$\left(\hat{V}_{A0}\right)_1 = \frac{4}{\pi}\frac{V_{CC}}{2} = 1{,}273\left(\frac{V_{CC}}{2}\right) \tag{2.2}$$

Amplitude of the harmonics:

$$\left(\hat{V}_{A0}\right)_h = \frac{\left(\hat{V}_{A0}\right)_1}{h} \tag{2.3}$$

V_{A0} voltage can be expressed as a Fourier series:

$$V_{A0} = \sum_{h=1,3,5,\ldots}^{\infty} \frac{4}{h\cdot\pi}\frac{V_{CC}}{2} sen\left(h\cdot\omega\cdot t\right) \tag{2.4}$$

When the load is made up of resistance and inductance in series, the driving time of the switches vary from 90º to 180º, being fixed its value for the phase angle of the load.

The current flowing through a load formed by resistance and inductance, can be obtained from the following expression:

$$i_O = \sum_{h=1,3,5,\ldots}^{\infty} \frac{4}{h\cdot\pi}\frac{V_{CC}}{2}\frac{1}{\sqrt{R^2 + (f\cdot\omega\cdot L)^2}} sen\left(h\cdot\omega\cdot L - \vartheta_h\right) \tag{2.5}$$

And:

$$\vartheta_h = arctg\left(\frac{\left(h\cdot\omega\cdot L\right)}{R}\right) \tag{2.6}$$

If we denote I_{01}, to the effective value of the fundamental component of the current flowing through the load, the power consumed, due to that component (h=1), can be calculated as:

$$P_{O1} = V_{O1}\cdot I_{O1}\cdot\cos(\vartheta_1) = {I_{01}}^2\cdot R = \left[\frac{1}{\sqrt{2}}\frac{4}{h\cdot\pi}\frac{V_{CC}}{2}\frac{1}{\sqrt{R^2\cdot(h\cdot\omega\cdot L)^2}}\right]^2\cdot R \tag{2.7}$$

2.3 ANALYSIS BY FOURIER SERIES

To analyze the load current and calculate the power absorbed by it, the method of Fourier series (and above is used if the load is more complex than a single charge (R-L) is used [2-4].

It's convenient to express the voltage and current of the load in terms of a Fourier series. If there is no DC component in the output, we have:

$$V_O(t) = \sum_{h=1}^{\infty} V_h \cdot sen(h \cdot \omega \cdot t + \phi_h) \quad (2.8)$$

And

$$I_O(t) = \sum_{h=1}^{\infty} I_h \cdot sen(h \cdot \omega \cdot t + \phi_h) \quad (2.9)$$

The power absorbed by the load with a series resistance is calculated from $Irms^2 \cdot R$, where the rms current can be determined from the corresponding rms currents given to each component of the Fourier series:

$$I_{rms} = \sqrt{\sum_{h=1}^{\infty} {I_h}^2, rms} = \sqrt{\sum_{h=1}^{\infty} \left(\frac{I_h}{\sqrt{2}}\right)^2} \quad (2.10)$$

With:

$$I_h = \frac{V_h}{Z_h} \quad \textbf{(2.11)}$$

And *Zn* is the impedance of the load for harmonic *n*.

Similarly, can be determined absorbed power in the load resistance for each frequency in Fourier series. The total power is determined from:

$$P = \sum_{n=1}^{\infty} P_n = R \cdot \sum_{n=1}^{\infty} {I_{h,rms}}^2 \quad (2.12)$$

And: $I_{n,rms}$ is $I_n / \sqrt{2}$.

In the case of a square wave, the Fourier series contain odd harmonics, and can be represented as:

$$V_0(t) = \sum_{h,impares}^{\infty} \frac{4V_{CC}}{h\pi}(sen(h \cdot \omega \cdot t)) \quad (2.13)$$

2.4 TOTAL HARMONIC DISTORTION

In general, for all converters is very important to know and control the quality of the voltage or current outputs alternating [3-4].

The quality of a non-sinusoidal wave can be expressed in terms of factor DAT or THD. Assuming no DC component in the output, we have:

$$THD = \frac{\sqrt{\sum_{h=2}^{\infty}(V_{h,rms})^2}}{V_{1,rms}} = \frac{\sqrt{V_{rms}{}^2 - V_{1,rms}{}^2}}{V_{1,rms}} \quad (2.14)$$

The THD of the current is calculated by substituting the current and voltage in the previous equation.

The THD of the load current is usually of more interest than the output voltage. This definition of the THD factor is based on the Fourier series, so there is an advantage to the use the method of Fourier series in the analysis when you have to calculate the THD factor. Other measures of distortion, as the distortion factor can also be applied to describe the waveforms output inverters.

3 SELECTION AND THEORETICAL STUDY OF INVERTERS

3.1 SELECTION OF INVERTERS FOR THIS PROJECT

After making a study of all kinds of inverter circuits that there are at present, from classics to the latest discoveries, and considering the didactic purpose of this project is for students newly introduced to the world of inverter circuits, it was decided to conduct the study and simulation of the inverter circuits following [3-4]:

1- Half-Wave Inverter (Square Wave Control)

2- Half-Wave Inverter (PWM Control)

3- Asymmetric Inverter (Square Wave Control)

4- Asymmetric Inverter (PWM Control)

5- Full Wave H-Bridge Inverter (Square Wave Control)

6- Full Wave H-Bridge Inverter (Voltage Harmonic Cancellation)

7- Full Wave H-Bridge Inverter (Bipolar PWM Control)

8- Full Wave H-Bridge Inverter (Single-Pole PWM Control)

9- Three-Phase VSI 6-Step Inverter [Star Load] (Square Wave Control)

10- Three-Phase VSI 6-Step Inverter [Triangle Load] (Square Wave Control)

11- Three-Phase VSI 6-Step Inverter [Star Load] (PWM Control)

12- Three-Phase VSI 6-Step Inverter [Triangle Load] (PWM Control)

13- Three-Phase VSI 6-Step Inverter [Star Load] (Hysteresis Control)

14- Push-Pull Inverter (Square Wave Control)

15- Push-Pull Inverter (PWM Control)

16- Cascade Full Bridge Single-Phase Inverter (2 Levels)

17- Cascade Full Bridge Single-Phase Inverter (4 Levels)

18- Cascade Full Bridge Three-Phase Inverter [Star Load] (3 Levels)

19- NPC Inverter

20- NPC Inverter (5 Levels)

21- NPC Three-Phase Inverter [Star Load]

22- FC Inverter (5 Levels)

23- FC Three-Phase Inverter [Star Load] (3 Levels)

As can be seen, it begins with the simplest inverters circut, and increases the difficulty of each of them. The last types of inverters are called Multilevel Inverters.

3.2 THEORETICAL STUDY OF EACH INVERTER

This section will explain in detail each of the inverter circuits that have been studied and simulated throughout this project [2-4].

3.2.1 HALF-WAVE INVERTERS

3.2.1.1 HALF-WAVE INVERTER (SQUARE WAVE CONTROL)

Next figure (Figure 3) shows the schematic for this type of inverter. The modulation technique used makes the switches are controlled (TA +) and (TA-), with a delay between them of 180 º.

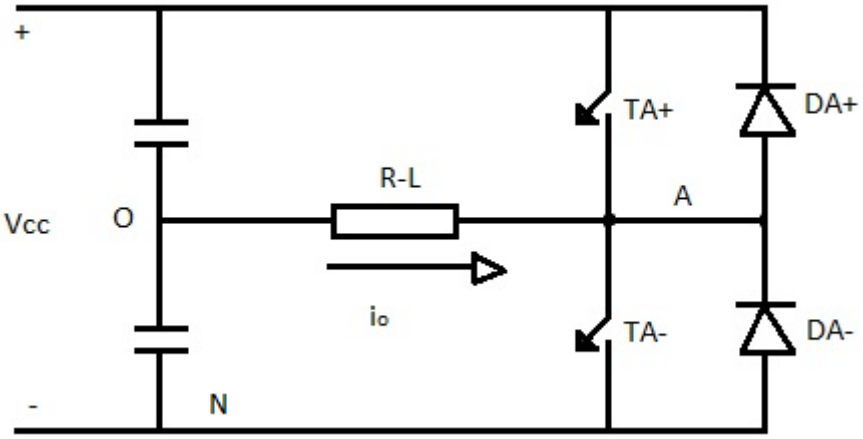


Figure 3 Single-Phase Half-Bridge Inverter.

When:

$$\text{(TA+) is ON} \rightarrow V_{A0} = \frac{Vcc}{2}$$

$$\text{(TA-) is ON} \rightarrow V_{A0} = -\frac{Vcc}{2}$$

In the Figure 4 are represented the most representative wave forms for a resistive load.

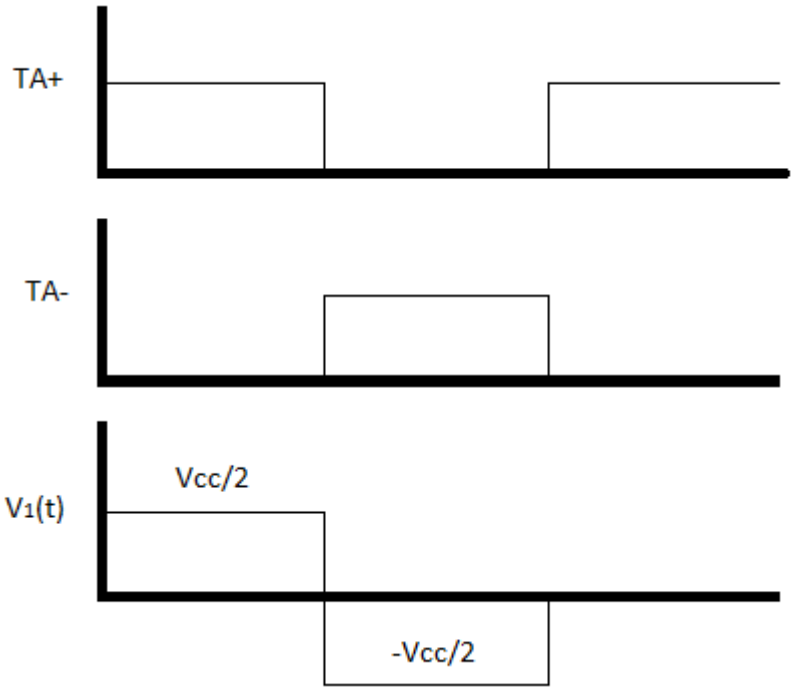


Figure 4 Waveforms of the transistors of Single-Phase Half-Bridge Inverter (1).

The following figure (Figure5) shows the current through the transistors:

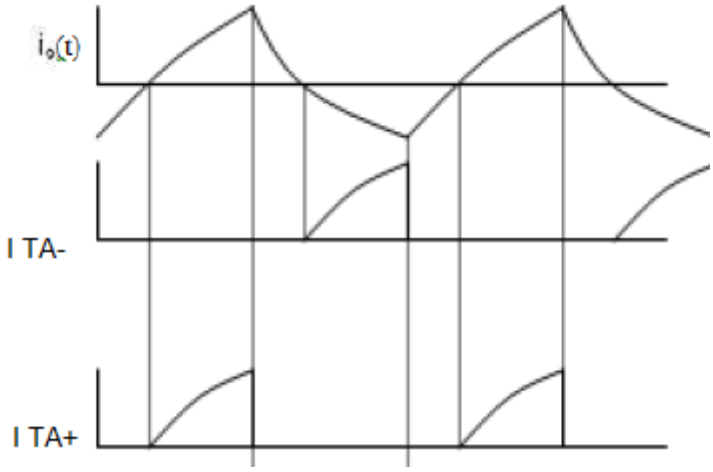


Figure 5 Waveforms of the transistors of Single-Phase Half-Bridge Inverter (2).

To calculate the current absorbed by the load, must first know the Fourier series expansion of a square wave form:

$$V_{(t)} = \sum_{n=1,3,5...}^{\infty} \frac{2 \cdot Vcc}{n \cdot \pi} \cdot \sin(n \cdot \omega \cdot t) \quad (3.1) \qquad V_1 = \frac{Vp1}{\sqrt{2}} = \frac{2 \cdot Vcc}{\sqrt{2} \cdot n} = 0.45 \cdot Vcc \quad (3.2)$$

$$V = \sqrt{V_1^2 + V_2^2 + ... + V_n^2} \quad (3.3) \qquad V_1 = \sqrt{\frac{2}{T} \int_0^{\pi/2} \frac{Vcc}{2} \cdot dt} = 0.5 \cdot Vcc \quad (3.4)$$

Current:

$$I_0(t) = \frac{V_o(t)}{Z} \quad (3.5) \qquad Z_n = \sqrt{R^2 + (n \cdot L \cdot \omega)^2} \quad (3.6) \qquad \varphi_n = arctg \frac{n \cdot L \cdot \omega}{R} \quad (3.7)$$

$$I_{o(t)} = \sum_{k=1,3,5...}^{\infty} \frac{2 \cdot Vcc}{n \cdot \pi \cdot \sqrt{R^2 + (n \cdot L \cdot \omega)^2}} \cdot \sin(n \cdot \omega \cdot t - \varphi_n) \quad (3.8)$$

$$I_1 = \frac{I_{p1}}{\sqrt{2}} = \sum_{n=1,3,5...}^{\infty} \frac{2 \cdot Vcc}{\sqrt{2} \cdot \pi \cdot \sqrt{R^2 + (L \cdot \omega)^2}} \quad (3.9)$$

$$I = \sqrt{I_1^2 + I_2^2 + ... + I_n^2} \quad (3.10) \qquad I_n = \frac{I_{pn}}{\sqrt{2}} = \sum_{n=1,3,5...}^{\infty} \frac{2 \cdot Vcc}{\sqrt{2} \cdot n \cdot \pi \cdot \sqrt{R^2 + (n \cdot L \cdot \omega)^2}} \quad (3.11)$$

Power consumed by the load:

$$P = \sum_{n=1,3,5...}^{\infty} V_n \cdot I_n \cdot \cos(\varphi_n) = \sum_{n=1,3,5...}^{\infty} \frac{2 \cdot Vcc^2}{n^2 \cdot \pi^2} \cdot \frac{1}{\sqrt{R^2 + (n \cdot L \cdot \omega)^2}} \cdot \cos(\varphi_n) \quad (3.12)$$

Only the first terms of the series are important because for high numbers of the harmonic amplitude of the Fourier component of voltage decreases and the impedances increases. This produces a small current for these harmonics.

From now on, we assume that we have an inverter circuit with the following values:

Vcc = 48v, R = 2Ω, L = 5mH, f = 50Hz

By performing all calculations, these results are obtained (Table 1):

n	fn (hz)	Vn (V)	Zn (Ω)	In (A)	Pn (W)
1	50	30,55	2,54	12,01	144,28
3	150	10,18	5,12	1,99	3,96
5	250	6,11	8,10	0,75	0,57
7	350	4,36	11,18	0,39	0,15
9	450	3,39	14,28	0,24	0,06

Table 1 Coefficients of the Fourier series for Single-Phase Half-Bridge Inverter.

To obtain the THD factor of the voltage and load current, it is used the Fourier series for a square wave and the equation defining the THD factor, shown in section 2.4. The rms value of a square wave voltage is equal to the peak value, and the component of the fundamental frequency is the first term being then:

$$V_{rms} = V_{cc} \qquad V_{1,\,rms} = \frac{V_1}{\sqrt{2}} = \frac{4V_{cc}}{\sqrt{2}\pi} \tag{3.13}$$

Voltage THD factor:

$$THDv = \frac{\sqrt{V_{rms}{}^2 - V_1{}^2,\,_{rms}}}{V_{1,\,rms}} = \frac{\sqrt{(\frac{10,18}{\sqrt{2}})^2 + (\frac{6,11}{\sqrt{2}})^2 + (\frac{4,36}{\sqrt{2}})^2 + (\frac{3,39}{\sqrt{2}})^2}}{(\frac{30,55}{\sqrt{2}})} = 0,4287 = 42,87\%$$

Current THD factor:

$$THDi = \frac{\sqrt{\sum_{n=2}^{\infty} (I_{n,\,rms})^2}}{I_{1,\,rms}} \approx \frac{\sqrt{(\frac{1,99}{\sqrt{2}})^2 + (\frac{0,75}{\sqrt{2}})^2 + (\frac{0,39}{\sqrt{2}})^2 + (\frac{0,24}{\sqrt{2}})^2}}{(\frac{12,01}{\sqrt{2}})} = 0,1811 = 18,11\%$$

In the Figure 6 shows the Fourier series, both the current (first graph) and the voltage (second graph) with the previous data and with Simulink (Matlab) graphs:

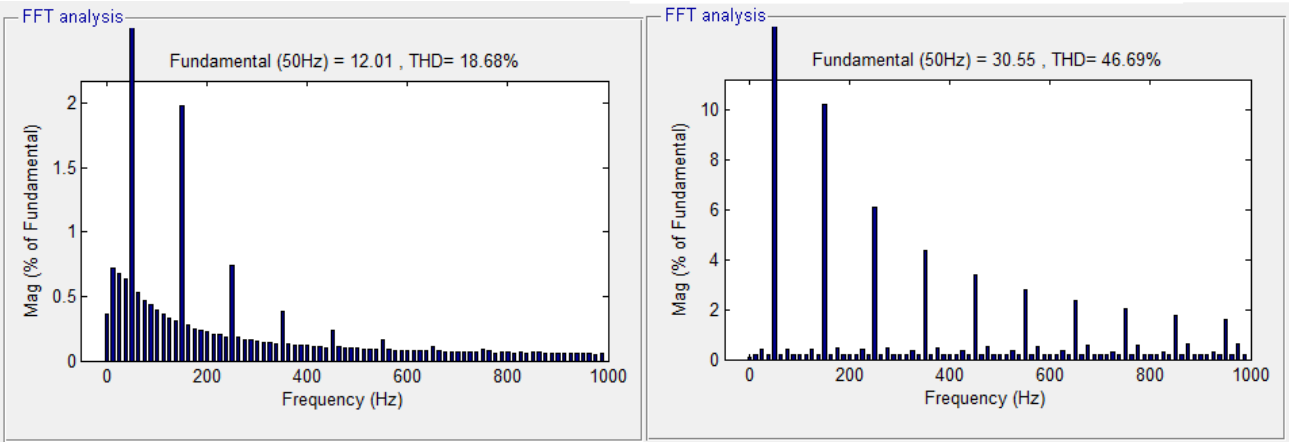


Figure 6 Frequency spectrum of the voltage and current of Single-Phase Half-Bridge Inverter (Square Wave Control).

3.2.1.2 HALF-WAVE INVERTER (PWM CONTROL)

The schematic of this inverter (Figure 7) is the same that "Half-wave inverter (Square Wave control)", but the control of this inverter is different: Pulse-Width Modulation **[2]**, **[6]**.

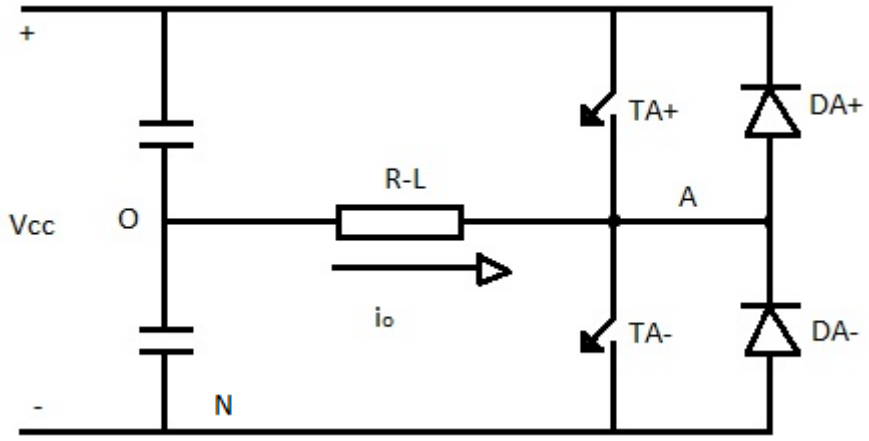


Figure 7 Single-Phase Half-Bridge Inverter.

When sinusoidal wave is higher tan triangular wave, the ouput load is +Vcc, and when is lower the opposite phenomenon occurs:

$$V_0 = \frac{+Vcc}{2} \qquad \text{when} \qquad V_{\sin} > V_{tri}$$

$$V_0 = -\frac{Vcc}{2} \qquad \text{when} \qquad V_{\sin} < V_{tri}$$

This version is bipolar PWM as the output takes alternating values between plus and minus the voltage of the DC source.

The switching scheme will implement the bipolar switching, for half-wave bridge inverter, is determined by comparing the instantaneous signals: reference and carrier, as seen in Figure 8, in the following manner:

T_{A+} is working when $V_{\sin} > V_{tri}$ $\left(V_0 = \frac{+Vcc}{2}\right)$

T_{A-} is working when $V_{\sin} < V_{tri}$ $\left(V_0 = -\frac{Vcc}{2}\right)$

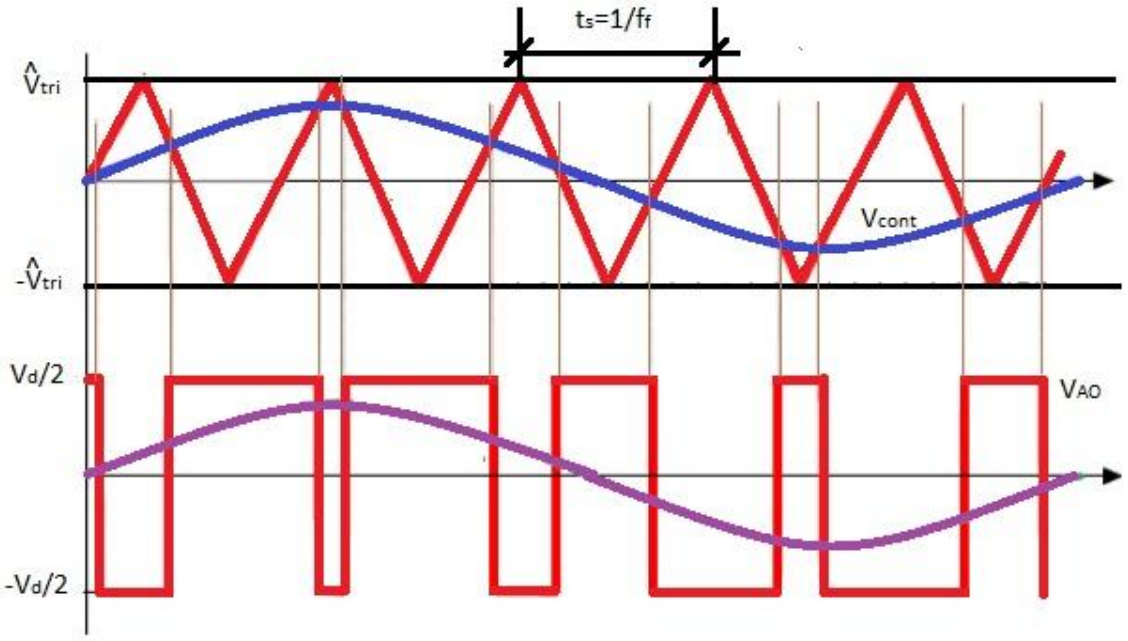


Figure 8 Waveform of Bipolar Width Modulated.

The Fourier series of bipolar PWM output shown in Figure 8 are calculated by examining each of the pulses. The triangular waveform is synchronized with the reference, as shown in Figure 8, and m_f is an odd integer that is chosen. Then the PWM output shows an odd symmetry, and can be expressed as the Fourier series:

$$V_0(t) = \sum_{n=1}^{\infty} \frac{V_n}{2}(Sen(n \cdot \omega_0 \cdot t))$$

For the k-th pulse of the PWM output in Figure 26, the Fourier coefficient is:

$$Vnk = \frac{2}{\pi}\int_0^T \frac{v(t)}{2} sen(n \cdot \omega_0 \cdot t) d(\omega_0 \cdot t) = \frac{1}{\pi}\left[\int_{\alpha k}^{\alpha k + \delta k} vcc \cdot sen(n \cdot \omega_0 \cdot t) d(\omega_0 \cdot t) + \int_{\alpha k + \delta k}^{\alpha k + 1} (-vcc) \cdot sen(n \cdot \omega_0 \cdot t) d(\omega_0 \cdot t)\right] \quad (3.14)$$

Calculating:

$$Vnk = \frac{Vcc}{n\pi}\left[\cos n\alpha_k + \cos n(\alpha_{k+1}) - 2\cos n(\alpha_k + \delta_k)\right] \quad (3.15)$$

Each V_n coefficient for the PWM waveform is the sum of V_{nk} for p pulses included in a period:

$$V_n = \sum_{k=1}^{p} V_{nk} \quad (3.16)$$

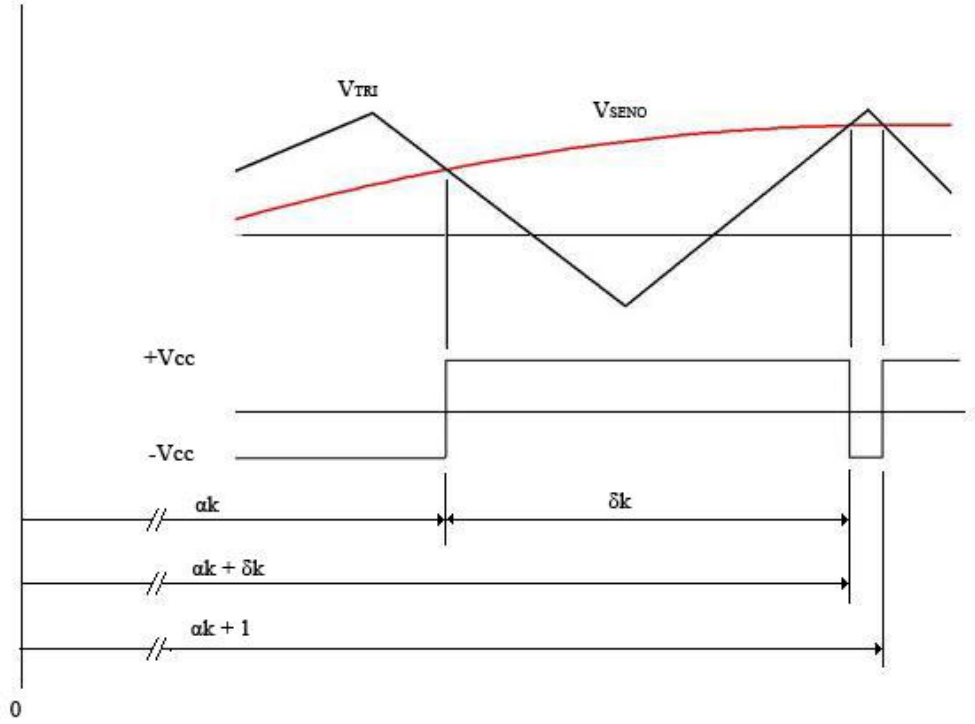


Figure 9 A PWM pulse to calculate the Fourier series for bipolar PWM.

The amplitudes of the harmonics is a function of m_a, because the width of each pulse depends on the relative amplitudes of the sinusoidal and triangular waves. The first harmonics in the output spectrum frequencies are around m_f. In the Table 2 are the first harmonics for bipolar PWM output. The Fourier coefficients are a function of m_f if m_f is high (≥ 9).

	m_a=1	**0,9**	**0,8**	**0,7**	**0,6**	**0,5**	**0,4**	**0,3**	**0,2**	**0,1**
n = 1	1	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
n = m_f	0,6	0,71	0,82	0,92	1,01	1,08	1,15	1,2	1,24	1,27
n = m_f ± 2	0,32	0,27	0,22	0,17	0,13	0,09	0,06	0,03	0,02	0

Table 2 Normalized Fourier coefficients V_n / V_{DC} for bipolar PWM.

Using the following data:

R = 2Ω, L = 5 mH, Vcc = 48 v, f = 50 Hz, m_a = 0,5 y m_f = 20 (f_{TRI} = (20) (50)=1.000Hz)

$$V_1 = m_a \cdot \frac{Vcc}{2} = 0{,}5 \cdot \frac{48}{2} = 12v$$

$$I_n = \frac{V_n}{Z_n} = \frac{V_n}{\sqrt{R^2 + (n \cdot \omega_0 L)^2}} \quad (3.17)$$

For the fundamental frequency:

$$I_n = \frac{12}{\sqrt{2^2 + (1 \cdot 2\pi 50 \cdot 0{,}005)^2}} = 4{,}72A$$

With m_f = 20, the first harmonics are held for n=20, 18 y 22. Using the table 2:

$$V_{20} = 1{,}08 \cdot \frac{48}{2} = 25{,}92v$$

$$V_{18} = V_{22} = 0{,}09 \cdot \frac{48}{2} = 2{,}16v$$

Currents corresponding to each of the harmonics are calculated from the equation (3.18).

To calculate the power of each frequency:

$$P_n = (I_{n,ef})2 \cdot R = (\frac{I_n}{\sqrt{2}})^2 R \quad (3.18)$$

By performing all calculations, these results are obtained (Table 3):

n	fn (hz)	Vn (V)	Zn (Ω)	In (A)	Pn (W)
1	50	12,00	2,54	4,72	22,27
18	900	2,16	28,34	0,08	0,01
20	1000	25,92	31,48	0,82	0,68
22	1100	2,16	34,62	0,06	0,00

Table 3 Coefficients of the Fourier series for Bipolar (PWM Control) Inverter.

Higher level harmonics provide little power, so are without considering.

Voltage THD factor:

$$THDv = \frac{\sqrt{{V_{rms}}^2 - {V_1}^2, _{rms}}}{V_{1,\,rms}} = \frac{\sqrt{(\frac{2{,}16}{\sqrt{2}})^2 + (\frac{25{,}92}{\sqrt{2}})^2 + (\frac{2{,}16}{\sqrt{2}})^2}}{(\frac{12}{\sqrt{2}})} = 2{,}1749 = 217{,}49\%$$

Current THD factor:

$$THDi = \frac{\sqrt{\sum_{n=2}^{\infty} (I_{n,\,rms})^2}}{I_{1,\,rms}} \approx \frac{\sqrt{(\frac{0{,}08}{\sqrt{2}})^2 + (\frac{0{,}82}{\sqrt{2}})^2 + (\frac{0{,}06}{\sqrt{2}})^2}}{(\frac{4{,}72}{\sqrt{2}})} = 0{,}1757 = 17{,}57\%$$

Waveforms of voltage and load current used for the example can be seen in Figure 10:

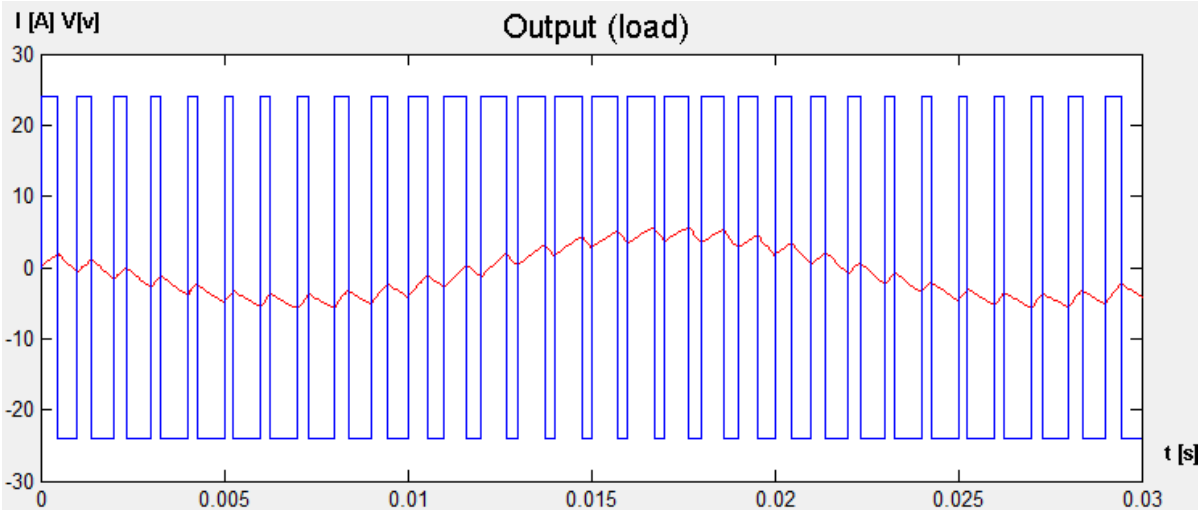


Figure 10 Voltage and current output of Bipolar (PWM Control) Inverter through the load.

In Figure 11 shows the Fourier series, both the current (first graph) and the voltage (second graph) with the previous data and with Simulink (Matlab) graphs:

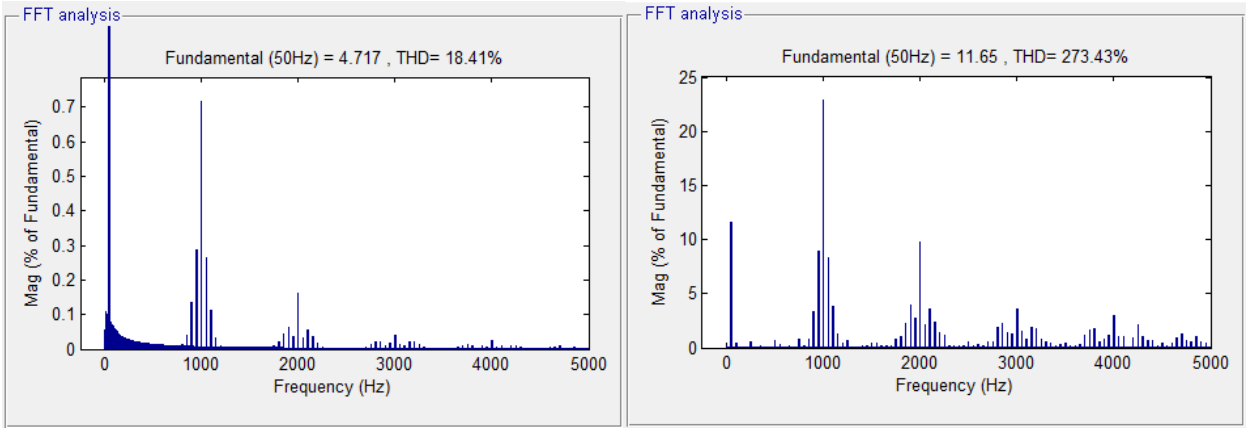


Figure 11 Frequency spectrum of the voltage and current of Half-Wave Inverter (PWM Control).

3.2.1.2.1 DEFINITIONS AND CONSIDERATIONS RELATING TO WIDTH PULSE MODULATION (PWM)

1º- Frequency modulation index m_f:

Fourier series of the PWM output voltage has a fundamental frequency that is the same as the reference signal frequency [6].

The values of some harmonics are sometimes higher than the Fundamental component. But as found at high frequencies with a low pass filter could be eliminated. Then we define Frequency modulation index:

$$m_f = \frac{f_{portadora}}{f_{referencia}} = \frac{f_{tri}}{f_{seno}} \tag{3.19}$$

If we increase the carrier frequency we are increasing the frequencies at which the harmonics are produced. This also produces higher losses in the inverter switches.

2º.-Amplitude modulation index m_a:

We define the index modulation amplitude m_a as:

$$m_a = \frac{V_{m,\,portadora}}{V_{m,\,referencia}} = \frac{V_{m,\,seno}}{V_{m,\,tri}} \tag{3.20}$$

The amplitude of the fundamental frequency of the output voltage is linearly proportional to m_a If $m_a \leq 1$:

$$V_1 = m_a \cdot Vcc \tag{3.21}$$

In this way m can be varied to change the amplitude of the output: If m_a is greater than 1, the output amplitude increases as the value of m_a, but not linearly.

I. With m_f and m_a we have to know some rules about output voltaje signal, harmonics, etc.

a) The amplitude of the fundamental harmonic of the output voltage is m_a times the input voltage. If we assume that m_f is high, we can assume, almost without error, the modulated output is constant in each cycle. The value of this output is + Vcc and-Vcc. The average value of the output voltage is:

$$V_{AB} = \frac{Vcc}{2}\left(\frac{T_{ON}}{T} - \frac{T_{Off}}{T}\right) = \frac{Vcc}{2}\left(\frac{T_{ON} - T_{Off}}{T}\right) \tag{3.22}$$

Therefore, in one cycle is demonstrated that:

$$\frac{T_{ON} - T_{Off}}{T} = \frac{V_{SEN}}{V_{TRI}} = m_a \tag{3.23}$$

Assuming that m_a <1, the only variable parameter from one cycle to another is the amplitude of the modulating wave, which is sinusoidal. Therefore we can rewrite the initial formulation for the first harmonic:

$$V_{an1} = \frac{V_{SEN}}{V_{TRI}} sen(\omega t) \cdot Vcc = m_a \cdot sen(\omega t) \cdot Vcc \tag{3.24}$$

We can say that the amplitude of the first harmonic output is m_a times the amplitude of the input voltage.

b) The harmonics of the output voltage appear as sidebands of the switching frequency and its multiples; this aspect is valid for values of m_f> 9, which can be taken as always true, except in exceptional cases of very high power. For the general case, we can say that the amplitude of the different harmonics is almost independent of the parameter m_f, and this only defines the frequency at which they appear, can be expressed such that the frequency of the different harmonics by the following expression:

$$f_s = \left(jm_f \pm k \right) f \tag{3.25}$$

f_s is the frequency of s order of harmonic corresponding to the k-sideband for j-times the rate of modulation.

For odd values of j, harmonics exist only for even values of k parameter; pairs of values for j, harmonics exist only for odd values of k.

In Table 4 the normalized amplitudes of the various harmonics are collected, depending on the modulation index m_f, are only represented those who have significant value to j = 4.

h/m_a	**0,2**	**0,4**	**0,6**	**0,8**	**1**
FUNDAMENTAL	**0,2**	**0,4**	**0,6**	**0,8**	**1**
m_f	1,242	1,15	1,006	0,818	0,601
$m_f \pm 2$	0,016	0,061	0,131	0,22	0,318
$m_f \pm 4$					0,018
$2m_f \pm 1$	0,19	0,326	0,37	0,314	0,181
$2m_f \pm 3$		0,024	0,071	0,139	0,212
$2m_f \pm 5$				0,013	0,033
$3m_f$	0,335	0,123	0,083	0,171	0,113
$3m_f \pm 2$	0,044	0,139	0,203	0,176	0,062
$3m_f \pm 4$		0,012	0,047	0,104	0,157
$3m_f \pm 6$				0,016	0,044
$4m_f \pm 1$	0,163	0,157	0,008	0,105	0,068
$4m_f \pm 3$	0,012	0,07	0,132	0,115	0,009
$4m_f \pm 5$			0,034	0,084	0,119
$4m_f \pm 7$				0,017	0,05

Table 4 Harmonic Normalized Table.

c) The m_f parameter must be an odd integer, so an odd symmetry is obtained in addition to a half-wave symmetry. Therefore, in the relation in the output voltage will exist only odd-order harmonics and disappear even-order harmonics. In the Fourier series expansion, there will be only sine terms.

II. With everything described so far in this section should take the following recommendations for the values of m_a and m_f. It should work with m_f values as high as possible, since the harmonics appear at high frequencies, making it easier to filter them, however, it should be in mind that the switching losses increase when raising the frequency. Taking into consideration the necessity to work outside the audible range, the frequency is usually chosen either above 20Khz or below 6kHz (in case of very high power), in order to avoid the frequencies in that range. Most authors set the value of 21 as the boundary for the value of this parameter can be considered high or low.
It can provide recommendations based on the value of this parameter (assuming m_a <1), taking the previous criterion:

$m_f < 21$:

- The sinusoidal and triangular signals must be synchronized, which necessarily requires that m_f is an integer value. The reason must be sought in the case that work with both unsynchronized signals, the output signal would have subharmonics, which is clearly undesirable. Therefore, if the output voltage should change its frequency, the triangular signal must also change.

- Must be an odd value, in order to take advantage of the symmetry of the waveform.

- The slopes of the signals V_{TRI} and V_{SIN} and must have opposite polarities and matching its zero crossing. This aspect is particularly important in the case of low values of m_f.

$m_f > 21$:

- The amplitudes of the subharmonics that can be generated by using asynchronous PWM are negligible. Therefore, if the value of m_f is high, it can set the frequency of the triangular signal and varying the frequency of the sinusoidal signal. However, if the load is a motor drive, it shouldn't be used and asynchronous mode, because although the low frequency harmonics are low amplitude, they can generate currents of high value, which are clearly undesirable.

$m_a > 1$:

- For the parameter m_a has been considered is always less than unity. If this parameter we is greater than unity, we are in the situation called overmodulation. In this situation, although the amplitude of the fundamental harmonic can be increased, the number of harmonics of the output is increased and they appear at frequencies lower. For this

operating system, it is recommended to work in a synchronized way. This situation must be avoided in uninterruptible power supplies, order to minimize possible distortion in the output voltage. However, it's common to use overmodulation. The input value is limited as follows:

$$m_a < \frac{4}{\pi}$$

For higher values of this parameter, the PWM concept is lost and degenerates into a square wave scheme.

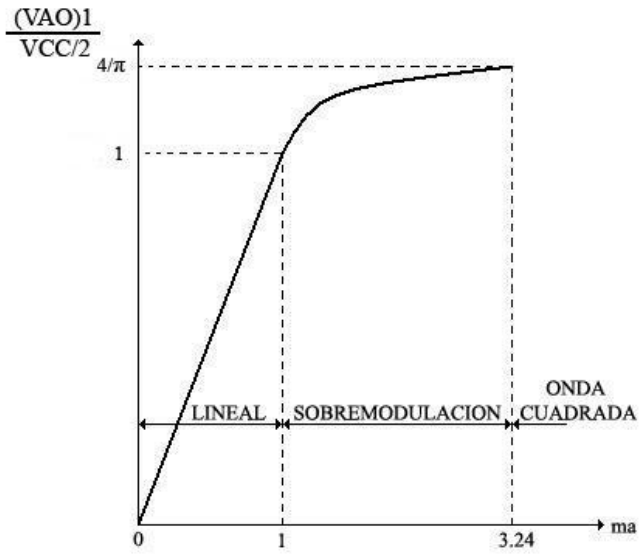


Figure 12 Peak value (normalized) of the fundamental role of m_a for m_a = 15.

3º.-Interruptores:

The switches on the circuit full wave bridge must be able to transport the current in either direction for pulse width modulation (PWM), like they do for operation with a square wave. Thus, feedback diodes are needed in the switching devices, as explained previously. Another consequence of using actual switches isn't open and close instantly. It's therefore necessary to consider the switching times in the control of the switches

4º.-Tensión de referencia:

Sinusoidal reference voltage must be generated within the control circuit of the inverter, or taken from an external reference. It might seem that the role of the inverter bridge is irrelevant, because there needs to be a sinusoidal voltage present before the bridge can generate a sinusoidal output. However, the reference signal requires very little power. The power supplied to the load comes from the source of continuous power, and this is the purpose pursued with the inverter. The reference

signal isn't restricted to a sinusoidal signal. The signal could be an audio signal, and the circuit full wave bridge could be used as PWM audio amplifier.

3.2.2 HALF-WAVE ASYMMETRIC INVERTERS

3.2.2.1 HALF-WAVE ASYMMETRIC INVERTER (SQUARE WAVE CONTROL)

The method to study and analyze this inverter is the same as has been used in "HALF-WAVE INVERTER (SQUARE WAVE CONTROL)". The only difference is in the output terminals, where the load is connected, as shown in the following Figure 13:

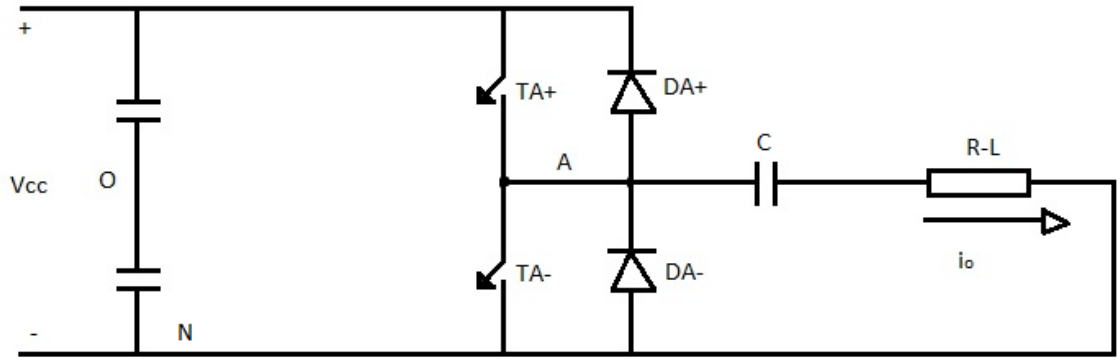


Figure 13 Half-Wave Asymmetric Inverter.

As seen in the scheme, the use of a capacitor is necessary because the DC component of the output voltage is eliminated by the series capacitor C.

In this way, It has been obtained the same table of values, the same formulations, the same THD's, and the same frequency Graph Fourier expansion.

Anyway, followings graphs (Figure 14) have been obtained by Simulink Matlab program with the simulation of this invertir (Half-wave asymmetric invertir):

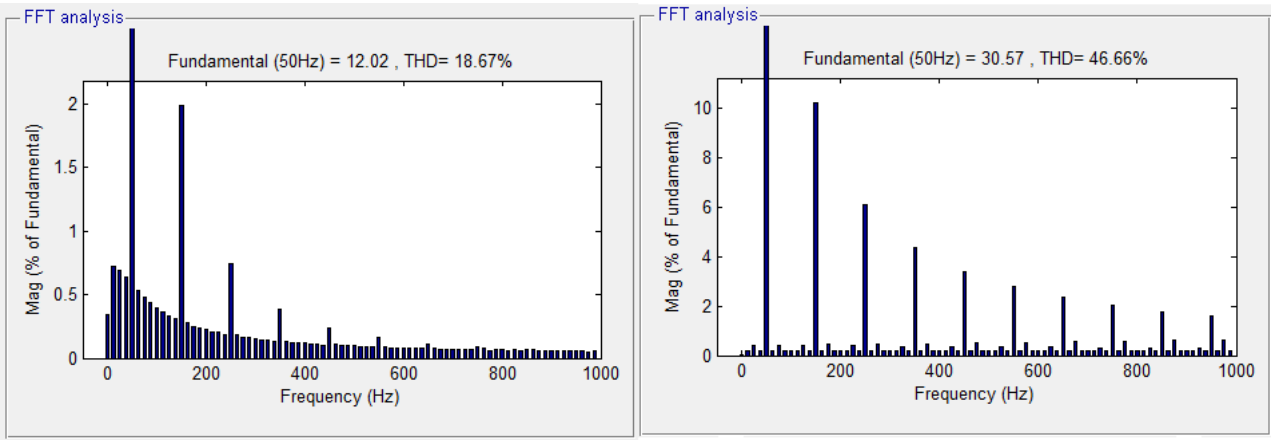


Figure 14 Frequency spectrum of the voltage and current of Half-Wave Asymmetric Inverter (Square Wave Control).

3.2.2.2 HALF-WAVE ASYMMETRIC INVERTER (PWM CONTROL)

The method to study and analyze this inverter is the same as has been used in "HALF-WAVE INVERTER (PWM CONTROL)". The only difference is in the output terminals, where the load is connected, as shown in the following Figure 15:

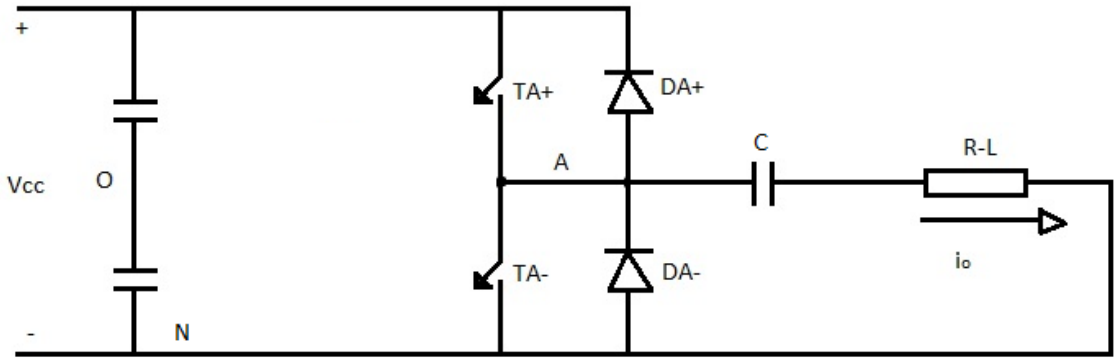


Figure 15 Half-Wave Asymmetric Inverter.

As seen in the scheme, the use of a capacitor is necessary because the DC component of the output voltage is eliminated by the series capacitor C.

In this way, It has been obtained the same table of values, the same formulations, the same THD's, and the same frequency Graph Fourier expansion.

Anyway, followings graphs (Figure 16) have been obtained by Simulink Matlab program with the simulation of this invertir (Half-wave asymmetric invertir):

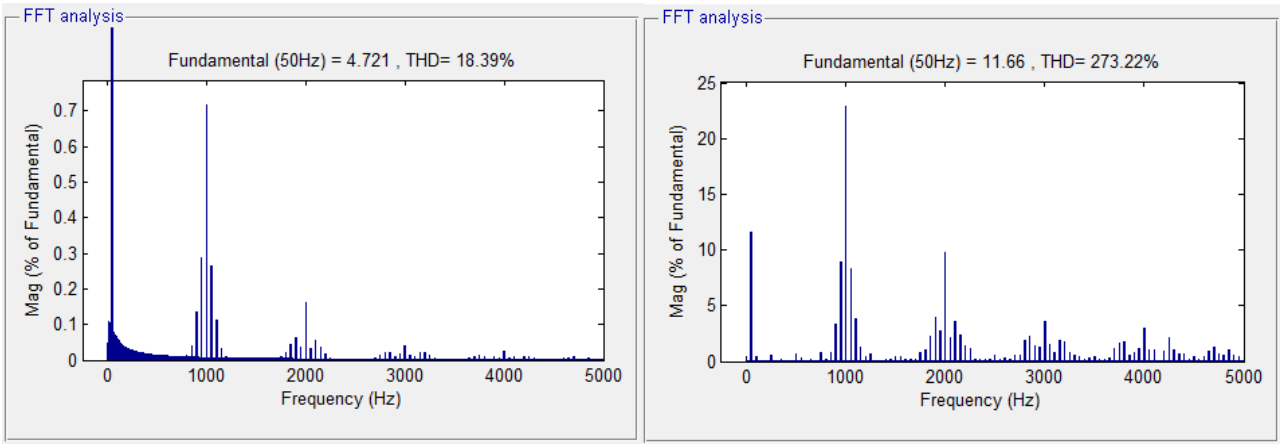


Figure 16 Frequency spectrum of the voltage and current of Half-Wave Asymmetric Inverter (PWM Control).

3.2.3 FULL WAVE H-BRIDGE INVERTERS

3.2.3.1 FULL WAVE H-BRIDGE INVERTER (SQUARE WAVE CONTROL)

Figure 17 shows the schematic for this type of inverter. The modulation technique used makes the switches are controlled in pairs (TA +, TB-) and (TA-, TB +), with a delay between them of 180 º.

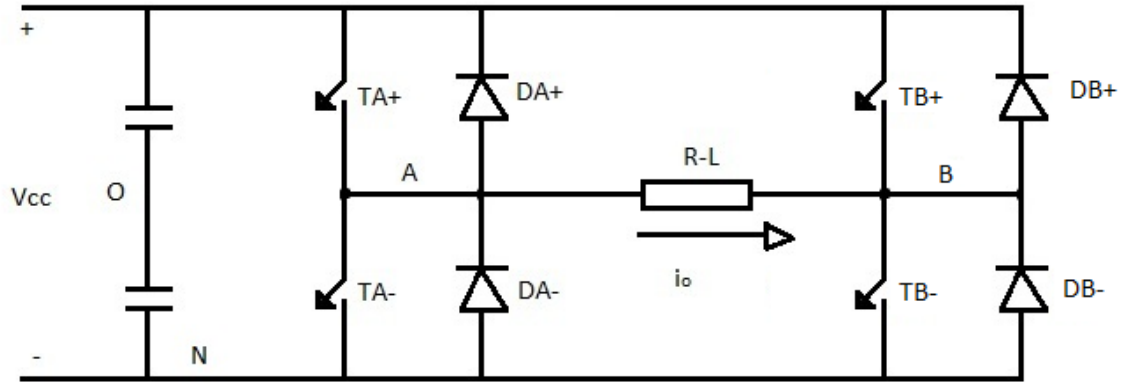


Figure 17 Full Wave H-Bridge Inverter.

When:

(TA+, TB-) are ON → $V_{A0} = \frac{Vcc}{2}$ y $V_{B0} = -\frac{Vcc}{2}$ => $V_{AB} = V_{A0} - V_{B0} = Vcc$

(TA-, TB+) are ON → $V_{A0} = -\frac{Vcc}{2}$ y $V_{B0} = \frac{Vcc}{2}$ => $V_{AB} = V_{A0} - V_{B0} = -Vcc$

In the figure are represented the most representative wave forms for a resistive load.

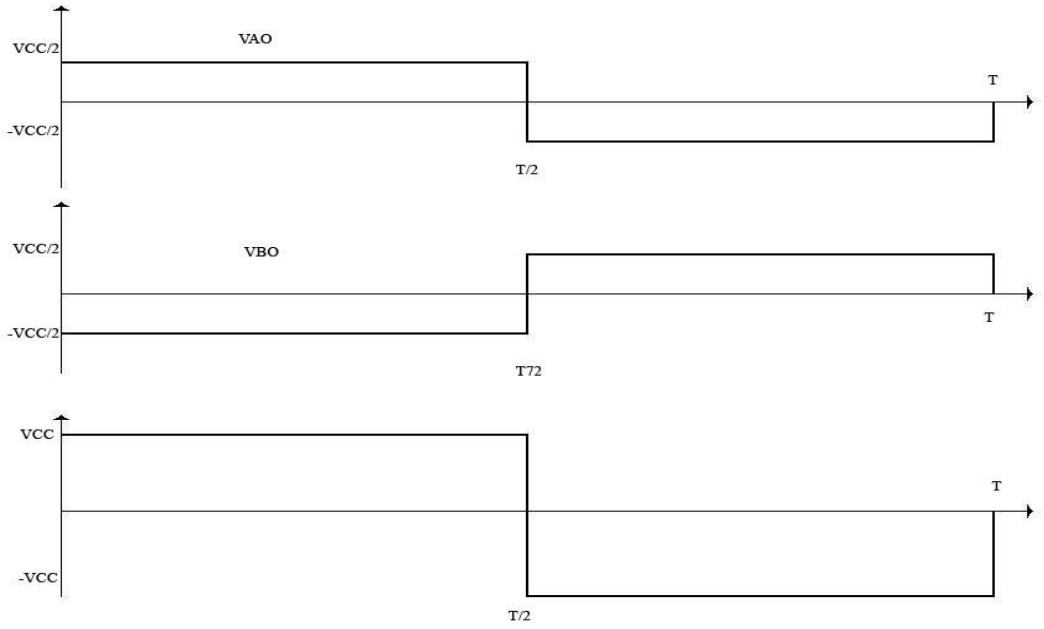


Figure 18 Waveforms of the transistors of Full Wave H-Bridge Inverter.

This output is not a sinusoidal form, but it is a AC wave.

Current wave form in the load depends of the kind of load. If we only use a resistance load, the output is like the figure 18, but if we use a load with inductance, output has more sinusoidal form than the only resistance load, because of the properties of the inductances filtered. An inductive load requires certain considerations when designing the switches on the full-bridge inverter, as the currents of the switches must be bidirectional.

Initially assume that in t=0, T_{A+} and T_{B-} are closed and the voltage in the load is +Vcc.

The current is expressed as the sum of the natural and forced replies:

$$i_0(t) = i_f(t) + i_n(t) = \frac{Vcc}{R} + Ae^{-t/\tau} \qquad ; 0 \le t \le \frac{T}{2} \tag{3.26}$$

A is a constant, which is calculated with a initial condition and $\tau = \frac{L}{R}$.

When $t = \frac{T}{2}$, T_{A+} y T_{B-} are opened, and T_{A-} y T_{B+} are closed. R-L Voltage change to –Vcc, and current has this form:

$$i_0(t) = \frac{-Vcc}{R} + Be^{\frac{-(t-\frac{T}{2})}{\tau}} \qquad ; \frac{T}{2} \le t \le T \tag{3.27}$$

Now, we calculate the equation 3-28 with t=0,

$$i_0(0) = \frac{Vcc}{R} + Ae^0 = \mathrm{I}_{\mathrm{m}\,in} \qquad \Rightarrow \qquad A = \mathrm{I}_{\mathrm{m}\,in} - \frac{Vcc}{R} \tag{3.28}$$

In this way, we calculate the equation 3-29 with $t = \frac{T}{2}$:

$$i_0(\frac{T}{2}) = \frac{-Vcc}{R} + Be^0 = \mathrm{I}_{\mathrm{m}\,ax} \quad \Rightarrow \qquad B = \mathrm{I}_{\mathrm{m}\,in} + \frac{Vcc}{R} \tag{3.29}$$

In steady state, current wave forms (equations 3-28 and 3-29) are converted to:

$i_0(t)$: $\qquad \frac{Vcc}{R} + \left(\mathrm{Im}in - \frac{Vcc}{R}\right)e^{\frac{-t}{\tau}} \qquad$ with $\qquad 0 \le t \le \frac{T}{2}$ (3.30)

$$-\frac{Vcc}{R} + \left(\mathrm{Im}ax + \frac{Vcc}{R}\right)e^{\frac{-(t-\frac{T}{2})}{\tau}} \qquad \text{with} \qquad \frac{T}{2} \le t \le T \tag{3.31}$$

We obtain a formula of I_{max}, assessing the first part of equation 3-32 with $t = \frac{T}{2}$:

$$i\left(\frac{T}{2}\right) = \mathrm{Im}ax = \frac{Vcc}{R} + \left(\mathrm{Im}in - \frac{Vcc}{R}\right)e^{\frac{-t}{2\tau}} \tag{3.32}$$

And: $\qquad -I_{MAX} = I_{MIN}$

substituting-I_{max} for I_{min} in the equation 3-33 and clearing I_{max}, we obtain:

$$\mathrm{Im}ax = -\mathrm{Im}in = \frac{Vcc}{R}\left(\frac{1 - e^{-T/2\tau}}{1 + e^{-T/2\tau}}\right) \tag{3.33}$$

In this way, the equations 3-32, 3-33 and 3-36 describe the current in a load RL steady state when a voltage is applied with a square wave. Figure 6 shows the resulting current in the load, the source and switches.

We can calculate the power absorbed with $Irms^2 \cdot R$. As the wave is symmetric, we can simplify the calculation by using only half of the period:

$$Irms = \sqrt{\frac{1}{T}\int_0^T i^2(t)d(t)} = \sqrt{\frac{2}{T}\int_0^{T/2}\left[\frac{Vcc}{R} + \left(\mathrm{Im}in - \frac{Vcc}{R}\right)e^{-t/\tau}\right]^2 d(t)} \tag{3.34}$$

When the switches are ideal, the power delivered by the source must be the same as the power absorbed by the load. The potency of a DC source is determined by:

$$Pcc = Vcc \cdot Is \tag{3.35}$$

The currents of the switches of Figure 19 shows that the switches in the full-bridge inverter must be capable of carrying both positive and negative currents to loads R-L. However, the actual electronic devices typically conduct current only in one direction. This problem, commented in section 2, is solved by placing feedback diodes in parallel with each switch. In the time interval in which the current in the switch should be negative, is the feedback diode which lets pass the current. The diodes are reversely biased when the current in the switch is positive. Figure 19 shows: the transistor and diode currents to a voltage with a square wave, and a load RL. The semiconductor power modules typically include diodes with feedback switches.

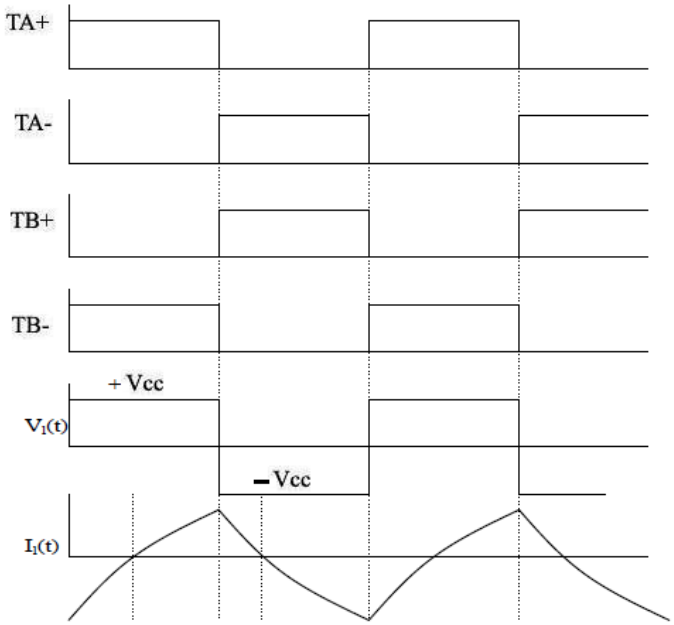


Figure 19 Current and voltage in a R-L load of Full Wave H-Bridge Inverter.

When transistors T_{A+} and T_{B-} are off, load current must be kept and will be transferred to the diodes D_{A-} and D_{B+}, causing the output voltage is -Vcc, while the current is not about to be canceled this will not conduct the T_{A-} and T_{B+} transistors.

On the other hand, the analysis by Fourier series of single-phase bridge inverter with square wave modulation will assume that we work with the following data:

Vcc = 48v, R = 2Ω, L = 5mH, f = 50Hz

Applying the equations:

$$V_n = \frac{4V_{cc}}{n\pi} = \frac{4(100)}{n\pi}$$

$$I_n = \frac{V_n}{Z_n} = \frac{V_n}{\sqrt{R^2 + (n\omega_0 L)^2}} = \frac{4(100)/n\pi}{\sqrt{10^2 + (n(2\pi 60)(0{,}02))^2}}$$

$$P_n = I_n{}^2, {}_{rms}R = (\frac{In}{\sqrt{2}})^2 R$$

Only the first terms of the series are important because for high numbers of the harmonic amplitude of the Fourier component of voltage decreases and the impedances increases. This produces a small current for these harmonics.

The following table shows this phenomenon (Table 5):

n	fn (hz)	Vn (V)	Zn (Ω)	In (A)	Pn (W)
1	50	61,12	2,54	24,03	577,53
3	150	20,37	5,12	3,98	15,84
5	250	12,22	8,10	1,51	2,27
7	350	8,73	11,18	0,78	0,61
9	450	6,79	14,28	0,48	0,23

Table 5 Coefficients of the Fourier series of Full Wave H-Bridge Inverter.

In Figure 20 shows the Fourier series, both the current (first graph) and the voltage (second graph) with the previous data and with Simulink (Matlab) graphs.

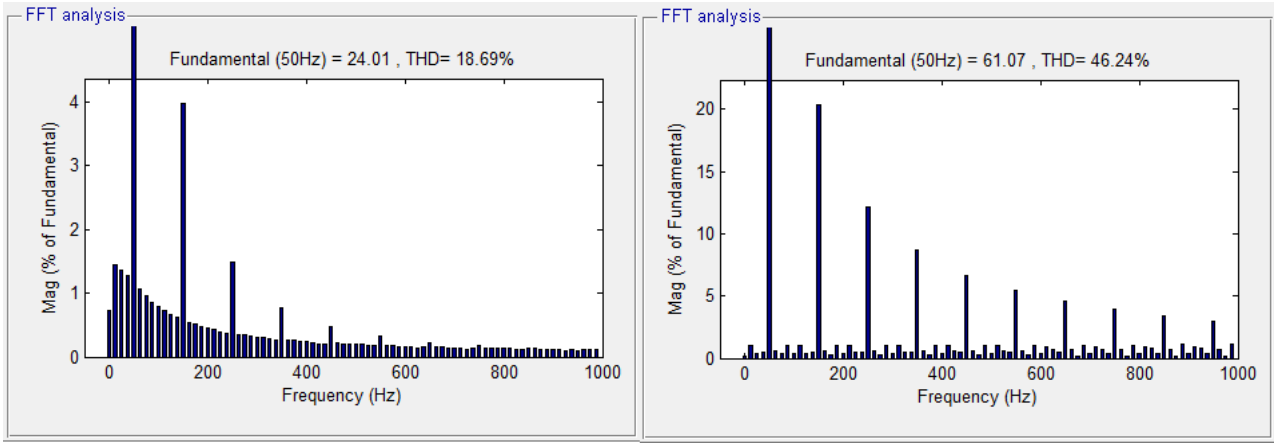


Figure 20 Frequency spectrum of the voltage and current of Full Wave H-Bridge Inverter (Square Wave Control).

To obtain the THD factor of the voltage and load current, it is used the Fourier series for a square wave and the equation defining the THD factor, shown in section 2.4. The rms value of a square wave voltage is equal to the peak value, and the component of the fundamental frequency is the first term being then:

$$V_{rms} = V_{cc}$$

$$V_{1,\,rms} = \frac{V_1}{\sqrt{2}} = \frac{4V_{cc}}{\sqrt{2}\pi}$$

Voltage THD factor:

$$THDv = \frac{\sqrt{{V_{rms}}^2 - {V_1}^2, _{rms}}}{V_{1,\,rms}} = \frac{\sqrt{{V_{cc}}^2 - (\frac{4V_{cc}}{\sqrt{2}\pi})^2}}{\frac{4V_{cc}}{\sqrt{2}\pi}} = \frac{\sqrt{48^2 - (\frac{192}{\sqrt{2}\pi})^2}}{\frac{192}{\sqrt{2}\pi}} = 0{,}483 = 48{,}3\%$$

Current THD factor:

$$THDi = \frac{\sqrt{\sum_{n=2}^{\infty}(I_{n,\,rms})^2}}{I_{1,\,rms}} \approx \frac{\sqrt{(\frac{3{,}97}{\sqrt{2}})^2 + (\frac{1{,}50}{\sqrt{2}})^2 + (\frac{0{,}78}{\sqrt{2}})^2 + (\frac{0{,}47}{\sqrt{2}})^2}}{(\frac{24{,}03}{\sqrt{2}})} = 0{,}183 = 18{,}3\%$$

3.2.3.2 FULL WAVE H-BRIDGE INVERTER (VOLTAGE HARMONIC CANCELLATION)

This control technique is similar to that explained in the previous case (Figure 21, square wave control). But the difference is that now each of the two parts of switches are controlled independently. If the phase shift between the two is 180 º turn to the previous case, but if the gap is (180 °-α) is different [6].

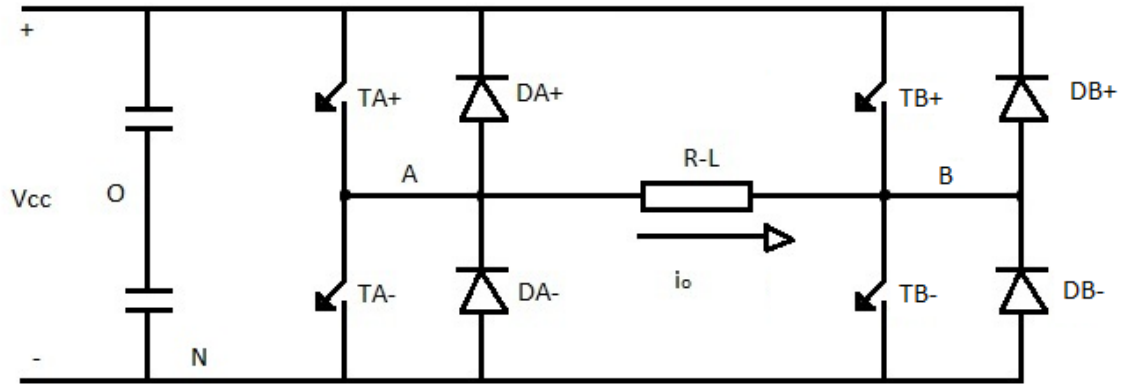


Figure 21 Full Wave H-Bridge inverter.

The two parts of switches are controlled independently. If the phase shift between the two is 180 º turn to the previous case, but if the gap is (180 °-α) is different.

The following graph (Figure 22) shows the waveform of the voltage applied to the load with this kind of control:

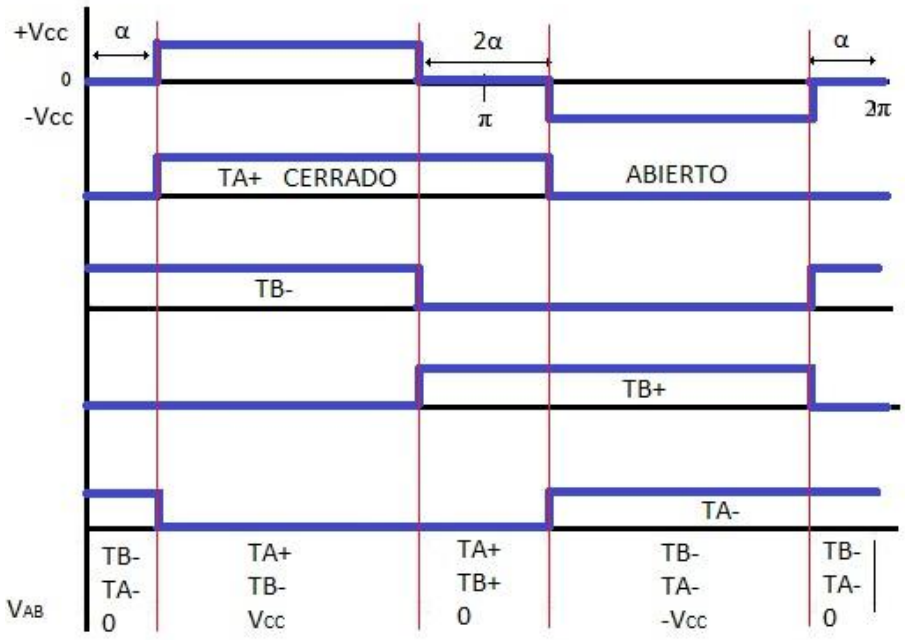


Figure 22 Inverter output for harmonic control and switching scheme for the Full Wave H-Bridge Inverter (Voltage Harmonic Cancellation).

As said before, when α = 0, we would be in the case of square wave modulation.

The following figure (Figure 23) shows the amplitudes of the harmonics depending on the values of α.

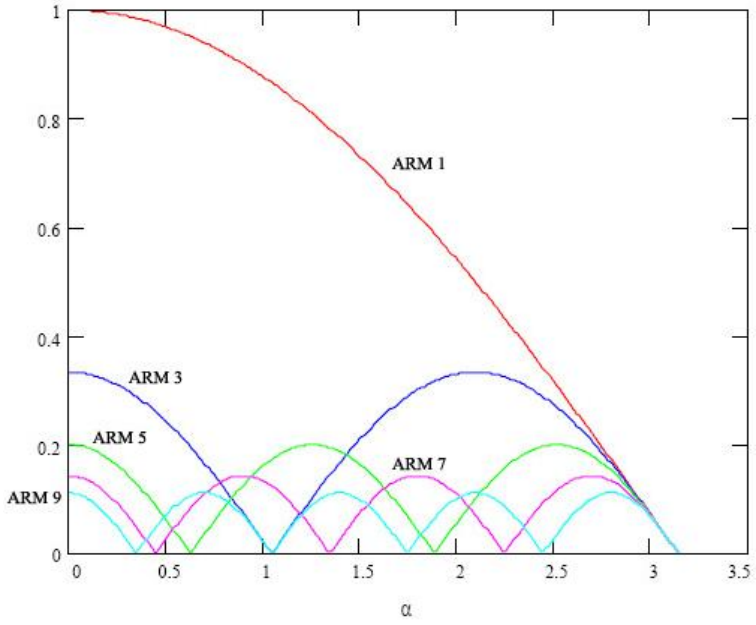


Figure 23 Amplitudes of the different harmonics, depending on the parameter α.

As we can see in Figure 22, the output voltage takes the following values: + Vdc, 0,-Vcc. We can control the output voltage by adjusting the α value.

The rms value of the waveform of the voltage, in Figure 22 is:

$$V_{rms} = \sqrt{\frac{1}{\pi}\int_{\alpha}^{\pi-\alpha} Vcc^2 d(\omega t)} = Vcc\sqrt{1-\frac{2\alpha}{\pi}} \tag{3.36}$$

The Fourier series of waveform is expressed as:

$$V_0(t) = \sum_{n,impares}^{\infty} V_n(Sin(n\cdot\omega_0\cdot t)) \tag{3.37}$$

Because of the symmetry, the amplitudes are:

$$Vn = \frac{2}{\pi}\int_{\alpha}^{\pi-\alpha} Vcc\cdot\sin(n\cdot\omega_0\cdot t)d(\omega_0 t) = (\frac{4Vcc}{n\cdot\pi})\cos(n\alpha) \tag{3.38}$$

α is the angle of zero voltage at each end of the pulse. The amplitude at the output for each output frequency is a function of α.

Particularly, the amplitude at the fundamental frequency is controlled by adjusting α:

$$V_1 = (\frac{4Vcc}{\pi})\cos(\alpha) \tag{3.39}$$

On the other hand, the harmonic content can also be controlled by adjusting α.

We work with the following data:

Vcc = 48v, R = 2Ω, L = 5mH, f = 50Hz

The harmonic n is deleted if:

$$\alpha = \frac{90^o}{n}$$

For example, applying the above formula, V_3 = 0 when α = 30 °. We could eliminate other harmonics give a correct value of α, because the cosine is 0.

The following table (Table 6) shows the coefficients of the Fourier series. Shows how the harmonic 3 (with α = 30º) is 0, also the harmonic 9 because the cosine is zero.

n	fn (hz)	Vn (V)	Zn (Ω)	In (A)	Pn (W)
1	50	52,93	2,54	20,81	433,20
3	150	0,01	5,12	0,00	0,00
5	250	-10,58	8,10	-1,31	1,70
7	350	-7,56	11,18	-0,68	0,46
9	450	-0,01	14,28	0,00	0,00

Table 6 Coefficients of the Fourier series of Full Wave H-Bridge Inverter (Voltage Harmonic Cancellation).

Graphically, in Figure 24, we can see how these harmonics have been eliminated, both the current (first graph) and the voltage (second graph) with the previous data and with Simulink (Matlab) graphs:

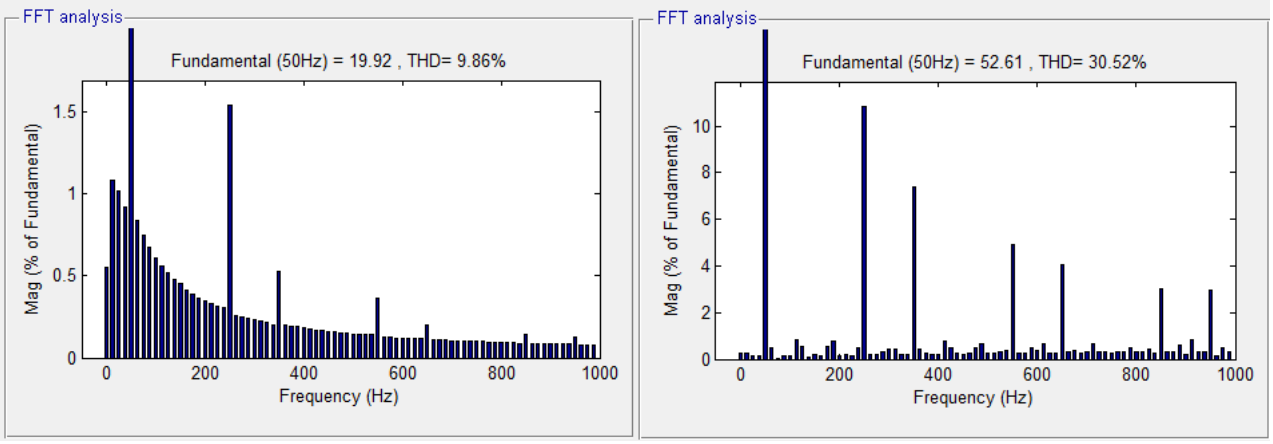


Figure 24 Frequency spectrum of the voltage and current Frequency spectrum for the voltage and current of Full Wave H-Bridge Inverter (Voltage Harmonic Cancellation)..

To obtain the THD factor of the voltage and load current, it is used the Fourier series for a square wave and the equation defining the THD factor, shown in section 2.4. The rms value of a square wave voltage is equal to the peak value, and the component of the fundamental frequency is the first term being then:

Voltage THD factor:

$$THDv = \frac{\sqrt{\sum_{n=2}^{\infty}(V_{n,\,rms})^2}}{V_{1,\,rms}} \approx \frac{\sqrt{(\frac{0}{\sqrt{2}})^2 + (\frac{10,58}{\sqrt{2}})^2 + (\frac{7,56}{\sqrt{2}})^2 + (\frac{0}{\sqrt{2}})^2}}{(\frac{52,93}{\sqrt{2}})} = 0,245 = 24,5\%$$

Current THD factor:

$$THDi = \frac{\sqrt{\sum_{n=2}^{\infty} (I_{n,\,rms})^2}}{I_{1,\,rms}} \approx \frac{\sqrt{(\frac{0}{\sqrt{2}})^2 + (\frac{1{,}31}{\sqrt{2}})^2 + (\frac{0{,}68}{\sqrt{2}})^2 + (\frac{0}{\sqrt{2}})^2}}{(\frac{20{,}81}{\sqrt{2}})} = 0{,}070 = 7{,}10\%$$

In summary, we can eliminate the harmonics we want by introducing values in the parameters in with this inverter with this type of modulation.

Depending on the number of these parameters introduced in each half cycle, we will have greater or lesser number of degrees of freedom, when deleting harmonics.

3.2.3.3 FULL WAVE H-BRIDGE INVERTER (BIPOLAR PWM CONTROL)

The scheme for this type of inverter can be see in the Figure 25.

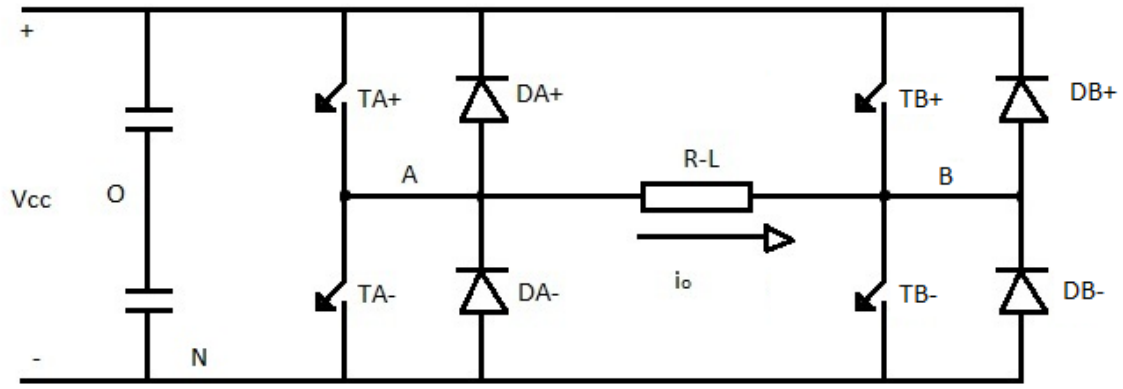


Figure 25 Full wave H-Bridge inverter.

When the instantaneous value of the sinusoidal reference is higher than the triangular bearer, the output is in + V_{CC}, and when the reference is less than the bearer, the output is in – V_{CC} [2]:

$$V_0 = +Vcc \qquad \text{when} \qquad V_{\sin} > V_{tri}$$

$$V_0 = -Vcc \qquad \text{when} \qquad V_{\sin} < V_{tri}$$

This version is bipolar PWM as the output takes alternating values between plus and minus the voltage of the DC source.

In this case the switches (TA+ TB-) and (TA- TB+) are controlled at a time (switching pairs).El esquema de conmutación que permitirá implementar la conmutación bipolar utilizando el puente inversor de onda completa de la figura 24 se determina comparando las señales instantáneas de referencia y portadora, como se puede observar en la figura 25, de la siguiente manera:

T_{A+} y T_{B-} is working when $V_{seno} > V_{tri}$ $(V_0 = +Vcc)$

T_{A-} y T_{B+} is working when $V_{seno} < V_{tri}$ $(V_0 = -Vcc)$

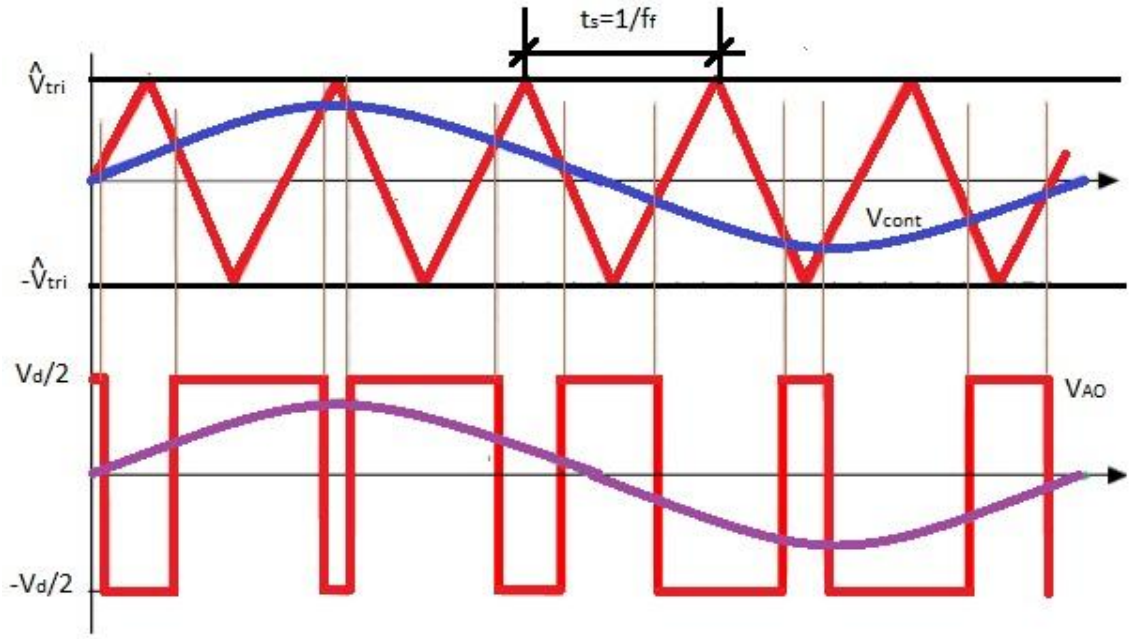


Figure 26 Bipolar Width Modulated.

The Fourier series of bipolar PWM output shown in Figure 29 is calculated by examining each of the pulses. The triangular waveform is synchronized with the reference, as shown in Figure 25, and m_f is an odd integer that is chosen. Then the PWM output shows an odd symmetry, and can be expressed as the Fourier series:

$$V_0(t) = \sum_{n=1}^{\infty} V_n(Sin(n\cdot\omega_0\cdot t)) \tag{3.40}$$

For the k-th pulse of the PWM output in Figure 26, the Fourier coefficient is:

$$Vnk = \frac{2}{\pi}\int_0^T v(t)\sin(n\cdot\omega_0\cdot t)d(\omega_0\cdot t) = \frac{2}{\pi}\left[\int_{\alpha k}^{\alpha k+\delta k} vcc\cdot\sin(n\cdot\omega_0\cdot t)d(\omega_0\cdot t) + \int_{\alpha k+\delta k}^{\alpha k+1}(-vcc)\cdot\sin(n\cdot\omega_0\cdot t)d(\omega_0\cdot t)\right] \tag{3.41}$$

Calculating:

$$Vnk = \frac{2Vcc}{n\pi}\left[\cos n\alpha_k + \cos n(\alpha_{k+1}) - 2\cos n(\alpha_k + \delta_k)\right] \tag{3.42}$$

Each V_n coefficient for the PWM waveform is the sum of V_{nk} for p pulses included in a period:

$$V_n = \sum_{k=1}^{p} V_{nk}$$

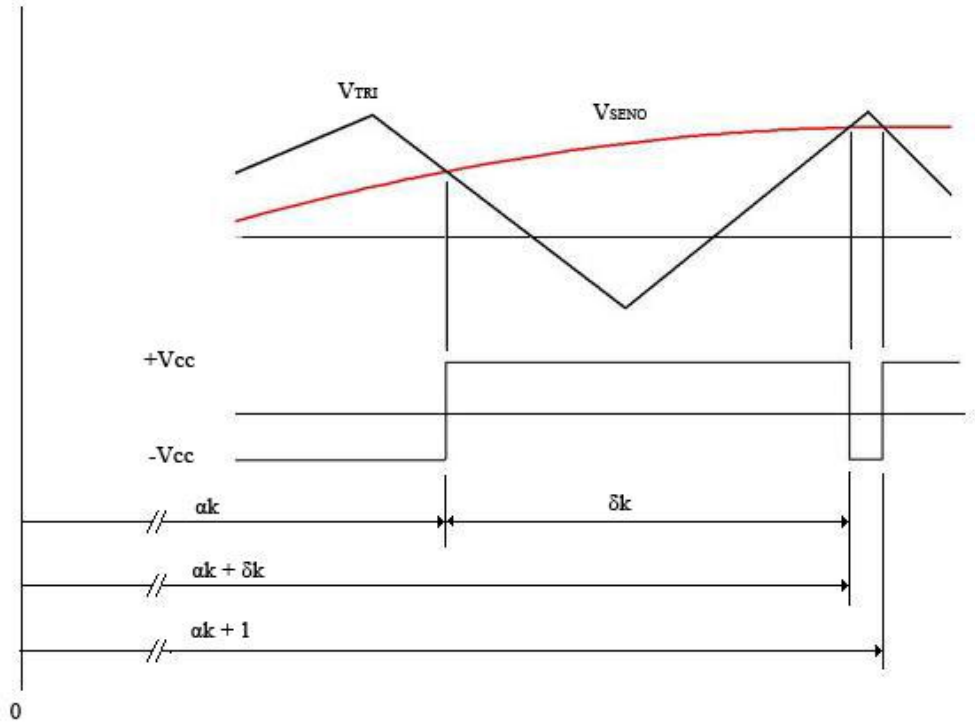


Figure 27 A PWM pulse to calculate the Fourier series for bipolar PWM.

The amplitudes of the harmonics is a function of m_a, because the width of each pulse depends on the relative amplitudes of the sinusoidal and triangular waves. The first harmonics in the output spectrum frequencies are around m_f. In the table 7 are the first harmonics for bipolar PWM output. The Fourier coefficients are a function of m_f if m_f is high (≥ 9).

	m_a=1	**0,9**	**0,8**	**0,7**	**0,6**	**0,5**	**0,4**	**0,3**	**0,2**	**0,1**
n = 1	1	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
n = m_f	0,6	0,71	0,82	0,92	1,01	1,08	1,15	1,2	1,24	1,27
n = m_f ± 2	0,32	0,27	0,22	0,17	0,13	0,09	0,06	0,03	0,02	0

Table 7 Normalized Fourier coefficients V_n / V_{DC} for bipolar PWM.

Using the following data:

R = 2Ω, L = 5 mH, Vcc = 48 v, f = 50 Hz, m_a = 0,5 y m_f = 20 (f_{TRI} = (20) (50)=1.000Hz)

$$V_1 = m_a \cdot Vcc = 0{,}5 \cdot 48 = 24v$$

$$I_n = \frac{V_n}{Z_n} = \frac{V_n}{\sqrt{R^2 + (n \cdot \omega_0 L)^2}}$$

For the fundamental frequency:

$$I_n = \frac{24}{\sqrt{2^2 + (1 \cdot 2\pi 50 \cdot 0{,}005)^2}} = 9{,}44A$$

With m_f = 20, the first harmonics are held for n=20, 18 y 22. Using the table 7:

$$V_{20} = 1{,}08 \cdot 48 = 51{,}84v$$

$$V_{18} = V_{22} = 0{,}09 \cdot 48 = 4{,}32v$$

Currents corresponding to each of the harmonics are calculated from the equation 3-49.

To calculate the power of each frequency:

$$P_n = (I_{n,\,ef})2 \cdot R = (\frac{I_n}{\sqrt{2}})^2 R$$

By performing all calculations, these results are obtained:

n	fn (hz)	Vn (V)	Zn (Ω)	In (A)	Pn (W)
1	50	24,00	2,54	9,44	89,06
18	900	4,32	28,34	0,15	0,02
20	1000	51,84	31,48	1,65	2,71
22	1100	4,32	34,62	0,12	0,02

Table 8 Coefficients of the Fourier series for Bipolar PWM inverter.

Higher level harmonics provide little power, so are without considering.

Current THD factor:

$$THDi = \frac{\sqrt{\sum_{n=2}^{\infty} (I_{n,\,rms})^2}}{I_{1,\,rms}} \approx \frac{\sqrt{(\frac{0{,}15}{\sqrt{2}})^2 + (\frac{1{,}65}{\sqrt{2}})^2 + (\frac{0{,}12}{\sqrt{2}})^2}}{(\frac{9{,}44}{\sqrt{2}})} = 0{,}175 = 17{,}50\%$$

Voltage THD factor:

$$THDv = \frac{\sqrt{\sum_{n=2}^{\infty} (V_{n,\,rms})^2}}{V_{1,\,rms}} \approx \frac{\sqrt{(\frac{4{,}32}{\sqrt{2}})^2 + (\frac{51{,}84}{\sqrt{2}})^2 + (\frac{4{,}32}{\sqrt{2}})^2}}{(\frac{24}{\sqrt{2}})} = 2{,}174 = 217{,}40\%$$

Waveforms of voltage and load current used for the example can be seen in Figure 28:

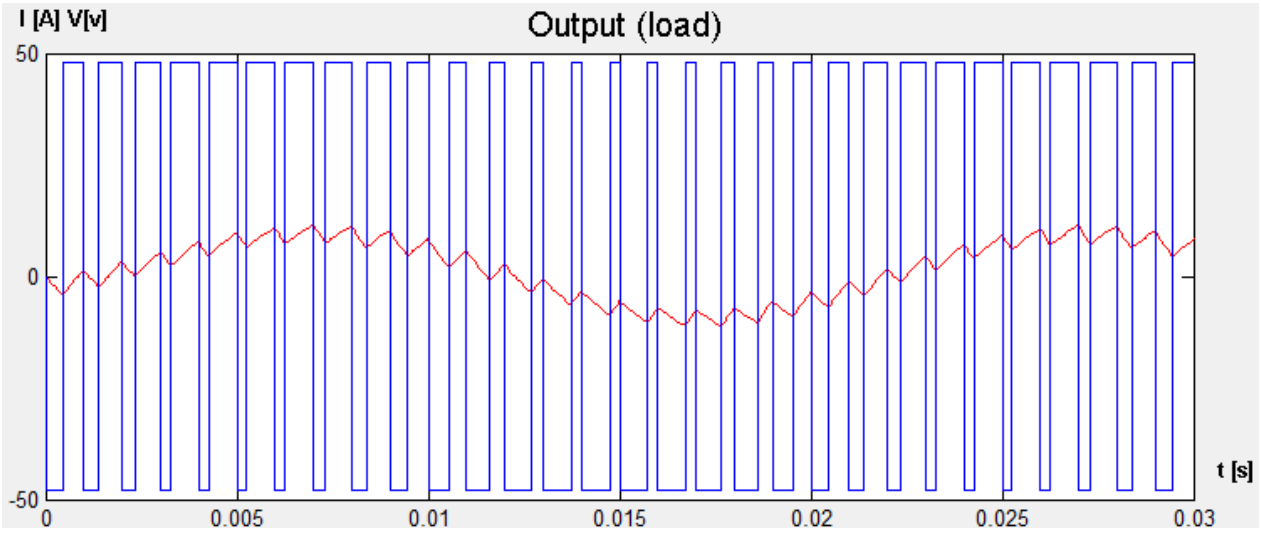


Figure 28 Voltage and current for bipolar PWM load.

In Figure 29 shows the Fourier series, both the current (first graph) and the voltage (second graph) with the previous data and with Simulink (Matlab) graphs:

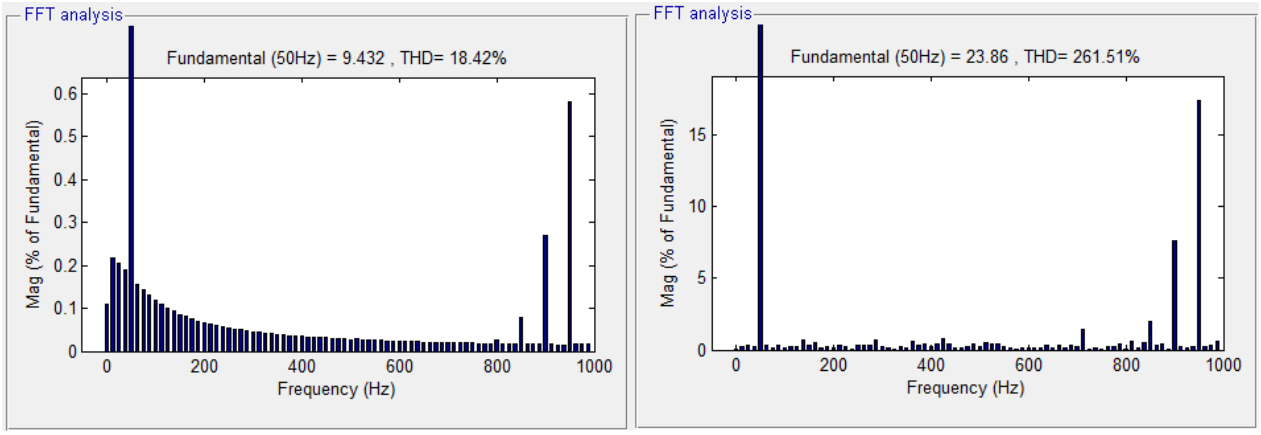


Figure 29 Frequency spectrum of the voltage and current of Full Wave H-Bridge (Bipolar PWM Control).

3.2.3.4 FULL WAVE H-BRIDGE INVERTER (SINGLE-POLE PWM CONTROL)

The scheme for this type of inverter can be see in the Figure 30.

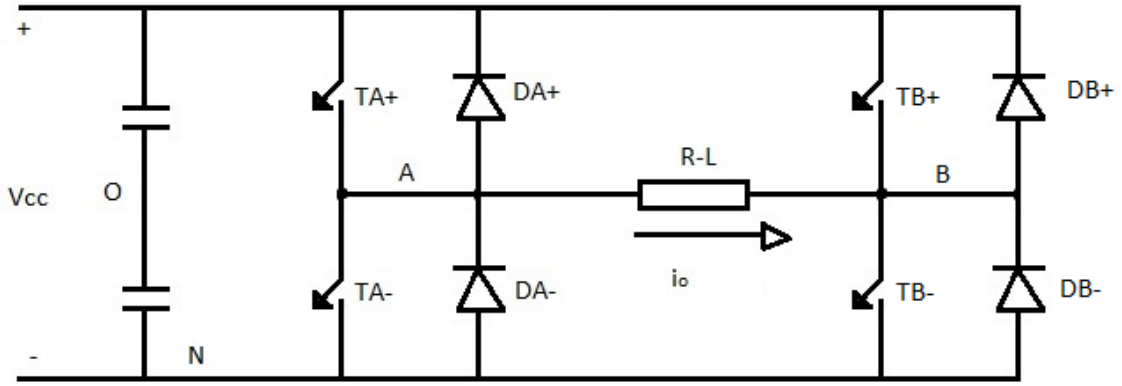


Figure 30 Full wave H-Bridge inverter.

In a unipolar switching scheme for pulse width modulation (PWM), the output is switched from high level to zero or low level to zero, instead of between high and low levels, as in the bipolar switching. A unipolar switching scheme has the following switches control [2]:

T_{A+} is on when $V_{SIN}>V_{TRI}$

T_{B-} is on when $-V_{SIN}<V_{TRI}$

T_{B+} is on when $-V_{SIN}>V_{TRI}$

T_{A-} is on when $V_{SIN}<V_{TRI}$

It's observed that the pairs of switches (T_{A+}, T_{A-}) y (T_{B+}, T_{B-}) are complementary, when a switch of one pair is closed the other is open. Voltages V_A y V_B in Figure 31 range between +Vcc and zero. The output voltage $V_o = V_{AB} = V_A - V_B$ is as shown in Figure 31.

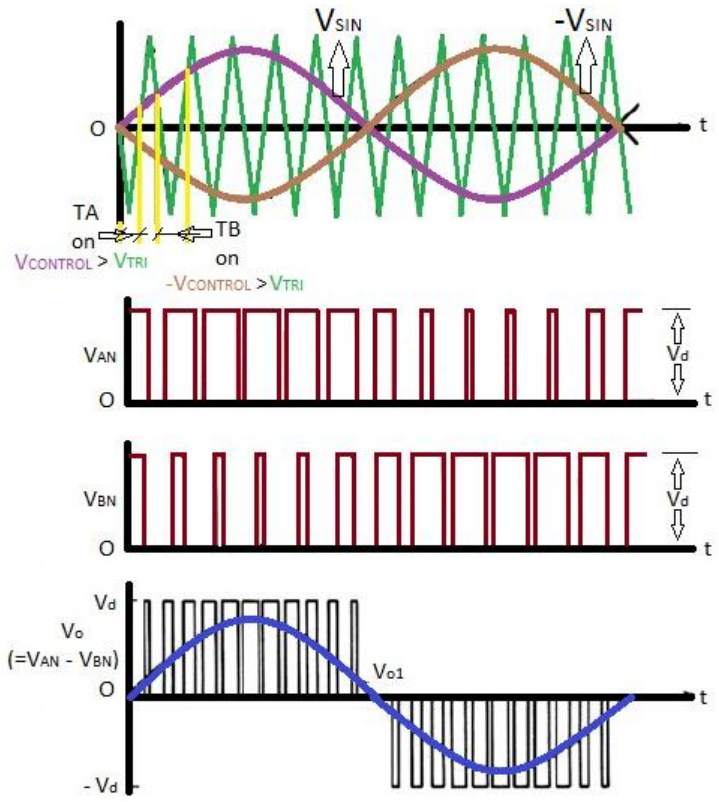


Figure 31 Waveform of Single-Pole Width Modulated.

Another unipolar switching scheme has only one pair of switches working to the carrier frequency while the other pair operates at the reference frequency, so we have two high frequency switches and two low frequency. This is the switching scheme:

T_{A+}	is on when	$V_{SIN}>V_{TRI}$	(high frequency)
T_{A-}	is on when	$V_{SIN}<V_{TRI}$	(high frequency)
T_{B-}	is on when	$V_{SEIN}>0$	(low frequency)
T_{B+}	is on when	$V_{SIN}<0$	(low frequency)

Where the sine and triangular wave are as shown in Figure 32. Alternatively, T_{B-} y T_{B+} may be the high frequency switches and T_{A-} y T_{A+} may be low frequency switches.

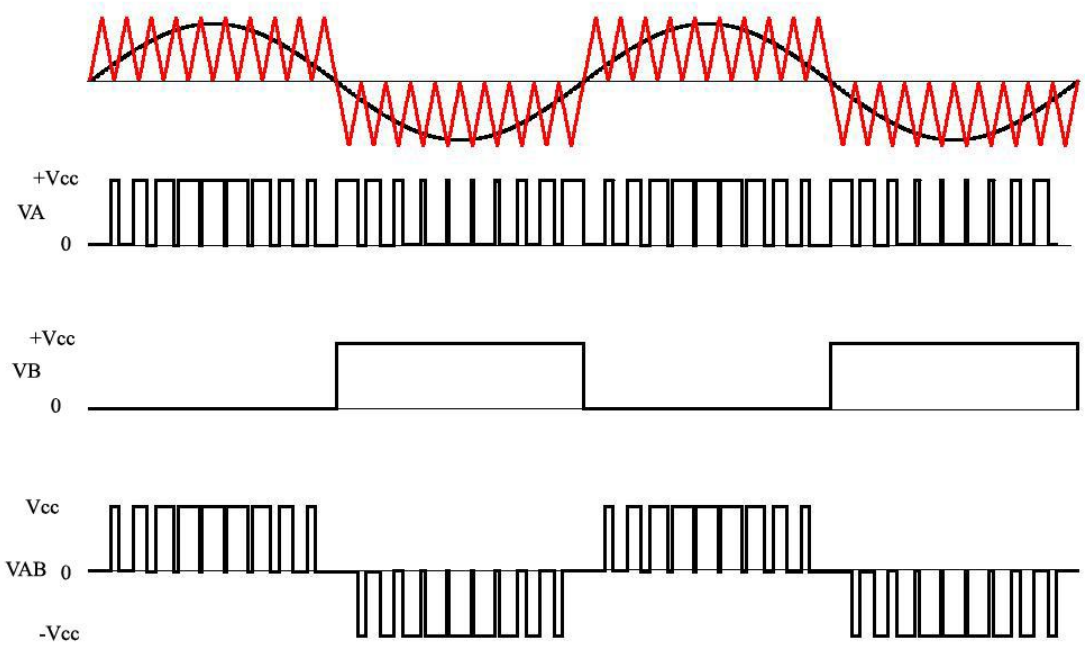


Figure 32 Unipolar PWM switches high and low frequency, reference and control signals and voltages V_A, V_B and V_{AB}.

With unipolar switching scheme of Figure 32, some of the harmonics which were in the spectrum of bipolar scheme are now absent. The harmonics in the output start approximately in $2m_f$, and is chosen an m_f that be an even integer.

The table 9 shows the first harmonic output for unipolar PWM control. Unipolar PWM scheme using switches high and low frequency, shown in Figure 32, will give results similar to those indicated in the table 9, but the harmonics estart around m_f instead of $2m_f$.

	m_a=1	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
n = 1	1	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
n = $2m_f$ ±1	0,18	0,25	0,31	0,35	0,37	0,36	0,33	0,27	0,19	0,1
n = $2m_f$ ± 3	0,21	0,18	0,14	0,1	0,07	0,04	0,02	0,01	0	0

Table 9 Normalized Fourier coefficients V_n/V_{cc} for Single-Pole PWM Control.

We work with the following data:

R = 2Ω, L = 5 mH, Vcc = 48 v, f = 50 Hz, m_a = 0,5 y m_f = 20 (f_{TRI} = (2) (50)=1.000Hz)

Calculating:

$$V_1 = m_a \cdot Vcc = 0{,}5 \cdot 48 = 24v$$

$$I_n = \frac{V_n}{Z_n} = \frac{V_n}{\sqrt{R^2 + (n \cdot \omega_0 L)^2}}$$

For the fundamental frequency:

$$I_n = \frac{24}{\sqrt{2^2 + (1 \cdot 2\pi 50 \cdot 0{,}005)^2}} = 9{,}44A$$

With m_f = 21, The first harmonics are for n = 39, 41, 43 y 45. Using the table 9:

$$V_{37} = V_{43} = 0{,}04 \cdot 48 = 1{,}92v$$

$$V_{39} = V_{41} = 0{,}36 \cdot 48 = 17{,}28v$$

The power for each frequency is calculated from:

$$P_n = (I_{n,ef})2 \cdot R = (\frac{I_n}{\sqrt{2}})^2 R$$

In the table 10 summarizes the frequencies of the voltages, currents and the resulting power at these frequencies:

n	fn (hz)	Vn (V)	Zn (Ω)	In (A)	Pn (W)
1	50	24,00	2,54	9,44	89,06
37	1850	1,92	58,15	0,03	0,00
39	1950	17,28	61,29	0,28	0,08
41	2050	17,28	64,43	0,27	0,07
43	2150	1,92	67,57	0,03	0,00

Table 10 Coefficients of the Fourier series of Single-Pole PWM Inverter.

Higher level harmonics provide little power, so are without considering.

Current THD factor:

$$THDi = \frac{\sqrt{\sum_{n=2}^{\infty} (I_{n,\,rms})^2}}{I_{1,\,rms}} \approx \frac{\sqrt{(\frac{0{,}03}{\sqrt{2}})^2 + (\frac{0{,}28}{\sqrt{2}})^2 + (\frac{0{,}27}{\sqrt{2}})^2 + (\frac{0{,}03}{\sqrt{2}})^2}}{(\frac{9{,}44}{\sqrt{2}})} = 0{,}041 = 4{,}1\%$$

Voltage THD factor:

$$THDv = \frac{\sqrt{\sum_{n=2}^{\infty}(V_{n,\,rms})^2}}{V_{1,\,rms}} \approx \frac{\sqrt{(\frac{1{,}92}{\sqrt{2}})^2 + (\frac{17{,}28}{\sqrt{2}})^2 + (\frac{17{,}28}{\sqrt{2}})^2 + (\frac{1{,}92}{\sqrt{2}})^2}}{(\frac{24}{\sqrt{2}})} = 1{,}023 = 102{,}3\%$$

Using the developmentally truncated Fourier series of the table 10, the THD factor is underestimated. However, as the load impedance increases and the harmonic amplitudes generally decrease as n increases, the previous estimate should be acceptable.

Waveforms of voltage and load current used for the example can be seen in Figure 33:

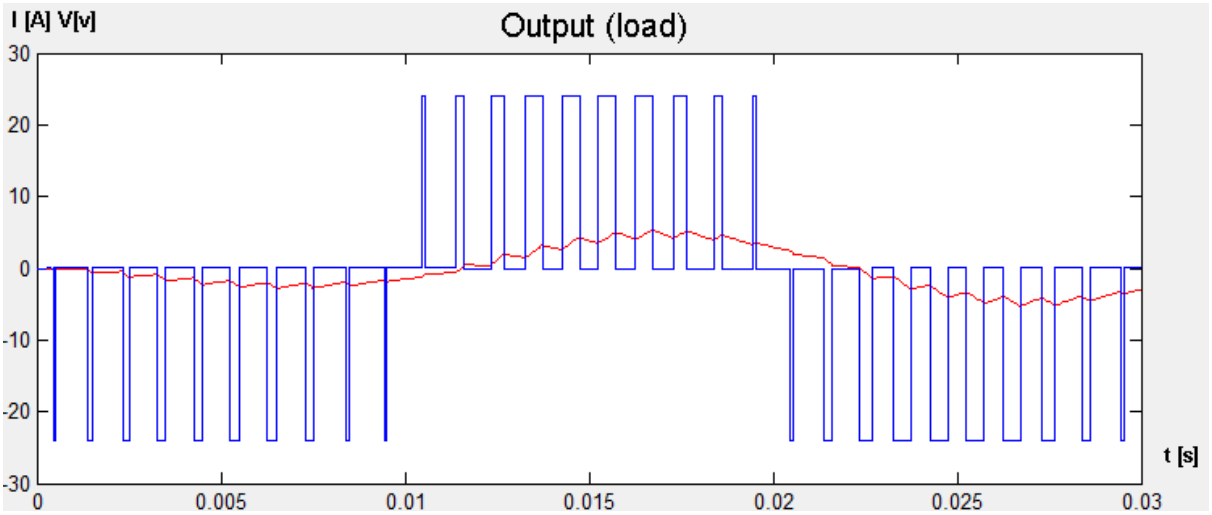


Figure 33 Voltage and current output of the load Single-Pole PWM Control.

In Figure 34 shows the Fourier series, both the current (first graph) and the voltage (second graph) with the previous data and with Simulink (Matlab) graphs:

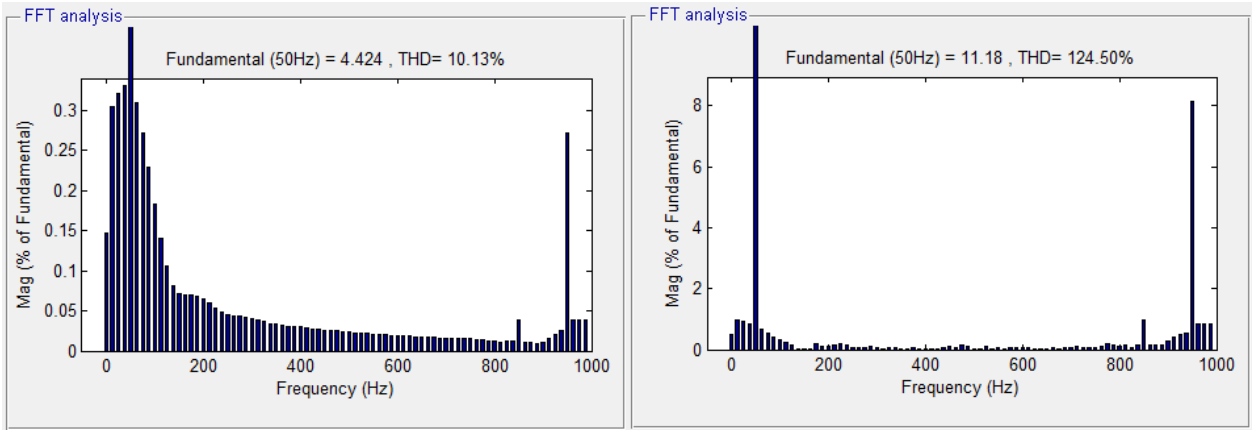


Figure 34 Frequency spectrum of the voltage and current of Full Wave H-Bridge Inverter (Single-Pole PWM Control).

3.2.4 THREE-PHASE V.S.I. 6-STEP INVERTERS

The three-phase voltage inverters are used mainly in high power applications. They can be obtained by connecting three single-phase inverters in parallel; considering that these control signals, are 120º out of phase, in order to obtain a balanced three-phase voltage system. The scheme for this type of inverter can be seen in Figure 35.

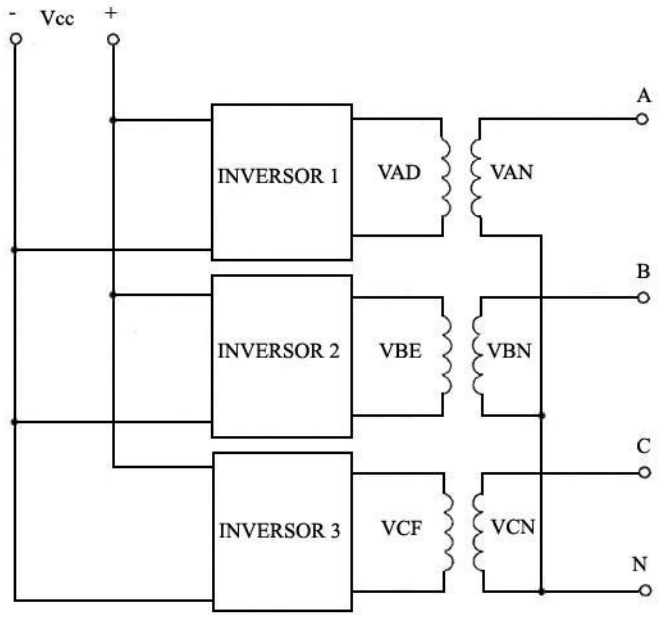


Figure 35 Three-Phase inverter from three phase inverters.

This solution requires three single-phase transformers, 12 controllable semiconductor elements and 12 diodes, so it's little used in practice, when working in the field of low and medium power.

Another solution, much more used, may be obtained using three-phase branches of a half-bridge inverter, with six semiconductor elements and 6 diodes, without having to use transformers.

3.2.4.1 THREE-PHASE V.S.I. 6-STEP INVERTER (SQUARE WAVE CONTROL)

The sequence of the six basis signals for the transistors of the circuit of Figure 35 are shown in Figure 36, where one can see that each 60º a transistor stop driving and the other located on the same branch of the bridge, starts driving.

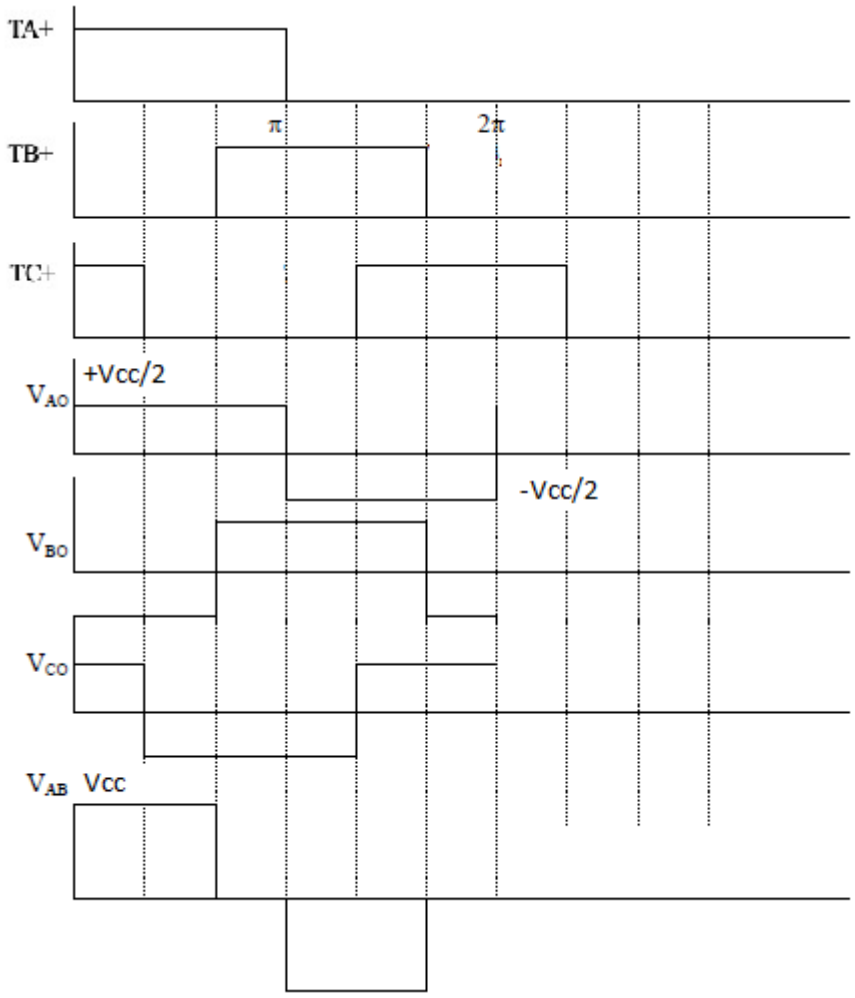


Figure 36 Waveforms of the transistors of Three-Phase V.S.I. 6-Step Inverter.

The sequence follows a certain order to the three output voltages resulting 120º out of phase with each other as shown in Figure 36. At any moment there are three driving transistors due to 180º each is active, this being the maximum possible driving time.

The positive signs of the output voltages are taken:

$$V_{AB} = V_A - V_B \qquad V_{BC} = V_B - V_C \qquad V_{CA} = V_C - V_A$$

and positive currents are leaving the inverter and into the load.

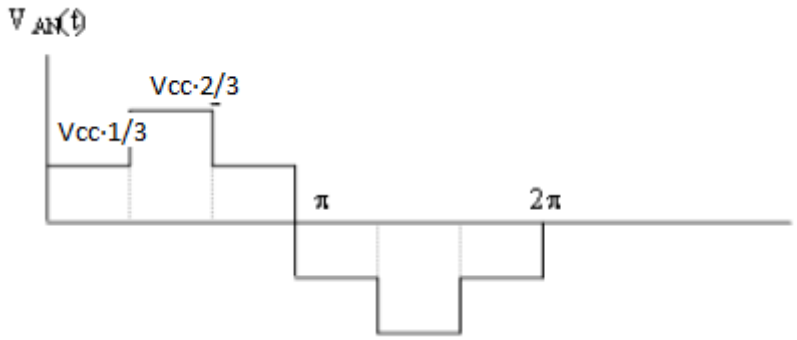


Figure 37 The output of a phase.

For any instant is between 60º and 120º the three transistors are driving are T_{A+}, T_{B}- and T_{C}-, therefore the terminal (A) is connected to + Vcc, while the terminals (B) and (C) are linked together and connected to the negative terminal of the source. Consequently:

$$V_{AB} = +V_{CC} \qquad V_{BC} = 0 \qquad V_{CA} = -V_{CC}$$

Each of the three output voltages, to be taken between two terminals of the bridge, depend on the driving of only two transistors, even though there are always three driving. For this reason, the half-cycles of these voltages are only 120º and not 180º, there being a dead time of 60º in each half cycle as shown in Figure 36.

These tensions are coming to the terminals of the load, the load in triangle or star.

3.2.4.1.1 TRIANGLE LOAD

The output voltages of the bridge are coincident with the phase voltages of the load, but the phase currents are different from the bridge output (Figure 38 and 39).

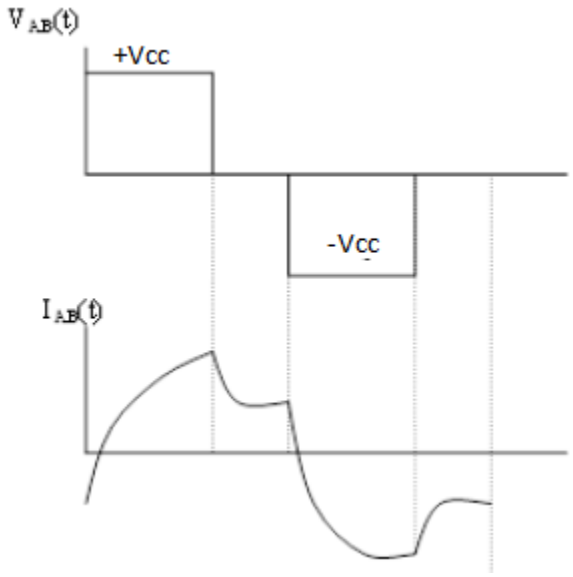


Figure 38 Waveform of the currents in the inverter load connected in triangle.

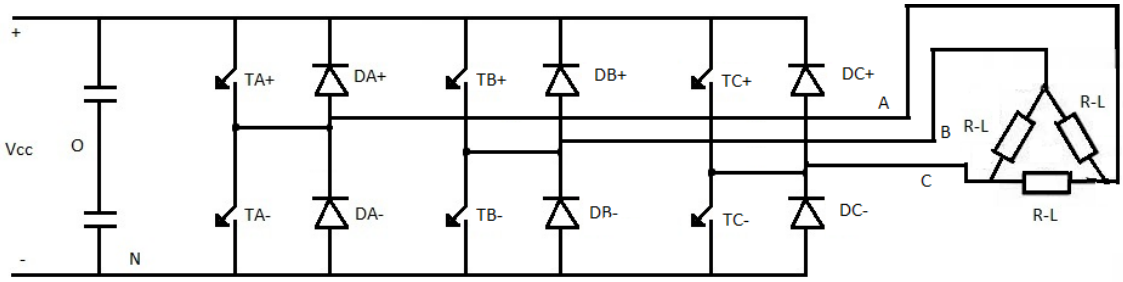


Figure 39 Three-Phase Full Bridge Inverter with load connected in triangle.

Taking as reference the voltage V_{AB} can be seen that:

- I_{AB} is formed by exponential segments, like all circuit currents and is out of phase at an angle ϕ in respect of V_{AB}.

- The offset of 120º between the branch currents of the load is keeping.

- $I_A = I_{AB} – I_{CA}$.

- The three output currents of the bridge I_A, I_B, I_C, maintain their offset of 120º from each other and 30º in delay on the respective branch currents I_{AB}, I_{BC} , I_{CA}.

- In a complete cycle T (o 2π) 12 switching that are not symmetrical occurs since transistors conduct for a longer time than the diodes. These switches are still an item at a time.

- The current through the transistor T_{A+} is much higher than the diode D_{A+} and are of opposite signs.

- The driving time T_{A+} is 180º - ϕ, and ϕ is the time that each diode is driving.

- At any time there are still three elements driving.

- In the source, in this case, the current at any time is negative, other words not crossing the x-axis because the active power of the load and the source can be observed by the pulse form of that current. This indicates that in addition to its average value, there are harmonic components that return to the source. To these loads R-L, reactive current flowing through the bridge and the load is higher than the source.

- For the current of the source has to cross the zero, ie there is no visible current with an oscilloscope is necessary that R active part of the load is small compared to the inductive component.

We work with the following data:

R = 2Ω, L = 5 mH, Vcc = 48 v, f = 50 Hz

On the other hand the voltage waveform expressed as a Fourier series, with the time origin at -π/6 has that the fundamental harmonic is:

$$V_{AB1} = \frac{2\sqrt{3}}{\pi} Vcc \cdot Sen(\omega t) \tag{3.43}$$

Its effective value:

$$V_{AB1} = \frac{\sqrt{6}}{\pi} Vcc \tag{3.44}$$

The fundamental harmonic of the current in this branch will be:

$$I_{AB1} = \frac{2\sqrt{3}}{\pi} \frac{Vcc}{Z_1} \cdot Sen(\omega t - \varphi_1) \tag{3.45}$$

And its effective value:

$$I_{AB1} = \frac{\sqrt{6}}{\pi} \frac{Vcc}{Z_1} \tag{3.46}$$

And

$$Z_1 = \sqrt{R^2 + (\omega L)^2} \tag{3.47}$$

is the load impedance at the fundamental frequency and ϕ_1 is the phase angle of I_{AB1} in delay of V_{AB1}.

Each harmonic current is calculated with the corresponding harmonic voltage with its respective impedance phase shift.

Thereby I_5 will be:

$$I_{AB5} = \frac{2\sqrt{3}}{5\pi}\frac{Vcc}{Z_5}\cdot Sen(\omega t - \varphi_5) \tag{3.48}$$

And its effective value:

$$I_{AB5} = \frac{\sqrt{6}}{5\pi}\frac{Vcc}{Z_5} \tag{3.49}$$

And

$$Z_5 = \sqrt{R^2 + (2\pi 5\, fL)^2} \qquad \text{y} \qquad \varphi_5 = arctg\left(\frac{2\pi 5\, fL}{R}\right) \tag{3.50}$$

In Figure 40 shows the Fourier series. Line current (first graph), phase current (second graph) and voltaje (V_{line} = V_{phase}) (third graph) Fourier series with the previous data and with Simulink (Matlab) graphs.

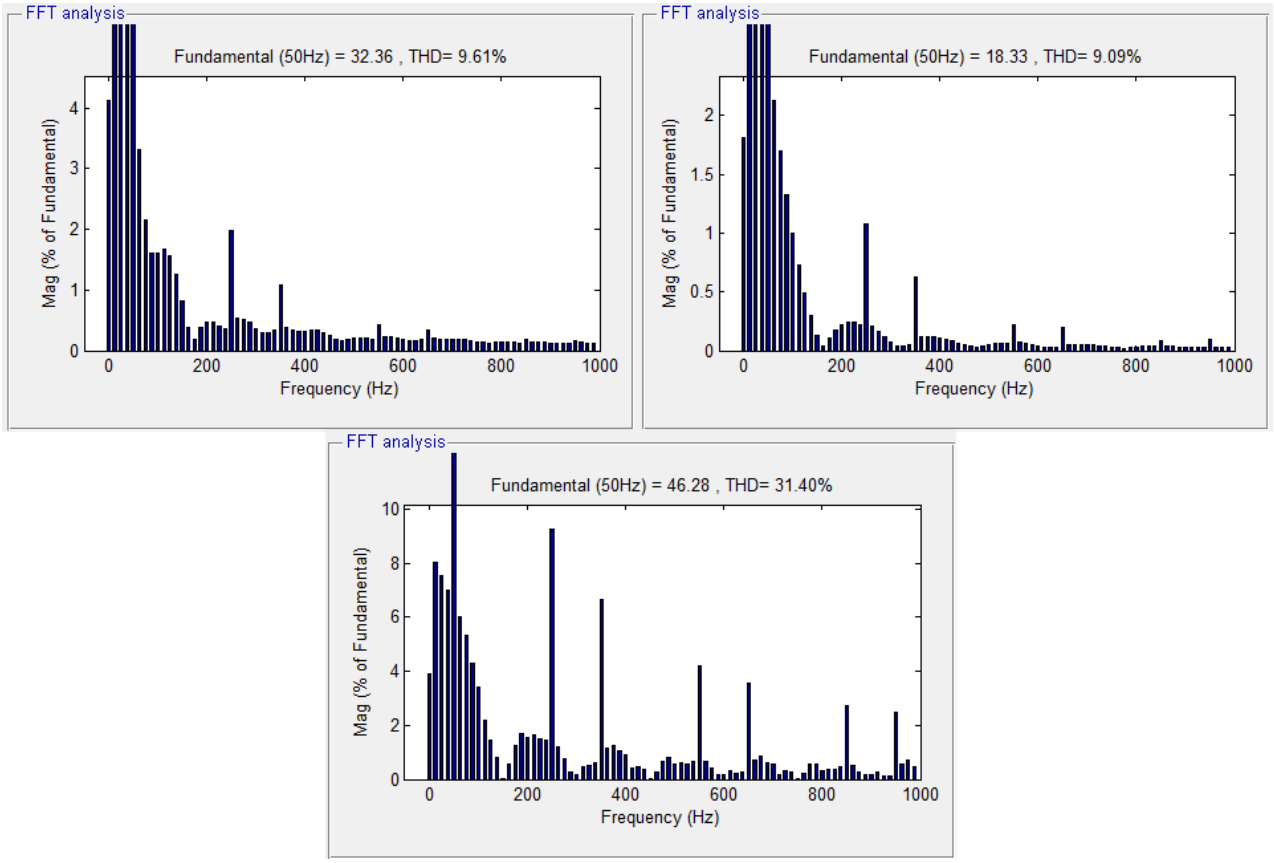


Figure 40 Frequency spectrum of the voltage and current of Three-Phase V.S.I. 6-Step Inverter with Load connected in Triangle (Square Wave Control).

The analysis of the phase currents of the load must be done for each section, being enough to analyze a one half cycle. In this way for I_{AB} we have a stretch of 0 to T/3 (0 to 2π/3) and the other from T/3 to T/2 (2π/3 to π).

1º-between 0 and T/3:

The differential equation for the branch ($_{AB}$) is: (current I_{AB} we will call I).

$$V_{CC} = I_R + L\frac{dI}{dt} \tag{3.51}$$

Which solution is:

$$I = \frac{Vcc}{R} + \left(+ Io - \frac{Vcc}{R} \right) e^{-t/\tau} \tag{3.52}$$

When $\tau = \frac{L}{R}$.

For t = T/3 follows that:

$$I = I^* = \frac{Vcc}{R} + \left(Io - \frac{Vcc}{R} \right) e^{-T/3\tau} \tag{3.53}$$

When I* is the final value of this stretch and will be the initial value for the next stretch.

2º.-Between T/3 and T/2:

Vcc = 0, then:

$$0 = I_R + L\frac{dI}{dt} \tag{3.54}$$

Which solution is:

$$I = I^* \cdot e^{-\left(\frac{t - \frac{T}{3}}{\tau} \right)} \tag{3.55}$$

Replacing I*:

$$I = \left[\frac{Vcc}{R} + \left(Io - \frac{Vcc}{R} \right) e^{-t/3\tau} \right] e^{-\left(\frac{t - T/3}{\tau} \right)} \tag{3.56}$$

For t = T/2 is I = -Io, then we replace in the equation before, we can calculate Io:

$$Io = -\frac{\frac{Vcc}{R}\left(e^{-\frac{T}{6\tau}} - e^{-\frac{T}{2\tau}} \right)}{1 + e^{-\frac{T}{2\tau}}} = -\frac{\frac{Vcc}{R}\left(e^{-\frac{\pi}{3\pi\omega}} - e^{-\frac{\pi}{\pi\omega}} \right)}{1 + e^{-\frac{\pi}{\tau\omega}}} \tag{3.57}$$

Knowing the values of the circuit Vcc, R y L, is calculated Io and then I*, in this way knowing I expressions for each.

In the table 11 you can see the values obtained for the applied load previously connected in triangle.

R = 2Ω, L = 5 mH, Vcc = 48 v, f = 50 Hz

n	fn(Hz)	V_l = V_ph (V)	ZABn(Ω)	I_ph(A)	I_l(A)
1	50	52,93	2,54	20,81	36,05
3	150	0,00	5,12	0,00	0,00
5	250	10,59	8,10	1,31	2,26
7	350	7,56	11,18	0,68	1,17
9	450	0,00	14,28	0,00	0,00
11	550	4,81	17,39	0,28	0,48
13	650	4,07	20,52	0,20	0,34

Table 11 Components of the Fourier series of Three-Phase V.S.I. 6-Step Inverter with Load connected in Triangle (Square Wave Control).

Voltage THD factor:

$$THDv = \frac{\sqrt{\sum_{n=2}^{\infty}(V_{n,\,rms})^2}}{V_{1,\,rms}} \approx \frac{\sqrt{(\frac{10,58}{\sqrt{2}})^2 + (\frac{7,56}{\sqrt{2}})^2 + (\frac{4,81}{\sqrt{2}})^2 + (\frac{4,07}{\sqrt{2}})^2}}{(\frac{52,92}{\sqrt{2}})} = 0,273 = 27,3\%$$

Current THD factor:

$$THDi = \frac{\sqrt{\sum_{n=2}^{\infty}(I_{n,\,rms})^2}}{I_{1,\,rms}} \approx \frac{\sqrt{(\frac{1,13}{\sqrt{2}})^2 + (\frac{0,67}{\sqrt{2}})^2 + (\frac{0,276}{\sqrt{2}})^2 + (\frac{0,198}{\sqrt{2}})^2}}{(\frac{20,81}{\sqrt{2}})} = 0,072 = 7,2\%$$

3.2.4.1.2 STAR LOAD

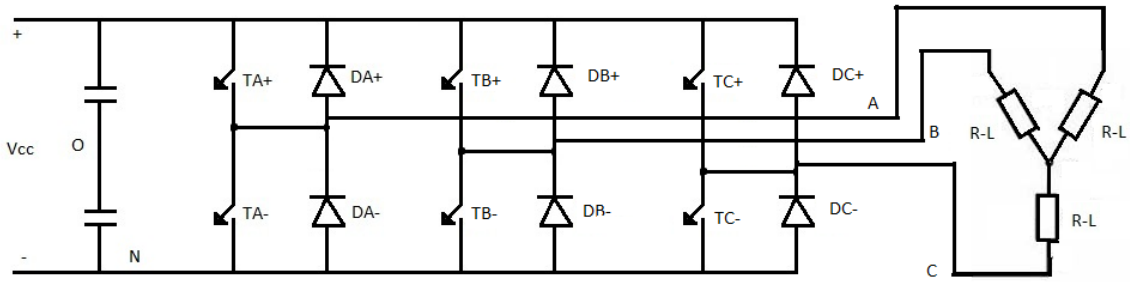


Figure 41 Three-Phase V.S.I. 6-Step Inverter.

The inverter is the same as Figure 39, except that the three-phase load is placed in star as shown in Figure 41.

The transistor keeps its excitation sequence and conduction of 180º and therefore waves of the figure 36 remain intact for this circuit. The bridge output current is equal to the phase current of the load, but the voltage of the charging phase, taken with respect to the star

center is different than that delivered to the bridge. Figure 42 shows the circuit having the load during the fraction of time between 60º and 120º, in which drive the transistors T_{A+}, T_{B-} y T_{C-} and terminals $_B$ and $_C$ are short circuited.

Since the load is equilibrated, the phase voltaje distribution is obtained indicated therein. This circuit changes every 60º in correspondence with the switching of the bridge, resulting in waves of the phase voltages V_{AN}, V_{BN} y V_{CN} of the figure 44.

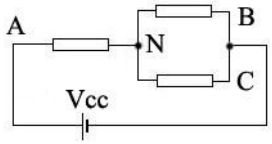


Figure 42 Load Circuit since T/6 to T/3 (π/3 to 2π/3).

Since the voltage waveform of the bridge hasn't changed, the harmonic composition of the voltages and currents, not vary, whatever the load.

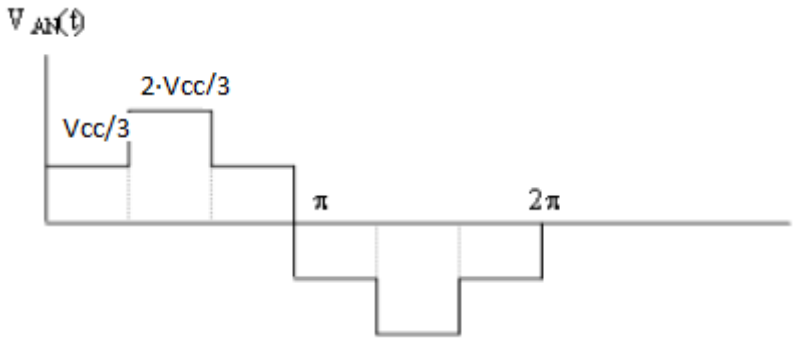


Figure 43 Phase Voltage at the output of Three-Phase V.S.I. 6-Step Inverter Inverter with Load connected in Star (Square Wave Control).

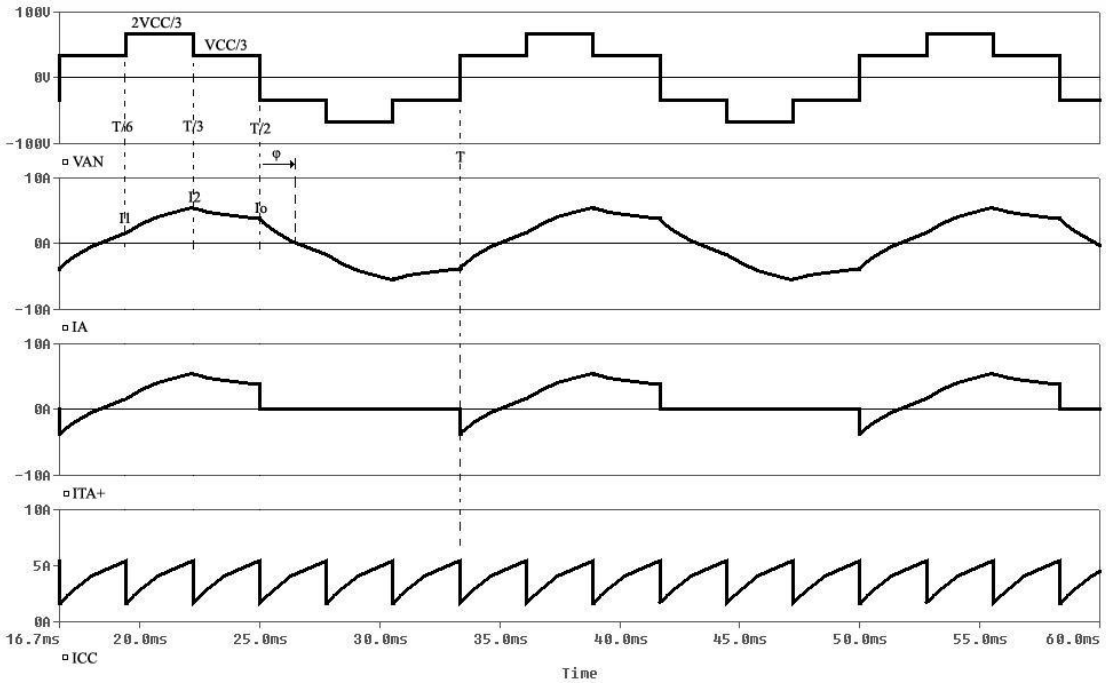


Figure 44 Waves of current in the inverter with Star load.

In Figure 44, the relations between the currents of the circuit is easily deduced. The harmonic content has not changed, but the phase current I_A is composed of three exponential curves in each half cycle.

1º.-Between 0 and T/6 (0 and π/3)

2º.- Between T/6 and T/3 (π/3 and 2π/3)

3º.- Between T/3 and T/2 (2π/3 and π)

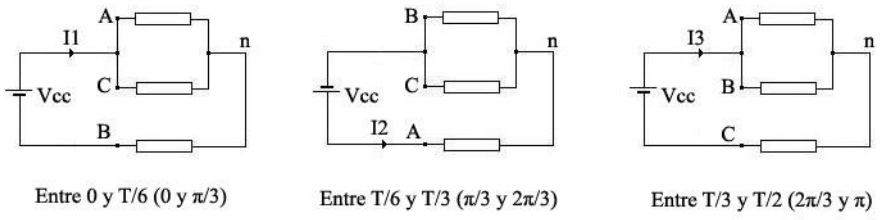


Figure 45 Equivalent circuits for each section.

1º.- The differential equation in this section is:

$$\frac{Vcc}{3} = I_R + L\frac{dI}{dt} \tag{3.58}$$

calculating:

$$I = \frac{Vcc}{3R} + \left(+ Io - \frac{Vcc}{3R} \right) e^{-t/\tau} \tag{3.59}$$

for t = T/6 (ωt = π/3) is i = I_1 (Final value of this curve and initial value for the next one)

2º.-In this part is:

$$\frac{2Vcc}{3} = I_R + L\frac{dI}{dt} \tag{3.60}$$

$$I = \frac{2Vcc}{3R} + \left(I1 - \frac{2Vcc}{3R} \right) e^{-\left(\frac{t-\pi/3}{\tau} \right)} \tag{3.61}$$

for t = T/3 (ωt = 2π/3) is i = I2 (which in turn is the initial value of the next tranche).

3º.-Again it's:

$$\frac{Vcc}{3} = I_R + L\frac{dI}{dt} \tag{3.62}$$

$$I = \frac{Vcc}{3R} + \left(I2 - \frac{Vcc}{3R} \right) e^{-\left(\frac{t-2\pi/3}{\tau} \right)} \tag{3.63}$$

for t = T/2 (ωt = π) is i = -Io (being able to calculate the three constants Io , I_1 and I_2)

The voltage waveform line by line V_{AB} can be expressed in a Fourier series as follows:

$$V_{AB} = \frac{a_0}{2} + \sum_{n=1}^{\infty} (a_n \cos(n\omega t) + b_n sen(n\omega t)) \tag{3.64}$$

Due to the quarter wave symmetry about the x axis, a_0 and a_n are zero. Assuming symmetry about the y axis en ωt = π/6, b_n can be written as follows:

$$Bn = \frac{1}{\pi}\left[\int_{-5\pi/6}^{-\pi/6} -Vcc \cdot d(\omega t) + \int_{\pi/6}^{-5\pi/6} Vcc \cdot d(\omega t) \right] = \frac{4Vcc}{n\pi} sen(\frac{n\pi}{2}) sen(\frac{n\pi}{3}) \tag{3.65}$$

Thereby, recognizing that the phase of V_{AB} is shifted π/6, and that the odd harmonics are zero, the instantaneous voltage V_{AB} is obtained between lines:

$$V_{AB} = \sum_{n=1,3,5,\ldots}^{\infty} \frac{4Vcc}{n\pi} sen(\frac{n\pi}{3}) sen(n(\omega t + \frac{\pi}{6})) \tag{3.66}$$

Both V_{BC} and V_{CA} can be determined with the above equation shifting 120º and 240º to V_{AB} respectively:

$$V_{BC} = \sum_{n=1,3,5,\ldots}^{\infty} \frac{4Vcc}{n\pi} sen(\frac{n\pi}{3}) sen(n(\omega t - \frac{\pi}{2})) \quad (3.67)$$

$$V_{CA} = \sum_{n=1,3,5,\ldots}^{\infty} \frac{4Vcc}{n\pi} sen(\frac{n\pi}{3}) sen(n(\omega t - \frac{7\pi}{6})) \quad (3.68)$$

It can be seen in equation 3-76 y 3-77, that the triple harmonic (n = 3, 9, 15, ...) would be zero in the line to line voltages.

The rms line to line voltage can be calculated:

$$VL = \sqrt{\frac{2}{2\pi} \int_0^{2\pi/3} Vcc^2 \cdot d(\omega t)} = \sqrt{\frac{2}{3}} Vcc \quad (3.69)$$

As seen in equation 3-75, the n-th component of the rms line voltage is:

$$VLn = \frac{4Vcc}{\sqrt{2}n\pi} sen(\frac{n\pi}{3}) \quad (3.70)$$

for n = 1 is the rms line voltaje of the fundamental harmonic:

$$VL_1 = \frac{4Vcc \cdot sen60^o}{\sqrt{2}\pi} \quad (3.71)$$

The rms value of the line-to-neutral voltages can be determined from the line voltage:

$$Vp = \frac{VL}{\sqrt{3}} = \frac{\sqrt{2}Vcc}{3} \quad (3.72)$$

With a resistive load, the diodes across transistors have no function. If the load is inductive, the current in each arm of the inverter would be delayed with respect to the voltage, as shown in Figure 44. When transistor T_{A-} is off, the only way the negative line current I_A is through D_{A+}, until the load current reverses polarity when. During this period, the transistor T_{A+} can't drive. Similarly, the transistor T_{A-} only start driving again when the current passes through zero and reverse polarity. Must to continuously control the control transistor, because time driving transistors and diodes depends on the power factor of the load.

For a connected load, the phase voltage is $V_{AN} = \frac{Vab}{\sqrt{3}}$, with delay of 30º in respect of V_{AB}. Consequently, the instantaneous phase voltages are:

$$V_{AN} = \sum_{n=1}^{\infty} \frac{4Vcc}{\sqrt{3}n\pi} sen(\frac{n\pi}{3}) sen(n\omega t) \qquad \text{para n = 1, 3,5,...} \quad (3.73)$$

$$V_{BN} = \sum_{n=1}^{\infty} \frac{4Vcc}{\sqrt{3}n\pi} sen(\frac{n\pi}{3}) sen(n(\omega t - \frac{2\pi}{3})) \qquad \text{para n = 1, 3,5,...} \qquad (3.74)$$

$$V_{CN} = \sum_{n=1}^{\infty} \frac{4Vcc}{\sqrt{3}n\pi} sen(\frac{n\pi}{3}) sen(n(\omega t - \frac{4\pi}{3})) \qquad \text{para n = 1, 3,5,...} \qquad (3.75)$$

The instantaneous phase voltage V_{AN} is divided between the load impedance

$$Z = R + jn\omega L \qquad (3.76)$$

Using the equation 3-82, line current I_A is:

$$I_A = \sum_{n=1,3,5,...}^{\infty} \left[\frac{4Vcc}{\sqrt{3}n\pi\sqrt{R^2 + (n\omega L)^2}} sen(\frac{n\pi}{3}) \right] sen(n\omega t - \varphi n) \qquad \varphi = arctg\left(\frac{2n\pi fL}{R} \right) \qquad (3.77)$$

We work with the following data:

R = 2Ω, L = 5 mH, Vcc = 48 v, f = 50 Hz

n	fn(Hz)	V_ph (V)	ZABn (Ω)	I_l = I_ph (A)	V_l (V)
1	50	30,56	2,54	12,02	52,93
3	150	0,00	5,12	0,00	0,00
5	250	6,11	8,10	0,75	10,59
7	350	4,37	11,18	0,39	7,56
9	450	0,00	14,28	0,00	0,00
11	550	2,78	17,39	0,16	4,81
13	650	2,35	20,52	0,11	4,07

Table 12 Coefficients of the Fourier series of Three-Phase V.S.I. 6-Step Inverter with Load connected in Star (Square Wave Control)..

Voltage THD factor:

$$THDv = \frac{\sqrt{\sum_{n=2}^{\infty} (V_{n,\,rms})^2}}{V_{1,\,rms}} \approx \frac{\sqrt{(\frac{6,11}{\sqrt{2}})^2 + (\frac{4,37}{\sqrt{2}})^2 + (\frac{2,78}{\sqrt{2}})^2 + (\frac{2,35}{\sqrt{2}})^2}}{(\frac{30,56}{\sqrt{2}})} = 0,273 = 27,3\%$$

Current THD factor:

$$THDi = \frac{\sqrt{\sum_{n=2}^{\infty}(I_{n,\,rms})^2}}{I_{1,\,rms}} \approx \frac{\sqrt{(\frac{0,75}{\sqrt{2}})^2 + (\frac{0,39}{\sqrt{2}})^2 + (\frac{0,16}{\sqrt{2}})^2 + (\frac{0,11}{\sqrt{2}})^2}}{(\frac{12,02}{\sqrt{2}})} = 0,072 = 7,2\%$$

As could be imagine the harmonic content in both star and triangle is the same (THDv and THDi) the only difference is that the amplitudes of the voltage and current magnitudes. (THDi magnitudes are somewhat different for the margin of error in the accounts).

In Figure 46 shows the Fourier series. Line Voltage (first graph), phase Voltage (second graph) and Current (I_{line} = I_{phase}) (third graph) Fourier series with the previous data and with Simulink (Matlab) graphs.

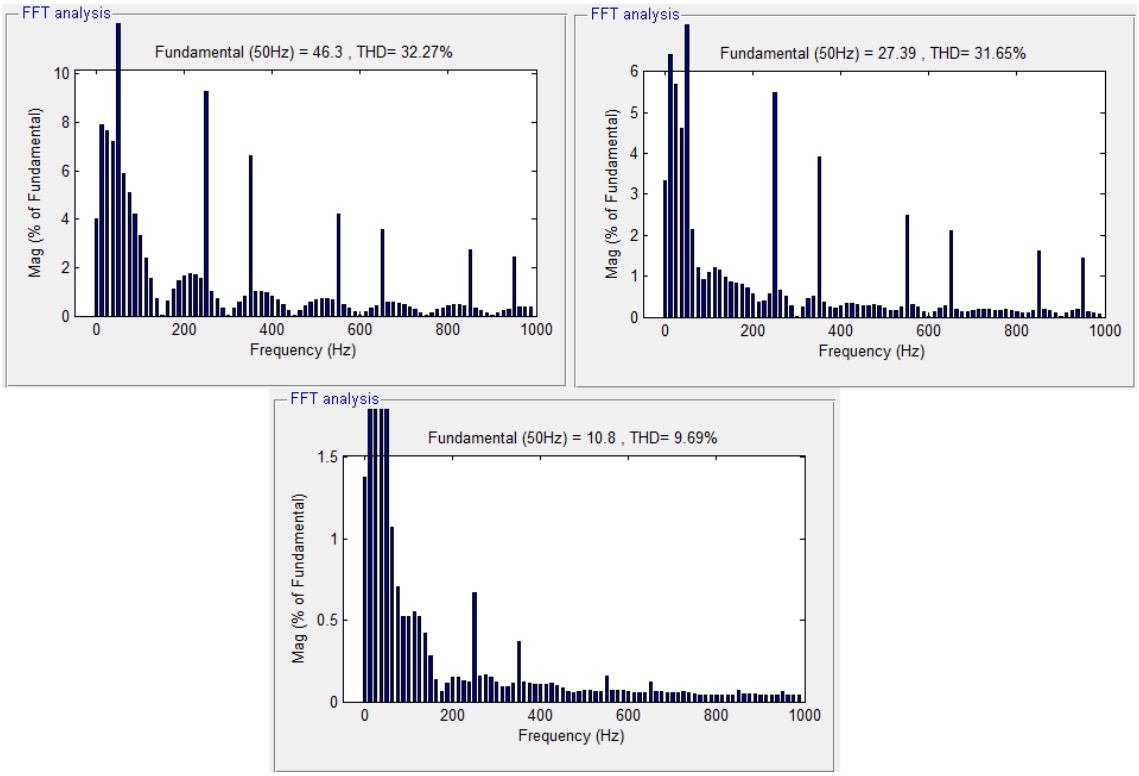


Figure 46 Frequency spectrum of the voltage and current of Three-Phase V.S.I. 6-Step Inverter with Load connected in Star (Square Wave Control).

3.2.4.2 THREE-PHASE V.S.I. 6-STEP INVERTER (PWM CONTROL)

Just like the single-phase inverters, the technique consists in obtaining three sinusoidal voltages, controllable magnitude and frequency from a voltage source input. For completely balanced three-phase voltage control commands for switches generate similarly to single phase case: the same triangular wave was compared with three sine of the same frequency (three waves modulating (V_{sinA}, V_{sinB}, V_{sinC}) and offset from one another 120º, As can be seen in Figure 47.

The control signals for the switches will be obtained as follows:

$$V_{\sin A} > V_{tri} \rightarrow T_{A+(ON)} \quad V_{AN} = Vcc$$

$$V_{\sin A} < V_{tri} \rightarrow T_{A-(ON)} \quad V_{AN} = 0$$

$$V_{\sin B} > V_{tri} \rightarrow T_{B+(ON)} \quad V_{BN} = Vcc$$

$$V_{\sin B} < V_{tri} \rightarrow T_{B-(ON)} \quad V_{BN} = 0$$

$$V_{\sin C} > V_{tri} \rightarrow T_{C+(ON)} \quad V_{CN} = Vcc$$

$$V_{\sin C} < V_{tri} \rightarrow T_{C-(ON)} \quad V_{CN} = 0$$

Waveforms V_A, V_B and line voltage V_{AB} can be seen in the Figure 48.

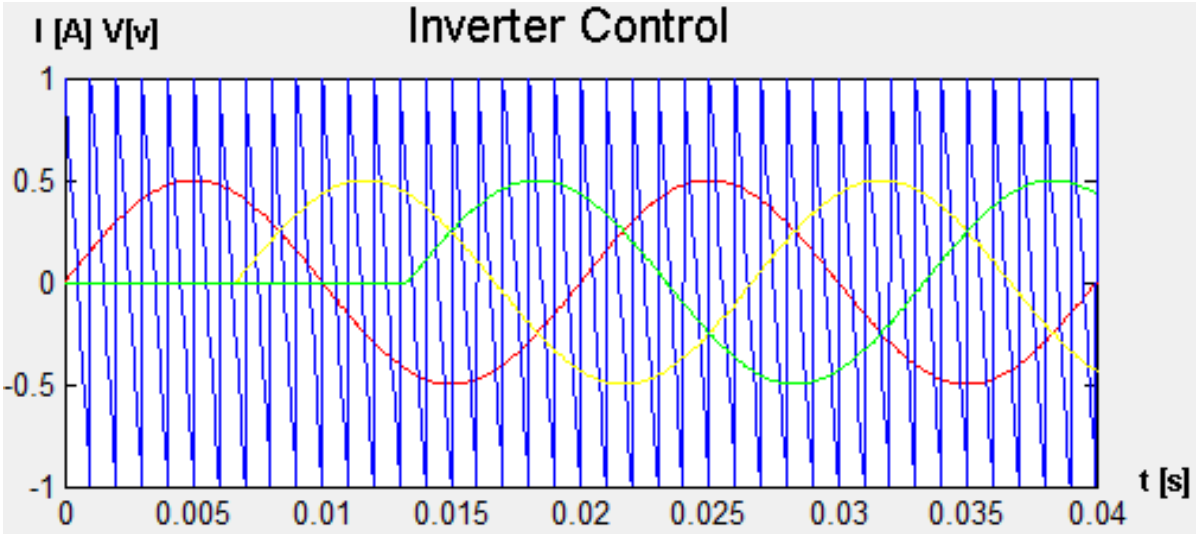


Figure 47 Waveforms of control switches for Three-Phase case (PWM Control).

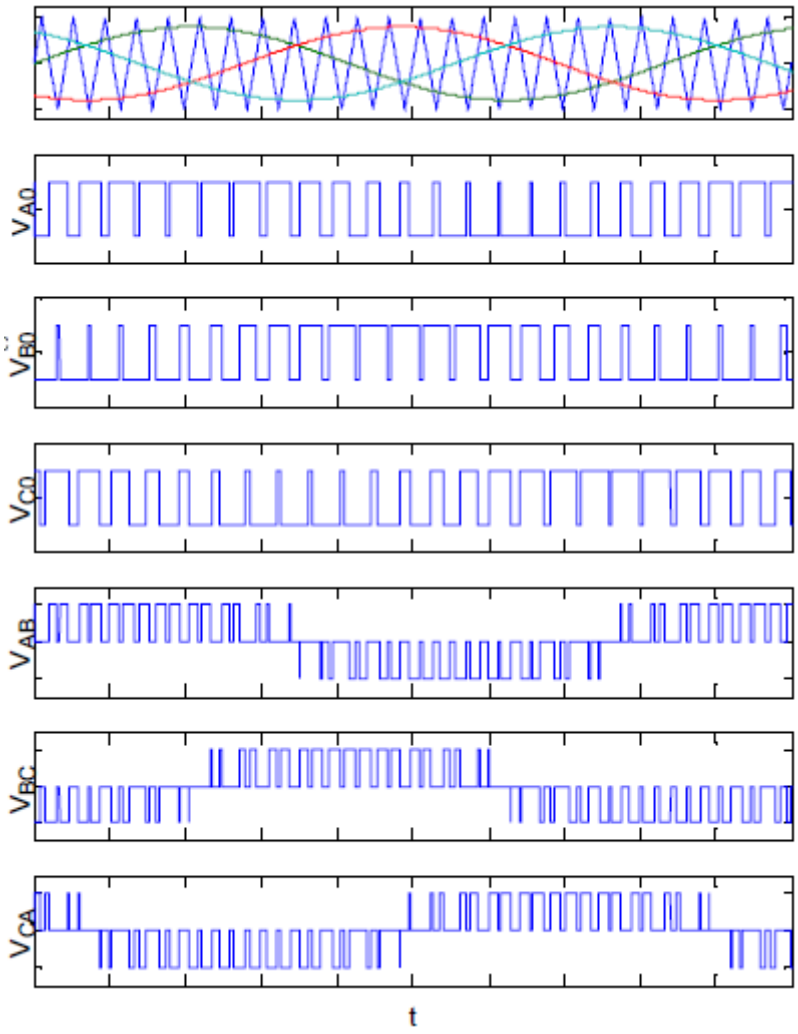


Figure 48 Width modulation three-phase inverter sine pulse.

In the output voltages of each branch (V_{XN}), There is a level of DC, to be always positive. However, this voltage disappears naturally in line voltages, (can be seen in Figure 48 like example of V_{AB}) and that all branches have the exact same DC level. This case is identical to the full bridge.

For the case of three-phase inverters, it's necessary to know only the harmonics that appear in the line voltage. In the frequency spectrum of the voltages V_{AN}, V_{BN}, and V_{CN} there are the odd harmonics sidebands of m_f and its multiples assuming odd m_f.

Considering m_f harmonics order (and the same applies to its odd multiples) the phase difference between the corresponding V_{AN} and of V_{BN} is $120^o \cdot m_f$; therefore, this difference will be zero if m_f is odd and a multiple of 3 $(120^o \cdot (2n+1) \cdot 3 \Leftrightarrow 360^o)$. Therefore, the spectrum is canceled of V_{AB}.

Arguing similarly, it's concluded that the odd harmonic multiples of m_f cancel out the line voltage if m_f and odd multiple of 3 is chosen. m_f is selected as odd multiples for 3 to remain odd numbers and therefore disappear even order harmonics.

Values of m_f and m_a don't differ greatly from those already exposed to the single phase case; can be summarized in the following sentences:

- For low values of m_f, in order to eliminate even harmonics synchronous PWM should be used with odd and integer m_f.

- Value of m_f must be multiple of 3, to eliminate the most relevant amplitude harmonics.

- The slopes of the carrier wave and modulating must be opposed in the zero crossings.

- For high values of m_f, the considerations are valid for single-phase.

- If working with overmodulation, it must respect the criteria for low values of m_f, regardless of the value of this.

3.2.4.2.1 OVERMODULATION PWM CONTROL

In the linear region, ($m_a \leq 1$) harmonic amplitude of the fundamental frequency varies linearly with m_a index. The analysis in the previous sections, we have the amplitude of the fundamental harmonic of the voltage V_{AN} that:

$$V_{AN1} = m_a \cdot \frac{Vcc}{2} \tag{3.78}$$

Therefore, the effective value of the line voltage to the fundamental frequency, as the voltages are offset 120º:

$$V_{AB} = \frac{\sqrt{3}}{2 \cdot \sqrt{2}} \cdot m_a \cdot Vcc \approx 0{,}612 \cdot m_a \cdot Vcc \tag{3.79}$$

Then it's possible to calculate the effective value of the harmonic without further multiply the table values normalized by the factor 0,612. The result is shown in the table 13.

h/m_a	0,2	0,4	0,6	0,8	1
FUNDAMENTAL	0,122	0,245	0,367	0,49	0,612
$m_f \pm 2$	0,01	0,037	0,08	0,135	0,195
$m_f \pm 4$				0,005	0,011
$2m_f \pm 1$	0,116	0,2	0,227	0,192	0,111
$2m_f \pm 5$				0,008	0,02
$3m_f \pm 2$	0,027	0,085	0,124	0,108	0,038
$3m_f \pm 4$		0,007	0,029	0,064	0,096
$4m_f \pm 1$	0,1	0,096	0,005	0,064	0,042
$4m_f \pm 5$			0,021	0,051	0,073
$4m_f \pm 7$				0,01	0,03

Table 13 Normalized rms values Vn/Vcc of the different harmonics.

For the case of modulation levels higher unit $m_a > 1$, the amplitude of the fundamental harmonic no longer varies linearly with that parameter. In Figure 49 it can be seen the variation of the effective value of the main harmonic depending on m_a. As can be seen, from the unit, the relationship ceases to be linear up to the maximum value is reached 0,78 ($0,78 = \frac{\sqrt{6}}{\pi}$), which corresponds to the square wave scheme. With this type of control appear more harmonics as sidebands of m_f and their multiples. However, the main harmonics have a smaller amplitude than in the previous case. Therefore, the losses due to unwanted harmonics will not be as high as inferred from the presence of more harmonics. Depending on the nature of the load and switching frequency, the losses due to these harmonics in overmodulation may be even lower than those produced in the linear zone PWM.

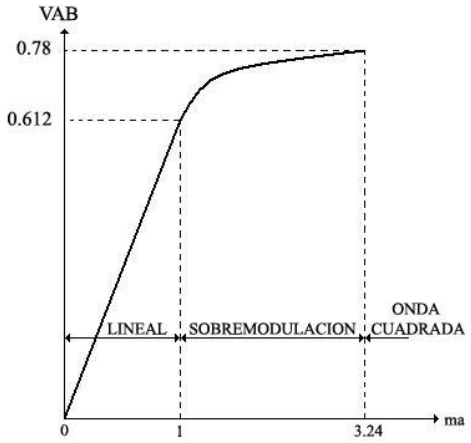


Figure 49 Variation of the normalized value of the fundamental harmonic with m_a.

3.2.4.2.2 TRIANGLE LOAD

The circuit used is shown in Figure 50. Since the load is connected in triangle, the bridge output voltage are coincident with the phase voltages of that load, but the phase currents differ from the bridge output.

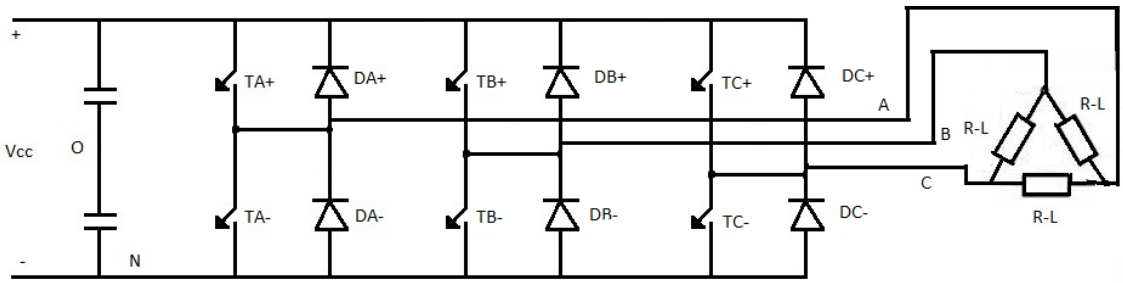


Figure 50 Three-Phase V.S.I. 6-Step Inverter with Load connected in Triangle.

In Figure 51 it can be seen the waveform of the voltage and the half-sine form of the current through the load.

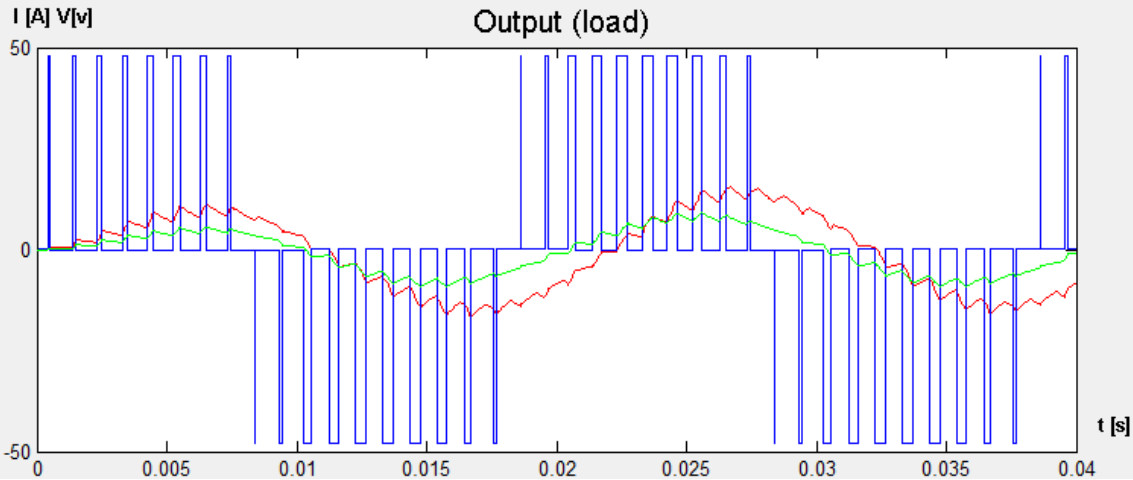


Figure 51 Waveform of voltage and current output of Three-Phase V.S.I. 6-Step Inverter with load connected in triangle.

In Figure 52 shows the Fourier series. Line current (first graph), phase current (second graph) and voltaje (V_{line} = V_{phase}) (third graph) Fourier series with the previous data and with Simulink (Matlab) graphs.

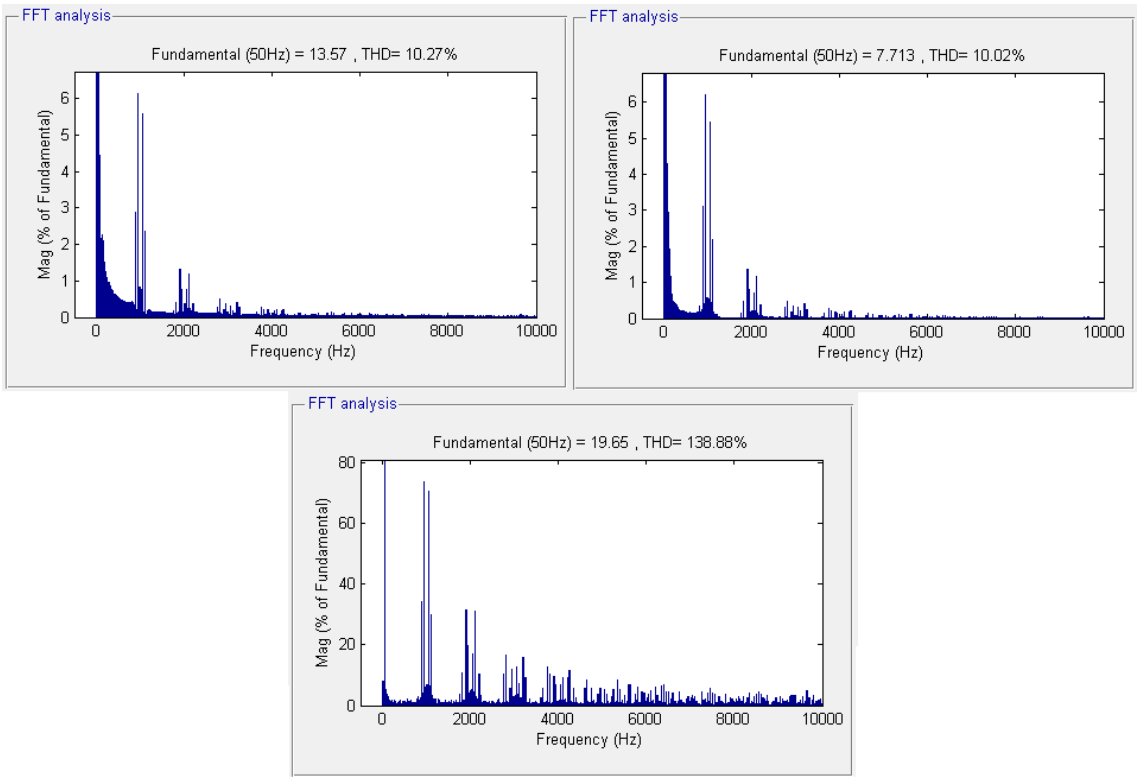


Figure 52 Frequency spectrum of the voltage and current of Three-Phase V.S.I. 6-Step Inverter with Load connected in Triangle (PWM Control).

We work with the following data:

R = 2Ω, L = 5 mH, Vcc = 48 v, f = 50 Hz, m_a = 0,5 y m_f = 20 (f_{TRI} = (20) (50)=1000Hz)

As the table 14 summarizes the frequencies of the voltages, currents and the resulting impedance for the first harmonics:

n	fn(Hz)	V_ph = V_l (V)	ZABn (Ω)	I_ph (A)	I_l (A)
1	50	20,32	2,54	7,99	13,84
18	900	3,82	28,34	0,13	0,23
22	1100	3,82	34,62	0,11	0,19
39	1950	14,55	61,29	0,24	0,41
41	2050	14,55	64,43	0,23	0,39
58	2900	6,41	91,13	0,07	0,12
62	3100	6,41	97,41	0,07	0,11

Table 14 Coefficients of the Fourier series of Three-Phase V.S.I. 6-Step Inverter with Load connected in Triangle (PWM Control).

THD factor (DAT) of the load current is calculated, approximating the rms current of the first harmonics by the terms listed in the table 14:

$$THDi = \frac{\sqrt{\sum_{n=2}^{\infty} (I_{n,\,rms})^2}}{I_{1,\,rms}}$$

$$\approx \frac{\sqrt{(\frac{0,13}{\sqrt{2}})^2 + (\frac{0,11}{\sqrt{2}})^2 + (\frac{0,24}{\sqrt{2}})^2 + (\frac{0,23}{\sqrt{2}})^2 + (\frac{0,07}{\sqrt{2}})^2 + (\frac{0,07}{\sqrt{2}})^2}}{(\frac{7,99}{\sqrt{2}})} = 0,0387 = 3,87\%$$

Voltage THD factor:

$$THDv = \frac{\sqrt{\sum_{n=2}^{\infty} (V_{n,\,rms})^2}}{V_{1,\,rms}}$$

$$\approx \frac{\sqrt{(\frac{3,82}{\sqrt{2}})^2 + (\frac{3,82}{\sqrt{2}})^2 + (\frac{14,55}{\sqrt{2}})^2 + (\frac{14,55}{\sqrt{2}})^2 + (\frac{6,41}{\sqrt{2}})^2 + (\frac{6,41}{\sqrt{2}})^2}}{(\frac{20,32}{\sqrt{2}})} = 0,8843 = 88,43\%$$

As could be imagine the harmonic content in both star and triangle is the same (THDv and THDi) the only difference is that the amplitudes of the voltage and current magnitudes. (THDi magnitudes are somewhat different for the margin of error in the accounts).

3.2.4.2.3 STAR LOAD

The circuit used is shown in Figure 53. With this circuit the voltage is applied to the connected load is the line voltage, ie V_{AN}, V_{BN} and $V_{CN,}$ when $_N$ is the neutral of the star load. In Figure 48 are represented V_{AN} and $V_{BN.}$

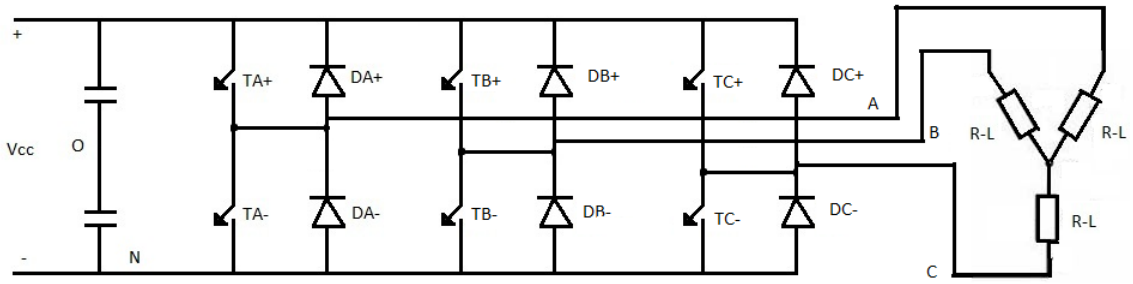


Figure 53 Three-Phase V.S.I. 6-Step Inverter with load connected in Star.

We work with the following data:

R = 2Ω, L = 5 mH, Vcc = 48 v, f = 50 Hz, m_a = 0,5 y m_f = 20 (f_{TRI} = (20) (50)=1000Hz)

$$V_{AN1} = m_a \cdot \frac{Vcc}{2} = 0{,}5 \cdot \frac{48}{2} = 12v$$

$$I_{AN} = \frac{V_{AN}}{Z_{AN}} = \frac{V_n}{\sqrt{R^2 + (n \cdot \omega_0 L)^2}}$$

For the fundamental frequency:

$$I_{AN1} = \frac{12}{\sqrt{2^2 + (1 \cdot 2\pi 50 \cdot 0{,}005)^2}} = 4{,}71A$$

As the table 15 summarizes the frequencies of the voltages, currents and the resulting impedance for the first harmonics:

n	fn(Hz)	V_l (V)	ZABn (Ω)	I_l = I_ph (A)	V_ph (V)
1	50	20,32	2,54	7,99	11,73
18	900	3,82	28,34	0,13	2,20
22	1100	3,82	34,62	0,11	2,20
39	1950	14,55	61,29	0,24	8,40
41	2050	14,55	64,43	0,23	8,40
58	2900	6,41	91,13	0,07	3,70
62	3100	6,41	97,41	0,07	3,70

Table 15 Coefficients of the Fourier series of Three-Phase V.S.I. 6-Step Inverter with Load connected in Star (PWM Control).

THD factor (DAT) of the load current is calculated, approximating the rms current of the first harmonics by the terms listed in the table 15:

$$THDi = \frac{\sqrt{\sum_{n=2}^{\infty} (I_{n,\,rms})^2}}{I_{1,\,rms}}$$

$$\approx \frac{\sqrt{(\frac{0{,}13}{\sqrt{2}})^2 + (\frac{0{,}11}{\sqrt{2}})^2 + (\frac{0{,}24}{\sqrt{2}})^2 + (\frac{0{,}23}{\sqrt{2}})^2 + (\frac{0{,}07}{\sqrt{2}})^2 + (\frac{0{,}07}{\sqrt{2}})^2}}{(\frac{7{,}99}{\sqrt{2}})} = 0{,}0387 = 3{,}87\%$$

Voltage THD factor:

$$THDv = \frac{\sqrt{\sum_{n=2}^{\infty} (V_{n,\,rms})^2}}{V_{1,\,rms}}$$

$$\approx \frac{\sqrt{(\frac{3,82}{\sqrt{2}})^2+(\frac{3,82}{\sqrt{2}})^2+(\frac{14,55}{\sqrt{2}})^2+(\frac{14,55}{\sqrt{2}})^2+(\frac{6,41}{\sqrt{2}})^2+(\frac{6,41}{\sqrt{2}})^2}}{(\frac{20,32}{\sqrt{2}})} = 0,8843 = 88,43\%$$

Using the development truncated Fourier series of the table 15, the THD factor is underestimated. However, as the load impedance increases and the harmonic amplitudes generally decrease as n increases, the previous estimate should be acceptable. In Figure 54 it can be seen the waveform of the voltage and the half-sine form of the current through the load.

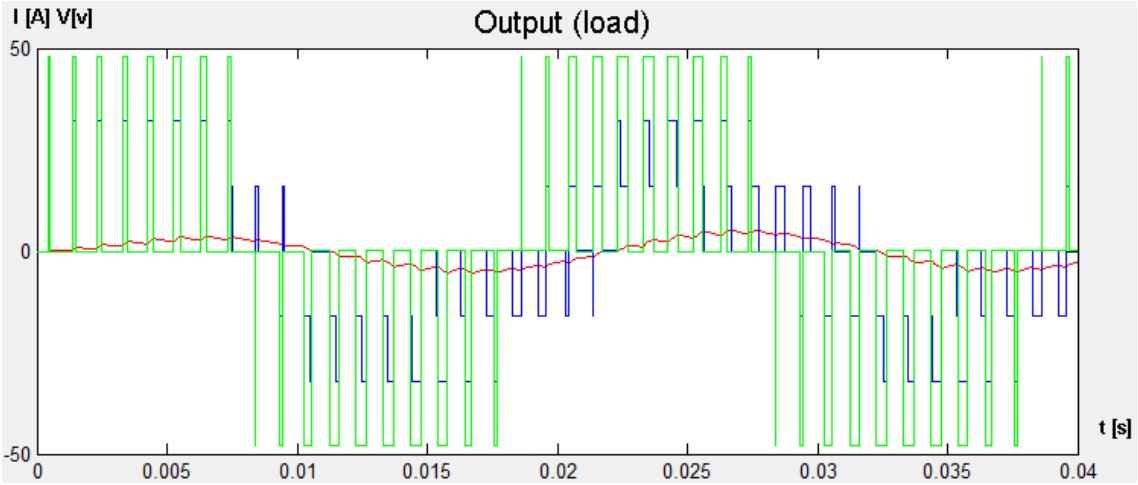


Figure 54 Voltage and current output of Three-Phase V.S.I. 6-Step Inverter with Load connected in Star (PWM Control).

In Figure 55 shows the Fourier series. Line Voltage (first graph), phase Voltage (second graph) and Current (I_{line} = I_{phase}) (third graph) Fourier series with the previous data and with Simulink (Matlab) graphs.

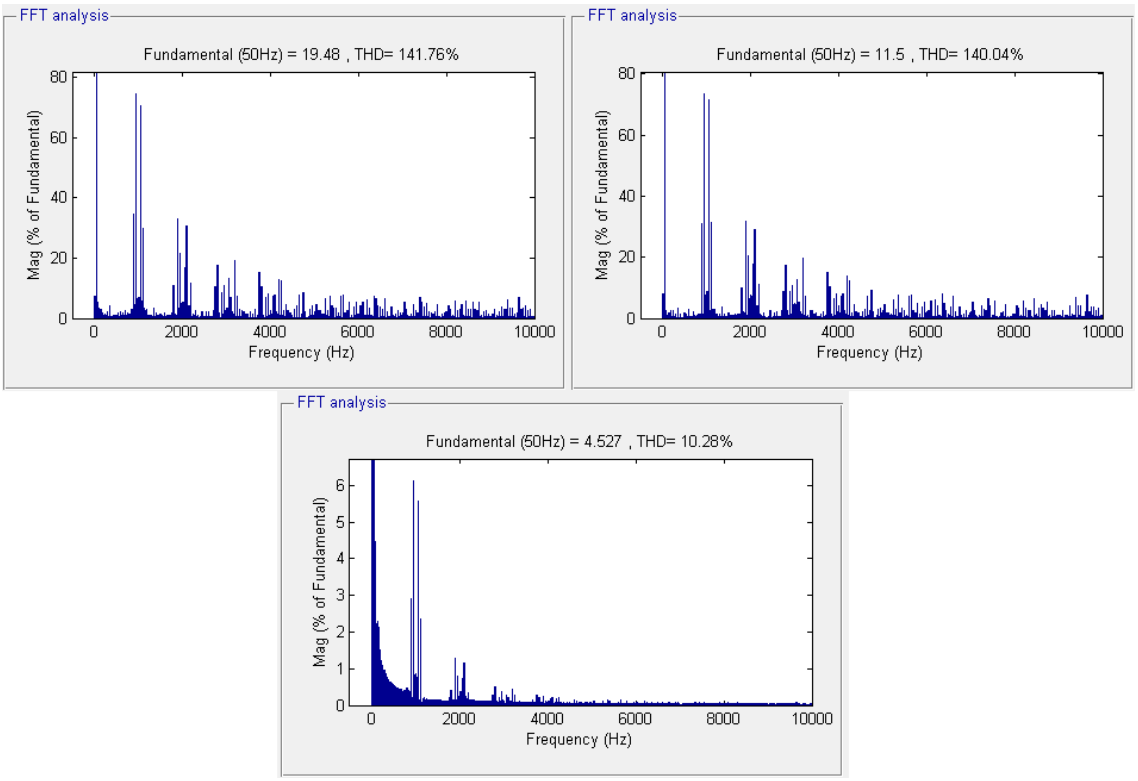


Figure 55 Frequency spectrum of the voltage and current of Three-Phase V.S.I. 6-Step Inverter with Load connected in Star (PWM Control).

3.2.4.3 THREE-PHASE V.S.I. 6-STEP INVERTER (HYSTERESIS CONTROL)

The current control by hysteresis band in three-phase circuits is performed by three current references displaced 120º (a reference for each phase of the inverter). For each phase, the current measurement is compared to the reference current defining a band (called hysteresis band). If the current measurement is within the band, inverter status doesn't change. If the current measurement is outside the band, it applies inverter status that makes the current re-enter to the band. The voltage applied to the phase will be positive if the current increases and negative in the opposite case.

To understand the operation of an easier way, the following figure (Figure 56) shows a single phase inverter with hysteresis control, which occurs previously explained:

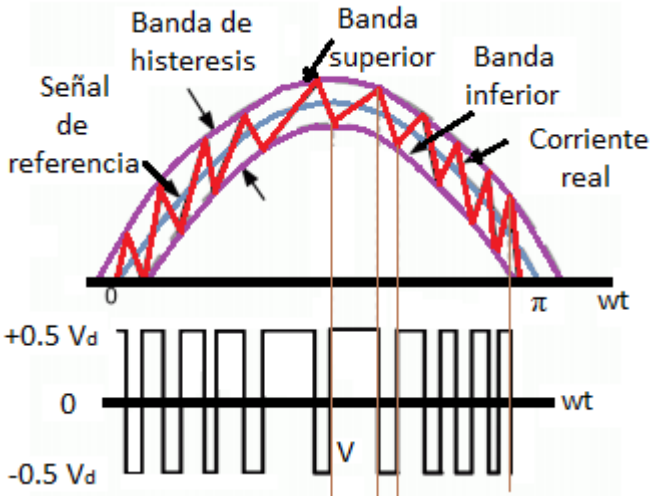


Figure 56 Waveforms Hysteresis Control.

3.2.4.3.1 STAR LOAD

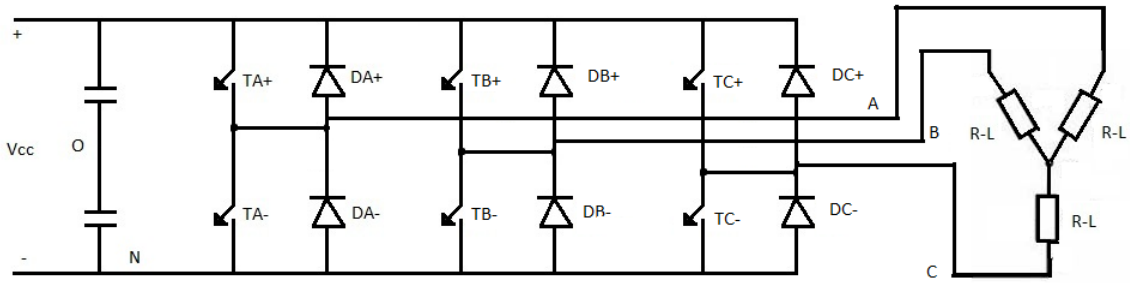


Figure 57 Three-Phase V.S.I. 6-Step Inverter with Load connected in Star (Hysteresis Control).

To perform all calculations on this inverter, the same procedure applies in the case of three-phase inverter with square wave control (See the previously explained theoretical development).

We work with the following data:

$$R = 2\Omega,\ L = 5\ mH,\ Vcc = 48\ v,\ f = 50\ Hz$$

By performing all calculations, these results are obtained:

n	fn(Hz)	V_ph (V)	ZABn (Ω)	I_l = I_ph (A)	V_l (V)
1	50	30,56	2,54	12,02	52,93
3	150	0,00	5,12	0,00	0,00
5	250	6,11	8,10	0,75	10,59
7	350	4,37	11,18	0,39	7,56
9	450	0,00	14,28	0,00	0,00
11	550	2,78	17,39	0,16	4,81
13	650	2,35	20,52	0,11	4,07

Table 16 Coefficients of the Fourier series of Three-Phase V.S.I. 6-Step Inverter with Load connected in Star (Hysteresis Control).

Here can be seen the waveform at the output of the load with this type of control (Figure 58):

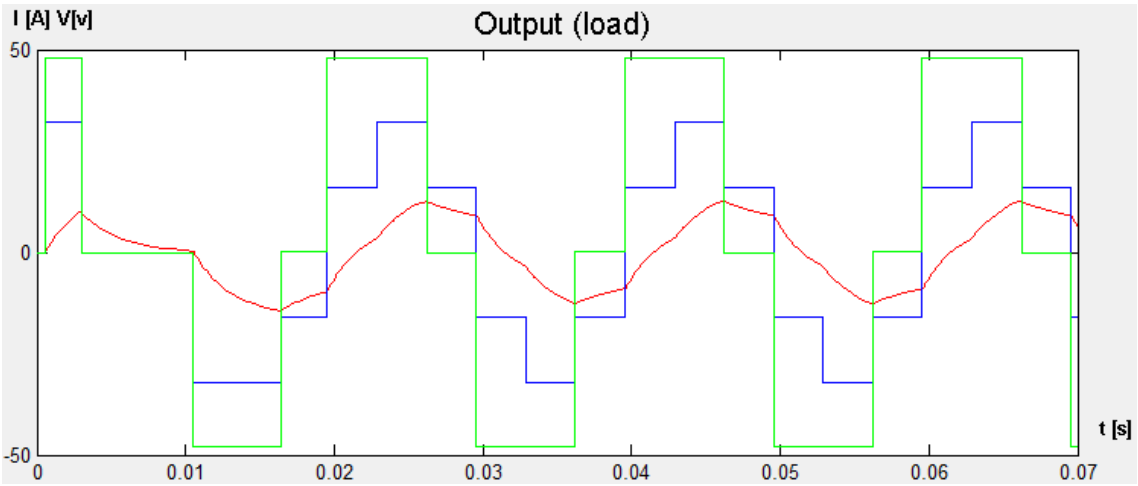


Figure 58 Voltage and current output of Three-Phase V.S.I. 6-Step Inverter with Load connected in Star (Hysteresis Control).

The yellow line represents the line voltage, the blue line corresponds to the phase voltage, and red is the current through the load.

Voltage THD factor:

$$THDv = \frac{\sqrt{\sum_{n=2}^{\infty} (V_{n,\,rms})^2}}{V_{1,\,rms}} \approx \frac{\sqrt{(\frac{6,11}{\sqrt{2}})^2 + (\frac{4,37}{\sqrt{2}})^2 + (\frac{2,78}{\sqrt{2}})^2 + (\frac{2,35}{\sqrt{2}})^2}}{(\frac{30,56}{\sqrt{2}})} = 0,273 = 27,3\%$$

Current THD factor:

$$THDi = \frac{\sqrt{\sum_{n=2}^{\infty} (I_{n,\,rms})^2}}{I_{1,\,rms}} \approx \frac{\sqrt{(\frac{0,75}{\sqrt{2}})^2 + (\frac{0,39}{\sqrt{2}})^2 + (\frac{0,16}{\sqrt{2}})^2 + (\frac{0,11}{\sqrt{2}})^2}}{(\frac{12,02}{\sqrt{2}})} = 0,072 = 7,2\%$$

As could be imagine the harmonic content in both star and triangle is the same (THDv and THDi) the only difference is that the amplitudes of the voltage and current magnitudes. (THDi magnitudes are somewhat different for the margin of error in the accounts).

In Figure 59 shows the Fourier series. Line Voltage (first graph), phase Voltage (second graph) and Current (I_{line} = I_{phase}) (third graph) Fourier series with the previous data and with Simulink (Matlab) graphs.

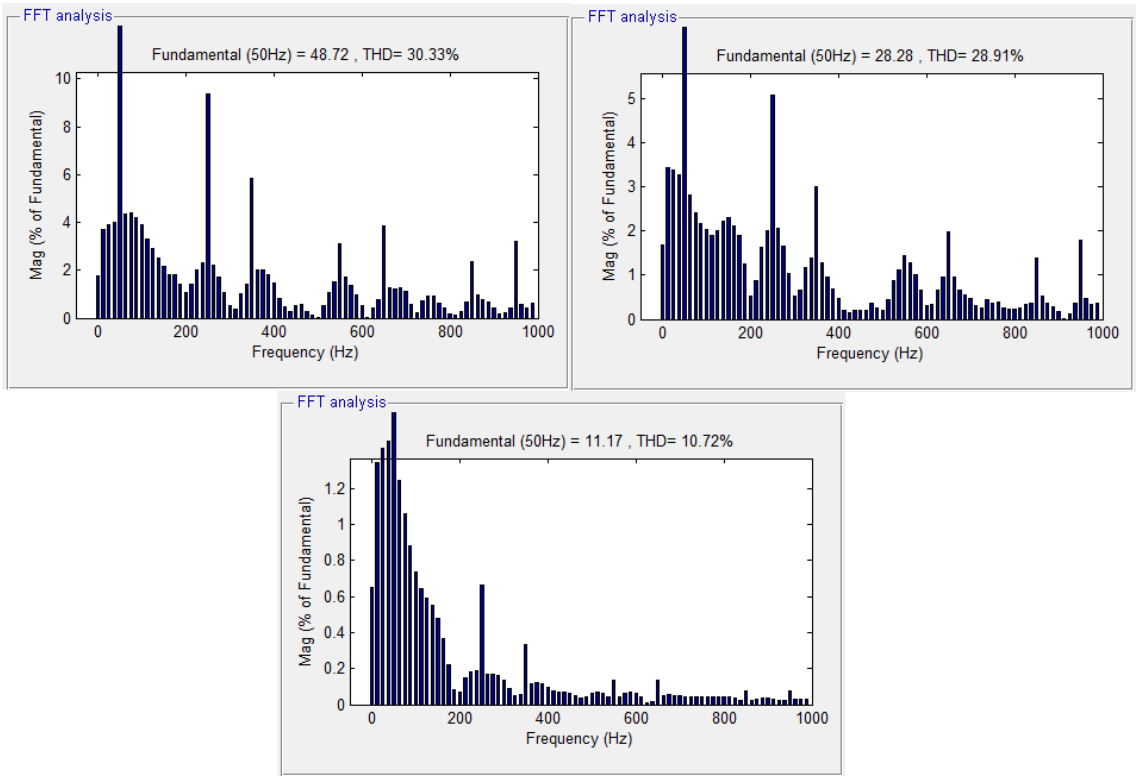


Figure 59 Frequency spectrum of the voltage and current of Three-Phase V.S.I. 6-Step Inverter with Load connected in Star (Hysteresis Control).

3.2.5 PUSH-PULL INVERTERS

3.2.5.1 PUSH-PULL INVERTER (SQUARE WAVE CONTROL)

The following figure (Figure 60) shows a Push-Pull inverter:

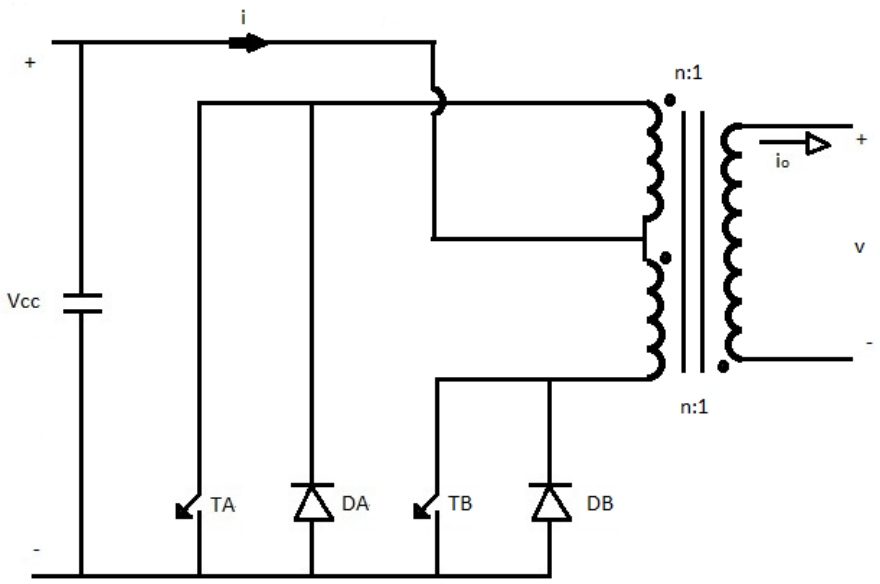


Figure 60 Push-Pull Inverter.

This inverter operates as follows:

$$\text{(TA) is ON and (TB) is OFF} \rightarrow V = \frac{Vcc}{n}$$

$$\text{(TB) is ON and (TA) is OFF} \rightarrow V = -\frac{Vcc}{n}$$

And:

$$V_{PEAKVOLTAGE} = \frac{4 \cdot Vcc}{\pi \cdot n} \tag{3.80}$$

Theoretically it's like a Half-Wave Inverter (Square Wave Control), but in the simulation there are voltage peaks simulation error.

To perform all calculations on this inverter, the same procedure applies in the case of Half-Wave Inverter (Square Wave Control) (See the previously explained theoretical development).

We work with the following data:

R = 2Ω, L = 5 mH, Vcc = 48 v, f = 50 Hz

By performing all calculations, these results are obtained (Table 17):

n	fn (hz)	Vn (V)	Zn (Ω)	In (A)	Pn (W)
1	50	30,55	2,54	12,01	144,28
3	150	10,18	5,12	1,99	3,96
5	250	6,11	8,10	0,75	0,57
7	350	4,36	11,18	0,39	0,15
9	450	3,39	14,28	0,24	0,06

Table 17 Coefficients of the Fourier series of Push-Pull Inverter (Square Wave Control).

Voltage THD factor:

$$THDv = \frac{\sqrt{V_{rms}{}^2 - V_1{}^2{}_{,\,rms}}}{V_{1,\,rms}} = \frac{\sqrt{(\frac{10,18}{\sqrt{2}})^2 + (\frac{6,11}{\sqrt{2}})^2 + (\frac{4,36}{\sqrt{2}})^2 + (\frac{3,39}{\sqrt{2}})^2}}{(\frac{30,55}{\sqrt{2}})} = 0,4287 = 42,87\%$$

Current THD factor:

$$THDi = \frac{\sqrt{\sum_{n=2}^{\infty} (I_{n,\,rms})^2}}{I_{1,\,rms}} \approx \frac{\sqrt{(\frac{1,99}{\sqrt{2}})^2 + (\frac{0,75}{\sqrt{2}})^2 + (\frac{0,39}{\sqrt{2}})^2 + (\frac{0,24}{\sqrt{2}})^2}}{(\frac{12,01}{\sqrt{2}})} = 0,1811 = 18,11\%$$

Here can be seen the waveform at the output of the load with this type of control (Figure 61):

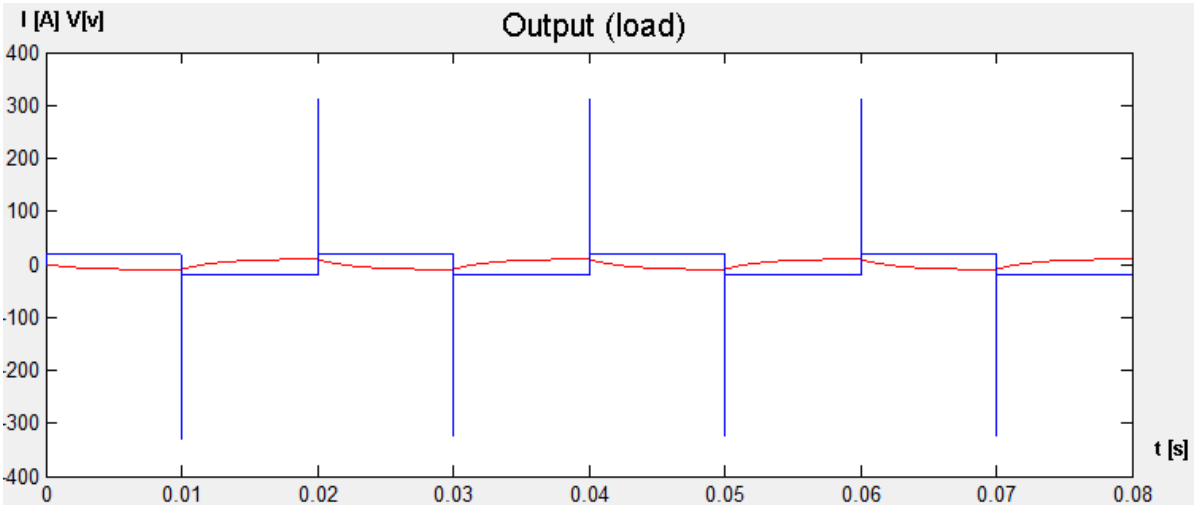


Figure 61 Voltage and current output of Push-Pull Inverter (Square Wave Control).

The blue line represents the voltage, and red is the current through the load. As it has said before, there are some simulation errors, and because of this can be seen some peaks in the output load waveform.

In Figure 62 shows the Fourier series, both the current (first graph) and the voltage (second graph) with the previous data and with Simulink (Matlab) graphs:

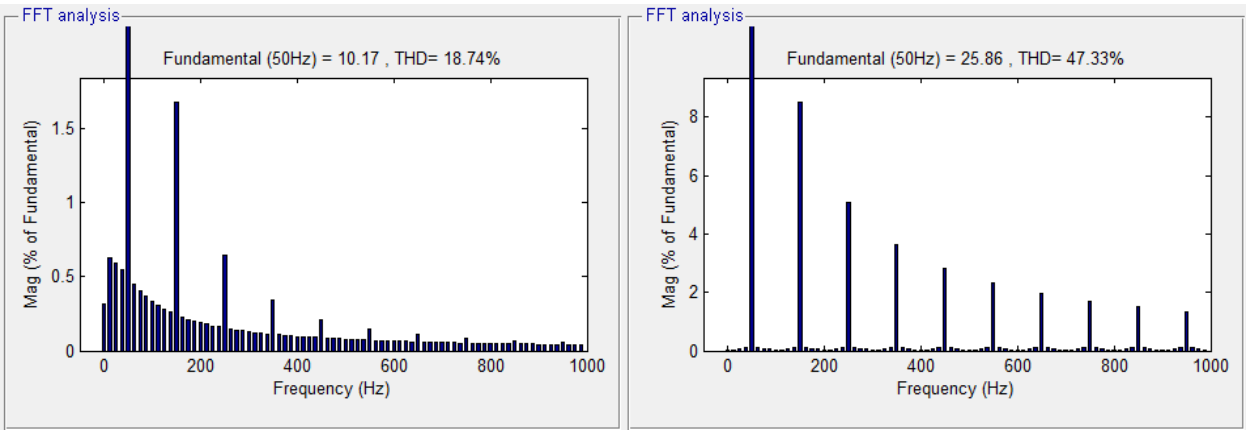


Figure 62 Frequency spectrum of the voltage and current of Push-Pull Inverter (Square Wave Control).

3.2.5.2 PUSH-PULL INVERTER (PWM CONTROL)

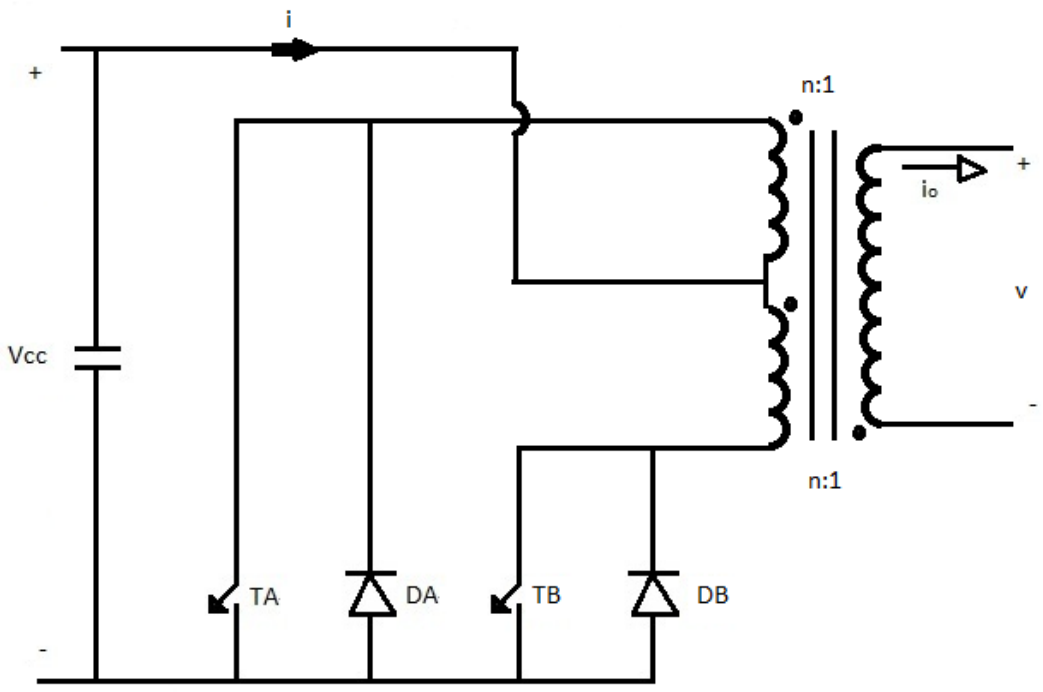


Figure 63 Push-Pull Inverter.

This inverter operates as follows:

$$V_{PEAKVOLTAGE} = m_a \cdot \frac{Vcc}{n} \quad \text{with} \quad m_a \leq 1 \tag{3.81}$$

Theoretically it's like a Half-Wave Inverter (PWM Control), but in the simulation there are voltage peaks simulation error.

To perform all calculations on this inverter, the same procedure applies in the case of Half-Wave Inverter (PWM Control) (See the previously explained theoretical development).

We work with the following data:

R = 2Ω, L = 5 mH, Vcc = 48 v, f = 50 Hz

In the following table (Table 18) can get the multiplying factor to calculate the voltage in each harmonic according to m_f and m_a. In this case, the most relevant harmonics are:

$$n = 1, \quad n = m_f \quad \text{and} \quad n = m_f \pm 2$$

	m_a=1	**0,9**	**0,8**	**0,7**	**0,6**	**0,5**	**0,4**	**0,3**	**0,2**	**0,1**
n = 1	1	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
n = m_f	0,6	0,71	0,82	0,92	1,01	1,08	1,15	1,2	1,24	1,27
n = m_f ± 2	0,32	0,27	0,22	0,17	0,13	0,09	0,06	0,03	0,02	0

Table 18 Normalized Fourier coefficients V_n / V_{DC} for PWM Control.

By performing all calculations, these results are obtained (Table 19):

n	**fn (hz)**	**Vn (V)**	**Zn (Ω)**	**In (A)**	**Pn (W)**
1	50	12,00	2,54	4,72	22,27
18	900	2,16	28,34	0,08	0,01
20	1000	25,92	31,48	0,82	0,68
22	1100	2,16	34,62	0,06	0,00

Table 19 Coefficients of the Fourier series of Push-Pull Inverter (PWM Control).

Voltage THD factor:

$$THDv = \frac{\sqrt{V_{rms}{}^2 - V_1{}^2, _{rms}}}{V_{1,\,rms}} = \frac{\sqrt{(\frac{2,16}{\sqrt{2}})^2 + (\frac{25,92}{\sqrt{2}})^2 + (\frac{2,16}{\sqrt{2}})^2}}{(\frac{12}{\sqrt{2}})} = 2,1749 = 217,49\%$$

Current THD factor:

$$THDi = \frac{\sqrt{\sum_{n=2}^{\infty} (I_{n,\,rms})^2}}{I_{1,\,rms}} \approx \frac{\sqrt{(\frac{0,08}{\sqrt{2}})^2 + (\frac{0,82}{\sqrt{2}})^2 + (\frac{0,06}{\sqrt{2}})^2}}{(\frac{4,72}{\sqrt{2}})} = 0,1757 = 17,57\%$$

Here can be seen the waveform at the output of the load with this type of control (Figure 64):

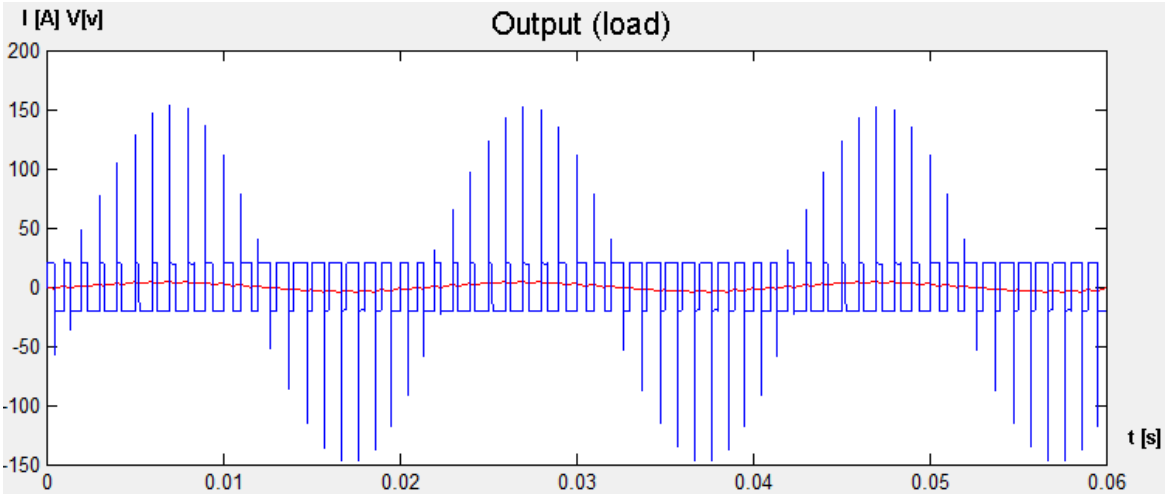


Figure 64 Voltage and current output of Push-Pull Inverter (PWM Control).

The blue line represents the voltage, and red is the current through the load. As it has said before, there are some simulation errors, and because of this can be seen some peaks in the output load waveform.

In Figure 65 shows the Fourier series, both the current (first graph) and the voltage (second graph) with the previous data and with Simulink (Matlab) graphs:

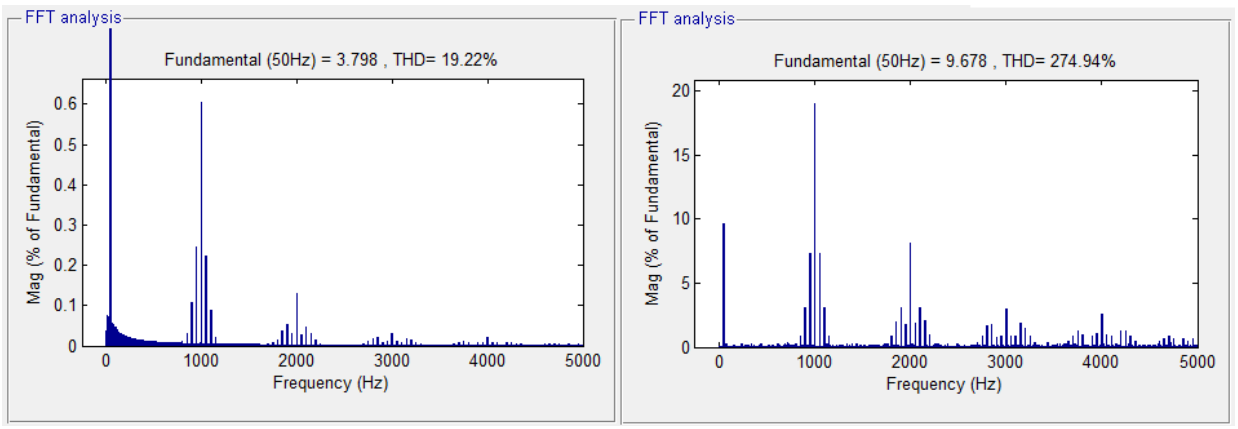


Figure 65 Frequency spectrum of the voltage and current of Push-Pull Inverter (PWM Control).

3.2.6 CASCADE FULL BRIDGE INVERTERS

This type of inverters are classified as: Multilevel Inverters.

It wbe explained later in more detail each of them, but first will comment on some advantages and disadvantages.

Advantages of cascade full bridge inverters:

- Construction can be modular, reducing installation complexity and price. The number of levels can be easily increased by adding more H-Bridges.
- Require less number of components than other multilevel inverters to achieve the same number of levels.
- It is fault tolerant. It may lose one of its stages and reduce its output to a level less and keep running.

And it has some disadvantages:

- Isolated DC sources are required. So it will be necessary to use a transformer with multiple secondary or independent transformers. This causes an increase in the cost of the inverter.

3.2.6.1 CASCADE FULL BRIDGE SINGLE-PHASE INVERTER (2 LEVELS)

The following figure (Figure 66) depicts the scheme of this type of inverter:

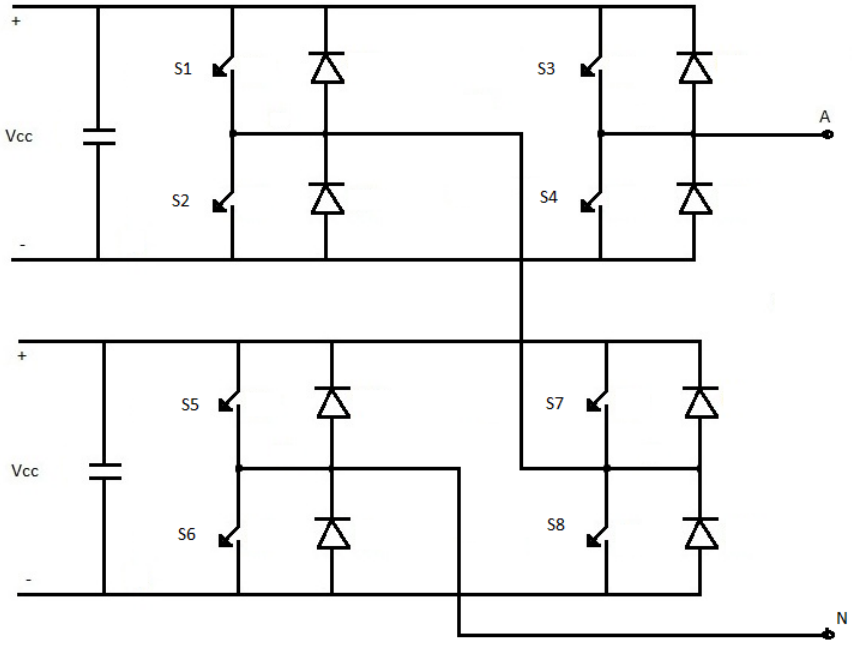


Figure 66 Cascade Full Brigde Single-Phase Inverter (2 Levels).

This type of inveter is the cascade connection of two H-Bridges, which is done by connecting in series two-stage H-bridge

This H-bridge inverter each can generate three different output voltages: + V_{CC}, 0, -V_{CC}. The resulting phase voltage is sintered by the sum of the voltages generated by each bridge. Therefore, the output voltage V_{an} can take 5 different values (Table 20):

CLOSED SWITCHES	VOLTAGE V_{AO}
S2, S3, S6, S8	{Vcc}
S2, S3, S6, S7	{2 Vcc}
Todos abiertos	0
S1, S4, S6, S8	{-Vcc}
S1, S4, S5, S8	{-2 Vcc}

Table 20 Control of transitors of Cascade Full Brigde Single-Phase Inverter (2 Levels).

In the figure below (Figure 67) can see how it works this inverter control, in which moment each transistor is working and what is not.

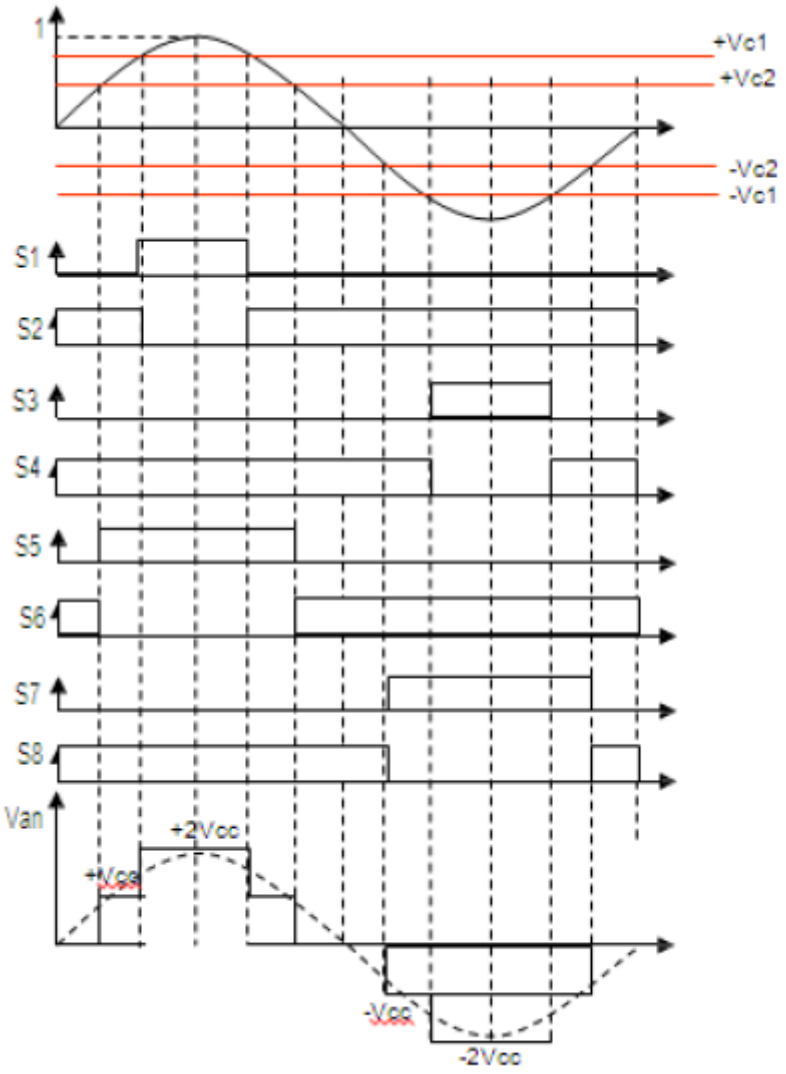


Figure 67 Waveform of transitors of Cascade Full Brigde Single-Phase Inverter (2 Levels).

Here can be seen the waveform at the output of the load with this type of control (Figure 68):

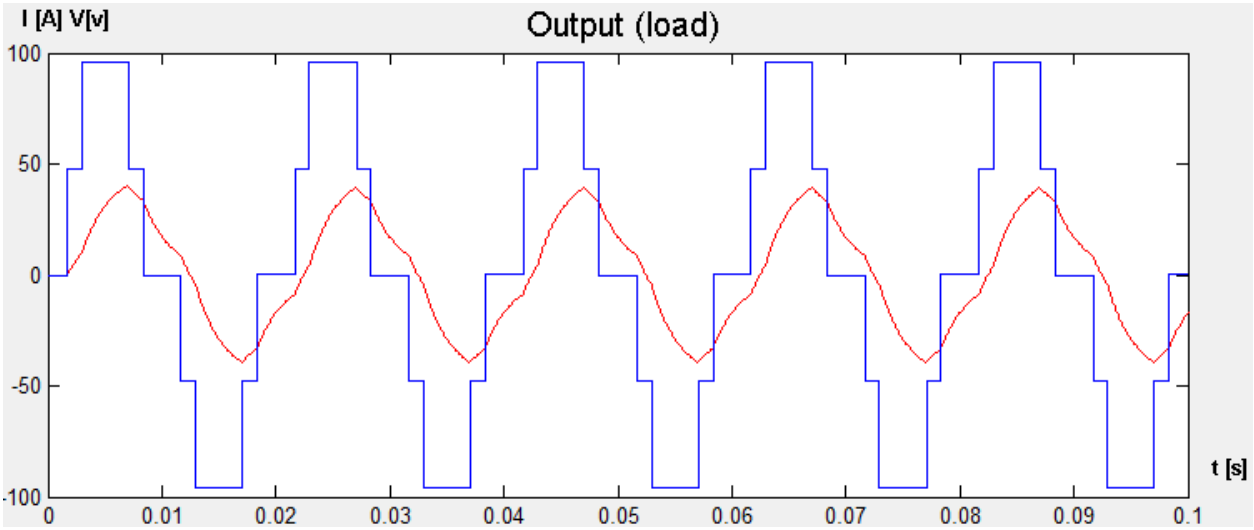


Figure 68 Voltage and current output of Cascade Full Brigde Single-Phase Inverter (2 Levels).

The blue line represents the voltage, and red is the current through the load.

In Figure 69 shows the Fourier series, both the current (first graph) and the voltage (second graph) with the previous data and with Simulink (Matlab) graphs:

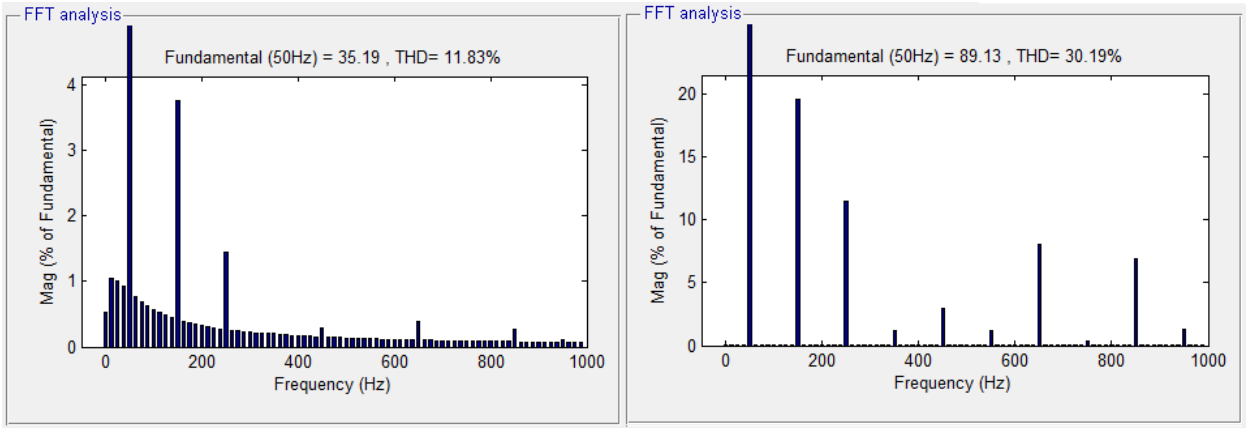


Figure 69 Frequency spectrum of the voltage and current of Cascade Full Brigde Single-Phase Inverter (2 Levels).

3.2.6.2 CASCADE FULL BRIDGE SINGLE-PHASE INVERTER (4 LEVELS)

The following figure (Figure 70) shows the scheme of this type of inverter:

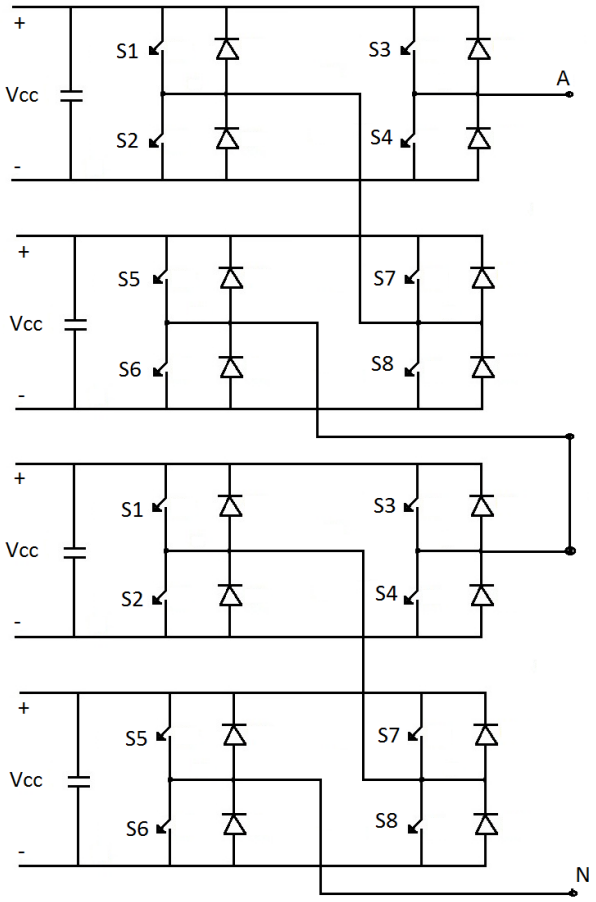


Figure 70 Cascade Full Bridge Single-Phase Inverter (4 Levels).

This inverter operates as follows:

Theoretically it's like a Cascade Full Bridge Single Phase Inverter (2 Levels), but there are 2 H-Bridge invertir more, but the control of them are the same that the two first (Table 21 and see the previously explained theoretical development of that inverter):

CLOSED SWITCHES	VOLTAGE V_{AO}
S2, S3, S6, S8	{Vcc}
S2, S3, S6, S7	{2 Vcc}
Todos abiertos	0
S1, S4, S6, S8	{-Vcc}
S1, S4, S5, S8	{-2 Vcc}

Table 21 Control of transitors of Cascade Full Bridge Single-Phase Inverter (4 Levels).

Here can be seen the waveform at the output of the load with this type of control (Figure 71):

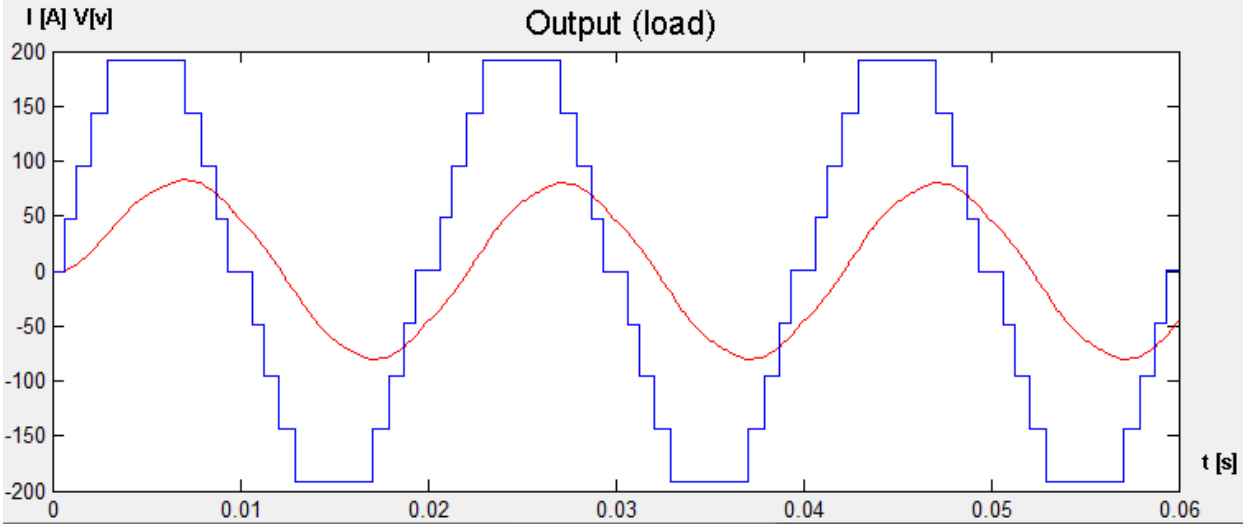


Figure 71 Cascade Full Bridge Single-Phase Inverter (4 Levels).

The blue line represents the voltage, and red is the current through the load.

In Figure 72 shows the Fourier series, both the current (first graph) and the voltage (second graph) with the previous data and with Simulink (Matlab) graphs:

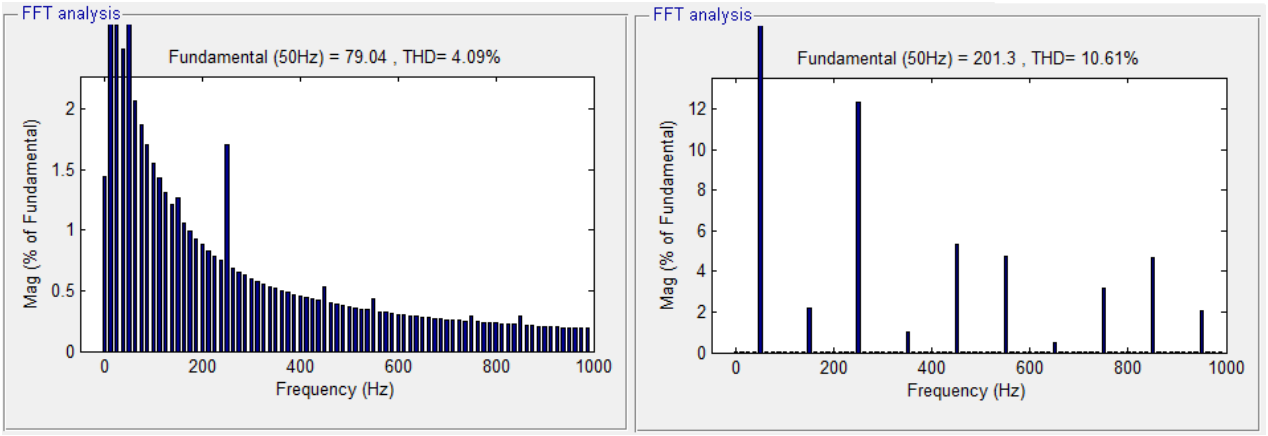


Figure 72 Frequency spectrum of the voltage and current of Cascade Full Brigde Single-Phase Inverter (4 Levels).

3.2.6.3 CASCADE FULL BRIDGE THREE-PHASE INVERTER (3 LEVELS)

3.2.6.3.1 STAR LOAD

The following figure (Figure 73) shows the scheme of this type of inverter:

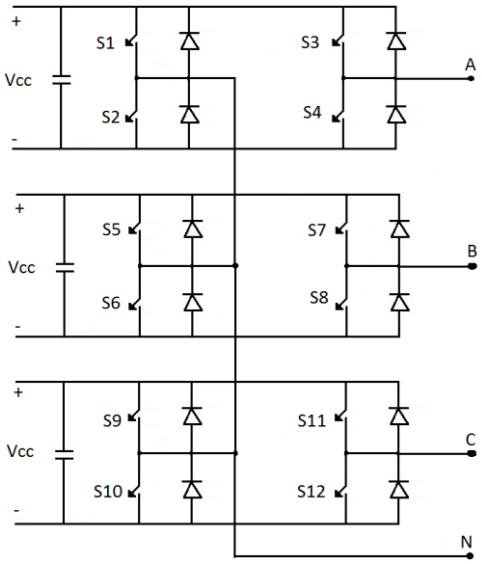


Figure 73 Cascade Full Bridge Three-Phase Inverter (3 Levels).

This inverter operates as follows:

In this case, the control of each of the three H-bridge Inverters is the same as in the case of Full Wave H-Bridge Inverter (See the previously explained theoretical development of that inverter) but each 120º out of phase from the previous inverter.

Here can be seen the waveform at the output of the load with this type of control (Figure 74):

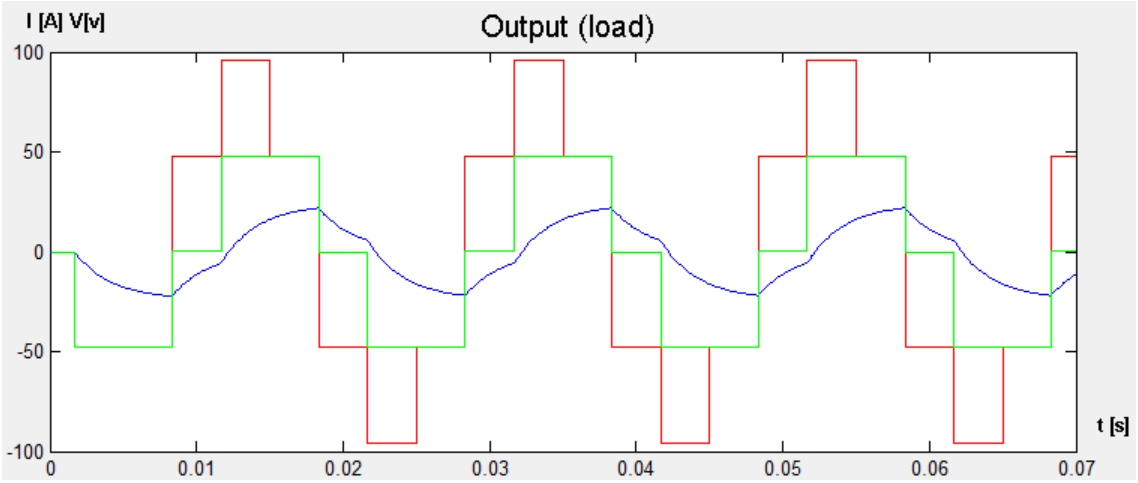


Figure 74 Voltage and current output of Cascade Full Bridge Three-Phase Inverter (3 Levels).

The red line represents the line voltage, the yellow line corresponds to the phase voltage, and blue is the current through the load.

In Figure 75 shows the Fourier series. Line Voltage (first graph), phase Voltage (second graph) and Current (I_{line} = I_{phase}) (third graph) Fourier series with the previous data and with Simulink (Matlab) graphs.

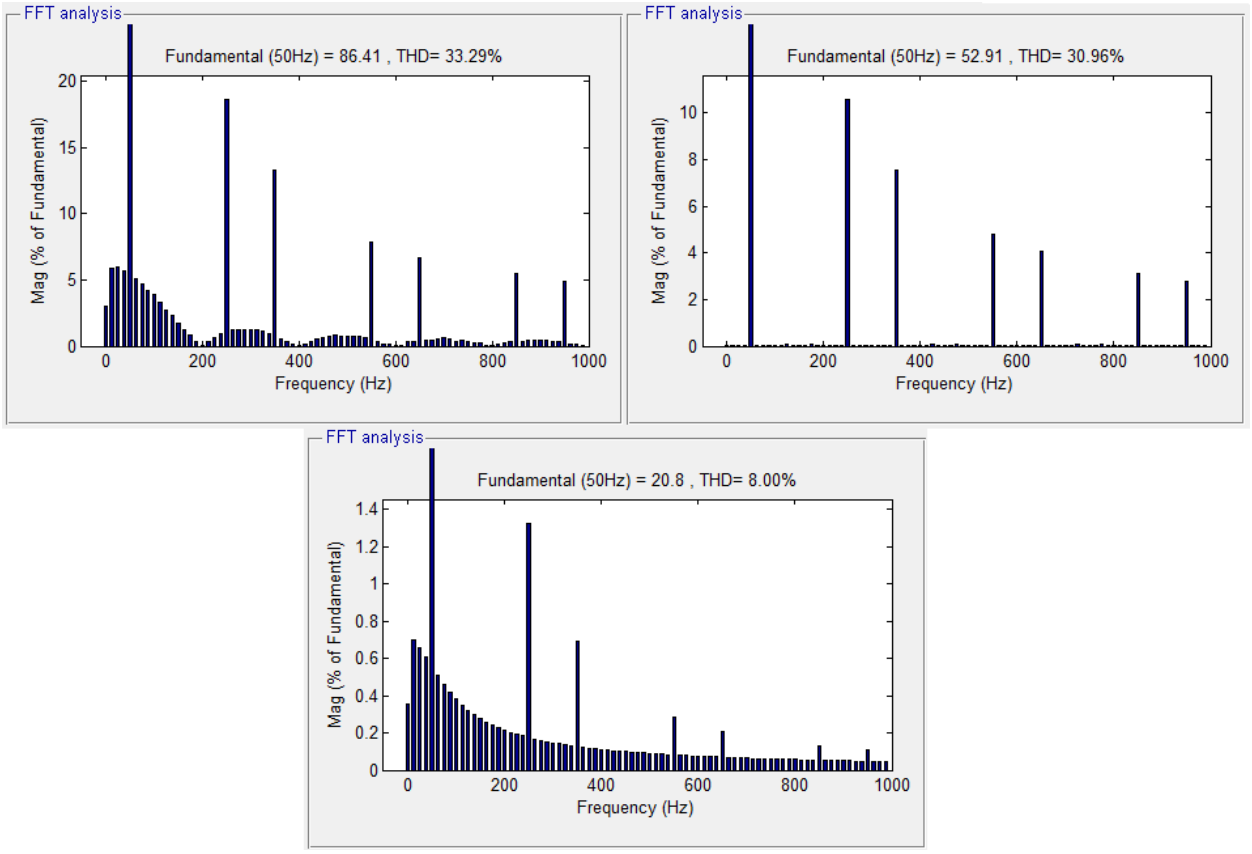


Figure 75 Frequency spectrum of the voltage and current of Cascade Full Brigde Three-Phase Inverter (3 Levels) with the load connected in Star.

3.2.7 NPC INVERTERS

These inverters are also a kind of Multilevel Inverters. The advantages of this type of inverter are:

- The blocking voltage of the switches is the input capacitor voltage $V_s/(n-1)$ in case of n levels.
- The number of capacitors are small in comparison to other types of multilevel inverters.
- Transformers isn't necessary.
- The efficiency is high.

- As the number of levels increases harmonic content is reduced.

On the contrary, there are also some disadvantages on this inverter:

- Fast recovery diodes are necessary.
- It's necessary that the voltages on the capacitors remain in balance at any point of work, so that the converter control is complicated. This balance is difficult as increases the number of levels, it may even be impossible in some operating conditions.

3.2.7.1 NPC INVERTER

The following figure (Figure 76) shows the scheme of this type of inverter:

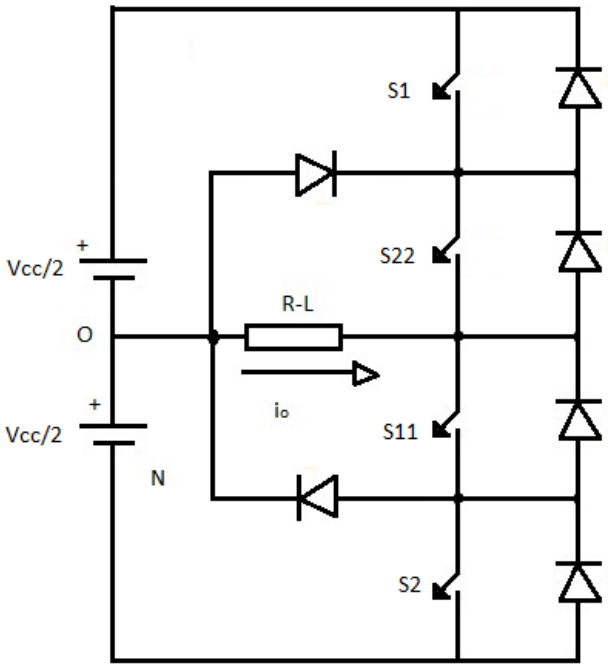


Figure 76 NPC Inverter.

This inverter operates as follows (Figure 77 and Table 22):

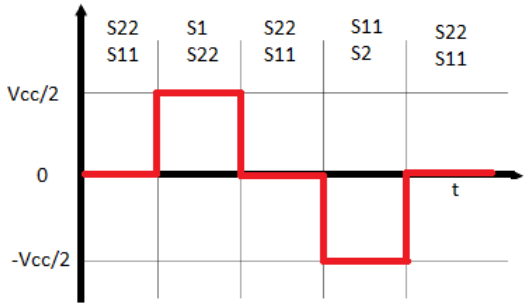


Figure 77 Control of transistors of NPC Inverter.

CLOSED SWITCHES	VOLTAGE V_{AO}
S1, S22	{Vcc/2}
S22, S11	0
S11, S2	{-Vcc/2}

Table 22 Control of transistors of NPC Inverter.

In the following figures (Figure 78 and 79) you can see how it works perfectly this type of invertir:

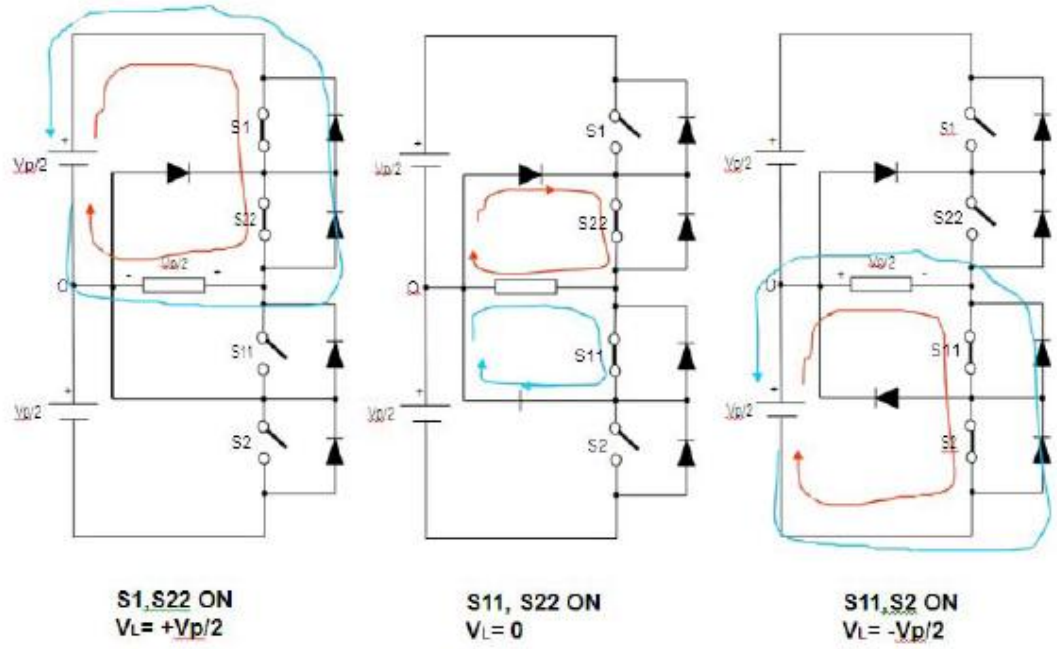


Figure 78 Current flow in NPC Inverter.

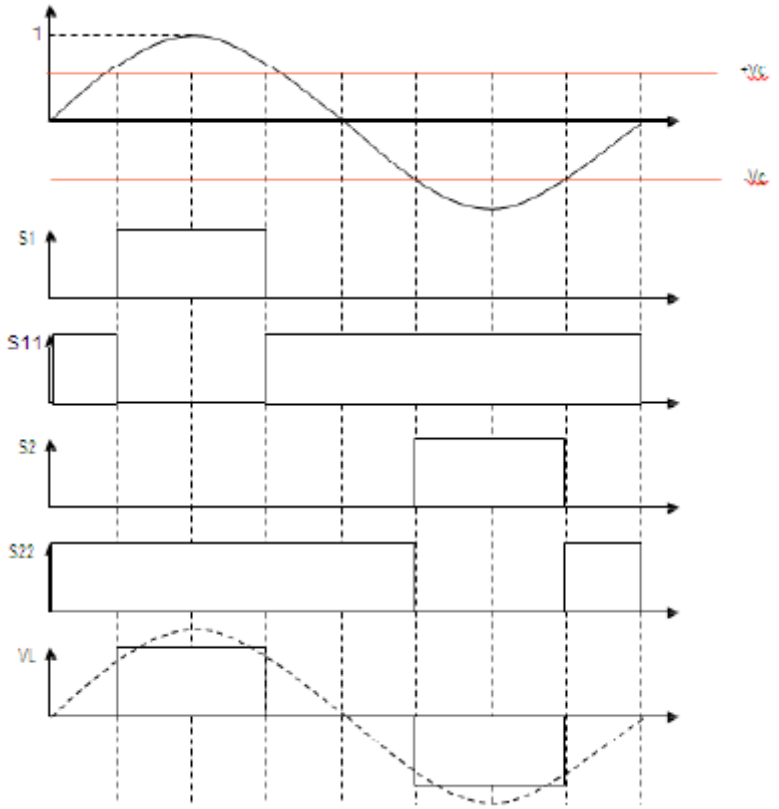


Figure 79 Waveform of transitors of NPC Inverter.

This NPC Inverter, shown in Figure 76, has many advantages over conventional 2-level inverters:

- The voltage should hold the switches is only half of the DC bus voltage, so it can withstand higher power levels.
- The harmonic distortion is less than the others.
- This inverter can be generalized, so that the principles used for three levels can be extrapolated to inverters with a higher number of voltage levels.

However, practical experience reveals certain technical difficulties that complicate its application in high-power converters. These difficulties are the following:

- Inverter such diodes are required to high speed and should be able to withstand high currents, and are subjected to a high reverse recovery stress.
- For inverters with more than three levels, the diodes withstand increasing tensions, so that must be connecting in series. This complicates the design and increases the reliability requirements and costs.
- The control methodology is complicated, since the NPC topology must be regulated the balance of capacitor voltages of the DC bus. For topologies over three levels, the regulation of this balance is still an unresolved issue.

Here can be seen (Figure 80) the waveform at the output of the load with this type of control:

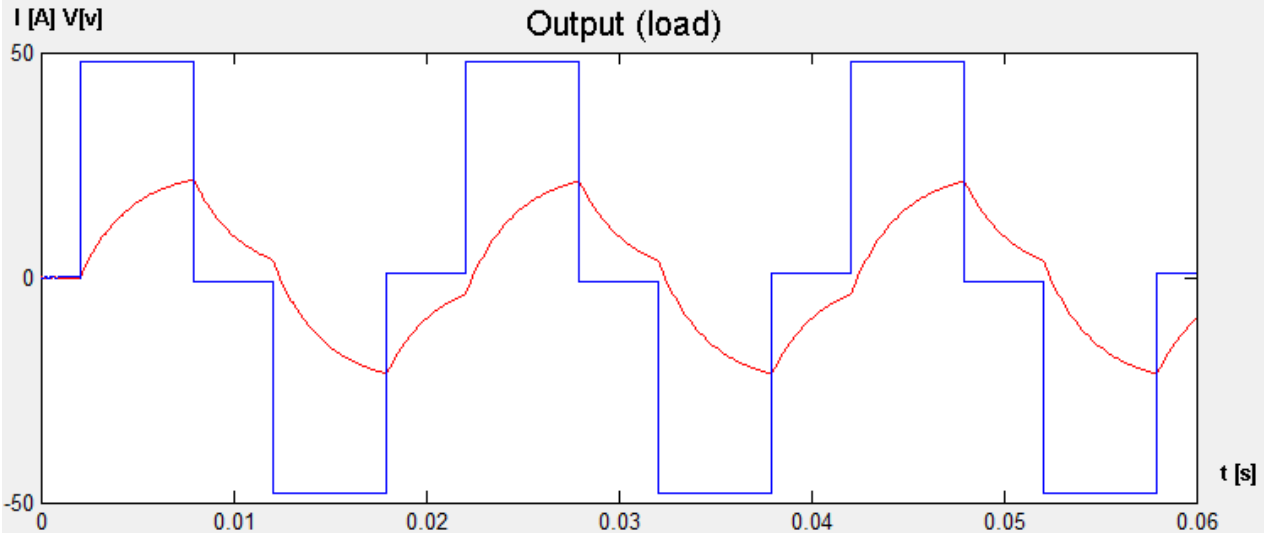


Figure 80 Voltage and current output of NPC Inverter.

The blue line represents the voltage, and red is the current through the load.

In Figure 81 shows the Fourier series, both the current (first graph) and the voltage (second graph) with the previous data and with Simulink (Matlab) graphs:

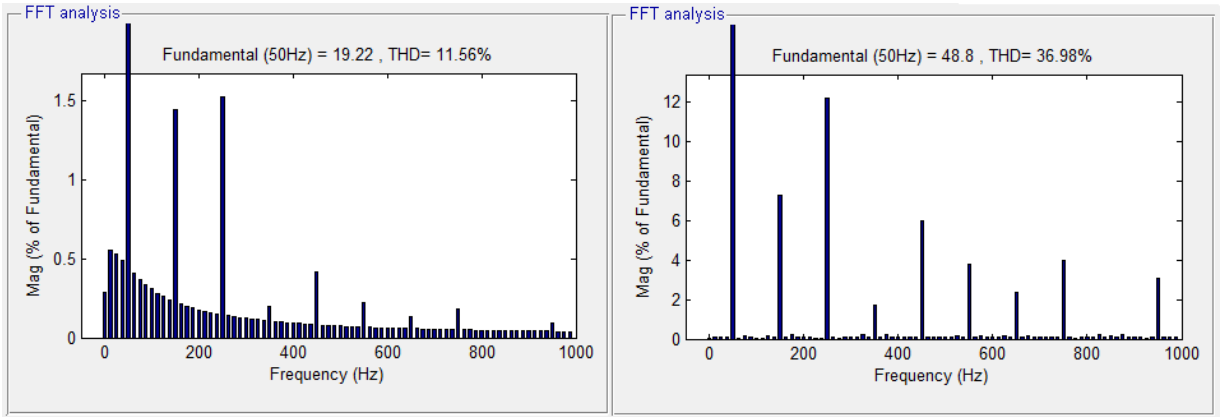


Figure 81 Frequency spectrum of the voltage and current of NPC Inverter.

3.2.7.2 NPC INVERTER (5 LEVELS)

The following figure (Figure 82) shows the scheme of this type of inverter:

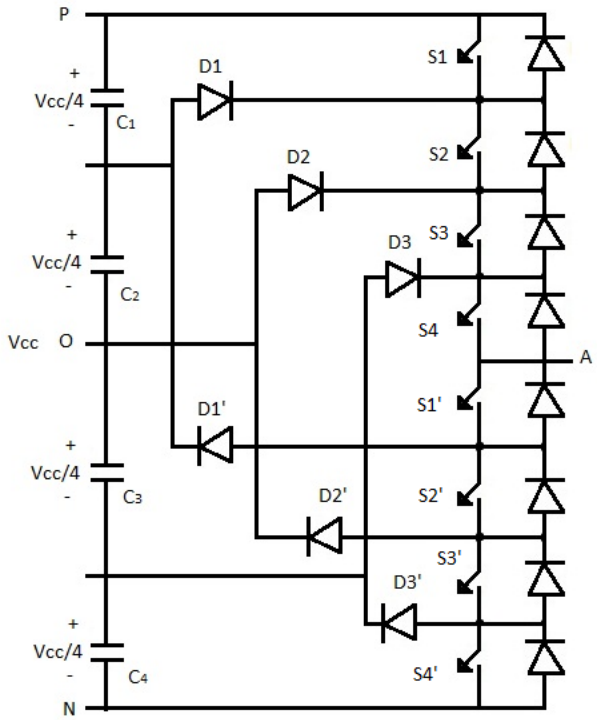


Figure 82 NPC Inverter (5 Levels).

This inverter operates as a continuation in the number of levels from previous (Table 23 and Figure 83):

CLOSED SWITCHES	VOLTAGE V_{AO}
S1, S2, S3, S4	{Vcc/2}
S2, S3, S4, S1'	{Vcc/4}
S3, S4, S1', S2'	0
S4, S1', S2', S3'	{-Vcc/4}
S1', S2', S3', S4'	{-Vcc/2}

Table 23 Control of transistors of NPC Inverter (5 Levels).

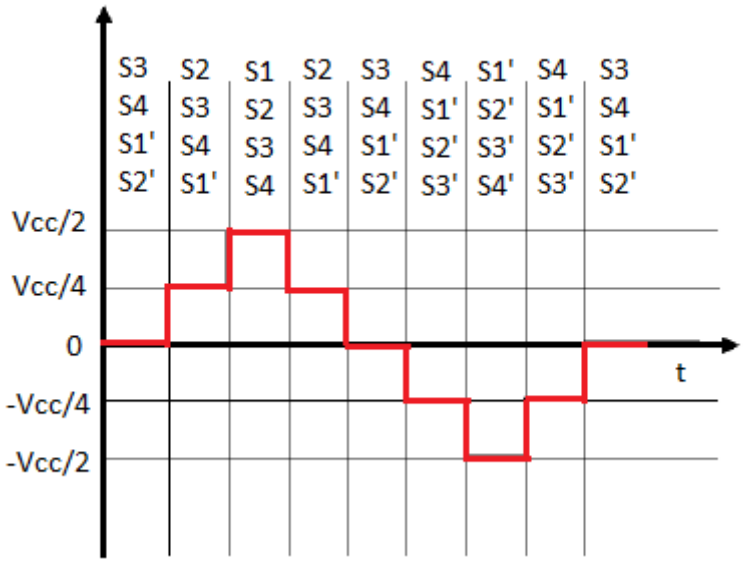


Figure 83 Waveform of transitors of NPC Inverter (5 Levels).

Here can be seen the waveform at the output of the load with this type of control (Figure 84):

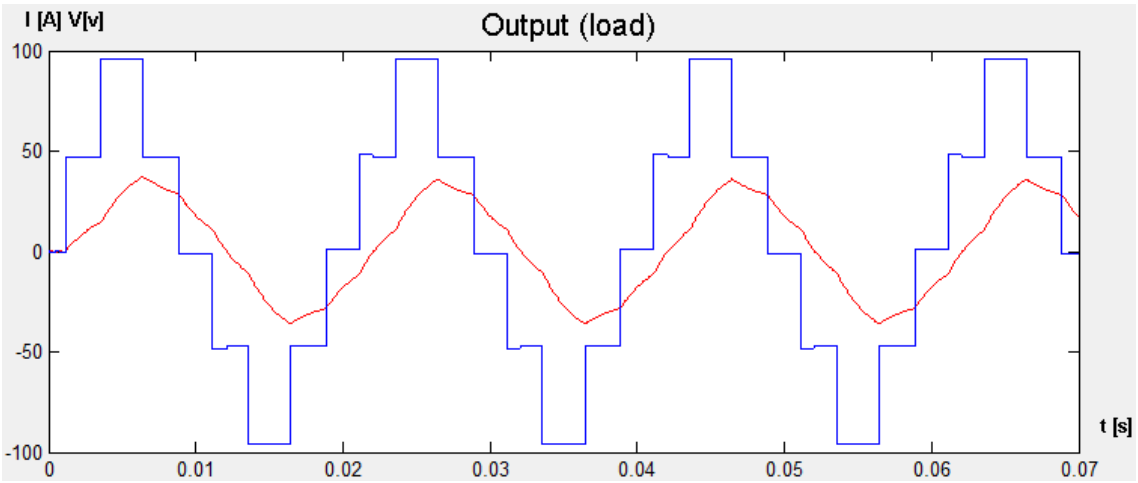


Figure 84 Voltage and current output of NPC Inverter (5 Levels).

The blue line represents the voltage, and red is the current through the load.

In Figure 85 shows the Fourier series, both the current (first graph) and the voltage (second graph) with the previous data and with Simulink (Matlab) graphs:

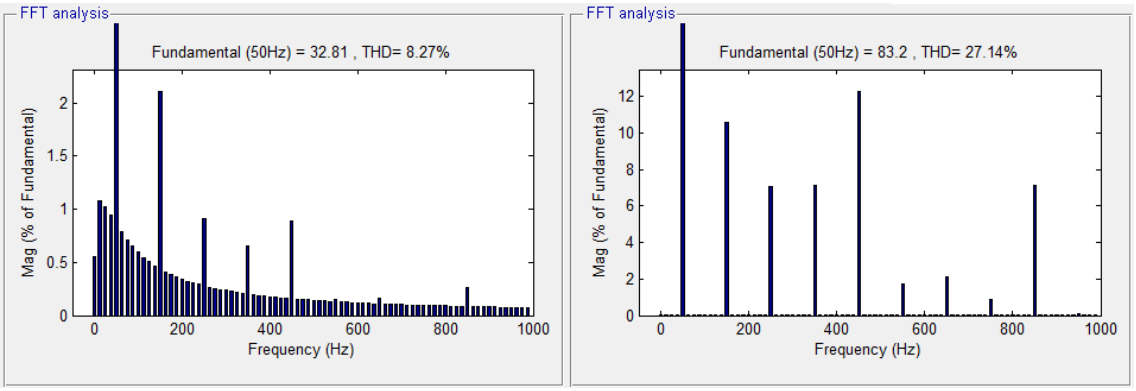


Figure 85 Frequency spectrum of the voltage and current of NPC Inverter (5 Levels).

3.2.7.3 NPC THREE-PHASE INVERTER

3.2.7.3.1 STAR LOAD

The following figure (Figure 86) shows the scheme of this type of inverter:

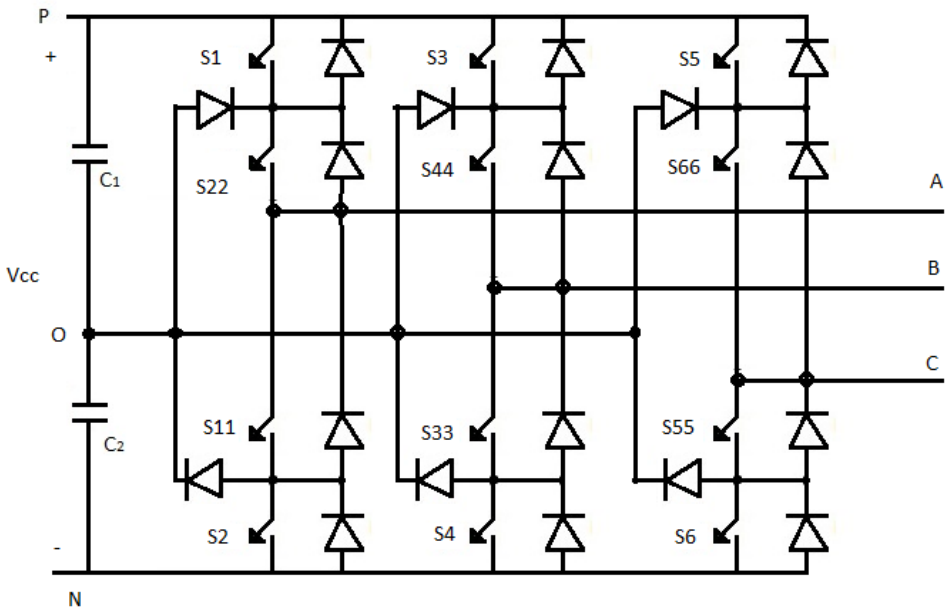


Figure 86 NPC Three-Phase Inverter.

This inverter operates as follows (Table 24). Each branch of circuit operates separately according to the control indicated in the following table, but each branch is 120º out of phase from the previous:

CLOSED SWITCHES	VOLTAGE V_{AO}
S1, S22	{Vcc/2}
S22, S11	0
S11, S2	{-Vcc/2}

Table 24 Control of transistors of NPC Three-Phase Inverter.

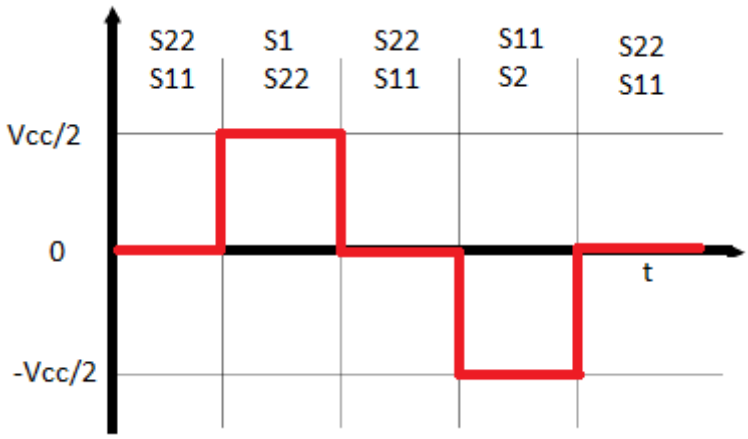


Figure 87 Waveform of transitors of NPC Three-Phase Inverter.

The waveform view up (Figure 87) here is corresponding to a single branch. The resultant whole circuit is shown below.

Here can be seen the waveform at the output of the load with this type of control (Figure 88):

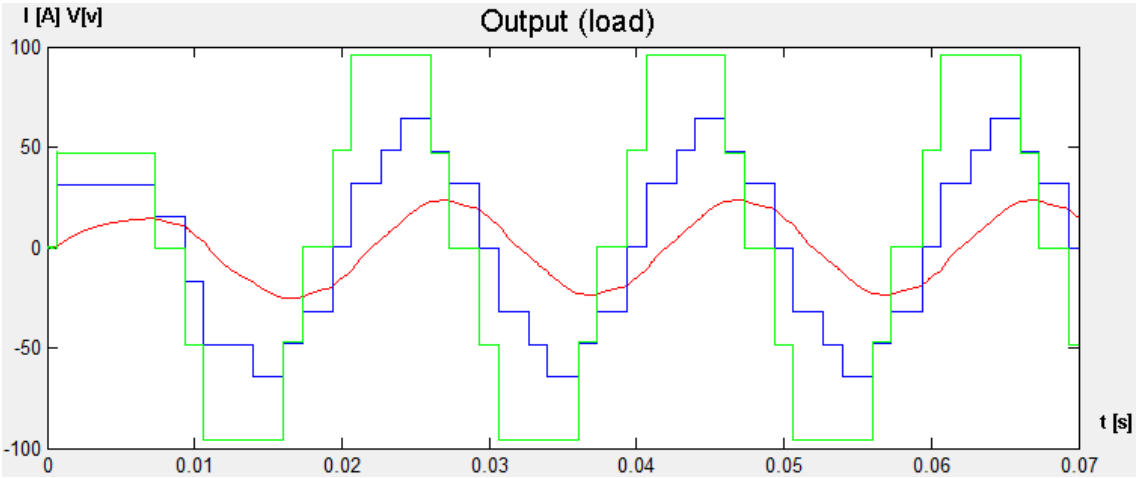


Figure 88 Voltage and current output of NPC Three-Phase Inverter.

The yellow line represents the line voltage, the blue line corresponds to the phase voltage, and red is the current through the load.

In Figure 89 shows the Fourier series. Line Voltage (first graph), phase Voltage (second graph) and Current (I_{line} = I_{phase}) (third graph) Fourier series with the previous data and with Simulink (Matlab) graphs.

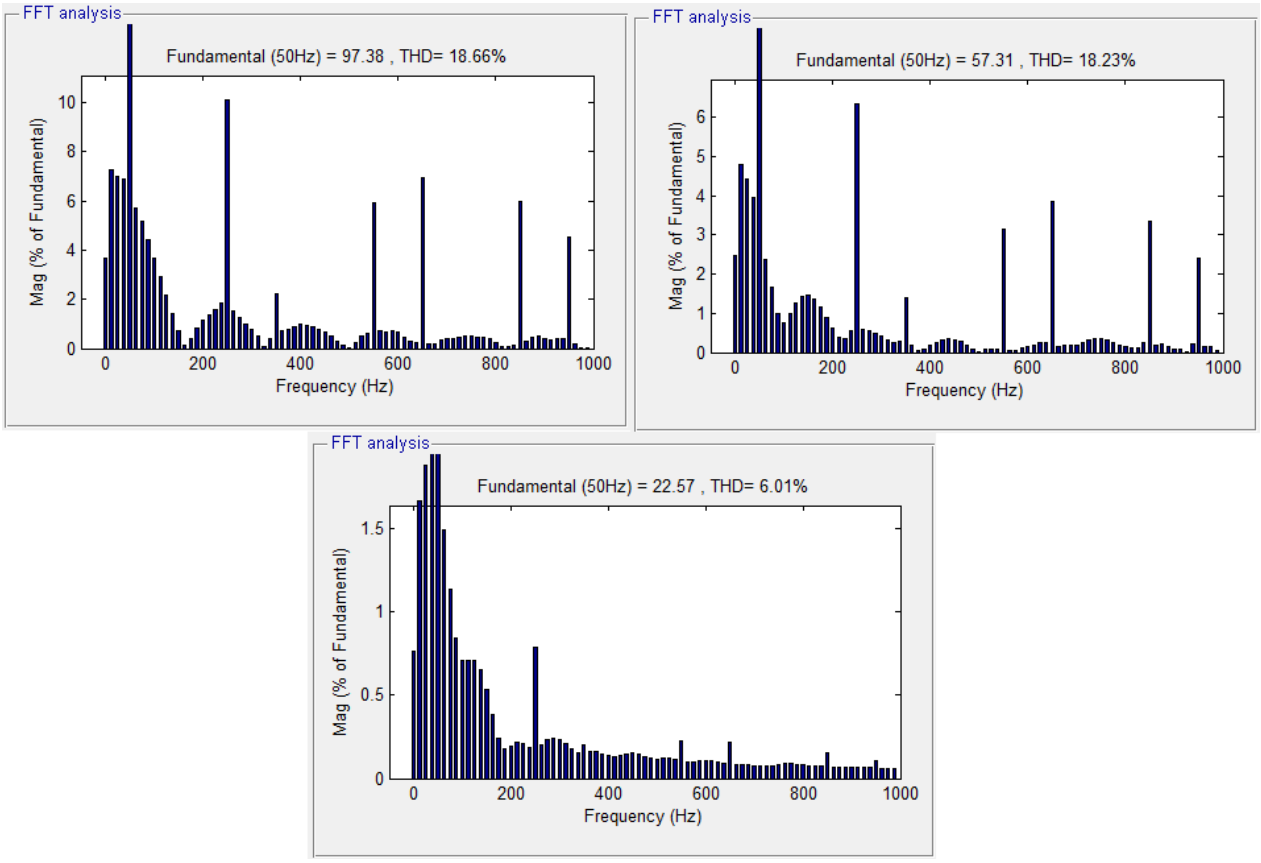


Figure 89 Frequency spectrum of the voltage and current of NPC Three-Phase Inverter with the load connected in Star.

3.2.8 FC INVERTERS

These inverters are also a kind of Multilevel inverters.The advantages of this type of inverter are:

- Due to the presence of the floating capacitor, the blocking voltage of the switches is $V_s/(n-1)$, as in the NPC inverter.
- When the number of levels is very high, the harmonic content will be very low, facilitating the use of filters with very small output.
- Can be added an additional branch switches to be able to generate a three-phase voltage, similar to the case NPC inverters.

On the contrary, there are also some disadvantages on this inverter:

- A large number of capacitors are used. The current flowing through all the floating capacitors is the same, therefore the value of the capacitors should be the same to maintain the same voltage ripple.

- The floating capacitors must withstand the load current, therefore, should be selected properly in order to not create excessive losses and not condition the maximum current of the inverter.

- Preloaded process of floating capacitors must exist.

3.2.8.1 FC INVERTER (5 LEVELS)

The following figure (Figure 90) shows the scheme of this type of inverter:

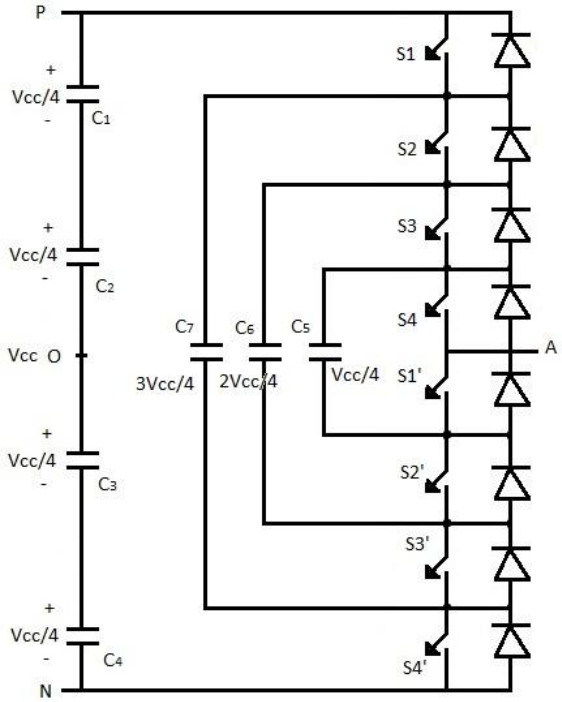


Figure 90 FC Inverter (5 Levels).

This inverter operates as follows (Table 25):

CLOSED SWITCHES	VOLTAGE V_{AO}
S1, S2, S3, S4	{Vcc/2}
S1, S2, S3, S1' o S2, S3, S4, S4' o S1, S3, S4, S3`	{Vcc/4}
S1, S2, S1', S2' o S3, S4, S3', S4' o S1, S3, S1', S3' o S1, S4, S2', S3' o S2, S4, S2', S4' o S2,	0
S4, S1', S2', S3'	{-Vcc/4}
S1', S2', S3', S4'	{-Vcc/2}

Table 25 Control of transistors of FC Inverter (5 Levels).

Here can be seen the waveform at the output of the load with this type of control (Figure 91):

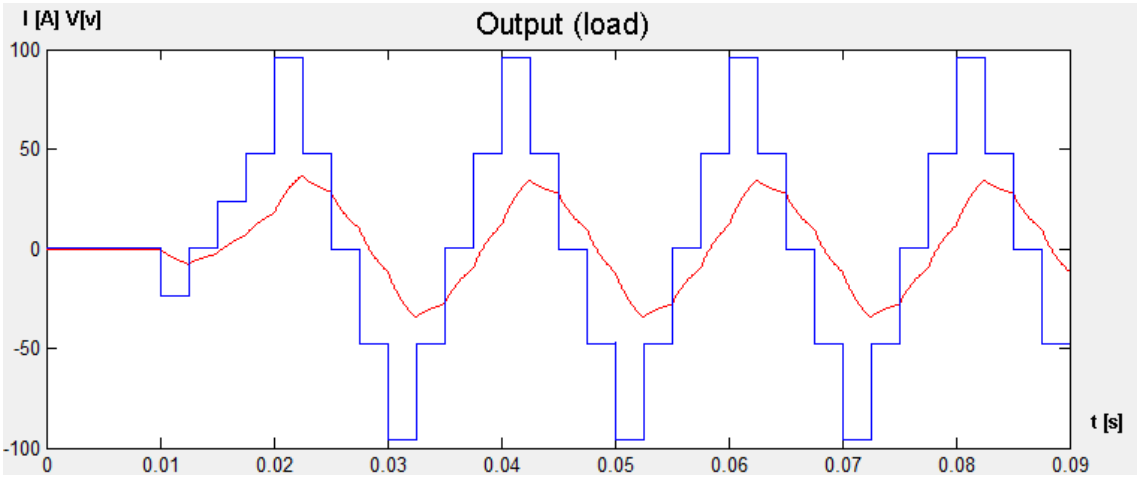


Figure 91 Voltage and current output of FC Inverter (5 Levels).

The blue line represents the voltage, and red is the current through the load.

In Figure 92 shows the Fourier series, both the current (first graph) and the voltage (second graph) with the previous data and with Simulink (Matlab) graphs:

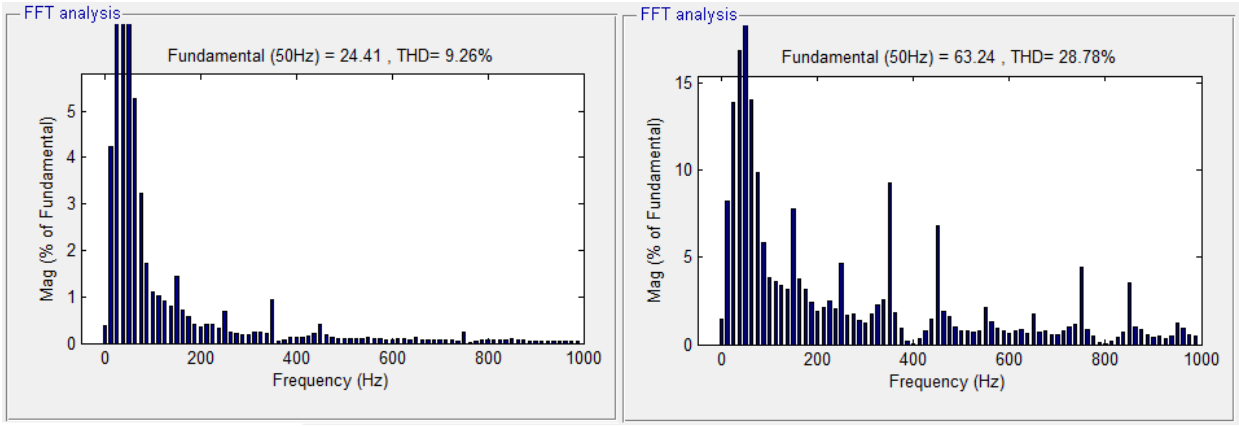


Figure 92 Frequency spectrum of the voltage and current of FC Inverter (5 Levels).

3.2.8.2 FC THREE-PHASE INVERTER (3 LEVELS)

3.2.8.2.1 STAR LOAD

The following figure (Figure 93) shows the scheme of this type of inverter:

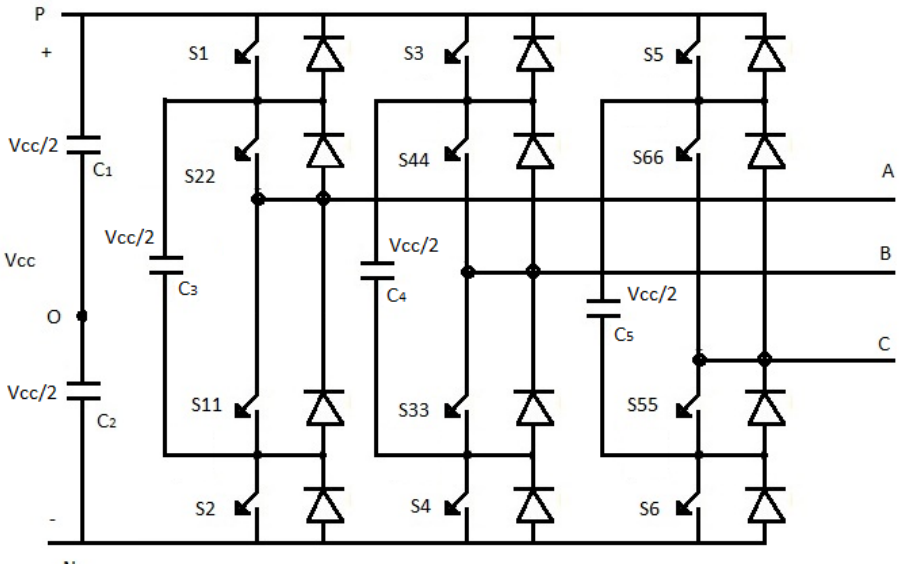


Figure 93 FC Three-Phase Inverter (3 Levels).

This inverter operates as follows (Table 26):

CLOSED SWITCHES	VOLTAGE V_{AO}
S1, S22	{Vcc/2}
S1, S11 o S22, S2	0
S11, S2	{-Vcc/2}

Table 26 Control of transistors of FC Three-Phase Inverter (3 Levels).

Here can be seen the waveform at the output of the load with this type of control (Figure 94):

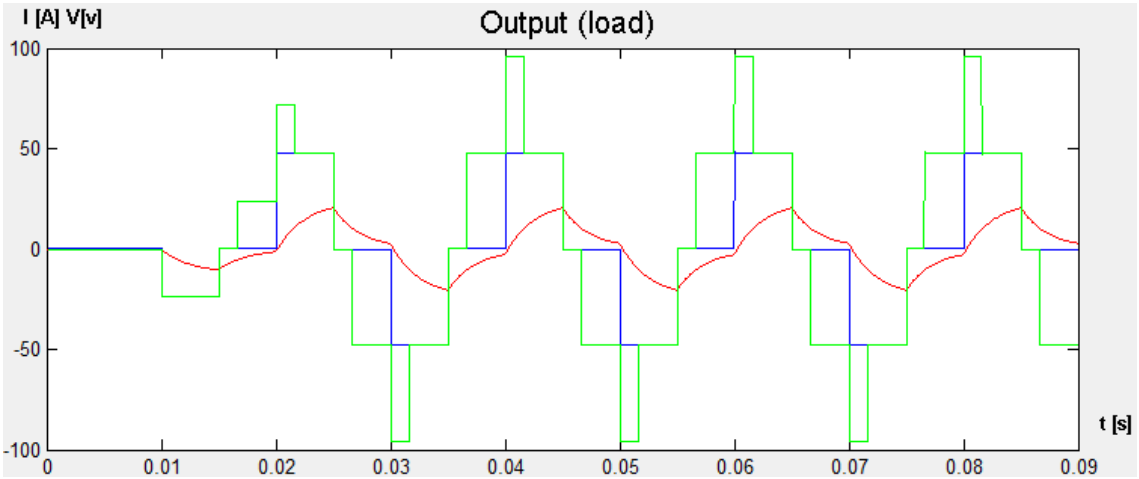


Figure 94 Voltage and current output of FC Three-Phase Inverter (3 Levels).

The yellow line represents the line voltage, the blue line corresponds to the phase voltage, and red is the current through the load.

In Figure 95 shows the Fourier series. Line Voltage (first graph), phase Voltage (second graph) and Current (I_{line} = I_{phase}) (third graph) Fourier series with the previous data and with Simulink (Matlab) graphs.

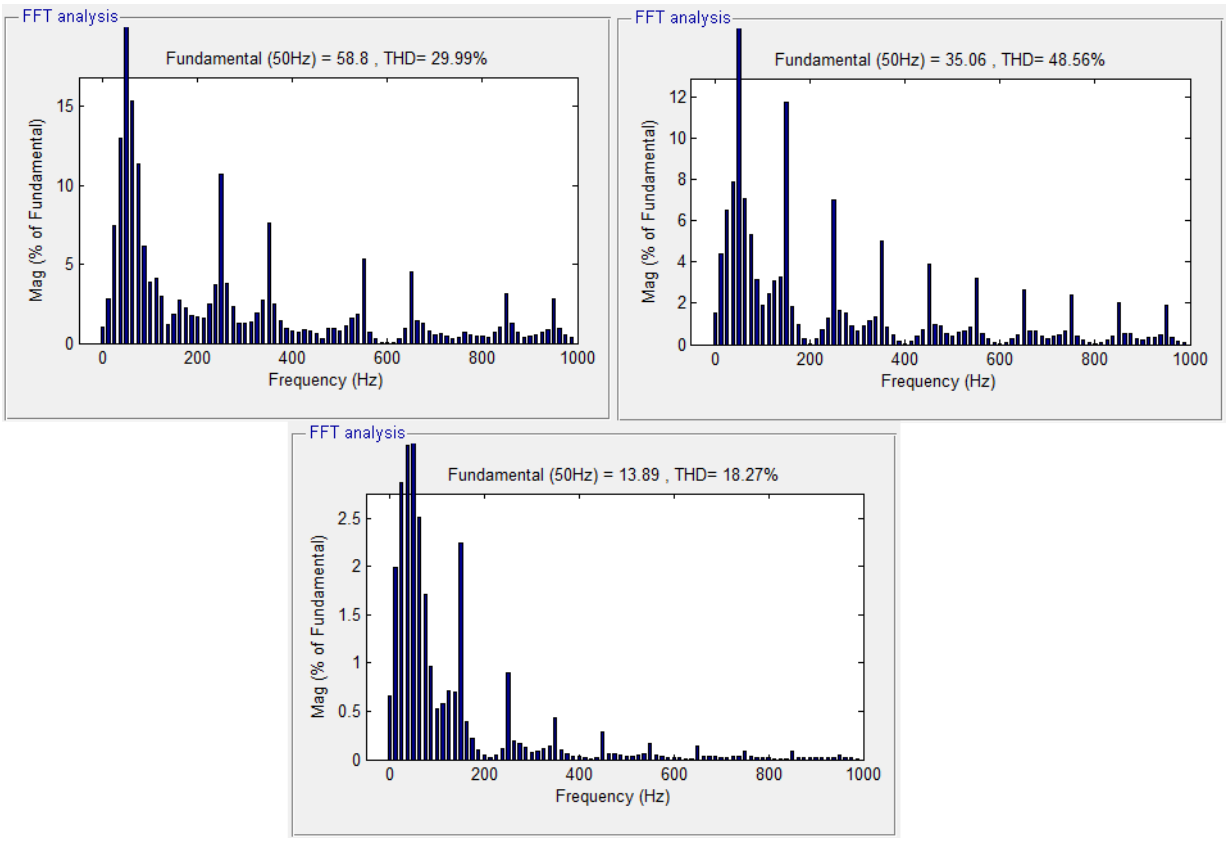


Figure 95 Frequency spectrum of the voltage and current of FC Three-Phase Inverter (3 Levels) with the load connected in Star.

4 APPLICATION FOR TEACHING STUDENTS

As mentioned previously, the purpose of this project is the creation of an educational application that supports engineering students to understand how to work the main types of inverter circuits.

The first step is the simulation of circuits previously analyzed using MATLAB SIMULINK program.

To do this first is briefly explained what MATLAB is what SIMULINK is in ANNEX I. This annex explains what MATLAB, which are all functions and a brief manual to introduce as the program is used. It also explains how to access SIMULINK and which are the first steps to know how to handle it without any problem.

After introduction to MATLAB and SIMULINK, has performed the simulations of the 23 inverter circuits which have been previously analyzed by SIMULINK.

The 23 simulations are available to students in order to open them, run them, study them and modify what the student wants. In ANNEX I are the steps to execute in case of not knowing how to do it, and all the most important schemes and waveforms for each circuit.

But when doing this project, it has been thought of how to make a graphical user interface where everything much easier to use for students.

MATLAB provides the ability to create a graphical interface created by programming code. This option is called MATLAB GUIDE, which is explained what it is and how it works in ANNEX III.

Therefore, it has created a graphical interface which is redirected into the 23 Simulink simulations. It's a very intuitive and easy to use interface where students can choose to study the inverter circuit they want. After selecting the circuit and loaded all the variables, the student can modify as he wants the control variables, input variables and the variables of the output load. After running the program, the student can see, in the interface, the graph of the waveform at the output (in the load), the graph of the control technique used in the circuit, and the numerical values most representative of the circuit. There is also an option where you can see the graph of the voltage and current harmonics in the load. There are also some theoretical support buttons where the student can support to fully understand the workings of all parts of the inverter circuit.

All this is explained in ANNEX III. In ANNEX IV is the programming code used to program this interface.

Finally, to complete the study of inverter circuits and facilitate the study to the students giving them more opportunities, it has made the simulation of inverter circuits in another circuit analysis program, called PSpice.

Therefore, students can access each of these PSpice simulations to study and analyze the working of the inverter circuits.

In ANNEX II can find a brief description of what is PSpice, and which are the first steps to run it. In ANNEX II are also all the schemes of simulations and the main waveforms of the circuits.

5 CONCLUSION

After analyzing and simulating all inverter circuits, and after comparing all values numeric (harmonic distortion, RMS values, output values, etc..) Can be observed between the values obtained from a theoretically way and the values obtained through simulation are slightly different.

This is because the margins of error that occur in the simulations, and also in the theoretical calculations are the ideal values, a bit far from reality.

With all these questions, students can do a study on calculation errors.

The objective of this project is not only for students, but can also be useful to the teachers when performing laboratory practices throughout the university career to propose to the students..

As it has mentioned above, this is a didactic application for engineering students who start studying the inverter circuits, but this is just the starting point. Can be proposed to other projects like this, which cover all kinds of converters, not only inverters circuits.

6 REFERENCES AND BIBLIOGRAPHY

The graphs and figures have been obtained from:

[1] JUAN LUIS VILLA, “Control y accionamiento de maquinas electricas” Escuela de ingeniería y Arquitectura, Universidad de Zaragoza.

[2] J. M. BURDIO, “Electrónica de Potencia” Escuela de ingeniería y Arquitectura, Universidad de Zaragoza.

[3] Web site: www.wikipedia.com

ANNEX II

DESCRIPTION OF THE COMPUTER PROGRAM *MATLAB* AND SIMULATION OF INVERTER CIRCUITS WITH *SIMULINK – MATLAB*

Author

Diego Salas Forns

Director

D. Francisco José Pérez Cebolla

INDEX

INDEX OF TABLES

INDEX OF FIGURES

1 DESCRIPTION OF THE COMPUTER PROGRAM "MATLAB"

1.1 WHAT IS "MATLAB"?

The name MATLAB is an acronym: MATrix LABoratory. Nowadays MATLAB is a very powerful program, with a friendly environment, which includes tools scientific and technical computing and graphical display as well as a programming language of high level [7], [9].

MATLAB is a mathematical software tool that offers Integrated Development Environment (IDE). It has an own programming language: Language M.

It is available for the following computing platforms: Unix, Windows, Mac OS X and GNU/Linux.

With this program we can manipulate matrix, representing data and functions, implement algorithms, creating graphical user interfaces (GUI) and communicate with programs in other languages and with other hardware devices.

MATLAB provides two additional tools: Simulink (multidomain simulation platform) and GUIDE (editor of graphical user interfaces - GUI).

In recent years has increased the number of services, such as digital signal processors programmed or create VHDL code.

1.2 HISTORY OF "MATLAB"

M programming language was created in the 70s, but the first version of MATLAB was created in 1984 by Cleve Moler. The main idea was to use subroutine packages written in Fortran Language in the courses of linear algebra and numerical analysis, without needing to write programs in that language.

In 2004 it was estimated that MATLAB was used by more than one million people in academic and business environments.

1.3 TOOL BOXES AND BLOCK PACKS

In MATLAB more than 35 boxes tool boxes and packs block (Simulink) are grouped. The following table (Table 27) shows these boxes:

TOOL BOXES	Simulink
Mathematics and Optimization	Modeling Fixed Point
Statistics and Data Analysis	Modeling Event-Based
Control System Design and Analysis	Modeling Physical
Signal Processing and Communications	Graphics Simulation
Image Processing	Control System Design and Analysis
Measuring and Test	Signal Processing and Communications
Computational Biology	Code Generation
Computational Finance	Quick Control Prototyping and SW/HW HIL
Application Development	Integrated Cards
Reports and database connection	Verification, Validation and Verification

Table 1 Tool boxes and block packs.

1.4 LIMITATIONS AND ALTERNATIVES

There have been criticism because MATLAB is a proprietary product of "The Mathworks," and users are limited and blocked by the seller.

But there is an additional tool called MATLAB Builder from "Application Deployment" to use functions MATLAB as library files that can be used for .NET or Java.

But the disadvantage is that the computer where the application has to be used requires MCR (MATLAB Component Runtime) for the MATLAB files work correctly. MCR may be freely distributed with the library files generated by MATLAB Compiler.

In the following list you can see alternatives to MATLAB:

- LabVIEW
- GNU Octave, software free likes a matlab.
- SAS
- Scilab
- Mathcad
- SciPy & Numerical Python
- Lenguaje R
- Álgebra computacional (SAC)

1.5 MATLAB IN THIS PROJECT

1.5.1 Simulink – MATLAB

Simulink is a visual programming environment that runs on MATLAB programming environment.

It is a programming environment of higher level of abstraction than the interpreted language Matlab (files with extensión .m). Simulink generates files with .mdl (of "model").

Simulink is a simulation tool models or systems, using block diagrams is achieved by making a theoretical simulation of different types of systems. You can see the output, both numerically and graphically.

It is important in the analysis of events, through system design (black boxes that perform some operation).

It is mainly used in Electrical Engineering issues related to digital signal processing (DSP), involving specific topics of biomedical engineering, telecommunications, among others. It is also widely used in Automatic Control Engineering and Systems.

1.5.1.1 HOW TO ACCESS FROM "MATLAB" TO Simulink

To access Simulink, first have to enter the MATLAB program. Then, you have two options to enter Simulink:

1. Write "simulink" in the command window of the program, as can be seen in the Figure 97:

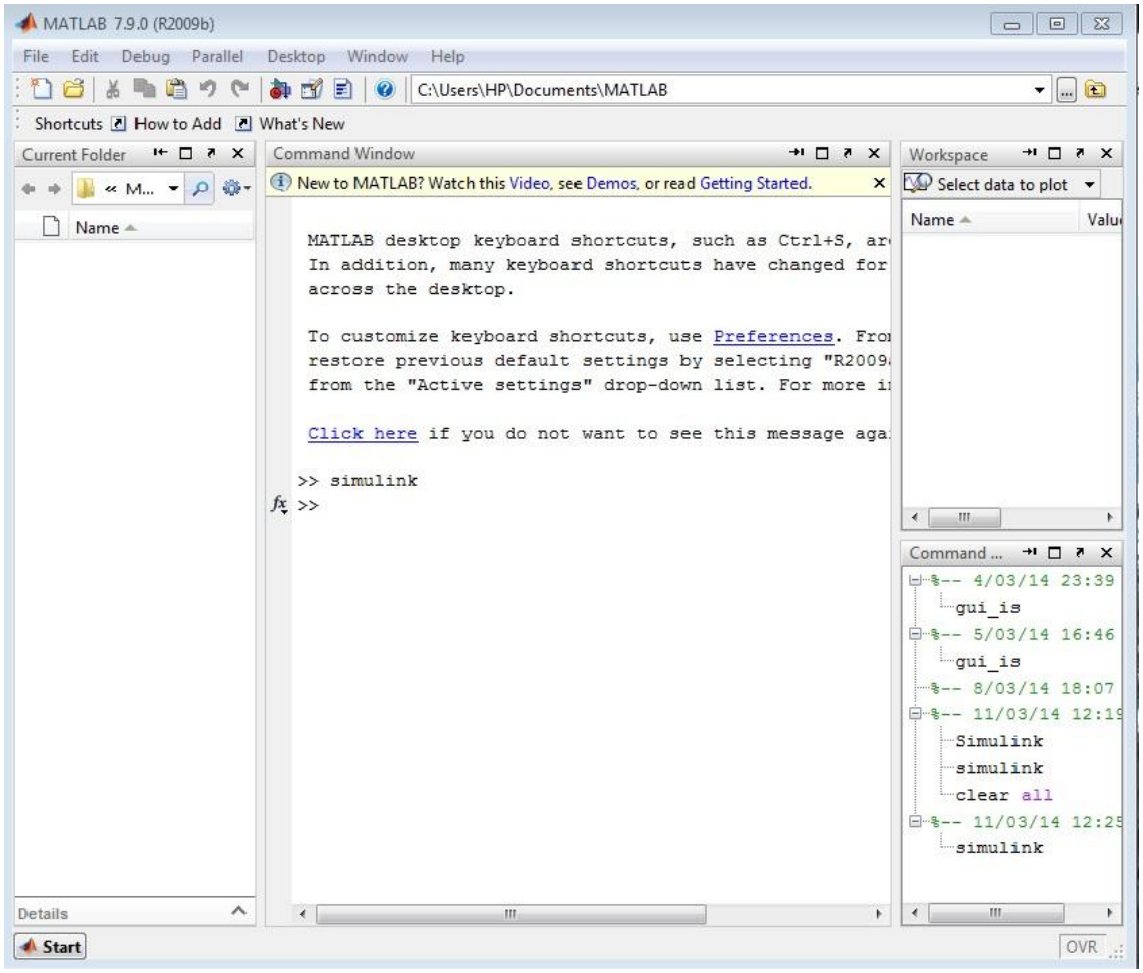


Figure 1 Matlab – Simulink (1).

And then it's open the Simulink window (Figure 98):

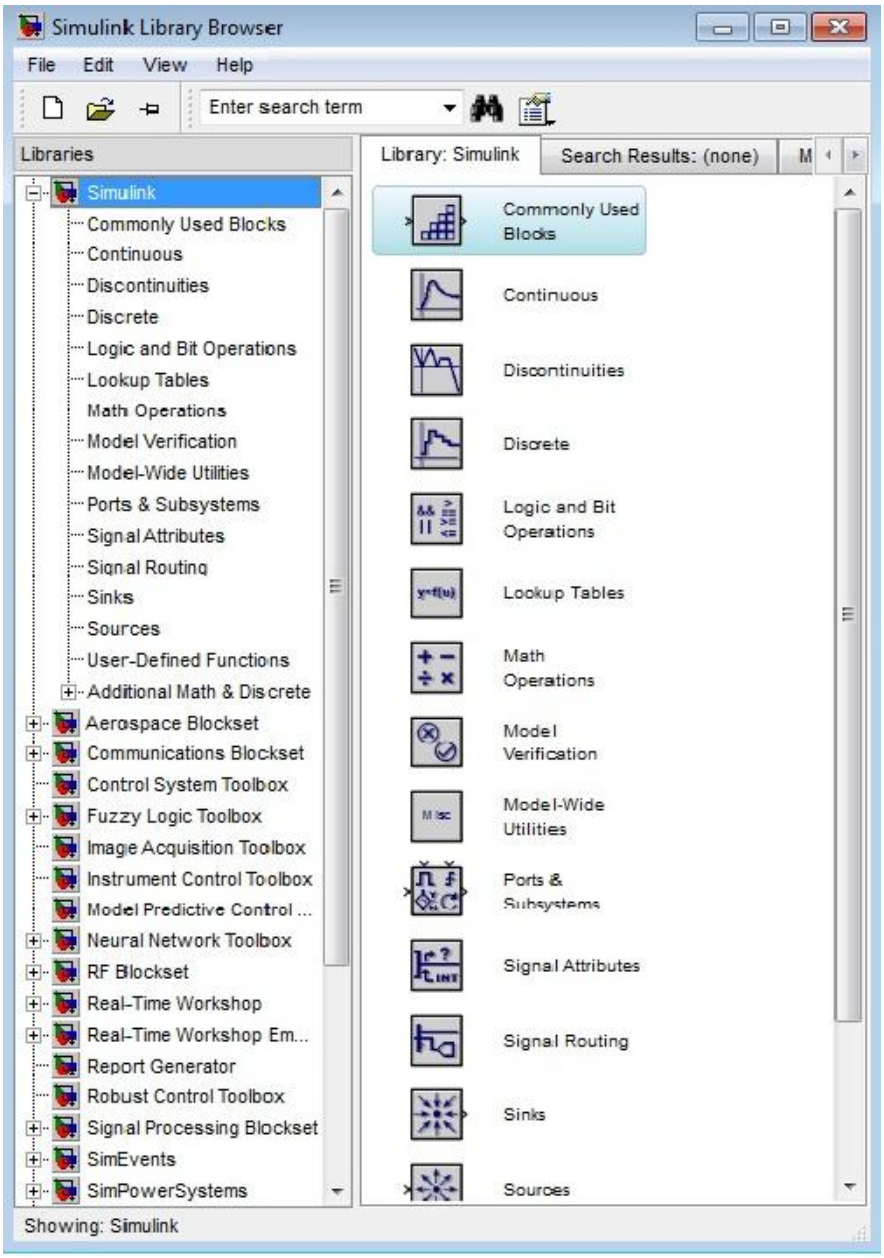


Figure 2 Matlab – Simulink (2).

2. Or press the button Simulink, as you can see in Figure 99:

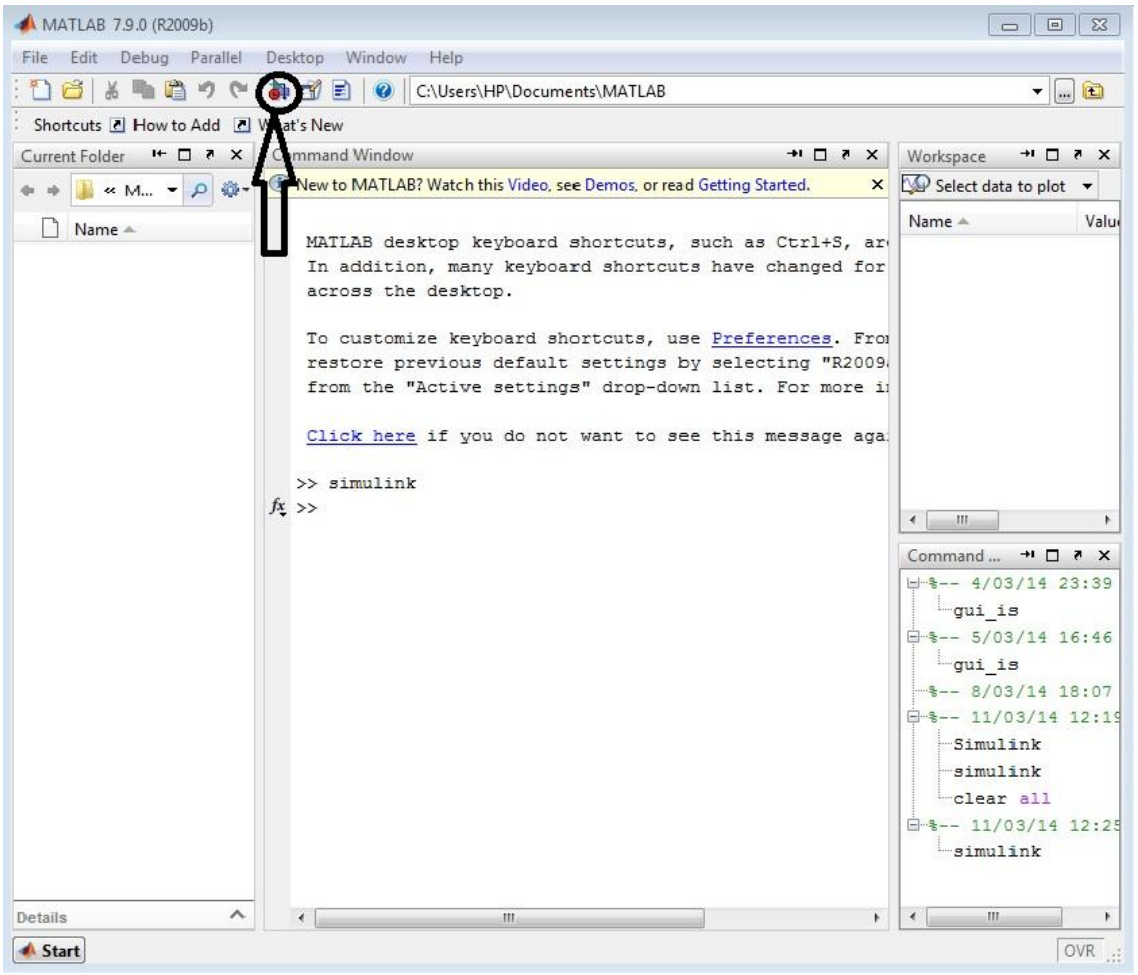


Figure 3 Matlab – Simulink (3).

To access the blocks that will be used in this project, we have to go to the option "SimPowerSystems" as the following figure (Figure 100) shows:

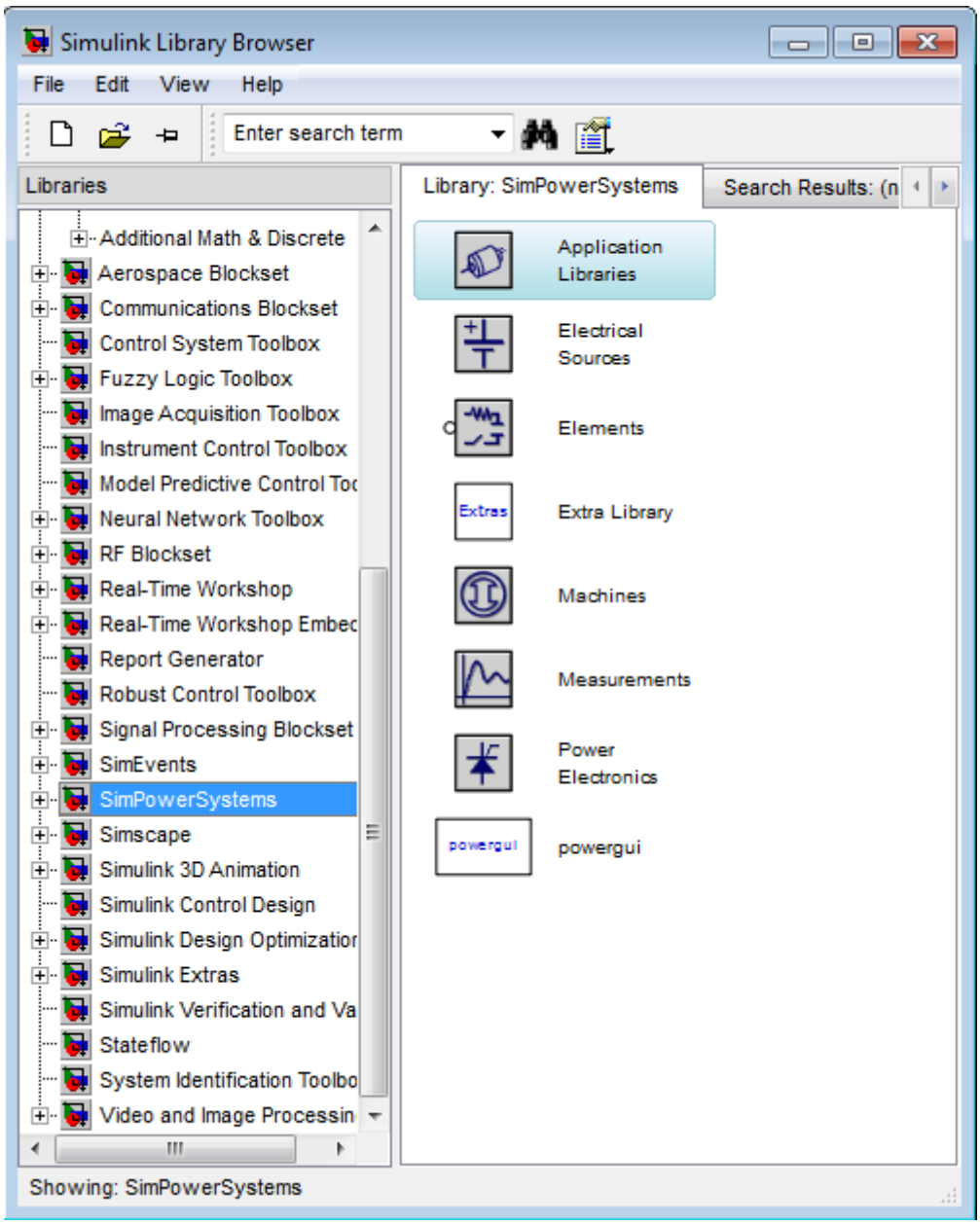


Figure 4 Matlab – Simulink (4).

1.5.2 GUI – MATLAB

GUI is a graphical user interface. With MATLAB can be programmed with M programming language and a platform to design computer graphics, in order to perform calculations and display the results in a more clear and intuitive way.

It can be seen in ANNEX IV of this project is the description of this graphical interface.

1.6 PROGRAM VERSIONS

Since 1984 there are several versions of this program, more than 35, but for the simulations with Simulink, and for building GUI for this Project, I have used the Version *MATLAB 7.8 R2009b* which stepped out into the market on 12th August of 2009.

2 SIMULATION OF INVERTERS WITH Simulink - MATLAB

2.1 NAME OF INVERTERS SIMULATED

In Annex I have been theoretically explained all investors covered by this project.

Below you can see in the list (Table 27), which type of investor is each Simulink file (.mdl):

.mdl	INVERTER
S_1	Half-Wave Inverter (Square Wave Control)
S_2	Half-Wave Inverter (PWM Control)
S_3	Asymmetric Inverter (Square Wave Control)
S_4	Asymmetric Inverter (PWM Control)
S_5	Full Wave H-Bridge Inverter (Square Wave Control)
S_6	Full Wave H-Bridge Inverter (Voltage Harmonic Cancellation)
S_7	Full Wave H-Bridge Inverter (Bipolar PWM Control)
S_8	Full Wave H-Bridge Inverter (Single-Pole PWM Control)
S_9	Three-Phase VSI 6-Step Inverter [Star] (Square Wave Control)
S_10	Three-Phase VSI 6-Step Inverter [Triangle] (Square Wave Control)
S_11	Three-Phase VSI 6-Step Inverter [Star] (PWM Control)
S_12	Three-Phase VSI 6-Step Inverter [Triangle] (PWM Control)
S_13	Three-Phase VSI 6-Step Inverter [Star] (Hysteresis Control)
S_14	Push-Pull Inverter (Square Wave Control)
S_15	Push-Pull Inverter (PWM Control)
S_16	Cascade Full Bridge Single-Phase Inverter (2 Levels)
S_17	Cascade Full Bridge Single-Phase Inverter (4 Levels)
S_18	Cascade Full Bridge Three-Phase Inverter (3 Levels)
S_19	NPC Inverter
S_20	NPC Inverter (5 Levels)
S_21	NPC Three-Phase Inverter
S_22	FC Inverter (5 Levels)
S_23	FC Three-Phase Inverter (3 Levels)

Table 2 Inverter Simulations with Matlab – Simulink.

2.2 BASIC STEPS TO START SIMULINK

2.2.1 WHERE ARE THE FILES?

The files of these simulations are in “Matlab Simulations” folder.

2.2.2 HOW TO OPEN THE FILES

To open files simulation, first have to go to the left window of the program (MATLAB), and select the folder where the files are saved, as shown in the following figure (Figure 101):

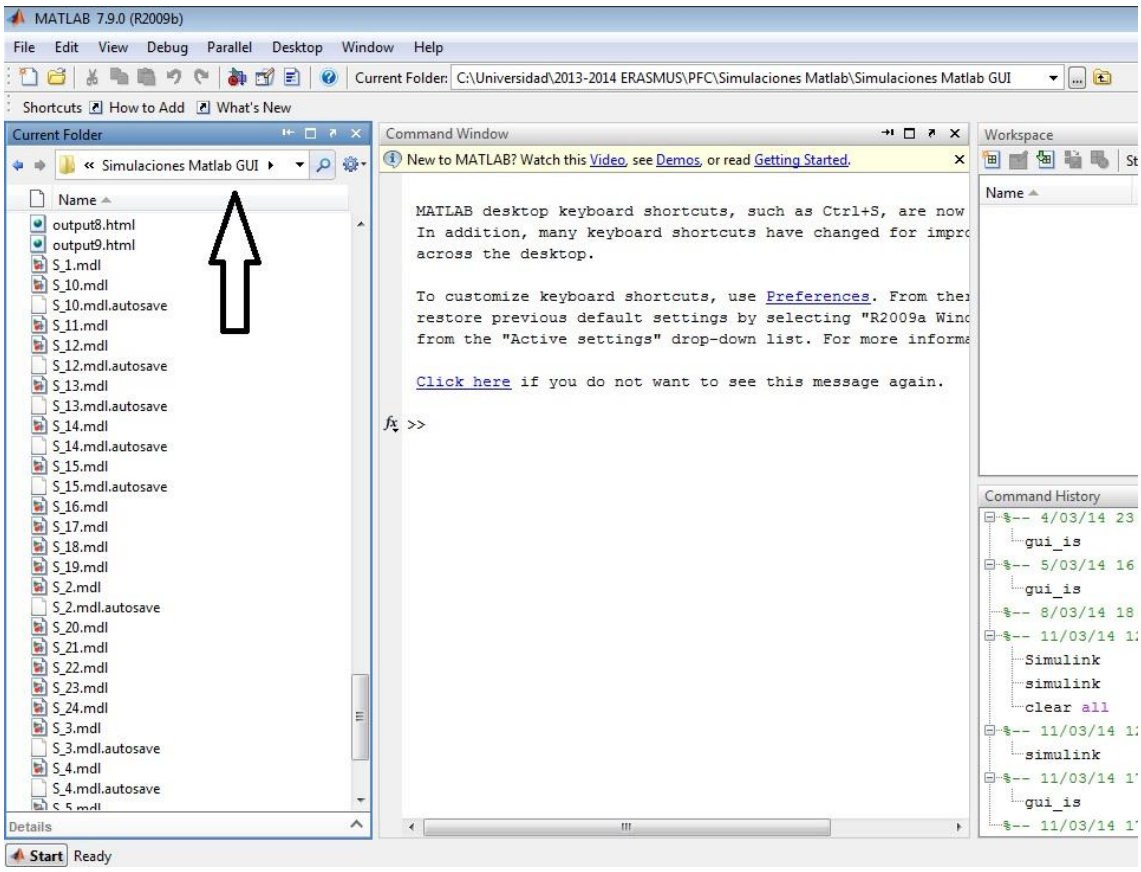


Figure 5 Matlab – Simulink (5).

After we have selected the folder, we have two options to open a simulation:

1. Writing the file name in the command window of MATLAB, for example (Figure 102):

>> S_1

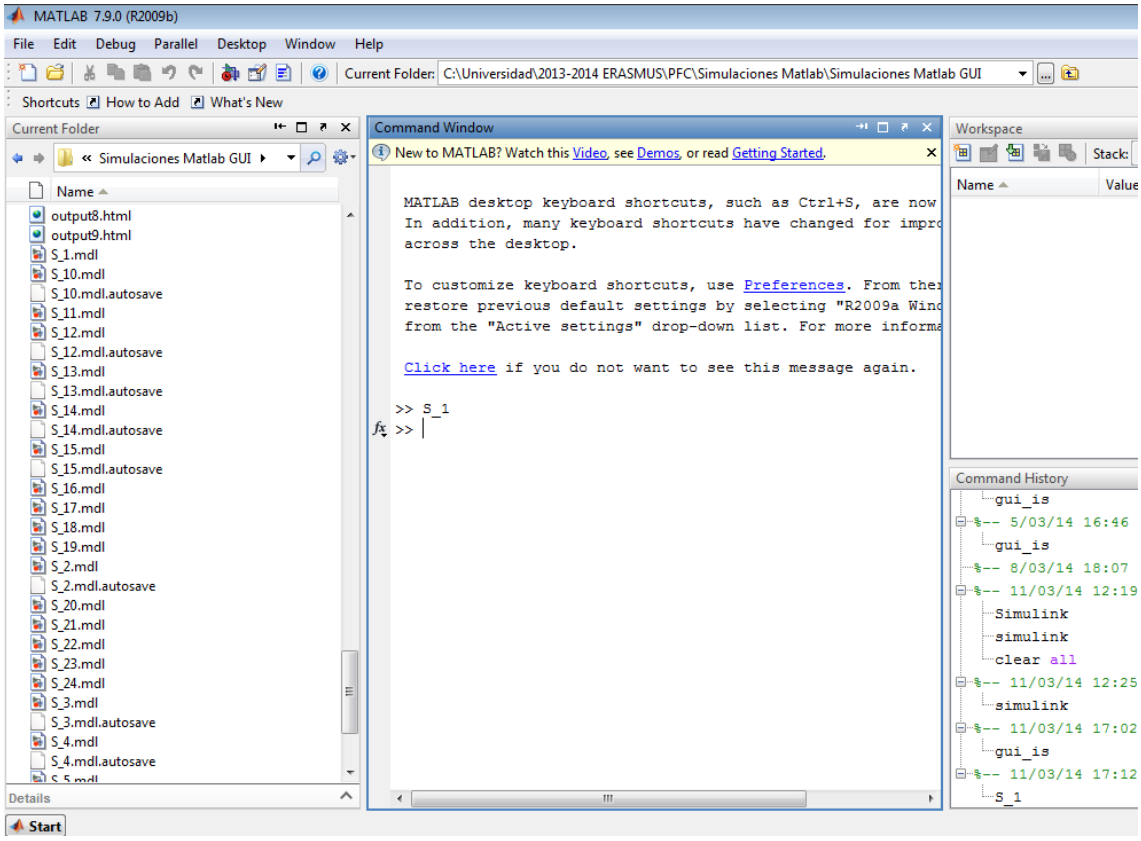


Figure 6 Matlab – Simulink (6).

2. Double-click directly on the file in the left window (Figure 103):

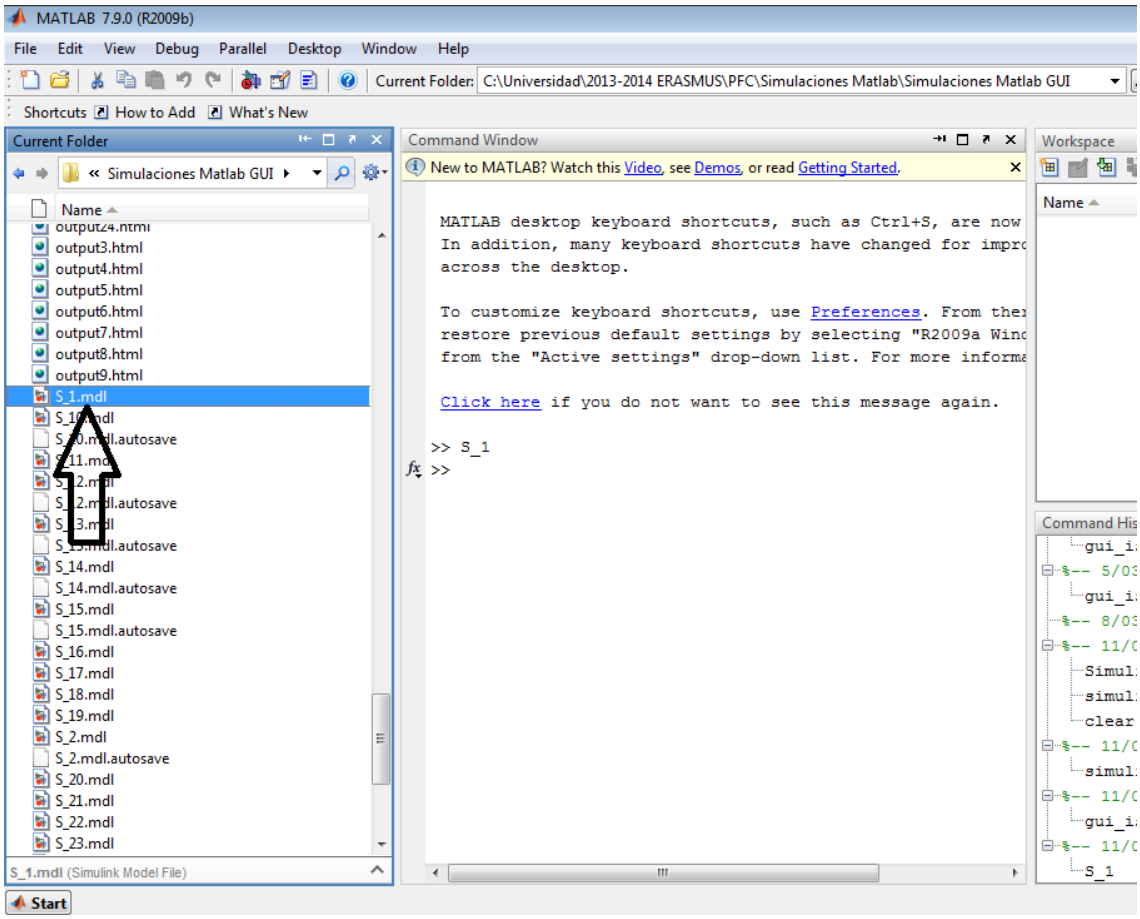


Figure 7 Matlab – Simulink (7).

2.2.3 HOW TO RUN THE FILES

When you have opened Simulink, to run the simulation press the button PLAY located at the top of the window (Figure 104). Right there beside you can change the time to be simulated.

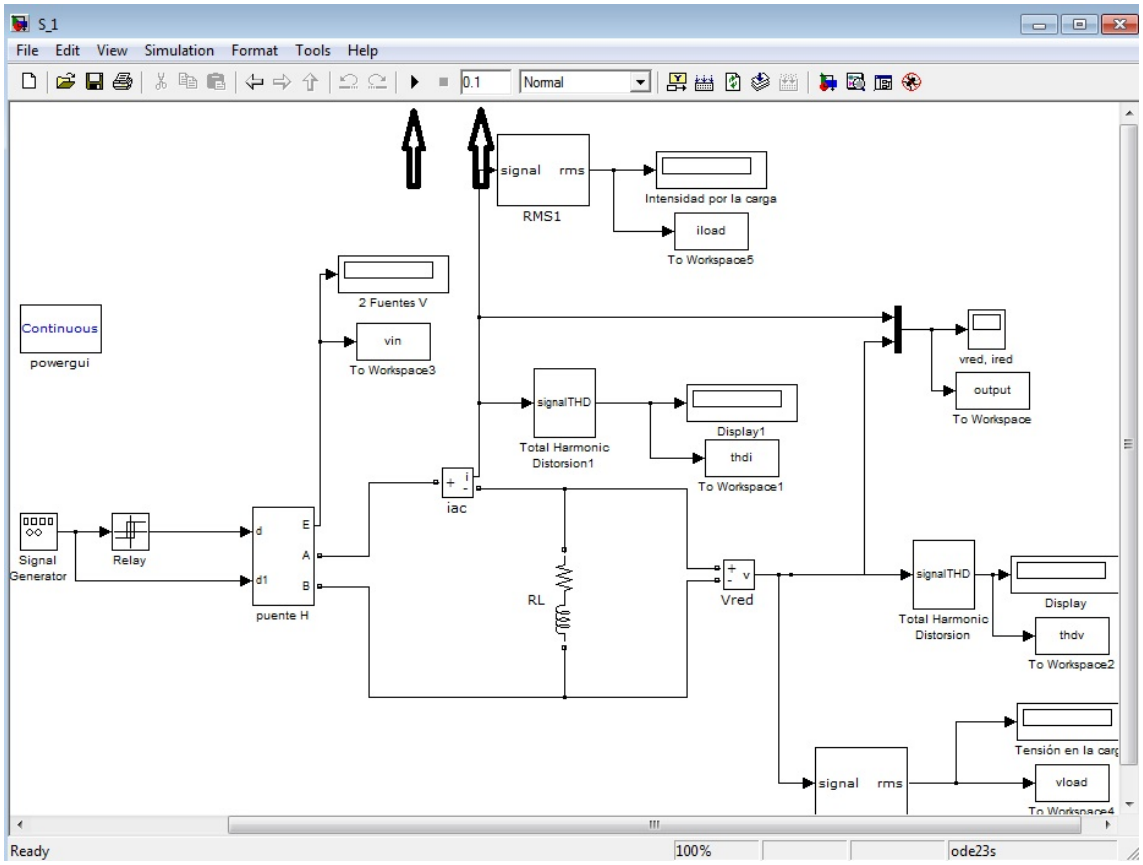


Figure 8 Matlab – Simulink (8).

These are just the basic steps. May be found all the information to use Simulink in the User's Guide MATLAB - Simulink.

2.3 ALL SIMULATIONS WITH SIMULINK

2.3.1 S_1 - Half-Wave Inverter (Square Wave Control)

When the file S_1 is opened, it may see the following block diagram corresponding to - Half-Wave Inverter (Square Wave Control). It can see that there are the load, de Signal Generator of the control and one “SubSystem” where the bridge is inside.

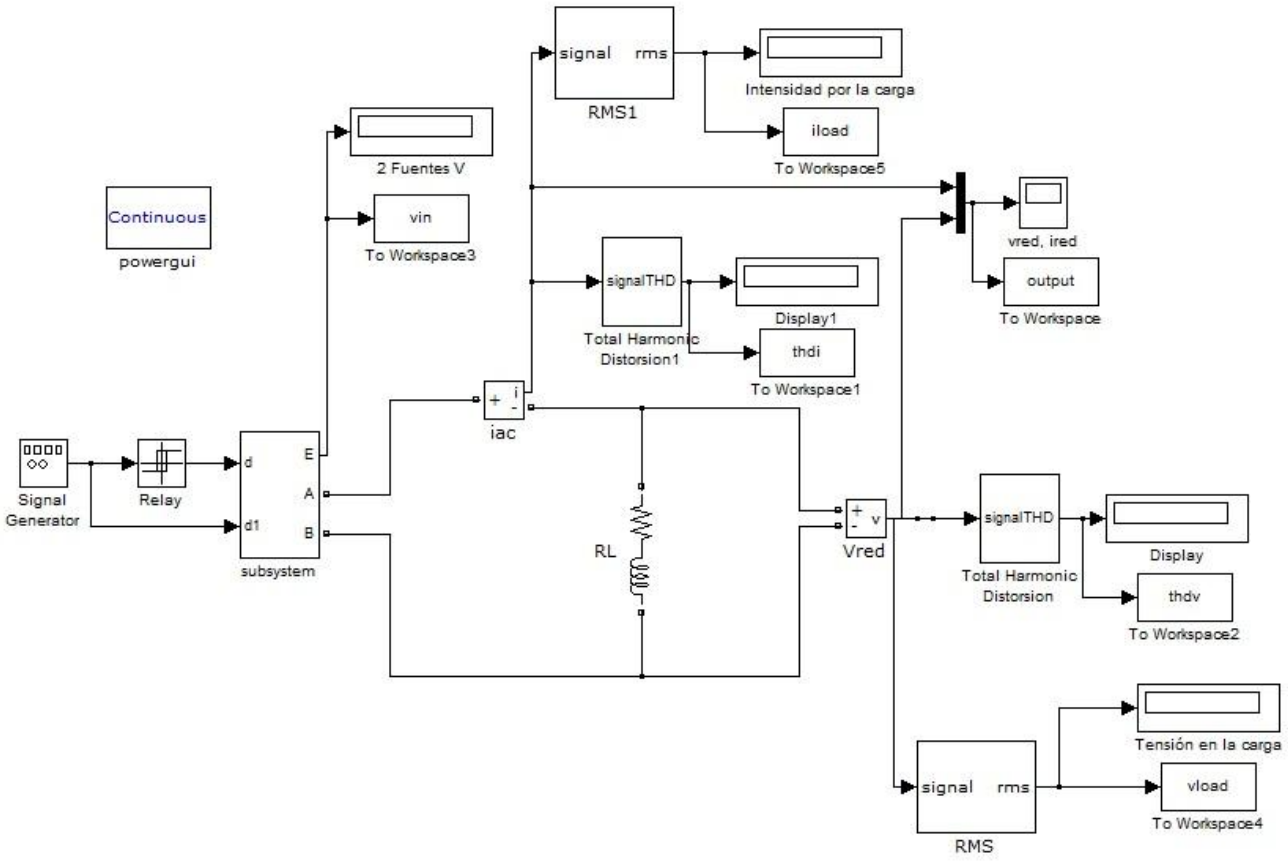


Figure 9 Inverter Scheme Simulink S_1.

In the next picture can be seen the bridge is located within the SubSystem:

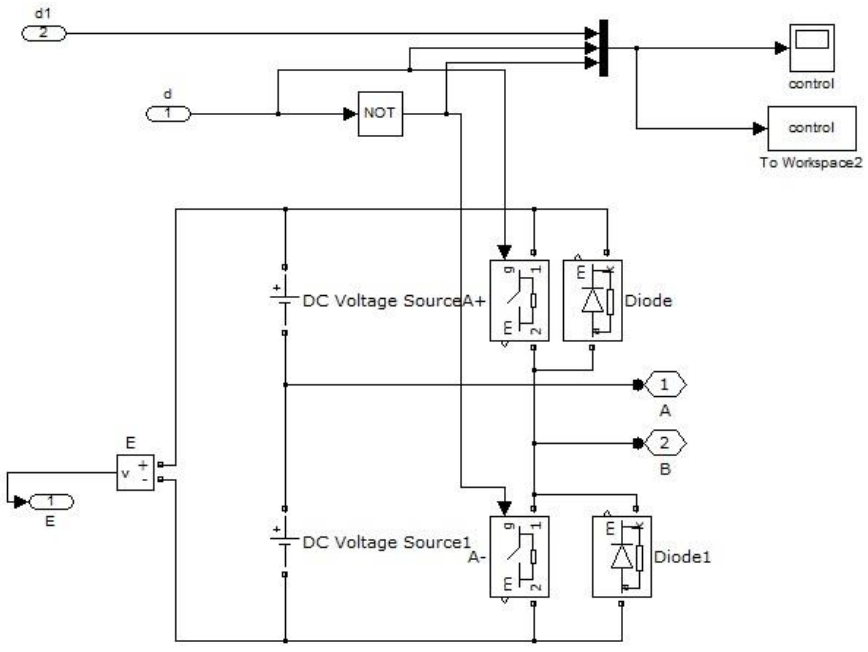


Figure 10 SubSystem Inverter Scheme Simulink S_1.

After pressing Play and the simulation is finished, it can double-click on the Scope to see the waveforms.

The first graph corresponds to the waveform of the output at the load. The yellow line represents the Current that runs through load and purple line represents the Voltage in it:

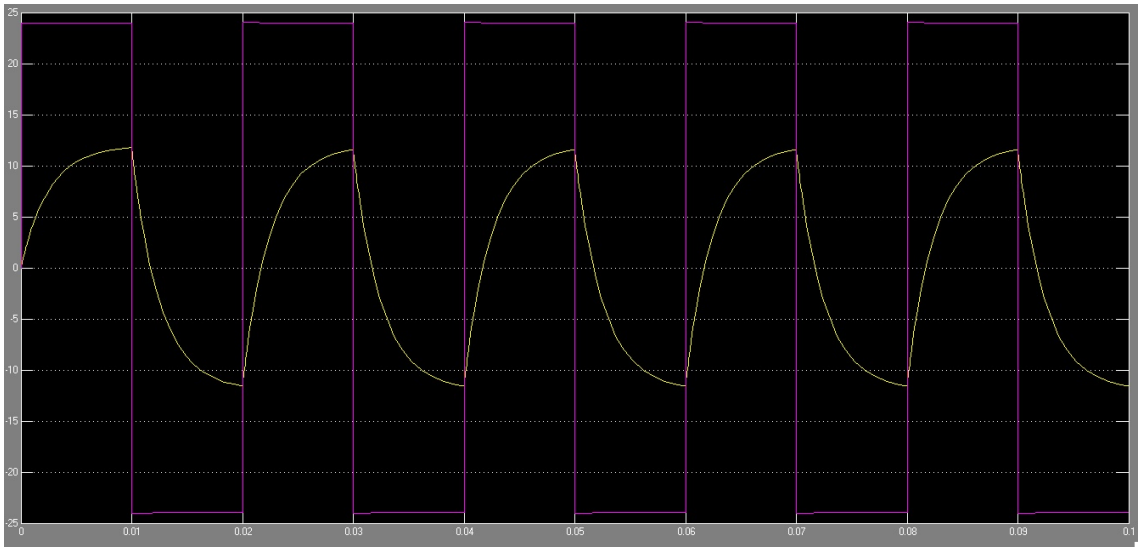


Figure 11 Output Load Inverter Simulink S_1.

Below it can see the waveform of the inverter control. Initially we have a square wave (yellow line). Depending on whether the value of the square wave is positive or negative, will be shot a transistor or the other (blue or purple line):

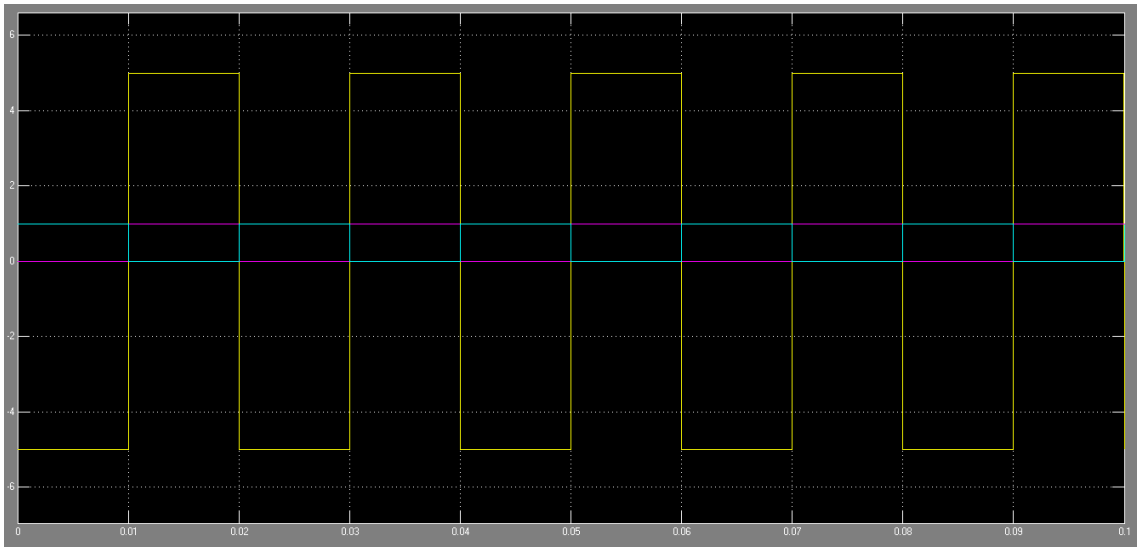


Figure 12 Control Inverter Simulink S_1.

The inverter control corresponds to that explained above in the theoretical information on the inverter.

All other investors in Simulink will proceed in the same way, so from now on only the characteristic information of each one of them will be written.

2.3.2 S_2 - Half-Wave Inverter (PWM Control)

Main block diagram:

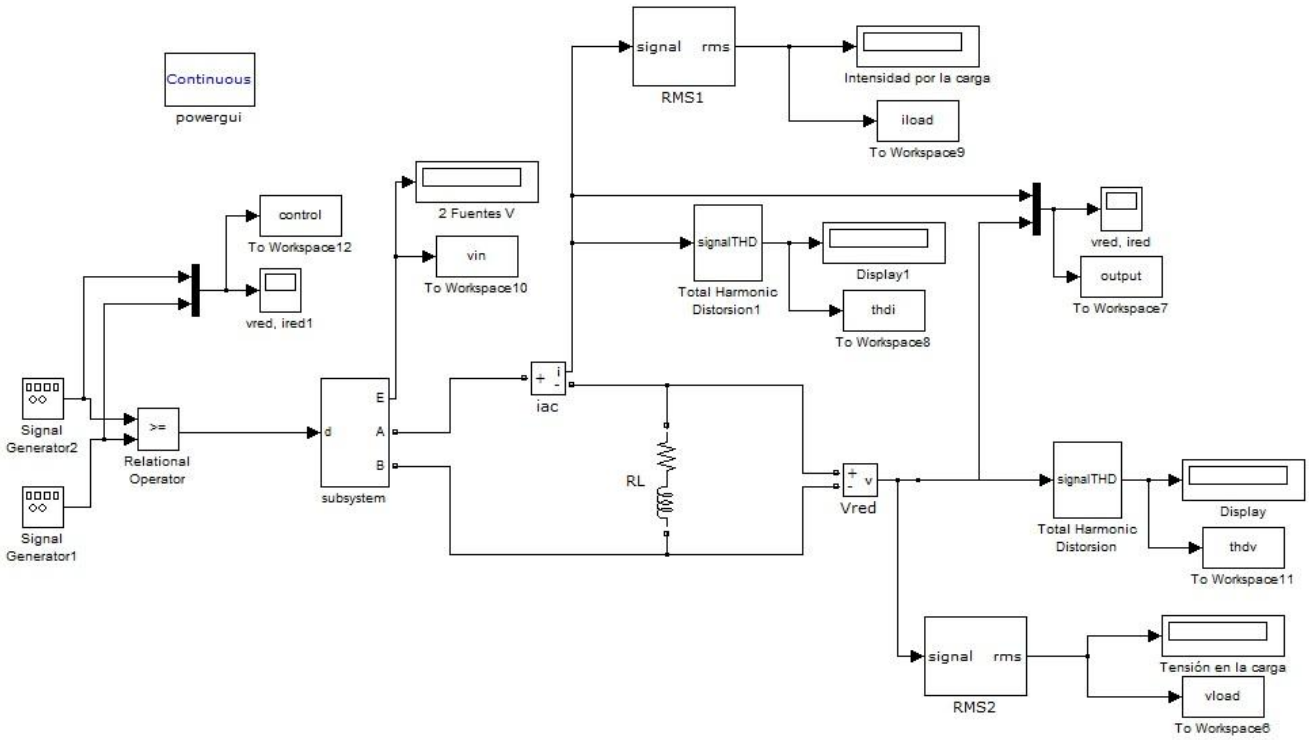


Figure 13 Inverter Scheme Simulink S_2.

SubSystem:

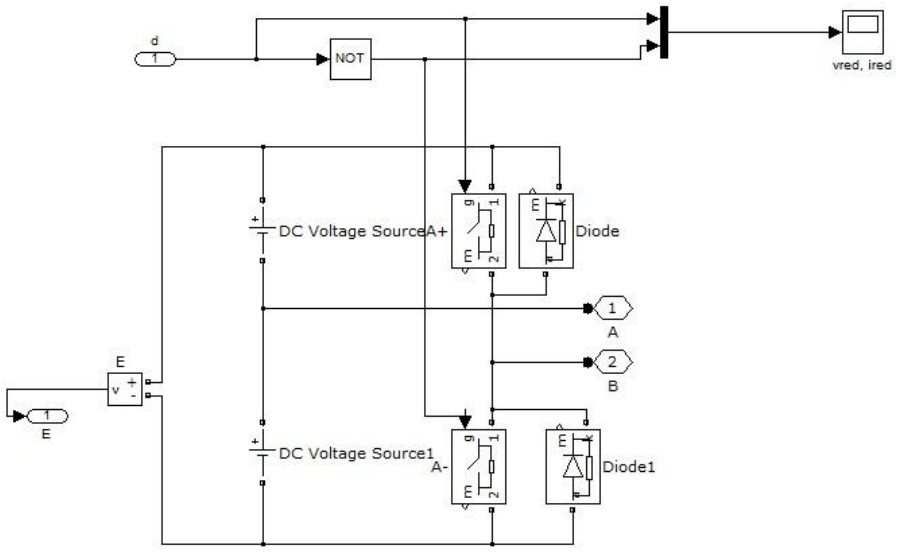


Figure 14 SubSystem Inverter Simulink Scheme S_2.

Graph of the waveform at the output load. It can see the sinusoidal form of the Current with some peaks (yellow line), and the purple one represents the Voltage in the output load. As it's a PWM control, the output voltage is like a square waveform but with different size of pulse width:

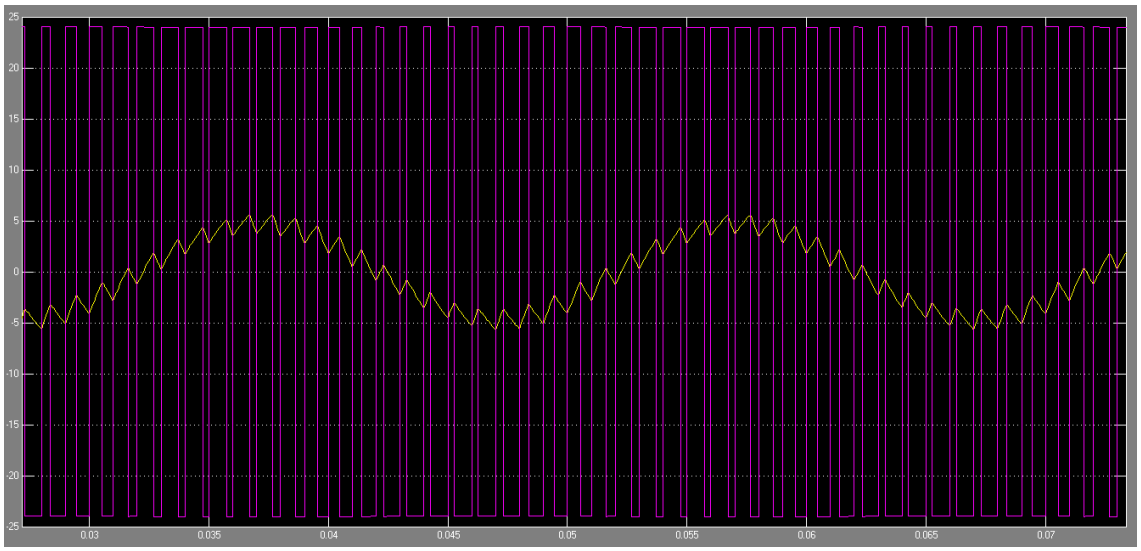


Figure 15 Output Load Inverter Simulink S_2.

Graph of the waveform of the inverter control. It can see the most popular PWM control system. There is a triangular waveform, and another sinusoidal waveform. When the Triangular is higher than Sinusoidal, one transistor is working and the other one not. And when de Triangular is less than Sinusoidal, the opposite phenomenon occurs:

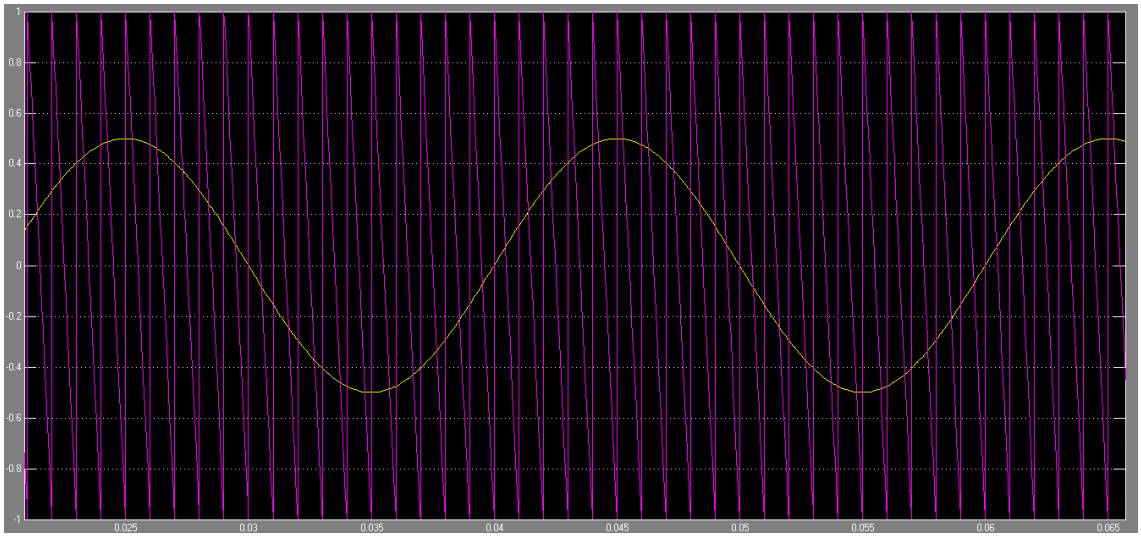


Figure 16 Control Inverter Simulink S_2.

2.3.3 S_3 - Asymmetric Inverter (Square Wave Control)

Main block diagram:

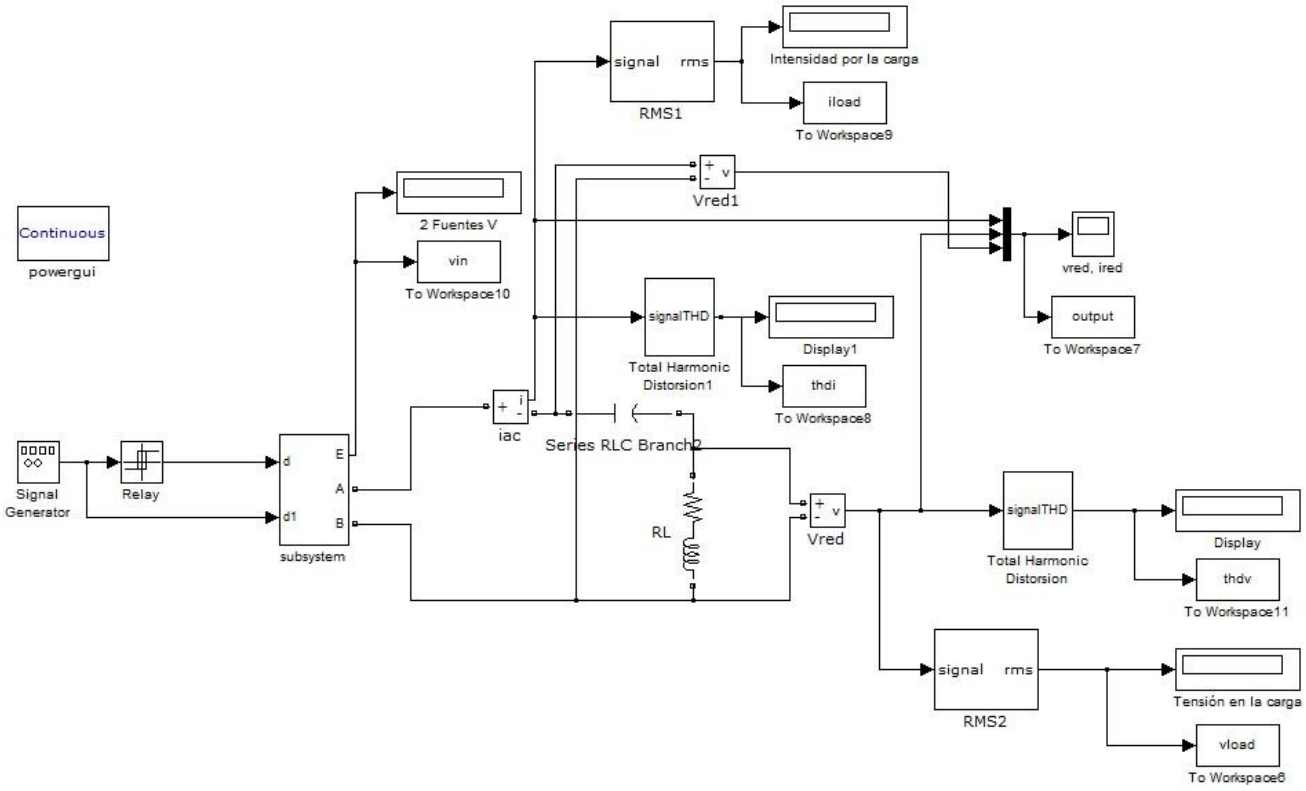


Figure 17 Inverter Scheme Simulink S_3.

SubSystem:

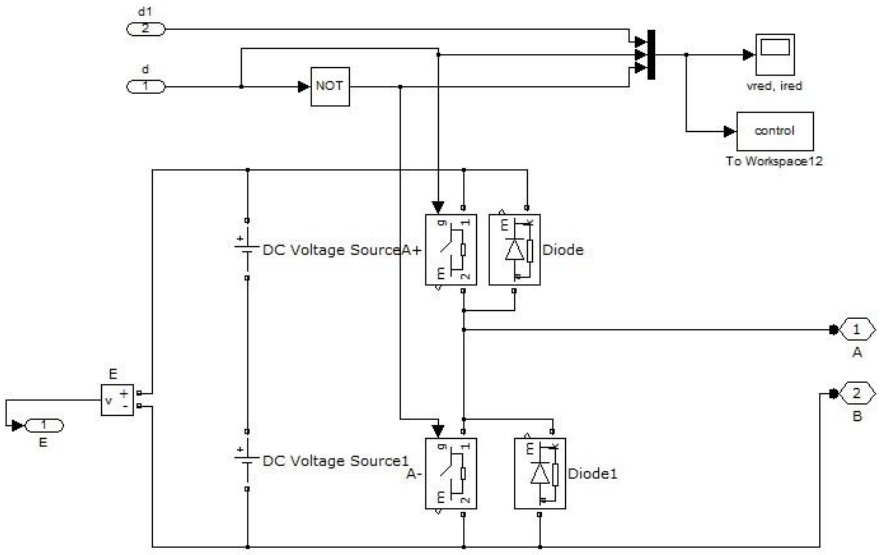


Figure 18 SubSystem Inverter Scheme Simulink S_3.

Graph of the waveform at the output load. The yellow line represents the Current that runs through load and purple line represents the Voltage in it. Blue line corresponds to the voltage of the output load with the filtering capacitor:

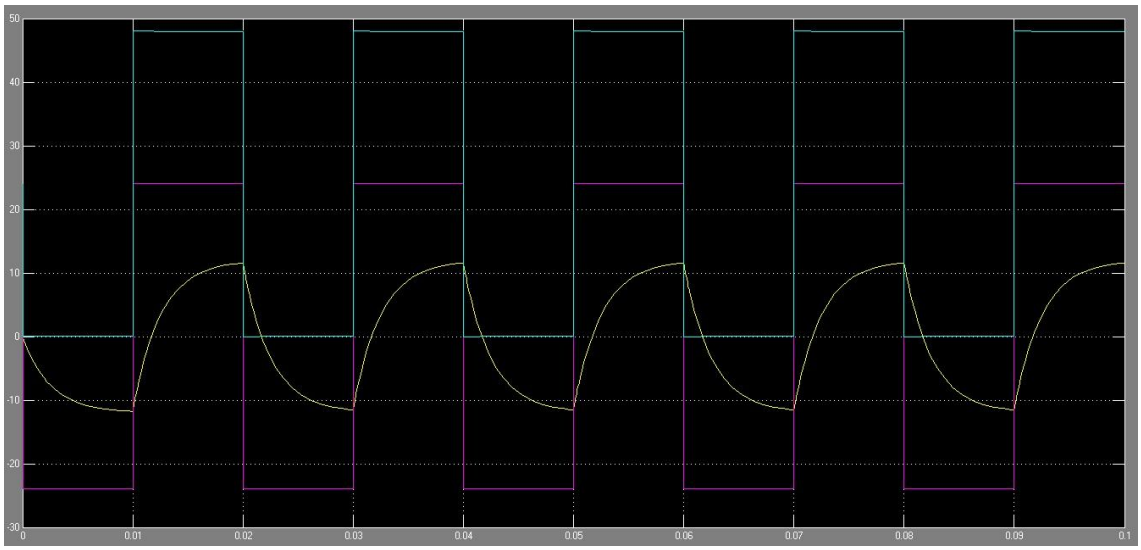


Figure 19 Output Load Inverter Simulink S_3.

Graph of the waveform of the inverter control (Square Wave Control):

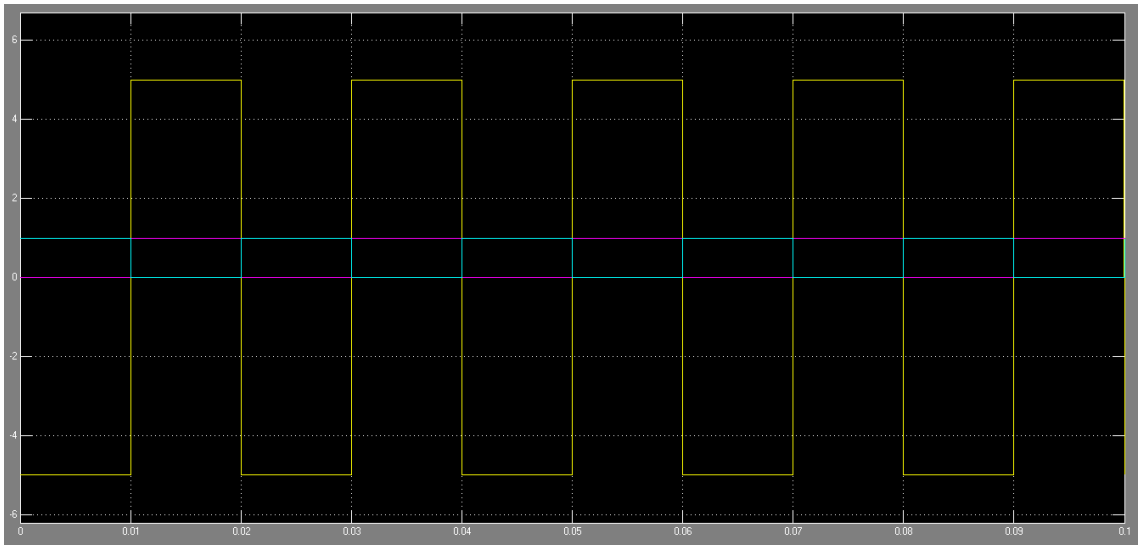


Figure 20 Control Inverter Simulink S_3.

2.3.4 S_4 - Asymmetric Inverter (PWM Control)

Main block diagram:

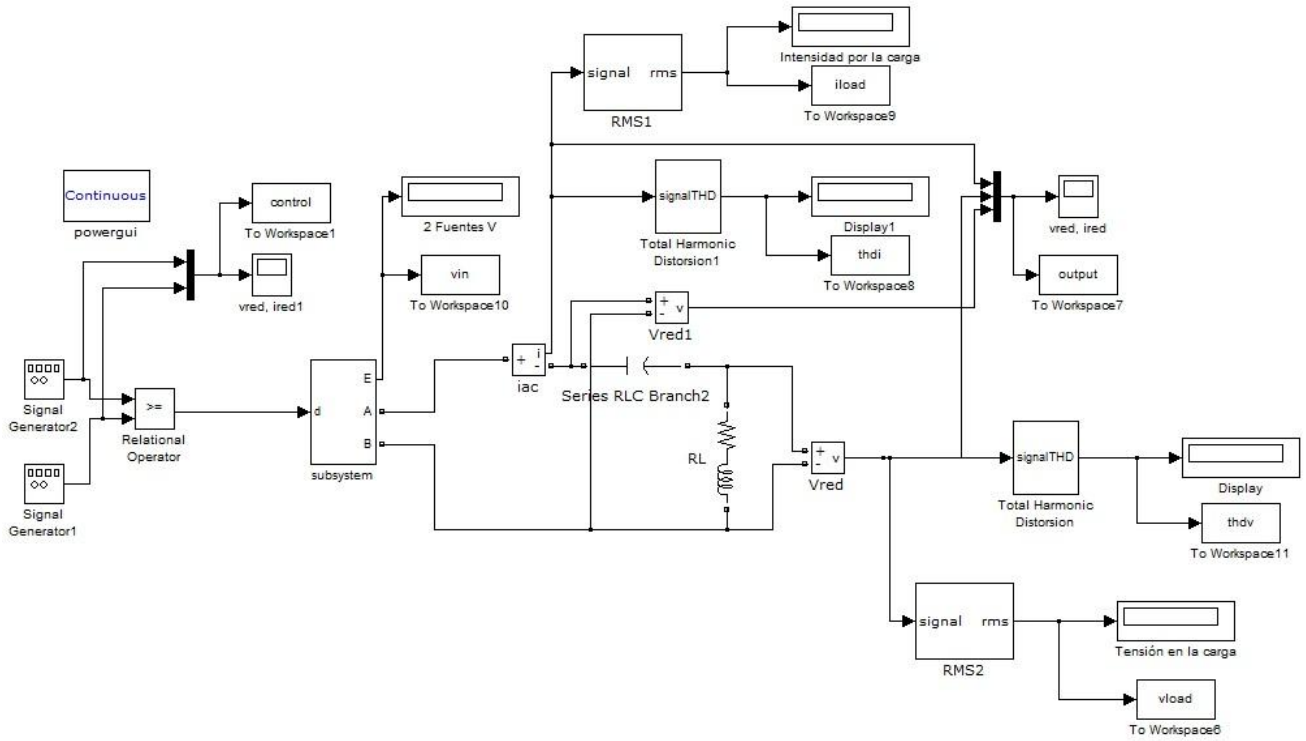


Figure 21 Inverter Scheme Simulink S_4.

SubSystem:

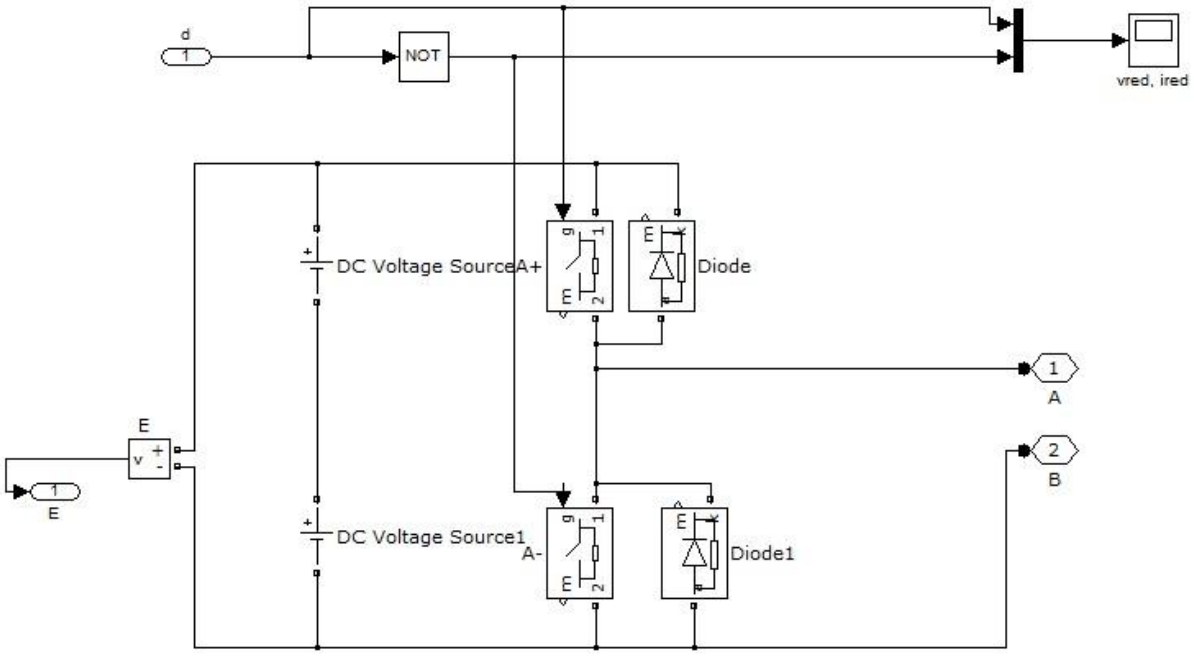


Figure 22 SubSystem Inverter Scheme Simulink S_4.

Graph of the waveform at the output load. The yellow line represents the Current that runs through load and purple line represents the Voltage in it. Blue line corresponds to the Voltage of the output load with the filtering capacitor:

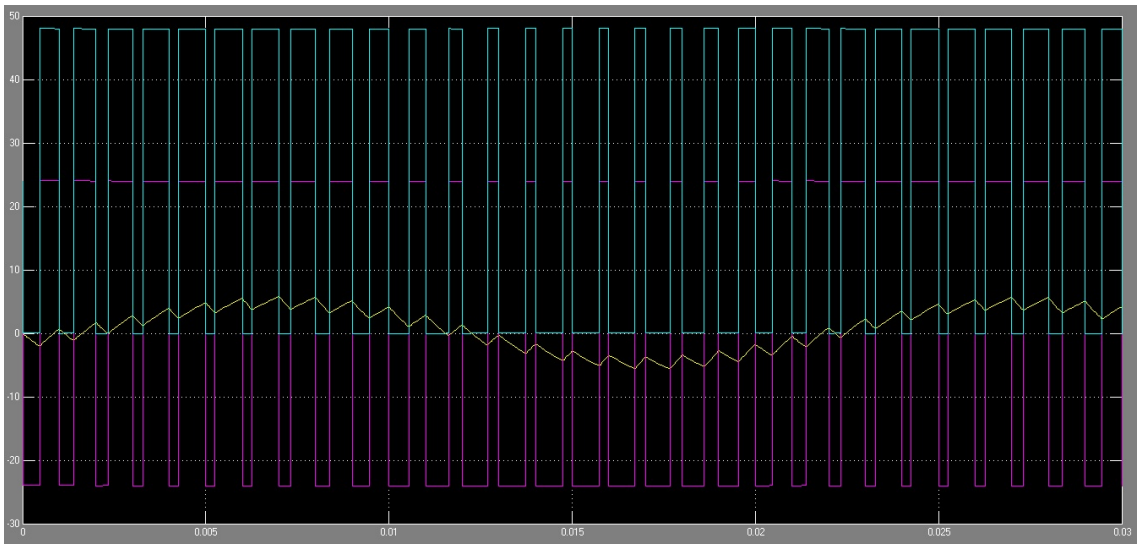


Figure 23 Output Load Inverter Simulink S_4.

Graph of the waveform of the inverter control (PWM Control):

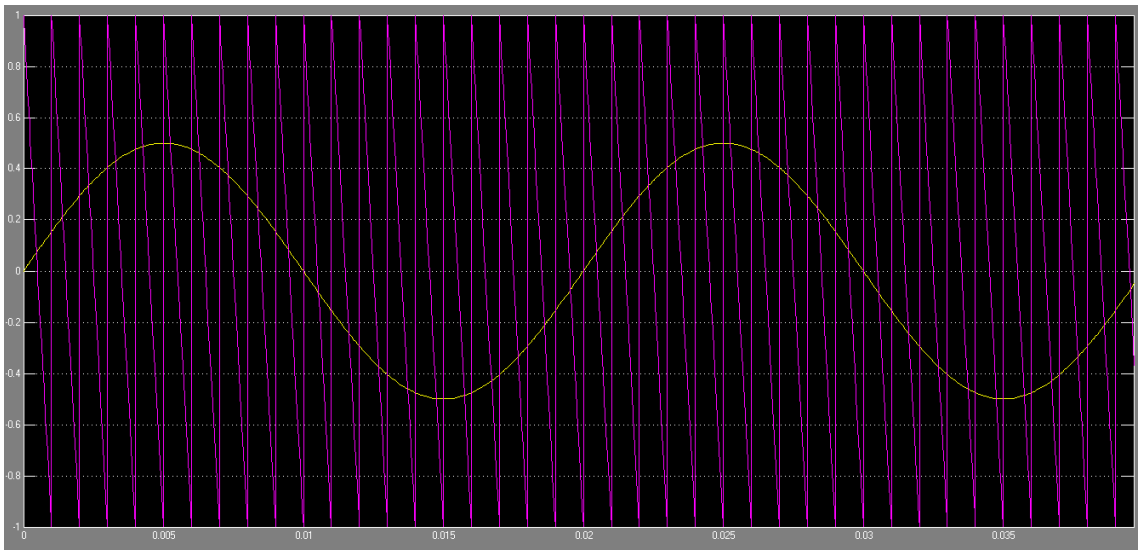


Figure 24 Control Inverter Simulink S_4.

2.3.5 S_5 - Full Wave H-Bridge Inverter (Square Wave Control)

Main block diagram:

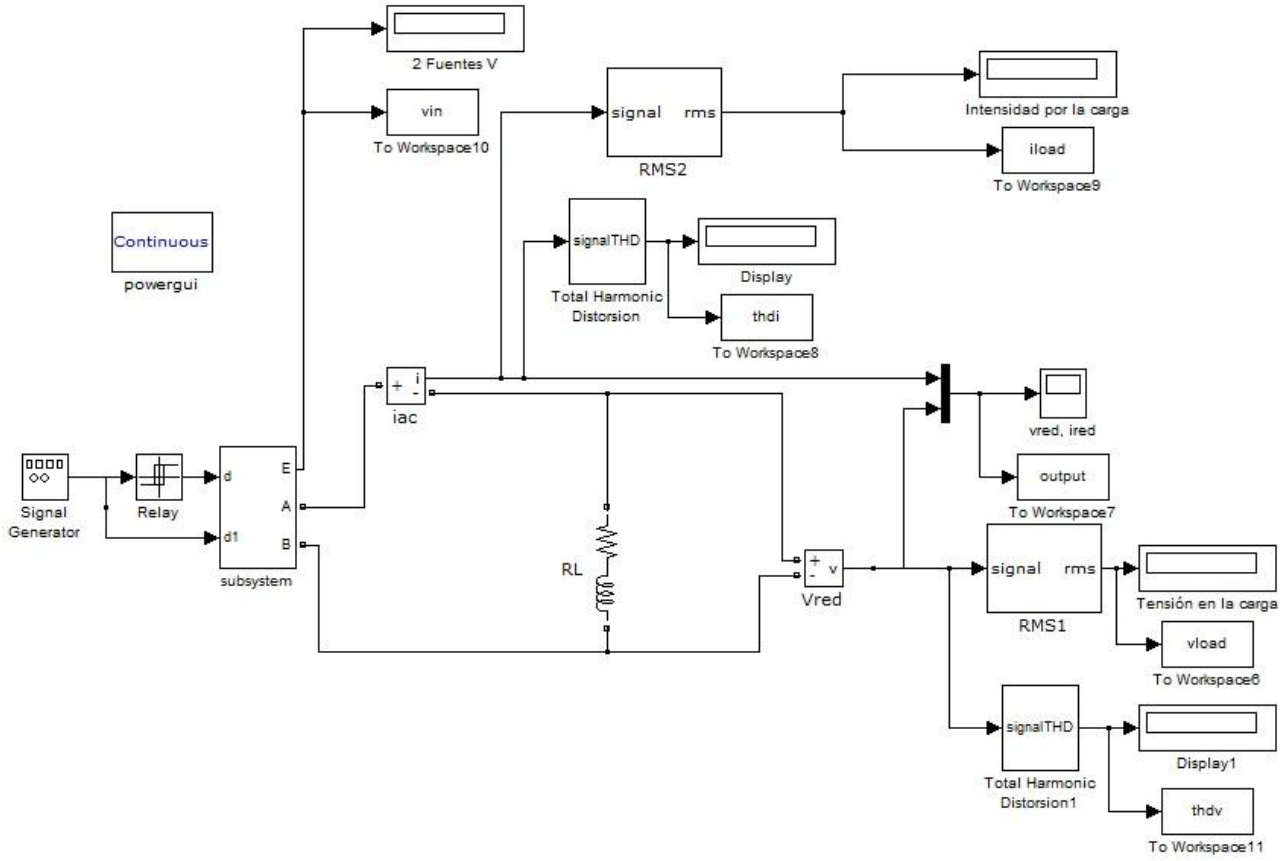


Figure 25 Inverter Scheme Simulink S_5.

SubSystem:

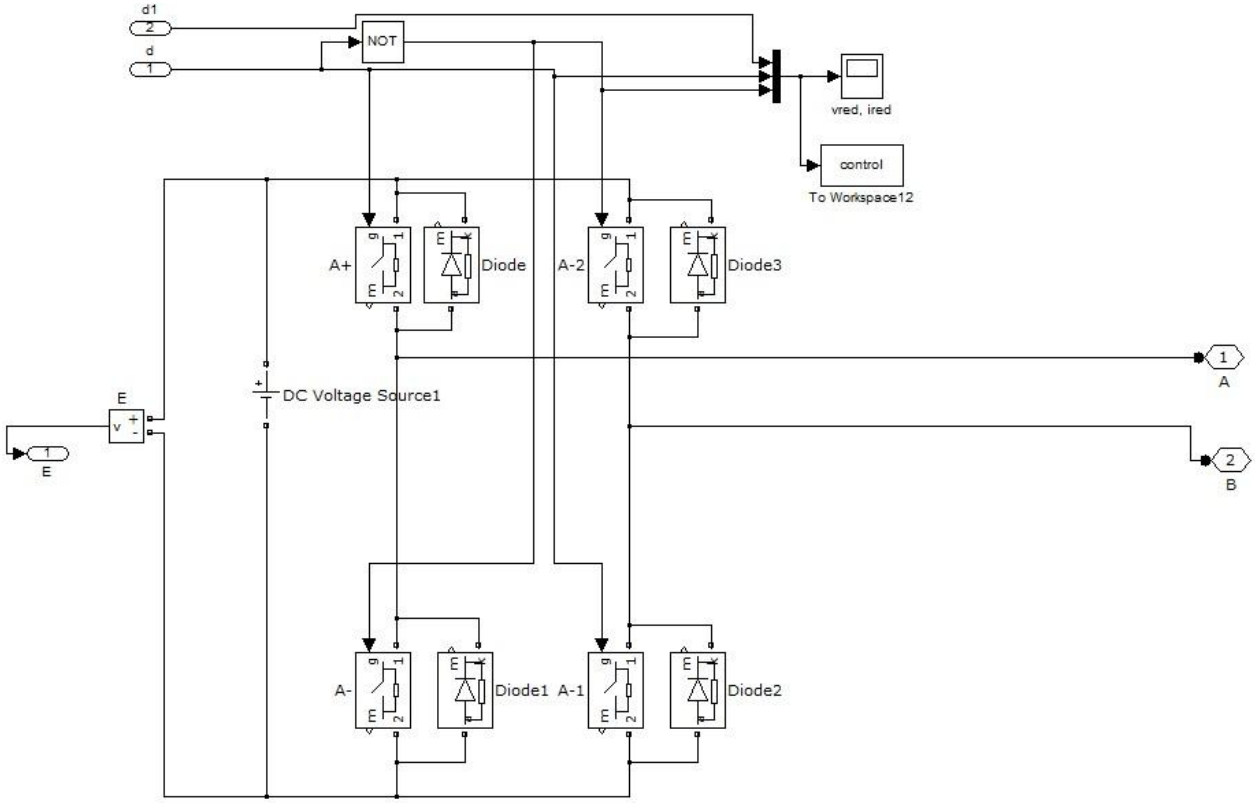


Figure 26 SubSystem Inverter Scheme Simulink S_5.

Graph of the waveform at the output load. The yellow line represents the Current that runs through load and purple line represents the Voltage in it:

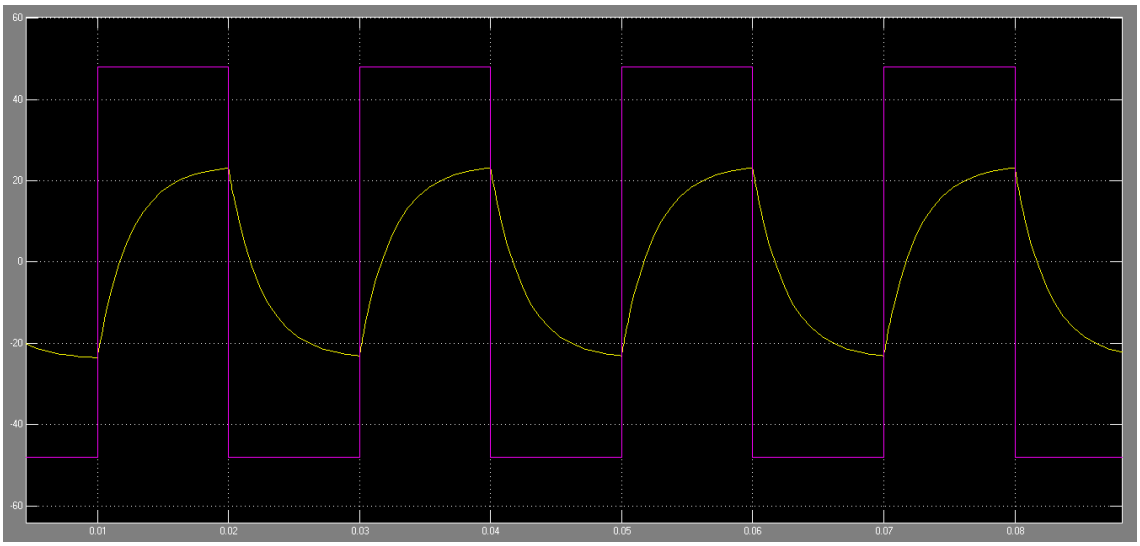


Figure 27 Output Load Inverter Simulink S_5.

Graph of the waveform of the inverter control (Square Wave Control):

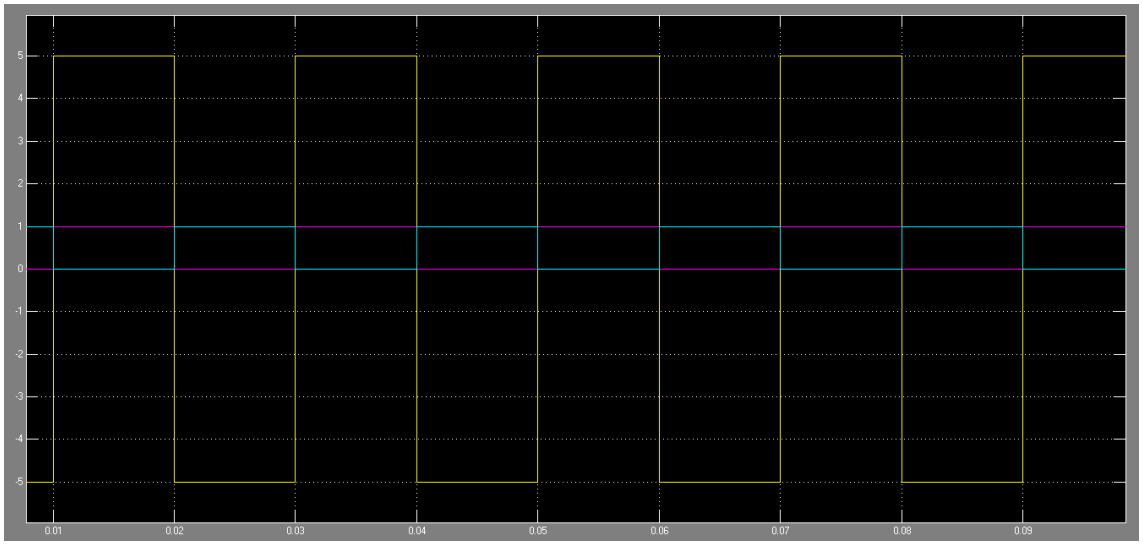


Figure 28 Control Inverter Simulink S_5.

2.3.6 S_6 - Full Wave H-Bridge Inverter (Voltage Harmonic Cancellation)

Main block diagram:

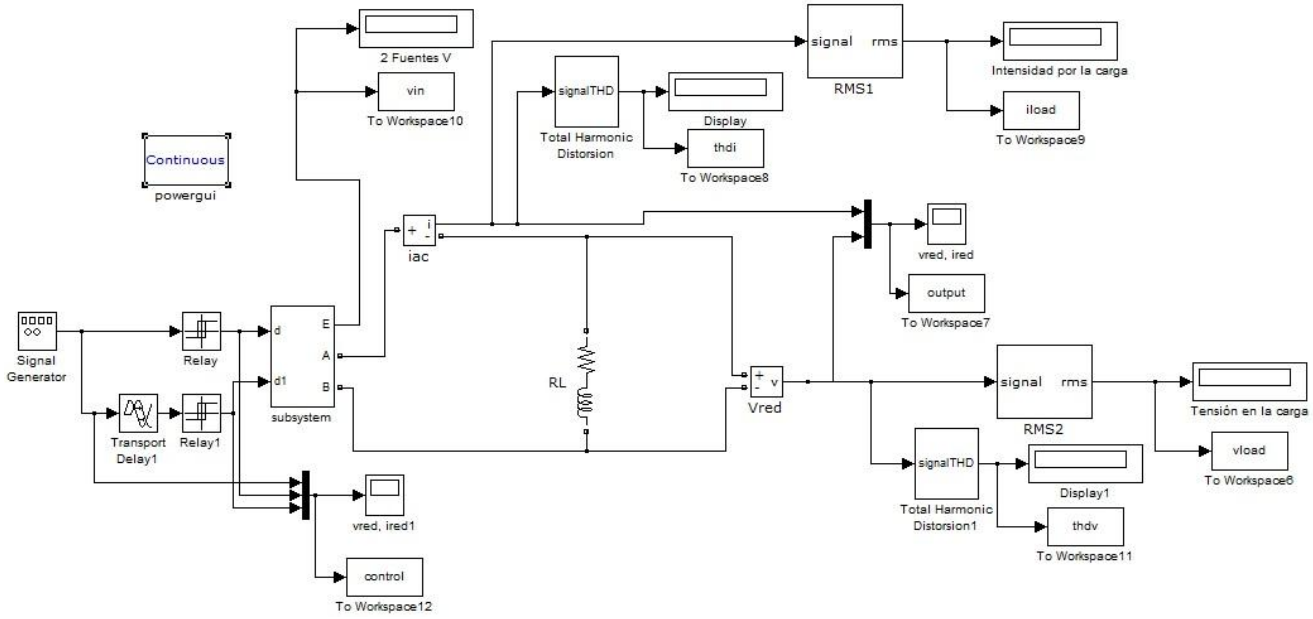


Figure 29 Inverter Scheme Simulink S_6.

SubSystem:

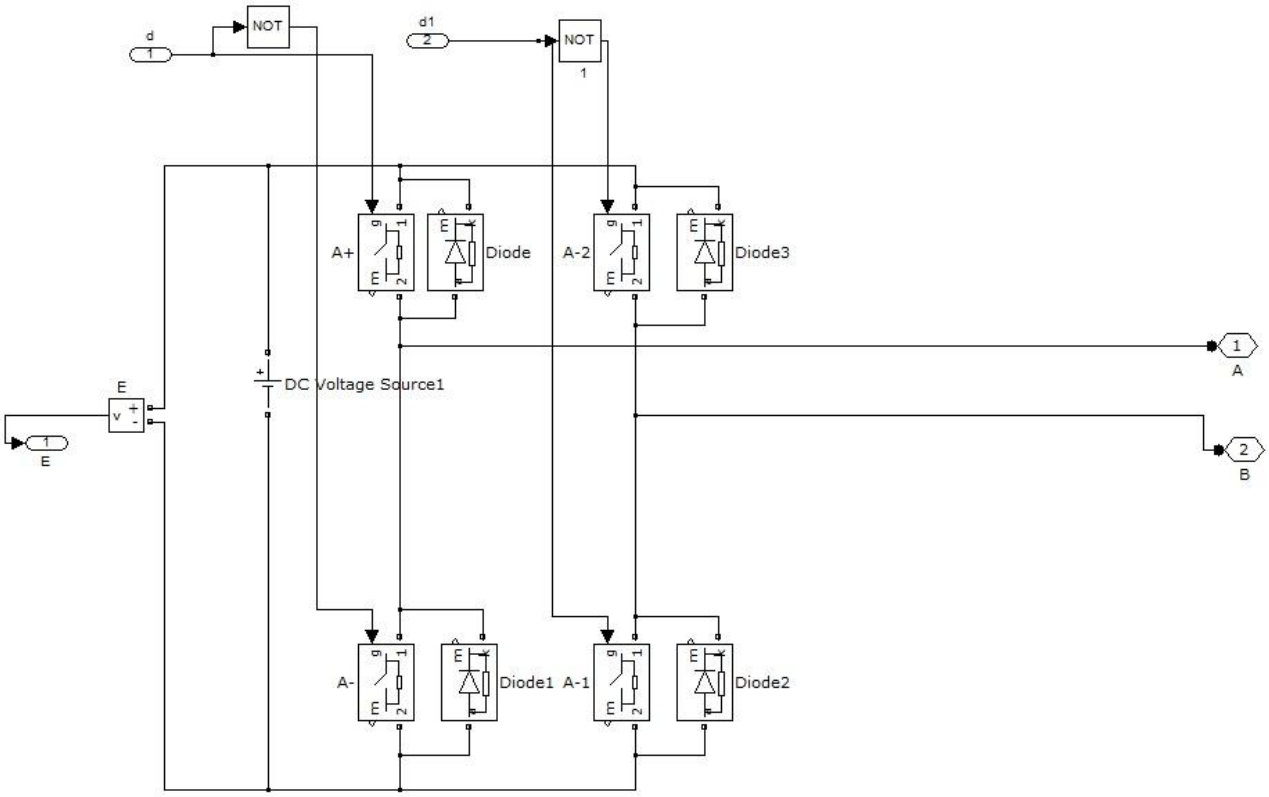


Figure 30 SubSystem Inverter Scheme Simulink S_6.

Graph of the waveform at the output load. The yellow line represents the Current that runs through load and purple line represents the Voltage in it:

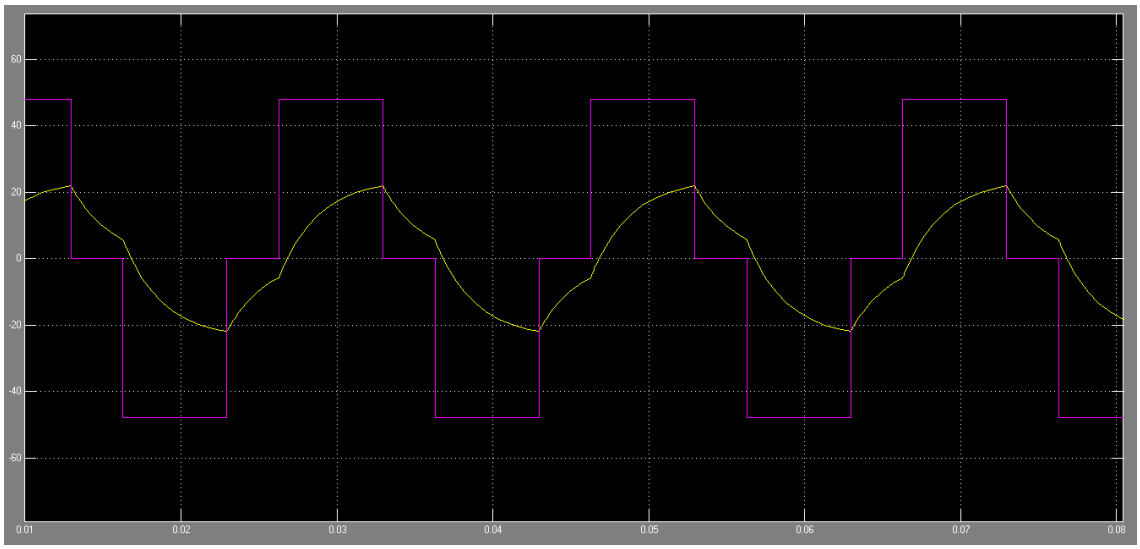


Figure 31 Output Load Inverter Simulink S_6.

Graph of the waveform of the inverter control (Square Wave Control). This system of control is independent in each of the sides of inverter. One square wave form for each one:

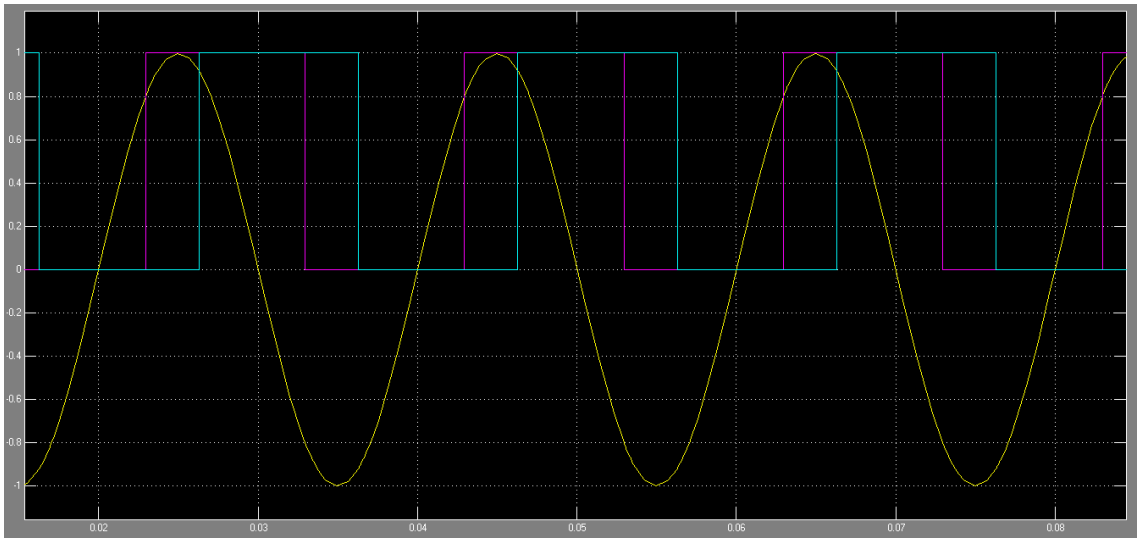


Figure 32 Control Inverter Simulink S_6.

2.3.7 S_7 - Full Wave H-Bridge Inverter (Bipolar PWM Control)

Main block diagram:

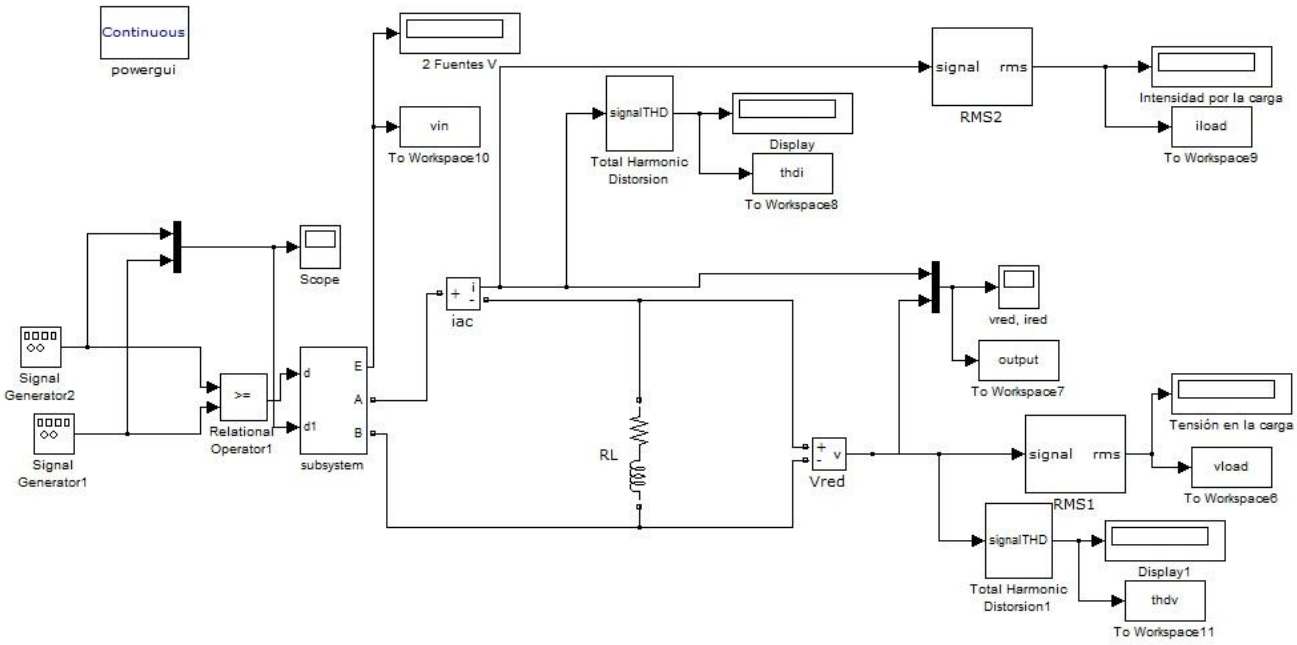


Figure 33 Inverter Scheme Simulink S_7.

SubSystem:

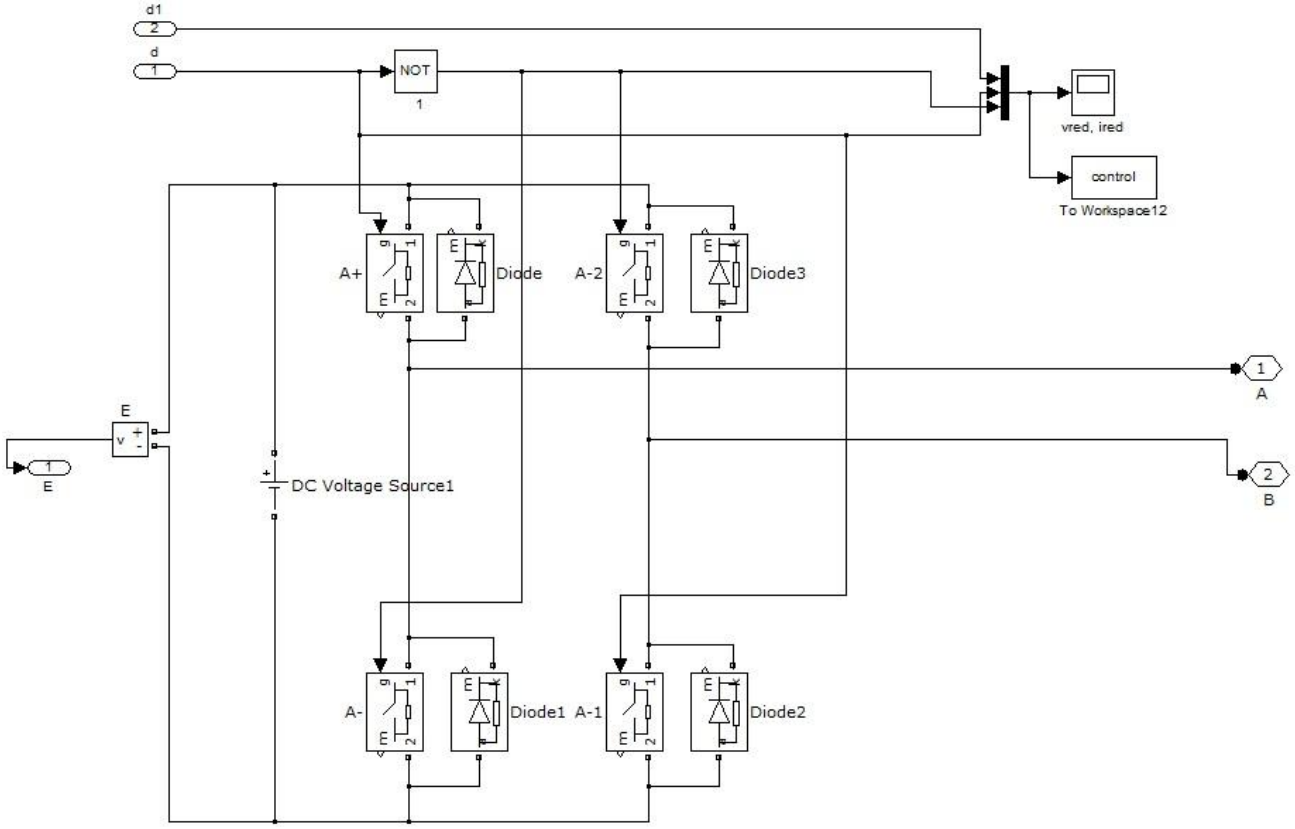


Figure 34 SubSystem Inverter Scheme Simulink S_7.

Graph of the waveform at the output load. The yellow line represents the Current that runs through load and purple line represents the Voltage in it:

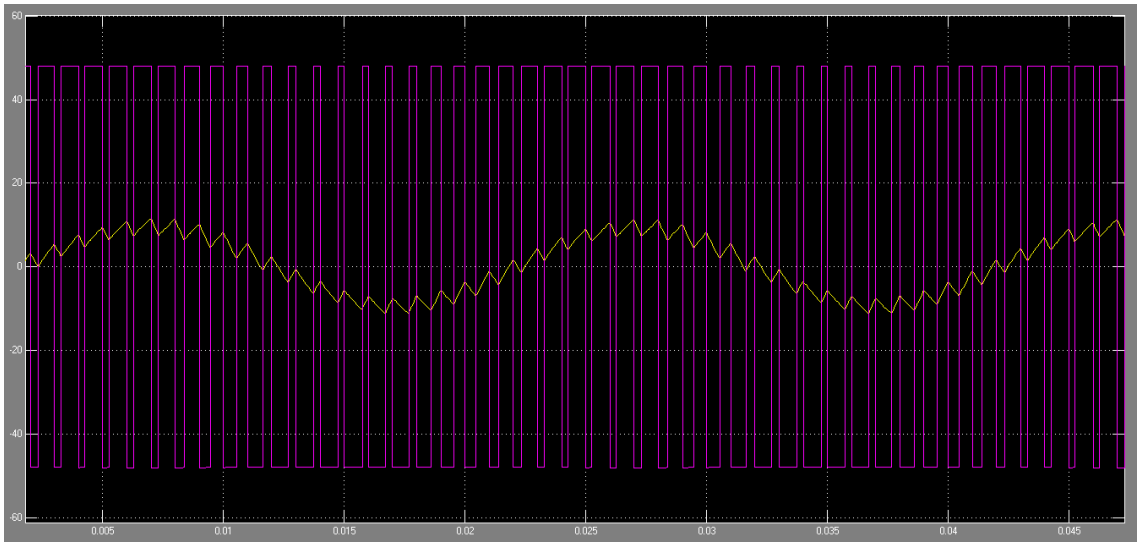


Figure 35 Output Load Inverter Simulink S_7.

Graph of the waveform of the inverter control (PWM Control):

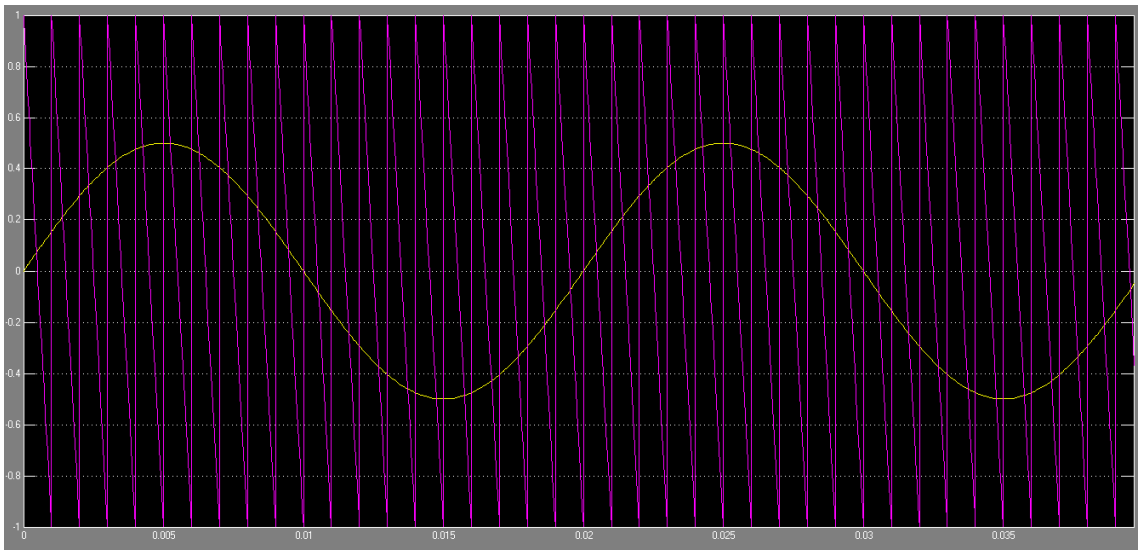


Figure 36 Control Inverter Simulink S_7.

2.3.8 S_8 - Full Wave H-Bridge Inverter (Single-Pole PWM Control)

Main block diagram:

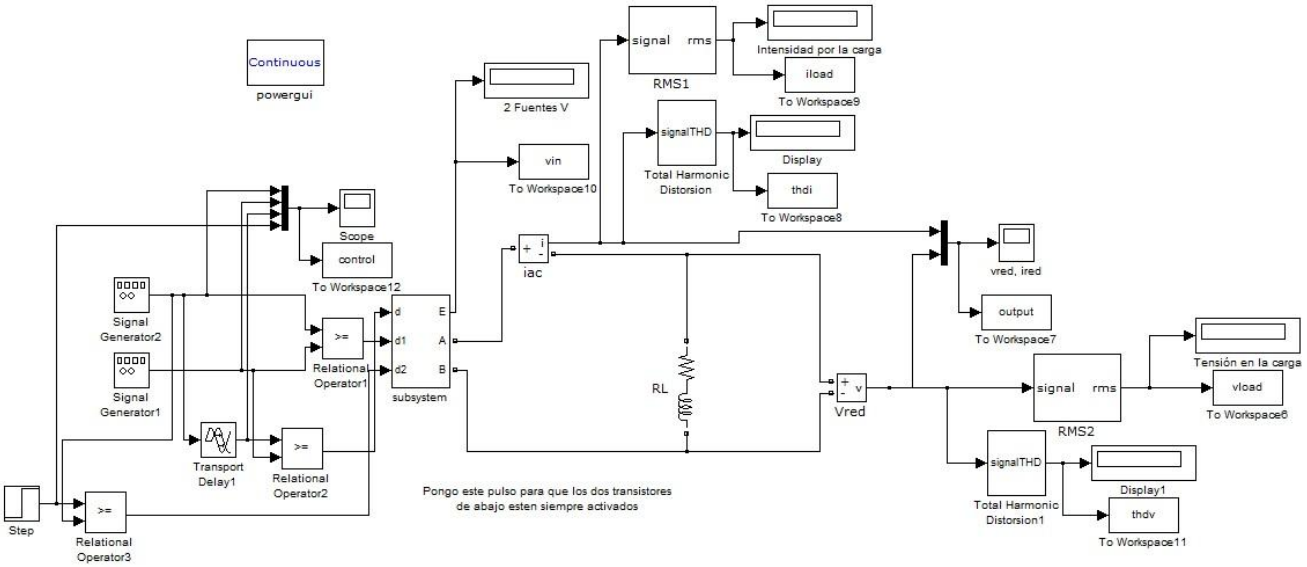


Figure 37 Inverter Scheme Simulink S_8.

SubSystem:

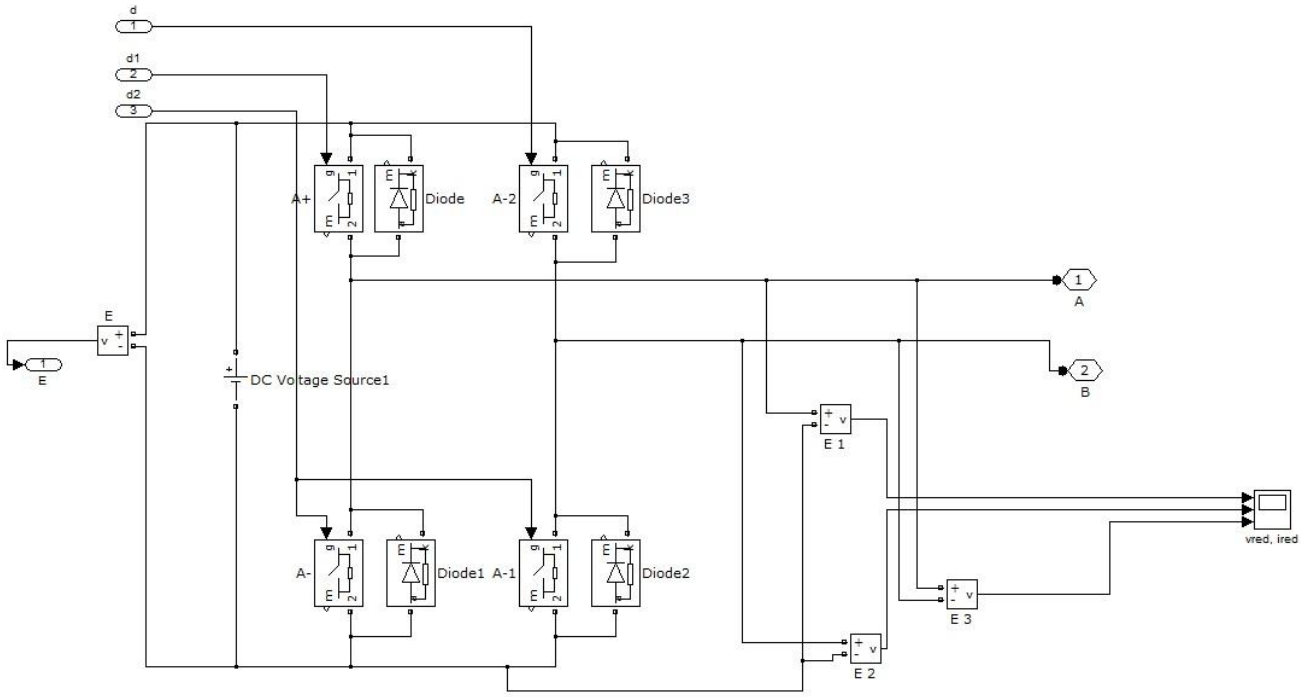


Figure 38 SubSystem Inverter Scheme Simulink S_8.

Graph of the waveform at the output load. The yellow line represents the Current that runs through load and purple line represents the Voltage in it:

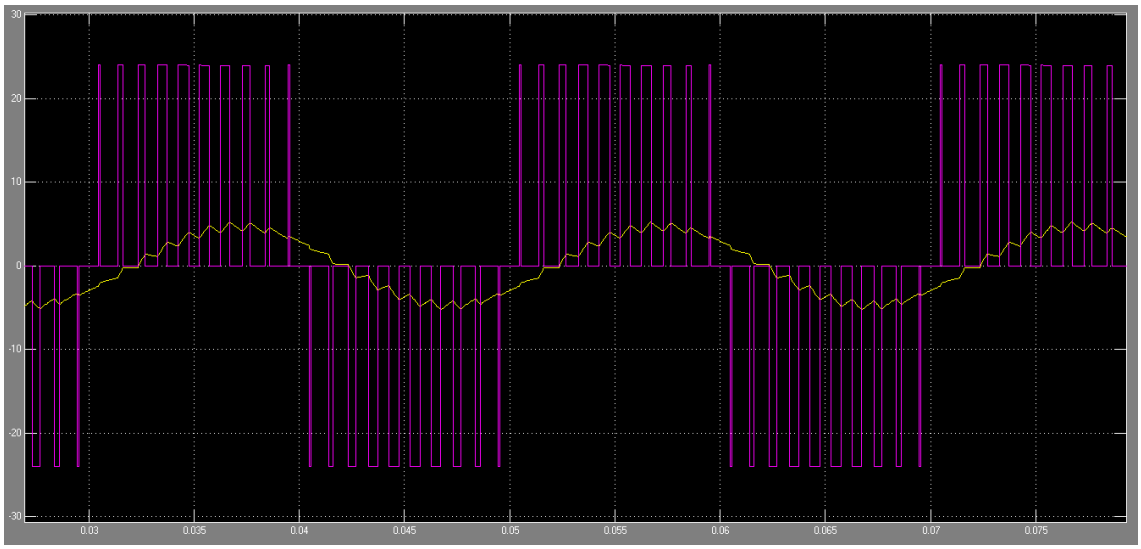


Figure 39 Output Load Inverter Simulink S_8.

Graph of the waveform of the inverter control (Three-Phase PWM Control):

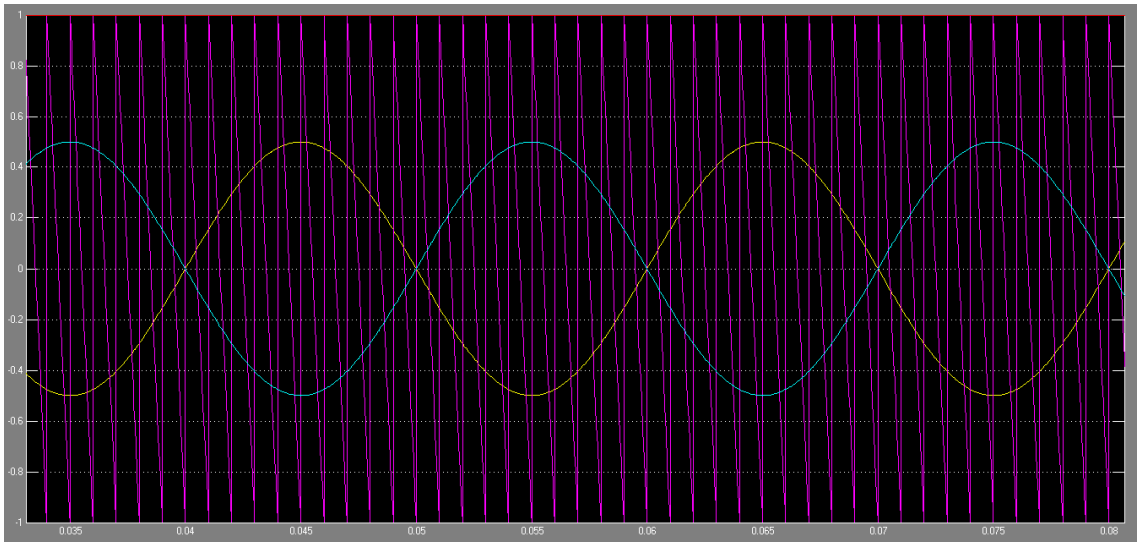


Figure 40 Control Inverter Simulink S_8.

2.3.9 S_9 - Three-Phase VSI 6-Step Inverter (Star Load) (Square Wave Control)

Main block diagram:

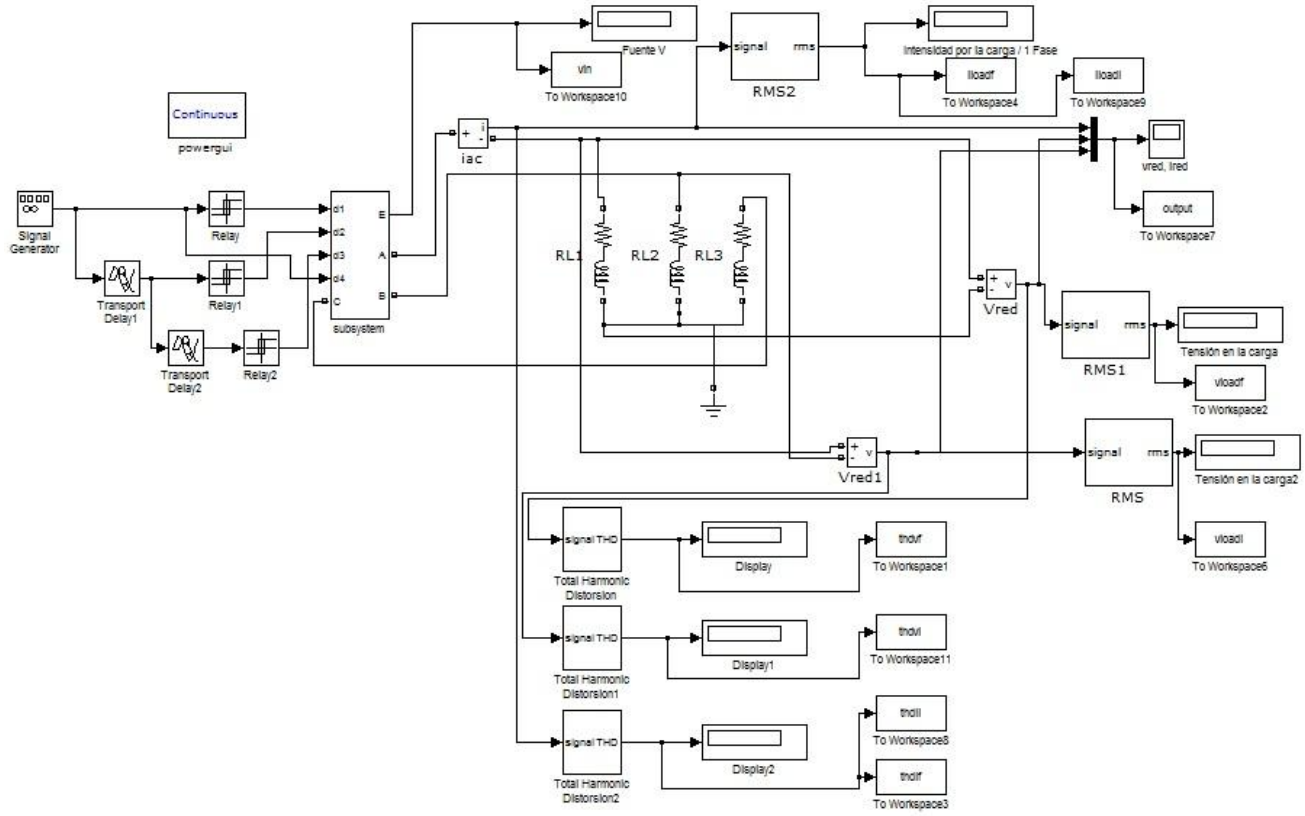


Figure 41 Inverter Scheme Simulink S_9.

SubSystem:

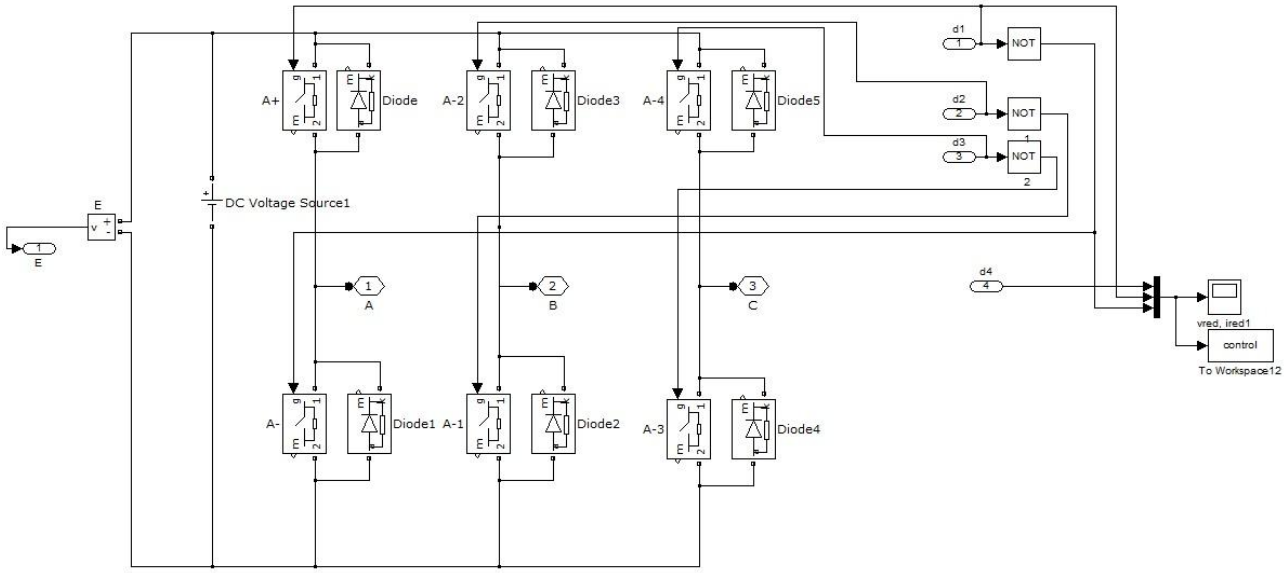


Figure 42 SubSystem Inverter Scheme Simulink S_9.

Graph of the waveform at the output load. The yellow line represents the Current that runs through load, purple line represents the Voltage (line) and blue line is the Voltage (phase) in it:

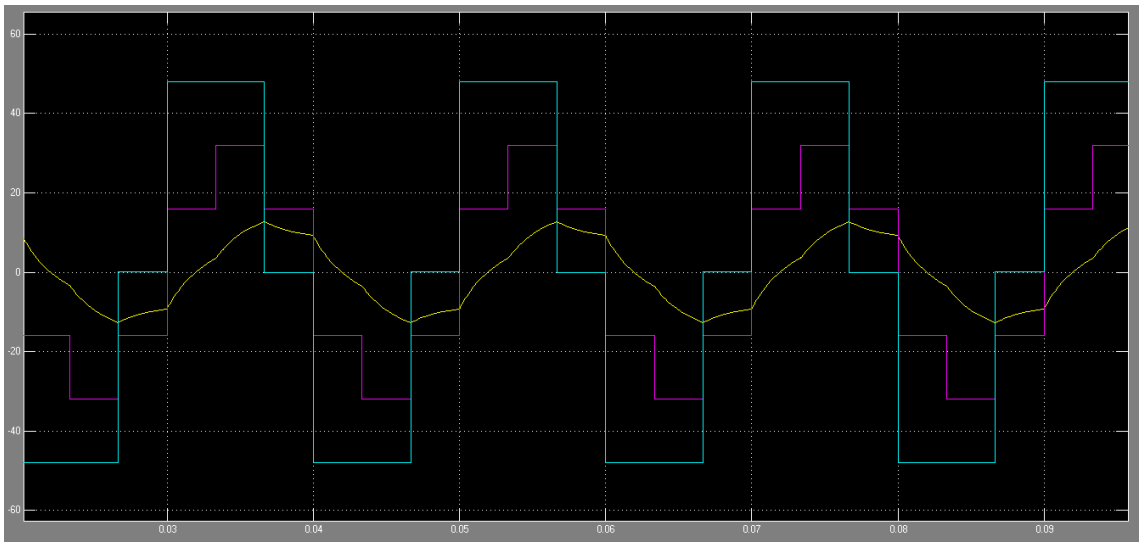


Figure 43 Output Load Inverter Simulink S_9.

Graph of the waveform of the inverter control (Square Wave Control):

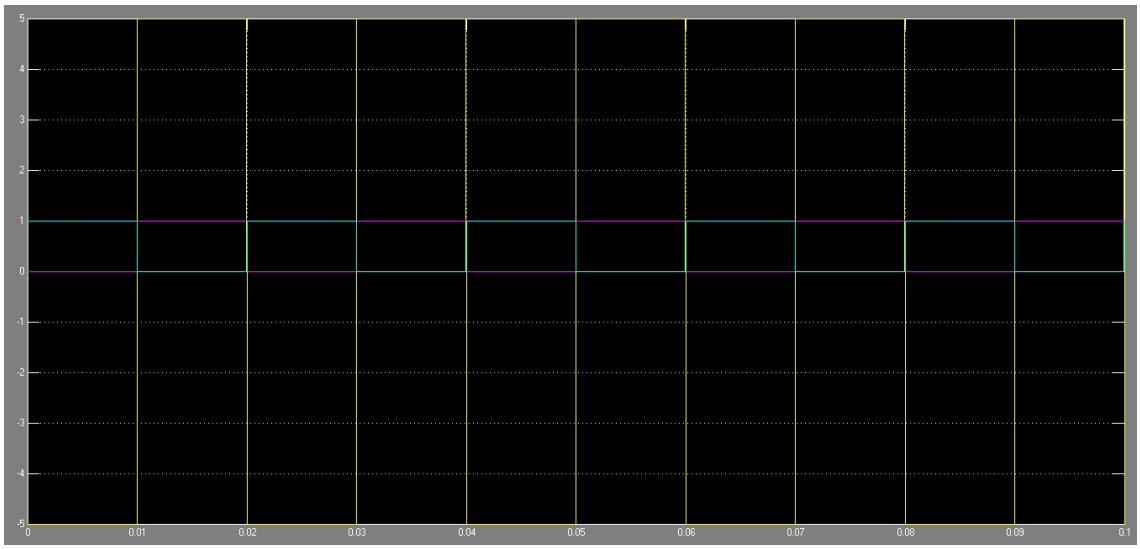


Figure 44 Control Inverter Simulink S_9.

2.3.10 S_10 - Three-Phase VSI 6-Step Inverter (Triangle Load) (Square Wave Control)

Main block diagram:

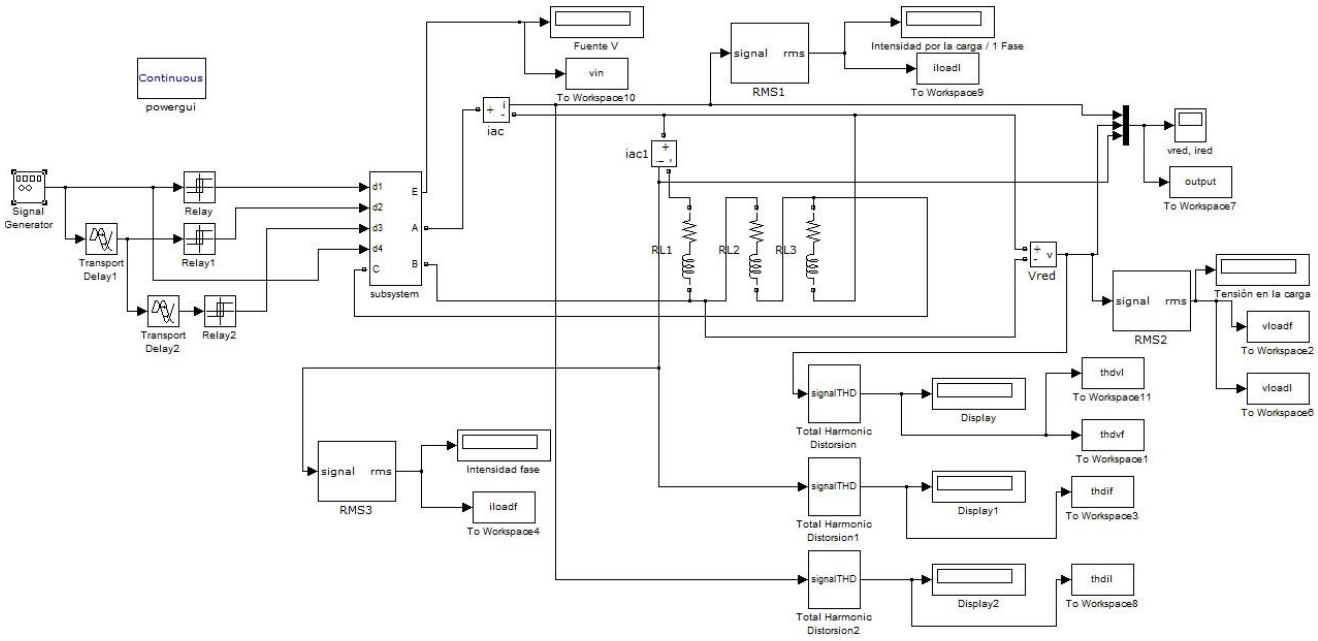


Figure 45 Inverter Scheme Simulink S_10.

SubSystem:

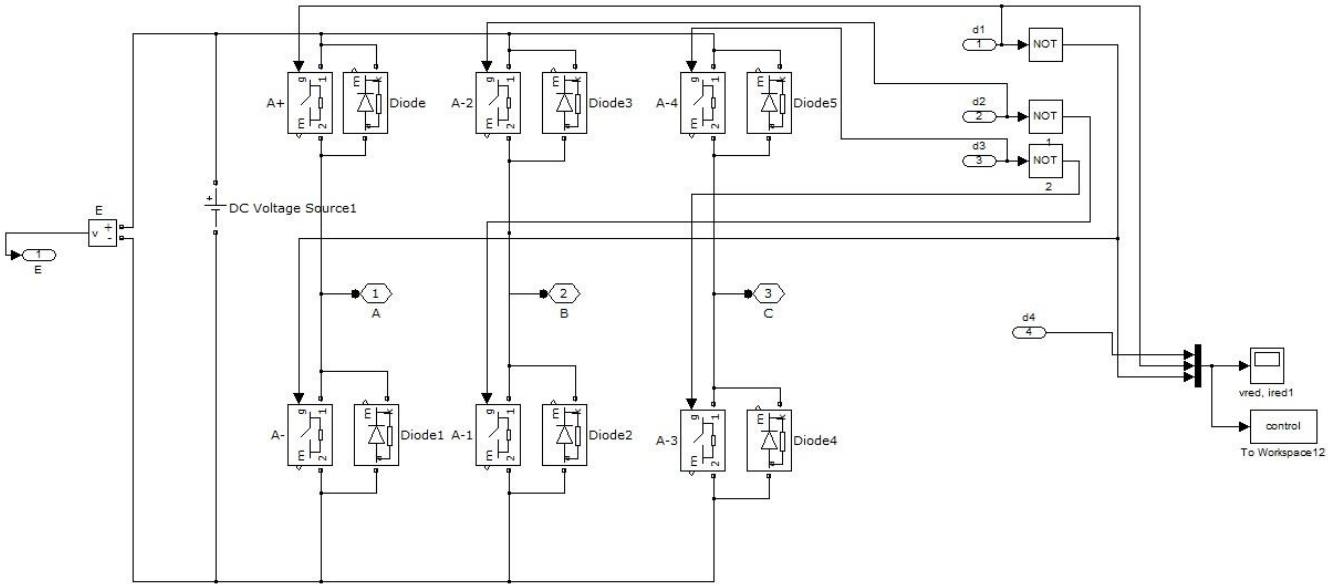


Figure 46 SubSystem Inverter Scheme Simulink S_10.

Graph of the waveform at the output load. The yellow line represents the Current (line) that runs through load and purple line represents the Voltage (Line and phase) and blue line is the Current (phase) that runs through load:

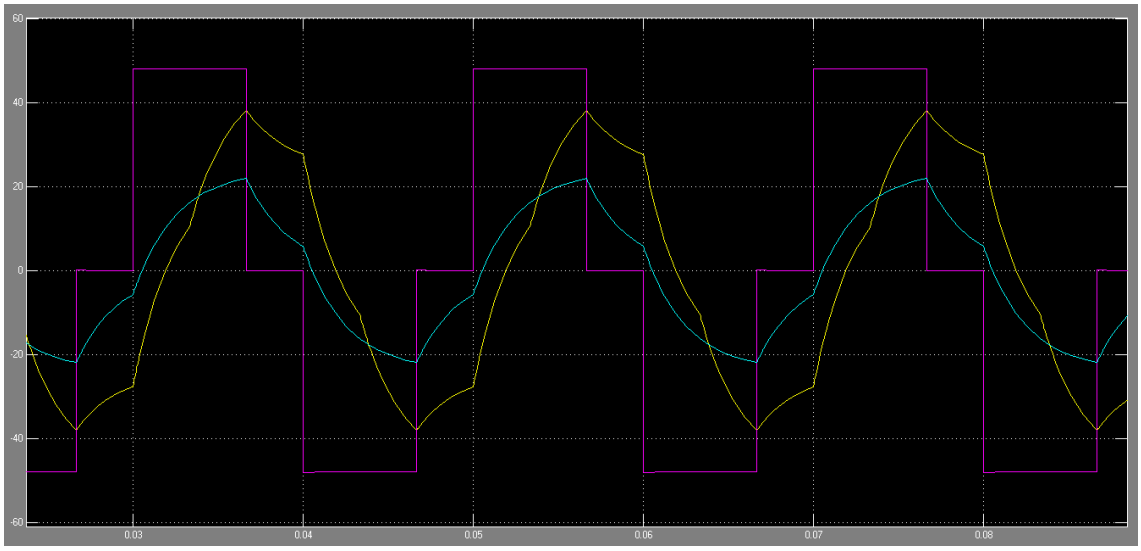


Figure 47 Output Load Inverter Simulink S_10.

Graph of the waveform of the inverter control (Square Wave Control):

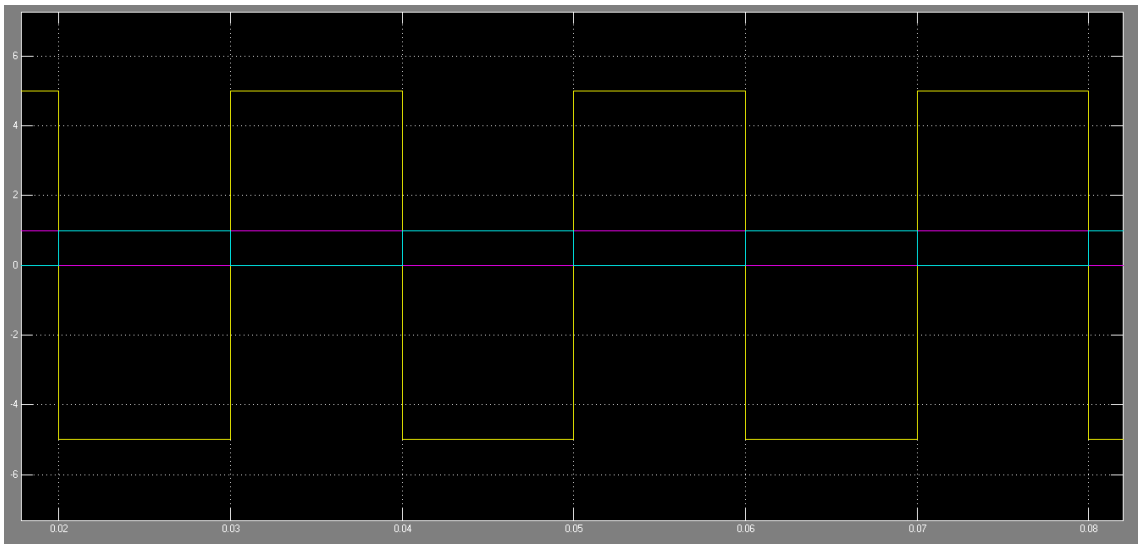


Figure 48 Control Inverter Simulink S_10.

2.3.11 S_11 -Three-Phase VSI 6-Step Inverter (Star Load) (PWM Control)

Main block diagram:

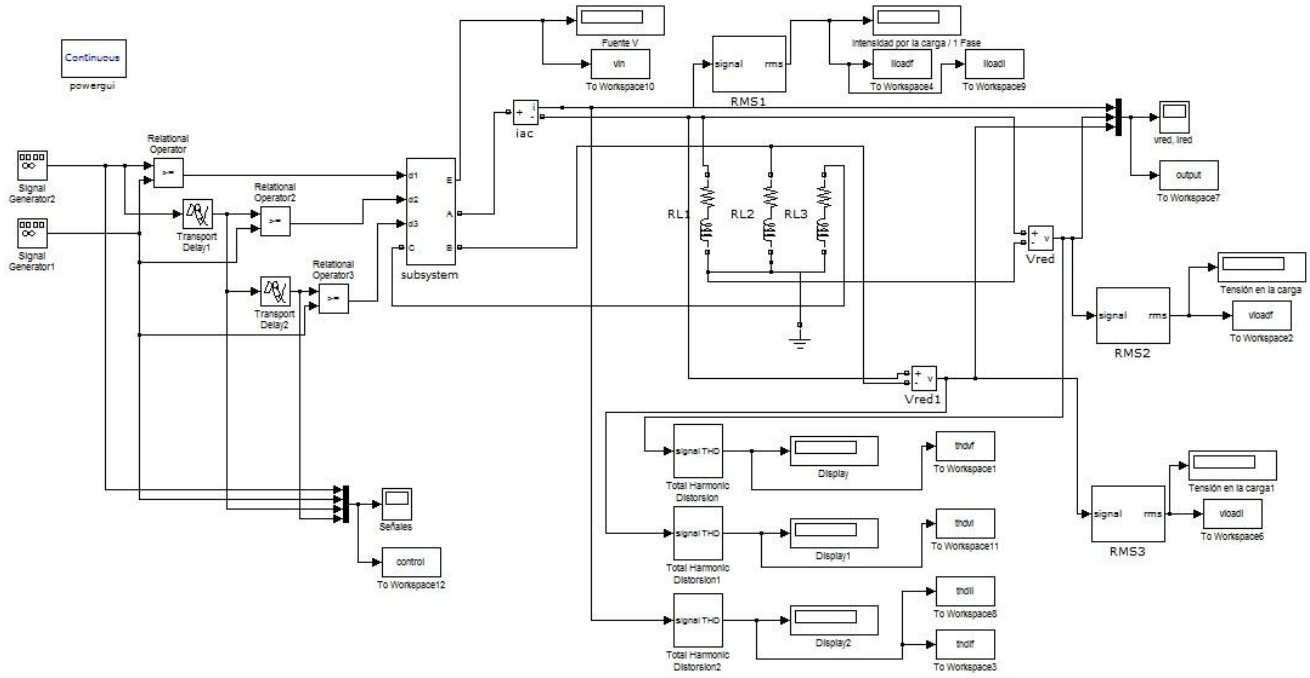


Figure 49 Inverter Scheme Simulink S_11.

SubSystem:

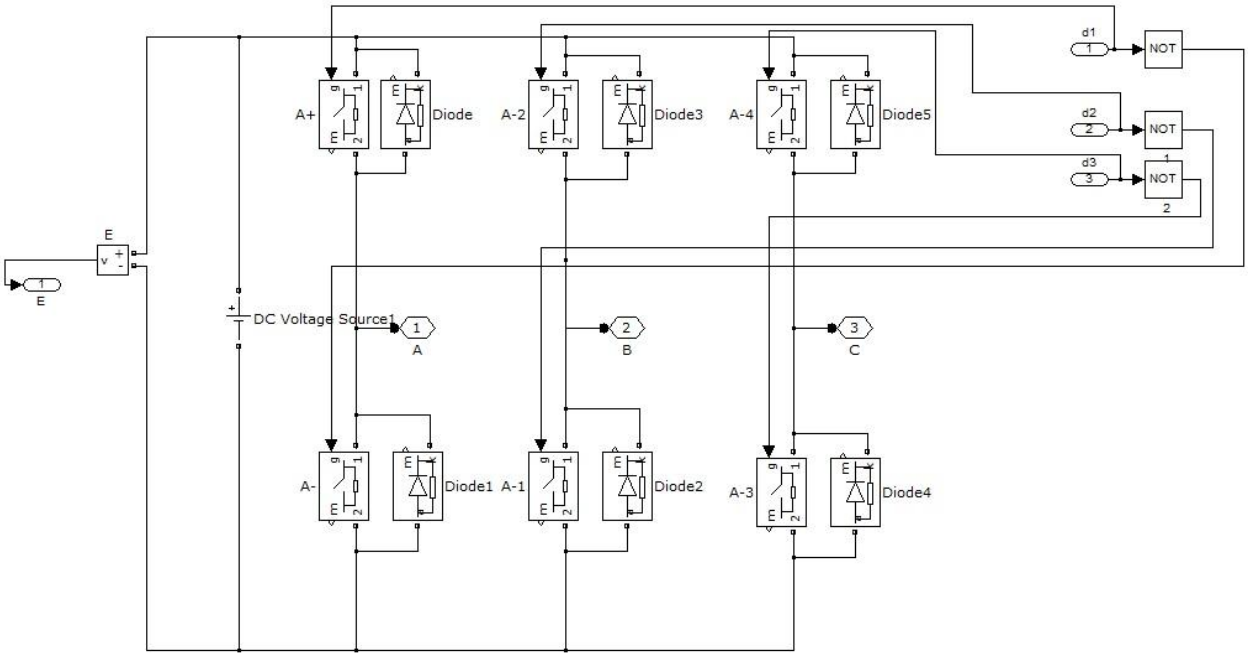


Figure 50 SubSystem Inverter Simulink Scheme S_11.

Graph of the waveform at the output load. The yellow line represents the intensity that runs through load and purple line represents the voltage in it:

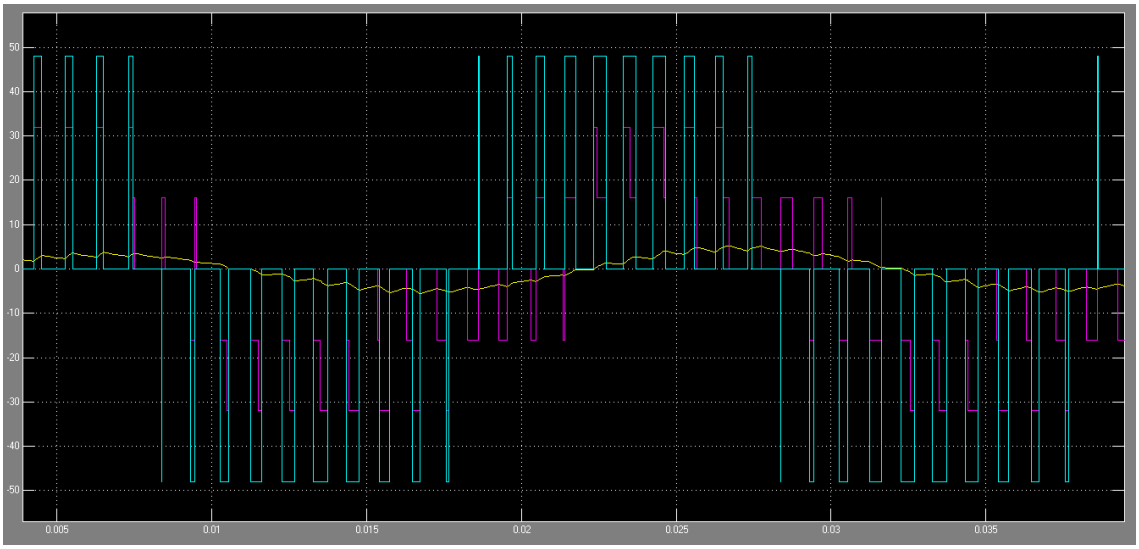


Figure 51 Output Load Inverter Simulink S_11.

Graph of the waveform of the inverter control:

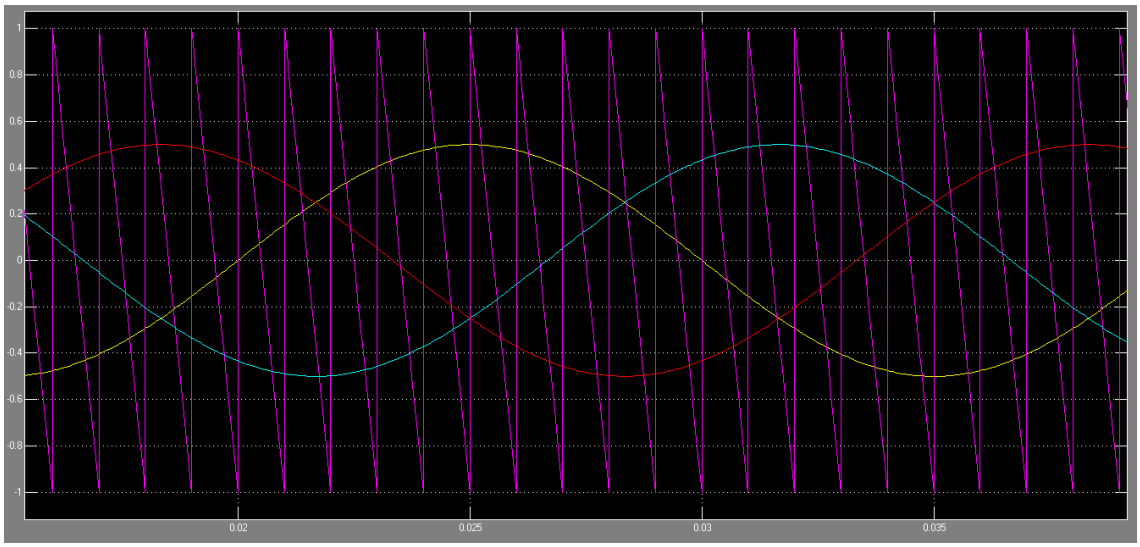


Figure 52 Control Inverter Simulink S_11.

2.3.12 S_12 - Three-Phase VSI 6-Step Inverter (Triangle Load) (PWM Control)

Main block diagram:

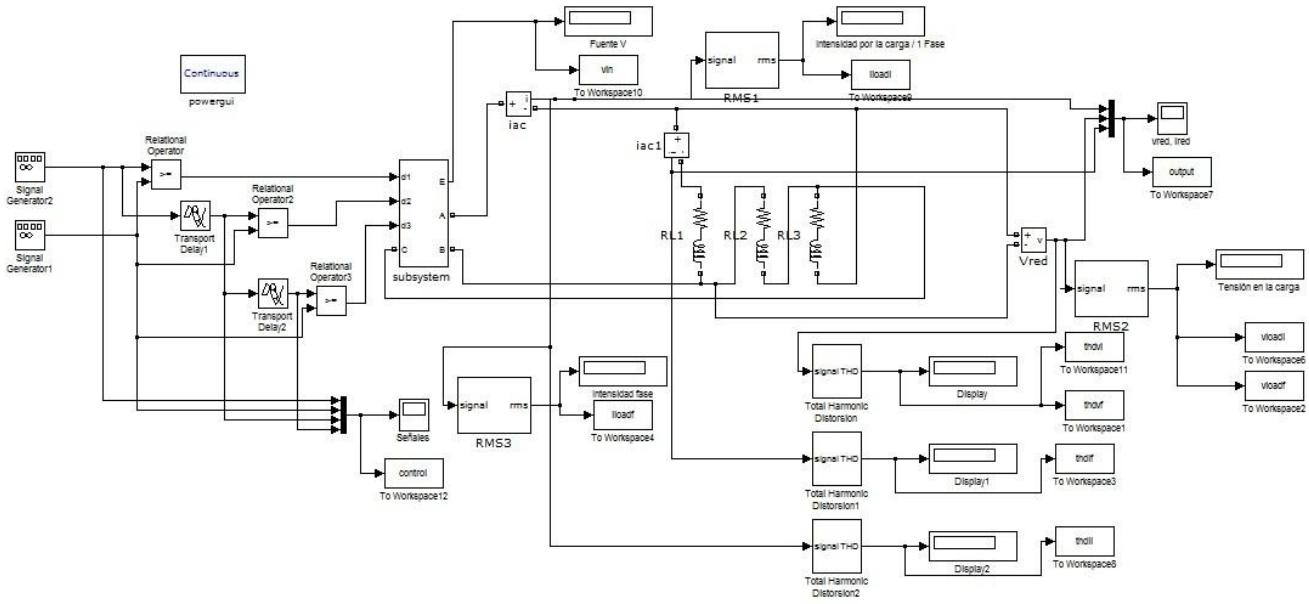


Figure 53 Inverter Scheme Simulink S_12.

SubSystem:

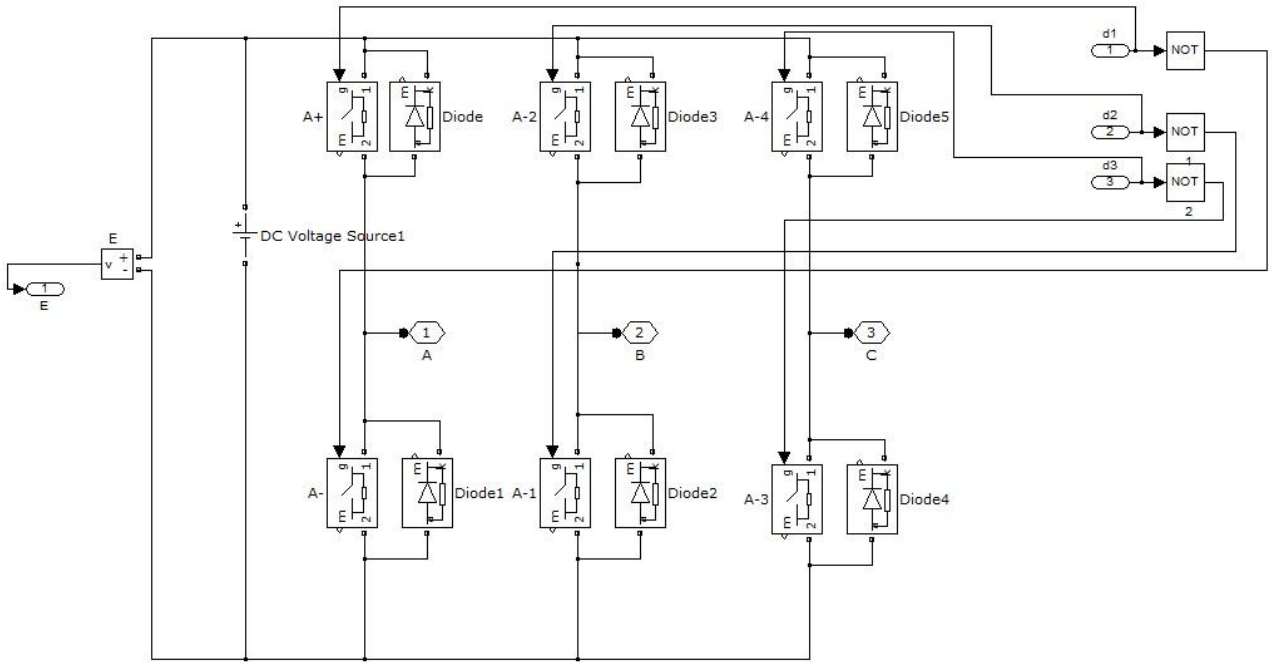


Figure 54 SubSystem Inverter Scheme Simulink S_12.

Graph of the waveform at the output load. The yellow line represents the Current (line) that runs through load and purple line represents the Voltage (Line and phase) and blue line is the Current (phase) that runs through load:

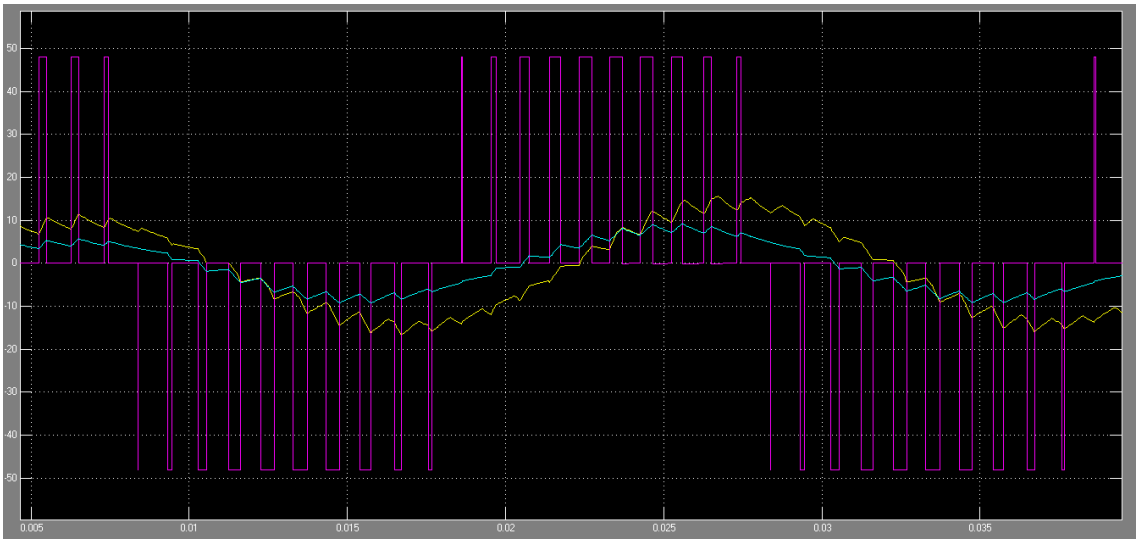


Figure 55 Output Load Inverter Simulink S_12.

Graph of the waveform of the inverter control (Three-Phase PWM Control):

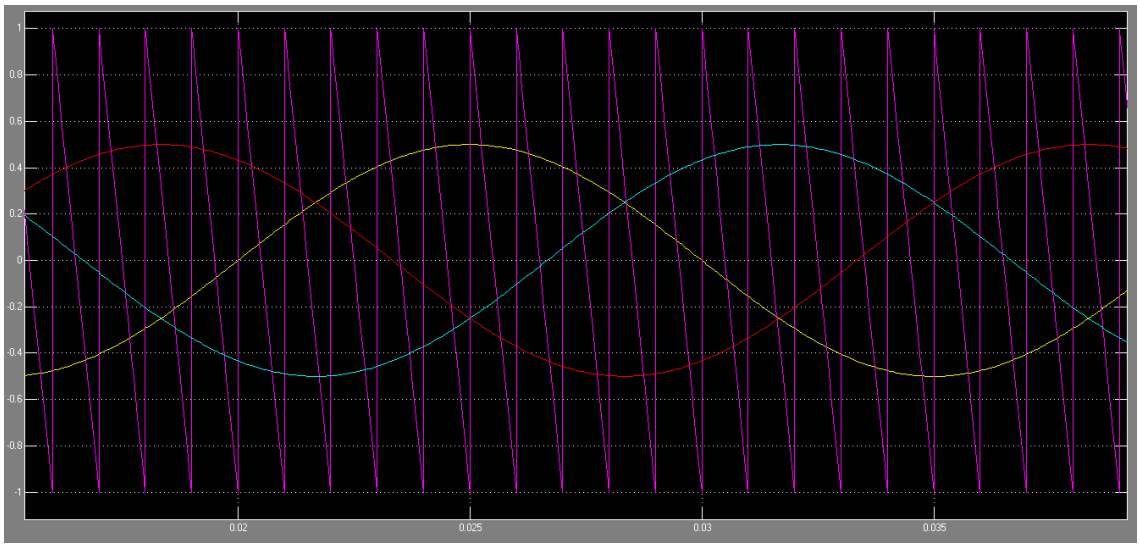


Figure 56 Control Inverter Simulink S_12.

2.3.13 S_13 - Three-Phase VSI 6-Step Inverter (Star Load) (Hysteresis Control)

Main block diagram:

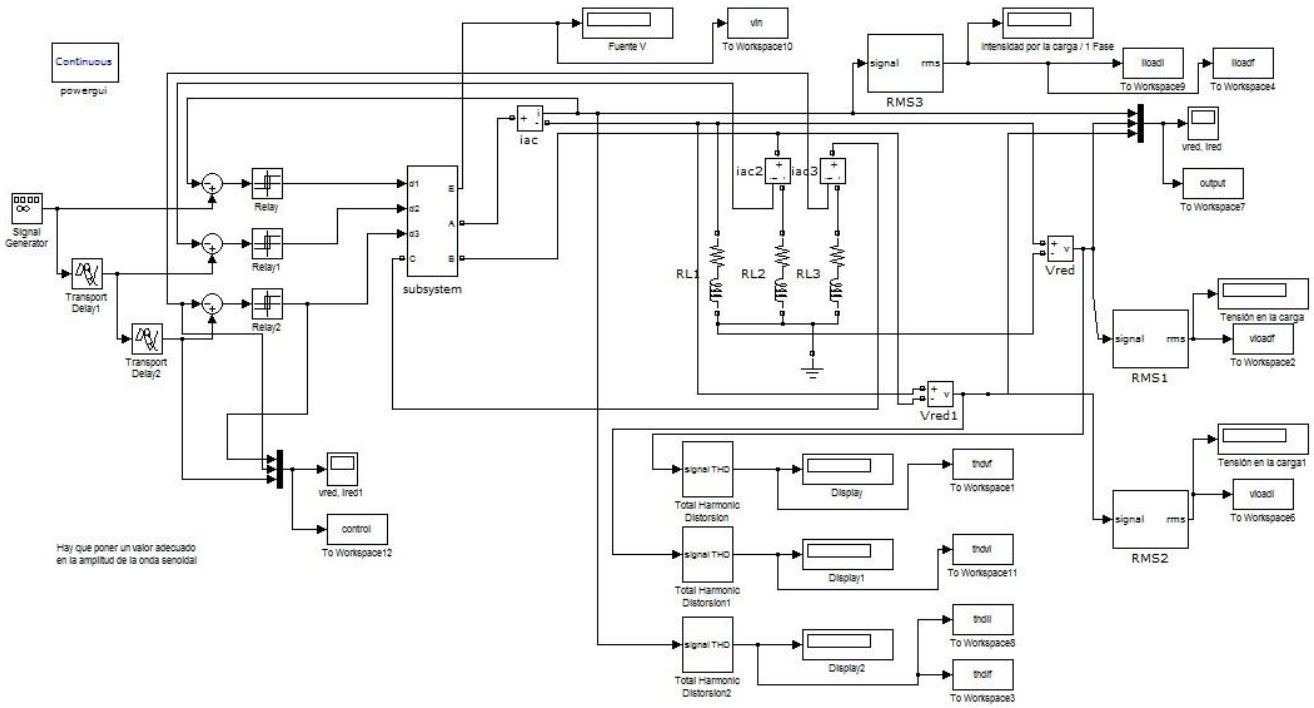


Figure 57 Inverter Scheme Simulink S_13.

SubSystem:

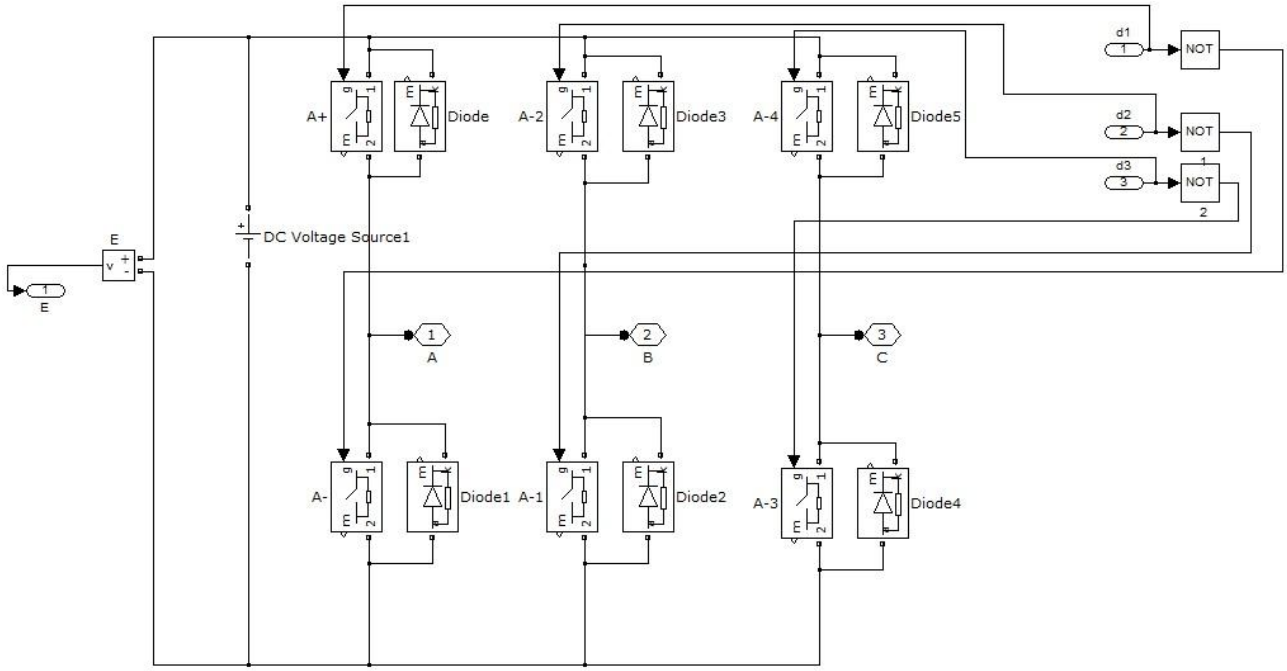


Figure 58 SubSystem Inverter Scheme Simulink S_13.

Graph of the waveform at the output load. The yellow line represents the Current that runs through load, purple line represents the Voltage (line) and blue line is the Voltage (phase) in it:

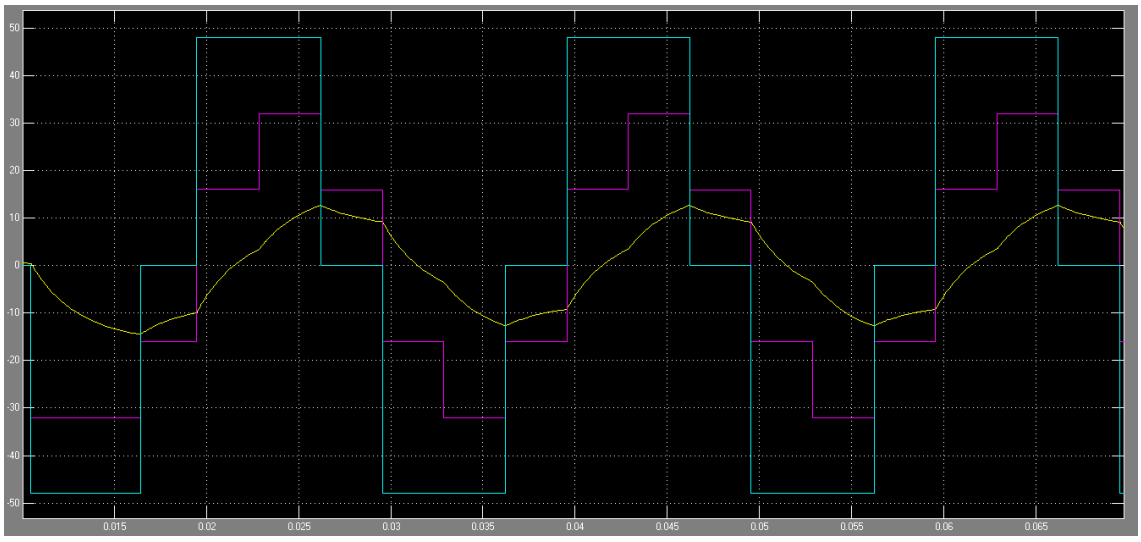


Figure 59 Output Load Inverter Simulink S_13.

Graph of the waveform of the inverter control (Hysteresis Control). This kind of control is developed in the theory of inverters:

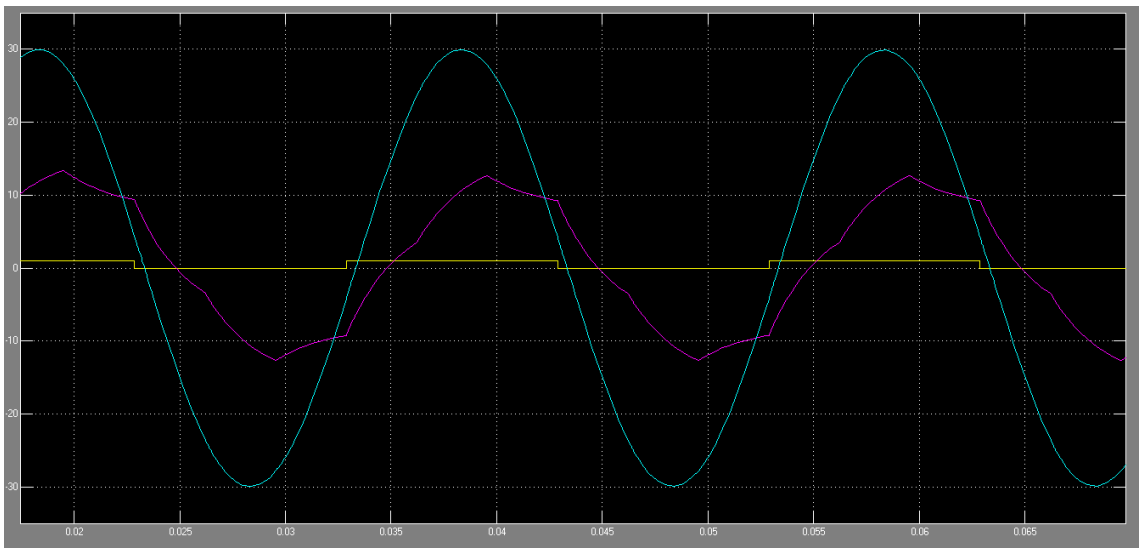


Figure 60 Control Inverter Simulink S_13.

2.3.14 S_14 - Push-Pull Inverter (Square Wave Control)

Main block diagram:

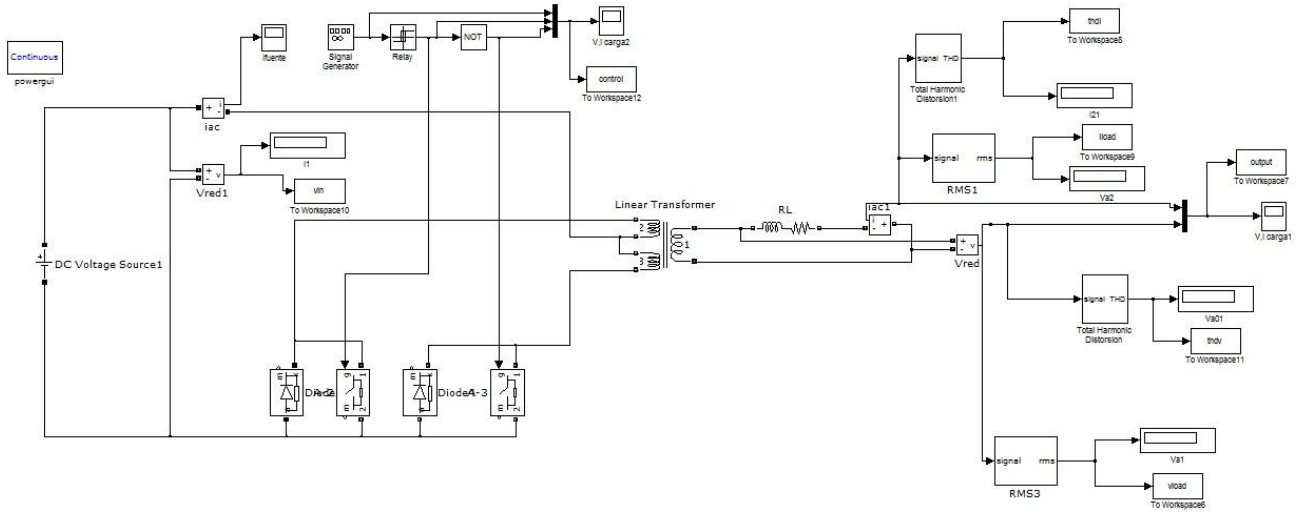


Figure 61 Inverter Scheme Simulink S_14.

Graph of the waveform at the output load. The yellow line represents the Current that runs through load and purple line represents the Voltage in it:

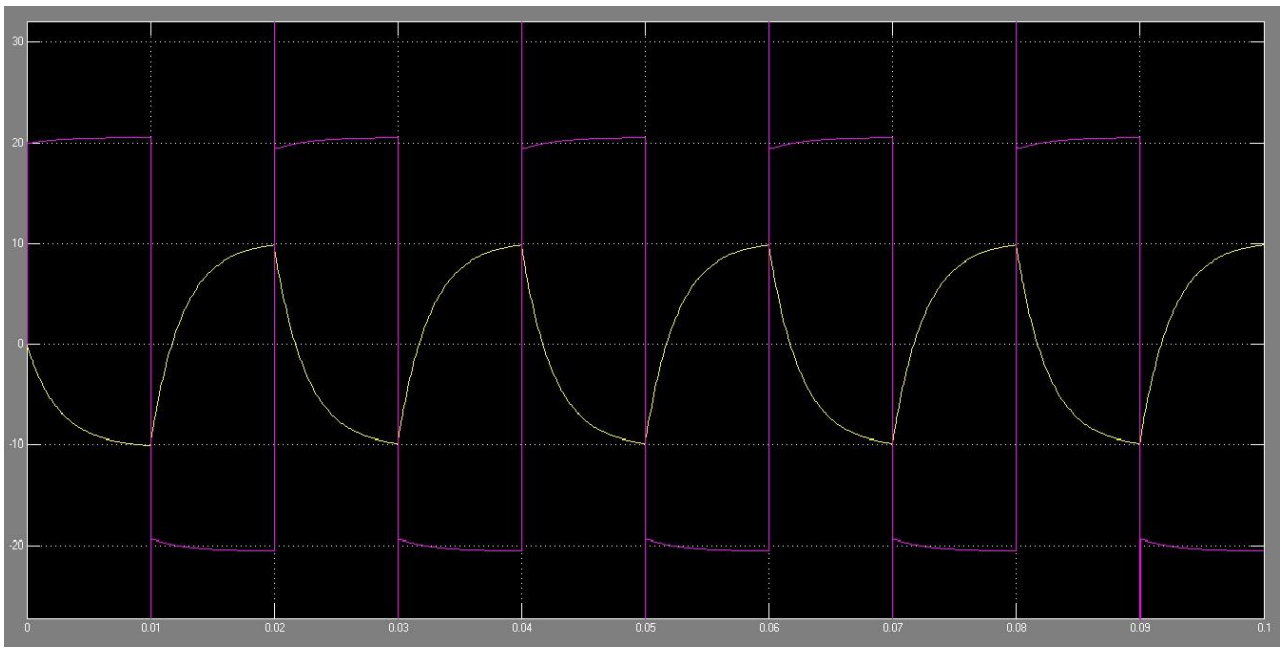


Figure 62 Output Load Inverter Simulink S_14.

Graph of the waveform of the inverter control (Square Wave Control):

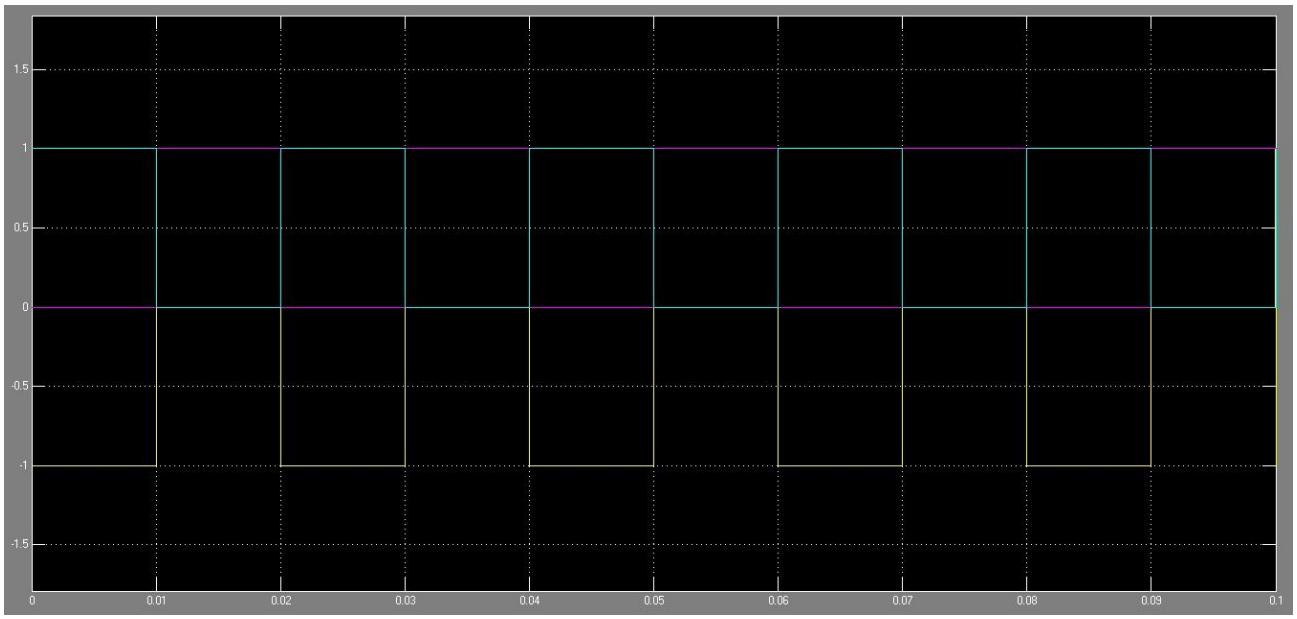


Figure 63 Control Inverter Simulink S_14.

2.3.15 S_15 - Push-Pull Inverter (PWM Control)

Main block diagram:

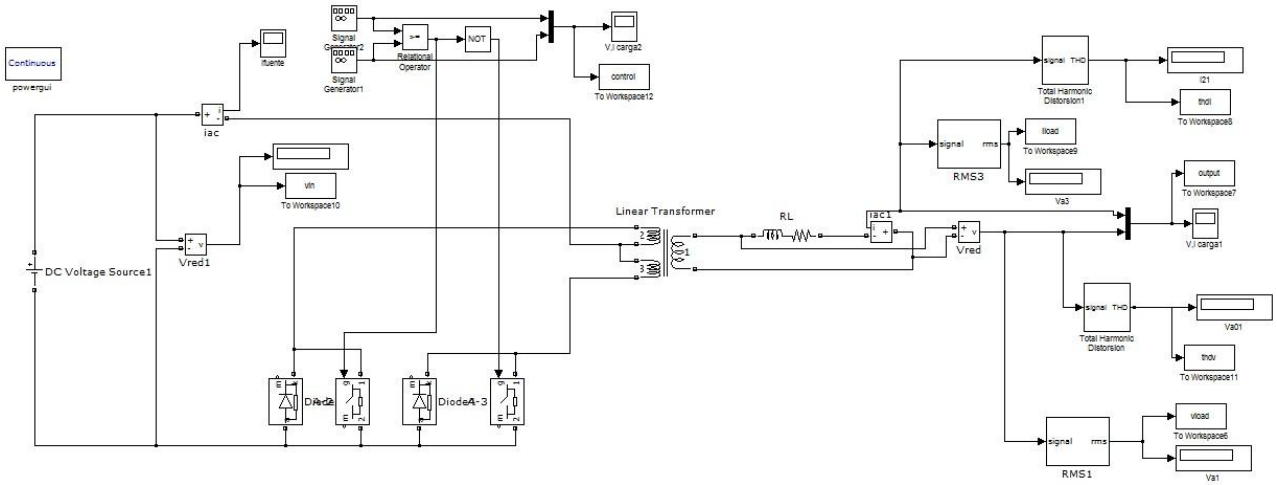


Figure 64 Inverter Scheme Simulink S_15.

Graph of the waveform at the output load. The yellow line represents the Current that runs through load and purple line represents the Voltage in it:

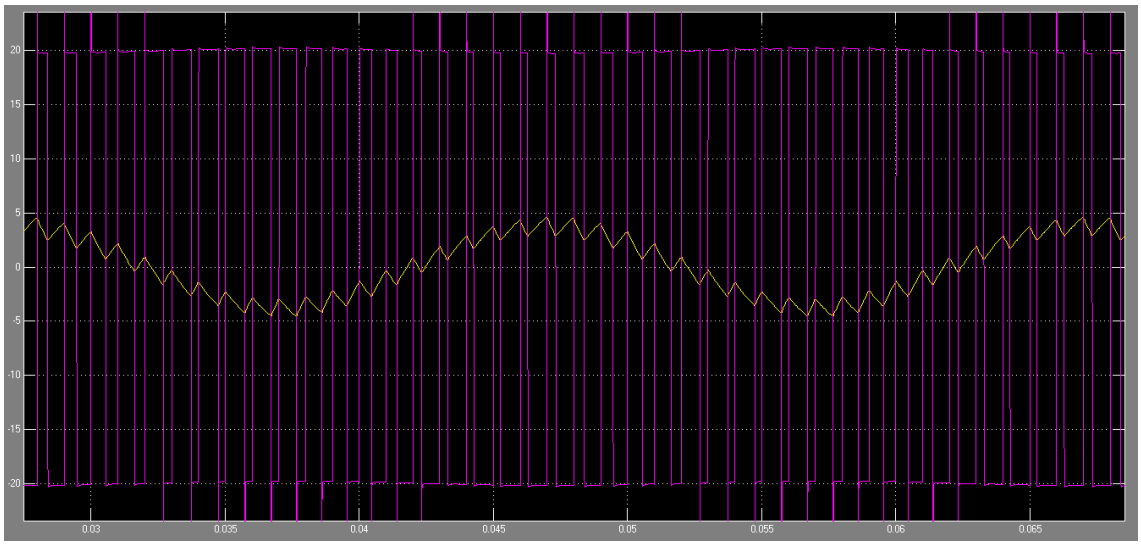


Figure 65 Output Load Inverter Simulink S_15.

Graph of the waveform of the inverter control (PWM Control):

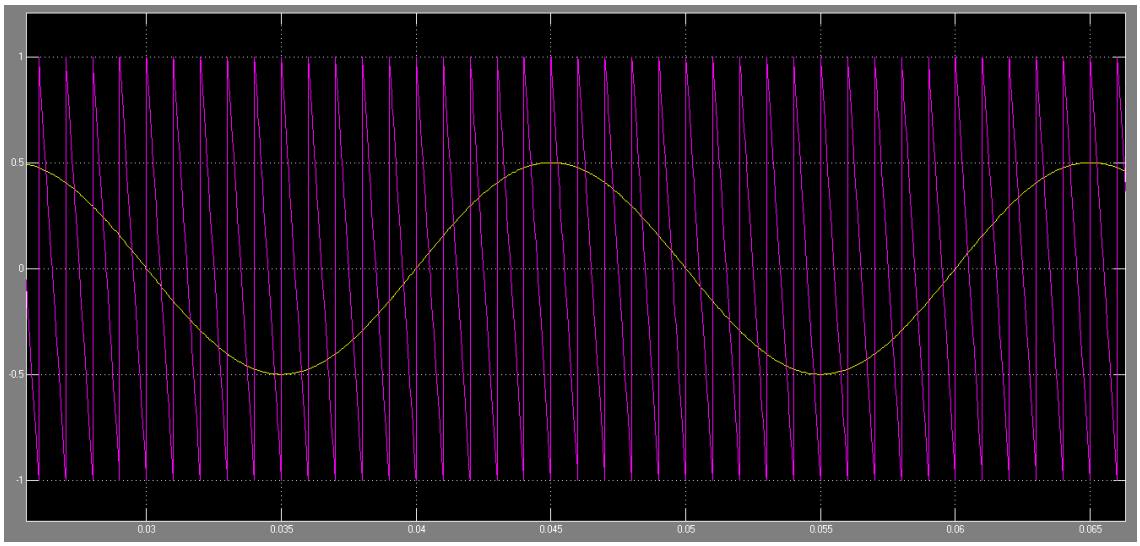


Figure 66 Control Inverter Simulink S_15.

2.3.16 S_16 - Cascade Full Bridge Single-Phase Inverter (2 Levels)

Main block diagram:

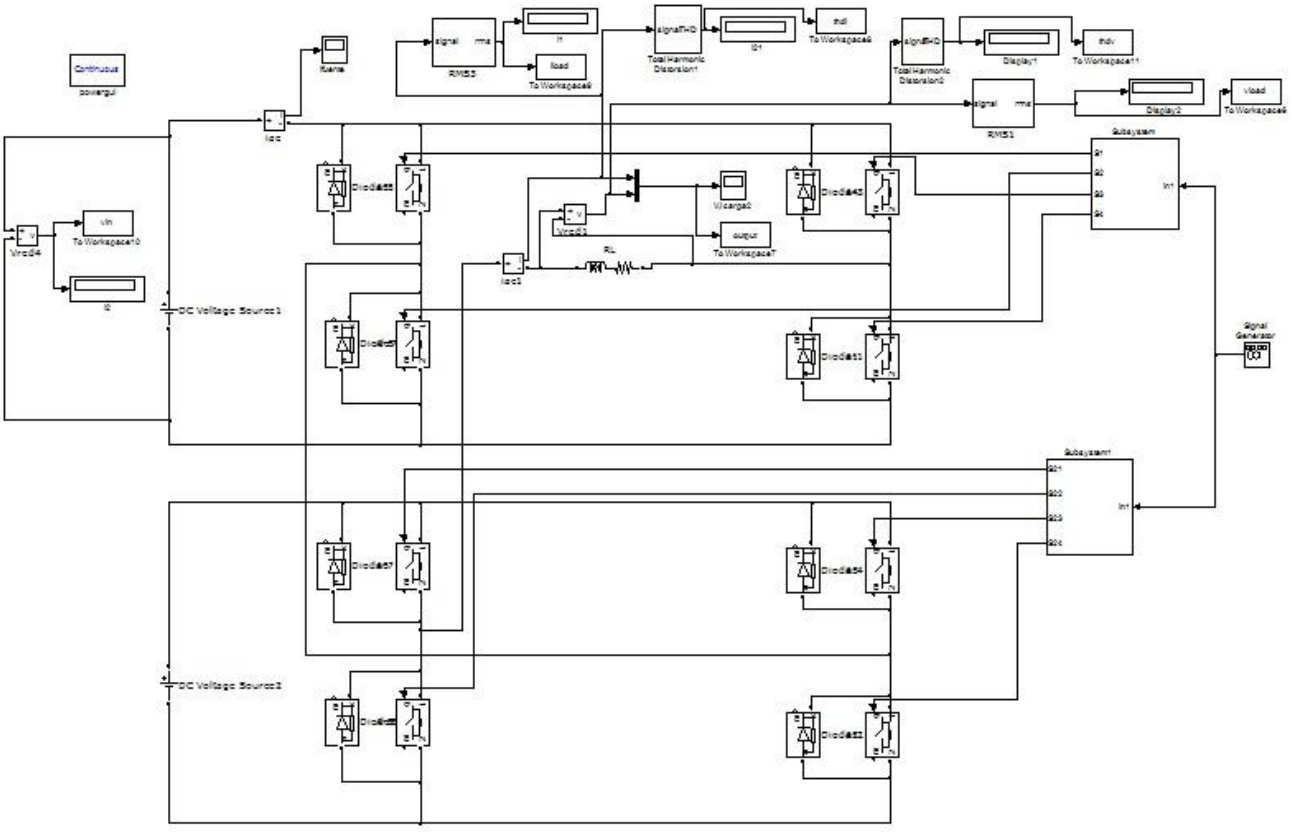


Figure 67 Inverter Scheme Simulink S_16.

SubSystem:

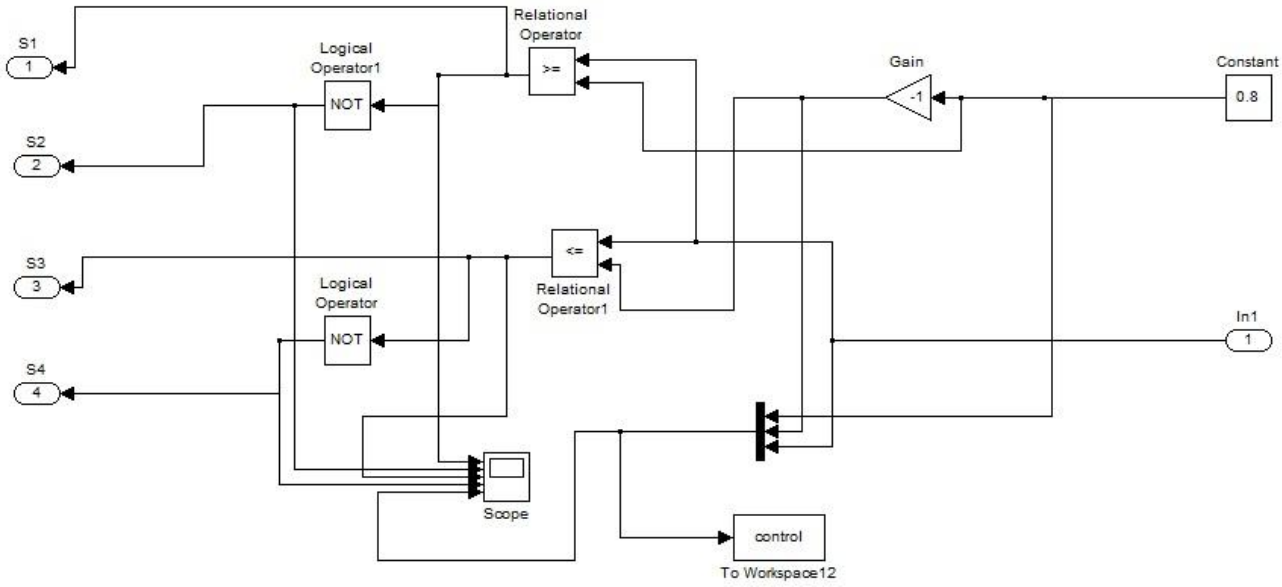


Figure 68 SubSystem Inverter Scheme Simulink S_16.

Graph of the waveform at the output load. The yellow line represents the Current that runs through load and purple line represents the Voltage in it:

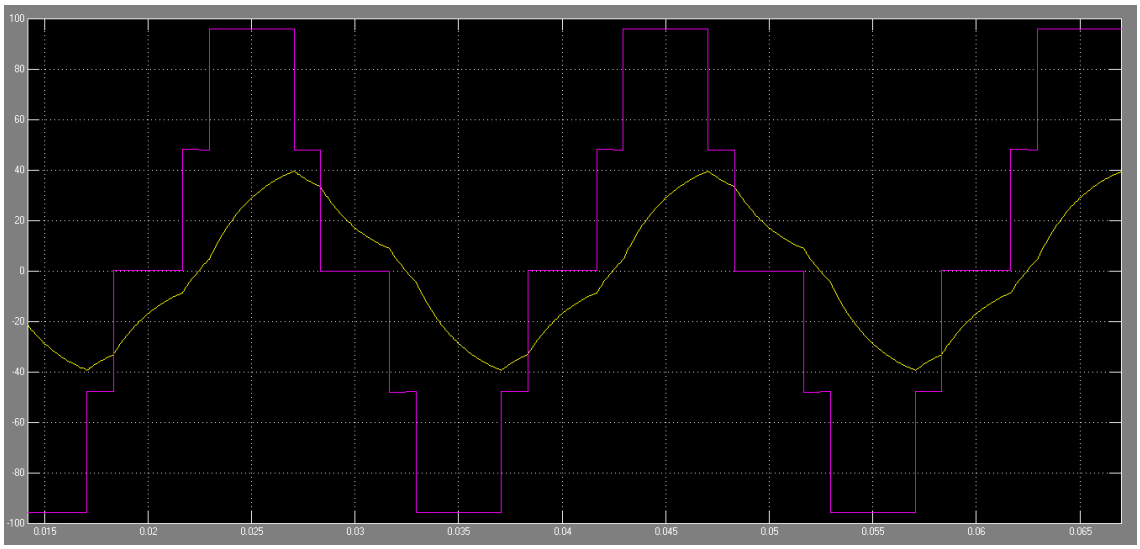


Figure 69 Output Load Inverter Simulink S_16.

Graph of the waveform of the inverter control. It can see when each transistor is open or closed:

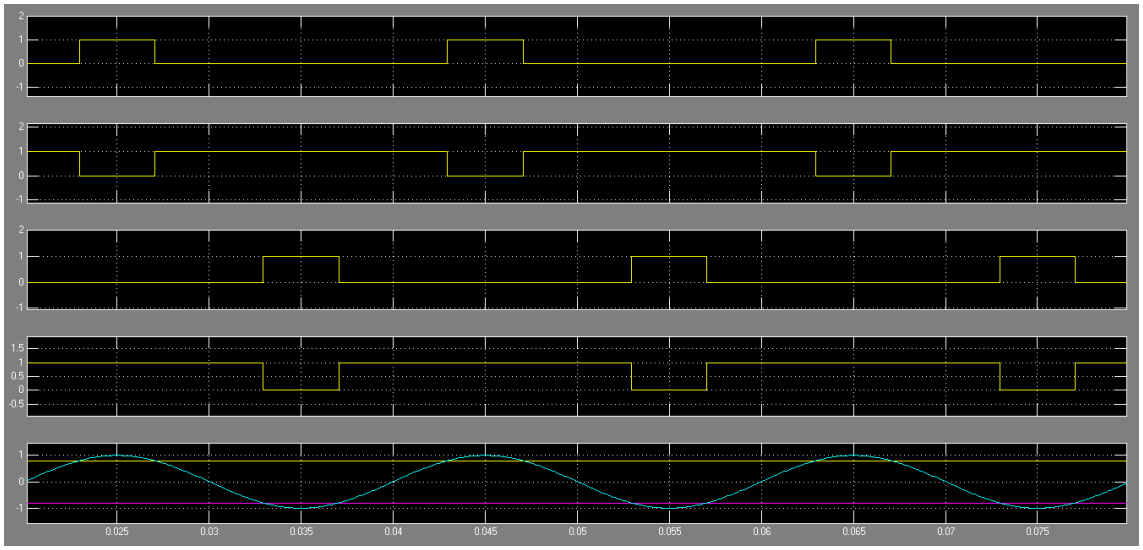


Figure 70 Control Inverter Simulink S_16.

2.3.17 S_17 - Cascade Full Bridge Single-Phase Inverter (4 Levels)

Main block diagram:

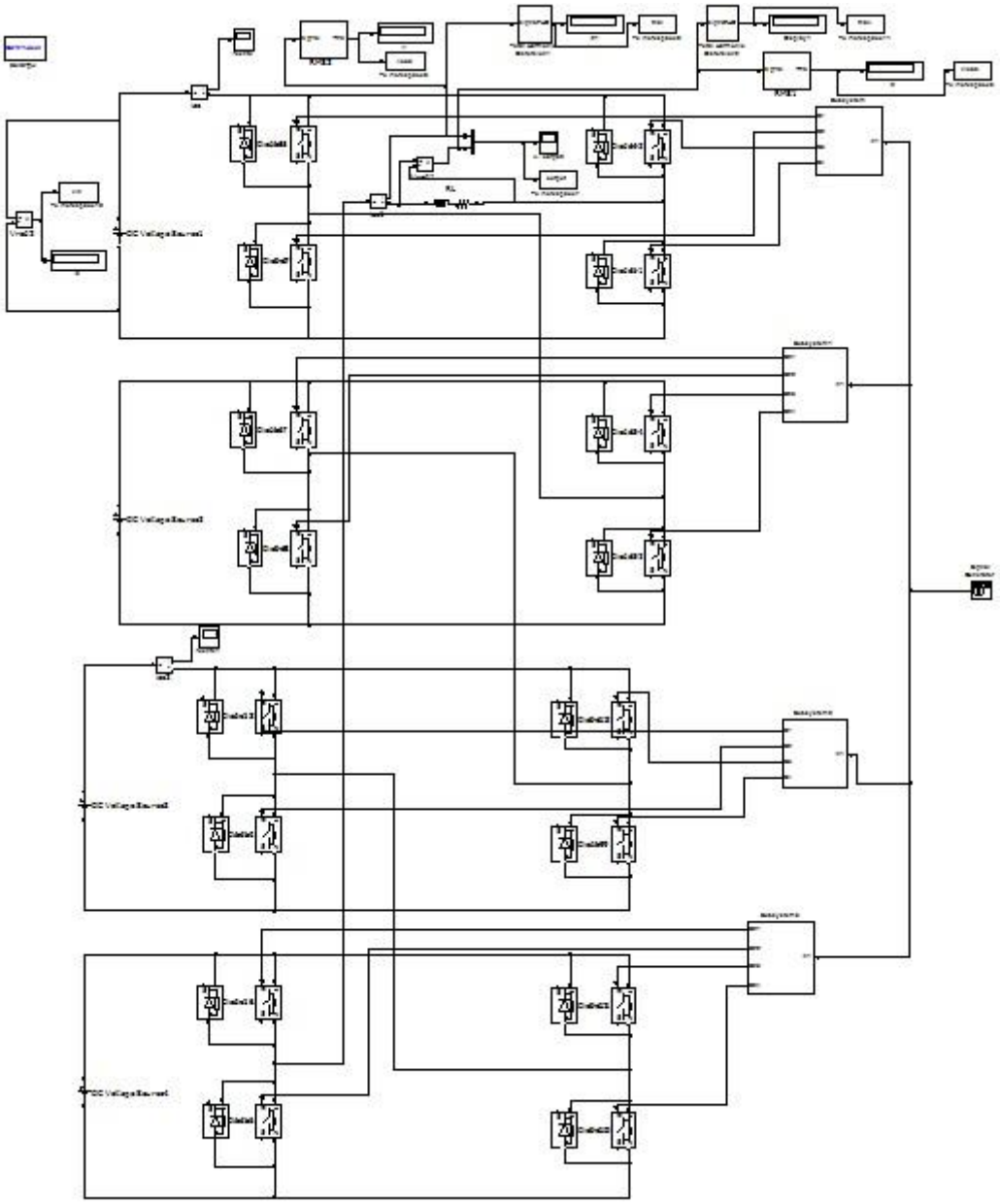


Figure 71 Inverter Scheme Simulink S_17.

SubSystem:

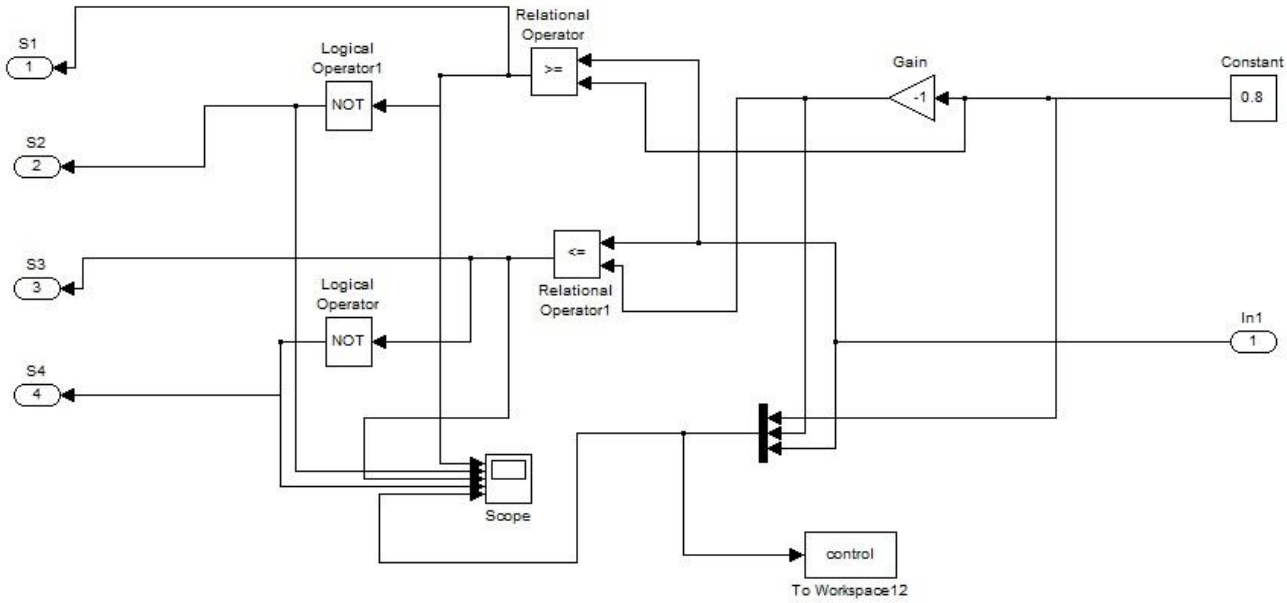


Figure 72 SubSystem Inverter Scheme Simulink S_17.

Graph of the waveform at the output load. The yellow line represents the Current that runs through load and purple line represents the Voltage in it:

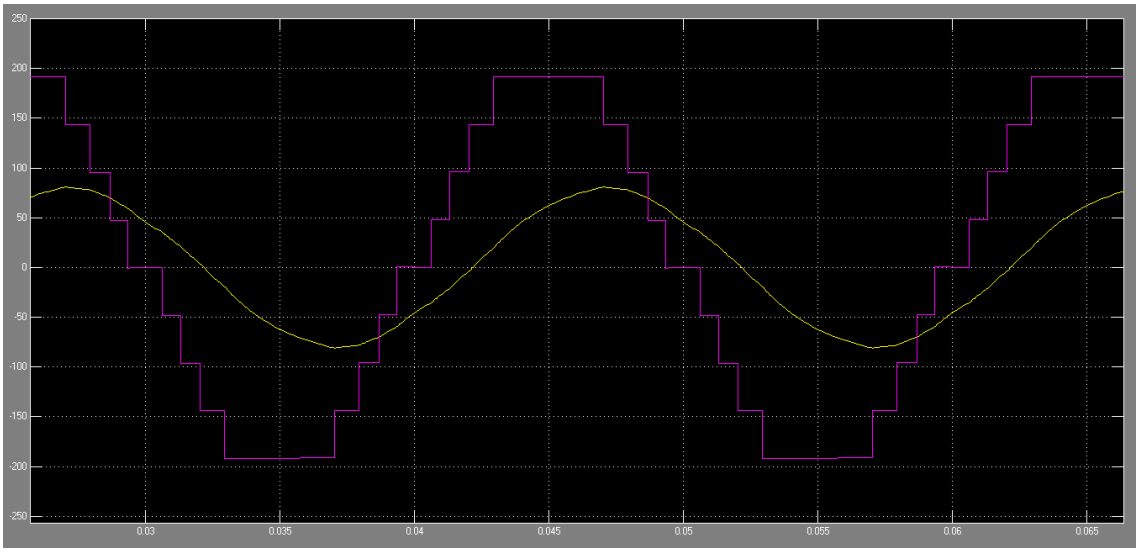


Figure 73 Output Load Inverter Simulink S_17.

Graph of the waveform of the inverter control. It can see when each transistor is open or closed:

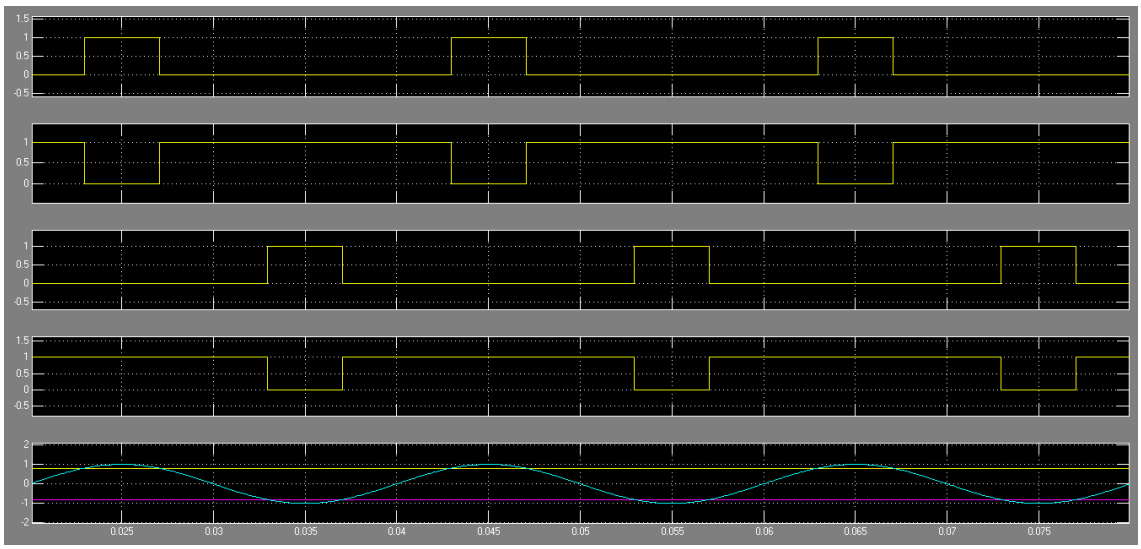


Figure 74 Control Inverter Simulink S_17.

2.3.18 S_18 - Cascade Full Bridge Three-Phase Inverter (Star Load) (3 Levels)

Main block diagram:

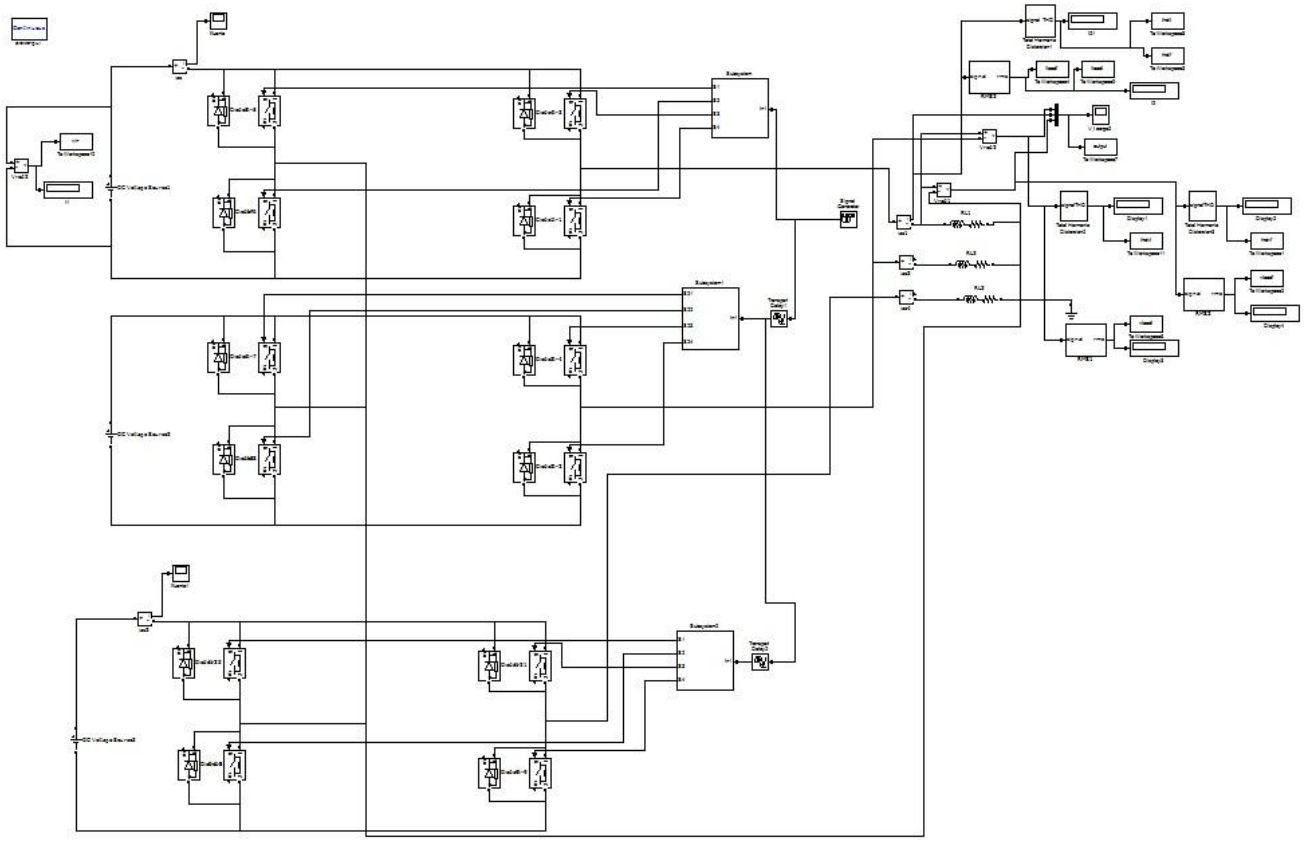


Figure 75 Inverter Scheme Simulink S_18.

SubSystem:

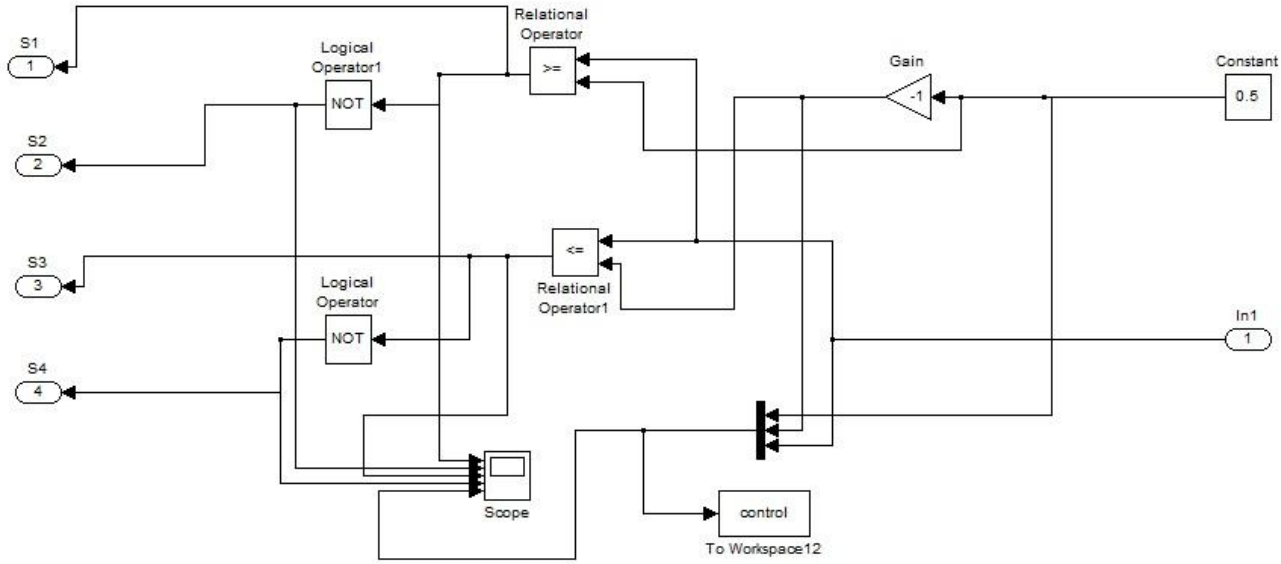


Figure 76 SubSystem Inverter Scheme Simulink S_18.

Graph of the waveform at the output load. The purple line represents the Current that runs through load, yellow line represents the Voltage (line) and blue line is the Voltage (phase) in it:

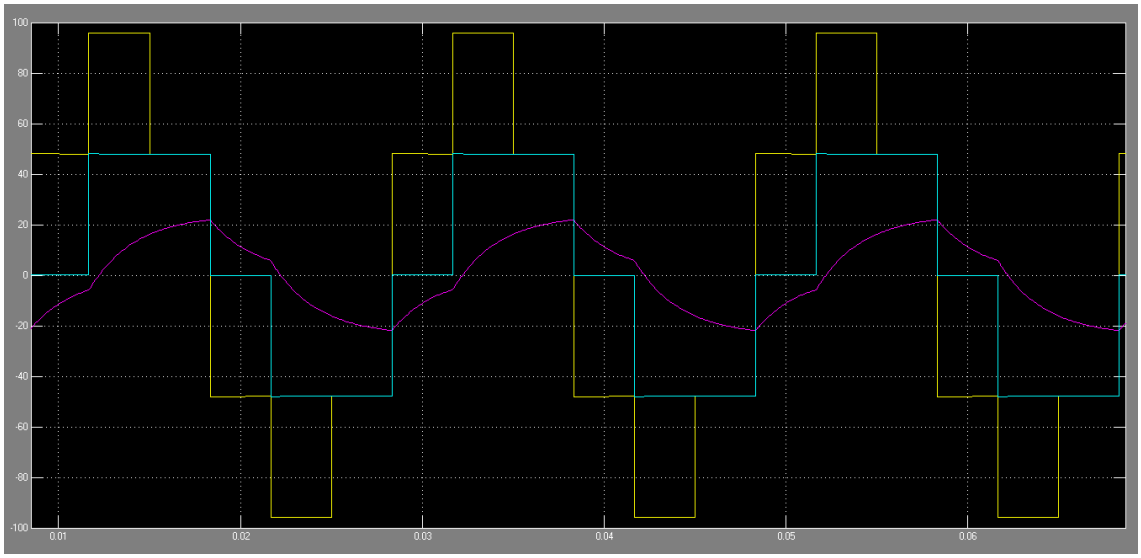


Figure 77 Output Load Inverter Simulink S_18.

Graph of the waveform of the inverter control. It can see when each transistor is open or closed:

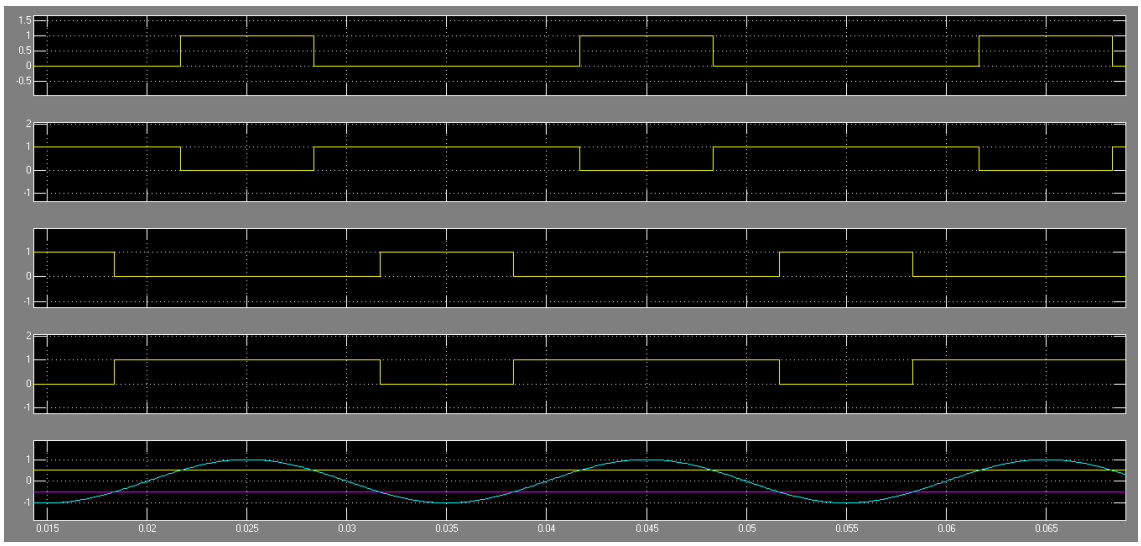


Figure 78 Control Inverter Simulink S_18.

2.3.19 S_19 - NPC Inverter

Main block diagram:

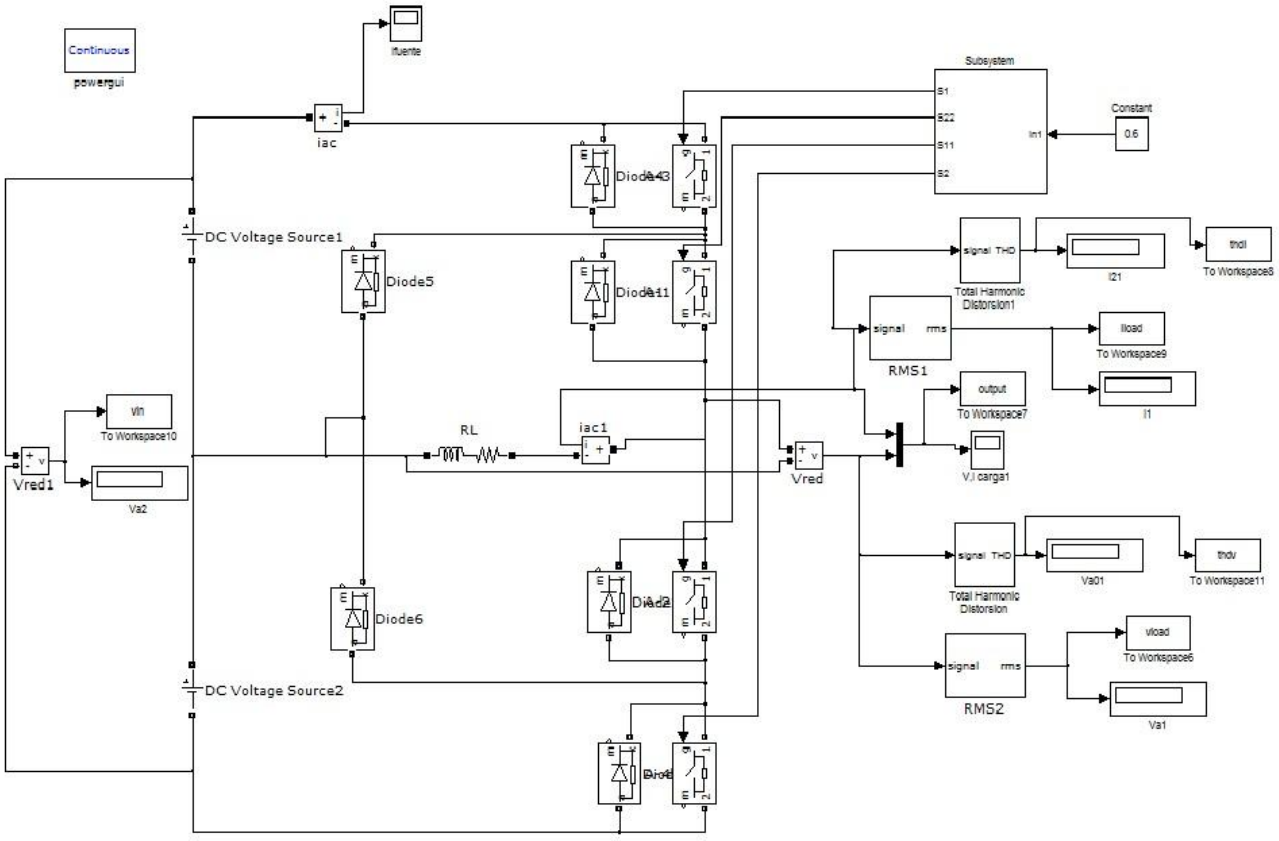


Figure 79 Inverter Scheme Simulink S_19.

SubSystem:

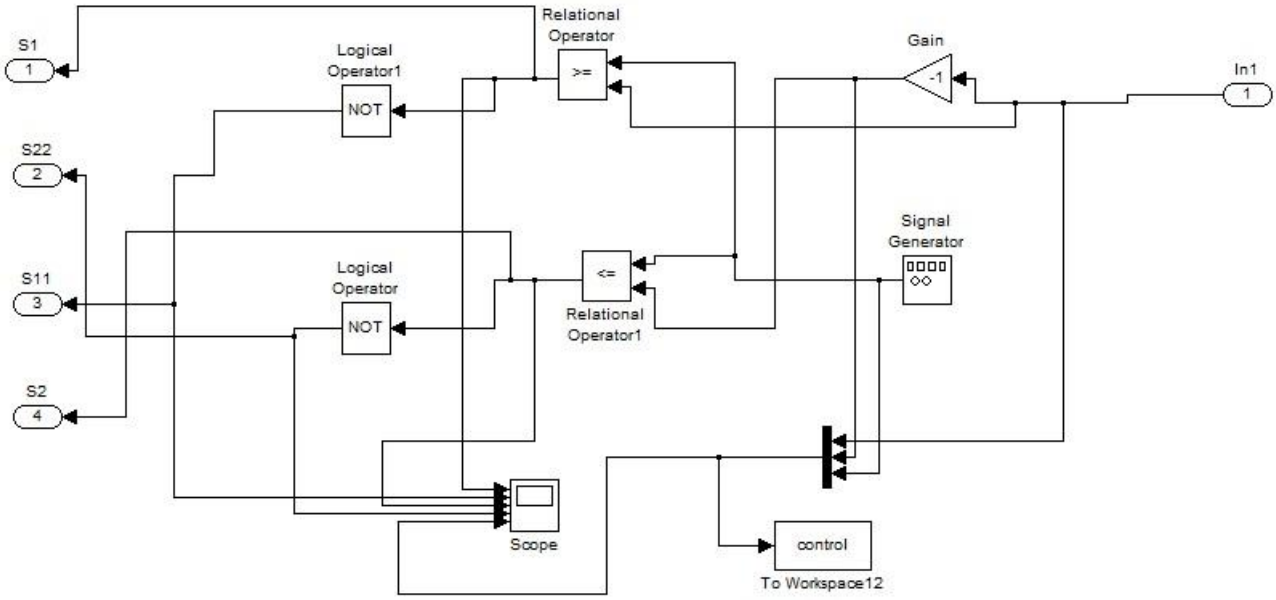


Figure 80 SubSystem Inverter Scheme Simulink S_19.

Graph of the waveform at the output load. The yellow line represents the Current that runs through load and purple line represents the Voltage in it:

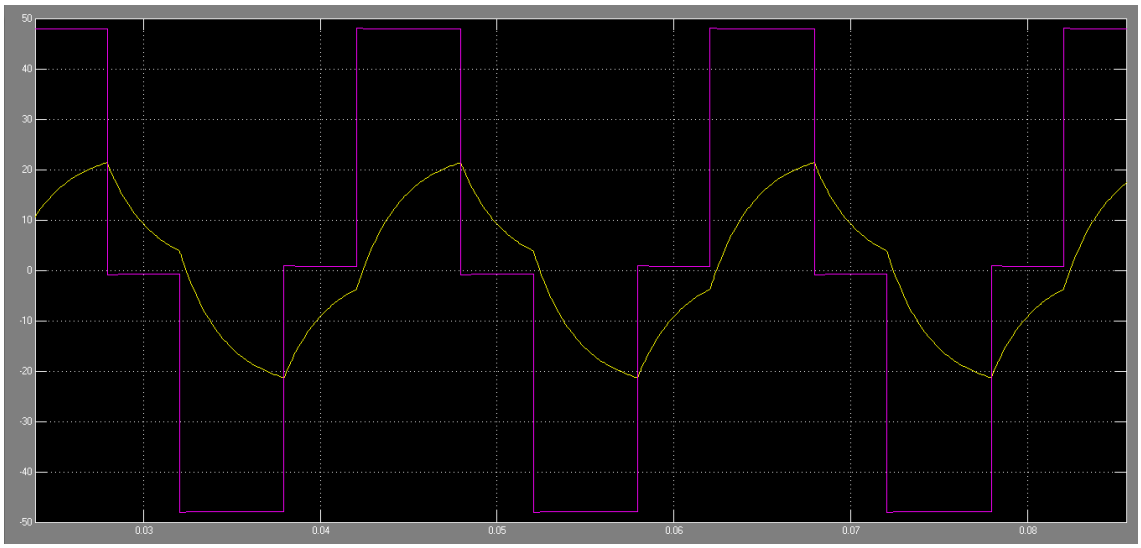


Figure 81 Output Load Inverter Simulink S_19.

Graph of the waveform of the inverter control. It can see when each transistor is open or closed:

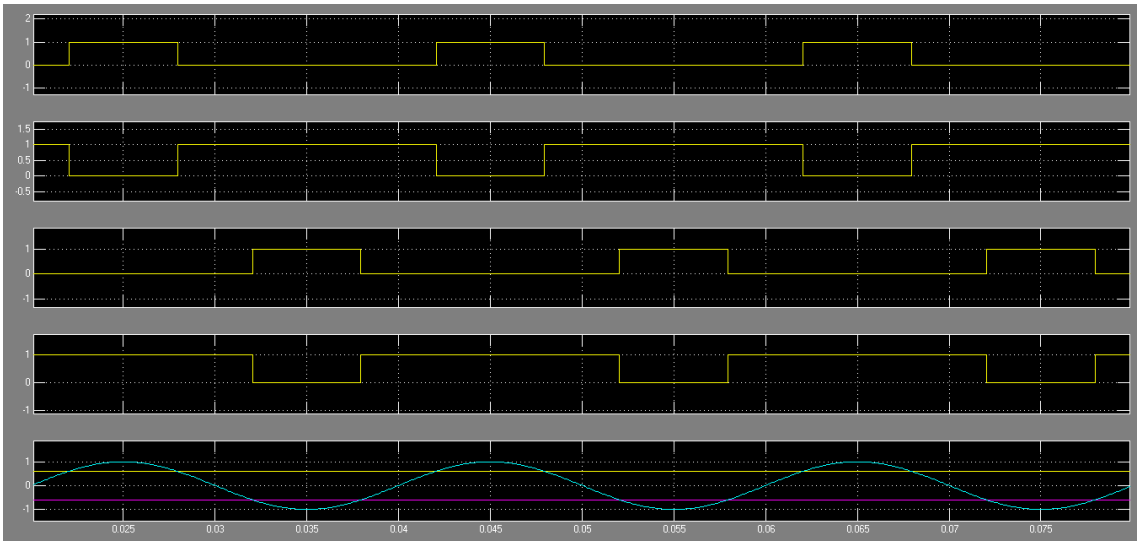


Figure 82 Control Inverter Simulink S_19.

2.3.20 S_20 - NPC Inverter (5 Levels)

Main block diagram:

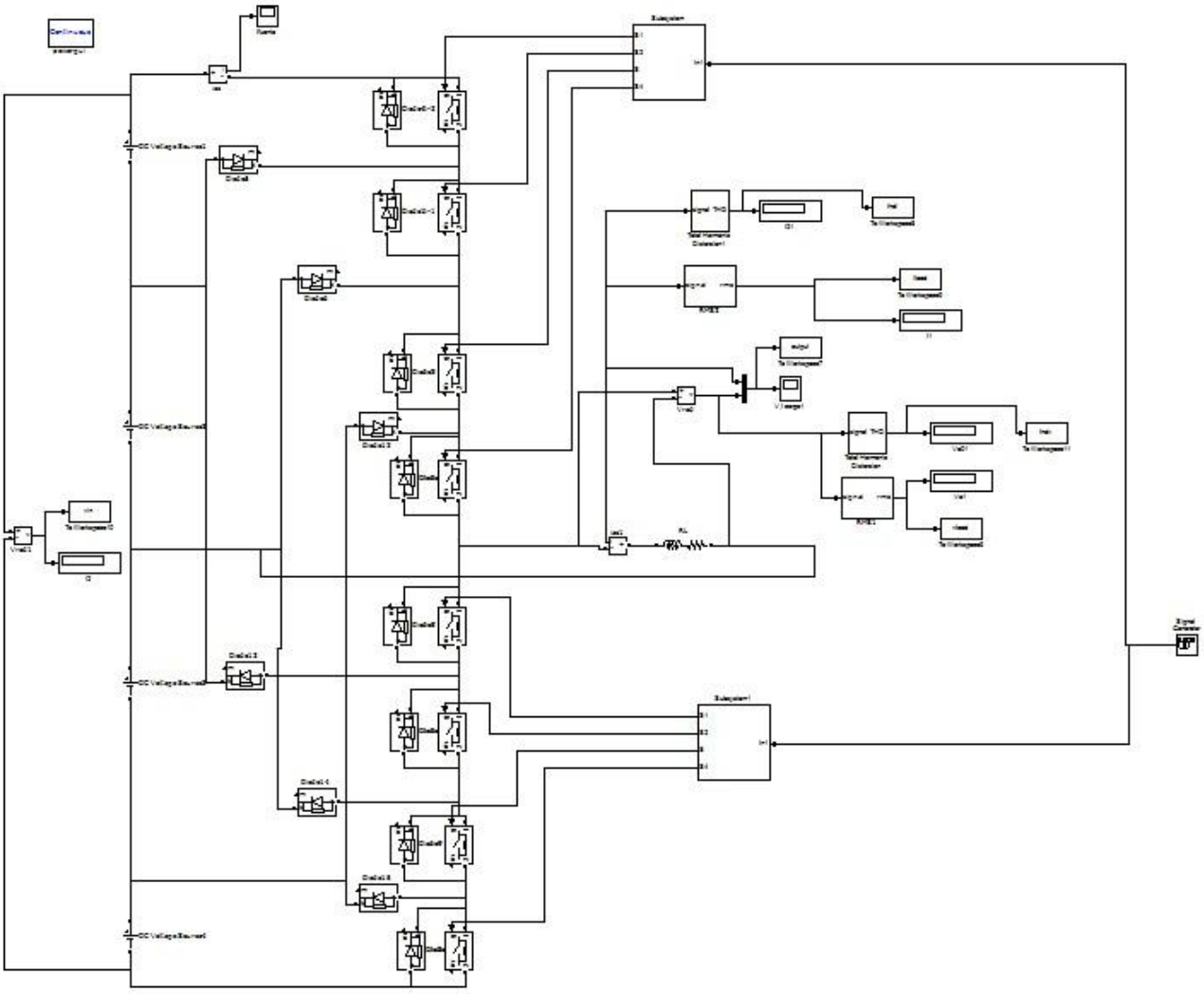


Figure 83 Inverter Scheme Simulink S_20.

SubSystem:

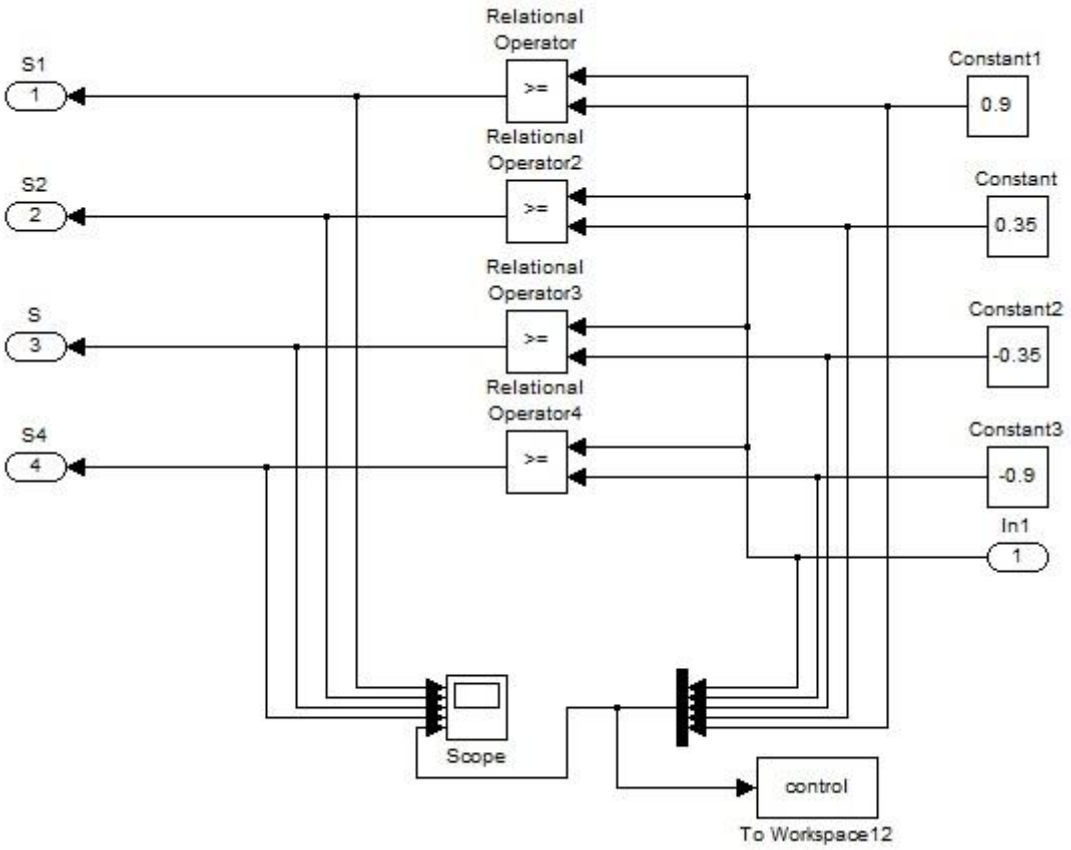


Figure 84 SubSystem Inverter Scheme Simulink S_20.

Graph of the waveform at the output load. The yellow line represents the Current that runs through load and purple line represents the Voltage in it:

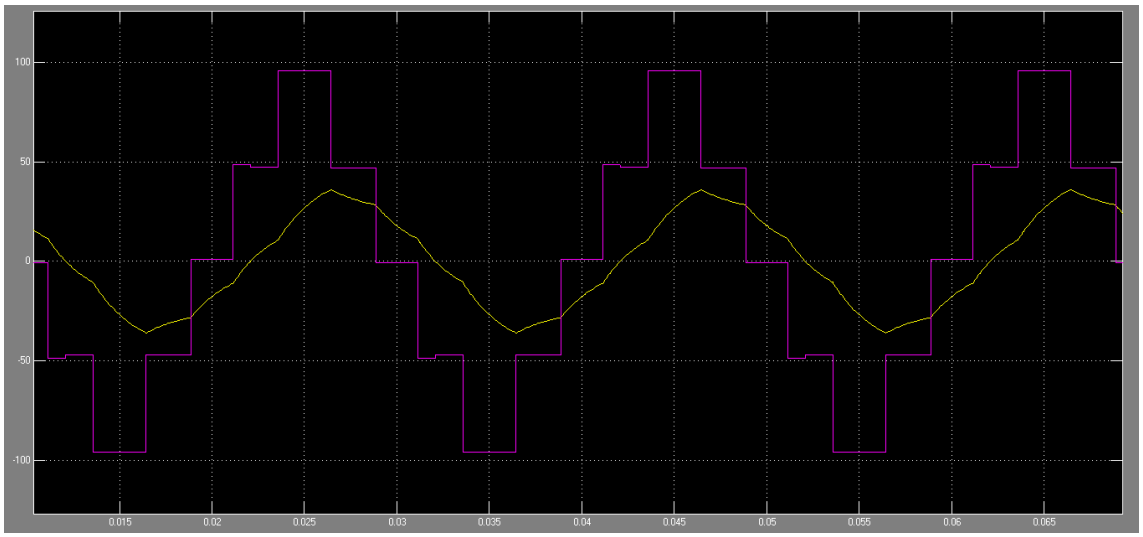


Figure 85 Output Load Inverter Simulink S_20.

Graph of the waveform of the inverter control. It can see when each transistor is open or closed:

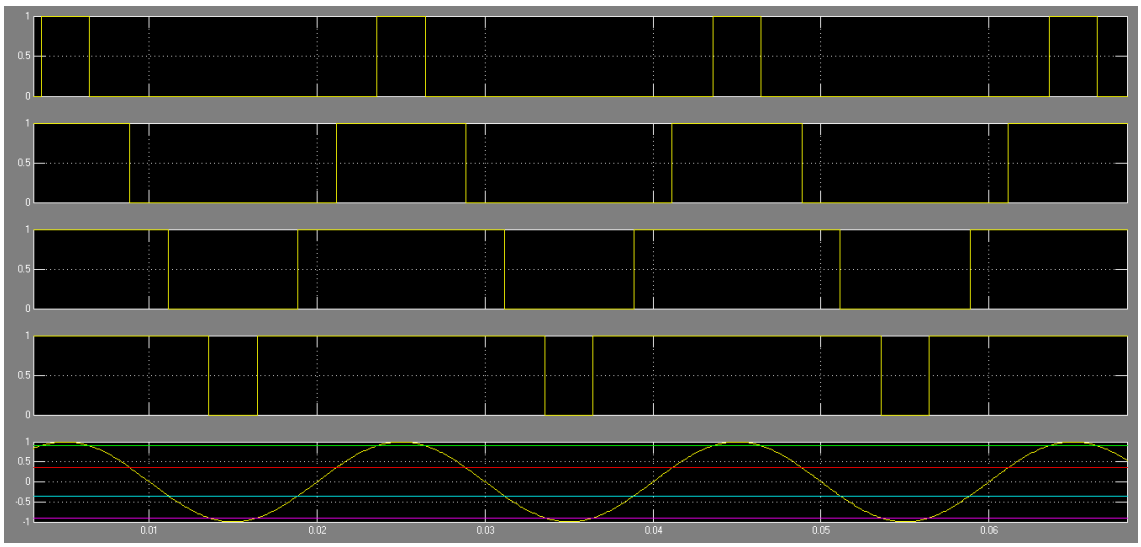


Figure 86 Control Inverter Simulink S_20.

2.3.21 S_21 - NPC Three-Phase Inverter [Star Load]

Main block diagram:

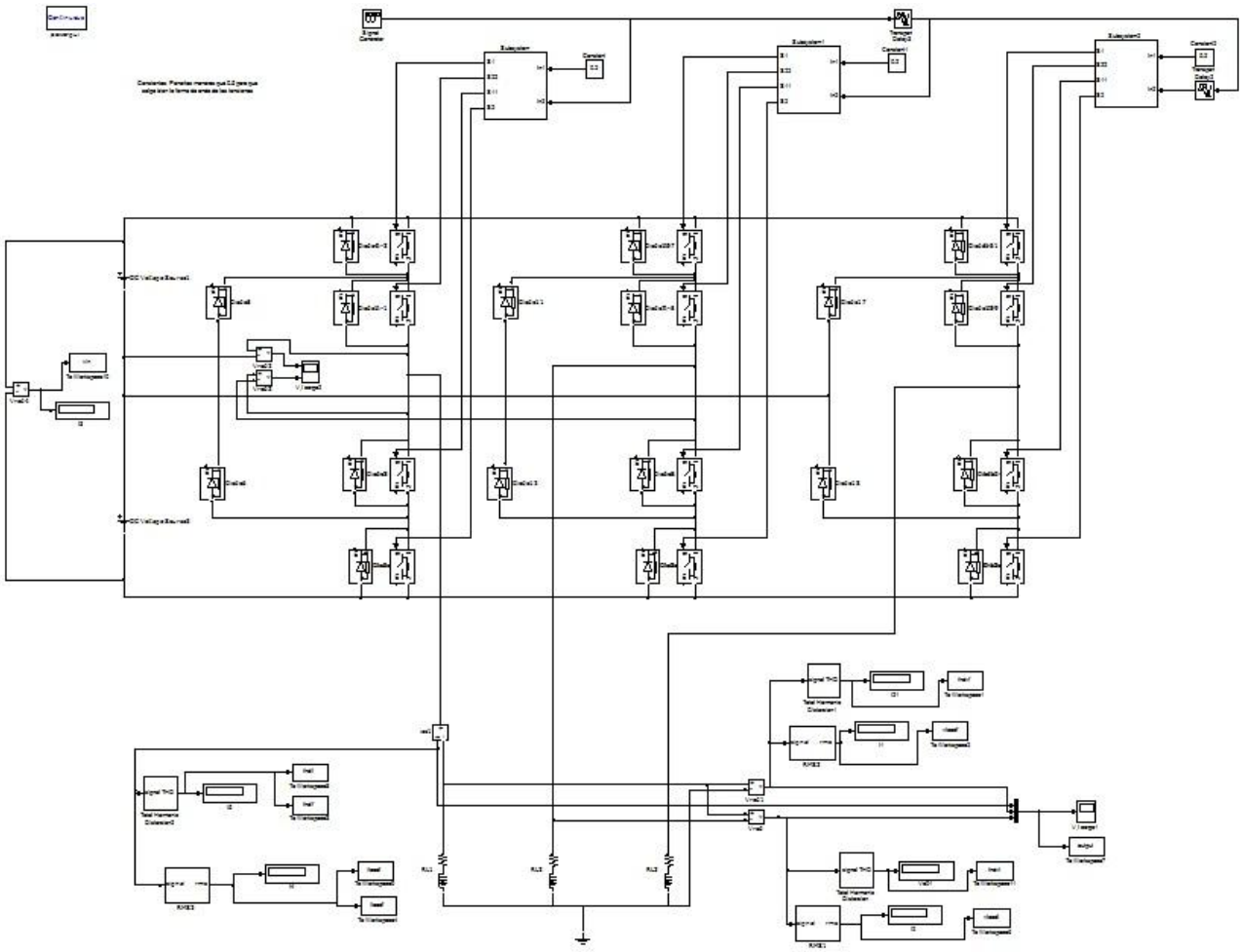


Figure 87 Inverter Scheme Simulink S_21.

SubSystem:

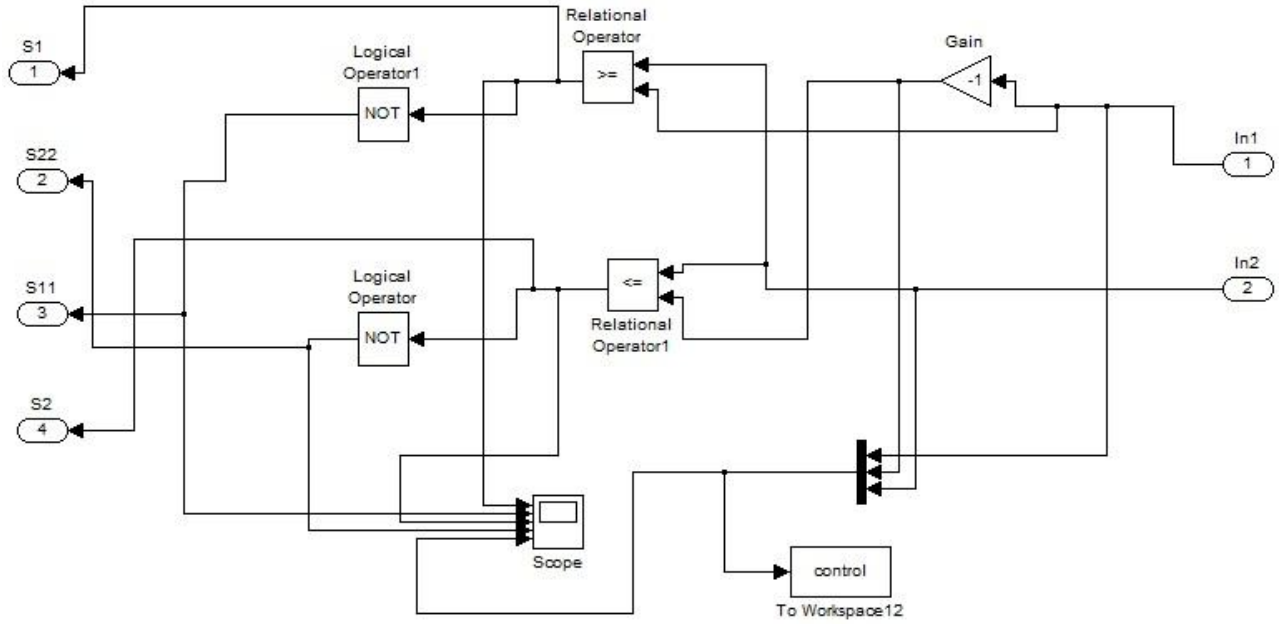


Figure 88 SubSystem Inverter Scheme Simulink S_21.

Graph of the waveform at the output load. The yellow line represents the Current that runs through load, blue line represents the Voltage (line) and purple line is the Voltage (phase) in it:

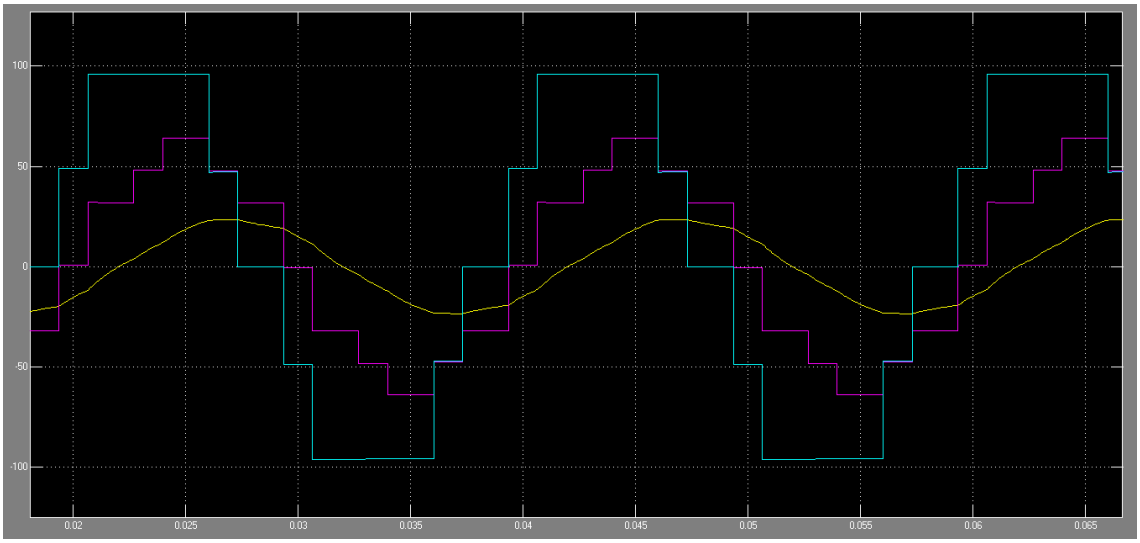


Figure 89 Output Load Inverter Simulink S_21.

Graph of the waveform of the inverter control. It can see when each transistor is open or closed:

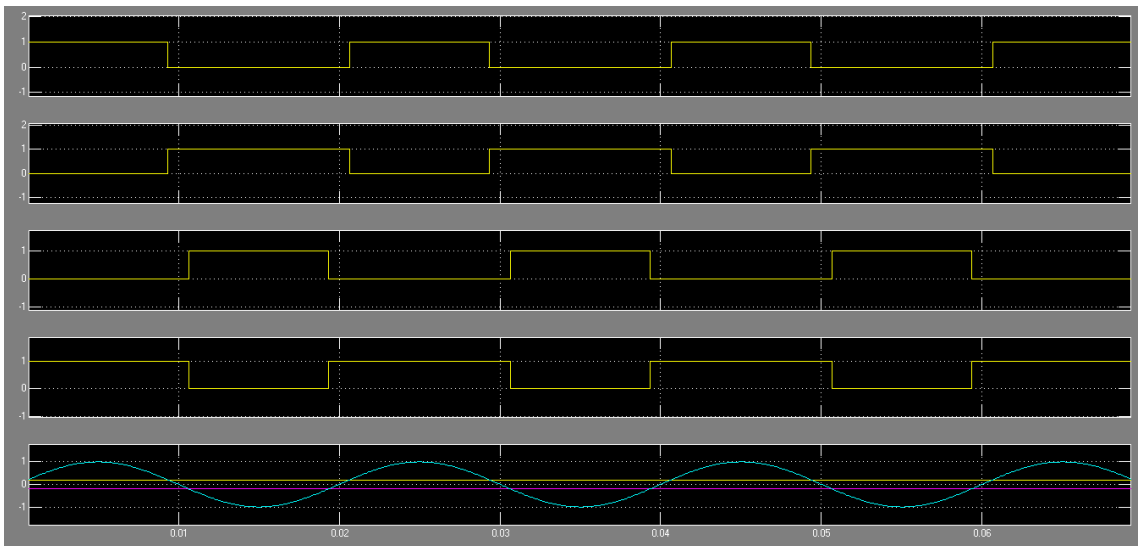


Figure 90 Control Inverter Simulink S_21.

2.3.22 S_22 - FC Inverter (5 Levels)

Main block diagram:

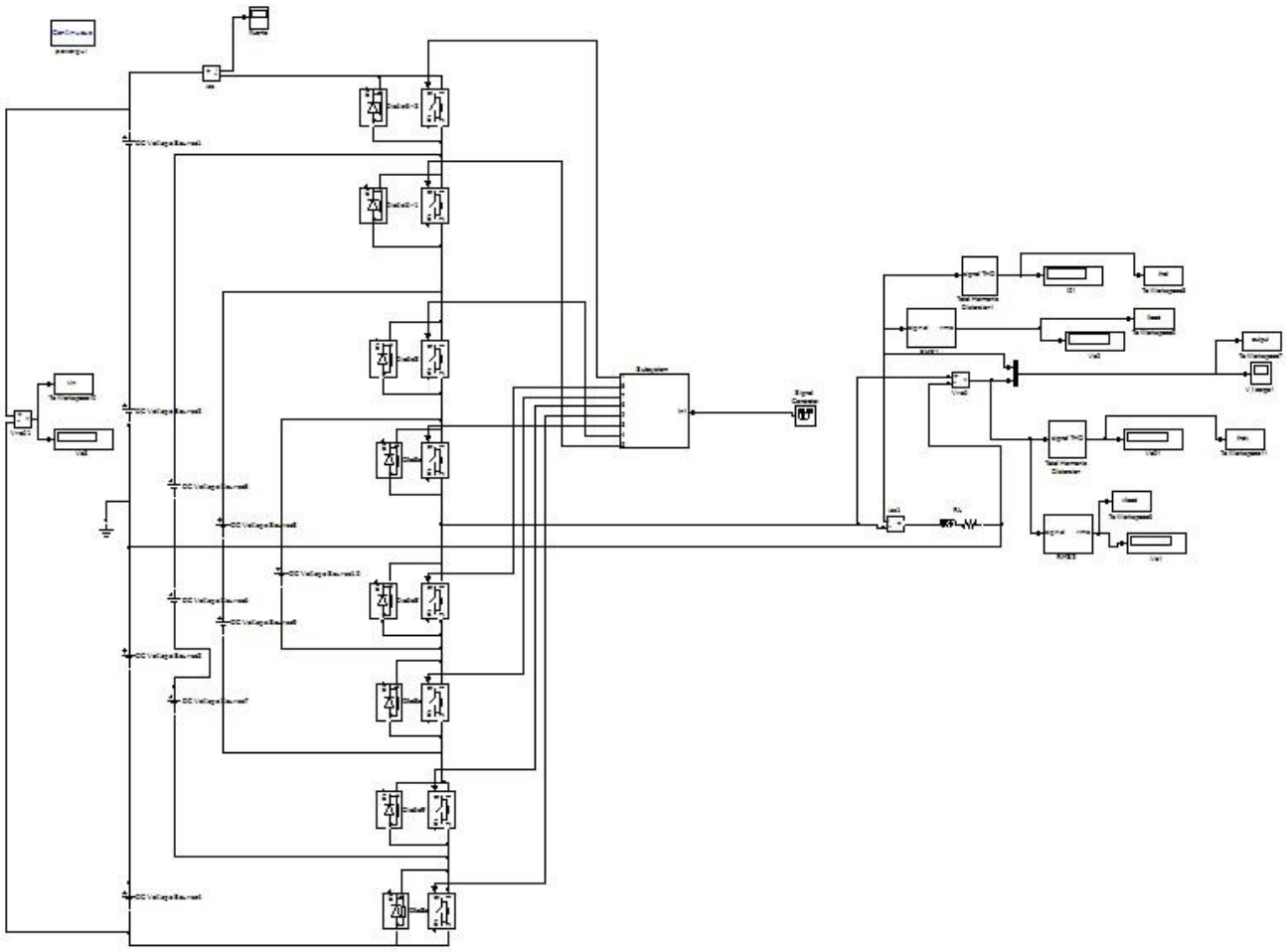


Figure 91 Inverter Scheme Simulink S_22.

SubSystem:

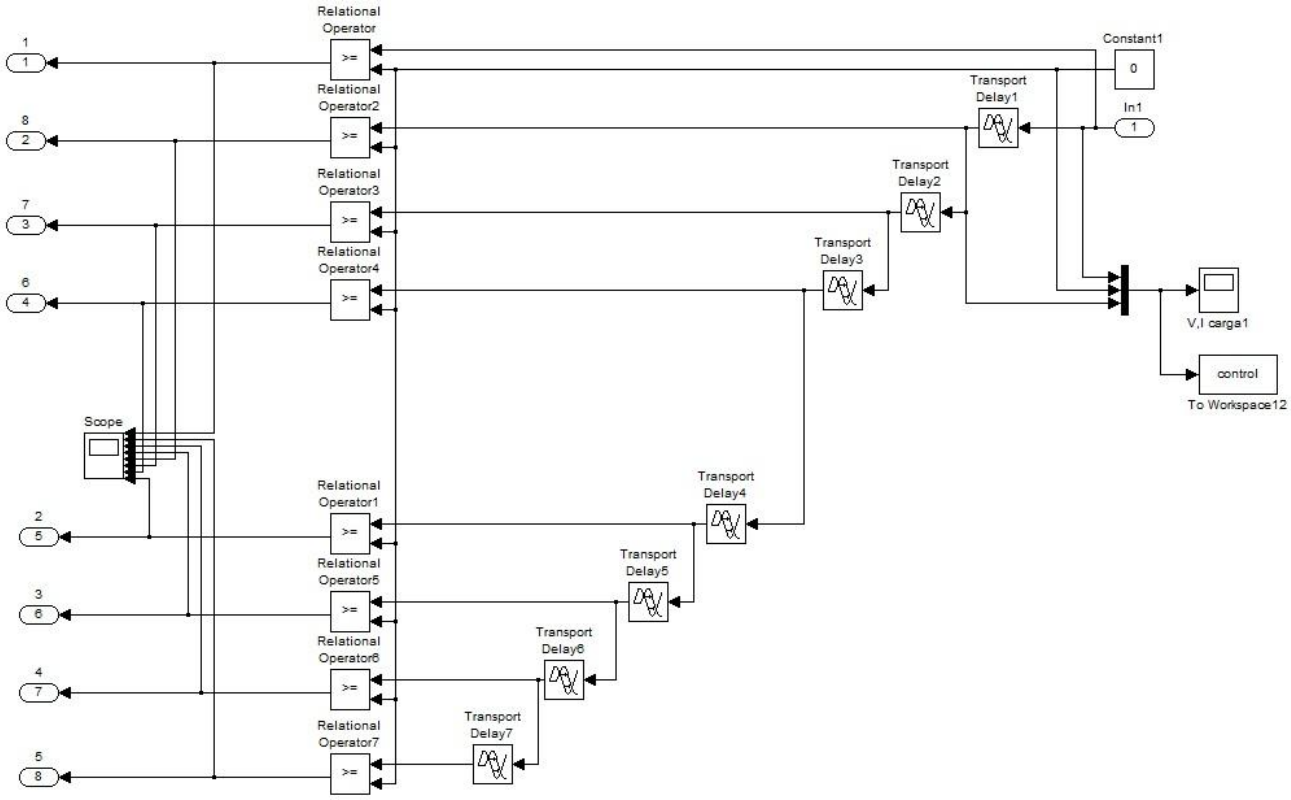


Figure 92 SubSystem Inverter Scheme Simulink S_22.

Graph of the waveform at the output load. The yellow line represents the Current that runs through load and purple line represents the Voltage in it:

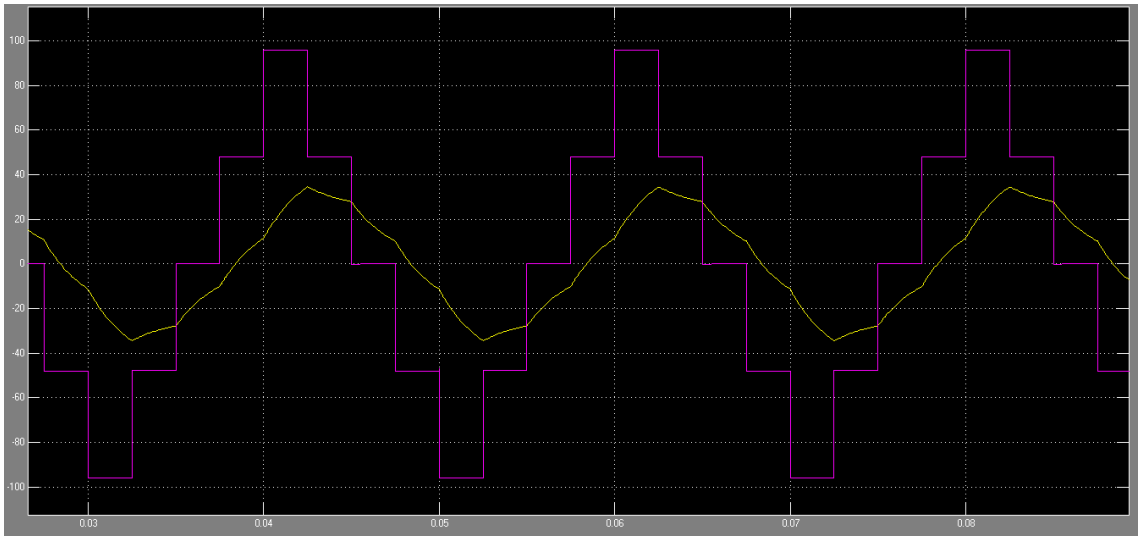


Figure 93 Output Load Inverter Simulink S_22.

Graphs of the waveform of the inverter control. It can see when each transistor is open or closed:

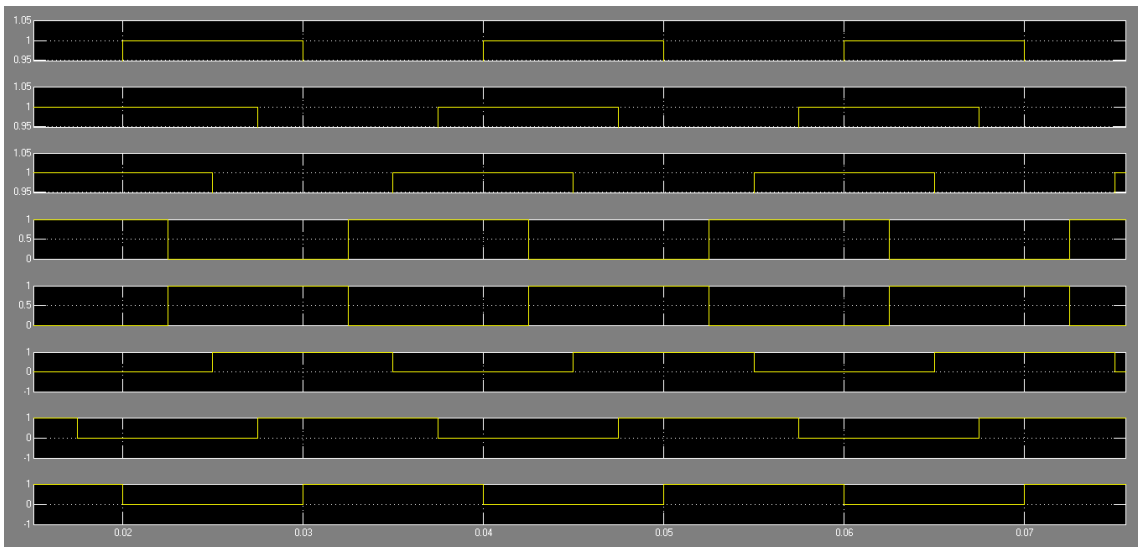


Figure 94 Control Inverter Simulink S_22 (1).

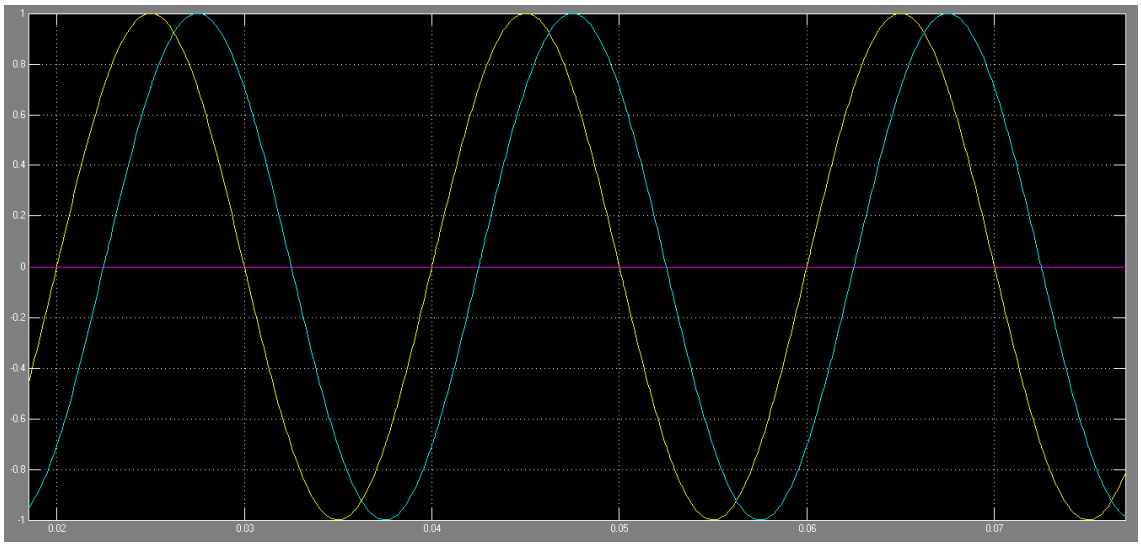


Figure 95 Control Inverter Simulink S_22 (2).

2.3.23 S_23 - FC Three-Phase Inverter (Star Load) (3 Levels)

Main block diagram:

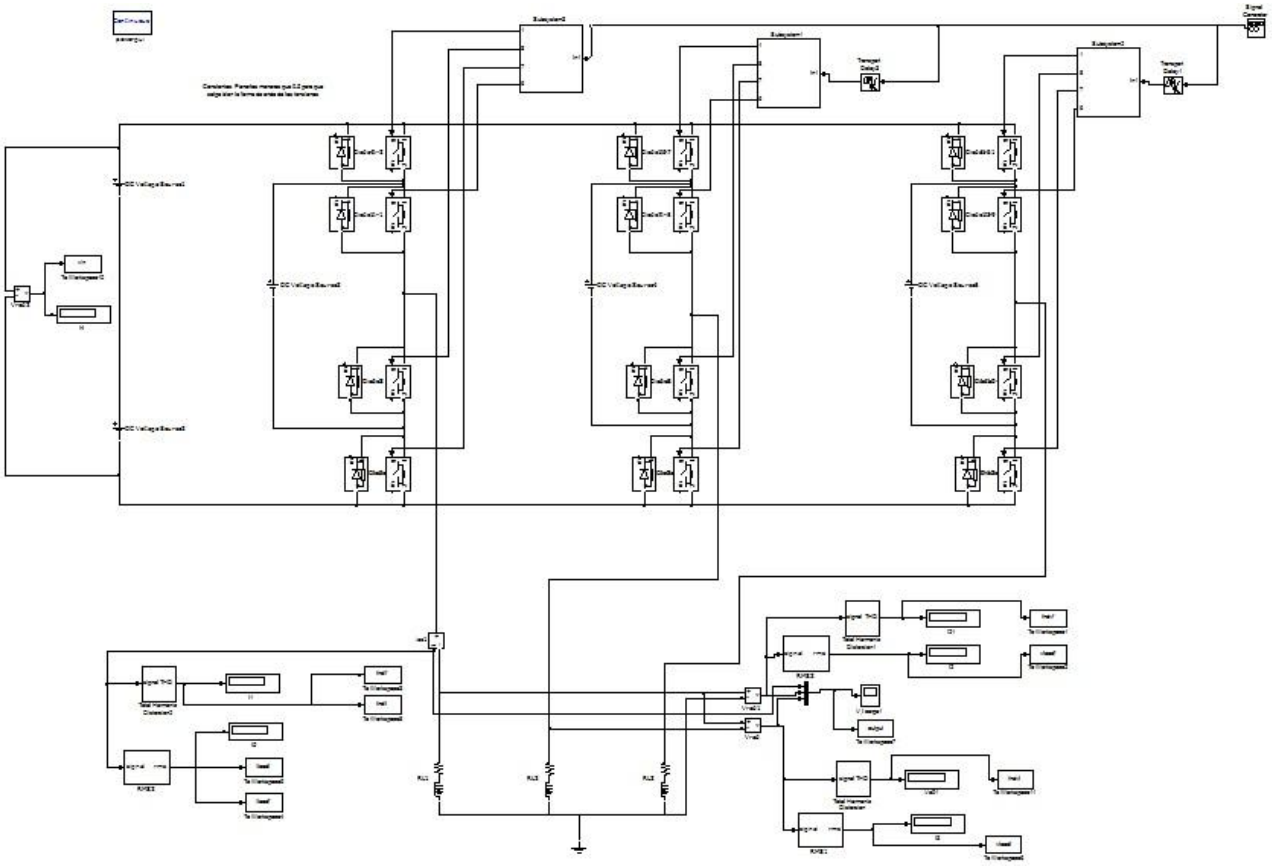


Figure 96 Inverter Scheme Simulink S_23.

SubSystem:

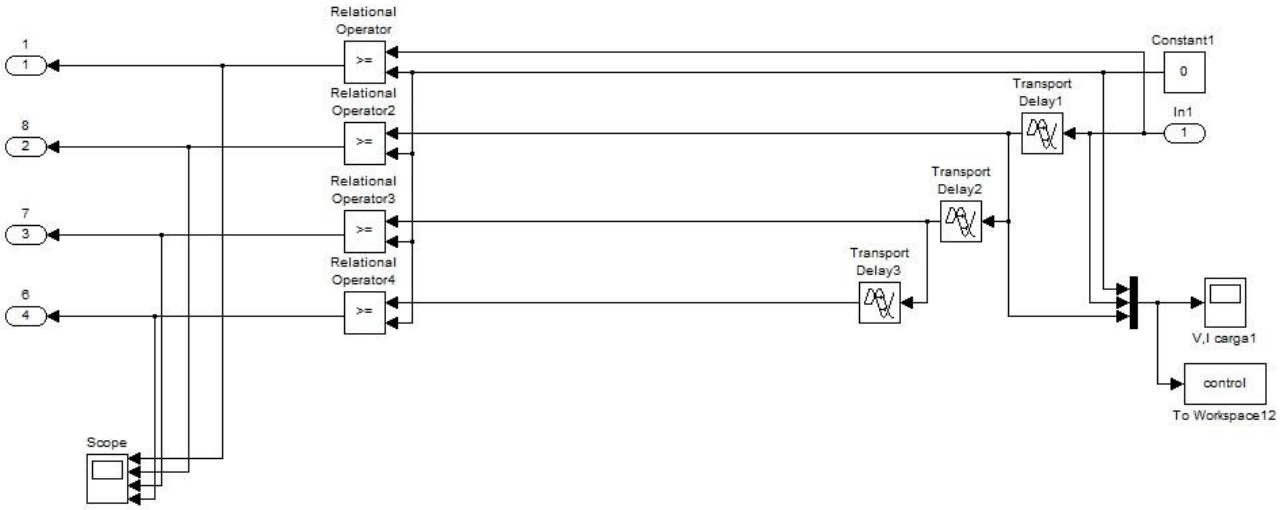


Figure 97 SubSystem Inverter Scheme Simulink S_23.

Graph of the waveform at the output load. The yellow line represents the Current that runs through load, blue line represents the Voltage (line) and purple line is the Voltage (phase) in it:

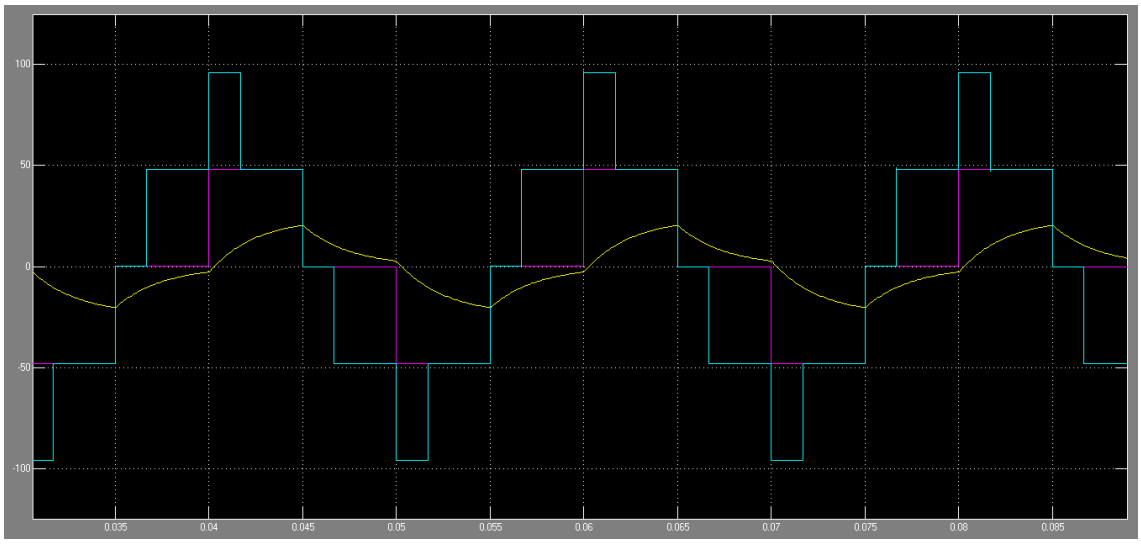


Figure 98 Output Load Inverter Simulink S_23.

Graphs of the waveform of the inverter control. It can see when each transistor is open or closed:

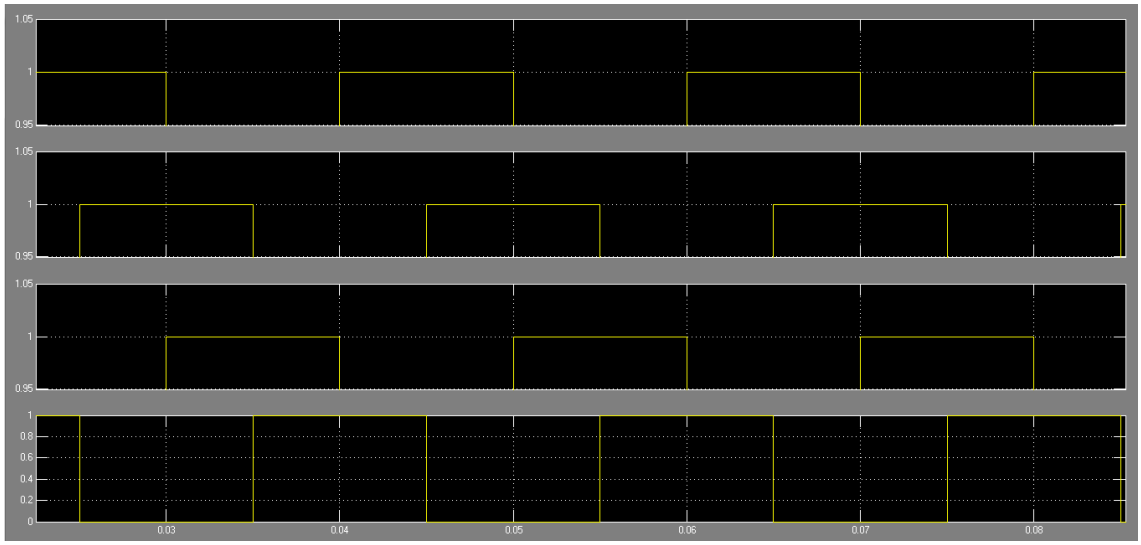


Figure 99 Control Inverter Simulink S_23 (1).

Figure 100 Control Inverter Simulink S_23 (2).

ANNEX III

DESCRIPTION OF THE *PSPICE* COMPUTER PROGRAM AND SIMULATION OF INVERTERS WITH *PSPICE*

Author

Diego Salas Forns

Director

D. Francisco José Pérez Cebolla

INDEX

INDEX OF TABLES

INDEX OF FIGURES

1. DESCRIPTION OF THE COMPUTER PROGRAM PSpice

1.1 WHAT IS PSpice?

SPICE is a program of simulation and design of analog and digital circuits. SPICE is an acronym for "Simulation Program with integrated Circuit Emphasis".

The purpose of this software is to simulate analog electronic circuits composed of resistors, capacitors, diodes, transistors, etc. For this, the following procedure is followed: First you have to describe the components, and then you have to describe the circuit and then choose the type of simulation (time, frequency, continuous, parametric, etc.).

The fundamental unit of programming Spice is the Netlist. It is an ASCII the file that contains the description of the circuit in Spice language as well as the different types of analysis (time domain, frequency etc.). After performing the Netlist, It is launched the “compiler" Spice that will tell us if there are mistakes or our circuit works correctly from the syntactic point of view. The results are displayed in the program Probe.

Spice, as expected, has developed into a graphical environment: Schematics. This tool allows us to make our circuits without needing to know the syntax Spice, with consequent savings time and effort. However, it is advisable to know the terminology Spice if we take full advantage of the program [10-11].

1.2 HISTORY OF PSpice

PSpice was created by the research laboratory of electronics at the University of Berkeley in California and the first time that was offered to the public was in 1975.

For personal computers and workstations there are various software packages that implement SPICE. Of these, the most popular is PSpice, created by MicroSim Corporation. It’s commercially available since 1984. Subsequent OrCAD Inc. purchased the product being the company that develops and distributes new versions of PSpice. Currently the latest version is 9.

SPICE1: It was developed by the CANCER program (acronym of Computer Analysis of Nonlinear Circuits, Excluding Radiation) at the University of Berkeley in the '60s. The program had its first presentation at a conference in 1973. It was programmed in FORTRAN language and used the technique of analysis of nodes to build the system of equations of the circuit.

SPICE2: In 1975 was launched the SPICE2 version with which popularized its use. This version of the program was also compiled into FORTRAN, it had more elements, transient analysis variable pitch, using the techniques of trapezoidal integration or integration Gear, getting the equations of the circuits by a modified traditional nodal analysis, which help remedy the inconveniences of the previous version and used a FORTRAN program innovation that allowed control memory.

SPICE3: The latest version of SPICE was 2G.6 FORTRAN version in 1983. The next version, SPICE3, was developed in "C" language, by Thomas Quarless, and the director was A. Richard in the year 1989. The SPICE3 version using the same syntax as its predecessors and had a "X Window" graphical interface.

It is an open source software, so it was widely used. SPICE code was distributed from its beginnings under a cost for the University of Berkeley. The program had the restriction of not being able to distribute in countries not considered friendly to the United States. Currently the program is covered by the BSD license.

SPICE promoted and formed the basis for other simulation programs in universities and industry. The first commercial version of SPICE was ISPICE. The most prominent commercial version of SPICE included: HSPICE and PSPICE. Academic versions of SPICE included XSPICE, developed at the Georgia Institute of Technology. Codes analog and digital analysis and Cider were added, allowing simulating semiconductor devices.

1.3 PARTS OF PSpice

PSpice is a combination of several programs. Each performs specific tasks within PSpice. Then briefly explain each of these four parts:

SCHEMATICS: t's where the circuit is drawn. It is a graphic editor that is used to draw and design the circuit to be simulated. Allows the user to incorporate the components stored in libraries, connect to make the circuit, change the characteristic parameters of these components and specify the type of analysis which it is required.

PSpice A/D: It is the program responsible to simulate the circuit created by the Schematics. From a file that describes the schema of the circuit, produces an output file will depend on the type of test used.

PROBE: It is a program to visualize the simulation results generated by PSpice A/D. It can be used to observe any voltage or current in the circuit. Also allows operations and display between signals such power signals. It is capable of displaying signals in function of time and as a function of frequency (frequency analysis).

DESIGN MANAGER: Allows the user to manage files related to the design he made. This program always opens every time start running any of the above programs.

1.4 TYPES OF ANALYSIS THAT EXECUTES PSpice

All versions of PSpicehave several types of analysis, such as:

- DC - Transfer function
- AC - Frequency response of circuit
- Transient - Evolution in time of the circuit.

And depending on the version used of PSpice, implemented more advanced analysis can be found as follows:

- Operating Point CD
- Analysis of CA
- Analysis of CA Single Frequency
- Transient Analysis
- Fourier Analysis
- Noise Analysis
- Noise Figure Analysis
- Analysis of Distortion
- Analysis of CD
- Worst-case analysis
- Behavior of Monte Carlo
- Sensitivity
- Parameter Sweeping
- Temperature Sweeping

1.5 PROGRAM VERSIONS

There are two types of distributions PSpice: a students, in which the number of components that can be used is limited, and a professional without limitations. While the first is free and can be downloaded from the Internet (*www.pspice.com*) the second costs a few thousand Euros.

There are many versions of OrCAD PSpice, but which has been used in this project is OrCAD PSpiceFull Version 9.2.

2. SIMULATION OF INVERTERS WITH PSpice

2.1 NAME OF INVERTERS SIMULATED

In Annex I have been theoretically explained all investors covered by this project.

Below you can see in the list (Table 29), which type of investor is each PSpice file (.opj extensión):

.opj	INVERTER
S_1	Half-Wave Inverter (Square Wave Control)
S_2	Half-Wave Inverter (PWM Control)
S_3	Asymmetric Inverter (Square Wave Control)
S_4	Asymmetric Inverter (PWM Control)
S_5	Full Wave H-Bridge Inverter (Square Wave Control)
S_6	Full Wave H-Bridge Inverter (Voltage Harmonic Cancellation)
S_7	Full Wave H-Bridge Inverter (Bipolar PWM Control)
S_8	Full Wave H-Bridge Inverter (Single-Pole PWM Control)
S_9	Three-Phase VSI 6-Step Inverter [Star] (Square Wave Control)
S_10	Three-Phase VSI 6-Step Inverter [Triangle] (Square Wave Control)
S_11	Three-Phase VSI 6-Step Inverter [Star] (PWM Control)
S_12	Three-Phase VSI 6-Step Inverter [Triangle] (PWM Control)
S_13	Full Wave H-Bridge Inverter (Hysteresis Control)
S_14	Push-Pull Inverter (Current Mode)
S_15	Push-Pull Inverter (Reg.)
S_16	Cascade Full Bridge Single-Phase Inverter (2 Levels)
S_17	Cascade Full Bridge Single-Phase Inverter (4 Levels)
S_18	Cascade Full Bridge Three-Phase Inverter (3 Levels)
S_19	NPC Inverter
S_20	NPC Inverter (5 Levels)
S_21	NPC Three-Phase Inverter
S_22	FC Inverter (5 Levels)
S_23	FC Three-Phase Inverter (3 Levels)

Table 1 Inverter Simulations with PSpice.

2.2 BASIC STEPS TO START PSpice

2.2.1 WHERE ARE THE FILES?

The files of these simulation are in “PSpice Simulations” folder.

2.2.2 HOW TO OPEN THE FILES

To open files simulation, first have to open “Caputure CIS” program, then have to click File >> Open>> Project…, and select the folder where the files are saved, as shown in Figure 197:

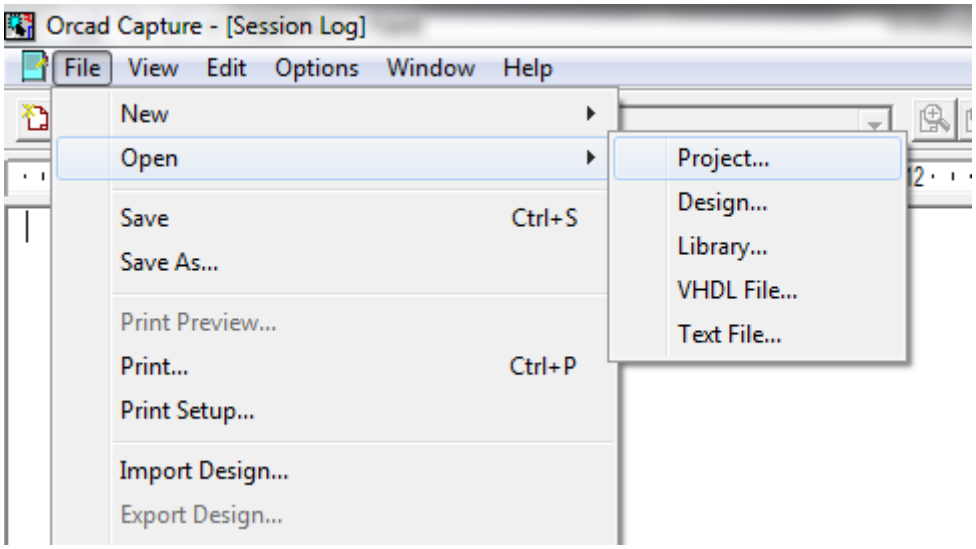


Figure 1 PSpice (1).

Now, you have to find a ".opj" file. After that, you have to select and open some folders and try to find a file like this (Figure 198):

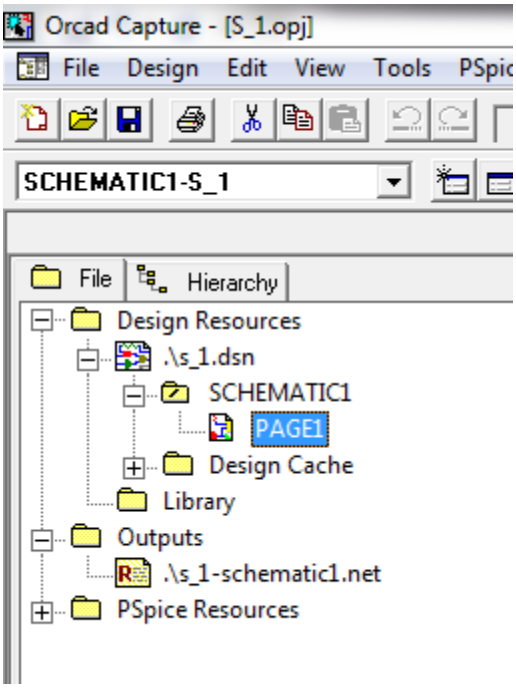


Figure 2 PSpice (2).

Double-Click and the simulation is opened.

2.2.3 HOW TO RUN THE FILES?

When you have opened Orcad Capture (Capture CIS), to run the simulation press the button PLAY (Run PSpice) located at the top of the window (Figure 199):

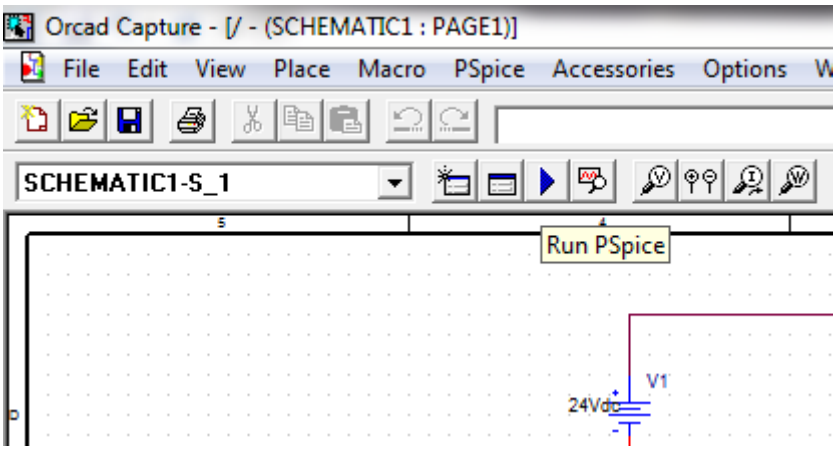


Figure 3 PSpice (3).

When the simulation is finished, a SCHEMATIC window is opened drawing the wave form that you have chosen before, for example, as you can see in the following picture (Figure 200):

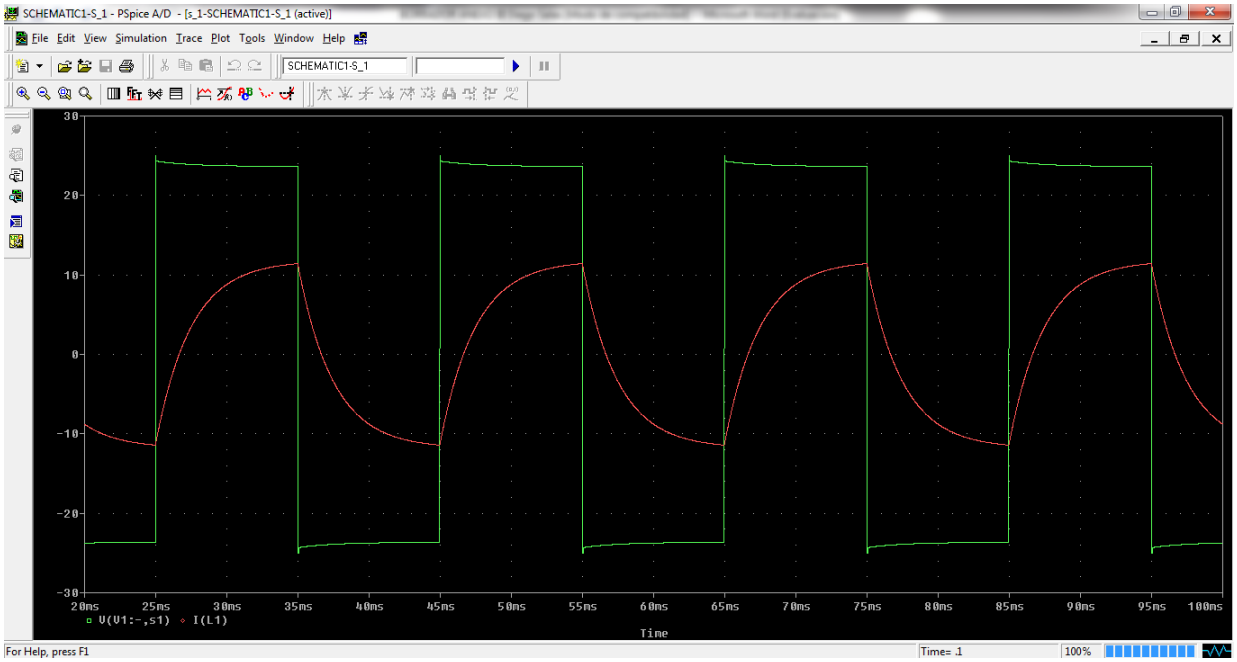


Figure 4 PSpice (4).

These are just the basic steps. May be found all the information to use PSpice ORCad in the User's Guide " PSpice -ORCad".

*Inverter Control is the same that has been explained in Annex II (Simulink-MATLAB).

2.2.4 ELECTRONIC ELEMENTES USED IN THESE SIMULATIONS WITH PSpice

Next table shows the electronic elements used in the construction of the PSpice simulations:

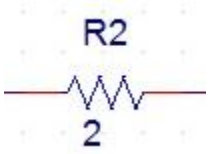
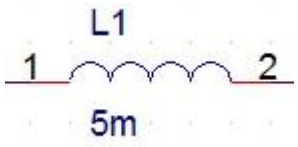
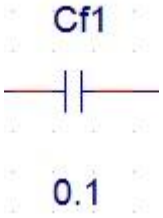
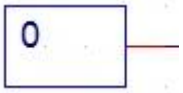
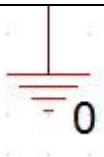
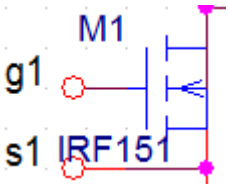
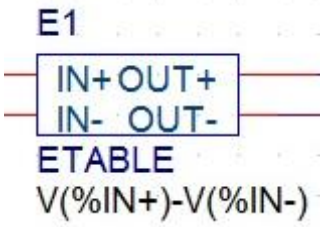
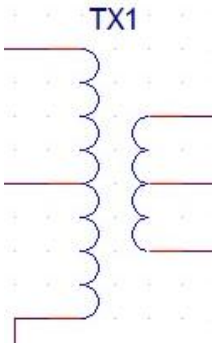
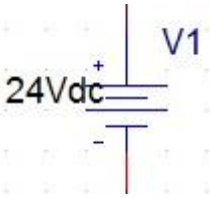
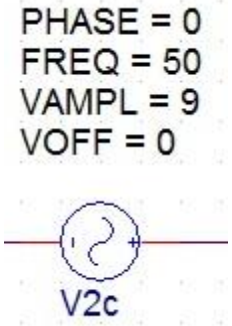
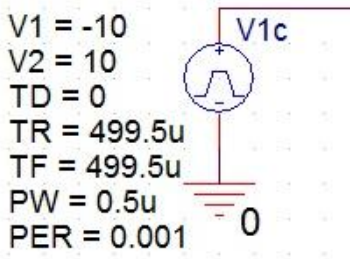
 RESISTENCE	 INDUCTANCE	 CAPACITOR
 DIODE	 CONSTANT	 GROUND
 TRANSISTOR	 ETABLE	
 DC VOLTAGE SOURCE	 AC VOLTAGE SOURCE	 VOLTAGE SOURCE

Table 2 Electronic elements

2.3 ALL SIMULATIONS WITH PSPICE

2.3.1 S_1 - Half-Wave Inverter (Square Wave Control)

The following image represents the inverter circuit:

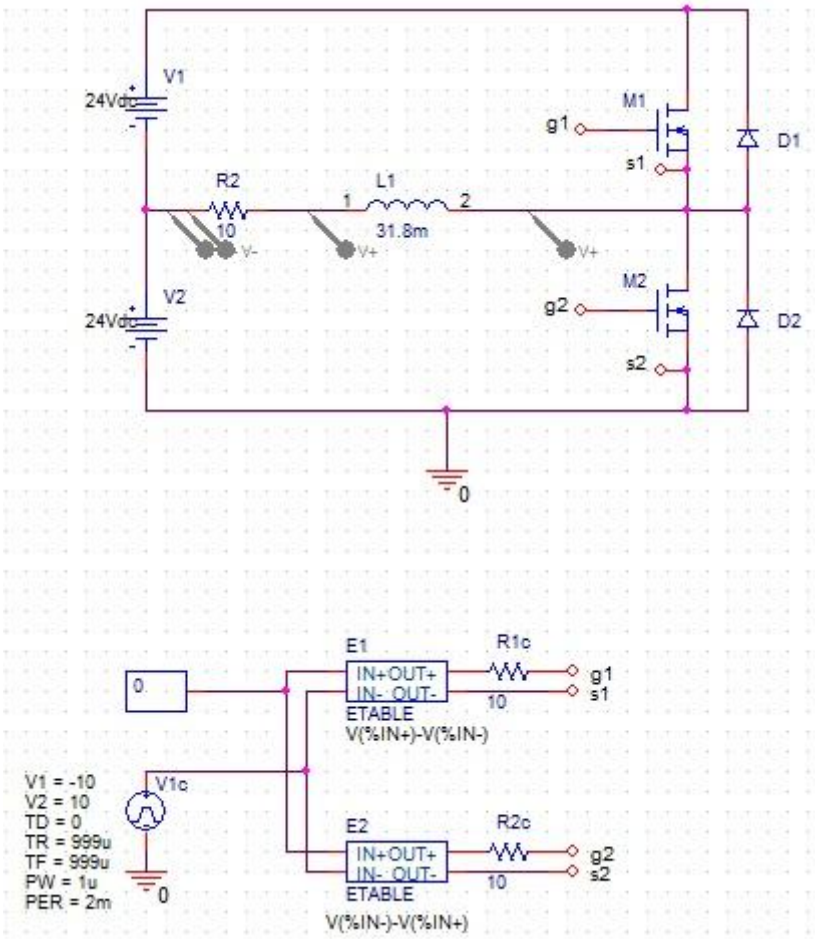


Figure 5 Inverter Scheme PSpice S_1.

The first graph corresponds to the waveform of the output at the load. The red line represents the Current that runs through load and green line represents the Voltage in it:

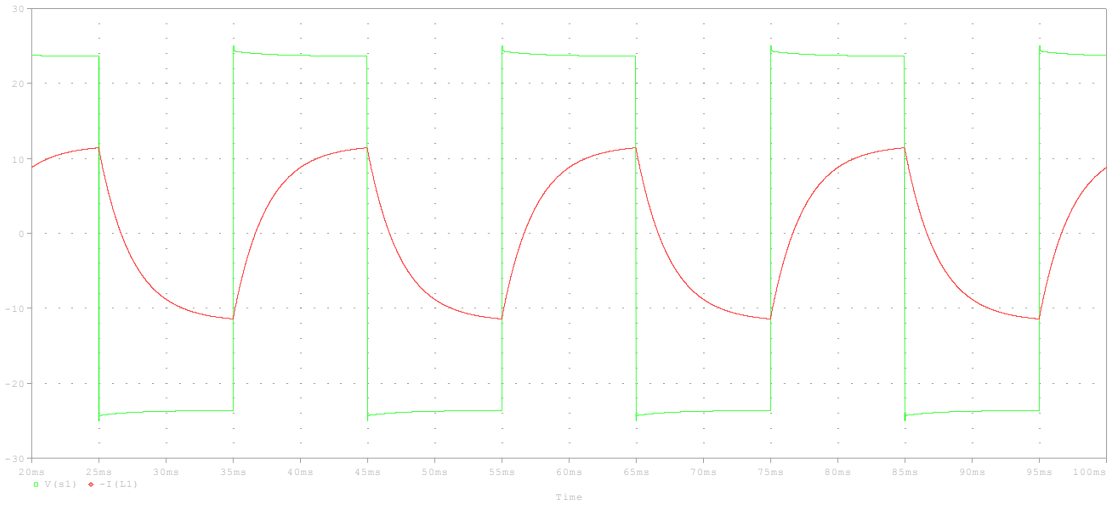


Figure 6 Output Inverter PSpice S_1.

The following graphs correspond to Fourier analysis. They represent frequency analysis of the harmonics of each waveform. Each graph goes with its own color line:

Figure 7 Fourier series Voltage of Inverter PSpice S_1.

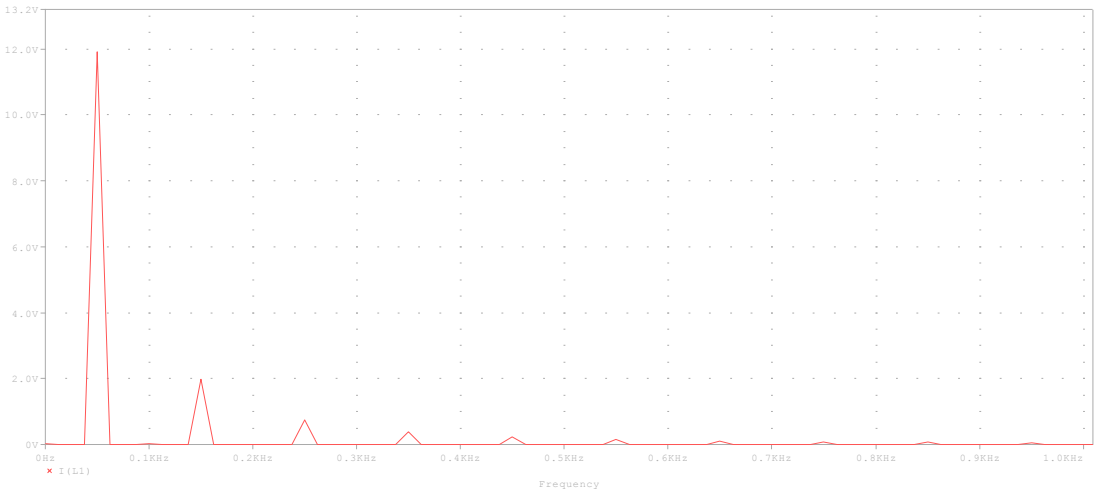


Figure 8 Fourier series Current of Inverter PSpice S_1.

2.3.2 S_2 - Half-Wave Inverter (PWM Control)

The following image represents the inverter circuit:

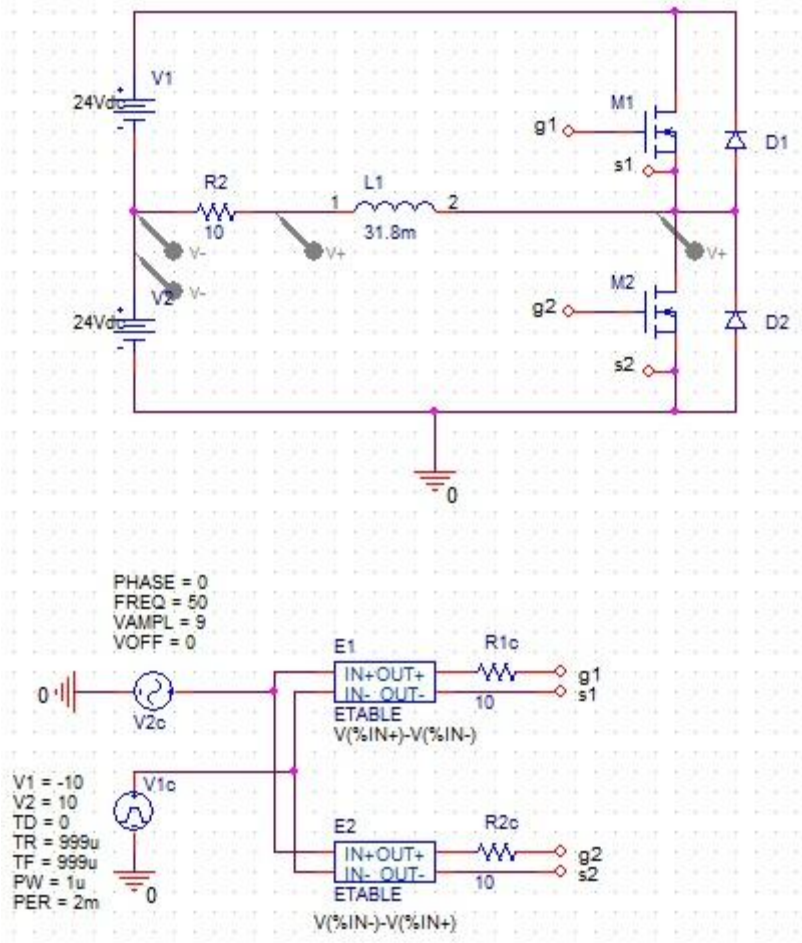


Figure 9 Inverter Scheme PSpice S_2.

The first graph corresponds to the waveform of the output at the load. The red line represents the Current that runs through load and green line represents the Voltage in it:

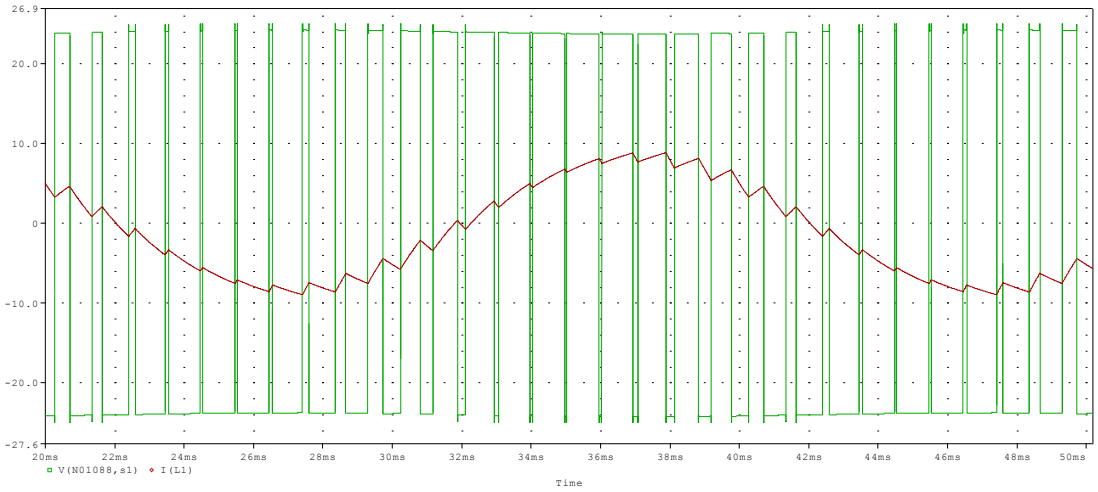


Figure 10 Output Inverter PSpice S_2.

The following graphs correspond to Fourier analysis. They represent frequency analysis of the harmonics of each waveform. Each graph goes with its own color line:

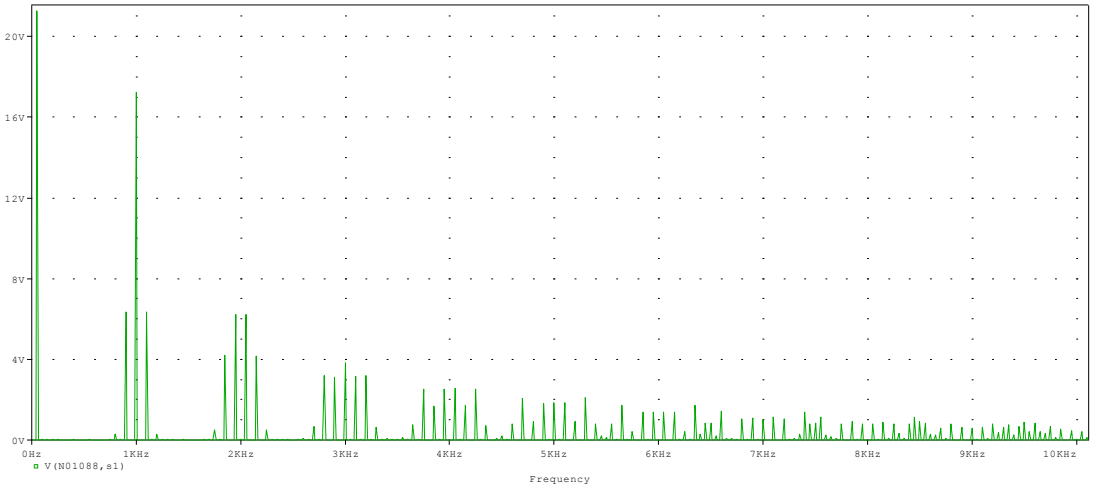


Figure 11 Fourier series Voltage of Inverter PSpice S_2.

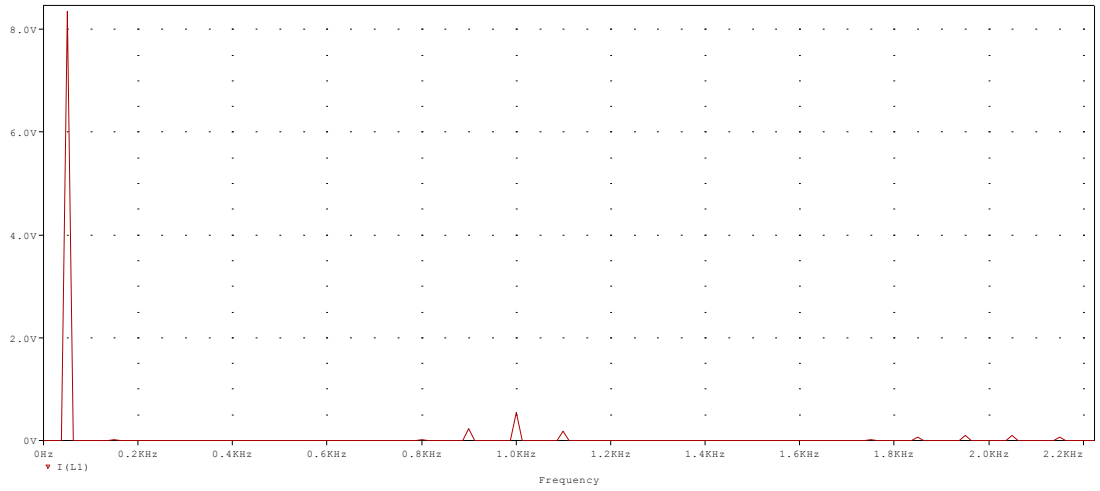


Figure 12 Fourier series Current of Inverter PSpice S_2.

2.3.3 S_3 - Asymmetric Inverter (Square Wave Control)

The following image represents the inverter circuit:

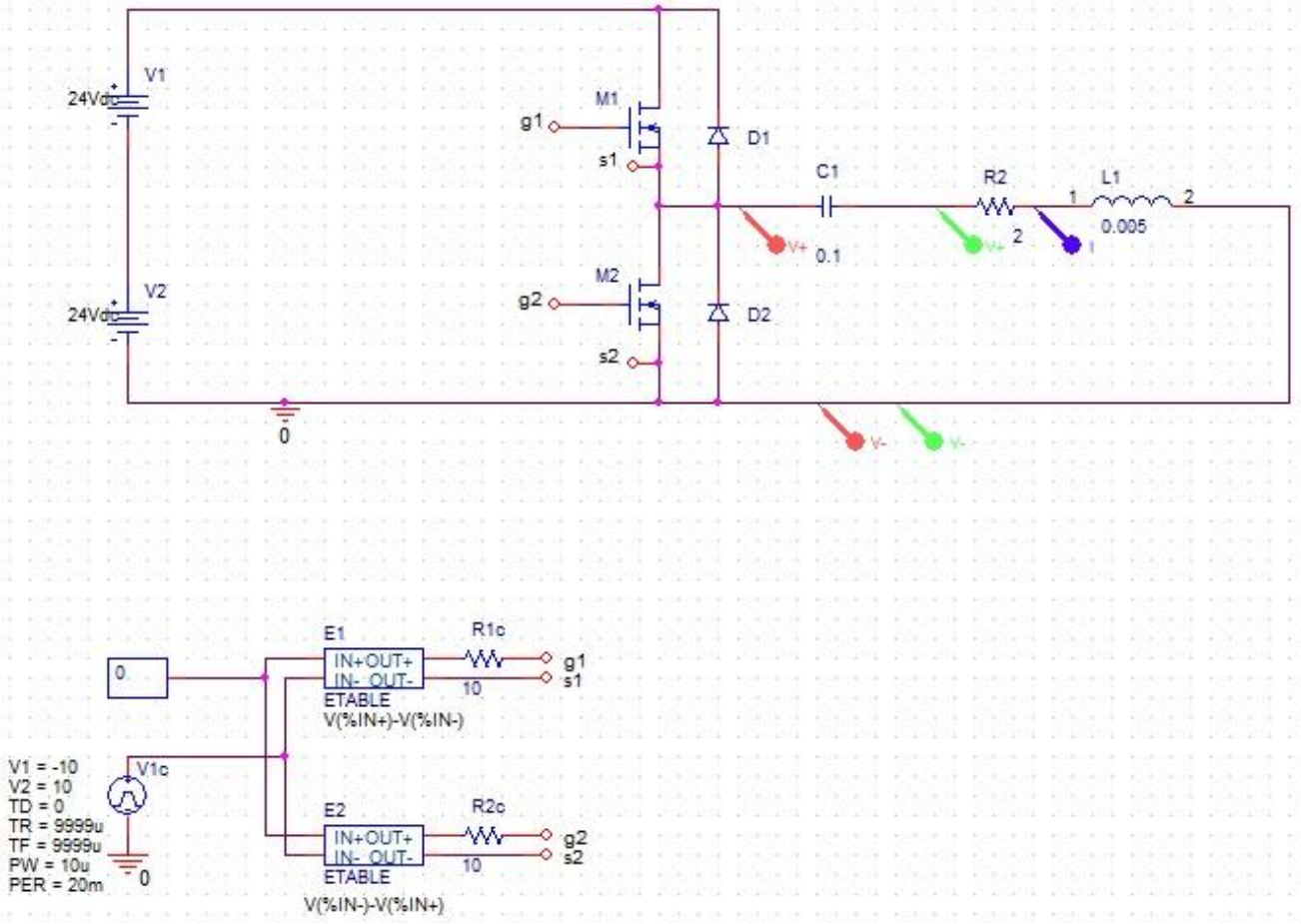


Figure 13 Inverter Scheme PSpice S_3.

The first graph corresponds to the waveform of the output at the load. The blue line represents the Current that runs through load and green line represents the Voltage in it. Red line corresponds to the voltage of the output load with the filtering capacitor:

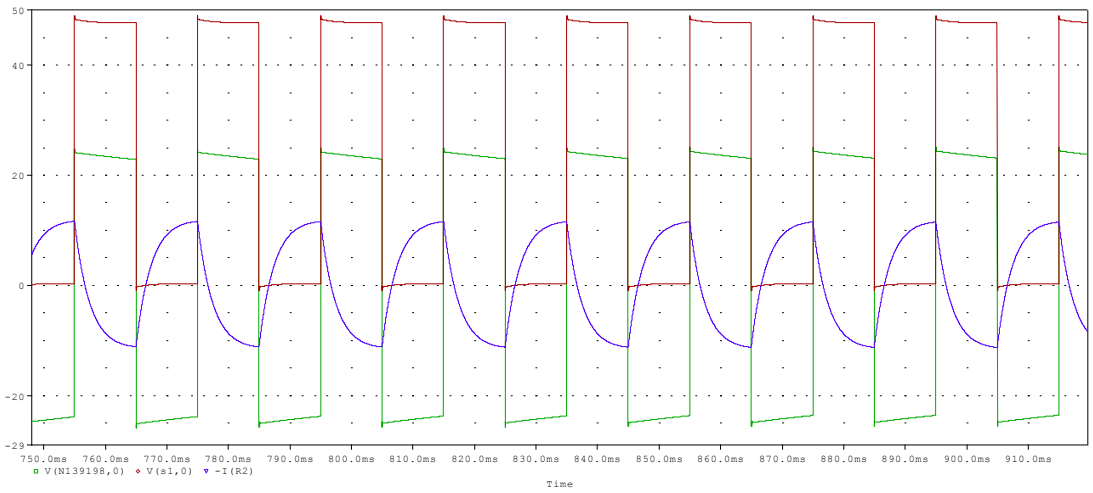


Figure 14 Output Inverter PSpice S_3.

The following graphs correspond to Fourier analysis. They represent frequency analysis of the harmonics of each waveform. Each graph goes with its own color line:

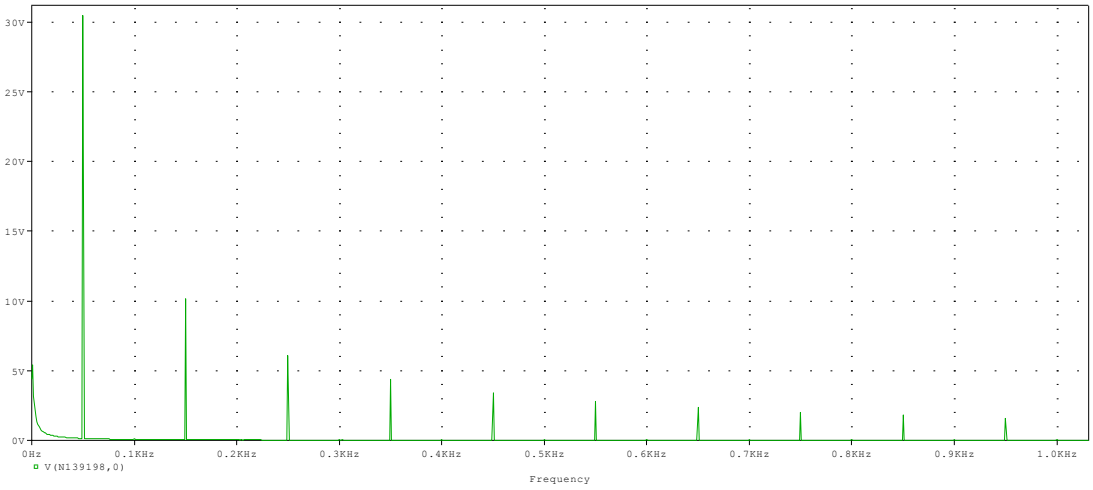


Figure 15 Fourier series Voltage (after Capacitor) of Inverter PSpice S_3.

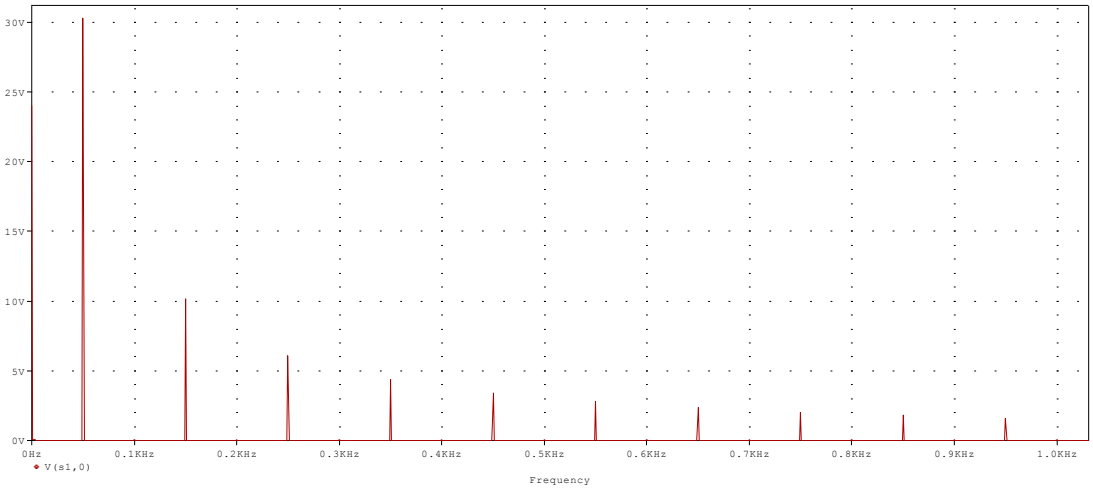


Figure 16 Fourier series Voltage (before Capacitor) of Inverter PSpice S_3.

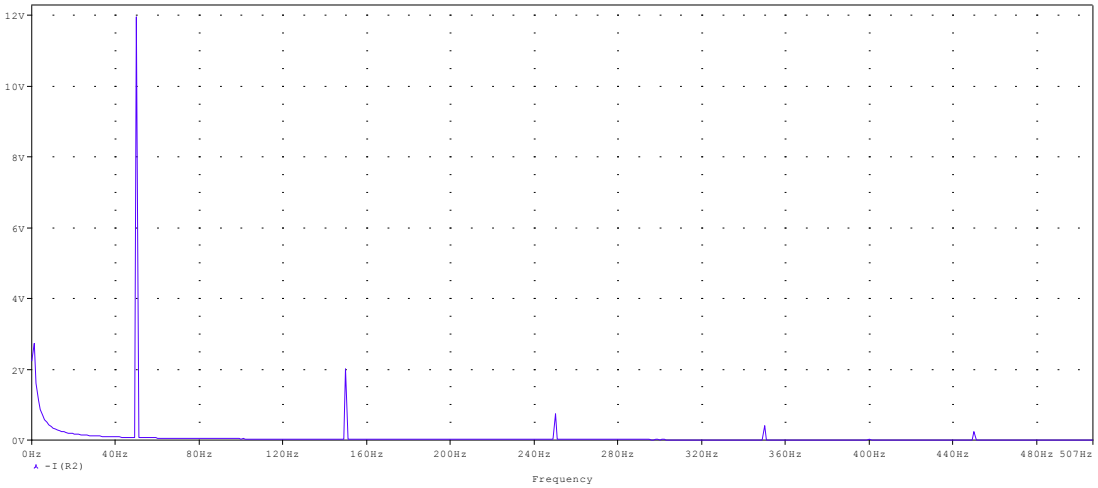


Figure 17 Fourier series Current of Inverter PSpice S_3.

2.3.4 S_4 - Asymmetric Inverter (PWM Control)

The following image represents the inverter circuit:

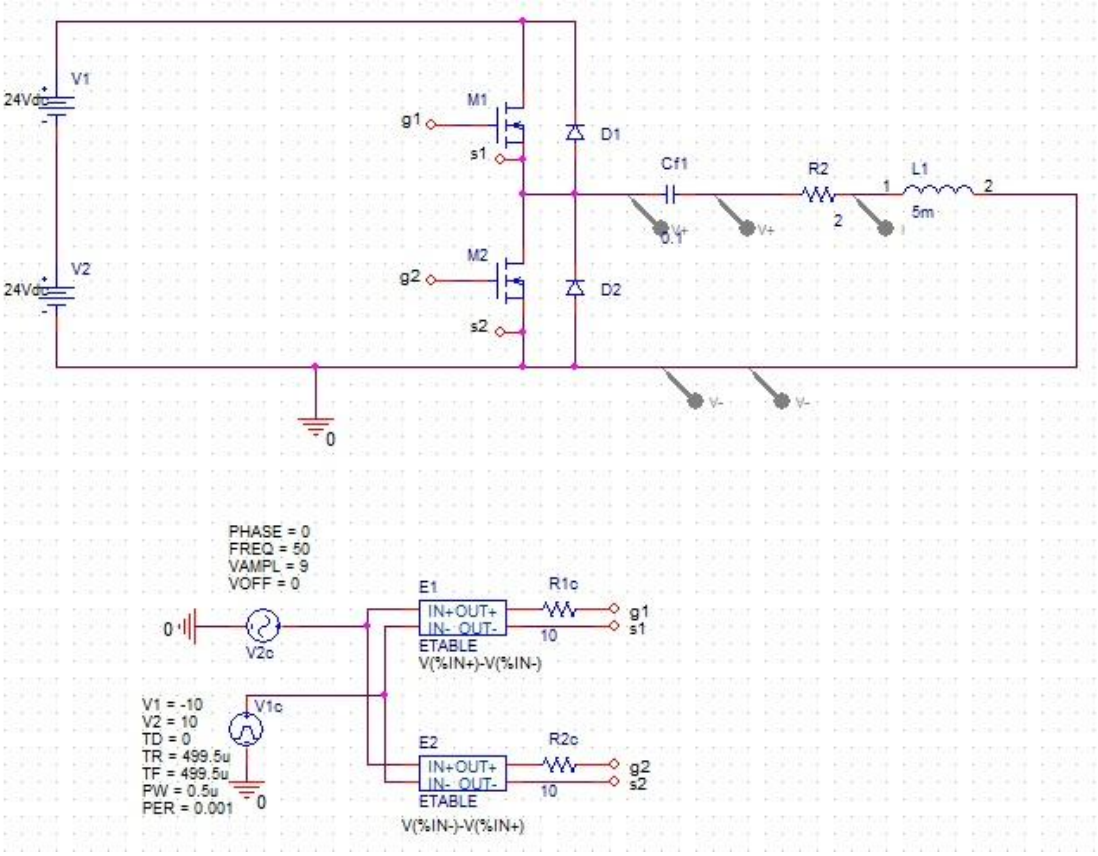


Figure 18 Inverter Scheme PSpice S_4.

The first graph corresponds to the waveform of the output at the load. The blue line represents the Current that runs through load and green line represents the Voltage in it. Red line corresponds to the voltage of the output load with the filtering capacitor:

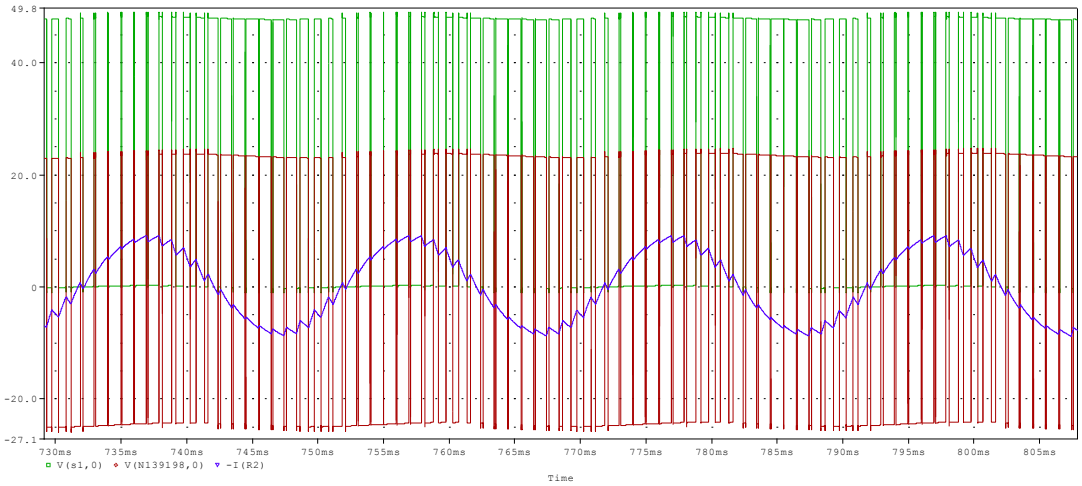


Figure 19 Output Inverter PSpice S_4.

The following graphs correspond to Fourier analysis. They represent frequency analysis of the harmonics of each waveform. Each graph goes with its own color line:

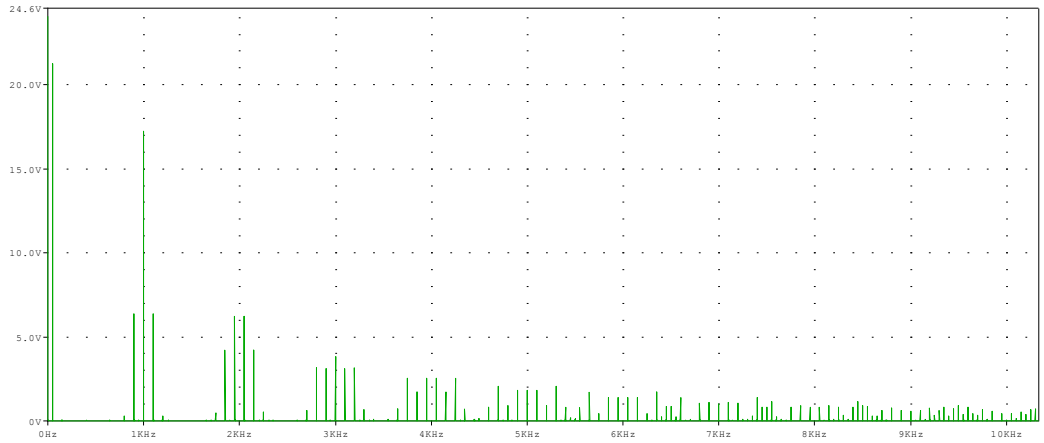


Figure 20 Fourier series Voltage (after Capacitor) of Inverter PSpice S_4.

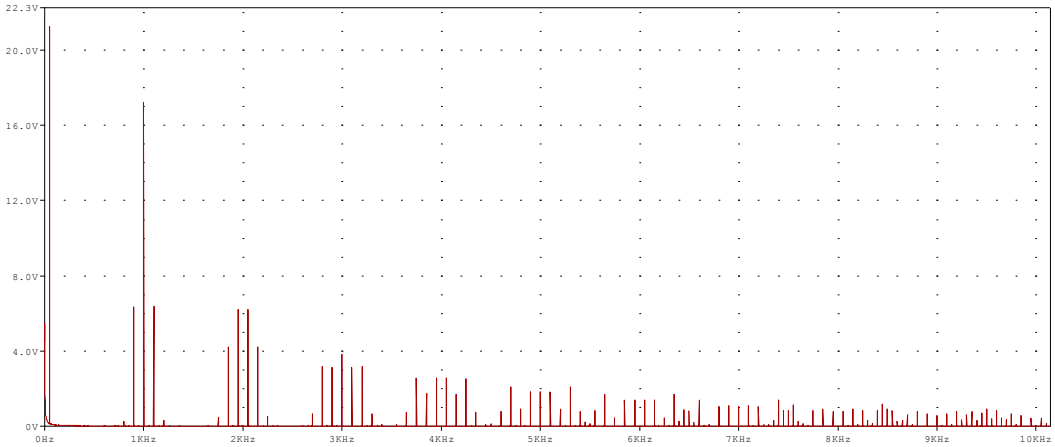


Figure 21 Fourier series Voltage (before Capacitor) of Inverter PSpice S_4.

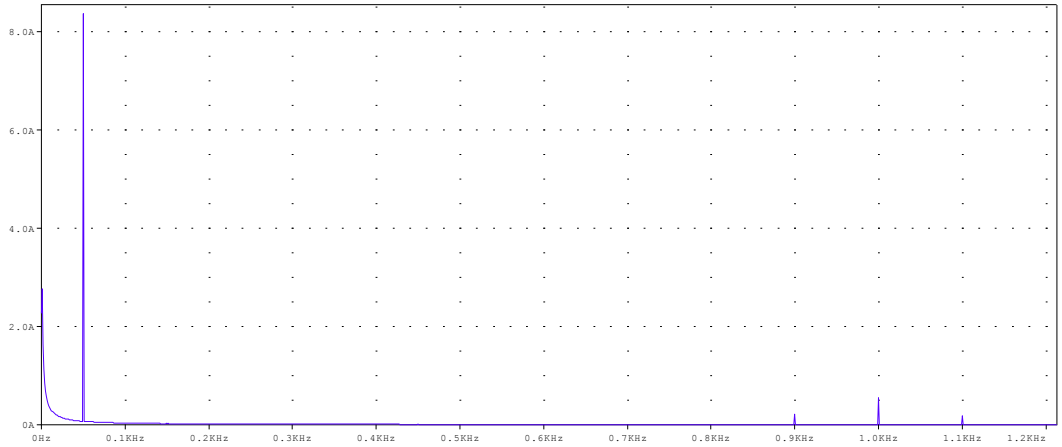


Figure 22 Fourier series Current of Inverter PSpice S_4.

2.3.5 S_5 - Full Wave H-Bridge Inverter (Square Wave Control)

The following image represents the inverter circuit:

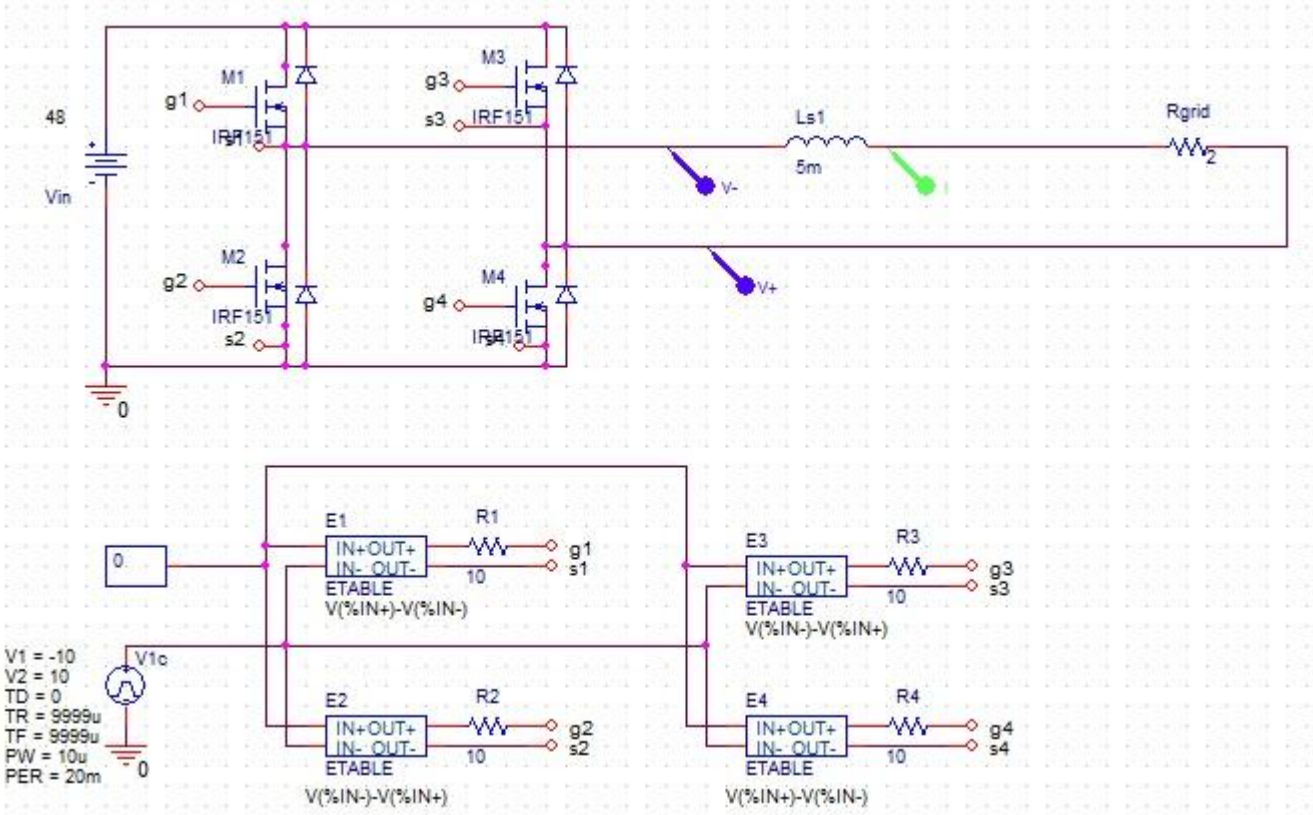


Figure 23 Inverter Scheme PSpice S_5.

The first graph corresponds to the waveform of the output at the load. The green line represents the Current that runs through load and blue line represents the Voltage in it:

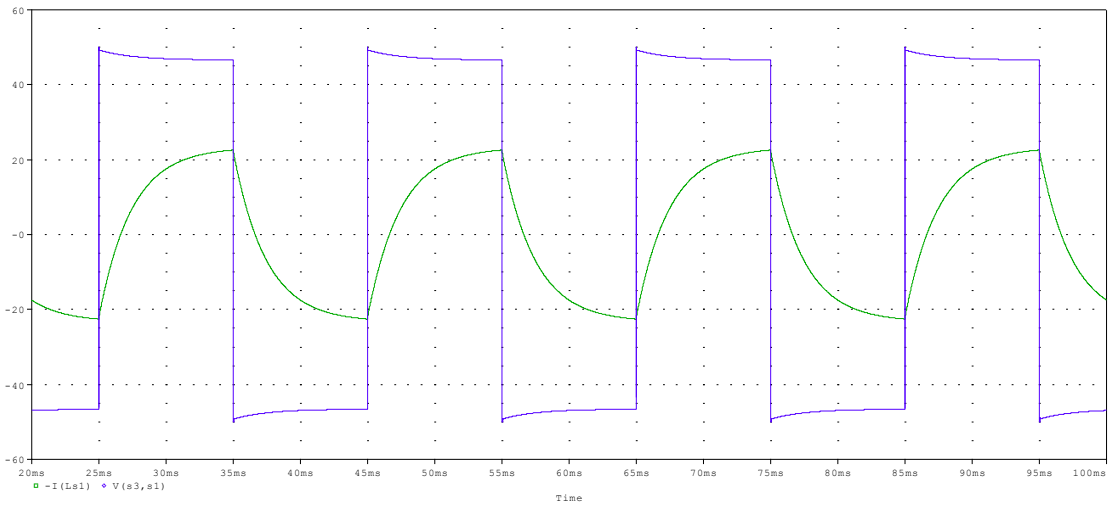


Figure 24 Output Inverter PSpice S_5.

The following graphs correspond to Fourier analysis. They represent frequency analysis of the harmonics of each waveform. Each graph goes with its own color line:

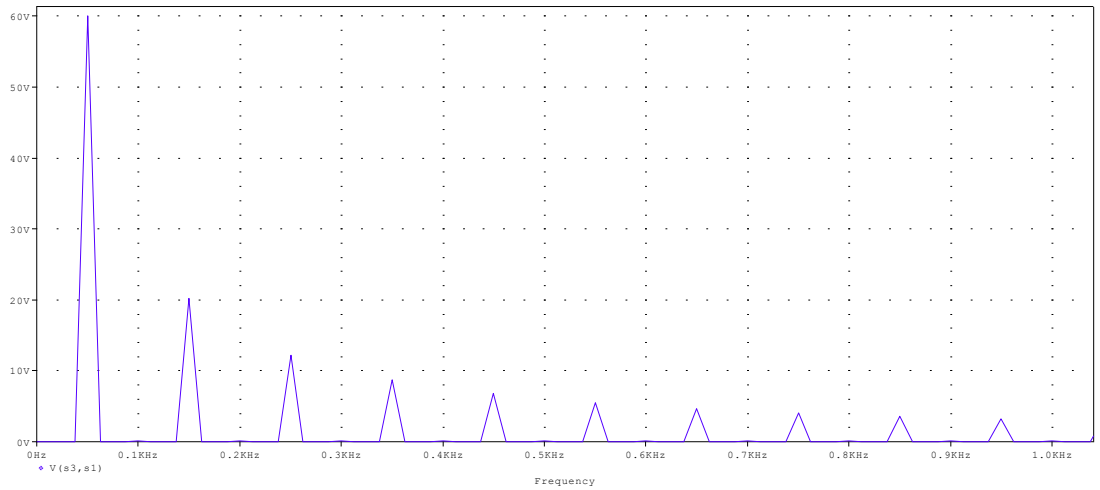


Figure 25 Fourier series Voltage of Inverter PSpice S_5.

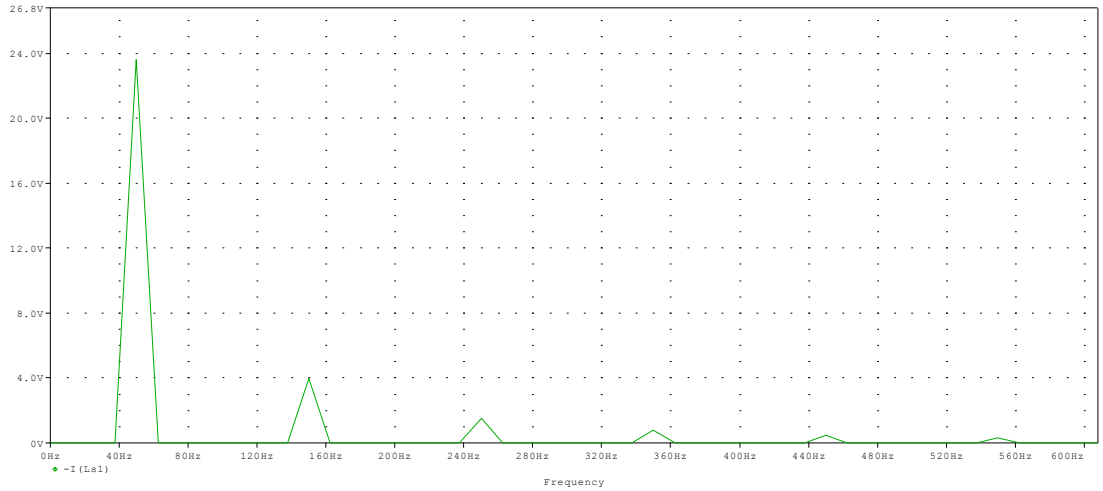


Figure 26 Fourier series Current of Inverter PSpice S_5.

2.3.6 S_6 - Full Wave H-Bridge Inverter (Voltage Harmonic Cancellation)

The following image represents the inverter circuit:

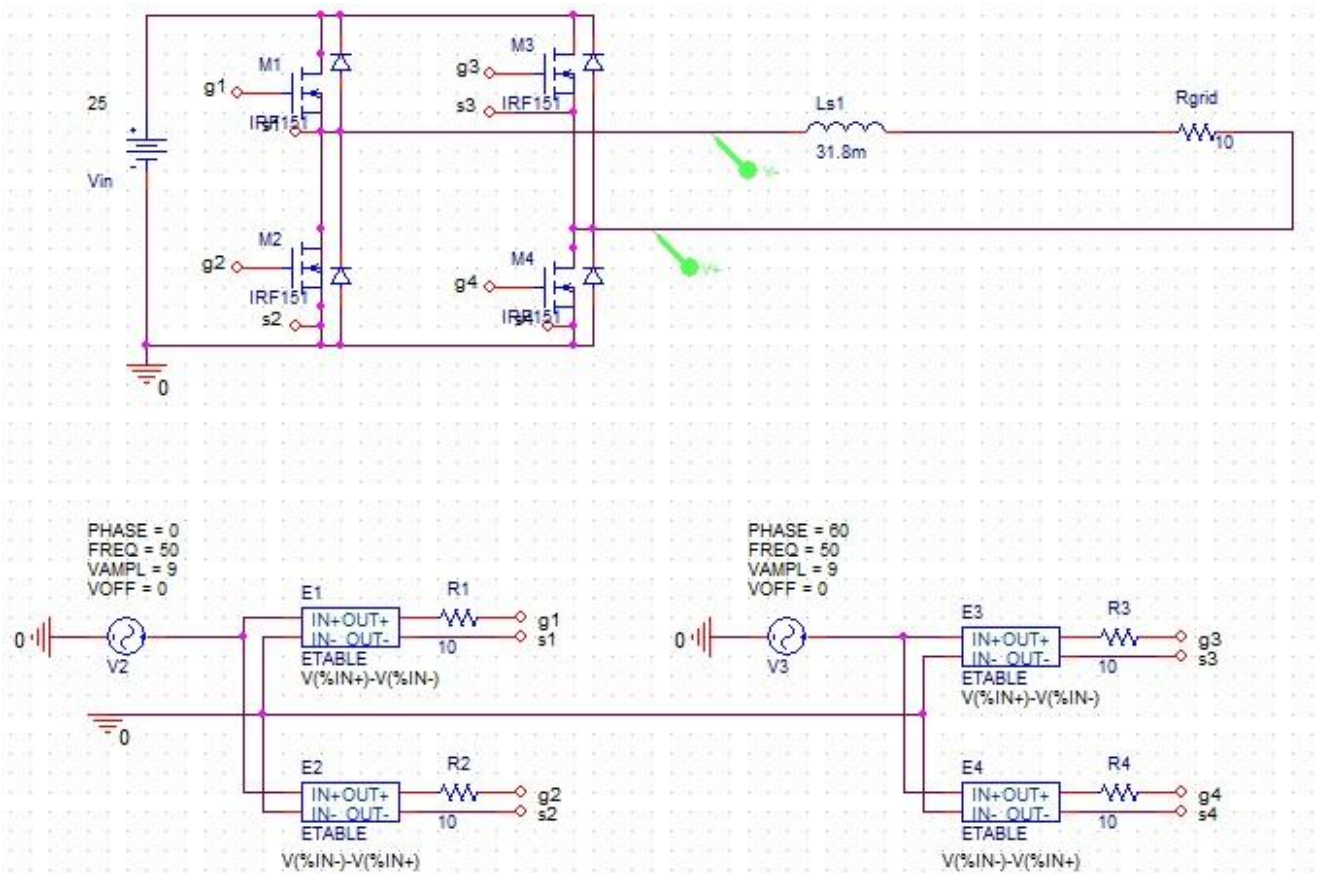


Figure 27 Inverter Scheme PSpice S_6.

The first graph corresponds to the waveform of the output at the load. The red line represents the Current that runs through load and green line represents the Voltage in it:

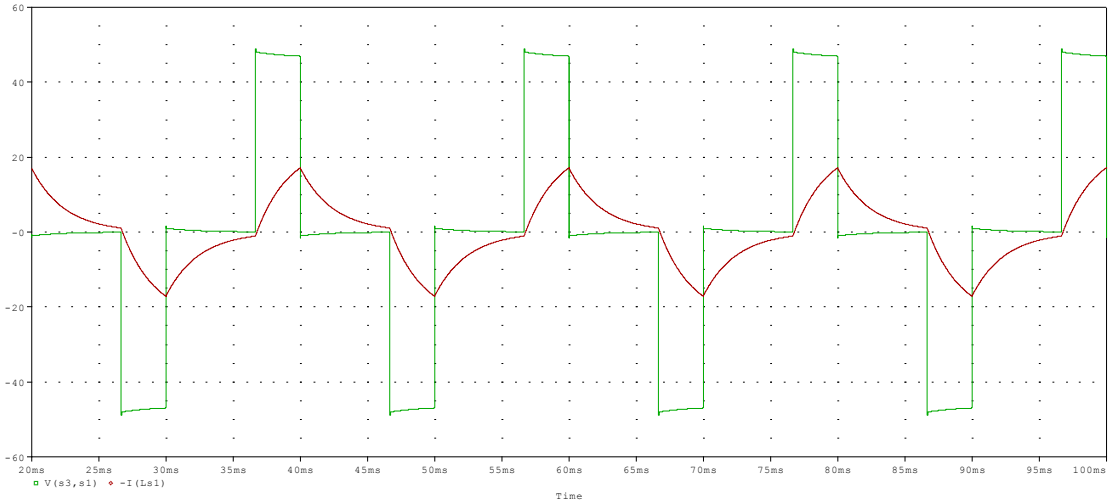


Figure 28 Output Inverter PSpice S_6.

The following graphs correspond to Fourier analysis. They represent frequency analysis of the harmonics of each waveform. Each graph goes with its own color line:

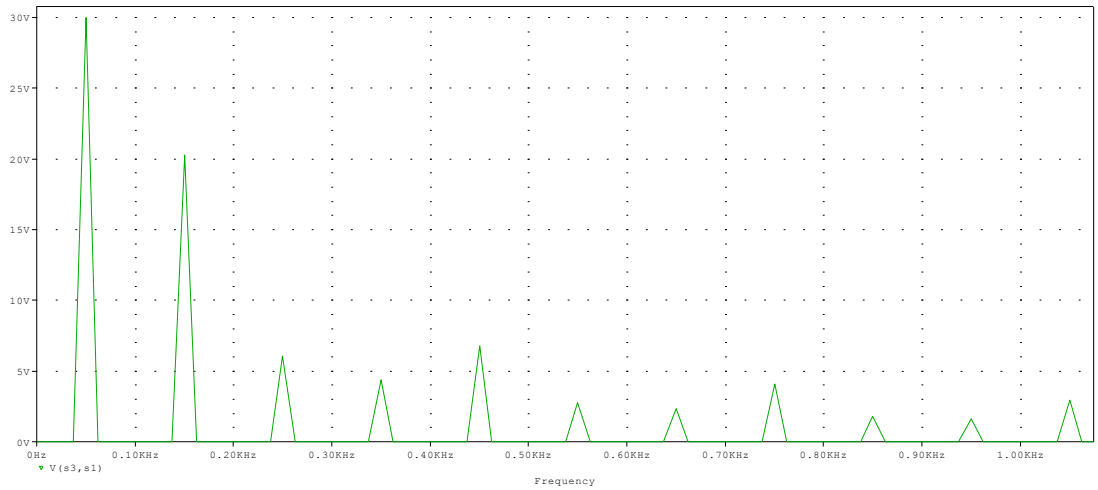


Figure 29 Fourier series Voltage of Inverter PSpice S_6.

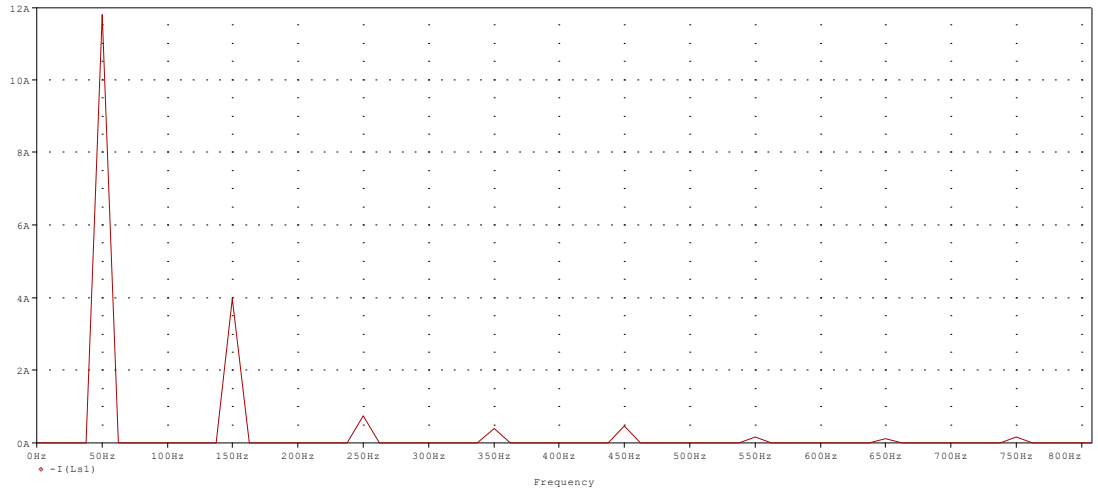


Figure 30 Fourier series Current of Inverter PSpice S_6.

2.3.7 S_7 - Full Wave H-Bridge Inverter (Bipolar PWM Control)

The following image represents the inverter circuit:

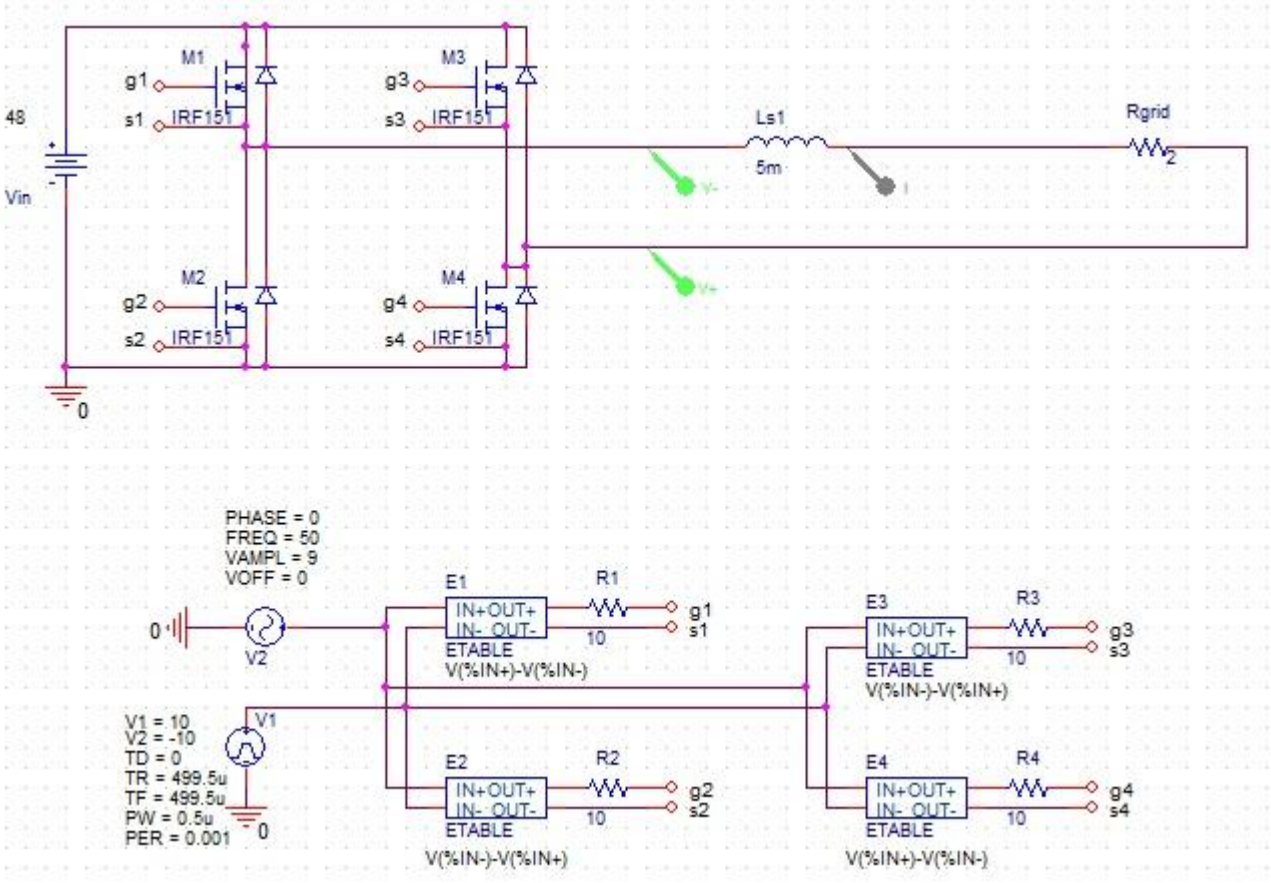


Figure 31 Inverter Scheme PSpice S_7.

The first graph corresponds to the waveform of the output at the load. The red line represents the Current that runs through load and green line represents the Voltage in it:

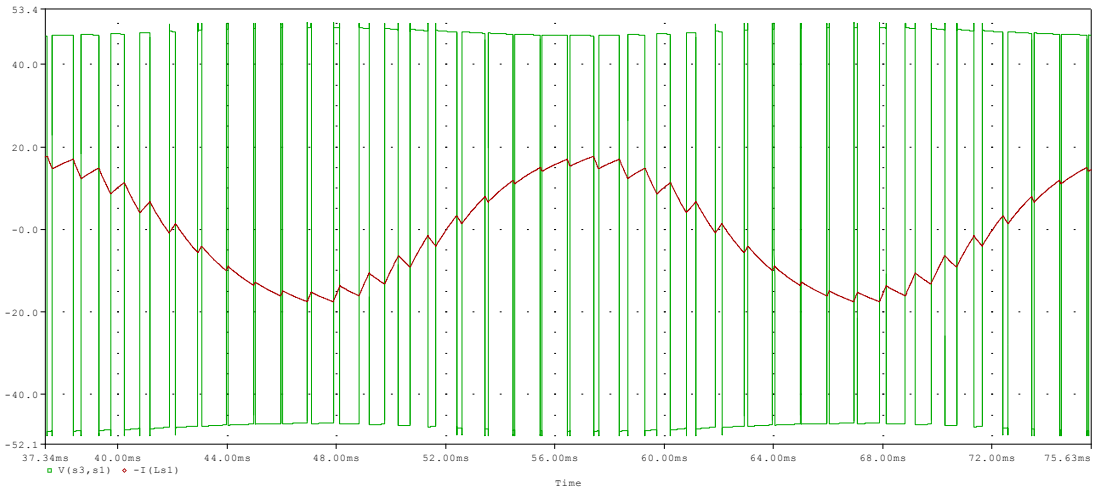


Figure 32 Output Inverter PSpice S_7.

The following graphs correspond to Fourier analysis. They represent frequency analysis of the harmonics of each waveform. Each graph goes with its own color line:

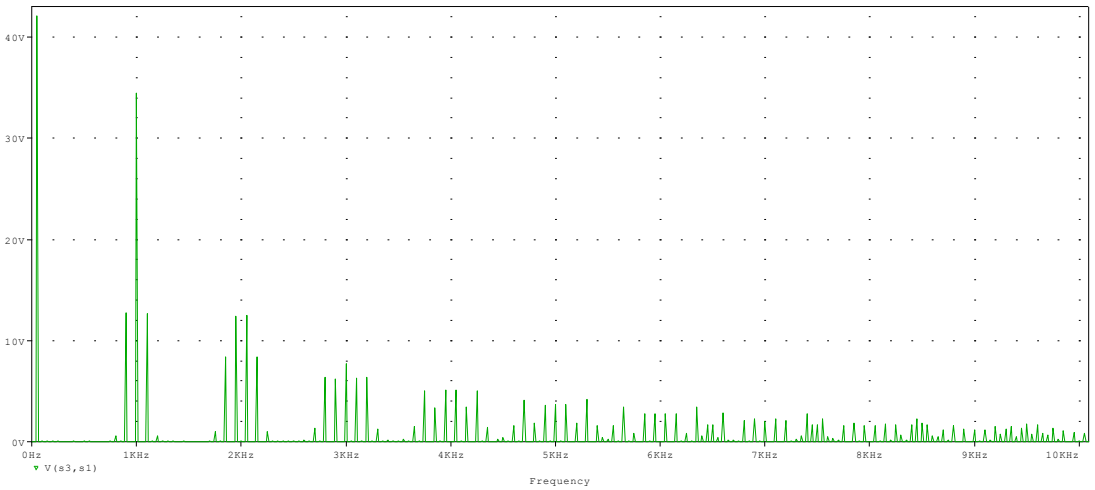


Figure 33 Fourier series Voltage of Inverter PSpice S_7.

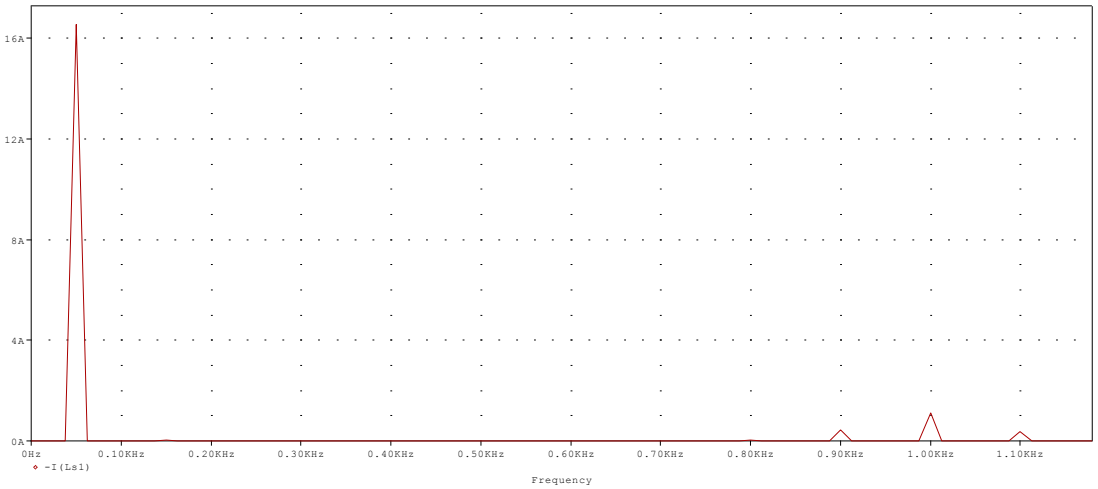


Figure 34 Fourier series Current of Inverter PSpice S_7.

2.3.8 S_8 - Full Wave H-Bridge Inverter (Single-Pole PWM Control)

The following image represents the inverter circuit:

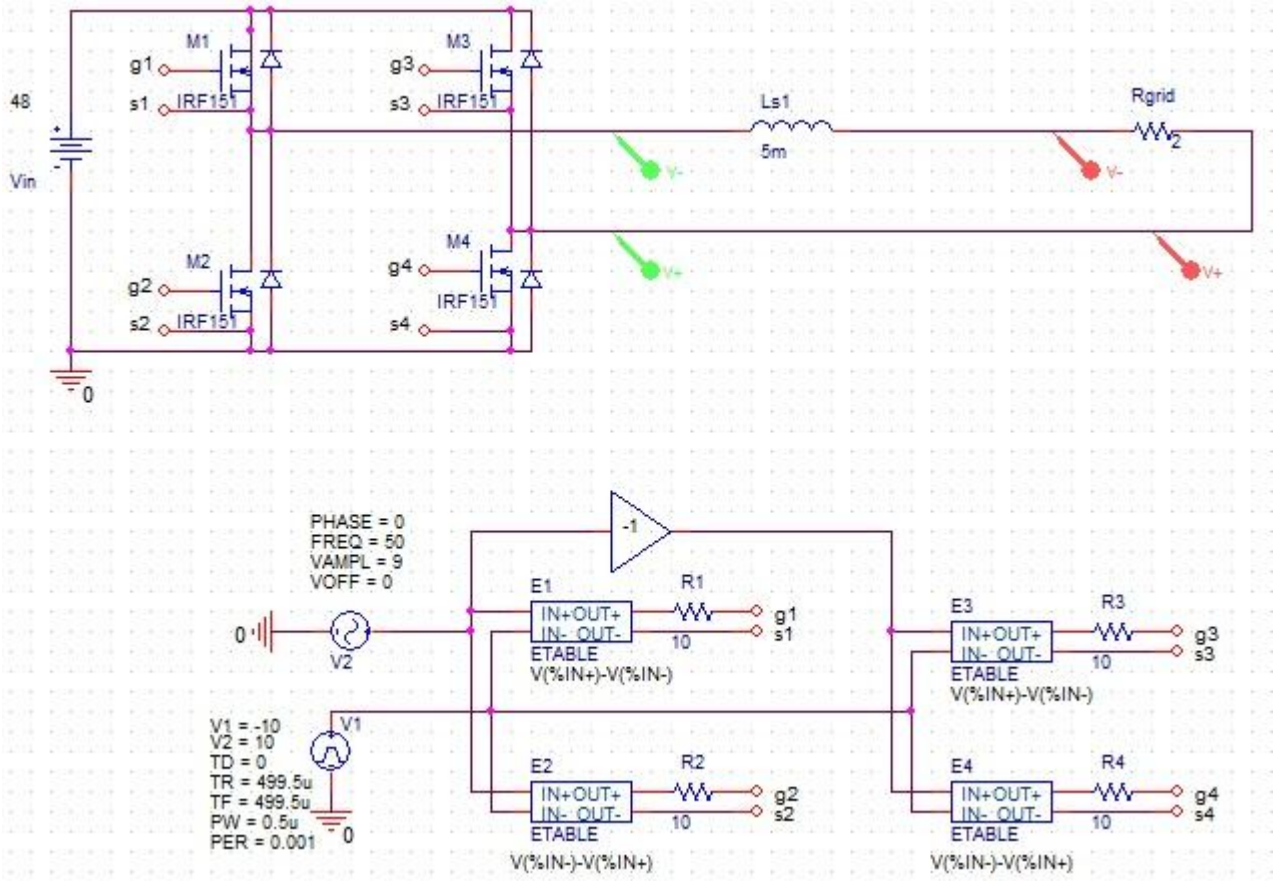


Figure 35 Inverter Scheme PSpice S_8.

The first graph corresponds to the waveform of the output at the load. The red line represents the Current that runs through load and green line represents the Voltage in it:

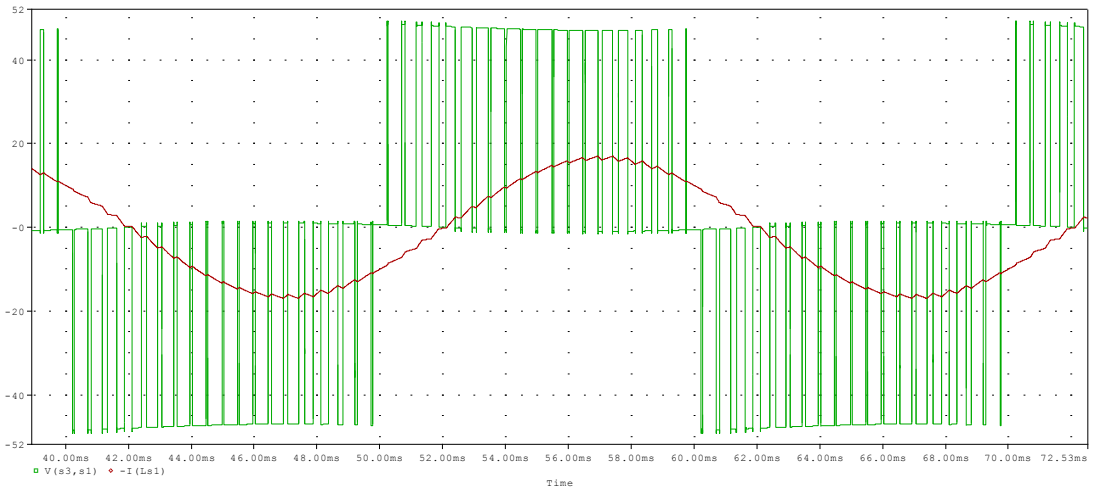


Figure 36 Output Inverter PSpice S_8.

The following graphs correspond to Fourier analysis. They represent frequency analysis of the harmonics of each waveform. Each graph goes with its own color line:

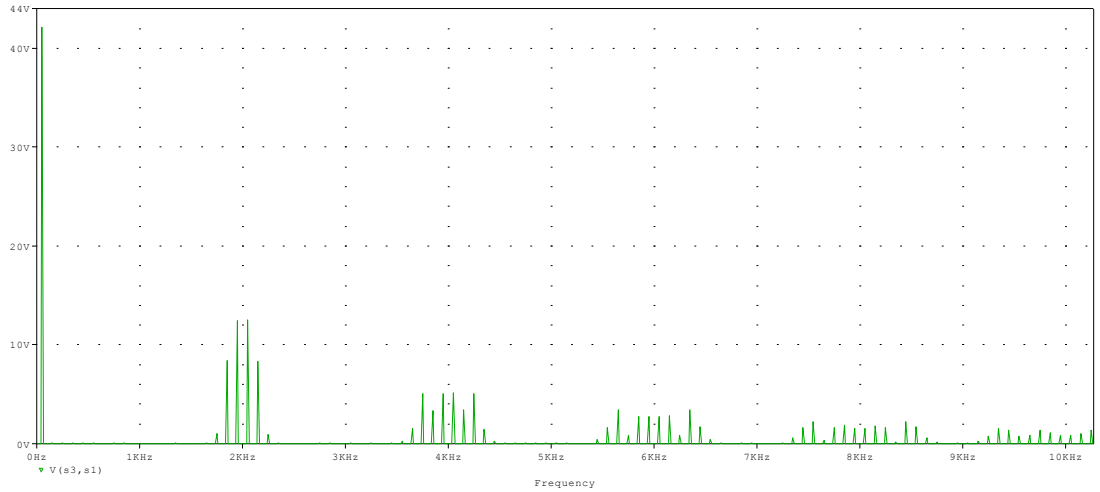


Figure 37 Fourier series Voltage of Inverter PSpice S_8.

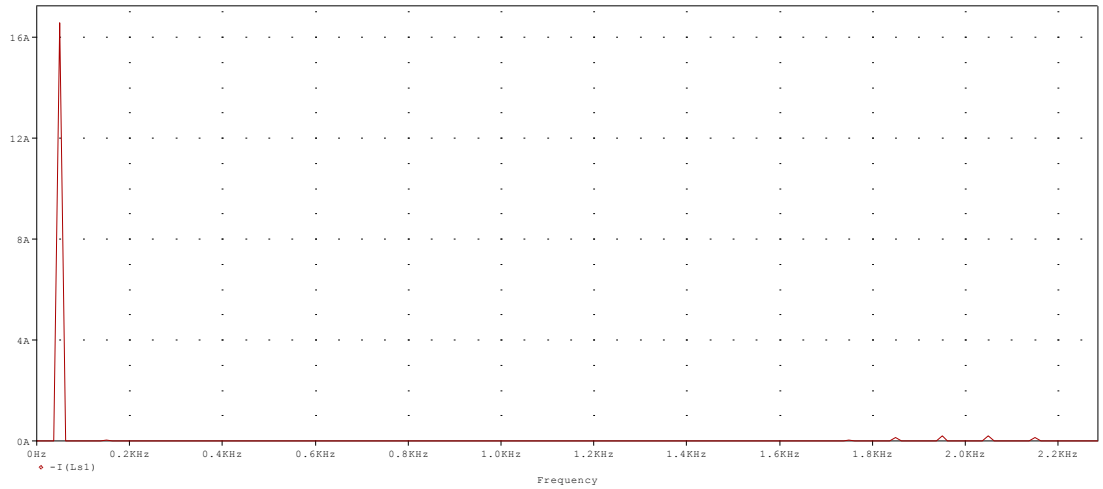


Figure 38 Fourier series Current of Inverter PSpice S_8.

2.3.9 S_9 - Three-Phase VSI 6-Step Inverter (Star Load) (Square Wave Control)

The following image represents the inverter circuit:

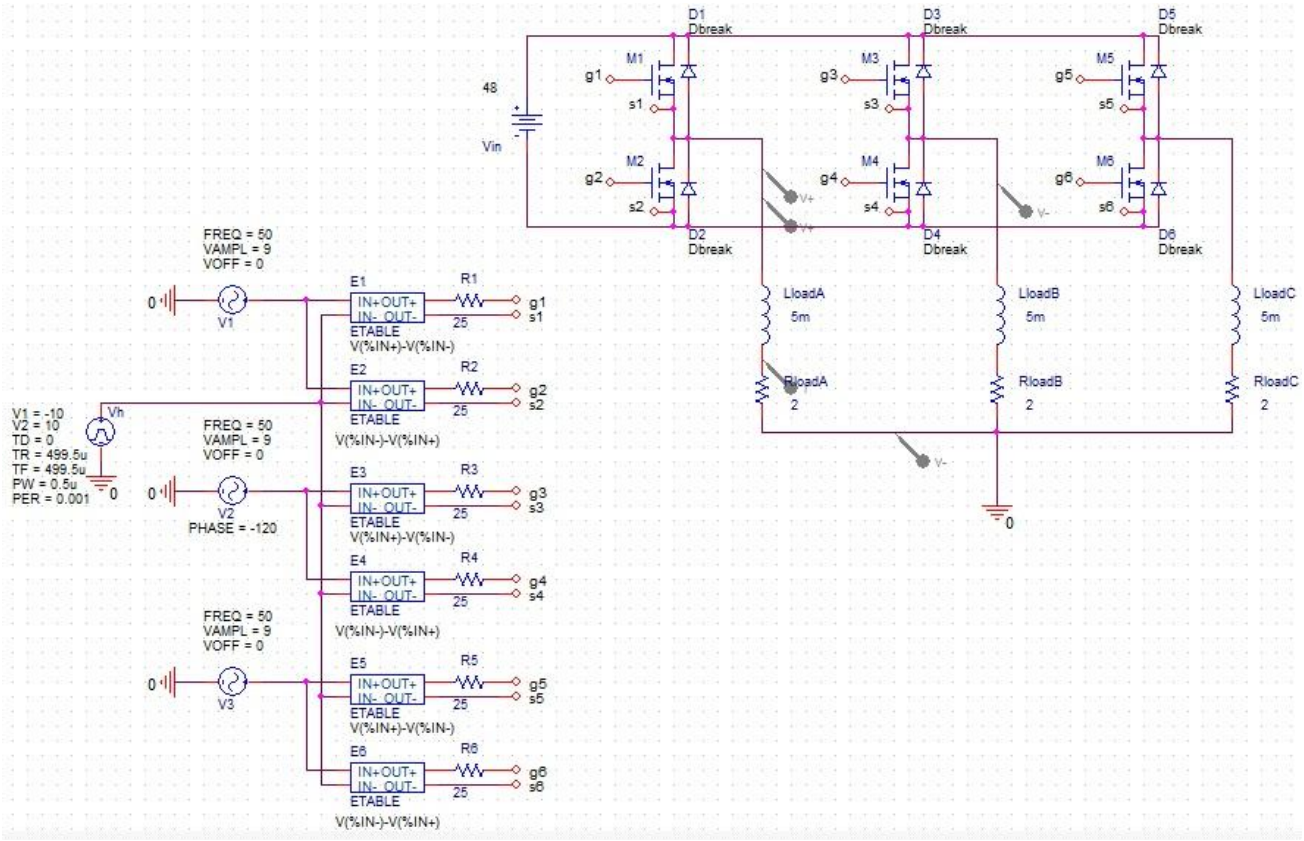


Figure 39 Inverter Scheme PSpice S_9.

Graph of the waveform at the output load. The red line represents the Current that runs through load, blue line represents the Voltage (line) and green line is the Voltage (phase) in it:

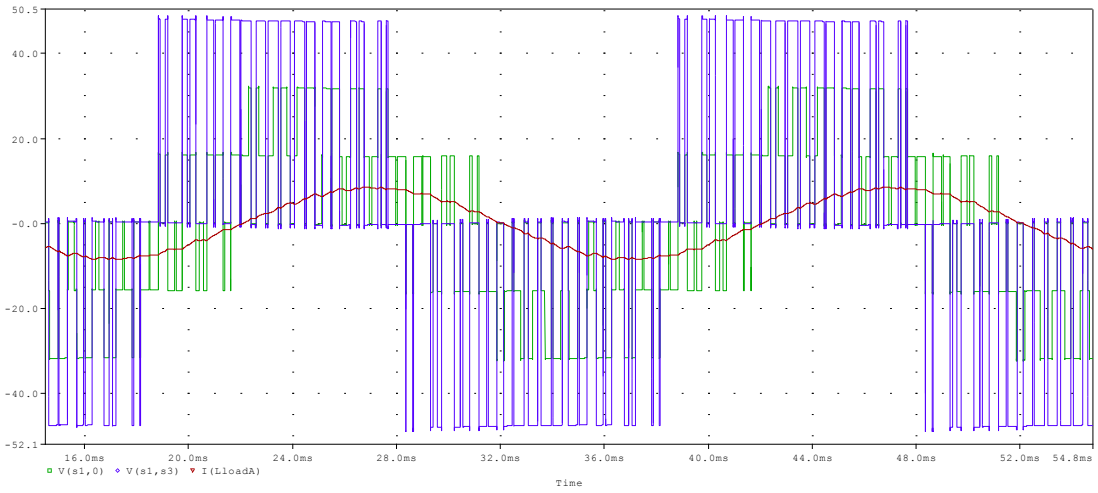


Figure 40 Output Inverter PSpice S_9.

The following graphs correspond to Fourier analysis. They represent frequency analysis of the harmonics of each waveform. Each graph goes with its own color line:

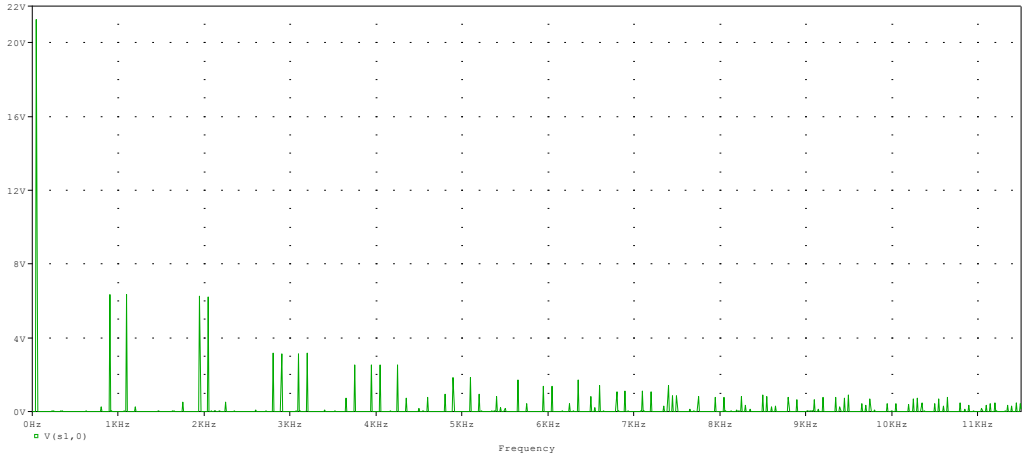


Figure 41 Fourier series Phase Voltage of Inverter PSpice S_9.

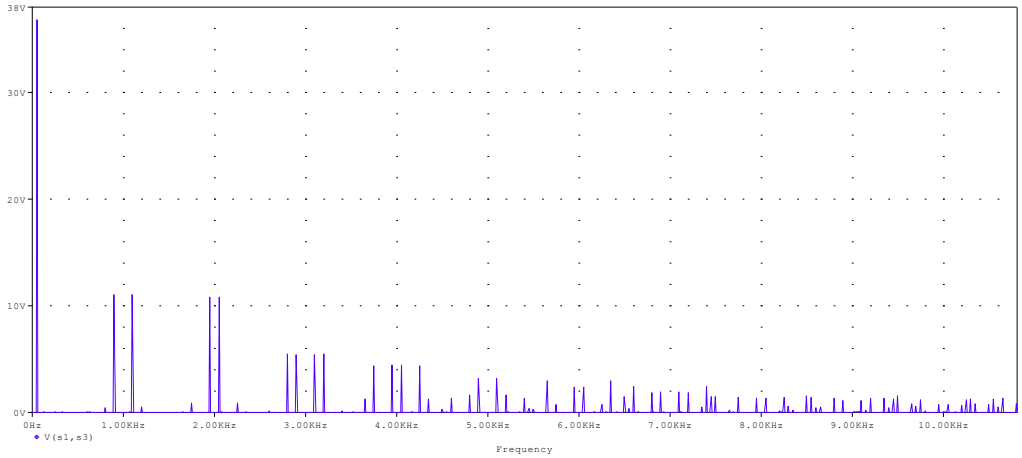


Figure 42 Fourier series Line Voltage of Inverter PSpice S_9.

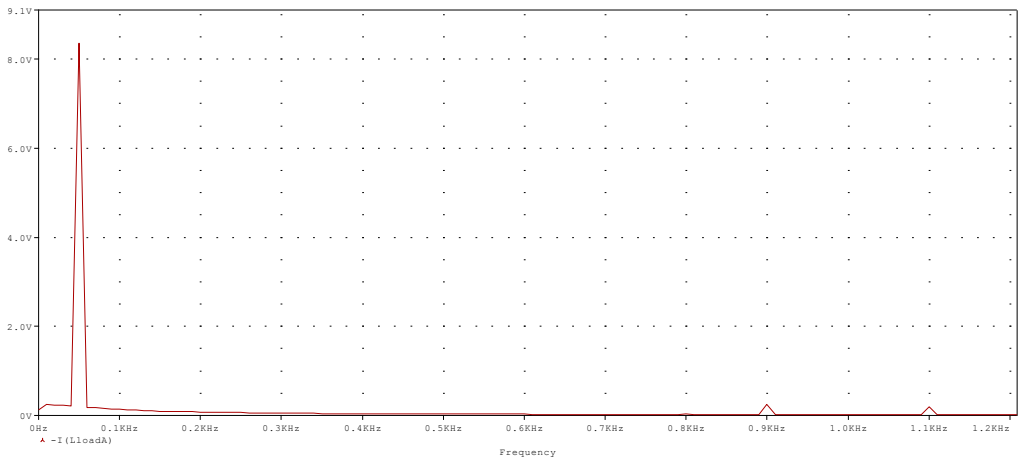


Figure 43 Fourier series Current of Inverter PSpice S_10.

2.3.10 S_10 - Three-Phase VSI 6-Step Inverter (Triangle Load) (Square Wave Control)

The following image represents the inverter circuit:

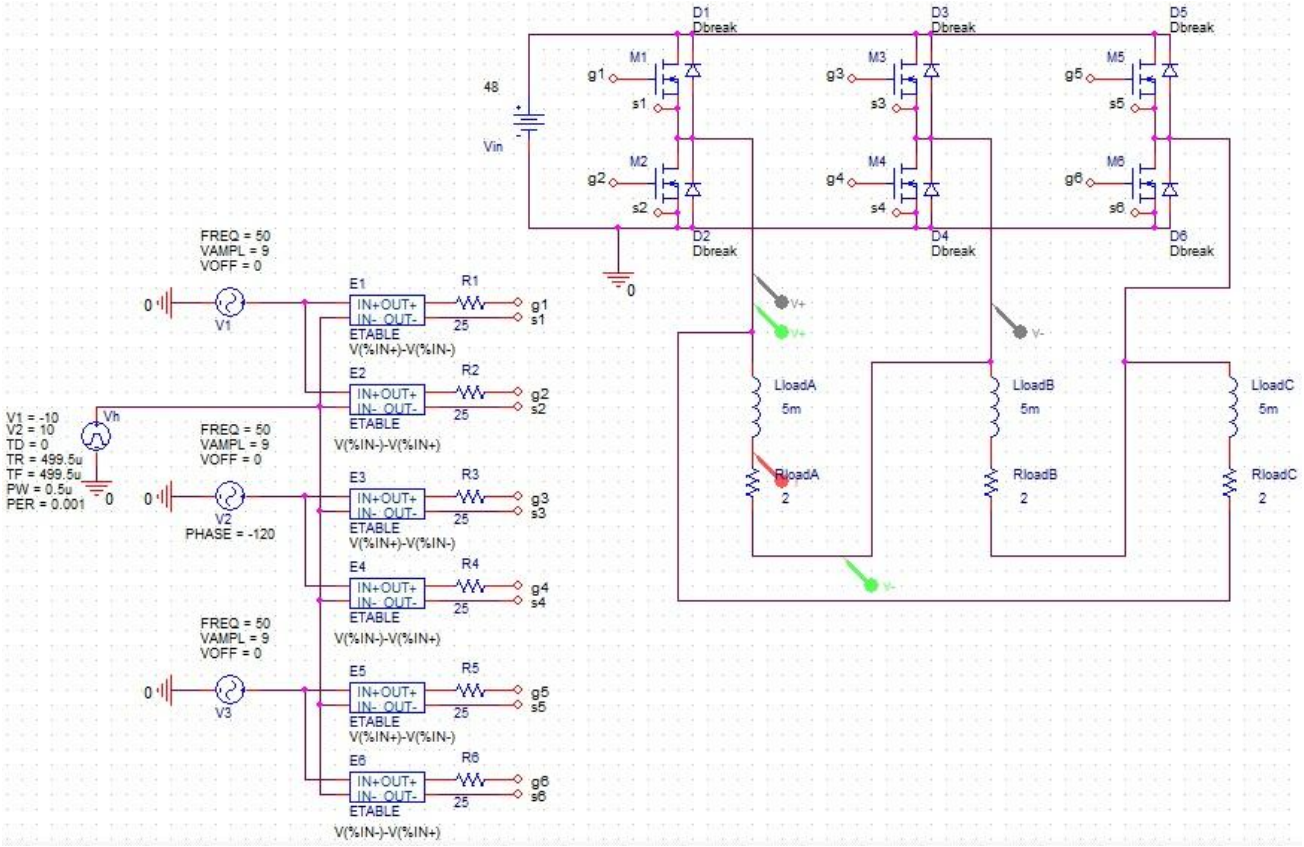


Figure 44 Inverter Scheme PSpice S_10.

Graph of the waveform at the output load. The blue line represents the Current (line) that runs through load and green line represents the Voltage (Line and phase) and red line is the Current (phase) that runs through load:

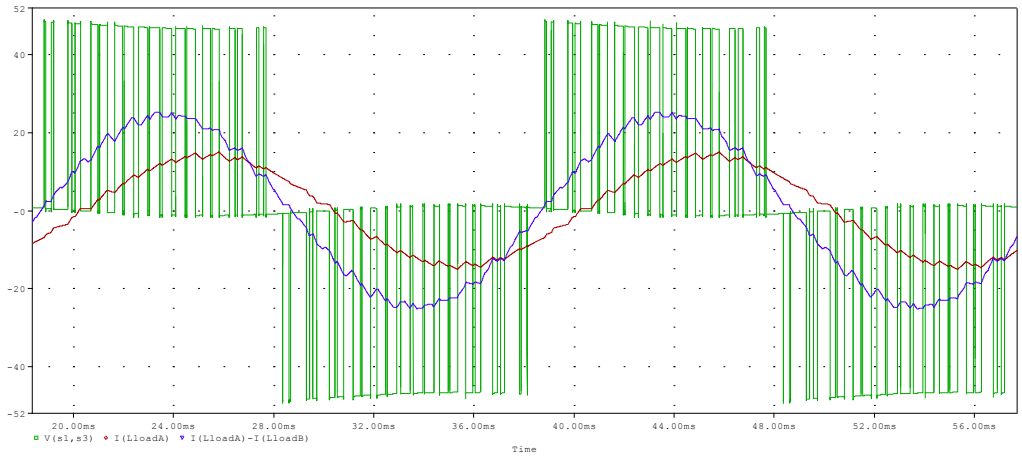


Figure 45 Output Inverter PSpice S_10.

The following graphs correspond to Fourier analysis. They represent frequency analysis of the harmonics of each waveform. Each graph goes with its own color line:

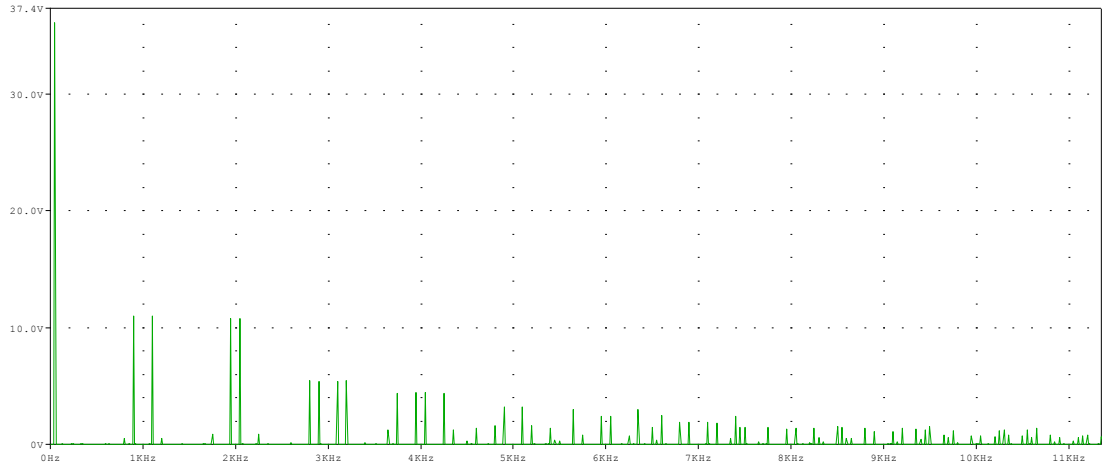


Figure 46 Fourier series Voltage of Inverter PSpice S_10.

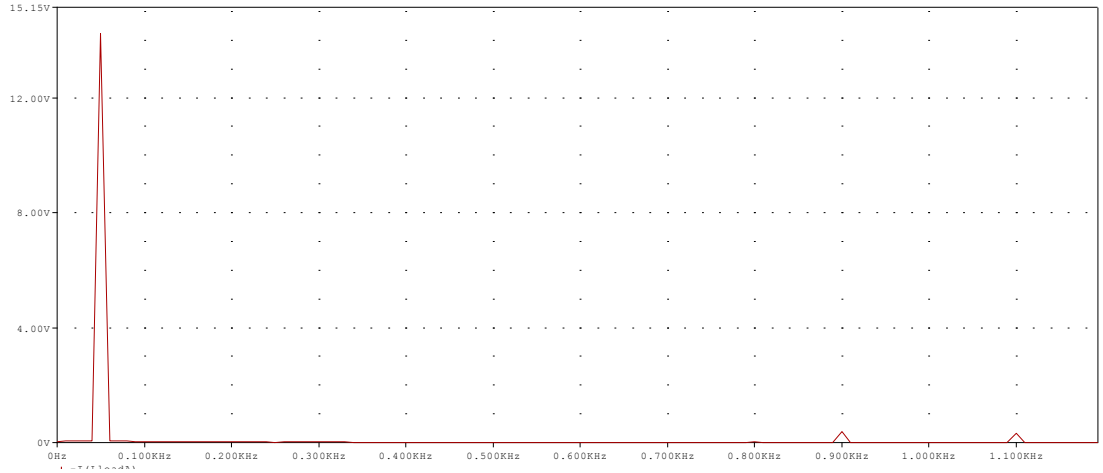


Figure 47 Fourier series Phase Current of Inverter PSpice S_10.

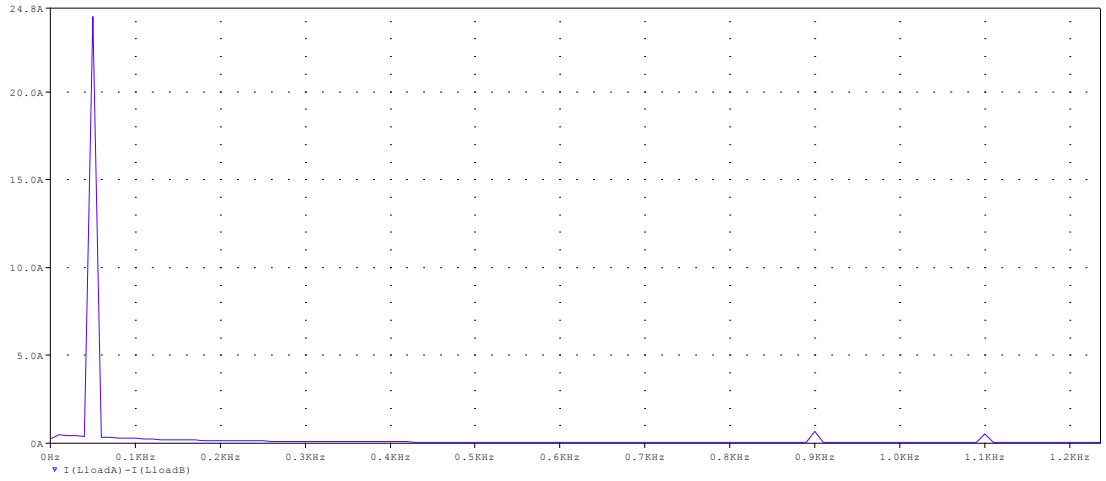


Figure 48 Fourier series Line Current of Inverter PSpice S_10.

2.3.11 S_11 -Three-Phase VSI 6-Step Inverter (Star Load) (PWM Control)

The following image represents the inverter circuit:

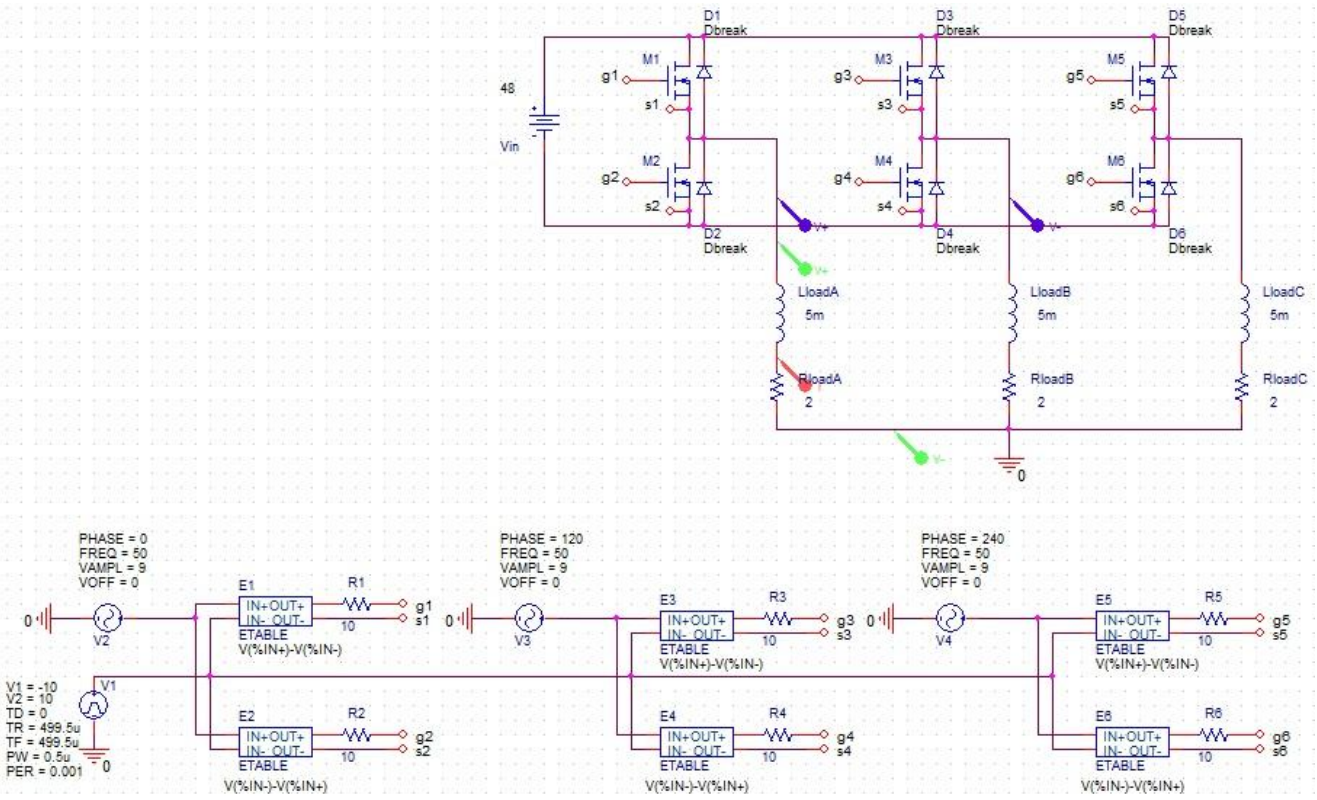


Figure 49 Inverter Scheme PSpice S_11.

Graph of the waveform at the output load. The red line represents the Current that runs through load, blue line represents the Voltage (line) and green line is the Voltage (phase) in it:

Figure 50 Output Inverter PSpice S_11.

The following graphs correspond to Fourier analysis. They represent frequency analysis of the harmonics of each waveform. Each graph goes with its own color line:

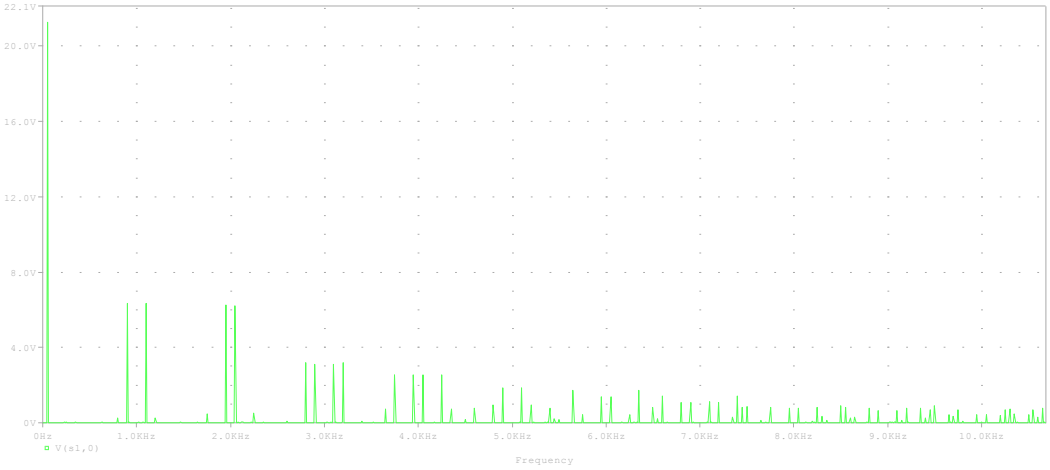


Figure 51 Fourier series Phase Voltage of Inverter PSpice S_11.

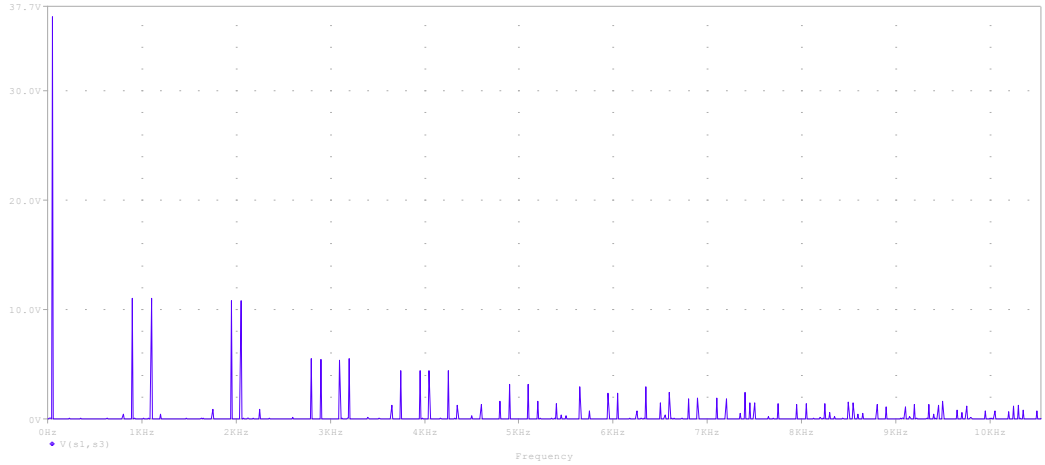


Figure 52 Fourier series Line Voltage of Inverter PSpice S_11.

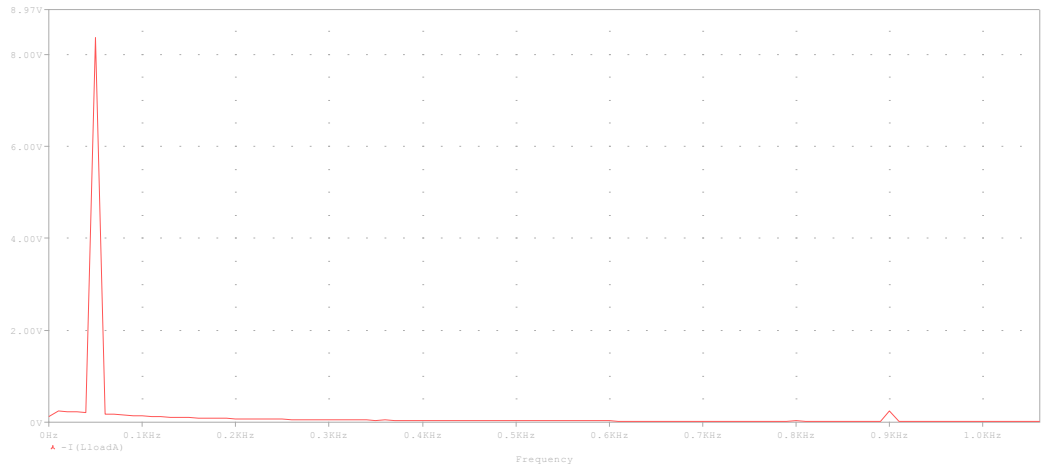


Figure 53 Fourier series Current of Inverter PSpice S_11.

2.3.12 S_12 - Three-Phase VSI 6-Step Inverter (Triangle Load) (PWM Control)

The following image represents the inverter circuit:

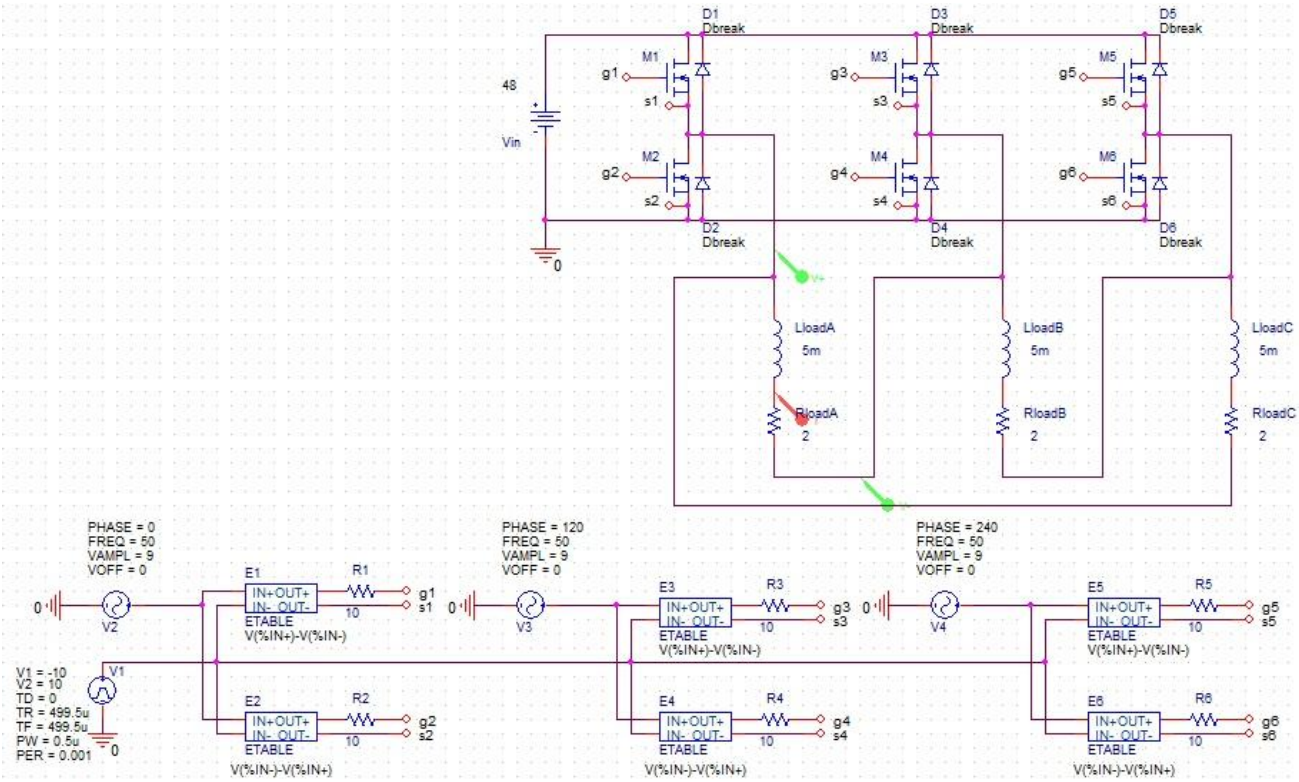


Figure 54 Inverter Scheme PSpice S_12.

Graph of the waveform at the output load. The blue line represents the Current (line) that runs through load and green line represents the Voltage (Line and phase) and red line is the Current (phase) that runs through load:

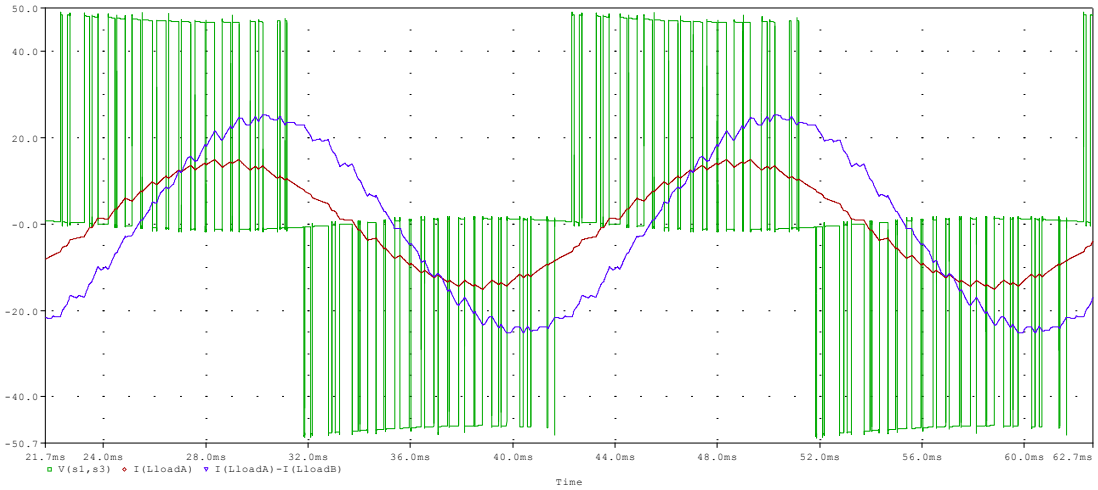


Figure 55 Output Inverter PSpice S_12.

The following graphs correspond to Fourier analysis. They represent frequency analysis of the harmonics of each waveform. Each graph goes with its own color line:

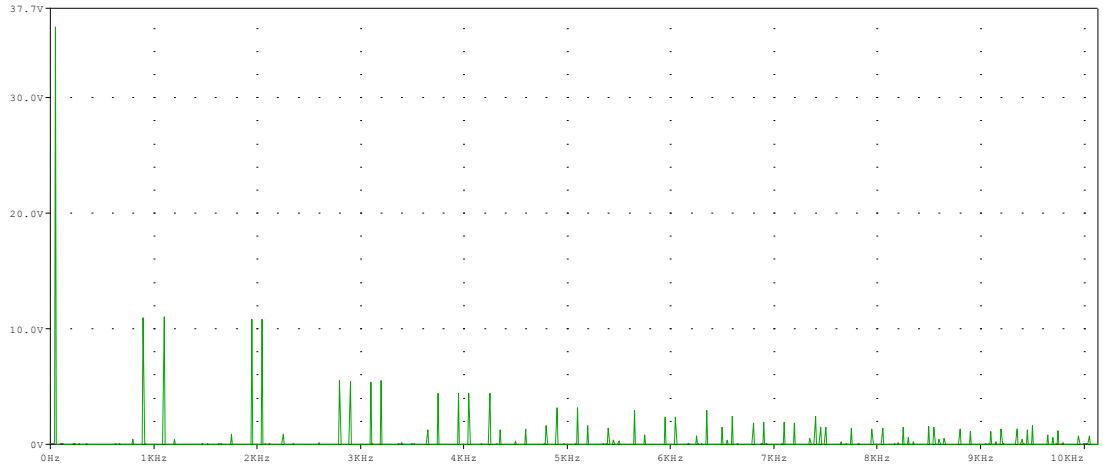


Figure 56 Fourier series Voltage of Inverter PSpice S_12.

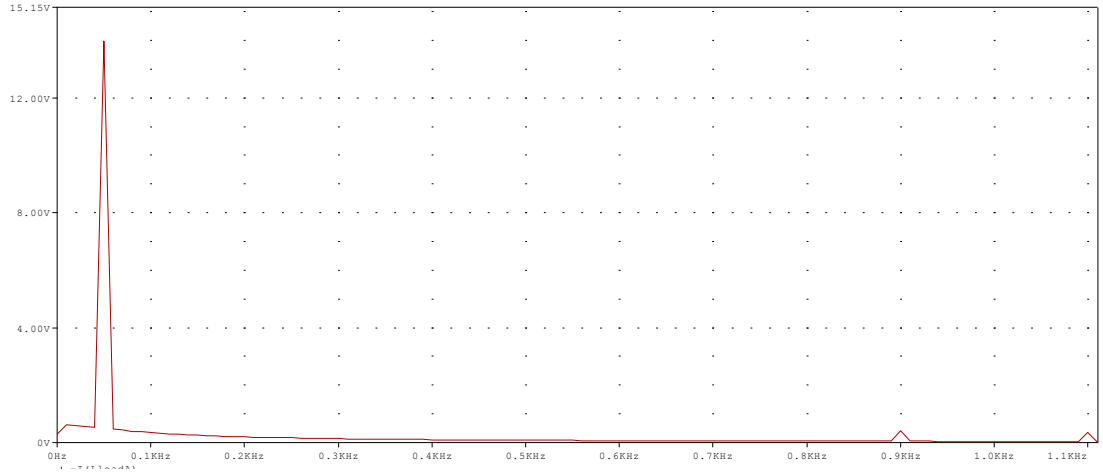


Figure 57 Fourier series Phase Current of Inverter PSpice S_12.

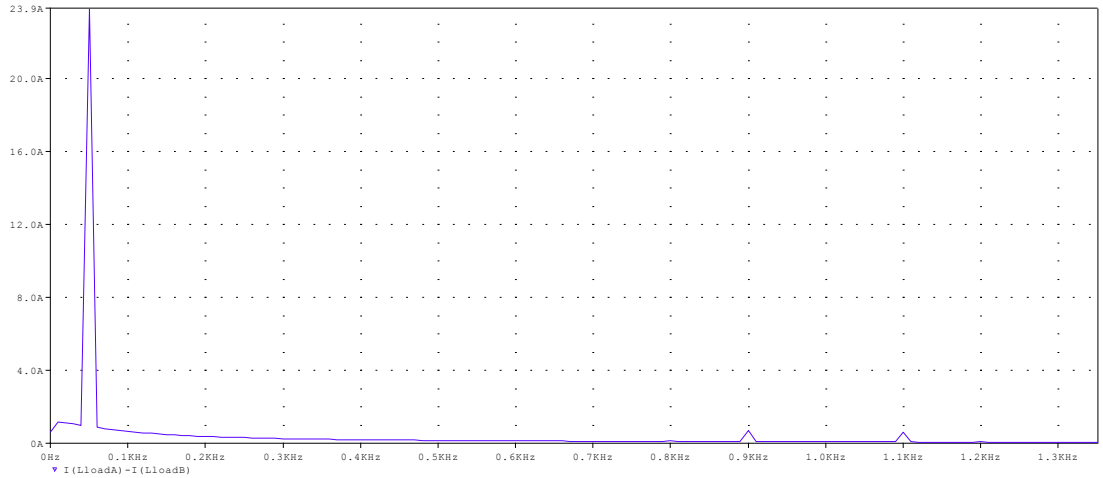


Figure 58 Fourier series Line Current of Inverter PSpice S_12.

2.3.13 S_13 - Full Wave H-Bridge Inverter (Hysteresis Control)

The following image represents the inverter circuit:

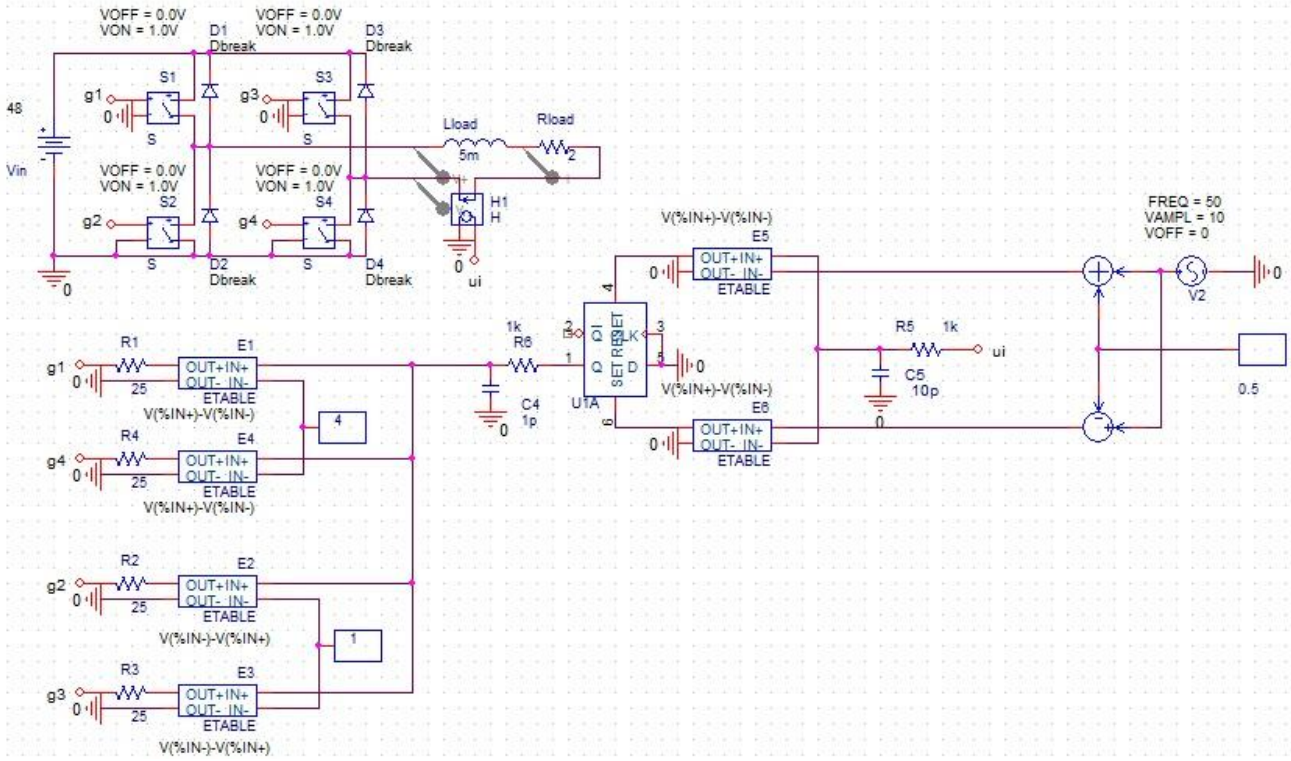


Figure 59 Inverter Scheme PSpice S_13.

The first graph corresponds to the waveform of the output at the load. The red line represents the Current that runs through load and green line represents the Voltage in it:

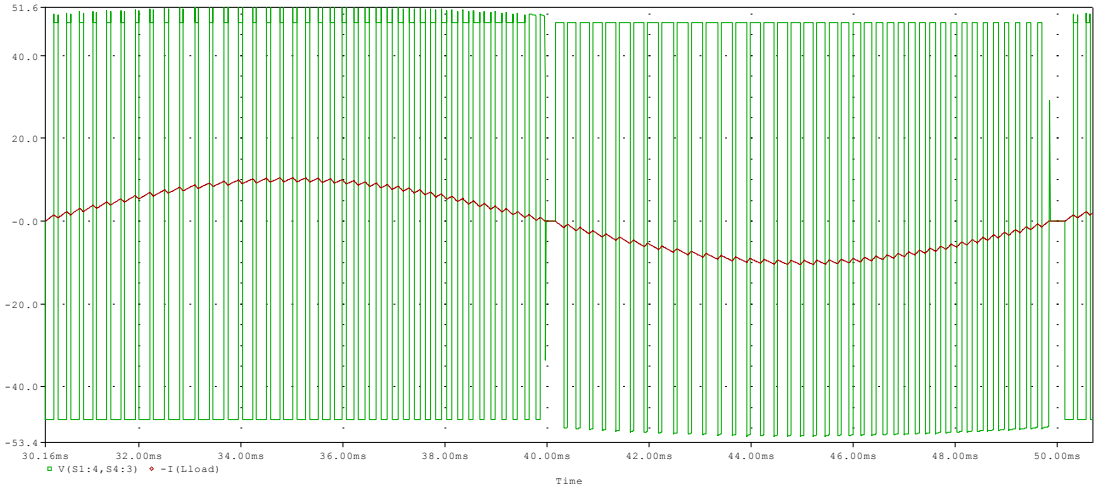


Figure 60 Output Inverter PSpice S_13.

The following graphs correspond to Fourier analysis. They represent frequency analysis of the harmonics of each waveform. Each graph goes with its own color line:

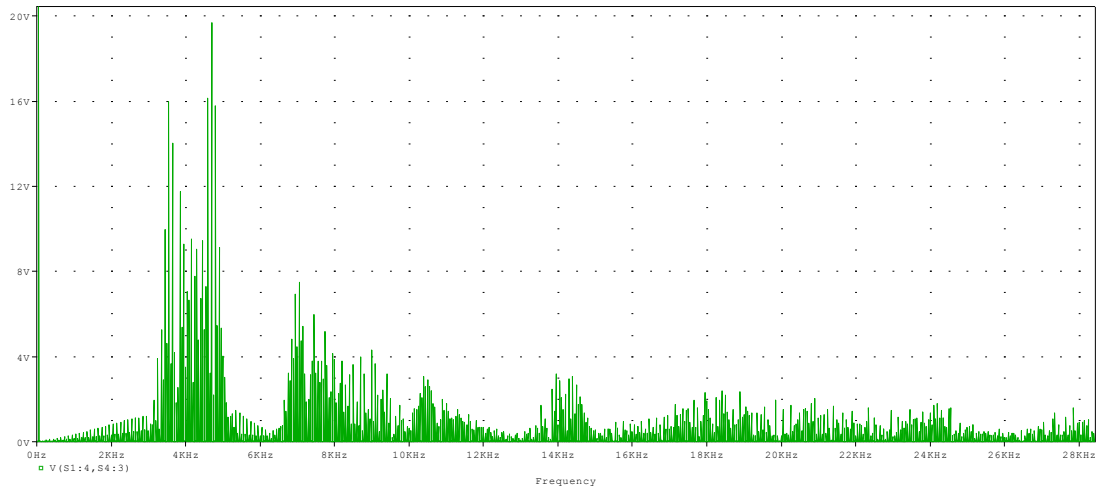


Figure 61 Fourier series Voltage of Inverter PSpice S_13.

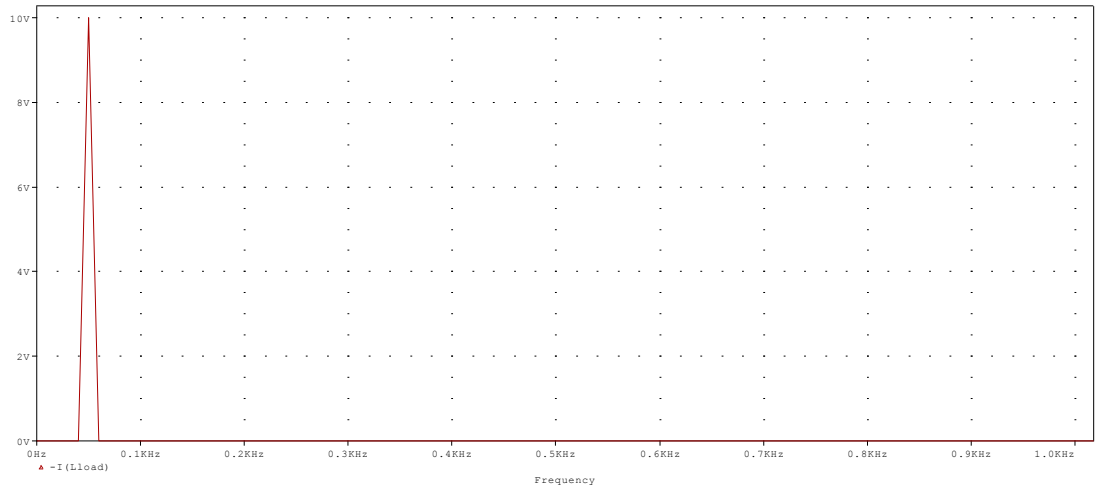


Figure 62 Fourier series Current of Inverter PSpice S_13.

2.3.14 S_14 - Push-Pull Inverter (Square Wave Control)

The following image represents the inverter circuit:

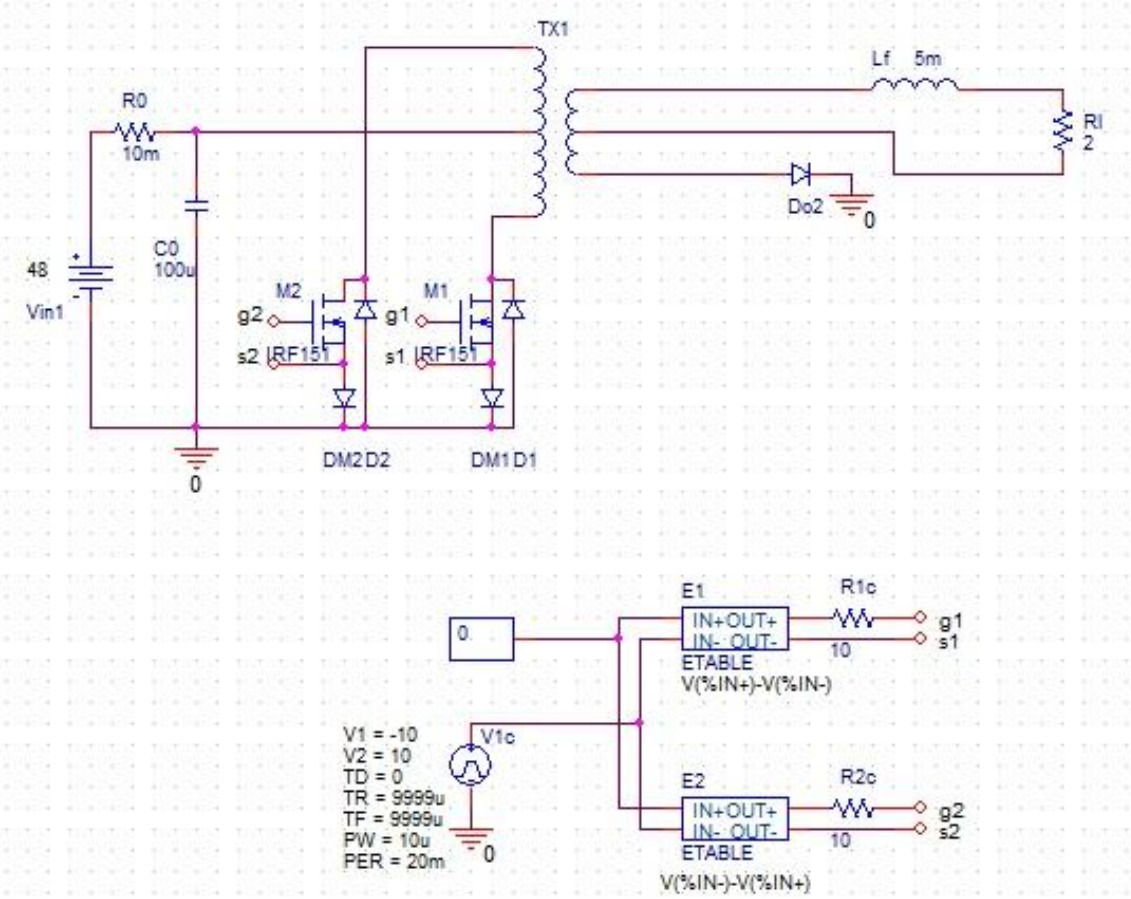


Figure 63 Inverter Scheme PSpice S_14.

The first graph corresponds to the waveform of the output at the load. The red line represents the Current that runs through load and green line represents the Voltage in it:

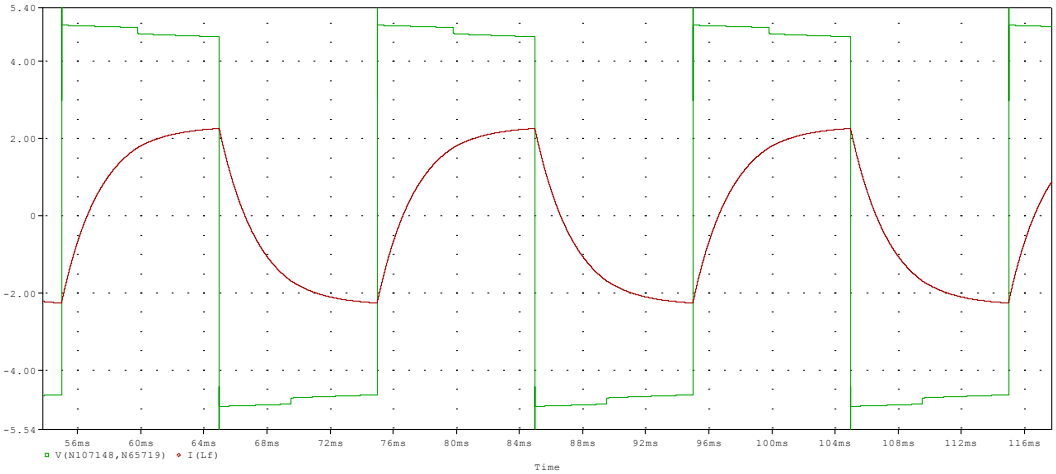


Figure 64 Output Inverter PSpice S_14.

The following graphs correspond to Fourier analysis. They represent frequency analysis of the harmonics of each waveform. Each graph goes with its own color line:

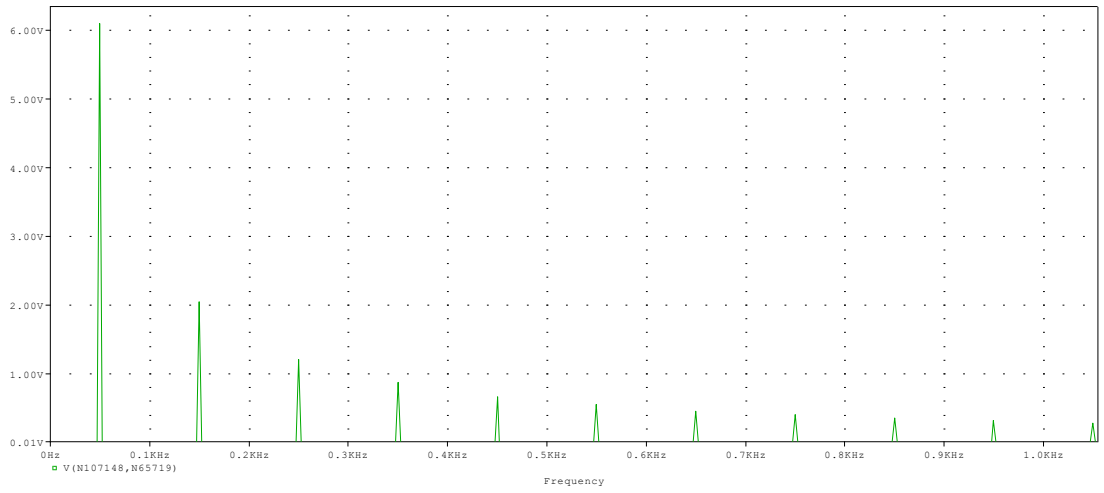


Figure 65 Fourier series Voltage of Inverter PSpice S_14.

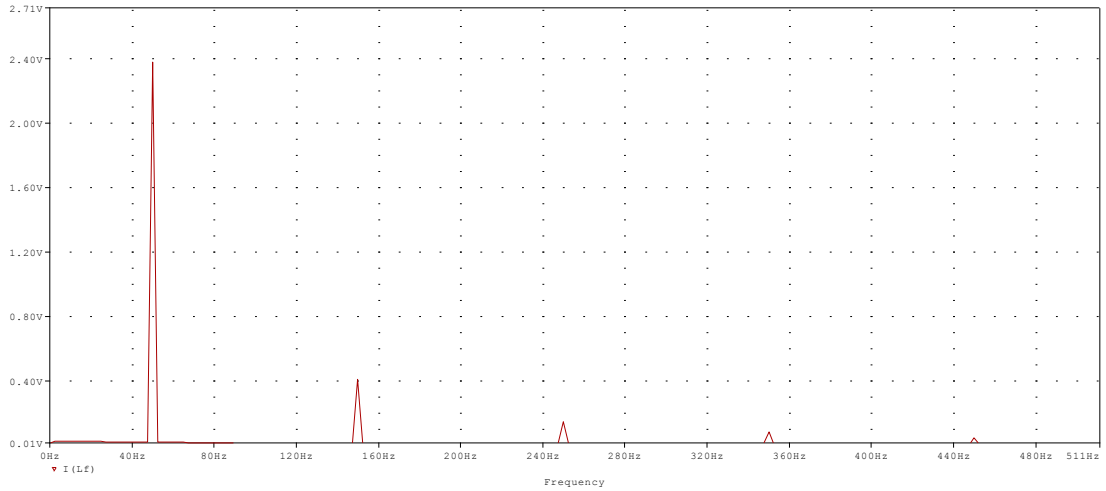


Figure 66 Fourier series Current of Inverter PSpice S_14.

2.3.15 S_15 - Push-Pull Inverter (PWM Control)

The following image represents the inverter circuit:

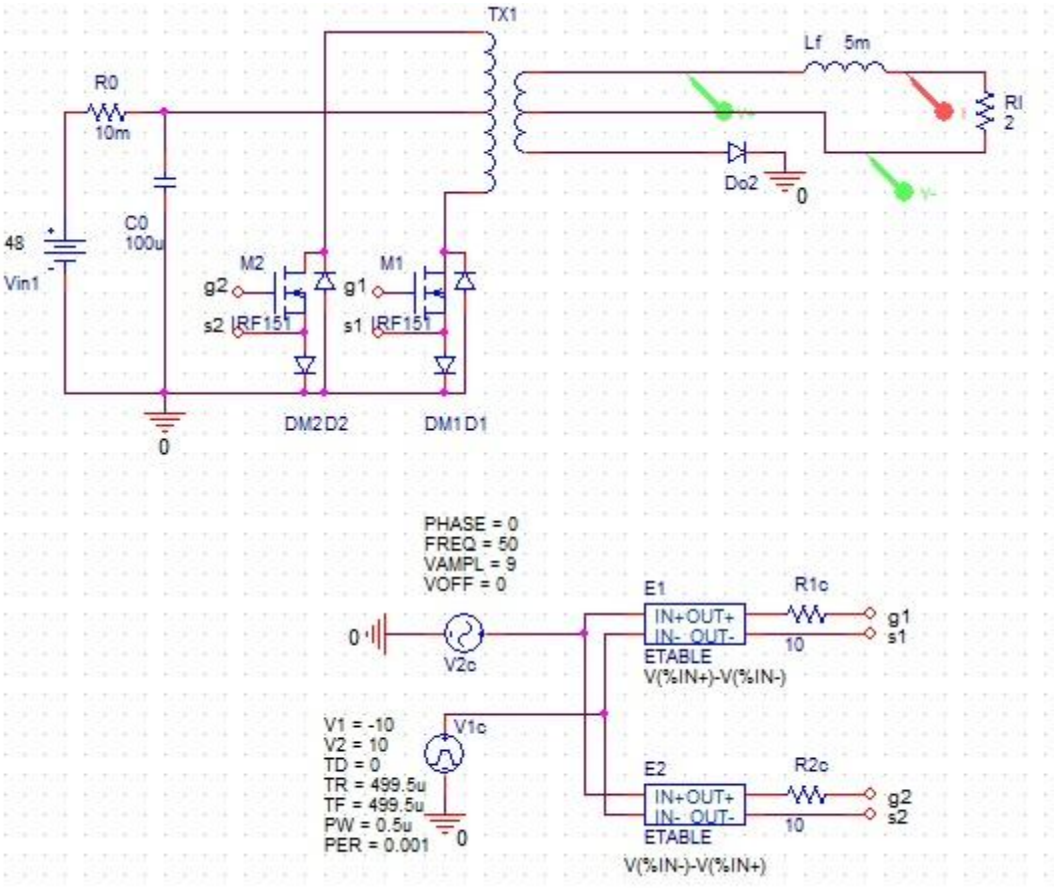


Figure 67 Inverter Scheme PSpice S_15.

The first graph corresponds to the waveform of the output at the load. The red line represents the Current that runs through load and green line represents the Voltage in it:

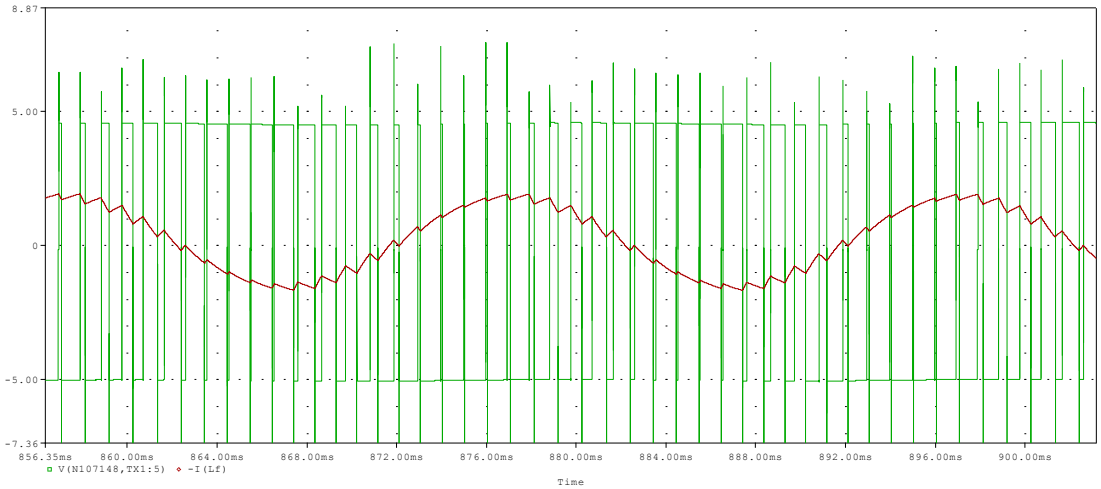


Figure 68 Output Inverter PSpice S_15.

The following graphs correspond to Fourier analysis. They represent frequency analysis of the harmonics of each waveform. Each graph goes with its own color line:

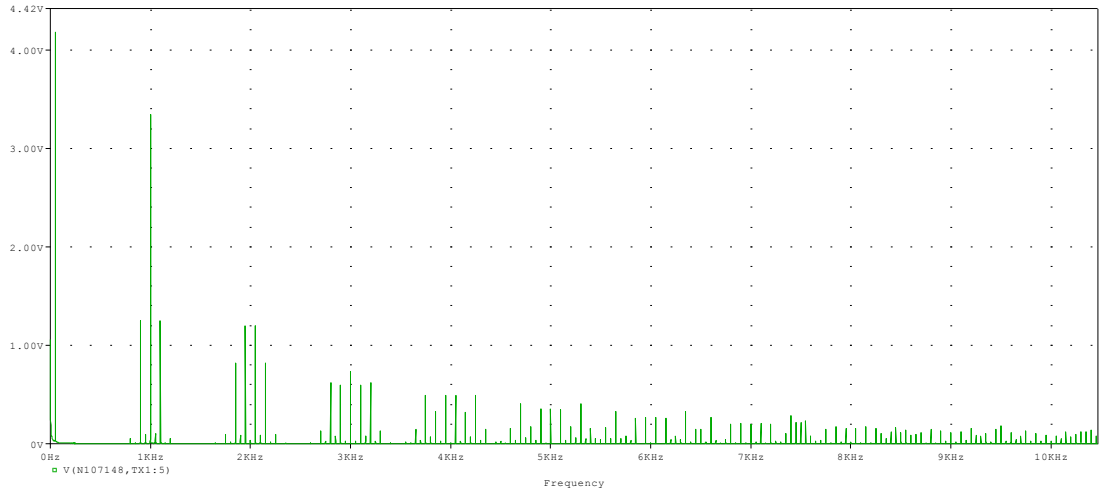


Figure 69 Fourier series Voltage of Inverter PSpice S_15.

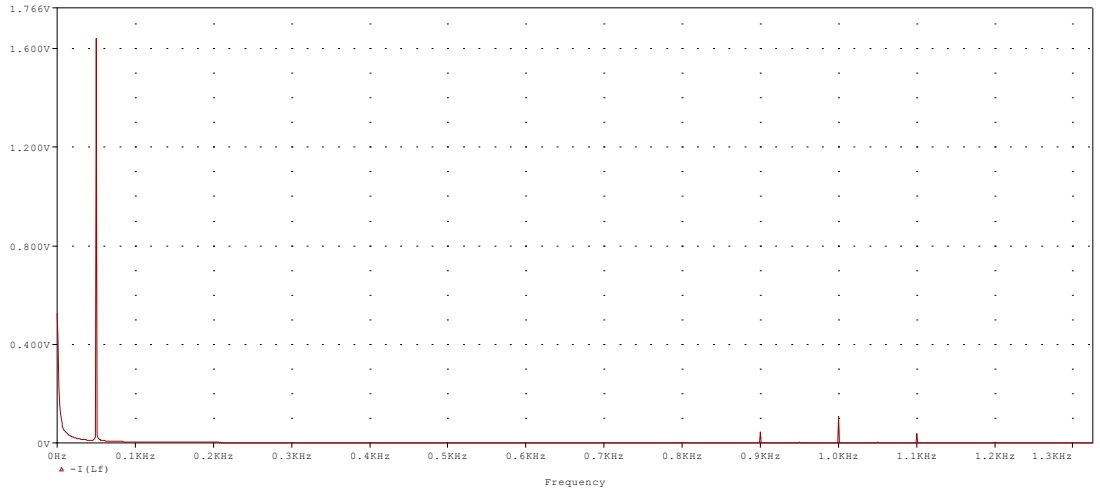


Figure 70 Fourier series Current of Inverter PSpice S_15.

2.3.16 S_16 - Cascade Full Bridge Single-Phase Inverter (2 Levels)

The following image represents the inverter circuit:

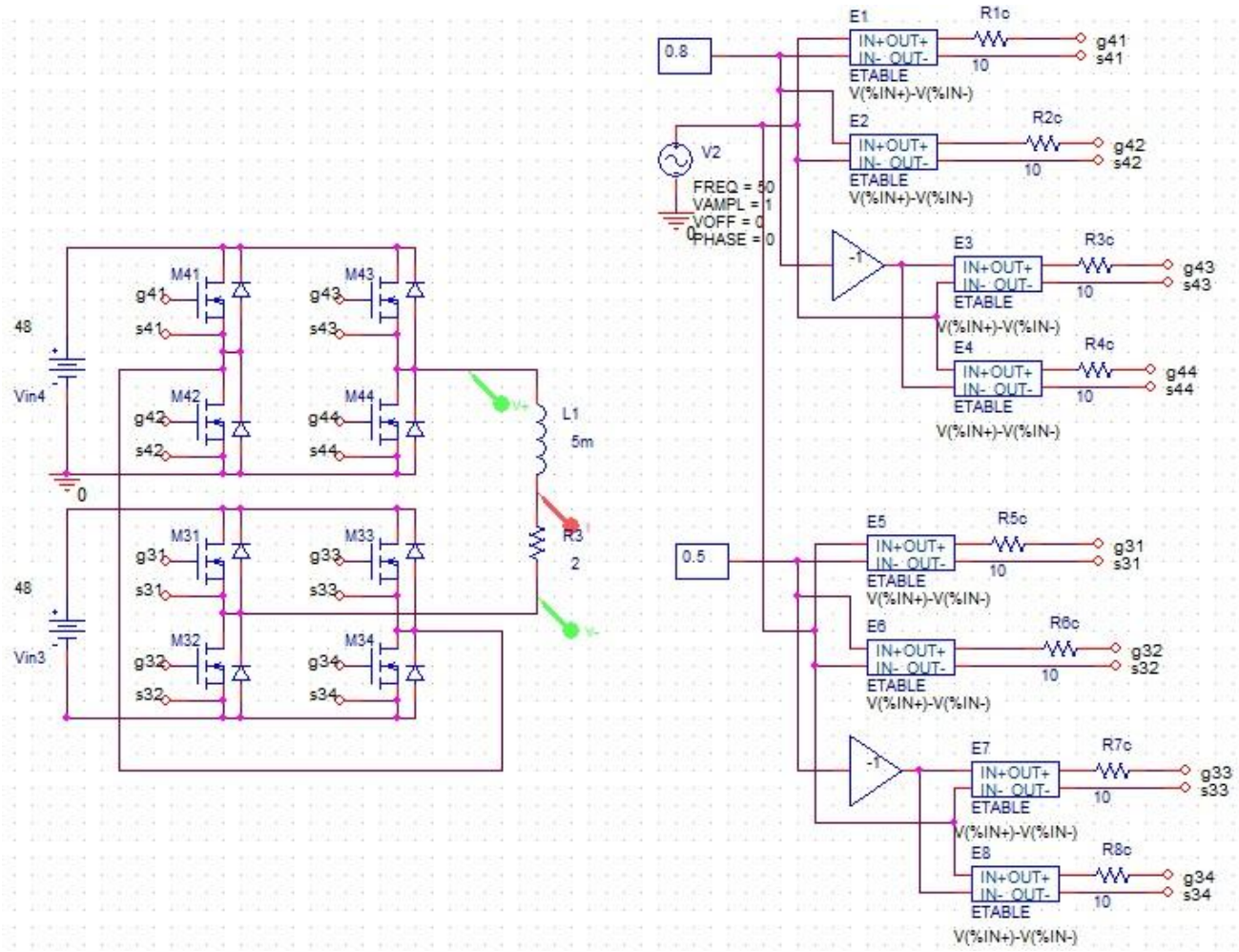


Figure 71 Inverter Scheme PSpice S_16.

The first graph corresponds to the waveform of the output at the load. The red line represents the Current that runs through load and green line represents the Voltage in it:

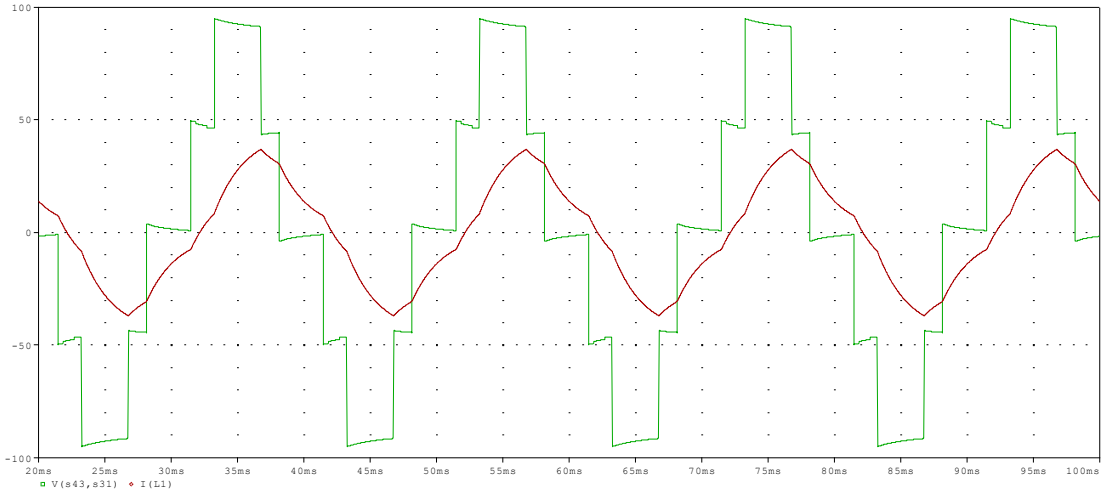


Figure 72 Output Inverter PSpice S_16.

The following graphs correspond to Fourier analysis. They represent frequency analysis of the harmonics of each waveform. Each graph goes with its own color line:

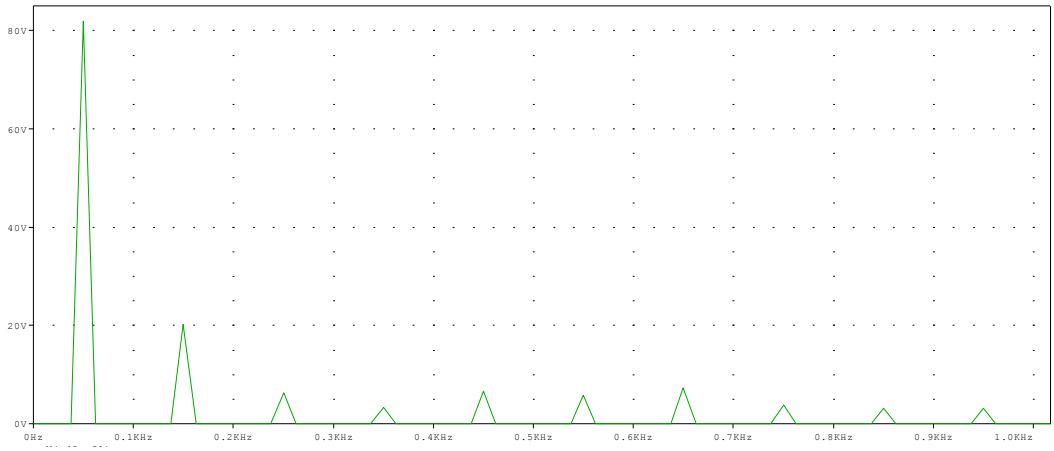


Figure 73 Fourier series Voltage of Inverter PSpice S_16.

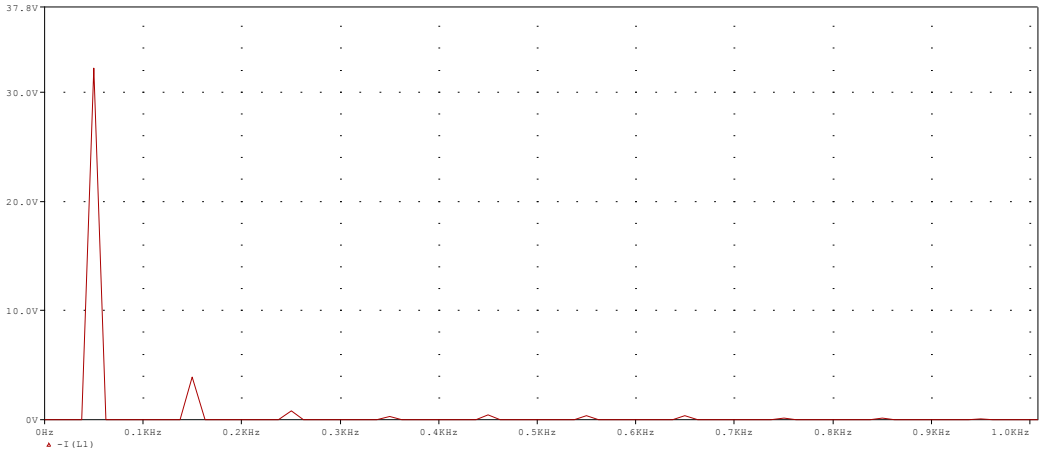


Figure 74 Fourier series Current of Inverter PSpice S_16.

2.3.17 S_17- Cascade Full Bridge Single-Phase Inverter (4 Levels)

The following image represents the inverter circuit:

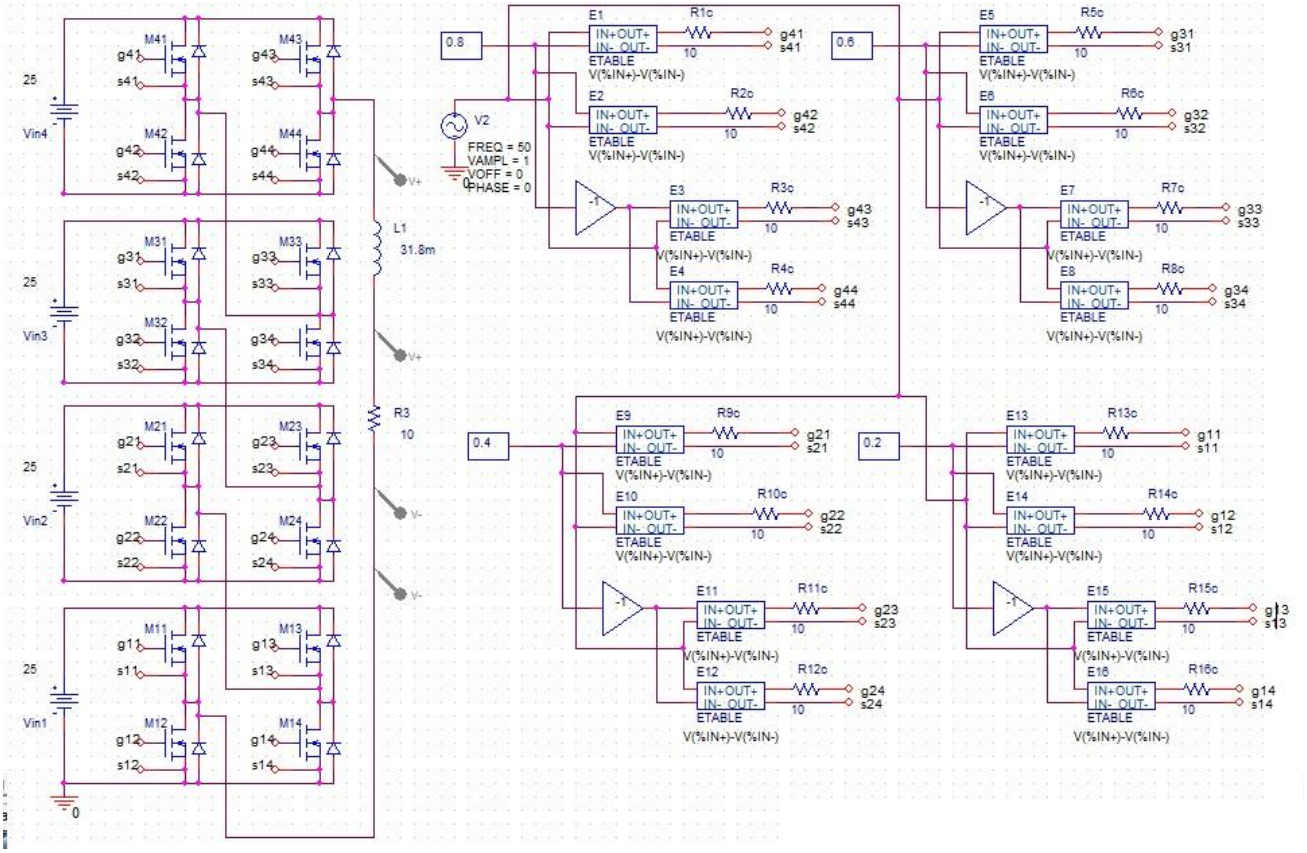


Figure 75 Inverter Scheme PSpice S_17.

The first graph corresponds to the waveform of the output at the load. The red line represents the Current that runs through load and green line represents the Voltage in it:

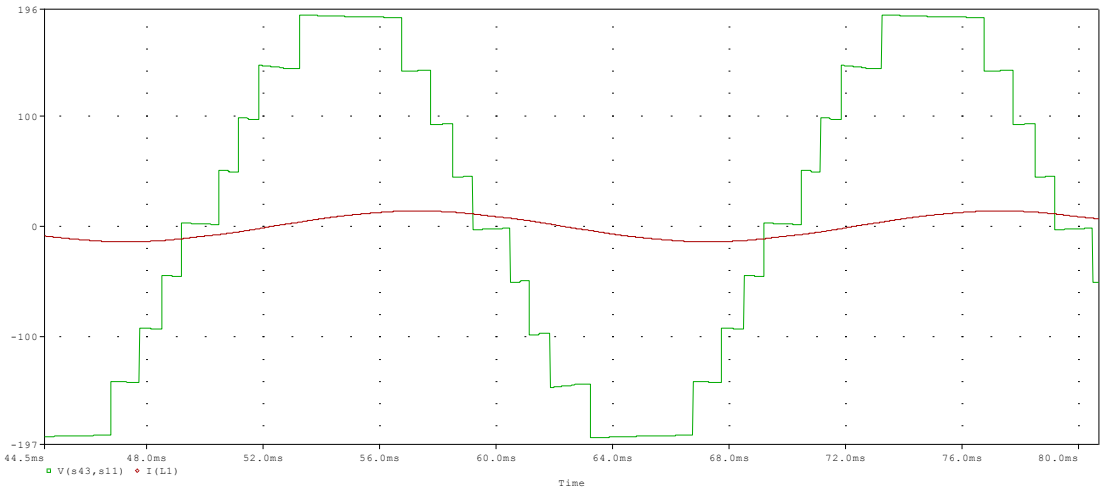


Figure 76 Output Inverter PSpice S_17.

The following graphs correspond to Fourier analysis. They represent frequency analysis of the harmonics of each waveform. Each graph goes with its own color line:

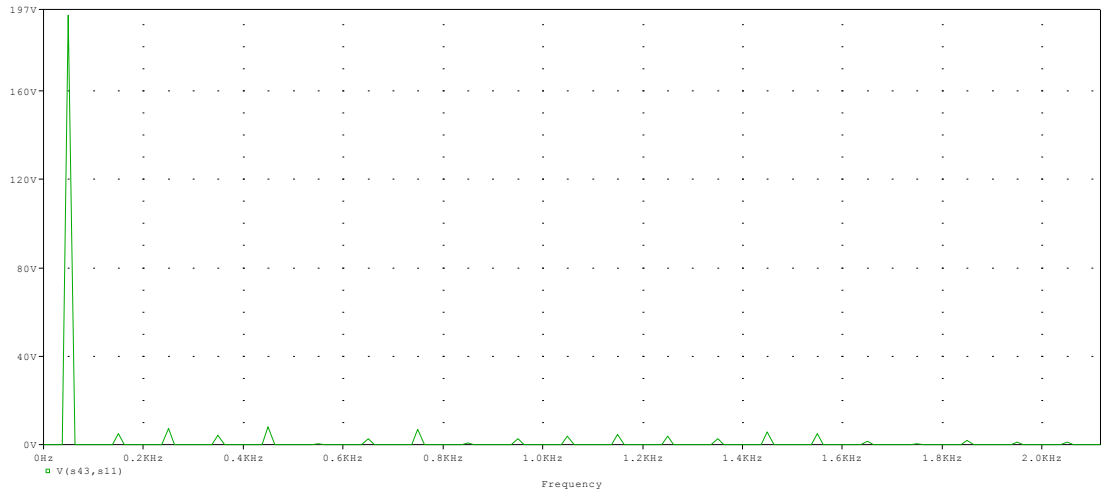


Figure 77 Fourier series Voltage of Inverter PSpice S_17.

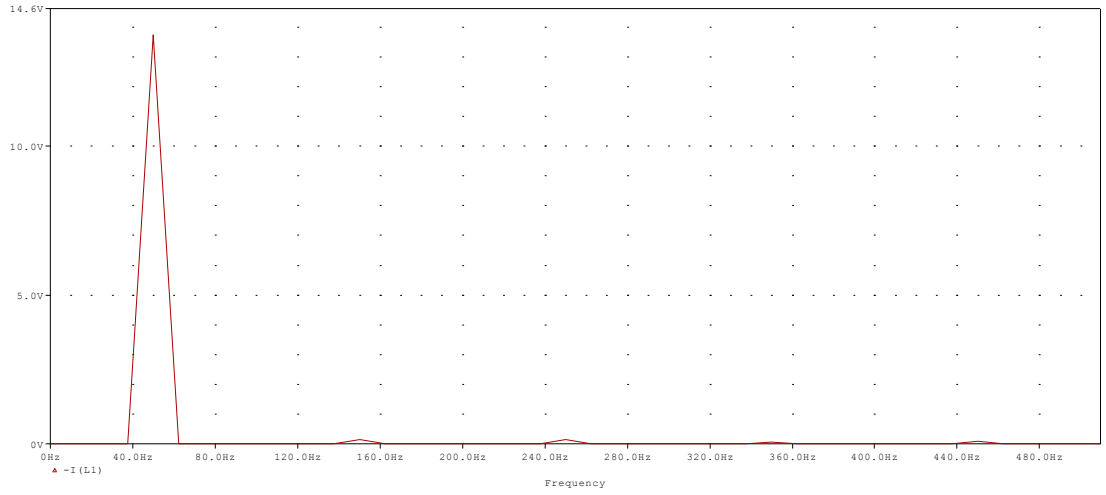


Figure 78 Fourier series Current of Inverter PSpice S_17.

2.3.18 S_18 - Cascade Full Bridge Three Phase Inverter (Star Load) (3 Levels)

The following image represents the inverter circuit:

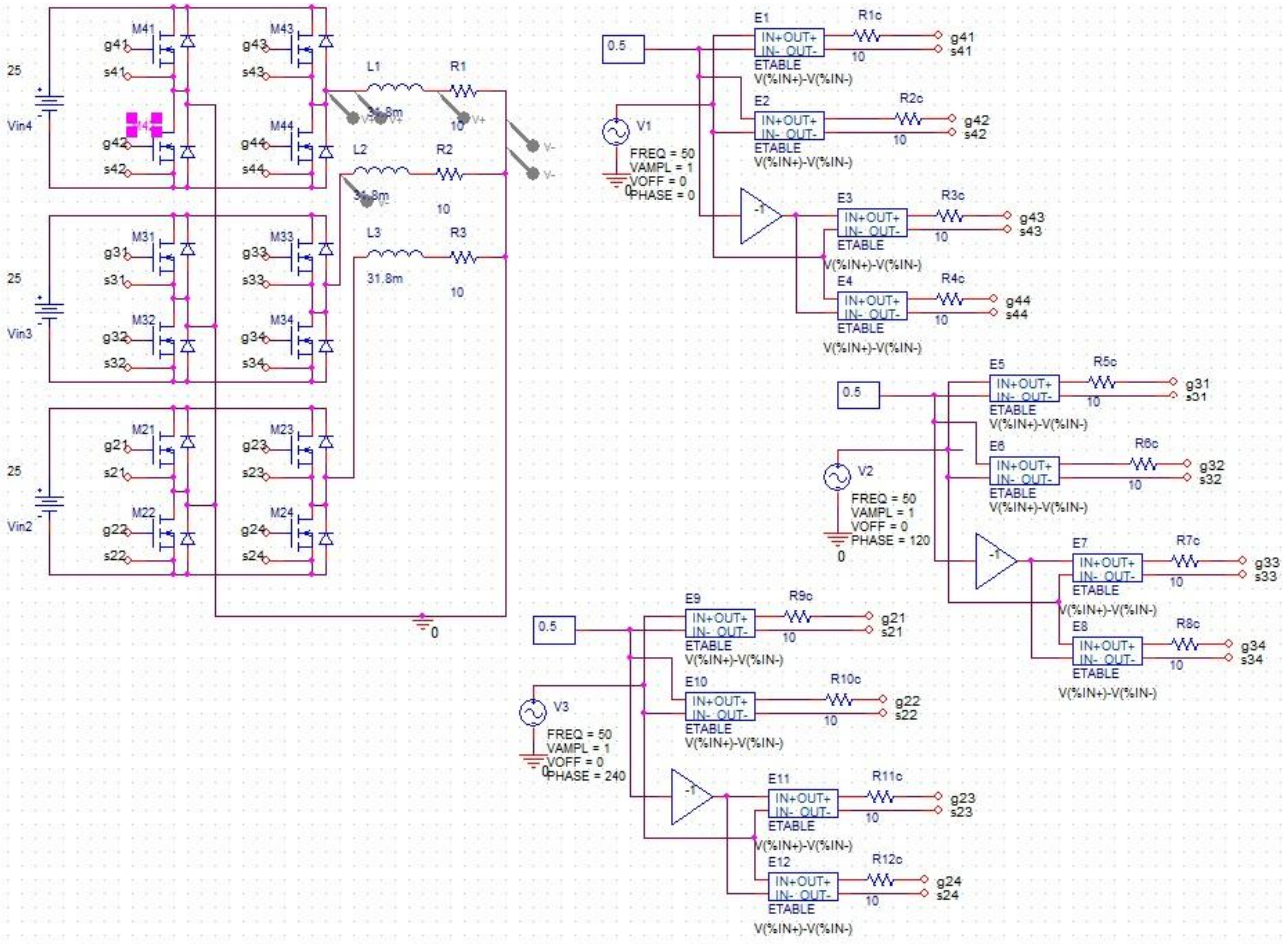


Figure 79 Inverter Scheme PSpice S_18.

Graph of the waveform at the output load. The blue line represents the Current that runs through load, red line represents the Voltage (line) and green line is the Voltage (phase) in it:

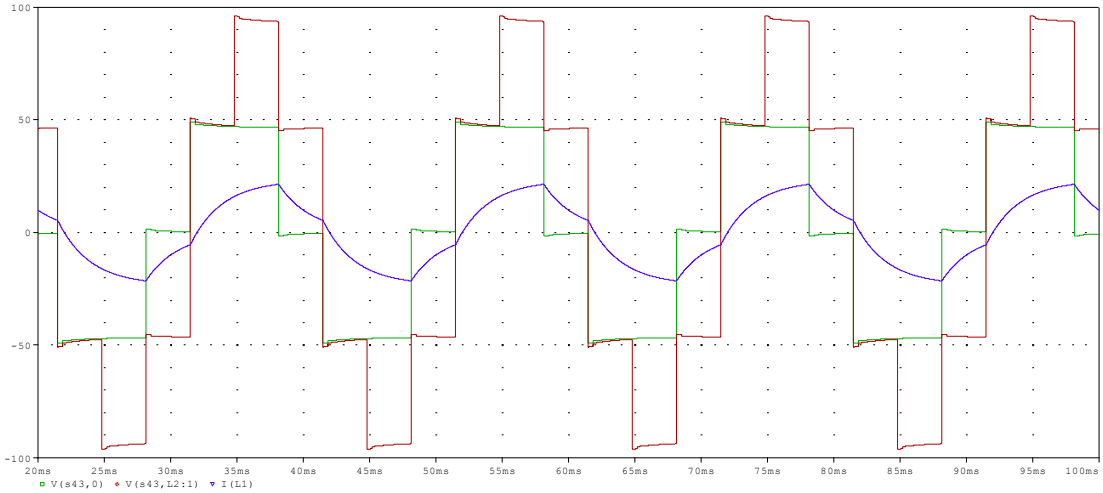


Figure 80 Output Inverter PSpice S_18.

The following graphs correspond to Fourier analysis. They represent frequency analysis of the harmonics of each waveform. Each graph goes with its own color line:

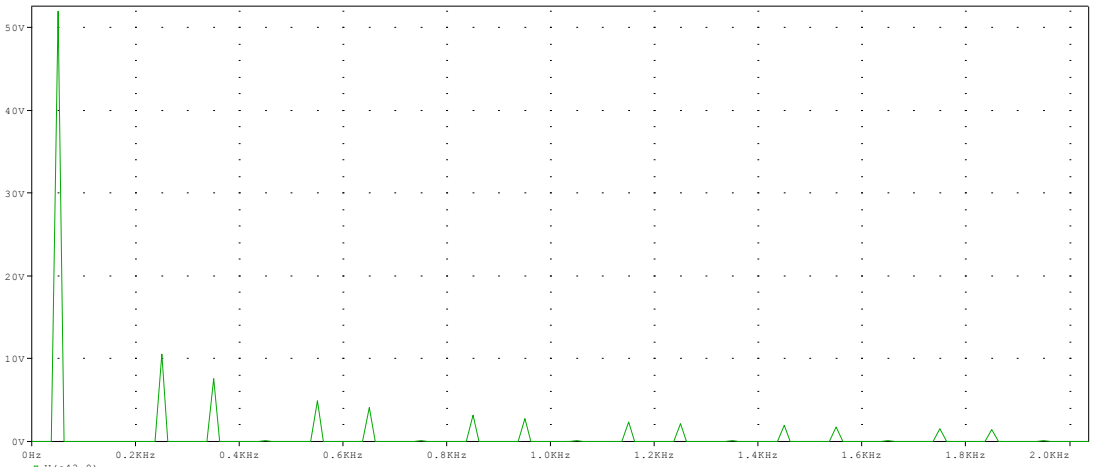


Figure 81 Fourier series Phase Voltage of Inverter PSpice S_18.

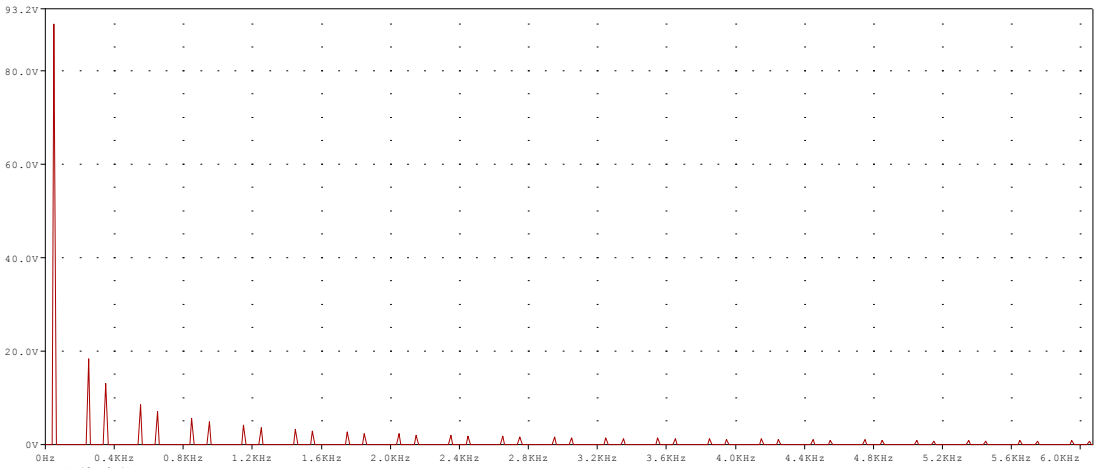


Figure 82 Fourier series Line Voltage of Inverter PSpice S_18.

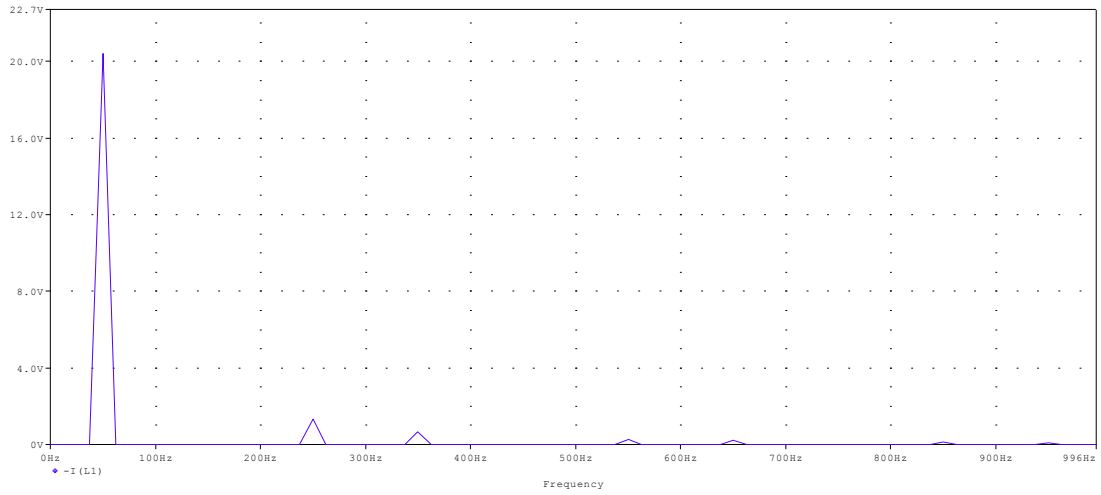


Figure 83 Fourier series Current of Inverter PSpice S_18.

2.3.19 S_19 - NPC Inverter

The following image represents the inverter circuit:

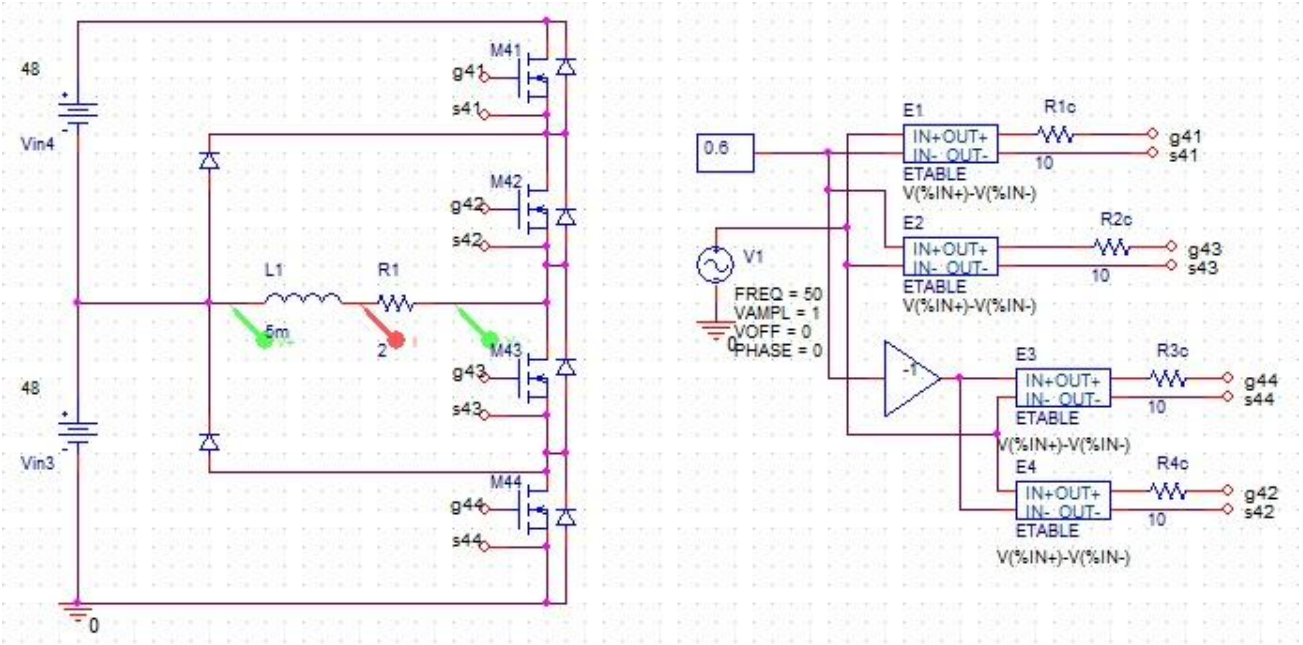


Figure 84 Inverter Scheme PSpice S_19.

The first graph corresponds to the waveform of the output at the load. The red line represents the Current that runs through load and green line represents the Voltage in it:

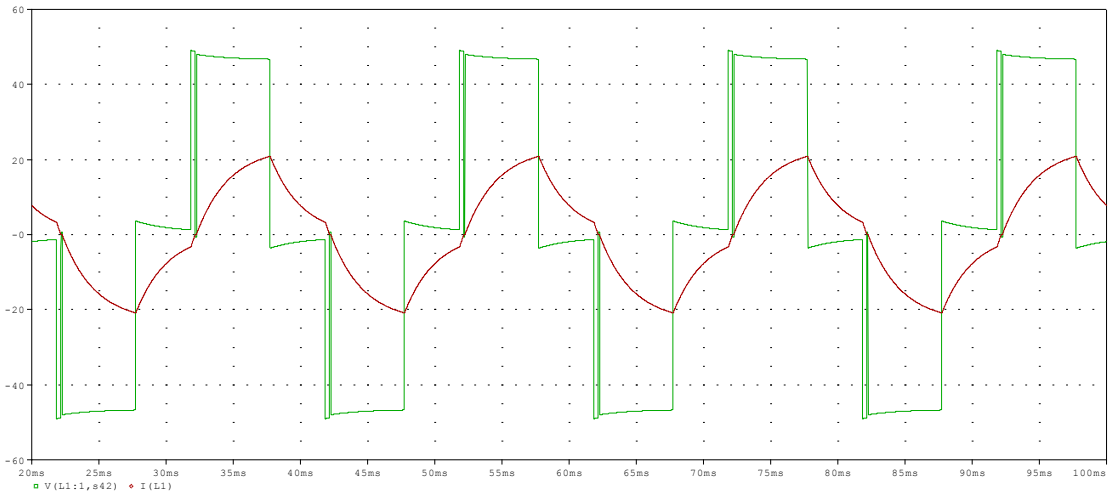


Figure 85 Output Inverter PSpice S_19.

The following graphs correspond to Fourier analysis. They represent frequency analysis of the harmonics of each waveform. Each graph goes with its own color line:

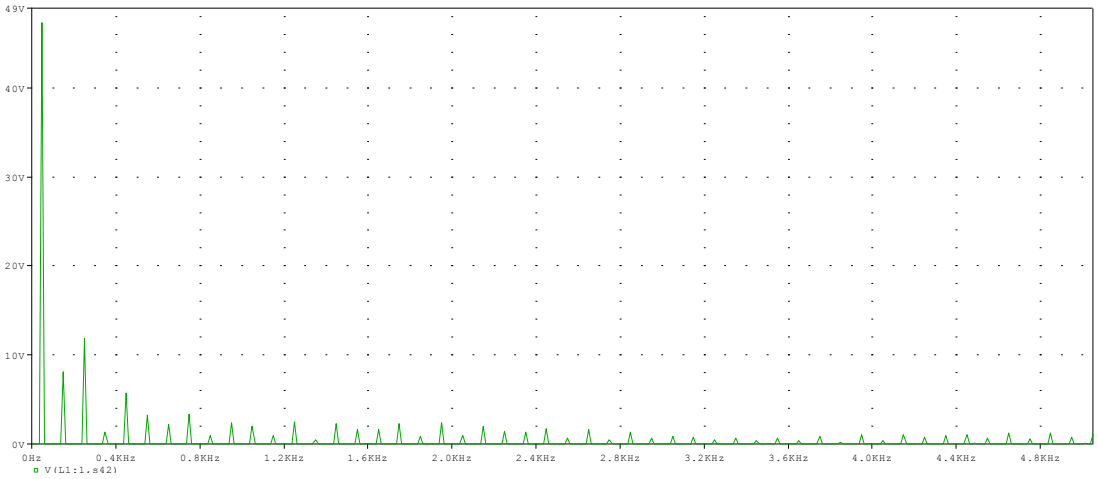


Figure 86 Fourier series Voltage of Inverter PSpice S_19.

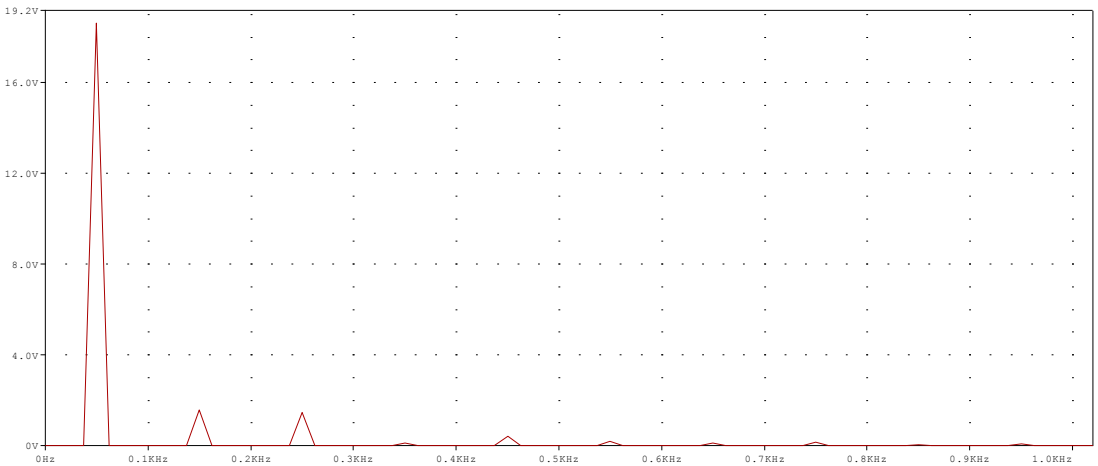


Figure 87 Fourier series Current of Inverter PSpice S_19.

2.3.20 S_20 - NPC Inverter (5 Levels)

The following image represents the inverter circuit:

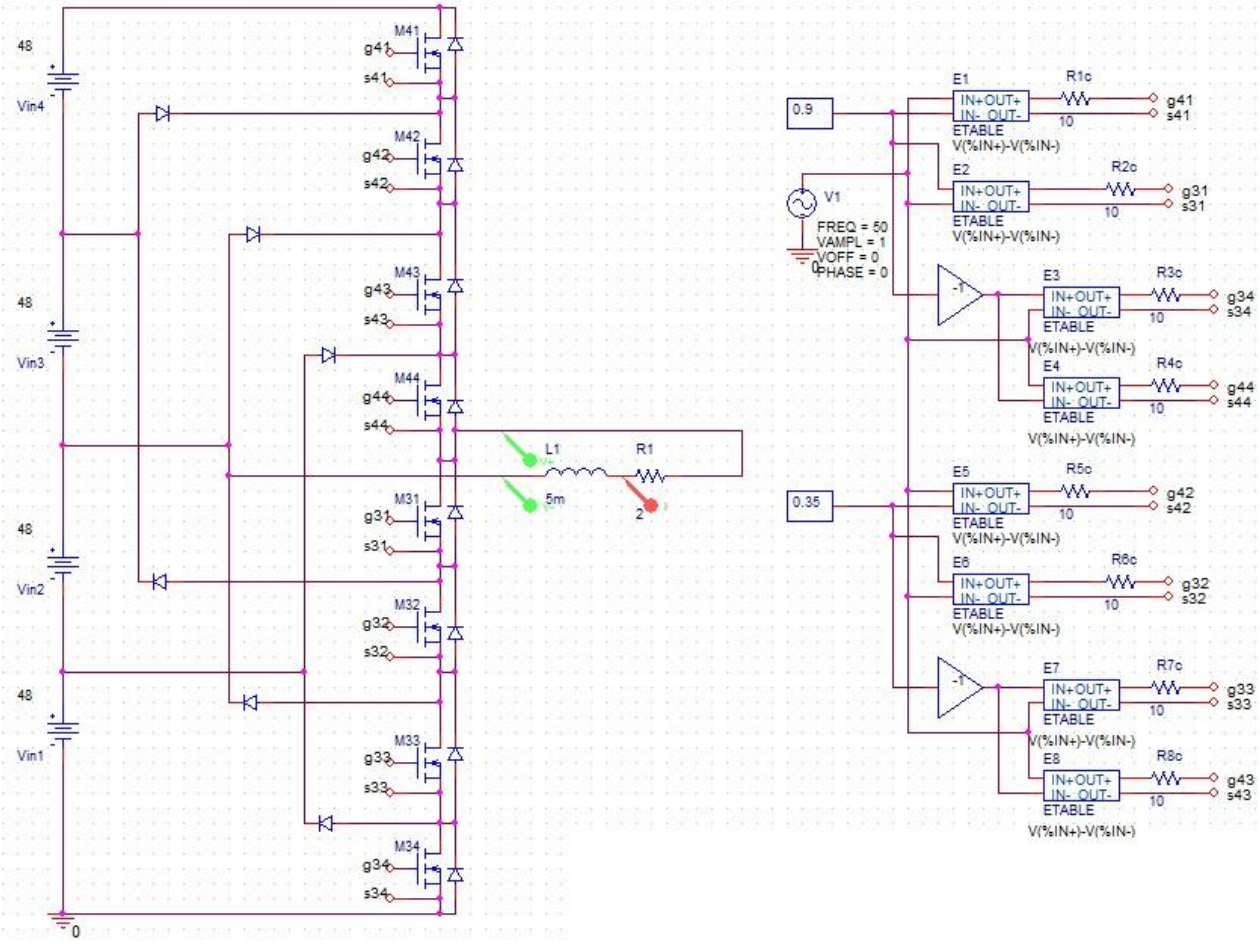


Figure 88 Inverter Scheme PSpice S_20.

The first graph corresponds to the waveform of the output at the load. The red line represents the Current that runs through load and green line represents the Voltage in it:

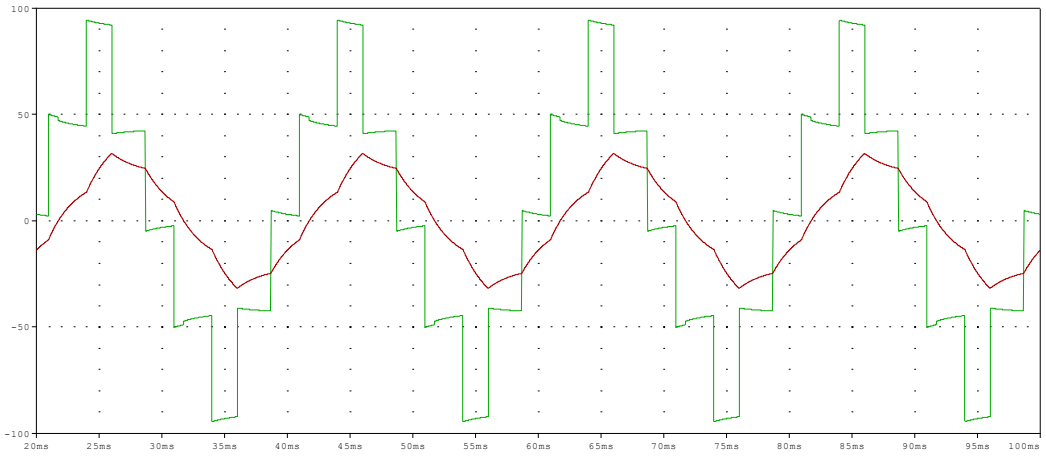


Figure 89 Output Inverter PSpice S_20.

The following graphs correspond to Fourier analysis. They represent frequency analysis of the harmonics of each waveform. Each graph goes with its own color line:

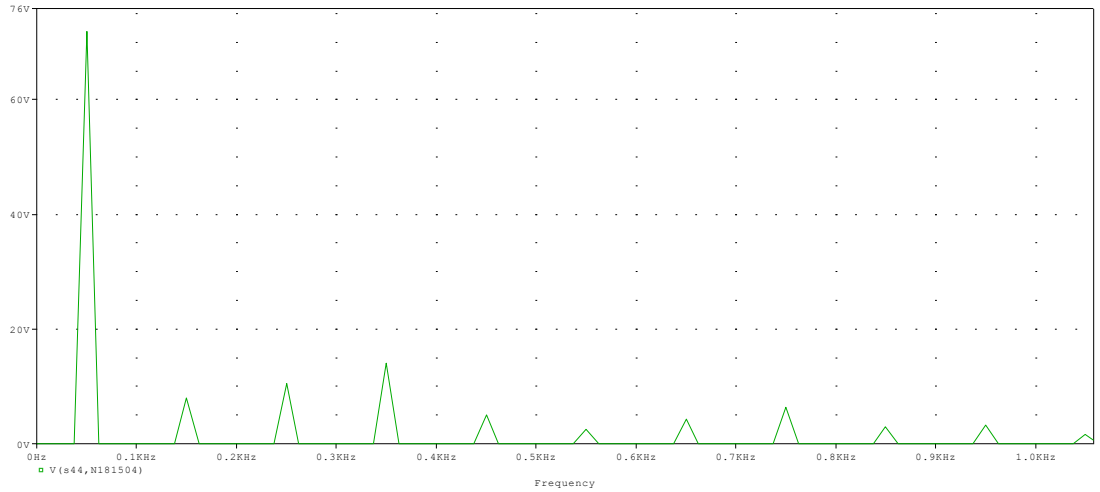


Figure 90 Fourier series Voltage of Inverter PSpice S_20.

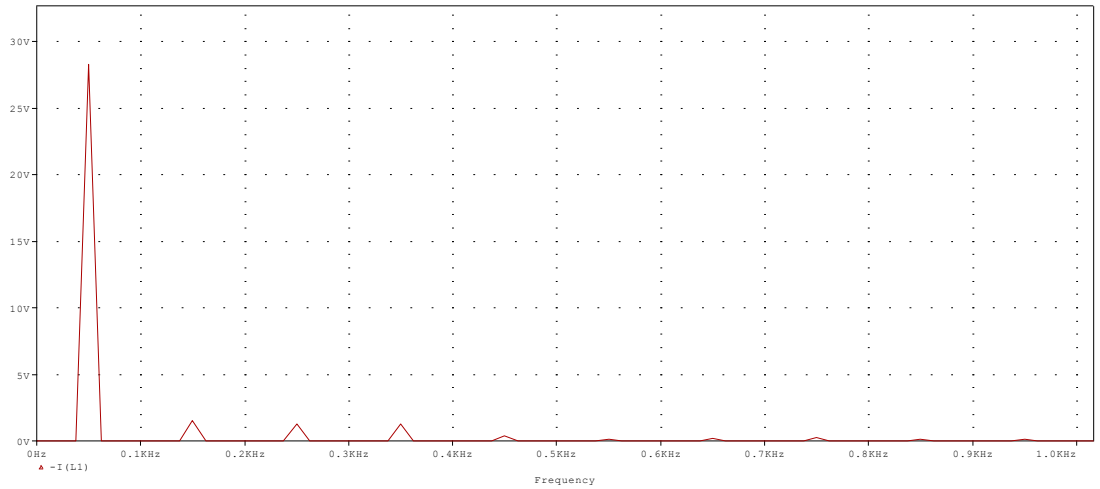


Figure 91 Fourier series Current of Inverter PSpice S_20.

2.3.21 S_21 - NPC Three-Phase Inverter (Star Load)

The following image represents the inverter circuit:

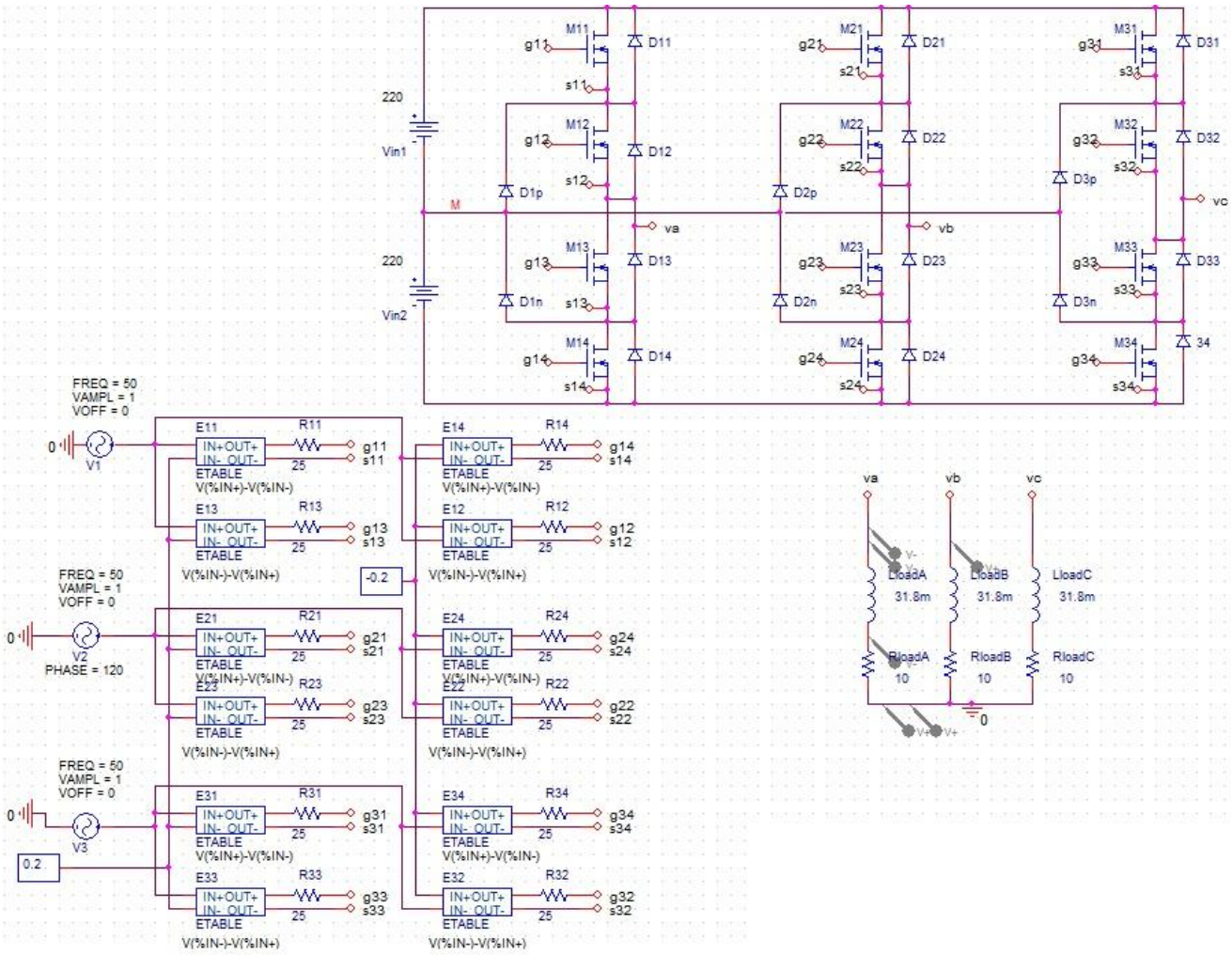


Figure 92 Inverter Scheme PSpice S_21.

Graph of the waveform at the output load. The blue line represents the Current that runs through load, red line represents the Voltage (line) and green line is the Voltage (phase) in it:

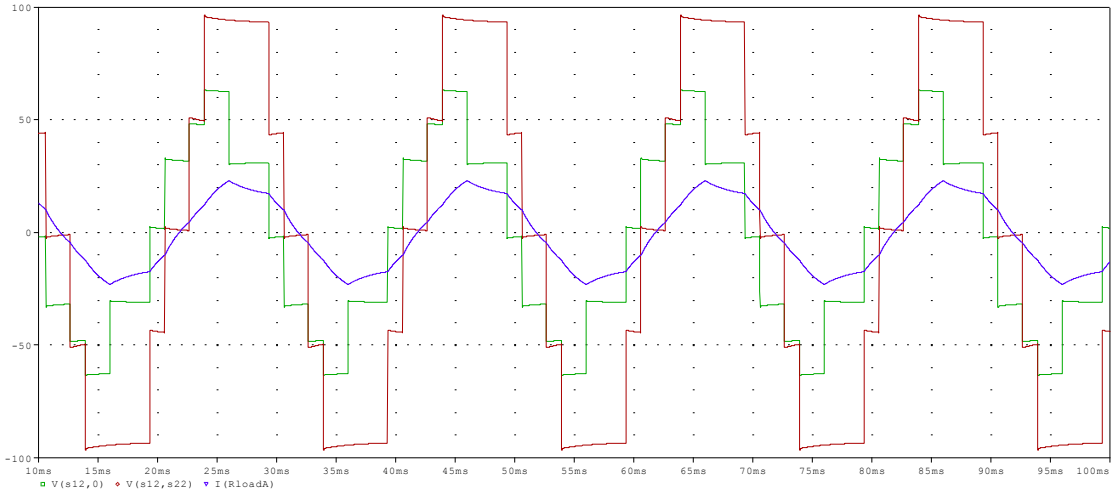


Figure 93 Output Inverter PSpice S_21.

The following graphs correspond to Fourier analysis. They represent frequency analysis of the harmonics of each waveform. Each graph goes with its own color line:

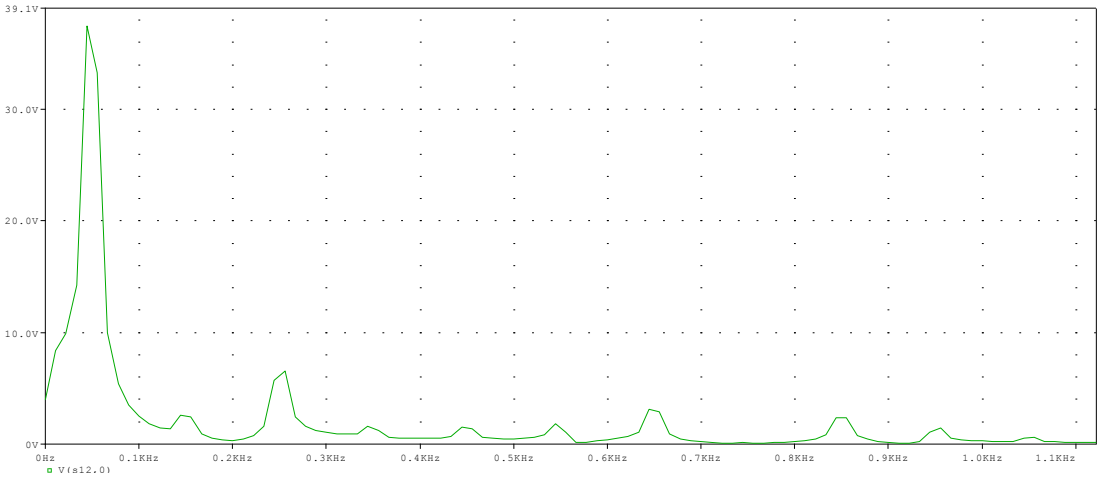


Figure 94 Fourier series Phase Voltage of Inverter PSpice S_21.

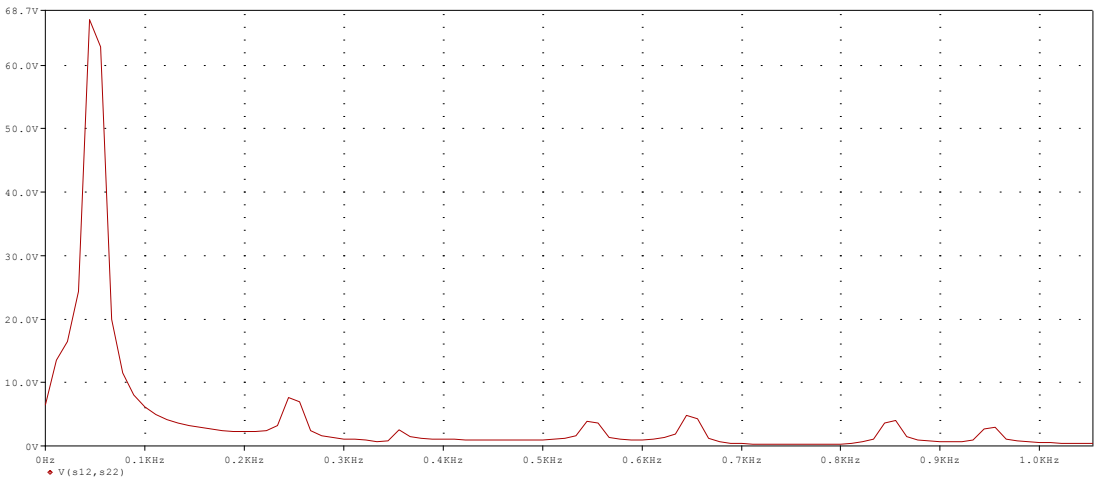


Figure 95 Fourier series Line Voltage of Inverter PSpice S_21.

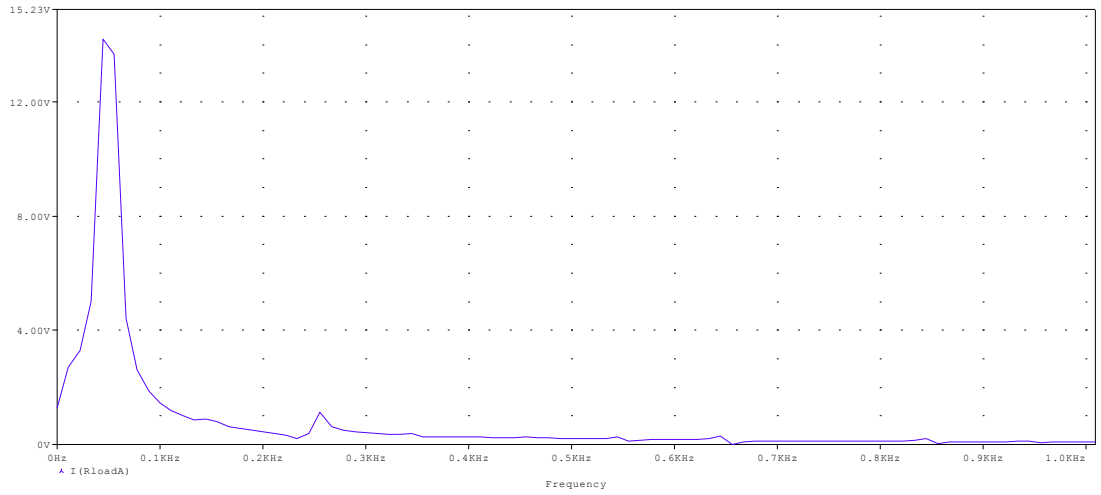


Figure 96 Fourier series Current of Inverter PSpice S_21.

2.3.22 S_22 - FC Inverter (5 Levels)

The following image represents the inverter circuit:

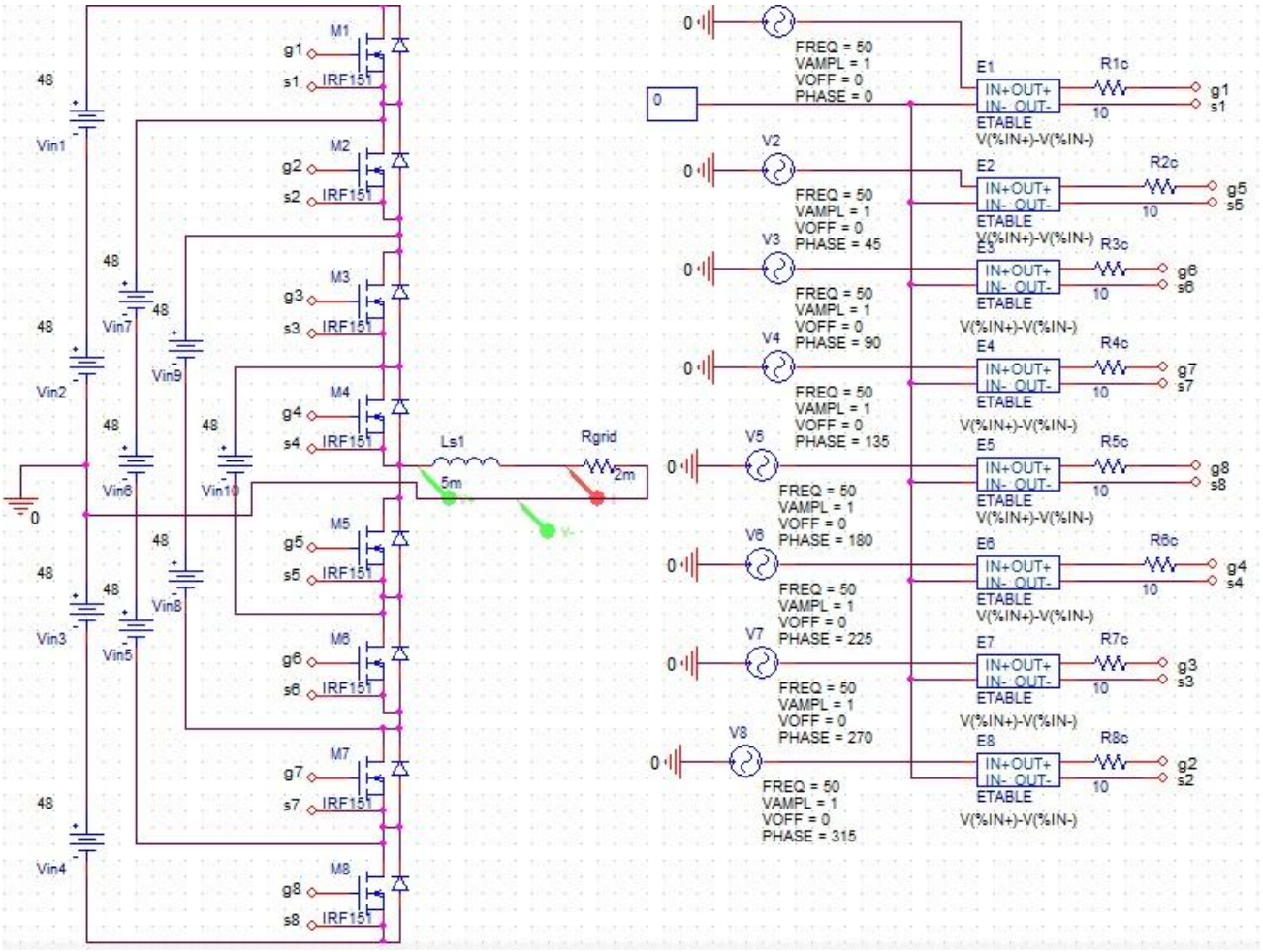


Figure 97 Inverter Scheme PSpice S_22.

The first graph corresponds to the waveform of the output at the load. The red line represents the Current that runs through load and green line represents the Voltage in it:

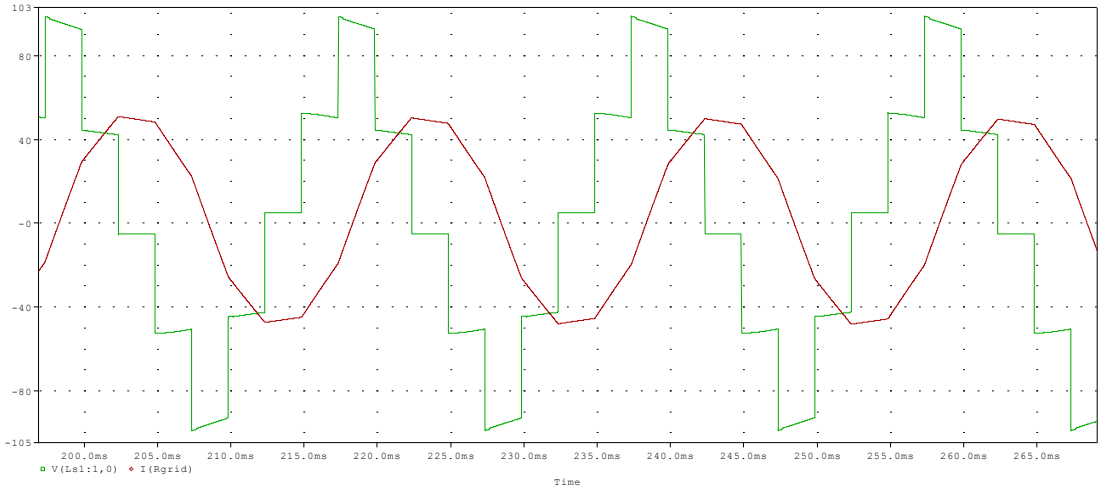


Figure 98 Output Inverter PSpice S_22.

The following graphs correspond to Fourier analysis. They represent frequency analysis of the harmonics of each waveform. Each graph goes with its own color line:

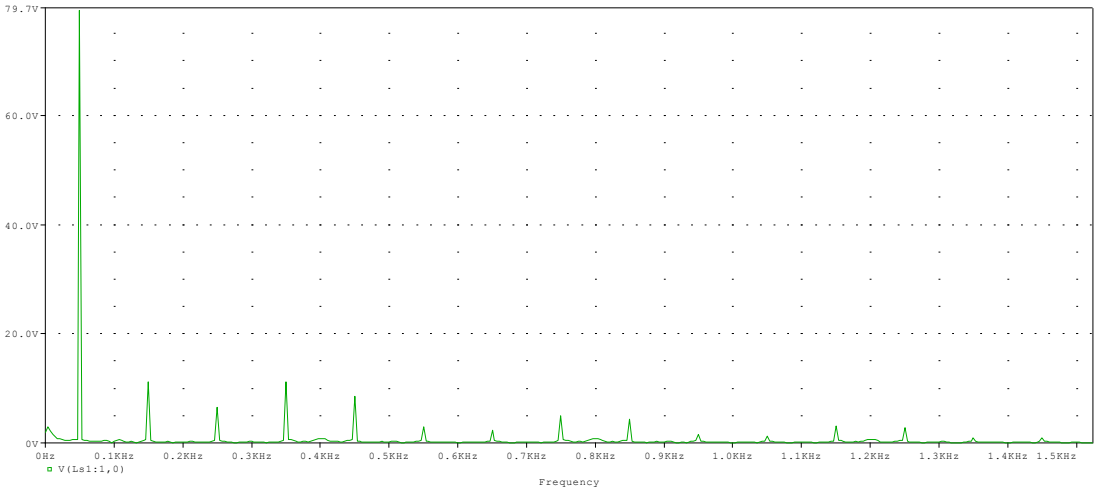


Figure 99 Fourier series Voltage of Inverter PSpice S_22.

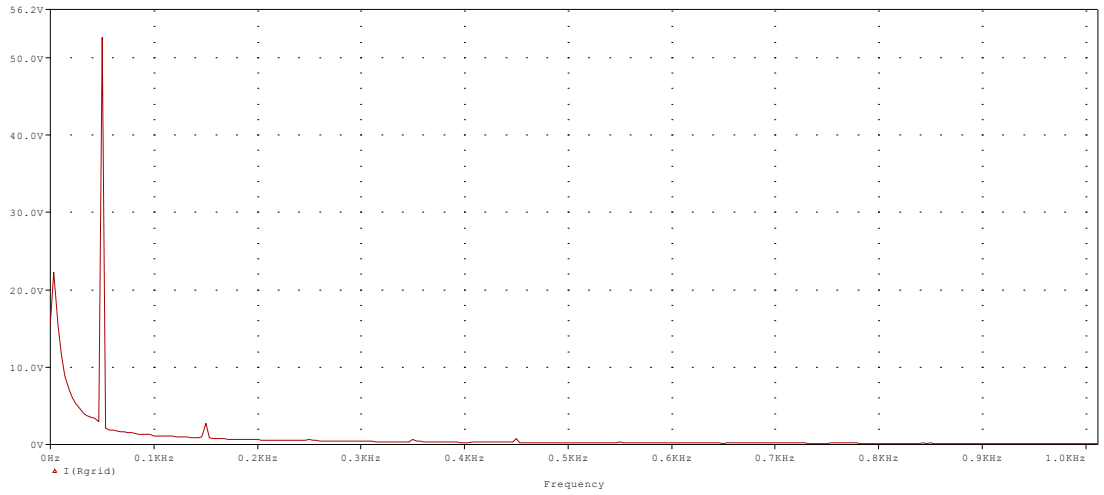


Figure 100 Fourier series Current of Inverter PSpice S_22.

2.3.23 S_23 - FC Three-Phase Inverter (Star Load) (3 Levels)

The following image represents the inverter circuit:

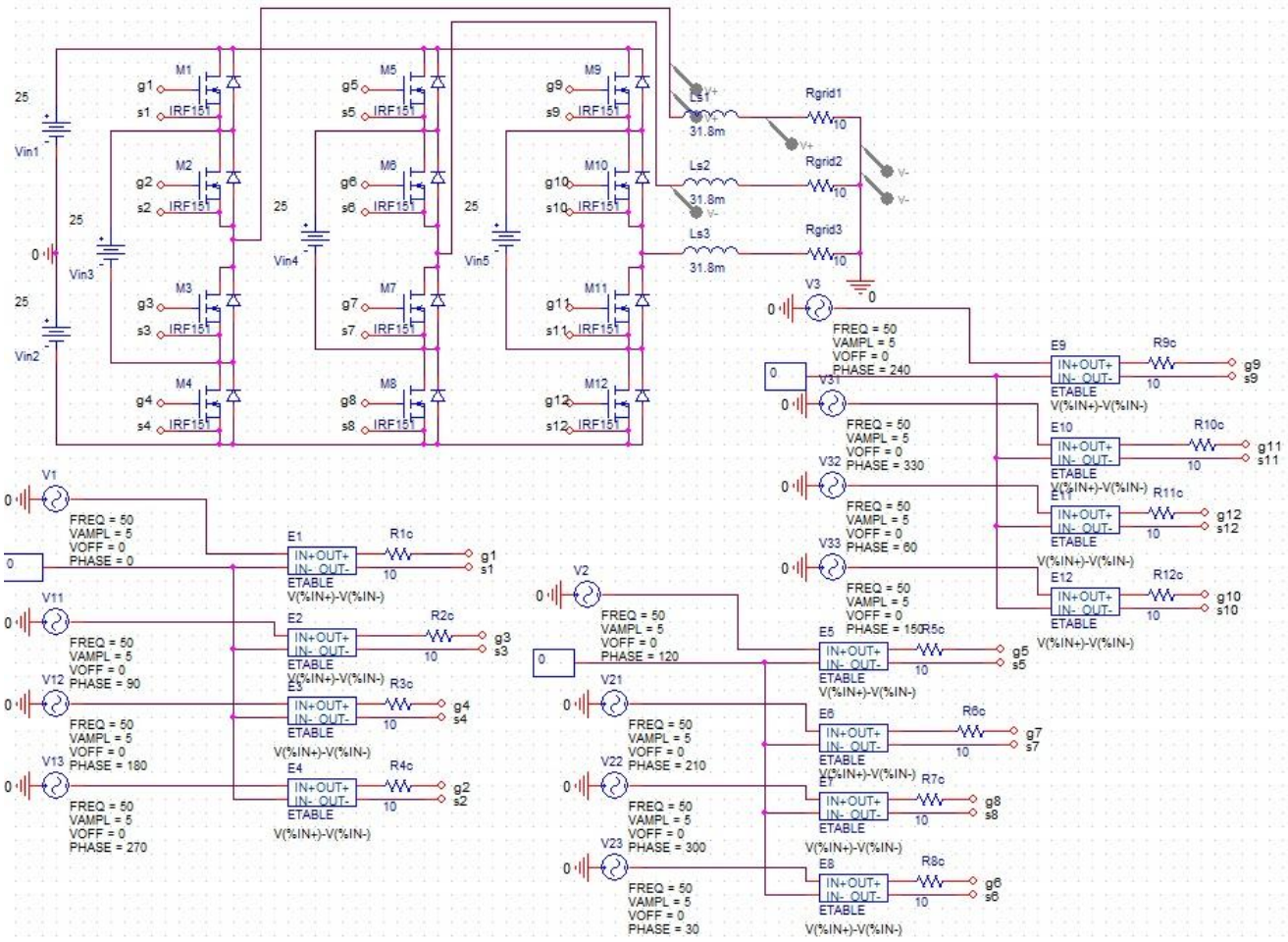


Figure 101 Inverter Scheme PSpice S_23.

Graph of the waveform at the output load. The blue line represents the Current that runs through load, red line represents the Voltage (line) and green line is the Voltage (phase) in it:

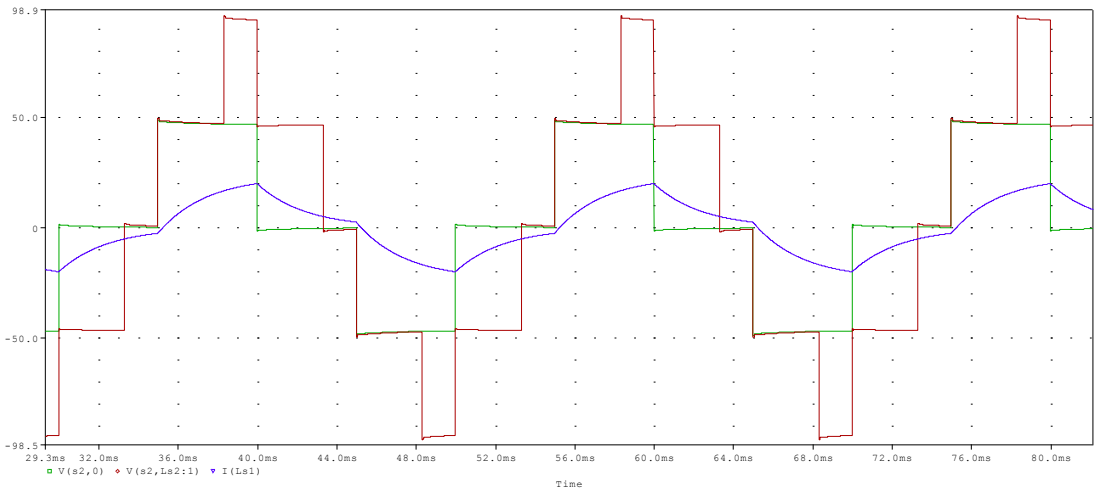


Figure 102 Output Inverter PSpice S_23.

The following graphs correspond to Fourier analysis. They represent frequency analysis of the harmonics of each waveform. Each graph goes with its own color line:

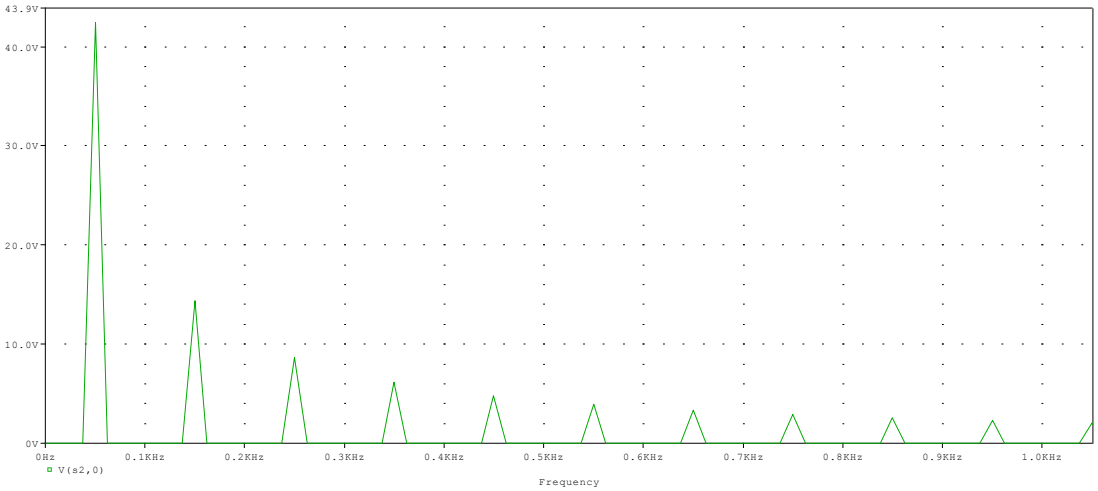


Figure 103 Fourier series Phase Voltage of Inverter PSpice S_23.

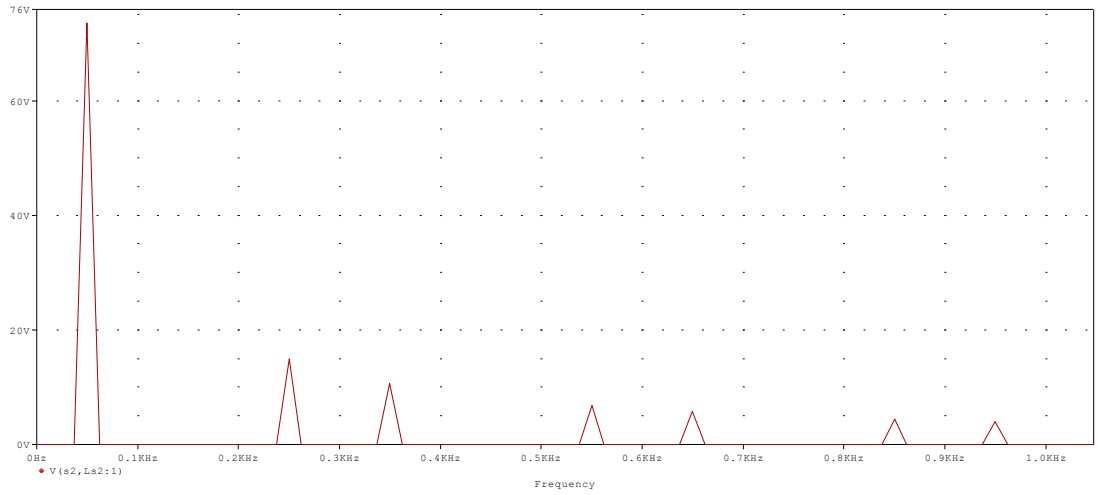


Figure 104 Fourier series Line Voltage of Inverter PSpice S_23.

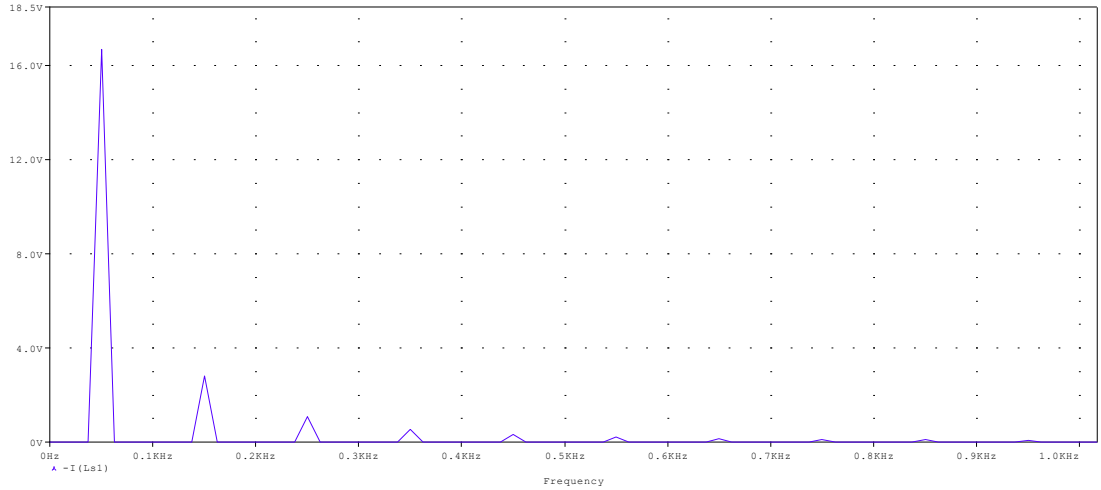


Figure 105 Fourier series Current of Inverter PSpice S_23.

ANNEX IV

DESCRIPTION OF THE *GUIDE MATLAB* COMPUTER PROGRAM

Author

Diego Salas Forns

Director

D. Francisco José Pérez Cebolla

INDEX

INDEX OF FIGURES

1 DESCRIPTION OF THE COMPUTER PROGRAM *GUIDE MATLAB*

1.1 WHAT IS *GUIDE MATLAB*?

Graphical User Interface, also known as GUI, is a computer program that acts as user interface, and that consists of providing a simple visual environment to allow communication with the operating system of a machine or, as in our case, a computer, using a set of images and graphic objects to represent the information and actions available in the interface.

Behind this interface must have a computer program which are implemented all operations and tasks that the user is sending from the interface.

Can be appointing a well-known interface like all Windows desktop environments, the X-Window GNU/Linux or Mac OS X, Aqua.

The creation of these interfaces is to create an easier to use and more intuitive visual environment that the software behind it [12-13].

1.2 HOW TO CREATE A *GUIDE MATLAB*

To create a GUI, you first have to get into MATLAB. Once inside MATLAB, you have to click the button that is shown in the image below (Figure 1):

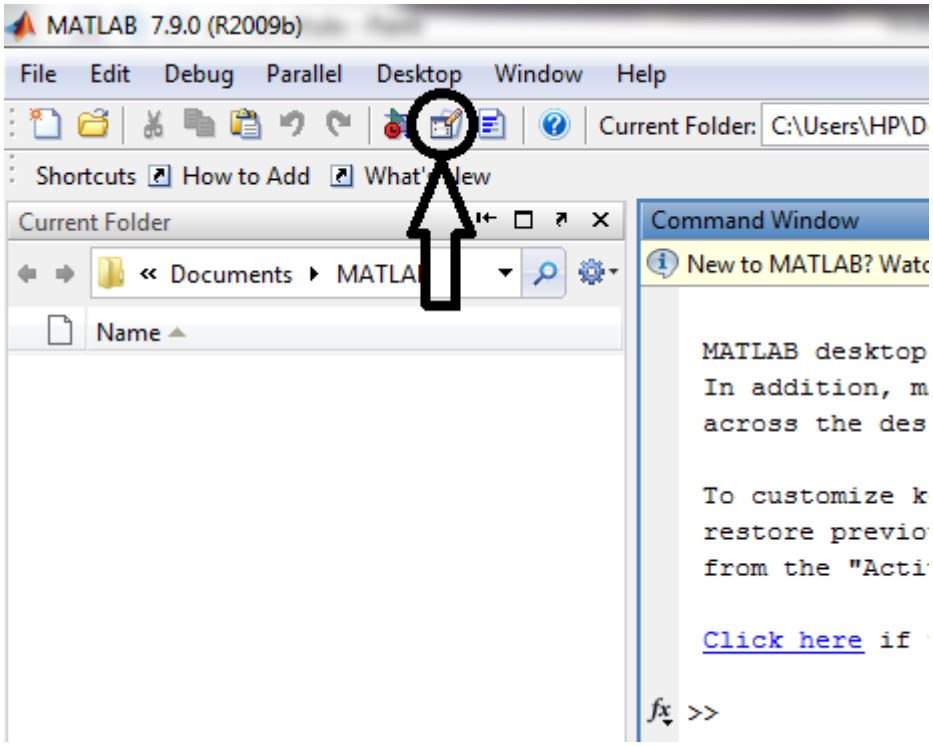


Figure 1 *GUIDE MATLAB* (1).

Clicking on that button, a new window will emerge, in which, following the instructions in the user manual of MATLAB, you can create a GUI as you want (Figure 2):

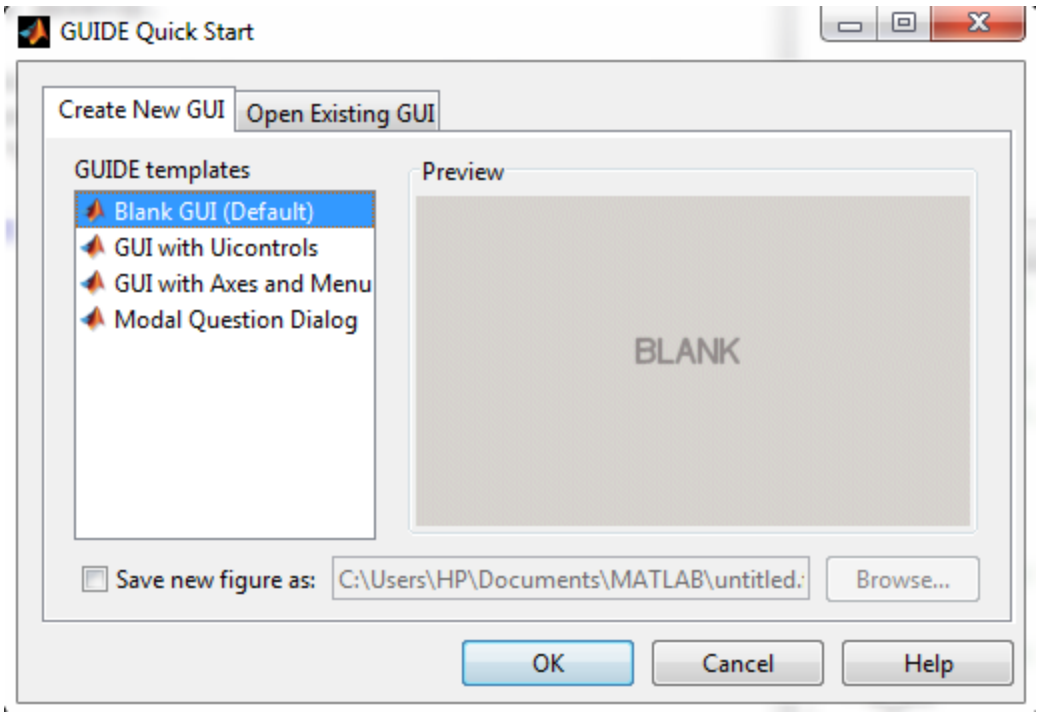


Figure 2 *GUIDE MATLAB* (2).

1.3 HOW TO ACCESS THE CODE PROGRAMMING

Having introduced all kinds of elements to be taken into the interface (Push Button, Slider, Edit text, Static text, Axes, etc.) There is a basic code that is automatically created, from which you have to enter code to make it work.

For example, for this project was created the following interface (Figure 3):

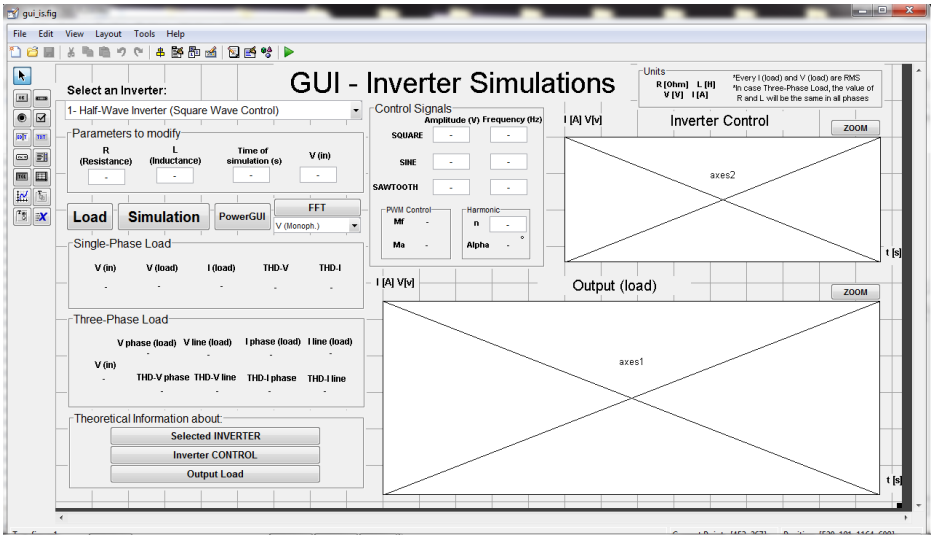


Figure 3 *GUIDE MATLAB* (3).

And to access the programming code of each element, need to click the right mouse button on the element you want to program and click: >> View Callbacks >> Callback.

For example with Push Button "Load" (Figure 4):

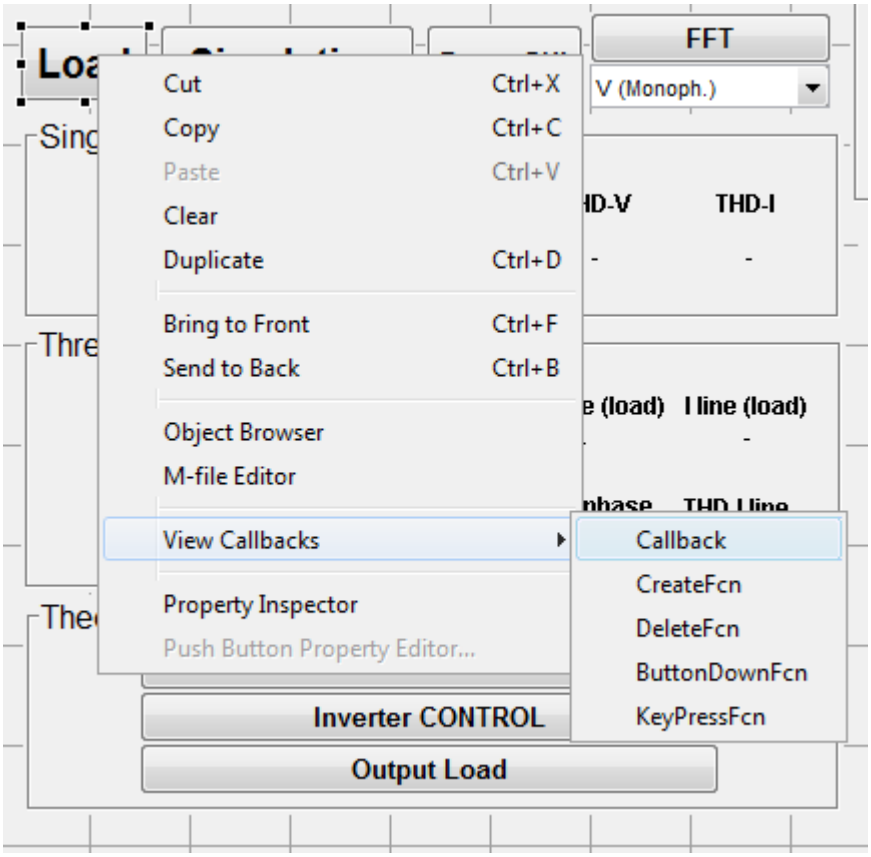


Figure 4 *GUIDE MATLAB* (4).

And will emerge a new window with the programming code in the place where the programming code of this element is found. In this case, the Push Button "Load" (Figure 5):

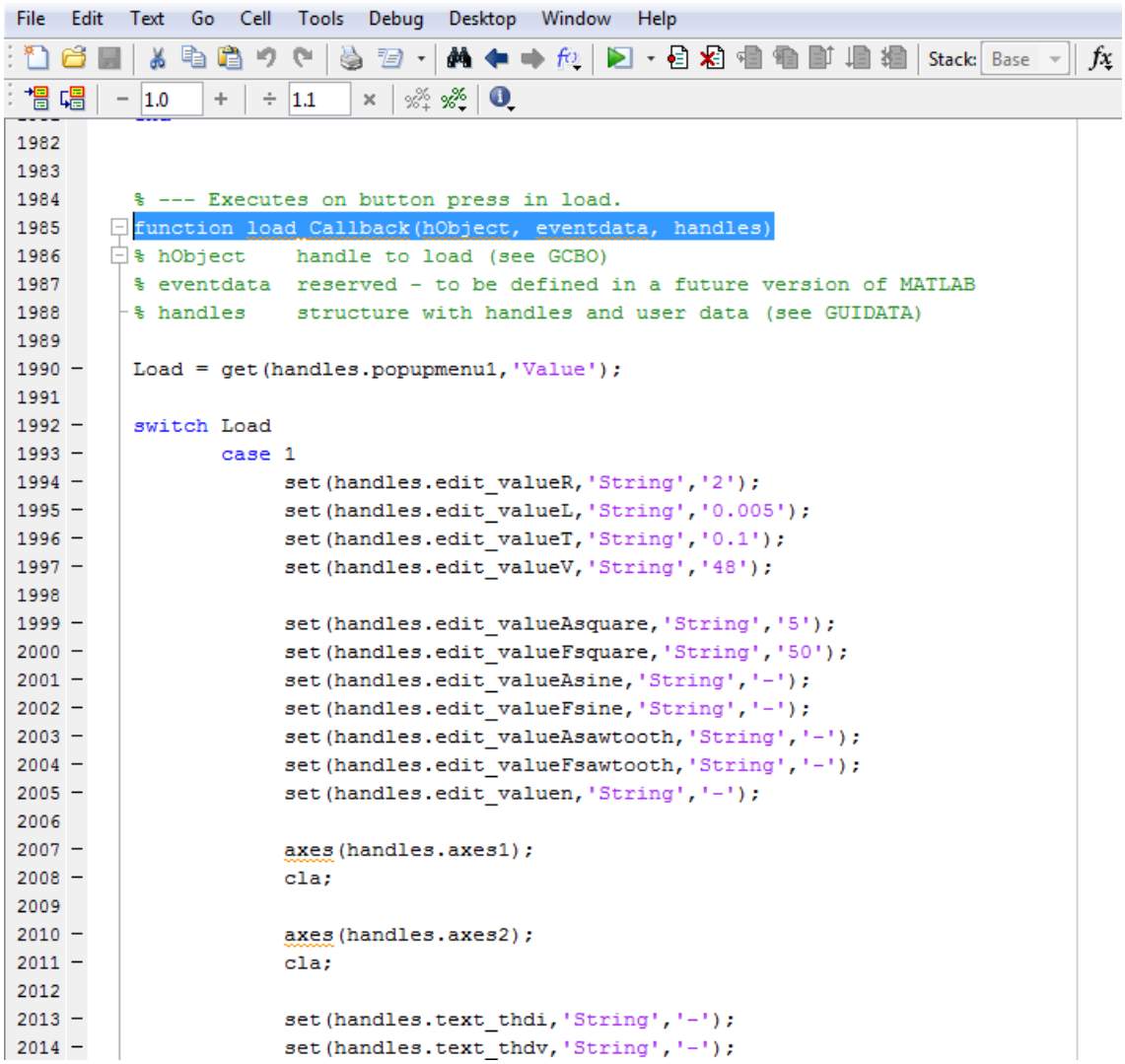


Figure 5 *GUIDE MATLAB* (5).

1.4 *GUIDE MATLAB* IN THIS PROJECT

As can be seen in Annex V, GUIDE MATLAB is used to create an interface that is able to control the simulation of "Simulink". Each simulation of "Simulink" has some parameters that can be modified:

- Input Voltage V(in).

- Resistance R.

- Inductance L.

- Time of Simulation.

- Every control signal of the inverter.

The main objective of the interface that was created in this project is to modify these parameters and get the waveform generated in the load as well as the control signal, and also the most representative numerical values as:

- V (load). Phase or line in case of Three-phase load.

- I (load). Phase or line in case of Three-phase load.

- THD-V. Phase or line in case of Three-phase load.

- THD-I. Phase or line in case of Three-phase load.

There are also some buttons, which when clicked a window pops up with theoretical information about:

- The selected inverter.

- Inverter Control.

- Output Load.

- FFT (Fourier analysis signals)

The following image (Figure 6) shows an example of the simulation of an inverter with the created interface:

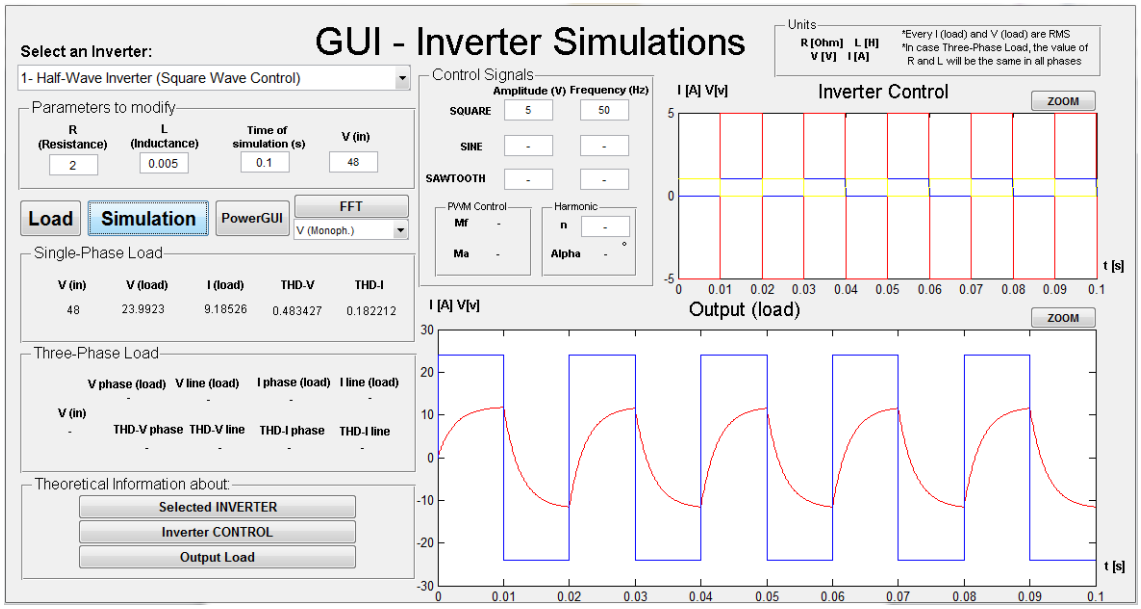


Figure 6 *GUIDE MATLAB* (6).

At first, you have to choose the inverter that you want to study, then you have to push “Load” button. Now, you can modify the parameters that you want, and finally you have to click “Simulation” button.

In the graphical user interface can be seen the graphic output load and inverter control. But to observe the frequency Fourier analysis (FFT), you first have to click the button "PowerGUI" and the following window opens (Figure 7):

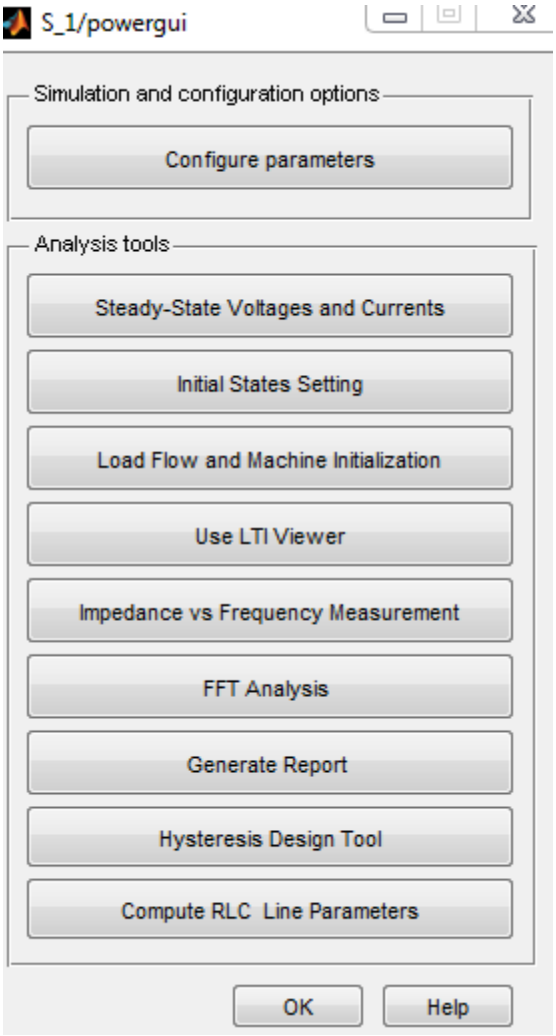


Figure 7 *GUIDE MATLAB* (7).

After opening the window is a button labeled “FFT Analysis”. Inside that, next window is opened (Figure 8):

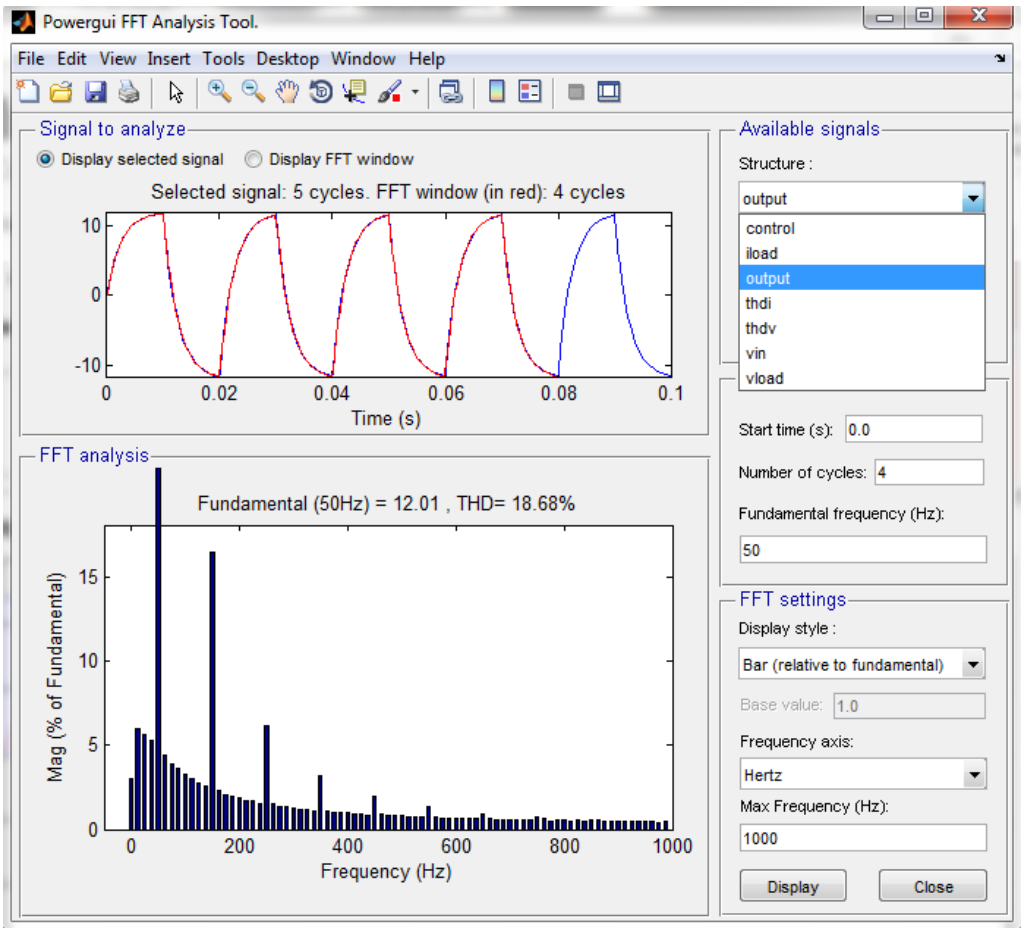


Figure 8 *GUIDE MATLAB* (8).

In “Available signals” you have to choose the “output” variable, and there you can see every output load signal. Finally you click “Display” and the FFT Analysis is plotted on the axes below.

Other way to open the FFT windows with the current and voltage frequency analysis is pushing “FFT” button, as it can see in figure 9:

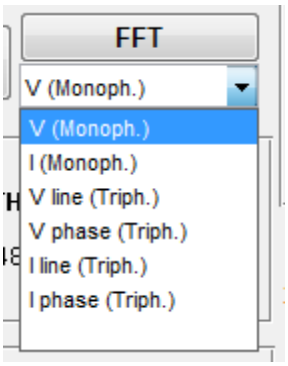


Figure 9 *GUIDE MATLAB* (9).

Depending on whether it is single phase or three phases, it has to choose the current or voltage monophasic or triphasic, and then, next windows are opened (for example V monophasic half wave Inverter, square wave control):

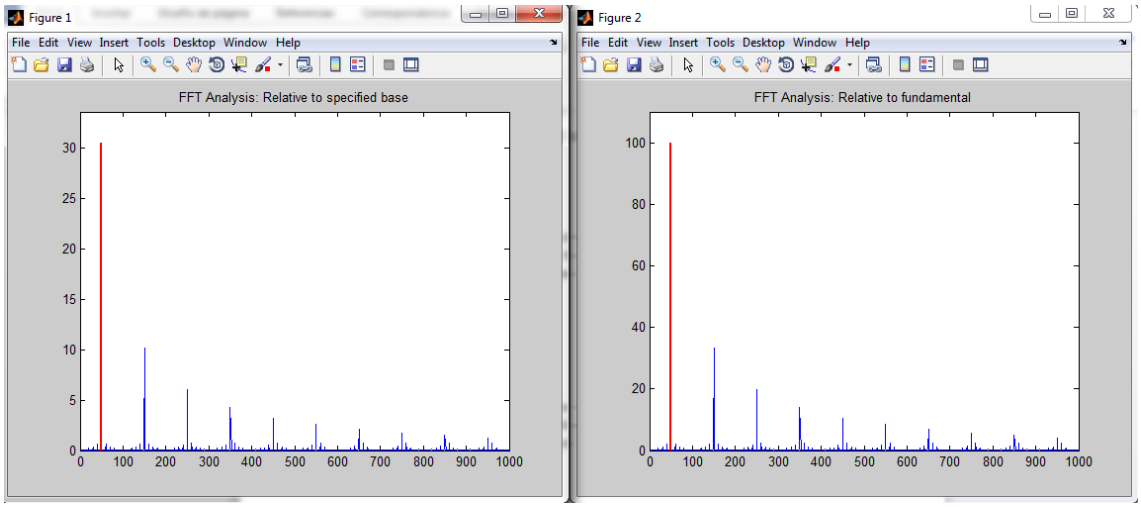


Figure 10 *GUIDE MATLAB* (10).

1.5 PROGRAM VERSIONS

Since 1984 there are several versions of this program, more than 35, but for the simulations with Simulink, and for building GUI for this Project, I have used the Version *MATLAB 7.8 R2009b* which stepped out into the market on 12th August of 2009.

ANNEX V

PROGRAMMING CODE *GUIDE MATLAB* TO CREATE THE INTERFACE.

Author

Diego Salas Forns

Director

D. Francisco José Pérez Cebolla

INDEX

INDEX OF FIGURES

1 PROGRAMMING CODE *GUIDE MATLAB* OF THE INTERFACE.

1.1 DESCRIPTION

Then it will show the end code of programming for interface that was created for this project [8], [12-13].

The following image shows all the elements that have been used to create the interface of this project. The next step is to access their "Callbacks", as explained in Annex IV, and proceed to writing programming code (Figure 1):

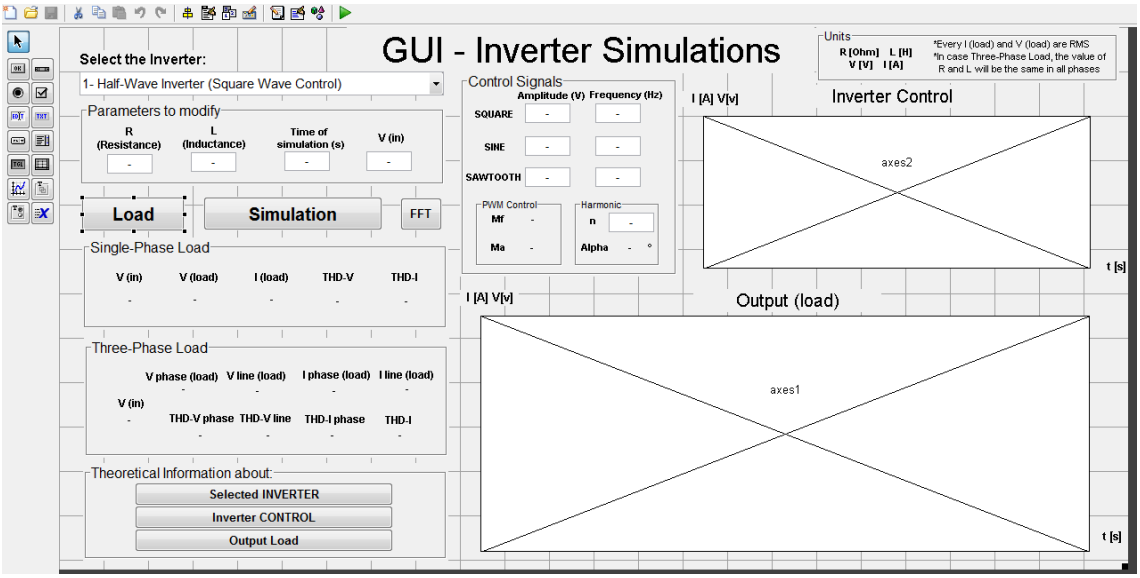


Figure 1 Editing screen interface.

1.2 PROGRAMMING CODE

```
function varargout = gui_is(varargin)
% GUI_IS M-file for gui_is.fig
%      GUI_IS, by itself, creates a new GUI_IS or raises the
existing
%      singleton*.
%
%      H = GUI_IS returns the handle to a new GUI_IS or the
handle to
%      the existing singleton*.
%
%      GUI_IS('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in GUI_IS.M with the given input
arguments.
%
```

```
%      GUI_IS('Property','Value',...) creates a new GUI_IS or
raises the
%      existing singleton*.  Starting from the left, property
value pairs are
%      applied to the GUI before gui_is_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes
property application
%      stop.  All inputs are passed to gui_is_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI
allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help gui_is

% Last Modified by GUIDE v2.5 24-Jul-2014 10:03:02

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @gui_is_OpeningFcn, ...
                   'gui_OutputFcn',  @gui_is_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before gui_is is made visible.
function gui_is_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to gui_is (see VARARGIN)

% Choose default command line output for gui_is
handles.output = hObject;
```

```
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes gui_is wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command
line.
function varargout = gui_is_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see
VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns
popupmenu1 contents as cell array
%        contents{get(hObject,'Value')} returns selected item
from popupmenu1


% --- Executes during object creation, after setting all
properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: popupmenu controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

1.2.1 THEORETICAL INFORMATION ABOUT:

*1.2.1.1 **SELECTED INVERTER***

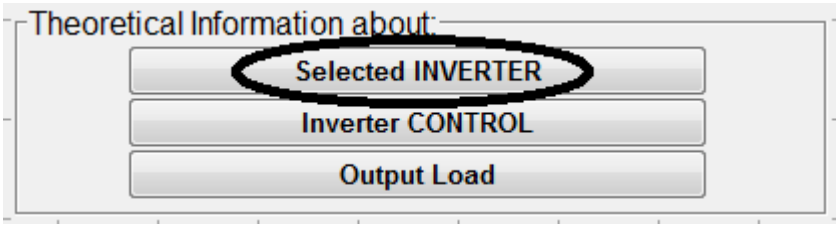


Figure 2 Theoretical Information about (1).

When it's done click the "Selected INVERTER" button (Figure 2), a window is opened with the theoretical information about that inverter.

```
% --- Executes on button press in theory_inverter.
function theory_inverter_Callback(hObject, eventdata, handles)
% hObject    handle to theory_inverter (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
Simulation = get(handles.popupmenu1,'Value');

web(['help' int2str(Simulation) '.html']);
```

*1.2.1.2 **INVERTER CONTROL***

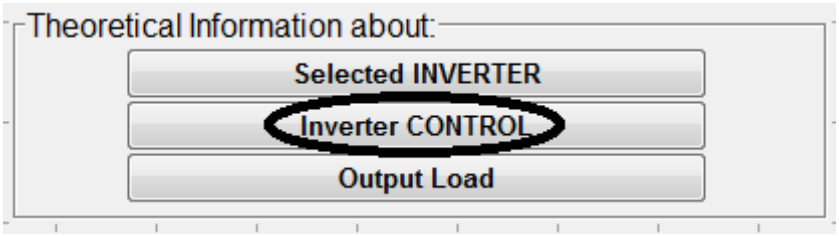


Figure 3 Theoretical Information about (2).

When it's done click the "Inverter CONTROL" button (Figure 3), a window is opened with the theoretical information about that invertir control.

```
% --- Executes on button press in theory_control.
function theory_control_Callback(hObject, eventdata, handles)
% hObject    handle to theory_control (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
Simulation = get(handles.popupmenu1,'Value');
```

```
web(['control' int2str(Simulation) '.html']);
```

*1.2.1.3 **OUTPUT LOAD***

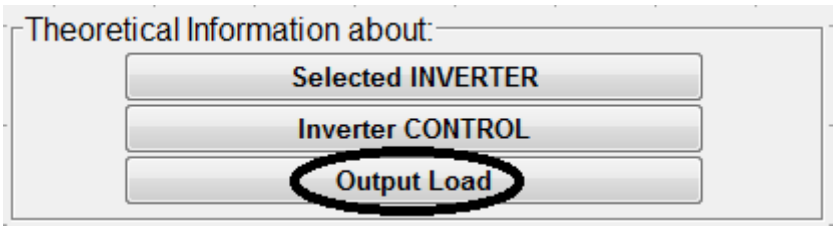


Figure 4 Theoretical Information about (3).

When it's done click the "Output Load" button (Figure 4), a window is opened with the theoretical information about the output load of that inverter.

```
% --- Executes on button press in theory_output.
function theory_output_Callback(hObject, eventdata, handles)
% hObject    handle to theory_output (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
Simulation = get(handles.popupmenu1,'Value');

web(['output' int2str(Simulation) '.html']);

function enlarge_output_Callback(hObject, eventdata, handles)
% hObject    handle to enlarge_output (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)



function edit_valueR_Callback(hObject, eventdata, handles)
% hObject    handle to edit_valueR (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_valueR
as text
%        str2double(get(hObject,'String')) returns contents of
edit_valueR
%        as a double

% --- Executes during object creation, after setting all
properties.
function edit_valueR_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_valueR (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit_valueL_Callback(hObject, eventdata, handles)
% hObject    handle to edit_valueL (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_valueL
as text
%        str2double(get(hObject,'String')) returns contents of
edit_valueL as a double


% --- Executes during object creation, after setting all
properties.
function edit_valueL_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_valueL (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit_valueT_Callback(hObject, eventdata, handles)
% hObject    handle to edit_valueT (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_valueT
as text
```

```
%        str2double(get(hObject,'String')) returns contents of
edit_valueT as a double


% --- Executes during object creation, after setting all
properties.
function edit_valueT_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_valueT (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

1.2.2 SIMULATION

Figure 5 Button simulation.

When it is done click on the button named "Simulation" (Figure 5), the program performs the following iterations:

```
% --- Executes on button press in simulation.
function simulation_Callback(hObject, eventdata, handles)
% hObject    handle to simulation (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

1.2.2.1 PARAMETERS TO MODIFY

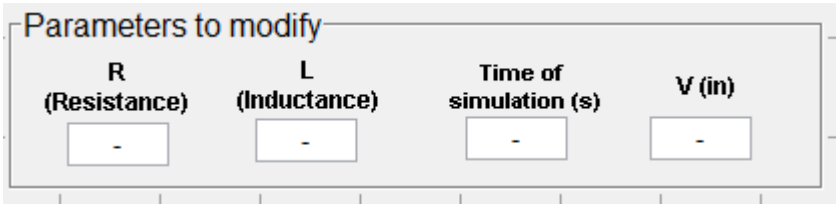


Figure 6 Parameters to modify.

First, the program saves the values that have been written in Parameters to Modify (Figure 6):

```
R_value = get(handles.edit_valueR,'String');
L_value = get(handles.edit_valueL,'String');
T_value = get(handles.edit_valueT,'String');
V_value = get(handles.edit_valueV,'String');
```

*1.2.2.2 **CONTROL SIGNALS***

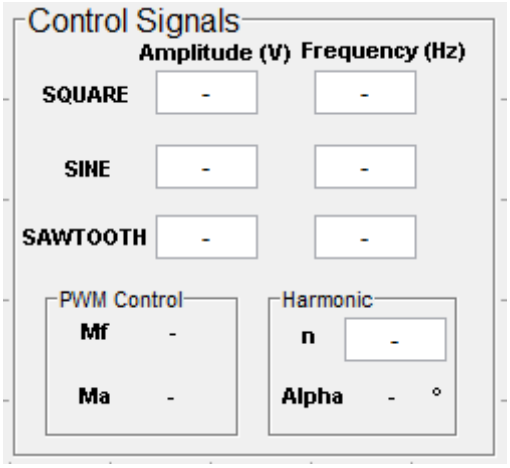


Figure 7 Control Signals.

Second, the program saves the values that have been written in Control Signals (Figure 7):

```
Asquare_value = get(handles.edit_valueAsquare,'String');
Fsquare_value = get(handles.edit_valueFsquare,'String');
Asine_value = get(handles.edit_valueAsine,'String');
Fsine_value = get(handles.edit_valueFsine,'String');
Asawtooth_value = get(handles.edit_valueAsawtooth,'String');
Fsawtooth_value = get(handles.edit_valueFsawtooth,'String');
n_value = get(handles.edit_valuen,'String');
```

To make the program work, the values entered must meet some conditions: The appropriate boxes must not be empty, must be numbers, the numbers must be greater than 0, etc.

```
if ~isempty(R_value)&& ~isempty(L_value) && ~isempty(V_value) &&
~isempty(T_value) && isnumeric(str2double(R_value))&&
isnumeric(str2double(L_value)) && isnumeric(str2double(T_value))
&& isnumeric(str2double(V_value)) && str2double(V_value)>0 &&
str2double(R_value)>0 && str2double(L_value)>0 &&
str2double(T_value)>0
```

*1.2.2.3 **POP-UP MENU "SELECT THE INVERTER"***

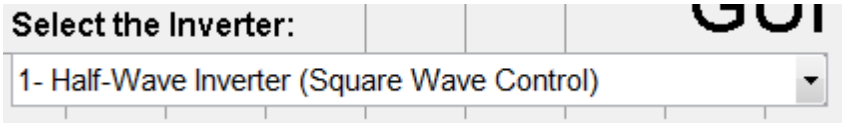


Figure 8 Select the Inverter.

Inside the Pop-Up menu (Figure 8), 23 different inverters, each has a position of 1 to 23. The following function is used to get a numerical value depending on the chosen inverter.

```
    Simulation = get(handles.popupmenu1,'Value');
```

If chosen the first inverter, the following function enters in Case 1:

```
switch Simulation
        case 1
```

Now, has to load the Simulink file corresponding to this inverter. In this case the file "S_1":

```
            load_system('S_1');
```

And then with the parameters introduced above, the program proceeds to change the appropriate values into *Simulink* file:

```
set_param(['S_1/RL'],'Resistance',R_value,'Inductance',L_value);
            set_param(['S_1'],'StopTime',T_value);
            V = str2double(V_value)/2;
            V = num2str(V);
            set_param(['S_1/subsystem/DC Voltage
Source'],'Amplitude',V);
            set_param(['S_1/subsystem/DC Voltage
Source1'],'Amplitude',V);
            set_param(['S_1/Signal
Generator'],'Amplitude',Asquare_value,
'Frequency',Fsquare_value);
```

As before, it must prove that the parameters entered are correct. If not correct emerges an error window:

```
            if ~isempty(Asquare_value)&& ~isempty(Fsquare_value)
&& isnumeric(str2double(Asquare_value))&&
isnumeric(str2double(Fsquare_value)) &&
str2double(Asquare_value)>0 && str2double(Fsquare_value)>0
                sim ('S_1');
            else
                errordlg('ERROR: Invalid Number of Control
Signals');
            end;
```

In *Simulink* file, there are some Workspace variables, which are vectors. Therefore, it has to create new variables that the last value of these vectors is saved:

```
time = output.time;
output1 = output.signals.values(:,1);
output2 = output.signals.values(:,2);
control1 = control.signals.values(:,1);
control2 = control.signals.values(:,2);
control3 = control.signals.values(:,3);
thdi = thdi.signals.values(:,1);
thdv = thdv.signals.values(:,1);
vload = vload.signals.values(:,1);
iload = iload.signals.values(:,1);
vin = vin.signals.values(:,1);
```

1.2.2.4 AXES 1

Variables: "Output" and "Control" are plotted in the following axes over time (Figure 9 and 10).

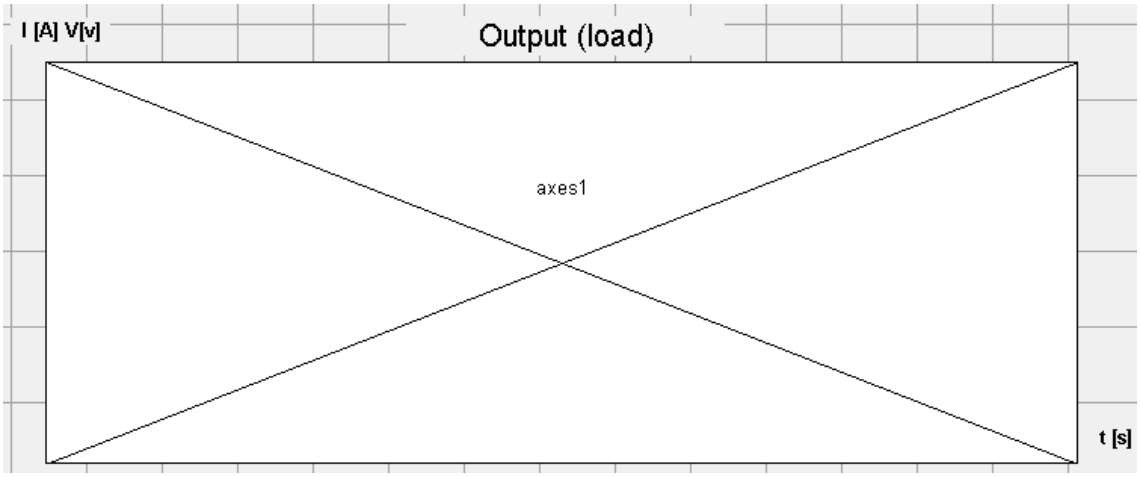


Figure 9 Axes 1

```
axes(handles.axes1);
cla;

plot(handles.axes1,time,output1,'r');
hold on;
plot(handles.axes1,time,output2,'b');
hold off;
```

1.2.2.5 **AXES 2**

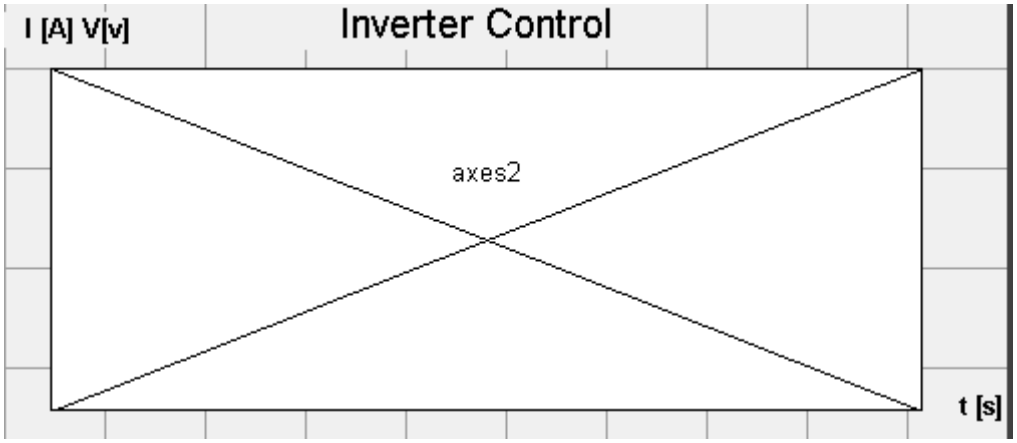


Figure 10 Axes 2.

```
axes(handles.axes2);
cla;

plot(handles.axes2,time,control1,'r');
hold on;
plot(handles.axes2,time,control2,'b');
plot(handles.axes2,time,control3,'y');
hold off;
```

*1.2.2.6 **OUTPUT VALUES***

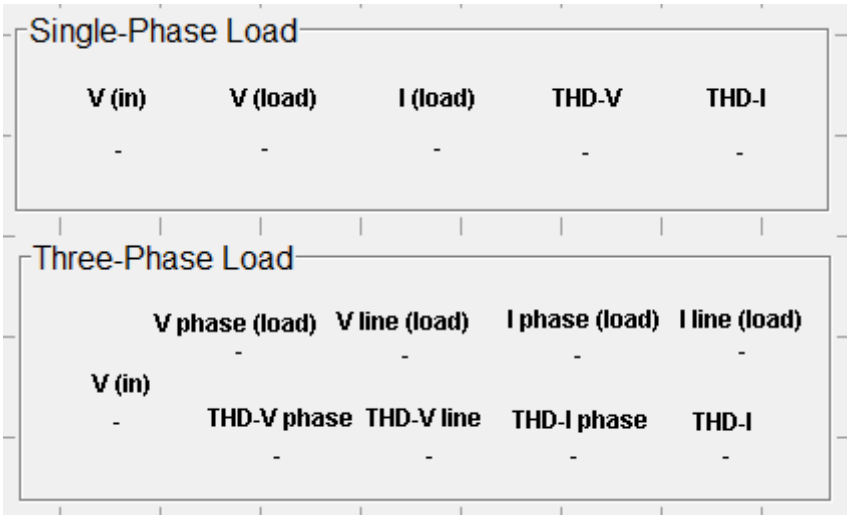


Figure 11 Output Values.

This example corresponds to a single phase load. So be displayed values corresponding to the single phase load above variables.

```
set(handles.text_thdi,'String',abs(thdi(end)));
set(handles.text_thdv,'String',abs(thdv(end)));
```

```
            set(handles.text_vin,'String',abs(vin(end)));
            set(handles.text_vload,'String',abs(vload(end)));
            set(handles.text_iload,'String',abs(iload(end)));
```

And as is observed in this example, the values corresponding to a three phase load are with a '-':

```
            set(handles.text_thdif,'String','-');
            set(handles.text_thdil,'String','-');
            set(handles.text_thdvf,'String','-');
            set(handles.text_thdvl,'String','-');
            set(handles.text_vin3,'String','-');
            set(handles.text_vloadf,'String','-');
            set(handles.text_vloadl,'String','-');
            set(handles.text_iloadf,'String','-');
            set(handles.text_iloadl,'String','-');
```

The same happens with the following 22 cases. It shows a three phases case as an example, and the especial case 6 (6- Full Wave H-Bridge Inverter (Voltage Harmonic Cancellation)):

```
        case 2

        case 3

        case 4

        case 5

        case 6

load_system('S_6');

set_param(['S_6/RL'],'Resistance',R_value,'Inductance',L_value);
            set_param(['S_6'],'StopTime',T_value);
            V = V_value;
            set_param(['S_6/subsystem/DC Voltage
Source1'],'Amplitude',V);
            set_param(['S_6/Signal
Generator'],'Amplitude',Asine_value, 'Frequency',Fsine_value);

            Alpha = 90/str2double(n_value);
            A = Alpha*2;
            B = 360/A;
            C = 1/str2double(Fsine_value);
            D = C/B;
            Delay_value = num2str(D);
            set_param(['S_6/Transport
Delay1'],'DelayTime',Delay_value);

            if ~isempty(Asine_value)&& ~isempty(Fsine_value) &&
isnumeric(str2double(Asine_value))&&
isnumeric(str2double(Fsine_value))  && str2double(Asine_value)>0
&& str2double(Fsine_value)>0
                if ~isempty(n_value) &&
isnumeric(str2double(n_value)) && str2double(n_value)>1
```

```
                    if mod(n_value,2)==1
                       sim ('S_6');
                    else
                       errordlg('ERROR: "n" is a even number.
"n" has to be a ODD number');
                    end;
                else
                    errordlg('ERROR: Invalid Number of
Harmonic');
                end;
            else
                errordlg('ERROR: Invalid Number of Control
Signals');
            end;



            time = output.time;
            output1 = output.signals.values(:,1);
            output2 = output.signals.values(:,2);
            control1 = control.signals.values(:,1);
            control2 = control.signals.values(:,2);
            control3 = control.signals.values(:,3);
            thdi = thdi.signals.values(:,1);
            thdv = thdv.signals.values(:,1);
            vload = vload.signals.values(:,1);
            iload = iload.signals.values(:,1);
            vin = vin.signals.values(:,1);

            axes(handles.axes1);
            cla;

            plot(handles.axes1,time,output1,'r');
            hold on;
            plot(handles.axes1,time,output2,'b');
            hold off;

            axes(handles.axes2);
            cla;

            plot(handles.axes2,time,control1,'r');
            hold on;
            plot(handles.axes2,time,control2,'b');
            plot(handles.axes2,time,control3,'y');
            hold off;

            set(handles.text_thdi,'String',abs(thdi(end)));
            set(handles.text_thdv,'String',abs(thdv(end)));
            set(handles.text_vin,'String',abs(vin(end)));
            set(handles.text_vload,'String',abs(vload(end)));
            set(handles.text_iload,'String',abs(iload(end)));

            set(handles.text_thdif,'String','-');
            set(handles.text_thdil,'String','-');
```

```
            set(handles.text_thdvf,'String','-');
            set(handles.text_thdvl,'String','-');
            set(handles.text_vin3,'String','-');
            set(handles.text_vloadf,'String','-');
            set(handles.text_vloadl,'String','-');
            set(handles.text_iloadf,'String','-');
            set(handles.text_iloadl,'String','-');

            set(handles.text_Alpha,'String',Alpha);


        case 7

        case 8

        case 9

            load_system('S_9');

set_param(['S_9/RL1'],'Resistance',R_value,'Inductance',L_value)
;

set_param(['S_9/RL2'],'Resistance',R_value,'Inductance',L_value)
;

set_param(['S_9/RL3'],'Resistance',R_value,'Inductance',L_value)
;
            set_param(['S_9'],'StopTime',T_value);
            V = V_value;
            set_param(['S_9/subsystem/DC Voltage
Source1'],'Amplitude',V);
            set_param(['S_9/Signal
Generator'],'Amplitude',Asquare_value,
'Frequency',Fsquare_value);

            if ~isempty(Asquare_value)&& ~isempty(Fsquare_value)
&& isnumeric(str2double(Asquare_value))&&
isnumeric(str2double(Fsquare_value)) &&
str2double(Asquare_value)>0 && str2double(Fsquare_value)>0
                sim ('S_9');
            else
                errordlg('ERROR: Invalid Number of Control
Signals');
            end;

            time = output.time;
            output1 = output.signals.values(:,1);
            output2 = output.signals.values(:,2);
            output3 = output.signals.values(:,3);
            control1 = control.signals.values(:,1);
            control2 = control.signals.values(:,2);
            control3 = control.signals.values(:,3);
            thdif = thdif.signals.values(:,1);
            thdil = thdil.signals.values(:,1);
```

```
    thdvf = thdvf.signals.values(:,1);
    thdvl = thdvl.signals.values(:,1);
    vloadf = vloadf.signals.values(:,1);
    vloadl = vloadl.signals.values(:,1);
    iloadf = iloadf.signals.values(:,1);
    iloadl = iloadl.signals.values(:,1);
    vin = vin.signals.values(:,1);

    axes(handles.axes1);
    cla;

    plot(handles.axes1,time,output1,'r');
    hold on;
    plot(handles.axes1,time,output2,'b');
    plot(handles.axes1,time,output3,'y');
    hold off;

    axes(handles.axes2);
    cla;

    plot(handles.axes2,time,control1,'r');
    hold on;
    plot(handles.axes2,time,control2,'b');
    plot(handles.axes2,time,control3,'y');
    hold off;

    set(handles.text_thdif,'String',abs(thdif(end)));
    set(handles.text_thdil,'String',abs(thdil(end)));
    set(handles.text_thdvf,'String',abs(thdvf(end)));
    set(handles.text_thdvl,'String',abs(thdvl(end)));
    set(handles.text_vin3,'String',abs(vin(end)));
    set(handles.text_vloadf,'String',abs(vloadf(end)));
    set(handles.text_vloadl,'String',abs(vloadl(end)));
    set(handles.text_iloadf,'String',abs(iloadf(end)));
    set(handles.text_iloadl,'String',abs(iloadl(end)));

    set(handles.text_thdi,'String','-');
    set(handles.text_thdv,'String','-');
    set(handles.text_vin,'String','-');
    set(handles.text_vload,'String','-');
    set(handles.text_iload,'String','-');

case 10

case 11

case 12

case 13

case 14

case 15
```

```
        case 16

        case 17

        case 18

        case 19

        case 20

        case 21

        case 22

        case 23


         otherwise
              disp('Other Simulation');
    end
else
    errordlg('ERROR: Invalid Number of Parameters to modify');

end



function edit_valueV_Callback(hObject, eventdata, handles)
% hObject    handle to edit_valueV (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_valueV
as text
%        str2double(get(hObject,'String')) returns contents of
edit_valueV as a double


% --- Executes during object creation, after setting all
properties.
function edit_valueV_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_valueV (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end



function edit_valueAsquare_Callback(hObject, eventdata, handles)
% hObject    handle to edit_valueAsquare (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of
edit_valueAsquare as text
%        str2double(get(hObject,'String')) returns contents of
edit_valueAsquare as a double


% --- Executes during object creation, after setting all
properties.
function edit_valueAsquare_CreateFcn(hObject, eventdata,
handles)
% hObject    handle to edit_valueAsquare (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit_valueFsquare_Callback(hObject, eventdata, handles)
% hObject    handle to edit_valueFsquare (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of
edit_valueFsquare as text
%        str2double(get(hObject,'String')) returns contents of
edit_valueFsquare as a double


% --- Executes during object creation, after setting all
properties.
function edit_valueFsquare_CreateFcn(hObject, eventdata,
handles)
% hObject    handle to edit_valueFsquare (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit_valueAsine_Callback(hObject, eventdata, handles)
% hObject    handle to edit_valueAsine (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of
edit_valueAsine as text
%        str2double(get(hObject,'String')) returns contents of
edit_valueAsine as a double


% --- Executes during object creation, after setting all
properties.
function edit_valueAsine_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_valueAsine (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit_valueFsine_Callback(hObject, eventdata, handles)
% hObject    handle to edit_valueFsine (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of
edit_valueFsine as text
```

```
%        str2double(get(hObject,'String')) returns contents of
edit_valueFsine as a double


% --- Executes during object creation, after setting all
properties.
function edit_valueFsine_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_valueFsine (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit_valueAsawtooth_Callback(hObject, eventdata,
handles)
% hObject    handle to edit_valueAsawtooth (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of
edit_valueAsawtooth as text
%        str2double(get(hObject,'String')) returns contents of
edit_valueAsawtooth as a double


% --- Executes during object creation, after setting all
properties.
function edit_valueAsawtooth_CreateFcn(hObject, eventdata,
handles)
% hObject    handle to edit_valueAsawtooth (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit_valueFsawtooth_Callback(hObject, eventdata,
handles)
% hObject    handle to edit_valueFsawtooth (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of
edit_valueFsawtooth as text
%        str2double(get(hObject,'String')) returns contents of
edit_valueFsawtooth as a double


% --- Executes during object creation, after setting all
properties.
function edit_valueFsawtooth_CreateFcn(hObject, eventdata,
handles)
% hObject    handle to edit_valueFsawtooth (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit_valuen_Callback(hObject, eventdata, handles)
% hObject    handle to edit_valuen (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_valuen
as text
%        str2double(get(hObject,'String')) returns contents of
edit_valuen as a double


% --- Executes during object creation, after setting all
properties.
function edit_valuen_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_valuen (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
```

```
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit_valueTpulse_Callback(hObject, eventdata, handles)
% hObject    handle to edit_valueTpulse (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of
edit_valueTpulse as text
%        str2double(get(hObject,'String')) returns contents of
edit_valueTpulse as a double


% --- Executes during object creation, after setting all
properties.
function edit_valueTpulse_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_valueTpulse (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

1.2.3 LOAD

Figure 12 Button Load.

The role of this button (Figure 12) is to set a standart values in the appropriate boxes, after having chosen the investor to be analyzed, and before clicking the button "Simulation".

```
% --- Executes on button press in load.
function load_Callback(hObject, eventdata, handles)
% hObject    handle to load (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

Load = get(handles.popupmenu1,'Value');

switch Load
       case 1
            set(handles.edit_valueR,'String','2');
            set(handles.edit_valueL,'String','0.005');
            set(handles.edit_valueT,'String','0.1');
            set(handles.edit_valueV,'String','48');

            set(handles.edit_valueAsquare,'String','5');
            set(handles.edit_valueFsquare,'String','50');
            set(handles.edit_valueAsine,'String','-');
            set(handles.edit_valueFsine,'String','-');
            set(handles.edit_valueAsawtooth,'String','-');
            set(handles.edit_valueFsawtooth,'String','-');
            set(handles.edit_valuen,'String','-');

            axes(handles.axes1);
            cla;

            axes(handles.axes2);
            cla;

            set(handles.text_thdi,'String','-');
            set(handles.text_thdv,'String','-');
            set(handles.text_vin,'String','-');
            set(handles.text_vload,'String','-');
            set(handles.text_iload,'String','-');

            set(handles.text_thdif,'String','-');
            set(handles.text_thdil,'String','-');
            set(handles.text_thdvf,'String','-');
            set(handles.text_thdvl,'String','-');
            set(handles.text_vin3,'String','-');
            set(handles.text_vloadf,'String','-');
            set(handles.text_vloadl,'String','-');
            set(handles.text_iloadf,'String','-');
            set(handles.text_iloadl,'String','-');

            set(handles.text_mf,'String','-');
            set(handles.text_ma,'String','-');
            set(handles.text_Alpha,'String','-');
```

The same happens with the following 22 cases. It shows a three phases case as an example, and the especial case 6 (6- Full Wave H-Bridge Inverter (Voltage Harmonic Cancellation)):

```
case 2

case 3

case 4

case 5

case 6

    set(handles.edit_valueR,'String','2');
    set(handles.edit_valueL,'String','0.005');
    set(handles.edit_valueT,'String','0.1');
    set(handles.edit_valueV,'String','48');

    set(handles.edit_valueAsquare,'String','-');
    set(handles.edit_valueFsquare,'String','-');
    set(handles.edit_valueAsine,'String','1');
    set(handles.edit_valueFsine,'String','50');
    set(handles.edit_valueAsawtooth,'String','-');
    set(handles.edit_valueFsawtooth,'String','-');
    set(handles.edit_valueFsawtooth,'String','-');
    set(handles.edit_valuen,'String','3');

    axes(handles.axes1);
    cla;

    axes(handles.axes2);
    cla;

    set(handles.text_thdi,'String','-');
    set(handles.text_thdv,'String','-');
    set(handles.text_vin,'String','-');
    set(handles.text_vload,'String','-');
    set(handles.text_iload,'String','-');

    set(handles.text_thdif,'String','-');
    set(handles.text_thdil,'String','-');
    set(handles.text_thdvf,'String','-');
    set(handles.text_thdvl,'String','-');
    set(handles.text_vin3,'String','-');
    set(handles.text_vloadf,'String','-');
    set(handles.text_vloadl,'String','-');
    set(handles.text_iloadf,'String','-');
    set(handles.text_iloadl,'String','-');
    set(handles.text_Alpha,'String','30');

    set(handles.text_mf,'String','-');
    set(handles.text_ma,'String','-');

case 7

case 8
```

```
case 9

    set(handles.edit_valueR,'String','2');
    set(handles.edit_valueL,'String','0.005');
    set(handles.edit_valueT,'String','0.1');
    set(handles.edit_valueV,'String','48');

    set(handles.edit_valueAsquare,'String','5');
    set(handles.edit_valueFsquare,'String','50');
    set(handles.edit_valueAsine,'String','-');
    set(handles.edit_valueFsine,'String','-');
    set(handles.edit_valueAsawtooth,'String','-');
    set(handles.edit_valueFsawtooth,'String','-');
    set(handles.edit_valuen,'String','-');

    axes(handles.axes1);
    cla;

    axes(handles.axes2);
    cla;

    set(handles.text_thdi,'String','-');
    set(handles.text_thdv,'String','-');
    set(handles.text_vin,'String','-');
    set(handles.text_vload,'String','-');
    set(handles.text_iload,'String','-');

    set(handles.text_thdif,'String','-');
    set(handles.text_thdil,'String','-');
    set(handles.text_thdvf,'String','-');
    set(handles.text_thdvl,'String','-');
    set(handles.text_vin3,'String','-');
    set(handles.text_vloadf,'String','-');
    set(handles.text_vloadl,'String','-');
    set(handles.text_iloadf,'String','-');
    set(handles.text_iloadl,'String','-');

    set(handles.text_mf,'String','-');
    set(handles.text_ma,'String','-');
    set(handles.text_Alpha,'String','-');

case 10

case 11

case 12

case 13

case 14

case 15
```

```
        case 16

        case 17

        case 18

        case 19

        case 20

        case 21

        case 22

        case 23

    otherwise
        disp('Other Simulation');
end;



function edit_valueDelay_Callback(hObject, eventdata, handles)
% hObject    handle to edit_valueDelay (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of
edit_valueDelay as text
%        str2double(get(hObject,'String')) returns contents of
edit_valueDelay as a double


% --- Executes during object creation, after setting all
properties.
function edit_valueDelay_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_valueDelay (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

1.2.4 PowerGUI

Figure 13 PowerGUI Button.

Within each Simulink simulation, there is a box called "PowerGUI", in which can be visualized different parameters and graphs, including frequency analysis of the harmonics with the Fourier transform.

This function (Figure 13) is responsible for loading all necessary values that box called "PowerGUI" to work correctly.

```
% --- Executes on button press in powergui.
function powergui_Callback(hObject, eventdata, handles)
% hObject    handle to powergui (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

R_value = get(handles.edit_valueR,'String');
L_value = get(handles.edit_valueL,'String');
T_value = get(handles.edit_valueT,'String');
V_value = get(handles.edit_valueV,'String');
Asquare_value = get(handles.edit_valueAsquare,'String');
Fsquare_value = get(handles.edit_valueFsquare,'String');
Asine_value = get(handles.edit_valueAsine,'String');
Fsine_value = get(handles.edit_valueFsine,'String');
Asawtooth_value = get(handles.edit_valueAsawtooth,'String');
Fsawtooth_value = get(handles.edit_valueFsawtooth,'String');
n_value = get(handles.edit_valuen,'String');

Simulation = get(handles.popupmenu1,'Value');

switch Simulation
        case 1
            load_system('S_1');

set_param(['S_1/RL'],'Resistance',R_value,'Inductance',L_value);
            set_param(['S_1'],'StopTime',T_value);
            V = str2double(V_value)/2;
            V = num2str(V);
            set_param(['S_1/subsystem/DC Voltage
Source'],'Amplitude',V);
            set_param(['S_1/subsystem/DC Voltage
Source1'],'Amplitude',V);
            set_param(['S_1/Signal
Generator'],'Amplitude',Asquare_value,
'Frequency',Fsquare_value);
            cy=(str2double(T_value)*str2double(Fsquare_value)-
1);
```

```
            cy=ceil(cy);
            cy=num2str(cy);
            set_param(['S_1/powergui'],'SampleTime',T_value,
'frequency', Fsquare_value, 'cycles', cy, 'fundamental',
Fsquare_value);

            if ~isempty(Asquare_value)&& ~isempty(Fsquare_value)
&& isnumeric(str2double(Asquare_value))&&
isnumeric(str2double(Fsquare_value)) &&
str2double(Asquare_value)>0 && str2double(Fsquare_value)>0
                evalin('base','sim(''S_1'')');
                powergui;
            else
                errordlg('ERROR: Invalid Number of Control
Signals');
            end;
```

The same happens with the following 22 cases. It shows a three phases case as an example, and the especial case 6 (6- Full Wave H-Bridge Inverter (Voltage Harmonic Cancellation)):

```
        case 2

        case 3

        case 4

        case 5

        case 6
             load_system('S_6');

set_param(['S_6/RL'],'Resistance',R_value,'Inductance',L_value);
            set_param(['S_6'],'StopTime',T_value);
            V = V_value;
            set_param(['S_6/subsystem/DC Voltage
Source1'],'Amplitude',V);
            set_param(['S_6/Signal
Generator'],'Amplitude',Asine_value, 'Frequency',Fsine_value);
            cy=(str2double(T_value)*str2double(Fsine_value)-1);
            cy=ceil(cy);
            cy=num2str(cy);
            set_param(['S_6/powergui'],'SampleTime',T_value,
'frequency', Fsine_value, 'cycles', cy, 'fundamental',
Fsine_value);

            Alpha = 90/str2double(n_value);
            A = Alpha*2;
            B = 360/A;
            C = 1/str2double(Fsine_value);
            D = C/B;
            Delay_value = num2str(D);
```

```
            set_param(['S_6/Transport
Delay1'],'DelayTime',Delay_value);

            if ~isempty(Asine_value)&& ~isempty(Fsine_value) &&
isnumeric(str2double(Asine_value))&&
isnumeric(str2double(Fsine_value))  && str2double(Asine_value)>0
&& str2double(Fsine_value)>0
                if ~isempty(n_value) &&
isnumeric(str2double(n_value)) && str2double(n_value)>1
                    evalin('base','sim(''S_6'')');
                    powergui;
                else
                    errordlg('ERROR: Invalid Number of
Harmonic');
                end;
            else
                errordlg('ERROR: Invalid Number of Control
Signals');
            end;


        case 7

        case 8

        case 9
             load_system('S_9');

set_param(['S_9/RL1'],'Resistance',R_value,'Inductance',L_value)
;

set_param(['S_9/RL2'],'Resistance',R_value,'Inductance',L_value)
;

set_param(['S_9/RL3'],'Resistance',R_value,'Inductance',L_value)
;
            set_param(['S_9'],'StopTime',T_value);
            V = V_value;
            set_param(['S_9/subsystem/DC Voltage
Source1'],'Amplitude',V);
            set_param(['S_9/Signal
Generator'],'Amplitude',Asquare_value,
'Frequency',Fsquare_value);
            cy=(str2double(T_value)*str2double(Fsquare_value)-
1);
            cy=ceil(cy);
            cy=num2str(cy);
            set_param(['S_9/powergui'],'SampleTime',T_value,
'frequency', Fsquare_value, 'cycles', cy, 'fundamental',
Fsquare_value);

            if ~isempty(Asquare_value)&& ~isempty(Fsquare_value)
&& isnumeric(str2double(Asquare_value))&&
```

```
isnumeric(str2double(Fsquare_value)) &&
str2double(Asquare_value)>0 && str2double(Fsquare_value)>0
                evalin('base','sim(''S_9'')');
                powergui;
            else
                errordlg('ERROR: Invalid Number of Control
Signals');
            end;

        case 10

        case 11

        case 12

        case 13

        case 14

        case 15

        case 16

        case 17

        case 18

        case 19

        case 20

        case 21

        case 22

        case 23

    otherwise
        disp('Other Simulation');
end;
```

1.2.5 FFT

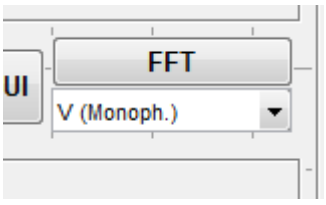


Figure 14 FFT button

Following senences cover to the programming code for the FFT Analysis (Fourier Analysis). First it has to be selected the type of wave (current or voltage, monophase or three phases), and then pulse de FFT button (figure 14).

```
% --- Executes on button press in fft_button.
function fft_button_Callback(hObject, eventdata, handles)
% hObject    handle to fft_button (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
R_value = get(handles.edit_valueR,'String');
L_value = get(handles.edit_valueL,'String');
T_value = get(handles.edit_valueT,'String');
V_value = get(handles.edit_valueV,'String');
Asquare_value = get(handles.edit_valueAsquare,'String');
Fsquare_value = get(handles.edit_valueFsquare,'String');
Asine_value = get(handles.edit_valueAsine,'String');
Fsine_value = get(handles.edit_valueFsine,'String');
Asawtooth_value = get(handles.edit_valueAsawtooth,'String');
Fsawtooth_value = get(handles.edit_valueFsawtooth,'String');
n_value = get(handles.edit_valuen,'String');

if ~isempty(R_value)&& ~isempty(L_value) && ~isempty(V_value) &&
~isempty(T_value) && isnumeric(str2double(R_value))&&
isnumeric(str2double(L_value)) && isnumeric(str2double(T_value))
&& isnumeric(str2double(V_value)) && str2double(V_value)>0 &&
str2double(R_value)>0 && str2double(L_value)>0 &&
str2double(T_value)>0

    Simulation = get(handles.popupmenu1,'Value');

switch Simulation
        case 1
            load_system('S_1');

set_param(['S_1/RL'],'Resistance',R_value,'Inductance',L_value);
            set_param(['S_1'],'StopTime',T_value);
            V = str2double(V_value)/2;
            V = num2str(V);
            set_param(['S_1/subsystem/DC Voltage
Source'],'Amplitude',V);
            set_param(['S_1/subsystem/DC Voltage
Source1'],'Amplitude',V);
            set_param(['S_1/Signal
Generator'],'Amplitude',Asquare_value,
'Frequency',Fsquare_value);


            if ~isempty(Asquare_value)&& ~isempty(Fsquare_value)
&& isnumeric(str2double(Asquare_value))&&
```

```
isnumeric(str2double(Fsquare_value)) &&
str2double(Asquare_value)>0 && str2double(Fsquare_value)>0
                sim ('S_1');
            else
                errordlg('ERROR: Invalid Number of Control
Signals');
            end;

            time = output.time;
            output1 = output.signals.values(:,1);
            output2 = output.signals.values(:,2);
            control1 = control.signals.values(:,1);
            control2 = control.signals.values(:,2);
            control3 = control.signals.values(:,3);
            thdi = thdi.signals.values(:,1);
            thdv = thdv.signals.values(:,1);
            vload = vload.signals.values(:,1);
            iload = iload.signals.values(:,1);
            vin = vin.signals.values(:,1);


            set(handles.text_thdi,'String',abs(thdi(end)));
            set(handles.text_thdv,'String',abs(thdv(end)));
            set(handles.text_vin,'String',abs(vin(end)));
            set(handles.text_vload,'String',abs(vload(end)));
            set(handles.text_iload,'String',abs(iload(end)));

            set(handles.text_thdif,'String','-');
            set(handles.text_thdil,'String','-');
            set(handles.text_thdvf,'String','-');
            set(handles.text_thdvl,'String','-');
            set(handles.text_vin3,'String','-');
            set(handles.text_vloadf,'String','-');
            set(handles.text_vloadl,'String','-');
            set(handles.text_iloadf,'String','-');
            set(handles.text_iloadl,'String','-');

            fft_button = get(handles.popupmenu2,'Value');

            switch fft_button
                case 1
                    n_repet=20;
                    ts=0.1/length(time);
                    senal_completa=repmat(output2',[1,n_repet]);
                    d=0;
                    for t=0:1:length(senal_completa)-1
                        t_senal(d+1)=ts*d;
                        d=d+1;
                    end
                    senal=senal_completa-mean(senal_completa);
                    T=t_senal(end);    %duración temporal de la
señal (lo que tengo) normalmente 0.4
                    Fs=1/ts; %frecuencia muestreo
```

```
                    t=linspace(0,T,T*Fs); % eje temporal de 0
hasta T con T*FS puntos (el número de muestras que tengo)
                    f=linspace(-Fs/2, Fs/2, T*Fs); % eje
frecuencial de -0.5 (-pi en radianes) hasta 0.5 (pi en
radianes)con T*FS puntos (el número de muestras que tengo)
                    Fsenal=fft(senal,length(f))/length(f); %fft
de la señal si dividimos por el numero de muestras que tengo lo
normalizo respecto al número de muestras que tengo (valor real)
                    amplitud_fft=fftshift(abs(Fsenal));    %
Determinación del armónico fundamental

armonico_fundamental=round(f(max(find(amplitud_fft==max(amplitud
_fft))))); % Determinación de las amplitudes reales de los
armónicos
                    amplitud_armonicos=2*amplitud_fft;
                    amplitud_continua=mean(senal_completa);  %
se correspondería con la que debería existir a frecuencia cero
en el gráfico
                    amplitud_continua=mean(senal);
                    N_arm=20; % número de armónicos que se
representan
                    % Representación incluyendo solo frecuencias
positivas con continua
                    % habiendo arreglado la amplitud de los
armónicos, la frecuencia alcanza el
                    % valor de la fundamental por el valor del
armónico
                    % N_arm*armonico_fundamental
                    ind_inf=min(find(f>=0));
                    amplitud_final=[amplitud_continua
amplitud_armonicos(1,ind_inf:end)];
                    f_final=[0 f(1,ind_inf:end)];
                    figure;
                    plot(f_final,amplitud_final,'b')
                    axis([0 N_arm*armonico_fundamental 0
max(amplitud_final)*1.1])
                    title('FFT Analysis: Relative to specified
base')
                    % frecuencia fundamental y su amplitud

amplitud_final_normalizada=100*(amplitud_final./amplitud_final(m
ax(find(amplitud_final==max(amplitud_final)))));

frecuen_fundamental=f_final(max(find(amplitud_final==max(amplitu
d_final))));
                    hold on
                    xgg=[frecuen_fundamental
frecuen_fundamental];
                    ygg=[0 max(amplitud_final)];
                    plot(xgg,ygg,'-r','LineWidth',2)
                    % Representación normalizada respecto al
armonico fundamental
                    figure;
                    plot(f_final,amplitud_final_normalizada,'b')
```

```
                    axis([0 N_arm*armonico_fundamental 0
max(amplitud_final_normalizada)*1.1])
                    title('FFT Analysis: Relative to
fundamental')
                    hold on
                    xgg=[frecuen_fundamental
frecuen_fundamental];
                    ygg=[0 100];
                    plot(xgg,ygg,'-r','LineWidth',2)
                    axis([0 N_arm*armonico_fundamental 0
max(amplitud_final_normalizada)*1.1])


                case 2
                    n_repet=20;
                    ts=0.1/length(time);
                    senal_completa=repmat(output1',[1,n_repet]);
                    d=0;
                    for t=0:1:length(senal_completa)-1
                        t_senal(d+1)=ts*d;
                        d=d+1;
                    end
                    senal=senal_completa-mean(senal_completa);
                    T=t_senal(end);    %duración temporal de la
señal (lo que tengo) normalmente 0.4
                    Fs=1/ts; %frecuencia muestreo
                    t=linspace(0,T,T*Fs); % eje temporal de 0
hasta T con T*FS puntos (el número de muestras que tengo)
                    f=linspace(-Fs/2, Fs/2, T*Fs); % eje
frecuencial de -0.5 (-pi en radianes) hasta 0.5 (pi en
radianes)con T*FS puntos (el número de muestras que tengo)
                    Fsenal=fft(senal,length(f))/length(f); %fft
de la señal si dividimos por el numero de muestras que tengo lo
normalizo respecto al número de muestras que tengo (valor real)
                    amplitud_fft=fftshift(abs(Fsenal));    %
Determinación del armónico fundamental

armonico_fundamental=round(f(max(find(amplitud_fft==max(amplitud
_fft))))); % Determinación de las amplitudes reales de los
armónicos
                    amplitud_armonicos=2*amplitud_fft;
                    amplitud_continua=mean(senal_completa);  %
se correspondería con la que debería existir a frecuencia cero
en el gráfico
                    amplitud_continua=mean(senal);
                    N_arm=20; % número de armónicos que se
representan
                    % Representación incluyendo solo frecuencias
positivas con continua
                    % habiendo arreglado la amplitud de los
armónicos, la frecuencia alcanza el
                    % valor de la fundamental por el valor del
armónico
                    % N_arm*armonico_fundamental
```

```
                    ind_inf=min(find(f>=0));
                    amplitud_final=[amplitud_continua
amplitud_armonicos(1,ind_inf:end)];
                    f_final=[0 f(1,ind_inf:end)];
                    figure;
                    plot(f_final,amplitud_final,'r')
                    axis([0 N_arm*armonico_fundamental 0
max(amplitud_final)*1.1])
                    title('FFT Analysis: Relative to specified
base')
                    % frecuencia fundamental y su amplitud

amplitud_final_normalizada=100*(amplitud_final./amplitud_final(m
ax(find(amplitud_final==max(amplitud_final)))));

frecuen_fundamental=f_final(max(find(amplitud_final==max(amplitu
d_final))));
                    hold on
                    xgg=[frecuen_fundamental
frecuen_fundamental];
                    ygg=[0 max(amplitud_final)];
                    plot(xgg,ygg,'-r','LineWidth',2)
                    % Representación normalizada respecto al
armonico fundamental
                    figure;
                    plot(f_final,amplitud_final_normalizada,'r')
                    axis([0 N_arm*armonico_fundamental 0
max(amplitud_final_normalizada)*1.1])
                    title('FFT Analysis: Relative to
fundamental')
                    hold on
                    xgg=[frecuen_fundamental
frecuen_fundamental];
                    ygg=[0 100];
                    plot(xgg,ygg,'-r','LineWidth',2)
                    axis([0 N_arm*armonico_fundamental 0
max(amplitud_final_normalizada)*1.1])

                case 3
                    errordlg('ERROR: This is a Single-Phase
circuit, you have to change FFT options');

                case 4
                    errordlg('ERROR: This is a Single-Phase
circuit, you have to change FFT options');

                case 5
                    errordlg('ERROR: This is a Single-Phase
circuit, you have to change FFT options');

                case 6
                    errordlg('ERROR: This is a Single-Phase
circuit, you have to change FFT options');
```

```
                otherwise
             disp('Other Simulation');
            end;
```

The same happens with the following 22 cases. It shows a three phases case as an example, and the especial case 6 (6- Full Wave H-Bridge Inverter (Voltage Harmonic Cancellation)):

```
        case 2

        case 3

        case 4

        case 5

        case 6
            load_system('S_6');

set_param(['S_6/RL'],'Resistance',R_value,'Inductance',L_value);
            set_param(['S_6'],'StopTime',T_value);
            V = V_value;
            set_param(['S_6/subsystem/DC Voltage
Source1'],'Amplitude',V);
            set_param(['S_6/Signal
Generator'],'Amplitude',Asine_value, 'Frequency',Fsine_value);

            Alpha = 90/str2double(n_value);
            A = Alpha*2;
            B = 360/A;
            C = 1/str2double(Fsine_value);
            D = C/B;
            Delay_value = num2str(D);
            set_param(['S_6/Transport
Delay1'],'DelayTime',Delay_value);

            if ~isempty(Asine_value)&& ~isempty(Fsine_value) &&
isnumeric(str2double(Asine_value))&&
isnumeric(str2double(Fsine_value))  && str2double(Asine_value)>0
&& str2double(Fsine_value)>0
                if ~isempty(n_value) &&
isnumeric(str2double(n_value)) && str2double(n_value)>1
                    if mod(n_value,2)==1
                       sim ('S_6');
                    else
                       errordlg('ERROR: "n" is a even number.
"n" has to be a ODD number');
                    end;
                else
                    errordlg('ERROR: Invalid Number of
Harmonic');
                end;
            else
```

```
                errordlg('ERROR: Invalid Number of Control
Signals');
            end;

            time = output.time;
            output1 = output.signals.values(:,1);
            output2 = output.signals.values(:,2);
            control1 = control.signals.values(:,1);
            control2 = control.signals.values(:,2);
            control3 = control.signals.values(:,3);
            thdi = thdi.signals.values(:,1);
            thdv = thdv.signals.values(:,1);
            vload = vload.signals.values(:,1);
            iload = iload.signals.values(:,1);
            vin = vin.signals.values(:,1);

            set(handles.text_thdi,'String',abs(thdi(end)));
            set(handles.text_thdv,'String',abs(thdv(end)));
            set(handles.text_vin,'String',abs(vin(end)));
            set(handles.text_vload,'String',abs(vload(end)));
            set(handles.text_iload,'String',abs(iload(end)));

            set(handles.text_thdif,'String','-');
            set(handles.text_thdil,'String','-');
            set(handles.text_thdvf,'String','-');
            set(handles.text_thdvl,'String','-');
            set(handles.text_vin3,'String','-');
            set(handles.text_vloadf,'String','-');
            set(handles.text_vloadl,'String','-');
            set(handles.text_iloadf,'String','-');
            set(handles.text_iloadl,'String','-');

            set(handles.text_Alpha,'String',Alpha);

            fft_button = get(handles.popupmenu2,'Value');

            switch fft_button
                case 1
                    n_repet=20;
                    ts=0.1/length(time);
                    senal_completa=repmat(output2',[1,n_repet]);
                    d=0;
                    for t=0:1:length(senal_completa)-1
                        t_senal(d+1)=ts*d;
                        d=d+1;
                    end
                    senal=senal_completa-mean(senal_completa);
                    T=t_senal(end);    %duración temporal de la
señal (lo que tengo) normalmente 0.4
                    Fs=1/ts; %frecuencia muestreo
                    t=linspace(0,T,T*Fs); % eje temporal de 0
hasta T con T*FS puntos (el número de muestras que tengo)
```

```
                    f=linspace(-Fs/2, Fs/2, T*Fs); % eje
frecuencial de -0.5 (-pi en radianes) hasta 0.5 (pi en
radianes)con T*FS puntos (el número de muestras que tengo)
                    Fsenal=fft(senal,length(f))/length(f); %fft
de la señal si dividimos por el numero de muestras que tengo lo
normalizo respecto al número de muestras que tengo (valor real)
                    amplitud_fft=fftshift(abs(Fsenal));    %
Determinación del armónico fundamental

armonico_fundamental=round(f(max(find(amplitud_fft==max(amplitud
_fft))))); % Determinación de las amplitudes reales de los
armónicos
                    amplitud_armonicos=2*amplitud_fft;
                    amplitud_continua=mean(senal_completa);  %
se correspondería con la que debería existir a frecuencia cero
en el gráfico
                    amplitud_continua=mean(senal);
                    N_arm=20; % número de armónicos que se
representan
                    % Representación incluyendo solo frecuencias
positivas con continua
                    % habiendo arreglado la amplitud de los
armónicos, la frecuencia alcanza el
                    % valor de la fundamental por el valor del
armónico
                    % N_arm*armonico_fundamental
                    ind_inf=min(find(f>=0));
                    amplitud_final=[amplitud_continua
amplitud_armonicos(1,ind_inf:end)];
                    f_final=[0 f(1,ind_inf:end)];
                    figure;
                    plot(f_final,amplitud_final,'b')
                    axis([0 N_arm*armonico_fundamental 0
max(amplitud_final)*1.1])
                    title('FFT Analysis: Relative to specified
base')
                    % frecuencia fundamental y su amplitud

amplitud_final_normalizada=100*(amplitud_final./amplitud_final(m
ax(find(amplitud_final==max(amplitud_final)))));

frecuen_fundamental=f_final(max(find(amplitud_final==max(amplitu
d_final))));
                    hold on
                    xgg=[frecuen_fundamental
frecuen_fundamental];
                    ygg=[0 max(amplitud_final)];
                    plot(xgg,ygg,'-r','LineWidth',2)
                    % Representación normalizada respecto al
armonico fundamental
                    figure;
                    plot(f_final,amplitud_final_normalizada,'b')
                    axis([0 N_arm*armonico_fundamental 0
max(amplitud_final_normalizada)*1.1])
```

```
                    title('FFT Analysis: Relative to
fundamental')
                    hold on
                    xgg=[frecuen_fundamental
frecuen_fundamental];
                    ygg=[0 100];
                    plot(xgg,ygg,'-r','LineWidth',2)
                    axis([0 N_arm*armonico_fundamental 0
max(amplitud_final_normalizada)*1.1])

                case 2
                    n_repet=20;
                    ts=0.1/length(time);
                    senal_completa=repmat(output1',[1,n_repet]);
                    d=0;
                    for t=0:1:length(senal_completa)-1
                        t_senal(d+1)=ts*d;
                        d=d+1;
                    end
                    senal=senal_completa-mean(senal_completa);
                    T=t_senal(end);    %duración temporal de la
señal (lo que tengo) normalmente 0.4
                    Fs=1/ts; %frecuencia muestreo
                    t=linspace(0,T,T*Fs); % eje temporal de 0
hasta T con T*FS puntos (el número de muestras que tengo)
                    f=linspace(-Fs/2, Fs/2, T*Fs); % eje
frecuencial de -0.5 (-pi en radianes) hasta 0.5 (pi en
radianes)con T*FS puntos (el número de muestras que tengo)
                    Fsenal=fft(senal,length(f))/length(f); %fft
de la señal si dividimos por el numero de muestras que tengo lo
normalizo respecto al número de muestras que tengo (valor real)
                    amplitud_fft=fftshift(abs(Fsenal));    %
Determinación del armónico fundamental

armonico_fundamental=round(f(max(find(amplitud_fft==max(amplitud
_fft))))); % Determinación de las amplitudes reales de los
armónicos
                    amplitud_armonicos=2*amplitud_fft;
                    amplitud_continua=mean(senal_completa);  %
se correspondería con la que debería existir a frecuencia cero
en el gráfico
                    amplitud_continua=mean(senal);
                    N_arm=20; % número de armónicos que se
representan
                    % Representación incluyendo solo frecuencias
positivas con continua
                    % habiendo arreglado la amplitud de los
armónicos, la frecuencia alcanza el
                    % valor de la fundamental por el valor del
armónico
                    % N_arm*armonico_fundamental
                    ind_inf=min(find(f>=0));
                    amplitud_final=[amplitud_continua
amplitud_armonicos(1,ind_inf:end)];
```

```
                    f_final=[0 f(1,ind_inf:end)];
                    figure;
                    plot(f_final,amplitud_final,'r')
                    axis([0 N_arm*armonico_fundamental 0
max(amplitud_final)*1.1])
                    title('FFT Analysis: Relative to specified
base')
                    % frecuencia fundamental y su amplitud

amplitud_final_normalizada=100*(amplitud_final./amplitud_final(m
ax(find(amplitud_final==max(amplitud_final)))));

frecuen_fundamental=f_final(max(find(amplitud_final==max(amplitu
d_final))));
                    hold on
                    xgg=[frecuen_fundamental
frecuen_fundamental];
                    ygg=[0 max(amplitud_final)];
                    plot(xgg,ygg,'-r','LineWidth',2)
                    % Representación normalizada respecto al
armonico fundamental
                    figure;
                    plot(f_final,amplitud_final_normalizada,'r')
                    axis([0 N_arm*armonico_fundamental 0
max(amplitud_final_normalizada)*1.1])
                    title('FFT Analysis: Relative to
fundamental')
                    hold on
                    xgg=[frecuen_fundamental
frecuen_fundamental];
                    ygg=[0 100];
                    plot(xgg,ygg,'-r','LineWidth',2)
                    axis([0 N_arm*armonico_fundamental 0
max(amplitud_final_normalizada)*1.1])

                case 3
                    errordlg('ERROR: This is a Single-Phase
circuit, you have to change FFT options');

                case 4
                    errordlg('ERROR: This is a Single-Phase
circuit, you have to change FFT options');

                case 5
                    errordlg('ERROR: This is a Single-Phase
circuit, you have to change FFT options');

                case 6
                    errordlg('ERROR: This is a Single-Phase
circuit, you have to change FFT options');

                otherwise
             disp('Other Simulation');
            end;
```

```
        case 7

        case 8

        case 9
            load_system('S_9');

set_param(['S_9/RL1'],'Resistance',R_value,'Inductance',L_value)
;

set_param(['S_9/RL2'],'Resistance',R_value,'Inductance',L_value)
;

set_param(['S_9/RL3'],'Resistance',R_value,'Inductance',L_value)
;
            set_param(['S_9'],'StopTime',T_value);
            V = V_value;
            set_param(['S_9/subsystem/DC Voltage
Source1'],'Amplitude',V);
            set_param(['S_9/Signal
Generator'],'Amplitude',Asquare_value,
'Frequency',Fsquare_value);

            if ~isempty(Asquare_value)&& ~isempty(Fsquare_value)
&& isnumeric(str2double(Asquare_value))&&
isnumeric(str2double(Fsquare_value)) &&
str2double(Asquare_value)>0 && str2double(Fsquare_value)>0
                sim ('S_9');
            else
                errordlg('ERROR: Invalid Number of Control
Signals');
            end;

            time = output.time;
            output1 = output.signals.values(:,1);
            output2 = output.signals.values(:,2);
            output3 = output.signals.values(:,3);
            control1 = control.signals.values(:,1);
            control2 = control.signals.values(:,2);
            control3 = control.signals.values(:,3);
            thdif = thdif.signals.values(:,1);
            thdil = thdil.signals.values(:,1);
            thdvf = thdvf.signals.values(:,1);
            thdvl = thdvl.signals.values(:,1);
            vloadf = vloadf.signals.values(:,1);
            vloadl = vloadl.signals.values(:,1);
            iloadf = iloadf.signals.values(:,1);
            iloadl = iloadl.signals.values(:,1);
            vin = vin.signals.values(:,1);

            set(handles.text_thdif,'String',abs(thdif(end)));
            set(handles.text_thdil,'String',abs(thdil(end)));
            set(handles.text_thdvf,'String',abs(thdvf(end)));
```

```
            set(handles.text_thdvl,'String',abs(thdvl(end)));
            set(handles.text_vin3,'String',abs(vin(end)));
            set(handles.text_vloadf,'String',abs(vloadf(end)));
            set(handles.text_vloadl,'String',abs(vloadl(end)));
            set(handles.text_iloadf,'String',abs(iloadf(end)));
            set(handles.text_iloadl,'String',abs(iloadl(end)));

            set(handles.text_thdi,'String','-');
            set(handles.text_thdv,'String','-');
            set(handles.text_vin,'String','-');
            set(handles.text_vload,'String','-');
            set(handles.text_iload,'String','-');

            fft_button = get(handles.popupmenu2,'Value');

            switch fft_button
                case 1
                    errordlg('ERROR: This is a Three-Phase
circuit, you have to change FFT options');

                case 2
                    errordlg('ERROR: This is a Three-Phase
circuit, you have to change FFT options');

                case 3
                    n_repet=20;
                    ts=0.1/length(time);
                    senal_completa=repmat(output3',[1,n_repet]);
                    d=0;
                    for t=0:1:length(senal_completa)-1
                        t_senal(d+1)=ts*d;
                        d=d+1;
                    end
                    senal=senal_completa-mean(senal_completa);
                    T=t_senal(end);    %duración temporal de la
señal (lo que tengo) normalmente 0.4
                    Fs=1/ts; %frecuencia muestreo
                    t=linspace(0,T,T*Fs); % eje temporal de 0
hasta T con T*FS puntos (el número de muestras que tengo)
                    f=linspace(-Fs/2, Fs/2, T*Fs); % eje
frecuencial de -0.5 (-pi en radianes) hasta 0.5 (pi en
radianes)con T*FS puntos (el número de muestras que tengo)
                    Fsenal=fft(senal,length(f))/length(f); %fft
de la señal si dividimos por el numero de muestras que tengo lo
normalizo respecto al número de muestras que tengo (valor real)
                    amplitud_fft=fftshift(abs(Fsenal));    %
Determinación del armónico fundamental

armonico_fundamental=round(f(max(find(amplitud_fft==max(amplitud
_fft))))); % Determinación de las amplitudes reales de los
armónicos
                    amplitud_armonicos=2*amplitud_fft;
```

```
                    amplitud_continua=mean(senal_completa);  %
se correspondería con la que debería existir a frecuencia cero
en el gráfico
                    amplitud_continua=mean(senal);
                    N_arm=20; % número de armónicos que se
representan
                    % Representación incluyendo solo frecuencias
positivas con continua
                    % habiendo arreglado la amplitud de los
armónicos, la frecuencia alcanza el
                    % valor de la fundamental por el valor del
armónico
                    % N_arm*armonico_fundamental
                    ind_inf=min(find(f>=0));
                    amplitud_final=[amplitud_continua
amplitud_armonicos(1,ind_inf:end)];
                    f_final=[0 f(1,ind_inf:end)];
                    figure;
                    plot(f_final,amplitud_final,'y')
                    axis([0 N_arm*armonico_fundamental 0
max(amplitud_final)*1.1])
                    title('FFT Analysis: Relative to specified
base')
                    % frecuencia fundamental y su amplitud

amplitud_final_normalizada=100*(amplitud_final./amplitud_final(m
ax(find(amplitud_final==max(amplitud_final)))));

frecuen_fundamental=f_final(max(find(amplitud_final==max(amplitu
d_final))));
                    hold on
                    xgg=[frecuen_fundamental
frecuen_fundamental];
                    ygg=[0 max(amplitud_final)];
                    plot(xgg,ygg,'-r','LineWidth',2)
                    % Representación normalizada respecto al
armonico fundamental
                    figure;
                    plot(f_final,amplitud_final_normalizada,'y')
                    axis([0 N_arm*armonico_fundamental 0
max(amplitud_final_normalizada)*1.1])
                    title('FFT Analysis: Relative to
fundamental')
                    hold on
                    xgg=[frecuen_fundamental
frecuen_fundamental];
                    ygg=[0 100];
                    plot(xgg,ygg,'-r','LineWidth',2)
                    axis([0 N_arm*armonico_fundamental 0
max(amplitud_final_normalizada)*1.1])

                case 4
                    n_repet=20;
                    ts=0.1/length(time);
```

```
                    senal_completa=repmat(output2',[1,n_repet]);
                    d=0;
                    for t=0:1:length(senal_completa)-1
                        t_senal(d+1)=ts*d;
                        d=d+1;
                    end
                    senal=senal_completa-mean(senal_completa);
                    T=t_senal(end);    %duración temporal de la
señal (lo que tengo) normalmente 0.4
                    Fs=1/ts; %frecuencia muestreo
                    t=linspace(0,T,T*Fs); % eje temporal de 0
hasta T con T*FS puntos (el número de muestras que tengo)
                    f=linspace(-Fs/2, Fs/2, T*Fs); % eje
frecuencial de -0.5 (-pi en radianes) hasta 0.5 (pi en
radianes)con T*FS puntos (el número de muestras que tengo)
                    Fsenal=fft(senal,length(f))/length(f); %fft
de la señal si dividimos por el numero de muestras que tengo lo
normalizo respecto al número de muestras que tengo (valor real)
                    amplitud_fft=fftshift(abs(Fsenal));    %
Determinación del armónico fundamental

armonico_fundamental=round(f(max(find(amplitud_fft==max(amplitud
_fft))))); % Determinación de las amplitudes reales de los
armónicos
                    amplitud_armonicos=2*amplitud_fft;
                    amplitud_continua=mean(senal_completa);  %
se correspondería con la que debería existir a frecuencia cero
en el gráfico
                    amplitud_continua=mean(senal);
                    N_arm=20; % número de armónicos que se
representan
                    % Representación incluyendo solo frecuencias
positivas con continua
                    % habiendo arreglado la amplitud de los
armónicos, la frecuencia alcanza el
                    % valor de la fundamental por el valor del
armónico
                    % N_arm*armonico_fundamental
                    ind_inf=min(find(f>=0));
                    amplitud_final=[amplitud_continua
amplitud_armonicos(1,ind_inf:end)];
                    f_final=[0 f(1,ind_inf:end)];
                    figure;
                    plot(f_final,amplitud_final,'b')
                    axis([0 N_arm*armonico_fundamental 0
max(amplitud_final)*1.1])
                    title('FFT Analysis: Relative to specified
base')
                    % frecuencia fundamental y su amplitud

amplitud_final_normalizada=100*(amplitud_final./amplitud_final(m
ax(find(amplitud_final==max(amplitud_final)))));
```

```
frecuen_fundamental=f_final(max(find(amplitud_final==max(amplitu
d_final))));
                    hold on
                    xgg=[frecuen_fundamental
frecuen_fundamental];
                    ygg=[0 max(amplitud_final)];
                    plot(xgg,ygg,'-r','LineWidth',2)
                    % Representación normalizada respecto al
armonico fundamental
                    figure;
                    plot(f_final,amplitud_final_normalizada,'b')
                    axis([0 N_arm*armonico_fundamental 0
max(amplitud_final_normalizada)*1.1])
                    title('FFT Analysis: Relative to
fundamental')
                    hold on
                    xgg=[frecuen_fundamental
frecuen_fundamental];
                    ygg=[0 100];
                    plot(xgg,ygg,'-r','LineWidth',2)
                    axis([0 N_arm*armonico_fundamental 0
max(amplitud_final_normalizada)*1.1])

                case 5
                    n_repet=20;
                    ts=0.1/length(time);
                    senal_completa=repmat(output1',[1,n_repet]);
                    d=0;
                    for t=0:1:length(senal_completa)-1
                        t_senal(d+1)=ts*d;
                        d=d+1;
                    end
                    senal=senal_completa-mean(senal_completa);
                    T=t_senal(end);    %duración temporal de la
señal (lo que tengo) normalmente 0.4
                    Fs=1/ts; %frecuencia muestreo
                    t=linspace(0,T,T*Fs); % eje temporal de 0
hasta T con T*FS puntos (el número de muestras que tengo)
                    f=linspace(-Fs/2, Fs/2, T*Fs); % eje
frecuencial de -0.5 (-pi en radianes) hasta 0.5 (pi en
radianes)con T*FS puntos (el número de muestras que tengo)
                    Fsenal=fft(senal,length(f))/length(f); %fft
de la señal si dividimos por el numero de muestras que tengo lo
normalizo respecto al número de muestras que tengo (valor real)
                    amplitud_fft=fftshift(abs(Fsenal));    %
Determinación del armónico fundamental

armonico_fundamental=round(f(max(find(amplitud_fft==max(amplitud
_fft))))); % Determinación de las amplitudes reales de los
armónicos
                    amplitud_armonicos=2*amplitud_fft;
```

```
                    amplitud_continua=mean(senal_completa);  %
se correspondería con la que debería existir a frecuencia cero
en el gráfico
                    amplitud_continua=mean(senal);
                    N_arm=20; % número de armónicos que se
representan
                    % Representación incluyendo solo frecuencias
positivas con continua
                    % habiendo arreglado la amplitud de los
armónicos, la frecuencia alcanza el
                    % valor de la fundamental por el valor del
armónico
                    % N_arm*armonico_fundamental
                    ind_inf=min(find(f>=0));
                    amplitud_final=[amplitud_continua
amplitud_armonicos(1,ind_inf:end)];
                    f_final=[0 f(1,ind_inf:end)];
                    figure;
                    plot(f_final,amplitud_final,'r')
                    axis([0 N_arm*armonico_fundamental 0
max(amplitud_final)*1.1])
                    title('FFT Analysis: Relative to specified
base')
                    % frecuencia fundamental y su amplitud

amplitud_final_normalizada=100*(amplitud_final./amplitud_final(m
ax(find(amplitud_final==max(amplitud_final)))));

frecuen_fundamental=f_final(max(find(amplitud_final==max(amplitu
d_final))));
                    hold on
                    xgg=[frecuen_fundamental
frecuen_fundamental];
                    ygg=[0 max(amplitud_final)];
                    plot(xgg,ygg,'-r','LineWidth',2)
                    % Representación normalizada respecto al
armonico fundamental
                    figure;
                    plot(f_final,amplitud_final_normalizada,'r')
                    axis([0 N_arm*armonico_fundamental 0
max(amplitud_final_normalizada)*1.1])
                    title('FFT Analysis: Relative to
fundamental')
                    hold on
                    xgg=[frecuen_fundamental
frecuen_fundamental];
                    ygg=[0 100];
                    plot(xgg,ygg,'-r','LineWidth',2)
                    axis([0 N_arm*armonico_fundamental 0
max(amplitud_final_normalizada)*1.1])

                case 6
                    n_repet=20;
                    ts=0.1/length(time);
```

```
                    senal_completa=repmat(output1',[1,n_repet]);
                    d=0;
                    for t=0:1:length(senal_completa)-1
                        t_senal(d+1)=ts*d;
                        d=d+1;
                    end
                    senal=senal_completa-mean(senal_completa);
                    T=t_senal(end);    %duración temporal de la
señal (lo que tengo) normalmente 0.4
                    Fs=1/ts; %frecuencia muestreo
                    t=linspace(0,T,T*Fs); % eje temporal de 0
hasta T con T*FS puntos (el número de muestras que tengo)
                    f=linspace(-Fs/2, Fs/2, T*Fs); % eje
frecuencial de -0.5 (-pi en radianes) hasta 0.5 (pi en
radianes)con T*FS puntos (el número de muestras que tengo)
                    Fsenal=fft(senal,length(f))/length(f); %fft
de la señal si dividimos por el numero de muestras que tengo lo
normalizo respecto al número de muestras que tengo (valor real)
                    amplitud_fft=fftshift(abs(Fsenal));    %
Determinación del armónico fundamental

armonico_fundamental=round(f(max(find(amplitud_fft==max(amplitud
_fft))))); % Determinación de las amplitudes reales de los
armónicos
                    amplitud_armonicos=2*amplitud_fft;
                    amplitud_continua=mean(senal_completa);  %
se correspondería con la que debería existir a frecuencia cero
en el gráfico
                    amplitud_continua=mean(senal);
                    N_arm=20; % número de armónicos que se
representan
                    % Representación incluyendo solo frecuencias
positivas con continua
                    % habiendo arreglado la amplitud de los
armónicos, la frecuencia alcanza el
                    % valor de la fundamental por el valor del
armónico
                    % N_arm*armonico_fundamental
                    ind_inf=min(find(f>=0));
                    amplitud_final=[amplitud_continua
amplitud_armonicos(1,ind_inf:end)];
                    f_final=[0 f(1,ind_inf:end)];
                    figure;
                    plot(f_final,amplitud_final,'r')
                    axis([0 N_arm*armonico_fundamental 0
max(amplitud_final)*1.1])
                    title('FFT Analysis: Relative to specified
base')
                    % frecuencia fundamental y su amplitud

amplitud_final_normalizada=100*(amplitud_final./amplitud_final(m
ax(find(amplitud_final==max(amplitud_final)))));
```

```
frecuen_fundamental=f_final(max(find(amplitud_final==max(amplitu
d_final))));
                    hold on
                    xgg=[frecuen_fundamental
frecuen_fundamental];
                    ygg=[0 max(amplitud_final)];
                    plot(xgg,ygg,'-r','LineWidth',2)
                    % Representación normalizada respecto al
armonico fundamental
                    figure;
                    plot(f_final,amplitud_final_normalizada,'r')
                    axis([0 N_arm*armonico_fundamental 0
max(amplitud_final_normalizada)*1.1])
                    title('FFT Analysis: Relative to
fundamental')
                    hold on
                    xgg=[frecuen_fundamental
frecuen_fundamental];
                    ygg=[0 100];
                    plot(xgg,ygg,'-r','LineWidth',2)
                    axis([0 N_arm*armonico_fundamental 0
max(amplitud_final_normalizada)*1.1])

                otherwise
             disp('Other Simulation');
            end;

        case 10

        case 11

        case 12

        case 13

        case 14

        case 15

        case 16

        case 17

        case 18

        case 19

        case 20

        case 21

        case 22
```

```
        case 23
else
    errordlg('ERROR: Invalid Number of Parameters to modify');

end


% --- Executes on selection change in popupmenu2.
function popupmenu2_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns
popupmenu2 contents as cell array
%        contents{get(hObject,'Value')} returns selected item
from popupmenu2


% --- Executes during object creation, after setting all
properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: popupmenu controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

1.2.6 ZOOM INVERTER CONTROL

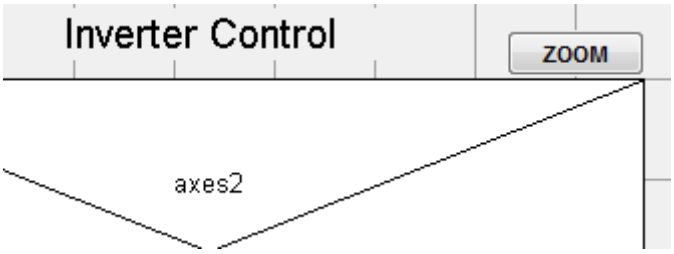


Figure 15 Inverter Control ZOOM

It repeats exactly the same process as when the "Simulation" button is pressed, but the graphics instead be drawn in one axis, it's written "figure", and this makes a separate window where it is generated the same graph.

```
% --- Executes on button press in ZoomIC.
function ZoomIC_Callback(hObject, eventdata, handles)
% hObject    handle to ZoomIC (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
R_value = get(handles.edit_valueR,'String');
L_value = get(handles.edit_valueL,'String');
T_value = get(handles.edit_valueT,'String');
V_value = get(handles.edit_valueV,'String');
Asquare_value = get(handles.edit_valueAsquare,'String');
Fsquare_value = get(handles.edit_valueFsquare,'String');
Asine_value = get(handles.edit_valueAsine,'String');
Fsine_value = get(handles.edit_valueFsine,'String');
Asawtooth_value = get(handles.edit_valueAsawtooth,'String');
Fsawtooth_value = get(handles.edit_valueFsawtooth,'String');
n_value = get(handles.edit_valuen,'String');

if ~isempty(R_value)&& ~isempty(L_value) && ~isempty(V_value) &&
~isempty(T_value) && isnumeric(str2double(R_value))&&
isnumeric(str2double(L_value)) && isnumeric(str2double(T_value))
&& isnumeric(str2double(V_value)) && str2double(V_value)>0 &&
str2double(R_value)>0 && str2double(L_value)>0 &&
str2double(T_value)>0

    Simulation = get(handles.popupmenu1,'Value');

switch Simulation
        case 1

            load_system('S_1');

set_param(['S_1/RL'],'Resistance',R_value,'Inductance',L_value);
            set_param(['S_1'],'StopTime',T_value);
            V = str2double(V_value)/2;
            V = num2str(V);
            set_param(['S_1/subsystem/DC Voltage
Source'],'Amplitude',V);
            set_param(['S_1/subsystem/DC Voltage
Source1'],'Amplitude',V);
            set_param(['S_1/Signal
Generator'],'Amplitude',Asquare_value,
'Frequency',Fsquare_value);


            if ~isempty(Asquare_value)&& ~isempty(Fsquare_value)
&& isnumeric(str2double(Asquare_value))&&
```

```
isnumeric(str2double(Fsquare_value)) &&
str2double(Asquare_value)>0 && str2double(Fsquare_value)>0
                sim ('S_1');
            else
                errordlg('ERROR: Invalid Number of Control
Signals');
            end;

            time = output.time;
            output1 = output.signals.values(:,1);
            output2 = output.signals.values(:,2);
            control1 = control.signals.values(:,1);
            control2 = control.signals.values(:,2);
            control3 = control.signals.values(:,3);
            thdi = thdi.signals.values(:,1);
            thdv = thdv.signals.values(:,1);
            vload = vload.signals.values(:,1);
            iload = iload.signals.values(:,1);
            vin = vin.signals.values(:,1);

            figure;

plot(time,control1,'r',time,control2,'b',time,control3,'y');
            title('Inverter Control - Sim 1');

            set(handles.text_thdi,'String',abs(thdi(end)));
            set(handles.text_thdv,'String',abs(thdv(end)));
            set(handles.text_vin,'String',abs(vin(end)));
            set(handles.text_vload,'String',abs(vload(end)));
            set(handles.text_iload,'String',abs(iload(end)));

            set(handles.text_thdif,'String','-');
            set(handles.text_thdil,'String','-');
            set(handles.text_thdvf,'String','-');
            set(handles.text_thdvl,'String','-');
            set(handles.text_vin3,'String','-');
            set(handles.text_vloadf,'String','-');
            set(handles.text_vloadl,'String','-');
            set(handles.text_iloadf,'String','-');
            set(handles.text_iloadl,'String','-');
```

The same happens with the following 22 cases. It shows a three phases case as an example, and the especial case 6 (6- Full Wave H-Bridge Inverter (Voltage Harmonic Cancellation)):

```
        case 2

        case 3

        case 4

        case 5
```

```
        case 6

            load_system('S_6');

set_param(['S_6/RL'],'Resistance',R_value,'Inductance',L_value);
            set_param(['S_6'],'StopTime',T_value);
            V = V_value;
            set_param(['S_6/subsystem/DC Voltage
Source1'],'Amplitude',V);
            set_param(['S_6/Signal
Generator'],'Amplitude',Asine_value, 'Frequency',Fsine_value);

            Alpha = 90/str2double(n_value);
            A = Alpha*2;
            B = 360/A;
            C = 1/str2double(Fsine_value);
            D = C/B;
            Delay_value = num2str(D);
            set_param(['S_6/Transport
Delay1'],'DelayTime',Delay_value);

            if ~isempty(Asine_value)&& ~isempty(Fsine_value) &&
isnumeric(str2double(Asine_value))&&
isnumeric(str2double(Fsine_value))  && str2double(Asine_value)>0
&& str2double(Fsine_value)>0
                if ~isempty(n_value) &&
isnumeric(str2double(n_value)) && str2double(n_value)>1
                    if mod(n_value,2)==1
                       sim ('S_6');
                    else
                       errordlg('ERROR: "n" is a even number.
"n" has to be a ODD number');
                    end;
                else
                    errordlg('ERROR: Invalid Number of
Harmonic');
                end;
            else
                errordlg('ERROR: Invalid Number of Control
Signals');
            end;

            time = output.time;
            output1 = output.signals.values(:,1);
            output2 = output.signals.values(:,2);
            control1 = control.signals.values(:,1);
            control2 = control.signals.values(:,2);
            control3 = control.signals.values(:,3);
            thdi = thdi.signals.values(:,1);
            thdv = thdv.signals.values(:,1);
            vload = vload.signals.values(:,1);
            iload = iload.signals.values(:,1);
            vin = vin.signals.values(:,1);
```

```
            figure;

plot(time,control1,'r',time,control2,'b',time,control3,'y');
            title('Inverter Control - Sim 6');

            set(handles.text_thdi,'String',abs(thdi(end)));
            set(handles.text_thdv,'String',abs(thdv(end)));
            set(handles.text_vin,'String',abs(vin(end)));
            set(handles.text_vload,'String',abs(vload(end)));
            set(handles.text_iload,'String',abs(iload(end)));

            set(handles.text_thdif,'String','-');
            set(handles.text_thdil,'String','-');
            set(handles.text_thdvf,'String','-');
            set(handles.text_thdvl,'String','-');
            set(handles.text_vin3,'String','-');
            set(handles.text_vloadf,'String','-');
            set(handles.text_vloadl,'String','-');
            set(handles.text_iloadf,'String','-');
            set(handles.text_iloadl,'String','-');

            set(handles.text_Alpha,'String',Alpha);

        case 7

        case 8

        case 9

            load_system('S_9');

set_param(['S_9/RL1'],'Resistance',R_value,'Inductance',L_value)
;

set_param(['S_9/RL2'],'Resistance',R_value,'Inductance',L_value)
;

set_param(['S_9/RL3'],'Resistance',R_value,'Inductance',L_value)
;
            set_param(['S_9'],'StopTime',T_value);
            V = V_value;
            set_param(['S_9/subsystem/DC Voltage
Source1'],'Amplitude',V);
            set_param(['S_9/Signal
Generator'],'Amplitude',Asquare_value,
'Frequency',Fsquare_value);

            if ~isempty(Asquare_value)&& ~isempty(Fsquare_value)
&& isnumeric(str2double(Asquare_value))&&
isnumeric(str2double(Fsquare_value)) &&
str2double(Asquare_value)>0 && str2double(Fsquare_value)>0
                sim ('S_9');
            else
```

```
                errordlg('ERROR: Invalid Number of Control
Signals');
            end;

            time = output.time;
            output1 = output.signals.values(:,1);
            output2 = output.signals.values(:,2);
            output3 = output.signals.values(:,3);
            control1 = control.signals.values(:,1);
            control2 = control.signals.values(:,2);
            control3 = control.signals.values(:,3);
            thdif = thdif.signals.values(:,1);
            thdil = thdil.signals.values(:,1);
            thdvf = thdvf.signals.values(:,1);
            thdvl = thdvl.signals.values(:,1);
            vloadf = vloadf.signals.values(:,1);
            vloadl = vloadl.signals.values(:,1);
            iloadf = iloadf.signals.values(:,1);
            iloadl = iloadl.signals.values(:,1);
            vin = vin.signals.values(:,1);

            figure;

plot(time,control1,'r',time,control2,'b',time,control3,'y');
            title('Inverter Control - Sim 9');

            set(handles.text_thdif,'String',abs(thdif(end)));
            set(handles.text_thdil,'String',abs(thdil(end)));
            set(handles.text_thdvf,'String',abs(thdvf(end)));
            set(handles.text_thdvl,'String',abs(thdvl(end)));
            set(handles.text_vin3,'String',abs(vin(end)));
            set(handles.text_vloadf,'String',abs(vloadf(end)));
            set(handles.text_vloadl,'String',abs(vloadl(end)));
            set(handles.text_iloadf,'String',abs(iloadf(end)));
            set(handles.text_iloadl,'String',abs(iloadl(end)));

            set(handles.text_thdi,'String','-');
            set(handles.text_thdv,'String','-');
            set(handles.text_vin,'String','-');
            set(handles.text_vload,'String','-');
            set(handles.text_iload,'String','-');

        case 10

        case 11

        case 12

        case 13

        case 14

        case 15
```

```
        case 16

        case 17

        case 18

        case 19

        case 20

        case 21

        case 22

       case 23

        otherwise
             disp('Other Simulation');
    end
else
    errordlg('ERROR: Invalid Number of Parameters to modify');

end
```

1.2.7 ZOOM OUTPUT

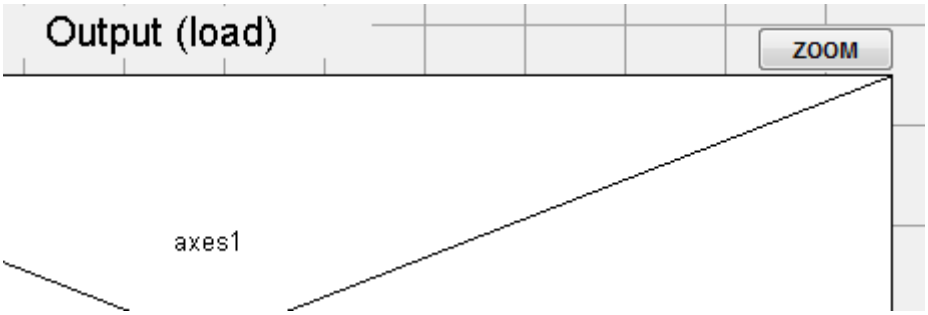


Figure 16 Output ZOOM

It repeats exactly the same process as when the "Simulation" button is pressed, but the graphics instead be drawn in one axis, it's written "figure", and this makes a separate window where it is generated the same graph.

```
% --- Executes on button press in ZoomO.
function ZoomO_Callback(hObject, eventdata, handles)
% hObject    handle to ZoomO (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
R_value = get(handles.edit_valueR,'String');
L_value = get(handles.edit_valueL,'String');
T_value = get(handles.edit_valueT,'String');
```

```
V_value = get(handles.edit_valueV,'String');
Asquare_value = get(handles.edit_valueAsquare,'String');
Fsquare_value = get(handles.edit_valueFsquare,'String');
Asine_value = get(handles.edit_valueAsine,'String');
Fsine_value = get(handles.edit_valueFsine,'String');
Asawtooth_value = get(handles.edit_valueAsawtooth,'String');
Fsawtooth_value = get(handles.edit_valueFsawtooth,'String');
n_value = get(handles.edit_valuen,'String');

if ~isempty(R_value)&& ~isempty(L_value) && ~isempty(V_value) &&
~isempty(T_value) && isnumeric(str2double(R_value))&&
isnumeric(str2double(L_value)) && isnumeric(str2double(T_value))
&& isnumeric(str2double(V_value)) && str2double(V_value)>0 &&
str2double(R_value)>0 && str2double(L_value)>0 &&
str2double(T_value)>0

    Simulation = get(handles.popupmenu1,'Value');

switch Simulation
        case 1

            load_system('S_1');

set_param(['S_1/RL'],'Resistance',R_value,'Inductance',L_value);
            set_param(['S_1'],'StopTime',T_value);
            V = str2double(V_value)/2;
            V = num2str(V);
            set_param(['S_1/subsystem/DC Voltage
Source'],'Amplitude',V);
            set_param(['S_1/subsystem/DC Voltage
Source1'],'Amplitude',V);
            set_param(['S_1/Signal
Generator'],'Amplitude',Asquare_value,
'Frequency',Fsquare_value);


            if ~isempty(Asquare_value)&& ~isempty(Fsquare_value)
&& isnumeric(str2double(Asquare_value))&&
isnumeric(str2double(Fsquare_value)) &&
str2double(Asquare_value)>0 && str2double(Fsquare_value)>0
                sim ('S_1');
            else
                errordlg('ERROR: Invalid Number of Control
Signals');
            end;

            time = output.time;
            output1 = output.signals.values(:,1);
            output2 = output.signals.values(:,2);
            control1 = control.signals.values(:,1);
            control2 = control.signals.values(:,2);
            control3 = control.signals.values(:,3);
            thdi = thdi.signals.values(:,1);
            thdv = thdv.signals.values(:,1);
```

```
            vload = vload.signals.values(:,1);
            iload = iload.signals.values(:,1);
            vin = vin.signals.values(:,1);

            figure;
            plot(time,output1,'r',time,output2,'b');
            title('Output (Load) - Sim 1');

            set(handles.text_thdi,'String',abs(thdi(end)));
            set(handles.text_thdv,'String',abs(thdv(end)));
            set(handles.text_vin,'String',abs(vin(end)));
            set(handles.text_vload,'String',abs(vload(end)));
            set(handles.text_iload,'String',abs(iload(end)));

            set(handles.text_thdif,'String','-');
            set(handles.text_thdil,'String','-');
            set(handles.text_thdvf,'String','-');
            set(handles.text_thdvl,'String','-');
            set(handles.text_vin3,'String','-');
            set(handles.text_vloadf,'String','-');
            set(handles.text_vloadl,'String','-');
            set(handles.text_iloadf,'String','-');
            set(handles.text_iloadl,'String','-');
```

The same happens with the following 22 cases. It shows a three phases case as an example, and the especial case 6 (6- Full Wave H-Bridge Inverter (Voltage Harmonic Cancellation)):

```
        case 2

        case 3

        case 4

        case 5

        case 6

            load_system('S_6');

set_param(['S_6/RL'],'Resistance',R_value,'Inductance',L_value);
            set_param(['S_6'],'StopTime',T_value);
            V = V_value;
            set_param(['S_6/subsystem/DC Voltage
Source1'],'Amplitude',V);
            set_param(['S_6/Signal
Generator'],'Amplitude',Asine_value, 'Frequency',Fsine_value);

            Alpha = 90/str2double(n_value);
            A = Alpha*2;
            B = 360/A;
            C = 1/str2double(Fsine_value);
            D = C/B;
```

```
            Delay_value = num2str(D);
            set_param(['S_6/Transport
Delay1'],'DelayTime',Delay_value);

            if ~isempty(Asine_value)&& ~isempty(Fsine_value) &&
isnumeric(str2double(Asine_value))&&
isnumeric(str2double(Fsine_value))  && str2double(Asine_value)>0
&& str2double(Fsine_value)>0
                if ~isempty(n_value) &&
isnumeric(str2double(n_value)) && str2double(n_value)>1
                    if mod(n_value,2)==1
                       sim ('S_6');
                    else
                       errordlg('ERROR: "n" is a even number.
"n" has to be a ODD number');
                    end;
                else
                    errordlg('ERROR: Invalid Number of
Harmonic');
                end;
            else
                errordlg('ERROR: Invalid Number of Control
Signals');
            end;

            time = output.time;
            output1 = output.signals.values(:,1);
            output2 = output.signals.values(:,2);
            control1 = control.signals.values(:,1);
            control2 = control.signals.values(:,2);
            control3 = control.signals.values(:,3);
            thdi = thdi.signals.values(:,1);
            thdv = thdv.signals.values(:,1);
            vload = vload.signals.values(:,1);
            iload = iload.signals.values(:,1);
            vin = vin.signals.values(:,1);

            figure;
            plot(time,output1,'r',time,output2,'b');
            title('Output (Load) - Sim 6');

            set(handles.text_thdi,'String',abs(thdi(end)));
            set(handles.text_thdv,'String',abs(thdv(end)));
            set(handles.text_vin,'String',abs(vin(end)));
            set(handles.text_vload,'String',abs(vload(end)));
            set(handles.text_iload,'String',abs(iload(end)));

            set(handles.text_thdif,'String','-');
            set(handles.text_thdil,'String','-');
            set(handles.text_thdvf,'String','-');
            set(handles.text_thdvl,'String','-');
            set(handles.text_vin3,'String','-');
            set(handles.text_vloadf,'String','-');
            set(handles.text_vloadl,'String','-');
```

```
            set(handles.text_iloadf,'String','-');
            set(handles.text_iloadl,'String','-');

            set(handles.text_Alpha,'String',Alpha);

        case 7

        case 8

        case 9

            load_system('S_9');

set_param(['S_9/RL1'],'Resistance',R_value,'Inductance',L_value)
;

set_param(['S_9/RL2'],'Resistance',R_value,'Inductance',L_value)
;

set_param(['S_9/RL3'],'Resistance',R_value,'Inductance',L_value)
;
            set_param(['S_9'],'StopTime',T_value);
            V = V_value;
            set_param(['S_9/subsystem/DC Voltage
Source1'],'Amplitude',V);
            set_param(['S_9/Signal
Generator'],'Amplitude',Asquare_value,
'Frequency',Fsquare_value);

            if ~isempty(Asquare_value)&& ~isempty(Fsquare_value)
&& isnumeric(str2double(Asquare_value))&&
isnumeric(str2double(Fsquare_value)) &&
str2double(Asquare_value)>0 && str2double(Fsquare_value)>0
                sim ('S_9');
            else
                errordlg('ERROR: Invalid Number of Control
Signals');
            end;

            time = output.time;
            output1 = output.signals.values(:,1);
            output2 = output.signals.values(:,2);
            output3 = output.signals.values(:,3);
            control1 = control.signals.values(:,1);
            control2 = control.signals.values(:,2);
            control3 = control.signals.values(:,3);
            thdif = thdif.signals.values(:,1);
            thdil = thdil.signals.values(:,1);
            thdvf = thdvf.signals.values(:,1);
            thdvl = thdvl.signals.values(:,1);
            vloadf = vloadf.signals.values(:,1);
            vloadl = vloadl.signals.values(:,1);
            iloadf = iloadf.signals.values(:,1);
            iloadl = iloadl.signals.values(:,1);
```

```
            vin = vin.signals.values(:,1);

            figure;

plot(time,output1,'r',time,output2,'b',time,output3,'y');
            title('Output (Load) - Sim 9');

            set(handles.text_thdif,'String',abs(thdif(end)));
            set(handles.text_thdil,'String',abs(thdil(end)));
            set(handles.text_thdvf,'String',abs(thdvf(end)));
            set(handles.text_thdvl,'String',abs(thdvl(end)));
            set(handles.text_vin3,'String',abs(vin(end)));
            set(handles.text_vloadf,'String',abs(vloadf(end)));
            set(handles.text_vloadl,'String',abs(vloadl(end)));
            set(handles.text_iloadf,'String',abs(iloadf(end)));
            set(handles.text_iloadl,'String',abs(iloadl(end)));

            set(handles.text_thdi,'String','-');
            set(handles.text_thdv,'String','-');
            set(handles.text_vin,'String','-');
            set(handles.text_vload,'String','-');
            set(handles.text_iload,'String','-');

        case 10

        case 11

        case 12

        case 13

        case 14

        case 15

        case 16

        case 17

        case 18

        case 19

        case 20

        case 21

        case 22

        case 23

        otherwise
             disp('Other Simulation');
    end
```

```
else
    errordlg('ERROR: Invalid Number of Parameters to modify');

end
```