Islamic University of Gaza

Deanery of Graduate Studies

Faculty of Engineering

Electrical Engineering Department

Design of GA-Fuzzy Controller for Buck DC-DC Converters

A Thesis Submitted To The Faculty Of Engineering.

In Partial Fulfillment of the Requirements For The

Degree of Master of Science in Electrical Engineering

Prepared By:

Mohammed A. Kaabar

Advisor

Dr. Basil Hamed

1435هـ - 2014 م

صفحة نتيجة الحكم على البحث (نتيجة الحكم من قبل لجنة المناقشة)

DEDICATION

I dedicate this thesis to my Parents in recognition of their endless help and support. I also dedicate this work to my wife and to my children, Ahmed, Leila and Laian who are my most precious thing in this life.

ACKNOWLEDGMENTS

First and foremost, at the beginning, I thank ALLAH for giving me the strength and health to let this work see the light.

I would like to express my gratitude and appreciation to Dr. Basil Hamed for all the help and guidance he provided throughout my graduate program.

Special thanks go Dr. Assad Abu-Jasser and Dr. Mohammed Mushtaha - thesis examiners- for their patience, guidance, and generous supports during this research.

I would like to thank my family, parents and my wife for their encouragement, patience, and assistance overall years. I am forever indebted to my parents, who have always kept me in their prayers.

ABSTRACT

By

Mohammed A. Kabar

The Islamic University of Gaza, 2014

Gaza, Palestine

Fuzzy logic control (FLC) systems have been tested in many technical and industrial applications as a useful modeling tool that can handle the uncertainties and nonlinearities of modern control systems. The main drawback of the FLC methodologies in the industrial environment is challenging for selecting the number of optimum tuning parameters. In this dissertation, a method has been proposed for finding the optimum membership functions, output gain and inputs gain of a fuzzy system using genetic algorithm (GA). A synthetic algorithm combined from fuzzy logic control and GA is used to design a controller for DC-DC converter. To exhibit the effectiveness of proposed algorithm, it is used to optimize the triangle membership functions of the fuzzy model DC-DC converter system as a case study. It is clearly proved that the optimized fuzzy controllers provided better performance than a fuzzy model for the same system in terms of settling time, steady state error and ripple voltage at the output response. The evaluation of the output has been carried out by software simulation using MATLAB Simulink.

الملخص

انظمة التحكم الضبابية تم اختيارها في العديد من التطبيقات التقنية والصناعية كأداة تتحكم في أنظمة التحكم الحديثة الغير خطية والمجهولة. الغرض من استخدامات أنظمة التحكم الضبابية في البيئة الصناعية هو اختيار قيم المتغيرات المثالية.في هذه الأطروحة، يتم عرض طريقة مقترحة للتحكم في شكل الأعضاء ومعاملات التكبير للمداخل والمخرج للمتحكم الضبابي لإيجاد الشكل والقيم المثالية لها باستخدام الخوارزمية الجينية.تتكون الخوارزمية المركبة من المتحكم الضبابي مع الخوارزمية الجينية في تصميم محول ثابت - ثابت. لعرض التأثيرات الخوارزمية يستخدم لتفعيل وظائف الأعضاء في المتحكم الضبابي للمحول ثابت - ثابت كحالة دراسية. من الواضح بأن المتحكم الضبابي زود أداء أفضل من المتحكم الضبابي لوحده لنفس النظام من حيث زمن التسوية، خطأ الحالة المستقرة وتموج الجهد في استجابة الجهد الناتج. تم تقييم النتائج بواسطة برنامج الماتلاب/ السيميولينك.

Computer Software Used:

- Microsoft ®Windows 7 Operating Systems
- Microsoft ®Word 2010
- MATLAB ®Version 2010b
- Simulink Version
- Acrobat Reader X

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER1 INTRODUCTION

Switch mode DC-DC converters efficiently convert a direct DC input voltage into a regulated DC output voltage. Compared to linear power supplies, switching power supplies provide much more efficiency and power density. Switching power supplies employ solid-state devices such as transistors and diodes to operate as a switch: either completely on or completely off. Energy storage elements, including capacitors and inductors, are used for energy transfer and work as a low-pass filter. The buck converter and the boost converter are the two fundamental topologies of switch mode DC-DC converters. Most of the other topologies are either buck-derived or boost-derived converters, because their topologies are equivalent to the buck or the boost converters.

1.1 Switch-Mode DC-DC Converters

Switch-mode DC-DC converters are used to convert the direct DC input to a controlled DC output at a desired voltage level. Switch-mode DC-DC converters include buck converters, boost converters, buck-boost converters, Cuk converters and full-bridge converters, etc. Among these converters, the buck converter and the boost converter are the basic topologies. Both the buck-boost and Cuk converters are combinations of the two basic topologies. The full-bridge converter is derived from the buck converter [1].

There are usually two modes of operation for DC-DC converters: continuous and discontinuous. The current flowing through the inductor never falls to zero in the continuous mode. In the discontinuous mode, the inductor current falls to zero during the time the switch is turned off. Only operation in the continuous mode is considered in this dissertation [1].

1.1.1 Buck Converter

The buck converter, shown in figure 1.1, converts the unregulated source voltage V_{in} into a lower output voltage V_{out} . The NPN transistor shown in figure 1.1 acts as a switch. The ratio of the ON time (t_{ON}) when the switch is closed to the entire

switching period (T) is defined as the duty cycle. The corresponding pulse width modulation (PWM) signal is shown in figure 1.2 [2].

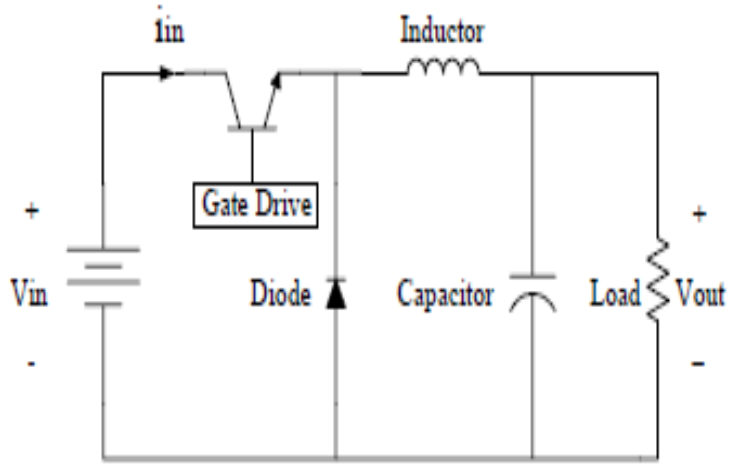


Figure 1.1: Buck Converter

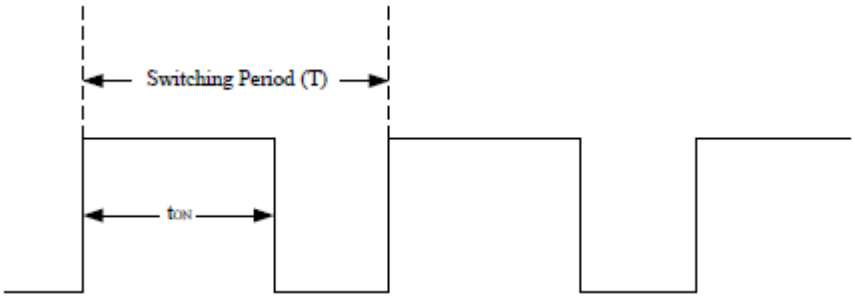


Figure 1.2: PWM Signal to Control the Switches in the DC-DC Converters

The equivalent circuit in figure 1.3 is valid when the switch is closed. The diode is reverse biased, and the input voltage supplies energy to the inductor, capacitor and the load. When the switch is opened as shown in figure 1.4, the diode conducts the capacitor supplies energy to the load, and the inductor current flows through the capacitor and the diode. The output voltage is controlled by varying the duty cycle. During steady state, the ratio of output voltage over input voltage is D, which is given by (1.1) [3].

$$D = \frac{V_{out}}{V_{in}} \qquad (1.1)$$

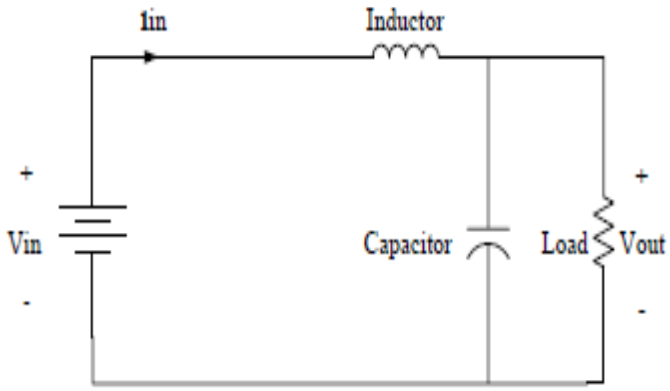


Figure 1.3: Equivalent circuit of the buck converter when the switch is closed

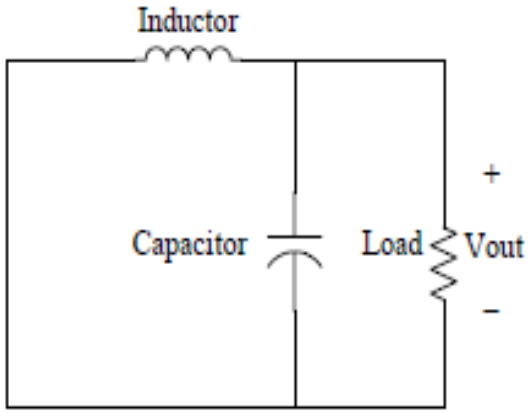


Figure 1.4: Equivalent Circuit of the Buck Converter When the Switch is Open

1.2 Control of DC-DC Converters

The output voltage of the switch-mode DC-DC converters is regulated to be within a specified range in response to changes in the input voltage and the load current. There are two control methods for DC-DC converters: voltage mode control and current mode control [3].

In voltage mode control, the converter's output voltage is compared with a reference to generate the voltage error signal. The duty cycle is adjusted based on the error signal to make the output voltage follow the reference value. Frequency response methods are usually used in the design of voltage mode controllers for DC-DC converters. Small signal model of the converters is first obtained by linearizing the power stage of the converters around an operating point, and then a compensator is designed based on the small signal model. Typical compensators include phase-lead compensator, phase-lag compensator and lead-lag compensator. In analog control, the

compensators are implemented using operational amplifiers and appropriate values of resistors and capacitors to obtain the desired transfer function. In digital control, the control algorithm is implemented on a microcontroller or DSP. Current mode control for a DC-DC converter is a two-loop system. An additional inner current loop is added to the voltage loop. The current loop monitors the inductor current and compares it with its reference value. The reference value for the inductor current is generated by the voltage loop [1].

1.3 Literature Review

The research problem addressed in this dissertation is the design and implementation of controllers for buck converters using Fuzzy control methods. Digital control for DC-DC converters is theoretically interesting because it is a multi-disciplinary research. Theory in the areas of power electronics, systems and control, and computer systems are all needed to conduct research in digital control of DC-DC converters. The increasing interest in digital control of switch mode power supplies is shown in international conference proceedings and journal publications in the past few years.

- In June 1995, Wang and Lee designed a fuzzy controller for basic DC-DC converters and then compared the computer simulation results with those for current-mode control in buck, boost and buck-boost converters [4]. It was concluded from the comparison of start-up responses and load regulation tests that the current-mode controlled buck converter had a faster transient response and better load regulation, while the fuzzy controller for both boost and buck-boost converters had less steady-state error and better transient response.
- In May 1996, J. Arias, A. Arias, et al. proposed a design procedure for a fuzzy logic controller for a buck converter [5]. The control rules were derived from analysis of the system dynamics in the state plane which use try an error to generate membership.
- In Jan 1997, Mattevelli and Spiazzi investigated a general-purpose fuzzy controller for DC-DC converters [6]. The fuzzy controller improved performance in terms of overshoot limitations and sensitivity to parameter variations compared to standard controllers. Simulation results for buck-boost and Sepic converters were presented.

- In Feb 1999, Campo and Tarela investigated the consequences of the finite word length on the performance of a digital fuzzy logic controller [7]. There were three types of error as a result of the finite word length: AD conversion errors, membership function errors and arithmetic errors. Simulation results showed that bias and limit cycles were generated due to the quantization.
- In September 2002, Viswanathan, Srinivasan and Oruganti studied the development of a universal fuzzy controller for a boost converter [8]. Simulation results were compared with the results of a PI controller under varying operating points. The performance of the fuzzy controller was superior to the performance of the PI controller.
- In June 2004, Perry and Sen proposed a design procedure that integrated linear control techniques with fuzzy logic [9]. The small signal model for the converters and linear design techniques were used in the initial stages of fuzzy controller design. Simulation and experimental results were presented and compared with results of a digital PI controller.
- In July 2011, Feshki Farahani proposed a design of a fuzzy controller and comparing with PI digital controller [10]. These comparisons show that the fuzzy controller has faster dynamic when compared with the PI digital classic.

1.4 Thesis Contribution

In this thesis, we will use fuzzy controller for DC-DC Buck converter. By using Matlab/Simulink we will design the fuzzy logic controllers and apply the GA (Genetic Algorithm) to optimize the memberships fuzzy controller to improve the settling time and the ripple of the output voltage of the system.

1.5 Outlines of the Thesis

This thesis consists of five chapters. Chapter 2 presents the DC-DC Converter model. Chapter 3 presents a review and introduction to fuzzy logic and its application, fuzzy sets operations, the main concepts in fuzzy sets such as membership functions, and linguistic variable. Chapter 4 presents a review and introduction to genetic algorithm; its uses and main concepts in genetic algorithm such as cross over, mutation, reproduction. Chapter 5 presents analysis and simulation results of the fuzzy controller without GA and with GA, while the last chapter concludes this thesis.

CHAPTER 2 DC-DC CONVERTERS

2.1 Converters

In electrical engineering, power engineering and the electric power industry, power conversion is converting electric energy from one form to another. On the application side, the end user uses the power generated to run his appliances. These appliances in turn use different types of voltage (direct or alternating) with different magnitude levels. Hence, there is a need to convert these voltages as per requirement. That is why converters come into picture? One may ask a question, why don't we reduce the magnitude levels using a potential divider? If at all the voltage or power level is the only concern for the appliances. But, there lies an urge to reduce the loss component in the circuit (to improve efficiency), make it compact, incorporate changes in parameters if required (like frequency adjustments), as well as the most important reason to make it economical.

Considering all the above factors, the necessity of a converter is well defined. Converters are classified into 4 types depending on the type and magnitude level of voltages is concerned:

1. AC-AC converters
2. AC-DC converters
3. DC-DC converters
4. DC-AC converters

We deals with the DC-DC converters, so let us have a quick introduction about the DC-DC converters.

2.2 DC-DC CONVERTER

A DC-DC converter is an electronic circuit power class converter which converts a source of direct current (DC) from one voltage level to another [11]. If the voltage magnitude level at the output is greater than the input, then the converter is called a boost converter, while if the voltage magnitude level at the output is less than the input, then the converter is called a buck converter. Buck converter is explained in

detail in this document. Buck converters are, as explained above, converts a source of direct current from higher voltage magnitude level to a lower voltage magnitude level.

One may think, why do not we use a linear regulator for reducing the voltage? The answer is that the linear regulators prove to be efficient over the switching regulators (such as the above converters) when the output is lightly loaded or when the desired output voltage is very close to the source voltage. Also linear regulators are void of magnetic circuit involving transformers or inductors thus resulting in a relatively smaller circuit. Other than these conditions if a linear regulator is used, the BJT which is usually used for regulation acts as a resistor, with the power loss across it. In order to make loss due to resistive drop almost negligible, we use switching regulators.

Buck converters are used only if the voltage level to be reduced is less than the order of 10. If the scaling is more than 10, then we need to use an isolated type, flyback converters. Considering the non-isolated buck converters, let us go for the detailed study of buck converters firstly.

2.2.1 Buck Converters

Figure 2.1 shows the typical buck converter diagram. As shown, the converter consists of the following components.

1- Direct Voltage Source - V_g
2- Power Converter (represented as Switch) - S_W
3- Filter Inductor - L_f
4- Filter Capacitor -C_f
5- Load Resistor - R_L
6- Freewheeling Diode - D_f

1. Voltage Source

Voltage Source is the basis of the converter circuit. In a typical buck converter as shown in Figure 2.1, one can see that the source voltage is direct voltage. The characteristics of an ideal direct voltage source is to provide a constant voltage independent of the current drawn from it by the load.

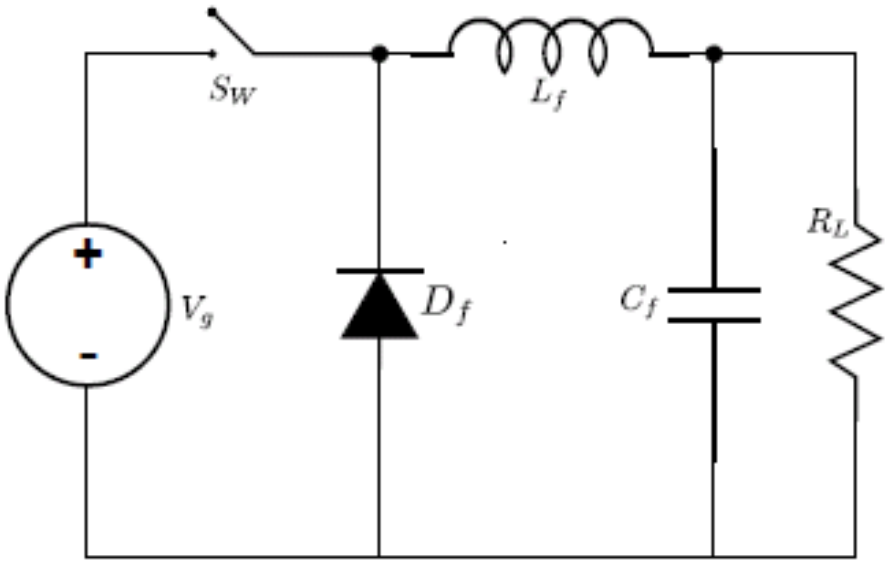


Figure 2.1: Typical Buck Converter

Practically, the voltage sources fall short of ideal and the terminal voltage drops with the increase in the output current drawn by the load. For simplicity, we have considered the source as ideal in our analysis.

2. Power Converter

A Power Converter is referred to a power electronic circuit that converts voltage and current from one form to another. An operative unit for electronic power conversion, comprising electronic valve devices, transformers and filters which necessary and auxiliaries in electronic Power Converter [12]. So, the term converter refers to a power electronic circuit that takes in electric power in one form and gives out electric power in another form [13]. A converter is a combination of one or more solid state electronic switching devices with filtering elements like inductors and capacitors (void of resistors to reduce power loss and hence increase efficiency) connected in different topology, depending on the application.

The switching element is an important part of a converter and is explicitly shown in the previous figure. Control of the switch is provided externally with the help of triggering circuits, control circuits or even firing circuits. This converter block is termed as a power circuit or power electronic circuit. The control circuit operates with the feedback signal from the output, and sometimes with an addition of feed forward signal from the input and a reference signal.

3. Filter Inductor

The filter inductor is required to smoothen out the current ripple as well as to reduce the high $\frac{di}{dt}$ caused at the instant of switching. This will not only improve the regulation of the output, but it protects also the switch to be stressed due to switching surge in current during switching.

4. Filter Capacitor

The Filter Capacitor is required to smoothen the output voltage waveform. It filters the rippled waveform and provides a constant output voltage across the load.

5. The Load

Load is any appliance which is connected across the output terminals of the converter. This can be a simple resistive load as shown in figure 2.1.

6. The Freewheeling Diode

Let us analyze the above circuit shown in figure 2.1 without the freewheeling diode and the capacitor (for simplicity). First let us close the switch S_W at time t_X, a current is established in the circuit through the load; and then if the switch is opened at the instant t_Y, the inductor L_f is charged and as soon as the switch is opened, the inductor looks for a path to discharge its stored energy. Otherwise, this leads to a sparking across the switch terminals leading to the malfunction or destruction of the switch itself due to the high amount of heat dissipated across the switch.

In order to provide a path for the inductor to dissipate the current, we require a freewheeling diode, which provides a path for the inductor current to flow through the load as soon as the switch is opened.

2.2.2 Analysis of a Typical Buck Converter

Oxford Dictionary [14] defines the term “Buck” as lowest of a particular rank. One might ask a question, as the Buck converter involves the conversion of direct voltage to direct voltage. So, how can it be termed a conversion? The answer is that a

fixed direct voltage is converted into a variable direct voltage source without changing the power level (neglecting power losses in the switch).

The analysis of the buck converter is as shown in figure 2.2[11], and the corresponding waveforms are as shown in figure 2.3. Assuming that the current flow in the inductor is continuous, we are now analyzing the buck converter in continuous conduction mode (CCM). From the basic principles, we know that,

$$e_L = L_f \frac{di}{dt} \qquad (2.1)$$

Assuming that the inductor current rises linearly in the inductor as shown in the waveform 2.3, we can say that.

$$V_g - V_o = L_f \frac{I_2 - I_1}{t_1} \qquad (2.2)$$

Where V_o is the output constant voltage across the load resistor.

$$t_1 = L_f \frac{\Delta I}{V_g - V_o} \qquad (2.3)$$

where $\Delta I = I_2 - I_1$.

This is for the period for which is ON. This time period t_1 is called ON time period. And is donated by $t_{ON} = kT_S$.

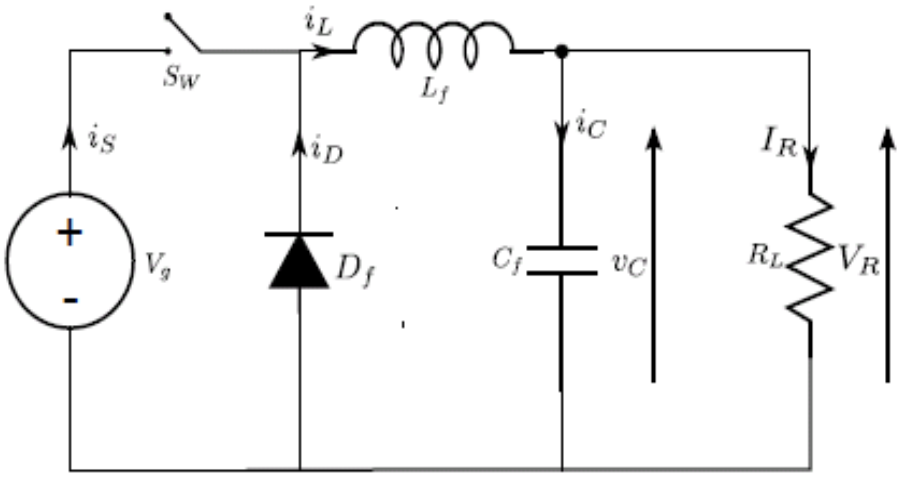


Figure 2.2 Buck Converter

For time period, $(1-k)T_S = t_{OFF}$, that is the time period t_2 for which, the switch is OFF, the input is Zero, so V_g = 0.

$$0 - V_o = L_f \frac{I_1 - I_2}{t_2} \tag{2.4}$$

$$V_o = L_f \frac{\Delta I}{t_2} \tag{2.5}$$

$$t_2 = L_f \frac{\Delta I}{V_o} \tag{2.6}$$

From equation 2.3 and 2.6, we get,

$$\Delta I = \frac{(V_g - V_o)t_1}{L_f} = \frac{V_o t_2}{L_f} \tag{2.7}$$

Simplifying the above equation 2.7, to get an expression for average voltage V_o, we get

$$V_o = kV_g \tag{2.8}$$

Where $k = \frac{t_{ON}}{T_s}$ is the duty cycle of the switch.

Neglecting the losses in the switch, we can write the input power is equal to output power.

$$V_g I_g = V_o I_o \tag{2.9}$$

where,

I_g is the source current

I_o is the average output current

From the above equation 2.9, we can write,

$$I_o = \frac{1}{k} I_g \tag{2.10}$$

To find the switching period, consider the equation

$$T_s = \frac{1}{f_s} = t_1 + t_2 = \frac{L_f \Delta I}{V_g - V_o} + \frac{L_f \Delta I}{V_o} \tag{2.11}$$

$$T_s = \frac{L_f V_g \Delta I}{V_o (V_g - V_o)} \tag{2.12}$$

To find the peak to peak ripple current, use equation 2.12, we get

$$\Delta I = \frac{V_o (V_g - V_o)}{f_s L_f V_g} \tag{2.13}$$

or,

$$\Delta I = \frac{V_g k (1 - k)}{f_s L_f} \tag{2.14}$$

We assume the load current is free of ripple and whatever ripple is present in the inductor current will be smoothened by the capacitor. Hence, we can write,

$$i_L = i_C + i_o \tag{2.15}$$

$$\Delta i_L = \Delta i_C \tag{2.16}$$

So that $\Delta i0 = 0$. Now, consider the waveform of inductor current i_L and i_C given in figure 2.3. We can see that both the waveform add up to make the steady state output current i_o to be constant. For upper half of the current ripple triangle, we can write the equation of the area of the triangle as $\frac{1}{2}\left(\frac{t_1}{2} + \frac{t_2}{2}\right)\frac{\Delta I}{2}$ which is nothing but equal to $\frac{\Delta I}{4}$ for a period of $\frac{t_1}{2} + \frac{t_2}{2} = \frac{T_s}{2}$ .Now for this period, the capacitor charges. So the charging of the capacitor, I_c is given by

$$I_C = \frac{\Delta I}{4} \tag{2.17}$$

Now the capacitor ripple voltage is given by,

$$v_C = \frac{1}{C_f}\int i_C dt + v_C(0^-) \tag{2.18}$$

$$\Delta v_C = \frac{1}{C_f}\int_{t=0}^{T_s/2} \Delta i_C + v_C(0^-) \qquad (2.19)$$

Or, we can write,

$$\Delta v_C = \frac{\Delta Q}{C_f} = \frac{1}{C_f}\frac{1}{2}\frac{T_s}{2}\frac{\Delta I}{2} = \frac{T_s \Delta I}{8C_f} \qquad (2.20)$$

Where we have considered the charging cycle of the capacitor with ΔQ as the charge accumulated during the period $\frac{t_1}{2} + \frac{t_2}{2} = \frac{T_s}{2}$ as explained above.

Now, putting the value of ΔI from the equation 2.13 and 2.14, correspondingly we get,

$$\Delta v_C = \frac{V_o(V_g - V_o)}{8L_f C_f V_g f_s^2} \qquad (2.21)$$

$$\Delta v_C = \frac{V_g k(1-k)}{8L_f C_f f_s^2} \qquad (2.22)$$

To find the critical values of the inductor and capacitor, the condition for continues inductor current and capacitor voltage is,

$$I_L = \frac{\Delta I}{2} \qquad (2.23)$$

$$V_o = \frac{\Delta v_C}{2} \qquad (2.24)$$

Substituting the value of I_L from equation 2.23 in equation 2.14, we get

$$\frac{V_g k(1-k)}{f_s L_f} = 2I_L = 2I_o = \frac{2kV_g}{R_L} \qquad (2.25)$$

$$\Rightarrow L_C = L_f \geq \frac{(1-k)R_L}{2f_s} \qquad (2.26)$$

Now, substituting the value of V_o from equation 2.24 in equation 2.22, we get

$$\frac{V_g k(1-k)}{8L_f C_f f_s^2} = 2V_o = 2kV_g \qquad (2.27)$$

$$\Rightarrow C_C = C_f \geq \frac{1-k}{16 L_f f_s^2} \qquad (2.28)$$

We can show the Continuous Conduction Mode Waveforms in the figure 2.3[11]

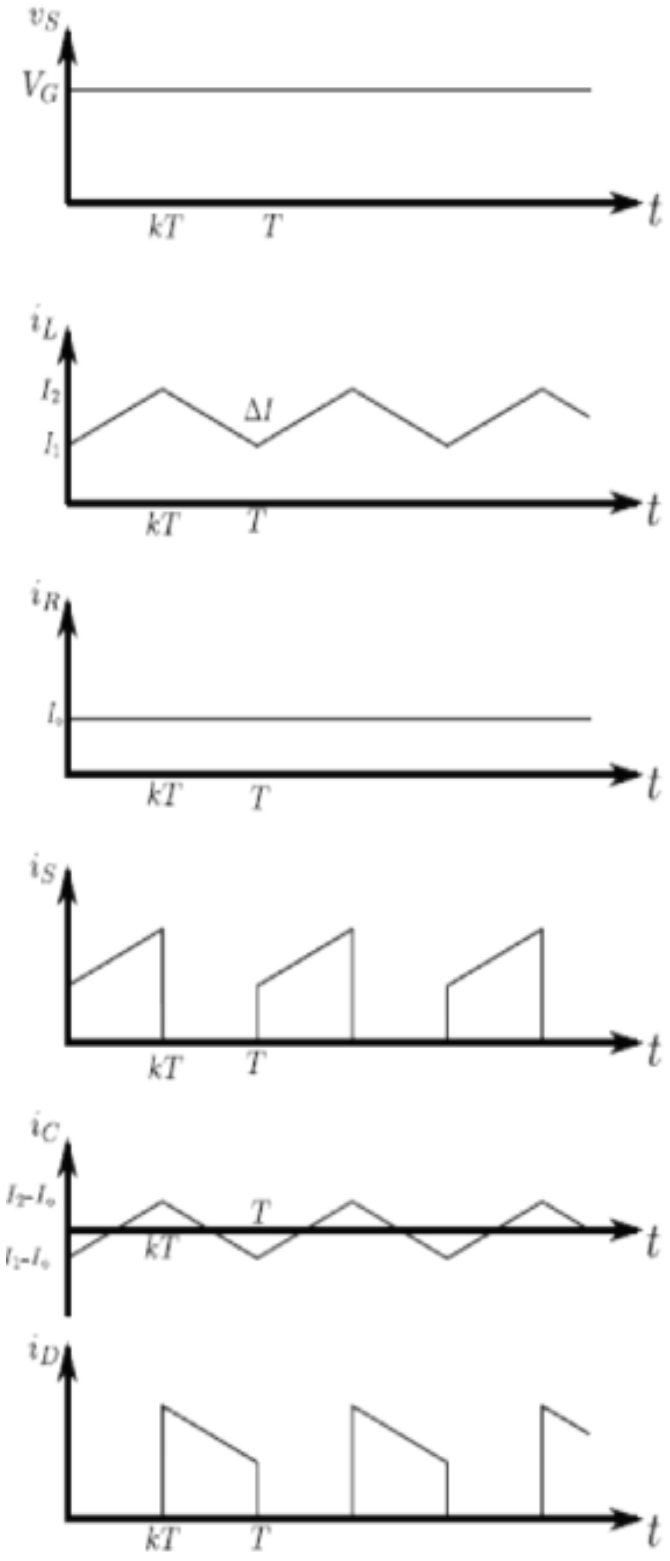


Figure 2.3: Continuous Conduction Mode Waveforms

CHAPTER 3 FUZZY CONTROL

3.1 Fuzzy Logic

Fuzzy logic is a superset of conventional (Boolean) logic that has been extended to handle the concept of partial truth- truth values between "completely true" and "completely false". As its name suggests, it is the logic underlying modes of reasoning which are approximate rather than exact. The importance of fuzzy logic derives from the fact that most modes of human reasoning and especially common sense reasoning are approximate in nature [15].

Fuzzy logic was developed by Lotfi A. Zadeh in the 1965s in order to provide mathematical rules and functions which permitted natural language queries. Fuzzy logic provides a means of calculating intermediate values between absolute true and absolute false with resulting values ranging between 0.0 and 1.0. Fuzzy logic calculates the shades of gray between black/white and true/false.

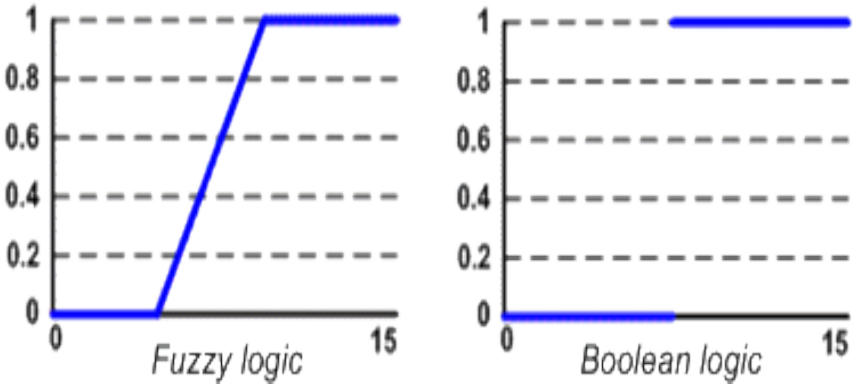


Figure 3.3: Fuzzy Logic vs. Boolean Logic

Fuzzy Logic deals with those imprecise conditions about which a true/false value cannot be determined. Much of this has to do with the vagueness and ambiguity that can be found in everyday life. For example, the question: *Is it HOT outside?* Probably would lead to a variety of responses from those asked. These are often labeled as subjective responses, where no one answers is exact. Subjective responses are relative to an individual's experience and knowledge. Human beings are able to exert this higher level of abstraction during the thought process. For this reason,

Fuzzy Logic has been compared to the human decision making process. Conventional Logic (and computing systems for that matter) is by nature related to the Boolean Conditions (true/false). What Fuzzy Logic attempts to encompass is that area where a partial truth can be established, that is a gradient within the true/false realm [15].

3.2 The Operations of Fuzzy Sets

In the classical sets there are basic operations that are found, such as intersection and union between sets, complement of set. If we have two sets (**A** and **B**) and these sets are subsets of universe (**U**).

3.2.1 Union

The membership function of the Union of two fuzzy sets A and B with membership functions μ_A and μ_B respectively is defined as the maximum of the two individual membership functions. This is called the *maximum* criterion.

$$\mu_{A\cup B} = max\,(\mu_A, \mu_B) \qquad (3.1)$$

The Union operation in Fuzzy set theory as shown in figure 3.2 is the equivalent of the **OR** operation in Boolean algebra.

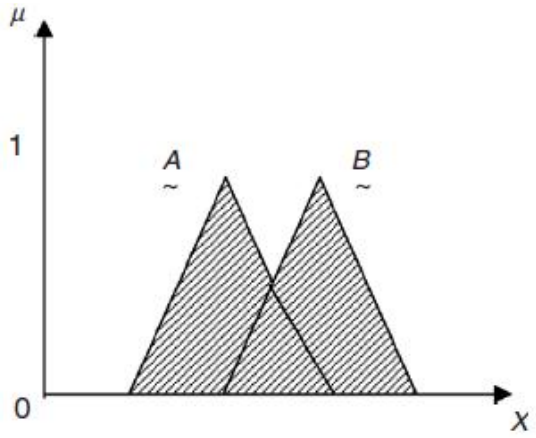


Figure 3.2: A OR B

3.2.2 Intersection

The membership function of the Intersection of two fuzzy sets A and B with membership functions μ_A and μ_B respectively is defined as the minimum of the two individual membership functions. This is called the *minimum* criterion.

$$\mu_{A\cap B} = min\,(\mu_A, \mu_B) \qquad (3.2)$$

The Intersection operation in Fuzzy set theory as shown in figure 3.3 is the equivalent of the **AND** operation in Boolean algebra.

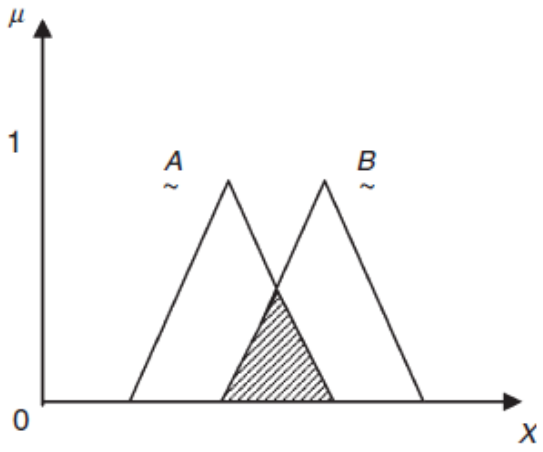


Figure 3.3: A AND B

3.2.3 Complement

The membership function of the Complement of a Fuzzy set A with membership function μ_A is defined as the negation of the specified membership function. This is called the *negation* criterion.

$$\mu_{\bar{A}} = 1 - \mu_A \qquad (3.3)$$

The Complement operation in Fuzzy set theory as shown in figure 3.4 is the equivalent of the **NOT** operation in Boolean algebra.

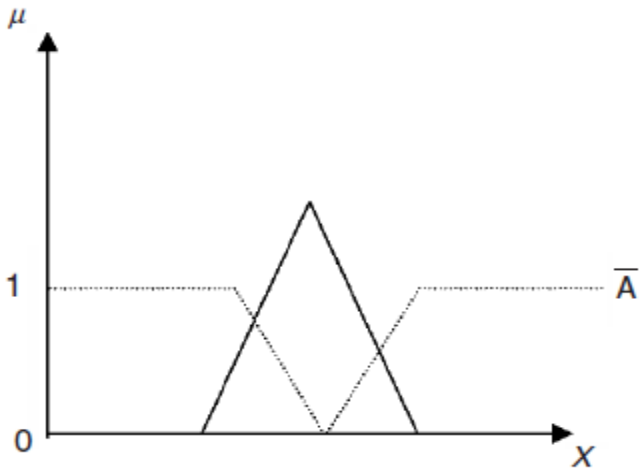


Figure 3.4: NOT A

Some properties of fuzzy set operations are given in Table 3.1[16] [17].

Table 3.1 Some Properties of Fuzzy Sets Operations

Law of contradiction	$A \cap \bar{A}=\emptyset$
Law of excluded middle	$A \cup \bar{A}= I$
De Morgan's laws	$\overline{(A \cap B)} = \bar{A} \cup \bar{B}$ $\overline{(A \cup B)} = \bar{A} \cap \bar{B}$
Involution (Double negation)	$\bar{\bar{A}}=A$
Commutative	$A \cap B = B \cap A$ $A \cup B = B \cup A$
Associative	$A \cap (B \cap C) = (A \cap B) \cap C$ $A \cup (B \cup C) = (A \cup B) \cup C$
Distributive	$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

3.3 Membership Function

Unlike the aforementioned conventional set, a fuzzy set (Zadeh, 1965) expresses the degree to which an element belongs to a set. Hence the characteristic function of a fuzzy set is allowed to have values between 0 and 1, which denotes the degree of membership of an element in a given set.

If X is a collection of objects denoted generically by x, then a "fuzzy set" A in X is defined as a set of ordered pairs [18]:

$$A = \{(x, \mu_A(x)) \mid x \in X\} \qquad (3.4)$$

where $\mu_A(x)$ is called "membership function" (or MF for short) for the fuzzy set A. The MF maps each element of X to a membership grade (or membership value) between 0 and 1 as shown in figure 3.5.

Obviously, the definition of a fuzzy set is a simple extension of the definition of a classical set in which the characteristic function is permitted to have any values between 0 and 1. If the values of the membership function are restricted to either 0 or 1, then A is reduced to a classical set and $\mu_A(x)$ is the characteristic function of A [18].

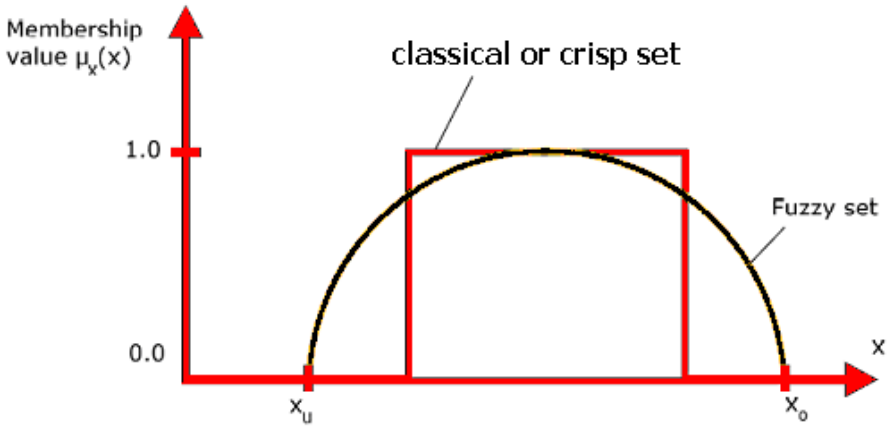


Figure 3.5: Membership Value in Fuzzy Set

3.3.1 Features of Membership Function

Features of Membership Function as shown in figure 3.6 divided to [18]:

1- Support

The "support" of a fuzzy set A is the set of all points x in X such that $\mu_A(x) > 0$:

$$support\ (A) = \{\, x \mid \mu_A(x) > 0 \,\} \qquad (3.5)$$

2- Core:

The "core" of a fuzzy set is the set of all points x in X such that $\mu_A(x) = 1$:

$$core\ (A) = \{\, x \mid \mu_A(x) = 1 \,\} \qquad (3.6)$$

3- Crossover points

A "crossover point" of a fuzzy set A is a point $x \in X$ at which $\mu_A(x) = 0.5$:

$$crossover\ (A) = \{\, x \mid \mu_A(x) = 0.5 \,\} \qquad (3.7)$$

4- Boundaries:

Comprise the elements x of the universe $0 < \mu_A(x) < 1$

5-α-cut:

The "α-cut" or "α-level set" of a fuzzy set A is a crisp set defined by

$$A\alpha = \{ x \mid \mu_A(x) \geq \alpha \} \qquad (3.8)$$

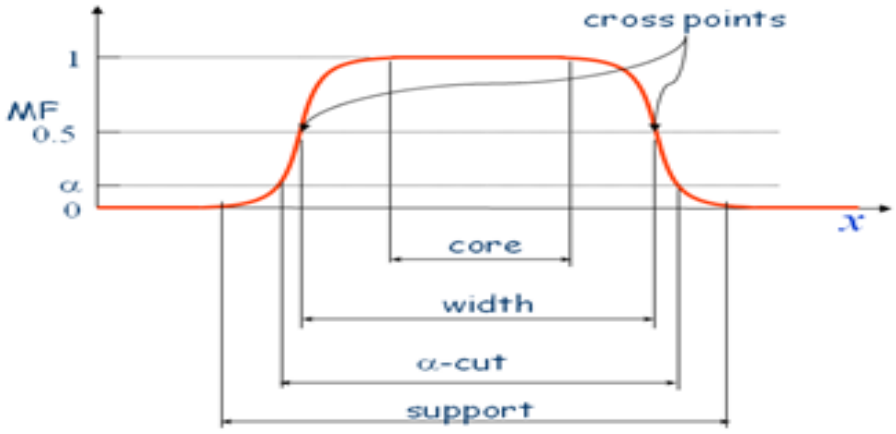


Figure 3.6: MF Terminology

3.3.2 Types of Membership Function

In the classical sets there is one type of membership function but in fuzzy sets there are many different types of membership function, now will show some of these types [18].

1- Triangular MFs

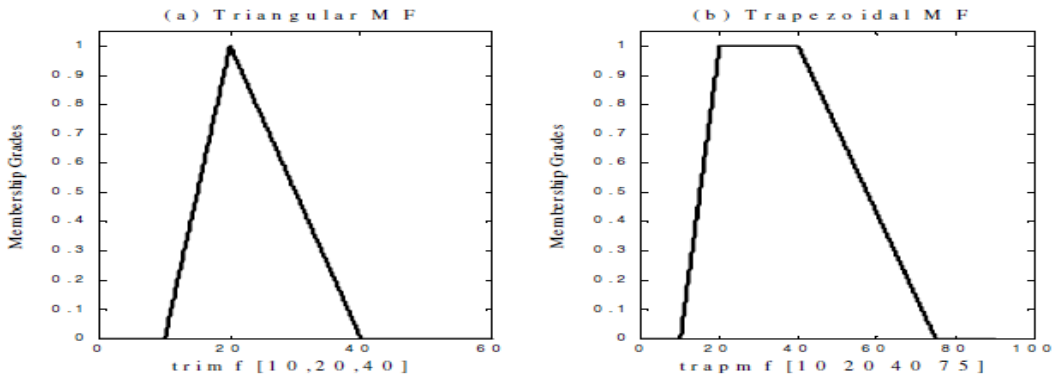
A "triangular MF" as shown in fig 3.7(a) is specified by three parameters {a, b, c} as follows:

$$y = triangle\,(x; a, b, c) = \begin{cases} 0, & x \leq a. \\ \dfrac{x-a}{b-a}, & a \leq x \leq b. \\ \dfrac{c-x}{c-b}, & b \leq x \leq c. \\ 0, & c \leq x. \end{cases} \qquad (3.9)$$

2- Trapezoidal MFs

A "trapezoidal MF" as shown in fig 3.7(b) is specified by four parameters {a, b, c, d} as follows:

$$\text{trapezoid}\,(x; a, b, c, d) = \begin{cases} 0, & x \leq a. \\ \dfrac{x-a}{b-a}, & a \leq x \leq b. \\ 1, & b \leq x \leq c. \\ \dfrac{d-x}{d-c}, & c \leq x \leq d. \\ 0, & d \leq x. \end{cases} \qquad (3.10)$$



(a)Triangular MF **(b)Trapezoidal MF**

Figure 3.7: Examples of Two Types of Parameterized MFs

3- Gaussian MFs

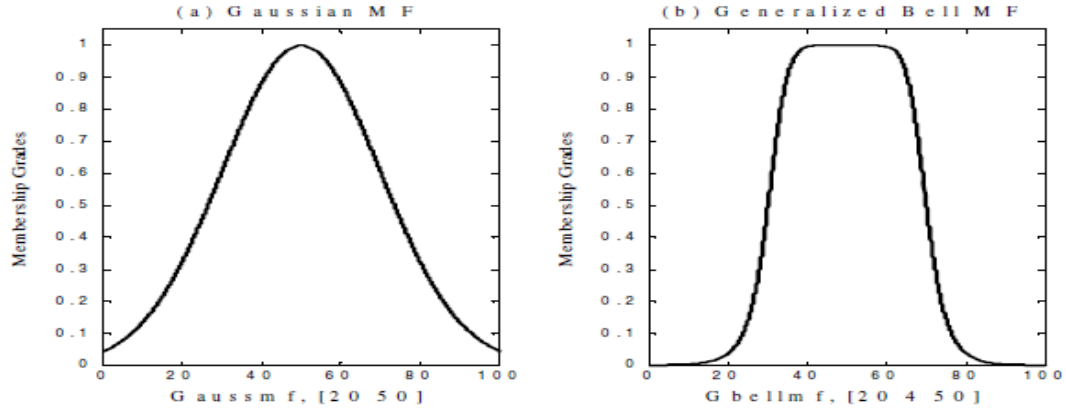
A "Gaussian MF" as shown in figure 3.8(a) is specified by two parameters $\{c, \sigma\}$:

$$\text{gaussian}\,(x;\ c\ ,\sigma) = e^{\frac{-1(x-c)}{\sigma}} \qquad (3.11)$$

4- Generalized bell MFs

A "generalized bell" as shown in fig 3.8(b) is specified by three parameters $\{a, b, c\}$:

$$bell(x; a, b, c) = \frac{1}{1 + 1(x-c)/a|^{2b}} \qquad (3.12)$$



(a)Gaussian **(b) Generalized Bell**

Figure 3.8: Examples of Two Classes of Parameterized Continuous MFs

5- Sigmoidal MFs

A "Sigmoidal MF" as shown in figure 3.9 is defined by the following equation:

$$sig(x;a,c) = \frac{1}{1 + exp\,[-a(x-c)]} \qquad (3.13)$$

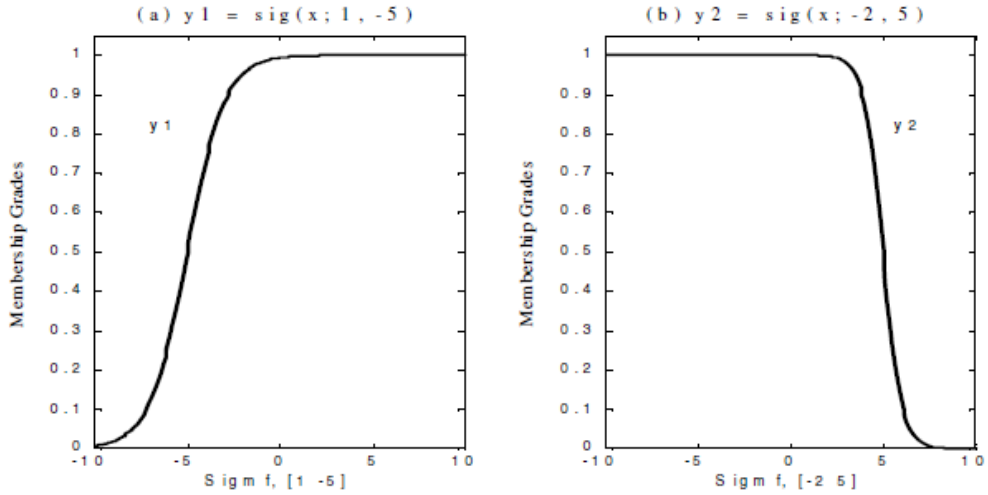


Figure 3.9: Two Sigmoidal Functions

3.4 General Structure of Fuzzy Logic Control "FLC" System

The basic parts of every fuzzy controller are displayed in figure 3.10. The fuzzy logic controller (FLC) is composed of a fuzzification interface, knowledge base, inference engine, and defuzzification interface.

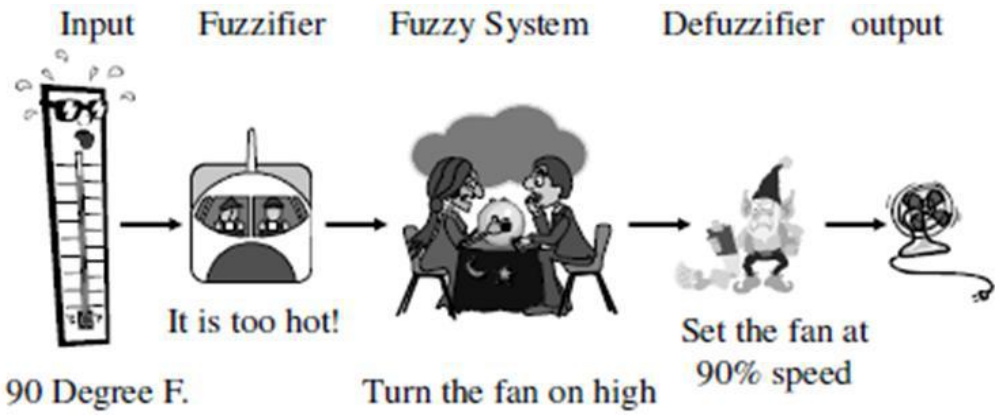


Figure 3.10: General Structure of Fuzzy Systems

3.4.1 Fuzzification

The first step in fuzzy logic processing the crisp inputs is transformed into fuzzy inputs as shown in figure 3.11. This transformation is called fuzzification. The system must turn numeric values into language and corresponding domains to allow the fuzzy inference engine to inference to transform crisp input into fuzzy input, membership functions must be first be defined for each input. Once membership functions are defined, fuzzification takes a real time input value, such as temperature, and compares it with the stored membership function information to produce fuzzy input values. Fuzzification plays an important role in dealing with uncertain information that might be objective in nature [19].

It converts the *crisp input* to a *linguistic variable* using the membership functions stored in the fuzzy knowledge base.

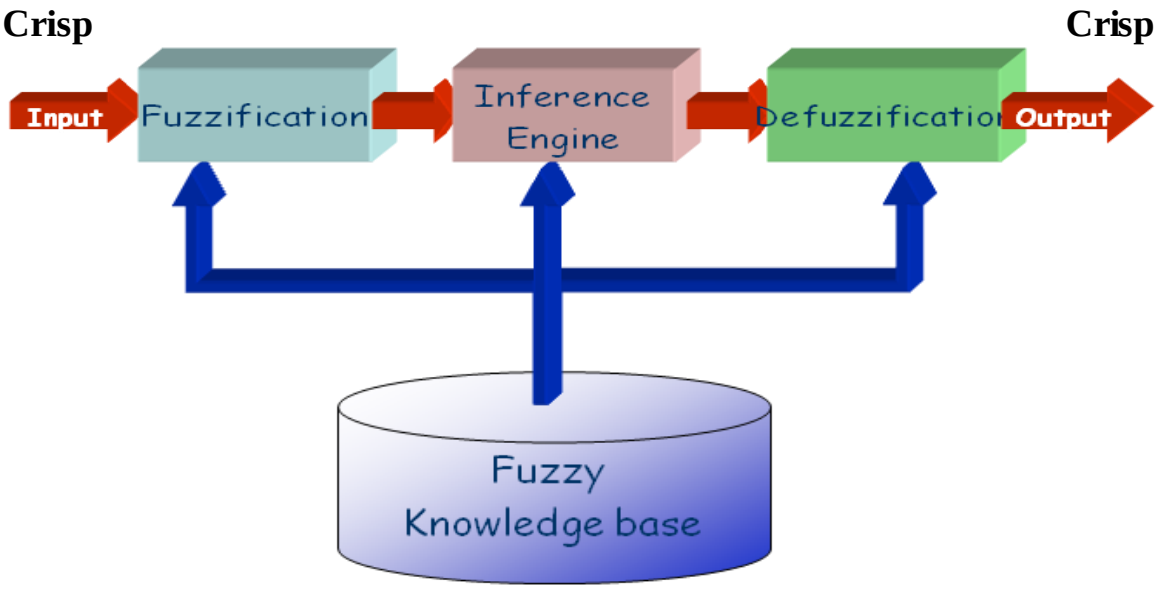


Figure 3.11: Basic Parts of Fuzzy Logic Controller

3.4.2. Knowledge Base

Knowledge base is the inference basis for fuzzy control. It defines all relevant language control rules and parameters. The knowledge base is the core of a fuzzy control system.

The knowledge base of Fuzzy Logic Controller (FLC) is comprised of two parts [20]:

1. Database.

2. Rule base

A linguistic controller contains rules in the (if-then) format. The Rule base is the cornerstone of the fuzzy model. The expert knowledge, which is assumed to be given as a number of if-then rules, is stored in a fuzzy rule base. The rules may use several variables in both the condition and the conclusion of the rules.

- **Linguistic Variables**

A "Linguistic variable" is characterized by a quintuple $(x, T(x), X, G, M)$ in which x is the name of the variable; $T(x)$ is the "term set" of x-that is, the set of its "linguistic values" or "linguistic terms"; X is the universe of discourse, G is a "syntactic rule" which generates the terms in $T(x)$; and M is a "semantic rule" which associates with each linguistic value A its meaning $M(A)$, where $M(A)$ denotes a fuzzy set in X[18].

- **Fuzzy Rules**

As was pointed out by Zadeh in his work on this area (Zadeh, 1973), conventional techniques for system analysis are intrinsically unsuited for dealing with humanistic systems, whose behavior is strongly influenced by human judgment, perception, and emotions. This is a manifestation of what might be called the "principle of incompatibility"[18]:

A "fuzzy if-then rule" (also known as “fuzzy rule”, “fuzzy implication” or "fuzzy conditional statement") assumes the form

$$\boldsymbol{If}\ x\ is\ A\ \boldsymbol{then}\ y\ is\ B \qquad (3.14)$$

where A and B are linguistic values defined by fuzzy sets on universes of discourse X and Y, respectively. Often "x is A" is called "antecedent" or "premise", while "y is B" is called the "consequence" or "conclusion".

Examples of fuzzy if-then rules are widespread in our daily linguistic expressions, such as the following:

- If pressure is high, then volume is small.
- If the road is slippery, then driving is dangerous.
- If the speed is high, then apply the brake a little.

3.4.3 Fuzzy Inference Systems

In this section we describe the type of fuzzy inference systems that have been widely used in the applications.

The "Mamdani fuzzy inference system" (Mamdani & Assilian, 1975) was proposed as the first attempt to control a steam engine and boiler combination by a set of linguistic control rules obtained from experienced human operators. Figure 3.12 is an illustration of how a two-rule Mamdani fuzzy inference system derives the overall output z when subjected to two numeric inputs x and y.

In Mamdani's application, two fuzzy inference systems were used as two controllers to generate the heat input to the boiler and throttle opening of the engine cylinder, respectively, to regulate the steam pressure in the boiler and the speed of the engine. Since the engine and boiler take only numeric values as inputs, a defuzzifier was used to convert a fuzzy set to a numeric value [18].

3.4.4 Defuzzification

Defuzzification refers to the way a numeric value is extracted from a fuzzy set as a representative value. In general, there are five methods for defuzzification. A brief explanation of each defuzzification strategy follows [18].

- **Centroid of area Z_{COA}:**

$$Z_{COA} = \frac{\int \mu_A(Z) Z dZ}{\mu_A(Z) dZ} \qquad (3.15)$$

where $\mu_A(z)$ is the aggregated output MF. This is the most widely adopted defuzzification strategy, which is reminiscent of the calculation of expected values of probability distributions. Various defuzzification methods are illustrated in figure 3.13.

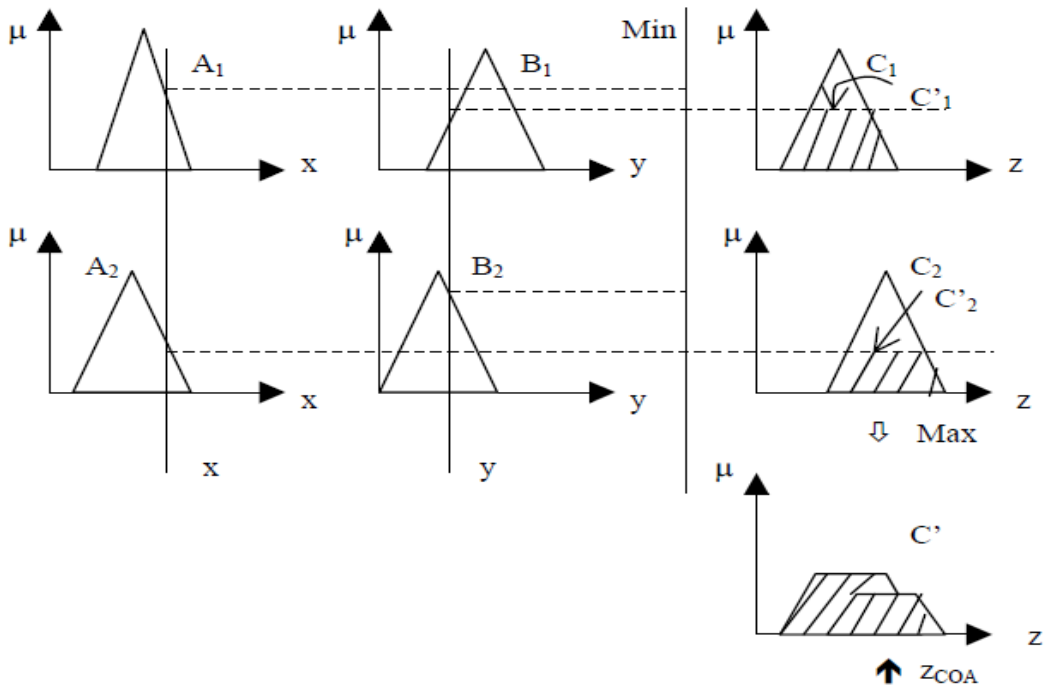


Figure 3.12: Mamdani Fuzzy Inference System using the Min and Max Operators

- **Bisector of area Z_{BOA} :**

$$\int_{\alpha}^{Z_{BOA}} \mu_A(Z)dZ = \int_{Z^{BOA}}^{\beta} \mu_A(Z)dZ \qquad (3.16)$$

where $\alpha = min\{z \mid z \in Z\}$ and $\beta = max\{z \mid z \in Z\}$.

- **Mean of maximum Z_{MOM}:** is the average of the maximizing z at which the MF reaches a maximum $\mu*$. Mathematically,

$$Z_{MOM} = \frac{\int ZdZ}{\int dZ} \qquad (3.17)$$

- **Smallest of maximum Z_{SOM}**: is the minimum (in terms of magnitude) of the maximizing z.
- **Largest of maximum Z_{LOM}**: is the maximum (in terms of magnitude) of the maximizing z.

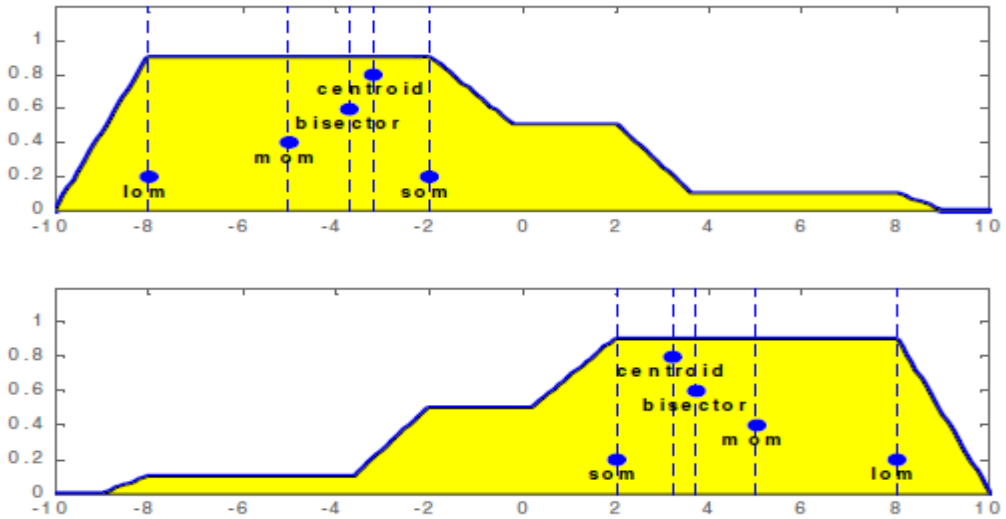


Figure 3.13: Various Defuzzification Methods for obtaining a Numeric Output

In this thesis the Mamdani model will be used and will have two inputs error and change of errors, and the output will be change of duty cycle. All inputs and output have seven membership functions. And the Z_{COA} defuzzification method will be used. The rule base will be as the next form:

IF error is zero AND change of error is zero THEN change of duty cycle is zero.

These rules will be written by the experience.

CHAPTER 4 GENETIC ALGORITHM

4.1 Introduction

Genetic Algorithm is reliable and robust method for searching solution spaces [21]. GA is general purpose search algorithm which uses principles inspired by neutral genetic to find solutions to problems [22][23]by using Survival of the fittest principle. The basic idea is to maintain a population of chromosomes, which represent candidate to the concrete problems that will be solve, through a process of computation and controlled variation. Each structure of chromosome in the population represent one of the possible solution of the problem and the fitness test of these chromosomes can determine which chromosomes are used to form new chromosomes that will be used in computational process. As in natural the new chromosomes are created by some operations such as crossover and mutations. There is another operation which called reproduction. This operation is added to achieve the survival of the fittest principle. In recent years, GA is used in many applications specially in optimization and search problems and had a great measure of success; the main reason of this success that it can start from any solutions, and generate other solutions that converge to the optimal solution in less time versus other classical search tools (enumerative, heuristic). "GA are theoretically and empirically proven to provide a robust search in complex spaces, thereby offering a valid approach to problems requiring efficient and effective searches"[24].

4.2 Basic Model of a Genetic Algorithm

The basic genetic algorithm is as follows [25]:

- [Start] Genetic random population of n chromosomes (suitable solutions for the problem)
- [Fitness] Evaluate the fitness $f(x)$ of each chromosome x in the population
- [New population] Create a new population by repeating following steps until the new population is complete
  - ❖ [Selection] select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to get selected).

- [crossover] With a crossover probability, cross over the parents to form new offspring (children). If no crossover was performed, offspring is the exact copy of parents.
- [Mutation] With a mutation probability, mutate new offspring at each locus(position in chromosome)
- [Accepting] Place new offspring in the new population.
- [Replace] Use new generated population for a further sum of the algorithm.
- [Test] If the end condition is satisfied, stop, and return the best solution in current population.
- [Loop] Go to step2 for fitness evaluation.

Figure 4.1 shows a basic model of a genetic algorithm, the algorithm is as follows.

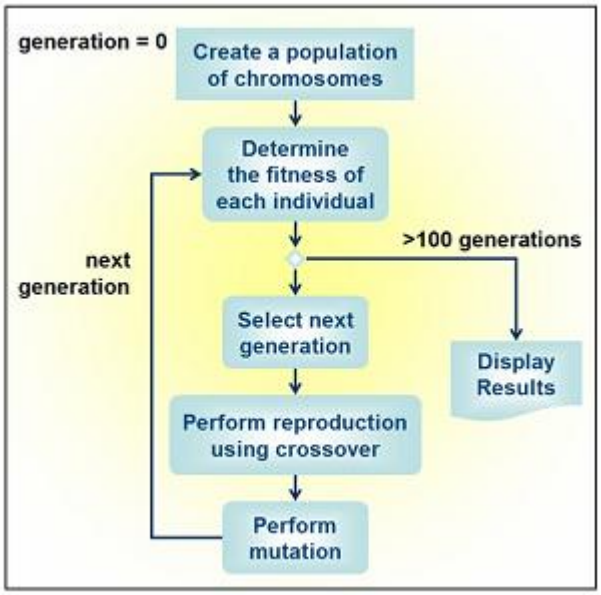


Figure 4.1: The Basic Genetic Algorithm

The human designer who want to solve optimization problem using GA must address five issues[24].

1- A genetic representation of candidate solutions,
2- A way to create an initial population of solutions,
3- An evaluation function which describes the quality of each individual,
4- Genetic operators that generate new variants during reproduction, and
5- Values for the parameters of the GA, such as population size, number of generations and probabilities of applying genetic operators.

4.3 Genetic Algorithm vs. Other Optimization Techniques

The principle of GAs is simple: imitate genetics and natural selection by a computer program [25]: The parameters of the problem are coded most naturally as a DNA-like linear data structure, a vector or a string. Sometimes, when the problem is naturally two or three-dimensional also corresponding array structures are used. A set, called population, of these problem dependent parameter value vectors is processed by GA. To start there is usually a totally random population, the values of different parameters generated by a random number generator. Typical population size is from few dozens to thousands. To do optimization we need a cost function or fitness function as it is usually called when genetic algorithms are used. By a fitness function we can select the best solution candidates from the population and delete the not so good specimens.

The nice thing when comparing GAs to other optimization methods is that the fitness function can be nearly anything that can be evaluated by a computer or even something that cannot! In the latter case it might be a human judgment that cannot be stated as a crisp program, like in the case of eyewitness, where a human being selects among the alternatives generated by GA. So, there are not any definite mathematical restrictions on the properties of the fitness function. It may be discrete, multimodal etc.

The main criteria used to classify optimization algorithms are as follows: continuous /discrete, constrained / unconstrained and sequential / parallel. There isa clear difference between discrete and continuous problems. Therefore it is instructive to notice that continuous methods are sometimes used to solve inherently discrete problems and vice versa. Parallel algorithms are usually used to speed up processing. There are, however, some cases in which it is more efficient to run several processors in parallel rather than sequentially. These cases include among others such, in which there is high probability of each individual search run to get stuck into a local extreme. Irrespective of the above classification, optimization methods can be further classified into deterministic and non-deterministic methods. In addition optimization algorithms can be classified as local or global. In terms of energy and entropy local search corresponds to entropy while global optimization depends essentially on the fitness i.e. energy landscape.

Genetic algorithm differs from conventional optimization techniques in following ways [25]:

1- GAs operate with coded versions of the problem parameters rather than parameters themselves i.e., GA works with the coding of solution set and not with the solution itself.
2- Almost all conventional optimization techniques search from a single point but GAs always operate on a whole population of points(strings) i.e., GA uses population of solutions rather than a single solution for searching. This plays a major role to the robustness of genetic algorithms. It improves the chance of reaching the global optimum and also helps in avoiding local stationary point.
3- GA uses fitness function for evaluation rather than derivatives. As a result, they can be applied to any kind of continuous or discrete optimization problem. The key point to be performed here is to identify and specify a meaningful decoding function.
4- GAs use probabilistic transition operates while conventional methods for continuous optimization apply deterministic transition operates i.e., GAs does not use deterministic rules.

4.4 Applications of Genetic Algorithm

Genetic algorithms have been used for difficult problems, for machine learning and also for evolving simple programs. They have been also used for some art, for evolving pictures and music. A few applications of GA are as follow [25]:

- Nonlinear dynamical systems-predicting, data analysis
- Robot trajectory planning
- Evolving LISP programs (genetic programming)
- Strategy planning
- Finding shape of protein molecules
- TSP and sequence scheduling
- Control-gas pipeline, pole balancing, missile evasion, pursuit
- Design-semiconductor layout, aircraft design, keyboard configuration, communication networks
- Scheduling-manufacturing, facility scheduling, resource allocation

- Machine Learning-Designing neural networks, both architecture and weights, improving classification algorithms, classifier systems
- Signal Processing-filter design
- Combinatorial Optimization-set covering, traveling salesman (TSP), Sequence scheduling, routing, bin packing, graph coloring and partitioning

4.5 GA Operations

The process of genetic algorithm is used to solve the structure of chromosome to represent the possible combinations of answers, according to population size; the population size can be fixed or random according to the optimization problem [25]. The chromosome structure represents all the variables that the designer wants to find the optimal values of it. Every variable represents gene, and the main structure of the genes is 0 and 1 combinations. Real numbers can also be used to indicate the number of floating. The chromosomes of each string represents a separate individual, and each individual represent one solution of the problem, so each population contain series of individual and the decision of each individual will have a so-called fitness value. This value is calculated by fitness function; the individual which have higher fitness value has higher probability to appear in the next generation more than the lower fitness value. The new population is generated after some processes, such selection, replication (reproduction), mating (crossover), Mutation and other evolutionary mechanisms. The process of evolution over generations eventually converges to the optimal solution.

4.6 GA Elements

The two distinct elements in the GA are individuals and populations. An individual is a single solution while the population is the set of individuals currently involved in the search process.

4.6.1 Individuals

An individual is a single solution. Individual groups together two forms of solutions as given below [25]:

1- The chromosome, which is the raw ‘genetic’ information (genotype) that the GA deal

2- The phenotype, which is the expressive of the chromosome in the terms of the model.

A chromosome is subdivided into genes. A gene is the GA's representation of a single factor for a control factor. Each factor in the solution set corresponds to gene in the chromosome. Figure 4.2 shows the representations of a genotype. Chromosomes are encoded by bit strings are given below in figure. 4.3.

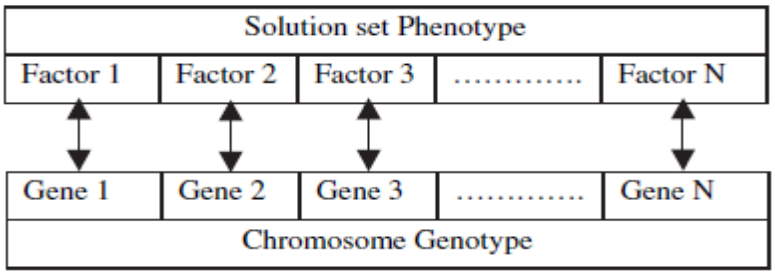


Figure 4.2: Representation of Genotype and Phenotype

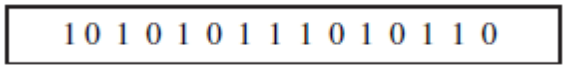


Figure 4.3: Representation of Chromosome

Genes are the basic "instructions" for building a Generic Algorithms. A chromosome is a sequence of genes. Genes may describe a possible solution to a problem, without actually being the solution. A gene as we show in figure 4.4 is a bit string of arbitrary lengths. The bit string is a binary representation of number of intervals from a lower bound a bit string of length 'n' can represent (2n-1) intervals. The size of the interval would be (range)/ (2n-1).

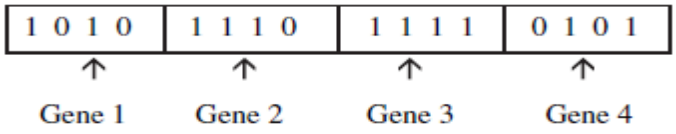


Figure 4.4: Representation of a Gene

4.6.2 Population

A population is a collection of individuals as shown in figure 4.5. A population consists of a number of individuals being tested, the phenotype parameters defining the individuals and some information about search space. Two important aspects of population used in Genetic Algorithms are:

1- The initial population generation.

2- The population size.

Population	Chromosome 1	1 1 1 0 0 0 1 0
	Chromosome 2	0 1 1 1 1 0 1 1
	Chromosome 3	1 0 1 0 1 0 1 0
	Chromosome 4	1 1 0 0 1 1 0 0

Figure 4.5: Population Representation

Initial population often consists of random individuals but in some cases the designer suggests some solutions to be in the population. The size of it depend on the complexly of the problem. In the ideal case the first population must have large number of individuals to cover all the range of solution space. All possible alleles of each should be present in the population. Sometimes some of the solutions expected can be used to seed the initial population. Thus, the fitness of these individuals will be high which helps the GA to find the solution faster. The population size can cause some problems. The large population is useful to find the best solution but it was established that the time required by a GA to converge is (N x log N) where N is population size [25]

4.7 Chromosome Coding

Encoding is a process of representing individual genes. The process can be performed using bits, numbers, trees, arrays, lists or any other objects. Encoding depends mainly on solving the problem. For example, one can encode directly real or integer numbers. Figures 4.6, 4.7, 4.8, 4.9 and 4.10 show the kinds of coding.

1- Binary Encoding:

The most common way of encoding is a binary string, which would be represented as in figure. 4.6. Binary encoding gives many possible chromosomes with a smaller number of alleles. On the other hand this encoding is not natural for many problems and sometimes corrections must be made after genetic operation is completed. Binary coded strings with 1s and 0s are mostly used. The length of the string depends on the accuracy. In this,

- Integers are represented exactly
- Finite number of real numbers can be represented
- Number of real numbers represented increases with string length

Chromosome 1	1 1 0 1 0 0 0 1 1 0 1 0
Chromosome 2	0 1 1 1 1 1 1 1 1 1 0 0

Figure 4.6: Binary Encoding

2- Real Encoding:

Every chromosome is a string of numbers, which represents the number in sequence. Sometimes corrections have to be done after genetic operation is completed. In permutation encoding, every chromosome is a string of integer/real values, which represents number in a sequence.

Chromosome A	1 5 3 2 6 4 7 9 8
Chromosome B	8 5 6 7 2 3 1 4 9

Figure 4.7: Real Encoding

3- Octal Encoding:

This encoding uses string made up of octal numbers (0-7).

Chromosome 1	03467216
Chromosome 2	15723314

Figure 4.8: Octal Encoding

4- Hexadecimal Encoding

This encoding uses string made up of hexadecimal numbers (0-9, A-F).

Chromosome 1	9CE7
Chromosome 2	3DBA

Figure 4.9: Hexadecimal Encoding

5- Value Encoding

Every chromosome is a string of values and the values can be anything connected to the problem. This encoding produces best results for some special problems

Chromosome A	1.2324 5.3243 0.4556 2.3293 2.4545
Chromosome B	ABDJEIFJDHDIERJFDLDFLFEGT
Chromosome C	(back), (back), (right), (forward), (left)

Figure 4.10: Value Encoding

6- Tree Encoding

This encoding is mainly used for evolving program expressions for genetic programming. Every chromosome is a tree of some objects such as functions and commands of a programming language.

4.8 Fitness Function.

The fitness of an individual in a genetic algorithm is the value of an objective function for its phenotype. And it is used to evaluate how good the different individuals in the population are. The fitness function depends on the problem that will be solved, for example in Traveling Sales Man (TSM) problem may be the time that the sales man will take it along traveling. So the fitness value can be defined as function of the objective function $g(x)$.

$$Fitness.value = f\big(g(x)\big) \qquad (4.1)$$

For calculating fitness, the chromosome has to be first decoded and the objective function has to be evaluated. The fitness not only indicates how good the solution is, but also corresponds to how close the chromosome is to the optimal one [25]. When the optimization problem is single criterion, it is simple because there is one goal to achieve, when the problem is multi criterion the optimization problem will be more complex because if the solution is optimal for one criterion it may be worst for another one. The most difficult fitness functions are the ones needed to evaluate non-numerical data [26], as the developer must find other metrics or ways to find a numerical evaluation of non-numerical data. An example of this is provided by Mitchell [27], who describes the problem of finding the optimal sequence of amino acids that can be folded to a desired protein structure. The acids are represented by the alphabet $\{A \dots Z\}$, and thus no numerical value can be straightforwardly calculated. The used fitness function calculates the energy needed to bend the given sequence of amino acids to the desired protein. In control applications there are different fitness function that may be used [28].

$$fitness.value = \int_0^\infty e^2(t)dt \quad sum\ of\ squared\ error \tag{4.2}$$

Where (e) is the error signal, this function can track error quickly, but easily gives rise to oscillation.

$$fitness.value = \int_0^\infty |e\ (t)|\ dt \quad sum\ of\ absolute\ error \tag{4.3}$$

This function can obtain good response, but its selection performance is not good.

$$fitness.value = \int_0^\infty t\ e^2\ (t)dt \quad sum\ of\ time\ weighted\ squared\ error \tag{4.4}$$

This function can gives fast tracking and good response.

4.9 Selection

Selection or reproduction is the first process after finding the fitness values of the solutions. In this process the developer will choose the pairs of parents that will be crossed. This step is to decide how to perform selection. On other words, who are the individuals of that population will be used to create the next offspring that will be

used for next generation. The purpose of this step is to stress the individuals of the population that will be selected with the higher fitness. The problem is how to select the chromosomes that will cross; there are many methods that can be used [25].

4.9.1 Roulette Wheel Selection.

Figure 4.11 shows the Roulette selection method which is one of the traditional GA selection techniques. That is called because this method works in a way that is similar to a roulette wheel. Each individual in a population is allocated a share of a wheel; the size of the share depends on the individuals fitness. The individuals that have higher fitness have big share. The individuals that have lower fitness have small share. That means the lower fitness of the individuals may have no chance to be in the roulette. A pointer is spun (a random number generated) and the individual to which it points is selected. This continues until the requisite number of individuals has been selected. The Roulette wheel will have a problem when the fitness values differ very much. If the best chromosome fitness is 90%, its circumference occupies 90% of Roulette wheel, and then other chromosomes have too few chances to be selected.

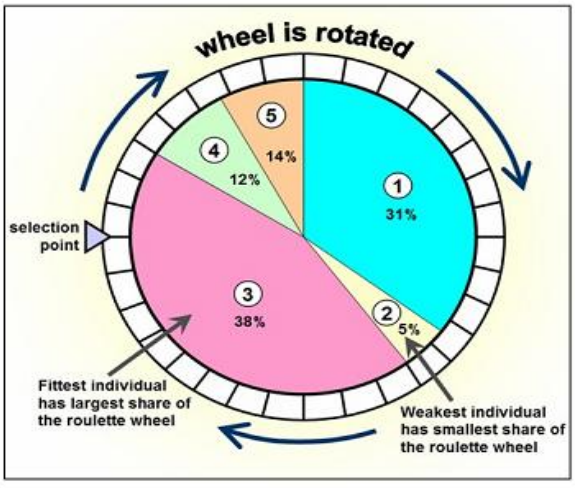


Figure 4.11: Roulette Wheel Selection.

4.9.2 Rank Selection

Rank Selection ranks the population and every chromosome receives fitness from the ranking [25]. It results in slow convergence It also keeps up

selection pressure when the fitness variance is low. Here, rank selection is programmed as follow:

1. Select first pair at random.
2. Generate random number R between 0 and 1.
3. If $R < r$ use the first individual as a parent. If the $R >= r$ then use the second individual as the parent
4. Repeat to select the second parent.

4.9.3 Stochastic Universal Sampling

Figure 4.12 shows Stochastic Universal Sampling method in this method the individuals represent a line that is divided into number of adjacent segments, such that each individuals segment is equal in size to its fitness exactly as in roulette-wheel selection. Then, create equally space pointers that are placed over the line. The numbers of these pointers (N Pointer) depends on the number of the individuals that will be selected; the distance between the pointers is given as 1/NP, and the position of the first pointer is given by a randomly generated number in the range [0, 1/N Pointer].

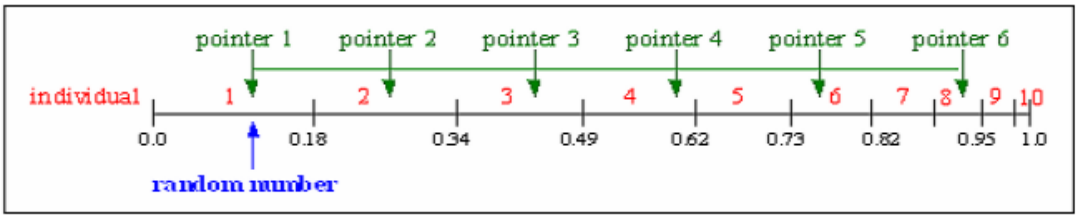


Figure 4.12: Stochastic Universal Sampling

4.10 Crossover

Crossover is the process of taking two parent solutions and producing from them child. After the selection (reproduction) process, the population is enriched with better individuals. Reproduction makes clones of good strings but does not create new ones. Crossover operator is applied to the mating pool with the hope that it creates a better offspring. Crossover is a recombination operator that proceeds in three steps [25]:

I. The reproduction operator selects at random a pair of two individual strings for the mating.
II. A cross site is selected at random along the string length.
III. Finally, the position values are swapped between the two strings following the cross site.

4.10.1 Single-Point Crossover

The traditional genetic algorithm uses single point crossover, where the two mating chromosomes are cut once at corresponding points and the sections after the cuts exchanged. Here, a cross-site or crossover point is selected randomly along the length of the mated strings and bits next to the cross-sites are exchanged. If appropriate site is chosen, better children can be obtained by combining good parents else it severely hampers string quality.

Figure 4.13 illustrates single point crossover and it can be observed that the bits next to the crossover point are exchanged to produce children. The crossover point can be chosen randomly.

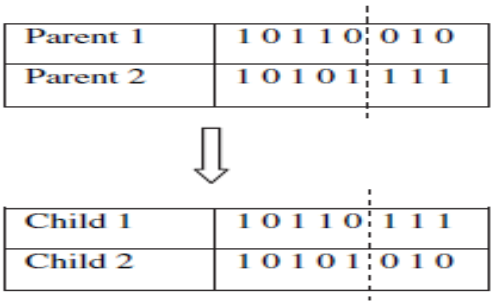


Figure 4.13: Single Point Crossover

4.10.2 Two-Point Crossover

In two-point crossover, two crossover points are chosen and the contents between these points are exchanged between two mated parents. These points are exchanged between two mated parents as shown in figure 4.14.

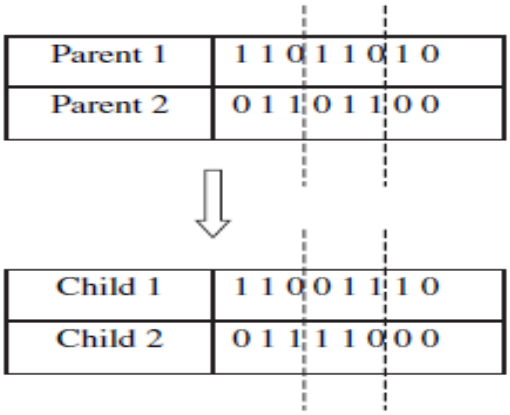


Figure 4.14: Two-Point Crossover

4.10.3 Multi-Point Crossover (N-Point crossover)

There are two ways in this crossover. One is even number of cross-sites and the other odd number of cross-sites. In the case of even number of cross-sites, cross-sites are selected randomly around a circle and information is exchanged. In the case of odd number of cross-sites, a different cross-point is always assumed at the string beginning.

4.10.4 Uniform Crossover

Uniform crossover is another crossover technique. In this technique, the random mask is used, and this mask has same length as the chromosome. This mask consists of 1s and 0s. If a bit in the mask is 1 then the corresponding bit in the first child will come from the first parent and the second parent will contribute that bit to the second offspring. If the mask bit is 0, the first parent contributes to the second child and the second parent to the first child as shown in figure 4.15.

Parent 1	1 0 1 1 0 0 1 1
Parent 2	0 0 0 1 1 0 1 0
Mask	1 1 0 1 0 1 1 0
Child 1	1 0 0 1 1 0 1 0
Child 2	0 0 1 1 0 0 1 1

Figure 4.15: Uniform Crossover

4.10.5 Three Parent Crossover

In this crossover technique, three parents are randomly chosen. Each bit of the first parent is compared with the bit of the second parent. If both are the same, the bit is taken for the offspring otherwise; the bit from the third parent is taken for the offspring. This concept is illustrated in figure 4.16.

Parent 1	1 1 0 1 0 0 0 1
Parent 2	0 1 1 0 1 0 0 1
Parent 3	0 1 1 0 1 1 0 0
Child	0 1 1 0 1 0 0 1

Figure 4.16: Three Parent Crossover

4.11 Mutation

Mutation means swap one bit in binary coding or changes one number if the chromosome consists of numbers [29]. For binary coding, for doing this process there are one way; first choosing random number at every individual if the number less than specified number which is chose before by the programmer, then this individual will have mutation process. But how many bits will be changed there are many ways. First choosing random number between1and the total numbers of chromosome length and swap the bits which meet that number. Second method is choosing random number between 0 and 1 at every bit of the chromosome if the number less than specified number then this bit will be swapped i.e., if it is a 1 change it to 0 or vice versa. This mutation probability is generally kept quite low and is constant throughout the lifetime of the GA. However, a variation on this basic algorithm changes the mutation probability throughout the lifetime of the algorithm, starting with a relatively high rate and steadily decreasing it as the GA progresses. This allows the GA to search more for potential solutions at the outset and to settle down more as it approaches convergence.

4.12 Elitism

The first best chromosome or the few best chromosomes are copied to the new population. The rest is done in a classical way [25]. Such individuals can be lost if

they are not selected to reproduce or if crossover or mutation destroys them. This significantly improves the GA's performance. Elitism can be used to eliminate the chance of any undesired loss of information during the mutation stage. Moreover, the execution time is less.

4.13 Genetic Fuzzy Systems

In a very broad sense, a Fuzzy System (FS) is any fuzzy logic-based system where fuzzy logic can be used either as the basis for the representation of different forms of system knowledge or to model the interactions and relationships among the system variables. FSs proven to be an important tool for modeling complex systems in which, due to complexity or imprecision, classical tools are unsuccessful. Genetic algorithms are search algorithms that use operations found in natural genetics to guide the trek through a search space. GA is theoretically and empirically proven to provide robust search capabilities in complex spaces, offering a valid approach to problems requiring efficient and effective searching. The following components of the knowledge base (KB) are potential candidates for optimization [24].

1- Data base (DB) components: scaling functions and membership function parameters.

2- Rule base (RB) components: "IF-THEN" rule consequents.

4.13.1 Genetic Tuning of the Data Base

The tuning of the scaling gains and fuzzy membership functions is an important task in FLC design. Scaling gains applied to the inputs and outputs of an FLC. Because the most FLC is normalized, the universes of discourse in which the fuzzy membership functions are defined all inputs and outputs are in the range [-1 1]. The individual refers to scaling gain and by using fitness function can calculate the best individual which gives the best scaling function. In the case of membership function, the parameters of the membership are tuned; Triangular membership functions are usually encoded by the left, right, and center of the membership

4.13.2 Genetic Learning of the Rule Base

In this way the membership of the fuzzy inputs and outputs and the scaling gain do not changed, but the sequence of IF-THEN rules will be modified to gives the

best result. In this way the individual is represented the one rule or the all rules. The RB is represented by a relational matrix, a decision table or a list of rules. In this thesis the fuzzy membership inputs and output membership functions will be used as variables that will be optimized using GA. Every triangular membership has three variable can effect on its shape; so each chromosome will has the number of genes that effect on the membership shape. For example every triangular membership can represented be three genes because it has three parameters that control of its shape (center edge, left edge and right edge). If the inputs of fuzzy controller have seven triangular memberships, then every chromosome of the population will have twenty one genes (7x3) (in this thesis there are ten variable for membership and three variables for gains). While the fitness functions will be the integral of the absolute value of error.

CHAPTER 5 DESIGNING FUZZY CONTROLLER FOR BUCK CONVERTER

5.1. Overall System Design

This chapter presents the process used to design fuzzy controllers for the DC-DC Buck converter using Try and error method and using Genetic Algorithm to improve the settling time, steady state error and ripple of the output voltage of the system.

5.2DC-DC Buck Converter

The operation of converter is fairly simple with an inductor, two switches and a capacitor that control the current of the inductor. Firstly, connecting the inductor to source voltage to store energy in the inductor, and then discharging the inductor into the load as shown in figure 5.1.

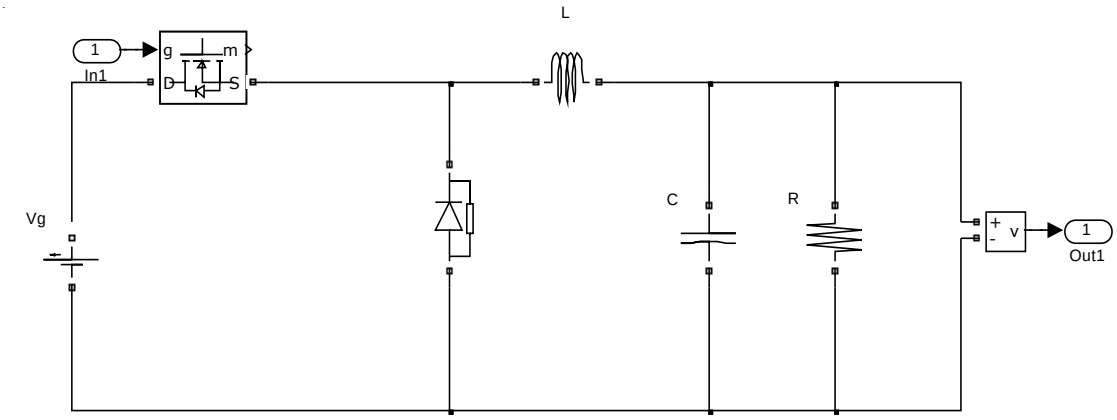


Figure 5.1: Buck Converter

The simulation DC-DC converted to one subsystem block using Matlab\ simulink program as shown in figure 5.2.

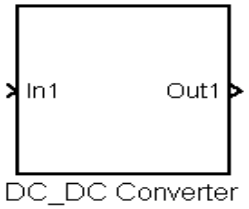


Figure 5.2: DC-DC Converter Subsystem

5.2.1 Pulse Width Modulation (PWM)

PWM signals are pulse trains with fixed frequency and magnitude and variable pulse width. However, the width of the pulses (duty cycle) changes from pulse to pulse according to a modulating signal as illustrated in figure 5.3. When a PWM signal is applied to the gate of the power transistor, it causes turn on and off intervals of the transistor to change from one PWM period to another according to the same modulating signal [19].

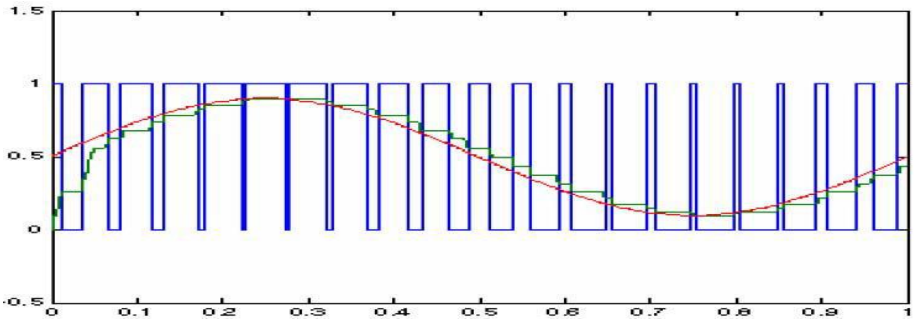
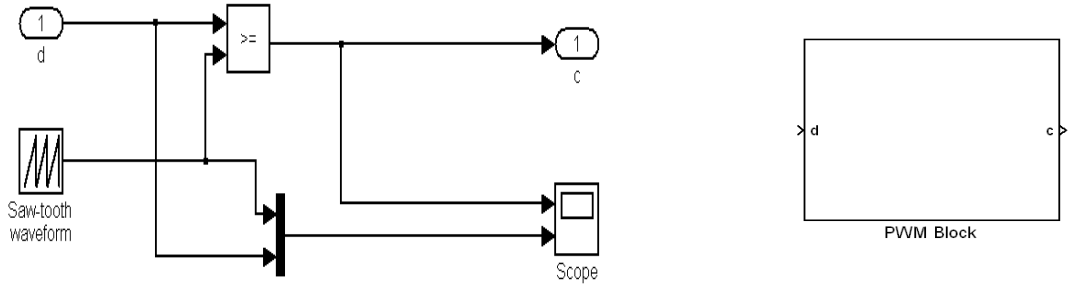


Figure 5.3: Pulse Width Modulation Waveforms

The simulation PWM converted to one subsystem block with input from FLC while the output goes to the DC-DC converter Block as shown in figure 5.4-b.



(a) Block **(b)Subsystem**

Figure 5.4: PWM Simulation using Matlab

5.3 Fuzzy Logic Controller.

Unlike Boolean logic which states that the value of any variable is either 0 or 1, fuzzy logic allows states between them; i.e. membership value. The grade of membership value of fuzzy variable can be described in linguistic term [9]. Fuzzy major features are the use of linguistic variables rather than numerical variables.

Linguistic variables are defined as the variables whose values are defined in a usual language (e.g. small and big), may be represented by fuzzy set.

The general structure of FLC controller as shown in chapter 3 is clear in figure 5.5. It comprises four principal components:

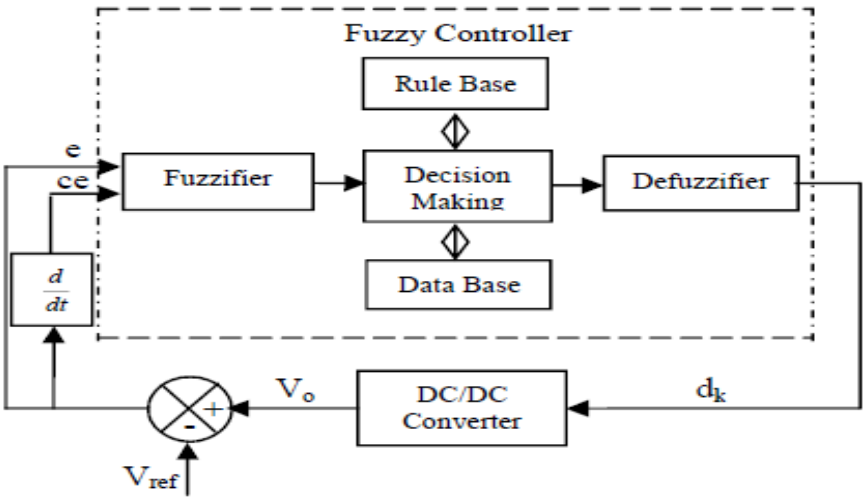


Figure 5.5: Basic Configuration of FLC

1- Fuzzifier

A fuzzyfication interface converts input data into suitable linguistic values.

2- Rule Base and Data Base

Both are known as knowledge base. It consists of data base with necessary linguistic definition and control rule set.

3- Decision Making

Decision making logic simulates a human decision process, infers the fuzzy control action from the knowledge of the control rules and the linguistic variable definition.

4- Defuzzifier

A defuzzification interface yields a non-fuzzy control action from an inferred fuzzy control action. The inputs of the fuzzy logic controller as shown in figure

5.6 are the error e and the change of error ce, which are defined in equation 5.1 and 5.2 as:

$$e = V_o - V_{ref} \qquad (5.1)$$

$$ce = e_k - e_{k-1} \qquad (5.2)$$

where V_o is the present output voltage, V_{ref} is the reference input voltage, and subscript k denotes values taken at the beginning of the k^{th} switching cycle. The output of the fuzzy controller is the duty cycle and is defined in equation 5.3 as:

$$d_k = d_{k-1} + h\Delta d_k \qquad (5.3)$$

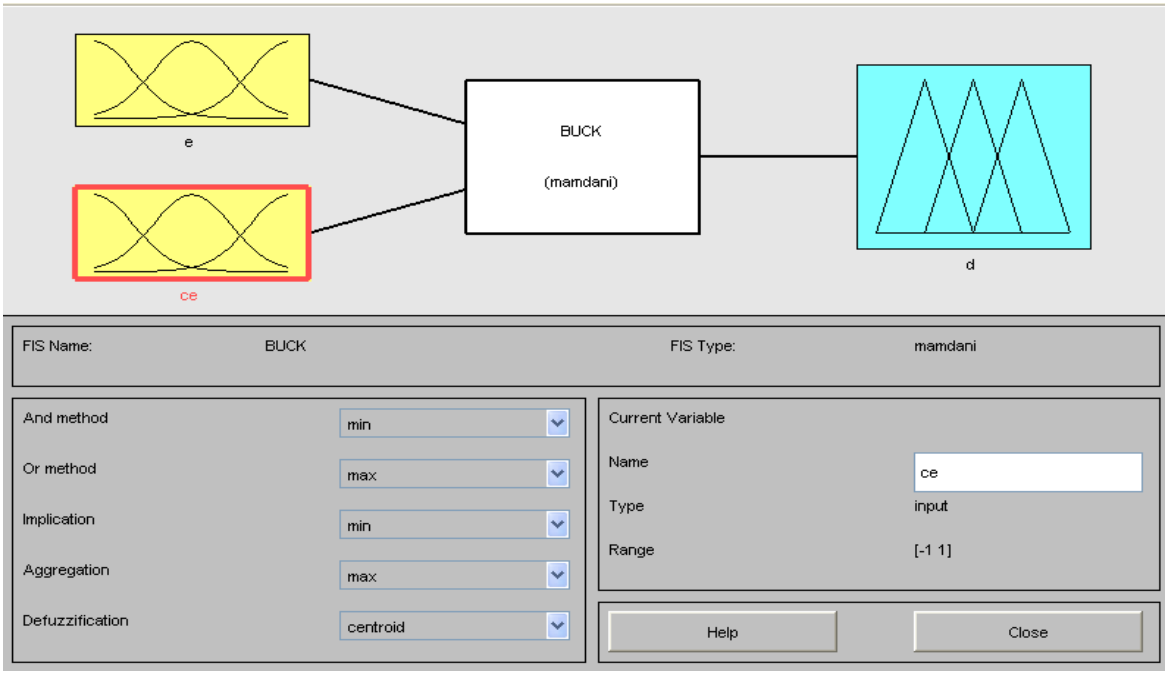


Figure 5.6: The Editor of Fuzzy Inference System using Matlab

The fuzzy logic control for DC-DC buck converter is made from two inputs and one output variable, as shown in figure 5.6, which are the error and change of error as an input variables, while the change of duty cycle is output variable. Each control variable has been divided into seven partitions. These partitions, which are called membership function have named into seven fuzzy subsets: PB (Positive Big), PM (Positive Medium), PS (Positive Small), ZO (Zero), NS (Negative Small), NM (Negative Medium) And NB (Negative Big). Partitions of fuzzy subsets and the shape of the membership function are shown in figure 5.7; i.e. (error variable) and (change

of error variable) and figure 5.8 (output variable).The triangular shapes of the membership function of this arrangement presume that for any particular input there is only one dominant fuzzy subset. The rules for the fuzzy controlled buck converter are tabulated in Table 5.1. From the tabulated table, the fuzzy rule base is formulated into 49 rules.

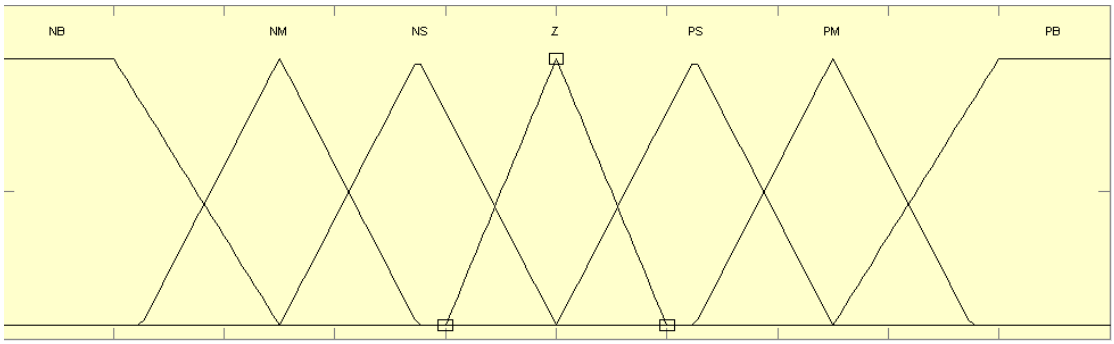


Figure 5.7: Membership Function of Input Variable 'error' and 'change of error'

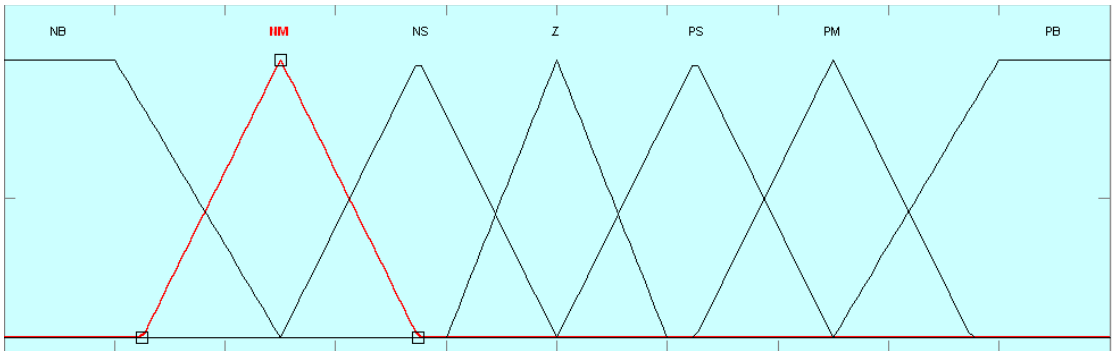


Figure 5.8: Membership Function of Output Variable

The next step after designing the membership functions is to write the rules of the fuzzy controller. These rules are chosen based on knowledge and experience. Table 5.1 shows the rules base as a matrix

Table 5.1: The Rule Base with 49 Rules

		ERROE						
		NB	NM	NS	Z	PS	PM	PB
CHANGE ERROR	NB	NB	NB	NB	NB	NM	NS	Z
	NM	NB	NB	NB	NM	NS	Z	PS
	NS	NB	NB	NM	NS	Z	PS	PM
	Z	NB	NM	NS	Z	PS	PM	PB
	PS	NM	NS	Z	PS	PM	PB	PB
	PM	NS	Z	PS	PM	PB	PB	PB
	PB	Z	PS	PM	PB	PB	PB	PB

Figure 5.9 shows the surface of fuzzy controller using the previous rules. Surface shows the relation between the inputs and output at any point in the intervals [-1 1] using the Centroid defuzzification method as introduced previously using equation (3.15).

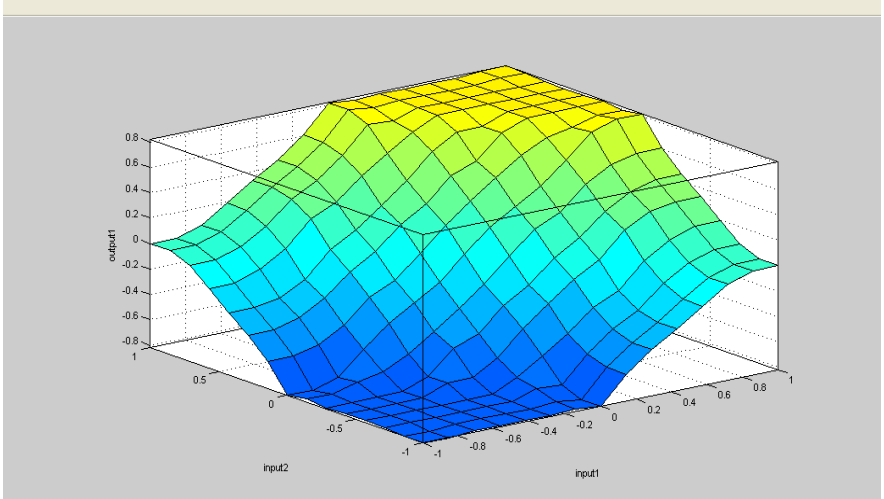


Figure 5.9: Surface of Fuzzy Controller

The simulation has been constructed using Matlab-Simulink environment is shown in figure 5.10. The results are obtained by adjusting the duty cycles with respect to voltage reference $14V$.

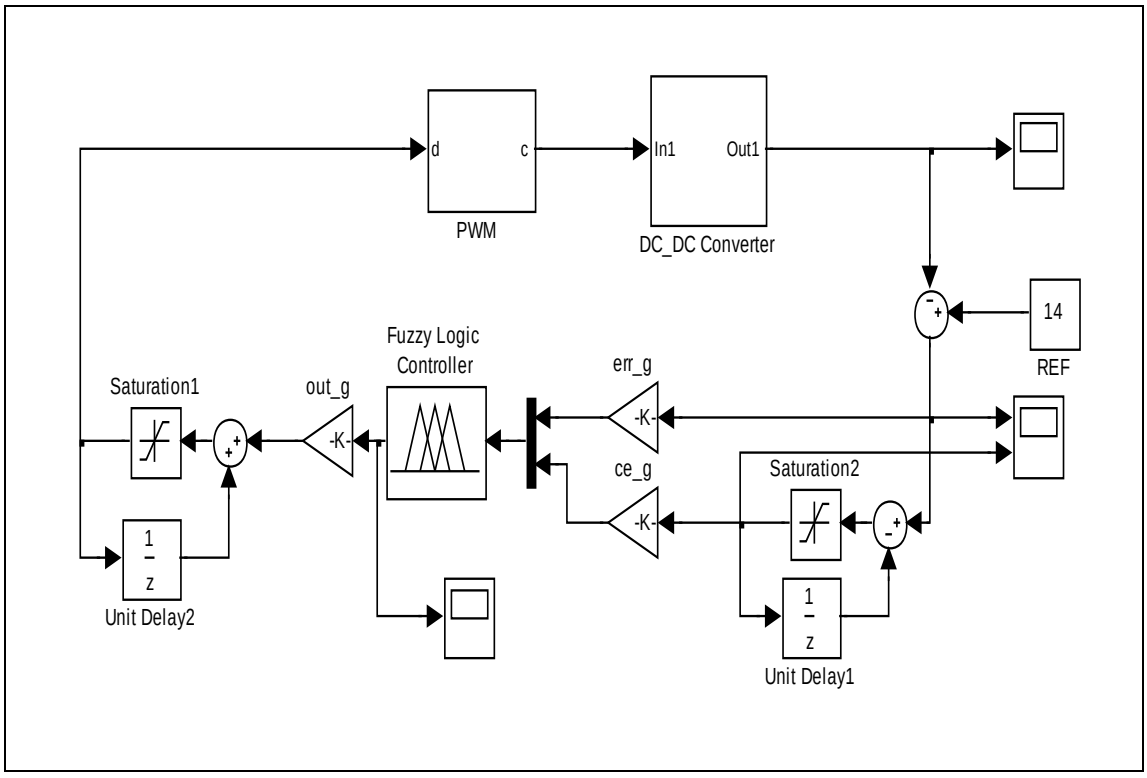


Figure 5.10:Matlab/Simulink Model (BUCK Converter Closed Loop with Fuzzy Controller)

5.4. Simulation Result

Simulated results obtained by Matlab/Simulink are shown in figures 5.11 and 5.12.The performance of the control scheme is assessed in terms of time domain specifications associated with output voltage response. As seen from figure 5.11, the compensated buck converter shows an under damped output voltage response and achieves the performance specifications of 0.83% overshoot and 0.045 ms settling time. The output voltage reaches its steady state value of $13.97V$ which gives a 1.5% steady state error. The steady state ripple is about $51.4mV$ which represent 0.37% of the steady state value of $14V$. The converter has a satisfactory time response. We can conclude that the results give a good satisfactory response. That means that fuzzy logic controller is a good alternative way to control power converters than the classical controllers.

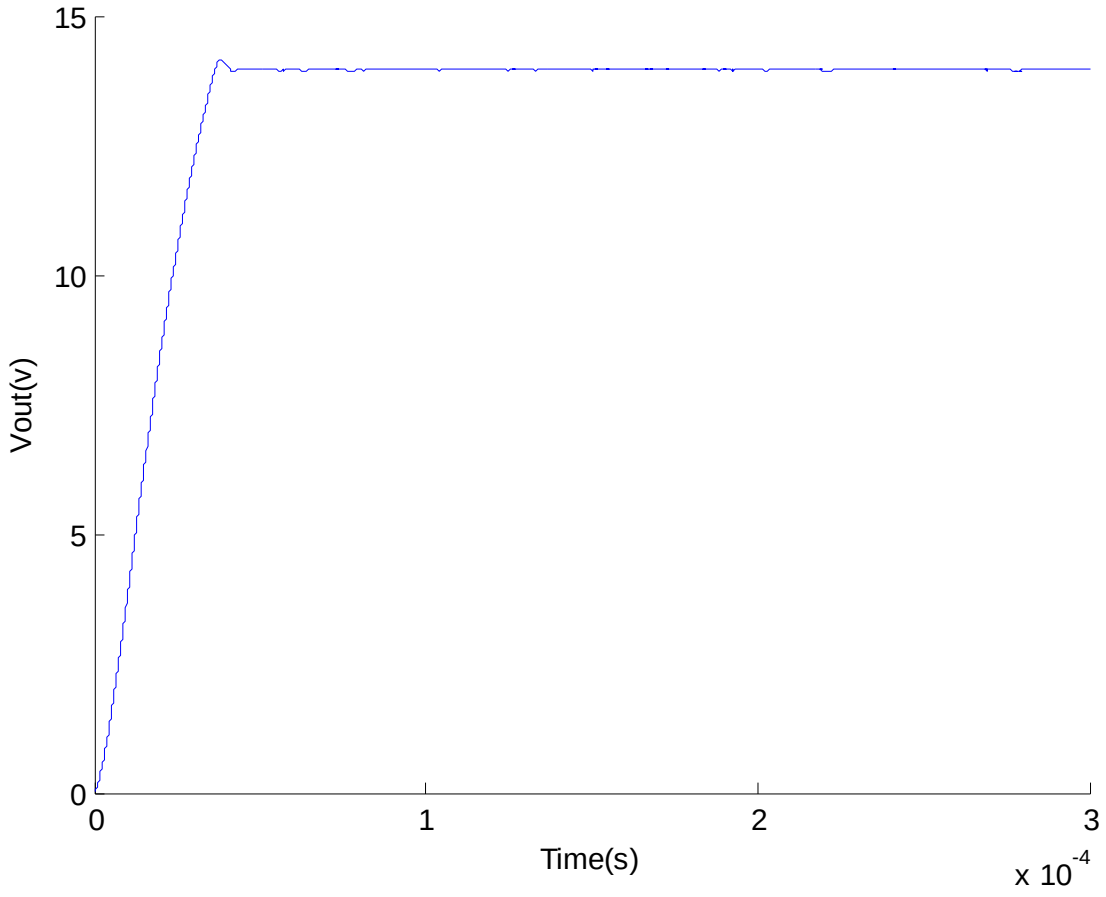


Figure 5.11: Responses FL Controller

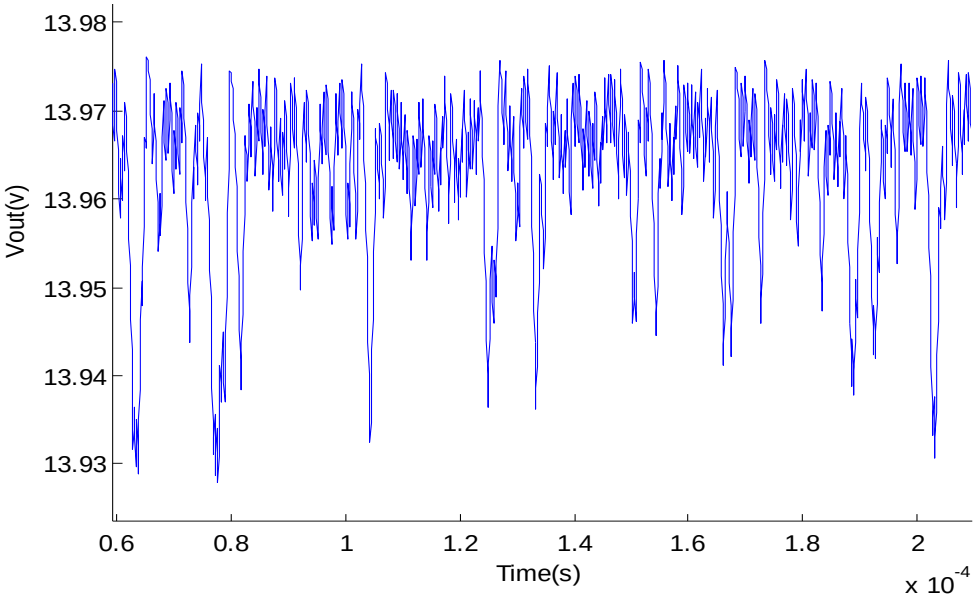


Figure 5.12: Ripples in the Output Voltage

Table 5.2: Output voltage follows the changes in reference voltage

V_{in} (V)	V_{ref} (V)	OS%	Tr (ms)	Ts (ms)
24	14	0.86%	0.033	0.042
24	13	0.23%	0.0296	0.039
24	12	0.225%	0.027	0.038

Table5.3: Converter Parameters

Symbol	Symbol Component Description	Values
R	Load Resistance	1 Ω
RL	Series Resistance of Inductor	80 mΩ
RC	Series Resistance of Capacitor	5 mΩ
L	Inductor of Output Filter	1 µH
C	Capacitor of Output Filter	200 µF
Vin	Input Voltage	24.0 V
Vout	Output Voltage	14.0 V

5.5 Fuzzy Controller Design with GA for Buck DC-DC Converter

Here are many parameters that affect on the control process of the Buck DC-DC converter model in addition to the shape of the memberships of the inputs and output of fuzzy controller; i.e. (change of error gain), err_g (error gain) and out_g (fuzzy output gain). These gain parameters can be tuned to give near optimal results. GA is used to optimize these parameters and optimize the shape of memberships of fuzzy controller. There are many ways to have good response, first is to optimize gain parameters only without any change in memberships shape. Second is to optimize the memberships shape without any change in gain parameters. The last way is optimizing both gain parameters and memberships shape. GA Matlab code (Appendix A) is used to optimize the memberships shape and gain parameters. This program is divided into two codes; main code is responsible for performing the GA steps such as selection, crossover, mutation and make new population. The second code is responsible for testing the new population to calculate the fitness function and out the best fitness value. In fuzzy controllers, there are two inputs and one output. Every input and output has seven membership functions, five of these membership functions are triangular memberships and the others trapezoidal membership function. But in this thesis some simplification are used in writing GA code.

1- The interval of the inputs and output will still at [-1 1] range.

2- The symmetric point will still at zero.

3- The inputs and output will have the same shape.

After this simplification the number of the variables will be 7 for membership functions and 3 for the gain parameters. The fitness function that will be used is the integral of the absolute values of the error, in the control design the fitness value need to be minimized. The number of individuals per one population will be 50, the number of generation will be 100, the crossover probability will be 0.7. The first step in Matlab program, generate random population consist of 50 individual, every individual consists of 13variable every variable is coded in 10 bit binary form. In the second step, it converts the binary code of each variable to real number. In the third step, one call fuzzy control program and simulink program and find the fitness value of the fitness function (the integral of absolute values of error).

Fourth, implementing crossover and mutation, and finally generate the new population. The stop condition can be after exact times or after the exact value of fitness value. in this thesis the stop condition is after 100 generation.

Figure 5.13 shows the memberships shape of the inputs and output. Figure 5.14 shows the Responses FL controller with GA and figure 5.15 shows the ripple of output voltage with GA.

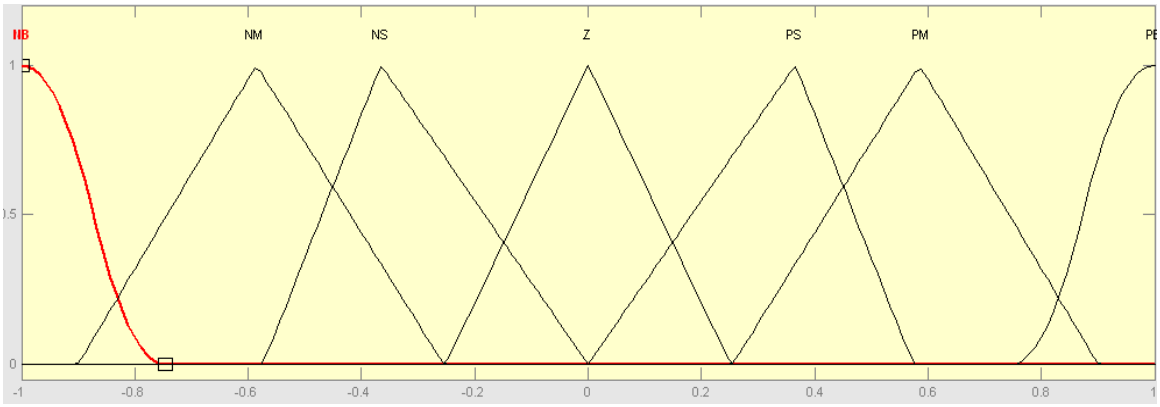


Figure 5.13: Membership Functions of the Fuzzy Controller with GA

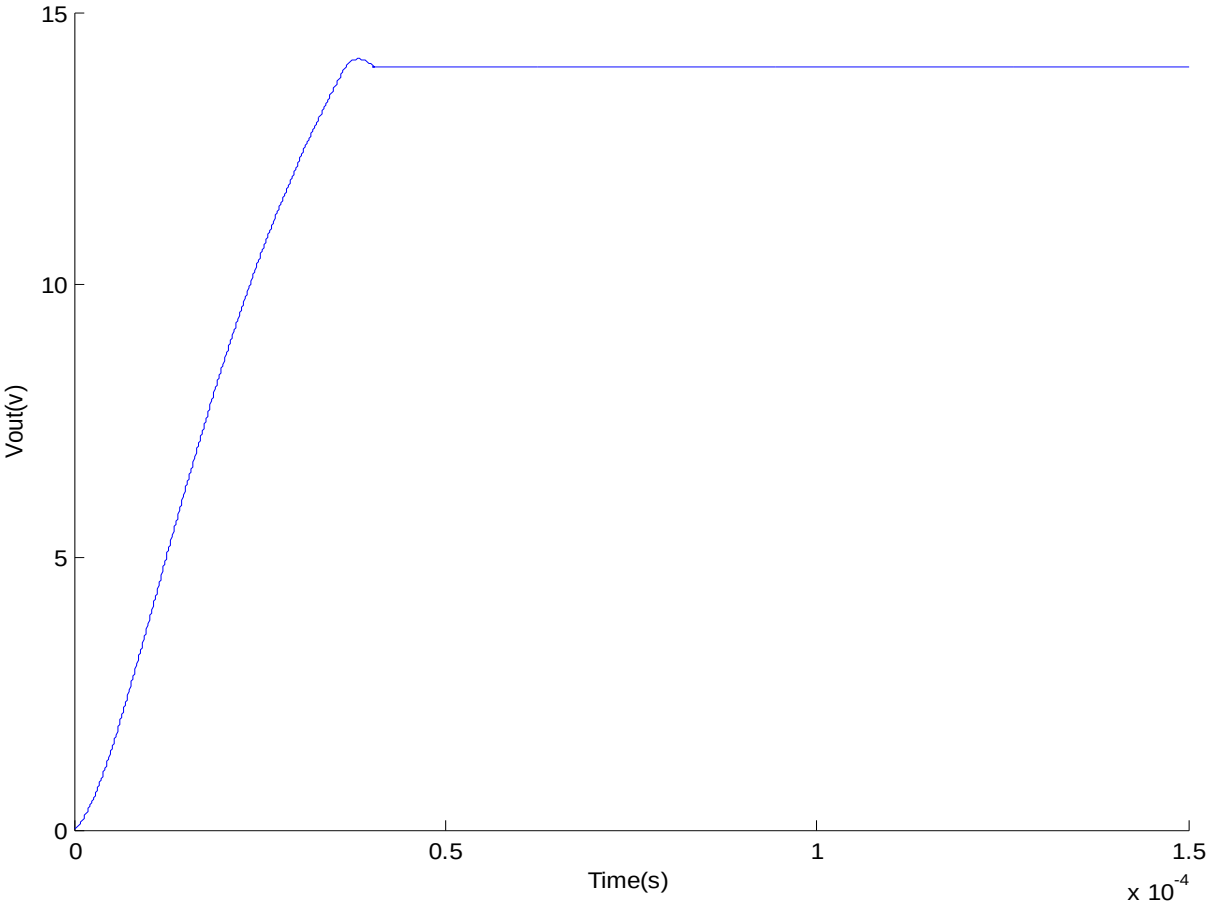


Figure 5.14: Responses FL Controller with GA

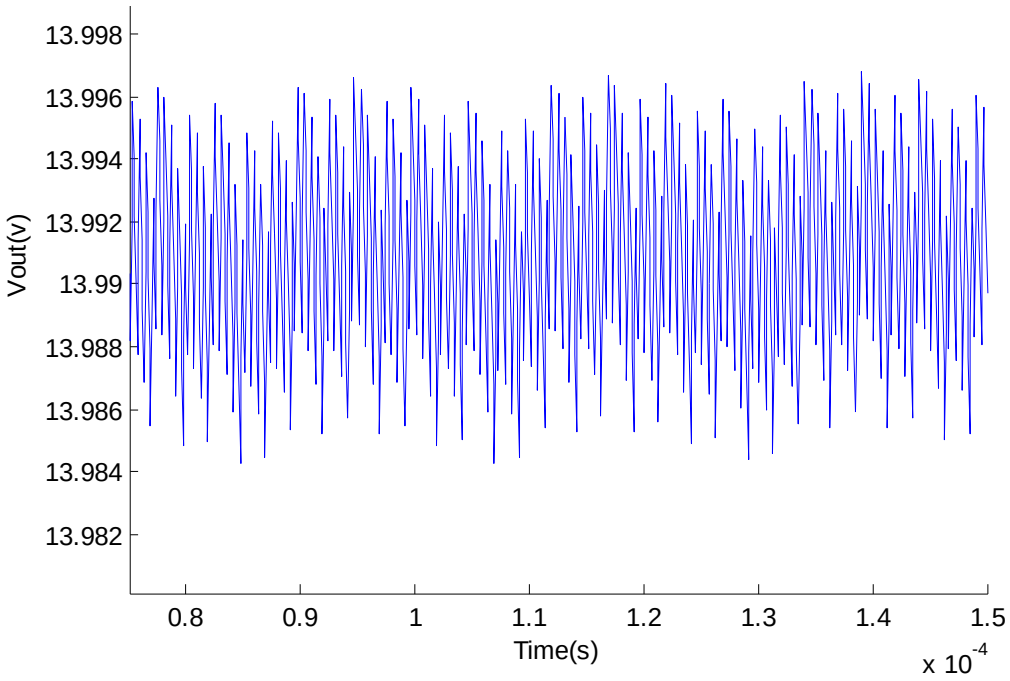


Figure 5.15: Ripples in the Output Voltage with GA

Table 5.4 shows the difference between parameters with and without GA optimization

Table 5.4: Parameters Values with and without GA Optimization

Parameters	With GA	Without GA
Overshoot	1.01%	0.83%
Settling time	0.041ms	0.043ms
Rising time	0.033ms	0.033ms
Error	0.5%	1.5%
Ripple	12mV	51.4mV

CHAPTER 6 CONCLUSION

6.1. Conclusion

Issues in the design and implementation of controllers for buck converters have been discussed in this dissertation. There are many control methods that may be used to design controllers for DC-DC converters. Generally, these methods fall into two categories: linear and nonlinear control methods.

Among the various techniques of artificial intelligence, the most popular and widely used technique in control systems is fuzzy control. Fuzzy controllers were designed based on the general knowledge of the converters.

In this thesis, fuzzy logic controller was designed to improve the system response such as settling time, overshoot and ripple of the output voltage. The advantages of fuzzy controllers are: exact mathematical models are not required for the design of fuzzy controllers, complexities associated with nonlinear mathematical analysis are relatively low, and fuzzy controllers are able to adapt to changes in operating points. The fuzzy controller was designed with Matlab software program. The GA optimization method was used to optimize the membership function of the inputs and output of the fuzzy controller and also to optimize the gains. The Buck model was tested with the new membership functions and new gains and the results were better than the results of old fuzzy controller.

6.2. Future Research

1- Sugeno method can be used instead of Mamdani model and make comparison between two methods.
2- Using fuzzy supervised PID techniques to give better results.
3- We can implement the FLC for (Buck Converter) by using the modern technique in FPGA.
4- In the future research the GA can be used in learning of rule base with Tuning of the data base to give better results.

REFERENCES

[1] Liping Guo, Thesis, "Design And Implementation Of Digital Controllers For Buck And Boost Converters Using Linear And Nonlinear Control Methods", Auburn, Alabama, August 2006.

[2] Y. S. Lee, Computer-Aided Analysis and Design of Switch-Mode Power Supplies, Marcel Dekker, Inc., New York, Basel, Hong Kong, 1993.

[3] N. Mohan, T. M. Undeland, W. P. Robbins, Power Electronics: Converters, Applications, and Design, John Wiley & Sons, Inc., 1995.

[4] F. H. Wang and C. Q. Lee, "Comparison of Fuzzy Logic and Current-Mode Control Techniques in Buck, Boost and Buck/Boost Converters", 1995 IEEE 26th Annual Power Electronics Specialists Conference, Vol. 2, pp. 1079 - 1085, June 1995.

[5] J. Arias, A. Arias, S. Gomariz and F. Guinjoan, "Generating design rules for buck converter-based fuzzy controllers", 1996 IEEE International Symposium on Circuits and Systems, Vol. 1, pp. 585 - 588, May 1996.

[6] P. Mattavelli and G. Spiazzi, "General-purpose fuzzy controller for DC-DC converters", IEEE Transactions on Power Electronics, Vol. 12, No. 1, pp. 79-86, January1997.

[7] I. Campo and J. M. Tarela, "Consequences of the Digitization on the Performance of a Fuzzy Logic Controller", IEEE Transaction on Fuzzy Systems, Vol. 7, No. 1, pp. 85-92**, Feb** 1999.

[8] K. Viswanathan, D. Srinivasan and R. Oruganti, "A Universal Fuzzy Controller for a Non-linear Power Electronic Converter", IEEE International Conference on Fuzzy Systems, Vol. 1, pp. 46-51, 2002.

[9] A. Perry, G. Feng, Y. Liu and P. C. Sen, "A new design method for PI-like fuzzy logic controllers for DC-DC converters", 35th Annual IEEE Power Electronics Specialists Conference, Aachen, Germany, pp. 3751-3757, 2004.

[10] Farahani, H. Feshki , "Designing and Implementation of a Fuzzy Controller for DC-DC Converters and Comparing with PI Digital Controller", Journal of Applied Sciences Research; July 2011, Vol. 7 Issue 7, p276.

[11] The Institute of Electrical and Electronics Engineers Inc., "IEEE Standard Dictionary of Electrical and Electronics Terms", Revised 2nd Edition

[12]International Electrotechnical commission

(IEC)http://www.electropedia.org/iev.iev.nsf/index?openform&part=551

[13] Power Electronics, 2/e, R S Ananda Murthy and V Nattarasu, Pearson Sanguine publications, 2010, ISBN: 9788131732403

[14] The Oxford English Dictionary http://oxforddictionaries.com/

[15] http://home.agh.edu.pl/~vlsi/AI/rozmyta_en/

[16] Robert Fuller, "Fuzzy Reasoning and Fuzzy Optimization," TUCS General Publications, No. 9, Turku Centre for Computer Science, Abo, 1998.

[17] M. Ibrahim, Fuzzy Logic for Embedded Systems **and** applications, Elsevier Science, MA, USA, 2004.

[18] Oscar Castillo and Patricia Melin, "Type-2 Fuzzy Logic: Theory and Applications," Springer-Verlag Berlin Heidelberg, 2008, ISBN: 978-3-540-76283-6

[19] Mohammed S. EL-Moghany, Sun and Maximum Power Point Tracking in Solar Array Systems Using Fuzzy Controllers via FPGA, Master Thesis, Islamic University-Gaza, 2011.

[20] H.-J. Zimmermann, *Fuzzy Sets Theory - and Its Applications*, Kluwer Academic Publishers, 1990.

[21] Robert Fulle´r, "Fuzzy Reasoning and Fuzzy Optimization," TUCS General Publications, No. 9, Turku Centre for Computer Science, Abo, 1998.

[22] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, MA, 1989.

[23] J.H. Holland, *Adaptation in Natural and Artificial systems*, University of Michigan Press, Ann Arbor, 1975.

[24] O. Cordon, F. Herrera, F. Hoffman and L. Magdalena, .Genetic Fuzzy System Evolutionary Tuning and Learning of Fuzzy Knowledge Bases," Worlds Scientific, Singapore, 2001.

[25] S.N. Sivanandam and S.N. Deepa, *Introduction to Genetic Algorithms*, Springer, New York, 2008.

[26] OutiRaiha, Applying Genetic Algorithms in Software Architecture Design, University of Tampere, Department of Computer Sciences Computer Science ,M.Sc thesis,February, 2008.

[27] M. Mitchell, *An Introduction to Genetic Algorithms*, MIT Press, 3rd edition. Springer. New York, 1996.

[28] Hung-Cheng Chen. and Sheng-Hsiung Chang, "Genetic Algorithms Based Optimization Design of a PID Controller for an Active Magnetic Bearing," IJCSNS International Journal of Computer Science and Network Security, VOL.6 No.12, December 2006.

[29] Hosam Abu Elreesh, "Design of GA-Fuzzy Controller For Magnetic Levitation Using FPGA", Master Thesis, Islamic University-Gaza, June 2011.

APPENDIX A GA MATLAB PROGRAMS

1- GA FUZZY MATLAB CODE MAIN PROGRAM

```
%-------------------------------------------------------------------------------------------

clc
clear
globalrinyouttimefe_absset_p
Ts=1e-7;
NE1=readfis('NE');
open_system('Buck');
MAXGEN = 100;%100; % maximum Number of generations
NVAR = 13; % Generation gap, how many new individuals are created
GGAP = .5; % Generation gap, how many new individuals are created
PRECI = 10; % Binary representation precision
NIND = 50; % No. of individuals per subpopulations
% First, a field descriptor is set up
FieldD = [rep([PRECI],[1, NVAR]); rep([-0.1;0.1],[1, NVAR]);...
rep([1; 0; 1 ;1], [1, NVAR])];
%-------------------------------------------------------
% The population is then initialized
%-------------------------------------------------------
FieldD(2,1)=0.9;FieldD(2,2)=14.5;FieldD(2,3)=14.5;FieldD(2,12)=2.4;FieldD(2
,13)=0;
FieldD(3,1)=1.1;FieldD(3,2)=15.5;FieldD(3,3)=15.5;FieldD(3,12)=2.8;FieldD(3
,13)=0.2;
FieldD;
BsJ=0;
E = crtbp(NIND, NVAR*PRECI);
E(1,:)=[1,0,0,0,0,0,0,0,1,1,0,1,1,0,0,1,0,1,0,0,1,1,0,0,0,1,1,1,1,1,1,1,1,0
,0,1,1,0,1,0,0,0,0,1,1,0,0,1,0,0,1,0,1,1,0,0,1,0,0,0,1,1,1,1,1,1,0,0,0,1,1,
1,0,1,0,0,0,1,0,1,0,1,0,1,1,1,0,0,1,1,1,0,0,1,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0
,1,1,1,0,0,1,0,1,0,1,0,1,1,0,1,1,1,0,1,0,1;];
E(2,:)=[1,0,0,0,0,0,0,0,1,1,0,1,1,0,1,0,1,1,0,1,1,0,0,0,0,0,0,0,0,0,1,1,1,0
```

```
,0,1,1,0,1,0,0,0,0,1,0,1,0,1,1,1,0,1,1,1,0,0,0,0,1,0,0,1,0,1,0,0,1,0,0,1,1,
0,1,1,0,0,0,0,1,1,1,0,0,1,1,1,0,0,0,1,1,1,0,1,1,0,0,1,1,0,1,1,1,1,1,0,1,0,0,1
,0,1,1,0,1,1,0,1,0,1,0,0,1,1,1,0,0,1,0,1,1;];

E(3,:)=[1,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,1,
0,1,1,0,0,1,1,1,0,0,0,0,0,0,1,0,0,0,0,0,1,1,1,1,0,1,0,1,0,1,0,1,0,0,1,0,0,1,1
,0,1,0,1,1,1,1,0,0,1,0,1,0,1,0,1,1,1,1,1,0,0;];
E(4,:)=[1,0,0,0,0,0,1,1,0,1,0,0,1,1,0,0,0,0,0,0,1,1,0,0,0,0,1,0,1,1,0,1,1,1,0,0,0
,0,0,0,0,1,0,0,1,0,1,0,0,1,1,1,0,0,0,0,0,1,1,0,0,1,0,1,1,0,0,0,0,0,1,1,0,1,1,0,
1,1,1,1,0,1,0,0,0,0,0,0,1,0,1,0,0,1,0,0,0,0,0,1,1,0,0,0,0,1,0,1,1,1,0,0,1,0,1,0,0
,1,0,0,1,1,0,0,0,0,0,1,1,0,1,1,0,1,1,0,0,0,0;];
for kg=1:1:MAXGEN
time(kg)=kg;
Kfuzzy=bs2rv(E,FieldD);
for s=1:1:NIND
%*************** Step 1 : Evaluate BestJ ****************************
Kfuzzyi=Kfuzzy(s,:);
[Kfuzzyi,BsJ]=mysystem(Kfuzzyi,BsJ);
BsJi(s)=BsJ;
end
[OderJi,IndexJi]=sort(BsJi);
BestJ(kg)=OderJi(1);
BJ=BestJ(kg);
Ji=BsJi+1e-10;
fi=1./Ji;
[Oderfi,Indexfi]=sort(fi);
Bestfi=Oderfi(NIND);
BestS=E(Indexfi(NIND),:);
kg
BJ
BestS;
%********** Step 2 : Select and Reproduct Operation*******************
fi_sum=sum(fi);
```

```
fi_Size=(Oderfi/fi_sum)*NIND;

fi_S=floor(fi_Size); %Selecting Bigger fi value
kk=1;
for i=1:1:NIND
for j=1:1:fi_S(i) %Select and Reproduce
TempE(kk,:)=E(Indexfi(i),:);
kk=kk+1; %kk is used to reproduce
end
end
%************ Step 3 : Crossover Operation ************
pc=0.99;
n=ceil(100*rand);
for i=1:2:(NIND-1)
temp=rand;
%if pc>temp %Crossover Condition
for j=n:1:100
TempE(i,j)=E(i+1,j);
TempE(i+1,j)=E(i,j);
end
%end
end
TempE(NIND,:)=BestS;
E=TempE;
%*********** Step 4: Mutation Operation ******************************
pm=0.001-[1:1:NIND]*(0.001)/NIND; %Bigger fi, smaller pm
for i=1:1:NIND
for j=1:1:3*PRECI
temp=rand;
if pm(i)>temp %Mutation Condition
if TempE(i,j)==0
TempE(i,j)=1;
else
TempE(i,j)=0;
```

```
end
end
end
end
TempE(NIND,:)=BestS;
E=TempE;
%*****************************************************************
**
end
Bestfi
BestS
Kfuzzyi
Best_J=BestJ(MAXGEN)
figure(1);
plot(time,BestJ);
xlabel('Times');ylabel('Best J');
```

2- GA FUZZY MATLAB CODE CALL SIMULINK PROGRAM

```
%-----------------------------------------------------------
function [Kfuzzyi,BsJ]=fuzzy_gaf(Kfuzzyi,BsJ)
global rinyouttimef
%-------------------------------------------------------------
a=newfis('NE');
a=addvar(a,'input','e',[-1,1]); %Parameter e
a=addmf(a,'input',1,'NB','zmf',[-1,-0.6668-Kfuzzyi(10)]);
a=addmf(a,'input',1,'NM','trimf',[-1-Kfuzzyi(9),-0.6668-Kfuzzyi(8), -
0.3332-Kfuzzyi(7)]);
a=addmf(a,'input',1,'NS','trimf',[-0.6668-Kfuzzyi(6),-0.3332-
Kfuzzyi(5),0]);
a=addmf(a,'input',1,'Z','trimf',[-0.3332-Kfuzzyi(4),0,0.3332+Kfuzzyi(4)]);
a=addmf(a,'input',1,'PS','trimf',[0,0.3332+Kfuzzyi(5),0.6666+Kfuzzyi(6)]);
a=addmf(a,'input',1,'PM','trimf',[0.3332+Kfuzzyi(7),0.6666+Kfuzzyi(8),1+Kfu
zzyi(9)]);
a=addmf(a,'input',1,'PB','smf',[0.6666+Kfuzzyi(10),1]);
a=addvar(a,'input','ec',[-1,1]); %Parameter ec
a=addmf(a,'input',2,'NB','zmf',[-1,-0.6668-Kfuzzyi(10)]);
a=addmf(a,'input',2,'NM','trimf',[-1-Kfuzzyi(9),-0.6668-Kfuzzyi(8), -
0.3332-Kfuzzyi(7)]);
a=addmf(a,'input',2,'NS','trimf',[-0.6668-Kfuzzyi(6),-0.3332-
Kfuzzyi(5),0]);
a=addmf(a,'input',2,'Z','trimf',[-0.3332-Kfuzzyi(4),0,0.3332+Kfuzzyi(4)]);
a=addmf(a,'input',2,'PS','trimf',[0,0.3332+Kfuzzyi(5),0.6666+Kfuzzyi(6)]);
a=addmf(a,'input',2,'PM','trimf',[0.3332+Kfuzzyi(7),0.6666+Kfuzzyi(8),1+Kfu
zzyi(9)]);
a=addmf(a,'input',2,'PB','smf',[0.6666+Kfuzzyi(10),1]);
a=addvar(a,'output','u',[-1,1]); %Parameter u
a=addmf(a,'output',1,'NB','zmf',[-1,-0.6668-Kfuzzyi(10)]);
a=addmf(a,'output',1,'NM','trimf',[-1-Kfuzzyi(9),-0.6668-Kfuzzyi(8), -
0.3332-Kfuzzyi(7)]);
a=addmf(a,'output',1,'NS','trimf',[-0.6668-Kfuzzyi(6),-0.3332-
```

```
Kfuzzyi(5),0]);
a=addmf(a,'output',1,'Z','trimf',[-0.3332-Kfuzzyi(4),0,0.3332+Kfuzzyi(4)]);
a=addmf(a,'output',1,'PS','trimf',[0,0.3332+Kfuzzyi(5),0.6666+Kfuzzyi(6)]);

a=addmf(a,'output',1,'PM','trimf',[0.3332+Kfuzzyi(7),0.6666+Kfuzzyi(8),1+Kf
uzzyi(9)]);
a=addmf(a,'output',1,'PB','smf',[0.6666+Kfuzzyi(10),1]);

%---------------------------------------------------------------
rulelist=[1 1 1 1 1; %Edit rule base
1 2 1 1 1;
1 3 1 1 1;
1 4 1 1 1;
1 5 2 1 1;
1 6 3 1 1;
1 7 4 1 1;

2 1 1 1 1;
2 2 1 1 1;
2 3 1 1 1;
2 4 2 1 1;
2 5 3 1 1;
2 6 4 1 1;
2 7 5 1 1;

3 1 1 1 1;
3 2 1 1 1;
3 3 2 1 1;
3 4 3 1 1;
3 5 4 1 1;
3 6 5 1 1;
3 7 6 1 1;

4 1 1 1 1;
4 2 2 1 1;
4 3 3 1 1;
```

```
4 4 4 1 1;
4 5 5 1 1;
4 6 6 1 1;
4 7 7 1 1;

5 1 2 1 1;
5 2 3 1 1;
5 3 4 1 1;
5 4 5 1 1;
5 5 6 1 1;
5 6 7 1 1;
5 7 7 1 1;

6 1 3 1 1;
6 2 4 1 1;
6 3 5 1 1;
6 4 6 1 1;
6 5 7 1 1;
6 6 7 1 1;
6 7 7 1 1;

7 1 4 1 1;
7 2 5 1 1;
7 3 6 1 1;
7 4 7 1 1;
7 5 7 1 1;
7 6 7 1 1;
7 7 7 1 1];
a=addrule(a,rulelist);
a=setfis(a,'DefuzzMethod','centroid');
writefis(a,'NE');
NE1=readfis('NE');
ke=num2str(Kfuzzyi(1));
kce=num2str(Kfuzzyi(2));
ku=num2str(Kfuzzyi(3));
```

```
ki=num2str(Kfuzzyi(12));
kp=num2str(Kfuzzyi(13));
set_param('Buck/err_g','Gain',ke);
set_param('Buck/ce_g','Gain',kce);
set_param('Buck/out_g','Gain',ku);
[t,x,y]=sim('Buck');
clear t;
clear x;
clear y;

BsJ=0;
ts=1e-7;
for k=1:1:201
timef(k)=k*ts;
Ji(k)=(e_abs.time(k)*e_abs.signals.values(k));
BsJ=BsJ+Ji(k);
end
end
```