CONTROLLER DESIGN VIA H_∞ OPTIMAL CONTROL, QUANTITATIVE FEEDBACK THEORY AND FUZZY LOGIC CONTROL: AN APPLICATION TO LOAD FREQUENCY CONTROL

This dissertation has been submitted to the department of Electrical and Electronic Engineering, University of Cape Town, in partial fulfilment of the requirements for an M.Sc. in Electrical Engineering.

By

C D Boesack

29 September 2000

ACKNOWLEDGEMENTS

The following are acknowledged for their patient guidance and assistance during the course of this project.

Our Father in heaven for wisdom and understanding, to Him be the glory and honour.

Prof. M Braae for his patient supervision and guidance during the progress of this project. His supervision is gratefully acknowledged.

Prof. A Petroianu and Prof. S Maclaren, for their supervision during the initial stages of the project.

Chris for his assistance in the machines laboratory in setting up the experimental generating unit.

Eskom for financial support and giving me the opportunity to further my studies by means of a masters degree in electrical engineering. Mr. Graeme Chown for his help and assistance and all Eskom staff who contributed to the success of this project.

My parents for their continual encouragement and support throughout the during of my studies, and without their sacrifice, my education would have been void.

SYNOPSIS

This thesis describes the application of robust controller design techniques namely H_∞ optimal control, fuzzy logic control and quantitative feedback theory to the load frequency control problem of power systems. It thus forms a comparative study of these design methods. Load frequency control is the closed loop control of electrical generating units to regulate the system frequency at its nominal value in the presence of load disturbances. Load frequency control also includes the regulation of the tie–line interchange powers.

Robustness is a closed loop characteristic defined as the ability of the control system to perform satisfactorily in the presence of model uncertainty and unmodelled system dynamics. These (model uncertainty and unmodelled system dynamics) are inherent to any physical process, and the performance of the control system depends to a great deal on the ability of the controller to account for these uncertainties. The above controller design methods characteristically incorporates uncertainty into the design of the controller. Thus with its application to power systems, where the dynamics of the system varies continuously, the above controller strategies forms a viable solution to the load frequency control problem.

Results obtained from this study indicate that similar transient response characteristics can be obtained for all the above-mentioned controller structures. The distinguishing factor between these controllers are their different design approaches and the techniques used in obtaining the resultant controller. H_∞ optimal control is a robust frequency domain controller synthesis method, however it suffers from the disadvantage that the controller zeros cancels all poles of the plant, which may be undesirable. This can be overcome by using the two-degree of freedom H_∞ problem formulation or the loop shaping approach proposed by McFarlane and Glover. Despite this, the application of H_∞ is well documented.

Quantitative feedback theory (QFT) incorporates uncertainty parametrically as plant templates on the Nichols chart, where performance specifications are represented as frequency domain bounds. The QFT technique, being a two-degree of freedom design method effectively maintains tracking performance and disturbance rejection properties over a wide operating region. One disadvantage however, to this method, is that it requires knowledge of frequency domain loop shaping which may prove difficult to the novice designer.

Common to H_∞ optimal control, quantitative feedback theory and PI control is that of a required mathematical model of the plant. In studies for load frequency control an accurate model of the system may not be readily obtained, which motivates the use of a fuzzy logic controller. Due to its attributes, the fuzzy logic controller exhibits favourable performance in regards to parameter robustness and in the ease with which the controller is designed.

Thus based on the results presented in this document, the conclusion is reached that all the above controllers are acceptable for load frequency control. It is thus recommended that these controllers be considered for their application to load frequency control. This is because of their versatility and ability to stabilise the system over a wide operating region.

TABLE OF CONTENTS

LIST OF FIGURES

TITLE **PAGE NUMBER**

LIST OF TABLES

CHAPTER 1

1. INTRODUCTION

Modern robust control affords the opportunity to design and synthesise feedback controllers able to regulate a given process acceptably over a wide operating region as is possible. This includes the effective closed loop control of the process in the presence of model uncertainty and in the presence of random disturbances. Controller design methods encapsulated under the term modern robust control includes H_∞ optimal control, fuzzy logic control and quantitative feedback theory. These design techniques are renown for their ability to deal effectively with model uncertainty (Maciejowski, 1989; Kandel and Langholz, 1994), which cannot be neglected in any physical process, hence motivating feedback control.

This project describes the application of these controller design methods including the classical PI controller to the load frequency control problem of power systems. Thus a comparative study of the performance of these controllers with its application to load frequency control is thus pursued. For this purpose an experimental laboratory motor generator unit emulating the operation of power system control in regard to frequency regulation is constructed and the designed control law is implemented digitally.

Load frequency control can be described as the closed control of electrical generating units to maintain the system frequency at its nominal value. Due to load imbalances or the power mismatch between the generated power and that of the demanded load, frequency deviations will occur. It is thus essential for the effective operation of the system and that of the connected load that the frequency remain at its nominal value or within certain tolerance bands.

H_∞ optimal control is a frequency domain controller synthesis method, where an objective function is minimised in order to satisfy multiple control objectives.

Similarly quantitative feedback theory is a frequency domain controller design and analysis tool, where the design is performed on the Nichols chart. Fuzzy control however, describes the characteristics of the system based on a fuzzy model and calculates the required control by a set of linguistic rules.

Thus the objectives of this project can be summarised as follows.

1. To review the operation and control of electrical power systems, concentrating on the load frequency control problem as described in literature.

2. To study modern controller design techniques such as H_∞ optimal control, fuzzy logic control and quantitative feedback theory. For comparative purposes the classical PI controller is also examined.

3. To implement these controller methods and comment on their effectiveness to load frequency control and to analyse their performance comparatively.

4. To draw conclusions and present recommendations based on the results obtained during the course of the project.

The organisation of this report is as follows:

Chapter 2 introduces the subject of load frequency control, detailing the relevant theoretical concepts of power system control and operation. It further expounds on the control methodology employed for load frequency control.

Chapter 3 describes H_∞ optimal control, presenting the mathematical framework for the synthesis of the controller. It thus serves as a tutorial section where its emphasis is on the H_∞ mixed sensitivity control problem, where the designer is allowed to design for multiple performance objectives. These include stability, tracking and disturbance rejection specifications.

Chapter 4 focuses on the operation and design of fuzzy logic controllers. It presents the various design stages involved in the design of the controller, from fuzzification and inferencing to defuzzification. This section then concludes by commenting on the stability of the fuzzy logic controller.

Chapter 5 concentrates on the robust controller design technique namely, quantitative feedback theory and forms the final section dedicated to the theoretical aspects of the above control laws. It defines quantitative feedback theory as a frequency domain controller design method.

Chapter 6 thus embarks on the application of these methods, describing the experimental laboratory unit and the implementation of the designed controllers. It further presents the design of the controllers followed by simulation and actual response data. This section concludes by presenting a comparative study of these methods and making appropriate inferences in each case.

Chapter 7 concludes the project by stating the advantages and disadvantages of the designed controllers and their effectiveness to load frequency control, followed by recommendations based on the content of this report.

CHAPTER 2

2. LOAD FREQUENCY CONTROL

This section describes the operation of load frequency control by taking the form of a literature survey, detailing the basic conceptual ideas involved in the efficient operation of power system control. Electrical power systems are concerned with the conversion of energy from one form to another, namely from stored energy in fossil fuels (coal, oil), kinetic energy of water (hydro-plants) or via nuclear energy (nuclear power plants) to electrical energy where it is utilised by power consumers (Miller and Malinowski, 1993). Electrical power systems consists primarily of turbine governors, turbines, alternators, transmission lines, transformers and loads connected to the power grid at various stages of the power system (Glover and Sarma, 1994). It is thus vital to the proficient operation of the power network that the system frequency, real and reactive powers and voltage levels remain constant or at desired specifications at any given point in time.

In order to regulate these quantities power utilities employ the use of supplementary or supervisory controllers to achieve desired system performance specifications (Handschin and Petroianu, 1991). Automatic generation control (AGC) forms part of this control methodology where it forms a corner stone in the operation of interconnected power systems in regard to frequency regulation. AGC aims at operating the generating units within the power system at minimal cost, utilising the least control effort to achieve this objective. Thus there are two distinct elements comprising automatic generation control, namely a control objective (frequency regulation) and secondly, an economic objective (minimisation of expended cost). The control objective is known as load frequency control (LFC) and the economic objective is known as the economic dispatch problem (ED). The following section briefly describes the constituents of AGC.

2.1. LOAD FREQUENCY CONTROL

Load frequency control in fulfilment of the AGC objectives regulates the system frequency and tie–line interchange power deviations to zero. The requirements for such a controller is that satisfactory transient performance must be maintained and that the closed loop system must satisfy the NERC criteria. NERC (North American Electric Reliability Council) is an international power system regulatory body (NERC operating manual). The main input to the LFC controller is the area control error (ACE) defined as the sum of the tie–line power deviation and the frequency deviation multiplied by a frequency bias factor β. Mathematically the ACE error is defined as,

$$ACE = \Delta P_{tie} + \beta * \Delta f$$

where ΔP_{tie} = Tie-line power deviation [MW].
Δf = frequency deviation [Hz].
β = frequency bias factor $\left[MW/_{0.1Hz}\right]$

The diagram shown in figure 1 below illustrates the operation of the load frequency controller within an interconnected control area. The LFC controller regulates all generating units operating under the instruction of AGC where the proportion (determined by participation factors) of the control signal shared by all generating units is calculated by the economic load dispatch routine. The control input to the plant, the generating unit, is the speed reference input of the turbine governor. The operating characteristic of the governor enables the system frequency to swing in unison throughout the entire control area, thus the power system operates synchronously (Wood and Wollenberg, 1996). It is however more important to note that synchronisation is achieved by the positive synchronising torque (ie. slope of sin(δ)).

As shown in figure 1 all tie–line interchange powers leaving the control area are summed to form the total tie–line interchange power. The deviation of the tie-line

power from its nominal value is added to the system frequency deviation to form the ACE error of the individual control area. P_{tie} and f represents the tie-line power and system frequency respectively, the subscript 0 denotes the nominal value. It is thus the function of the load frequency controller to drive this error to zero. The following section describes various controller design methods that have been successfully applied to the design of the LFC controller.

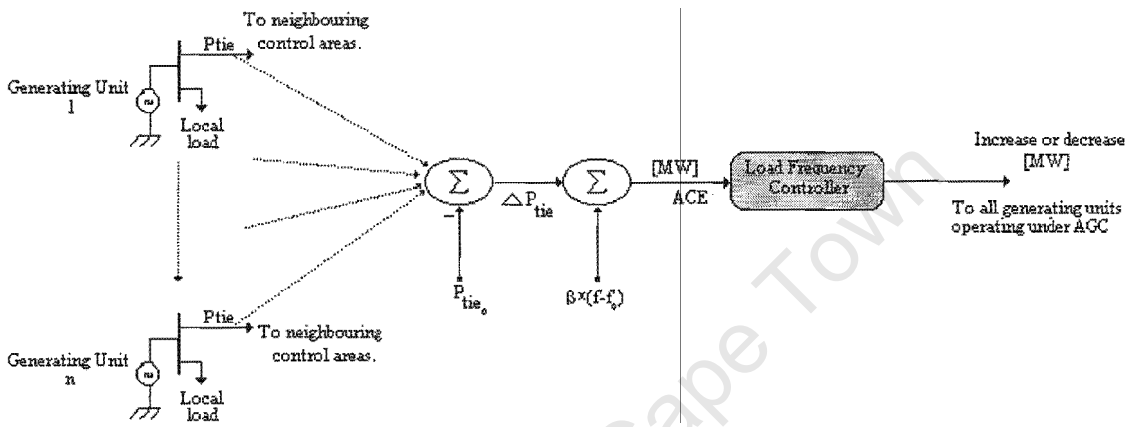


FIGURE 1: LFC FOR A SINGLE CONTROL AREA.

2.1.1. OPTIMAL CONTROL

Optimal control is a synthesis method aimed at minimising a performance index in which the state vector and control inputs are weighted. Appropriate selection of these weighting matrices ensures that the desired closed loop characteristics are achieved. In LFC we want to minimise the control effort utilised in order to reduce cost, therefore in optimal control this is conveniently accomplished by weighting matrices Q and R on the state vector and on the control input vector respectively.

Fosha and Elgerd (Fosha and Elgerd, 1970) approached the load frequency control problem via optimal control. Simulation studies performed on a two area interconnected power system reveals an improvement in the damping of the system over existing control strategies present at the time, being the classical optimisation

procedure of finding the optimal integral gain and the frequency bias factor β, known as tie-line bias control.

Considering robustness properties Wang et al. (Wang and Zhou and Wen, 1993) incorporated uncertainty matrices in the state space description of the system model to represent model parameter variations. Studies were conducted on a single control area with the inclusion of generation rate constraint. Results showed that the stability of the system could be guaranteed in the presence of model parameter uncertainties.

2.1.2. ADAPTIVE CONTROL

Most controller design methods utilise a mathematical model of the plant to be controlled. However this model may not accurately represent the dynamics of a particular plant or may vary with time, leading to a controller design that is not able to stabilise the system in the presence of these model uncertainties (Åström, 1983). In view of this, adaptive control is a viable solution to plants where there are considerable model parameter uncertainties (Pierre, 1987; Yamashita and Miyagi, 1991).

Adaptive control is a control scheme in which the controller attempts to correct itself due to variations in model parameters. The self-tuning regulator (STR) forms part the adaptive control structure in which input and output measurements are used to obtain a model of the plant. Thus as the model of the plant varies the controller will adjust in such a way as to compensate for these changes. The operation of the STR is to firstly estimate or identify the model of the plant followed a means of adjusting the controller parameters leading to the required performance specifications (Maciejowski, 1989).

The motivating argument for the application of adaptive control to power systems and more specifically to LFC is that the plant parameters are constantly changing. These changes are the result of loads being disconnected, equipment being switched on, new

installations or power failures in certain areas of the control area. If the controller cannot account for these variations dire consequences can result.

I. Vajk et al. (Vajk, Vajta, Keviczky, Haber, Hetthessy and Kovacs, 1985) has reported successful implementation of an adaptive regulator meeting the requirements for load frequency control. It is reported that improved dynamics and a more efficient control effect under wide operating conditions results. Yamashita and Miyagi (Yamashita and Miyagi, 1991) used the STR to account for the interaction between the power/frequency and reactive power/voltage control loops. The general model for STR is in the form of a difference equation where the parameters are updated recursively usually by least squares.

2.1.3. VARIABLE STRUCTURE CONTROL

Variable structure control (VSC) is also concerned with the robustness properties of the controller. Depending upon the performance of the system, the structure of the controller undergoes a change to facilitate an improvement in system dynamics. Since most modern control systems are digital in nature, Kumar et al (Kumar and Malik and Hope, 1987) investigated the effect of discretising the system model has on the performance of the closed loop system. It is noted that with comparison with the integral controller the VSC drives the ACE error to zero much faster.

2.1.4. FUZZY LOGIC CONTROL

G. A. Chown et al. (Chown and Hartman, 1997) implemented a fuzzy logic controller for automatic generation control to realise derivative action with the existing controller structure. Fuzzy logic control is based on the philosophy of placing the control action on a relative basis. As an example, if the ACE error is increasing, the control action should be adjusted as to drive the ACE error towards zero. In their

paper it is shown that satisfactory system performance can be achieved by the application of the fuzzy logic controller.

2.2. THE ECONOMIC DISPATCH PROBLEM

Included in automatic generation control is the economic load dispatch problem (ED). The economic dispatch routine attempts to control the generating units in the best economical way, minimising the control input to the plant, operating the plant in such a way as to minimise the operational costs of the units. The solution to this problem is achieved by means of optimisation techniques, neural networks (Yalcinoz and Short, 1997), genetic algorithms (Orero and Irving, 1996; Song and Chou, 1997) and linear programming techniques (Wood and Wollenberg, 1996).

Also included in the ED are program routines that predict the load of the system as a function of time. Load prediction methods employs the use of a regression model to model the system, taking into account factors such as weather and total system load at the time. Dispatching is done by adjusting the participation factors for each generating unit accordingly until the required power balance is achieved. The disadvantage of such an approach is that it has an effect on the performance of the system. By so doing the controller is detuned which may lead to oscillatory frequency behaviour (Chown, 1997).

2.3. THE EFFECTS OF FREQUENCY DEVIATION

Frequency deviation is a common problem within the power system control industry (Miller and Malinowski, 1993). This problem arises from the fact that the system load is constantly changing, making it very difficult for the power system operator to maintain the frequency at its nominal value. However the frequency deviation can be constrained to be within certain limits. The criteria that are used by Eskom, is that the frequency should not deviate by more than 50 mHz from its nominal value and it

should remain at its nominal for 90% of the time. Should the frequency deviate by a substantial amount permanent damage to power system equipment may result.

Many loads such as motors, transformers and clocks that are connected to the power system are frequency dependent. The operational performance of these loads depends upon the system frequency at any given point in time. In the case of motors the power output would vary in response to the change in frequency. This would be undesirable, leading to hazardous equipment operation and decreasing the level of security. In many circumstances theses motors are induction motors which operate synchronously with the system frequency.

Clocks or any timing instrumentation operating from the power supply would acquire a time error associated with any deviation in frequency. This can be illustrated by means of example. Consider a positive frequency deviation of 0.02 Hz, which is sustained for 2 hours, this would result in an accumulated time error of 2.4 seconds. In order to prevent time error accumulation, power utilities apply some sort of time error correction procedure. This is achieved by either increasing or decreasing the generation depending upon the time error to increase or decrease the system frequency. Should the frequency deviation exceed its prescribed limit, precautionary measures are taken to ensure the safety of personnel and that of the generating unit. Measures that are usually taken to ensure the safe operation of equipment are the tripping of the unit, or disconnecting the load, which is known as load shedding.

Frequency deviation is a direct result of under or over generation. When the generation within an interconnected area decreases or increases, the frequency changes by a factor known as the frequency bias factor β. The units for the frequency bias factor β is in MW per 0.1 Hz typically (Miller and Malinowski, 1993). Therefore for a change in generation of 10 MW will result in a frequency change of 1 Hz. This is known as the frequency response characteristic (FRC). The FRC depends on the governor droop settings of individual generating units as well as on the load of the system at the given time.

2.4. THE OPERATION OF POWER SYSTEMS

The process of electricity generation is shown in figure 2 below. It shows the electricity generation process from the procurement of the fossil fuel (coal), the pulverisation of the coal, the boiling of water to form super heated steam, which eventually drives the turbines and the generator to generate electricity.

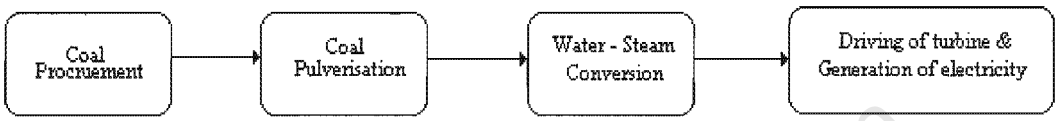


FIGURE 2: THE ENERGY CONVERSION PROCESS FOR ELECTRICITY GENERATION.

The last part of the above diagram is of importance to automatic generation control namely the driving of the turbines and the generation of electricity. Thus in maintaining the system frequency and powers at their desired values, control is required. The control of the system frequency and interchange powers is divided into two distinct controller sections, the primary and the secondary controllers. The primary control involves the rapid response of the system to load variations where fast controller action is required (Wood and Wollenberg, 1996). This is achieved by the turbine governor, where it senses the speed of the turbine shaft and depending upon the desired setpoint it applies corrective action to the main steam valve resulting in an increase or decrease of steam flow to the turbine. Shown in figure 3 below is an illustration showing of the operation of the turbine governor.

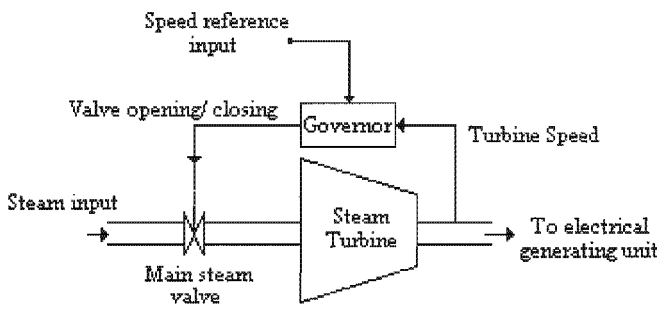


FIGURE 3: THE OPERATION OF THE PRIMARY CONTROLLER.

Changing the speed reference input of the governor alters the operating point of the system. Since any change in the speed of the turbine shaft is recognised by the governor and corrective action is applied to the position of the steam valve either by opening or closing it. The characteristic operation of the governor is known as the droop characteristic. The droop characteristic is the percentage change in frequency (or speed) that will cause a 100% change in output power (valve fully open or closed). Thus for example, a 5% droop characteristic will correspond to a 0.05Hz deviation in frequency, this will cause a 100% change in valve position causing a proportional change in output power. The droop characteristic is shown in figure 4 below.

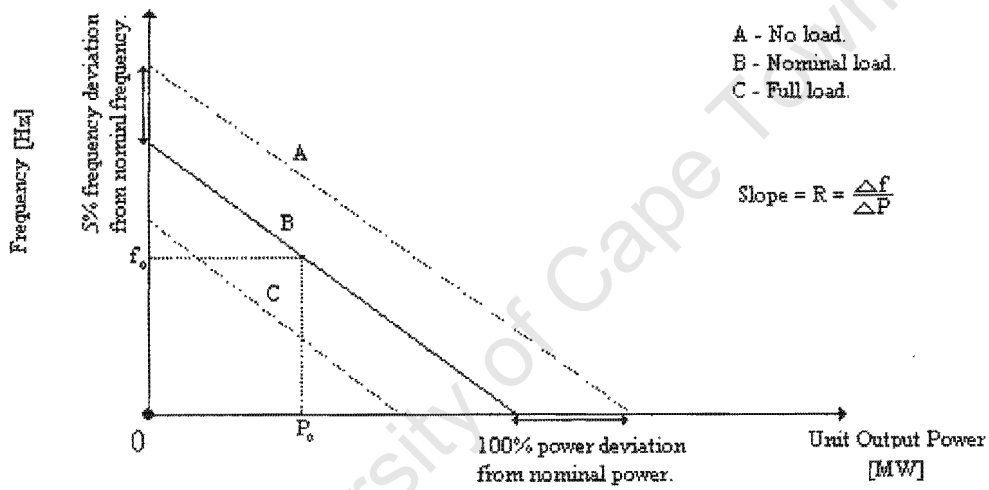


FIGURE 4: ILLUSTRATION OF THE DROOP CHARACTERISTIC.

Thus the droop characteristic R determines the slope of the graph shown in figure 4, that is, the change in unit output for a given change in frequency. Due to load variations the frequency of the system will also vary. An increase in load will result in a decrease in frequency and vice versa. This decrease in frequency will also be sensed by the governor, which alters the operating point of the system. Therefore to maintain the operating point of the system at its nominal value, the governor reference input must be changed requiring supplementary control or AGC also known as load frequency control.

The above statement can be verified by means of simulation. A commonly encountered model of the power system employed for load frequency control purposes are shown in figure 5 below (Wood and Wollenberg, 1996). See appendix 1.

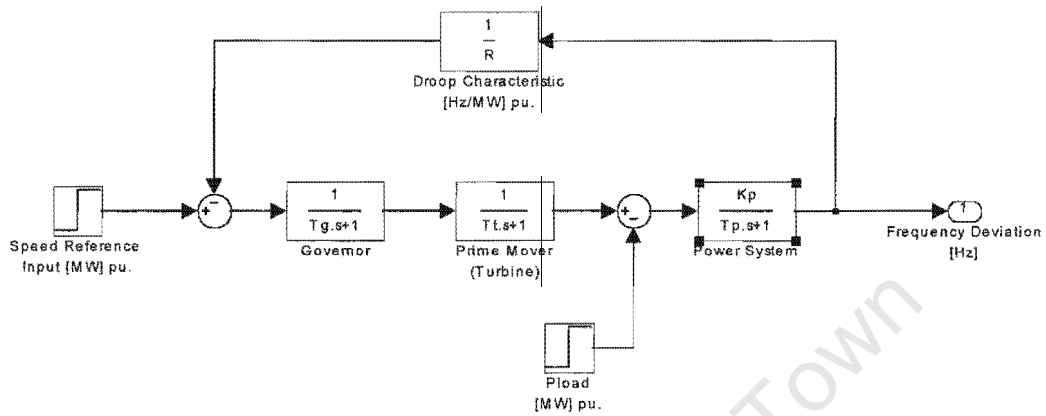


FIGURE 5: POWER SYSTEM MODEL OF LFC.

Model parameters are chosen as Kp = 120 Hz/MW, Tp = 20s, Tg = 0.08s, Tt = 0.3s, R = 2.4 Hz/MW, all parameters are per unit values (Bose and Atiyyan, 1980).

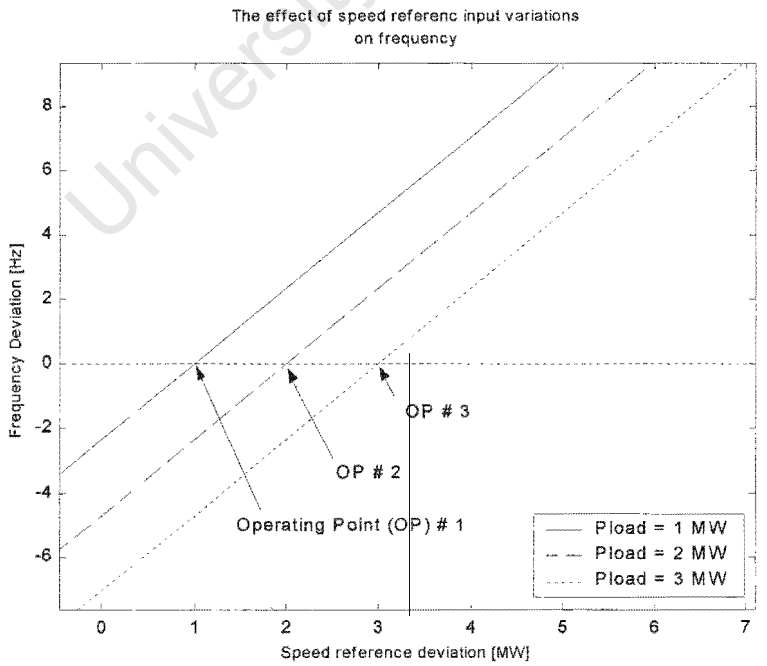


FIGURE 6: THE EFFECT OF VARYING THE SPEED REFERENCE INPUT ON FREQUENCY.

With reference to the model shown in figure 5, figure 6 illustrates the effect of varying the speed reference input of the turbine governor on the system frequency. Thus from figure 6 it is seen that in order to maintain the frequency at its nominal value in the presence of load variations the speed reference input of the turbine governor must be manipulated accordingly. An increase in the speed reference input will reflect an increase in system frequency. This has the effect of moving the operating point of the system. This motivates the use of supervisory controllers (LFC) since any load change will result in a frequency offset when no regulatory control is applied, despite the operation of the turbine governor.

2.5. INTERCONNECTED AREAS

The diagram below (figure 7) illustrates the operation of a two area interconnected power system. Each area is isolated from the other, except for the tie-line interchange power. This is to allow for contractual power arrangements as well as inter–area support in the event that one area cannot supply its demanded load capacity. Interconnected areas also add an extra level of security to the performance of the system.

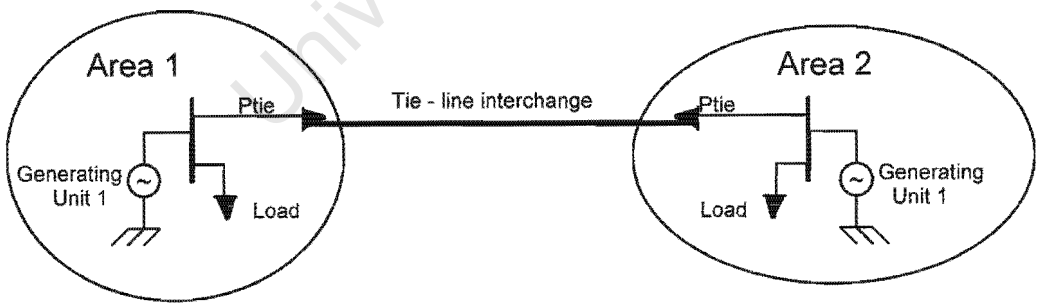


FIGURE 7: ILLUSTRATION OF TWO AREA INTERCONNECTED POWER SYSTEMS.

The requirement for the load frequency controller within an interconnected area is that it should only receive system status information from its own area. The only common factor is the tie-line interchange power, which should be kept constant as contracted

between the power utilities concerned. If this is not the case it leads to a loss of generation for the control area. This is known as inadvertent energy.

2.6. THE NERC CRITERIA

The NERC council is responsible for setting up international compliance standards on power utilities to regulate the power system to prescribed standards. The standards most relevant to load frequency control are given in terms of the area control error.

These criteria are outline below.

- **Criteria A1**: The ACE error must cross zero at least once within every 10 minute period.

- **Criteria A2:** The average ACE error must be within certain bounds (should not exceed $\pm L_d$) over all 10 minute period.

- **Criteria B1:** Following a disturbance, the ACE should return to zero within 10 minutes.

- **Criteria B2**: Following a disturbance, the ACE should begin to return to zero within one minute.

Since the load on the power system is constantly changing, the measure of the how well the control system is performing is accessed by the ACE error. The performance criteria under normal conditions are given by A1 and A2. Criteria B1 and B2 apply to disturbance conditions. The bound in criteria A2, namely L_d, limits the magnitude of the ACE error. It (L_d) represents the largest hourly load change experienced by the control area during the year.

CHAPTER 3

3. H_∞ OPTIMAL CONTROL

H_∞ optimal control also known as H_∞ optimisation is a frequency domain controller synthesis method, where prior to controller synthesis the desired closed loop specifications are formulated as weighting functions on chosen closed loop transfer function models (Williams, 1991). The name H_∞ refers to a set of transfer functions in the complex plane where these functions are analytic and bounded in the right half complex plane (i.e. they are stable transfer functions) and refers to a branch of mathematics dedicated to its study (Doyle et al, 1989). This synthesis procedure aims at minimising the infinity norm of the respective closed loop transfer function model chosen by the designer. The advantage gained by performing an H_∞ controller synthesis is that it guarantees closed loop stability and performance robustness. In addition it simplifies multivariable controller design, however this project is dedicated to a single input and single output (SISO) process. Though the solution to the synthesis problem is highly mathematical, during the controller design stage, namely the selection of weighting functions, this complexity is hidden from the designer motivating its extensive application (Matlab robust control toolbox, 1998).

3.1. H_∞ PROBLEM FORMULATION

This section presents the mathematical analysis necessary for the synthesis of the H_∞ controller, it is founded on the work performed by cited references Maciejowski (Maciejowski, 1989) and Skogestad and Postlewaite (Skogestad and Postlewaite, 1996) with other researches cited in this document. To apply H_∞ control theory to standard control problems of the form shown in figure 8 below, where the objective is to find a stabilising feedback controller K(s) satisfying tracking and disturbance

rejection properties, this configuration must first be formulated as an H_∞ synthesis problem. The plant G(s) is augmented with the functions W_1, W_2 and W_3 to form the augmented or generalised four port system shown in figure 9 below.

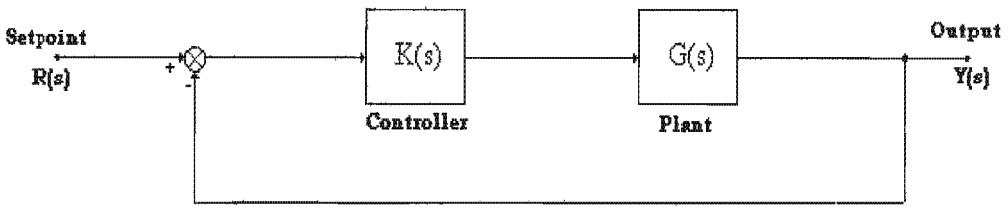


FIGURE 8: STANDARD CONTROL DIAGRAM.

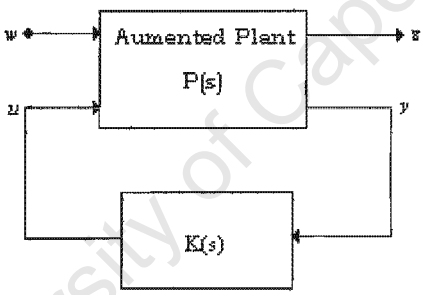


FIGURE 9: AUGMENTED PLANT AND FEEDBACK CONTROLLER.

P(s) is the augmented plant, K(s) is the synthesised controller, *w* is an exogenous input including all external signals such as setpoints and disturbances, *u* is the control input to the plant and y are all the measured signals or system outputs used for calculating the necessary control law. Characteristic to synthesis methods, H_∞ control theory defines an objective function as a measure of the level of success of the design (Francis and Zames, 1984). This objective function is defined as the minimisation of the infinity norm of the closed loop transfer function (or transfer function matrix for MIMO systems) between *w* and *z*. By describing the augmented plant P(s) as a matrix of the form,

$$P(s) = \begin{bmatrix} P_{11}(s) & P_{12}(s) \\ P_{21}(s) & P_{22}(s) \end{bmatrix} \quad (1)$$

the closed loop relation between w and z can be obtained. From figure 9 the following matrix equation can be defined.

$$\begin{bmatrix} z \\ y \end{bmatrix} = \begin{bmatrix} P_{11}(s) & P_{12}(s) \\ P_{21}(s) & P_{22}(s) \end{bmatrix} \begin{bmatrix} w \\ u \end{bmatrix} \quad (2)$$

Substituting for $u = K(s) * y$ and applying matrix multiplication to the above expression the following closed loop expression is obtained.

$$\begin{bmatrix} z \\ y \end{bmatrix} = \begin{bmatrix} P_{11}(s) * w + P_{12}(s) * K(s) * y \\ P_{21}(s) * w + P_{22}(s) * K(s) * y \end{bmatrix} \quad (3)$$
$$(4)$$

From (4),

$$y = [1 - P_{22}(s) * K(s)]^{-1} * P_{21}(s) w \quad (5)$$

Substituting equation (5) into equation (3) the closed loop transfer function between w and z is obtained. This transfer function is denoted as $F(K,P)$, which is a function of the feedback controller K(s) and the augmented plant P(s).

$$T_{z/w} = F(K,P) = P_{11}(s) + P_{12}(s) * K(s) * [I - P_{22}(s) * K(s)]^{-1} * P_{21}(s) \quad (6)$$

Thus the H_∞ synthesis problem is to find a stabilising controller that minimises the infinity norm of $F(K,P)$. This is written as,

$$\min_{K(s)} \|F(K,P)\|_\infty < \gamma \quad (7)$$

$\|F(K,p)\|_\infty$ is an operator norm, which by definition represents a matrix by means of a scalar number. Equation 7 denotes the minimisation of $\|F(K,p)\|_\infty$ over all stabilising

controllers K(s). Where γ, known as the stability margin is the minimum value for which the infinity norm of the closed loop transfer function matrix *F(P,K)* should satisfy for all stabilising controllers. The infinity norm is defined as the maximum upper singular value (or supremum) of a transfer function matrix (Maciejowski, 1989). Stated more generally, the infinity norm represents the relative size of the transfer function gain by expressing it as a single number (Robel, 1989). This is illustrated below.

$$\|G(j\omega)\|_\infty = \sup \bar{\sigma}(G(j\omega))$$

where, $\bar{\sigma}$ = upper singular value.

sup = supremum or the least upper bound.

A common H_∞ optimisation problem is the mixed sensitivity problem, in which more than one closed loop transfer function is minimised (Skogestad and Postlethwaite, 1996). Conventionally the closed loop sensitivity *S(s)* and complementary sensitivity *T(s)* functions are optimised with the control sensitivity *M(s)*. This is formulated as,

$$\min \left\| \begin{matrix} W_1(s)S(s) \\ W_2(s)M(s) \\ W_3(s)T(s) \end{matrix} \right\|_\infty < \gamma \qquad (8)$$

W_1, W_2 and W_3 are weighting functions on the sensitivity, control sensitivity and complementary sensitivity functions respectively. The motivation for optimising multiple closed loop transfer functions is to enable the achievement of multiple performance objectives (Skogestad and Postlethwaite, 1996). Since S(s) is the closed loop transfer function between the system output y(s) and the output disturbance d(s), W_1 will impose a performance constraint on the output disturbance rejection properties of the control system. Similarly, W_3 imposes constraints on the tracking performance of the system. Because these specifications, namely tracking and disturbance rejection specifications cannot be achieved simultaneously, the weighting functions are frequency dependent functions, enabling a frequency domain compromise of these specifications. By substituting equations (9), (10) and (11),

$$S(s) = \frac{1}{1 + K(s) * G(s)} \tag{9}$$

$$M(s) = \frac{K(s)}{1 + K(s) * G(s)} \tag{10}$$

$$T(s) = \frac{K(s) * G(s)}{1 + K(s) * G(s)} \tag{11}$$

into equation (8) and equating coefficients with that of equation (6) the following augmented plant can be found (Maciejowski, 1989).

$$P(s) = \left[\begin{array}{c|c} P_{11}(s) & P_{12}(s) \\ \hline P_{21}(s) & P_{22}(s) \end{array}\right] = \left[\begin{array}{c|c} W_1 & -W_1 G \\ 0 & W_2 \\ 0 & W_3 G \\ \hline I & G \end{array}\right] \tag{12}$$

Once appropriate weighting functions have been chosen and the augmented plant obtained, the controller synthesis procedure may proceed.

3.2. H_∞ CONTROLLER SYNTHESIS

An algorithm, which solves the H_∞ synthesis problem, is the Doyle and Glover procedure outlined in detail in (Doyle et al, 1989). This algorithm solves for the H_∞ controller by iteratively solving two Riccati equations and finds a sub-optimal controller satisfying the imposed frequency domain specifications of the closed loop system. This algorithm is one of the simpler H_∞ synthesis solutions however the mathematics is nontrivial (Maciejowski, 1989). This solution (the Glover and Doyle solution) is briefly described below. It is assumed that the augmented plant P(s) shown in figure 9 has the following state space realisation.

$$P(s) = \left[\begin{array}{c|cc} A & B_1 & B_2 \\ \hline C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{array}\right] \tag{13}$$

For simplifying the solution equations, D_{11} and D_{22} are assumed to be zero. Further assumptions made are,

i) (A, B_1) is stabilizable and (C_1, A) detectable.

ii) (A, B_2) is stabilizable and (C_2, A) detectable.

iii) $D'_{12}[C_1 D_{12}] = [0\ I]$.

iv) $\begin{bmatrix} B_1 \\ D_{21} \end{bmatrix} D'_{21} = \begin{bmatrix} 0 \\ I \end{bmatrix}$

Assumptions i) and ii) are required to ensure that the resultant controller stabilises the closed loop system, while that of iii) and iv) ensure that the matrices to the left of the matrix equation is both row and column rank respectively, this ensures that the synthesised controller is proper and hence realisable. A detailed study on the above assumptions can be found in the original paper of Doyle and Glover (Doyle et al, 1989).

The solution to the H_∞ synthesis problem involves the solution of two algebraic Riccati equations. These equations are as follows, where X_∞ and Y_∞ are solutions to the Riccati equations. The superscript T denotes transpose.

$$A^T X_\infty + X_\infty A + C_1{}^T C_1 + X_\infty \left(\gamma^{-2} B_1 B_1{}^T - B_2 B_2{}^T\right) X_\infty = 0 \tag{14}$$

$$A Y_\infty + Y_\infty A^T + B_1 B_1{}^T + Y_\infty \left(\gamma^{-2} C_1{}^T C_1 - C_2{}^T C_2\right) Y_\infty = 0 \tag{15}$$

Also the spectral radius of,

$$\rho(X_\infty \quad Y_\infty) \quad < \quad \gamma^2 \tag{16}$$

Upon satisfaction of the assumptions and the solving the algebraic Riccati equations, a sub optimal controller can be found and is defined as,

$$K_{sub}(s) = \left[\begin{array}{c|cc} \hat{A}_\infty & -Z_\infty L_\infty & Z_\infty B_2 \\ \hline F_\infty & 0 & I \\ -C_2 & I & 0 \end{array}\right] \tag{17}$$

Where,

$$\hat{A}_\infty = A + \gamma^{-2} B_1 B_1^{\ T} X_\infty + B_2 F_\infty + Z_\infty L_\infty C_2$$

$$F_\infty = -B_2^{\ T} X_\infty, \quad L_\infty = -Y_\infty C_2^{\ T}, \quad Z_\infty = \left(I - \gamma^{-2} Y_\infty X_\infty\right)^{-1}$$

It should be noted that the above controller explanation is fairly brief based of the complexity of the mathematics and for more detailed information the cited references should be examined.

3.3. H_∞ CONTROLLER DESIGN

The practical design of an H_∞ controller involves the selection of weighting functions W_1, W_2 and W_3. Appropriate choice of these functions ensures that the desired system performance specifications are achieved and maintained in the presence of model uncertainty. Presented below are a few guidelines as to the appropriate selection of these functions. From classical control theory where multiple control objectives are defined such as setpoint tracking and disturbance rejection, a frequency domain compromise between all closed loop objectives must be reached (Skogestad and Postlethwaite, 1996). In H_∞ control this is achieved by choosing W_1 as a low pass filter and W_2 and W_3 as high pass filters respectively.

3.3.1. SELECTION OF WEIGHT W_1

Weight W_1 is a performance weight on the closed loop sensitivity function S(s), equation (9), and as such it imposes frequency domain constraints on S(s) for output disturbance rejection properties of the feedback system shown in figure 8. Traditionally output disturbances occur over lower frequency ranges as compared to that of sensor noise and as a result it is desired that S(s) should be low over the frequency range where the energy of the disturbance model is dominant. That is, the 3dB frequency of the disturbance model (Skogestad and Postlethwaite, 1996). This is mathematically formulated as follows in equation (18), where the magnitude of S(s) should be less than or equal to the inverse of W_1.

$$|S(s)| \leq \left|\frac{1}{W_1(s)}\right| \quad \forall w$$

$$\Rightarrow \|W_1 S\|_\infty < 1 \qquad (18)$$

This is illustrated below by defining W_1 as a simple first order low pass filter in figure 10. Weight W_1 is chosen as $W_1 = \frac{1}{s+1}$. The hashed region in figure 10 indicates the region, which the sensitivity function S(s) should not violate in the frequency domain. In addition weight W_1 manipulates the transient performance of the closed loop system.

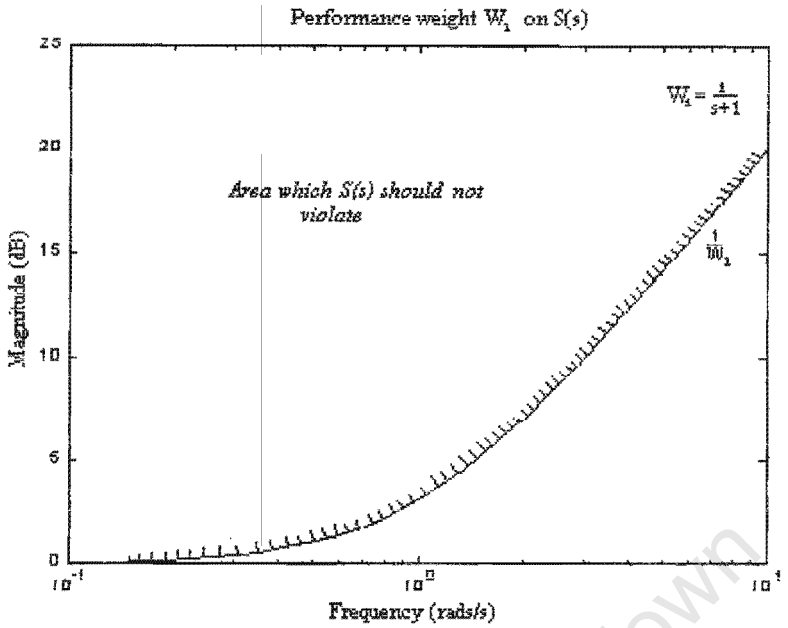


FIGURE 10: TYPICAL SHAPE OF PERFORMANCE WEIGHT W_1.

3.3.2. SELECTION OF WEIGHT W_2

W_2 is a performance weight on the control sensitivity M(s). It is desirable to minimise the amount of control action utilised at high frequencies to reduce actuator saturation or spurious high frequency modes (Katebi and Grimble and Zhang, 1997). Therefore to accomplish this, W_2 is chosen as a high pass filter, initially a first order transfer function is used. However, in practice it suffices to select W_2 as a small constant to satisfy rank conditions when synthesising the H_∞ controller. Its mathematical analogue is shown in equation (19).

$$|M(s)| \leq \left|\frac{1}{W_2(s)}\right| \qquad \forall w$$

$$\Rightarrow \quad \|W_2 M\|_\infty < 1 \qquad (19)$$

3.3.3. SELECTION OF WEIGHT W_3

Weight W_3 must be selected in compliance with W_1 via the relation S(s) + T(s) = 1. This ensures that the required tracking specifications are maintained and robustness due to additive or multiplicative model uncertainty is achieved (Kwakernaak, 1993). The requirement for setpoint tracking is that T(s), the complementary sensitivity function, must be 0dB at low frequencies, this implies that the inverse of W_3 must be high at low frequencies and small at high frequencies. This is to ensure sufficient noise attenuation properties. Mathematically this is expressed as,

$$|T(s)| \leq \left|\frac{1}{W_3(s)}\right| \qquad \forall w$$

$$\Rightarrow \|W_3 T\|_\infty < 1 \qquad (20)$$

Illustrated in figure 11 below is W_3 for a simple first order high pass filter, $W_3 = \frac{s}{s+1}$.

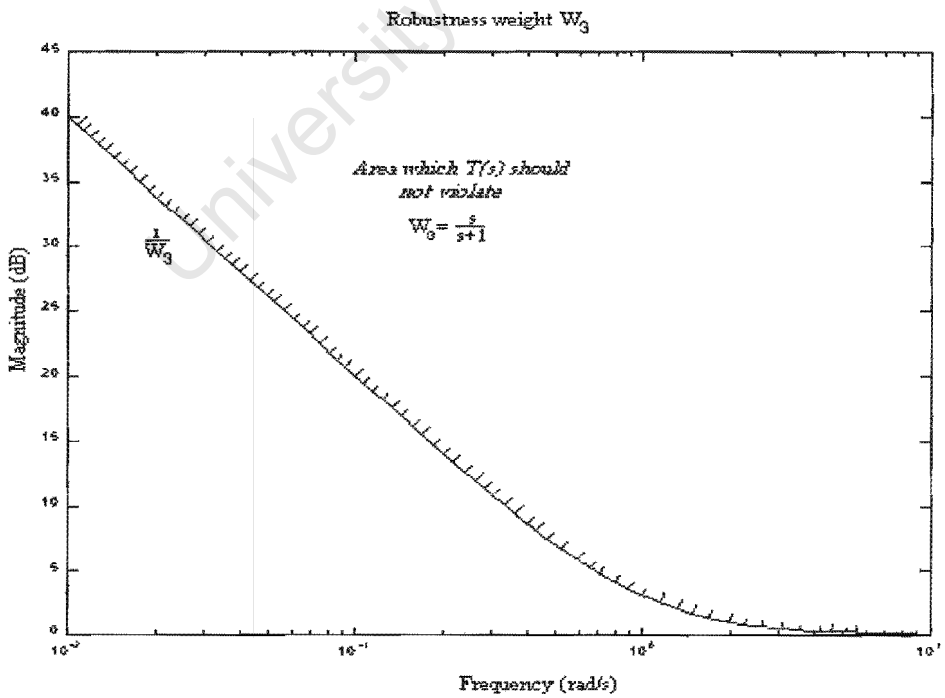


FIGURE 11: TYPICAL SHAPE OF WEIGHTING FUNCTION W_3.

CHAPTER 4

4. FUZZY LOGIC AND CONTROL

Pioneering the work in fuzzy set theory and its application to control systems was L.A Zadeh during the mid 1960's (Zadeh, 1965) when he concluded his seminal work on fuzzy set theory. The principle operation of fuzzy set theory or commonly known as fuzzy logic is to describe precise information by an imprecise system description (Reznik, 1997). This can be demonstrated by means of an example. Consider the height of a person named John. The height of John when described precisely can be said to be 1.5m, also known as the crisp system description. However when described in fuzzy terms John might be said to be of medium height, where heights less than 1.2m corresponds to the fuzzy linguistic term short and that greater than 1.7m to the linguistic term tall and anything in between as medium. The advantage gained by performing the transformation from crisp information to fuzzy information enable easier data processing and places the operational arena within a human context. This enables the designer to incorporate technical knowledge based on the behaviour of the plant into the design of the controller.

Being inherently nonlinear in operation, fuzzy logic control is a convenient design method for dealing with nonlinear processes, and as such, successful industrial applications of fuzzy logic control have been reported. Thus with its application to power systems, where the plant is large and characterised by nonlinear behaviour, it forms a good choice for controller design (Kandel and Langholz, 1994). Industrial applications of fuzzy logic control to power systems include, the start up control of a steam generator (Constantin von Altrock, Dirk Prussmann, Bernhard Krause, and Britta Franken, 1997), automatic generation control (Chown, G and Hartman, R.C, 1997) and supervisory control of a coal power plant (Pruessmann, D, 1997). Results presented in these citations indicate that fuzzy logic control operated favourably.

4.1. FUZZY SET THEORY

At the heart of fuzzy logic control is fuzzy set theory, where in contrast to the traditional notion of a set in which an object is either a member of the set or not (i.e. 1 or 0), fuzzy sets describes the membership of the object over a certain range between 1 and 0 (Zadeh, 1965). Thus for a fuzzy set A consisting of real numbers for example, a member value of set A is x, will have an associated degree of membership denoted by $\mu_A(x)$. This is graphically illustrated in figure 12 below.

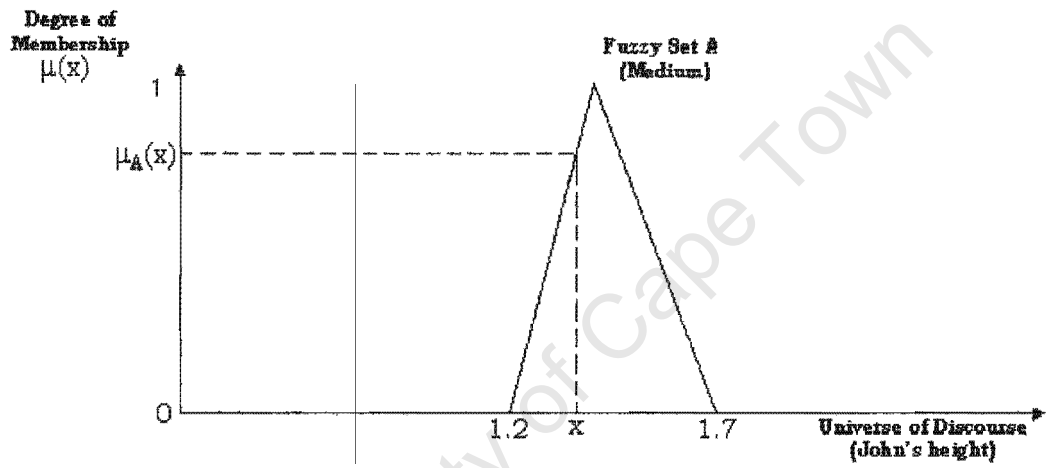


FIGURE 12: ILLUSTRATION OF FUZZY SET THEORY.

The degree of membership $\mu(x)$ in figure 12 reflects the measure to which the crisp input value x is a member of set A. Thus in accordance with the introductory example, all crisp values of Johns height which are members of set A will have a membership degree of $\mu_A(x)$ else it will be 0, indicating that it is not a member of set A. The shape of the fuzzy set shown above is triangular, which relates the crisp input to the fuzzy input, thus it is a convenient mapping tool between the two variables namely the universe of discourse and the degree of membership. Though there are other fuzzy set shapes to choose from (i.e. quadratic, exponential etc.), the triangular set is chosen based primarily on its mathematical ease and incorporating higher order functions as fuzzy sets does not readily reflect significant changes in control effort (Reznik, 1997).

4.2. FUZZY SET OPERATIONS

As stated above fuzzy set theory forms an integral part in fuzzy control and in order to maximise the operation of the resultant controller, fuzzy mathematical operations are performed on each fuzzy set. Detailed analysis of these operations are not given, only the relevant functions applicable to the proper functioning of the designed controller are described. The commonly encountered fuzzy operations are the union and intersection of fuzzy sets. The union of two sets refers to the combination of all elements within both sets, whereas the intersection of two sets, are the elements of the sets that are common in both sets. Figure 13 below demonstrates the union and intersection of fuzzy sets.

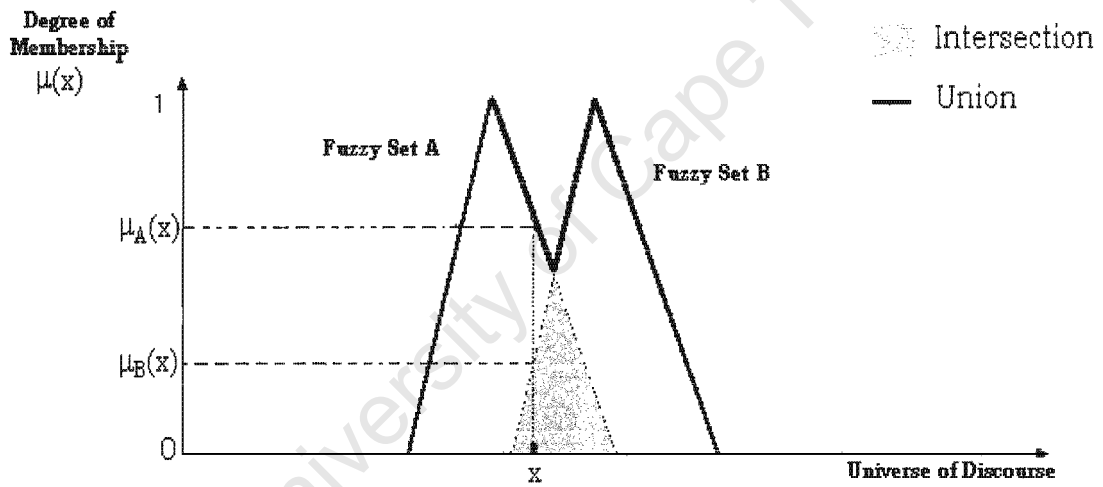


FIGURE 13: UNION AND INTERSECTION OF FUZZY SETS.

As seen from figure 13 the intersection of two fuzzy sets, are represented by the grey coloured region. The mathematical interpretation of this by considering the crisp input x, within the universe of discourse, yields two degree of membership functions, namely $\mu_A(x)$ and $\mu_B(x)$ for fuzzy sets A and B respectively. Thus to obtain one membership function for variable x to represent the collective effect due to the functions $\mu_A(x)$ and $\mu_A(x)$, the mathematical min (minimum) function is used. The

motivation for this is that the "chain is only as strong as its weakest link". Thus a commonly used definition for the intersection of fuzzy sets A and B are defined by,

$$\mu_{A\cdot B}(x) = \min(\mu_A(x), \mu_B(x))$$

In figure 13 the union of the two fuzzy sets are depicted by the region enclosed by the bold solid line. Since the union encapsulates all elements present in both sets the max (maximum) function is used as its mathematical equivalent. Thus the definition of the union of fuzzy sets are defined by,

$$\mu_{A+B}(x) = \max(\mu_A(x), \mu_B(x))$$

4.3. FUZZY CONTROL

Conventional controller design methods employ the use of a detailed mathematical model of the plant or process to be controlled. When these design methods fail to satisfy specified performance criteria based on the obtained model, alternative solutions to controller design are sought (Tong, 1977). One such method is fuzzy control. This method of controller design is based on a series of if and then statements. This is formulated as follows and are known as fuzzy rules.

```
if
   { Condition is true }
then
   { Do this }
end
```

The conditional part of the above statement is known as the antecedent part and that of the then part as the consequent part. The fuzzy logic controller consists of a set of rules, which are formulated based on 'rules of thumb'. The output of the controller

thus depends on the processing of the fuzzy rules. The success of the resultant controller depends to a great deal on the structuring of the fuzzy rules and upon the number of rules employed. The complexity or size of the rules increases in a polynomial manner depending upon the number of controller inputs and number of fuzzy sets used (Chwee, et al).

The fuzzy logic controller can be divided into three distinct sections, namely fuzzification, inferencing rules and defuzzification. The block diagram shown in figure 14 below illustrates the general structure of the fuzzy logic controller where the inputs and outputs are all crisp values used for measurement and control. Fuzzy inputs and fuzzy outputs are internal to the controller.

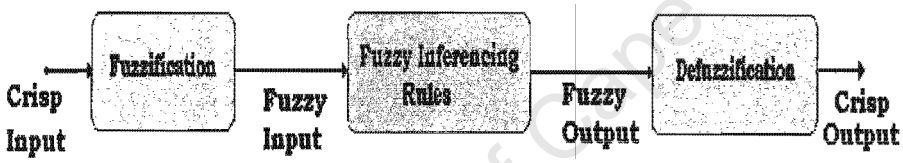


FIGURE 14: STRUCTURE OF THE FUZZY CONTROLLER.

4.3.1. FUZZIFICATION

Fuzzification refers to the mapping procedure from the crisp input value x to the fuzzy input $\mu(x)$. Traditionally the universe of discourse is divided into a number of fuzzy sets known as fuzzy partitions as shown in figure 15 below.

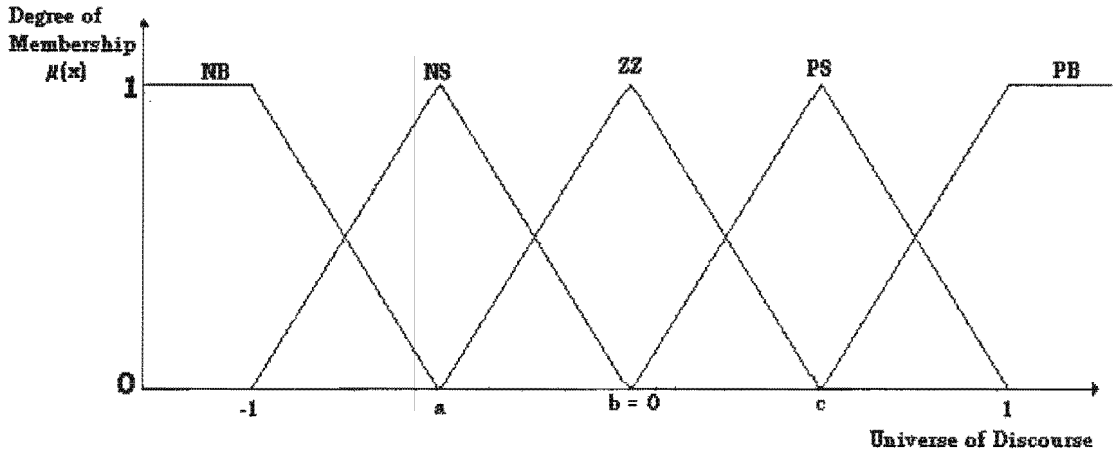


FIGURE 15: FUZZIFICATION OF CRISP INPUTS.

These fuzzy partitions are labelled by fuzzy linguistic terms describing the input quantitatively. The abbreviations NB, NS, ZZ, PS and PB stand for the quantitative terms negative big, negative small, zero, positive small and positive big respectively. As described above these functions are chosen based primarily on mathematical ease and are described as follows.

$$\mu(x) = \begin{cases} 0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots x < a \\ \dfrac{(x-a)}{(b-a)} \ldots\ldots\ldots\ldots\ldots a \le x \le b \\ \dfrac{(c-x)}{(c-b)} \ldots\ldots\ldots\ldots\ldots b \le x \le c \\ 0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots x > c \end{cases}$$

4.3.2. FUZZY INFERENCING RULES

Fuzzy inferencing rules forms the central data processing engine of the fuzzy logic controller. These rules are structured as a series of if and than statements where the appropriate control is calculated when a certain condition is met. Depending upon the value of the input and its related fuzzy linguistic term, a fuzzy inferencing rule will be 'fired'.

This can be shown by means of an example. Shown below are typical fuzzy rules for speed control of a DC motor with the following fuzzy sets.

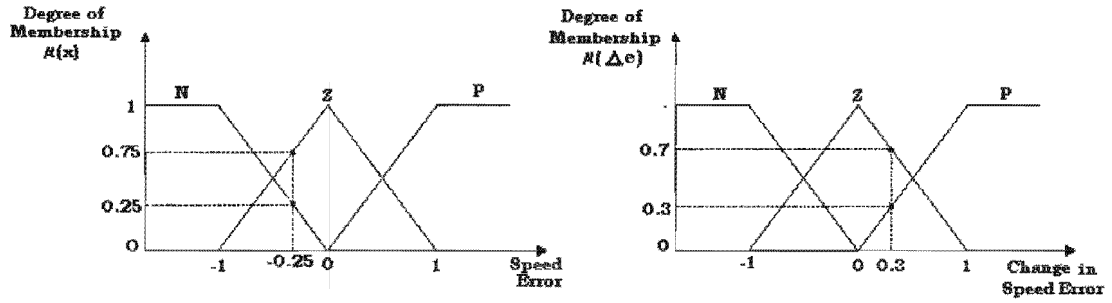


FIGURE 16: INPUT FUZZY SETS FOR EXAMPLE PROBLEM.

The inferencing rules are as follows.

1. *If the error is Negative and the Δerror is Negative then the Output is Negative.*
2. *If the error is Negative and the Δerror is Zero then the Output is Negative.* ..... *(fired)*
3. *If the error is Negative and the Δerror is Positive then the Output is Zero.* ..... *(fired)*
4. *If the error is Zero and the Δerror is Negative then the Output is Negative.*
5. *If the error is Zero and the Δerror is Zero then the Output is Zero.* ..... *(fired)*
6. *If the error is Zero and the Δerror is Positive then the Output is Positive.* ..... *(fired)*
7. *If the error is Positive and the Δerror is Negative then the Output is Zero.*
8. *If the error is Positive and the Δerror is Zero then the Output is Positive.*
9. *If the error is Positive and the Δerror is Positive then the Output is Positive.*

From the fuzzy sets shown in figure 16 above and from the inferencing rules, four fuzzy rules are fired, namely, rules 2, 3, 5 and 6. Each rule yields an appropriate degree of membership function as follows.

$$\text{Rule 2}: \quad \begin{aligned} \mu_{N,Z}(e,\Delta e) &= \min(\mu_N(-0.25), \mu_Z(0.3)) \\ &= \min(0.25, 0.7) \\ &= 0.25 \end{aligned}$$

Intersection of Fuzzy Sets N and Z

Rule 3 :
$$\mu_{N\cdot P}(e,\Delta e) = \min(\mu_N(-0.25), \mu_P(0.3)) = \min(0.25,0.3) = 0.3$$
Intersection of Fuzzy Sets N and P

Rule 5 :
$$\mu_{Z\cdot Z}(e,\Delta e) = \min(\mu_Z(-0.25), \mu_Z(0.3)) = \min(0.75,0.7) = 0.7$$
Intersection of Fuzzy Sets Z and Z

Rule 6 :
$$\mu_{Z\cdot P}(e,\Delta e) = \min(\mu_Z(-0.25), \mu_P(0.3)) = \min(0.75,0.3) = 0.3$$
Intersection of Fuzzy Sets Z and P

4.3.3. CLIPPING OF FUZZY OUTPUT SETS

The previous section concentrated on the calculation of the degree of membership functions corresponding to each fired inferencing rule. Thus in order to obtain a crisp control output, further processing of these values are necessary. The effect, which these membership values have on the output fuzzy sets, is to clip or scale them (Mamdani, 1974). Since each fired inferencing rule produces a clipped output fuzzy set, the final output results by taking the union of all the clipped output sets. Figure 17 below illustrates the procedure.

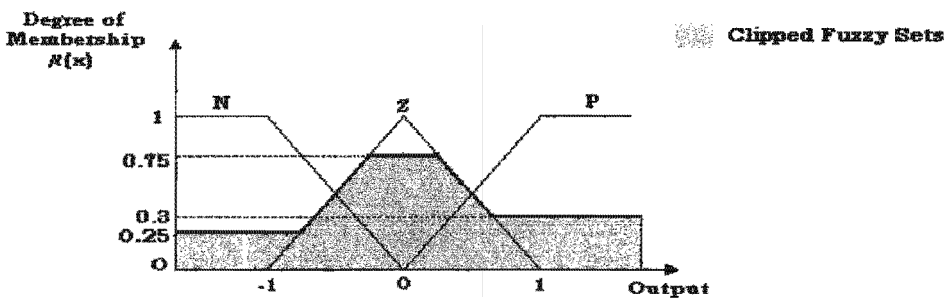


FIGURE 17: CLIPPED OUTPUT FUZZY SETS.

The shaded region in figure 17 represents the clipped output fuzzy sets corresponding to the error and change in error inputs at that time. Thus in order to obtain a crisp control output defuzzification is required.

4.3.4. DEFUZZIFICATION OF FUZZY OUTPUTS

Obtaining a crisp control output from a fuzzy output requires defuzzification. This process can be realised by a number of different defuzzification methods, such as the bisector, middle of maximum, largest of maximum, smallest of maximum and the centriod methods of which we will consider the latter (Kandel and Langholz, 1994).

The input to the defuzzification process is a fuzzy set which is the result from the aggregation or union of all clipped output fuzzy sets and yields the required control output. In the centroid method the universe of discourse of the output is divided into N sampled points, these points are used to calculate the control value. This is illustrated in figure 18 below where N is chosen as 4.

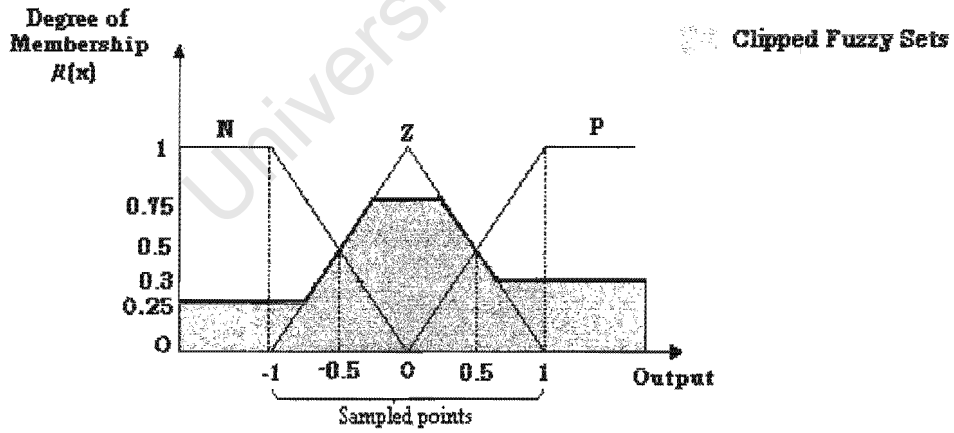


FIGURE 18: SAMPLING POINTS FOR DEFUZZIFICATION

The four sampling points are $y_{s1} = -1$, $y_{s2} = -0.5$, $y_{s3} = 0.5$ and $y_{s4} = 1$. These points are then compared with the consequent parts of each fired rule by using the min

function. This is illustrated as follows by considering the result for rule 2 and tabulating the results for remaining fired rules.

Rule 2: *If the error is Negative and the Δerror is Zero then the Output is Negative.*

$$\begin{aligned}\mu_{N.Z}(e,\Delta e) &= \min(\mu_N(-0.25), \mu_Z(0.3)\\ &= \min(0.25,0.7)\\ &= 0.25\end{aligned}$$

$$\mu_{s1}(y_{s1}) = \mu_{s1}(-1) = 1 \qquad (\textit{membership value for sampling point} - 1)$$

Therefore,

$$\mu_{D1} = \min(\mu_{NZ}(e,\Delta e), \underbrace{\mu_{s1}(y_{s1})}_{\substack{\textit{membership value corresponding to Output}\\ \textit{fuzzy set Negative (since consequent part is Negative)}}}$$

$$= \min(0.25,1)$$
$$= 0.25$$

Likewise for rules 3, 5 and 6.

$$\mu_{D2} = \min(\mu_{NP}(e,\Delta e), \underbrace{\mu_{s1}(y_{s1})}_{\text{Rule 3:}})$$

$$= \min(0.3,1)$$
$$= 0.3$$

$$\mu_{D3} = \min(\mu_{ZZ}(e,\Delta e), \underbrace{\mu_{s1}(y_{s1})}_{\text{Rule 5:}})$$

$$= \min(0.7,1)$$
$$= 0.7$$

$$\mu_{D4} = \min(\mu_{ZP}(e,\Delta e), \underbrace{\mu_{s1}(y_{s1})}_{\text{Rule 6:}})$$

$$= \min(0.3,1)$$
$$= 0.3$$

Taking the union of these values yields,

$$\begin{aligned} u_{S1} &= \max(\mu_{D1}, \mu_{D2}, \mu_{D3}, \mu_{D4}) \\ &= \max(0.25, 0.3, 0.7, 0.3) \\ &= 0.7 \end{aligned}$$

Repeating the above procedure for each sampled output point, the following tabulated values are obtained.

i	*Output Sample Value* y_{si}	*Consequent Value* u_{si}
1	-1	0.7
2	-0.5	0.5
3	0.5	0.5
4	1	0.5

TABLE 1: TABULATION OF OUTPUT DEFUZZIFICATION VALUES

The crisp control output can thus be calculated by the following equation.

$$\textit{Crisp Control Output } u = \frac{\sum_{i=1}^{N} y_{si} * u_{si}}{\sum_{i=1}^{N} u_{si}}$$

Where N is the number of output sample values, y_s the sample value and u_s the result obtained by taking the union of the consequent parts of each fired rule. Therefore the controller output is calculated as follows,

$$u = \frac{(-1)*(0.7)+(-0.5)*(0.5)+(0.5)*(0.5)+(1)*(0.5)}{0.7+0.5+0.5+0.5}$$

$$= -0.091\,V \quad \text{(assuming the control input is a voltage)}$$

The control value calculated above indicates that the control input should decrease by 0.091V. This value depends upon the choice of the sample points where points –1 and 1 dictates the maximum control deviation.

4.4. STABILITY OF THE FUZZY LOGIC CONTROLLER

A control system is said to be stable if when unperturbed (eg. no step test) its states tends towards an initial or desired state, this also applies when the system is perturbed (Zhang, X et al., 1994). In fuzzy logic control, linear control theory does not readily apply, since the controller cannot be mathematically defined. However, a nonlinear function approximation of the controller can be found, which enables nonlinear stability analysis. Methods of evaluating the stability of fuzzy systems are as follows. By observing the order in which the fuzzy rules are fired an indication of the stability of the system can be obtained. This method looks at the fuzzy rule trajectory (the order in which the fuzzy rules are fired), if this trajectory progressively decreases towards the centre of the fuzzy rules table where state deviations are zero, the system is said to be stable. However, if this trajectory has the reverse tendency, the system is unstable (Reznik, 1997). Braae and Rutherford (Braae and Rutherford, 1979) investigated the theoretical aspects of fuzzy systems and analysed stability by nonlinear phase plane trajectories. DeGlas (DeGlas) proposed an analysis via Lyapunonv's stability theory.

CHAPTER 5

5. QUANTITATIVE FEEDBACK THEORY

Quantitative feedback design theory is a frequency domain closed loop controller design technique introduced by Isaac M. Horowitz during the early 1970's (Horowitz, 1991). This design technique exploits the controller design facilities prevalent in the Nichols chart based on classical loop shaping ideas, though recently, developments have been made to automate the loop shaping process (Chen and Ballance, 1998; Chait and Chen and Hollot, 1999). Since in any controller design procedure it is expedient to the stability and transient response of the controlled system to consider model uncertainty, quantitative feedback theory incorporates uncertainty in the form of plant templates. These templates are geometric representations of all plant uncertainty drawn on the Nichols chart (Wu and Grimble and Breslin, 1998). Templates are constructed by expressing model uncertainty parametrically or non parametrically depending on the choice of the designer, this is achieved by defining the uncertainty over a given range of all possible parameter values.

Similar to conventional controller design methodologies such PI and LQG, and synthesis methods such as H_∞ optimal control, performance specifications must be incorporated into the QFT design. This is achieved by imposing performance bounds on selected or all closed loop transfer functions models. These specifications then translate to gain and phase bounds on the nominal open loop transfer function model L(s), which must not be violated for a successful design. The loop shaping procedure then finds an appropriate controller satisfying all performance bounds on the Nichols chart. This process is by no means trivial and depends to a great extent on the expertise of the feedback controller designer. This is significantly simplified if the problem at hand is convex, which means that a global optimum exists. Figure 19 below illustrates the system block diagram employed for a QFT design. It shows the plant G(s) to be controlled, the cascade feedback controller K(s) and the prefilter F(s). Thus the QFT design problem is to find K(s) and F(s) to meet system specifications

such as setpoint tracking, disturbance rejection and robustness properties. The resultant controller is known as a two degree of freedom controller.

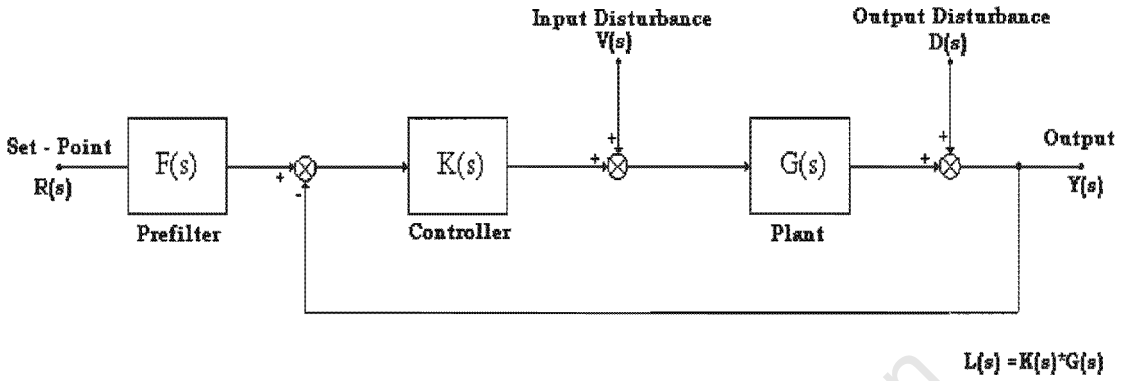


FIGURE 19: FEEDBACK CONTROL LOOP FOR QFT.

The application of QFT has varied extensively over the past decade in comparison to previous years. This is a direct result of the availability of good computer aided design software packages such as the Matlab QFT design toolbox (Matlab version 5.3, 1999). The application of QFT includes flight control systems (Wu and Grimble and Breslin, 1998), vibration control of piezoceramic – based smart structures (Choi and Cho and Park, 1999), digital control of multiple input multiple output or MIMO processes (Duncan, 1995), active vibration control (Yaniv and Horowitz, 1990) and power system stabiliser (Rao and Sen, 1999) to name but a few. As illustrated in the above cited references the QFT design technique has been successfully implemented and has satisfied the desired performance criterion.

Essentially QFT consists of four controller design steps. These include the formulation of time domain specifications as frequency domain constraints (Horowitz, 1993). Though time domain specifications are not compulsory since they merely form a tool for obtaining frequency domain specifications. In addition meeting frequency specification does not necessarily ensure that time domain specifications are met. Secondly the plant uncertainty is parametrically quantified by means of a parameter set. This is followed by the generation of plant templates on the Nichols chart. Thirdly the QFT design is progressed by the computation of QFT bounds on the nominal open

loop transfer function L(s). Finally the controller is designed satisfying the system performance and all QFT bounds by making use of classical loop shaping design techniques. This is followed by a prefilter design, necessary for translating the closed loop transfer function model to within a region satisfying the required system tracking specifications. The following section describes the above mentioned design procedure in more detail.

5.1. SYSTEM TRACKING SPECIFICATIONS

Since quantitative feedback theory is a frequency domain design methodology, time domain specifications must be formulated as frequency domain constraints (Horowitz and Sidi, 1972). A convenient transient analysis tool is the system output response due to a unit step input, since it examines both the high frequency and steady state properties of the system under investigation. Thus analysis by means of a unit step lends itself to the formulation of time domain specifications based on conventional concepts such as rise time t_r, settling time t_s, and maximum peak overshoot M_p (Horowitz, 1993).

Tolerances are placed on both the upper and lower excursions of the unit step response. This is formulated as follows.

$$a(t) \leq y(t) \leq b(t)$$

Where $y(t)$ = unit step response.

$a(t)$ = lower tracking bound specification.

$b(t)$ = upper tracking bound specification.

These are translated to the frequency domain as follows.

$$|A(jw)| \leq |T_R(jw)| \leq |B(jw)|$$

Where

$$T_R(jw) = F(s) * \frac{L(s)}{1 + L(s)}, \text{ the closed loop tracking transfer function.}$$

$A(s)$ = the lower frequency bound.

$B(s)$ = the upper frequency bound.

F(s) and L(s) are the prefilter and open loop transfer functions respectively.

Initial choices for the tracking specifications are the second or third order transfer function models (Horowitz and Sidi, 1972). Therefore finding bounds on the respective model parameters enables the corresponding time domain bounds to be satisfied or vice versa. Manipulation of the damping factors and natural frequencies of these models allows for easy achievement of tracking specifications. The structure of the dominant second order model is shown below.

$$A(s)=\frac{\omega_{nl}^2}{(s^2+2\varsigma_l s+\omega_{nl}^2)(s+\alpha_l)} \qquad \textit{lower bound specification}$$

$$B(s)=\frac{\omega_{nu}^2(s+\alpha_u)}{(s^2+2\varsigma_u s+\omega_{nu}^2)} \qquad \textit{upper bound specification}$$

Where ω_{nl}, ξ_l and ω_{nu}, ξ_u are the natural frequencies and damping factors for the upper and lower bounds respectively. The subscripts l and u denote lower and upper bounds. Notice the addition of a pole in the lower bound specification $A(s)$ and a zero in the upper bound specification $B(s)$. This is done to lower the lower bound specification and to raise the upper bound specification in the high frequency range. Shown in figure 20 and figure 21 below are typical tracking performance specifications and their respective frequency bounds. It is also noted that depending upon the desired specifications, the steady state gain of A(s) and B(s) does not have to be unity. A tolerance band is usually acceptable. Should tracking specifications be desired B(s) and or A(s) should be unity.

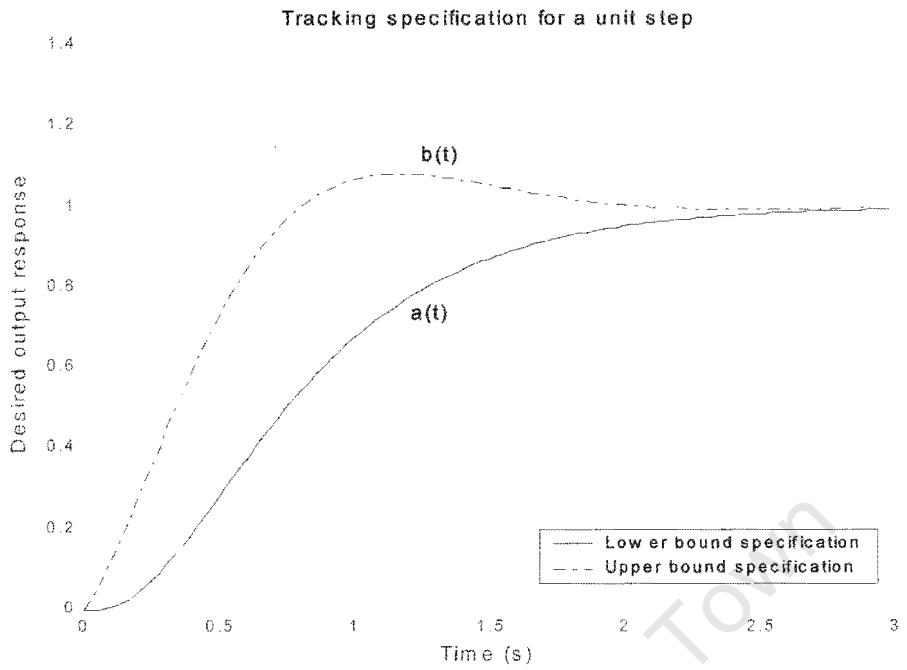


FIGURE 20: TRACKING SPECIFICATIONS ON THE SYSTEM OUTPUT.

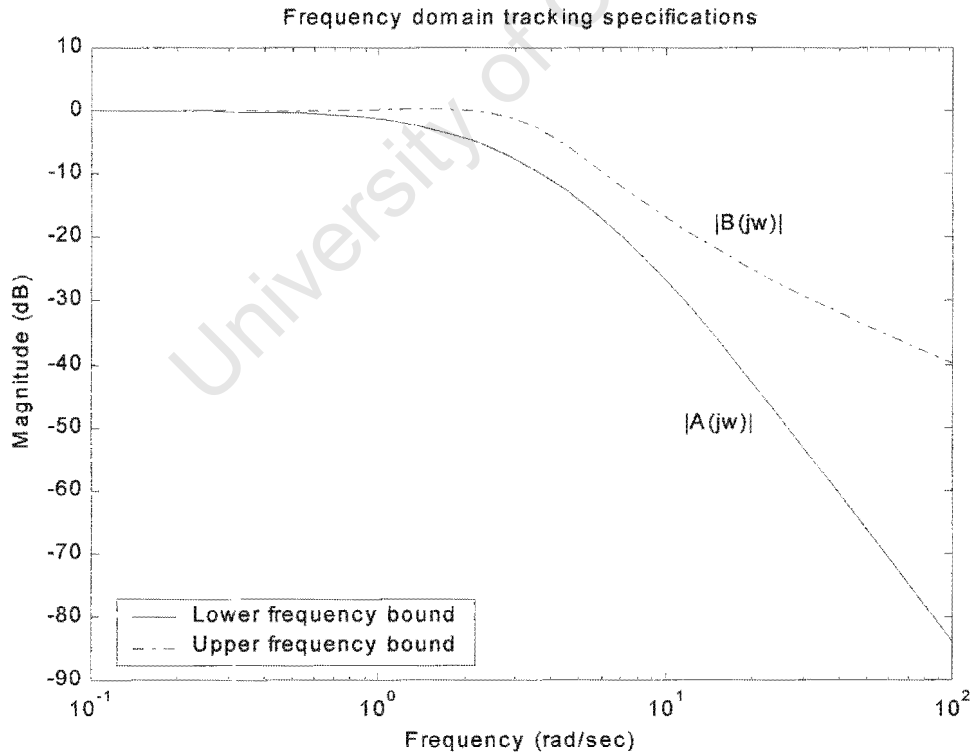


FIGURE 21: FREQUENCY DOMAIN TRACKING SPECIFICATIONS.

In addition to tracking specifications, disturbance rejection properties are also of concern in the QFT design. These are typically imposed on the closed loop transfer functions relating the output Y(s) to the input disturbance V(s) and the output disturbance signal D(s). Their transfer function models are as follows.

$$T_{D1} = \frac{Y(s)}{V(s)} = \frac{G(s)}{1 + L(s)} \leq W_{s1}$$

$$T_{D2} = \frac{Y(s)}{D(s)} = \frac{1}{1 + L(s)} \leq W_{s2}$$

Where W_{s1}, W_{s2} are the desired performance weights. These are generally chosen as constants or as a frequency dependent weight effective over the desired frequency range of disturbance rejection. Unlike the tracking bound specifications, which define, lower and upper bounds, only upper bounds are needed for disturbance rejection.

5.2. PLANT TEMPLATE GENERATION

Central to quantitative feedback theory is the concept of plant templates. These templates are geometric representations of all plant uncertainty described parametrically over a set of parameter values (Duncan, 1995). They are complex numbers representing the frequency response of the uncertain plant evaluated at discrete frequencies over a chosen set of frequencies. The design aim is to manipulate the position of the plant template on the Nichols chart from an undesirable position to a position satisfying the design specification. This must be done without any rotation of the plant template (Rao and Sen, 1999), this is farther exemplified by the fact that plant templates are represented in logarithmic form on the Nichols chart (ie. multiplication by a complex number (the controller) becomes the translation log(L) = log|L| + j arg(L)). This is illustrated as shown below.

$$G(s) = K \frac{\prod_{i=1}^{m} s + z_i}{\prod_{i=1}^{n} s + p_i}$$

$$\text{Where,} \quad z \in [a, b]$$
$$p \in [a_1, b_1]$$

Where z_i, p_i are zeros and poles of the uncertain plant G(s), while a and b are elements of the uncertain parameter set. The frequency response of all plants within the uncertain set is calculated and drawn on the Nichols chart. Depending upon the shape and complexity of these templates, template approximations must be made, usually achieved by manual construction. Figure 22 illustrates this.

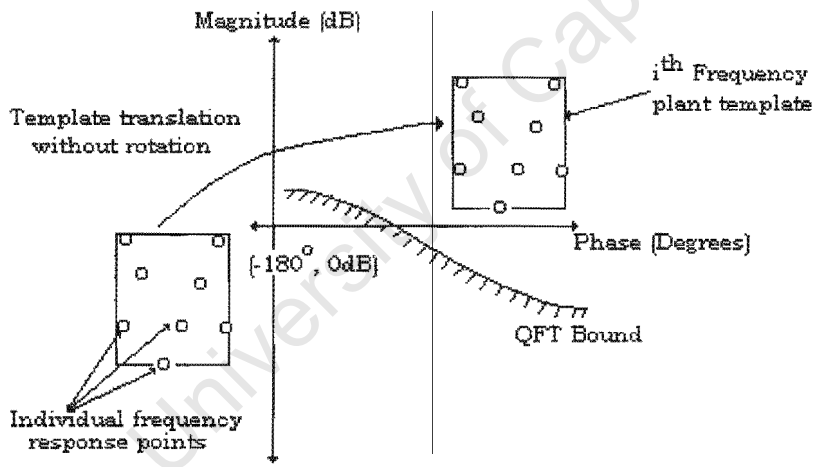


FIGURE 22: PLANT TEMPLATE MANIPULATION ON THE NICHOLS CHART.

5.3. QFT BOUNDS

As stated above QFT bounds are evaluated at discrete frequencies over a chosen frequency. This is followed by a QFT bound computation also evaluated at each frequency point. The underlining principle in computing QFT bounds is the concept of quadratic inequalities (Rodrigues and Chait and Hollot, 1997; Borghesani and

Chait and Yaniv, 1995). These bounds are derived from their closed loop transfer function models and the desired performance specifications.

5.4. QFT LOOP SHAPING

QFT loop shaping is the term given for the process of shaping the open loop transfer function (or loop transmission) model L(s) shown in figure 19, to achieve the desired system performance specifications. Shaping of the nominal loop transmission is accomplished by manipulating gain and phase relationships of the controller on the Nichols chart. This process follows a trial and error approach, where dynamic compensator elements such phase advance, phase lag, gain and integral terms are continually added to the structure of the nominal loop transmission until the desired QFT performance bounds are satisfied. The final controller then is the aggregation of all these dynamic elements. This procedure is tedious and time consuming, depending to a great deal on the experience of the designer.

Thus to facilitate the loop shaping process, developments have been made to automate the QFT controller design. This procedure entails the optimisation of controller gains with respect to an objective function, inclusive of the QFT performance bounds. Approaches to this problem include linear programming (Chait and Chen and Hollot, 1999; Bryant and Halikias, 1995), genetic algorithms (Chen and Ballance and Li, 1998) and nonlinear programming (Thompson and Nwokah, 1994). In the linear programming approach use is made of the Bode integral, which incorporates the gain and phase relationship of a G(s) transfer function model analytically (Gera and Horowitz, 1980). Though the automatic approach to loop shaping is convenient, it suffers from the disadvantage that the designer loses insight into the controller design. It has been proven that the optimal open loop frequency response or the Nichols chart contour of the nominal system L(s) must lie on or just above the QFT bound at each frequency point (Gera and Horowitz, 1980). Though this optimum condition is by no means a fixed constraint. Thus an acceptable approach would be to restrict the complexity of the controller and ensure that the high frequency dynamics are

minimised. The loop shaping objective then is to find an L(s), which satisfies the boundary constraints with minimum controller gain at high frequencies. The loop shaping approach adopted in this project will be based on manual loop shaping techniques.

5.5. PREFILTER DESIGN

Following the QFT controller design for disturbance rejection and robustness properties against model uncertainty, a prefilter design procedure is commenced. This is necessary to shift the closed loop transfer function T(s) to within an acceptable region satisfying the QFT tracking specifications. The controller K(s) shown in figure 19 only ensures that the variation in T(s) due to plant uncertainty is less that the difference of the upper and lower tracking bounds. That is,

$$\Delta \mid T(s) \mid \leq \mid B(jw) \mid - \mid A(jw) \mid .$$

Since the Bode plot is used in the prefilter design, the gain characteristics of the filter are manipulated until the tracking bounds are satisfied. The size of the uncertain set should not be chosen arbitrarily, but rather the one who built the plant should have fixed the uncertainty set

CHAPTER 6

6. CONTROLLER DESIGN APPLICATION

The preceding chapters presented the relevant theory and design approaches pertaining to H_∞ optimal control, quantitative feedback theory and fuzzy logic control. The common characteristic feature of these methods are, that they inherently consider plant uncertainty, whether it be parametric (structured uncertainty) or non–parametric (unstructured uncertainty). The application of these methods have varied substantially from complex systems such flight control (Wu and Grimble and Breslin, 1998) to relatively simpler systems such as consumer electronics (Reznik, 1997). In this chapter we present the application of the above controller strategies to an experimental turbo – alternator system. The experiment is aimed at emulating the operation of electric power systems, more specifically the load frequency control problem. It is a well known fact that power system parameters are constantly varying due to load changes, maintenance and temperature effects, and etcetera. Thus with the load frequency control problem in mind, the designed controller is required to regulate the system frequency satisfactorily and to account for system uncertainty and load disturbances, thus favourably motivating the use of the above controller design methods to load frequency control.

6.1. THE LABORATORY TURBO–ALTERNATOR SET

In conventional power systems, frequency control is achieved by manipulating the reference input of the speed governor, thereby increasing or decreasing the system generation. This process is simulated by realising the operation of load frequency control on an experimental turbo–alternator set, which consists of a DC motor driving an AC generator. This is equivalent to a single control area in power system terminology, where the control area has an equivalent droop characteristic and prime mover or turbine model (see Chapter 2). In our experiment the motor represents the

prime mover and the three–phase generator the electrical generating unit of the power system. The load frequency controller forms part of the supervisory control of the control area, applies control to the speed reference input of the turbine governor.

The previously discussed controller design methods namely, H_∞ optimal control, quantitative feedback theory and fuzzy logic control are proposed as the supervisory controllers for load frequency control, the PI controller is designed for comparative purposes. A comparative study between these methods is thus undertaken commenting on their effectiveness to the load frequency control problem. Shown figure 23 below is a block diagram of the experimental power system.

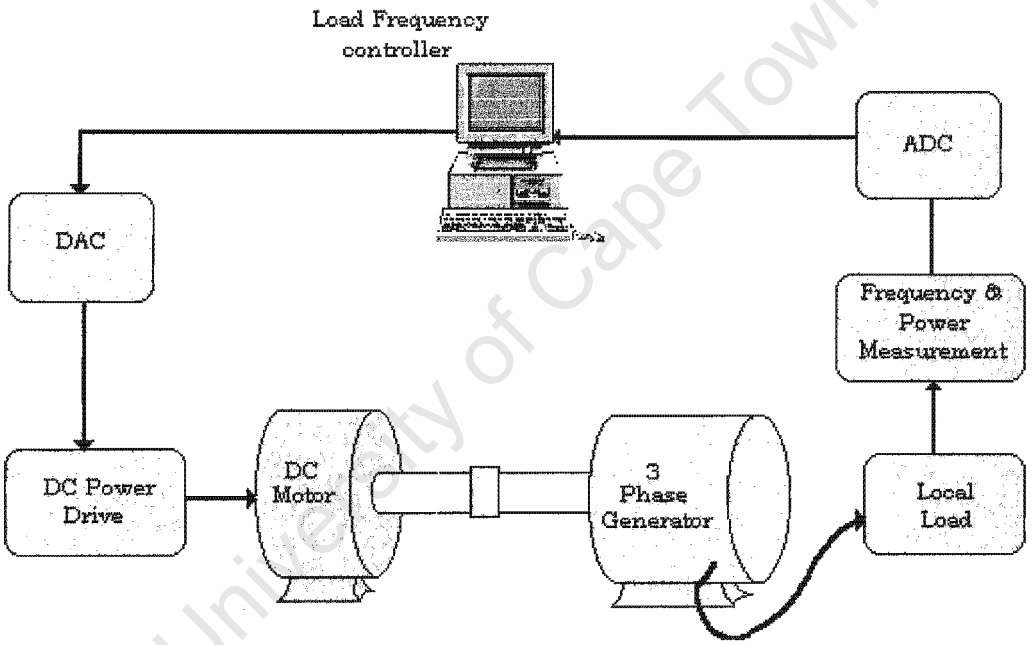


FIGURE 23: EXPERIMENTAL LOAD FREQUENCY CONTROL

Not shown in figure 23 are the excitation systems for both the DC motor and the generating unit. In addition the isolation circuitry protecting the micro–computer is also not shown (refer to appendix 6). The nominal specification data for the motor and generator are as follows. The DC motor used is a compoundly wound Mawdsley motor, rated at 250V at 28A with a rate speed of 3000rpm. The generator is a three phase generator, with 400V, 7.25A and 5KVA output voltage, current and power

ratings respectively. It operates at 50Hz with a power factor of 0.8. The excitation voltage used is 100V at 5A.

The output from the digital to analogue converter is a ±10V signal forming the control input, however the acceptance signal from the DC power drive is a signal in the voltage range between 0 and 10V. Thus appropriate voltage scaling of the DAC signal is required to form the control input u(t). A manual excitation system by means of a variac is used for the excitation of the field coils of the generator. The assumption is made that the effect of the excitation voltage on the system frequency is negligible. Though the turbo–alternator set differs in size and complexity from that of large power systems, it does provide a means of verifying the relative performance and quality of control of various controller design strategies. Though is it not the aim of the thesis to improve significantly on the speed of response, but on the robustness properties of the proposed controllers.

6.2. SYSTEM MODELLING AND PARAMETER IDENTIFICATION

Characterising the dynamic behaviour of a system by means of a mathematical model forms an integral part of control system design. Figure 26 shows the model of the open loop system, where G(s) is the plant and $G_d(s)$ is the disturbance model. It is required that these models accurately describe the various modes of the system under investigation. Methods of obtaining such models include rigorous mathematical analysis based on the dynamic properties of the system or via experimental data. The latter procedure is commonly applied taking the form of step test data. This is motivated by its simplicity and its ability to examine both the high and low frequency characteristics of the plant. Following this procedure unit step response data shown in figure 24 are obtained for the system illustrated by figure 23.

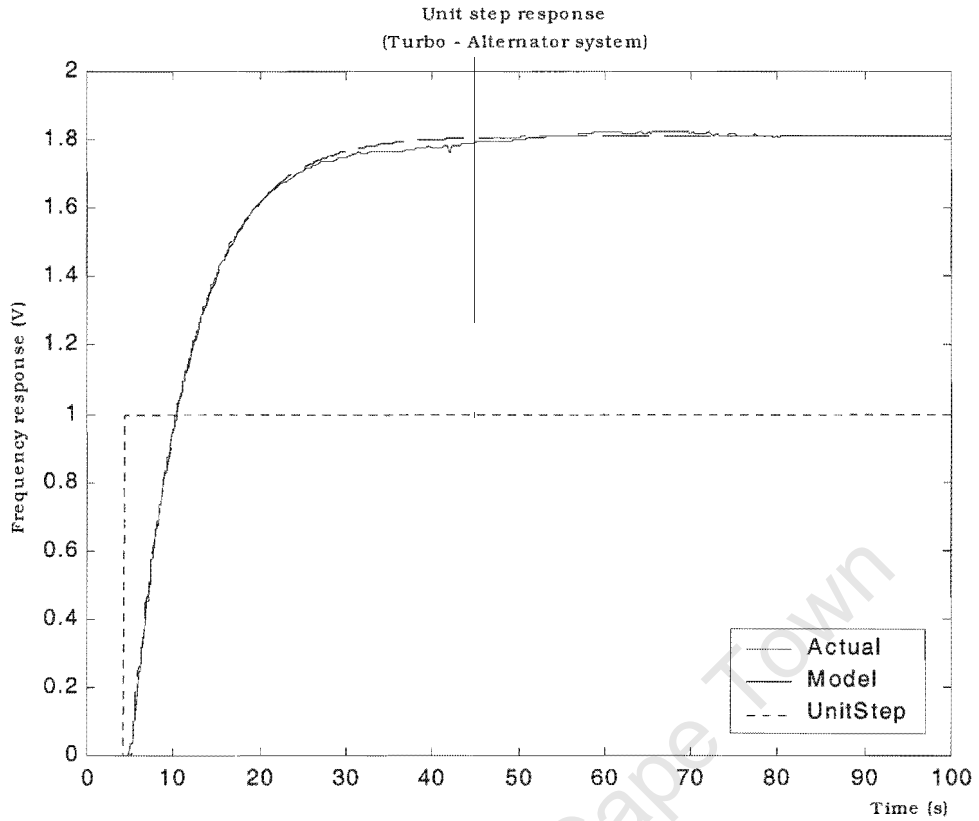


FIGURE 24: UNIT STEP RESPONSE FOR THE TURBO – ALTERNATOR SYSTEM.

To improve on the matching properties of the model an optimisation procedure was employed, using standard function routines (fminimax.m and lsqnonlin.m) found in the Matlab optimisation toolbox.

The resulting plant model for the system is shown below.

$$G(s) = \frac{10.02}{s^3 + 0.72s^2 + 37.33s + 5.53} e^{-1.00s}$$

Poles at s =
-0.28576 ± 6.0962i
-0.14848

The time delay of one second was included into the model since by examining the time response above (figure 24), there was a delay in response. It is a second order

system with a one second delay in response time. Second order because we have the DC power drive (time constant assumed negligible), DC motor and generator in series, with an associated time constant, but more specifically analysing the block diagram for a single control area shown in figure 5 of chapter 2 results in a third order system (see appendix 1), this motivates the structure of the above model by matching model parameters to fit the actual response data. Similar procedures are followed for a disturbance response due to a 3KW load increase is shown in figure 25 below.

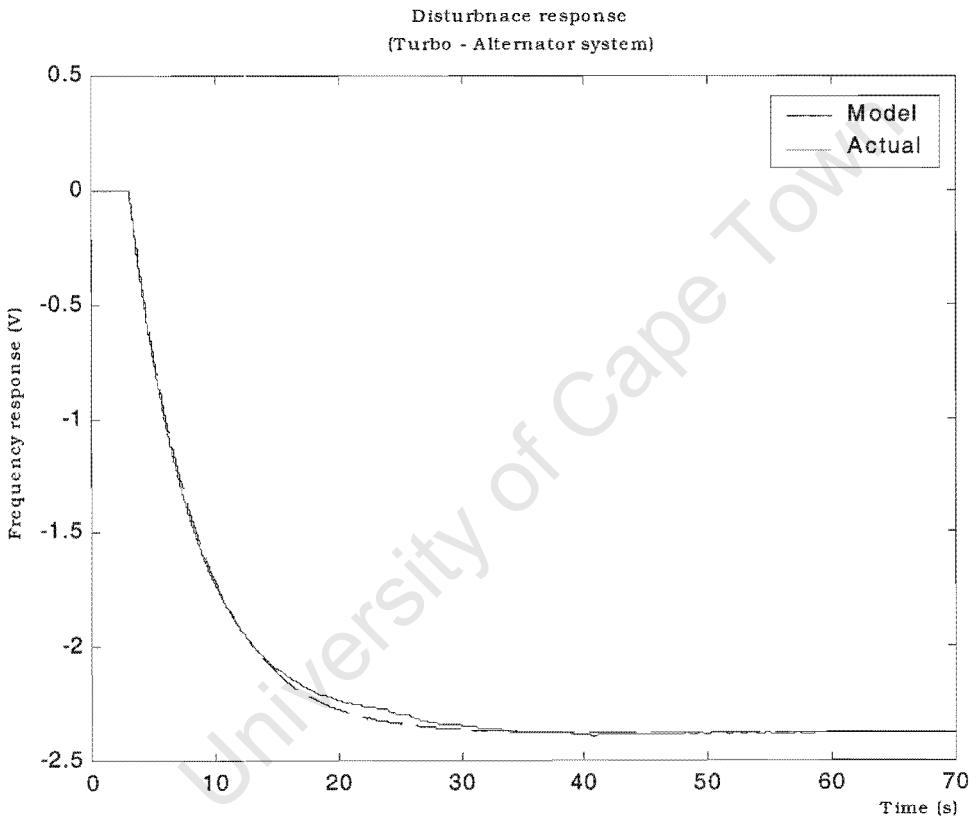


FIGURE 25: DISTURBANCE RESPONSE FOR TURBO – ALTERNATOR SYSTEM.

The resulting model is as follows.

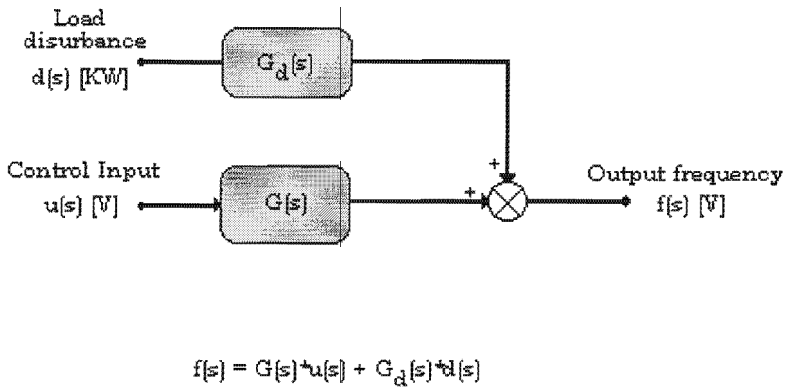


FIGURE 26: BLOCK DIAGRAM OF SYSTEM MODEL.

$$G_d(s) = \frac{+512}{s^3 + 0.82s^2 + 3541s + 646} \quad \frac{[V]}{[KW]}$$

Poles at s =
$-0.31878 \pm 59.504i$
-0.18244

From figure 5, the disturbance model (G_d(s) above) should have the exact same denominator term as G(s). However this is not so, since different identification runs were applied. Thus in block diagram form the model of the turbo–alternator system is shown in figure 26 above, where u(t) is the control input in volts (V), d(t) the load disturbance in kilowatts (KW) and the output frequency f(t) in volts (V). An experiment conducted which involved synchronising the experimental unit with mains proved to be unsuccessful due to large difference in "plant size" between that of the experimental unit and that of the mains supply.

6.3. SAMPLING TIME SELECTION

Since the control laws are implemented digitally, the selection of sampling times are important to the effectiveness of the designed controllers. Should the sampling time be too large, it may lead to aliasing, where high frequency signals exhibit low

frequency characteristics, which may lead to instability. On the other hand should the sampling time be too small there may be excessive control action due to the introduction of high loop gain. Thus a procedure for guiding the selection of sampling times, are as follows (Zafiriou and Morari, 1986; Kanniah et al, 1984).

Sampling time selection is also greatly influenced by the hardware cost and by floating point accuracy (overcome by using Goodwin's delta transform method). Selecting the sampling time too slow limits the achievable bandwidth of the open loop system and hence the performance. An initial choice for the sampling time, for the control system shown in figure 23 above is $t_s = 0.3s$. A bode plot of the plant G(s) is shown in figure 27 below, it shows that most of the system energy is below 6rad/s upon which it attenuates by approximately –60dB per decade. Following the Shannon sampling criteria which states that the sampling time should be at least twice that of the operating frequency of the system, in our case w = 6 rad/s, the following calculations are made.

$$Since \quad w = 2*\pi*f$$

$$\Rightarrow \quad f_s = 2*\left(\frac{w}{2*\pi}\right) \qquad Shannon's\ criteria$$

$$\Rightarrow \quad T_s = \frac{1}{f_s} = \frac{w}{\pi} = 1.9s$$

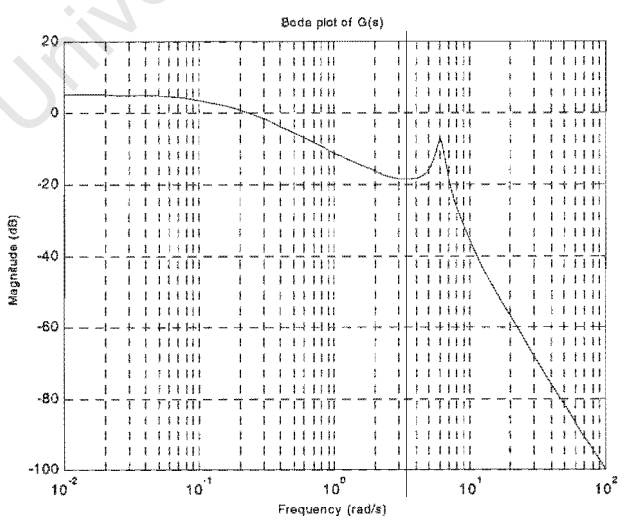


FIGURE 27: SYSTEM BODE PLOT

Since the initial sampling time of t_s = 0.3s is well below the calculated time of ts = 1.9s, subsequent work will be based on this initial value. Examining the complex pole of G(s) we see that there is a frequency component of 6.0926rad/s. This implies a period of approximately 1s ($\omega = 2\pi f, T = \frac{1}{f}$). Similarly, the complex pole of $G_d(s)$ yields a period of 0.1s. Thus the speed of response of $G_d(s)$ is much faster than that of the plant G(s). This implies that the sampling time should be less than 0.1s. As noted the above sampling time of 0.3s contradicts the previous discussion. However, on the practical experiment t_s = 0.3s worked well.

6.4. APPLICATION OF H_∞ OPTIMAL CONTROL

The controller design specifications in an H_∞ design are contained in the weighting functions. Thus by appropriate choice of these functions the desired closed loop specifications are imposed on the controlled system. The commonly encountered H_∞ design problem is the mixed sensitivity problem, where the sensitivity function S(s) and the complementary sensitivity function T(s) are shaped by frequency dependent transfer functions W_1 and W_3 respectively, imposing constraints on their respective magnitudes. In addition, for minimising the amount of control input utilised, a constraint W_2 can be placed on the control sensitivity function M(s). This is important in reducing excessive control action, thereby reducing the wear and tear on system actuators. This is important to power system control, since the amount of control action utilised can be minimised by the adjustment W_2.

The mixed sensitivity problem is posed as follows.

$$\min \left\| \begin{matrix} W_1 S \\ W_2 M \\ W_3 T \end{matrix} \right\|_\infty \leq \gamma$$

Gamma (γ) represents the number below, which the infinity norm of the mixed sensitivity should fall. Gamma is iteratively decreased until the minimum value for gamma is found, which satisfies the performance and robustness properties imposed by the weighting functions. In addition the controller should stabilise the nominal plant. Synthesis algorithms for solving the mixed sensitivity problem are available, such as the Doyle and Glover algorithm (Doyle and Glover et al, 1989). The Matlab functions hinf and hinfopt from the robust control toolbox solves for the H_∞ controller given the weighting transfer functions.

The sensitivity weight W_1 determines the performance and disturbance rejection properties of the controlled system, while W_3 manipulates the robustness properties of the system due to model uncertainty. The greater the uncertainty rejected by the controlled system the more robust the system is said to be. Based on the relation,

$$S + T = 1 \qquad \textit{(SISO system)}$$

care should be taken in the selection of the weighting functions. Since it is impractical to satisfy this relation over all frequencies, a frequency domain compromise must be reached. Thus from classical control theory, disturbances occur over low frequency ranges, and sensor noise at much higher frequencies. This implies that the sensitivity function should be small at low frequencies and the complementary sensitivity function small at high frequencies to reject the effect of sensor noise. In addition, for setpoint tracking W_3 should be 0dB (gain of 1) over the low frequency range.

6.4.1. CONTROLLER DESIGN 1

An initial choice for the weighting functions are first order transfer functions of the form shown below.

$$W_i(s) = K\frac{\frac{1}{M}s + \omega_B}{s + A * \omega_B}$$

Where, K is the gain, M the high frequency gain, A the low frequency gain and ω_B the desired system bandwidth. This is illustrated in figure 28 below. We notice that as the A parameter of the weight is decreased the system approximates a type 1 system, this is important for zero steady state error. The slope of the curve is 20 dB per decade.

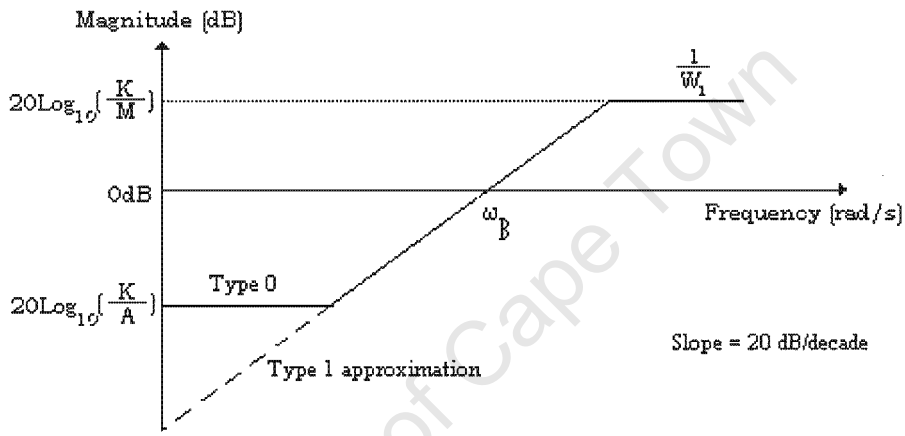


FIGURE 28: BODE PLOT OF TYPICAL WEIGHTING FUNCTION.

Using the Nyquist plot for a perturbed system and comparing it to the nominal system, the following condition can be shown to be true (Bode, 1947; Horowitz, 1963; Kwakernaak, 1993).

$$\frac{|L(j\omega) - L_0(j\omega)|}{|L_0(j\omega)|}.|T_o(j\omega)| < 1$$

$$|\Delta_m| = \frac{|L(j\omega) - L_0(j\omega)|}{|L_0(j\omega)|}$$

Where $L_0(jw)$ is the nominal open loop transfer function, $L(jw)$ is the uncertain (perturbed) system and $T_0(jw)$ is the nominal complementary sensitivity function. The

term Δ_m is a relative measure of the size of the uncertainty. Thus when Δ_m is a frequency dependent function, we see its relevance to H_∞ optimal control via the relation,

$$\|W_3.T_0\|_\infty < 1$$

$$|\Delta_m(j\omega)| \leq |W_3(j\omega)| \ldots\ldots\ldots\ldots\ldots\ldots \forall\, \omega$$

The term Δ_m is thus the multiplicative representation of uncertainty. Therefore the weighting function W_3 incorporates robustness properties into the system. Thus the greater the magnitude difference between the upper and lower frequency ranges of W_3, the more robust the system becomes, since more plant uncertainty is tolerated. This is illustrated in figure 29 below, where Δ_m for a ±20% parameter variation uncertainty is shown with the W_3 shown below. This indicates that the controlled system is expected to tolerate ±20% parameter uncertainty reasonably well.

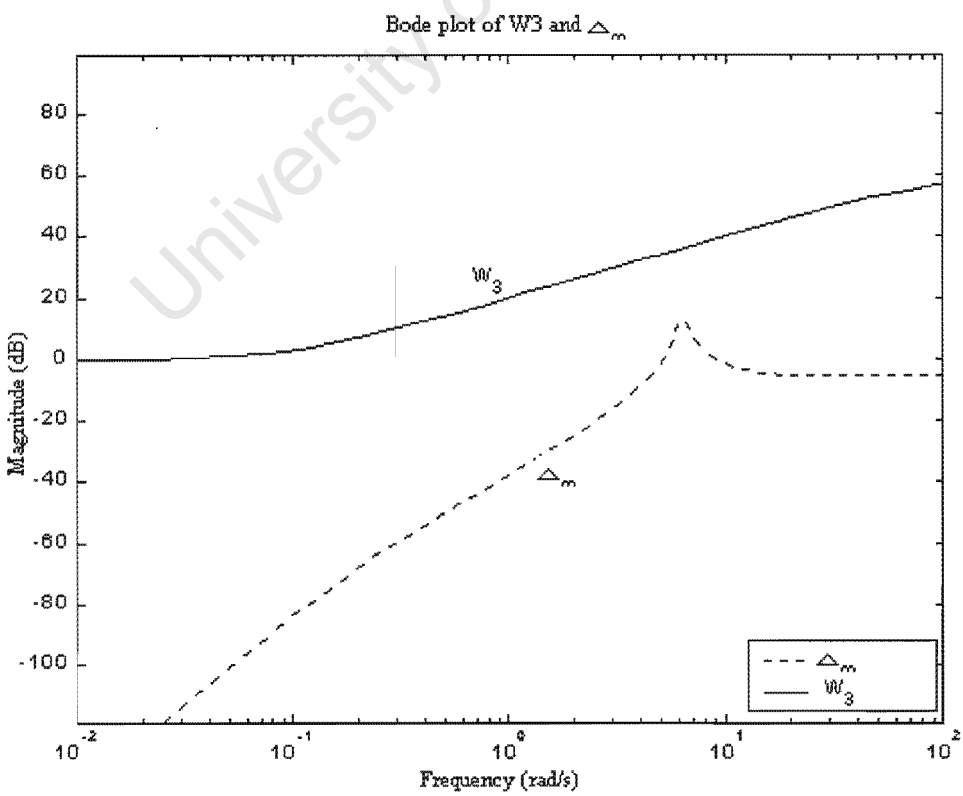


FIGURE 29: WEIGHT W_3 AND Δ_M FOR ±20% MULTIPLICATIVE UNCERTAINTY.

Selecting weighting functions as follows, theses values are selected from the Monte Carlo simulation results shown in appendix 2.

$$W_1 = \frac{s+0.1}{s+0.0001} \qquad (M = 0.01,\ A = 10^{-6},\ w_B = 10,\ K = 10^{-3})$$

$$W_2 = 3.5$$

$$W_3 = \frac{1000s+100}{s+100} \qquad (M = 0.01,\ A = 10,\ w_B = 10,\ K = 10)$$

Their corresponding bode plots are shown in figure 30 below.

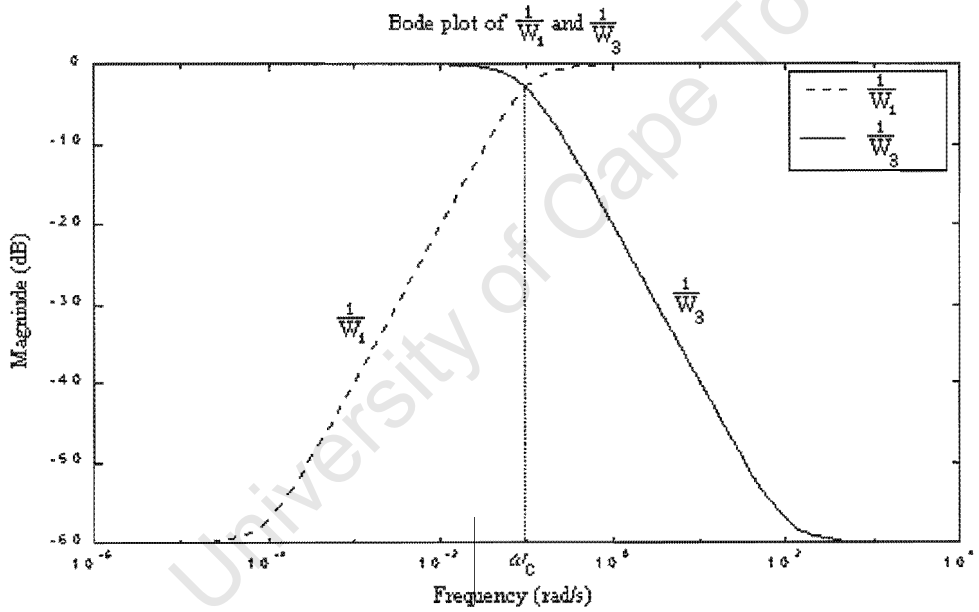


FIGURE 30: BODE PLOTS OF $1/W_1$ AND $1/W_3$.

The importance of the crossover frequency (ω_c) is that it affects the speed of response of the system, the larger this frequency the faster the closed loop response becomes. Weight W_2 is chosen as 3.5 since it was observed that the control input exhibits a high frequency spike depending on the value of W_3, thus W_3 was iteratively manipulated until an acceptable control input was obtained (see appendix 2, section A2.2).

Using the Matlab function "hinfopt", which iteratively solves for the H_∞ controller yielding the minimum infinity norm, the following controller was obtained. The plant described by section 6.2 above was used in the synthesis process. Dead time was approximated by a first order Pade approximation.

$$K_\infty(s) = \frac{1.611\,s^5 + 165.5\,s^4 + 493.3\,s^3 + 5556\,s^2 + 1.133e4\,s + 1783}{s^6 + 109.5\,s^5 + 1023\,s^4 + 7422\,s^3 + 2.404e4\,s^2 + 2.873e4\,s + 2.873}$$

$\gamma = 0.4258$ (optimal gamma)

The poles and zeros of this controller are tabulated in table 2 below.

	Plant G(s)	Controller $K_\infty(s)$	Closed loop system T(s)	Weight $1/W_3(s)$
Poles	-0.27619 + 5.6709i *	-100 *	-100 *	-0.1
	-0.27619 - 5.6709i *	-2.3302 + 6.0848i	-2.3416 + 6.1246i	
	-2 *	-2.3302 - 6.0848i	-2.3416 - 6.1246i	
	-0.17161 *	-2.4264 + 0.93793i	-0.27619 + 5.6709i *	
		-2.4264 - 0.93793i	-0.27619 - 5.6709i *	
		-0.0001	-2.3479 + 0.28641i	
			-2.3479 - 0.28641i	
			-2 *	
			-0.17161 *	
			-0.13436	
Zeros	2	-100 *	-100 *	-100
		-0.27619 + 5.6709i *	-0.27619 + 5.6709i *	
		-0.27619 - 5.6709i *	-0.27619 - 5.6709i *	
		-2 *	-2 *	
		-0.17161 *	2	
			-0.17161 *	

TABLE 2: POLES AND ZEROS OF H_∞ CONTROLLER 1.

Examining table 2 we see that the controller zeros cancels all poles of the plant G(s), examine tables in appendix 2. Secondly the H_∞ synthesis method manipulates the controller poles such that the closed loop poles of the system are in the vicinity of poles of the inverse of the weighting functions. This can be seen by observing the pole

positions of the complementary sensitivity function T(s) and the poles of $\frac{1}{W_3(s)}$ as shown in table 2. Since $\frac{1}{W_3(s)}$ has a dominant pole at s = -0.1, likewise T(s) has a dominant pole at s = -0.13436, thus the same characteristics are inferred. The poles and zeros marked by (*) indicate the poles and zeros that are cancelled by the controller K(s) and T(s). The red (*) denotes the plant poles that are cancelled by the controller zeros. We note that T(s) has two complex poles positioned at s = -2.3416 ± 6.1246i and at s = -2.3479 ± 0.28641i respectively, these correspond to controller poles s = 2.3302 ± 6.0848i and s = -2.4264 + 0.93793i. These are to nullify the effect of the nonminimum phase zero at s = 2 (as a result of the Pade approximation). The H_∞ synthesis method, more specifically the mixed sensitivity formulation, solves the problem of right half plane poles (unstable poles) by making use of a unique jω axis shifting property.

This characteristic of pole zero cancellation may be undesirable (Kuo, 1991) and solutions to circumvent this property are discussed by Sefton and Glover (Sefton and Glover, 1990) and Kwakernaak (Kwakernaak, 1993). A method that avoids this problem is the McFarlane and Glover loop shaping approach (McFarlance and Glover, 1992) where the open loop system is shaped by pre and post weighting functions on G(s) and the two degree of freedom H_∞ controller structure (Skogestad and Postlethwaite, 1996).

Converting the above controller to discrete time was realised by the bilinear transform with a sampling time of t_s = 0.3s. Time domain responses are shown below in figure 31, comparing the simulated results with that of the actually achieved response. Step and disturbance responses are shown in figure 31 and 32 with ±20% and ±30% parameter uncertainty.

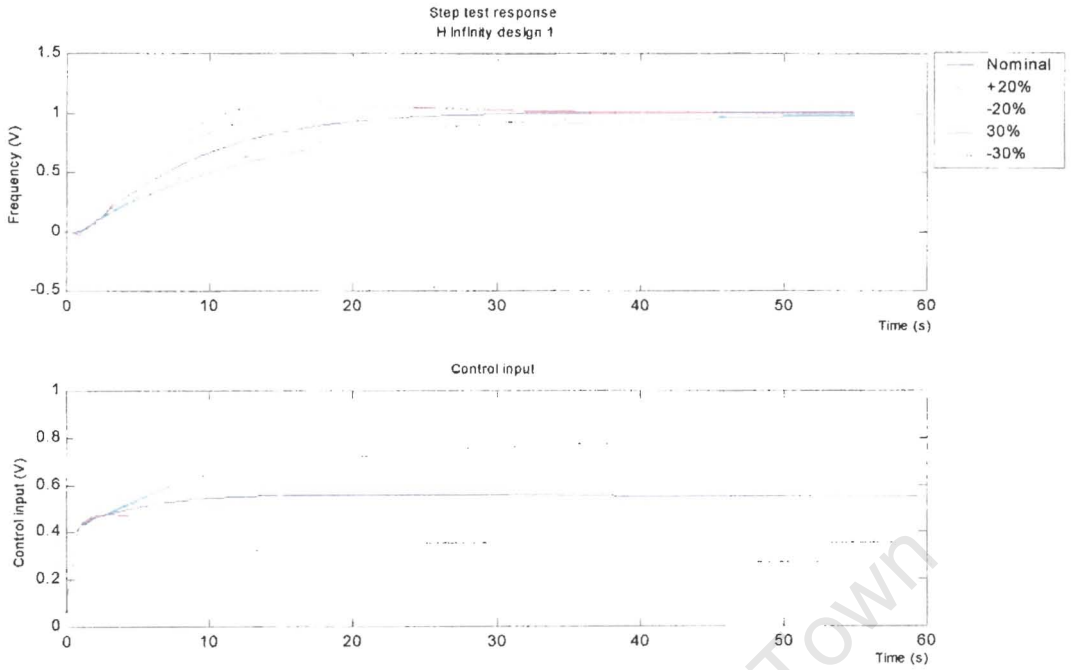


FIGURE 31: STEP RESPONSE FOR H_∞ CONTROLLER DESIGN 1.

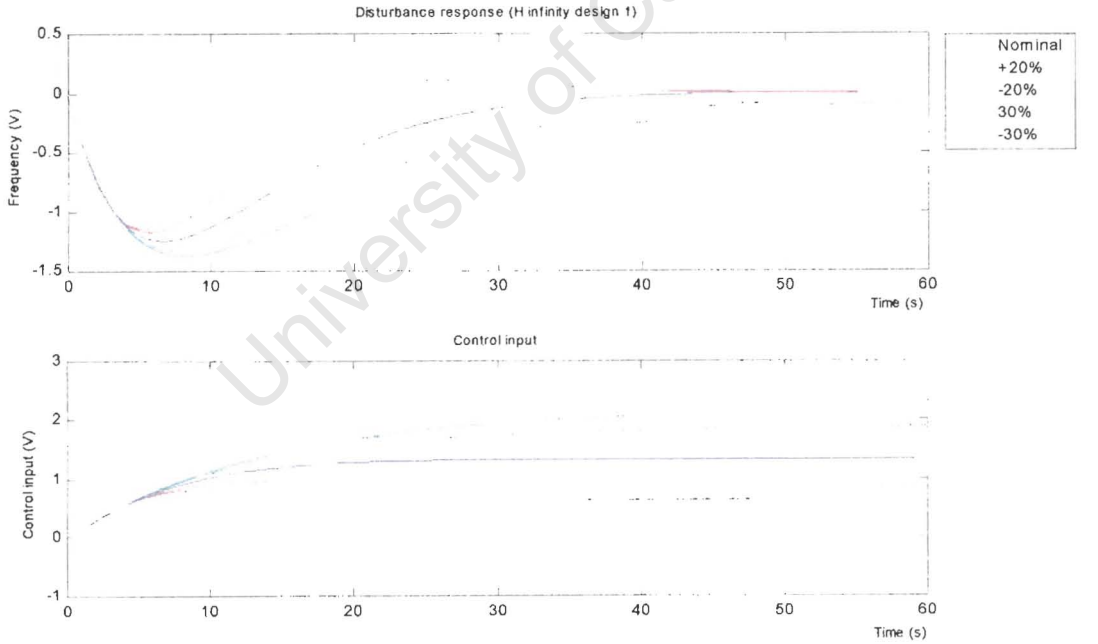


FIGURE 32: DISTURBANCE RESPONSE FOR H_∞ DESIGN1.

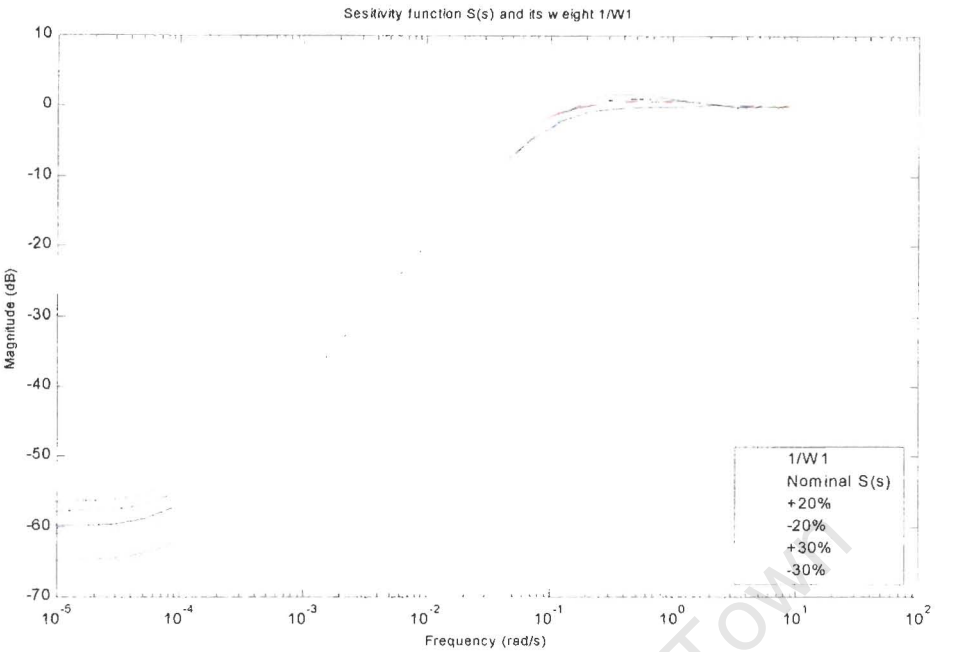


FIGURE 33: SENSITIVITY FUNCTION FOR VARIOUS PARAMETER VALUES.

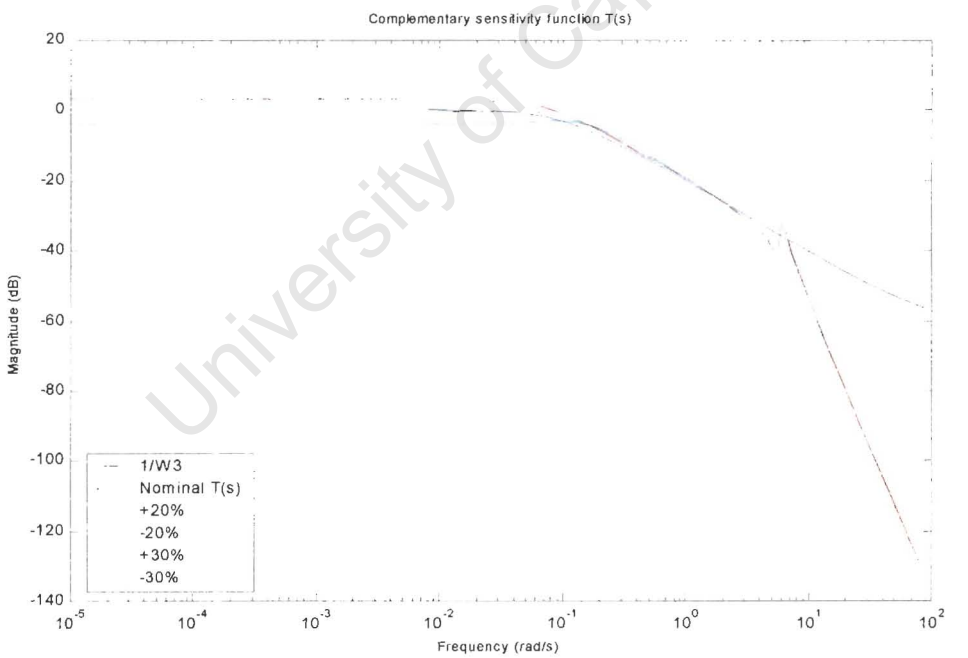


FIGURE 34: COMPLEMENTARY SENSITIVITY FUNCTIONS FOR VARIOUS PARAMETER VALUES.

Bode plots of the closed loop sensitivity functions S(s) and complementary sensitivity functions T(s) are shown above in figures 33 and 34. It can be seen that there is a

good correlation between the desired and actual frequency responses up to 2rad/s, however, the complementary sensitivity function deviates from that of the function $1/W_3$ at high frequencies by attenuating much faster than that of $1/W_3$. This does not pose any problem of concern since it will attenuate the effect of high frequency noise more effectively and is deemed satisfactory.

Examining the time response curves shown in figure 31, it is seen that steady state error is not zero, as can also seen from the structure of the controller. This is attributed to the fact that the level of integral action obtained by employing the weights shown above, depends upon the A parameter of W_1. Therefore, for pure integral action, A should ideally be zero. Actual time domain results are shown in figures 35 and 36. It is seen that the actual response is similar to that of the simulated response.

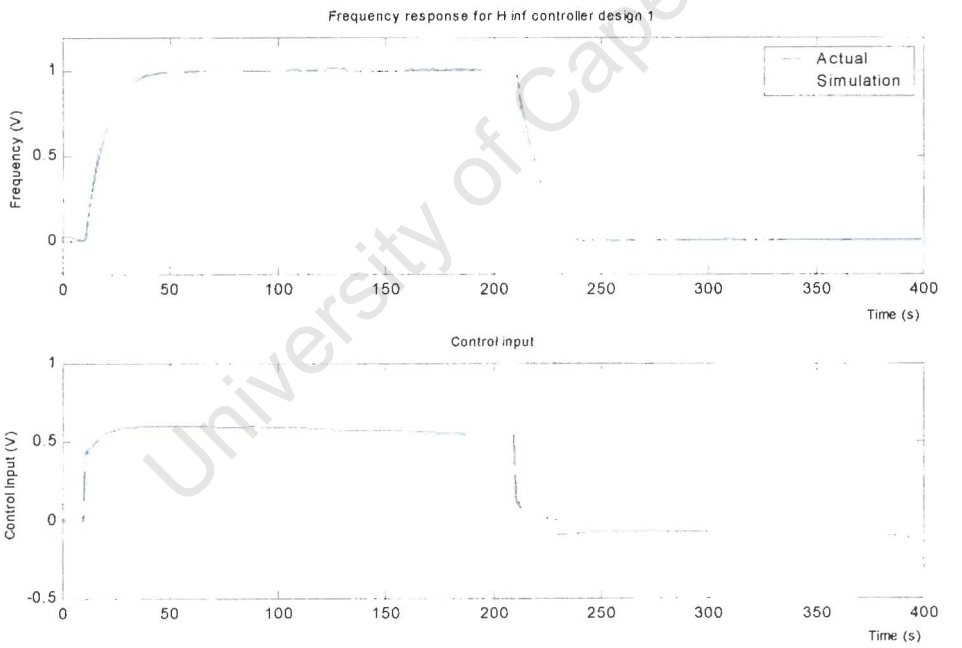


FIGURE 35: ACTUAL AND SIMULATED RESPONSE FOR H_∞ DESIGN 1.

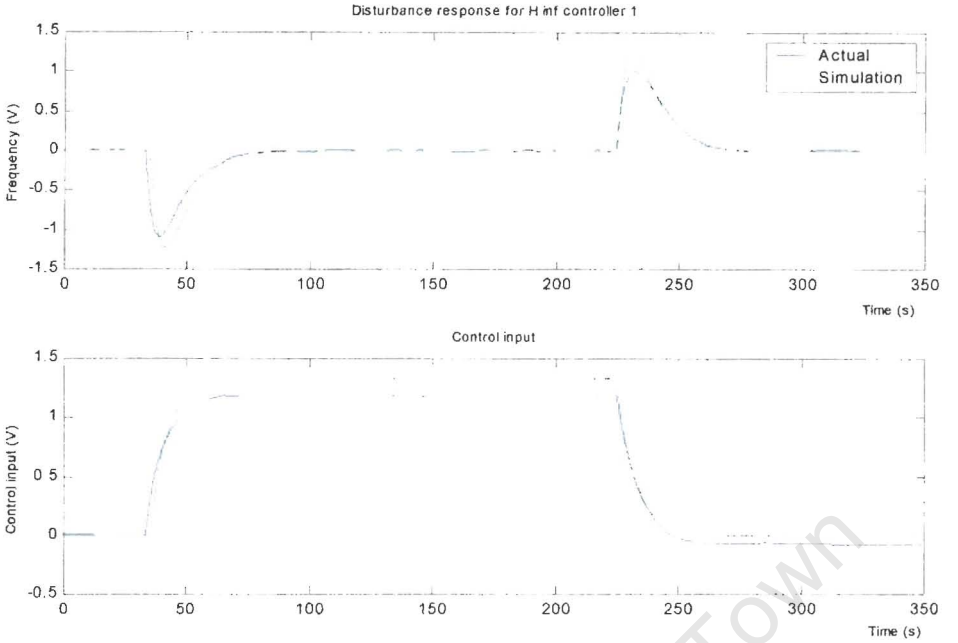


FIGURE 36: DISTURBANCE RESPONSE FOR H_∞ DESIGN 1.

6.4.2. CONTROLLER DESIGN 2

Choosing the weighting functions as follows yields the results shown below. The aim of the design is to implicitly add integral action into the controller.

$$W_1 = \frac{(s+0.65)^2}{s(s+3.65)}$$

$$W_2 = 2.25$$

$$W_3 = \frac{1000s+10}{s+100} \qquad (M = 0.01,\ A = 10,\ w_B = 10,\ K = 10)$$

The double zero in W_1 is to ensure that the weight is both proper and that there is sufficient roll at high frequencies to guarantee that the closed loop gain of S(s) is 0dB within this frequency range. Bode plots and response curves are shown in figures 37, 38 and 39 below.

Yielding the resulting controller.

$$K_\infty(s) = \frac{0.8576s^6 + 91.74s^5 + 637.6s^4 + 4075s^3 + 1.862e4s^2 + 2.662e4s + 4039}{s^7 + 113.8s^6 + 1493s^5 + 1.224e4s^4 + 5.075e4s^3 + 9.253e4s^2 + 6.313e4s + 6.843e\text{-}011}$$

$\gamma = 0.5313$ (optimal gamma)

As seen from the structure of the above controller, the complexity of the controller increases as the order of the performance weights increases. In comparison with design 1, design 2 has a higher optimal gamma value which indicates that design 1 is the more optimal design, in addition the integral action of the controller (design 2) improved as seen by the larger steady state gain. A table tabulating the poles and zeros of the controller shown above are given below in table 3.

	Plant G(s)	Controller $K_\infty(s)$	Closed loop system T(s)	Weight $1/W_3(s)$
Poles	-0.27619 + 5.6709i *	-100 *	-100 *	-0.1
	-0.27619 - 5.6709i *	-3.3831 + 6.5808i	-3.3769 + 6.596i	
	-2 *	-3.3831 - 6.5808i	-3.3769 - 6.596i	
	-0.17161 *	-3.65	-0.27619 + 5.6709i *	
		-1.6712 + 0.60492i	-0.27619 - 5.6709i *	
		-1.6712 - 0.60492i	-3.6004	
		-1.084e-015	-2 *	
			-2.063	
			-1.1908	
			-0.17161 *	
			-0.15065	
Zeros	2	-100 *	-100 *	-100
		-0.27619 + 5.6709i *	-0.27619 + 5.6709i *	
		-0.27619 - 5.6709i *	-0.27619 - 5.6709i *	
		-4.2575	-4.2575	
		-2 *	2	
		-0.17161 *	-2 *	
			-0.17161 *	

TABLE 3: POLES AND ZEROS FOR H_∞ DESIGN 2.

As observed from table 3, the H_∞ controller zeros cancels the poles of the plant G(s). In addition we also see the placing of redundant pole and zeros at s = -100. Thus appropriate controller reduction can be done by the removal these spurious poles and zeros.

Bode plots of S(s) the sensitivity function (figure 37), T(s) the complementary sensitivity function (figure 38) and time domain results (figures 39, 40 and 41) are shown below. Time domain results are shown for ±20% and ±30% uncertainty (see note below) variation in the pole and zero positions of the plant.

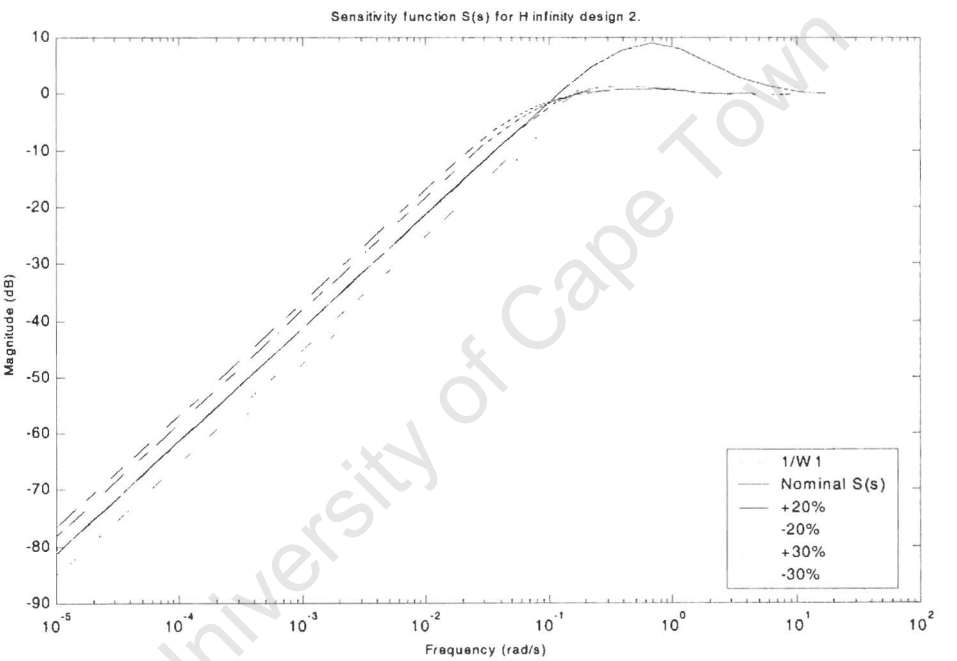


FIGURE 37: BODE PLOT OF SENSITIVITY FUNCTION FOR H_∞ DESIGN 2.

Note: The plant model is of the form shown below, where z and p denotes zeros and poles respectively. N with its respective subscripts is the number of poles and zeros. The uncertainty considered is ±20% and ±30% of the pole/zero positions from their nominal value (ie. z_i ±20%).

$$G(s) = \frac{\prod_{i_z=1}^{N_z}(s + z_{i_z})}{\prod_{i_p=1}^{N_p}(s + p_{i_p})}$$

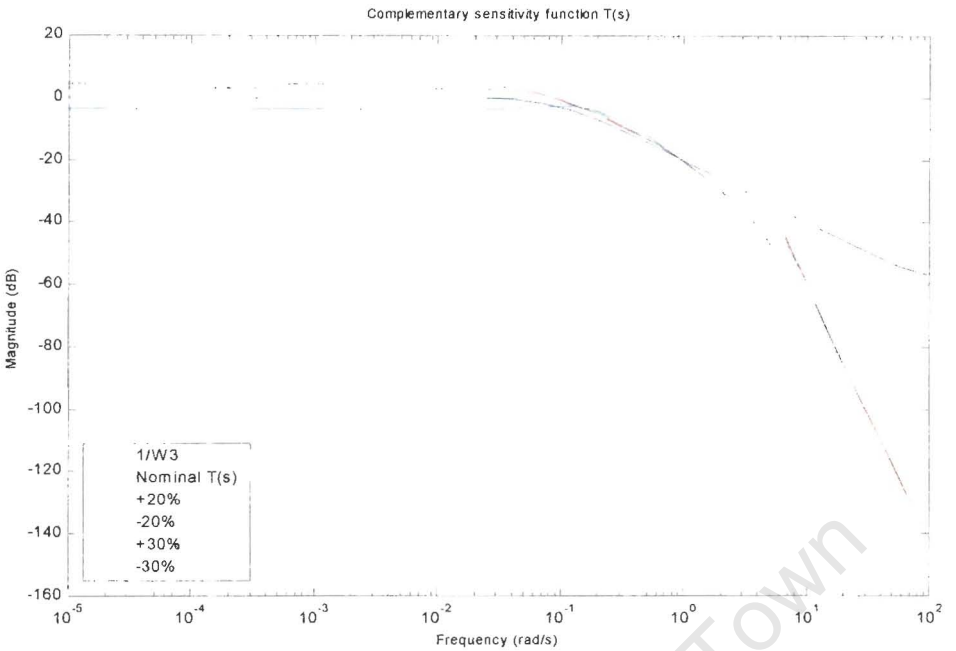


FIGURE 38: BODE PLOT OF THE COMPLEMENTARY SENSITIVITY FUNCTION FOR H_∞ DESIGN 2.

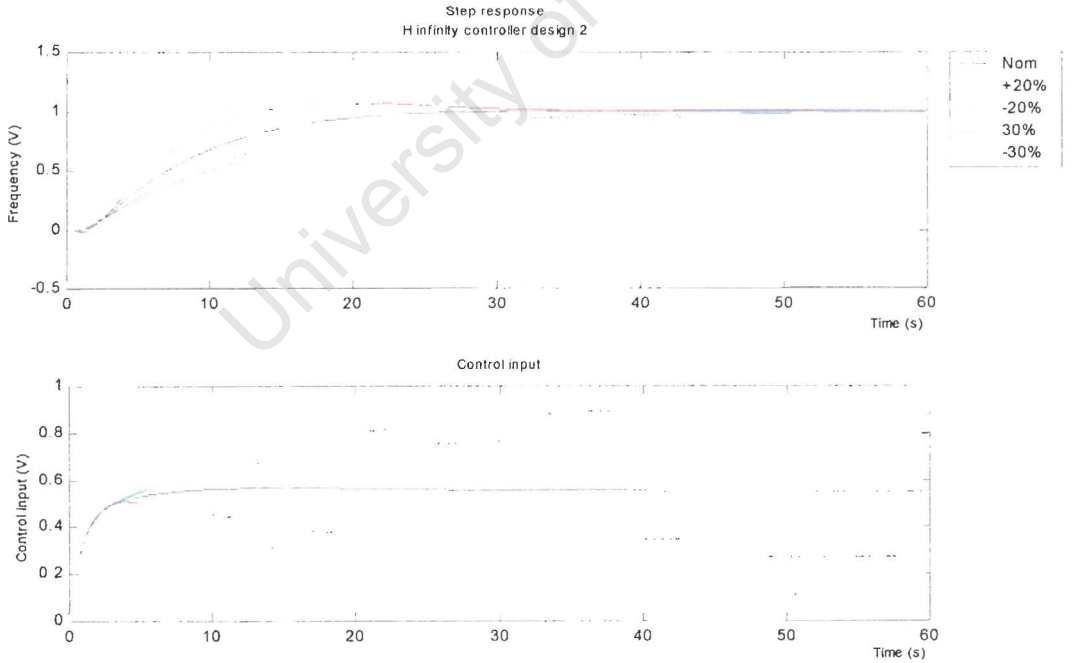


FIGURE 39: STEP RESPONSE FOR H_∞ CONTROLLER DESIGN 2.

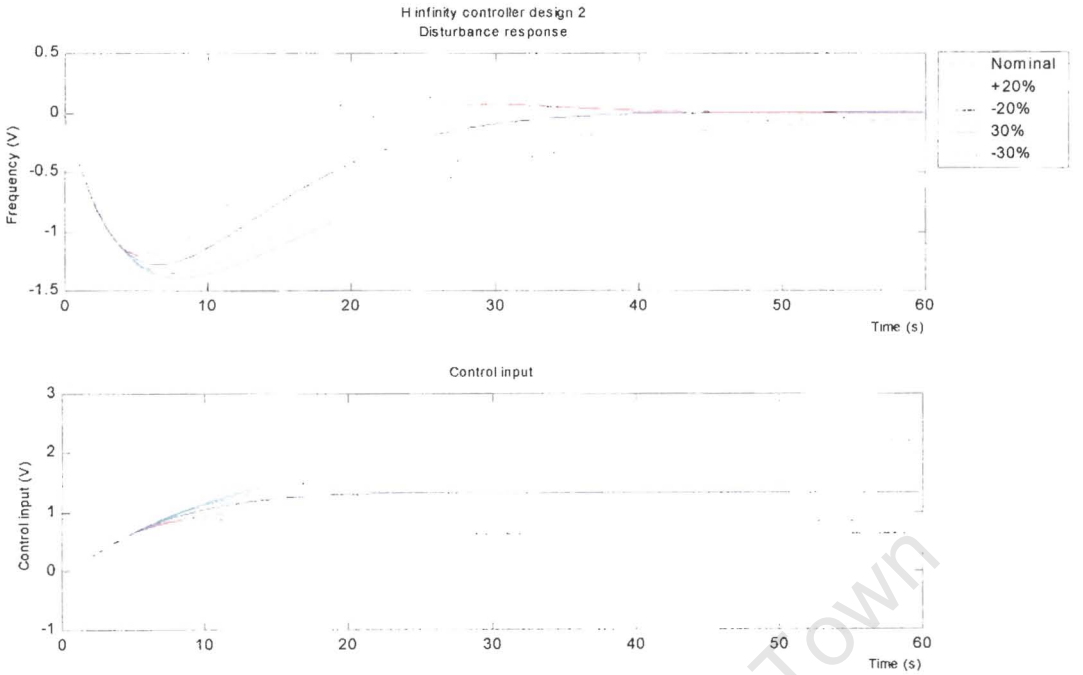


FIGURE 40: DISTURBANCE RESPONSE FOR H_∞ CONTROLLER DESIGN 2.

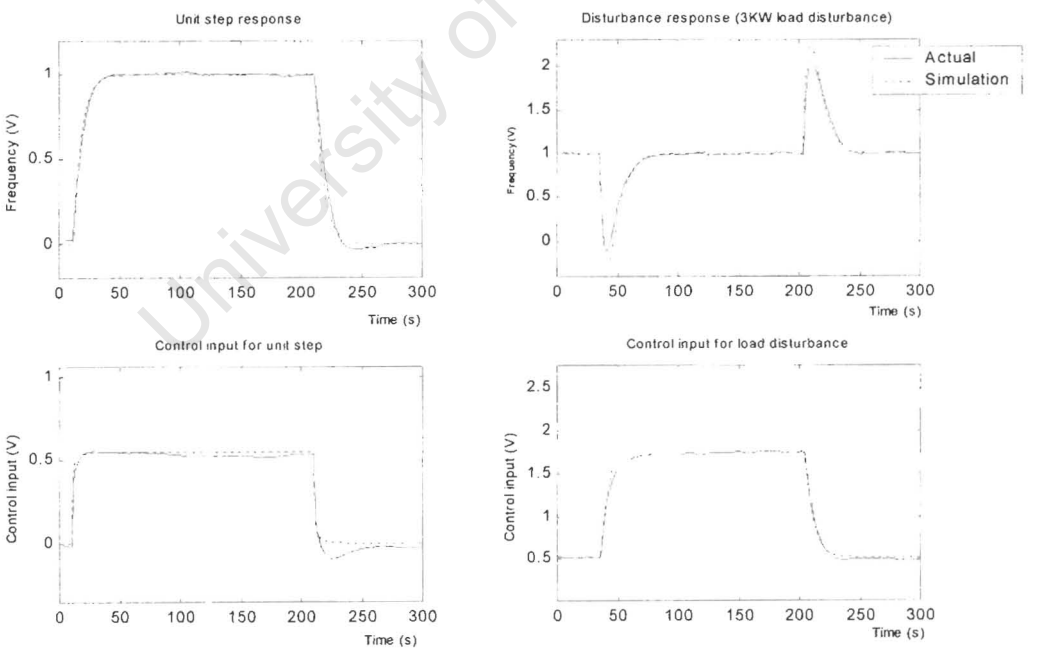


FIGURE 41: H_∞ DESIGN 2 RESPONSE.

Simulation results in comparison with the actual measured results are shown above in figure 41. The tracking performance and disturbance rejection properties are observed to be satisfactory. However, on the negative step response, the actual response is characterised by a momentary overshoot, this can be explained by the nonlinear behaviour of the motor, generator and load system. In addition the rotational inertia of the generator shaft contributes to this overshoot. The disturbance response is also shown above, illustrating satisfactory disturbance rejection properties of the controller. There is also a discrepancy between the actual and simulated results, which is attributed to modelling inaccuracies.

6.5. APPLICATION OF FUZZY LOGIC CONTROL

Described in this section is the application of fuzzy logic control to the experimental load frequency control system. The design of the fuzzy logic controller is a threefold process, consisting of fuzzification, fuzzy input processing by means of inferencing rules and defuzzification. The structure of the controller chosen is that of the standard PI fuzzy controller, which is shown in figure 42 below.

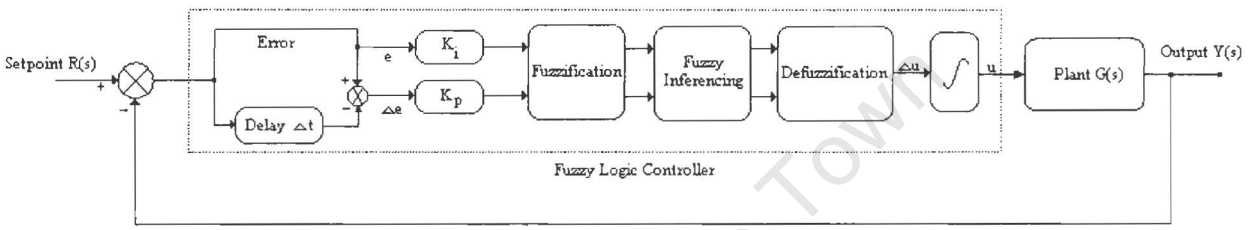


FIGURE 42: STRUCTURE OF FUZZY LOGIC CONTROLLER.

Where K_i and K_p are the integral and proportional gains respectively and Δt is the sample time delay. The controller is of PI type, which can be described mathematically as follows,

$$Fuzzy_{PI}(t) = u(t) = K_p e(t) + K_I \int e(t)dt$$

Where, e(t) is the input error signal.

However due to the difficulty of formulating the inferencing rules based on the integral of the error (Reznik, 1997), the above equation is differentiated to obtain the following, which is the change in control input.

$$\Delta Fuzzy_{PI}(t) = \Delta u(t) = K_P * \Delta e(t) + K_I * e(t)$$

Where the output of the above control strategy is integrated to revert back to the PI controller structure.

6.5.1. FUZZIFICATION

Fuzzification involves the conversion of crisp input to fuzzy inputs. This is achieved by the use of membership functions. A convenient choice of the membership functions, are the triangular shaped functions. These map the entire universe of discourse of the crisp input to a value between 0 and 1. The reason for fuzzification is that concise information can be described by relative terms (or in colloquial language), this enables a human interpretation of data. Shown in figure 43 below are the membership functions for the design.

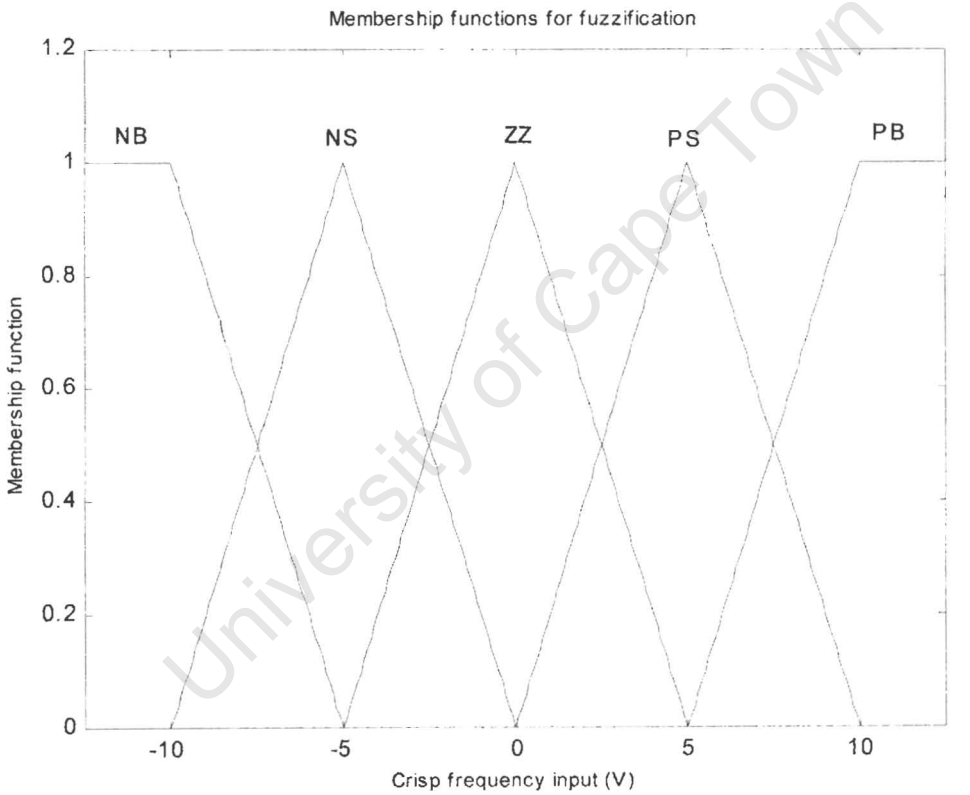


FIGURE 43: MEMBERSHIP FUNCTIONS FOR FUZZIFICATION.

To improve on the versatility of the fuzzy logic controller the inputs and outputs of the controller should be normalised, however the input range was considered to be ±10V.

6.5.2. FUZZY INFERENCING

Inferencing is the decision making utility of the controller and calculates the control effort required by considering the membership value and its respective linguistic term, such as negative big (NB) or positive small (PS). Shown below are the fuzzy inferencing table used in calculating the required control input.

Change in Error $\triangle e(t)$ / Error $e(t)$	NB	NS	ZZ	PS	PB
NB	NB	NB	NB	NS	ZZ
NS	NB	NB	NS	ZZ	PS
ZZ	NB	NS	ZZ	PS	PB
PS	NS	ZZ	PS	PB	PB
PB	ZZ	PS	PB	PB	PB

TABLE 4: FUZZY INFERENCING TABLE.

6.5.3. DEFUZZIFICATION

Defuzzification is the reverse of fuzzification and is a mapping procedure that converts the fuzzy control output to an equivalent crisp control output. The most well known defuzzification procedure is the centroid method. This method samples the universe of discourse of the control output at discrete points chosen by the designer and calculates the required control action depending upon the fuzzy outputs.

6.5.4. THE CONTROL SURFACE

In fuzzy control the shape of the control surface is of importance, since it conveys information on the performance quality of the resultant controller. It is thus desired that this surface be as smooth as possible without any sudden transition from one

extreme control value to another. The control surface for the above fuzzy controller is shown below, figure 44.

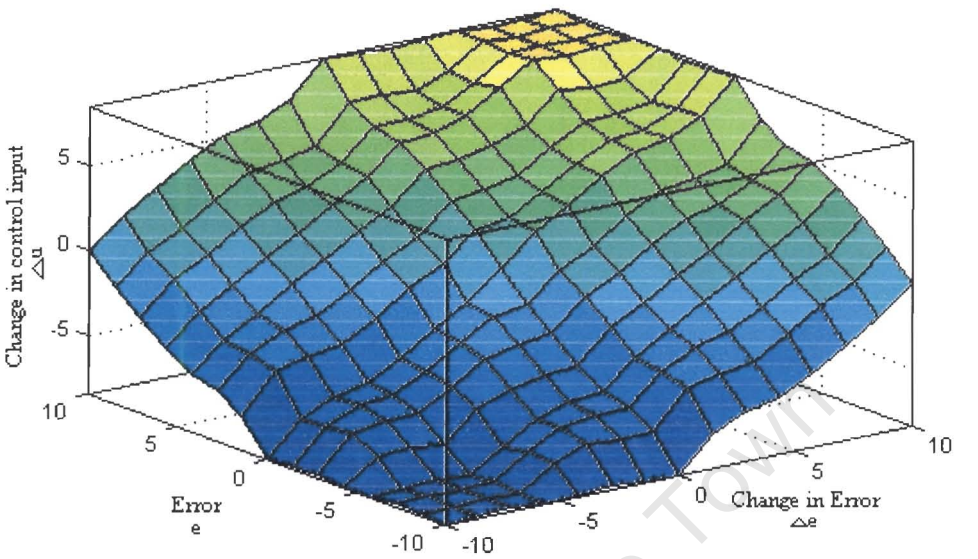


FIGURE 44: FUZZY CONTROL SURFACE.

6.5.5. FUZZY CONTROLLER PERFORMANCE RESULTS

Choosing parameters K_i and K_p by trial and error suitable values were obtained as follows (see appendix 3).

$$K_i = 0.035$$

$$K_p = 0.6$$

Simulation and experimental results are shown in figures 45,46, 47 and 48 below, showing the nominal response in comparison with ±20% and ±30% uncertainty in pole/zero positions.

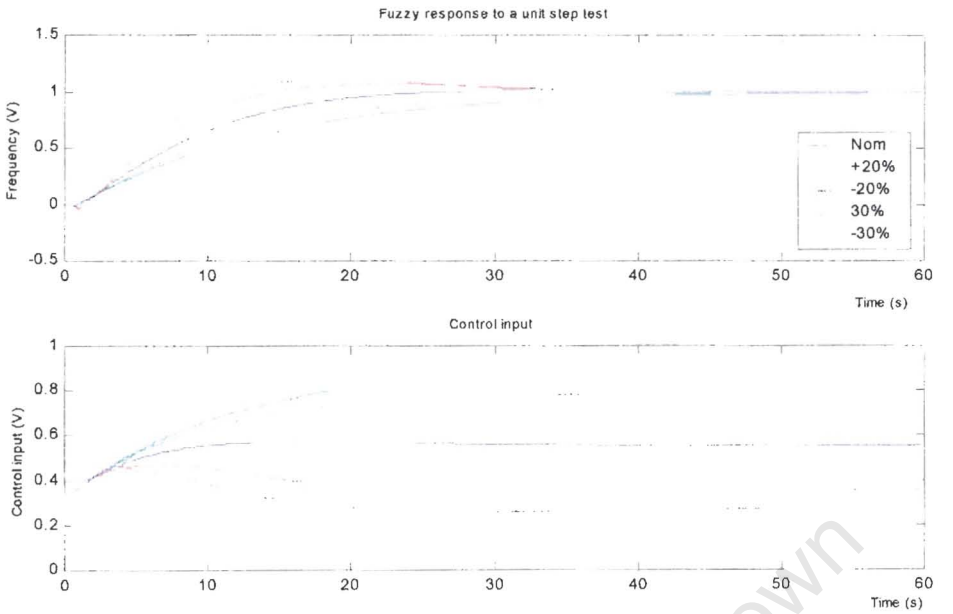


FIGURE 45: UNIT STEP RESPONSE OF FUZZY CONTROLLER.

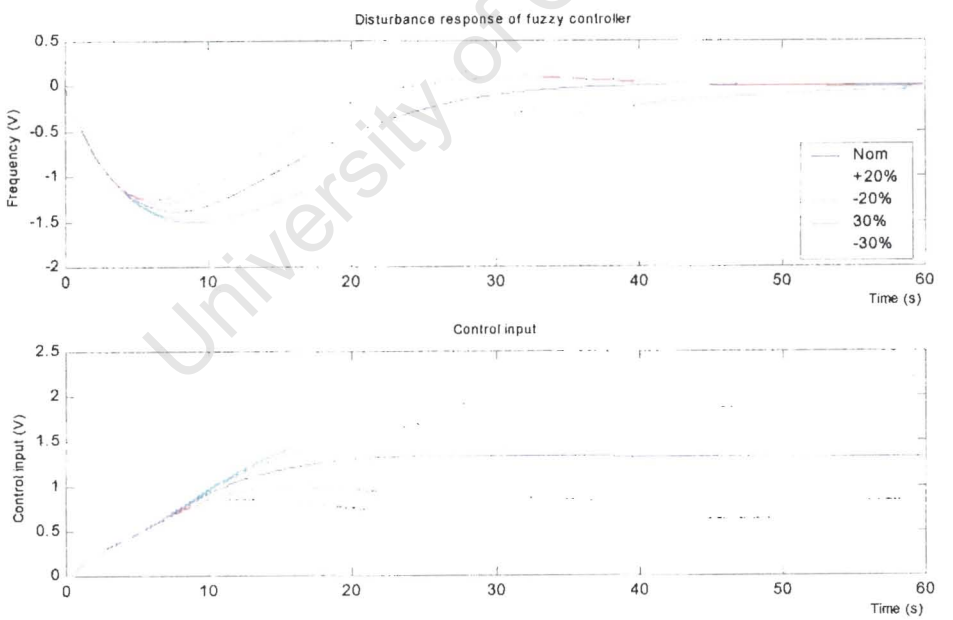


FIGURE 46: DISTURBANCE RESPONSE OF FUZZY CONTROLLER.

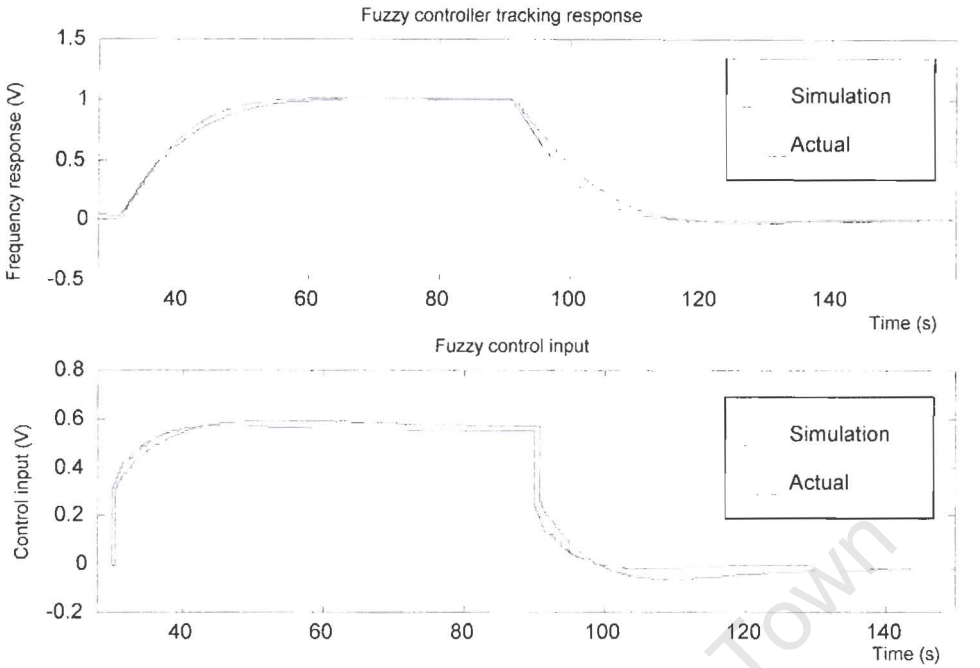


FIGURE 47: FUZZY CONTROLLER TRACKING RESPONSE (ACTUAL AND SIMULATION).

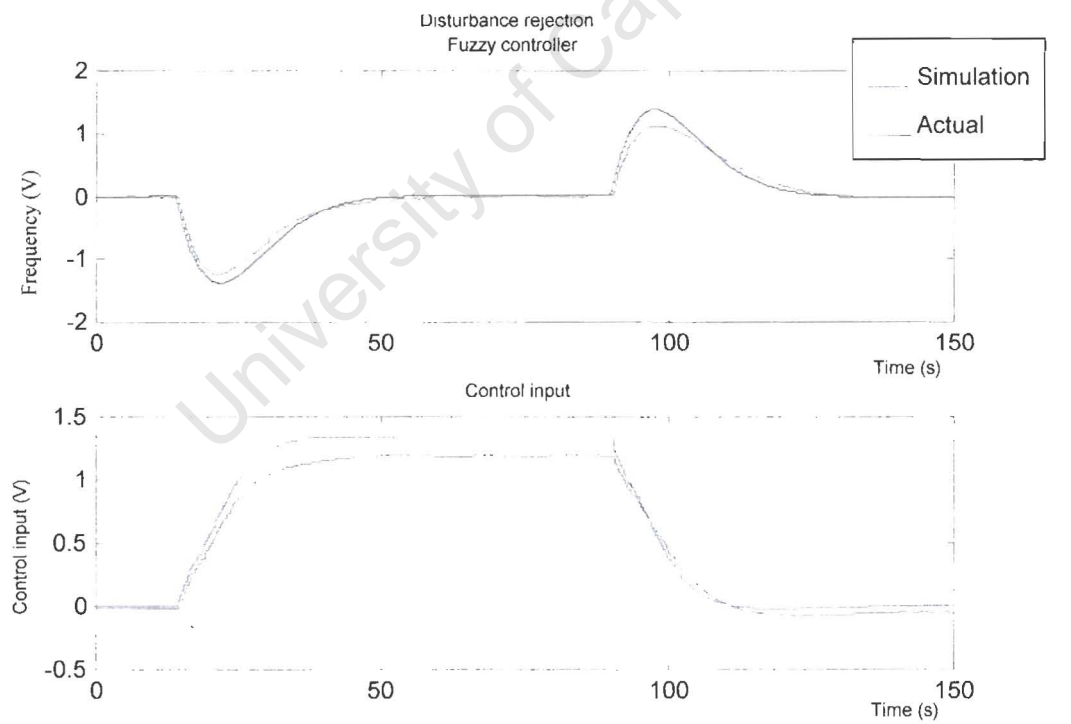


FIGURE 48: FUZZY CONTROLLER DISTURBANCE RESPONSE (ACTUAL AND SIMULATION).

6.6. APPLICATION OF QUANTITATIVE FEEDBACK THEORY

During the initial design stages of QFT, time domain specifications are formulated as frequency domain constraints on the magnitude and phase characteristics of the open loop transfer function L(s). Developed by Horowitz (Horowitz, 1993), QFT deals exclusively with model uncertainty, where these are represented as plant templates on the Nichols chart. Thus by considering the frequency domain constraints on L(s) and the corresponding plant templates, QFT bounds are calculated at discrete frequencies chosen from a given frequency set. This frequency set is chosen by the designer to reflect the most critical or dominant frequency modes of the system. A point of concern when choosing these frequencies are that QFT bounds are evaluated at these frequencies and by choosing them widely dispersed would fail to calculate the intermediate QFT bound. Described below is the application of QFT to the turbo–alternator system.

6.6.1. DESIGN SPECIFICATIONS

The design specifications of the closed loop control system are shown below.

- Robust Stability margin

$$\left|\frac{L(s)}{1+L(s)}\right| \leq 1.2 \qquad \text{for all } G(s),\ w \in [0, \infty]$$

Gain margin ≈ 5.26 dB.

Phase margin ≈ 60°.

- Tracking specifications

$$|A(s)| \leq |T(s)| \leq |B(s)|$$

where,

$$A(s) = \frac{0.005}{(s^2 + 0.2s + 0.01)(s + 0.5)} \quad \text{(lower tracking specification)}$$

$$B(s) = \frac{0.03*(s+3)}{s^2 + 0.6s + 0.09} \quad \text{(upper tracking specification)}$$

$$T(s) = \frac{L(s)}{1 + L(s)}$$

These specifications are selected to achieve robust stability and setpoint tracking. The upper and lower bounds are chosen such that there is no overshoot, and that the settling time of the system due to a unit step input is between 16s (upper bound) and 50s (lower bound), Robust stability is expressed as the conventional M–circle, where desired gain margins and phase margins are 5.26dB and 60° respectively. The second specification is that of tracking, with a lower and upper tracking bound A(s) and B(s) respectively. The frequency set is chosen as ω_{QFT} = [0.001 0.01 0.1 1 4 6 10 20 50 100 250] in radians per second. This frequency array was chosen such that it incorporates the most dominant system response characteristics. We notice the frequency at w = 6rad/s, this is included since most of the uncertainty is pronounced in this region.

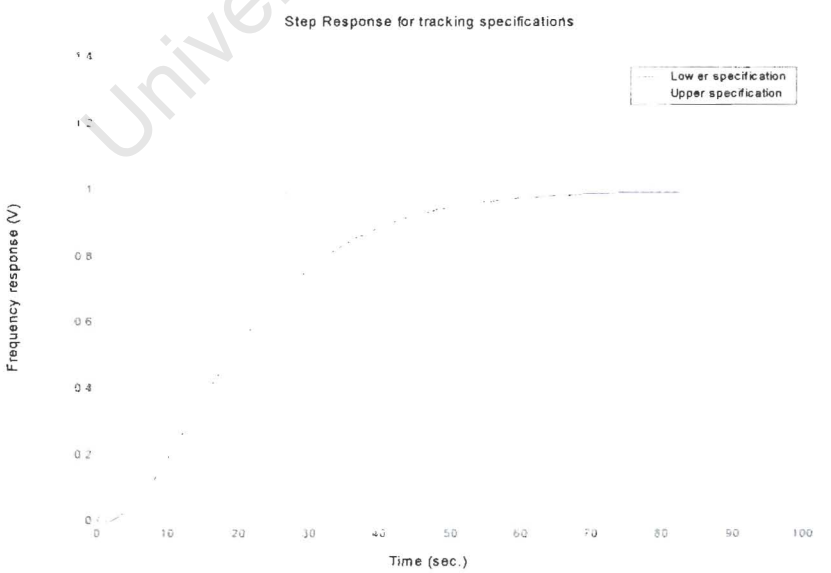


FIGURE 49: QFT TRACKING SPECIFICATIONS.

6.6.2. PLANT TEMPLATES

Plant templates of the model shown in section 6.1 above is illustrated below, figure 50. These represent the frequency response of the uncertain plant by considering ±20% parameter uncertainty in the plant model. As can be seen the plant templates tend towards a vertical line of length,

$$V = 20 * \log_{10}(K\ \max) - 20 * \log_{10}(K\ \min) \quad dB$$

Where K_{max} and K_{min} are the maximum and minimum plant gains. This is apparent from the fact that all transfer functions tends to,

$$L(s)_{s\to\infty} \to \frac{K}{s^e}$$

as the variable s tends to infinity. K is the plant gain and e is the excess of poles over zeros. Plant templates are shown below for the plant G(s).

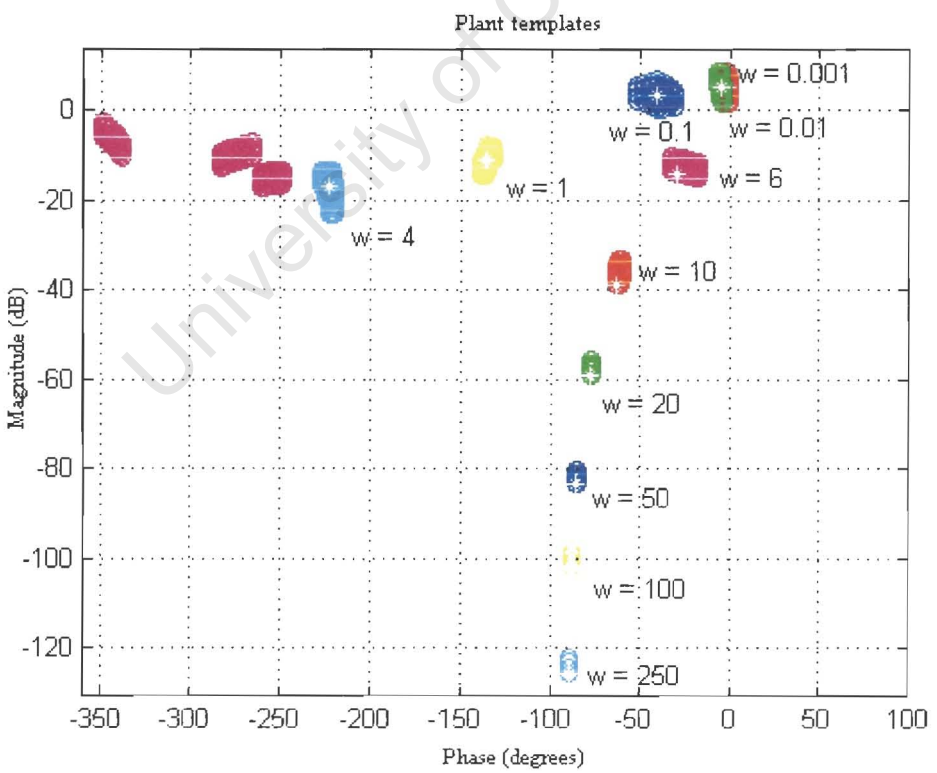


FIGURE 50: QFT PLANT TEMPLATES.

6.6.3. QFT BOUNDS

Shown in figure 51 and 52 below are the QFT bounds for the specifications shown above (namely, robust stability and QFT tracking bounds). These bounds were computed by using the function routines available from the Matlab QFT toolbox.

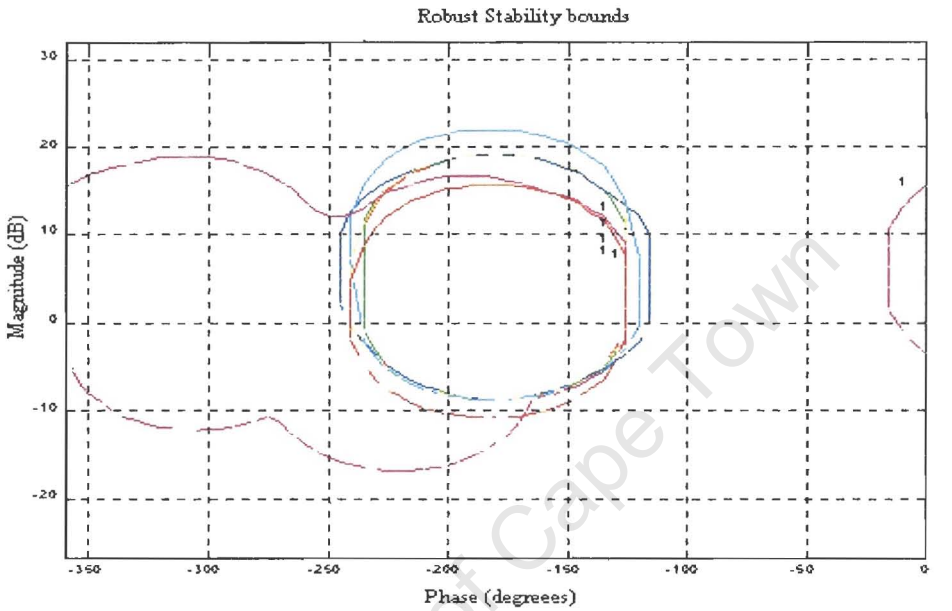


FIGURE 51: QFT TRACKING BOUNDS.

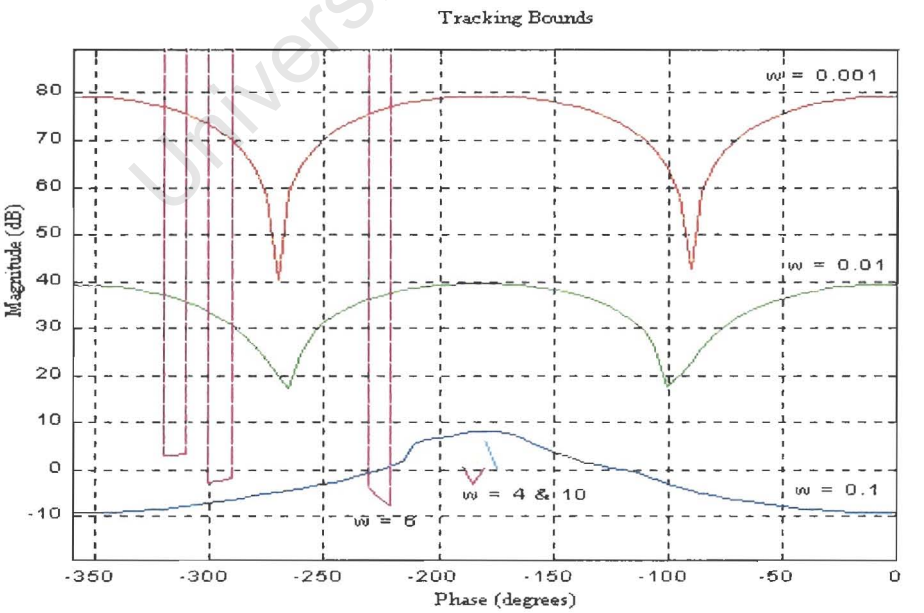


FIGURE 52: QFT TRACKING BOUNDS.

6.6.4. QFT LOOP SHAPING

QFT loop shaping is the process of shaping the open loop transfer function to meet the specified QFT bounds. This process follows a trial and error approach where dynamic compensator elements are added to the open loop transfer function until the bounds are satisfied. Though this procedure is transparent to the designer, it is tedious and time consuming resulting in a controller design of high complexity, however this is the procedure that will be followed. The loop shaping procedure followed in finding an appropriate controller K(s) is as follows. Starting with a unity gain controller K(s) = 1, we plot the resulting loop transmission L(s) on the Nichols chart with the QFT bounds. This is shown in the diagram below.

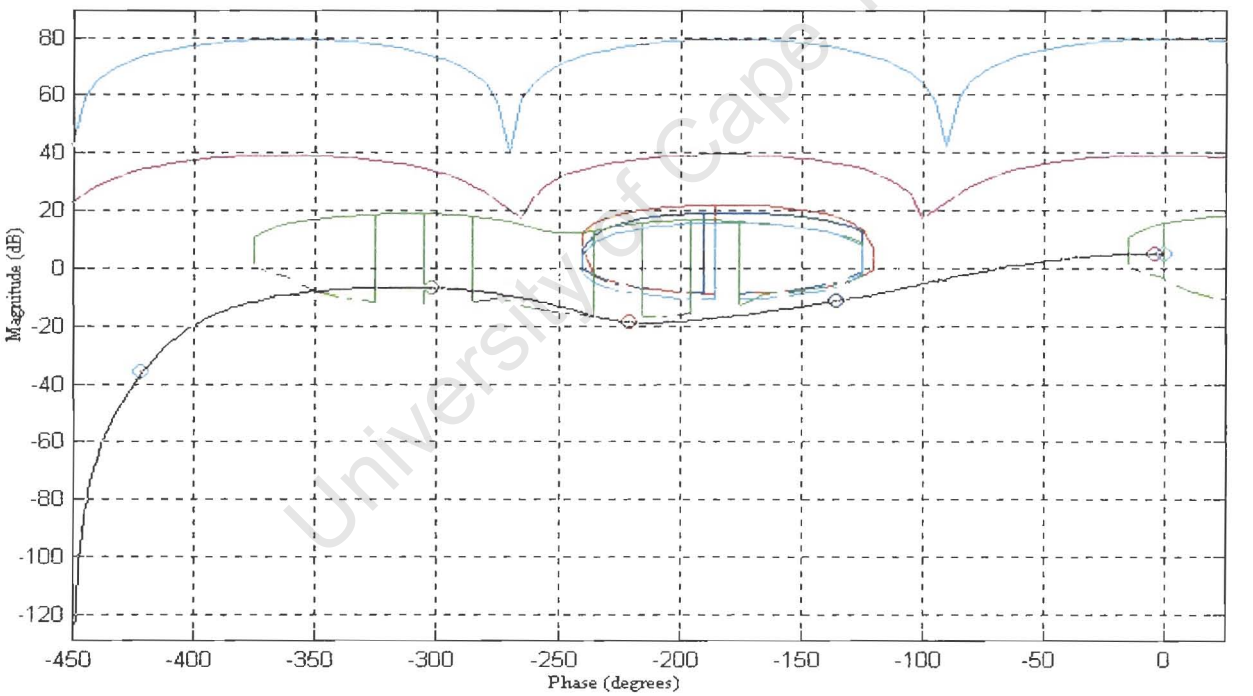


FIGURE 53: NICHOLS CHART OF L(S) FOR K(S) = 1.

As can be seen from figure 53, to meet the bounds at frequencies w = [0.001 0.01 0.1], lag of approximately -90° should be added to the system and the gain should be increased by 40dB. This can be achieved by adjusting the loop gain accordingly and inserting an integrator into the system. This also sets the corresponding type number

for zero steady state error. The Nichols chart of the open loop transfer function L(s) and its satisfied bounds are shown in figure 54 below. A point of interest is that we must force the gain of the open loop at high frequencies to be attenuated as quickly as possible, this is necessary to minimize excessive control action at high frequencies (eg. due to a step input). Though the Nichols chart contour satisfies all its specified bounds, it is by no means an optimal trajectory. The optimal design should follow the universal high frequency bound until approximately –180°, where it should attenuate rapidly.

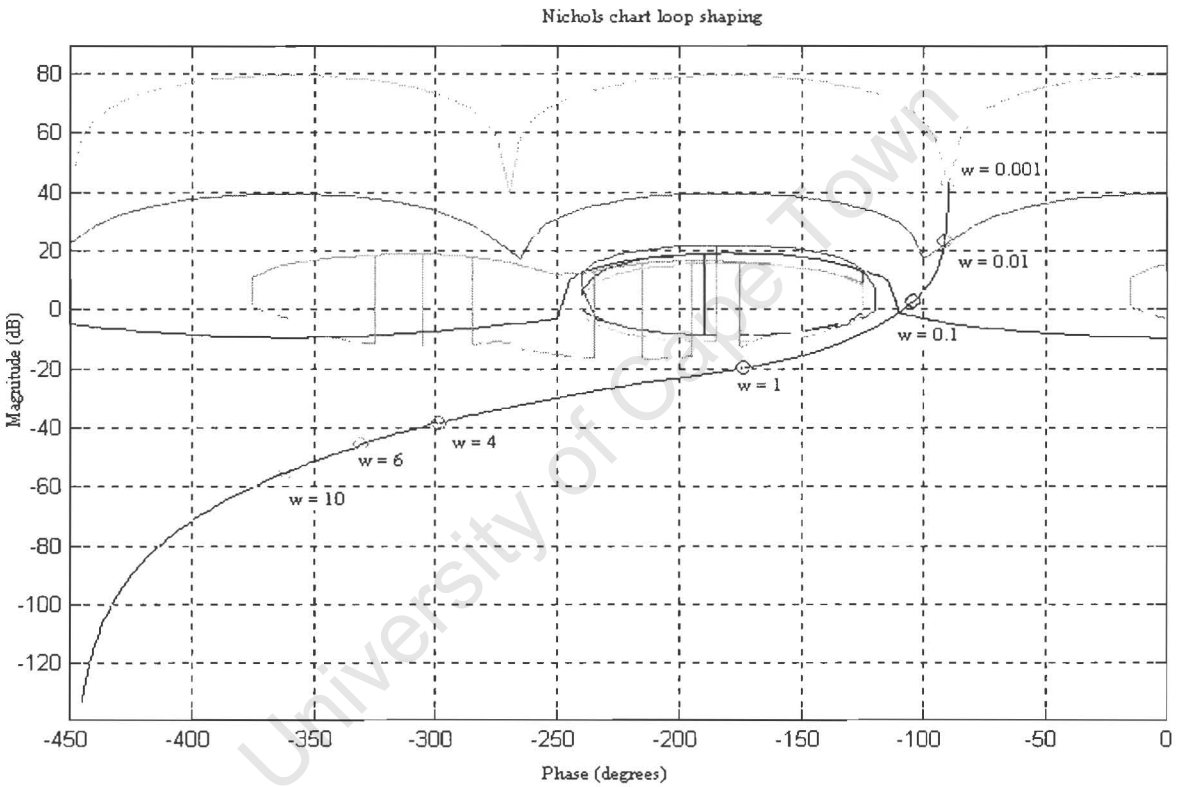


FIGURE 54: NICHOLS CHART OF OPEN LOOP TRANSMISSION.

The resultant controller obtained after a few iterations is given below.

$$K_{QFT}(s) = \frac{0.3226\,s^5 + 0.5963\,s^4 + 12.5\,s^3 + 15.67\,s^2 + 8.156\,s + 0.9874}{s^5 + 17.83\,s^4 + 45.07\,s^3 + 40.63\,s^2 + 12.39\,s}$$

A table tabulating the poles and zeros of this controller is shown below.

	Plant G(s)	Controller $K_\infty(s)$
Poles	-0.27619 + 5.6709i -0.27619 - 5.6709i -2 -0.17161	0 -15.003 -1.0079 -1 -0.81937
Zeros	2	-0.28012 + 6.1051i -0.28012 - 6.1051i -0.56001 + 0.41679i -0.56001 - 0.41679i -0.16816

TABLE 5: POLES AND ZEROS OF QFT DESIGN 1.

It is a fifth order controller with an integrator to set the type number. The motivation for the other poles and zeros are to keep the transfer function proper and realizable, this is due to the fact that by adding the integrator, lag is introduced into the system. Therefore to compensate for this lag, zeros are added to satisfy the bounds at each frequency. It tends to be easier to satisfy the bounds at lower frequencies and then move towards higher frequencies. Since the design of the controller takes place on the Nichols chart, the stability of the control system is guaranteed by ensuring that the open loop gain and phase characteristics does not pass to the left of the Nichols chart origin (-180°, 0dB). From table 5 it can be seen that though there is no exact pole zero cancellation, the zeros of the controller, s = -0.28012 ±6.1051i and s = -0.16816 are in the vicinity of the plant poles s = -0.27619 ± 5.6709i and s = -0.17161. For all practical purpose these poles and zeros cancel. An alternative design would be to use a PI type control law with a lead/lad if required.

6.6.5. PREFILTER DESIGN

As described in chapter 5, following the design of the feedback controller is the design of an appropriate prefilter. The prefilter shifts the closed loop frequency response model to within a region satisfying the required tracking specifications. Shown in figure 55 below is a bode plot of the nominal closed loop transfer function T(s) without a prefilter (ie. F(s) = 1) for ±20% plant uncertainty.

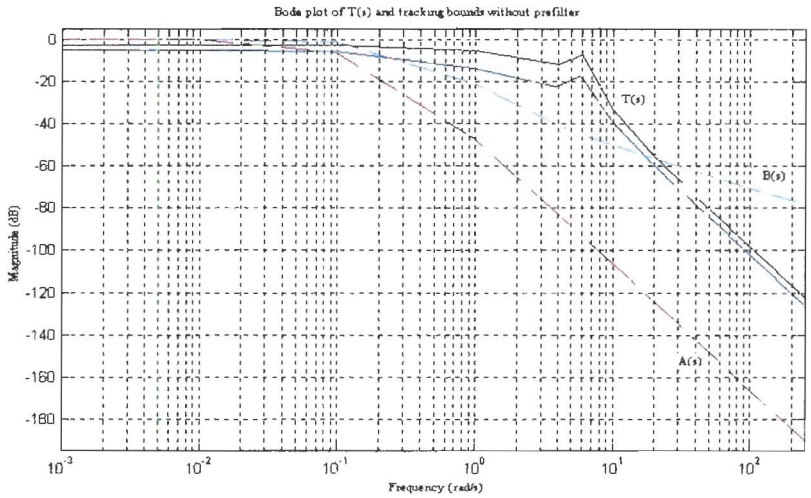


FIGURE 55: BODE OF T(S) WITHOUT PREFILTER P(S).

The prefilter F(s) was thus chosen as the following second order model.

$$F_{QFT}(s) = \frac{0.07555\,s + 0.2083}{s^2 + 1.649\,s + 0.2083}$$

	T(s)	$F_{QTF}(s)$
Poles	-14.981	-1.5112
	-0.28575 + 6.0962i	-0.13784
	-0.28575 - 6.0962i	
	-2.4898	
	-0.89303 + 0.56852i	
	-0.89303 - 0.56852i	
	-0.40002	
	-0.16057 + 0.077385i	
	-0.16057 - 0.077385i	
Zeros	-0.28012 + 6.1051i	-2.7571
	-0.28012 - 6.1051i	
	2	
	-0.56001 + 0.41679i	
	-0.56001 - 0.41679i	
	-0.16816	

TABLE 6: POLES AND ZEROS OF T(S) AND $F_{QFT}(s)$.

Table 6 above compares the closed loop poles of the complementary sensitivity function T(s) with that the prefilter F(s). With reference to the time domain response

shown in figure 57, the sluggish response due to a unit step input can be explained by the dominant pole at s = -0.13784 of the prefilter. The two poles of F(s) gives the required damping by moving T(s) between the upper and lower tracking bounds, while the zero positioned at s = -2.76 increases the gain at high frequencies. The resultant closed loop system F(s)*T(s) is shown below.

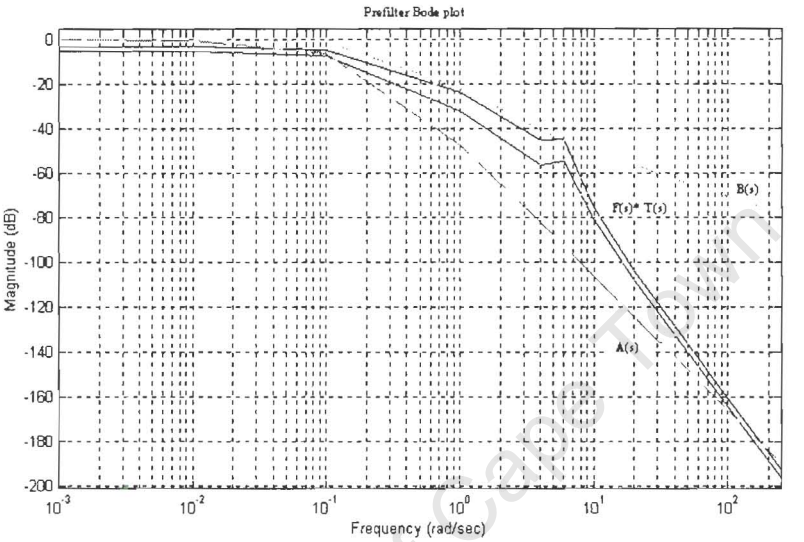


FIGURE 56: PREFILTER BODE PLOT.

The droop as observed in figure 56 around 0.1rad/s accounts for the slow tail response in figure 57. There are deviations at both low and high frequencies as seen above in figure 56, however this is acceptable since for tracking at low frequencies the gain of the prefilter should be 1, while that of high frequencies is of less consequence because it is not expected that tracking should realised at these frequencies. Time domain simulations in comparison with experimentally observed results are shown in figures 57, 58, 59 and 60 below. As expected there are bound violations at low frequency, this can be improved by choosing an appropriate prefilter that is more stringent over the low frequency range, however this may require a high order filter.

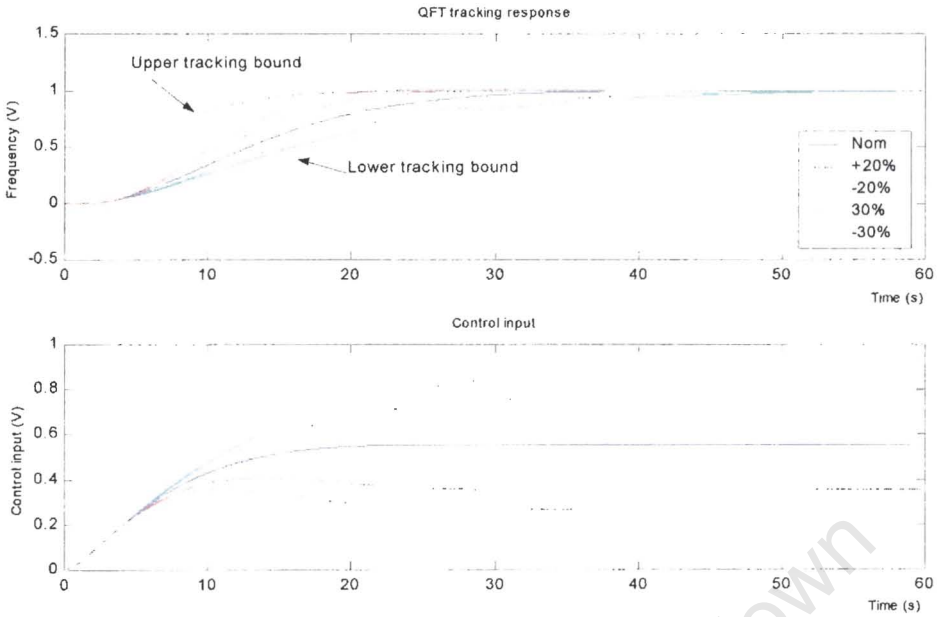


FIGURE 57: QFT TRACKING RESPONSE FOR DESIGN 1.

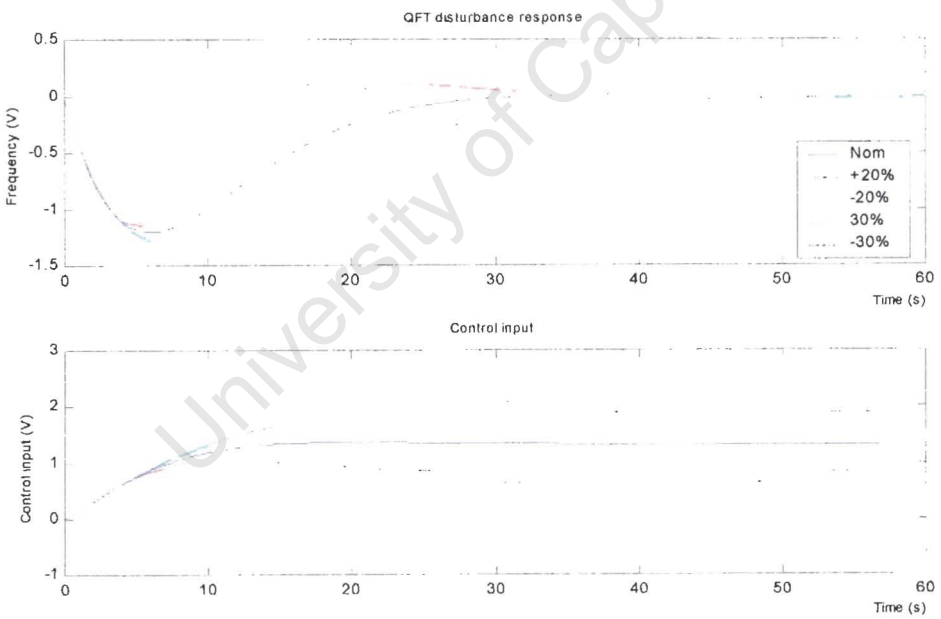


FIGURE 58: QFT DISTURBANCE RESPONSE FOR DESIGN 1.

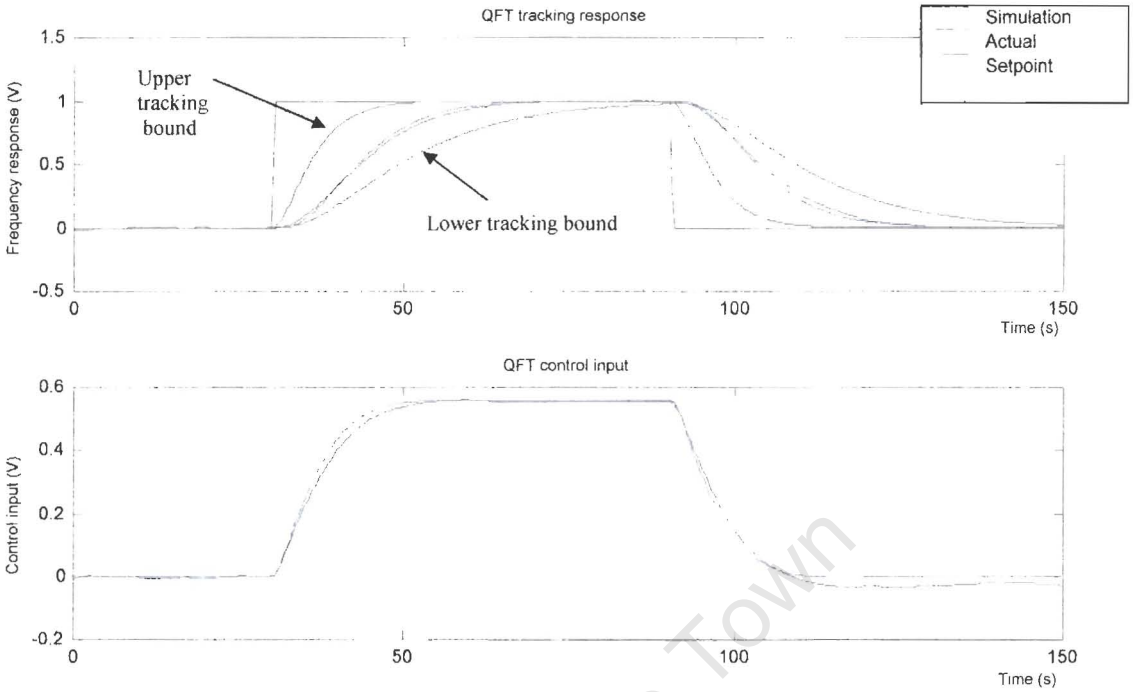


FIGURE 59: QFT TRACKING RESPONSE (SIMULATION AND ACTUAL).

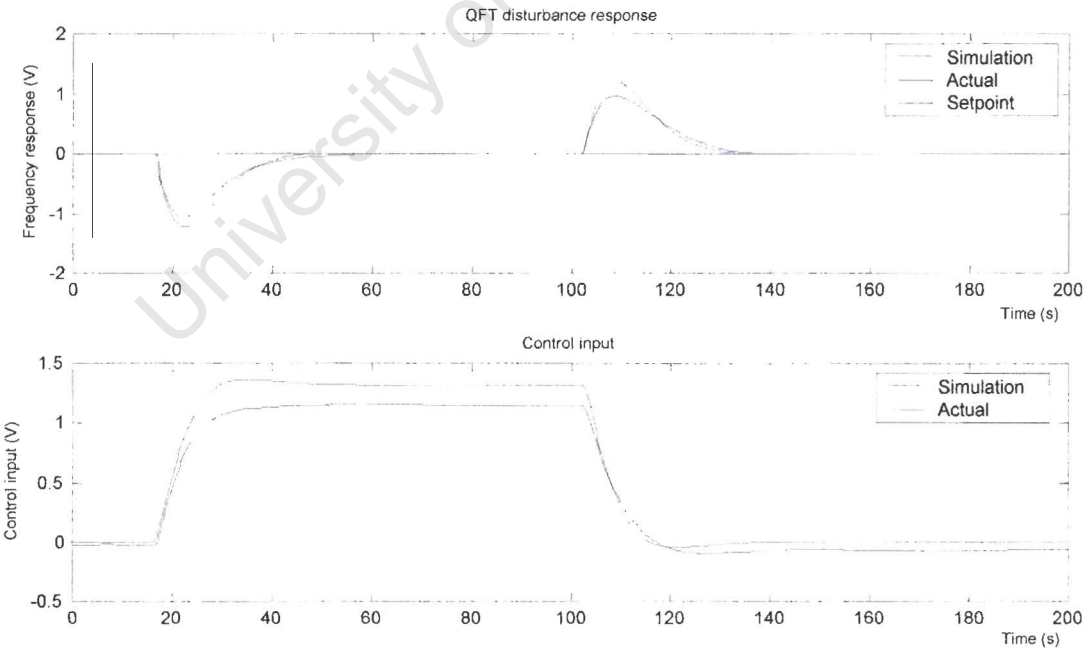


FIGURE 60: QFT DISTURBANCE RESPONSE (SIMULATION AND ACTUAL).

6.6.6. QFT DESIGN FOR DISTURBANCE REJECTION

This section illustrates the QFT design technique based on disturbance rejection specifications. The disturbance rejection specification is chosen as the worst case output disturbance response, being 1 – A(s), where A(s) is the lower tracking bound or a suitably chosen specification defined by tracking specifications (Horowitz, 1993). A plot of the tracking specifications are shown in figure 61 below, it also shows the worst case disturbance rejection response due to a unit step input and their corresponding tracking specifications. The specifications are chosen as follows.

- Robust Stability margin

$$\left|\frac{L(s)}{1+L(s)}\right| \leq 1.2 \qquad \text{for all } G(s), w \in [0, \infty]$$

Gain margin ≈ 5.26 dB.

Phase margin ≈ 60°.

- Tracking specifications

$$|A(s)| \leq |T(s)| \leq |B(s)|$$

where,

$$A(s) = \frac{0.0050125}{(s^2 + 0.2s + 0.01003)(s + 0.5)} \quad \text{(lower tracking specification)}$$

$$B(s) = \frac{0.03033 * (s + 3)}{s^2 + 0.6s + 0.0901} \quad \text{(upper tracking specification)}$$

$$T(s) = \frac{L(s)}{1 + L(s)}$$

- Disturbance rejection specification (worst case disturbance).

$$T_d(s) = \frac{0.012019}{(s^2 + 0.4s + 0.04006)(s + 0.3)}$$

$$|S(s)| \leq |1 - T_d(s)|$$

$$S(s) = \frac{1}{1 + L(s)}$$

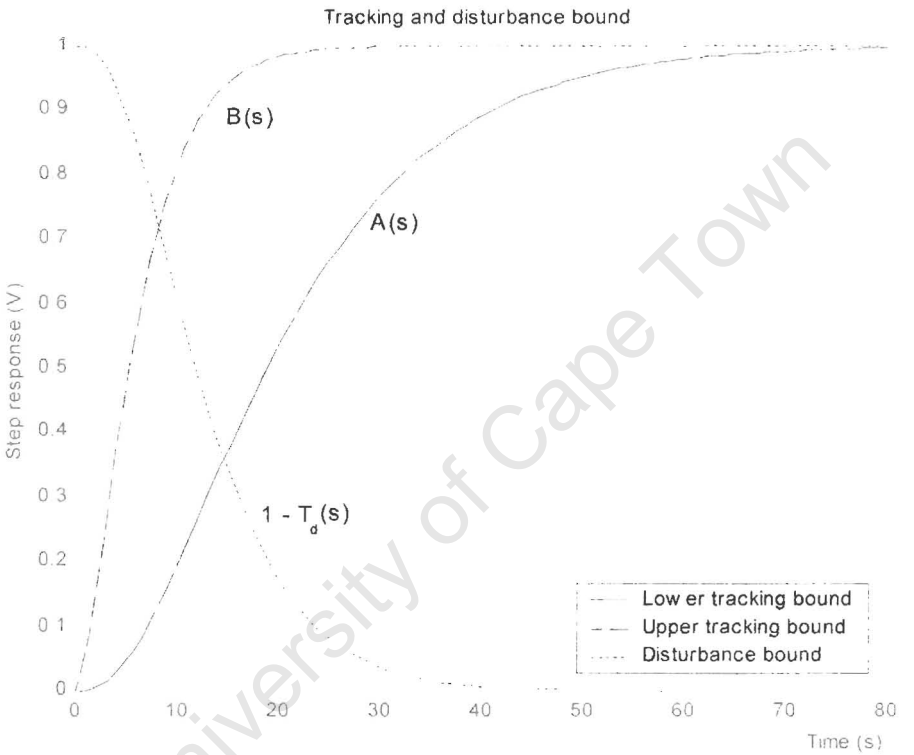


FIGURE 61: TRACKING SPECIFICATION AND DISTURBANCE REJECTION SPECIFICATION.

Computing the QFT robust stability, tracking and disturbance rejection bounds yields similar Nichols charts as those shown above (figures 51 and 52) and are not repeated here. The final QFT design is thus shown below in figure 53, it shows the satisfied QFT bounds for a sub optimal QFT controller. The loop shaping process follows a similar trial and error approach of that described above (sections 5.4 and 6.6.4), by first satisfying the low frequency bounds and progressively moving towards satisfying the higher frequency bounds. Figure 62 illustrates the final QFT design. The resultant controller is,

$$K_{QFT}(s) = \frac{0.003807\,s^5 + 0.04805\,s^4 + 0.221\,s^3 + 1.739\,s^2 + 2.033\,s + 0.258}{s^5 + 7.912\,s^4 + 13.21\,s^3 + 10.2\,s^2 + 3.609\,s}$$

	Plant G(s)	Controller $K_\infty(s)$
Poles	-0.27619 + 5.6709i	0
	-0.27619 - 5.6709i	-5.9679
	-2	-0.9063
	-0.17161	-0.5189 + 0.63087i
		-0.5189 - 0.63087i
Zeros	2	-10.749
		-0.27444 + 6.0791i
		-0.27444 - 6.0791i
		-1.1787
		-0.14443

TABLE 7: POLES AND ZEROS FOR QFT DESIGN 2.

Seen from table 7 there are approximate pole zero cancellation between controller zeros (s = -0.27444 ± 6.0791i, s = -0.14443) and plant poles (s = -0.27619 ± 5.6709i, s = -0.17161). From the Nichols chart below at w = 0.1 and w = 1, there is a violation of the bounds. This is not severe

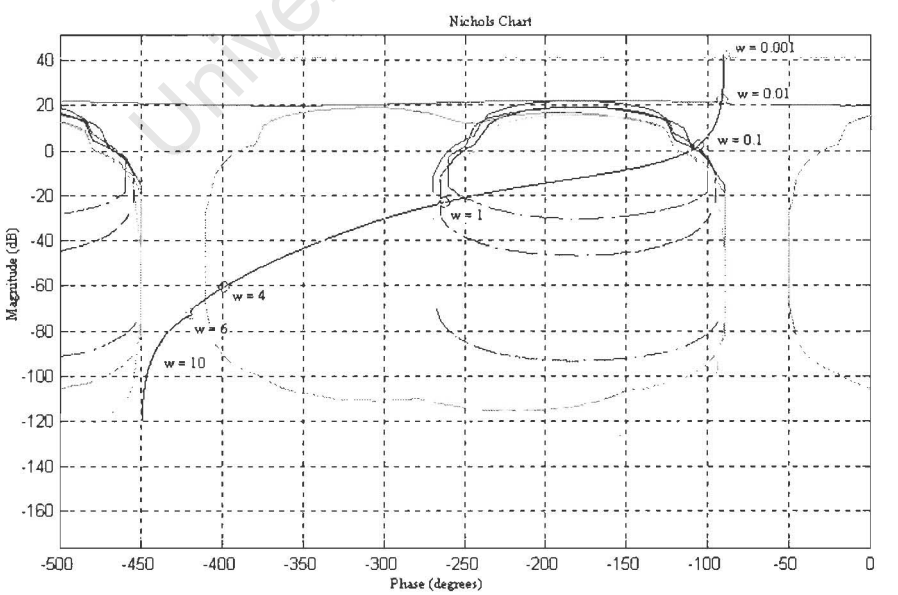


FIGURE 62: QFT DESIGN BASED ON DISTURBANCE REJECTION.

The prefilter chosen for the design is shown below, it moves the closed transfer function T(s) to within the region satisfying the tracking specifications. Its bode plot is shown below in figure 63.

$$F_{QFT}(s) = \frac{0.06426s + 0.1265}{s^2 + 1.165s + 0.1265}$$

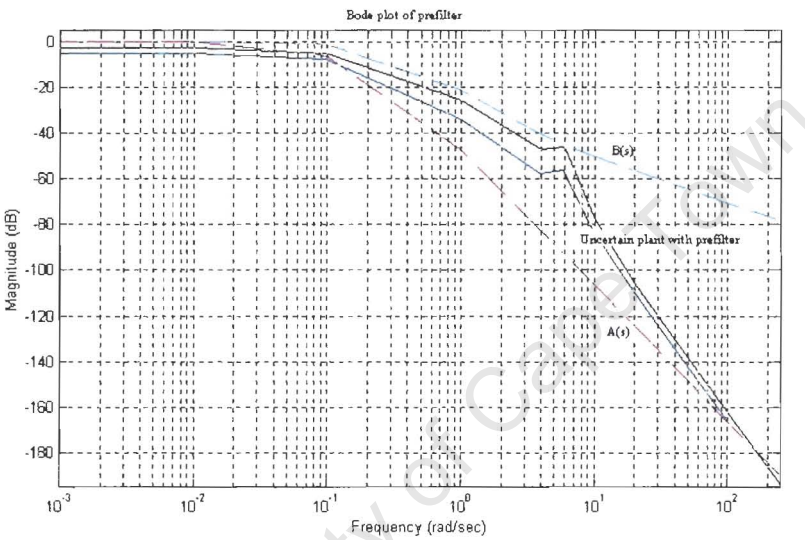


FIGURE 63: QFT PREFILTER DESIGN.

6.6.7. SIMULATION AND EXPERIMENTAL RESULTS FOR THE DISTURBANCE DESIGN

Figures 64 and 65 shows the tracking and disturbance rejection response of the above designed QFT controller. The simulation and experimentally observed results correspond reasonably well. However due the effect of the prefilter over the high frequency range (the region just after the unit step in figure 64), the tracking response is sluggish. This can be remedied by increasing the prefilter gain over this frequency.

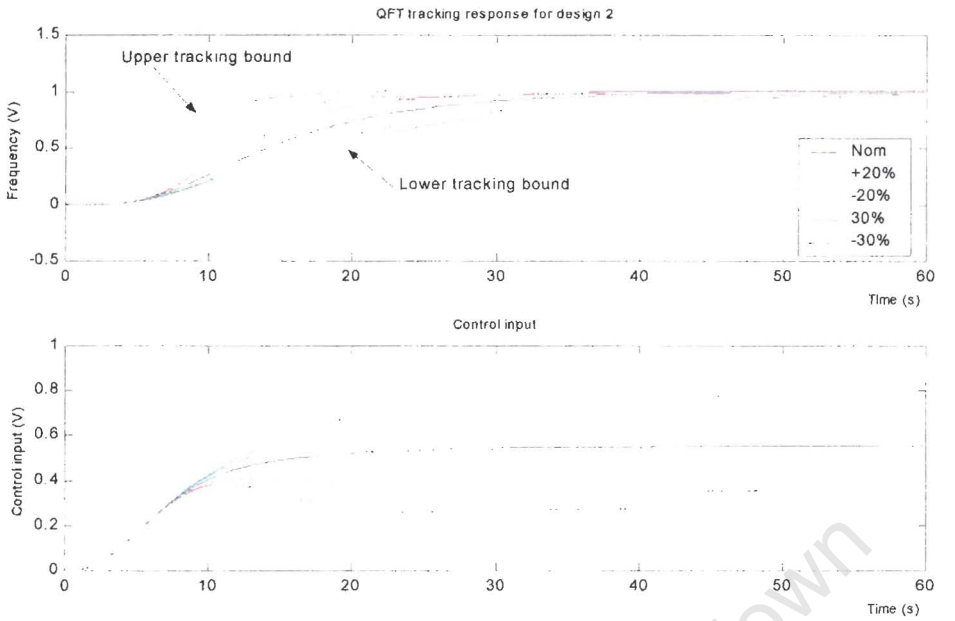


FIGURE 64: TRACKING RESPONSE FOR QFT DESIGN 2

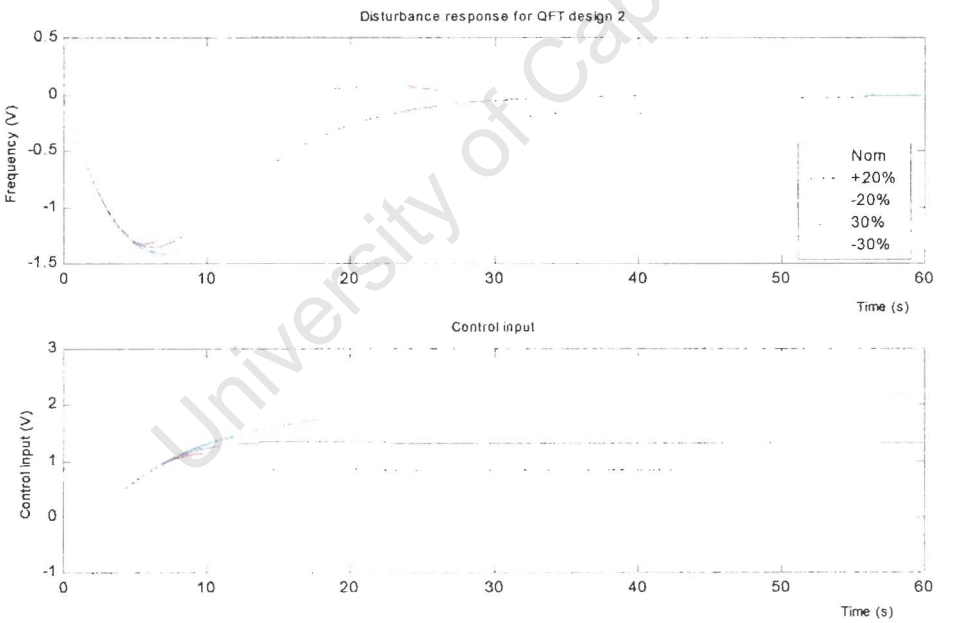


FIGURE 65: DISTURBANCE RESPONSE FOR QFT DESIGN 2.

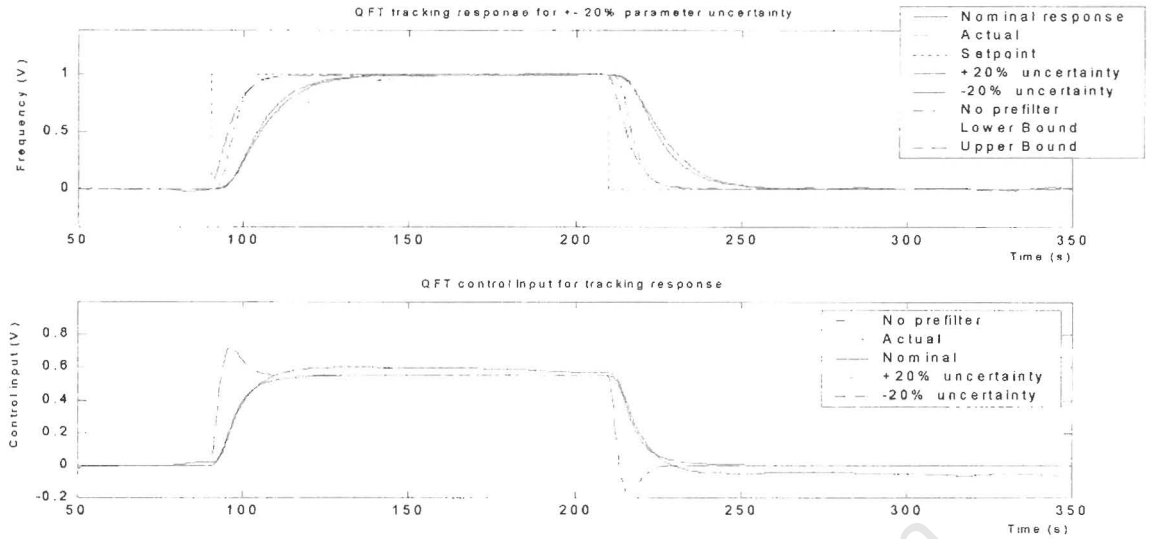


FIGURE 66: QFT TRACKING RESPONSE.

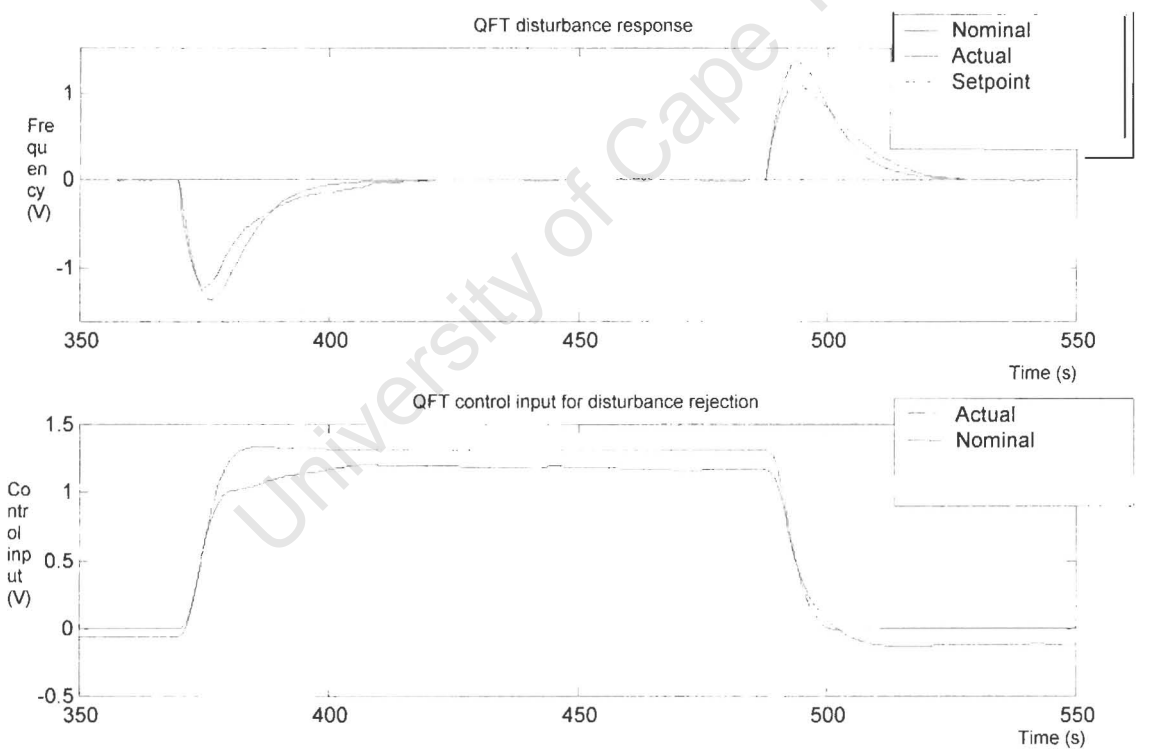


FIGURE 67: QFT DISTURBANCE RESPONSE.

6.7. PI CONTROLLER DESIGN

Though modern control theory results in excellent control system performance, the consideration of classical controller design techniques such as PI, should not be neglected. This is based on the simplicity of the controller design. For this reason a PI controller is designed for the process shown in figure 23. The tracking specifications for the PI control system are chosen in sync with that of the QFT controller design described by the previous section. The structure of the PI controller is shown below, where the design objective is to find parameters K and T such that the closed loop system is stable and satisfies the tracking specifications.

$$K_{PI}(s) = \frac{K * (Ts + 1)}{s}$$

The following controller parameters obtained via simulation satisfies the above specifications.

$$K = 0.5 \qquad T = 0.15$$

Time domain simulations and the actually observed results are shown below (figures 68, 69, 70 and 71). Also shown are the robustness properties of the controller for ±20% parameter uncertainty in all G(s) parameters.

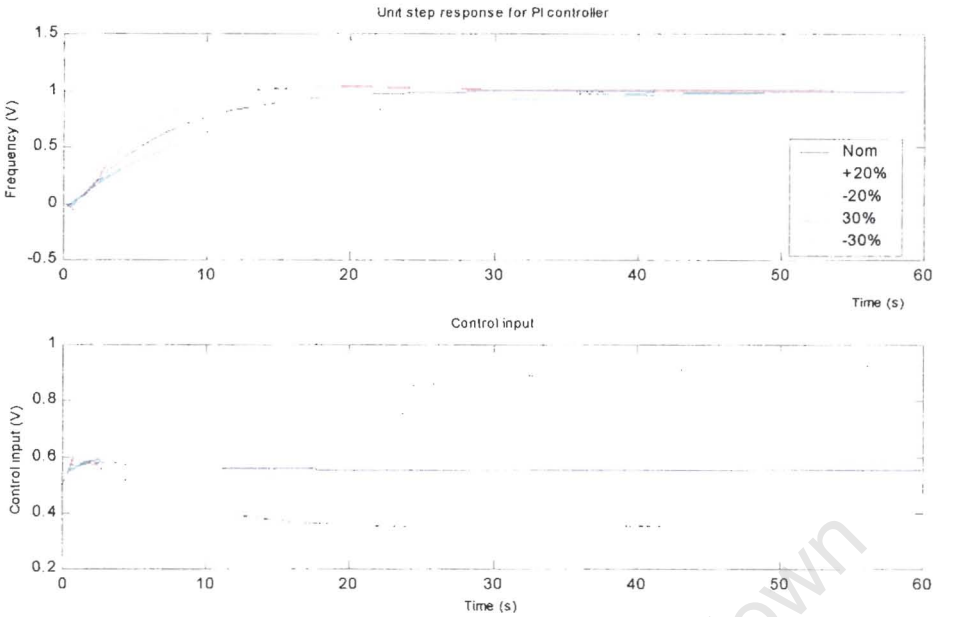


FIGURE 68: UNIT STEP RESPONSE FOR PI CONTROLLER.

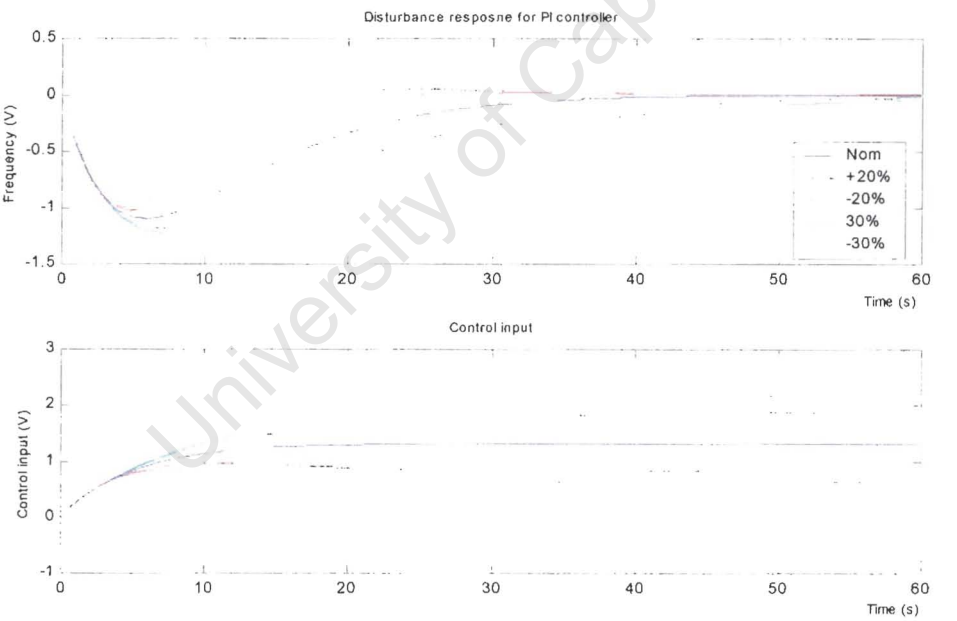


FIGURE 69: DISTURBANCE RESPONSE FOR PI CONTROLLER.

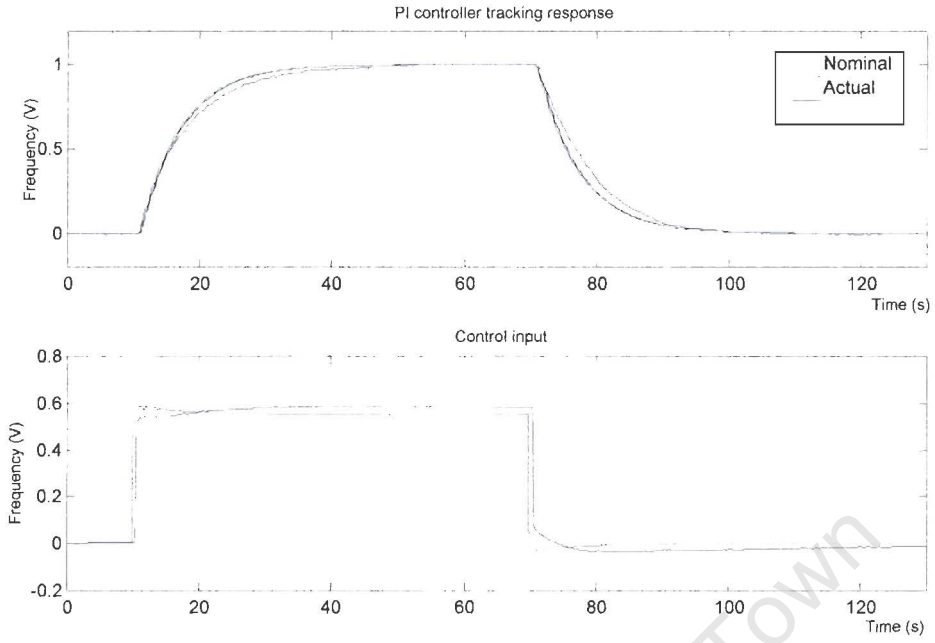


FIGURE 70: PI CONTROLLER TRANSIENT RESPONSE.

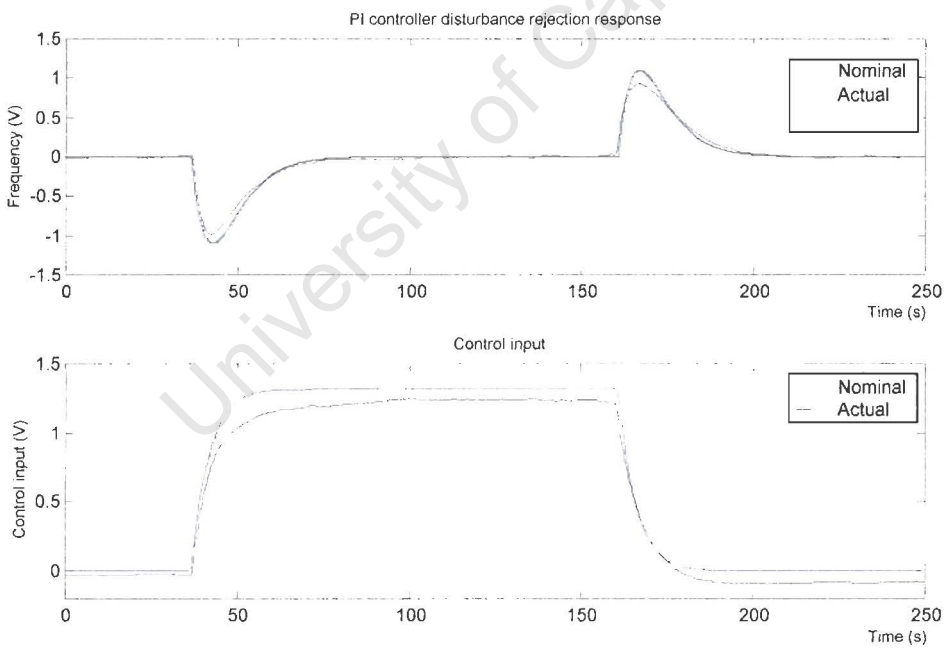


FIGURE 71: PI DISTURBANCE REJECTION RESPONSE.

6.8. CONTROLLER DESIGN COMPARISON

A term prevalent in the chemical engineering industry is that of dynamic operability, in which the control quality of the closed loop system is accessed (Morari, 1983). The quality of control is thus a function of the controller design variables, the structure of the controller, the manipulated inputs chosen for control and that of the plant characteristics (Ross, 1997). The effectiveness of the resultant control system is then tested in regard to robustness or sensitivity of the system to uncertainty and whether the system performance is maintained over the varying operating region. The following section attempts to expound on the quality of control of the above designed controllers, it is by no means an operability study but aims at commenting on the effectiveness of these controllers (H_∞, QFT, Fuzzy, PI) to the power system load frequency control problem.

In evaluating the quality of control, a performance measure generally is defined (Kirk, 1970). The choice of a performance index should be done prior to the design of the controller, however the performance index has been chosen towards the end of the design. Thus to a certain extent this comparison is biased. The selection of the performance measure depends on its ability to accurately represent the desired performance of the closed loop system. In our design problem we want to minimise the frequency deviation and that of the control input utilised. Therefore the following performance index is chosen. Where k_1 and k_2 are constants signifying the relative importance of the error deviation and that of the control input deviation respectively.

$$J = \frac{1}{T}\int_0^T k_1 * e(t)^2 + k_2 * u(t)^2 dt$$

Where T is the time period over which the performance measure is applied. Shown in figures 72 and 73 below are time domain simulation results of the above described controllers for a unit step input (figure 72) followed by a 3KW load disturbance response (figure 73). As seen from these figures similar response characteristics can

be achieved for all the designed controllers. However, the PI controller uses substantially more control input than that utilised by the other controller structures (H_∞, Fuzzy and QFT), in that it initial almost instantaneously increases its control energy. This may have negative effects on the actuators of the control system (eg. the turbine governors) and is not desirable.

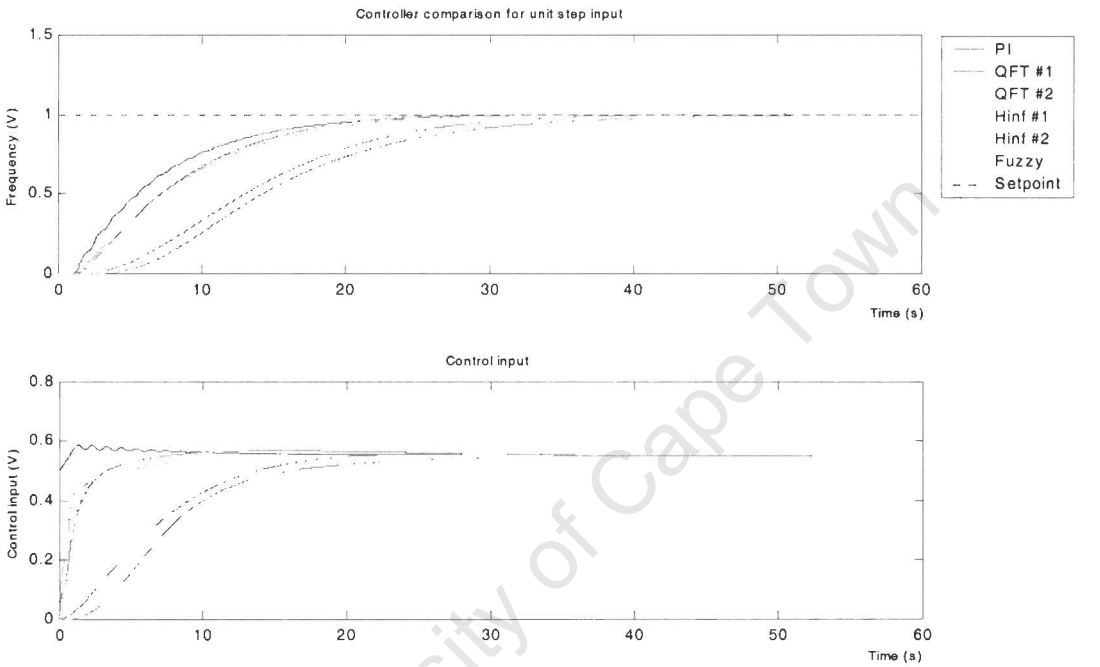


FIGURE 72: CONTROLLER COMPARISON PLOT FOR A UNIT STEP INPUT.

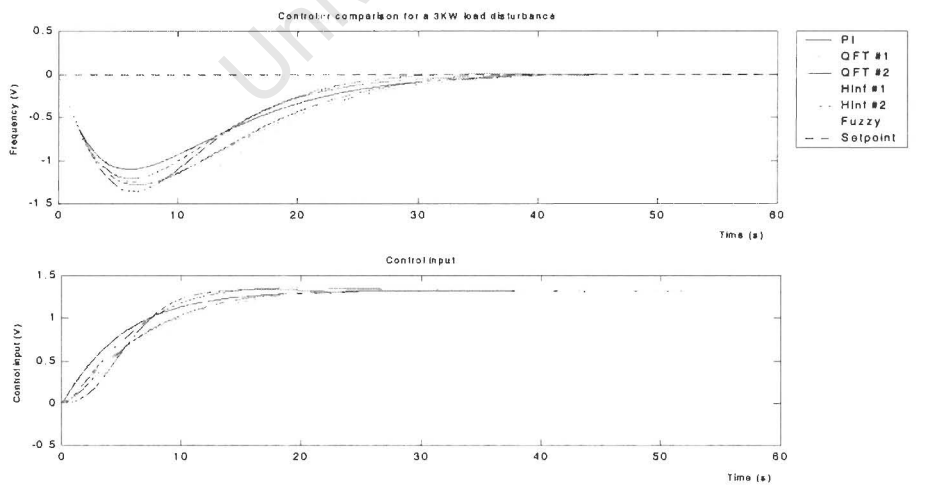


FIGURE 73: CONTROLLER COMPARISON FOR LOAD DISTURBANCE.

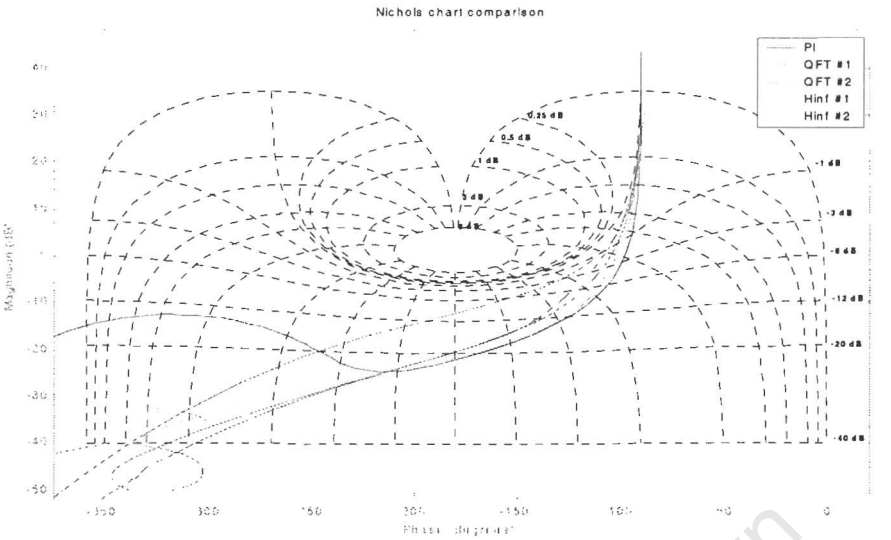


FIGURE 74: NICHOLS CHART COMPARISON.

By using these time domain simulation results (figures 72 and 73) and the open loop frequency characteristics of the system as shown on the Nichols chart in figure 74, the following tabulated closed loop characteristics can be observed (table 8) for a unit step input.

NA = not applicable, %OS = percentage overshoot, GM = gain margin, PM = phase margin, BW = bandwidth, T_d = delay time, T_r = rise time, T_s = settling time, e_o = steady state error, u_o = steady state control input.

	GM (dB)	PM (°)	BW (rad/s)	%OS (%)	T_d (s)	T_r (s)	T_s (s)	e_o (V)	u_o (V)
H_∞ #1	21.95	78.07	0.13	0.08	7.17	15.84	22.28	$890.60*10^{-9}$	0.55
H_∞ #2	20.44	75.29	0.15	0.3	7.19	14.37	20.53	$9.37*10^{-15}$	0.55
FLC	NA	NA	NA	0.87	7.45	14.64	20.34	0	0.55
QFT #1	20.58	71.52	0.11	0	13.06	19.53	29.56	0	0.55
QFT #2	12.29	69.01	0.10	0	14.26	21.43	34.54	0	0.55
PI	22.24	81.86	0.16	0	5.53	13.90	19.84	0	0.55

TABLE 8: TABULATED CLOSED LOOP CHARACTERISTICS.

As tabulated in table 8, judging the robustness properties of the controlled system by means of gain (GM) and phase margins (PM), the PI controller is more robust with a gain margin of 22.24dB and a phase margin of 81.86°. This is achieved at the expense of the control input used. Reducing the gain of the controller will increase the gain margin, however this is at the sacrifice of system performance. The H_∞ design 1 and QFT design 1 are comparatively similar in robustness properties. The only discrepancy between these controllers are their response times under closed loop control. Thus in order to evaluate the performance of the closed loop system, figures 75 and 76 below show the frequency of occurrence of the error signal as a percentage of time. Figure 75 shows the frequency response due to a unit step input and figure 76 shows the response due to a 3KW load disturbance.

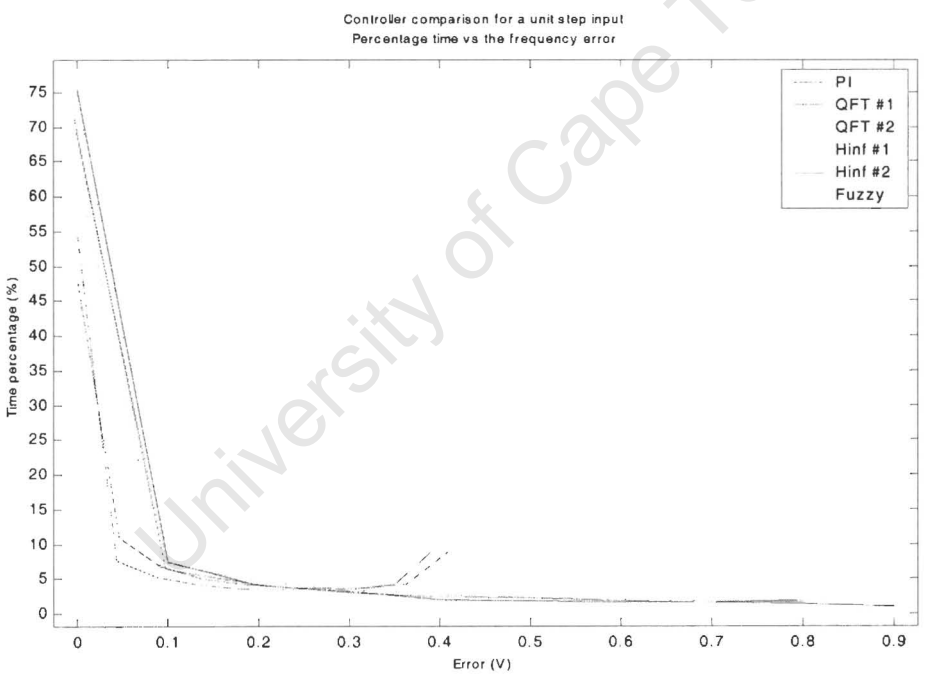


FIGURE 75: THE ERROR AS A PERCENTAGE OF TIME FOR A UNIT STEP INPUT.

As seen from figure 75, the error signal is zero for 75% of the time in the case of the PI controller followed by 70% for the H_∞ design (1 & 2) together with that of the fuzzy logic controller and 55% and 48% for the QFT design 1 and 2 respectively. Also shown is the maximum error excursion of the response, being 0.9V for the PI,

This indicates that the QFT design is best at minimising the frequency error deviation due to a unit step input. Figure 76 below illustrates the performance of the system due to load disturbance. It shows that the QFT design regulates the system error deviation to zero 57% of the time and that of controllers PI, H_∞ and fuzzy logic to approximately 55% of the time.

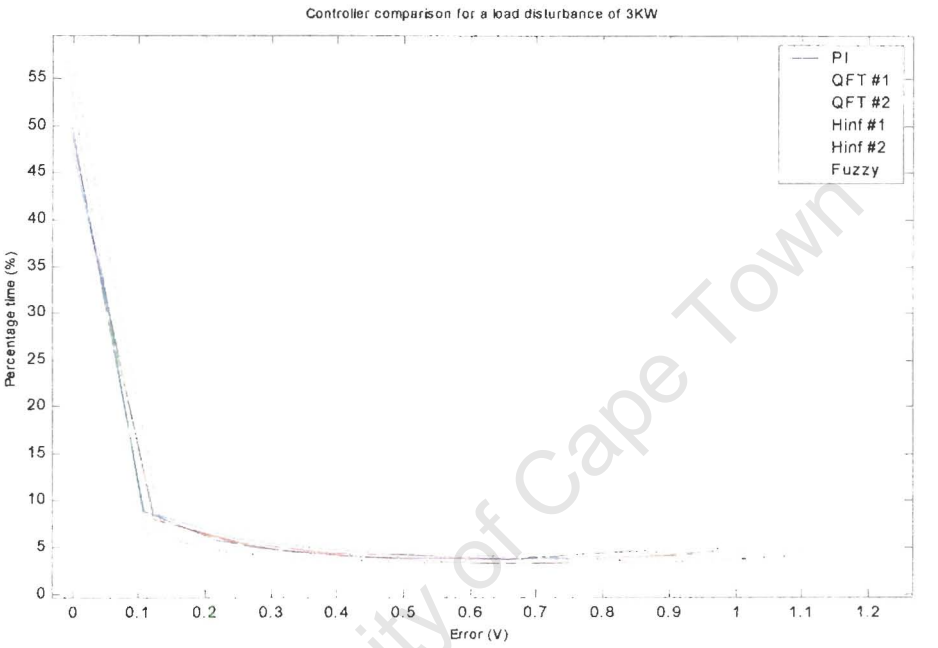


FIGURE 76: THE ERROR AS A PERCENTAGE OF TIME FOR 3KW LOAD DISTURBANCE.

To complement the above discussion on the robustness properties of the system, the following figure illustrates the performance of the system by varying the plant gain (figure 77) and that of the time delay (figure 78). It shows the above performance index as a function of these variables. Placing equal emphasis on the error and control input signals; the weighting gains k_1 and k_2 are chosen as 0.5. These graphs show that the controllers can maintain system performance over a large operating region.

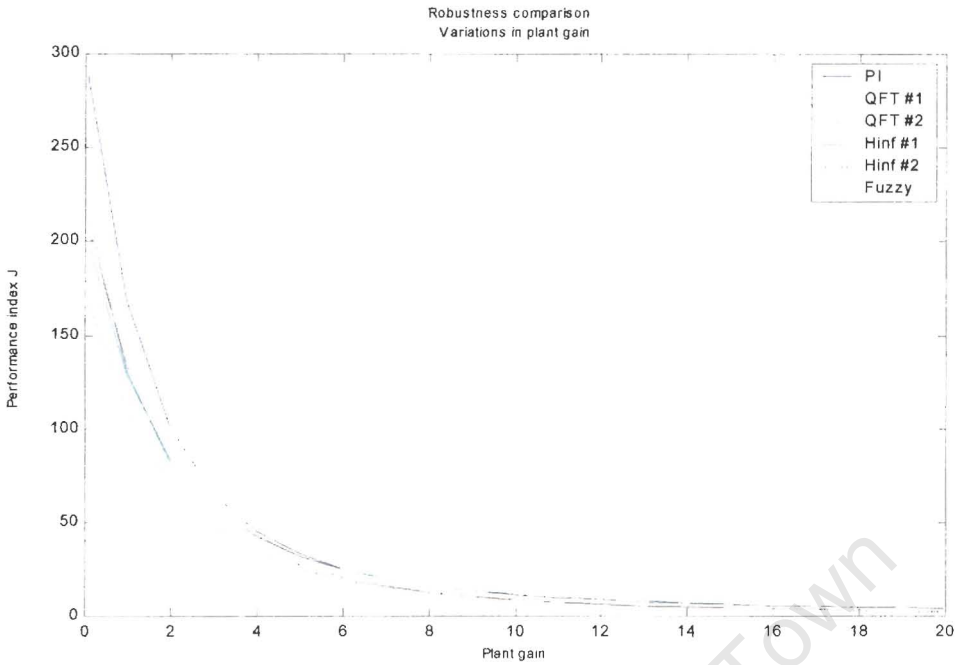


FIGURE 77: VARIATION OF PLANT GAIN ON ROBUSTNESS.

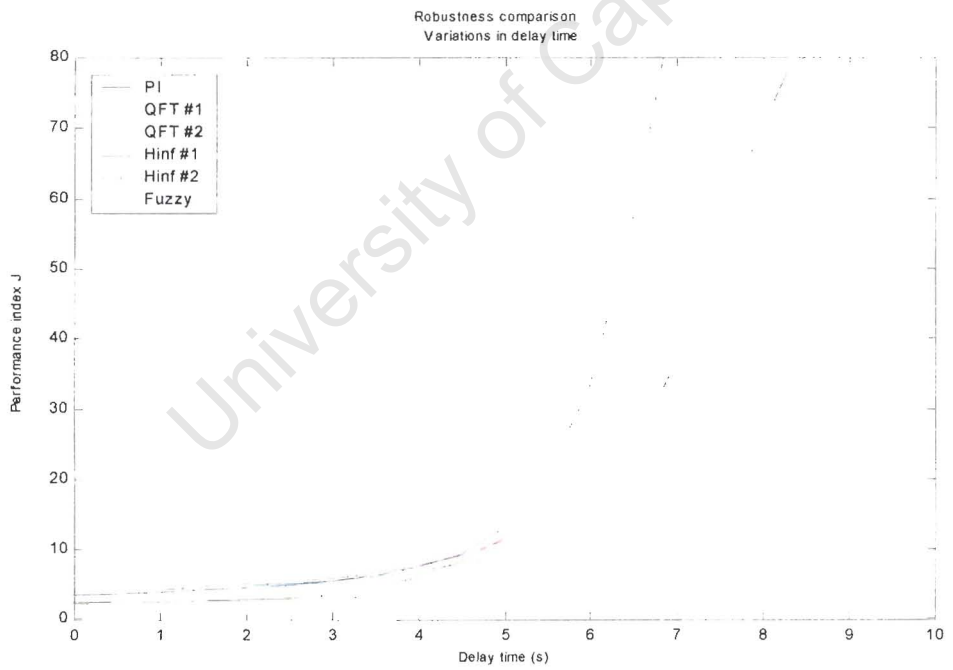


FIGURE 78: ROBUSTNESS COMPARISON DUE TO DELAY TIME VARIATION.

CHAPTER 7

7. CONCLUSION

This thesis describes the application of robust controller design techniques to the load frequency control problem of power systems. The controller design methods investigated are H_∞ optimal control, quantitative feedback theory and fuzzy logic control. For comparison purposes the classical PI controller is described. Presented are the basic theoretical aspects of load frequency control. This is followed by theoretical expositions of the mathematical frameworks involved in the above controller design strategies.

It is shown that these control methodologies perform satisfactorily over a wide operating region, maintaining the stability of the system, robust performance and sufficient disturbance rejection properties. With these properties in mind, the above controllers are proposed as candidate solutions to load frequency control with the exception of the PI controller. Though the PI controller in comparison with H_∞, QFT and fuzzy logic control compared favourably, it lacks the extra degree of freedom necessary in the shaping of the loop transmission which is vital to load frequency control design.

The H_∞ and QFT design methods are model based techniques, contrary to that of the fuzzy logic controller, which is a nonmathematical approach dependent on expert knowledge of the plant. This characteristic of the fuzzy logic controller is desirable in circumstances where a mathematical model of the plant is not available or difficult to obtain, such as load frequency control.

The design intricacy of the H_∞ synthesis method is that of the selection of weighting functions. These weighting functions impose the desired closed loop specifications of the system on the synthesis process. The complexity of the controller thus depends on

the complexity of the weighting functions and that of the plant. An undesirable feature of the mixed sensitivity H_∞ synthesis problem formulation is that of pole zero cancellation. In this formulation the controller zeros cancels all poles of the plant G(s), thus it removes undesirable system modes and inserts poles reflecting the desired closed loop characteristics. Despite this disadvantage, after a few Monte Carlo simulations the H_∞ controller design yields a robust controller meeting system specifications.

In QFT however, the designer has direct access to the synthesis of the controller. This method is a step by step process where dynamic controller elements are added sequentially and is transparent to the designer. There are similarities between the H_∞ and QFT design methods, these include the fact that both techniques are frequency domain methods. Secondly, performance constraints are imposed as frequency dependent functions on closed loop transfer functions, ie. the sensitivity function S(s), complementary sensitivity function T(s) and the control sensitivity M(s).

Fuzzy logic control, as presented performs acceptably, however it is noted that the conventional approach to fuzzy controller design following fuzzzification, inferencing and defuzzification is to manipulate the proportional and integral gains of controller. However this manipulation of gains changes the effective universe of discourse of the system thus changing the membership functions. Though this may not prove to be disadvantageous, it is a point worth noting. Due to its lack of mathematical rigour, the assessment of stability of the fuzzy system proves to be an arduous task. This motivates stability analysis by means of simulation. This is in contrast to H_∞ and QFT where stability is based on their pole positions.

Thus modern robust controller design methods such as those described above provide adequate closed loop regulation in the presence of uncertainties and output disturbances. The recommendation is thus made that these controllers be considered as solutions to the load frequency control problem of power systems.

Bibliography

Asgharian, R (1994), *" A robust H_∞ power system stabilizer with no adverse effect on shaft torsional modes"*, IEEE transactions on energy conversion, Vol. 9, No.3, pp. 475 – 481.

Asgharian, R and Tavakoli, S.A (1996), *" A systematic approach to performance weights selection design of robust H^∞ PSS using genetic algorithms"*, IEEE transactions on energy conversion, Vol. 11, No. 1, pp 111 – 117.

Åström, K.J (1983), *" Theory and applications of adaptive control – a survey"*, Automatica, Vol. 19, No. 5, pp 471 – 486.

Azvine, B and Wynne, R.J (1992), " A review of quantitative feedback theory as a robust control system design technique", Trans Inst. MC, Vol. 14, No. 5, pp 265 – 279.

Bose, A and Atiyyah, I (1980), *"Regulation error in load frequency control"*, IEEE transactions on power apparatus and systems, Vol. PAS-99, No. 2, pp 650 - 657.

Braae, M and Rutherford, D.A (1979), *" Theoretical and linguistic aspects of the fuzzy logic controller"*, Automatica, Vol. 15, pp. 553 – 577.

Bryant, G.F. and Halikias, G.D. (1995), *"Optimal loop shaping for systems with large parameter uncertainty via linear programming"*. Int. J. Control, Vol. 62, No.3, pp. 557 – 568.

Chait, Y and Chen, Q and Hollot, C.V (1999), *"Automatic loop – shaping of QFT controllers via linear programming"*.

Chen, S and Malik, O.P (1995), " *H_∞ optimisation based power system stabiliser design*", IEE proceedings on generation, transmission and distribution, Vol. 142, No. 2, pp. 179 – 184.

Chen, S.B and Fan, Y.H and Zhang, F.E (1997), " *Case study comparison of robust linear quadratic design and mixed – sensitivity H_∞ control*", IEE proceedings on control theory and applications, Vol. 144, No. 5, pp 476 – 480.

Chen, W and Balance, D.J and Li, Y (1998), " Automatic loop shaping in QFT using genetic algorithms".

Chen, W and Ballance, D.J (1997), *"On choice of the nominal plant in quantitative feedback theory"*.

Cherry, A.S and Jones, R.P (1995), " *Fuzzy logic control of an automatic suspension system*". IEE proceedings on control theory and applications, Vol. 142, No. 2, pp. 149 – 160.

Choi, S.B and Cho, S.S (1999), " *Vibration and position tracking control of piezoceramic based smart structures via QFT*", Journal of dynamic systems, measurement and control, Vol. 121, pp. 27 – 33.

Chown, G and Hartman, R.C (1997), " *Design and experience with a fuzzy logic controller for automatic generation control (AGC)*", IEEE PICA conference, Columbus Ohio.

Chwee, K.N and Li, Y , *"Reduced rule-based and direct implementation of fuzzy logic control systems"*.

Doyle, J.C and Glover, K and Khargonekar, P.P and Francis, B.A (1989), " *State – space solutions to standard H_2 and H_∞ control problems*", IEEE transactions on automatic control, Vol. 34, No. 8, pp. 831 – 847.

Duncan, G.A (1995), *" Digital control system design for a unique nonlinear MIMO process using QFT technique"*, IEE proceedings on control theory and applications, Vol. 142, No. 5, pp. 466 – 474.

Fosha, C.E and Elgerd, O. I (1970), *" The megawatt – frequency control problem: A new approach via optimal control theory"*, IEEE Transactions on power apparatus and systems, Vol. PAS-89, pp. 563 – 577.

Fosha, C.E and Elgerd, O. I (1970), *"The megawatt – frequency control problem: A new approach via optimal control theory"*, IEEE Transactions on power apparatus and systems, Vol. PAS-89, pp. 563 – 577.

Francis, B.A and Zames, G (1984), *" On H^{∞} optimal sensitivity theory for SISO feedback systems"*, IEEE transactions on automatic control, Vol. AC-29, No. 1 pp 9 – 16.

Gera, A and Horowitz, I (1980), *" Optimization of the loop transfer function"*, International journal of control, Vol. 31, No. 2, pp. 389 – 398.

Glover, J.D and Sarma (1994), *"Power system analysis and design"*, Imprint Boston: PWS Pub.

Graham, D and Lathrop, R.C (1953), *" The synthesis of optimum transient response"*, AIEE, pp. 273 – 288.

Handschin, E and Petroianu, A (1991), *"Energy management systems"*, Springer-Verlag Berlin, Heidelberg.

Hayakawa, K and Matsumoto, K and Yamashita, M and Suzuki, Y and Fujimori, K and Kimura, H (1999), *" Robust H^{∞} output feedback control of decoupled automobile active suspension systems"*, IEEE transactions on automatic control, Vol. 44, No. 2, pp 392 – 396.

Horowitz, I (1991), *" Survey of quantitative feedback theory (QFT)"*, International journal of control, Vol. 53, No. 2, pp. 255 – 291.

Horowitz, I and Sidi, M (1972), *" Synthesis of feedback systems with large plant ignorance for prescribed time domain tolerances"*, International journal of control, Vol. 16, No. 2, pp. 287 – 309.

Horowitz, I. (1973), *"Optimum loop transfer function in single loop minimum – phase feedback systems"*. Int. J. Control, Vol. 18, No. 1, pp. 97 – 113.

Horowitz, I. (1993), *"Quantitative feedback design theory (QFT), Vol. 1"*. QFT Publications.

Jaleeli, N and VanSlyck, L.S and Ewart, D and Fink, L.H and Hoffmann, A.G (1992), *" Understanding automatic generation control"*, Transactions on power systems, Vol. 7, No. 3, pp 1106 – 1121.

Kandel A and Langholz, G (1994), *"Fuzzy conrol systems"*, CRC Press, Inc.

Katebi, M.R and Grimble, M.J and Zhang, Y (1997), *" H∞ robust control design for dynamic ship positioning"*, IEE proceedings on control theory and applications, Vol. 144, No. 2, pp 110 – 120.

Kiszka, J.B and Gupta, M.M and Nikiforuk, P.N (1985), *" Energetistic stability of fuzzy dynamic systems"*, IEEE transactions on systems, man and cybernetics, Vol. SMC-15, No. 6, pp. 783 – 792.

Kumar, A. and Malik, O. P. and Hope, G.S. (1987), *"Discrete variable structure controller for load frequency control of multiarea interconnected power systems"*, IEE Proceedings, Vol. 134, Pt.C, No. 2, pp. 116 – 120.

Kumar, J and Ng, K-H and Sheble, G (1997), " *AGC simulator for price based operation part 1: A model*", IEEE transactions on power systems, Vol. 12, No. 2, pp. 527 – 532.

Kumar, J and Ng, K-H and Sheble, G (1997), " *AGC simulator for price based operation part 2: Case study results*", IEEE transactions on power systems, Vol. 12, No. 2, pp. 533 – 538.

Kumar, S.R and Majumder, D.D (1984), " *Application of circle criteria for stability analysis of line SISO and MIMO systems associated with fuzzy logic controller*", IEEE transactions on systems, man and cybernetics, Vol. SMC-14, No. 2, pp. 345 – 349.

Kuo, B.C (1991), *"Automatic control systems"*, 6th edition, Prentice Hall Inc.

Kwakernaak, H (1993), " *Robust control and H_∞ optimization – tutorial paper*", Automatica, Vol. 29, No. 2, pp. 255 – 273.

Lim, K.Y and Wang, Y and Zhou, R (1996), " *Robust decentralized load frequency control of multi – area power systems*", IEE proceedings on generation, transmission and distribution, Vol. 143, No. 5, pp. 377 – 386.

Maciejowski, J (1989), " *Multivariable feedback design*", Addison – Wesley publishers Ltd.

Mamdani, E.H (1974), *"Application of fuzzy algorithms to control a simple dynamic plant"*, Proc. IEE, Vol. 121, pp. 1858 – 1888.

McFarlane, D and Glover, K (1992), " *A loop shaping design procedure using $H\infty$ synthesis*", IEEE transactions on automatic control, Vol. 37, No. 6, pp 759 – 769.

Meisma, G (1995), " *Unstable and nonproper weights in H_∞ control*", Automatica, Vol. 31, No. 11, pp 1655 – 1658.

Messer, A.C and Grimble, M.J (1993), " *Introduction to robust ship track – keeping control design"*, Trans Inst. MC, Vol. 15, No. 3, pp 104 – 110.

Miller, R.H and Malinowski, J.H (1993), *"Power system operation"*, Third edition , Mcgraw-Hill, Inc.

Miniesy, S.M and Bohn, E.V (1971), "*Optimum load frequency continuous control of unknown deterministic power demand"*, IEEE transactions, pp 1910 – 1915.

Mollman, L.A and Kennedy, T (1968), " *Interrelationship of time error, frequency deviation, and inadvertent flow on an interconnected system"*, IEEE transactions on power apparatus and systems, Vol. PAS – 87, No. 2, pp 520 – 526.

Nwokah, O.D.I and Thompson, D.F (1989), *"Algebraic and topological aspects of quantitative feedback theory"*, Vol. 50, No. 4, pp 1057 – 1069.

Orero, S.O and Irving, M.R (1996), *" Economic dispatch of generators with prohibited operating zones: a genetic algorithm approach"*, IEE proceedings on generation, transmission and distribution, Vol. 143, No. 6, pp 529 – 534.

Perng, J.W and Han, K.C and Tsai, S.J and Han, K.W (1998), *" State – space solution of the standard H_∞ control problem for a strip gauge control"*, IEE proceedings on control theory and applications, Vol. 145, No. 3, pp. 291 – 298.

Pierre, D.A (1987), *" A perspective on adaptive control of power systems"*, IEEE transactions on power systems, Vol. PWRS – 2, No. 2, pp. 387 – 396.

Pruessmann, D. (1997), *"Fuzzy logic supervisory control for coal power plant"*, FuzzyTech application note.

Rao, P.S and Sen, I (1999), *" Robust tuning of power system stabilizers using QFT"*, IEEE transactions on control systems technology, Vol. 7, No. 4, pp. 478 – 486.

Reznik, L (1997), *" Fuzzy controllers"*, Newnes, An imprint of Butterworth - Heinemann.

Robel, G (1989), *"On computing the Infinity norm"*, IEE transactions on automatic control, Vol. 34, No.8.

Rodrigues, J.M and Chait, Y and Hollot, C.V (1997), *" An efficient algorithm for computing QFT bounds"*, Transactions on ASME, Vol. 119, pp. 548 - 552.

Sidi, M (1976), *" Feedback synthesis with plant ignorance, nonminimum phase, and time domain tolerances"*, Automatica, Vol. 12, pp. 265 – 271.

Skogestad, S and Postlethwaite, I (1996), *" Multivariable feedback control"*, John Wiley & Sons Ltd.

Song, Y.H and Chou, C.S.V (1997), *" Advanced engineered conditioning genetic approach to power economic dispatch"*, IEE proceedings on generation, transmission and distribution, Vol. 144, No. 3, pp. 285 – 292.

Tong, R.M (1977), *" A control engineering review of fuzzy systems"*, Automatica, Vol. 13, pp 559 – 569.

Usry, R. O (1968), *"Inadvertent energy interchange – causes, remedies and balancing"*, IEEE transactions on power apparatus and systems, Vol. PAS – 87, No. 2, pp 513 – 520.

Vajk, I and Vajta, M and Keviczky, L and Haber, R and Hetthessy, J and Kovacs K (1985), *" Adaptive load frequency control of the Hungarian Power system"*, Automatica, Vol. 21, No. 2, pp 129 – 137.

Vajk, I. and Vajta, M. and Keviczky, L. and Haber, R. and Hetthessy, J. and Kovacs K. (1985), *"Adaptive load frequency control of the Hungarian Power system"*, Automatica, Vol. 21, No. 2, pp 129 – 137.

VanDoren, V.J (1999), *"Assessing control loop performance"*, Control Engineering, pp 91 – 96.

Wang, Y and Zhou, R and Wen, C. (1993), *" Robust load frequency controller design for power systems"*, IEE Proceedings – C, Vol. 140, pp. 11 – 16.

Wang, Y. and Zhou, R. and Wen, C. (1993), *"Robust load frequency controller design for power systems"*, IEE Proceedings – C, Vol. 140, pp. 11 – 16.

Williams, S.J (1991), *"H_∞ for the layman"*, Measurement and control, Vol. 24, pp 18 – 21.

Wood, A.J and Wollenberg, B.F. (1996). *"Power generation, operation and control"*. John Wiley & Sons, INC.

Wu, S.F and Grimble, M.J and Breslin, S.G (1998), *" Introduction to quantitative feedback theory for lateral robust flight control system design"*, Control engineering practice, Vol. 6, pp 805 – 828.

Yalcinoz, T and Short, M.J (1997), *" Large scale economic dispatch using an improved Hopfield neural network"*, IEE proceedings on generation, transmission and distribution, Vol. 144, No. 2, pp. 181 – 185.

Yamashita, K and Miyagi H (1991). *" Multivariable self-tuning regulator for load frequency control system with interaction of voltage on load demand"*, IEE Proceedings – D, Vol. 138, No. 2, pp 177 – 183.

Yamashita, K. and Miyagi H, (1991). *"Multivariable self-tuning regulator for load frequency control system with interaction of voltage on load demand"*, IEE Proceedings – D, Vol. 138, No. 2, pp 177 – 183.

Yaniv, O (1995), *" Robust feedback synthesis for margins at the plant input"*, *Automatica*, Vol. 31, No. 2, pp. 333 – 336.

Yaniv, O and Horowitz, I. (1990), *"Quantitative feedback theory for vibration control synthesis"*. Int. J. Control. Vol. 51, No. 6, pp. 125 – 1258.

Zadeh, L.A (1965). *" Fuzzy sets. Information and control"*.

Zafiriou, E and Morari, M (1986), *" Design of robust digital controllers and sampling time selection for SISO systems"*, International journal of control, Vol. 44, No. 3, pp 711 – 735.

Zolotas, A.C and Halikias, G.D (1999), *" Optimal design of PID controllers using the QFT method"*, IEE proceedings on control theory and applications, Vol. 146, No. 6, pp. 585 – 589.

APPENDIX 1

A.1. THE MODEL OF A SINGLE CONTROL AREA

Figure 79 below shows the model of the system that will be used throughout course of this project. G(s) is the open loop plant model and $G_d(s)$ is the disturbance model, while u(s), d(s) and f(s) are the system input, load disturbance and output frequency respectively.

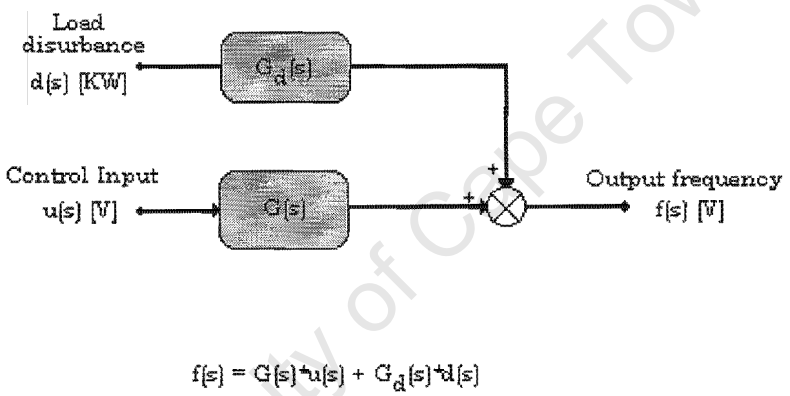


FIGURE 79: THE OPEN LOOP MODEL OF THE SYSTEM.

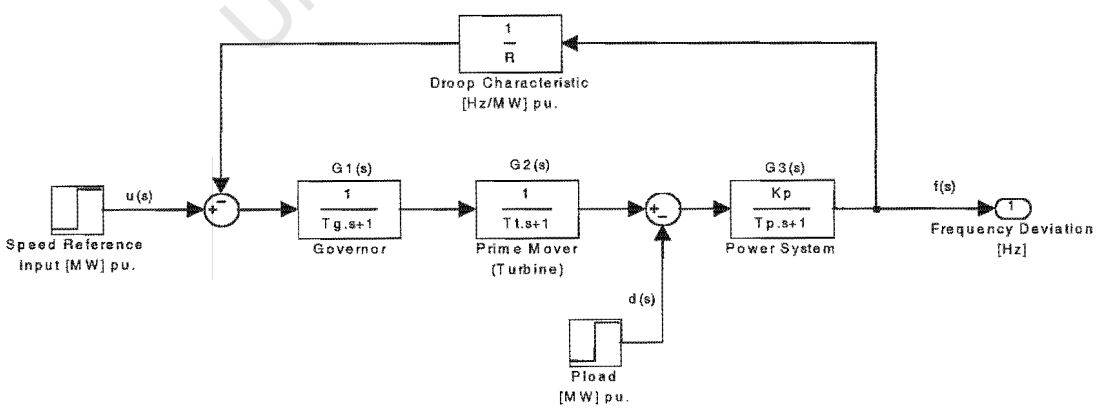


FIGURE 80: POWER SYSTEM MODEL FOR LOAD FREQUENCY CONTROL.

Figure 80 above is the power system model commonly used for load frequency control studies (Miller and Malinowski, 1994; Wood and Wollenberg, 1996; Glover and Sarma, 1994). Using block diagram algebra and arranging the model shown in figure 80 to resemble that of figure 79 the following transfer function can be obtained.

- The transfer function between u(s) and f(s) by setting d(s) = 0 (see figure 81).

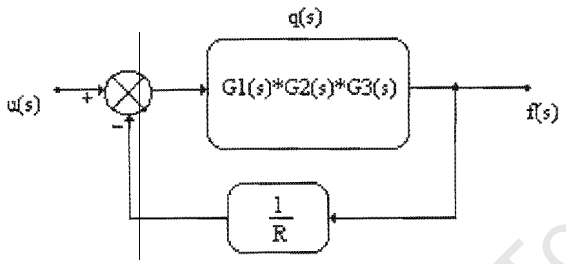


FIGURE 81: BLOCK DIAGRAM FOR u(s) AND f(s).

$$G(s) = \frac{f(s)}{u(s)} = \frac{q(s)}{1 + \frac{1}{R} * q(s)}$$

$$= \frac{G1(s) * G2(s) * G3(s)}{1 + \frac{1}{R} * G1(s) * G2(s) * G3(s)}$$

$$= \frac{Kp}{(Tgs + 1)(Tts + 1)(Tps + 1) + \frac{Kp}{R}}$$

- The transfer function between d(s) and f(s) by setting u(s) = 0 (see figure 82).

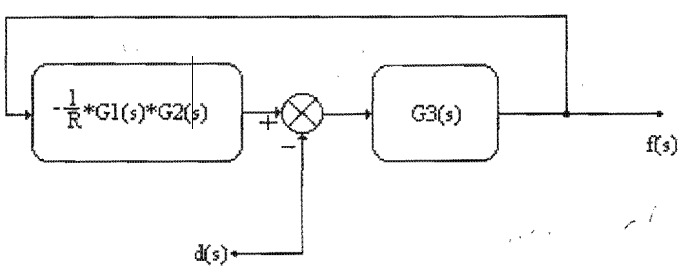


FIGURE 82: BLOCK DIAGRAM BETWEEN D(S) AND F(S).

$$y(s) = \left(-\frac{1}{R} * G1(s) * G2(s) * y(s) - d(s) \right) * G3(s)$$

$$\Rightarrow \left(1 + \frac{1}{R} * G1(s) * G2(s) * G3(s) \right) * y(s) = -G3(s) * d(s)$$

$$\therefore \frac{y(s)}{d(s)} = \frac{-G3(s)}{\left(1 + \frac{1}{R} * G1(s) * G2(s) * G3(s) \right)}$$

$$\Rightarrow G_d(s) = \frac{y(s)}{d(s)} = \frac{-Kp * (Tgs + 1)(Tts + 1)}{(Tgs + 1)(Tts + 1)(Tps + 1) + \frac{Kp}{R}}$$

Thus the steady state error of the system in response to a load disturbance as seen from the above transfer function $G_d(s)$ is proportional to the droop characteristic R. That is,

$$\Delta f(s) = -R * \Delta d(s)$$

where $\Delta f(s)$ is the frequency deviation and $\Delta d(s)$ is the load deviation (Wood and Wollenberg, 1996). This motivates the use for supplementary controllers. Model parameters are chosen as Kp = 120 Hz/MW, Tp = 20s, Tg = 0.08s, Tt = 0.3s, R = 2.4 Hz/MW, all parameters are per unit values (Bose and Atiyyan, 1980).

APPENDIX 2

A.2. H_∞ CONTROLLER DESIGN BY TRIAL AND ERROR

This section aims at obtaining appropriate weighting function parameters for weights W_1, W_2 and W_3 for the standard H_∞ controller synthesis problem. The weights are defined as,

$$W_i(s) = K\frac{\frac{1}{M}s + \omega_B}{s + A * \omega_B} \qquad i = 1,3$$

$$W_2(s) = C \qquad \text{(a constant)}$$

Initially ω_B is chosen as 10 for all weighting functions. The section below tabulates the poles and zeros of the synthesised controller followed by a time domain plot of the output response (for both the step and disturbances responses). Thus this appendix can be divided into the following sections.

Section A2.1 : Variations in A for weight W_1 (table 4, figures 70 and 71).
Variations in M for weight W_1 (table 5, figures 72 and 73).

Section A2.2 : Variations in C for weight W_2 (figures 74 and 75).

Section A2.3 : Variations in A for weight W_3 (table 6, figures 76 and 77).
Variations in M for weight W_3 (table 7, figures 78 and 79).

SECTION A2.1

W_1 : (A varies, M = 0.01, K = 10e-3, ω_B = 10); W2 = 3.5 ; W3 : (A = 10, M = 0.01, K = 10, ω_B = 10).

	Plant G(S)	H_∞ controller #1 (A = 10e-6)	H_∞ controller #2 (A = 10e-4)	H_∞ controller #3 (A = 20e-4)	H_∞ controller #4 (A = 50e-4)	H_∞ controller #5 (A = 80e-4)
Poles	-0.27619 + 5.6709i -0.27619 - 5.6709i -2 -0.17161	-100 -2.3302 + 6.0848i -2.3302 - 6.0848i -2.4264 + 0.9379i -2.4264 - 0.9379i -0.0001	-100 -2.3152+ 6.0747i -2.3152 - 6.0747i -2.9752+ 0.6825i -2.9752 - 0.6825i -0.01	-100 -2.3433 + 6.1139i -2.3433 - 6.1139i -1.669 + 0.5569i -1.669 - 0.5569i -0.02	-100 -2.3443 + 6.1311i -2.3443 - 6.1311i -1.9682 -0.35422 -0.05	-100 -2.345+ 6.1241i -2.345 - 6.1241i -1.7738 -0.97184 -0.08
Zeros	2	-100 -0.27619 + 5.6709i -0.27619 - 5.6709i -2 -0.17161	-100 -0.27619+ 5.6709i -0.27619 - 5.6709i -2 -0.17161	-100 -0.27619+ 5.6709i -0.27619 - 5.6709i -2 -0.17161	-100 -0.27619+ 5.6709i -0.27619 - 5.6709i -2 -0.17161	-100 -0.27619+ 5.6709i -0.27619 - 5.6709i -2 -0.17161
Gain	-10.02	1.6113	2.0512	0.58364	0.052299	0.069249
Gamma (γ)	N/A	0.42578	0.46094	0.5	0.67578	0.91406

TABLE 9: POLES AND ZEROS TABLE OF PLANT G(S) AND CONTROLLER K(S).

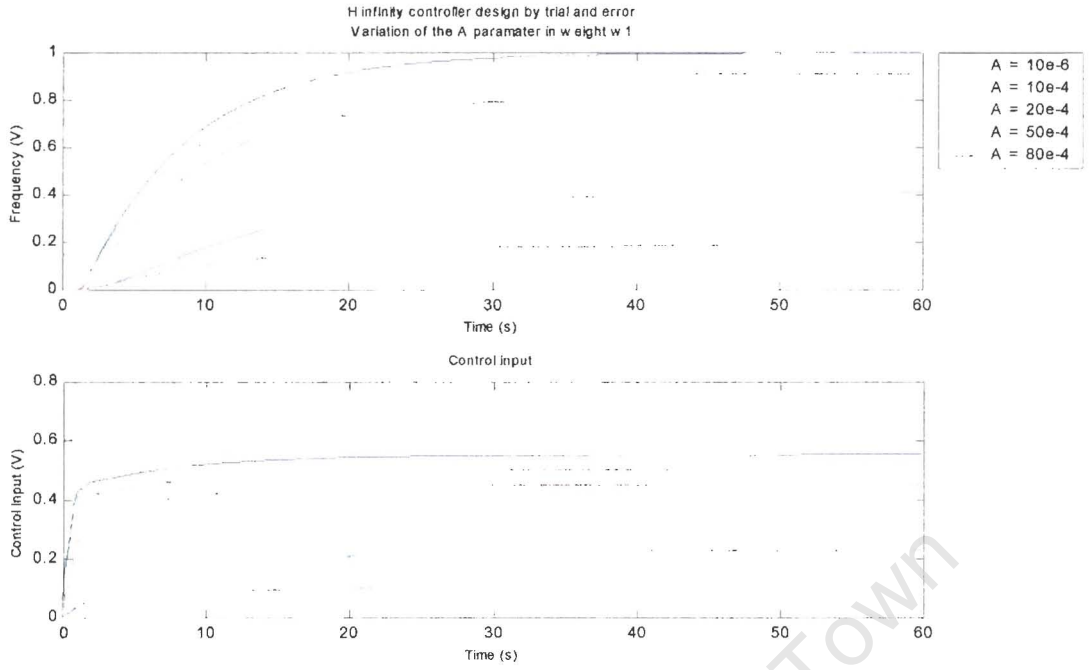


FIGURE 83: EFFECT VARYING THE A PARAMTER IN W_1.

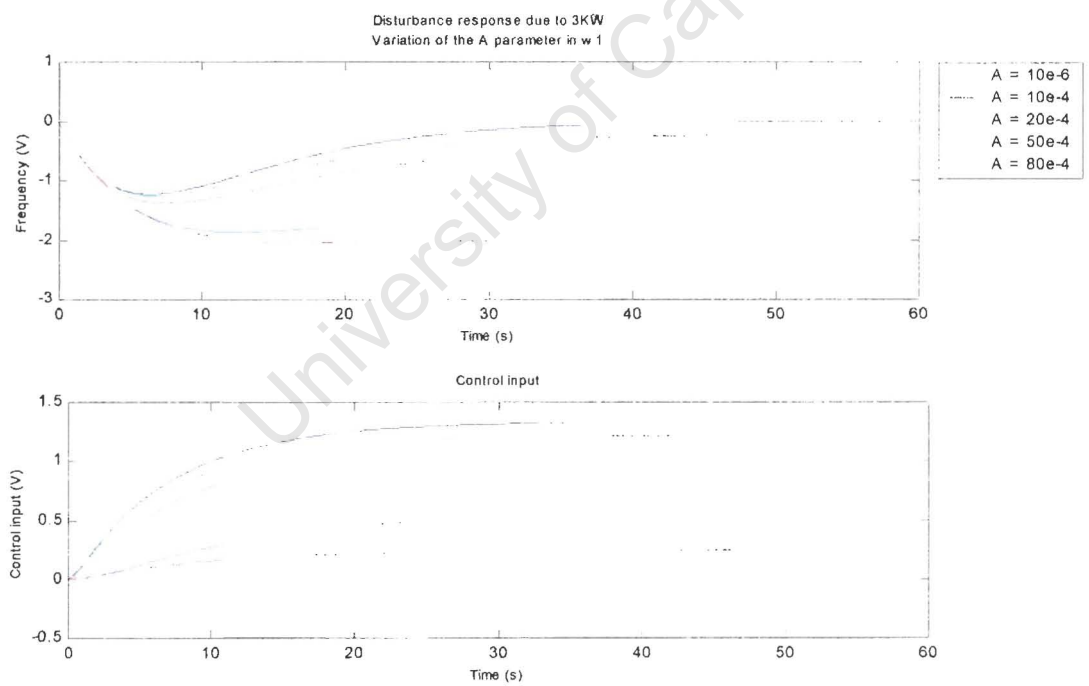


FIGURE 84: EFFECT OF VARYING THE A PARAMETER OF W_1 ON THE DISTURBANCE RESPONSE.

W_1 : (A = 10e-6, M = varies, K = 10e-3, ω_B = 10); W2 = 3.5 ; W3 : (A = 10, M = 0.01, K = 10, ω_B = 10).

	Plant G(S)	H_∞ controller #6 (M = 0.0025)	H_∞ controller #7 (M = 0.00375)	H_∞ controller #8 (M = 0.005)	H_∞ controller #9 (M = 0.01)	H_∞ controller #10 (M = 0.05)
Poles	-0.27619 + 5.6709i -0.27619 - 5.6709i -2 -0.17161	-100 -1.7338 + 5.9869i -1.7338 - 5.9869i -5.6685 + 2.2425i -5.6685 - 2.2425i -0.0001	-100 -10.235 -2.0362 + 6.0579i -2.0362 - 6.0579i -3.4208 -0.0001	-100 -2.233 + 6.0277i -2.233 - 6.0277i -5.1045 -3.2169 -0.0001	-100 -2.3302 + 6.0848i -2.3302 - 6.0848i -2.4264 + 0.9379i -2.4264 - 0.9379i -0.0001	-100 -30.732 -2.2372 + 6.1367i -2.2372 - 6.1367i -2.3759 -0.0001
Zeros	2	-100 -0.27619 + 5.6709i -0.27619 - 5.6709i -2 -0.17161	-100 -0.27619 + 5.6709i -0.27619 - 5.6709i -2 -0.17161	-100 -0.27619 + 5.6709i -0.27619 - 5.6709i -2 -0.17161	-100 -0.27619 + 5.6709i -0.27619 - 5.6709i -2 -0.17161	-100 -0.27619 + 5.6709i -0.27619 - 5.6709i -2 -0.17161
Gain	-10.02	2.0431	3.6323	2.5913	1.6113	20.18
Gamma (γ)	N/A	0.24219	0.3418	0.39453	0.42578	0.43359

TABLE 10: POLES AND ZEROS FOR H_∞ DESIGN.

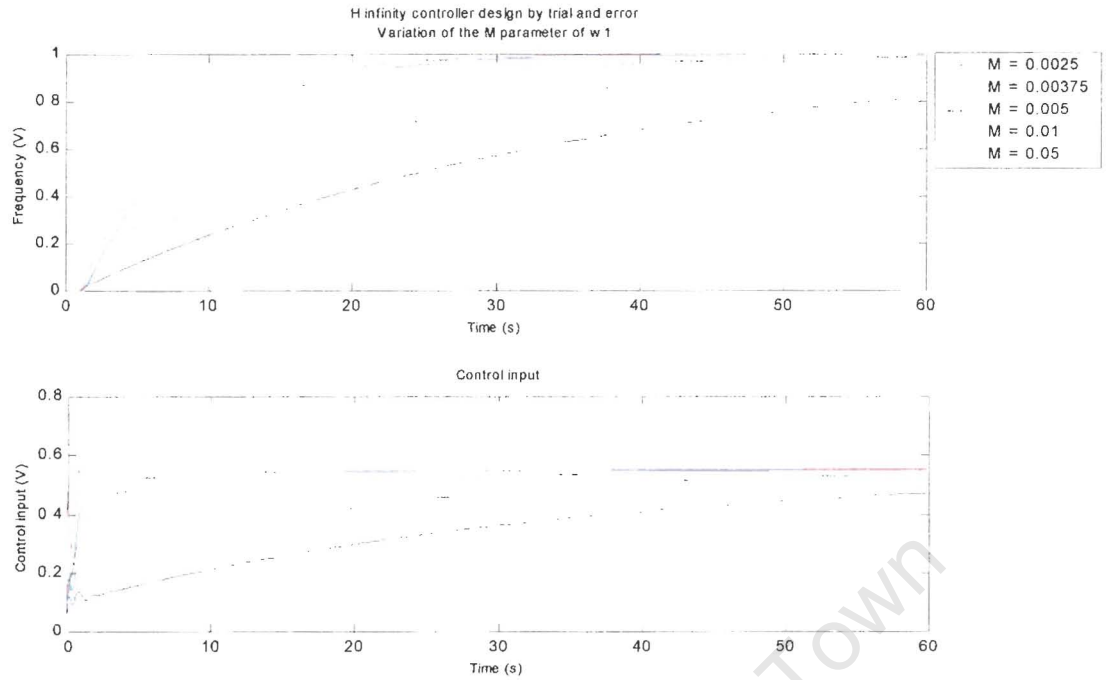


FIGURE 85: VARIATION OF THE M PARAMETER IN WEIGHT W_1.

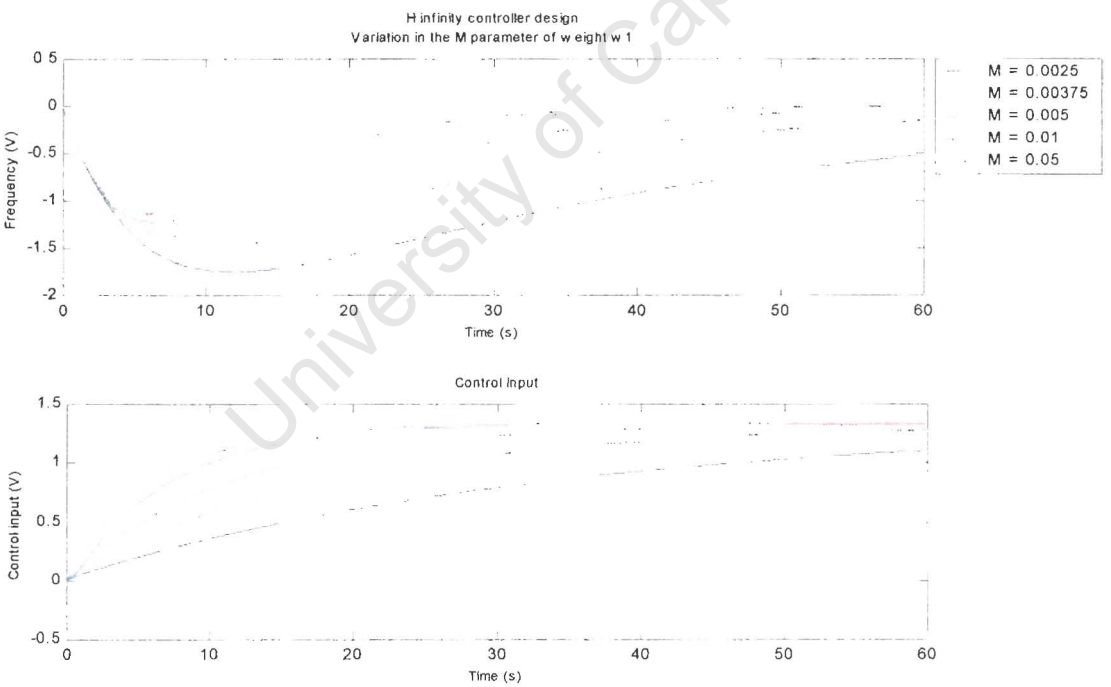


FIGURE 86: DISTURBANCE REJECTION RESPONSE.

SECTION A2.2

W_1 : (A = 10e-6, M = 0.01, K = 10e-3, ω_B = 10); W_2 = varies ; W_3 : (A = 10, M = 0.01, K = 10, ω_B = 10).

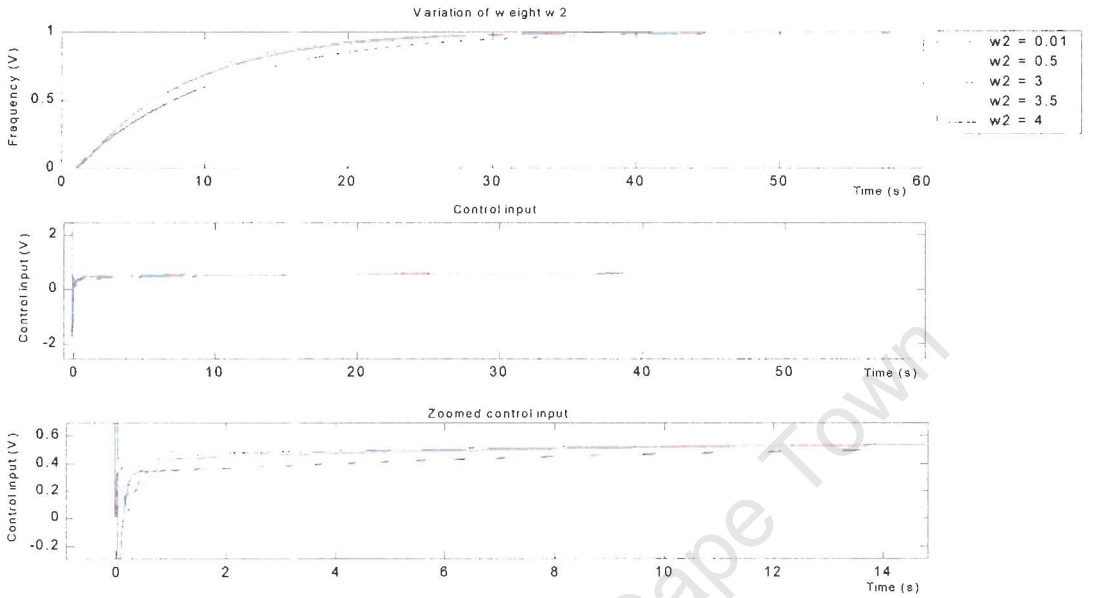


FIGURE 87: THE EFFECT OF VARYING WEIGHT W_2 DUE TO A STEP INPUT.

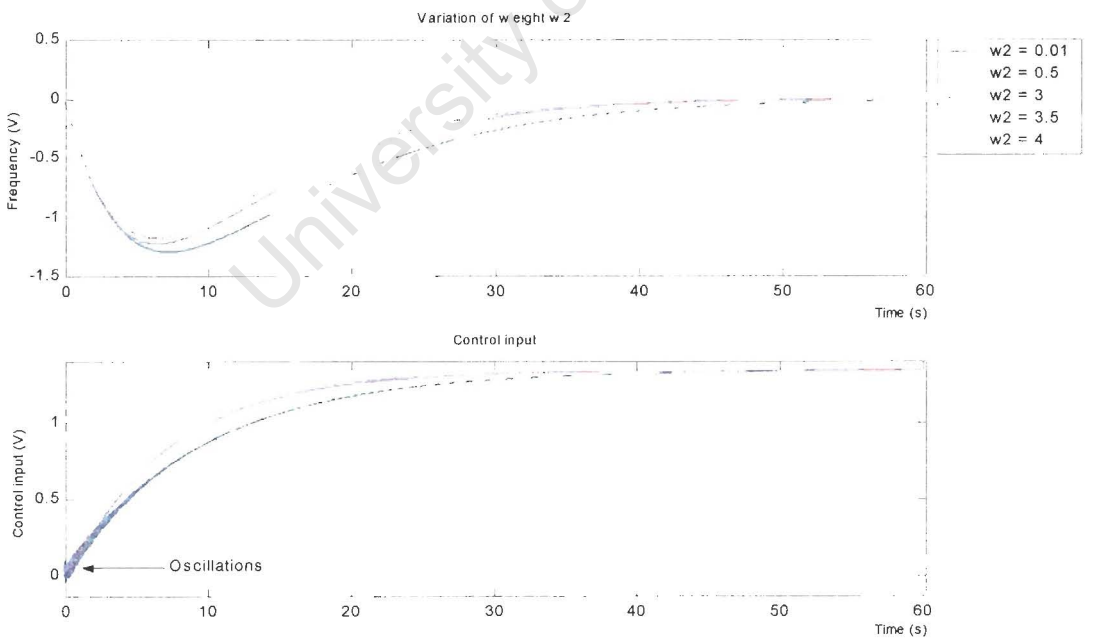


FIGURE 88: THE EFFECT OF VARYING WEIGHT W_2 ON THE DISTURBANCE RESPONSE.

SECTION A2.3

W_1 : (A = 10e-6, M = 0.01, K = 10e-3, ω_B = 10); W2 = 3.5 ; W3 : (A = varies, M = 0.01, K = 10, ω_B = 10).

	Plant G(s)	H_∞ controller #11 (A = 0.5)	H_∞ controller #12 (A = 1)	H_∞ controller #13 (A = 5)	H_∞ controller #14 (A = 10)	H_∞ controller #15 (A = 50)
Poles	-0.27619 + 5.6709i -0.27619 - 5.6709i -2 -0.17161	-6.8177 + 12.699i -6.8177 - 12.699i -13.8 -1.9958 -0.15375 -0.0001	-14.977 -7.0017 + 11.963i -7.0017 - 11.963i -1.7004 + 0.43683i -1.7004 - 0.43683i -0.0001	-50.013 -4.0143 + 6.9211i -4.0143 - 6.9211i -7.8437 -2.4772 -0.0001	-100 -2.3302 + 6.0848i -2.3302 - 6.0848i -2.4264 + 0.93793i -2.4264 - 0.93793i -0.0001	-500 -0.52904 + 5.6658i -0.52904 - 5.6658i -2.6276 + 0.88547i -2.6276 - 0.88547i -0.0001
Zeros	2	-0.27619 + 5.6709i -0.27619 - 5.6709i -5 -2 -0.17161	-10 -0.27619 + 5.6709i -0.27619 - 5.6709i -2 -0.17161	-50 -0.27619 + 0.6709i -0.27619 - 5.6709i -2 -0.17161	-100 -0.27619 + 5.6709i -0.27619 - 5.6709i -2 -0.17161	-500 -0.27619 + 5.6709i -0.27619 - 5.6709i -2 -0.17161
Gain	-10.02	0.27835	3.7822	6.3991	1.611	1.5577
Gamma (γ)	N/A	0.049316	0.097656	0.33984	0.42578	0.47656

TABLE 11: VARIATION OF THE A PARAMETER OF WEIGHT w_3.

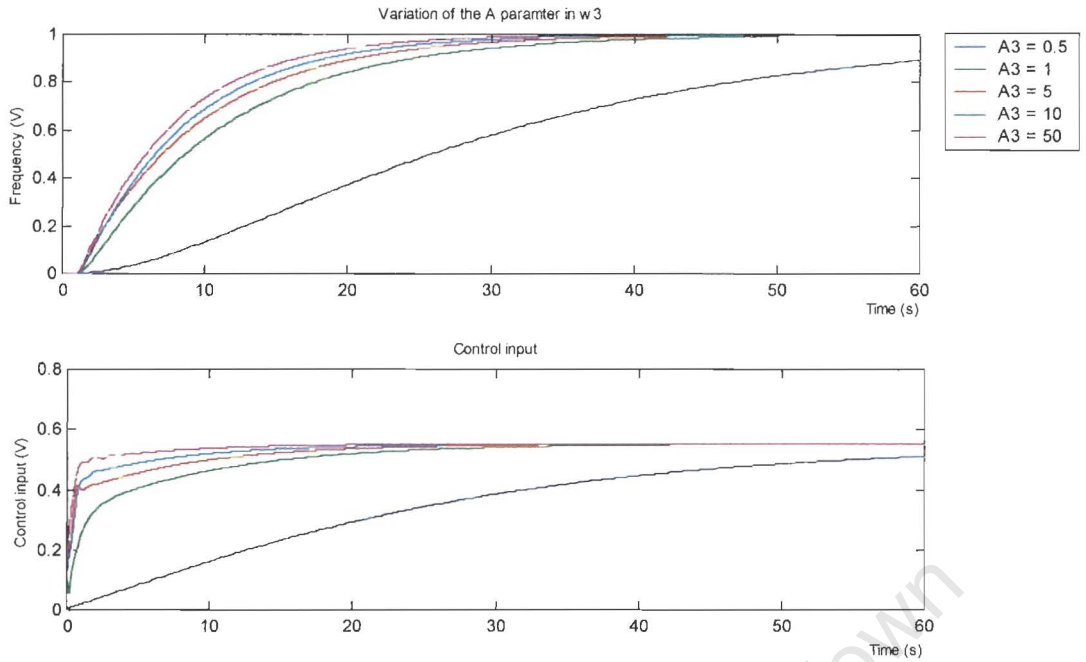


FIGURE 89: VARIATIONS OF THE A PARAMTER IN WEIGHT w3 (STEP RESPONSE).

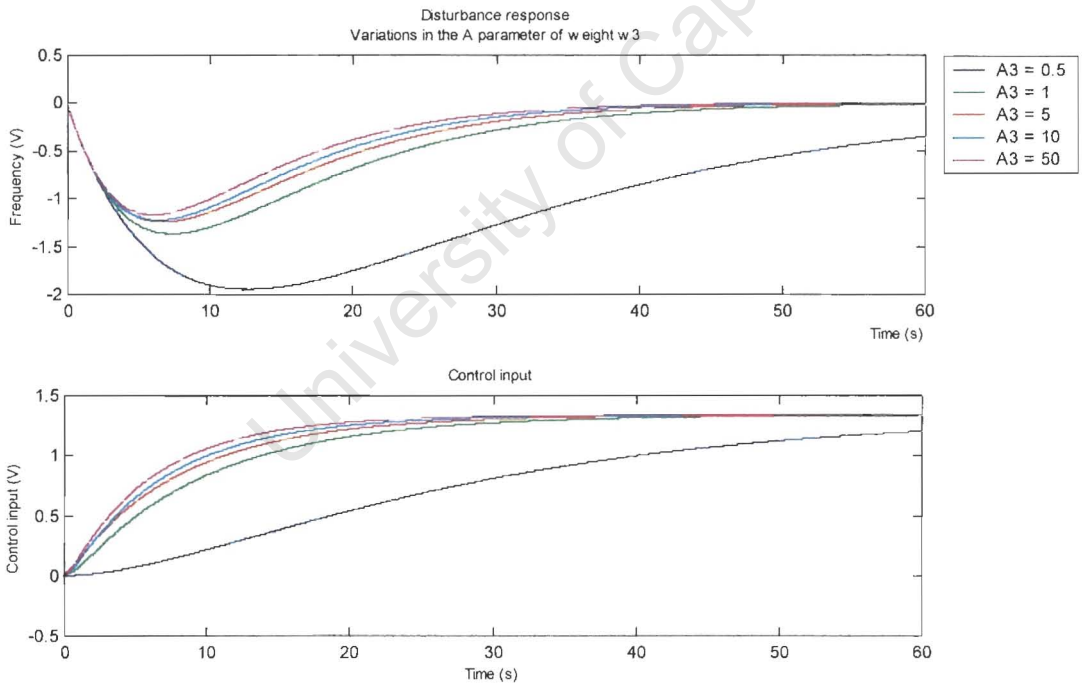


FIGURE 90: VARIATIONS OF THE A PARAMETER IN WEIGHT w_3 (DISTURBANCE RESPONSE).

W_1 : (A = 10e-6, M = 0.01, K = 10e-3, ω_B = 10); W2 = 3.5 ; W3 : (A = 10, M = varies, K = 10, ω_B = 10).

	Plant G(s)	H_∞ controller #16 (M = 0.006)	H_∞ #17 (M = 0.008)	H_∞ #18 (M = 0.01)	H_∞ #19 (M = 0.1)	H_∞ #20 (M = 1)
Poles	-0.27619 + 5.6709i -0.27619 - 5.6709i -2 -0.17161	-100 -32.668 -3.4603 + 6.6776i -3.4603 - 6.6776i -2.3463 -0.0001	-100 -2.8025 + 6.2675i -2.8025 - 6.2675i -3.5238 -3.0809 -0.0001	-100 -2.3302 + 6.0848i -2.3302 - 6.0848i -2.4264 + 0.93793i -2.4264 - 0.93793i -0.0001	-100 -0.30946 + 5.6568i -0.30946 - 5.6568i -3.3205 + 0.20311i -3.3205 - 0.20311i -0.0001	-100 -0.22092 + 5.6516i -0.22092 - 5.6516i -3.1241 + 0.59186i -3.1241 - 0.59186i -0.0001
Zeros	2	-100 -0.27619 + 5.6709i -0.27619 - 5.6709i -2 -0.17161	-100 -0.27619 + 5.6709i -0.27619 - 5.6709i -2 -0.17161	-100 -0.27619 + 5.6709i -0.27619 - 5.6709i -2 -0.17161	-100 -0.27619 + 5.6709i -0.27619 - 5.6709i -2 -0.17161	-100 -0.27619 + 5.6709i -0.27619 - 5.6709i -2 -0.17161
Gain	-10.02	20.524	2.6987	1.6113	2.512	2.2873
Gamma (γ)	N/A	0.41406	0.42188	0.42578	0.4375	0.4375

TABLE 12: VARIATION OF THE M PARAMETER OF WEIGHT w_3.

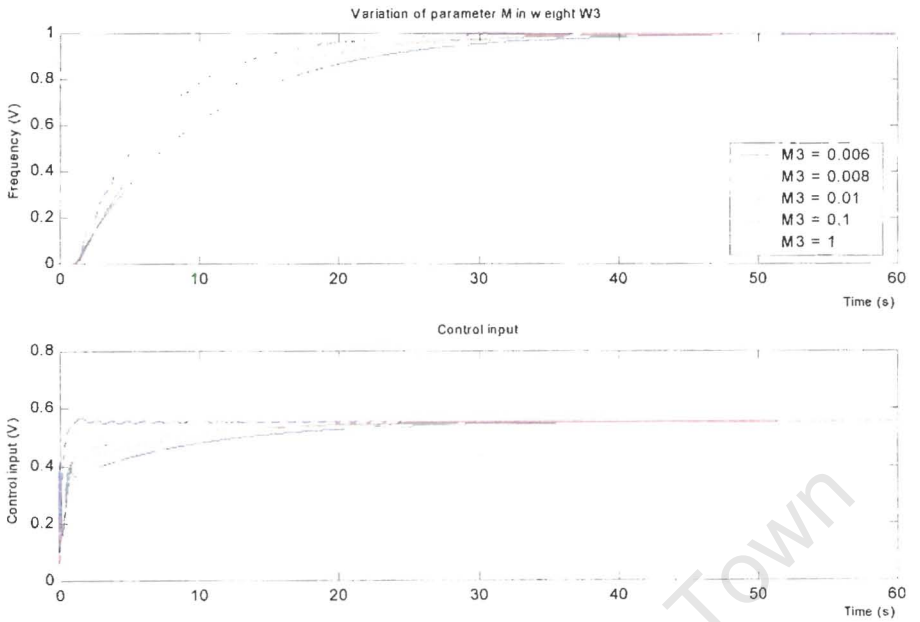


FIGURE 91: VARIATION OF THE M PARAMETER IN WEIGHT W_3 FOR A STEP INPUT.

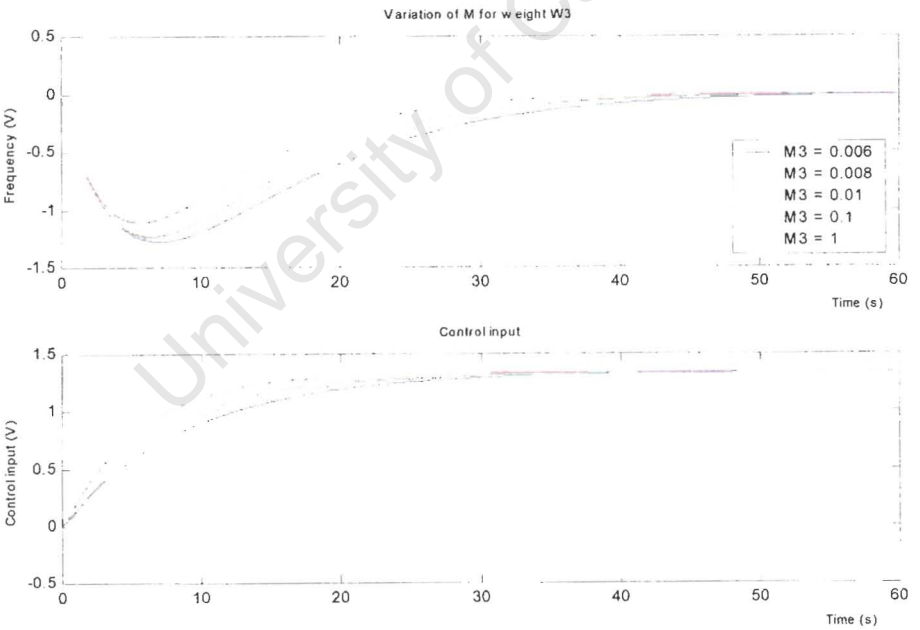


FIGURE 92: VARIATION OF PARAMETER M IN WEIGHT W_3 (DISTURBANCE RESPONSE).

APPENDIX 3

A.3. FUZZY CONTROLLER (MONTE CARLO SIMULATION)

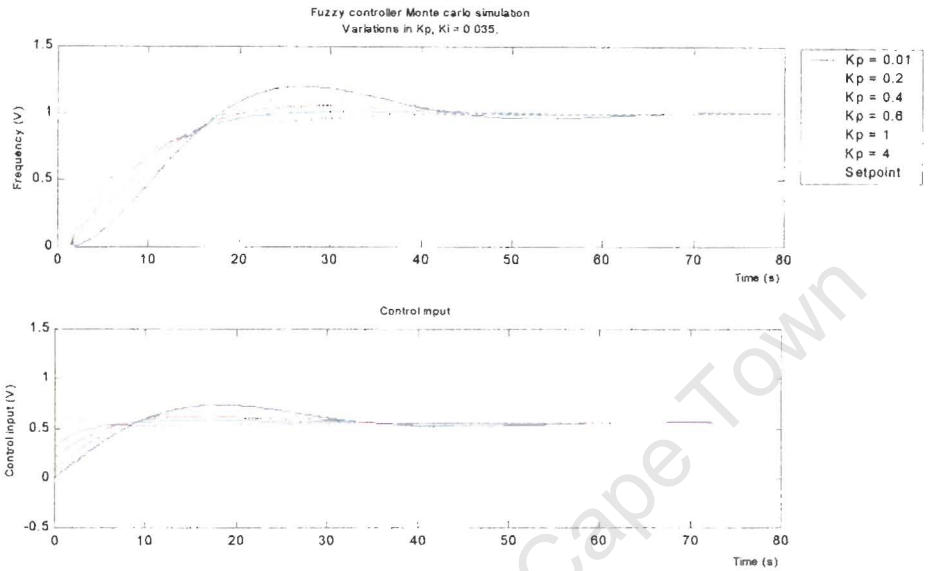


FIGURE 93: UNIT STEP RESPONSE FOR FUZZY CONTROLLER (VARIATIONS IN K_P).

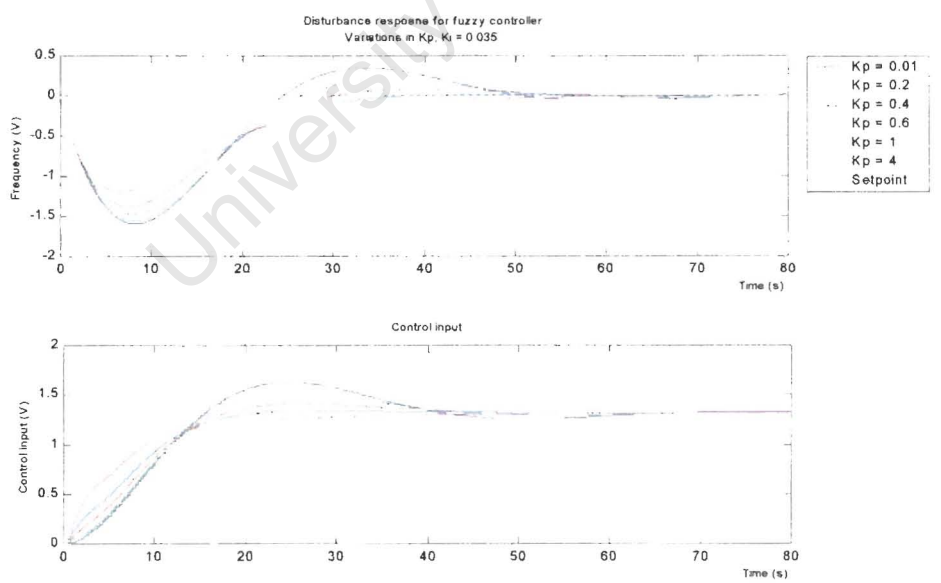


FIGURE 94: DISTURBANCE RESPONSE FOR FUZZY LOGIC CONTROLLER (VARIATIONS IN K_P).

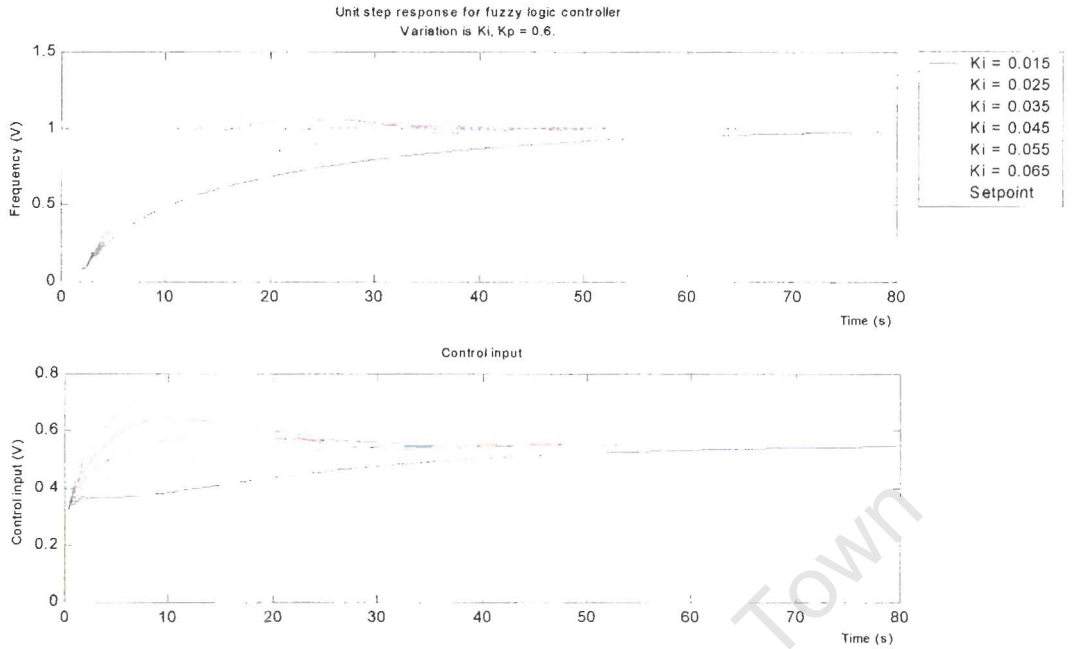


FIGURE 95: FUZZY CONTROLLER RESPONSE (VARIATIONS IN K_i).

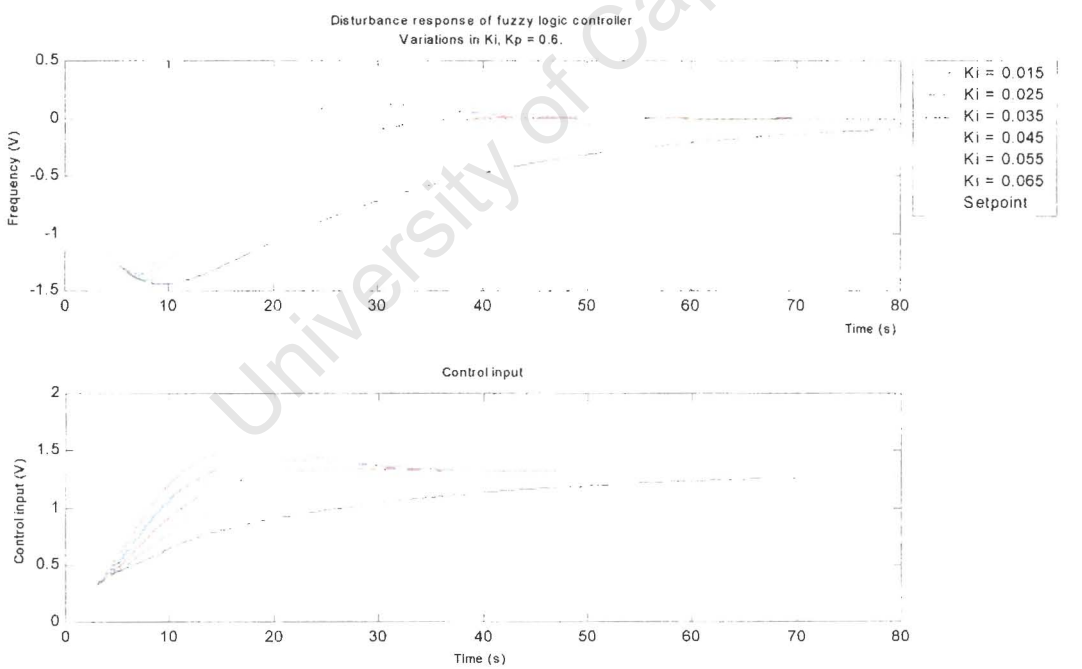


FIGURE 96: FUZZY CONTROLLER DISTURBANCE RESPONSE (VARIATIONS IN K_i).

APPENDIX 4

A.4. MATLAB CODE FOR H_∞ CONTROLLER SYNTHESIS

```
% H infinity optimal controller for Load frequency control.
% Matlab code for calculating the H infinity optimal control law.
% The function used is the matlab "hinfopt" function which iteratively
% solves for the h infinity controller.

clear;
clc;

% G(s) model.
% The model is a second order system with a 1 second
% delay is modelled as a fist order Pade approximation.

Ts = 0.3;
numg = 10.02;
deng = [1 0.724 32.33 5.532];

numd = [521.6];
dend = [1 0.82 3541 646.1];

G = tf(numg,deng);          % G(s) model.
[num,den] = pade(1,1);      % 1 second delay.
d = tf(num,den);

Gs1 = G*d;

[num,den] = tfdata(Gs1,'v');


% Conversion to state space.

[ag,bg,cg,dg] = tf2ss(num,den);
ssg = mksys(ag,bg,cg,dg);

% Weighting functions.

M = 0.01;
A = 10e-6;
wb = 10;

w1 = 10e-3*tf([1/M wb],[1 A*wb]);%w1 for disturbance rejection and integral action.

%design 2.

%A = 0.65;
%B = 3.65;

%w1 = tf([1 2*A A^2],[1 B 0]);
```

```
w2 = 3.5;  % design 1.

%w2 = 2.25;  % design 2.

M = 1e-2;
A = 10;
wb = 10;

w3 = 10*tf([1/M wb],[1 A*wb]);
% The H infinity control law calculation.

[TSS] = augtf(ssg,w1,w2,w3);                    % the augmentation of the plant.
[SS_CPopt,SS_CLopt,HINFOopt] = HINFOPT(TSS);  % H infinity controller synthesis.
[AGopt,BGopt,CGopt,DGopt] = BRANCH(SS_CLopt);
[numcopt,dencopt] = ss2tf(AGopt,BGopt,CGopt,DGopt);
Kopt = tf(numcopt,dencopt);                     %the resultant H infinity controller.


[Kznum,Kzden] = c2dm(numcopt,dencopt,Ts,'tustin');    % H inf controller.

sim('NewHinfDesign11',500);                          % control system simulation

figure(1);
[mag,phase,freq1] = bode(1/w1);
for c = 1:length(mag)
    mag1(c) = mag(c);
end

[mag,phase,freq3] = bode(1/w3);
for c = 1:length(mag)
   mag3(c) = mag(c);
end

semilogx(freq1,20*log10(mag1),'r',freq3,20*log10(mag3),'g');grid;
legend('1/W1','1/W3');

figure(3);
plot(time,freq);title('Frequency');grid;

figure(4);
plot(time,con);title('Control Input');grid;

% Analysis of closed loop system.
% This section plot the respective bode plots of
% the sensitivity function S(s) and complementary sensitivity T(s)
% and their weighting functions.

%Sensitivity function.

   S = 1/(1 + Kopt*G);

   [Smag,phase,Sfreq1] = bode(S);
   for c = 1:length(Smag)
      Smag1(c) = Smag(c);
   end

   figure(5);
```

```
semilogx(Sfreq1,20*log10(Smag1),'r',freq1,20*log10(mag1),'g');grid;
legend('S','1/W1');
title('Sensitivity function');

% Complementary Sensitivity function.

T = (Kopt*G)*S;

[Tmag,phase,Tfreq1] = bode(T);
for c = 1:length(Tmag)
   Tmag1(c) = Tmag(c);
end

figure(6);
semilogx(Tfreq1,20*log10(Tmag1),'r',freq3,20*log10(mag3),'g');grid;
legend('T','1/W3');
title('Complementary Sensitivity function');
```

APPENDIX 5

A.5. MATLAB CODE FOR QFT CONTROLLER DESIGN

```
% Matlab code for QFT controller desig.
% The matlab QFT control design toolbox is used.
clear;
clc;

% G(s) model with first order pade approximation.
K0 = 10.02;
A0 = 0.724;
B0 = 37.33;
C0 = 5.532;

n0 = [K0];
d0 = [1 A0 B0 C0];

% delayed process.

n10 = [-K0 2*K0];
d10 = [1 (A0+2) (B0+2*A0) (C0+2*B0) 2*C0];

%n10 = [K];
%d10 = [1 A B C];

% Plant template generation (for +-20% uncertainty).

c1 = 1;

for K = linspace((K0-0.2*K0),(K0+0.2*K0),4)

   for A = linspace((A0-0.2*A0),(A0+0.2*A0),4)

      for B = linspace((B0-0.2*B0),(B0+0.2*B0),4)

        for C = linspace((C0-0.2*C0),(C0+0.2*C0),4)

          n1(c1,:) = [-K 2*K];
          d1(c1,:) = [1 (A+2) (B+2*A) (C+2*B) 2*C];

          c1 = c1 + 1;

      end
  end
 end
end

% Frequency vector used for the design.

w = [0.001 0.01 0.1 1 4 6 10 20 50 100 250];
```

```
P = freqcp(n1,d1,w);

plottmpl(w,[],P,1);
% Robust stability specifications.

w1 = 1.2;

ws1 = [0.001 0.01 0.1 1 4 6 10];

bnd1 = SISOBNDS(1,w,ws1,w1,P);

% Tracking specification.

Kl = zpk([],[-0.1-0.005i -0.1+0.005i -0.5],[0.0050125]);   % relaxed specs.
Ku = zpk([-3],[-0.3-0.01i -0.3+0.01i],[0.03003333]);

%Kl = tf(1,[0.11 9.44 10.33 1]);  % more (too) stringent specs.
%Ku = tf([0.002 1],[0.001 3 1]);

Kdl = zpk([],[-0.2-0.008i -0.2+0.008i -0.3],[0.0120192]);  % more stringent lower
spec.

[nTl,dTl] = tfdata(Kl,'v');
[nTu,dTu] = tfdata(Ku,'v');

% Frequency vector for tracking specifications
% and for calculating the QFT tracking bounds.

w7 = [0.001 0.01 0.1 1 4 6 10];

[numl,denl] = tfdata(Kl,'v');
ws7l = freqcp(numl,denl,w);
ws7l = abs(ws7l);
[numu,denu] = tfdata(Ku,'v');
ws7u = freqcp(numu,denu,w);
ws7u = abs(ws7u);
ws7 = [ws7u;ws7l];

% bnd7 = SISOBNDS(7,w,w7,ws7,P);


%disturbance specification.
% Disturbance rejection vector.

ws2 = [0.001 0.01 0.1 1 4 6 10];

Kdl = tf(Kdl);

ws = (1 - Kdl);
[num,den] = tfdata(ws,'v');
ws = freqcp(num,den,w);
ws = abs(ws);

bnd2 = SISOBNDS(2,w,ws2,ws,P);

plotbnds(bnd1,1);
plotbnds(bnd2,2);
```

```
%plotbnds(bnd7,7);

%bnd3 = grpbnds(bnd1,bnd2,bnd7);
bnd3 = grpbnds(bnd1,bnd2);

plotbnds(bnd3);

inbnds = sectbnds(bnd3);
plotbnds(inbnds);
title('Intersection of bounds');

numc = [0.8576 91.74 637.6 4075 1.862e4 2.66e4 4039];
denc = [1 11.3 1493 1.224e4 5.075e4 9.253e4 6.313e4 6.843e-11];


numc = [0.3226 0.5963 12.5 15.67 8.156 0.9874];
denc = [1 17.83 45.07 40.63 12.39 0];

nc = [0.3226    0.5963   12.5000   15.6700    8.1560    0.9874];   %QFT design 1
controller for relaxed system.
dc = [1.0000   17.8300   45.0700   40.6300   12.3900         0];

nc = [0.03161 0.1749 1.701 6.609 16.4 19 2.098];  %QFT design 2 more stringent
controller.
dc = [1 10.58 42.7 80.61 68.54 18.95 0];

nc = [0.003807 0.04805 0.221 1.739 2.033 0.258];   % QFT controller for disturbance
rejection.
dc = [1 7.912 13.21 10.2 3.609 0];

% LPshape the semi automatic QFT loop shaping environment.
LPSHAPE(w,inbnds,n10,d10,0,nc,dc);

%pfshape(7,w,w,ws7,P);

% Prefilter design.
% Using the matlab Pfshape function.
nf = 0.13;
df = [1 0.13];

nf = [0.07555 0.2083];   %QFT design 1, for relaxed control system.
df = [1 1.649 0.2083];

nf = [1.095 0.2679 38.18];  % QFT design 2.
df = [1 13.14 56.91];

nf = [0.561 0.00776 17.15]; % QFT design 2 - fine.
df = [1 25.57 17.15];

nf = [0.06426 0.1265];
df = [1 1.165 0.1265];

PFSHAPE(7,w,w,ws7,P,0,[],[],nf,df);

% QFT analysis.

[tout,xstate,yout] = sim('QFTrun',200);
```

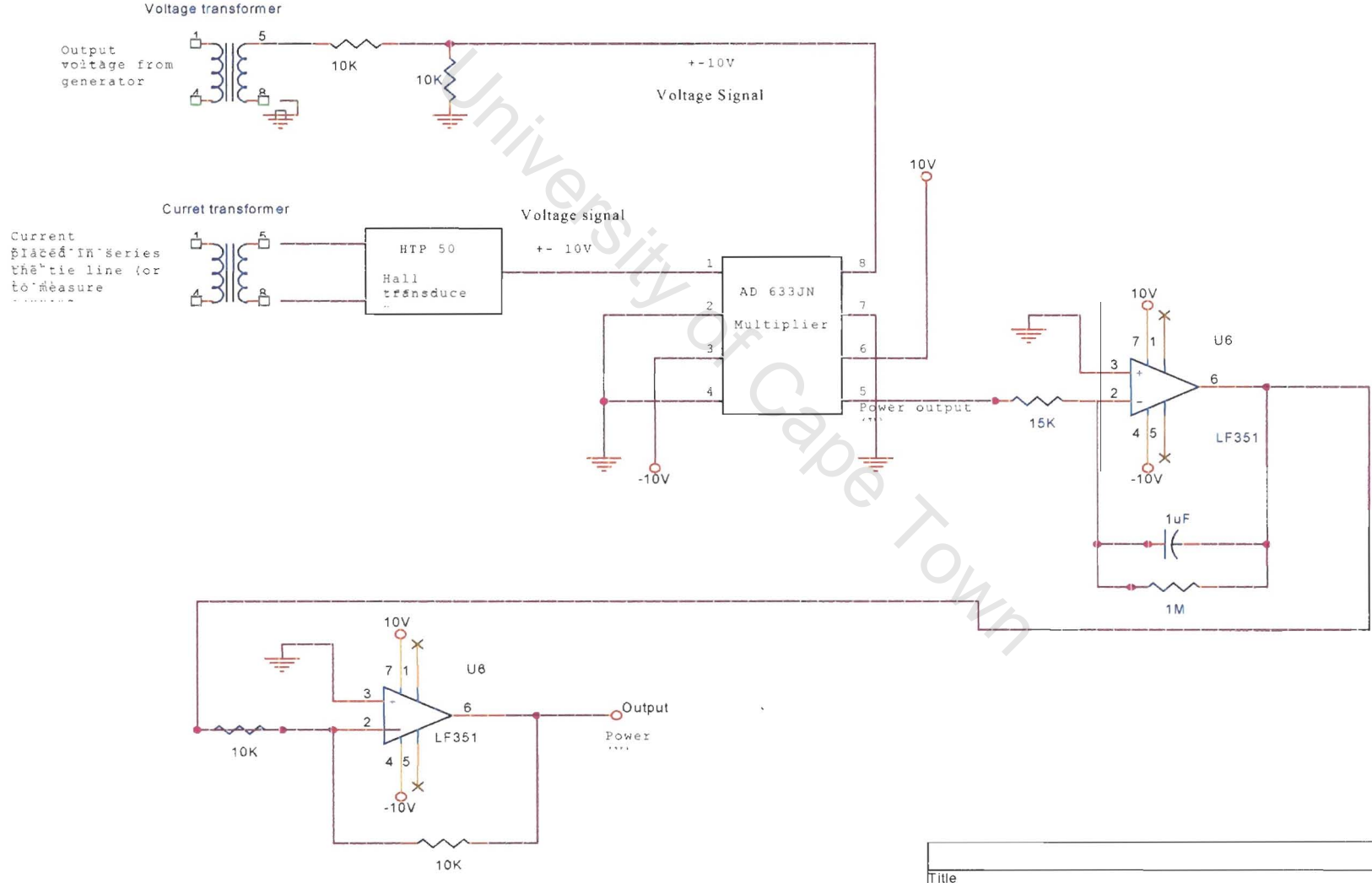
```
figure(24);
plot(tout,yout);
plot(tout,yout(:,2));
figure(25);
hold on;
plot(tout,yout(:,1));

[yTl,tTl] = step(Kl,100);
[yTu,tTu] = step(Ku,100);

[tout,xstate,yout] = sim('QFTrun',200);

%[tout1,xstate,yout1] = sim('Track',60);

figure(26);
hold on;
plot(tout,yout(:,1),tTl+11,yTl,tTu+11,yTu);

figure(27);
hold on;
plot(tout,yout(:,2));

[Kznum,Kzden] = c2dm(nc,dc,0.3,'tustin');
[Fznum,Fzden] = c2dm(nf,df,0.3,'tustin');

K0 = 10.02 + 0*10.02;
A0 = 0.724 + 0*0.724;
B0 = 37.33 + 0*37.33;
C0 = 5.532 + 0*5.532;

% delayed process.

n10 = [-K0 2*K0];
d10 = [1 (A0+2) (B0+2*A0) (C0+2*B0) 2*C0];


F = tf(nf,df);
G = tf(n10,d10);
K = tf(nc,dc);

T = F*((K*G)/(1+K*G));

[m,phase,w] = bode(T);

for c = 1:length(m)

   m1(c) = 20*log10(m(c));
end


[ml,phase,wl] = bode(Kl);

for c = 1:length(ml)

   ml1(c) = 20*log10(ml(c));
end
```

```
[mu,phase,wu] = bode(Ku);

for c = 1:length(mu)
   mu1(c) = 20*log10(mu(c));
end

figure(28);
hold on;
semilogx(w,m1,w1,m11,wu,mu1);

K0 = 10.02 + 0.2*10.02;
A0 = 0.724 + 0.2*0.724;
B0 = 37.33 + 0.2*37.33;
C0 = 5.532 + 0.2*5.532;

% delayed process.

n10 = [-K0 2*K0];
d10 = [1 (A0+2) (B0+2*A0) (C0+2*B0) 2*C0];


F = tf(nf,df);
G = tf(n10,d10);
K = tf(nc,dc);

T = F*((K*G)/(1+K*G));

[m,phase,w] = bode(T);

for c = 1:length(m)

   m1(c) = 20*log10(m(c));
end


[m1,phase,w1] = bode(K1);

for c = 1:length(m1)

   m11(c) = 20*log10(m1(c));
end

[mu,phase,wu] = bode(Ku);

for c = 1:length(mu)

   mu1(c) = 20*log10(mu(c));
end

figure(28);
hold on;
semilogx(w,m1,w1,m11,wu,mu1);

K0 = 10.02 - 0.2*10.02;
A0 = 0.724 - 0.2*0.724;
B0 = 37.33 - 0.2*37.33;
C0 = 5.532 - 0.2*5.532;
```

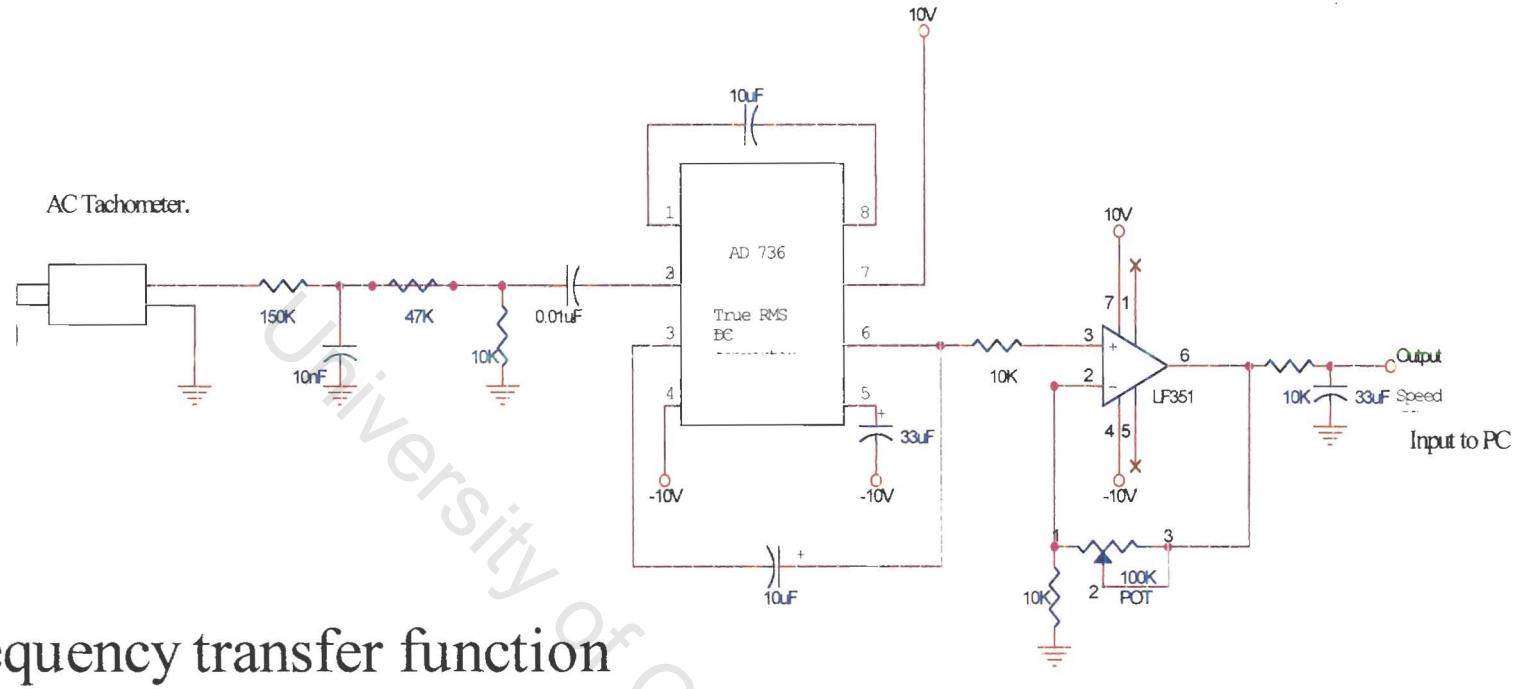
```
% delayed process.

n10 = [-K0 2*K0];
d10 = [1 (A0+2) (B0+2*A0) (C0+2*B0) 2*C0];

F = tf(nf,df);
G = tf(n10,d10);
K = tf(nc,dc);

T = F*((K*G)/(1+K*G));

[m,phase,w] = bode(T);

for c = 1:length(m)

   m1(c) = 20*log10(m(c));
end

[m1,phase,w1] = bode(K1);

for c = 1:length(m1)

   m11(c) = 20*log10(m1(c));
end

[mu,phase,wu] = bode(Ku);

for c = 1:length(mu)

   mu1(c) = 20*log10(mu(c));
end

figure(28);
hold on;
semilogx(w,m1,w1,m11,wu,mu1);

K0 = 10.02 + 0.2*10.02;
A0 = 0.724 + 0.2*0.724;
B0 = 37.33 + 0.2*37.33;
C0 = 5.532 + 0.2*5.532;

% delayed process.

n210 = [-K0 2*K0];
d210 = [1 (A0+2) (B0+2*A0) (C0+2*B0) 2*C0];

K0 = 10.02 - 0.2*10.02;
A0 = 0.724 - 0.2*0.724;
B0 = 37.33 - 0.2*37.33;
C0 = 5.532 - 0.2*5.532;

% delayed process.

n102 = [-K0 2*K0];
d102 = [1 (A0+2) (B0+2*A0) (C0+2*B0) 2*C0];
```

APPENDIX 6

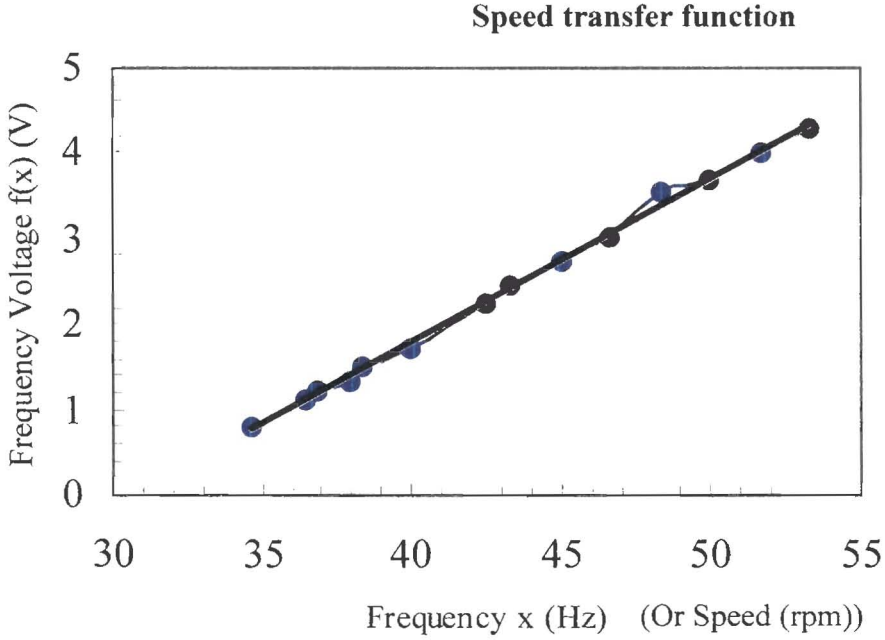
A.6. ELECTRONIC CIRCUITRY FOR MEASUREMENT AND CONTROL



Title	Power sensor module	
Size A	Document Number 3	Rev
Date: Thursday, September 14, 2000	Sheet 1 of 1	



Frequency transfer function



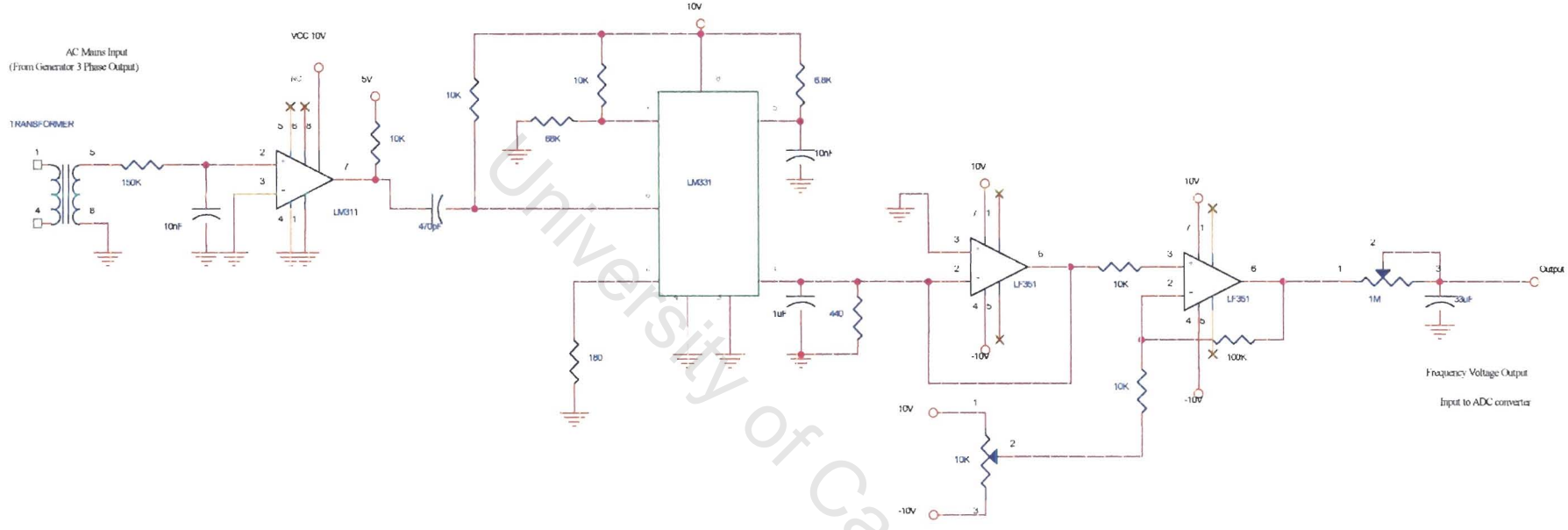
Output frequency voltage

Staight line approximation

$f(x) = 0.1909x - 5.8367$

$f(x) = 0.0032x_{rpm} - 5.8367$

Title: **Speed sensor module**		
Size: A	Document Number: 2	Rev
Date: Thursday, September 14, 2000	Sheet 1 of 1	



Title	Frequency Sensor Module	
Size: A4	Document Number: 1	Rev
Date: Thursday, September 14, 2000	Sheet 1 of 1	

APPENDIX 7

A.7. PHOTOGRAPHS OF EXPERIMENTAL UNIT

Figure 97 (right) shows the complete experimental load frequency control system. It shows the motor generator set (front center), DC power drive (extreme left), load frequency controller (PC, left rear), power supply and electronic measurements circuitry (rear centre) and the excitation system used for the excitation of the AC generator (front right).

FIGURE 97: EXPERIMENTAL LOAD FREQUENCY CONTROL UNIT.

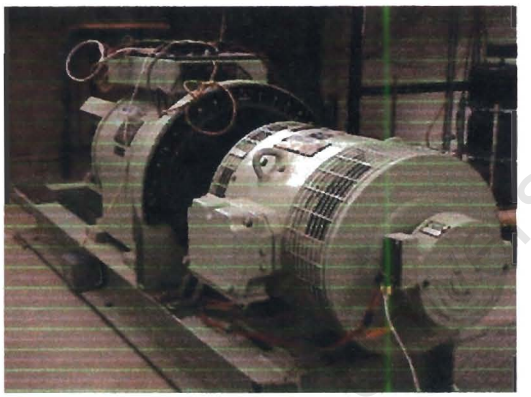
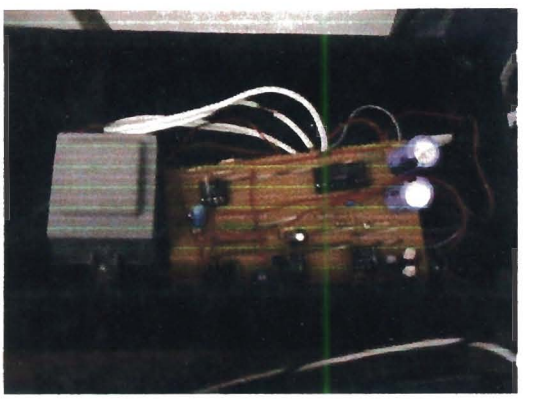


Figure 98 (left) shows the motor generator unit. It consists of a DC motor driving a three phase generator. The generator is positioned to the further left and that of the motor to the nearer right of the observer.

FIGURE 98: MOTOR GENERATOR SYSTEM.



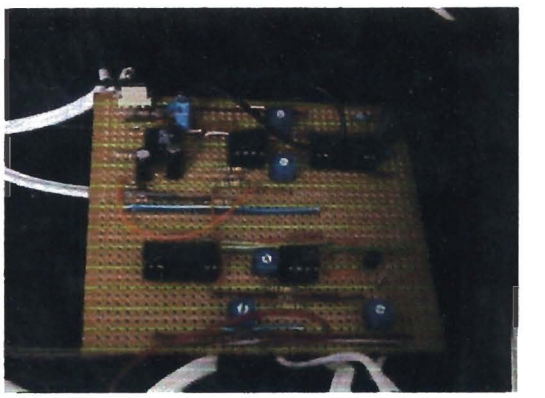


FIGURE 99: THE ELECTRONIC MEASUREMENT CIRCUIRY.

Figure 99 above shows the electronic measurement circuitry for the frequency sensor (left) and the speed sensor (right).

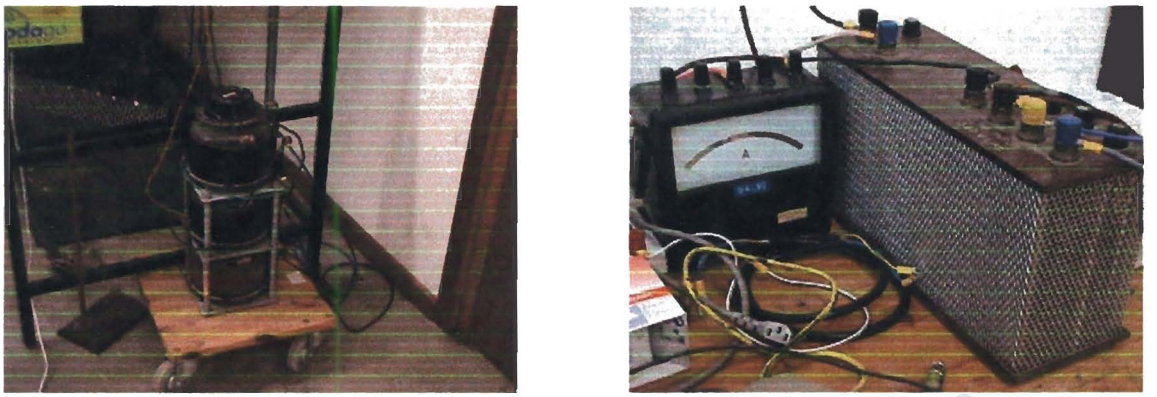


FIGURE 100: EXCITATION SYSTEM.

Shown above in figure 100 is the excitation system used for the excitation of the generation. It consists of a three phase variac (left) and a 100V AC to DC converter.

Figure 101 (right) shows the load used for the experiment. Each heating element is approximately 1KW in power.

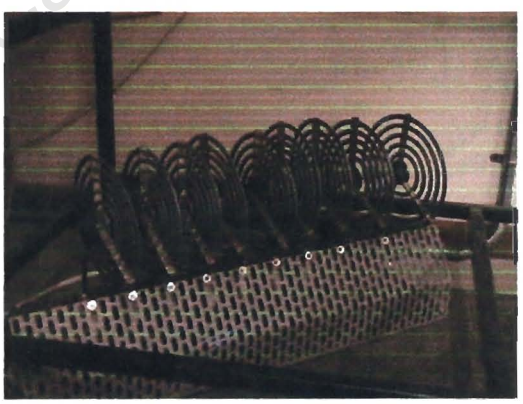


FIGURE 101: ELECTRICAL LOAD.

APPENDIX 8

A.8. VISUAL C++ IMPLEMENTATION CODE

Visual C++ implementation code is given below. Only the relevant source code is shown. The main function routine that performs all the various calling routines is the LFCView.cpp program (shown below). It performs the required control and displays the response data on the screen in real time. Threads are used to achieve this, one thread performs the control task and the other updates the screen as new display data are obtained. The disadvantage to the approach is that depending upon the priority of the process the quality of the displayed results may be delayed. The thread priority used thus, is normal priority.

The various controllers are implemented by using the bilinear transformation, with the exception of the QFT controller designs, these are implemented using the discrete state space implementation. Since the it was observed that the bilinear transformation does not seen to maintain the integral action of the controller when using the QFT design (from the Matlab QFT toolbox), there was a discrepancy between the actual and simulated results in terms of steady state accuracy. This was not the case with the state space implementation.A brief listing the functions routines called are given below.

- LFCView.cpp & LFCView.h : main program, implements the H_∞ optimal control law, QFT controller and the PI controller. Also it displays the response data on the screen.

- Fuzzy_Controller.cpp & Fuzzy_Controller.h : implements the fuzzy logic controller.

- T_Sample.cpp & T_Sample.h : counts the number of clock cycles, thus counting the elapsed time required for the next sampling instant. The input to this function is the sampling time in seconds.

- Ntdt2801.cpp & Ntdt2802.h : code used to interface with the DT2801 ADC/DAC converter. This code was written by Warren.

```
// LFCView.cpp : implementation of the CLFCView class
#include "time.h"
#include "stdafx.h"
#include "LFC.h"
#include "math.h"
#include "Fuzzy_Controller.h"
#include "Ntdt2801.h"
#include "T_Sample.h"
#include "LFCDoc.h"
#include "LFCView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

volatile int threadController = 0;
volatile int who = 0;
UINT ControlProc_Fuzzy(LPVOID param);
UINT ControlProc_Hinf(LPVOID param);
UINT ControlProc_QFT(LPVOID param);
UINT ControlProc_PI(LPVOID param);
UINT ControlProc_StepTest(LPVOID param);
UINT ControlProoc_LoadModelling(LPVOID param);
/////////////////////////////////////////////////////////////////////////////
// CLFCView
IMPLEMENT_DYNCREATE(CLFCView, CView)
BEGIN_MESSAGE_MAP(CLFCView, CView)
        //{{AFX_MSG_MAP(CLFCView)
        ON_COMMAND(ID_MENUITEM32772, OnStopLFC)
        ON_COMMAND(ID_MENUITEM32773, OnFuzzyControl)
        ON_COMMAND(ID_MENUITEM32774, OnHInfinityControl)
        ON_COMMAND(ID_MENUITEM32775, OnQFTControl)
        ON_COMMAND(ID_MENUITEM32776, OnStepTest)
        ON_COMMAND(ID_MENUITEM32777, OnStopModelling)
        ON_COMMAND(ID_MENUITEM32778, OnLoadModelling)
        ON_COMMAND(ID_MENUITEM32779, OnPI)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()
/////////////////////////////////////////////////////////////////////////////
// CLFCView construction/destruction
CLFCView::CLFCView()
{
        // TODO: add construction code here
        tacho_speed_port = 0;
        frequency_port = 1;
        Droop = 0.2;
        frequency = 0;
        tacho_speed = 0;
        Droop_the_system = 0;
        Speed_reference = 0;
        Saftronics_input =0;
        t = 0;
        Error = 0;
        Stop_control = 0; // stop LFC if Stop_control == 1;
        ttemp = 0;
        x0 = 0;
```

```
        for(int c=0;c <= 599;c++)
        {
          xposition[c] = 0;
          yposition[c] = 0;
          cposition[c] = 0;
          setposition[c] = 0;
        }
}
CLFCView::~CLFCView()
{}
BOOL CLFCView::PreCreateWindow(CREATESTRUCT& cs)
{
        // TODO: Modify the Window class or styles here by modifying
        //  the CREATESTRUCT cs
        return CView::PreCreateWindow(cs);
}
////////////////////////////////////////////////////////////////////////
// CLFCView drawing
void CLFCView::OnDraw(CDC* pDC)
{
        CLFCDoc* pDoc = GetDocument();
        ASSERT_VALID(pDoc);
        // TODO: add draw code for native data here
        pDC->SetMapMode(MM_LOMETRIC); //ENGLISH);
        pDC->MoveTo(100,-1000);
        pDC->LineTo(2500,-1000);
        pDC->TextOut(2450,-1100,"Time (s)");
        // Draws grid lines.
        char s[25];
        for(int time = 10;time <= 120;time = time + 10)  // x axis.
        {
            wsprintf(s,"%d",time);
            pDC->TextOut(70 + (time*200)/10,-1010,s);  // time
            pDC->MoveTo(100 + (time*200)/10,-1000);  // Lines.
            pDC->LineTo(100 + (time*200)/10,-980);
        }
        for(int volt = -10;volt <= 10;volt = volt + 2)  // y axis.
        {
            wsprintf(s,"%d",volt);
            pDC->TextOut(10,30 - 1000 + (volt*800)/10,s);  // Voltage
            pDC->MoveTo(100,-1000 + (volt*800)/10);  // Lines.
            pDC->LineTo(120,-1000 + (volt*800)/10);
        }
        pDC->MoveTo(100,-100);
        pDC->LineTo(100,-1900);
        pDC->TextOut(80,-100,"Freq (V)");
    for(int index = 0; index < 599;index++)
        {
        CPen pen(PS_SOLID,1,RGB(255,0,0));
        CPen* oldPen = pDC->SelectObject(&pen);
        pDC->MoveTo(xposition[index] + 100,yposition[index] - 1000);
        pDC->LineTo(xposition[index+1] + 100,yposition[index+1] - 1000);
        pDC->SelectObject(oldPen);

        CPen penc(PS_DASH,1,RGB(0,0,255));
        CPen* oldPenc = pDC->SelectObject(&penc);
        pDC->MoveTo(xposition[index] + 100,cposition[index] - 1000);
        pDC->LineTo(xposition[index+1] + 100,cposition[index+1] - 1000);
        pDC->SelectObject(oldPenc);
```

```
        CPen penset(PS_DOT,1,RGB(0,255,0));
        CPen* oldPenset = pDC->SelectObject(&penset);
        pDC->MoveTo(xposition[index] + 100,setposition[index] - 1000);
        pDC->LineTo(xposition[index+1] + 100,setposition[index+1] - 1000);
        pDC->SelectObject(oldPenset);
    }
if(who == 1)
 {
    wsprintf(s,": Fuzzy Control");
    pDC->TextOut(800,-15,s);
 }
else
 if(who == 2)
 {
    wsprintf(s,": Hinf Control");
    pDC->TextOut(800,-15,s);
 }
else
if(who == 3)
 {
   wsprintf(s,": QFT Control");
   pDC->TextOut(800,-15,s);
 }
else
 if(who == 4)
 {
     wsprintf(s,": PI Control");
     pDC->TextOut(800,-15,s);
 }
       else
        {
        wsprintf(s,"              ");
        pDC->TextOut(500,-155,s);
        }
 if(threadController == 1)
 {
     wsprintf(s,"LFC ON ");
     pDC->TextOut(500,-15,s);
 }
else
 {
     wsprintf(s,"LFC OFF ");
     pDC->TextOut(500,-15,s);
     wsprintf(s,"                              ");
     pDC->TextOut(500,-155,s);
     who = 0;
if(threadController == 2)
 {
    wsprintf(s,"Step Test");
    pDC->TextOut(500,-55,s);
 }
if(threadController == 3)
 {
    wsprintf(s,"Load Modelling");
    pDC->TextOut(500,-55,s);
 }
 }
 }
////////////////////////////////////////////////////////////////////////
// CLFCView diagnostics
```

```
#ifdef _DEBUG
void CLFCView::AssertValid() const
{
  CView::AssertValid();
}
void CLFCView::Dump(CDumpContext& dc) const
{
        CView::Dump(dc);
}
CLFCDoc* CLFCView::GetDocument() // non-debug version is inline
{
        ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CLFCDoc)));
        return (CLFCDoc*)m_pDocument;
}
#endif //_DEBUG
/////////////////////////////////////////////////////////////////////////////
// CLFCView message handlers
UINT ControlProc_Fuzzy(LPVOID param)
{
  CLFCView *theView = (CLFCView*)param;
  NT_DT2801 theApp;
  T_Sample wait_sample_time;
  FILE *stream;
  FILE *streamx;
  stream = fopen("FData.txt","w");
  streamx = fopen("FDatax.txt","w");
  double frequency = 0;
  double tacho_speed = 0;
  double power = 0;
  double t = 0;
  int tacho_speed_port = 0;
  int frequency_port = 3;
  int power_port = 2;
  double R = 2.4;  //Regulation.
  int Control_input_port = 0;
  double Drooped_speed = 0;
  double control = 1;
  double error = 0;
  double xI = -15.16;
  double xI_1 = -15.16;
  double con = 0;
  double delta = 0;
  double erp = 0;
  double error1 = 0;
  double t0 = 0;
  double tc = 10;
  int stepped = 0;
  double setpoint = 2.5;
  Fuzzy_Controller Fuzzy;
 while(threadController)
  {
   wait_sample_time.Sample_Time();
  // Reading ADC.
  frequency = theApp.ADC(frequency_port);
   tacho_speed = theApp.ADC(tacho_speed_port);
  power = theApp.ADC(power_port);
  // Adding system droop.
  // Drooped_speed  = (0.8 - tacho_speed)/R;
```

```
  // Calculating control.
 //    error = 2 - frequency;
 if(t >= 30)
  {
     setpoint = 3.5;
  }
if(t >= 90)
{
   setpoint = 2.5;
}
error = setpoint - tacho_speed;
// error = 2 - tacho_speed;
delta = (error - erp)*0.6;
erp = error;
error1 = error*0.035;
// Fuzzy Controller.
con = Fuzzy.FuzzyController(error1,delta);
xI = con + xI_1;
control = 0.3*xI_1;
xI_1 = xI;
theView->Plot(t,tacho_speed,control,setpoint);
theApp.DAC(0,control);  // output control.
//theApp.DAC(1,-control);
t = t + wait_sample_time.Ts();
fprintf(stream,"%f %f %f %f %f %f \n",t,tacho_speed,frequency,power,control,error);
fprintf(streamx,"%f %f %f\n",t,xI_1,xI);
who = 1;
//theView->Invalidate();
}
 fclose(stream);
 fclose(streamx);
 theApp.~NT_DT2801();
 return 0;
}
UINT ControlProc_Hinf(LPVOID param)
{
 CLFCView *theView = (CLFCView*)param;
 NT_DT2801 theApp;
  T_Sample wait_sample_time;
  FILE *stream;
  FILE *streamx;
  stream = fopen("HData.txt","w");
  streamx = fopen("HDatax.txt","w");
  double frequency = 0;
  double tacho_speed = 0;
  double power = 0;
  double t = 0;
  int tacho_speed_port = 0;
  int frequency_port = 3;
  int power_port = 2;
 double R = 2.4;  //Regulation.
 int Control_input_port = 0;
 double Drooped_speed = 0;
 double control = 1;
 double error = 0;
 double setpoint = 2.5;
 double xn_7 = -696;
 double xn_6 = -696;
```

```
  double xn_5 = -696;
  double xn_4 = -696;
  double xn_3 = -696;
  double xn_2 = -696;
  double xn_1 = -696;
  double x = -696;
 while(threadController)
{
 wait_sample_time.Sample_Time();
// Reading ADC.
 frequency = theApp.ADC(frequency_port);
 tacho_speed = theApp.ADC(tacho_speed_port);
 power = theApp.ADC(power_port);
// Adding system droop.
// Drooped_speed  = (0.8 - tacho_speed)/R;
// Calculating control.
//error = 2 - frequency;
if(t >= 30)
  {
     setpoint = 3.5;
    }
if(t >= 90)
  {
     setpoint = 2.5;
  }
error = setpoint - tacho_speed;
// Hinf controller design 1.

//  x = error + 1.0604*xn_1 + 0.0287*xn_2 - 0.2930*xn_3 + 0.4771*xn_4 - 0.3667*xn_5 +
0.0935*xn_6;
//  control = 0.1168*x + 0.0097*xn_1 - 0.0716*xn_2 + 0.0507*xn_3 - 0.0845*xn_4 -
0.0523*xn_5 + 0.0475*xn_6;
// Hinf controller design 2.
x =  error + 1.4560*xn_1 - 0.1634*xn_2 - 0.4984*xn_3 + 0.4432*xn_4 - 0.4166*xn_5 +
0.2139*xn_6 - 0.0347*xn_7;
control = 0.0643*x - 0.0089*xn_1 - 0.0406*xn_2 + 0.0367*xn_3 - 0.0527*xn_4 - 0.0185*xn_5
+ 0.0325*xn_6 - 0.0058*xn_7;
xn_7 = xn_6;
xn_6 = xn_5;
xn_5 = xn_4;
xn_4 = xn_3;
xn_3 = xn_2;
xn_2 = xn_1;
xn_1 = x;
theApp.DAC(0,control);  // output control.
theView->Plot(t,tacho_speed,control,setpoint);
t = t + wait_sample_time.Ts();
fprintf(stream,"%f %f %f %f %f %f\n",t,tacho_speed,frequency,power,control,error);
fprintf(streamx,"%f %f %f %f %f %f %f %f\n",t,x,xn_1,xn_2,xn_3,xn_4,xn_5,xn_6,xn_7);
who = 2;
//theView->Invalidate();
 }
fclose(stream);
fclose(streamx);
theApp.~NT_DT2801();
return 0;
}
UINT ControlProc_QFT(LPVOID param)
{
  CLFCView *theView = (CLFCView*)param;
```

```
NT_DT2801 theApp;
T_Sample wait_sample_time;
FILE *stream;
FILE *streamx;
stream = fopen("QData.txt","w");
streamx = fopen("QDatax.txt","w");
double frequency = 0;
double tacho_speed = 0;
double power = 0;
double t = 0;
int tacho_speed_port = 0;
int frequency_port = 3;
int power_port = 2;
double R = 2.4;  //Regulation.
int Control_input_port = 0;
 double Drooped_speed = 0;
 double control = 1;
 double error = 0;
 double setpoint = 2.5;
//double xn_7 = -1100;
//double xn_6 = -1100;
//double xn_5 = -1100;
//double xn_4 = -1100;
//double xn_3 = -1100;
//double xn_2 = -1100;
//double xn_1 = -1100;
//double x = -1100;
double step = 2.5;
double xF = 65.352968;
double xFn_1 = 65.352968;
 //double xFn_2 = 65;
 // QFT design based on tracking specifications.
//        double A[5][5] = {  {2.8729, -2.7027, 0.6022, 0.3914, -0.1638},
//                            {1.0000,     0,     0,     0,     0},
//                            {    0, 1.0000,     0,     0,     0},
//                            {    0,     0, 1.0000,     0,     0},
//                            {    0,     0,     0, 1.0000,     0}};

//        double B[5][1] = { { 1 },
//                           { 0 },
//                           { 0 },
//                  { 0 },
//                  { 0 }};
//
//        double C[1][5] = {  {0.0122,   0.1485,   -0.4441,   0.4057,   -0.1218}};
//        double D = 0.1553;
// QFT design for disturbance rejection.
double   A[5][5]   =   {{3.49871011577202,   -4.64788348703597,   2.81891790225276,   -
0.70062488839118, 0.03088035740237},
         {1.00000000000000,          0,          0,          0,          0},
         {         0, 1.00000000000000,          0,          0,          0},
         {         0,          0, 1.00000000000000,          0,          0},
         {         0,          0,          0, 1.00000000000000,          0}};

double B[5][1]  = {{  1 },
                   {  0 },
                   {  0 },
                   {  0 },
                   {  0 }};
```

```
double C[1][5] ={ { 0.01728476210795, -0.02912510453265, 0.01478264225133, -
0.00427861435167, 0.00158507289009}};
double D = 0.00908825601860;
//double x[5] = { -9220.208336, -9219.969509, -9219.726427, -9219.477683, -9219.221853};
// QFT 1
double x[5] = { -18620.702065, -18621.139941, -18621.567605, -18621.991796, -
18622.416593}; // QFT 2;
 double temp = 0;
  double u = 1;
  double y = 0;
  double state = 0 ;
  double xtemp[5];
// QFT 1;
//            double xF1 = 166.965076;
//            double xF2 = 166.965082;
//            double xF1next = 166.965076;
//            double xF2next = 166.965082;
// QFT 2;
double xF1 = 258.585035;
double xF2 = 258.585035;
double xF1next = 258.585035;
double xF2next = 258.585035;
  while(threadController)
{
 wait_sample_time.Sample_Time();
// Reading ADC.
frequency = theApp.ADC(frequency_port);
tacho_speed = theApp.ADC(tacho_speed_port);
power = theApp.ADC(power_port);
// Adding system droop.
 //Drooped_speed  = (0.8 - tacho_speed)/R;
// Calculating control.
//error = 2 - frequency;
if(t >= 90)
{
 step = 3.5;
}
if(t >= 210)
{
step = 2.5;
}
// Prefilter.
// xF = step + 0.8476*xFn_1;
// setpoint = 1.7698*xF - 1.6174*xFn_1;
//xF = step - 0.1329*xFn_1 + 0.8547*xFn_2;
//setpoint = 0.255*xF + 0.1391*xFn_1 - 0.1159*xFn_2;
//xF = step + 0.9302*xFn_1;
//setpoint = 0.3304*xF - 0.2606*xFn_1;
//xF = step + 0.9302*xFn_1;
//setpoint = 0.3317*xF - 0.2619*xFn_1;
// xF = step + 0.9617*xFn_1;
// setpoint = 0.01913*xF + 0.01913*xFn_1;
//xF = step + 0.9617*xFn_1;
//setpoint = 0.01913*xF + 0.01913*xFn_1;
//xFn_2 = xFn_1;
//xFn_1 = xF;
//state = 0.03752236007948*state + 0.01912702305051*step;
//            setpoint = 0.96174595389897*state + step;
```

```
  //   xF = step + 0.96174595389897*xFn_1;
  //   setpoint = 0.01912702305051*xF + 0.01912702305051*xFn_1;
  //   xFn_1 = xF;
  // QFT prefilter for tracking design.
 // setpoint = 0.02782879605870*xF1 - 0.01304716882353*xF2 + 0.01279455255607*step;
 //  xF1next = 1.58991059966890*xF1 - 0.60488380233248*xF2 + 1*step;
 //  xF2next = xF1;
 //       xF1 = xF1next;
//       xF2 = xF2next;
// QFTT prefilter for disturbance design.
setpoint =  0.02278945983691*xF1 - 0.01322396350591*xF2 + 0.01060231806954*step;
xF1next = 1.69354097382698*xF1 - 0.70320897336417*xF2 + step;
xF2next = 1.00000000000000*xF1 + 0*xF2 + 0*step;
xF1 = xF1next;
xF2 = xF2next;
 u = setpoint - tacho_speed; /*setpoint*/
 // Calculates the current control vector.
for(int j = 0; j<5; j++)
{
 temp = temp + C[0][j]*x[j];
}
y = temp + D*u;
temp = 0;
// Calculates the next state vector.
for(int i = 0; i<5; i++)
{
for(int j = 0; j<5; j++)
{
  temp = temp + A[i][j]*x[j];
}
                xtemp[i] = temp + B[i][0]*u;
                temp = 0;
        }
        for(i = 0;i<5;i++)
        {
                x[i] = xtemp[i];
        }
  // QFT Controller.
// x = error + 0.1690*xn_1 + 1.7068*xn_2 + 0.2159*xn_3 - 1.2222*xn_4 - 0.0384*xn_5 +
0.2067*xn_6 - 0.0378*xn_7;
// control = 0.0620*x - 0.0024*xn_1 - 0.0354*xn_2 + 0.0243*xn_3 - 0.0578*xn_4 -
0.0137*xn_5 + 0.0332*xn_6 - 0.0061*xn_7;

 //x = error + 2.078*xn_1 - 1.231*xn_2 + 0.1531*xn_3;
 //control = 0.4069*x - 0.349*xn_1 - 0.4053*xn_2 + 0.3505*xn_3;
 //x = error + 1.401*xn_1 - 0.05921*xn_2 - 0.4058*xn_3 + 0.06667*xn_4 - 0.002466*xn_5;
 //control = 0.3688*x - 0.3476*xn_1 + 0.01954*xn_2 + 0.09967*xn_3 - 0.3824*xn_4 +
0.2538*xn_5;
 //x = error + 2.123*xn_1 - 1.437*xn_2 + 0.3404*xn_3 - 0.02617*xn_4 + 4.754e-005*xn_5;
 //control = 0.404*xn_1 - 0.536*xn_2 + 0.3398*xn_3 - 0.455*xn_4 + 0.2557*xn_5;
 // x = error + 2.873*xn_1 - 2.703*xn_2 + 0.6022*xn_3 + 0.3914*xn_4 - 0.1638*xn_5;
 // control = 0.1553*x - 0.434*xn_1 + 0.5682*xn_2 - 0.5376*xn_3 + 0.3449*xn_4 -
0.09635*xn_5;
// x = error + 2.406*xn_1 - 1.901*xn_2 + 0.5462*xn_3 - 0.05155*xn_4;
// control = 0.1177*x - 0.01564*xn_1 - 0.1941*xn_2 + 0.01623*xn_3 + 0.07698*xn_4;
//x = error + 3.492*xn_1 - 4.716*xn_2 + 3.058*xn_3 - 0.9415*xn_4 + 0.1078*xn_5;
//control = 0.002212*x - 0.001929*xn_1 + 0.0001043*xn_2 + 0.000517*xn_3 - 0.002266*xn_4
+ 0.001462*xn_5;
//   xn_7 = xn_6;
//   xn_6 = xn_5;
```

```
//              xn_5 = xn_4;
//              xn_4 = xn_3;
//              xn_3 = xn_2;
//              xn_2 = xn_1;
//              xn_1 = x;
//             x = error + xn_1;
//             control = 0.5112*x - 0.4887*xn_1;
//    xn_1 = x;
//control = (control + Drooped_speed);
theApp.DAC(0,y);  // output control.
theView->Plot(t,tacho_speed,y,step);
t = t + wait_sample_time.Ts();
fprintf(stream,"%f %f %f %f %f %f %f\n",t,tacho_speed,frequency,power,y,step,error);
fprintf(streamx,"%f %f %f %f %f %f %f %f %f\n",t,x[0],x[1],x[2],x[3],x[4],xF1,xF2);
who = 3;
// theView->Invalidate();
}
fclose(stream);
fclose(streamx);
theApp.~NT_DT2801();
return 0;
}
UINT ControlProc_PI(LPVOID param)
{
 CLFCView *theView = (CLFCView*)param;
 NT_DT2801 theApp;
 T_Sample wait_sample_time;
FILE *stream;
FILE *streamx;
stream = fopen("PIData.txt","w");
streamx = fopen("PIDatax.txt","w");
double frequency = 0;
 double tacho_speed = 0;
 double power = 0;
 double t = 0;
 int tacho_speed_port = 0;
 int frequency_port = 3;
 int power_port = 2;
 double R = 2.4;  //Regulation.
int Control_input_port = 0;
double Drooped_speed = 0;
double control = 1;
double error = 0;
double setpoint = 2.5;
double xn_1 = -211;
 double x = -211;
while(threadController)
{
wait_sample_time.Sample_Time();
// Reading ADC.
 frequency = theApp.ADC(frequency_port);
 tacho_speed = theApp.ADC(tacho_speed_port);
 power = theApp.ADC(power_port);
// Adding system droop.
//              Drooped_speed  = (0.8 - tacho_speed)/R;
// Calculating control.
//error = 2 - frequency;
if(t >= 60)
{
setpoint = 3.5;
```

```
                    }
                    if(t >= 120)
                    {
                    setpoint = 2.5;
                    }
            error = setpoint - tacho_speed;
  // PI control.
 x = error + xn_1;
control = 0.5112*x - 0.4887*xn_1;
xn_1 = x;
theApp.DAC(0,control);  // output control.
theView->Plot(t,tacho_speed,control,setpoint);
t = t + wait_sample_time.Ts();
fprintf(stream,"%f %f %f %f %f %f\n",t,tacho_speed,frequency,power,control,error);
fprintf(streamx,"%f %f\n",t,x,xn_1);
who = 4;
//theView->Invalidate();
}
  fclose(stream);
   fclose(streamx);
   theApp.~NT_DT2801();
   return 0;
}
UINT ControlProc_StepTest(LPVOID param)
{
 AfxMessageBox("Step Test Begun");
 CLFCView *theView = (CLFCView*)param;
 NT_DT2801 theApp;
 T_Sample wait_sample_time;
 FILE *stream;
 stream = fopen("StepTest.txt","w");
 double frequency = 0;
 double tacho_speed = 0;
 double power = 0;
 double t = 0;
 int tacho_speed_port = 0;
 int frequency_port = 3;
 int power_port = 2;
 double R = 2.4;  //Regulation.
 int Control_input_port = 0;
 double Drooped_speed = 0;
 double control = -6;
while(threadController == 2)
{
 wait_sample_time.Sample_Time();
// Reading ADC.
 frequency = theApp.ADC(frequency_port);
 tacho_speed = theApp.ADC(tacho_speed_port);
 power = theApp.ADC(power_port);
// Adding system droop.
//Drooped_speed  = (1.4 - tacho_speed)/R;
// Calculating control.
 if(t >= 30)
 {
 control = -5;
 }
//control = control + Drooped_speed;
theApp.DAC(0,control);  // output control.
theView->Plot(t,tacho_speed,control,control);
t = t + wait_sample_time.Ts();
```

```
 fprintf(stream,"%f %f %f %f %f \n",t,tacho_speed,frequency,power,control);
 //theView->Invalidate();
  }
fclose(stream);
return 0;
}
UINT ControlProc_LoadModelling(LPVOID param)
{
  AfxMessageBox("Load Modelling Begun");
  CLFCView *theView = (CLFCView*)param;
   NT_DT2801 theApp;
   T_Sample wait_sample_time;
   FILE *stream;
   stream = fopen("LoadTest.txt","w");
   double frequency = 0;
   double tacho_speed = 0;
   double power = 0;
   double t = 0;
int tacho_speed_port = 0;
int frequency_port = 3;
int power_port = 2;
double R = 2.4;  //Regulation.
int Control_input_port = 0;
double Drooped_speed = 0;
double control = 3.5;
while(threadController == 3)
{
 wait_sample_time.Sample_Time();
// Reading ADC.
 frequency = theApp.ADC(frequency_port);
   tacho_speed = theApp.ADC(tacho_speed_port);
   power = theApp.ADC(power_port);
   // Adding system droop.
  //Drooped_speed  = (1.4 - tacho_speed)/R;
  //control = control + Drooped_speed;
  theApp.DAC(0,control);  // output control.
  t = t + wait_sample_time.Ts();
  fprintf(stream,"%f %f %f %f %f \n",t,tacho_speed,frequency,power,control);
 //theView->Invalidate();
  }
 fclose(stream);
 return 0;
}
void CLFCView::OnStopLFC()
{
        Stop_control = 0;
        threadController = 0;
   who = 0;
}
void CLFCView::OnFuzzyControl()
{
        // TODO: Add your command handler code here
        threadController = 1;
   //HWND hWnd = GetSafeHwnd();
   AfxBeginThread(ControlProc_Fuzzy,this,THREAD_PRIORITY_NORMAL);
}
void CLFCView::OnHInfinityControl()
{
        // TODO: Add your command handler code here
```

```
   threadController = 1;
  // HWND hWnd = GetSafeHwnd();
   AfxBeginThread(ControlProc_Hinf,this,THREAD_PRIORITY_NORMAL);
}
void CLFCView::OnQFTControl()
{
            // TODO: Add your command handler code here
 threadController = 1;
//HWND hWnd = GetSafeHwnd();
AfxBeginThread(ControlProc_QFT,this,THREAD_PRIORITY_NORMAL);
}
void CLFCView::OnPI()
{
// TODO: Add your command handler code here
threadController = 1;
AfxBeginThread(ControlProc_PI,this,THREAD_PRIORITY_NORMAL);
}
void CLFCView::OnStepTest()
{
// TODO: Add your command handler code here
threadController = 2;
AfxBeginThread(ControlProc_StepTest,this,THREAD_PRIORITY_IDLE);
}
void CLFCView::OnStopModelling()
{
            // TODO: Add your command handler code here
            threadController = 0;
}
void CLFCView::OnLoadModelling()
{
            // TODO: Add your command handler code here
            threadController = 3;
   AfxBeginThread(ControlProc_LoadModelling,this,THREAD_PRIORITY_IDLE);
}
void CLFCView::Plot(double x, double y,double c,double set)
{
 UINT tempx[600];   //temporary time vector.
 UINT tempy[600];   // output.
 UINT tempc[600];   // control.
 UINT tempset[600]; // setpoint.
for(int index = 0; index < 599;index++)
{
tempx[index] = xposition[index+1];
tempy[index] = yposition[index+1];
tempc[index] = cposition[index+1];
tempset[index] = setposition[index+1];
}
for(index = 0; index < 599;index++)
{
xposition[index] = tempx[index];
yposition[index] = tempy[index];
cposition[index] = tempc[index];
setposition[index] = tempset[index];
}
xposition[599] = UINT((ttemp*200)/10); //100mm = 10s.
yposition[599] = UINT((y*800)/10) ;  // 800mm = 10V.
cposition[599] = UINT((c*800)/10) ;  // 800mm = 10V.
setposition[599] = UINT((set*800)/10);
ttemp = ttemp + (x - x0);
x0 = x;
```

```
if(ttemp > 120)
{
ttemp = 0;
for(index = 0;index <= 599;index++)
{
   xposition[index] = 0;
  yposition[index] = 0;
  cposition[index] = 0;
 setposition[index] = 0;
}
Invalidate(TRUE);      } Invalidate(FALSE);}
}
```

```
#include "stdafx.h"
#include "Fuzzy_Controller.h"
#include <iostream.h>
#include <math.h>
#include <stdlib.h>

Fuzzy_Controller::Fuzzy_Controller()
{
        mu_NB = 0;
        NBvalue = 0;
        mu_NS = 0;
        NSvalue = 0;
        mu_ZZ = 0;
        ZZvalue = 0;
        mu_PS = 0;
        PSvalue = 0;
        mu_PB = 0;
        PBvalue = 0;
        control = 0;
};

double Fuzzy_Controller::NBmf(double x)
{
   double a1 = -10;
   double a2 = -5;
   double a3 = 0;
   double a4 = 5;
   double a5 = 10;
   double vect[2] = {0,0};
   if(x <= a2)
   {
     mu_NB = (a2 - x)/(a2 - a1);
     if(x < a1)
          {
             mu_NB = 1;
          }
     NBvalue = 1;
   }
   else
   {
     mu_NB = 0;
     NBvalue = 0;
   }
  return mu_NB;
}

int Fuzzy_Controller::NB()
{
 return NBvalue;
}

double Fuzzy_Controller::NSmf(double x)
{
  double a1 = -10;
  double a2 = -5;
  double a3 = 0;
  double a4 = 5;
  double a5 = 10;
```

```
  if((x >= a1) && (x <= a2))
  {
    mu_NS = (x-a1)/(a2 - a1);
    NSvalue = 1;
  }
  else
  {
    mu_NS = 0;
    NSvalue = 0;
  }
  if((x >= a2) && (x <= a3))
  {
    mu_NS = (a3-x)/(a3-a2);
    NSvalue = 1;
  }
 else
  {
    mu_NS = mu_NS;
    NSvalue = NSvalue;
  }
  return mu_NS;
}

int Fuzzy_Controller::NS()
{
            return NSvalue;
}

double Fuzzy_Controller::ZZmf(double x)
{
  double a1 = -10;
  double a2 = -5;
  double a3 = 0;
  double a4 = 5;
  double a5 = 10;

 if((x >= a2) && (x <= a3))
 {
   mu_ZZ = (x-a2)/(a3 - a2);
   ZZvalue = 1;
 }
 else
 {
   mu_ZZ = 0;
   ZZvalue = 0;
 }
 if((x >= a3) && (x <= a4))
 {
   mu_ZZ = (a4-x)/(a4-a3);
    ZZvalue = 1;
 }
 else
 {
   mu_ZZ = mu_ZZ;
   ZZvalue = ZZvalue;
 }
return mu_ZZ;
}
```

```
int Fuzzy_Controller::ZZ()
{
        return ZZvalue;
}

double Fuzzy_Controller::PSmf(double x)
{
  double a1 = -10;
  double a2 = -5;
  double a3 = 0;
  double a4 = 5;
  double a5 = 10;
  if((x >= a3) && (x <= a4))
  {
    mu_PS = (x-a3)/(a4 - a3);
    PSvalue = 1;
  }
  else
  {
    mu_PS = 0;
    PSvalue = 0;
  }
  if((x >= a4) && (x <= a5))
  {
    mu_PS = (a5-x)/(a5-a4);
    PSvalue = 1;
  }
  else
  {
 mu_PS = mu_PS;
    PSvalue = PSvalue;
  }
  return mu_PS;
}
int Fuzzy_Controller::PS()
{
        return PSvalue;
}

double Fuzzy_Controller::PBmf(double x)
{
   double a1 = -10;
    double a2 = -5;
    double a3 = 0;
    double a4 = 5;
    double a5 = 10;
  if((x >= a4) && (x <= a5))
  {
    mu_PB = (x-a4)/(a5-a4);
    PBvalue = 1;
  }
  else
  {
    mu_PB = 0;
    PBvalue = 0;
  }
  if(x >= a5)
  {
    mu_PB = 1;
    PBvalue = 1;
```

```
 }
 else
 {
   mu_PB = mu_PB;
   PBvalue = PBvalue;
 }
return mu_PB;
}

int Fuzzy_Controller::PB()
{
        return PBvalue;
}
double Fuzzy_Controller::Inferencing(double mu_NBE,int NBE, double mu_NSE, int NSE,
double mu_ZZE, int ZZE,double mu_PSE,int PSE,double mu_PBE,int PBE,double mu_NBD,int
NBD, double mu_NSD, int NSD, double mu_ZZD, int ZZD,double mu_PSD,int PSD,double
mu_PBD,int PBD)
{
/*
[NBE,mu_NBE] = NBmf(error);
[NSE,mu_NSE] = NSmf(errr);
[ZZE,mu_ZZE] = ZZeromf(error);
[PSE,mu_PSE] = PSmf(error);
[PBE,mu_PBE] = PBmf(error);

%Delta.
[NBD,mu_NBD] = NBmf(delta);
[NSD,mu_NSD] = NSmf(delta);
[ZZD,mu_ZZD] = ZZeromf(delta);
[PSD,mu_PSD] = PSmf(delta);
[PBD,mu_PBD] = PBmf(delta);
*/

   double NB1 = 0;
   double NB2 = 0;
   double NB3 = 0;
   double NB4 = 0;
   double NB5 = 0;
   double NB6 = 0;
   double NS1 = 0;
   double NS2 = 0;
   double NS3 = 0;
   double NS4 = 0;
   double ZZ1 = 0;
   double ZZ2 = 0;
   double ZZ3 = 0;
   double ZZ4 = 0;
  double ZZ5 = 0;
   double PS1 = 0;
   double PS2 = 0;
   double PS3 = 0;
   double PS4 = 0;
   double PB1 = 0;
   double PB2 = 0;
   double PB3 = 0;
   double PB4 = 0;
   double PB5 = 0;
   double PB6 = 0;

//Inferencing rules.
```

```
if((NBE == 1) && (NBD == 1))       // Rule 1.
{
  NB1 = __min(mu_NBE,mu_NBD);
}
else
{
  NB1 = 0;
}
if((NBE == 1) && (NSD == 1))       // Rule 2.
{
  NB2 = __min(mu_NBE,mu_NSD);
}
else
{
  NB2 = 0;
}
if((NBE == 1) && (ZZD == 1))       //Rule 3.
{
  NB3 = __min(mu_NBE,mu_ZZD);
}
else
{
  NB3 = 0;
}
if((NBE == 1) && (PSD == 1))       // Rule 4.
{
  NS1 = __min(mu_NBE,mu_PSD);
}
else
{
  NS1 = 0;
}
if((NBE == 1) && (PBD == 1))       // Rule 5.
{
  ZZ1 = __min(mu_NBE,mu_PBD);
}
else
{
  ZZ1 = 0;
}
if((NSE == 1) && (NBD == 1))       // Rule 6.
{
  NB4 = __min(mu_NSE,mu_NBD);
}
else
{
  NB4 = 0;
}
if((NSE == 1) && (NSD == 1))       // Rule 7.
{
  NB5 = __min(mu_NSE,mu_NSD);
}
else
{
  NB5 = 0;
}
if((NSE == 1) && (ZZD == 1))       // Rule 8.
{
  NS2 = __min(mu_NSE,mu_ZZD);
}
```

```
else
{
  NS2 = 0;
}
if((NSE == 1) && (PSD == 1))       // Rule 9.
{
  ZZ2 = __min(mu_NSE,mu_PSD);
}
else
{
  ZZ2 = 0;
}
if((NSE == 1) && (PBD == 1))       // Rule 10.
{
  PS1 = __min(mu_NSE,mu_PBD);
}
else
{
  PS1 = 0;
}
if((ZZE == 1) && (NBD == 1))       // Rule 11.
{
  NB6 = __min(mu_ZZE,mu_NBD);
}
else
{
  NB6 = 0;
}
if((ZZE == 1) && (NSD == 1))       // Rule 12.
{
  NS3 = __min(mu_ZZE,mu_NSD);
}
else
{
  NS3 = 0;
}
if((ZZE == 1) && (ZZD == 1))       // Rule 13.
{
  ZZ3 = __min(mu_ZZE,mu_ZZD);
}
else
{
  ZZ3 = 0;
}
if((ZZE == 1) && (PSD == 1))       // Rule 14.
{
  PS2 = __min(mu_ZZE,mu_PSD);
}
else
{
  PS2 = 0;
}
if((ZZE == 1) && (PBD == 1))       // Rule 15.
{
  PB1 = __min(mu_ZZE,mu_PBD);
}
else
{
  PB1 = 0;
}
```

```
if((PSE == 1) && (NBD == 1))       // Rule 16.
{
  NS4 = __min(mu_PSE,mu_NBD);
}
else
{
  NS4 = 0;
}
if((PSE == 1) && (NSD == 1))       // Rule 17.
{
  ZZ4 = __min(mu_PSE,mu_NSD);
}
else
{
  ZZ4 = 0;
}
if((PSE == 1) && (ZZD == 1))       // Rule 18.
{
  PS3 = __min(mu_PSE,mu_ZZD);
}
else
{
  PS3 = 0;
}
if((PSE == 1) && (PSD == 1))       // Rule 19.
{
  PB2 = __min(mu_PSE,mu_PSD);
}
else
{
  PB2 = 0;
}
if((PSE == 1) && (PBD == 1))       // Rule 20.
{
  PB3 = __min(mu_PSE,mu_PBD);
}
else
{
  PB3 = 0;
}
if((PBE == 1) && (NBD == 1))       // Rule 21.
{
  ZZ5 = __min(mu_PBE,mu_NBD);
}
else
{
  ZZ5 = 0;
}
if((PBE == 1) && (NSD == 1))       // Rule 22.
{
  PS4 = __min(mu_PBE,mu_NSD);
}
else
{
  PS4 = 0;
}
if((PBE == 1) && (ZZD == 1))       // Rule 23.
{
  PB4 = __min(mu_PBE,mu_ZZD);
}
```

```
else
{
  PB4 = 0;
}
if((PBE == 1) && (PSD == 1))       // Rule 24.
{
  PB5 = __min(mu_PBE,mu_PSD);
}
else
{
  PB5 = 0;
}
if((PBE == 1) && (PBD == 1))       // Rule 25.
{
  PB6 = __min(mu_PBE,mu_PBD);
}
else
{
  PB6 = 0;
}
// Defuzzification.
double NB = 0;
double NS = 0;
double ZZ = 0;
double PS = 0;
double PB = 0;
NB = __max(__max(__max(NB1,NB2),__max(NB3,NB4)),__max(NB5,NB6));
NS = __max(__max(NS1,NS2),__max(NS3,NS4));
ZZ = __max(__max(__max(ZZ1,ZZ2),__max(ZZ3,ZZ4)),ZZ5);
PS = __max(__max(PS1,PS2),__max(PS3,PS4));
PB = __max(__max(__max(PB1,PB2),__max(PB3,PB4)),__max(PB5,PB6));

double mu_NBzr1 = 0, mu_NSzr1 = 0,mu_ZZzr1 = 0,mu_PSzr1 = 0,mu_PBzr1 = 0;
double mu_NBzr2 = 0, mu_NSzr2 = 0,mu_ZZzr2 = 0,mu_PSzr2 = 0,mu_PBzr2 = 0;
double mu_NBzr3 = 0, mu_NSzr3 = 0,mu_ZZzr3 = 0,mu_PSzr3 = 0,mu_PBzr3 = 0;
double mu_NBzr4 = 0, mu_NSzr4 = 0,mu_ZZzr4 = 0,mu_PSzr4 = 0,mu_PBzr4 = 0;
 // yr1 = -8;
  // yr1 = -10;
    mu_NBzr1 = __min(NB,0.6);
    mu_NSzr1 = __min(NS,0.4);
    mu_ZZzr1 = __min(ZZ,0);
    mu_PSzr1 = __min(PS,0);
    mu_PBzr1 = __min(PB,0);
   // yr2 = -3;
    mu_NBzr2 = __min(NB,0);
    mu_NSzr2 = __min(NS,0.6);
    mu_ZZzr2 = __min(ZZ,0.4);
    mu_PSzr2 = __min(PS,0);
    mu_PBzr2 = __min(PB,0);
    // yr3 = 5;
    mu_NBzr3 = __min(NB,0);
    mu_NSzr3 = __min(NS,0);
    mu_ZZzr3 = __min(ZZ,0.4);
    mu_PSzr3 = __min(PS,0.6);
    mu_PBzr3 = __min(PB,0);
    // yr4 = 8;
    // yr4 = 10
    mu_NBzr4 = __min(NB,0);
    mu_NSzr4 = __min(NS,0);
    mu_ZZzr4 = __min(ZZ,0);
```

```
   mu_PSzr4 = __min(PS,0.4);
   mu_PBzr4 = __min(PB,0.6);
  double mu_yr1 = 0;
   double mu_yr2 = 0;
   double mu_yr3 = 0;
   double mu_yr4 = 0;

mu_yr1                                                              =
__max(__max(__max(mu_NBzr1,mu_NSzr1),__max(mu_ZZzr1,mu_PSzr1)),mu_PBzr1);
            mu_yr2                                                  =
__max(__max(__max(mu_NBzr2,mu_NSzr2),__max(mu_ZZzr2,mu_PSzr2)),mu_PBzr2);
            mu_yr3                                                  =
_max(__max(__max(mu_NBzr3,mu_NSzr3),__max(mu_ZZzr3,mu_PSzr3)),mu_PBzr3);
   mu_yr4                                                           =
__max(__max(__max(mu_NBzr4,mu_NSzr4),__max(mu_ZZzr4,mu_PSzr4)),mu_PBzr4);

double C = 0;
C = (mu_yr1*(-8) + mu_yr2*(-3) + mu_yr3*(3) + mu_yr4*(8))/(mu_yr1 + mu_yr2 + mu_yr3
+mu_yr4);
return C;
}
double Fuzzy_Controller::FuzzyController(double error, double delta)
{
Fuzzy_Controller theApp;
double mu_NBE = 0;
double mu_NSE = 0;
double mu_ZZE = 0;
double mu_PSE = 0;
double mu_PBE = 0;
int NBE = 0;
int NSE = 0;
int ZZE = 0;
int PSE = 0;
int PBE = 0;
double mu_NBD = 0;
double mu_NSD = 0;
double mu_ZZD = 0;
double mu_PSD = 0;
double mu_PBD = 0;
int NBD = 0;
int NSD = 0;
int ZZD = 0;
int PSD = 0;
int PBD = 0;
//double error = 10;
//double delta = 10;
double c = 0;
// Fuzzification of Error.
mu_NBE = theApp.NBmf(error);
NBE = theApp.NB();
mu_NSE = theApp.NSmf(error);
NSE = theApp.NS();
mu_ZZE = theApp.ZZmf(error);
ZZE = theApp.ZZ();
mu_PSE = theApp.PSmf(error);
PSE = theApp.PS();
mu_PBE = theApp.PBmf(error);
PBE = theApp.PB();
```

```
// Fuzzification of Delta.
mu_NBD = theApp.NBmf(delta);
NBD = theApp.NB();
mu_NSD = theApp.NSmf(delta);
NSD = theApp.NS();
mu_ZZD = theApp.ZZmf(delta);
ZZD = theApp.ZZ();
mu_PSD = theApp.PSmf(delta);
PSD = theApp.PS();
mu_PBD = theApp.PBmf(delta);
PBD = theApp.PB();
// Inferencing and Defuzzification.

c                                                                    =
theApp.Inferencing(mu_NBE,NBE,mu_NSE,NSE,mu_ZZE,ZZE,mu_PSE,PSE,mu_PBE,PBE,m
u_NBD,NBD, mu_NSD,NSD,mu_ZZD,ZZD,mu_PSD,PSD,mu_PBD,PBD);

return c;
}
```

```
class Fuzzy_Controller
{
public:
Fuzzy_Controller();  //constructor.
// Membership functions for fuzzification.
double NBmf(double x);
int NB();
double NSmf(double x);
int NS();
double ZZmf(double x);
int ZZ();
double PSmf(double x);
int PS();
double PBmf(double x);
int PB();
// Inferencing Rules.
double Inferencing(double mu_NBE,int NBE, double mu_NSE, int NSE, double mu_ZZE, int ZZE,double mu_PSE,int PSE,double mu_PBE,int PBE,double mu_NBD,int NBD, double mu_NSD, int NSD, double mu_ZZD, int ZZD,double mu_PSD,int PSD,double mu_PBD,int PBD);
double FuzzyController(double error, double delta);
private:
double mu_NB;
int NBvalue;
double mu_NS;
int NSvalue;
double mu_ZZ;
int ZZvalue;
double mu_PS;
int PSvalue;
double mu_PB;
int PBvalue;
double control;
};
```