

FUZZY LOGIC BASED WIDE-RANGE VIBRATION CONTROL SYSTEM

A DISSERTATION

*submitted in partial fulfilment of the
requirements for the award of the degree*

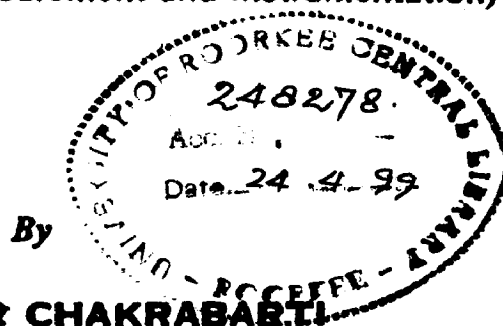
of

MASTER OF ENGINEERING

in

ELECTRICAL ENGINEERING

(With Specialization in Measurement and Instrumentation)



SUJIT KUMAR CHAKRABARTI



**DEPARTMENT OF ELECTRICAL ENGINEERING
UNIVERSITY OF ROORKEE
ROORKEE-247 667 (INDIA)**

MARCH, 1999

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this dissertation entitled 'FUZZY LOGIC BASED WIDE-RANGE VIBRATION CONTROL SYSTEM', in partial fulfilment of the requirement for the award of the degree of Master of Engineering with specialisation in Measurement and Instrumentation, submitted in the Department of Electrical Engineering, University of Roorkee, Roorkee is an authentic record of my own work carried out during the period from July 1998 to March 1999 under the guidance of Dr. H.K. Verma, Professor and Dr. Vinod Kumar, Professor, Department of Electrical Engineering, University of Roorkee, Roorkee.

The matter embodied in the dissertation has not been submitted by me for the award of any other degree.


Dated : 24th March 1999

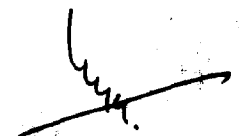
Place : Roorkee



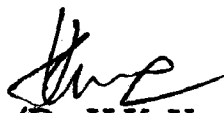
(SUJIT KUMAR CHAKRABARTI)

CERTIFICATE

This is to certify that the above statement made by the candidate is correct  the best of our knowledge.



(Dr. Vinod Kumar)
Professor
Deptt. of Electrical Engineering
University of Roorkee
Roorkee - 247 667, INDIA



(Dr. H.K. Verma)
Professor
Deptt. of Electrical Engineering
University of Roorkee
Roorkee - 247 667, INDIA

ACKNOWLEDGMENT

My first and foremost note of gratitude goes to my guide Dr. H.K. Verma. I had always more to learn from observing him than from direct instructions. His intelligence, experience, perseverance and unmatched technical enthusiasm are things whose memories, I will always carry along and these will remain unending sources of inspiration for me.

My co-guide, Dr. Vinod Kumar also deserves a special mention of thankfulness. His practical attitude and helpfulness with full lack of pretense, are precious lessons I gained.

I also thank Dr. S.C. Saxena, Head, Department of Electrical Engineering who showed all the enthusiasm possible in providing me with the facilities required.

The laboratory staff in Instrumentation and Signal Processing Laboratory of Department of Electrical Engineering must also be thanked with full heart. Mr. Mishra, Mr. Kansal, Mr. Moajjam Ali, and Rajesh - Thank you.

If an acknowledgement is due for adding the flavour of human emotions to my endeavours then it is to my friends here, without whom this very wonderful experience of doing the dissertation would have been ridden with loneliness.

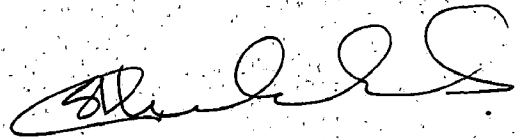
Shripad Deshpande (M.Tech CST) remained my enthusiastic companion throughout my stay here. His intelligence and simplicity are not a source of fondness and liveliness, just for me, but for many.

Mr. N.L. Prajapati, from whom I will always draw lessons of modesty, simplicity, patience, and love was one of the radical sources of inspiration. He played the role of an excellent teacher by making me learn and practise more, in the veil of lending assistances to him.

The agonies and ecstasies of the experience of doing this dissertation were born in full with me by my three friends - Amitabh, Deependra, and Manish. A note of thanks to those lively companions.

Y. Srikanth (M.E. SEOR), P.K. Roy (M.Tech CST), Harish Agarwal (M.Tech CST), and B.V. Naheshwara Rao (M.E. Mech) were friends who lent a vent to my parallel self-my very talkative philosophical self.

Last but not the least, a mention of love and respect for my parents. While they persistently stuck to their view of me as a little child, I struggled hard to grow up in there eyes. My failure to do so could be the reason of many a successful venture.



(SUJIT KUMAR CHAKRABARTI)

ABSTRACT

A PC based generalised vibration control system has been developed using fuzzy logic. The hardware system on which the controller software has been designed is a vibration table meant primarily for testing energy-meters of various types. Generalisation of the controller aims at making it free to the maximum possible extent, from the dependencies on the behaviour of the hardware and on load variations. This is attempted by providing neat interfaces for conveniently changing the controller characteristics as and when required. The structure of the controller is based on fuzzy patterns ; hence the changes to be made during improvisations have a better prospect of meeting the intuitions. The control was implemented in real-time on the aforementioned vibration test-system, and its performance tested. The control system has been proved in regards to its controlling capability, thereby validating the fuzzy logic controller.

The control system software has been developed in Turbo C++. The operating system was MS-DOS 6.22.

CONTENTS

	Page No.
CANDIDATE'S DECLARATION	(i)
ACKNOWLEDGMENT	(ii)
ABSTRACT	(iv)
Chapter One	
INTRODUCTION	1
1.1 General	1
1.2 Types of Control Systems	2
1.3 Fuzzy Logic Controller	4
1.4 Vibration Control System	4
1.5 Statement of the Problem	4
Chapter Two	
FUZZY CONTROLLERS	6
2.1 Paradigm of Fuzzy Logic	6
2.2 Theory of Fuzzy Logic	8
2.3 Basic Fuzzy Controller	12
Chapter Three	
VIBRATION CONTROL SYSTEM	14

3.1	Introduction	14
3.2	Vibration Exciter	14
3.3	Power Amplifier	16
3.4	Acceleration Transducers	16
3.5	Signal Conditioning	16
3.6	Data Acquisition System	17
3.7	Fuzzy Vibration Controller Structure	18
3.8	Self-Tuning Fuzzy Controller	21

Chapter Four

	FUZZY VIBRATION CONTROLLER DESIGN	24
4.1	Introduction	24
4.2	Class Description	24
4.3	Class Hiercharchy	25
4.4	Fuzzification	28
4.5	Computation of Weights of Rules	29
4.6	Computation of Final Weights	32
4.7	Weighing of Output Fuzzy Sets	33
4.8	Obtaining the Output Fuzzy Set	36
4.9	Defuzzification	37

Chapter Five

INTERFACE SOFTWARE	39
5.1 Introduction	39
5.2 Initialisation	39
5.3 Acquisition of Feedback Data	41
5.4 Outputting One Cycle of Sinusoid	41
5.5 The Complete Control Procedure	42
5.6 Computation of One Control Sample	44
5.7 User Interface	44
Chapter Six	
RESULTS AND CONCLUSIONS	47
6.1 Performance Record	47
6.2 Discussions	47
6.3 Conclusions	51
6.4 Scopes of Future Work	52
REFERENCES	54
APPENDIX	56

INTRODUCTION

1.1 GENERAL

Control systems have found a wide range of applications in fields ranging from simple to sophisticated. Space-vehicle systems, missile guidance systems, aircraft autopilotings systems process control and biomedical engineering are some of them. The theory of control system is quite a well advanced field. Research and development in the field of control systems finds one of the greatest number of members and patrons.

The application of control system has itself come a long way. While the very first instance of the application of control can hardly be traced, its journey from the day of inception has been from perhaps a manually controlled single element open loop control system to multi state variable learning and adaptive control systems. Implementation of control systems, has been possible due to the advent of high performance actuating elements (pneumatic, hydraulic and electric among others), electrical and electronic devices (for signal conditioning and control law implementation), and digital computers (for data processing).

Even as the field of control systems undergoes an unending evolution and progress an introduction to the established types would still be instructive for a broad classification.

The following section gives a brief idea about these classes of controllers.

1.2 TYPES OF CONTROL SYSTEMS [1,2]

1.2.1 Open-loop control system

These systems contain a forward path and there is no way by which an output can have an effect on the input automatically. That is, there is no feedback arrangement in such control systems. Simple systems, with fairly satisfactory calibrations are open loop control systems. They are always stable.

1.2.2 Closed Loop Control System

In these system there are feedbacks made from the output to the input. This brings into action the idea of automatic control. Such systems are relatively insensitive to noise and interference. Thus components of poorer quality can be afforded for saving on cost. However, stability is a major consideration in the design of closed loop control systems.

Such controller are found in a number of varieties e.g. feedback, controllers, cascade controllers, feedforward controllers, ratio controllers and PID controllers.

1.2.3 Adaptive Control Systems

Adaptation is the property of self adjustment and self modification in accordance with the unpredictable changes in conditions of environment or structure. The concept of adaptive control has a great deal of appeal to the system designer since it will also accommodate moderate engineering design errors or uncertainties and will compensate for the failure of minor system components, thereby increasing the overall system reliability.

1.2.3 Learning Control Systems

Almost all open loop control systems have a human operator who provides the primary control operations by observing the system performance. As the time goes by, the operator becomes more experienced in carrying out operations for optimal system performance. Strictly speaking, by the induction of human operators, open loop control systems not only become closed loop, they become learning control systems.

To replace this human operator by a feedback system closely matching him in performance, that system must have training capability. Neural networks and genetic algorithms have contributed significantly to the realisation of such systems.

1.2.4 Expert Control Systems

These are knowledge based systems having capacity not only to learn, but to behave similar to human experts. The knowledge base is prepared with the help of human experts. Once in action, these systems learn and augment their knowledge suitably, and can even have a capability of behaving intelligently in unforeseen conditions.

The trend is towards simulating the human being in control system, thus making them fully automatic. At the cutting edge, knowledge based systems are there. The ultimate milestone would be to simulate the 'human wisdom' which, perhaps, has more to do with man's ability to emote and feel, than with his ability to think. Man has to go a long way before this can be achieved.

1.3 FUZZY LOGIC CONTROLLER

Whenever there is tremendous complexity, or non linearity, or ambiguity in various aspects of a control system design, there is a search for methods for imitating the elements responsible for this behaviour.

Fuzzy logic controllers have provided an efficient method of doing that. First implemented in Japan for industrial purposes, now fuzzy logic controllers are becoming increasingly popular. Known for their closeness to human beings in interpreting data, they provide an efficient way of accommodating non-linearities, ambiguities and uncertainties.

1.4 VIBRATION CONTROL SYSTEM[3]

Before being commissioned, a batch of energy meter should pass through a number of type tests. Vibration test is one of these. The computerised vibration test system for energy meters available in the I&SP Lab of EED UOR subjects a test object (energy meter) to vibrations of varying frequency, amplitude and duration as per IS13010 / IS : 13779. The vibrations is controlled by the personal computer. The control law used is PID.

The hardware used in this dissertation is common to that of the above control system. The control is, however based on fuzzy logic. Details of the hardware are presented in chapter three and other details of the controller are given in chapters four and five.

1.5 STATEMENT OF THE PROBLEM

This dissertation has been carried out in three steps which are as follows :

1. A software was developed which constitutes the basic controller. The control system employs fuzzy logic to frame its control law. The development of the software aims at making the controller generalised enough for being used in a variety of applications with little or no changes to be made in the software itself. The software provides facilities of changing the controller settings, to save them in files and to load them from existing ones, as and when required.
2. In this step the basic fuzzy logic control system was implemented on a practical system for controlling vibrations of a vibration exciter used to perform vibration tests on electrical energy meters. The settings of the controller were manually tuned to improve the controller performance to the maximum extent possible.
3. An additional feature of self tuning capability was incorporated in the final stage. The performance of the controller with and without self-timing feature were studied and compared to with the existing PID controller [3].

The results are summarized in chapter six of this report. Conclusion of the work are also brought out in chapter six.

FUZZY CONTROLLERS

2.1 THE PARADIGM OF FUZZY LOGIC [4,5]

In his well known paper Warren Weaver (1948) classifies all problems into three groups, namely, 'organised simplicity', 'disorganised complexity' and 'organised complexity'.

Organised simplicity refers to such problems which are easily modeled and contain only or limited number of variables. Such problems are best handled by classical algebra and calculus. Disorganised complexity is present in problem having very large number of variables and sufficient randomness. Such problems are suitably handled with statistical mathematics.

These two groups of problems, according to Warren Weaver, lie at the two extremes of the universe of all problems we face. Unfortunately, only a tiny portion of all problems faced by humans fall in these two categories. Most problems -- and the most important ones -- fall in the third group : organised complexity. These problems do not have such a small number of variables as could be accommodated within a mathematical model following paradigms of classical mathematics ; nor do they have enough randomness as would make statistics a suitable tool. Yet these problem exist, and are too important to be abandoned. Many ways to deal with such problems have come up. Fuzzy logic is one of them.

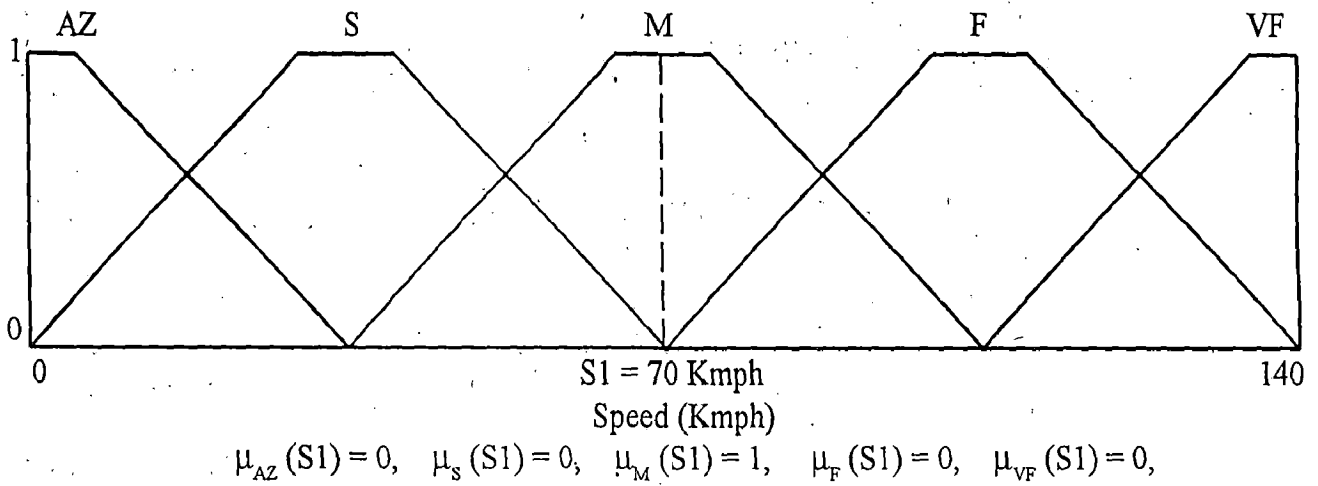


Fig. 2.1 Speed of a car represented as a fuzzy variable

Fuzzy logic tries to break out of the constraints of binary logic. In presence of uncertainties and ambiguities, a statement need not be completely true or false. Its degree of truth and false hood is allowed to be partial ; to lie anywhere in the range $[0,1]$. It is proposed in the fuzzy logic literature that human do not interpret data on strict numerical terms. Rather, values are seen as linguistic data (small, very large, a bit harder etc.) and the association of actual numerical data to these linguistic terms are done through certain patterns which describe these linguistic concepts.

Fuzzy logic tries to initiate this method of abstraction of data into information and then to knowledge.

Research on the theory of fuzzy sets has been growing steadily since its inception in mid 1960s. Research on broad variety of applications has also been very active and has produced impressive results. Some of the fields where fuzzy logic funds popularity are : Control, decision making, pattern recognition, simulation of non-linear system and many others.

2.2 THEORY OF FUZZY LOGIC

Fuzzy logic tries to imitate the way a human being makes a decision. Thus what is involved in the heart of the decision making processes based on fuzzy logic is not strict numerical data. Rather, they are certain linguistic data, fuzzy variables associated to them, and a set of rules operating on these variables. A brief introduction to these concepts is given in this section. More detailed explanation of the working of fuzzy logic is given in chapter three and chapter four with reference to our fuzzy logic controller.

(a) Linguistic variable

When a man uses a particular quantity for making a decision, he does not go beyond a precision as defined by assigning a linguistic value to that quantity.

For example, a man requires to be somewhat tall and somewhat dark to be handsome. While we judge the handomeness of a man, we hardly bother about precisely measuring his height. Yet, there will be a general agreement among many regarding whether the man is handsome or not.

Such variables are called linguistic variables. For example, tall, dark, handsome, somewhat hot or very hot, bad or very bad, bright or glaring etc.

To involve fuzzy logic, such concepts are used, and crisp, numeric data are converted into or extracted from such data (called as fuzzification and defuzzification respectively) as per standard fuzzy logical rules.

(b) Fuzzy variable

In a fuzzy system, linguistic concepts (or variables) are defined on various fuzzy sets. Then, these variables become fuzzy variables.

Thus, take the example of the speed of car. It may take any value between zero and 140 kmph. This range may be divided into five linguistic variables namely, approximately zero (AZ), low (L), medium (M), fast (F) and very fast (VF). The fuzzy variable speed can then be defined by the five linguistic terms converted into fuzzy sets as shown in Fig. 2.1.

The fuzzysset corresponding to each linguistic concept is also called as a membership function of that linguistic variable. A membership function is a representation

of the grade by which a given crisp value belongs to that linguistic variable. Thus the questions to which the definition of membership functions provides us an answer are such as: If the speed of the care is 70 kmph, what is its membership to ' APPROXIMATELY ZERO' fuzzy set ?

The shapes of the membership functions can be anything that matches the basic cognitive idea of that linguistic concept. the membership grade can take any value in the range $[0,1]$, and the domain over which the function is defined is the same as the range of the given quantity.

(c) Rule - Base :

Depending upon the application, there is a set of binary rules which suggests the linguistic value of the output variable(s) depending upon the linguistic value of the input variable(s). This set of rules is called a rule base.

Examples:

- If the speed of the approaching car is fast and its distance from you is small the move out of its way.
- If temperature is very hot switch off the oven.
- If demand on power is increasing, and transformer is getting very hot then pump the cooling oil slightly faster.

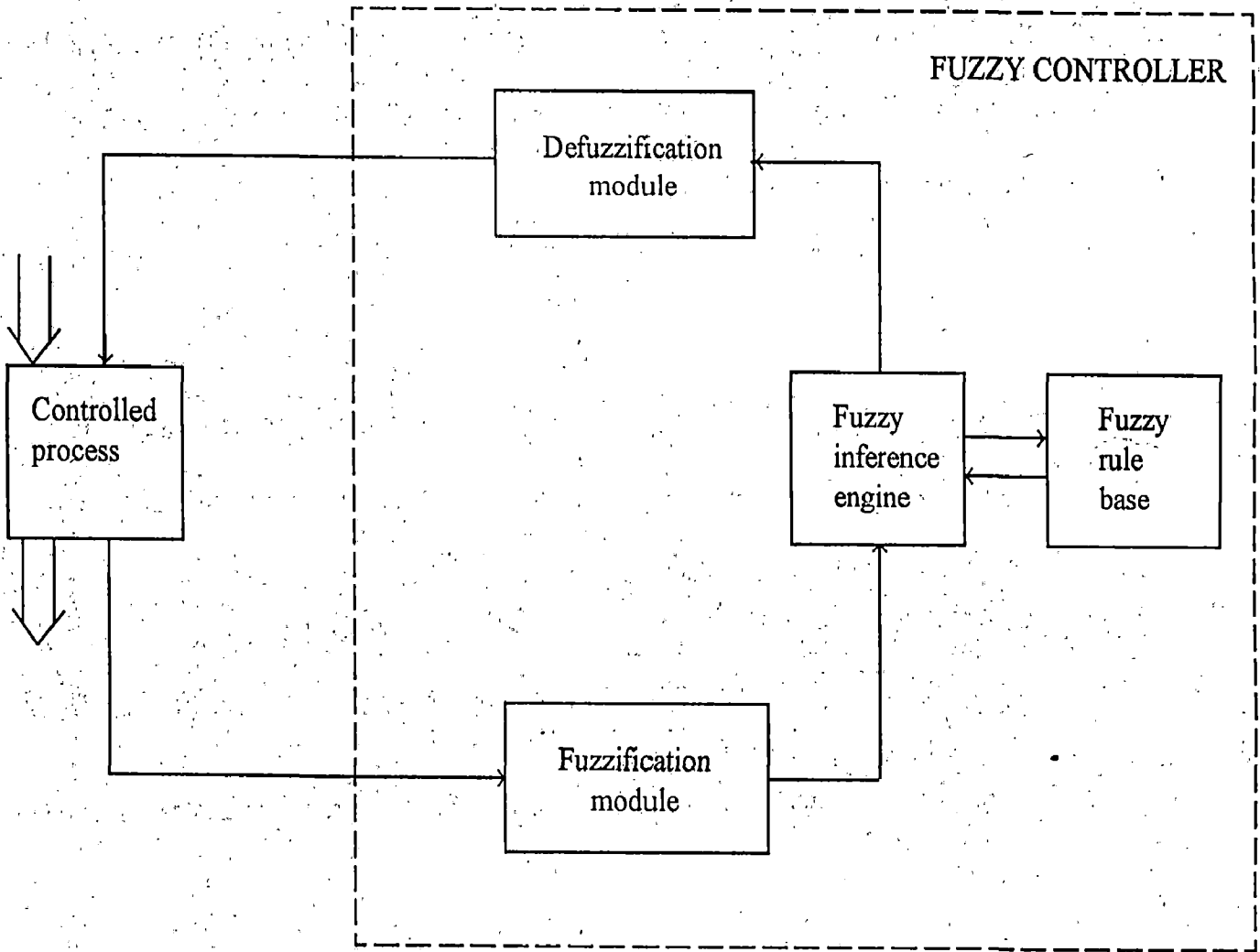


Fig. 2.2 Block diagram of a basic fuzzy controller

2.3 BASIC FUZZY CONTROLLER [4]

Fig. 2.2 shows the block diagram of a basic fuzzy controller. It shows that a basic fuzzy controller consists of four modules, namely, a fuzzy rule base, a fuzzy inference engine and fuzzification and defuzzification modules. In the sequence of the computation of one control sample, the roles of various modules of the basic fuzzy controller are described below :

a) **Fuzzification Module** : The computation of the next control sample, be it a control system of any type, depends on the present condition of the controlled plant. These plant conditions are transduced (measured) by appropriate measuring instruments. the measured value thus obtained at the output of the sampler (or analog-to-digital converter) is usually crisp or distinct. For implementation of fuzzy inferences, these values are required in fuzzy form. This transformation from crisp to fuzzy form of the plant condition measured value is done by the fuzzification module.

The output of the fuzzification module is the set of membership values of the measured variable to the various membership functions (fuzzy sets) defined for that variable.

b) **Inference Engine** : The fuzzified measured variables are then made use of by the inference engine for the computation of the fuzzy control output. The interference process includes selection of proper rules from the rule-base; weighing of the membership functions of the output fuzzy variable; and finally the formation of the fuzzy output sample by proper combination of the weighed membership functions.

The output of the inference engine is a single fuzzyset corresponding to the output control sample.

c) **Rule-Base** : This is the data base containing usually linguistic rules based upon linguistic variables as recognised by the fuzzy controller's inference engine.

For example :

IF the temperature is very high

AND the pressure is slightly low

THEN the heat change should be slightly negative.

The above rule to one is the many rules that can be contained in the rule-base.

Obviously, there should be membership functions clearly defined in the controller for the linguistic terms appearing in the rule-base. For example, there should be a fuzzyset (membership function) for the linguistic terms very high, slightly low and slightly negative for the fuzzy variable defined for temperature, pressure and heat change, respectively.

These rules are used by the inference engine of the controller for the computation of the fuzzy control output.

d) **Defuzzification Module** : The output of the inference engine is in fuzzy form. It has to be converted into crisp form (defuzzified) as the final step of computation of the crisp control output sample. This work is done by the defuzzification module.

There are a number of defuzzification methods namely, centre of area method, centre of maxima and mean of maxima method. One of these methods that is found most suitable in context to the target problem, can be used.

VIBRATION CONTROL SYSTEM

3.1 INTRODUCTION

The vibration control system can be divided into two parts : namely, the hardware and the software. the hardware consists of the vibration test system (vibration exciter, power amplifier, signal conditioners and transducer) and PC. The software consists of the basic fuzzy logic control and its interfaces with the hardware and user.

The hardware was adopted as it existed in a completely developed stage. The development was done in the Instrumentation and Signal Processing Laboratory of Deptt. of Electrical Engg., University of Roorkee; and has been under use with a digital PID controller for carrying out vibration tests on electrical energy meters [3]. The details of this hardware system have been provided in section 3.2 though 3.6.

The fuzzy controller was fully developed and implemented during this dissertation work. The details of its structure have been provided in section 3.7. The block schematic of the vibration control system is shown in Fig. 3.1.

3.2 VIBRATION EXCITER

Vibration exciter (or vibration table) on which the test object is mounted. The vibration exciter vibrates at the frequency of the input sinusoidal signal fed to it. The amplitude of the vibration is dependent on the amplitude of the input signal.

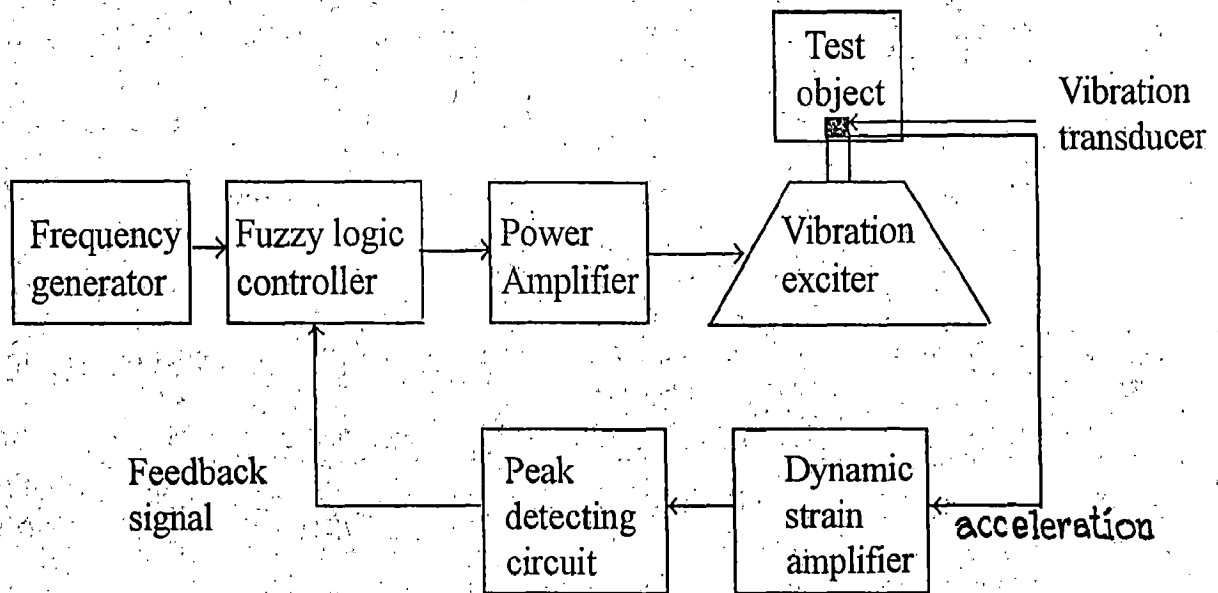


Fig.3.1 Block diagram of fuzzy logic vibration control system

3.3 POWER AMPLIFIER

The sinusoidal input signal to the vibration exciter is power amplified by this power-amplifier. There are digital displays for the frequency of the input signal, and for the output voltage and current on the front panel. Also provided on the front panel is knob for gain adjustment.

3.4 ACCELERATION TRANSDUCERS

For the transduction of the acceleration of the vibration exciter, to be used as the measured value in the feedback path of the vibration control system, two types of acceleration pickups are available here.

a) Piezo electric transducer : gives better performance at higher frequency range, while performance is not satisfactory at lower frequency range. This pickup is suitable for applications with high working frequency.

b) Piezo - resistive transducer : gives better performance at lower frequency range. Performance deteriorates at higher frequency. the pickup is available at cheaper costs as compared to the piezoelectric pickup. The transducer consists of a strain - gauge element.

Our application being a low frequency one (10-150 Hz) and due to lower cost, the strain gauge acceleration pick-up was preferred and has been used all along.

3.5 SIGNAL CONDITIONING

Devices used for conditioning the feed back signal conditioning from the acceleration pickup for data acquisition are as follows :

3.5.1 Dynamic Strain Amplifier

The signal coming from the strain-gauge acceleration pick-up is too weak to be acquired directly by a digital computer. For this reason a strong amplification is provided to this signal by the dynamic strain amplifier. The front panel provides course and fine balance for the bridge amplifier. There is a frequency selection Switch allowing the user to select the cut off frequency of the low - pass filter incorporated in the amplifier unit.

3.5.2 Peak - Detector :

Ours is a discrete time controller. That is, each control action is taken at a discrete instant of time. The output signal is a sinusoid whose amplitude is controlled (adjusted) on the basis of the peak acceleration value during the previous cycle of sinusoid. thus, the measured value of interest is the peak acceleration value of each cycle of vibration. This is captured at the output of the dynamic strain amplifier by this peak detector circuit.

3.6 DATA ACQUISITION SYSTEM

The data is acquired by a personal computer through add-on data acquisition card.

3.6.1 Personal Computer

An IBM PC compatible has been used. A fast processor, viz. 486 DX or pentium is required when the controller is operating in explicit calculation mode.

3.6.2 Data Acquisition Card

Dynalog's PCL-208 card has been used as the data acquisition card. This card has one 16 channel analog to digital converter with 12 bit precision. Two digital-to-analog converters and an 8254 programmable timer counter are also available on-board, which, in conjunction with the 1 MHz or 10 MHz internally provided clock signal, is used for time keeping purposes. Relevant details about the register addresses, jumper and switch positions are given in the Appendix.

3.7 FUZZY VIBRATION CONTROLLER STRUCTURE

Fig. 3.2 shows the complete structure of fuzzy logic vibration controller. the software has been shown enclosed in the outer block named 'fuzzy logic controller'. The other blocks are the hardware elements described in section 3.2 through 3.6.

The multiplier 'II' is a novel combination of software and hardware multiplication available in DAC (a concept devised by Dr. H.K. Verma and Dr. Vinod Kumar).

Both the digital to analog converters in the PCL-208 data acquisition card are used in the way illustrated in the figure 3.3. A reference voltage of $-V_{ref}$ volts D.C. will give an analog output range of 0 Volt to V_{ref} volts to the multiplying type DAC s mounted on board PCL - 208. The DAC 0 is given the internal reference of -5V D.C. Thus, its output can vary in the range 0 volt to +5 volt.

This output of DAC 0 is given as the reference voltage to the DAC1 as shown. Now the output of the multiplier is available at the output of DAC1.

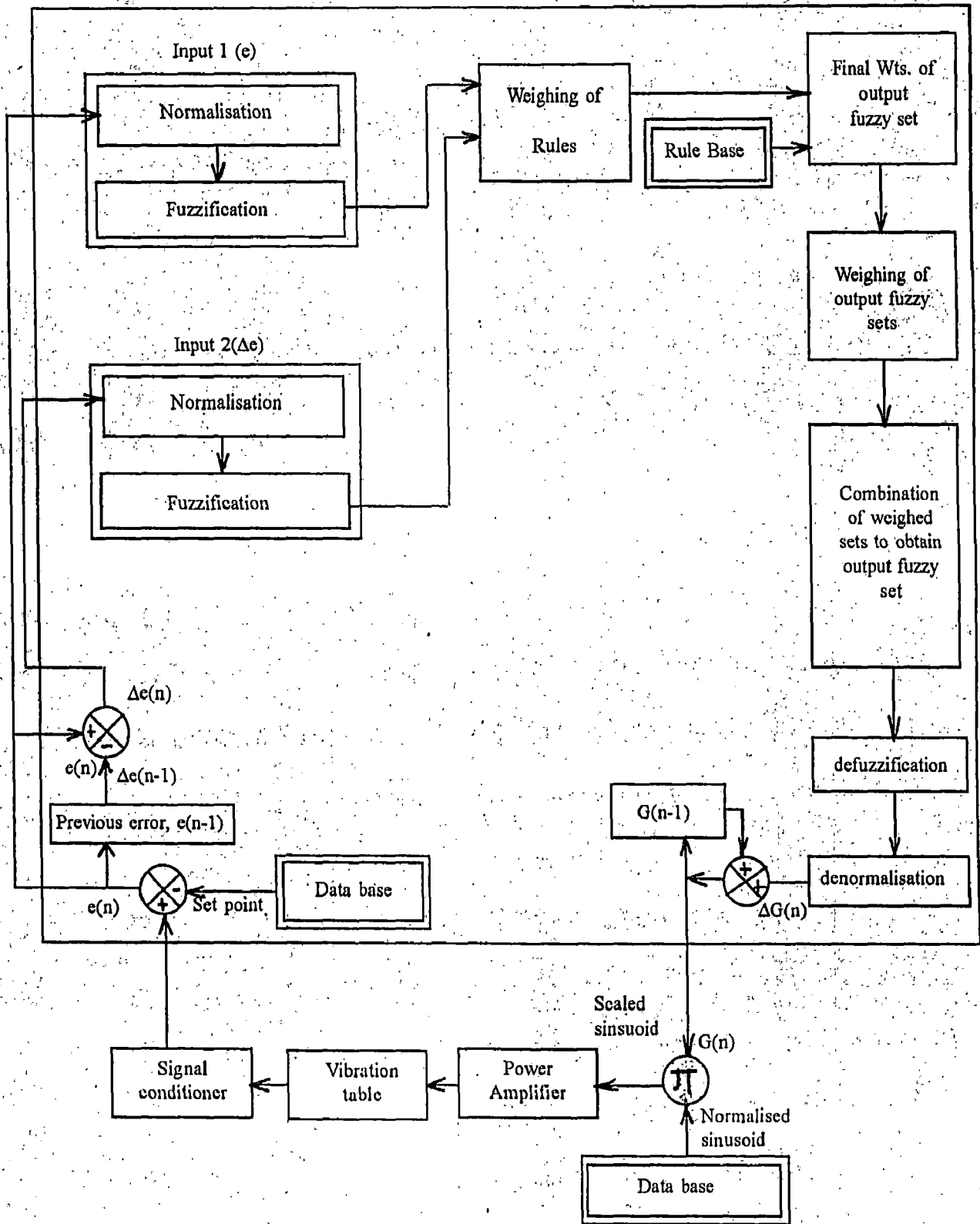


Fig. 3.2 Block Diagram of Internal Structure of Fuzzy Logic Vibration Controller

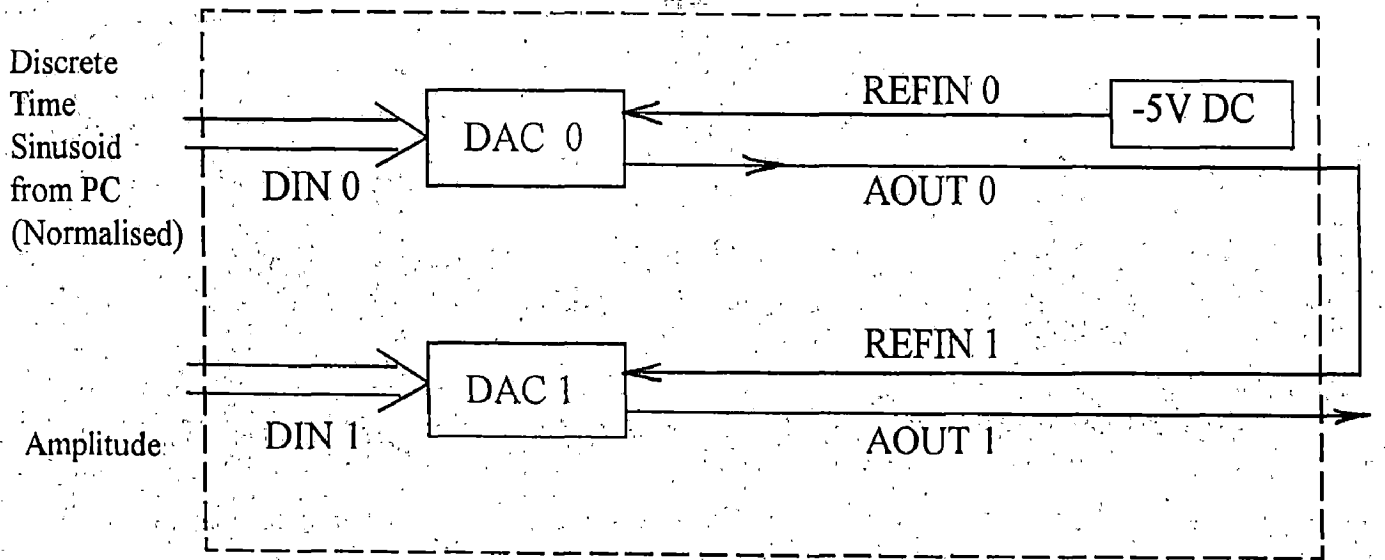


Fig.3.3 Multiplier details

A normalized sinusoidal signal is given out to the input of the DAC 0, while the factor by which it has to be multiplied (or scaled) is written out to DAC 1 digital input. the result is a sinusoid properly scaled available at AOUT1 as shown.

Inside the fuzzy controller block there are a number of inner blocks which are explained below.

a) Input 1 and Input 2

These two are input blocks, one corresponding to the error(e) and the other corresponding to the change of error (Δe). These contain functions for normalisation and fuzzification of respective input variable.

b) Weighing of rules

This function decides the weights by which each rule in the rule base is fired.

c) Inference Engine

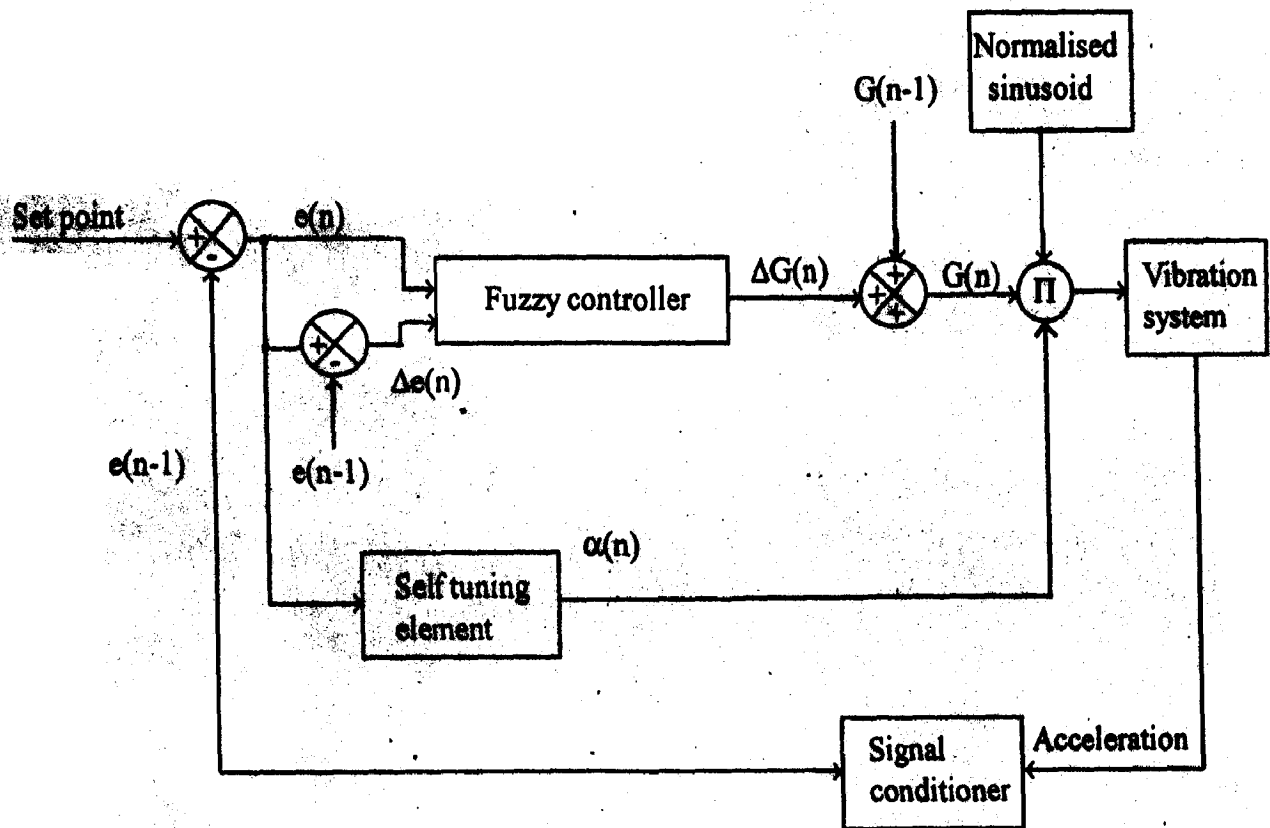
This involves many functions. Salient ones are shown in block named 'final weights of output fuzzy sets', 'weighing of output fuzzy sets', 'combination of weighed set to obtain output fuzzy set' and 'defuzzification'.

d) Denormalisation

This contains the key to deciding the exact strength of the control action. A high denormalisation factor would cause strong control, i.e. rapid correction. The final tuning of the controller is to be done through the adjustment of this parameters

3.8 SELF TUNING FUZZY CONTROLLER [6]

A minor modification in the peripheral structure of the controller lends a self-tuning controller that has the capability of uneven strength of control in the entire range.



LEGEND

- $e(n)$ = error, present sample
- $\Delta e(n)$ = error change, present sample
- $e(n-1)$ = error, previous sample
- $\Delta e(n-1)$ = error change, previous sample
- $\Delta G(n)$ = controller output gain adjustment
- $G(n)$ = controller output gain
- $\alpha(n)$ = output gain scaling factor

Fig. 3.4 Self tuning fuzzy controller

In our application this feature was implemented by making the self tuning scaling factor a linear function of error.

This element provides a wide range of possibilities of making the controller self-tuning. In this software the self tuning element is derived from the internal building blocks of the basic fuzzy controller ; hence enjoys all its flexibilities pertaining to its shape.

The block diagram of the self-tuning version of the fuzzy vibration control system is shown in fig. 3.4.

VIBRATION CONTROL SYSTEM

4.1 INTRODUCTION

This chapter discusses the software design aspects of the fuzzy vibration controller, the structure of which was discussed in section 3.7. The software has been implemented in C++ [7,8,9,10]. The development has been kept object-oriented in order to ensure maximum flexibility, expendability and clarity of the program. Chapter five will discuss the software details in continuation dealing with interfaces with hardware and user.

4.2 CLASS DESCRIPTIONS

Following user defined classes are created to form the fuzzy vibration controller software.

a) **Class Fuzzy_set** : This is the object representing a fuzzy set or membership function.

This contains information about the shape and position of a fuzzy set in the form of an array of points. Thus the structure of a membership function is stored in a piecewise linearised manner.

b) **Class Var** : This represents a fuzzy variable. This, therefore, consists of an array, named region [], of fuzzy-sets each representing a membership function (M.F.) of the fuzzy MFs variable. As many can be contained in the variable as required depending on the user. Default number of fuzzy sets is 3. Var contains computational facilities on the fuzzysets. It can also check its own errors as per the constraints of the software.

c] **Class Input:** This represents the input variable. It is derived from class Var and contains facilities of normalisation and fuzzification of the read in data.

d] **Class Engine :** This represents the inference engine of the fuzzy controller and involves all fuzzy logical computations and stores them in an output fuzzyset. This, thus, is derived from class Var and class Fuzzy_set. Engine can check for any errors existing in its settings as per the constraint of the software.

e] **Class Controller :** This is the outermost object representing the fuzzy controller. This is derived for the class Engine and contains many functions for operation of the controller.

f] **Class Menu:** This object contain the menu screens for user interfacing. The menu screens are built in text mode and give facilities to the user to use the software as required. main () function acts as the driver of the program. The relevant objects are created and function are called as per the commands of the user through this function.

4.3 CLASS HIERARCHY

Fig. 4.1 shows the class hierarchy as used in this software. Inheritance is used produce this class hierarchy. Also shown, within boxes each representing a class, the classes composing the respective class. Composition has been used wherever this structure is required.

Individual meaning a each class has been described in section 4.2. This section explains their significance in relation to one another.

The class Fuzzy_set is the innermost object.

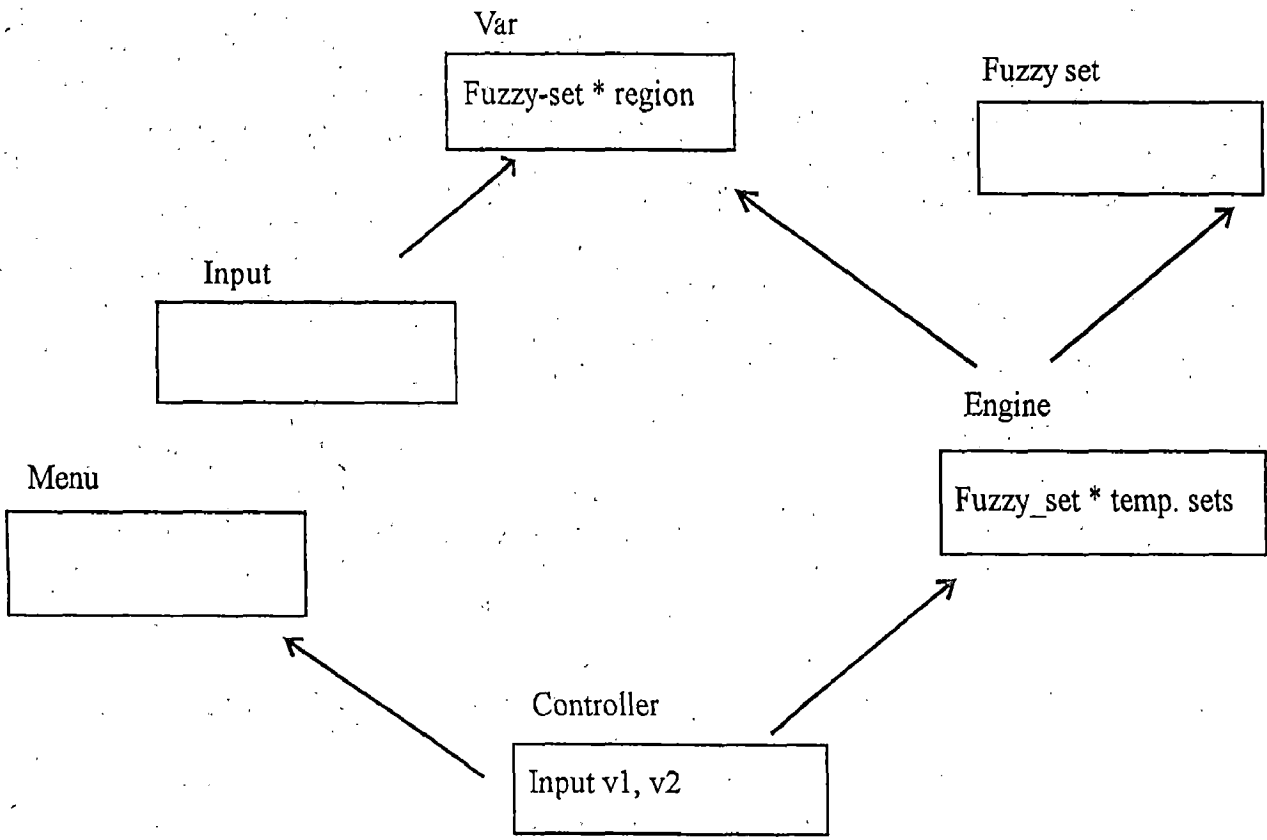


Fig. 4.1 Class Hierarchy

The class var is composed of an array of Fuzzy_set's . Each fuzzy_set represents the membership function corresponding to a linguistic concept pertaining to various values of that variable.

The class Input is a derived class of Var. It provides all facilities of conversion of the crisp input sample into a normalised form and then into a fuzzy variable (fuzzification). Thus fuzzy variable corresponding to this input is nothing but the base class Var.

The class Engine is derived from Var and Fuzzy_set. The fuzzy variable corresponding to the output variable is the base class Var. The output of the computation of one output control sample is primarily in the form of a fuzzyset. This information is stored in the base class Fuzzy_set at the end of each sample of control.

This class also contains an array of Fuzzy_sets with the name Fuzzy-set temp_sets[]. This is required as a temporary storage for the computation of the control output. Its significance is explained in the algorithm of truncating the output membership in section 4.7.

The class Controller is the object representing the complete fuzzy vibration control. Hence all facilities pertaining to it are accommodated in this class class Menu and class Engine are base classes of this class. Menu contains menu screens for menu driven user interfaces. Engine is the inference engine. The input values (the error and 'error change') are brought into the controller by the means of two objects V1 V2 which are of the type Input. They are contained in the class Controller.

Section 4.4 to 4.9 contain the algorithms in the implementation of the controller. When objects are created (i.e. when their construction functions are called) various activities occur steps as initialization. Description of these initialization steps are

foregone with . Only the computational algorithms are described (variable names used in algorithm may not conform with those in the software)

4.4 FUZZIFICATION (void input: : fuzzify())

ALGORITHM

1. Obtain the input sample of the variable, say x_1 .
2. Normalize the sample in the range [0,1].
3. Set counter $I = 0$.
4. Find the membership grade of x_1 to I th membership function (M.F.) defined for the fuzzy variable defined for x , and store as I th value of array mem_grade[].
5. IF $I =$ no. of M.F.S defined for x ,
THEN go to 6.
ELSE
Increment I by 1.
Go to 4.
6. Return.

EXPLANATION

Each fuzzy variable consists of a number of fuzzy sets. Fuzzification of a given sample of a variable means the determination of the membership grade of the crisp value to each of these fuzzysets. The above function does this after the acquisition of each input sample.

Once an input sample is acquired and normalised, this function is called for each input variable. The output of this procedure is an array mem_grade [] each element of which is the membership grade of the input sample to the corresponding fuzzyset of the given input variable.

4.5 COMPUTATION OF WEIGHTS OF RULES (Engine :: fill_FAM ())

ALGORITHM

1. Set counter I = 0.
2. Set counter J = 0.
3. Find the minimum out of I th element of V1. mem-grades [] and J th element of V2. mem-grades [] and store it in FAM [I] [J].

4. IF J = number of M.F.S defined for V2.

THEN

Go to 5.

ELSE

Increment J by 1.

Go to 3.

5. IF I = number of M.F.S defined for V1.

THEN

Go to 6.

ELSE

Increment I by 1

Go to 2.

6. Return

Explanation

The rule base of fuzzy system is often stored in the form of an array called fuzzy associative memory (FAM). The number of dimensions of a FAM is equal to number of input variables, each dimension corresponding to one input variable. Each element of the FAM is a fuzzy rule.

For example, let there be two input variables, say e and Δe . Each defined as a fuzzy variable containing three membership functions, LOW, MEDIUM and HIGH, coded as 0,1 and 2 respectively. Similarly, the output variable is also defined as a fuzzy variable containing three membership functions, LOW, MEDIUM and HIGH, coded similarly as 0,1 and 2 respectively.

Since the system has two input variables, the FAM (named `rule_base[]` in our software) is a two dimensional array having three rows and three columns. In a typical case `rule_base [] []` will look as shown in Fig.4.2.

Δe

	0	1	2
0	2	2	1
1	2	1	0
2	1	0	0

`rule_base [] [] = e`

Fig. 4.2 An example of fuzzy associative memory

For example, take the element rule_base [1] [2] = 0. This rule means that :

If e is MEDIUM (1) AND Δe is HIGH (2),

THEN output is LOW (0).

Now, the weight of this rule for a given value of sample of each input variable e and Δe depends on the membership of the given normalised values of e and Δe to MEDIUM M.F. of e and HIGH M.F. of Δe respectively.

Mathematically,

$$\text{Weight of rule_base [1] [2]} = \min (\mu_e [\text{MEDIUM}], \mu_{\Delta e} [\text{HIGH}]). \quad (4.1)$$

The min operator is allowed to be of any form as described in fuzzy logic literature. The choice depends more on the pragmatic applicability than on any theoretical base. In our case min operator has been used in its purest form, i.e.

$$\begin{aligned} \min (\mu_e (i), \mu_{\Delta e} (j)) &= \mu_e (i) \text{ if } \mu_e (i) \leq \mu_{\Delta e} (j) \\ &= \mu_{\Delta e} (j) \text{ if } \mu_e (i) > \mu_{\Delta e} (j) \end{aligned} \quad (4.2)$$

The rule that comes out of this process having a non-zero weight, is said to have been fired. The weights of each rule are separately evaluated and stored in an array identical in dimensions to the rule base [] []. In our software this variable matrix has been named as FAM [] [] (which should not be confused with FAM, which is represented by rule_base [] [] in our software). Each element of FAM [] [] contains, for each input sample, the weight of the rule represented by the corresponding element of the rule_base [] [] matrix. For each input sample, each element of FAM [] [] is computed as per the eq. 4.1.

This is accomplished by calling the function `fill_FAM ()` whose algorithm has been discussed in this section above.

4.6 COMPUTATION OF FINAL WEIGHTS

(void Engine : fill_temp_memberships (void))

Each rule in `rule_base [] []` suggests a fuzzy value (in terms of the fuzzy membership functions desired for the output variable). The truth value of that fuzzy value is the same as the weight of the given rule. One output variable fuzzy set may be suggested by many rules in the rule base. Each rule will do that with a different value of weight. Which has to be selected as the final weight of the given fuzzy set? The answer is 'max' operator.

Corresponding to each output variable fuzzy set, there is a list of rules which suggest that fuzzy set as their output. When the weight array `FAM [] []` is filled as described in section 4.5, the final weight of each output fuzzy set is computed as the maximum of the weights of all rules with that fuzzy set as the output.

The list of all rules corresponding to each output fuzzy set is contained in an array `tmg []`.

The algorithm for computation of final weights for each output fuzzy set is given below.

ALGORITHM

1. Set Counter I = 0.
2. Set counter J = 0.

3. Set MAX to the J^{th} rule corresponding to the I^{th} output fuzzy set.
4. Increment J by 1.
5. IF J = no. of rules corresponding to the I^{th} fuzzy set,

THEN
 Go to 6.

ELSE IF W_t (J^{th} Rule of I^{th} fuzzy set) > MAX,

THEN go to 3.

ELSE go to 4.
6. Increment I by 1.
7. IF I = no. of output fuzzy sets.

THEN go to 8.

ELSE go to 2.
8. Set temp_membership [i] to max.
9. Return.

The final weight of each output fuzzy set is contained in an array named temp_memberships [].

4.7 WEIGHING OF OUTPUT FUZZY SETS (void Engine:: truncate(void))

Once the final weights of output fuzzy sets has been decided by the process described in section 4.6, the next step is to weigh the associated output fuzzy sets with their respective weights.

For example, if fuzzy sets for the output fuzzy variable are defined as LOW(0), MEDIUM (1), and HIGH(2), their weights are stored in the array temp_memberships [].

If, temp_memberships [] = {0.5, 0.8, 0}.

then for the question, how much LOW, MEDIUM, and HIGH should the output variable be? the answer is : 0.5, 0.8 and 0 respectively. The actual output fuzzy set will thus be a weighed combination of these fuzzy sets.

Many strategies are available for weighing the fuzzy sets. We have employed the method of truncation of the fuzzy set. An example of truncation of a fuzzy set is shown in Fig.4.3.

For the example shown in Fig.4.3, the input fuzzy set is stored in a piecewise linear manner as $\{(x_0, y_0), (x_1, y_1), (x_2, y_2)\}$ where $y_0 = y_2 = 0$ and $y_1 = 1$.

On truncating the fuzzy set at 0.5 level the resultant fuzzy set will be stored again piece wise linearly as $\{(x_0, y_0), (x_3, y_3), (x_4, y_4), (x_2, y_2)\}$ where $y_3 = y_4 = 0.5$.

The algorithm to bring about this effect been provided below.

ALGORITHM

1. Obtain the point(s) where the horizontal line at the given memberships level, say μ , cuts the given fuzzy set. Let this point or pair of points be called α_μ .
2. The truncated fuzzy set will contain all points on left side of the member of α_μ which is on the rising edge of the original fuzzy set. Copy all such points to the resultant fuzzy set, if they exist.

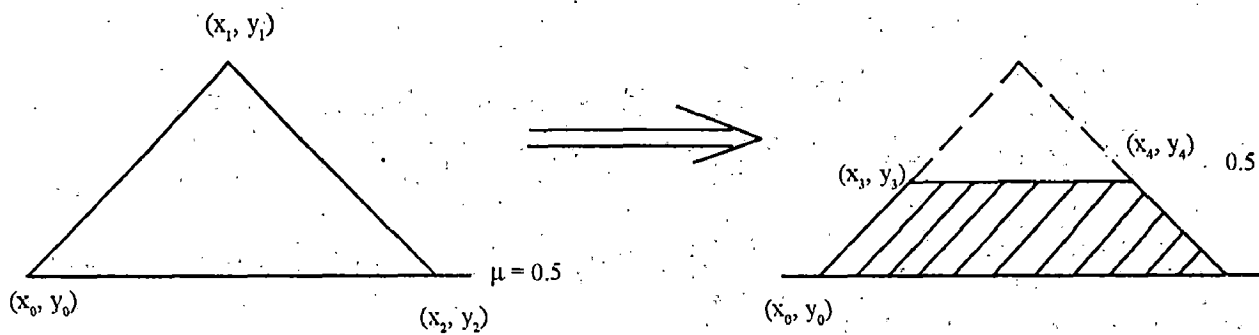


Fig. 4.3 Truncation of a fuzzy sets

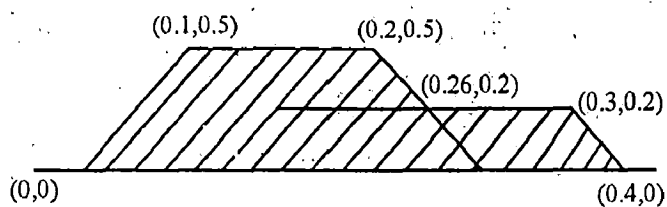
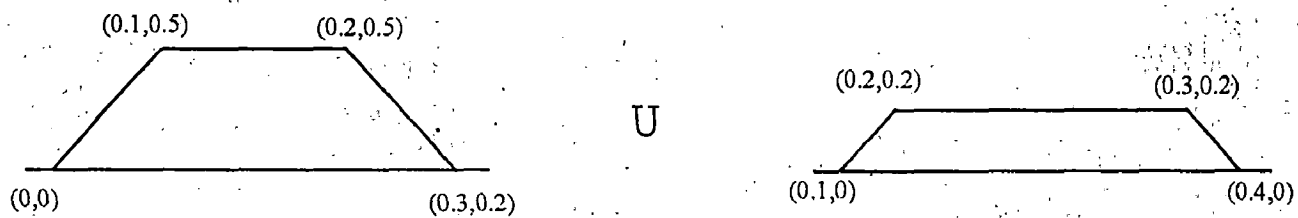


Fig. 4.4 Composite set of two truncated fuzzy sets

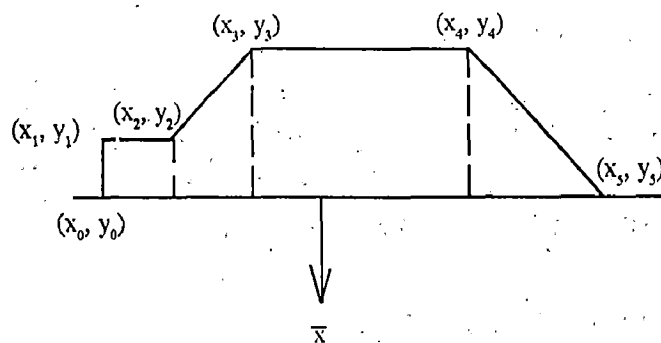


Fig. 4.5 An output fuzzy set partitioned for defuzzification

3. Copy the rising edge member of α_μ as the next member of truncated fuzzy set, if it does not coincide with any point already copied.
4. Copy the next member of α_μ if it exists to the resultant fuzzy set.
5. The truncated fuzzy set will contain all points on the right side of the member of α_μ which is on the descending edge of the original fuzzy set. Copy all such points to the resultant fuzzy set, if they exist.

During this process track has to be kept of the number of points copied into the resultant fuzzy set. Because this number is now the number of points contained in the truncated fuzzy set, it is an important parameter for further computations.

Each output fuzzy set is truncated. These truncated fuzzy sets are temporarily stored in an array temp_sets [].

4.8 OBTAINING THE OUTPUT FUZZY SET

(Fuzzy_set Fuzzy_set :: operator || (Fuzzy_set &))

and Fuzzy_set Fuzzy_set :: operator |= (Fuzzy_set &))

1. Copy the first fuzzy set of temp_sets [] into the final output fuzzy set.
2. Set counter I to 1.
3. Find the union of the present output fuzzy set and the Ith fuzzy set of temp_sets [] and store as new value of output fuzzy set.
4. IF I = no. of fuzzy sets in output variable
THEN

Go to 5.

ELSE increment I by 1

Go to 3.

5. Return.

Step 3 of the above algorithm is best understood with an example

Let fuzzy set $A = \{(0,0), (0.1,0.5), (0.2, 0.5), (0.3, 0)\}$

Let fuzzy set $B = \{(0.1,0), (0.2, 0.2), (0.3, 0.2), (0.4, 0)\}$

then fuzzy set $C = A \cup B = \{(0, 0), (0.1, 0.5), (0.2, 0.5), (0.26, 0.2), (0.3, 0.2)$
 $(0.4, 0)\}$

Graphically, this is shown in fig.4.4

Out of the weighed output fuzzy sets, the process of obtaining the final output fuzzy set is described in the above algorithm. With completion of the above procedure, the output control variable is available in fuzzy form.

4.9 DEFUZZIFICATION (void Engine :: defuzzify (void))

The control fuzzy variable is available. Now it has to be defuzzified.

There are a number of alternative methods of defuzzification as mentioned in section 2.3. We have chosen 'Centre of area' method of defuzzification. The output fuzzy set may typically look as shown in Fig.4.5.

The defuzzification process can be visualised with the help of the dotted lines partitioning the output fuzzy set into a number of trapezoids.

The defuzzified value can be obtained by the following process.

$$\bar{x} = \frac{\sum A_i x_i}{\sum A_i}$$

where $\sum A_i x_i$ = Summation of the products of the area of each trapezoidal part and the x co-ordinate of its centroid.

$\sum A_i$ = Summation of the area of each trapezoidal part.

The calculation of the normalised output control sample gets completed here. It is further denormalised and the new controller gain is calculated by adding the new control adjustment to the previous gain.

The details of hardware control software are discussed in chapter five.

INTERFACE SOFTWARE

5.1 INTRODUCTION

Design of the controller software was discussed in chapter four. This chapter continues the discussion on the software, dealing primarily with the interface of the controller software with the external world.

The controller software is connected with the external world on two sides, viz., on the system hardware side, and on the user side. Section 5.2 through 5.6 describe the interface with the hardware. Section 5.7 gives brief information about the user interface.

Certain relevant details of PCL-208 card are complementary to the information in this chapter. These are presented in APPENDIX[11].

5.2 INITIALISATION (void Controller::init_208 (int))

The PCL-208 card uses both its DAC's for outputting the control output signal; the ADC for acquiring the feedback signal from the peak detector; and the 8254 programmable timer - counter for time-keeping during the operation of the controller[12].

Thus the process of initialisation of PCL-208 card involves the following steps.

ALGORITHM

1. Set counter 1 in mode 3 BCD count.

2. Set counter 2 in mode 2 BCD count.
3. Load counter 1 register (16 bit) with appropriate value for the required output frequency.
4. Load counter 2 register (16 bit) with appropriate value for the required output frequency.
5. Select channel 0 of the input multiplexer.
6. Select pacer triggering.
7. Enable pacer triggering.
8. Send proper gain value to the DAC1.
9. Return

Step 5 makes the multiplexer scan through only its channel 0 input. This is because only one analog input is connected to its input, i.e. the feedback signal from the peak detector.

The internal clock of the PCL-208 card is used for all time keeping in the system.

Hence we require pacer triggering. This is done through step 6 and 7.

As described in section 3.7, the new controller gain is computed and is sent out to the DAC1. This is done through the step 8 of the above algorithm.

It is to be noted that step 8 is basically a function in the software to be called when the controller gain is to be renewed. Hence complete initialisation is not to be done each time controller output occurs.

5.3 ACQUISITION OF FEEDBACK DATA (void Controller::sample(void))

ALGORITHM

1. Initialise CURRENT STATUS as FREE.
2. Assign value of CURRENT STATUS to PREVIOUS STATUS.
3. Acquire the new value of CURRENT STATUS.
4. IF CURRENT STATUS = FREE and PREVIOUS STATUS = BUSY.
THEN
 Go to 5.
ELSE
 Go to 2.
5. Read the new value from ADC PORT.
6. Scale and Store properly.
7. Return

Basically, CURRENT STATUS read the status of the ADC. The ADC keeps certain status bit BUSY (1) when it is in the process of converting an analog value to digital value. It makes it FREE (0) when the conversion is completed. Step 3 and 4 keep polling this status bit and on tracking a BUSY to FREE transition of it, read in the newly converted data from ADC port.

5.4 OUTPUTING ONE CYCLE OF SINUSOID (void Controller :: sinusoid_out (int))

ALGORITHM

1. Set counter I to 0.
2. Detect one BUSY to FREE transition of ADC.

3. Output Ith sample of a cycle to DAC 0.
4. Increment I by 1.
5. IF I equal to 32
 THEN
 Go to 6.
 ELSE
 Go to 2.
6. Return.

There is an array of 32 elements (named out_sin[]) that contains normalised value all together constituting a complete cycle of a sine-wave. Thus each sample is spaced equally at $\pi/16$ radian. These values are to sent out to DAC 0 one by one after an appropriate spacing in time depending upon the output frequency required.

The time spacing is obtained by loading the counter registers with appropriate counts. The analog to digital conversion occurs at a corresponding rate. Thus the time is kept by reading in the ADC status.

5.5 THE COMPLETE CONTROL PROCEDURE

(void Controller :: real_control (mode))

The internal structure of the fuzzy logic vibration controller was described in section 3.7. The software design aspects were discussed in Chapter 4 and in Sections 5.2 through 5.4. This procedure coordinates the complete controller section by timely

invoking all the procedures discussed earlier. The algorithm of this software is given below.

ALGORITHM

1. Initialise PCL 208 card, output denormalising factors, GAIN to 1 etc.
2. Get the first feedback INPUT SAMPLE i.e. invoke procedure of sampling (section 5.3).
3. Set counter I to 0.
4. Set counter J to 0.
5. Output the current value of GAIN (section 5.2).
6. Output one cycle of sinusoid (section 5.4).
7. Assign INPUT SAMPLE to PREVIOUS SAMPLE.
8. Invoke procedure for entire fuzzy logic calculation (section 5.6).
9. Adjust the GAIN by the denormalised output of the fuzzy logic inference engine.
10. Increment J by 1.
11. IF J = No. of cycles of the current frequency to be output
THEN Go to 12
ELSE Go to 5.
12. Increment I by 1.
13. IF I = No. of frequencies to be output
THEN Return.
ELSE Go to 3.

5.6 COMPUTATION OF ONE CONTROL SAMPLE

(void Controller :: sample_out (void))

After the input samples have been acquired as e and Δe the computation of the next controller output sample is carried out as follows:

ALGORITHM

1. Normalise e and Δe .
2. Fuzzify (section 4.4).
3. Find rule weights (section 4.5).
4. Find the final weights for all output fuzzy sets (section 4.6).
5. Weigh the output fuzzy sets (section 4.7).
6. Combine to get final output fuzzy set (section 4.8).
7. Defuzzify.
8. Return.

5.7 USER INTERFACE

This section enumerates and defines the basic interfaces provided to the user to operate the controller software in various ways.

Also described in this section are those features which may need changes on further improvisation.

5.7.1 Facilities

(a) Control :

This gives an option of running the real-time control, the simulated controller, the real-time control in rapid mode, and self-tuning mode.

The real-time control initiates the control action with explicit computation.

The simulated controller was (and can be) used in the initial stage of the controller setting to test whether the controller is stable and has correcting tendencies.

On choosing the real time control in rapid mode, a look up table of an appropriate size (as defined by the user) is created. The look up contains precalculated control values. Hence time of explicit on line computation is saved. However, a lookup sufficiently large (for accuracy) and small enough (for speed and memory) is a design consideration.

On initiation of the control action, the controller merely fetches the values from the look up as per the value of e and Δe , thus saving time.

b) Changing the Controller Setting

All the controller setting-viz. the number of fuzzy sets contained in each fuzzy variable, their shapes, the number of fuzzy rules and the rules themselves, can be changed by the user through the user interface.

c) Saving the Settings

The controller settings can be saved to the disc by this facility in a data.file specified by the user.

d) Loading the Settings

Settings can be loaded from a properly documented data file by this command. The software has its self checking mechanism which becomes active whenever a change is

made or settings are loaded from the disc. Appropriate error message will be displayed on occurrence of any violation of constraints. The software will not enter control actions unless existing faults are corrected.

5.7.2 Limitation

The most important limitation of the software is that it is build for only two feedback quantities, i.e., error and change in error. If any change in the number of variables is intended to be made, the same is not allowed. Moreover, no superficial or concentrated change will being about the necessary modification. A number of algorithms assume the number of variables as two. Any change in this number will have to be made at all such places.

RESULTS AND CONCLUSIONS

6.1 PERFORMANCE RECORD

The vibration table was subjected to vibrations at frequencies ranging from 10Hz to 150 Hz at steps of 10 Hz. Each frequency was maintained for 5 sec. The set point was kept constant at $1g$ (9.81 m/s^2) peak, for all cycles. The recording was done by making necessary additions to the controller software and using the same hardware as for control.

Fig. 6.1 shows the plot of the recorded data. The upper plot is the complete point by point plot. The lower plot is the plot obtained by averaging the response over 64 samples windows. Plotting of the graphs was done with DASylab package.

6.2 DISCUSSIONS

Certain important observations can be made on above record. An attempt is made below to provide explanation for these observations.

(A) Expansion of frequency scale toward higher frequency range :

The vibration table was subjected to each operating frequency for a constant duration, i.e. 5 sec. The number of cycles output for each frequency is, therefore, proportional to its value. To accommodate all the samples they have been kept equidistant in the plot. Thus, plot for a higher frequency spans over a larger range on the abscissa as compared to that for a lower frequency.

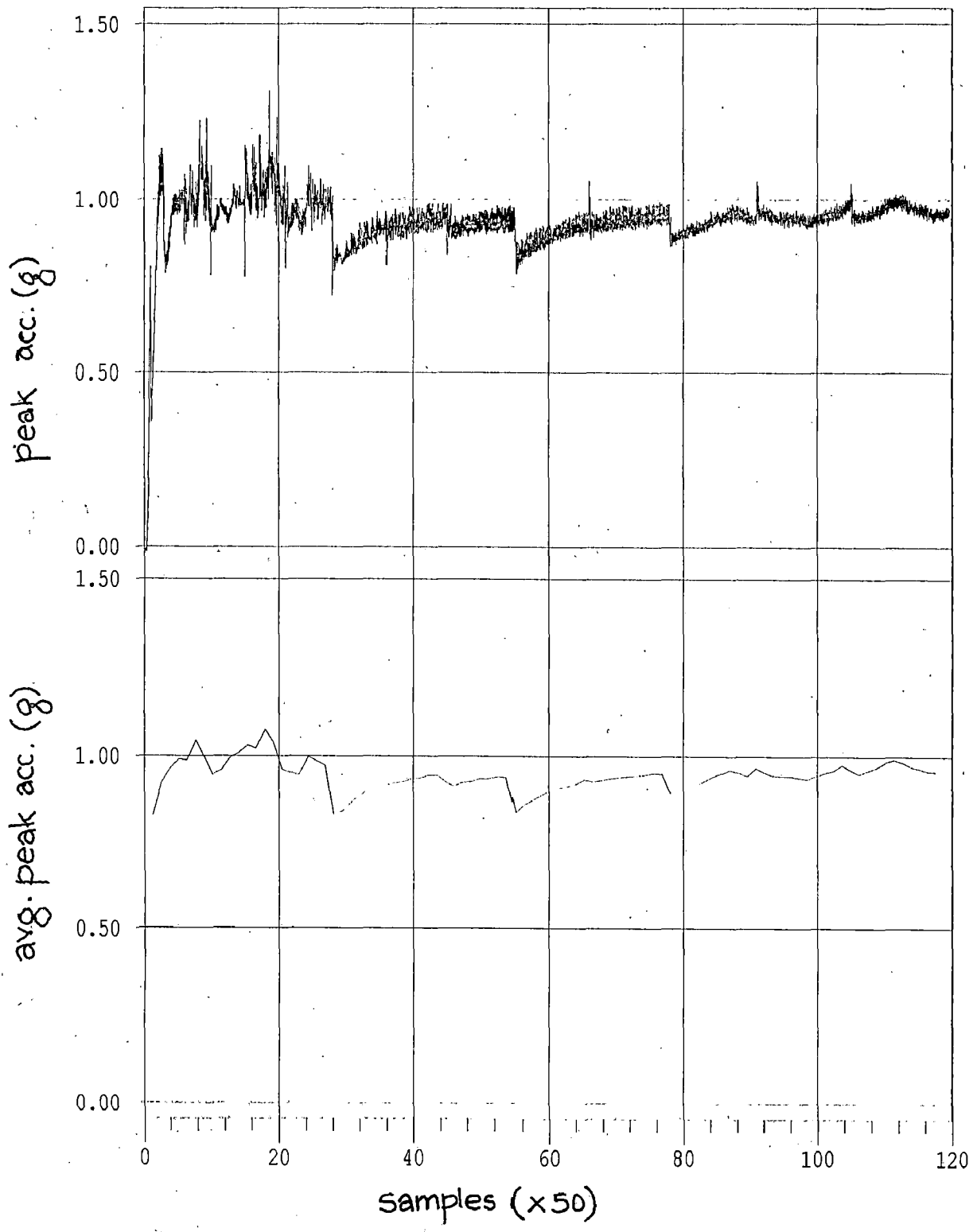


Fig. 6.1 Performance Record of Fuzzy Vibration Controller

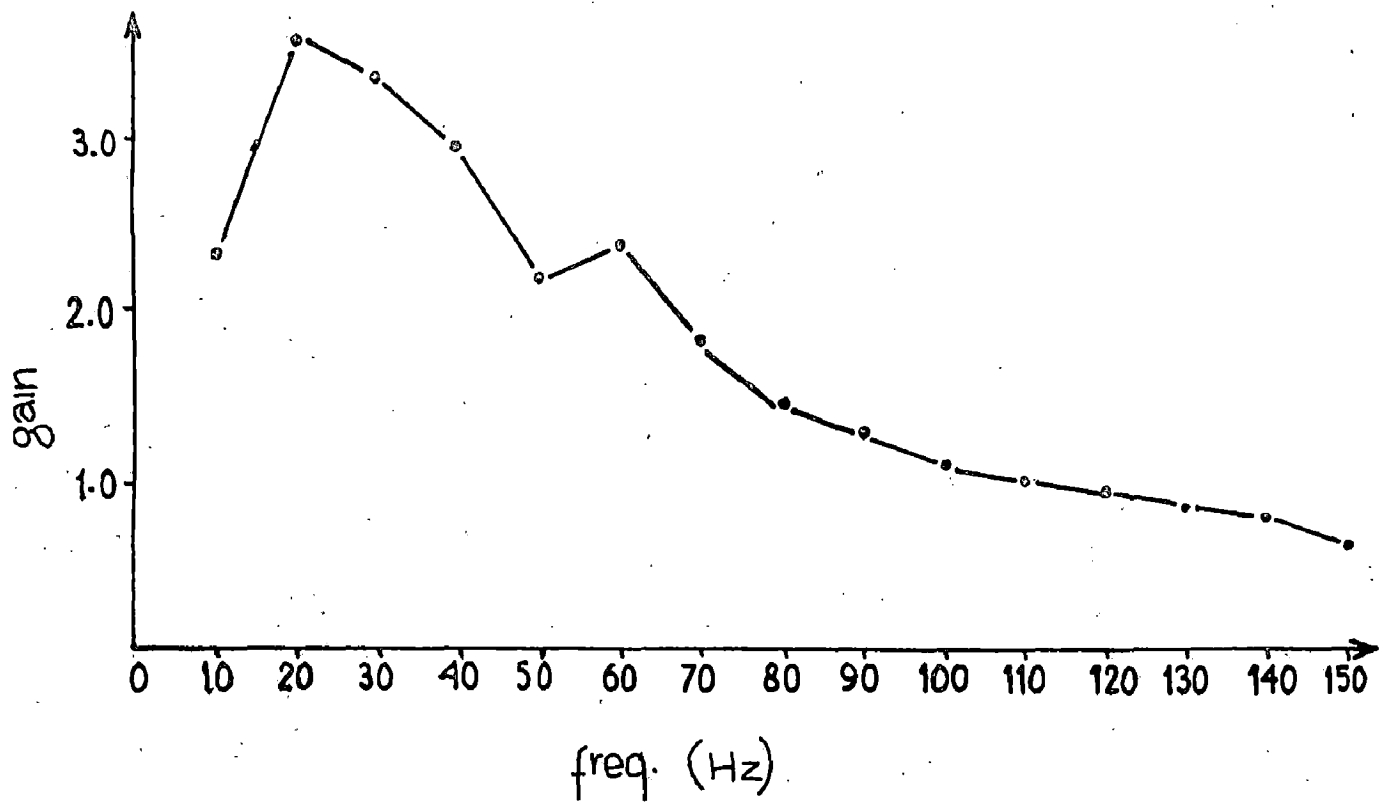


Fig. 6.2 Frequency Response of Vibration Table

(B) Plotting of average response :

This plotting was done to show the general performance of the controller in a clear way. While the complete plot is indicative of the controller's ability to limit oscillation (thickness of the envelope), the average plot shows the trend of the controller in keeping close to the setpoint.

(C) Controller performance at low frequencies :

Oscillations upto 25 percent are observed. The frequency response of the vibration table, presented in fig.6.2., exhibits two peaks -- at around 20 Hz, and another at around 50 Hz. There is a trough (dip) observed at around 40 Hz. Thus, the overall gain of the vibration table varies significantly over this frequency range, which is largely responsible for large oscillations in this region.

(D) Controller performance at high frequencies :

Controller performance visibly improves towards higher frequency. The reasons could be :

- (a) No significant change in the gain of the vibration table is present above 50 Hz.
- b) No crests or troughs are present in the higher frequency range.
- c) The step - change in frequency at higher frequency range is not as big as it is at lower frequency range, in proportion to the current frequency. Hence this does not act as a major destabilising factor as this does at lower frequencies.

218278



(E) Effect of step change in frequency:

The step change of 10 Hz every 5 seconds appears to have an observable effect on controller performance. Except at a few places, the peak acceleration shoots up or (usually) down near the points of frequency changes. The possible reasons are:

- a) The step change in frequency, especially when its ratio to the value of the current frequency is large, represents the introduction of higher frequency transients. They are likely to affect the output.
- b) Over the 5 seconds' running time at any constant frequency the output of the controller starts settling to a value as required to keep the peak acceleration close to the set value. However, the gain value with which the next frequency starts, remains the same, irrespective of the fact that there could be a significant difference in the vibration table gain at those two frequencies. Hence, the overall forward path gain undergoes a step change with the change in frequency. This, with full likelihood, may deviate the vibrations from their set value.

6.3 CONCLUSION

A PC based generalised fuzzy logic vibration controller has been designed and implemented in real time on the vibration test system for energy meters available in I and SP laboratories of EED, UOR.

The membership functions have been defined in a piecewise linear manner instead of any mathematical equations. The objective being to provide maximum flexibility on the user level to access, and modify the controller parameters, a comprehensive menu-driven user interface has been built. It gives the user a facility to change the controller settings, to

load them from and save them to the disc. An error-detecting mechanism has been incorporated that provides safety against erroneous parameters which violate the constraints of the software.

The controller has been tuned by adjusting the controller denormalisation factor. The record obtained as the controller after the tuning validates the controller in this application.

6.4 SCOPES OF FUTURE WORK

The 'generalised' design of the controller opens a number of possibilities of improvements in the controller designed in this dissertation. They are briefly discussed below.

6.4.1 General fine tuning.

The tuning that was attempted during the dissertation work was that of the denormalisation factor. However, there are a number of other possibilities by which the controller - tuning can be done.

- a) by changing the number of membership functions (M.F.S.) in each variable.
- b) by changing the shapes of the M.F.s.
- c) by changing the rule - base.

All this can be done by changing the source code or conveniently and safely through the menu of the software.

6.4.2 Incorporation of integral component

This software only uses error (e) and change in error (Δe) as the input variables. To improve the controller performance, summation of error (Ee) may also be used in addition. Unfortunately, that will significantly increase the computation time, which may become a major consideration in deciding whether to include this component or not.

6.4.3 Increasing the number of feedback variables

Since, no explicit modelling of the vibration table has been attempted, the relationship between the output (vibration) and various parameters (e.g. voltage, current or VA output of the power amplifier) may prove to be of importance. Hence arrangement may be made to feedback some of these quantities to the PC and make use of them as feedback variables. Again the computation time of the controller may pose a limit to the number of feedback variables that can be used.

6.4.4 Learning or adaptive controller

Incorporation of learning tools like artificial-neural network may prove to be very useful in converting this controller into a learning system. The internal structure of the controller provides sufficient flexibility to allow this improvement.

6.4.5 Graphic User Interface

Even through the controller software is fully menu driven, the user interface is all in text mode. A graphic user interface can make it more user-friendly.

REFERENCES

1. Nagrath, I.J. and Gopal, M.,
Control Systems Engineering, Second Edition
New Age International (P) Ltd., Publishers 1996.
2. Ogata, K.,
Modern Control Engineering, Second Edition
Prentice-Hall of India Pvt. Ltd., 1995.
3. Dr. H.K. Verma and Dr. Vinod Kumar
Computerised Vibration Test System for Energy Meters
Manual, 1996.
4. Klir, G.J. and Yuan, B.,
Fuzzy Sets and Fuzzy Logic; Theory and Applications
Prentice-Hall of India Pvt. Ltd., N. Delhi, 1997.
5. Ross, T.J.,
Fuzzy Logic with Engineering Applications
McGraw Hill, Inc., 1997.
6. Mudi, R.K., and Pal, N.R.,
A Self Tuning Fuzzy P.D. Controller
IETE Journal of Research, July-October, 1998.

7. Stroustrup, B.,
The C++ Programming Language, Third Edition
Addison - Wesley, 1998.
8. Lafore, R.,
Object Oriented Programming in Turbo C++
Galgotia Publications Pvt. Ltd., 1997.
9. Hubbard, J.R.,
Theory and Problems of Programming with C++
Schaum's outline series, McGraw-Hill, 1996.
10. Kernighan, B.W., and Ritchie, D.M.,
The C Programming Language, Second Edition,
Prentice-Hall of India Pvt. Ltd., N. Delhi, 1996.
11. Dynalog
Owner's Manual, PCL-208, Data Acquisition Card,
Dynalog Microsystems Pvt. Ltd.
12. Hall, D.V.,
Microprocessors and Interfacing
Tata-McGraw Hill Publishing Company Ltd., 1995.

Dynalog's PCL 208 data acquisition card has been used for interfacing purposes.

These contain the following two :

- a) Reading in the feedback data from the peak detector through ADC.
- b) Outputting sinusoidal output of appropriate amplitude to the vibration exciter through the DACs.

The method and algorithms used for both the actions have been described in chapter three, four and five. given here are certain relevant data related to the addresses and jumper positions of PCL 208, which are of interest in our application. Also given is the block diagram of PCL 208 card in fig. A1[11].

1. ADDRESS SPECIFICATION

When PCL 208 card is added - on to the PC buses, it occupies 16 consecutive I/O port addresses, viz., $BASE + 0$ to $BASE + 15$. The value of $BASE$ depends on the DIP switch $DIP SW2$ on board. We have set $SW2$ to have $BASE = 300 H$.

The significance of the addresses occupied by PCL 208 are as follows

PCL-208 BLOCK DIAGRAM

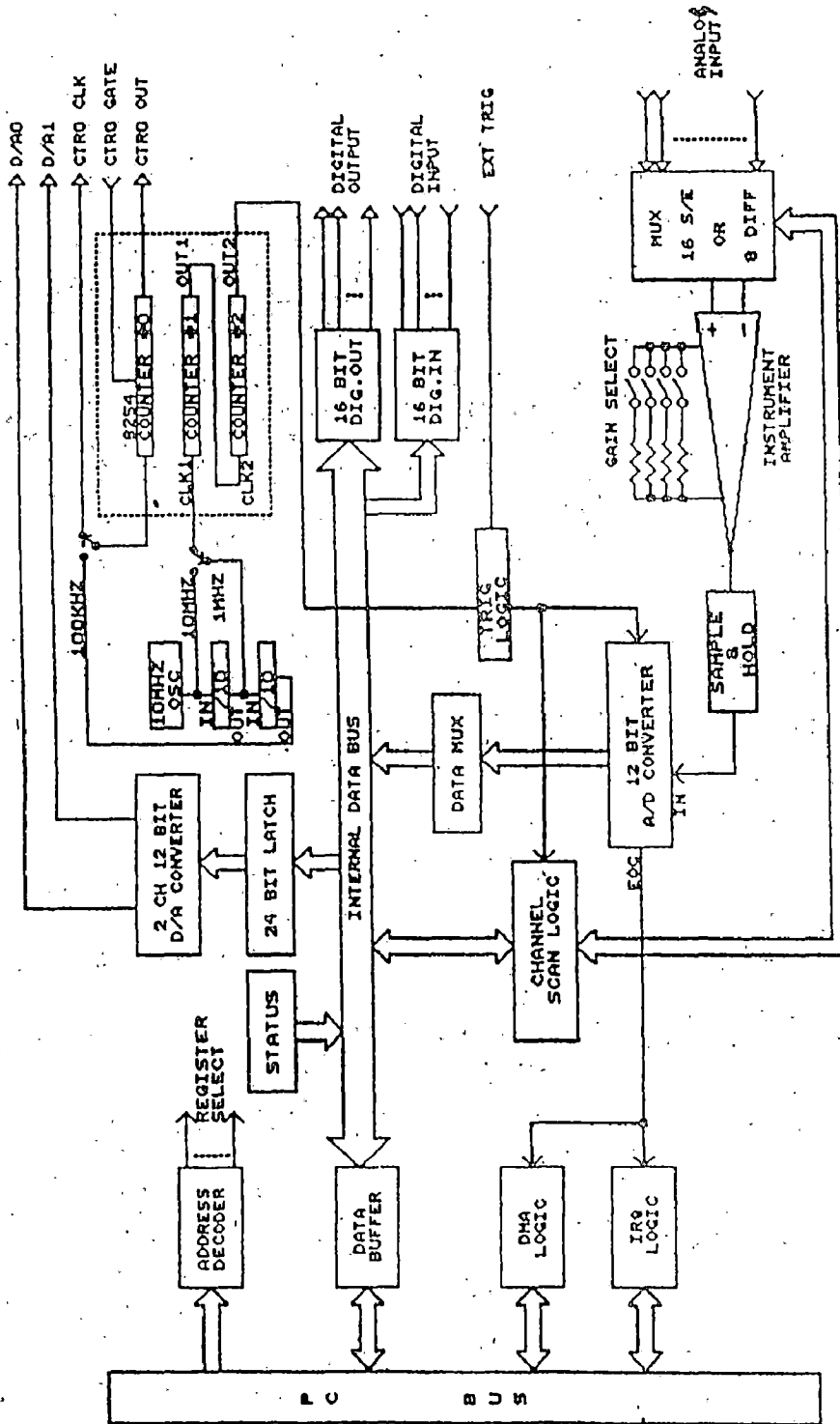


Fig. A1 : Block Diagram of PCL.208 Data Acquisition Card

Location	Read	Write
BASE + 0	A/D Low byte and channel number	Software A/D trigger
BASE + 1	A/D high byte	N/A
BASE + 2	MUX Scan channel	MUX scan channel
BASE + 3	D/I low byte (DI0 - D17)	D/O 0 low byte
BASE + 4	N/A	D/A 0 high byte
BASE + 5	N/A	D/A 0 low byte
BASE + 6	N/A	D/A 1 high byte
BASE + 7	N/A	D/A 1 high byte
BASE + 7	N/A	D/A 0 high byte
BASE + 8	PCL 208 Status	Clear interrupt request
BASE + 9	PCL 208 Control	PCL 208 control
BASE + 10	N/A	Counter enable
BASE + 11	D/I high byte (D18-D115)	D/A 0 high byte
BASE + 12	Counter 0	Counter 0
BASE + 13	Counter 1	Counter 1
BASE + 14	Counter 2	Counter 2
BASE + 15	N/A	Counter control

2. JUMPER SETTING

SW3 : For 16 channels analog input mode.

SW1 : For 1MHz timer clock selection.

JP1: To set DAC 0 reference voltage to -5V DC onboard supply

JP2 : To set DAC 1 reference voltage to external supply.