

An Alternative Pulse-Width Modulation Scheme for Automotive Inverters

Bulent Buyukbozkirli
Hafid Chrifi-Alaoui

May, 2001

Abstract: Future automotive 42-Volt DC electrical systems will use pulse-width modulated inverters to convert the bus voltage to a 3-phase AC voltage with desired amplitude and frequency. In this report we develop a method using Genetic Algorithms that finds switching algorithms that reduce the number of filter capacitors needed at the DC side of the inverter, thus decreasing the production cost of the inverters significantly while maintaining acceptable AC bus power quality.

Work done for McCleer Power, Inc. under the direction of C. R. MacCluer and P. J. McCleer, in partial fulfillment of the requirements of Michigan State University MTH 844.

Table of Contents

Introduction	1
A Simplified Example	3
Genetic Algorithms	4
Simulation Results	7
Application of Genetic Algorithms	11
Conclusion	15
Acknowledgments	16
References	17
Appendix A.	18

1 Introduction

DC voltage can be converted to a sinusoidal AC voltage, either single phase or three phase, with desired amplitude and frequency by means of electrical devices called “switch-mode inverters.” The basic idea of these inverters is sketched in Figure 1. The DC voltage is converted to an AC voltage that alternates between values 0 and a certain fixed voltage. Only the fundamental component of this voltage is to be used as the output. Thus, the high frequency harmonics are filtered and the desired sinusoidal voltage is obtained.



Figure 1 Passage from DC to AC voltage by switch-mode inverters.

Figure 2 shows the details of such an inverter. This is one leg of a 3-phase inverter, which provides an AC voltage V_A , by means of the switches T_{A+} and T_{A-} . The objective is to maintain the desired output power specifications while providing a voltage V_A with a specific amplitude and frequency but without unnecessary harmonics. The whole art of switching scheme design comes into play at this point.

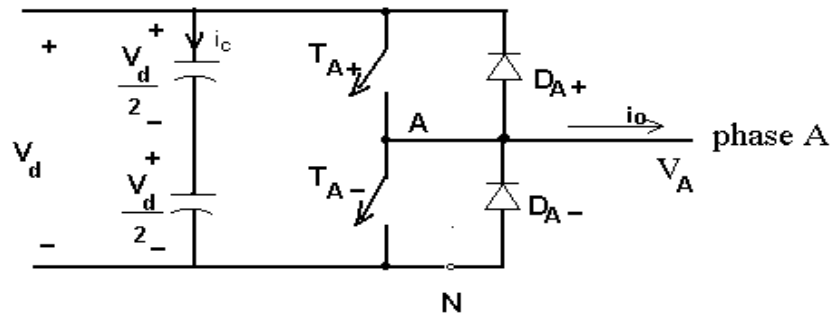


Figure 2 One leg of a 3-phase inverter

The ripples in the output current i_o result in ripples in the current i_c through the capacitors. This ripple current causes the capacitors to warm over time, which is very

undesirable since the temperature in the engine hood under normal operating conditions can approach 120°C. Such high temperatures can damage the capacitors.

Because of the high cost of these capacitors, any reduction in their number will reduce the cost of the inverter. Reducing the number of capacitors can be obtained by reducing the current i_c . Hence, the main focus of our design is to minimize i_c while still meeting the desired output voltage and/or current specifications.

There are two commonly used switching schemes: Pulse-Width-Modulated (PWM) switching and square-wave switching (SWS) [1]. In one of the standard PWM switching schemes, the duration of the ON-interval is determined such that the area under the output power signal during this interval is equivalent to the area under the desired output power signal during both this ON-interval and the following OFF-interval, hence an equivalent amount of output power is delivered to the load. In this report, we use this method to obtain a switching pattern that will be used as a reference to compare with our new design patterns.

In another PWM switching scheme, which is also known as sinusoidal PWM scheme, a fixed triangular reference signal is compared with a sinusoidal control signal whose amplitude and frequency can be changed, to determine the duration of ON and OFF intervals. When the control signal is smaller than the reference signal, the switch T_{A+} is turned off and T_{A-} is turned on. The switches change state when the control signal is larger than the reference signal. The amplitude and frequency of the output voltage are controlled by changing the control voltage.

In the square-wave switching SWS scheme, which can be considered as a crude special case of the PWM scheme, each switch of the inverter is on for one half-cycle of the desired output frequency.

The practice in automotive inverters is to control the switches by a central processing unit (CPU), which can be programmed to turn the switches on and off in any desired pattern. The switching is accomplished with high-current field-effect transistors (FET). The advantage of this method is the ability to control the fundamental component while eliminating some of the harmonics. However, one should be careful with the frequency of switching since, in practice, if a switch turns off in an inverter leg, the turn-on

of the other switch is delayed by a blanking time, which introduces low-order harmonics in the output.

2 A Simplified Example

In this section we analyze a simplified half-wave version of Figure 2 as shown in Figure 3. Our goal is to illustrate with this simple case that it is possible to find, by means of Genetic Algorithms, a switching pattern that satisfies desired specifications, yet decreases capacitor currents. The basics of Genetic Algorithms are described in Section 3.

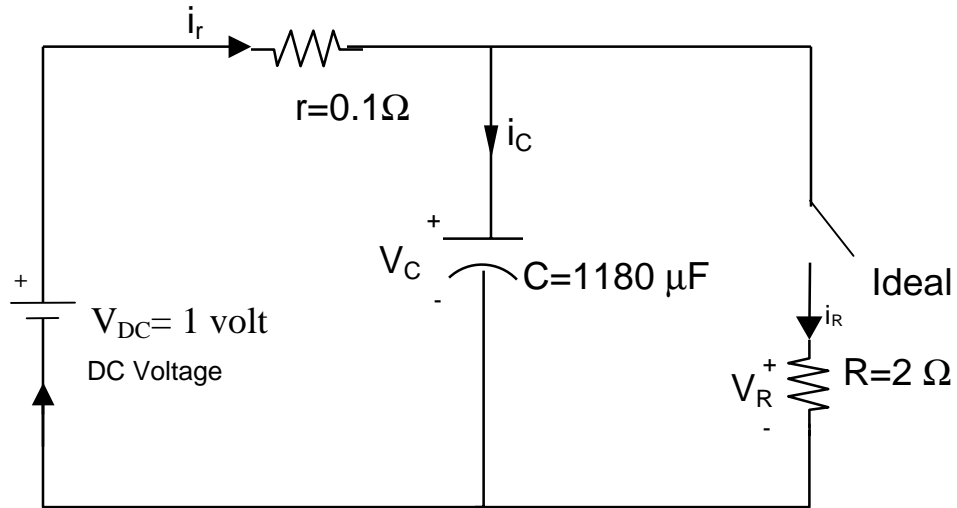


Figure 3 Current chopping by ideal switch

In our model, Figure 3, a resistive load is used for simplicity. The resistance r represents the wiring losses. All the values of the circuit elements are normalized so that the battery voltage is 1 volt.

The voltage V_c across the capacitor satisfies the differential equation

$$\frac{dV_c}{dt} + \left(\frac{1}{rC} + \frac{1}{RC} \right) V_c = \frac{V_{DC}}{rC}, \quad (1)$$

where $R = \infty$ when the switch is open. The solution to (1) is

$$V_c(t) = \frac{R}{r+R} V_{DC} + A e^{-\frac{1}{C} \left(\frac{1}{r} + \frac{1}{R} \right) t}, \quad (2)$$

where the constant A is determined by the initial value of V_c . Setting $G = 1/R$, $g = 1/r$ and finding the value of A in terms of the initial voltage $V_c(0)$, (2) becomes

$$V_c(t) = \frac{1}{rG+1}V_{DC} + (V_c(0) - \frac{1}{rG+1}V_{DC})e^{-\frac{(g+G)t}{C}}. \quad (3)$$

Here we take the conductivity $G=0.5$ when the switch is ON, and $G=0$ when the switch is OFF. Using solution (3) for the voltage it is quite easy to find the currents through the resistance R and r , since we have $i_R = V_c / R$ when the switch is on (otherwise zero), and $i_r = (V_{DC} - V_c) / r$. Section 4 contains the results of the simulation of this circuit using MATLAB.

3 Genetic Algorithms

Genetic Algorithms (GAs), developed by J. H. Holland in 1975, [2][3], are search algorithms based on the dynamics of natural selection and genetics. They represent the evolution of a given population in the parameter space towards an optimal configuration, the only one that is able to “survive” in relation with the given experimental data. More information and the mathematical foundations of GAs can be found in [4]. GAs have been applied in the control design of PWM digital controllers [5], but never to our knowledge to finding the optimal switching pattern of the power inverters as presented in this report. The wide interest in GAs is due to their computational simplicity and their capability of detecting global optima in the presence of many local minima.

Genetic algorithms are superior to other optimization and search procedures in four ways:

- **Direct use of a coding:** GAs require the natural parameter set of the optimization problem to be coded as a finite-length string over some finite alphabet. Thus, they work with a coding of the parameter set, not the parameters themselves.

- **Search from a population:** In many optimization methods, we move from a single point in the decision space to the next using some transition rule. By contrast, GAs work from a rich database of points simultaneously (a population of strings), climbing many peaks in parallel; thus, the probability of finding a false peak is reduced over methods that go point-to-point.

- **Blindness to auxiliary information:** To perform an effective search for better and better structures, GAs exploit all the information directly from the fitness

function (objective function) associated with individual strings and therefore do not require any auxiliary information or specific assumptions about the parameter space (such as continuity or differentiability).

• **Randomized operators:** GAs use probabilistic transition rules to guide their search, yet they are not a simple random search. Random choice is used as a tool to guide a search toward regions of the search space with likely improvement.

In order to illustrate how the genetic algorithm works, we consider the following simplified example: Suppose that the switch of our circuit turns on 5 times during each period of the output frequency. We want to determine the start time and duration of each turn-on period. So, we can take the time difference between the start times of each turn-on period as our parameter space and ask the genetic algorithm to find optimum turn-on time values for us. Thus, each member of our population is represented by a string (chromosome) of 5 integers, such as 3-6-2-7-4. Hence, “3” here means that the first turn-on will take place after 3 units from the start of the period, and “6” means that the second turn-on will be after 6 units from the end of the first turn-on period and so on. Each integer in this string, called the “field,” may take values in a specified discrete range (in this example, we use 3-bit values: 0,1,2,...,7). To find the actual time lengths, one needs to normalize the string 3-6-2-7-4 so that the addition of its entries equals to one period of the output signal.

Genetic algorithm starts with an initial population containing many such strings. Successive generations are produced by means of three fundamental operators from the previous generation: *reproduction*, *crossover*, and *mutation*. To illustrate these operators, let us take an initial population of only 4 strings, for simplicity,

3-6-2-7-4
4-2-3-3-1
6-3-2-5-2
7-4-5-1-5 .

A *fitness* value is assigned to each of these strings, which is a measure of how good a string is. In other words, it is the figure of merit that we want to maximize. Assume that we have the following fitness values for these members as shown in Table 1.

Table 1

No.	String	Fitness	% of Total
1	3-6-2-7-4	73	50.31
2	4-2-3-3-1	12	8.28
3	6-3-2-5-2	6	4.15
4	7-4-5-1-5	54	37.26
Total		145	100.0

Reproduction is a process in which individual strings are copied according to their *fitness* values. A simple reproduction allocates offspring strings using a roulette wheel with slots sized according to fitness (Figure 4.)

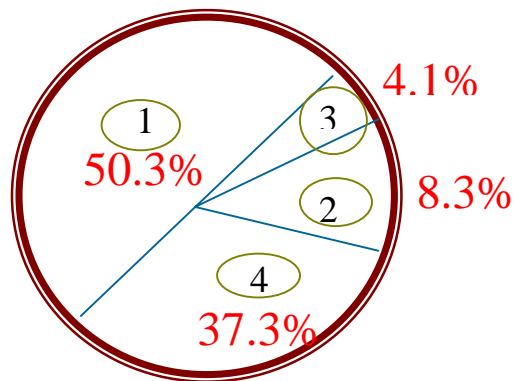


Figure 4 *Reproduction by roulette wheel*

In our example, it is most likely that first and the last strings will be reproduced in the second generation. A sample reproduction may give the following result:

3-6-2-7-4
7-4-5-1-5
4-2-3-3-1
3-6-2-7-4

Note that, the first member of the initial population is produced twice because of its high fitness value, while the third member is not represented anymore due to its small fitness value.

After reproduction, simple *crossover* may proceed in two steps. First, members of the newly reproduced strings are mated at random. Second, each pair of strings undergoes

crossing over as follows: an integer position k along the string is selected uniformly at random between 1 and 4 (i.e., the string length less one). Two new strings are created by swapping all characters after position k . For example, if 3-6-2-7-4 and 4-2-3-3-1 are mated and the position 3 is selected, then the crossover will generate 3-6-2-3-1 and 4-2-3-7-4, which are obtained by swapping the last two fields of the strings. Similarly, the mating of the other two strings with a swapping at the 2nd position would generate 7-4-2-7-4 and 3-6-5-1-5. Thus, our new population will be:

3-6-2-3-1
4-2-3-7-4
7-4-2-7-4
3-6-5-1-5

Lastly, the *mutation* is a random change in the value of a string's random field. Mutation happens usually with a very small probability, about 0.001. As an example, a mutation may change the 4th field of the string 7-4-2-7-4 as 7-4-2-3-4. Hence the final second generation is

3-6-2-3-1
4-2-3-7-4
7-4-2-3-4
3-6-5-1-5

After the fitness of the strings of this new generation is calculated, the third generation is produced following the same principles above. This procedure continues until the best string of the new generation converges to a required fitness.

Many other operators (e.g., *inversion*, *elition*, *gradient-bit search*, etc.) may also be added to accelerate and/or guarantee convergence.

4 Simulation Results

In this section, we synthesize a load power of $P(t) = A^2 R \sin^2 \omega t$, where A is the amplitude of the desired current through resistance R , i.e., $P(t)$ would be the power dissipated by R when the current is $i_R(t) = A \sin \omega t$ (see Figure 2.1). In our experiment, we take the frequency of the desired output current to be $f = \omega / (2\pi) = 200 \text{ Hz}$, with a switching frequency of 50 Hz , i.e., the switch turns ON an OFF 50 times during each cycle of the output 200 Hz .

First we find a symmetric PWM pattern assuming that the current through R remains fixed at the value $A=0.5$ amperes during the ON-intervals. The duration of ON and OFF intervals are found as follows: The time interval from $t=0$ to $t=0.005$ is divided into 50 equal sections. Then, for each section, the duration of ON-interval is calculated which would supply the same amount of energy to the load as the ideal sinusoidal power $P(t)$ would give during the given section. We will call this switching pattern as *Symmetric Linear Partition* pattern. Figure 5 illustrates this pattern on the graph of the corresponding power through R as a square-like graph. The sinusoidal graph in this figure is the graph of the desired ideal power. Note that, as the ideal power increases, the length of the ON-interval increases since one needs more integrated power to supply the energy that would be supplied in the ideal case.

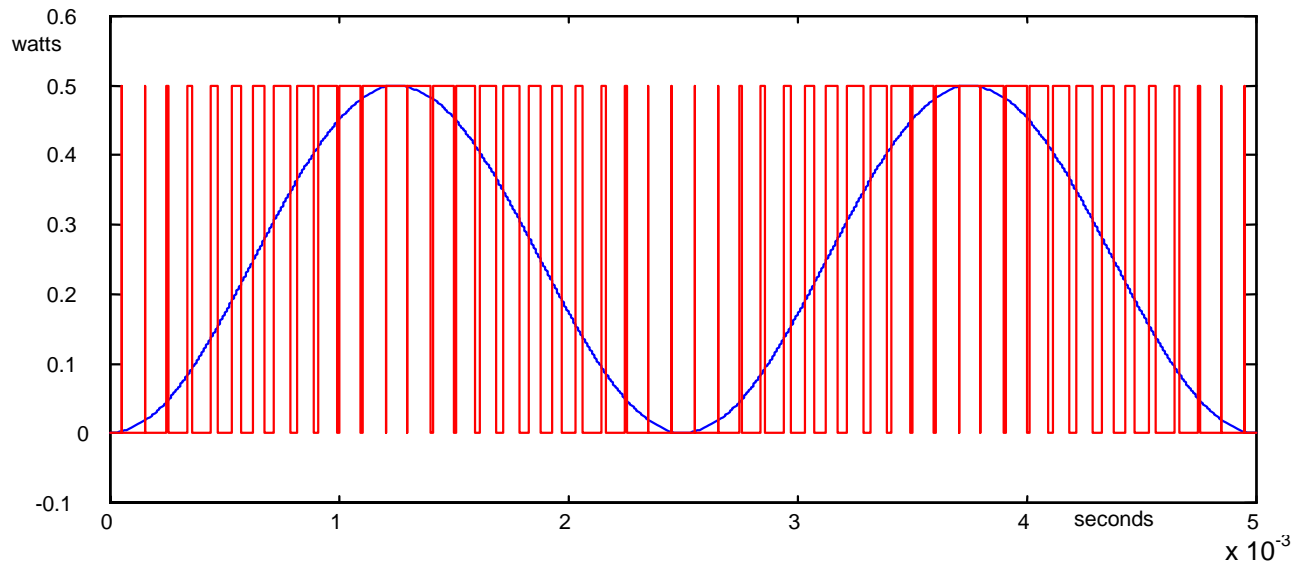


Figure 5 The synthesized and the ideal power through the resistance R .

In Figure 6(a) we see the graphs of the capacitor voltage V_C when the above ON-OFF pattern is applied. Here, we have a continuous graph that is either exponential growth or decay depending on the position of the switch (as expected). The corresponding battery current i_r behaves similarly as seen in Figure 6(b). We observe similar ripples on i_r .

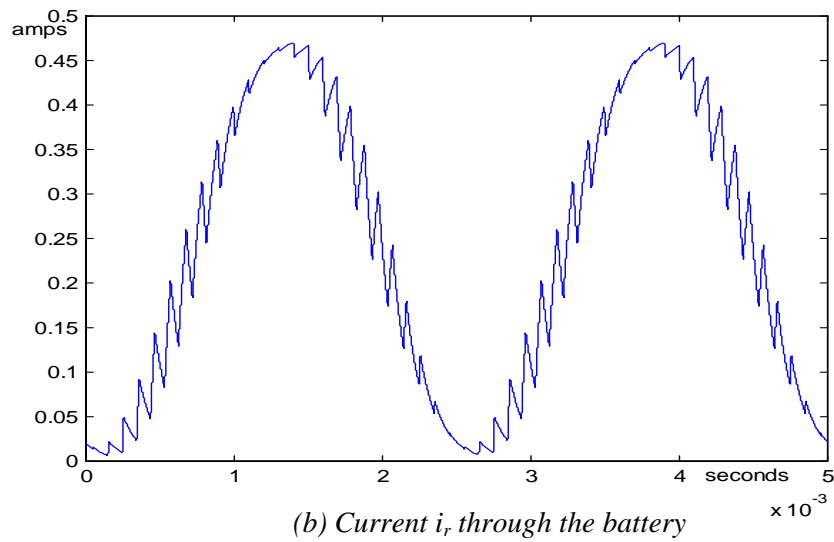
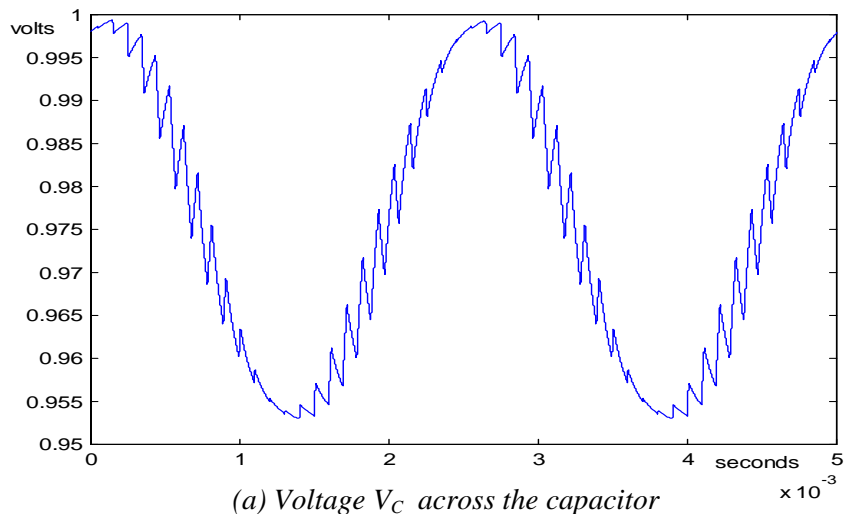
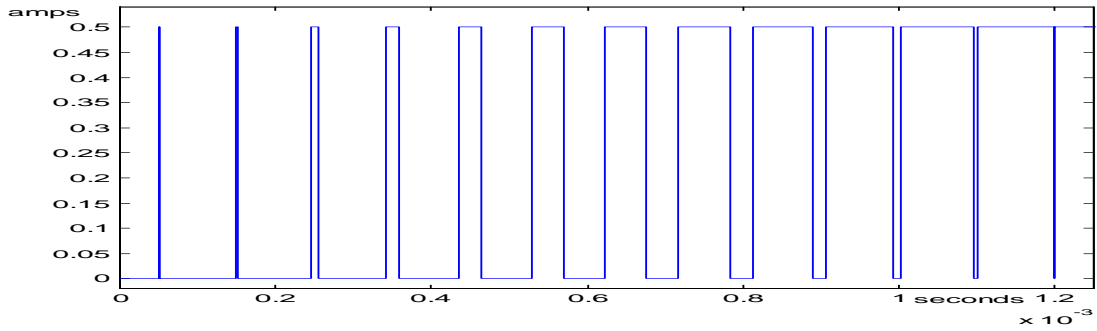
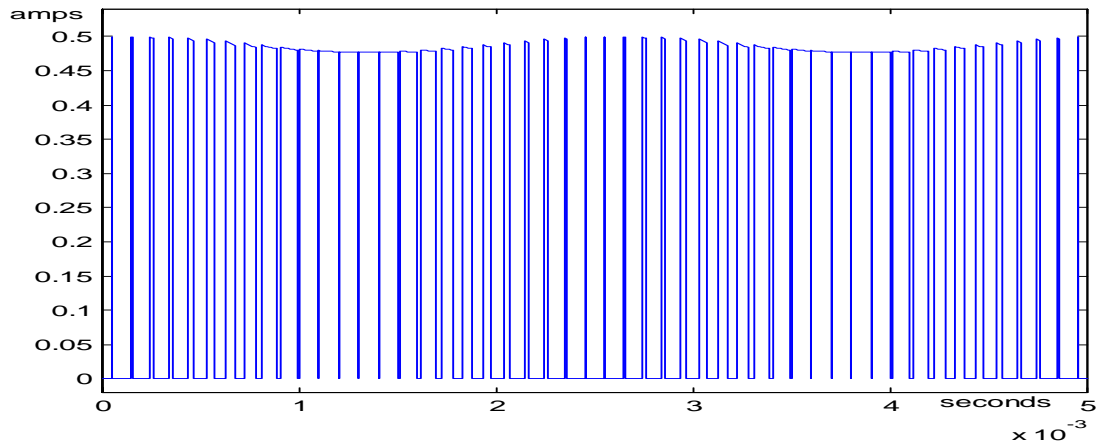


Figure 6

Figure 7 contains the graphs of both the current estimated, (which was assumed to be constant at $i_R=0.5$ ampere for ON-intervals), and the graph of the actual i_R when the switching pattern obtained above is applied. We observe that the actual current fluctuates around 0.49 ampere with small exponential increase or decrease during ON-intervals.



(a) Estimated load current i_R



(b) Actual load current i_R .

Figure 7

The difference between the estimated and the actual load current causes a loss in the desired power quality. Because of this loss, the actual energy supplied to the load R is less than the desired energy. Using the formula

$$\text{percentage energy loss} = 100 \times \frac{\int RA^2 \sin^2 \omega t \, dt - \int V_c \cdot i_R \, dt}{\int RA^2 \sin^2 \omega t \, dt},$$

where the integrals are taken over one period of $A \sin \omega t$, we have found this error to be 6.89%.

5 Application of Genetic Algorithms

In this section, we apply genetic algorithms to our simplified model of Figure 3. We used the software package, GALOPPS release 3.2, developed by Erik D. Goodman and the Genetic Algorithms Research and Applications Group at Michigan State University [6].

In our experiment, we considered two different cases: In Case 1, the string length is taken to be 101. The individual entries of each string are interpreted as the relative lengths of successive on and off intervals. Each field can take a 4-bit value, so the values 0,1,2,...,15 are the possible field values (Figure 8.) This switching scheme will be called *Non-symmetric Nonlinear Partition*.

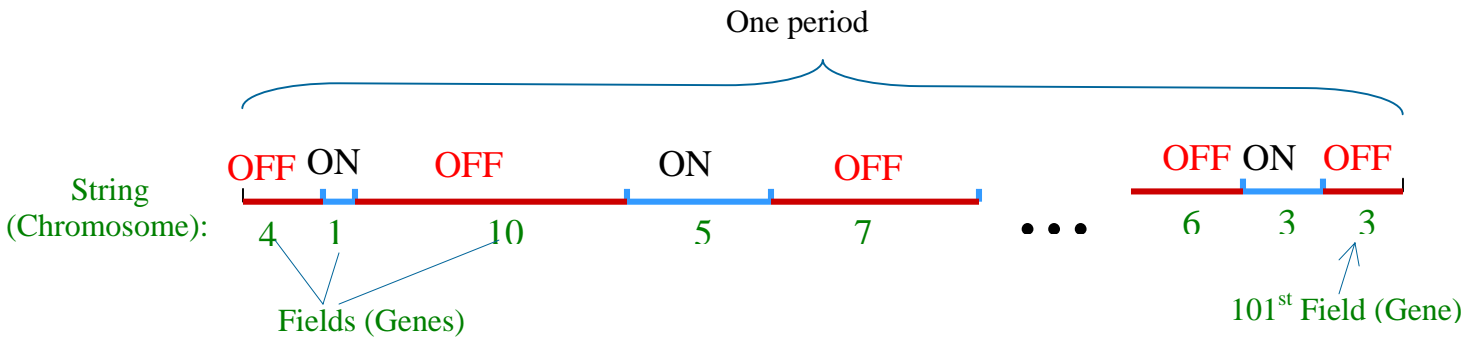


Figure 8 Strings (chromosomes) of Case 1: Non-symmetric Nonlinear Partition

In our experiment, we used a MATLAB routine to evaluate the fitness of each string. In this routine we convert the string values to actual time lengths. Since an arbitrary distribution of these intervals does not guarantee the desired shape of the output power, we have to include the measure of *goodness* of output power quality in the fitness function, as well as the measure of capacitor current. Thus, the fitness of a string is evaluated as follows: Consider the capacitor current i_c , and over one period calculate

$$Fit_1 = \frac{1}{\int i_c^2 dt} .$$

Then, during any on-interval, calculate the power $P_{\text{synthesized}}$ supplied to the load resistance. Also calculate the power P_{ideal} that would be supplied to the load, from the time when the given on-interval starts, to the beginning of the next on-interval, (as if the output current were a perfect sinusoidal signal with desired frequency and amplitude.)

Take

$$Fit_2 = \frac{1}{\sum_{\text{all ON-intervals}} (P_{\text{synthesized}} - P_{\text{ideal}})^2}.$$

A new fitness function F is taken as a weighted sum of Fit_1 and Fit_2 :

$$F = a \cdot Fit_1 + b \cdot Fit_2,$$

where constants a and b are chosen to bring Fit_1 and Fit_2 to the same order of magnitude.

GALOPP is run for 20 generations with 200 strings in each generation. Figure 9 shows the fitness F on the vertical axis, for each population.

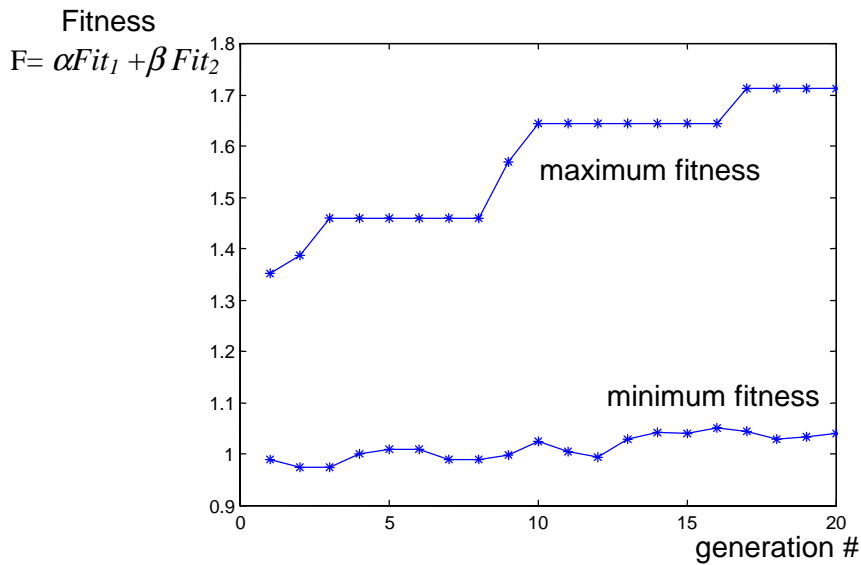


Figure 9 GALOPP results for Case 1

In Case 2, the string length is taken to be 50, in which the individual entries of each string are interpreted as the relative lengths of successive ON-intervals only (Figure 10.) Thus, the switch turns on 50 times during each period of the desired output signal. This switching scheme will be called as *Symmetric Nonlinear Partition*.

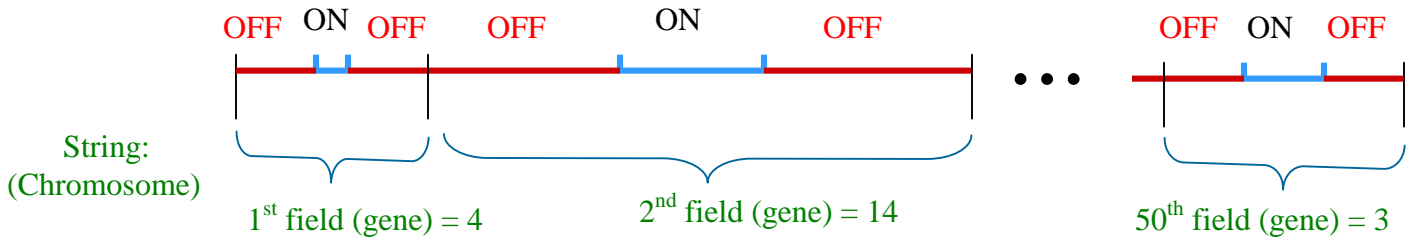


Figure 10 Strings (chromosomes) of Case 2: Symmetric Nonlinear Partition

In this case, we calculate the required duration of on-intervals, by a MATLAB routine, so that the desired output power requirement (within 10% of the ideal power, which would be supplied if the output current were an ideal sinusoidal signal) is satisfied. Then, using this pattern of ON and OFF intervals, the routine assigns the fitness of the string as only Fit_1 of the first case. After 20 generations, with 200 strings in each generation, we have obtained the results shown in Figure 11.

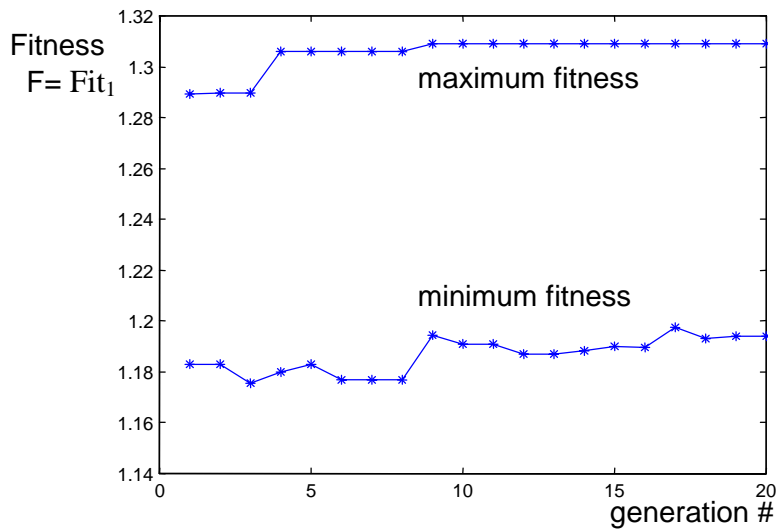


Figure 11 GALOPP results for Case 2

In both cases, Figures 9 and 11, we observe that the maximum fitness for each generation is getting better as the new generations are produced. We also observe an overall increase in the minimum fitness of each population, which means that there is an

improvement in the average member of each generation when the new generations are produced. In Case 1 (Figure 9) the maximum fitness value possibly did not converge to its best value since we see a regular increase in every 5 to 6 generations.

Table 2 below compares the commonly used symmetric linear partition of section 4 with Case 1 and Case 2. The second column is the percentage energy loss which is caused by the difference between the estimated current through the load R and the actual current obtained by the given ON-OFF switching design scheme. The percentage energy error for each case is within acceptable limits (within 10%). The third column shows that the switching pattern of Case 2 is better than the other two, as far as the amount of capacitor current is concerned. Case 1 was a very hard situation for GALOPP to work on since a switching pattern which is “good” for power quality may not usually be a “good” one for the capacitor current, thus making it quite difficult to improve both Fit_1 and Fit_2 at the same time. In the last column, we have the values measuring how well the output power imitates $RA^2 \sin^2 \omega t$. Here, we see that the switching pattern of Case 1 is not successful at all. On the other hand, when we compare Case 2 with linear partition case, we see that the improvement in the capacitor current is obtained at the expense of output power quality. However, Case 2 produces an output power that is quite close to the quality of the standard one. Although we have not looked at the Fourier analysis of the output power signal, we expect to see a higher harmonic distortion for Case 2 than the symmetric linear partition case.

Table 2 *Comparison of the three switching schemes*

	% energy loss	Fit₁ (Capacitor current)	Fit₂ (Output power quality)
Symmetric Linear Partition of section 4	6.8893	1.1920	26.5626
Case 1			
Non-Symmetric Nonlinear Partition	6.5420	0.8549	0.8569
Case 2			
Symmetric Nonlinear Partition	7.5295	1.3092	19.2824

In Figure 12, we see the comparison of all three cases over each generation for the capacitor current fitness Fit_1 only. This graph clearly shows that Nonlinear Symmetric Partitions obtained from GA application are better than the other two cases, as far as the capacitor current is concerned. Even in the first population that was selected randomly, we have a member that is better than the standard Symmetric Linear Partition. The following generations of Case 1 produce much better elements.

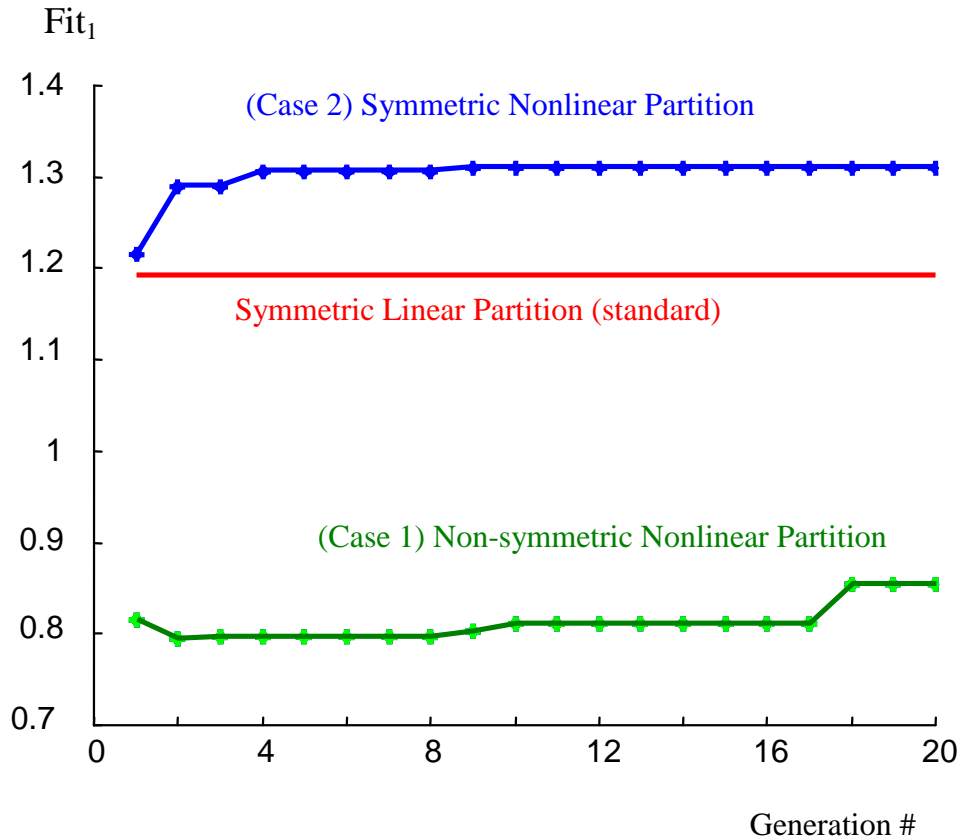


Figure 12 Comparison of GALOPP results of Case 1 and Case 2 with the standard pattern

6 Conclusion and Future work

In our simplified model, we achieved an improvement in the capacitor current at the expense of output power quality. The amount of this tradeoff can be controlled by the GA application. Even though we have started with a random selection of the initial population, we have achieved a successful result. Hence a few experiments proved that it is possible to find design schemes that are better than the standard patterns by means of GA.

One should be quite careful in choosing the proper *fitness* function, since it is the only information GALOPP receives about the particular problem. One may explore other possible *fitness* functions, e.g., harmonic distortion at the output, as well as different encoding of parameters for a better performance. A careful simulation of the circuit in consideration will be necessary to be able to compare different switching schemes. For example, the estimation errors in the calculation of the output current resulted in the output power loss in our analysis. That's why we had to take this into consideration when we compare different patterns. One may try to calculate the output current values and the capacitor voltages analytically, without any estimation errors, or instead of analytical calculations, one may use a circuit simulation application program, e.g., Simulink or Pspice, together with GALOPP. Once this simulation is achieved, one can easily apply our method presented in this report to both single-phase and three-phase inverters. In Appendix A, we present an analytical study of single-phase inverter with an induction motor model suggested by L. J. Giacoletto.

As a reference case, one may take the standard *sinusoidal* PWM switching scheme and compare it with the results of GA.

It may be necessary to run GALOPP several times for the same experiment since the production of new generations is based on probabilistic methods, which means that one may get different results by performing the experiment with the same parameters, even by starting with the same initial population.

In our experiments, the initial population, from which the following generations are produced, was created randomly. By a more careful selection of the initial population one may produce much better results.

7 Acknowledgments

The authors would like to express their thanks to C. R. MacCluer for his helpful suggestions and guidance, to E. D. Goodman for providing us with the GALOPP program and helping us to understand and apply GA to our specific problem, and to P. J. McCleer for suggesting this very interesting PWM problem to us.

8 References

- [1] N. Mohan, T. M. Undeland, W. P. Robbins, *Power Electronics*, second edition, New York: John Wiley & Sons.
- [2] J. H. Holland, *Genetic algorithms and the optimal allocations of trials*, SIAM Journal of Computing, 2 (2), 88-105, 1973.
- [3] J. H. Holland, *Adaptation in natural and artificial systems*, Ann Arbor: The University of Michigan Press, 1975.
- [4] David E. Goldberg, *Genetic Algorithms, in Search, Optimization & Machine Learning*, Addison-Wesley, 1989.
- [5] L.S. Shieh, W. Wang, J.S.H. Tsai, *Optimal digital design of hybrid uncertain systems using genetic algorithms*, IEE Proceedings online no. 19990082, 1999.
- [6] GALOPP , <http://garage.cps.msu.edu/software/software-index.html>

Appendix A

The circuit used in the analysis part of the project, is in fact a simplified version of the actual circuit, a picture of more realistic version of the actual circuit is shown in Fig A1:

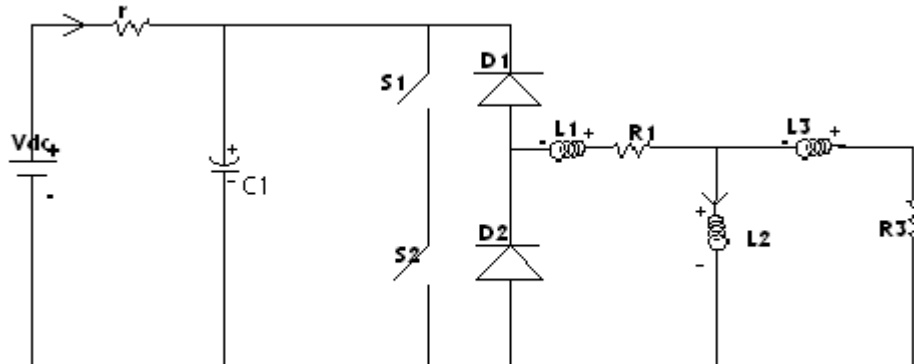


Figure A1

The induction motor equivalent circuit is introduced in the new version as a more realistic load. The two switches in the new version, allow the current to recirculate through diodes. When transistors are switched off, two cases are introduced:

The first case is when switch 1 is ON and switch 2 is OFF. Fig A2 describes the equivalent circuit.

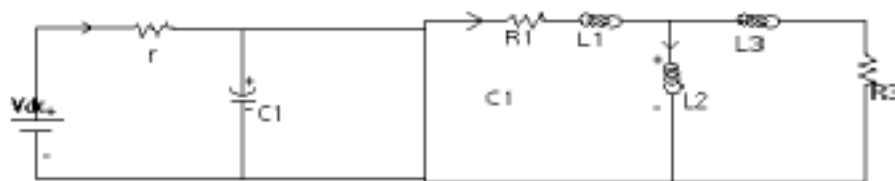


Figure A2

In general $U=Z*I$ (see Figure A3.)

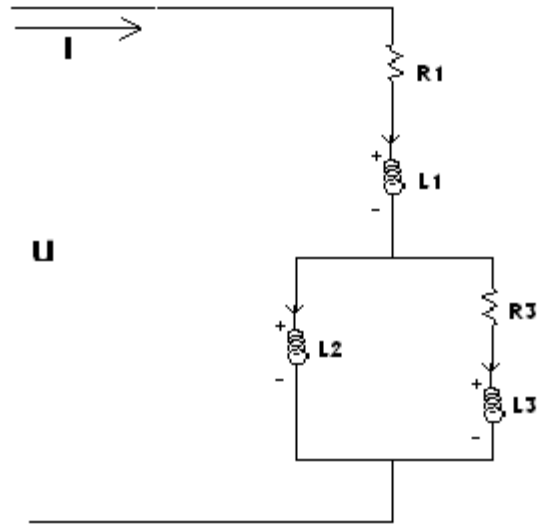


Figure A3

Now, we rewrite the circuit of Figure A2 as shown below in Figure A4

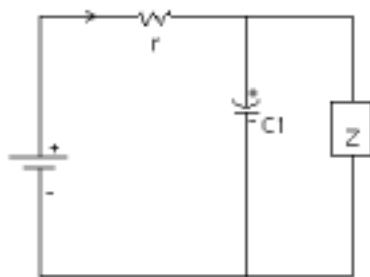


Figure A4

$$Z = R_1 + L_1 s + \frac{1}{\frac{1}{R_3 + L_3 s} + \frac{1}{L_2 s}} \quad \text{and} \quad U = (R_1 + L_1 s + \frac{R_3 + L_2 s + L_3 s}{R_3 L_2 s + L_2 L_3 s^2}) * I,$$

That is :

$$(R_2 L_2 s + L_2 L_3 s^2) * U = (R_1 + L_1 s) * (R_3 L_2 s + L_2 L_3 s^2) + R_3 + L_2 s + L_3 s;$$

And so has the time-domain model

$$R_3L_2 \frac{du}{dt} + L_2L_3 \frac{d^2u}{dt^2} = L_1L_2L_3 \frac{d^3i}{dt^3} + (R_3L_1L_2 + R_1L_2L_3) \frac{d^2i}{dt^2} + (R_1R_2L_2 + L_2 + L_3) \frac{di}{dt} + R_3 i.$$

When the PWM circuit as a whole is considered, we are led to the following system of state equations:

$$i_r = i_c + i, \quad (1)$$

$$V_{DC} - i_r * r = u, \quad (2)$$

$$C \frac{du}{dt} = i_c, \quad (3)$$

$$R_3L_2 \frac{du}{dt} + L_2L_3 \frac{d^2u}{dt^2} = L_1L_2L_3 \frac{d^3i}{dt^3} + (R_3L_1L_2 + R_1L_2L_3) \frac{d^2i}{dt^2} + (R_1R_3L_2 + L_2 + L_3) \frac{di}{dt} + R_3 i. \quad (4)$$

From (2) we have $i_r = \frac{V_{DC} - V_c}{r}$. If we substitute the i_r expression in (1) we get

$$\frac{V_{DC} - V_c}{r} = i_c + i.$$

Now if we use (3) to replace i_c by $C \frac{dV_c}{dt}$ we get

$$\frac{V_{DC} - V_c}{r} = C \frac{dV_c}{dt} + i.$$

Which yields to

$$i = C \frac{dV_c}{dt} - \frac{1}{r} V_c + \frac{V_{DC}}{r}. \quad (5)$$

Then by substituting equation (5) in (4) we can write a fourth order differential equation in V_c :

$$R_3L_2 \frac{dV_c}{dt} + L_2L_3 \frac{d^2V_c}{dt^2} = -L_1L_2L_3C \frac{d^4V_c}{dt^4} + \frac{L_1L_2L_3}{r} \frac{d^3V_c}{dt^3} + C(R_3L_1L_2 + R_1L_2L_3) \frac{d^3V_c}{dt^3} - \frac{1}{r} (R_3L_1L_2 + R_1L_2L_3) \frac{d^2V_c}{dt^2} - C(R_1R_3L_2 + L_2 + L_3) \frac{d^2V_c}{dt^2} - \frac{1}{r} (R_1R_3L_2 + L_2 + L_3) \frac{dV_c}{dt} - CR_3 \frac{dV_c}{dt} - \frac{R_3}{r} V_c + \frac{R_3}{r} V_{DC}.$$

So that the differential equation in the standard form

$$L_1 L_2 L_3 C \frac{d^4 V_c}{dt^4} + \left[\frac{L_1 L_2 L_3}{r} + C (R_3 L_1 L_2 + R_1 L_2 L_3) \right] \frac{d^3 V_c}{dt^3} + \left[L_2 L_3 + \frac{1}{r} (R_3 L_1 L_2 + R_1 L_2 L_3) + C (R_1 R_3 L_2 + L_2 + L_3) \right] \frac{d^2 V_c}{dt^2} + \left[R_3 L_2 + \frac{1}{r} (R_3 L_1 L_2 + R_1 L_2 L_3) + C R_3 \right] \frac{d V_c}{dt} + \frac{R_3}{r} V_c - \frac{R_3}{r} V_{DC} = 0. \quad (6)$$

An analytical solution of equation (6) is of great use here, since the Matlab code uses the output of the previous cycle as an initial condition for the current cycle. Numerical solution induces error terms, which accumulate over cycles and may have very undesirable effect the credibility of the results.

The second case is when switch (1) is OFF and switch (2) is ON. Thus there will be no current between the load side and the rest of the circuit (see Fig B1)

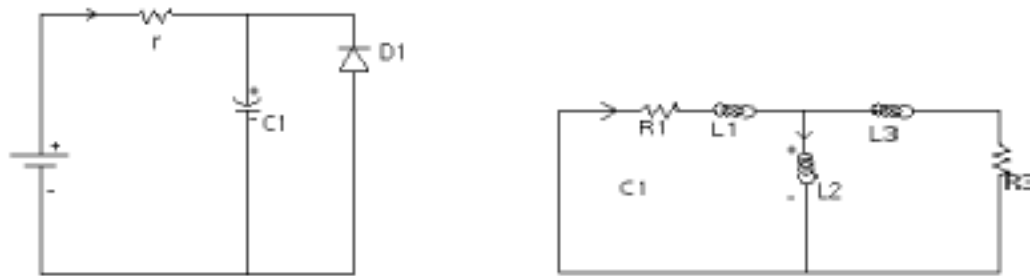


Figure B1 Under case 2 the circuit is equivalent to two independent components.

The response equation for the first component is given in analysis part of the project.

When analyzing the second component, we will concentrate on the current in the first resistor R_1 . The state equations are as follows:

$$i = i_2 + i_3, \quad (2.1)$$

$$L_1 \frac{di}{dt} + R_2 i = L_2 \frac{di_2}{dt}, \quad (2.2)$$

$$L_3 \frac{di_3}{dt} - R_3 i_3 = L_2 \frac{di_2}{dt}. \quad (2.3)$$

Substitute (2.1) in (2.2) and (2.3) to get

$$-L_1 \frac{di}{dt} + R_2 i = L_2 \frac{di}{dt} - L_2 \frac{di_3}{dt} \quad (2.4)$$

$$L_3 \frac{di_3}{dt} - R_3 i_3 = L_2 \frac{di}{dt} - L_2 \frac{di_3}{dt} \quad (2.5)$$

From (2) we can get $\frac{di_3}{dt}$ as a function of i

$$L_2 \frac{di_3}{dt} = (L_2 + L_1) \frac{di}{dt} + R_2 i. \quad (2.6)$$

That is

$$\frac{di_3}{dt} = \frac{L_2 + L_1}{L_2} \frac{di}{dt} + R_2 i. \quad (2.7)$$

On the other hand we can write (2.5) as

$$(L_3 + L_2) \frac{di_3}{dt} - R_3 i_3 = -L_2 \frac{di}{dt}. \quad (2.8)$$

Substituting $\frac{di_3}{dt}$ from (2.7) into (2.8) yields to

$$(L_3 + L_2) \left(\frac{L_2 + L_1}{L_2} \frac{di}{dt} + R_2 i \right) - R_3 i_3 = -L_2 \frac{di}{dt}. \quad (2.9)$$

Which yields to

$$\begin{aligned} i_3 &= \frac{1}{R_3} \left[(L_3 + L_2) \left(\frac{L_2 + L_1}{L_2} \frac{di}{dt} + R_2 i \right) + L_2 \frac{di_z}{dt} \right] = \\ & \left[\frac{(L_3 + L_2)(L_2 + L_1)}{R_3 L_2} + \frac{L_2}{R_3} \right] \frac{di}{dt} + \frac{(L_3 + L_2)R_2}{R_3} i. \end{aligned} \quad (2.10)$$

At the end we can use equation (2) to write a state equation in the parameter of interest i_3

$$\left(\frac{(L_3 + L_2)(L_2 + L_1)}{R_3 L_2} + \frac{L_2}{R_3} \right) \frac{d^2 i}{dt^2} + \frac{(L_3 + L_2)R_2}{R_3} \frac{di}{dt} = \frac{L_2 + L_1}{L_2} \frac{di}{dt} + R_2 i. \quad (2.11)$$

Giving the equation in standard form

$$\left(\frac{(L_3 + L_2)(L_2 + L_1)}{R_3 L_2} + \frac{L_2}{R_3} \right) \frac{d^2 i}{dt^2} + \left[\frac{(L_3 + L_2)R_2}{R_3} - \frac{L_2 + L_1}{L_2} \right] \frac{di}{dt} - R_2 i = 0. \quad (2.12)$$

The two cases studied in this section yield to ordinary differential equation with constant coefficients. A case easy to solve, still, in the first case the ODE is of order four, so we need four independent initial conditions to get an exact answer, that is we need to keep track of four independent variables from the previous cycle. And the same we need to keep track of three independent variables from the first case, to get initial conditions for the last two differential equations.