Investigation into Iterative Feedback Control

Martin I. Machaba

A dissertation submitted in a fulfillment of the requirements
for the degree of Master of Science in Electrical Engineering
in the University of Cape Town

Declaration

I declare that this dissertation is my own unaided work. It is being submitted for the degree of Masters of Science in Engineering in the University of Cape Town. It has not been submitted before for any degree or examination in another university.

Signature of Author..... Signed by candidate .....

Cape Town

20 December 2004

Abstract

The Iterative Feedback Tuning (IFT) method is investigated in this dissertation, starting with the history and origin of this algorithm. The objective of this thesis was to apply the IFT algorithm to a physical system and the system chosen was a DC motor configured for speed control. The inertia of the load for the DC motor is varied to ensure that the algorithm will make the necessary adjustments to compensate for changes in load. Since the IFT is closely related to the well known Model Reference Adaptive Control (MRAC) method application that uses the gradient approach to its adaption.

The equations for the MRAC gradient approach method are presented and applied to a DC motor for a speed control. The purpose of this application was to investigate how the MRAC gradient approach will behave in practice in order to give a base case against which to compare the IFT method. The differences between the theoretical and experimental responses of the MRAC are explained by simulation study and modification of the basic MRAC equations.

The one degree of freedom controller was chosen for IFT application to a DC motor system for its simplicity since it requires only two experiments to be carried out on the DC motor instead of three as in the case of the two degree of freedom controller. The appropriate signals were generated in experiment 1 and 2 of the IFT algorithm and the control parameters updated in the third step, the values of the controller parameters for which the quadratic criterion is a minimum were produced. The iterations were repeated for a different load.

Acknowledgements

Firstly I would like to thank God for his guidance and empowering me during the hard times of this dissertation period, my supervisor Prof Martin Braae for all his inputs and efforts, and my friend Anesu for your support.

Contents

List of figures

List of Tables

List of Abbreviations

A/D	Analog to Digital
Amp	Amplifier
BFGS	Broyden-Fletcher-Glodfard-Shanno
CEP	Control Engineering Practice
D/A	Digital to Analog
d.o.f	Degree of freedom
GUI	Graphic User Interface
IFT	Iterative feedback tuning
IMC	Internal Model Control
ISE	Integral square error
LTI	Linear time invariant
MIMO	Multiple inputs multiple outputs
MRAC	Model Reference Adaptive Control
PI	Proportional Integral
PID	Proportional Integral Differential
SISO	Single input single output
SOC	Self Organizing Control
STR	Self- Tuning-Regulator
ZN	Ziegler-Nichols

External Comments	Action taken
1. Given the huge volume of existing literature on adaptive control and iterative design, there is an apparent lack of thorough understanding of the theory through largely a partial inclusion of key papers on the topics. The thesis when dealing with theoretical derivations are not obvious. For example, the expression of (4.7) is typically a time-domain expression, while the derivation of (4.9), (4.10)on page 37, and (4.10) on page 38, (4.11) ect, are all in frequency-domain, I will give a non-exhaustive list of technical errors and inconsistencies at the end of the document.	• The literature review was rewritten from the beginning, a list of papers and journals on both adaptive control(MRAC) and IFT algorithm are listed. • The derivation from the time domain to frequency domain is been presented in the manner in all the IFT literature that were reviewed.
2. The choice of a DC motor to demostrate the IFT ideas is not well justified, is parameter shifting a mojor concern of a DC motor? If it is not, why IFT? If it is , how severely it is and how IFT addresses the problem? Even in the present form, the one-page chapter 7 is supposed to compared simulated results with experimental results. No meaningful conclusions were provided	• The inertia of the DC motor was varied to see the effect of varying load • Chapter 7 was written
3. The literature style and presentation are not satisfactory	• The dissertation is been rearranged to organized the style and the presentation
4. List of errors	• All the random errors have been eliminated • The reference style is fixed to be uniform and consistence

External Comments	Action taken
1. Given the huge volume of existing literature on adaptive control and iterative design, there is an apparent lack of thorough understanding of the theory through largely a partial inclusion of key papers on the topics. The thesis when dealing with theoretical derivations are not obvious. For example, the expression of (4.7) is typically a time-domain expression, while the derivation of (4.9), (4.10)on page 37, and (4.10) on page 38, (4.11) ect, are all in frequency-domain, I will give a non-exhaustive list of technical errors and inconsistencies at the end of the document.	• The literature review was rewritten from the beginning, a list of papers and journals on both adaptive control(MRAC) and IFT algorithm are listed. • The derivation from the time domain to frequency domain is been presented in the manner in all the IFT literature that were reviewed.
2. The choice of a DC motor to demostrate the IFT ideas is not well justified, is parameter shifting a mojor concern of a DC motor? If it is not, why IFT? If it is , how severely it is and how IFT addresses the problem? Even in the present form, the one-page chapter 7 is supposed to compared simulated results with experimental results. No meaningful conclusions were provided	• The inertia of the DC motor was varied to see the effect of varying load • Chapter 7 was written
3. The literature style and presentation are not satisfactory	• The dissertation is been rearranged to organized the style and the presentation
4. List of errors	• All the random errors have been eliminated • The reference style is fixed to be uniform and consistence

Chapter 1

Introduction

The optimization of industrial control systems can significantly improve their economic performance. Traditional methods of controller design, namely process modeling followed by the use of controller design methods require a high level of on-plant engineering expertise. Thus the ability to tune controllers on-line has always been an attractive but difficult problem in control engineering.

As an example the South African mineral extraction industry has for decades designed and implemented sophisticated optimizing control systems on their unit processes and plants- A historic example being that of a milling plant described by Hulbert [20]. In subsequent implementations on other milling plants it was soon realized that once a control law had been designed for a particular plant the structure of its control was known and the design method merely set the parameter values for the controller. For a while however every new application was still tackled using the traditional approach of system identification followed by controller design. This procedure is a costly engineering exercise [21], so considerable research effort was put into determining the viability of using more sophisticated methods, like the well known adaptive control techniques to produce the control laws.

The current project was launched to investigate the feasibility of automatically adjusting control parameters while avoiding the expensive system identification and controller design stages of the traditional approach to process control. Industrial experience in the mineral extraction industry had shown that the work had been successful in practice on

some Single Input Single Output (SISO) systems but had yet to be extended to the Multiple Input Multiple Output (MIMO).

Thus this thesis investigates a number of tuning methods in order to determine their practical characteristics by implementing them on a physical system. Following an extensive literature review, Iterative Feedback Tuning was chosen as the method to investigate. It is a relatively new method in process control and has the potential to become very popular in industry as noted in the recent paper by Lequin O, Gevers M, Mossberg M, Bosmans E and Triest L [5]. The main reason is that one does not need to know the model of the plant in order to design the parameters of the controller for that particular plant.

Objectives of the thesis

The main objective of this thesis was to investigate the Iterative Feedback Tuning (IFT) algorithm by applying it to a physical system. The system chosen was a DC motor since the literature review revealed that the IFT algorithm has not been tested on the DC motor configured for speed feedback. The IFT is closely related to the well known Model Reference Adaptive Control (MRAC) in that both are non-parametric methods that minimize a quadratic cost function that measures how closely the response of the controlled system matches that of a desired closed loop model. To provide a means of evaluating the IFT method against an existing technique the research started by implementing the MRAC on a DC motor. The experimental results were compared to those obtained from digital simulation studies. In the case of IFT, the structure for a feedback control for both one and two degree of freedom controller was investigated and specific equations were derived for a DC motor. Based on this the IFT algorithm was implemented and tested for the one degree of freedom controller while the MRAC was tested for a two degree of freedom controller.

Scope and limitation

The one degree of freedom controller was chosen for implementation of the IFT algorithm on a DC motor because of its simplicity as will be explained later. This method is relatively new (See reference [15], dated 1994) and has been used for a variety of tasks. For example: In 1998 it was used in the tuning of a linear time-invariant MIMO system [13] whilst in 2002 it was used on the tuning of controllers for a two-mass-spring system with friction [1]. It has also been applied to a magnetic suspension system [2], to the optimization of nonlinear systems [3] and the estimation of controller parameter sensitivity function [4].

Historical background

The IFT method was proposed in the mid nineteen nineties see [13, 14, 15, 30] although the idea of it has been there for decades [22, 31] in the sense that it derives from Model Reference Adaptive Control method.

Plan of development

The literature review of the IFT algorithm follows in the next chapter. Chapter 3 investigates and applies the MRAC gradient approach to a DC motor. Chapter 4 deals with the derivations of the IFT equations for a one degree of freedom controller. The IFT algorithm is applied to a DC motor in chapter 5 for a one degree of freedom controller. Chapter 6 shows the simulation for IFT one degree of freedom controller and the comparison between the simulation and the actual experimental results for a one degree of freedom controller is given. In order to verify that the minimum of the criterion is produced by the IFT algorithm, the criterion minimization is investigated further in chapter 7 and the controller parameters for which the criterion is a minimum are determined. The inertia of the load for the DC motor is changed to verify that the algorithm works properly and can track process changes. The conclusions are drawn from the study and future developments are listed in chapter 8. For completeness and ease of

reference the theory behind the IFT method is presented in appendix 2, and includes the derivation of IFT equations for the two degree of freedom controller

Chapter 2

Literature review

2.1 BACKGROUND

Based on the stated problem from the mineral extraction industry the initial aim of the research project was to survey the literature with a view to establishing the status quo in respect of automatic tuning methods for controllers, especially those encountered in an industrial setting. From this understanding the latest, most promising method was to be investigated in depth and to be applied to a laboratory installation in order to quantify the practical aspects of its application, such as algorithmic complexity, real-time operation, and more.

2.2 RESEARCH OBJECTIVE

The websites www.scirus.com and www.google.com were the main search engines used for researching the literature electronically (The former mainly for papers and electronic journals while the latter for useful web pages). The University of Cape Town Library's three departments namely, the loans desk, interlibrary loans and short loans centre were used to search for relevant books and journals.

The search revealed that the adaptive control methods had been known since the 1950s [22]. Recent literature indicated that the Model Reference Adaptive Control (MRAC) method had been superseded by a newer, non-parametric technique for tuning controller parameters – A technique known as *iterative feedback tuning* [8]. The method appears to have originated in the research work of Hjammarsson et Al [15] in 1994. International interest in the new method had been intense as indicated by the number of hits returned

by the two electronic search engines. Table 2.1 shows the number of journals and web pages that were found by the Scirus search engine using various key words as indicated.

Table 2.1: Scirus search results

Key words	No of Journals	No of web pages
"Iterative methods"	5070	22 805
"Iterative feedback methods"	65	280
"Iterative feedback control"	65	317
"Iterative feedback tuning"	22	100

Table 2.2 shows the number of web pages that were found by the Google search engine using the same key words.

Table 2.2: Google search results

Key words	No of web pages
"Iterative methods"	153 000
"Iterative feedback methods"	9
"Iterative feedback control"	24
"Iterative feedback tuning"	410

Table 2.2 shows the number of web pages that were found by using key words of the research topic.

The large number of hits achieved from the key words "Iterative methods" from both www.scirus.com and www.google.com illustrates that the concept of iterative method is been there for a long time. As the key words become specific the hits are narrowed, for example the key words "Iterative feedback tuning" produced only 22 hits on the www.scirus.com website. This implies that the IFT algorithm is very recent. Most of the 22 journals produced by Scirus are dated post the year 2000. The majority of the journal papers were produced by the Control Engineering Practice (CEP) while IEEE produced the least and Automatica produced some where between the two.

With both Google and Scirus the keywords "Iterative method" produced papers and webpages dated from the nineteen sixties to the nineteen nineties, while most of the

results from the keywords "Iterative feedback tuning" are dated late nineteen nineties and early two thousands (when this project was initiated).

2.3 LITERATURE REVIEW

At the start of this project the special section on algorithms and application of Iterative Feedback Tuning (IFT) published by Control Engineering Practice (CEP) in 2002 was used as the initial source of literature material. In addition Automatica and various IEEE Journals produced other papers on Iterative Feedback Tuning method and its applications.

In the Control Engineering Practice (CEP) journals of 2002 there are many papers on the application of Iterative Feedback Tuning (IFT) algorithm- See [1, 2, 3, 4, 5, 6, 8, 9, 10]. These are now discussed in more detail:

The IFT algorithm was applied to tune controllers for a two-mass-spring system with friction in [1], and tuning of controller with application to a magnetic suspension system in [2]. Iterative controller optimization for nonlinear systems described in [3], and the signal convolution method was used for estimating the controller parameter sensitivity in [4]. Iterative Feedback Tuning of PID parameters were compared with classical tuning rules in [5], and the decoupling controller for a 2×2 system was tuned using Iterative Feedback Tuning in [6]. A special section in [8] dealt with algorithms and the applications of IFT algorithm, while in [9] Iterative Feedback Tuning is applied to internal model controllers. The IFT method presented in [10] shows how the parameters of a frequency domain controller are tuned.

Since the IFT algorithm was proposed in 1994 it has been shown to work well when applied to various systems. For example, in [1] the algorithm was shown to be very successful in controlling the mass-spring system's position under heavy friction where a two degree-of-freedom IFT controller was applied to a servo system. Two strategies were adopted in order to deal with this heavy friction: The one is to separate the tuning of the feedback and feed forward controllers and the other is to employ the Broyden-Fletcher-

Glodfard-Shanno (BFGS) method as a quasi- Newton method into a parameter update law.

In [2] the IFT algorithm was used to tune controller parameters based on the correlation approach. This was done by making the output error between the desired and the achieved output response uncorrelated with the setpoint signal. By doing this the IFT method can be used as an objective technique for controller tuning. In [3] the data-driven model-free control design method that was proposed by Hjalmarsson in 1994 was extended to a case where both the plant and the controller are allowed to be nonlinear. In this paper it was shown that one can obtain an estimate of the gradient experimentally by using the actual system with slightly different setpoint signals. The IFT method proved to achieve a fast response to setpoint changes, faster settling time as well as less overshoot in [5] compared with other classical PID tuning method namely: The Ziegler-Nichols (ZN) tuning rules, the internal model control (IMC) method and the integral square error (ISE) method.

The Iterative Feedback Tuning algorithm was applied to internal model control (IMC) and Smith predictor in [9]. In this case the algorithm was altered by doing four experiments instead of three to accommodate the tuning of the Smith predictor.

The Automatica journals over the period 1995 to 2003 contained three papers on the application of IFT algorithm [7, 11, 12]. In [7] the relay auto-tuning of PID controllers using Iterative Feedback Tuning was applied to a process control problem, in which the PID controller was auto tuned to give specific bandwidth and phase margin using an IFT scheme. The algorithm was tested in the laboratory on a coupled tank and the theoretical results were demonstrated to be observed in practice. The paper in [11] introduces a model free tuning algorithm that is based on frequency domain properties of the closed loop system's signals. The scheme used for this paper exploited the feature of the Iterative Feedback Tuning method.

The iterative identification scheme was applied to a sugar cane crushing mill in [12]. This paper examined the Zangscheme iterative identification and controller design as applied in a modified form to a sugar cane crushing mill. The selection of excitation and data filtering for model identification is done by using the connections between identification and robust control design.

The Iterative Feedback Tuning scheme was used to tune controllers for linear time invariant (LTI) multivariable system in [13] where the algorithm was applied to a laboratory model of a helicopter. As mentioned previously the paper in [15] by Hjammarsson presents one of the first publications of the Iterative Feedback Tuning method in 1994.

It has since been applied for a variety of tasks from the simplest like optimal tuning of simple PID controllers [5] to more complex tasks like systematic design of controllers of increasing complexity. From the literature survey it was noted that application to a DC motor had not previously been considered in detail. Since the DC motor is an important electrical engineering device and has dynamics that are very dependent on its loading it was chosen as the system on which to evaluate the IFT method. These results would give a good indication of the potential of IFT for engineering applications in the mineral extraction industry. In particular aspects such as the effect of its parameters, and the cycle time of its algorithm were to be investigated.

The IFT method was developed to solve most of the problems experienced in industry concerning model identification. In the control industry many problems can be expressed or cast in terms of a cost function. To be able to find the solutions to these optimization problems the details of a plant's model need to be known [14]. In most industrial cases it is very difficult to model a plant and its disturbances in detail and as a result it is hard to obtain the best performance from the controller used. For this reason Iterative Feedback Tuning method was developed. The input – output data design methods have been proposed and suggested by many researchers including Hjalmarsson. The advantage of

these methods is that they do not depend on the model of the plant. They utilize the input – output data only.

The IFT algorithm is based on iterative tuning of the controller parameters along the gradient direction of a given cost function. Tuning of controller parameters based on experimental data is a procedure that has been very actively researched of late. In [16, 17] the introduction of data identification is dealt with and, [18, 19] shows other surveys that dealt with iterative identification prior to the proposal of the IFT algorithm in 1994. IFT method can directly tune the controller from experimental data and the algorithm's method draws on the ideas from Model Reference Adaptive Control - See [15].

Chapter 3

This section analyses, presents and applies an MRAC to an actual system before applying the IFT algorithm because the concept of using experimental data for model identification originates from MRAC. This will allow a comparison between the two methods to provide a motivation for the selection of one over the other

3.1 ADAPTIVE CONTROL

Adaptive control is a method that is been in existence for decades and this section of the dissertation is mainly referenced from the book *Adaptive Control* by *KJ Astrom and B Wittenmark (1989) [22]* that deals with MRAC systems.

When something adapts that means it changes its state or behavior in order to fit the new specification or circumstances. Model Reference Adaptive Control, is one of the main approaches to adaptive control. (Another is STR or Self-Tuning –Regulator)

For instance an adaptive regulator is a regulator that can modify or change its parameters and hence its behavior in response to changes in its process dynamics and disturbances [22]. A controller should be able to modify its signals due to changes in loop dynamics and disturbances, and this is the reason why feedback control was introduced. The adaptive controller does this but also modifies its parameter values for additional flexibility.

Historically adaptive control systems were difficult to define as indicated in [22] that suggested the definition: "An adaptive system is any physical system that can be

designed with an adaptive viewpoint." In 1973 the term SOC (or Self Organizing Control) system was introduced but was not accepted by the control world.

Until that point in time a meaningful definition for Adaptive Control which would make it possible to have a look at the regulator hardware and software in order to decide if it is adaptive or not had still not been found [22].

3.2. MODEL REFERENCE ADAPTIVE CONTROL

The block diagram for a MRAC loop is shown in figure 3.1 (reproduced from [22]) and illustrates the basic principle of this algorithm. The desired performance is expressed in terms of a reference model, which gives the desired response y_m to a setpoint signal u_c. The system also has a feedback loop composed of a regulator and a plant.

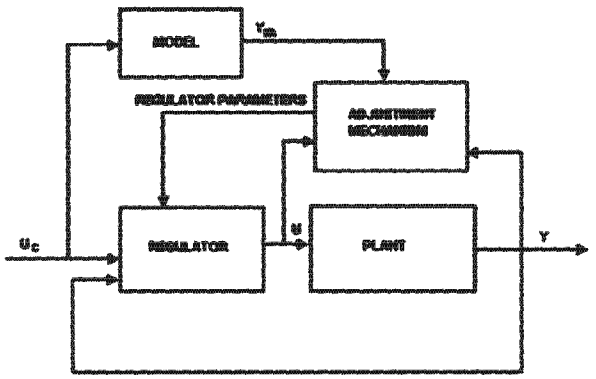

Figure 3.1: Block diagram for the MRAC system [22]

Figure 3.1 consists of two loops: The inner and the outer loop. The inner loop provides the ordinary control feedback while the outer loop adjusts the parameters of the regulator in the inner loop. This MRAC method makes the assumption that the inner loop is faster than the outer loop, (i.e that the variables in the inner loop changes more rapidly than the parameters in the outer loop [22]).

Whitaker and his team originally proposed the model reference adaptive control in 1958, and further work was done on the method in the sixties and seventies. Whitaker introduced two ideas on the method [22], which are:

- A model specifies the performance of the controlled system.
- The parameters of the regulator are adjusted based on the error between the reference model and the system.

Initially MRAC was derived for servo problems in deterministic continuous time systems. Later the ideas and the theory of MRAC were extended in order to cover discrete time systems and systems with stochastic disturbances [22].

The MRAC's analysis and design consists of three basic approaches known as the *Gradient* approach, the *Passivity Theory* approach and the *Lyapunov Function* approach.

This dissertation focuses on the gradient approach method because it is related to the Iterative Feedback Tuning.

3.2.1 The Gradient Approach

The gradient approach method does not always result in a stable closed-loop system, and as a result the application of stability theory is an alternative method to use. The gradient method is based on the assumption that the parameters in the system change more slowly than the variables.

The modelling error e_m in figure 3.2 is the difference between the output response y of the actual system and the desired output response y_m. The regulator has two degrees of freedom and contains two controller parameters, $t0$ and $s0$, that are changed based on this error e_m [22].

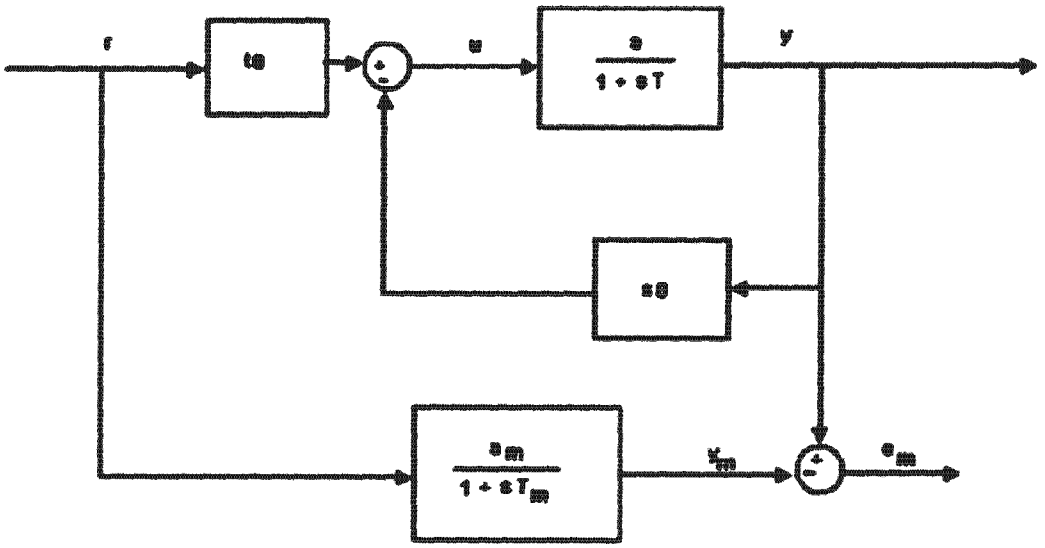

Figure 3.2: The block diagram of the MRAC gradient approach method shows the system for the DC motor speed control

The controller in the figure has the two loops associated with MRAC configurations:

- The inner loop, which is an ordinary feedback loop consisting of the process and the controller
- The outer loop, which adjust the controller parameters in such a way that the error $e_m = y - y_m$ is driven to zero

The MRAC algorithm is applied to a DC motor speed loop in which the process transfer function is as follows:

$$G(s) = \frac{a}{1+sT} \quad \text{..... (3.1)}$$

The assumption is made that the DC motor model is a simple one-pole system.

Following the example in [22], it is assumed that the controller has the form shown in figure 3.2:

$$u = t0 * r - s0 * y \quad \text{..... (3.2)}$$

The closed loop output response y of the process is:

$$y = \frac{a * t0}{1 + a * s0 + s * T} r \quad \text{..... (3.3)}$$

The model-following error e_m is defined as:

$$e_m = y - y_m \quad \text{..... (3.4)}$$

where y_m is the desired output response:

$$y_m = \frac{a_m}{1 + s * T_m} r \quad \text{..... (3.5)}$$

Thus, using equation (3.3), (3.4) and (3.5), the error becomes

$$e_m = \frac{a * t0}{1 + a * s0 + s * T} r - \frac{a_m}{1 + s * T_m} r \quad \text{..... (3.6)}$$

By comparing equation (3.3) and (3.5) the following controller parameter equations can be derived to give zero – error in equation (3.6):

$$t0 = \frac{a_m * T}{a * T_m} \quad \text{..... (3.7)}$$

$$s0 = \frac{T - T_m}{a * T_m} \quad \text{..... (3.8)}$$

For the feedback gain $s0$ to be positive the following constraint has to be satisfied:

$$T > T_m$$

where T is the time constant for the actual model and T_m is the time constant for the desired model. This will ensure that the desired model is faster than the process [22] as is also obvious from a root locus plot of this first order system.

The sensitivity derivatives [22] are obtained by taking partial derivatives of equation (3.6) with respect to the controller parameters $t0$ and $s0$

$$\frac{\partial e_m}{\partial t0} = \frac{a}{1 + a * s0 + s * T} r \quad \text{..... (3.9)}$$

$$\frac{\partial e_m}{\partial s0} = \frac{a^2 * t0}{\left(1 + a * s0 + s * T\right)^2} r \quad \text{..... (3.10)}$$

Equation (3.10) can be rewritten by using the actual output response in (3.3) to give the following equation:

$$\frac{\partial e_m}{\partial s0} = \left(\frac{a}{1 + a * s0 + s * T}\right) y \quad \text{..... (3.11)}$$

Equation (3.9) and (3.10) or (3.11) cannot be used directly in an MRAC algorithm because not all the parameters are known. In order to obtain realizable parameter adjustment rules approximations are required for the missing parameters [22].

To derive the approximation for the parameters it can be observed that the optimal value of the controller parameters $s0$ is:

$$1 + a * s0 + s * T = 1 + s * T_m \quad \text{..... (3.12)}$$

The next equation for updating the controller parameter can be derived using the approximation stated above:

$$\frac{\partial t0}{\partial t} = -\gamma\left(\frac{1}{1+s*T_m}r\right)*e_m \quad \text{..... (3.13)}$$

$$\frac{\partial s0}{\partial t} = \gamma\left(\frac{1}{1+s*T_m}y\right)*e_m \quad \text{..... (3.14)}$$

In equation (3.13) and (3.14) γ is the adaptation gain, and the convergence rate depends highly on this parameter [22].

Figure 3.3 shows equation (3.13) and (3.14) in block diagram format. This is for updating the controller parameters $s0$ and $t0$, to make sure the error e_m is driven to zero. The controller parameters $t0$ and $s0$ get updated at every iteration towards the best value without ever having to break the loop. The Model Reference Adaptive Control's gradient approach influenced the design of the next method known as Iterative Feedback Tuning.

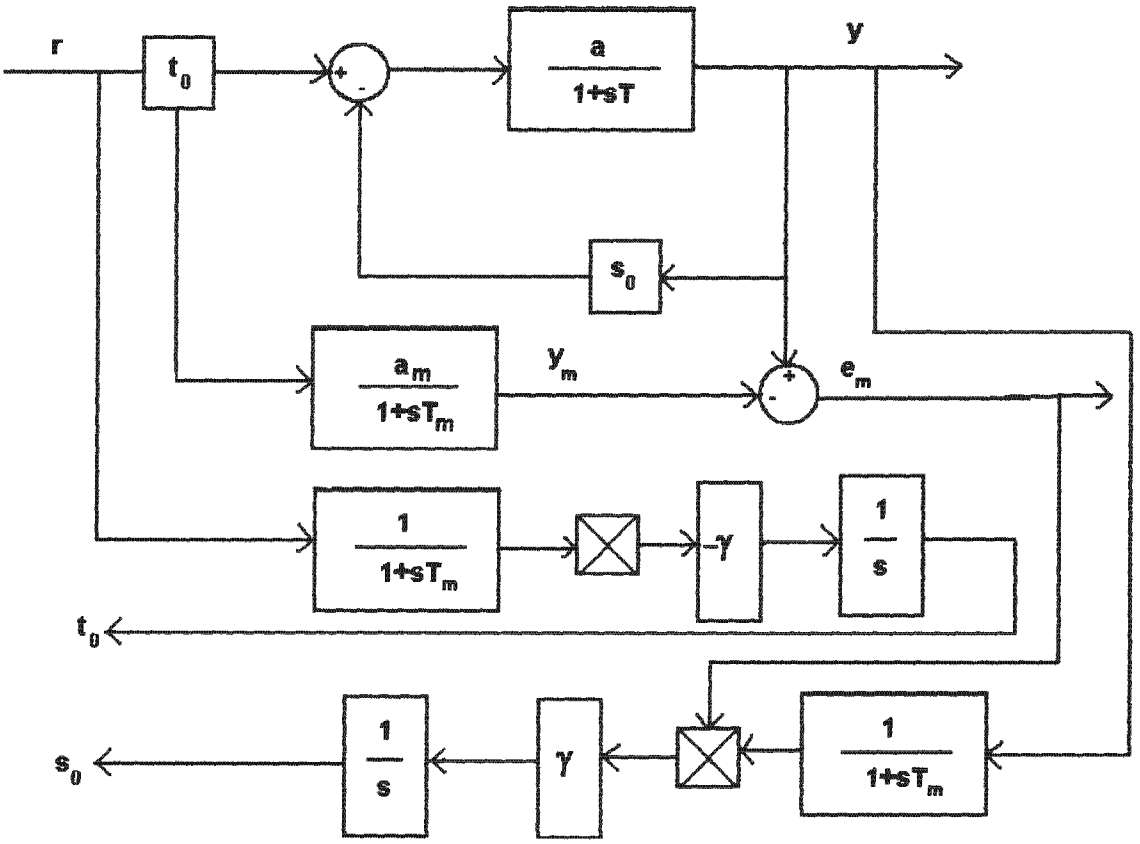

Figure 3.3: The block diagram of the MRAC gradient approach method with the regulator parameters updating

3.2.2 Application of MRAC gradient approach method on a DC motor

The MRAC method has been applied to a DC motor before - See [23, 24, 25, 26]. In [23] the MRAC is used for integrator control for precision positioning, in [24] the influence of parameters and structure of PI-type fuzzy-logic controller on a DC drive is applied. The multirate adaptive optimal control is applied to a DC motor in [25] and MRAC of air-lubricated capstan drive for precision positioning is applied in [26].

In [23] the single integrator controller is used to reduce the effect of friction so that good performance in precision positioning systems can be achieved. This paper proposed a double – integrator control in which two integrators are used to improve controller gain at lower frequencies. The double integrator was tested for precision positioning of an

aerostatic slide system in the presence of friction. The paper in [24] presents in detail the properties of the dc drive system with a fuzzy-logic speed controller in the presence of varying drive system inertia. The influence of the parameters and structure of a PI – type fuzzy controller on the dynamics of speed control was tested in the presence of disturbances caused by the varying inertia load torque.
The paper in [25] presents the implementation and theory of multirate adaptive optimal control system. This is done by combining the concept and theories of a multirate system, adaptive as well as optimal control. The real time control system is then studied with a number of simulations and control experiments using dc motor.

The MRAC gradient approach method was applied in this thesis to the DC motor for a speed control loop as this method was not found to have been applied to a DC motor for speed loop control before. The purpose of this application was to investigate how the MRAC gradient approach method will behave in practice in order to give a base case against which to compare the IFT method. In particular the research investigated various practical aspects such as how updating of the parameters would affect the system, how measurement noise would effect the system and whether there were any additional practical problem that the basic theory in [22] did not highlight.

3.2.3 THE MRAC PROGRAM

The program for the MRAC algorithm is written in Visual Basic and the main code statements are explained line by line:

Table 3.1 **Overview of the MRAC code**

COMPUTER CODE	DESCRIPTION
ts = Timer2.Interval / 1000	The sampling time t_s
Tm = 1	The time constant T_m
am = 1	The dc gain a_m
gama = 0.2	The positive real scalar λ
p = Exp(-ts / Tm)	Simplifying variables
*q = Exp(-am * ts)*	Simplifying variables
rta(j) = sgGenerator(rGen, Plot_t)	Compute the setpoint signal r_t
*ys0t = ((1 - q) * yt + am * q * ys0t) / am*	Simplifying variables
*yt0t = ((1 - q) * rt + am * q * yt0t) / am*	Simplifying variables
*s0t = s0t + ts * gama * em * ys0t*	Calculate the updated parameter
*t0t = t0t - ts * gama * em * yt0t*	Calculate the updated parameter
*ut = t0t * rt - s0t * yt*	Calculate the input signal u

COMPUTER CODE	DESCRIPTION
*ymt = (1 - p) * am * rt + p * ymt*	Calculate the desired output response y_m
em = yt - ymt	Calculate the error between the actual and the desired output response

The MRAC described in section 3.2.1 was coded in Visual Basic and it was executed on a PC in the Control and Instrumentation Laboratory. The experimental data collected during a typical run are shown in figure (3.4).

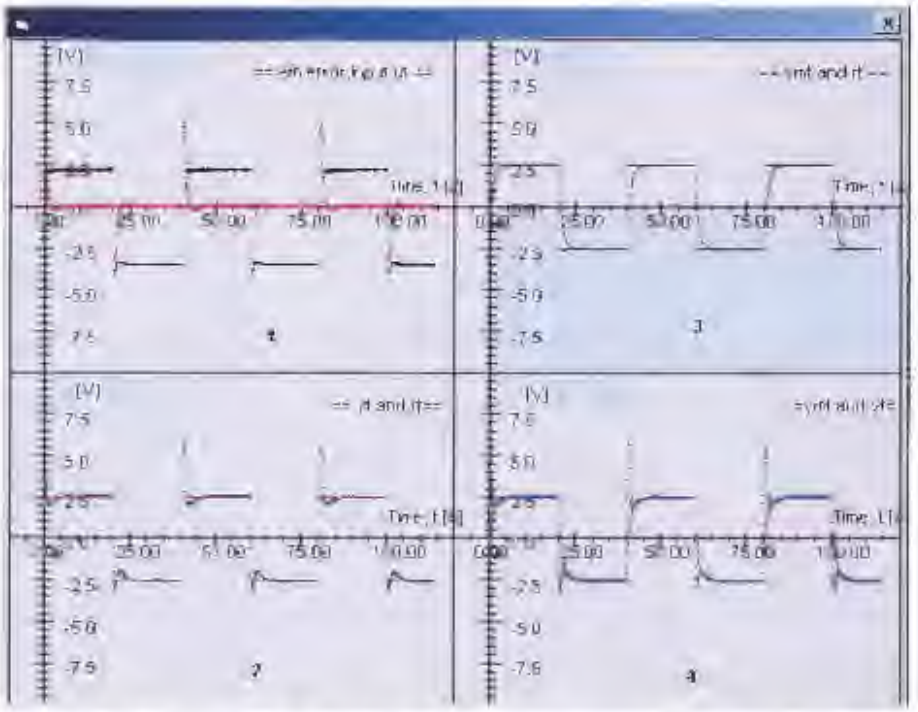

Figure 3.4: The results from the MRAC gradient approach method on a dc motor

The graph labeled 1 in figure 3.4 shows the input signal u in black given by equation (3.2) and the error signal e_m in red given by equation (3.4). Graph 2 shows the actual output response y in blue as well as the setpoint r in red, while graph 3 shows the desired response y_m in red as well as the setpoint r in black. Lastly graph 4 shows the desired response y_m in red and the actual response y in blue.

The steady state error e_m in graph 1 achieved the objective of the MRAC gradient approach method, namely that the error e_m be driven to zero as illustrated in graph 4. The actual output response y tracks the setpoint r perfectly at steady state after a settling time of about 5 seconds. Unfortunately there is an overshoot of approximately 60%

whenever the setpoint r changes. This was not predicted by the theory presented in the textbook [22].

It is extremely unlikely that the overshoot is due to any random event such as noise or disturbances since the overshoot is absolute consistent in each step response. As noted later in section 3.2.5 the overshoot is caused by assumptions made in the basic theory. The implications of these are that the MRAC gradient approach equations do not represents the physical motor system adequately.

3.2.4 The simulation of the MRAC gradient approach method

This section uses the Simulink simulation to investigate the reasons for the unexpected overshoot that was observed in figure (3.4).

The figure for the simulation of the MRAC gradient approach is shown in figure 3.5

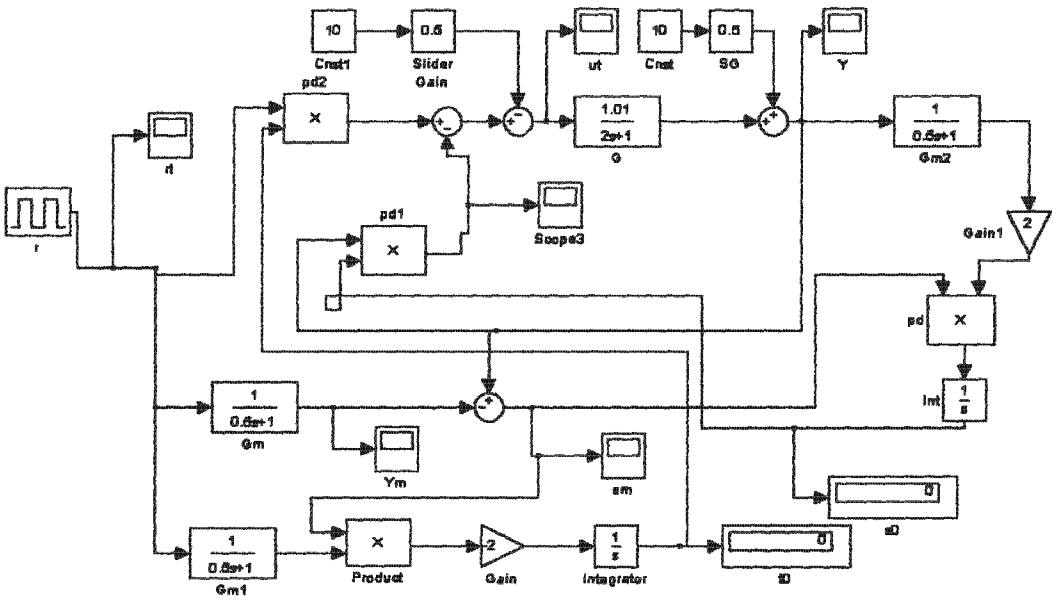

Figure 3.5: The simulation of the MRAC gradient approach method

It is important to note that the simulation has been configured to allow offsets on both the input and the output variables. In the initial simulation results these offsets are zero as required by the basic MRAC theory.

The following section shows the results obtained from the simulation of MRAC algorithm produced by the schematic shown in figure 3.5 under such conditions

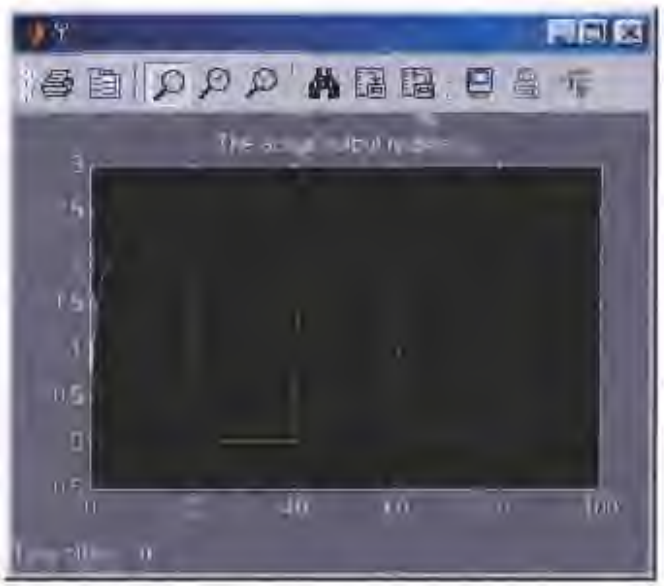

Figure 3.6: The simulation of the actual output response

The actual output response y to a square wave setpoint is shown in figure 3.6. As predicted by the theory, no overshoot is observed in the simulation responses.

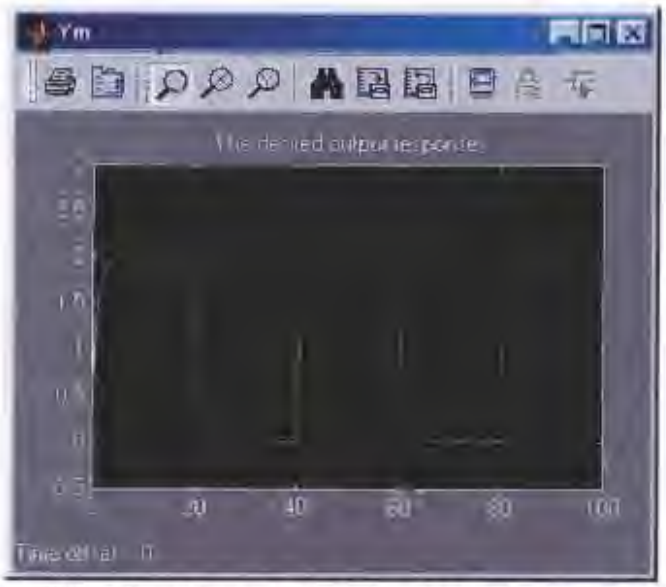

Figure 3.7: The simulation of the desired output response

Figure 3.7 shows the desired output response as predicted by the theory, this signal is almost identical to the one produced in the real experiment shown in figure 3.4 graph 3

Figure 3.8: The simulation of the input signal

The input signal u as predicted by the theory is shown in figure 3.8, it has an overshoot approximately 10% while the one produced by the real experiment in figure 3.4 has approximately 25% overshoot.

Figure 3.9: The simulation of the error signal

Figure 3.9 shows the error signal e_m as produced by the simulation, and it is as expected because the design and actual output response are equal.

3.2.5 The physical system

The next section explains the difference between the real system and the theoretical one. It was observed in section 3.2.1 that the actual output response y had an overshoot see figure (3.4), which was not expected according to the simulation of the output response see figure (3.5). A possible reason behind this could be due to the signal offsets U_0 and Y_0 that exits on the input and output response of the real system as shown in figure (3.10), which more precisely represents the complete diagram of practical system.

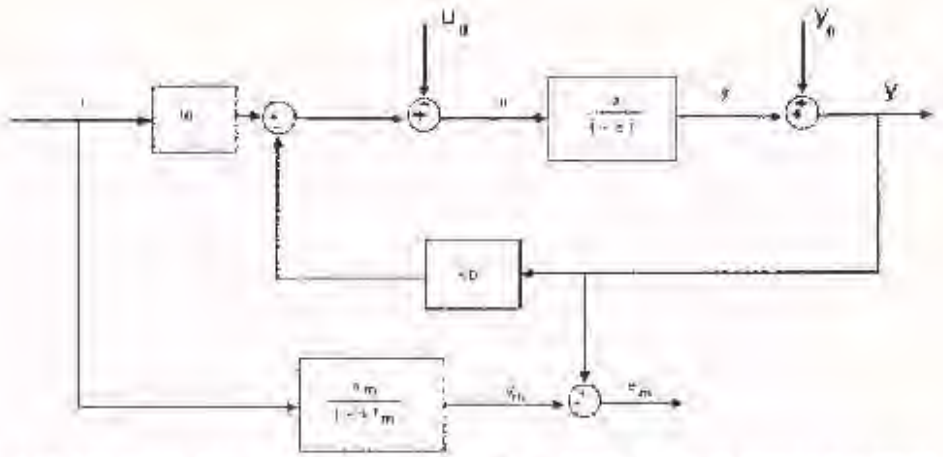

Figure 3.10: The block diagram of the real system of the MRAC gradient approach method

The effects that these offsets have on the MRAC equations are readily deduced from the original equations, given in above. Firstly the controller in figure 3.10 has the form:

$$u = t0 * r - s0 * y - s0 * Y_0 - U_0 \quad \text{..... (3.15)}$$

Equation (3.15) is written in terms of the offsets U_0 and Y_0

The closed loop actual output response y is:

$$y = \frac{a * t0 * r - a * U_0 - a * s0 * Y_0}{1 + a * s0 + s * T} \quad \text{..... (3.16)}$$

The real system output response is given by:

$$Y = y + Y_0 \quad \text{..... (3.17)}$$

substituting equation (3.16) into (3.17) gives the following equation:

$$Y = \frac{a * t0 * r - a * U_0 + Y_0 * (1 + s * T)}{1 + a * s0 + s * T} \quad \text{..... (3.18)}$$

The model-following error e_m is defined as:

$$e_m = Y - y_m \quad \text{..... (3.19)}$$

where y_m is the desired output response given by equation (3.5)

Thus, using equation (3.18), (3.5) and (3.19) the error becomes

$$e_m = \frac{a*t0*r - a*U_0 + Y_0*(1+s*T)}{1+a*s0+s*T} - \frac{a_m}{1+s*T_m}r \quad \text{..... (3.20)}$$

Since the inputs signals U_0 and Y_0 are independent of the controller parameter $t0$ and $s0$, the sensitivity derivatives are obtained by taking partial derivatives of equation (3.20) with respect to the controller parameter $t0$ and $s0$

$$\frac{\partial e_m}{\partial s0} = \frac{-a*\left(a*t0*r - a*U_0 + Y_0*(1+s*T)\right)}{\left(1+a*s0+s*T\right)^2} \quad \text{..... (3.21)}$$

$$\frac{\partial e_m}{\partial t0} = \frac{a}{1+a*s0+s*T}r \quad \text{..... (3.22)}$$

The sensitivity derivative with respect to the controller parameter $t0$ is the same as obtained in equation (3.9).

Equation (3.21) can be rewritten by using the real actual output response Y in (3.18) to give the following equation:

$$\frac{\partial e_m}{\partial s0} = \frac{-a*Y}{1+a*s0+s*T} \quad \text{..... (3.23)}$$

The gradient of the controller parameter $t0$ and $s0$ are independent of the desired model $\frac{a_m}{1+s*T_m}$ respectively hence the following equation can be written:

$$\frac{\partial e_m}{\partial s0} = \frac{\partial Y}{\partial s0} \quad \text{..... (3.24)}$$

$$\frac{\partial e_m}{\partial t0} = \frac{\partial Y}{\partial t0} \quad \text{..... (3.25)}$$

where Y is the actual output response from the real system and is given by equation (3.18)

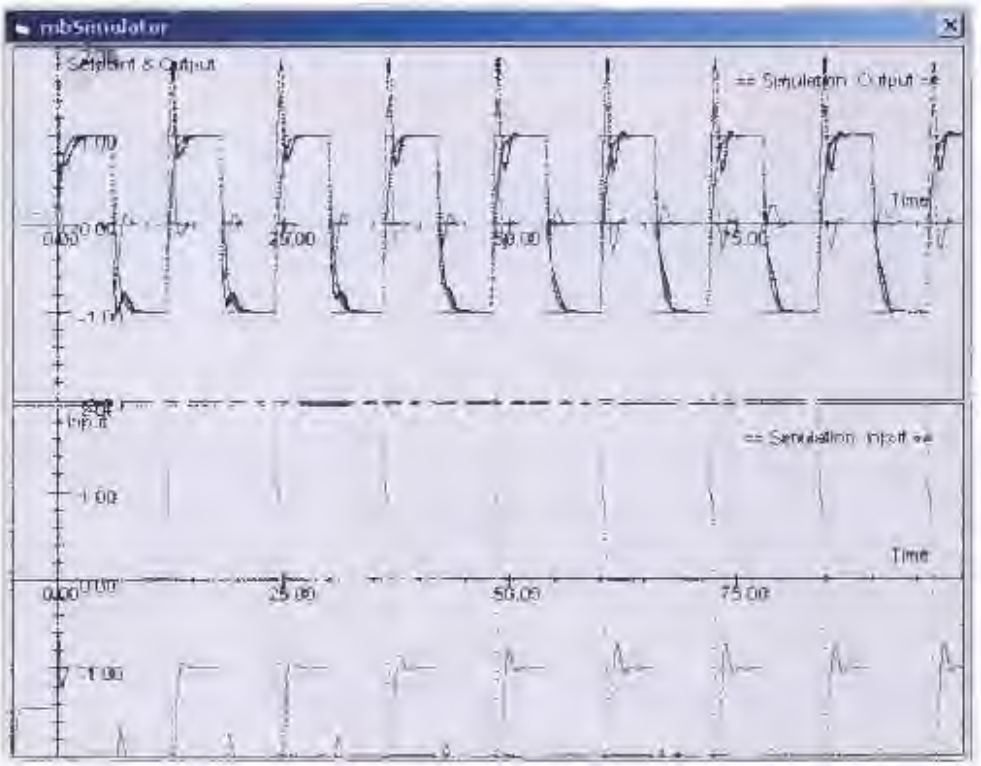

Figure 3.11: The simulation of the real system with offsets

Figure 3.11 shows the simulation of the schematic in figure 3.9 for the following setting: The offsets $U_0 = 1.00$ and $Y_0 = 3.00$, with these offsets the simulation illustrate the results obtained from the experiment.

The observation in figure 3.11 shows the overshoot experienced in the experiment, which the theory from [22] did not cover. The overshoot experience by the actual output response y could not be reduced hence IFT algorithm was suggested.

Chapter 4

Iterative Feedback Theory for a Single Variable System

For easy of the reference the general IFT theory is given in appendix 2, and will be used to develop a one degree of freedom system in this chapter. The one degree of freedom controller was chosen for IFT application to a system, while the two degree of freedom controller was chosen for MRAC application for this dissertation and the system chosen is a DC motor configured for speed control.

The one degree of freedom of controller was selected for its simplicity in that only two experiments need to be run on the system instead of the three required in the case of the two degree of freedom controller (for reasons that are explained in appendix 2). The type 1 or PI type controller was chosen for the application of the one degree of freedom controller because it is widely applied to industrial problems.

The DC motor system used in this experiment is not subject to any disturbances, therefore the design was modified from the general two degree of freedom figure in appendix 2 to suit the new situation. The DC motor does however contribute measurement noise to the system that is not accounted for in the theoretical equations in [14].

4.1 IFT WITH ONE DEGREE OF FREEDOM CONTROLLER

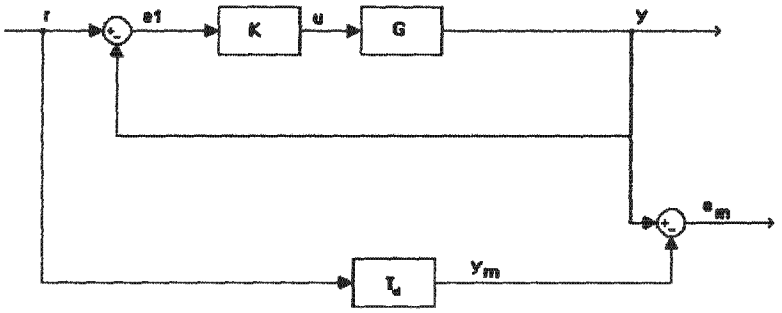

Figure 4.1: Block diagram of a one degree of freedom controller in a closed loop system

The block diagram in figure 4.1 shows the closed loop system of a one degree of freedom controller that is considered in this research on IFT.

Let K be a type 1 controller, G an unknown model representing the plant and T_d the desired model.

$$K(z) = \frac{\sigma_1 * z + \sigma_0}{(z-1)} \quad \text{..... (4.1)}$$

As mentioned in Appendix 2 the IFT equations minimize a quadratic criterion. The detailed mathematics was obtained from the literature and is presented in Appendix 2 for completeness and ease of reference. The essential equations can be summarized as follows:

In equation (4.1) above σ_0 and σ_1 are the scalar parameters of the chosen controller that need to be chosen to optimize the performance of the control system as defined by the quadratic criterion. These two parameters are elements in a vector of parameters, $\sigma = [\sigma_0 \quad \sigma_1]^T$. On the assumption that the noise and disturbances are zero the output response of the controlled system is given by

$$y = \left(\frac{G*K}{1+G*K} \right) * r \quad \text{..... (4.2)}$$

Let T be the transfer function of the closed loop system and let S be the sensitivity function. These are defined as:

$$T = \frac{G*K}{1+G*K} \quad \text{..... (4.3)}$$

$$S = \frac{1}{1+G*K} \quad \text{..... (4.4)}$$

Equation (4.2) can then be written as:

$$y = T*r \quad \text{..... (4.5)}$$

The modelling error in the IFT loop is defined as follows:

$$e_m = y - y_m \quad \text{..... (4.6)}$$

where y_m is the desired output response given by equation (A2.2) in appendix 2

4.1.1 Output signals for one degree of freedom controller

Following the standard literature on the IFT algorithm the quadratic criterion J is represented by:

$$J = \frac{1}{2} \sum_{1}^{N} e_m^2 \quad \text{..... (4.7)}$$

and the gradient of the criterion with respect to the controller parameters is represented by:

$$\frac{\partial J}{\partial \sigma} = -\sum_{1}^{N} e_m * \frac{\partial e_m}{\partial \sigma} \quad \text{..... (4.8)}$$

As stated in appendix 2 the gradient of the error e_m with respect to the vector of controller parameters σ depends only on the actual response y and not the desired response y_m. Thus the following equations provide the gradient for the cost function in terms of the elements of the controller parameter vector:

$$\frac{\partial e_m}{\partial \sigma_0} = \frac{\partial y}{\partial \sigma_0} \quad \text{..... (4.9)}$$

$$\frac{\partial e_m}{\partial \sigma_1} = \frac{\partial y}{\partial \sigma_1} \quad \text{..... (4.10)}$$

In order to determine the parameter that minimizes the quadratic criterion in equation (4.8), one needs to find the derivative of $\frac{\partial e_m}{\partial \sigma}$. This is equal to $\frac{\partial y}{\partial \sigma}$, because the desired model T_d is a constant and hence independent of the controller parameters (σ). As shown in appendix 2 the required gradient becomes:

$$\frac{\partial y}{\partial \sigma} = \frac{1}{K} * \left(\frac{G * K * r}{(1 + G * K)} \right) * \frac{\partial K}{\partial \sigma} - \frac{1}{K} * \left(\frac{G^2 * K^2 * r}{(1 + G * K)^2} \right) * \frac{\partial K}{\partial \sigma} \quad \text{..... (4.10)}$$

By substituting equation (4.3) and (4.4) into (4.10) the derivative can be rewritten as:

$$\frac{\partial y}{\partial \sigma} = \frac{1}{K} * \frac{\partial K}{\partial \sigma} * \left[T * r - T^2 * r \right] \quad \text{..... (4.11)}$$

Thus equation (4.11) is a simplified version of (4.10) and is the one that is used in the IFT method to adjust or tune the controller parameters.

In the general two degrees of freedom IFT controller, three experiments are needed in the algorithm but this is a one-degree of freedom controller so as explained in appendix 2 only two experiments needs to be generated.

Thus only two N - length setpoint signals are needed, $r_j^{(i)}$ where $i = 1,2$. The corresponding output signal is represented by:

$$r_j^{(1)} = r,\ y^{(1)} = T * r \quad \text{.....(4.12)}$$

$$r_j^{(2)} = \left(r_j^{(1)} - y^{(1)}\right),\ y^{(2)} = T * \left(r_j^{(1)} - y^{(1)}\right) = T * r - T^2 * r \quad \text{.....(4.13)}$$

It can be observed that the signal $y^{(2)}$ generated in equation (4.13) is exactly what is needed in equation (4.11), therefore equation (4.11) can be rewritten as:

$$\frac{\partial y}{\partial \sigma} = \frac{1}{K}\frac{\partial K}{\partial \sigma} y^{(2)} \quad \text{.....(4.14)}$$

In this application the controller parameters form a 2-vector hence $\sigma = [\sigma_0 \quad \sigma_1]$, the criterion minimization in equation (4.11) readily determined and the update of the controller parameters is given by the following equations:

$$\sigma_{0_{j+1}} = \sigma_0 - \gamma * r_{11}\frac{\partial J}{\partial \sigma_0} - \gamma * r_{12}\frac{\partial J}{\partial \sigma_1} \quad \text{.....(4.15)}$$

$$\sigma_{1_{j+1}} = \sigma_1 - \gamma * r_{21}\frac{\partial J}{\partial \sigma_0} - \gamma * r_{22}\frac{\partial J}{\partial \sigma_1} \quad \text{.....(4.16)}$$

These two equations are derived from the general one given in equation (A2.12) in appendix 2. The constants r_{11}, r_{12}, r_{21} and r_{22} are the elements of the matrix R_j given in equation (A2.12) in appendix 2.

4.1.2 Input signals for the one degree of freedom controller

From the IFT control loop shown in figure 4.1 the process input is obtained from the controller as shown in the transfer function equation:

$$u = K * (r - y) \quad \text{.....(4.17)}$$

The closed loop transfer function T given by equation (4.3), the sensitivity function S given by equation (4.4) and expression for the response given by equation (4.5) can be used in equation (4.17) to produce the following expression for the process input:

$$u = K * S * r \quad \text{.....(4.18)}$$

By generating the two actual output response signals, $y^{(1)}$ and $y^{(2)}$ in equations (4.12) and (4.13) the corresponding two inputs signal are the following:

$$u^{(1)} = K * S * r \quad \text{..... (4.19)}$$

$$u^{(2)} = K * (S * r - S * T * r) \quad \text{..... (4.20)}$$

where K is a controller given by equation (4.1)

The gradient of the input with respect to the controller parameter vector sigma (σ) is given by:

$$\frac{\partial u}{\partial \sigma} = \frac{1}{K} * \frac{\partial K}{\partial \sigma} * u^{(2)} \quad \text{..... (4.21)}$$

Since the controller parameter sigma (σ) is a vector, equation (4.21) can be defined further with respect to σ_1 and σ_2 respectively. This produces the two input-response gradients:

$$\frac{\partial u}{\partial \sigma_0} = \frac{1}{(\sigma_1 * z + \sigma_0)} * u^{(2)} \quad \text{..... (4.22)}$$

$$\frac{\partial u}{\partial \sigma_1} = \frac{z}{(\sigma_1 * z + \sigma_0)} * u^{(2)} \quad \text{..... (4.23)}$$

Similarly the output response's gradient with respect to the controller parameter (σ) in equation (4.11) can be rewritten further in terms of the controller parameters σ_0 and σ_1 respectively:

$$\frac{\partial y}{\partial \sigma_0} = \frac{1}{(\sigma_1 * z + \sigma_0)} * y^{(2)} \quad \text{..... (4.24)}$$

$$\frac{\partial y}{\partial \sigma_1} = \frac{z}{(\sigma_1 * z + \sigma_0)} * y^{(2)} \quad \text{..... (4.25)}$$

4.1.3 Summary of the IFT one degree of freedom controller algorithm:

With the given two-parameter controller K operating on the DC motor system, the signals $y^{(1)}$ and $y^{(2)}$ given by equation (4.12) and (4.13)

$$r_j^{(1)} = r, \text{ and } y^{(1)} = T * r \qquad \text{Experiment \#1}$$

$$r_j^{(2)} = \left(r_j^{(1)} - y^{(1)}\right) \text{ and } y^{(2)} = T * \left(r_j^{(1)} - y^{(1)}\right) = T * r - T^2 * r \qquad \text{Experiment \#2}$$

are the measured responses of the controlled motor system to the setpoints $r^{(1)}$ and $r^{(2)}$.

Once these experiments have been run the output gradient signals $\frac{\partial y}{\partial \sigma_0}$, $\frac{\partial y}{\partial \sigma_1}$ given by equation (4.24) and (4.25)

$$\frac{\partial y}{\partial \sigma_0} = \frac{1}{(\sigma_1 * z + \sigma_0)} * y^{(2)}$$

$$\frac{\partial y}{\partial \sigma_1} = \frac{z}{(\sigma_1 * z + \sigma_0)} * y^{(2)}$$

are computed by the IFT algorithm using equation (4.8). The gradient of the criterion $\frac{\partial J}{\partial \sigma}$ can be computed in terms of controller parameters σ_0 and σ_1 respectively.

$$\frac{\partial J}{\partial \sigma} = -\sum_{1}^{N} e_m * \frac{\partial y}{\partial \sigma}$$

Once the gradient of the quadratic criterion has been estimated, the next controller parameters σ_0 and σ_1 are updated recursively using equations (4.15) and (4.16) respectively.

$$\sigma_{0_{j+1}} = \sigma_0 - \gamma * r_{11} \frac{\partial J}{\partial \sigma_0} - \gamma * r_{12} \frac{\partial J}{\partial \sigma_1}$$

$$\sigma_{1_{j+1}} = \sigma_1 - \gamma * r_{21} \frac{\partial J}{\partial \sigma_0} - \gamma * r_{22} \frac{\partial J}{\partial \sigma_1}$$

The R matrix used in these parameter equations is either the identity matrix or the matric computed from equation (A2.31)

$$R_j = \frac{1}{N}\sum_{i=1}^{N}\left(est\left[\frac{\partial y}{\partial \sigma}(\sigma_j)\right] est\left[\frac{\partial y}{\partial \sigma}(\sigma_j)\right]^T + \lambda * est\left[\frac{\partial u}{\partial \sigma}(\sigma_j)\right] est\left[\frac{\partial u}{\partial \sigma}(\sigma_j)\right]^T \right)$$

that needs both the output and the input gradients with respect to the controller parameters. The latter are obtained from equations (4.22) and (4.23)

$$\frac{\partial u}{\partial \sigma_0} = \frac{1}{(\sigma_1 * z + \sigma_0)} * u^{(2)}$$

$$\frac{\partial u}{\partial \sigma_1} = \frac{z}{(\sigma_1 * z + \sigma_0)} * u^{(2)}$$

4.1.4 Summary of the IFT parameters

The following table shows all of the IFT parameters used in this research and their purpose in the algorithm.

Table 4.1: IFT parameters

The parameters	**Their role in the algorithm**
λ	This is the weighting factor and its role is to scale the second term in the matrix R_j given by equation (A2.31). The user determines this parameter.
γ	The constant γ is a positive scalar that sets the rate at which the algorithm will converge. The user determines this parameter.
N	The finite length N is the length for algorithm iteration. The user determines this parameter.
σ_0 and σ_1	The controller parameters. This parameters are determined by the IFT algorithm thought the initial values are set by the user..
t	This is the sampling time for the duration of each experiment, it is important as it gives the signal enough time to settle before the next experiment takes place. The user determines this parameter.

Chapter 5

An Applications for the One Degree of Freedom IFT controller

Having presented the theory for two -parameter one degree of freedom controller in chapter 4, it is now applied to a DC motor. The IFT algorithm has being applied to a DC motor in [4] where it is used in conjunction with a signal convolution method for estimation of controller parameter sensitivity functions. The reason for using the speed control of a DC motor on which to evaluate the IFT algorithm are that it is available in the laboratory and has parameters that can be varied in order to explore the parameter tracking ability of the IFT controller.

The reason for applying the one degree of freedom controller is its simplicity, and the fact that the sensitivity of the actual response of the physical system to modeling error, disturbances and load changes has not been considered in previous literature. As mentioned before only two experiments are required in this instance while three are required for a two degree of freedom controller.

Figure 5.1 is a picture of the DC motor (Servo motor) used for the application of the IFT one degree of freedom controller.

5.1 APPLICATION OF IFT FOR ONE DEGREE OF FREEDOM CONTROLLER TO A DC MOTOR

Figure 5.1: The picture of DC motor

The DC motor used in this research for the MRAC and IFT methods consists of the following apparatus:

- Pre – amplifier unit
- Attenuator unit
- Op Amp unit
- Input pot unit
- DC motor
- Reduction Gear Tacho unit
- Output pot unit
- Servo Amplifier
- Power supply
- Analog/Digital Converter
- Digital Multimeter
- PC or Computer

The A/D system used was a Data Translation DT2801 (or DT302). Its range is set to -10 to +10 [V] with a 12-bit resolution giving 20[V] for a 4095[Count] range. The DC motor is a +/-15[V] system so the experiments had to ensure that the range of the Data Translation card was never exceeded.

5.2 EXPERIMENT 1- IFT CONTROL OF THE DC MOTOR

The next section shows the results from the application of the IFT algorithm for the one degree of freedom controller. The starting values for the controller parameters σ_0 and σ_1 were chosen after a test step with random amplitudes on the DC motor in which it was found that the motor operate better at low values of the controller parameters. When the controller parameters approach or exceed the value of 1 the output from the motor speed control becomes unstable. Hence the following starting values were chosen for the controller parameters.

5.2.1 Conditions for the experiment:

- The controller parameters starting values

  $\sigma_0 = 0.03$

  $\sigma_1 = 0.03$

- The matrix R_j

  The matrix has constant elements that can have any value as long as it remains a positive matrix [14] hence the 2x2 identity matrix was chosen.

$$R_j = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \text{.....(5.1)}$$

The signals that were obtained when this IFT control was connected to the DC motor are shown in figure 5.2.

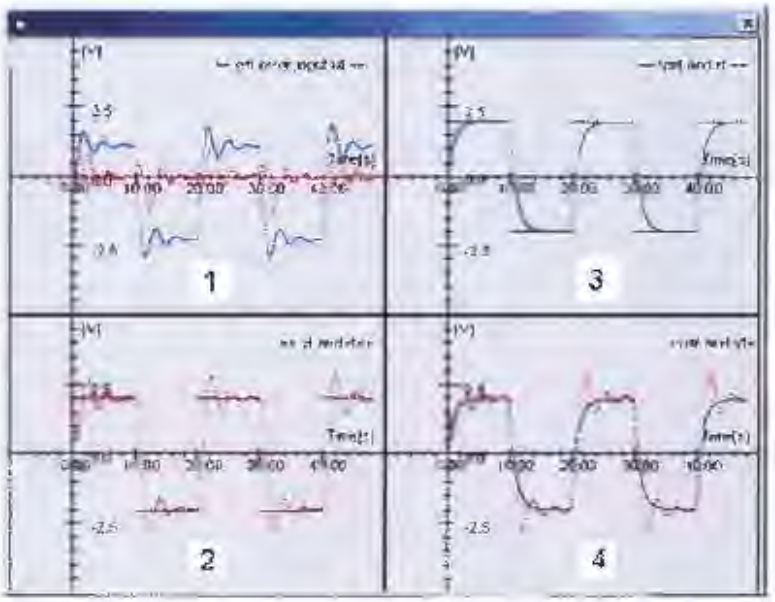

Figure 5.2: The experimental results of IFT Control of the DC Motor

Graph 1 on figure 5.2 shows error signal e_m in red given by equation (3.6) and the input signal $u^{(1)}$ in blue given by equation (3.17), while graph 2 shows the setpoint signal $r^{(1)}$ in black and the actual output response signal $y^{(1)}$ in red. Graph 3 shows the desired output response y_m in blue with the setpoint signal $r^{(1)}$, while graph 4 shows the desired output response y_m in blue with the actual output response $y^{(1)}$ in red.

For this experiment the setpoint $r^{(1)}$ had an amplitude of 2[V] and period of 20 seconds. The effect of changing the setpoint to 1[V] and 20 [s] is illustrated by repeating the experiment, as shown in Fig.5.3.

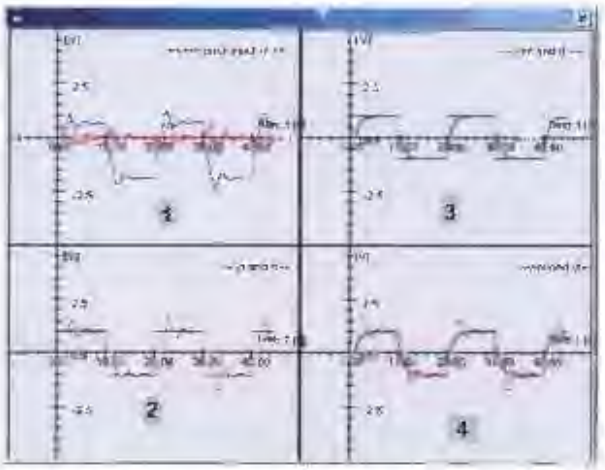

Figure 5.3: The experimental results for IFT control of the DC motor

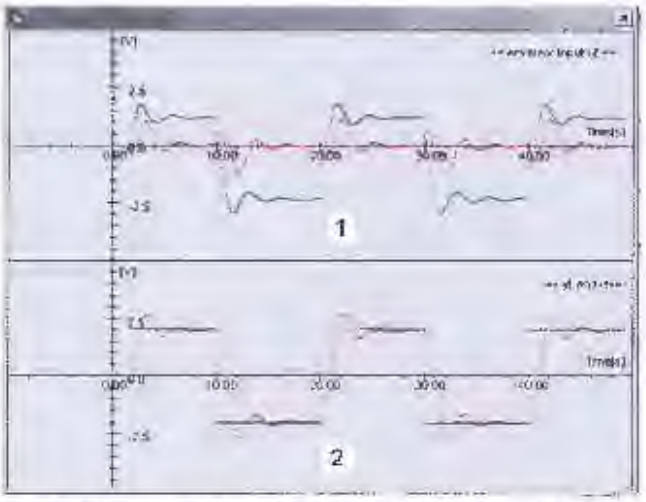

Figure 5.4: The results from the experiment 1 for the IFT algorithm

Figure 5.4 shows an enlargement of the first two graphs from figure 5.2. The input signal $u^{(1)}$ in blue and the error signal e_w in red are given in graph 1, while graph 2 shows the actual output response $y^{(1)}$ in red together with the setpoint $r^{(1)}$ in black in experiment 1 part of the IFT algorithm

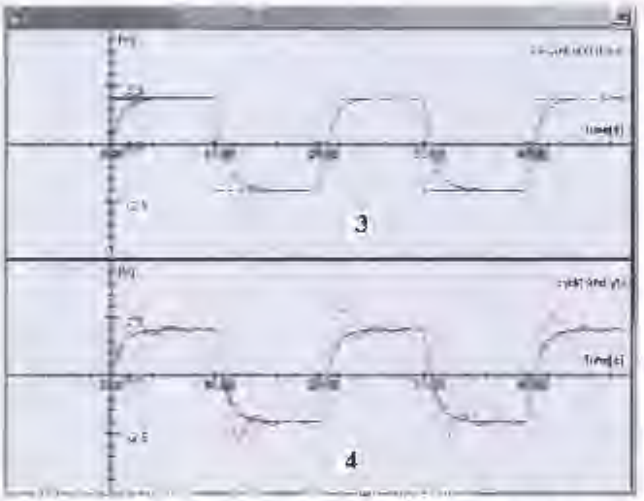

Figure 5.5: The results from the experiment 1 for the IFT algorithm

Graph 3 in figure 5.5 shows the desired output response y_m in blue at the top together with the setpoint signal $r^{(1)}$ in black, while graph 4 in figure 5.5 labeled 4 shows the desired output response y_m in blue together with the actual output response $y^{(1)}$ in red.

The first observation from this experiment shown in figure 5.3 and 5.4 is that changing the amplitude of the setpoint $r^{(1)}$ did not change the outcome of the shape of the actual output response $y^{(1)}$. Secondly the overshoot experienced by the actual output response $y^{(1)}$ is smaller than the one experienced in the MRAC algorithm y by comparison of the responses in figures 2.4 and figure 5.5.

5.3 RESULTS FROM EXPERIMENT 2 FOR THE DC MOTOR

Experiment 2 of the IFT algorithm uses the data that was generated in experiment 1. For example the setpoint $r^{(2)}$ for experiment 2 must be given by equation (3.13). This is generated in experiment 1 and used in experiment 2 and is shown in figure 5.6. One needs to make sure that the setpoint used in experiment 2 is exactly the one generated in experiment 1. The following figures show that.

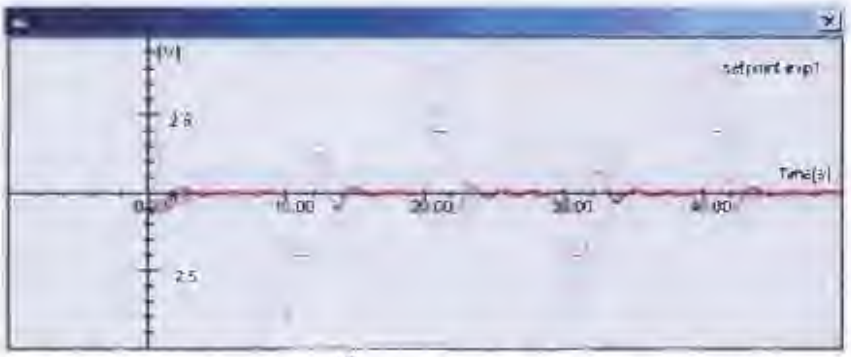

Figure 5.6: The $r_t - y_t^{(1)}$ graph from IFT's experiment 1

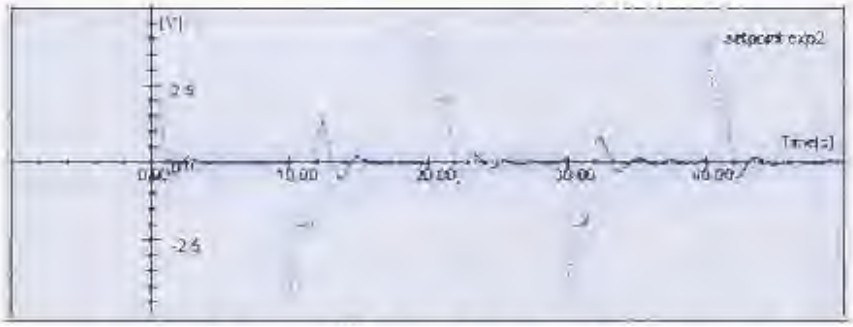

Figure 5.7: The $r_t^{(2)}$ graph from IFT's experiment 2

Figure 5.6 shows the data generated in experiment 1 namely the difference between the setpoint signal $r^{(1)}$ and the actual output response $y^{(1)}$, while figure 5.7 shows the data used in experiment 2 as the setpoint signal $r^{(2)}$. The two signals in figure 5.6 and figure 5.7 are identical therefore the results produced by the algorithm are for the correct setpoint.

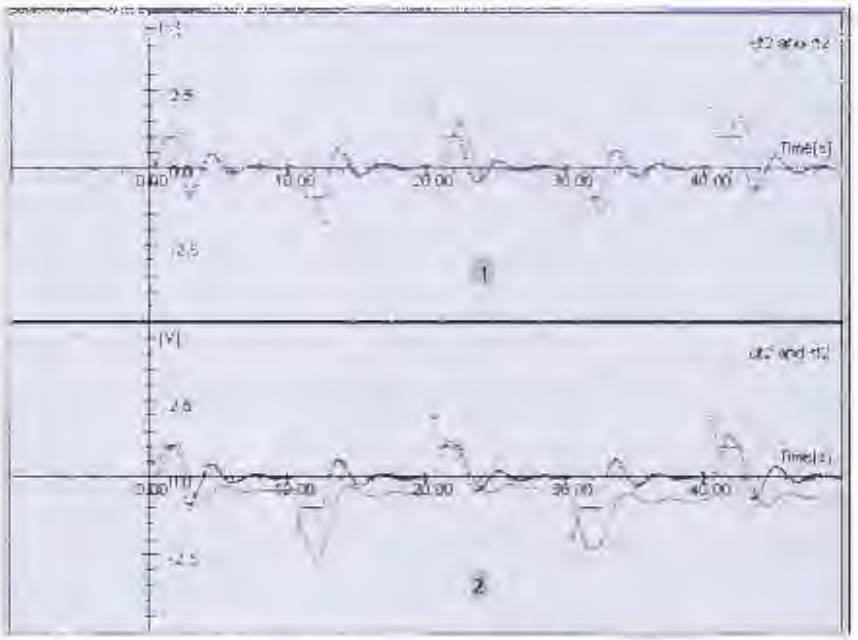

Figure 5.8: The results from experiment 2 for the IFT algorithm

Figure 5.8 shows the signals obtained from experiment 2 run by the IFT algorithm. In the graph labeled 1 is the setpoint signal $r^{(2)}$ in blue given by equation (5.13) which is the difference between the setpoint signal $r^{(1)}$ and the actual output response $y^{(1)}$ from experiment 1. The output response $y^{(2)}$ from the second experiment is shown in red. The graph 2 in figure 5.8 shows the input signal $u^{(2)}$ in red together with the setpoint signal $r^{(2)}$ in black.

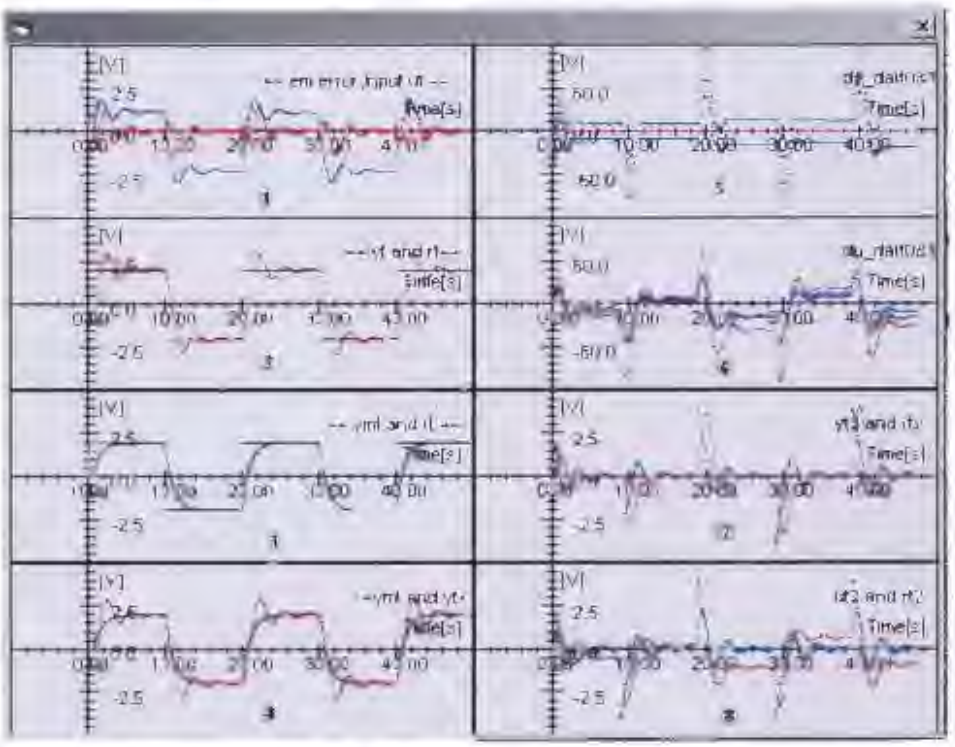

Figure 5.9: The results from both Experiment 1 and 2 of the IFT algorithm

This figure 5.9 shows results for both experiment 1 and 2 for the IFT algorithm when the matrix R_j is an identity matrix. Graphs labeled 1 to 4 are from experiment 1 while graphs 5 to 8 are from experiment 2.

Graph 7 shows the actual output response $y^{(2)}$ for experiment 2 in red when the setpoint signal $r^{(2)}$ in blue given by equation (4.13) is applied to the system. Graph 7 also shows that the actual output response $y^{(2)}$ is zero when the actual output response $y^{(1)}$ from experiment 1 graph labeled 2 has tracked the setpoint signal $r^{(1)}$. This is so because the setpoint signal $r^{(2)}$ to experiment 2 is the difference between the setpoint signal $r^{(1)}$ in experiment1 and the actual output response $y^{(1)}$. Therefore if $r^{(1)}$ and $y^{(1)}$ are equal it means that the setpoint signal in the second experiment $r^{(2)}$ is zero and hence the actual output response $y^{(2)}$ is also zero.

Graph 5 in figure 5.7 shows the output gradient with respect to the controller parameter σ_0 and σ_1, $\frac{\partial y}{\partial \sigma_0}$ in red and $\frac{\partial y}{\partial \sigma_1}$ in blue. Graph 6 shows the input gradient with respect to the controller parameter σ_0 and σ_1, $\frac{\partial u}{\partial \sigma_0}$ in red and $\frac{\partial u}{\partial \sigma_1}$ in blue. The equations for these time signals are those given in equation (5.22), (5.23), (5.20) and (5.21) respectively. Graph 8 shows the input signal $u^{(2)}$ in red together with the setpoint signal $r^{(2)}$ in blue.

5.3.1 Ringing in the IFT Responses

The double trace on some signals in graphs 5 and 6 of figure 5.9 were not expected from the theory of IFT system and were investigated in detail. This showed that they were due to the phenomena of ringing as demonstrated in the following section that deals with the ringing phenomena in the IFT responses.

The $z-plane$ is a useful tool in analysis of digital, sampled data systems. In particular the effect of pole-positions can be summarized by the diagram in figure.5.10 for real poles that show the position of ringing poles.

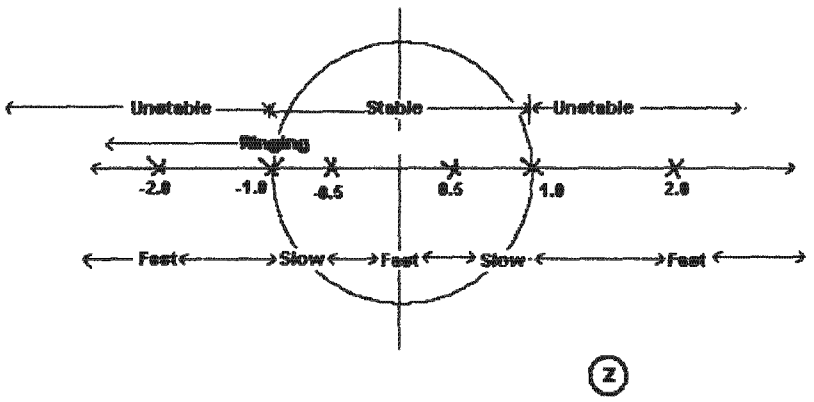

Figure 5.10: Real pole position in the z-plane

Specifically any poles on the negative real axis of the z-plane lead to the phenomena known as ringing.

An enlarged view of figure 5.9 is given in figure 5.11 clearly shows the apparent double trace in blue in graphs labeled 5 and 6. The investigation showed that the ringing phenomenon resulted from of the controller parameters σ_0 and σ_1 having the same value, in this case 0.03.

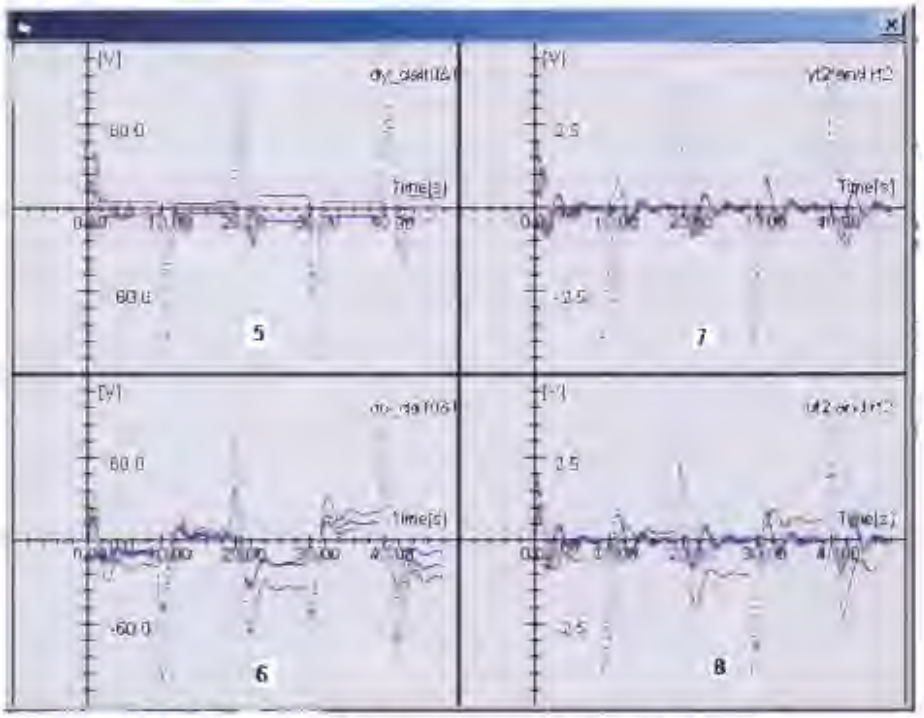

Figure 5.11: The results from experiment 2 of the IFT algorithm

With reference to equation 4.22, 4.23 4.24 and 4.25 poles for the gradient transfer function of the input signal with respect to the controller parameters $\frac{\partial u}{\partial \sigma_0}$ and $\frac{\partial u}{\partial \sigma_1}$, as well as the gradient of the actual output response with respect to the controller parameters $\frac{\partial y}{\partial \sigma_0}$ and $\frac{\partial y}{\partial \sigma_1}$ system are defined by $z = -\frac{\sigma_0}{\sigma_1}$. In the above case with the starting parameters set to be equal this means that the poles are on the unit circle in the z-plane and on the negative real axis and hence that the IFT will exhibit the ringing phenomena.

5.3.2 Varying the controller parameters sigma σ_0 and σ_1

Having identified the cause of the problem noted with the IFT algorithm, the position of this ringing pole was moved by altering the controller parameters σ_0 and σ_1, as described in the next section.

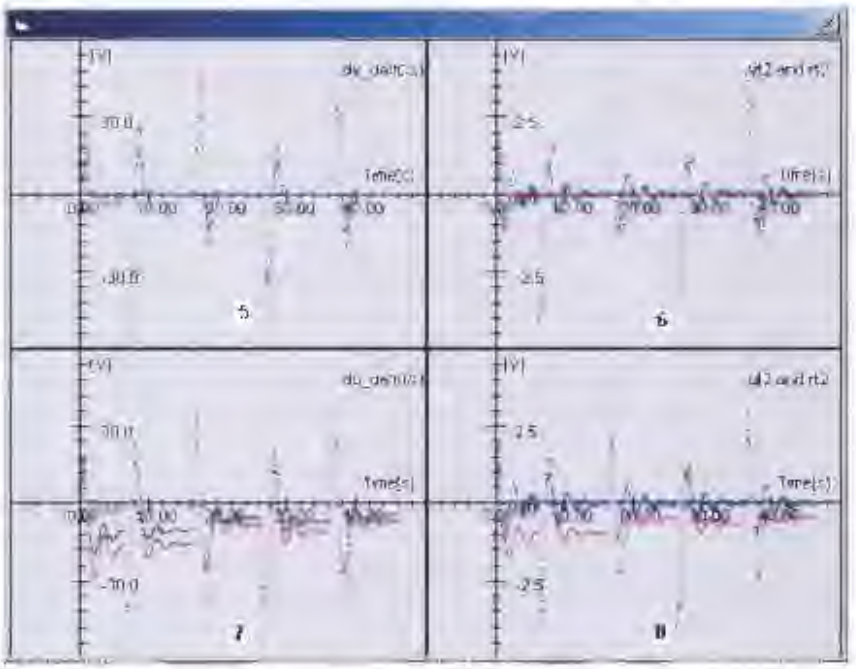

Figure 5.12 The results from experiment 2 without the oscillation(ringing) phenomena

Figure 5.12 shows the results from experiment 2 when the controller parameters have the following values $\sigma_0 = 0.03$ and $\sigma_1 = 0.06$ this place the pole $z = -0.5$, it can be observed that the ringing phenomena has disappeared, as can be seen in graphs 5 and 6. The IFT equations are stable since the pole falls inside the unit circle in the z-plane and the transients due to the pole of magnitude 0.5 decay within six samples.

5.4 THE IFT PROGRAM

The program for the IFT algorithm is written in Visual Basic and the main statements are explained line by line:

Table 5.1 **Overview of the IFT code** Experiment 1

COMPUTER CODE	DESCRIPTION
alf0 = alf0z.Text	Read the initial parameters
alf1 = alf1z.Tex	Read the initial parameters
rta(j) = sgGenerator(rGen, Plot_t)	Read the setpoint signal r, which is in a vector format
yta(j) = yt	Store the actual output response $y^{(1)}$
et1a(j) = rta(j) - yta(j)	Store the error, which is the setpoint signal to experiment 2
*uta(j) = alf1 * et1a(j) + alf0 * et1a(j) + uta(j - 1)*	Calculate the input signal $u^{(1)}$
*ymta(j) = am * rta(j) - am * p * rta(j) + p * ymta(j - 1)*	Calculate the desired output response y_m
emta(j) = yta(j) - ymta(j)	Calculate the error between the actual and the desired output response
*ja = ja + 0.5 * emta(j) ^ 2*	Do the summation of the criterion J

Table 5.2 **Overview of the IFT code** Experiment 2

COMPUTER CODE	DESCRIPTION
rt = et1a(j)	Read the setpoint signal, which was generated and store from experiment 1
yta2(j) = yt	Read the actual output response for $y^{(2)}$
et2a(j) = rt – yt	Calculate the error between the setpoint signal and the actual output response for experiment 2
*uta2(j) = alf1 * et2a(j) + alf0 * et2a(j) + uta2(j - 1)*	Calculate the input signal $u^{(2)}$
ut = uta2(j)	Store the input signal
*dy_dalf0t(j) = (yta2(j) - alf0 * dy_dalf0t(j)) / alf1*	Calculate the output gradient with respect to the controller parameter σ_0
*dy_dalf1t(j) = (yta2(j) - alf0 * dy_dalf1t(j - 1)) / alf1*	Calculate the input gradient with respect to the controller parameter σ_1
*du_dalf0t(j) = (uta2(j) - alf0 * du_dalf0t(j)) / alf1*	Calculate the input gradient with respect to the controller parameter σ_0
*du_dalf1t(j) = (uta2(j) - alf0 * du_dalf1t(j - 1)) / alf1*	Calculate the output gradient with respect to the controller parameter σ_1
dy_alf0_tot = dy_alf0_tot + dy_dalf0t(j)	Calculate the summation of the output gradient with respect to the controller parameter σ_0
dy_alf1_tot = dy_alf1_tot + dy_dalf1t(j)	Calculate the summation of the output gradient with respect to the controller parameter σ_1

COMPUTER CODE	DESCRIPTION
du_alf0_tot = du_alf_tot + du_dalf0t(j)	Calculate the summation of the input gradient with respect to the controller parameter σ_0
du_alf1_tot = du_alf_tot + du_dalf1t(j)	Calculate the summation of the input gradient with respect to the controller parameter σ_1
*dj_dalf0 = dj_dalf0 - emta(j) * dy_dalf0t(j)*	Calculate the criterion minimization for controller parameter σ_0
*dj_dalf1 = dj_dalf1 - emta(j) * dy_dalf1t(j)*	Calculate the criterion minimization for controller parameter σ_1
*r11t = ((dy_alf0_tot) ^ 2 + lamda * (du_alf0_tot) ^ 2) / 1000*	When R_j is defined by (3.30), one needs to calculate the indexes of $R_j = \begin{bmatrix} r11 & r12 \\ r21 & r22 \end{bmatrix}$, the next four lines this one included does that.
*r12t = (dy_alf0_tot * dy_alf1_tot + lamda * (du_alf0_tot * du_alf1_tot)) / 1000*	
*r21t = ((dy_alf1_tot * dy_alf0_tot) + lamda * (du_alf1_tot * du_alf0_tot)) / 1000*	
*r22t = ((dy_alf1_tot) ^ 2 + lamda * (du_alf1_tot) ^ 2) / 1000*	
*alf0 = alf0 - (gamaz * r11t * dj_dalf0) / 1000 - gamaz * r12t * (dj_dalf1) / 1000*	The calculation of the updating controller parameter σ_0
*alf1 = alf1 - (gamaz * r21t * dj_dalf0) / 1000 - gamaz * r22t * (dj_dalf1) / 1000*	The calculation of the updating controller parameter σ_1

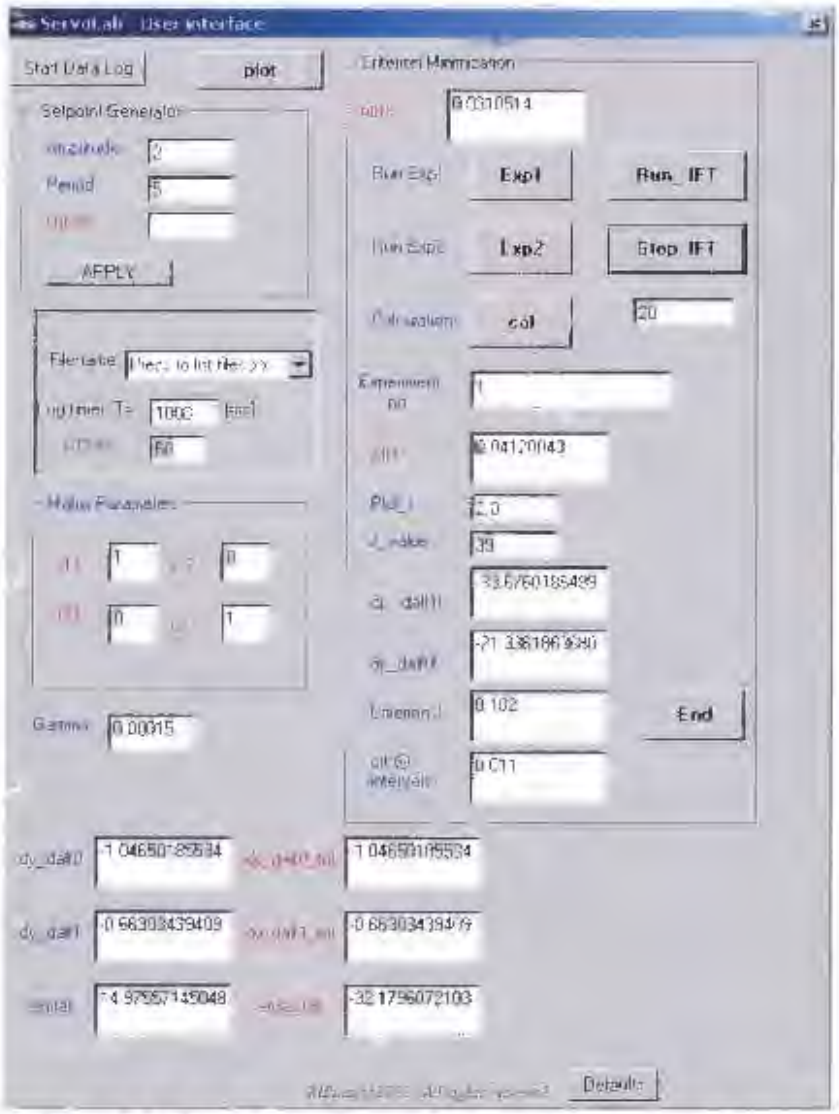

Figure 5.13: The Graphic user interface for the Visual Basic code

Figure 5.13 shows the Graphic User Interface (GUI) for the Visual Basic code for the IFT algorithm with all the parameters inputs depending on the user, it also shows and plots the relevant signals at every iteration.

5.5 BOUNDARIES FOR THE CONTROLLER PARAMETERS

The next section determines the domain of the controller parameters σ_0 and σ_1 for which the system is stable, unstable or oscillatory. This domain is predicted theoretically and verified experimentally.

5.5.1 The boundary as defined by the real system experimentally

The IFT algorithm assumes that one does not know the system or plant that is being controlled. For this section of the thesis however it is assumed that the plant is known in order to determine values of the controller parameters σ_0 and σ_1 that will give the best response for the DC motor. The best response is the one that minimizes the quadratic criterion for the motor. This provides a basis against which to compare the controller parameters produced by the IFT algorithm.

The transfer function model for the servo motor (DC motor) system in open loop was found from step response data.

$$g(s) = \frac{1.01}{(1+2.00s)} \cdot [v]/[v] \qquad (5.2)$$

This transfer function is converted to an equivalent step-invariant pulse-transfer function by z-transformation with a Zero- order Hold. The result is given in terms of the sampling time T_s

$$gh(z) = \frac{1.01 * \left(1 - e^{-\frac{T_s}{2}}\right)}{\left(z - e^{-\frac{T_s}{2}}\right)} / [v]/[v] \qquad (5.3)$$

In the experimental work the sampling time T_s=0.05 seconds

The controller $K(z)$ is given by equation (5.1) and can be modified in order to make the coefficient of the denominator variable z unity as shown below:

$$K(z)=\frac{\sigma_1 * \left(z+\frac{\sigma_0}{\sigma_1}\right)}{(z-1)} \quad \text{..... (5.4)}$$

The open loop pulse transfer function becomes:

$$q(z)=gh(z)*K(z)=\frac{1.01*\left(1-e^{\left(\frac{-T_s}{2}\right)}\right)*\sigma_1*\left(z+\frac{\sigma_0}{\sigma_1}\right)}{(z-1)*\left(z-e^{\left(\frac{-T_s}{2}\right)}\right)} \quad \text{..... (5.5)}$$

And the closed loop characteristic equation is given by:

$$\varphi_c(z)=(z-1)*\left(z-e^{\left(\frac{T_s}{2}\right)}\right)+\gamma*\left(z+\frac{\sigma_0}{\sigma_1}\right) \quad \text{..... (5.6)}$$

where γ is the root locus gain and T_s is the sampling time.

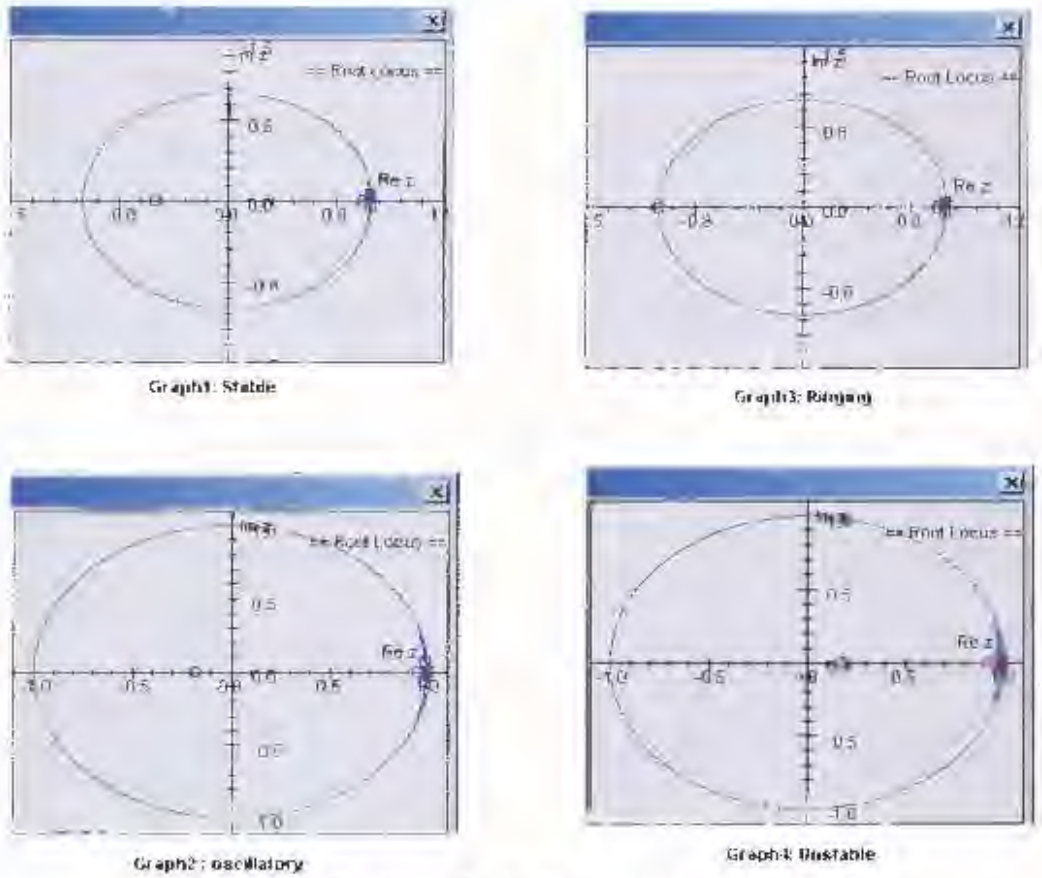

Figure 5.14: Root locus for the controller parameters σ_0 and σ_1

Figure 5.14 shows the four boundaries of the controller parameter σ_0 and σ_1. Graph 1 shows the root locus for $\sigma_0 = 0.03$ and $\sigma_1 = 0.06$ and the system is stable. Graph 2 shows the point where the actually response y starts to be oscillatory and the controller parameters have the value $\sigma_0 = 0.03$ and $\sigma_1 = 0.0165$ at that point. In graph 3 the ringing phenomena that was explained in section 5.3.1 is shown. Lastly in graph 4 if the pole $z = \frac{\sigma_0}{\sigma_1}$ goes to the outside the unit circle of a z - plane the system becomes unstable as will be shown later with equations for IFT.

The controller parameter σ_0 and σ_1 values were found by experiment to lie within the boundaries given in Table 5.3.

Table 5.3: The controller parameter variation

no	σ_0	σ_1	Results
1	0.03	0.03	Ringing
2	0.03	0.06	Stable
3	0.03	0.09	Stable
4	0.03	0.12	Stable
5	0.03	0.15	Stable
6	0.03	0.165	Stable
7	All values of σ_0	$\sigma_1 < \sigma_0$	Unstable
8	$+\backslash -\sigma_0$	$-\backslash +\sigma_1$	Unstable

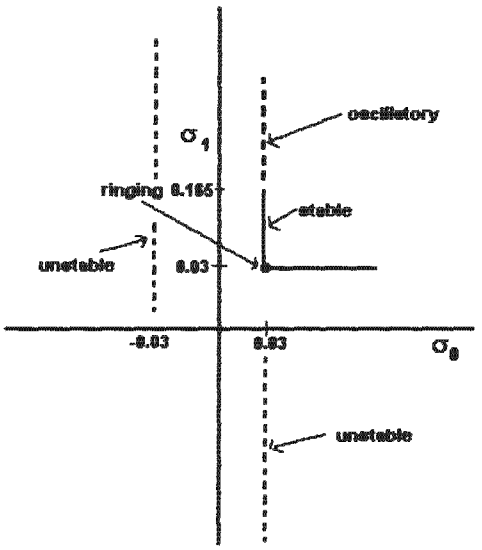

Figure 5.15: The controller parameter boundary

Figure 5.15 shows the regions and boundaries for the controller parameters σ_0 and σ_1 used in the DC motor experiment. The zero at $-\frac{\sigma_0}{\sigma_1}$ for the type 1 controller given by equation (5.4) must be inside the unit circle meaning that $\sigma_1 > \sigma_0$, since this is a zero for the controller K it is also a pole for the gradient of the output and input given by

equation (3.20), (3.21), (3.22) and (3.23). The IFT system starts to be oscillatory when $\sigma_1 > 0.165$ as this is the region the DC motor system does not operate properly.

The next section determines the domain for the controller parameters by theory and simulation

5.5.2 The boundary as defined by theory and simulation

The diagram for the IFT control system is repeated in figure 5.16

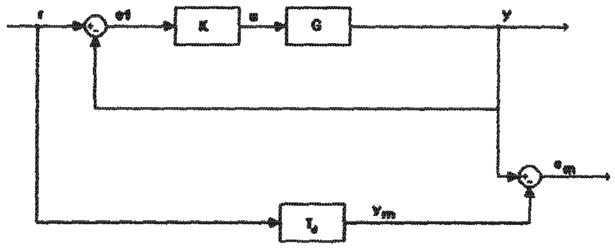

Figure 5.16: Block diagram of a one degree of freedom controller in a closed loop system

Let the type 1 controller be defined by the transfer function:

$$K(s) = \frac{\beta_0 + \beta_1 * s}{s} \quad \text{.........} (5.7)$$

where β_0 and β_1 are controller parameters in the $s-plane$

Let $k_1(z)$ be the $z-tranform$ of $K(s)$ in equation (5.7)

$$k_1(z) = \frac{(\beta_0 + \beta_1) * z - \beta_1}{(z-1)} \quad \text{.........} (5.8)$$

By comparing equation (5.8) and (5.1) the following equivalences between the parameters of the two controllers can be defined:

$$\sigma_0 = -\beta_1 \quad \text{.........} (5.9)$$

$$\sigma_1 = \beta_0 + \beta_1 \quad \text{.........} (5.10)$$

The error e_m in equation (4.6) has to be driven to zero for the actual controller parameters to satisfy the condition for which the quadratic criterion J is a minimum. This means that the transfer function of the closed loop system must be equal to that of the desired model:

$$\frac{G*K}{1+G*K} = \frac{a_m}{1+s*T_m} \quad \text{..... (5.11)}$$

where $G = \frac{a}{1+s*T}$ and $G*K = \frac{a*\beta_0 + a*\beta_1*s}{s*(1+s*T)}$ Hence

$$\frac{a*\beta_0 + a*\beta_1*s}{s*(1+s*T)+a*\beta_0 + a*\beta_1*s} = \frac{a_m}{1+s*T_m} \quad \text{..... (5.12)}$$

Cross multiplying the denominator and numerator of equation (5.12) gives:

$$(a*\beta_0 + a*\beta_1*s)*(1+s*T_m) = a_m*(s*(1+s*T)+a*\beta_0 + a*\beta_1*s) \quad \text{..... (5.13)}$$

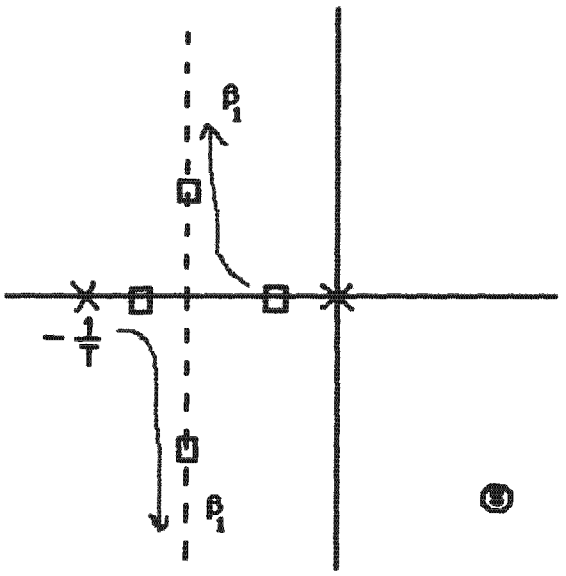

Figure 5.17: The root locus of the actual closed loop system

Comparing the coefficient of s the following equations can be derived from equation (5.12)

$$\beta_0 = \frac{1}{a * T_m} \quad \text{.....(5.14)}$$

$$\beta_1 = \frac{T}{a * T_m} \quad \text{.....(5.15)}$$

For the actual system the time constant $T = 2$ seconds and the gain $a = 1.01$, while the time constant of the desired system $T_m = 1$ second hence:

$$\beta_0 = 0.99 \quad \text{.....(5.16)}$$

$$\beta_1 = 1.98 \quad \text{.....(5.17)}$$

By substituting equation (5.16) and (5.17) into (5.9) and (5.10) yields the following values for the controller parameters sigma in the $z - plane$:

$$\sigma_0 = -1.98 \quad \text{.....(5.18)}$$

$$\sigma_1 = 2.97 \quad \text{.....(5.19)}$$

The pole for the input and output gradient with respect to the controller parameter sigma σ on the $z - plane$ is at $z = -0.667$,and this condition is explained in line 8 of table 5.3 as unstable.

5.6 THE IFT ALGORITHM WHEN R_j IS NOT AN IDENTITY MATRIX

For this experiment the matrix R_j is defined as:

$R_j = \frac{1}{N}\sum_{t=1}^{N}\left(est\left[\frac{\partial y}{\partial \sigma}(\sigma_j)\right] est\left[\frac{\partial y}{\partial \sigma}(\sigma_j)\right]^T + \lambda * est\left[\frac{\partial u}{\partial \sigma}(\sigma_j)\right] est\left[\frac{\partial u}{\partial \sigma}(\sigma_j)\right]^T \right)$ as stated in equation (A2.30)

$$est\left[\frac{\partial y}{\partial \sigma}(\sigma_j)\right] = \begin{bmatrix} \frac{\partial y}{\partial \sigma_0} \\ \frac{\partial y}{\partial \sigma_1} \end{bmatrix} \quad \text{.....(5.20)}$$

$$est\left[\frac{\partial u}{\partial \sigma}(\sigma_j)\right] = \begin{bmatrix} \frac{\partial u}{\partial \sigma_0} \\ \frac{\partial u}{\partial \sigma_1} \end{bmatrix} \quad \text{..... (5.21)}$$

These equations come from equation (4.23), (4.22), (4.20) and (4.21) respectively.

The first term in the matrix R_j works out to be:

$$\begin{bmatrix} \frac{\partial y}{\partial \sigma_0} \\ \frac{\partial y}{\partial \sigma_1} \end{bmatrix} * \begin{bmatrix} \frac{\partial y}{\partial \sigma_0} & \frac{\partial y}{\partial \sigma_1} \end{bmatrix} = \begin{bmatrix} \left(\frac{\partial y}{\partial \sigma_0}\right)^2 & \left(\frac{\partial y}{\partial \sigma_0} * \frac{\partial y}{\partial \sigma_1}\right) \\ \left(\frac{\partial y}{\partial \sigma_1} * \frac{\partial y}{\partial \sigma_0}\right) & \left(\frac{\partial y}{\partial \sigma_1}\right)^2 \end{bmatrix} \quad \text{..... (5.22)}$$

Similarly the second term in the matrix R_j works out to be:

$$\lambda * est\left[\frac{\partial u}{\partial \sigma}(\sigma_j)\right] * est\left[\frac{\partial u}{\partial \sigma}(\sigma_j)\right]^T = \begin{bmatrix} \lambda * \left(\frac{\partial u}{\partial \sigma_0}\right)^2 & \lambda * \left(\frac{\partial u}{\partial \sigma_0} * \frac{\partial u}{\partial \sigma_1}\right) \\ \lambda * \left(\frac{\partial u}{\partial \sigma_1} * \frac{\partial u}{\partial \sigma_0}\right) & \lambda * \left(\frac{\partial u}{\partial \sigma_1}\right)^2 \end{bmatrix} \quad \text{..... (5.23)}$$

Hence the elements of the matrix R_j are defined by the following scalar equations:

$$r_{11} = \frac{1}{N}\sum_{t=1}^{N}\left(\left(\frac{\partial y}{\partial \sigma_0}\right)^2 + \lambda * \left(\frac{\partial u}{\partial \sigma_0}\right)^2\right) \quad \text{..... (5.24)}$$

$$r_{12} = \frac{1}{N}\sum_{t=1}^{N}\left(\left(\frac{\partial y}{\partial \sigma_0} * \frac{\partial y}{\partial \sigma_1}\right) + \lambda * \left(\frac{\partial u}{\partial \sigma_0} * \frac{\partial u}{\partial \sigma_1}\right)\right) \quad \text{..... (5.25)}$$

$$r_{21} = \frac{1}{N}\sum_{t=1}^{N}\left(\left(\frac{\partial y}{\partial \sigma_1} * \frac{\partial y}{\partial \sigma_0}\right) + \lambda * \left(\frac{\partial u}{\partial \sigma_1} * \frac{\partial u}{\partial \sigma_0}\right)\right) \quad \text{..... (5.26)}$$

$$\Lambda \; \sum_{i=1}^{N}\left((\sigma\sigma_1)^2 \qquad (\partial\sigma_1)^2\right) \quad \text{..... (5.27)}$$

$$R_j = \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix} \text{..... (5.28)}$$

5.6.1 Results from the IFT algorithm for general R_j matrix

Once matrix R_j has been computed from data collected during initial experiments on the motor system, it can be incorporated in to the IFT algorithm. The matrix R_j for this experiment was found to be:

$$R_j = \begin{bmatrix} 495.92 & 370.62 \\ 370.62 & 276.97 \end{bmatrix} \text{..... (5.29)}$$

When observing this matrix's parameters, it is clear that the off-diagonal entities r_{12} and r_{21} have the same value. The reason behind this is that in this experiment one is dealing with scalars, hence the dot product of the term are equal $\left(\frac{\partial y}{\partial \sigma_0} * \frac{\partial y}{\partial \sigma_1}\right) = \left(\frac{\partial y}{\partial \sigma_1} * \frac{\partial y}{\partial \sigma_0}\right)$ where else if it were vectors the result is not necessarily true.

The results produced by the IFT algorithm with this matrix R_j are shown in figure 5.18

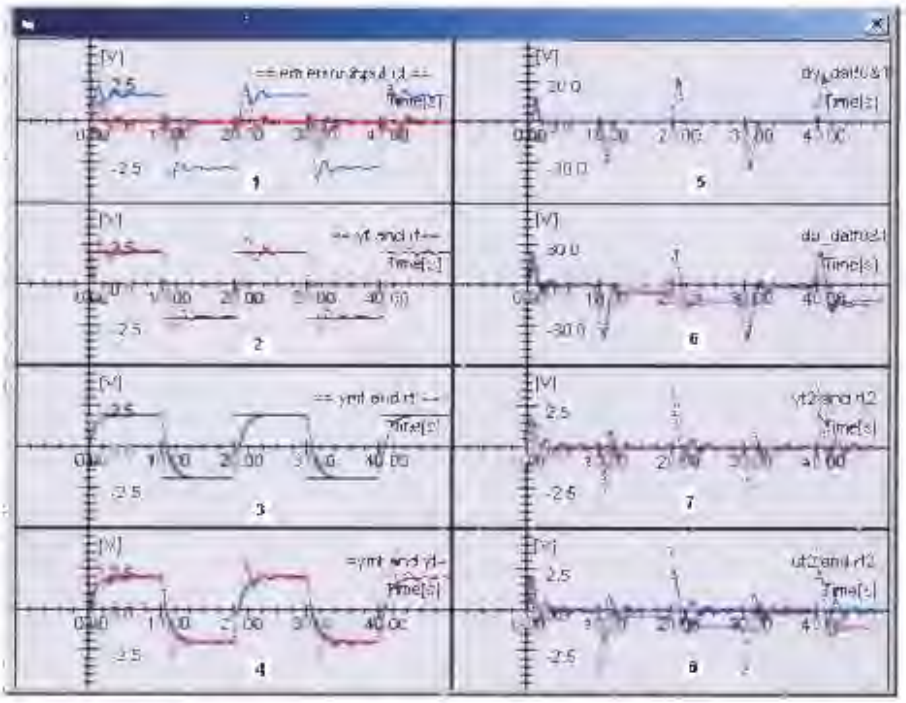

Figure 5.18. The IFT algorithm results when the R matrix is defined by the equation (A2.30) in appendix 2.

The signals in figure 5.18 look similar to those obtained using the identity matrix R_j given by equation (3.1) and shown in figure 5.2 for experiment 1 and figure 5.10 for experiment 2 , this means that for the DC motor problem the identity matrix worked as well as the R_j defined by equation (A2.30) in appendix 2. The literature stated [14] that this is not always the case for more complex problems.

Chapter 6

Simulation

The next section of the dissertation simulates the one degree of freedom IFT controller using Simulink and the results are compared to those obtained through experiments in chapter 5

6.1 SIMULATION OF IFT FOR ONE DEGREE OF FREEDOM CONTROLLER

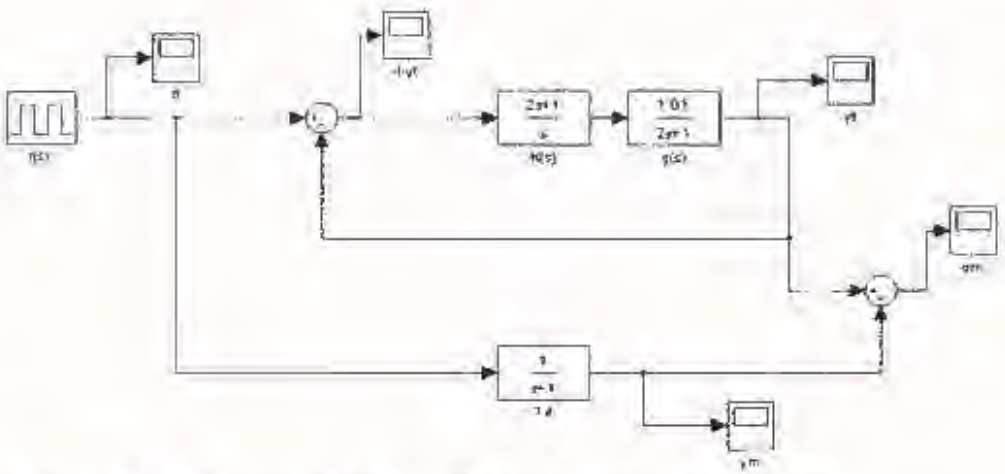

Figure 6.1. The simulation figure for one degree of freedom controller

Figure 6.1 shows the circuit for the simulation of a one degree of freedom controller for the first step of the IFT algorithm.

Figure 6.2: The simulation for the setpoint rt

Figure 6.2: shows the setpoint r used for the simulation as stated in the figure the amplitude is 1 unit and the period is 40 seconds. The simulation period is twice that of the experiment. The reason behind this is that with the simulation there is no limitation to the variables and 40 seconds is long enough for the signals to settle.

Figure 6.3: The simulation of the actual response y

Figure 6.3 shows the actual output response y as predicted by the simulation program, the time constant is 2.5 seconds.

Figure 6.4: The simulation of the desired output ym

Figure 6.4 shows the desired output response y_m as predicted by the Simulink program. It has the time constant of 2.5 seconds.

Figure 6.5: The simulated error em

Figure 6.6: The simulation of the error em

Figure 6.5 and figure 6.6 shows two simulated graphs for the error e_m which is the difference between the actual output response y and the desired output response y_m. The difference between the two graphs is the scaling for the y - *axis*, with the graph on the right zoom to scale. The error e_m is approximately zero and overshoots either to 0.004 or to -0.004 whenever the setpoint r_i changes.

Figure 6.7: The simulated input signal u

Figure 6.7 shows the input signal u_i as simulated by the program.

Figure 6.8: The simulated r-y1 signal

The signal simulated in figure 6.8 is generated in experiment 1 and used as a setpoint $r^{(2)}$ in experiment 2.

6.2 DISCUSSION OF THE SIMULATION RESULTS

The following section discusses the actual experimental results with the ones obtained from the simulation using Simulink.

- The actual output response $y^{(1)}$ generated from experiment 1 of the IFT algorithm shown in figure 5.2 graph 2 had a slight overshoot of 8% which is not found by the simulated figure 6.4. This observation led to a study of the effect of the electric time constant of the DC motor as presented in section 6.3 below.

- The overshoot experienced in the IFT algorithm is far less than that produced by using the MRAC gradient approach method in chapter 3. The reason for this is that the IFT algorithm eliminates disturbances as explained in appendix 2. These disturbances include the offsets on the input and output variables that were shown to be the cause of the overshoot in section 3.2.5

6.3 THE APPROXIMATION OF THE REAL SYSTEM

The simulation shown in section 6.1 takes the model or transfer function of the DC motor as:

$$g(s) = \frac{A}{(1 + s * T_m)} \quad \text{..... 6.1}$$

Where A is the gain and T_m is the mechanical time constant of the DC motor. The real transfer function of the DC motor is as follows:

$$g(s) = \frac{A}{(1 + s * T_m)(1 + s * T_e)} \quad \text{..... 6.2}$$

Where T_e is the electric time constant. For the simulation it was assumed that the electric time constant T_e is very small hence the transfer function in equation (6.1). The following section shows the simulation when the electric time constant is taken into account.

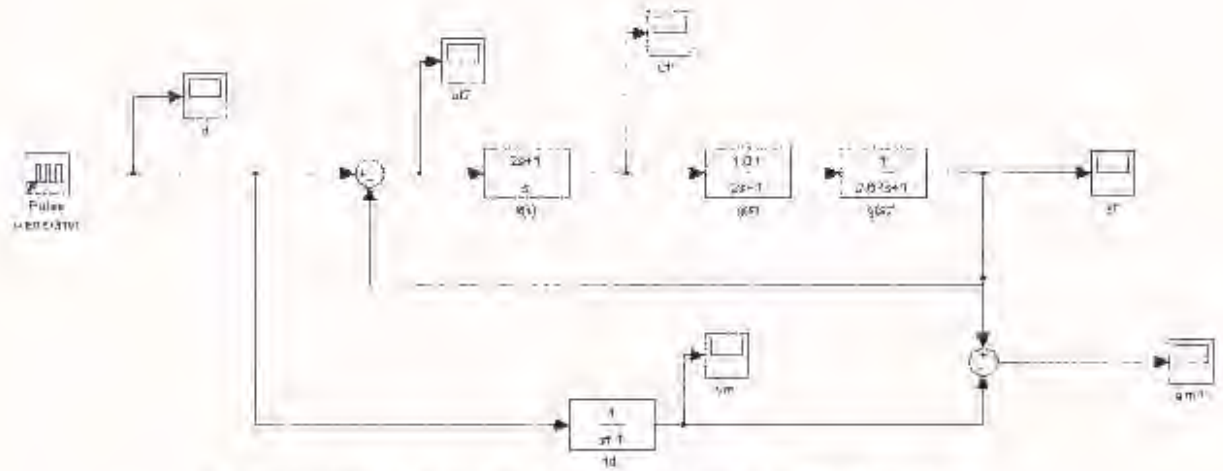

Figure 6.9: The simulation figure of the JFT one degree of freedom controller

Figure 6.9 shows the simulation circuit when the electric time constant T_e is taken into account. The mechanical time constant T_m is greater than T_e, therefore the simulation has the following mathematical expression for electric time constant:

$$T_e = \frac{T_m}{x} \quad \text{..... 6.3}$$

where x is positive scalar

The following figures shows simulation from Simulink when T_e is taken into account.

Figure 6.10: The simulation of the actual output response y

The simulation in figure 6.10 took place under the following condition:

$$T_e = \frac{T_m}{3} \quad \ldots\ldots 6.4$$

Figure 6.10 shows the actual output response experienced an overshoot of approximately 10%, therefore it could be assumed that the electric time constant contributed to the overshoot of y shown in graph 2 figure 5.2.

Figure 6.11: The simulated error signal e_m

Figure 6.11 shows the error signal e_m with the overshoot which resembles graph 1 figure 5.2.

Chapter 6

Simulation

The next section of the dissertation simulates the one degree of freedom IFT controller using Simulink and the results are compared to those obtained through experiments in chapter 5

6.1 SIMULATION OF IFT FOR ONE DEGREE OF FREEDOM CONTROLLER

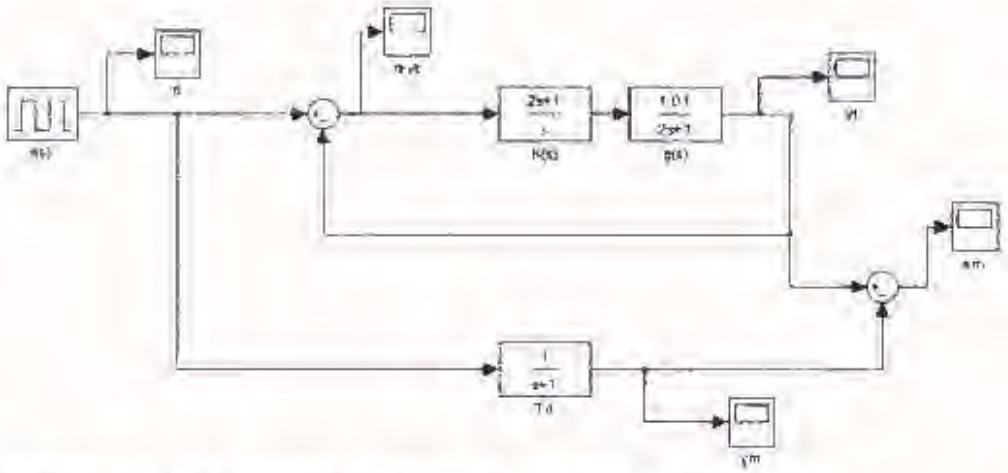

Figure 6.1: The simulation figure for one degree of freedom controller

Figure 6.1 shows the circuit for the simulation of a one degree of freedom controller for the first step of the IFT algorithm.

Figure 6.2: The simulation for the setpoint rt

Figure 6.2: shows the setpoint r used for the simulation as stated in the figure the amplitude is 1 unit and the period is 40 seconds. The simulation period is twice that of the experiment. The reason behind this is that with the simulation there is no limitation to the variables and 40 seconds is long enough for the signals to settle.

Figure 6.3: The simulation of the actual response y

Figure 6.3 shows the actual output response y as predicted by the simulation program, the time constant is 2.5 seconds.

Figure 6.4: The simulation of the desired output ym

Figure 6.4 shows the desired output response y_m as predicted by the Simulink program. It has the time constant of 2.5 seconds.

Figure 6.5: The simulated error em

Figure 6.6: The simulation of the error em

Figure 6.5 and figure 6.6 shows two simulated graphs for the error e_m which is the difference between the actual output response y and the desired output response y_m. The difference between the two graphs is the scaling for the $y-axis$, with the graph on the right zoom to scale. The error e_m is approximately zero and overshoots either to 0.004 or to -0.004 whenever the setpoint r changes.

Figure 6.7: The simulated input signal u

Figure 6.7 shows the input signal u, as simulated by the program.

Figure 6.8: The simulated r-y1 signal

The signal simulated in figure 6.8 is generated in experiment 1 and used as a setpoint $r^{(2)}$ in experiment 2.

6.2 DISCUSSION OF THE SIMULATION RESULTS

The following section discusses the actual experimental results with the ones obtained from the simulation using Simulink.

- The actual output response $y^{(1)}$ generated from experiment 1 of the IFT algorithm shown in figure 5.2 graph 2 had a slight overshoot of 8% which is not found by the simulated figure 6.4. This observation led to a study of the effect of the electric time constant of the DC motor as presented in section 6.3 below.

- The overshoot experienced in the IFT algorithm is far less than that produced by using the MRAC gradient approach method in chapter 3. The reason for this is that the IFT algorithm eliminates disturbances as explained in appendix 2. These disturbances include the offsets on the input and output variables that were shown to be the cause of the overshoot in section 3.2.3

6.3 THE APPROXIMATION OF THE REAL SYSTEM

The simulation shown in section 6.1 takes the model or transfer function of the DC motor as:

$$g(s) = \frac{A}{(1 + s * T_m)} \quad \text{..... 6.1}$$

Where A is the gain and T_m is the mechanical time constant of the DC motor. The real transfer function of the DC motor is as follows:

$$g(s) = \frac{A}{(1 + s * T_e)(1 + s * T_m)} \quad \text{..... 6.2}$$

Where T_e is the electric time constant. For the simulation it was assumed that the electric time constant T_e is very small hence the transfer function in equation (6.1). The following section shows the simulation when the electric time constant is taken into account.

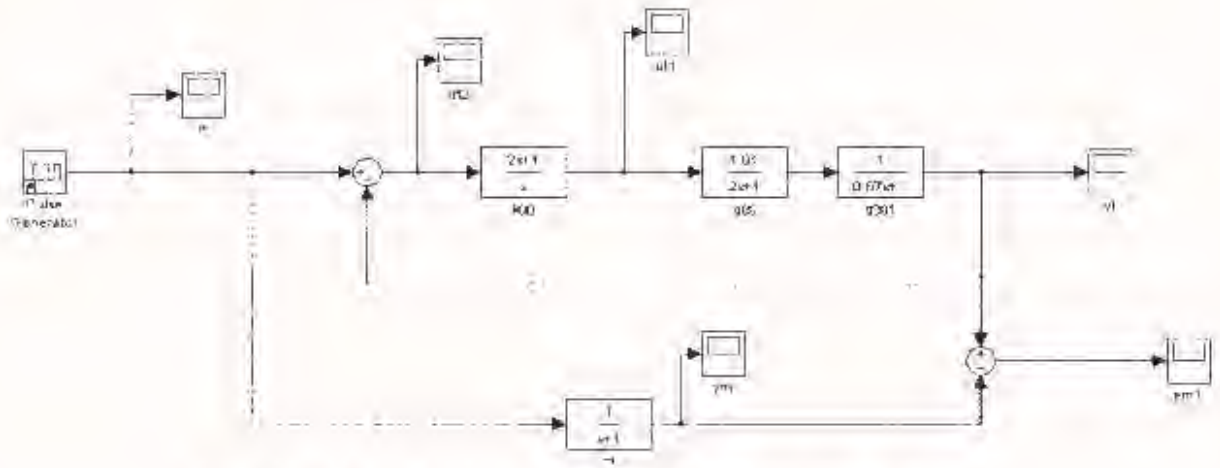

Figure 6.9: The simulation figure of the IFT one degree of freedom controller

Figure 6.9 shows the simulation circuit when the electric time constant T_e is taken into account. The mechanical time constant T_m is greater than T_e therefore the simulation has the following mathematical expression for electric time constant.

$$T_e = \frac{T_m}{x} \quad \text{..... 6.3}$$

where x is positive scalar

The following figures shows simulation from Simulink when T_e is taken into account:

Figure 6.10: The simulation of the actual output response y

The simulation in figure 6.10 took place under the following condition:

$$T_e = \frac{T_m}{3} \quad \ldots\ldots 6.4$$

Figure 6.10 shows the actual output response experienced an overshoot of approximately 10%, therefore it could be assumed that the electric time constant contributed to the overshoot of y shown in graph 2 figure 5.2.

Figure 6.11: The simulated error signal e_{ss}

Figure 6.11 shows the error signal e_{ss} with the overshoot which resembles graph 1 figure 5.2.

Chapter 7

Validation of the Iterative Feedback Tuning Results

The purpose of this chapter is to investigate in detail the quadratic cost function J of the one degree of freedom IFT in order to find the controller parameters for which the cost function is a minimum. The inertia of the load for the DC motor is also varied to ensure that the IFT is able to tune the controller parameters on-line in order to compensate for process changes due caused by different loads.

7.1 THE CONVERGENCE OF THE PARAMETER TO LOCAL OR GLOBAL MINIMA

When dealing with a problem of optimization, it is vital that the optimization algorithm is started with sensible initial parameters that in this case mean parameters that are within the correct region or domain in which the function optimized. Two methods are used here to find or search for the controller parameters for which the quadratic criterion is a minimum.

- The first search is done manual taking into account the stable region of the DC motor explained in chapter 5
- The second search is done automatically where by the Visual Basic code searches the entire region to find the controller parameters for which the quadratic criterion is a minimum.

In both cases the actual DC motor system is used and it is anticipated that the inherent noise and uncertainty will cause deviations from the ideals of the basic theory.

7.2 FINDING THE CRITERION MINIMUM

In order to find through manual search methods the parameter point where the cost function is a minimum, the following procedure was carried out on the DC motor system:

- Keep one controller parameter constant and vary the other until the minimum of the quadratic criterion is found in that axis direction. For example keep σ_0 constant and vary σ_1.

- When the controller parameter that is being varied reaches a point where the quadratic criterion is a minimum, then it is kept at that point and the other parameter, which was kept constant is varied along its axis until the minimum is found. For example σ_1 is now kept constant and σ_0 is varied.

- When the second controller parameter that is being varied reaches the minimum for the quadratic criterion, then the two points are taken as the parameter values for which the quadratic criterion surface is close to its minimum. (Further searching would be needed to find the actual optimum parameter point but was not carried out as the approximate position is assumed to be close enough to the optimum to allow a check on the results of the automated search.)

7.2.1 Manual tuning of the controller parameters for criterion minimization

The search for the criterion minimization's parameters is done manual in this section, one parameter is increased by a constant increment until the lowest value of the criterion J is obtained. The same steps are done to obtain the other parameter. This manual method provides a rough indication of the position of the cost function minimum in the sigma parameter space.

Table 7.1 Shows the results for the quadratic criterion J when the controller parameter σ_1 is varied and σ_0 is kept constant:

Number	Sigma 0 σ_0	Sigma 1 σ_1	Criterion J
1	0.020	0.022	0.1172
2	0.020	0.026	0.1100
3	0.020	0.028	0.0920
4	0.020	0.032	0.0880
5	0.020	0.036	0.07670
6	0.020	0.038	0.05970
7	0.020	0.042	0.07650
8	0.020	0.044	0.07470
9	0.020	0.046	0 08000
10	0.020	0.048	0.08050
11	0.020	0.050	0.09050

The results for the experiment shown in table 7.1 indicated $\sigma_1 = 0.038$ as the value for which the quadratic criterion is a minimum in the σ_1 axis when $\sigma_0 = 0.020$. The corresponding value for the criterion is 0.0597 as indicated in row 6 of table 7.1.

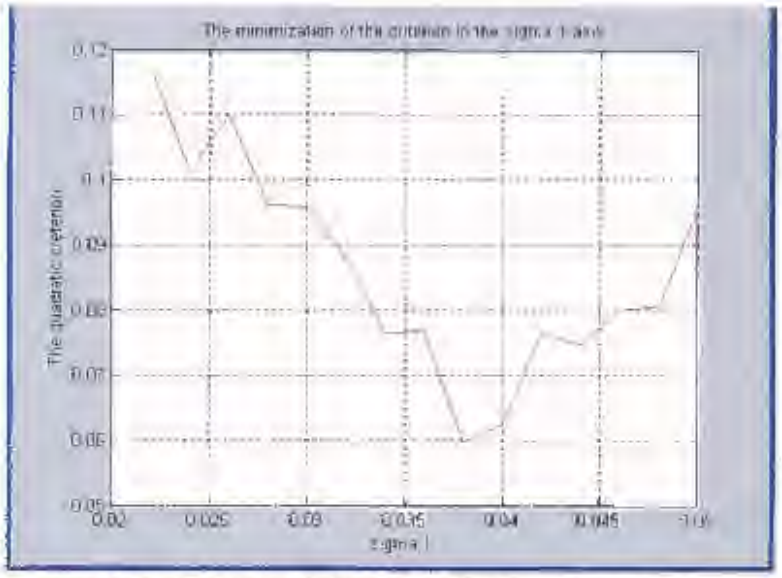

Figure 7.1: The minimization of the quadratic criterion in the σ_1 axis

Figure 7.1 shows the minimization of the quadratic criterion in the σ_1 axis as given by table 7.1. Notice the fluctuation in the quadratic function. This is ascribed to variations in the captured data due to measurement noise and other disturbances that are present in physical system.

The next section finds the value of the controller parameter in the σ_0 axis for which the criterion is a minimum when $\sigma_1 = 0.038$. The results for the quadratic criterion J are given in Table 7.2:

Number	Sigma 0 σ_0	Sigma 1 σ_1	Criterion J
1	0.022	0.038	0.0540
2	0.024	0.038	0.0495
3	0.026	0.038	0.0425
4	0.028	0.038	0.0500
5	0.030	0.038	0.0540

6	0.032	0.038	0.0550
7	0.034	0.038	0.0623
9	0.036	0.038	0.0760

The results for the experiment shown in table 7.2 indicated $\sigma_0 = 0.026$ as the value for which the quadratic criterion is a minimum in the σ_0 axis, the value for the criterion is 0.0425 as indicated in row 3 of table 7.2. The data shown in figure 7.2 gives an indication of the space of the quadratic criterion in the direction of the σ_0 axis

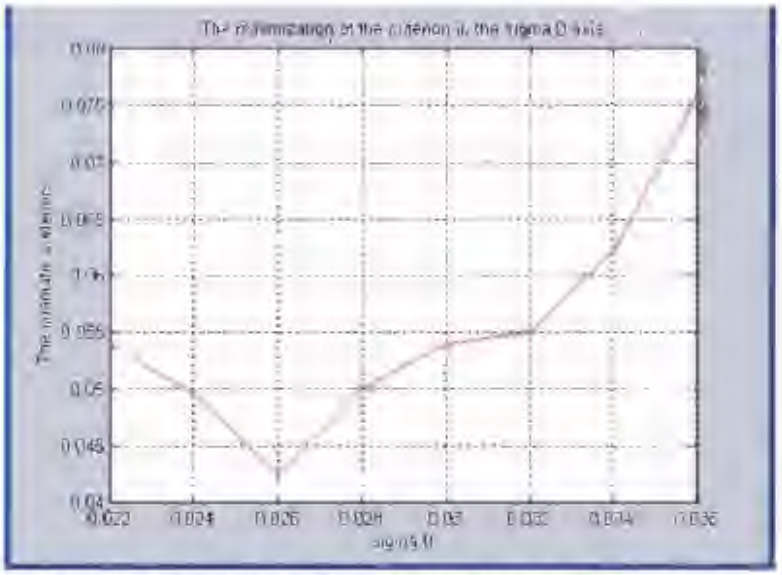

Figure 7.2: The minimization of the quadratic criterion in the σ_0 axis

The following conclusion can be made from these two manual experiments:
For the domain $\sigma_1 = [0.022 \quad 0.05]$ when $\sigma_0 = 0.02$ the minimum for the quadratic criterion was found to lie at $\sigma_1 = 0.038$, and for the domain $\sigma_0 = [0.022 \quad 0.036]$ when $\sigma_1 = 0.038$ the minimum for the quadratic criterion was found to lie at $\sigma_0 = 0.026$.

The domain for σ_0 is determined by the fact that $\sigma_1 > \sigma_0$ as explained in table 5.3. The values for the controller parameters for which the quadratic criterion is a minimum, for a the following given domain $\sigma = [0.022 \quad 0.05]$ for the DC motor given in the following

format $\sigma = [\sigma_0 \quad \sigma_1]$, is $\sigma = [0.026 \quad 0.038]$. The value for the quadratic criterion for this minimum for the given controller parameters is $J = 0.0425$.

7.2.2 Automatic tuning of parameters for criterion minimization

The following section determines automatically the parameters of the criterion minimization. This is done by the Visual Basic code for the IFT algorithm that is given in table 4.1 and table 4.2. The parameters are updated after experiment 1 and experiment 2 are completed. Each experiment runs for 50 seconds in order to ensure that a number of responses to the same setpoint change are averaged in the IFT.

To determine the parameters automatically the following procedure was implemented:

- The code was run without the boundaries stated in section 4.5.1 in order to avoid additional complications due to any nonlinear effects that constraints at such boundaries might cause.
- The starting points for the controller parameters were 10% and 5% away from the value of parameters for which the quadratic criterion is a minimum. This is done to ensure that the Visual Basic code can be able to tune the parameters automatically if they are displaced from the true position.
- One parameter is tuned automatically while the other one is kept constant, the one that is tuned is been displaced by 10% and 5% of its true value. True value meaning the value determined manually at which the criterion is near its minimum. This process is done for both parameters.
- The third stage allows the IFT to tune both controller parameters at the same time

The Visual Basic code for updating for the controller parameters given in table 4.2 is repeated in equation (7.1) and (7.2)

*alf0 = alf0 - (gamaz * r11t * dj_dalf0) / 1000 - gamaz * r12t * (dj_dalf1) / 1000*..... (7.1)
*alf1 = alf1 - (gamaz * r21t * dj_dalf0) / 1000 - gamaz * r22t * (dj_dalf1) / 1000*..... (7.2)

In the equation in (7.1) and (7.2), the only variable that depends on the user's input is the *gamaz* or γ as it is written in equation (3.15) and (3.16). The constant *gamaz* or γ is a positive scalar that sets the rate at which the algorithm will converge. If the value of this scalar is too large, the increments for the updating of the controller parameters will be high and the parameters might enter in the region stated in section 4.3.1 and result in the ringing phenomena or instability. While if the scalar is too small the rate of which the algorithm will converge is too slow. Designers of the IFT parameters have to find the trade-off between the two when determining the value of the scalar λ.

The following section tunes the controller parameter σ_1, while σ_0 is kept constant at 0.0234

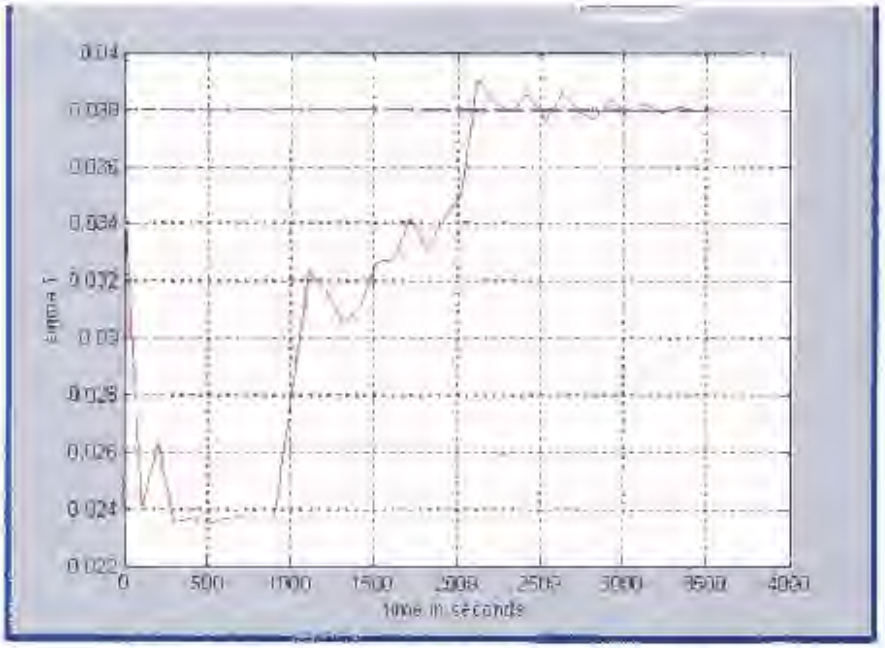

Figure 7.3: The auto tuning of σ_1 with σ_0 kept constant

Figure 7.3 shows the auto tuning of the controller parameter σ_1 with σ_0 kept constant. The starting value for the controller parameter is $\sigma_1 = 0.0342$, this value is 10% less than the value of the controller parameter σ_1 for which the criterion is a minimum. The controller parameter started going the opposite direction to the expected value, which is $\sigma_1 = 0.038$ shown by the blue trace in figure 7.3. The controller parameter started

increasing steadily between $t = [300 \quad 900]$ seconds, but the increase was very slow. The value of the positive scalar γ was increased from $\gamma = 0.00015$ to $\gamma = 0.0015$ at $t = 901$ seconds, that resulted in the increase of the rate at which the controller parameter σ_1 is changing. It took approximately 3500 seconds before the controller parameter σ_1 reached the expected value, after oscillating around the optimal value for 1000 seconds.

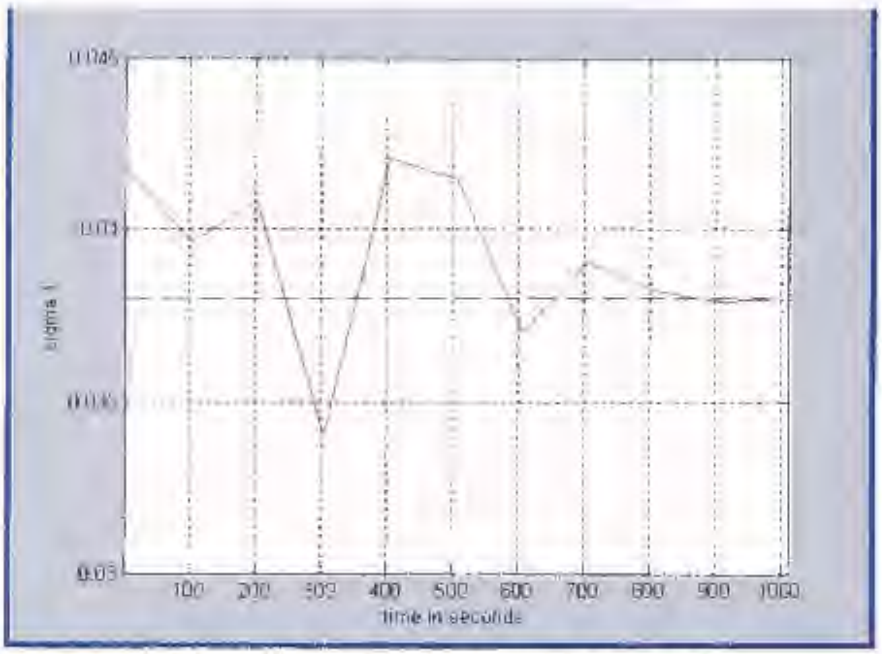

Figure 7.4: The auto tuning of σ_1 with σ_0 kept constant

Figure 7.4 shows the auto tuning of the controller parameter σ_1 with σ_0 kept constant at 0.0286. The starting value for the controller parameter is $\sigma_1 = 0.0418$, this value is 10% more than the value of the controller parameter σ_1 for which the criterion is a minimum. The controller parameter took approximately 1000 seconds to settle after initially oscillating. The reason for the parameter reaching its final value more quickly is that the value of the scalar which is $\gamma = 0.0015$.

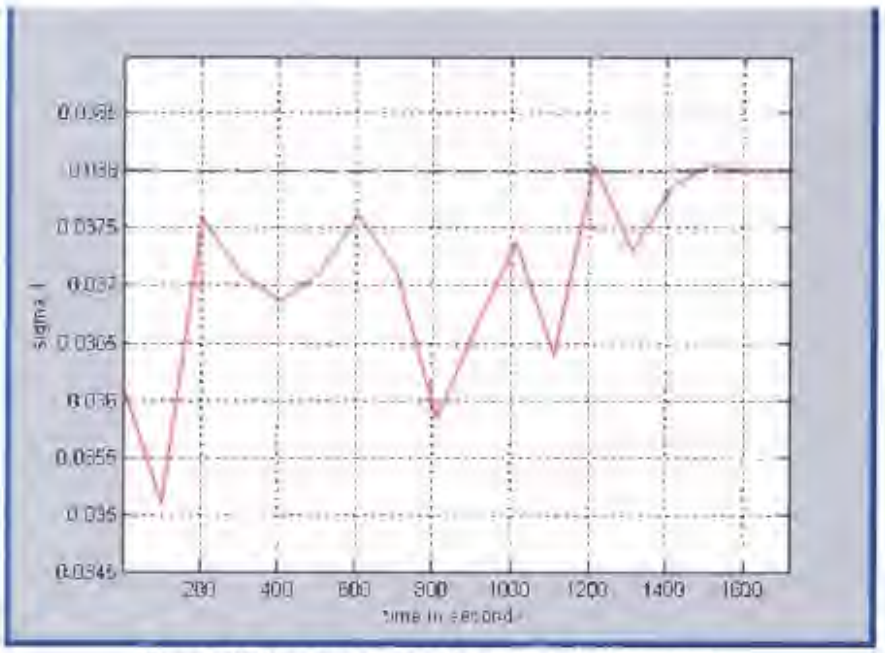

Figure 7.5: The auto tuning of σ_1 with σ_0 kept constant

Figure 7.5 shows the auto tuning of the controller parameter σ_1 with σ_0 kept constant at 0.0247. The starting value for the controller parameter is $\sigma_1 = 0.0361$, this value is 5% less than the value of the controller parameter σ_1 for which the criterion is a minimum. Once again the controller parameter started going the opposite direction to the expected value, which is $\sigma_1 = 0.038$ shown by the blue trace in figure 7.5. The controller parameter started increasing steadily until it reached the expected value of 0.038 after approximately 1600 seconds. The value for the positive scalar is $\gamma = 0.0015$.

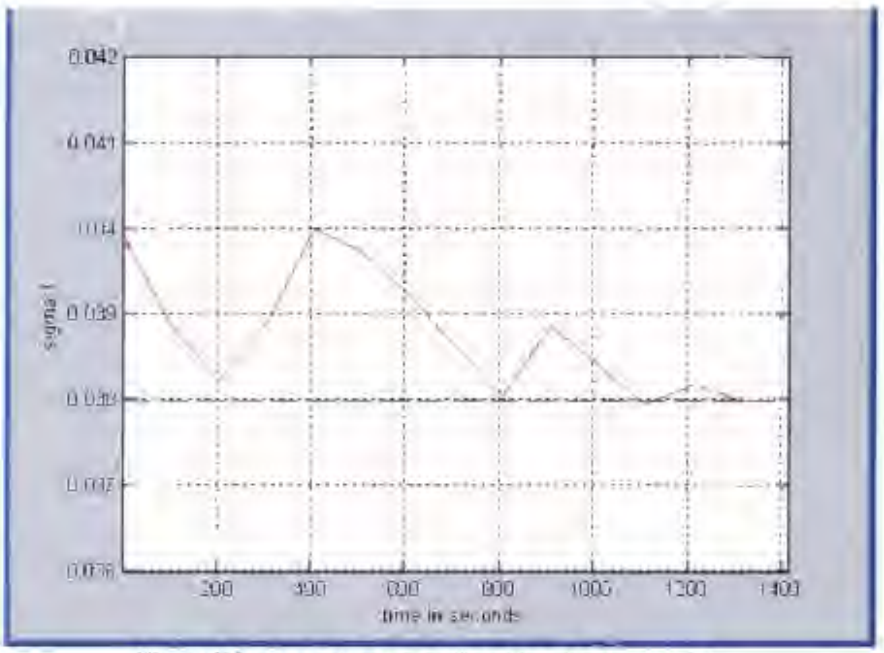

Figure 7.6: The auto tuning of σ_1 with σ_0 kept constant

Figure 7.6 shows the auto tuning of the controller parameter σ_1 with σ_0 kept constant at 0.0273. The starting value for the controller parameter is $\sigma_1 = 0.0399$, this value is 5% more than the value of the controller parameter σ_1 for which the criterion is near its minimum. The controller parameter started oscillating and decreasing until it reached the expected value of $\sigma_1 = 0.038$ shown by the blue trace in figure 7.6. It took approximately 1400 seconds to settle. The value for the positive scalar is $\gamma = 0.0015$.

The next section tunes the controller parameter σ_0, while σ_1 is kept constant.

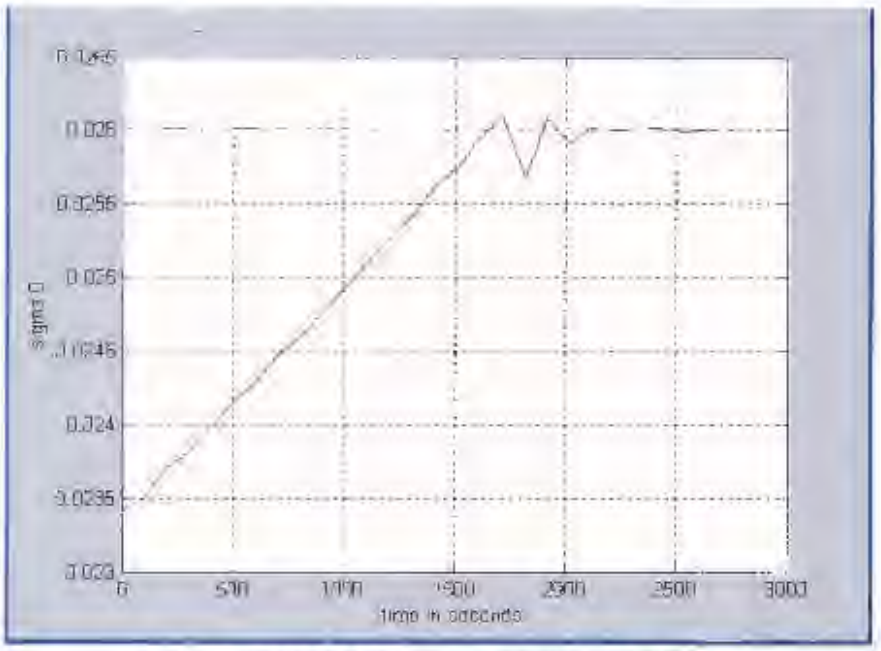

Figure 7.7: The auto tuning of σ_0 with σ_1 kept constant

Figure 7.7 shows the auto tuning of the controller parameter σ_0 with σ_1 kept constant at 0.0342. The starting value for the controller parameter is $\sigma_0 = 0.0234$, this value is 10% less than the value of the controller parameter σ_0 for which the criterion is a minimum. The controller parameter started increasing steadily toward the expected value of $\sigma_0 = 0.026$ shown by the blue trace in figure 7.7. It took approximately 3000 seconds to settle. The value for the positive scalar is $\gamma = 0.0003$. The value of γ was increased to increase the rate of change of the controller parameter.

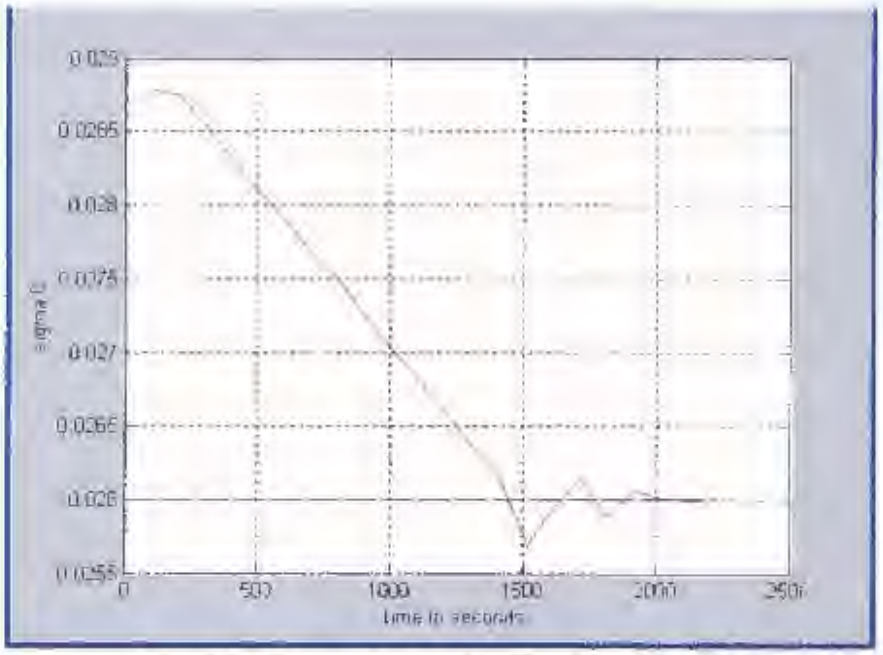

Figure 7.8: The auto tuning of σ_0 with σ_1 kept constant

Figure 7.8 shows the auto tuning of the controller parameter σ_0 with σ_1 kept constant at 0.0418. The starting value for the controller parameter is $\sigma_0 = 0.0286$, this value is 10% less than the value of the controller parameter σ_0 for which the criterion is near its minimum. The controller parameter started decreasing steadily toward the expected value of $\sigma_1 = 0.026$ shown by the blue trace in figure 7.8. It took approximately 2000 seconds to settle. The value for the positive scalar is $\gamma = 0.0003$.

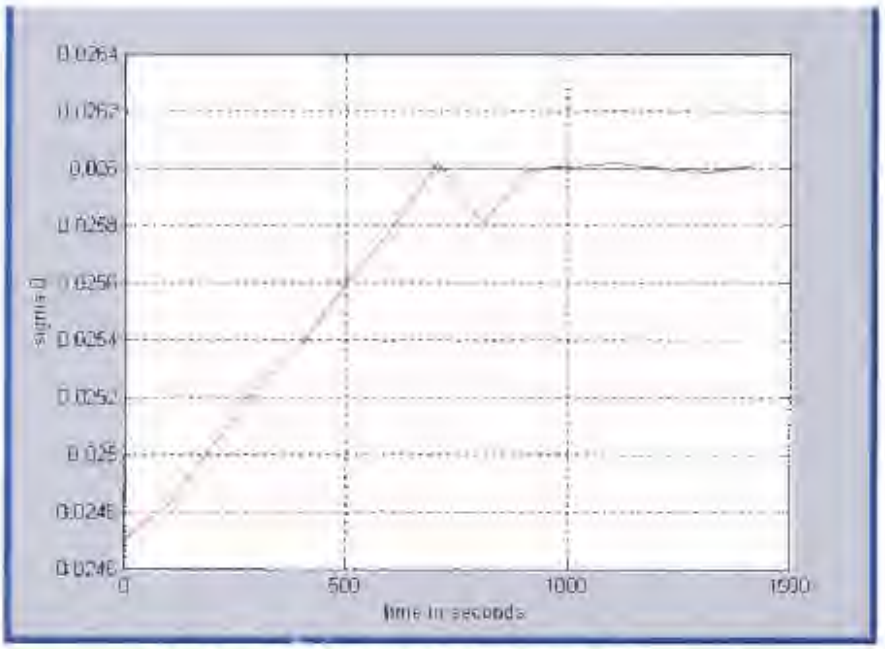

Figure 7.9: The auto tuning of σ_0 with σ_1 kept constant

Figure 7.9 shows the auto tuning of the controller parameter σ_0 with σ_1 kept constant at 0.0361. The starting value for the controller parameter is $\sigma_0 = 0.0247$, this value is 5% less than the value of the controller parameter σ_0 for which the criterion is near its minimum. The controller parameter started increasing steadily toward the expected value of σ_1 0.026 shown by the blue trace in figure 7.9. It took approximately 1500 seconds to settle. The value for the positive scalar is $\gamma = 0.0003$.

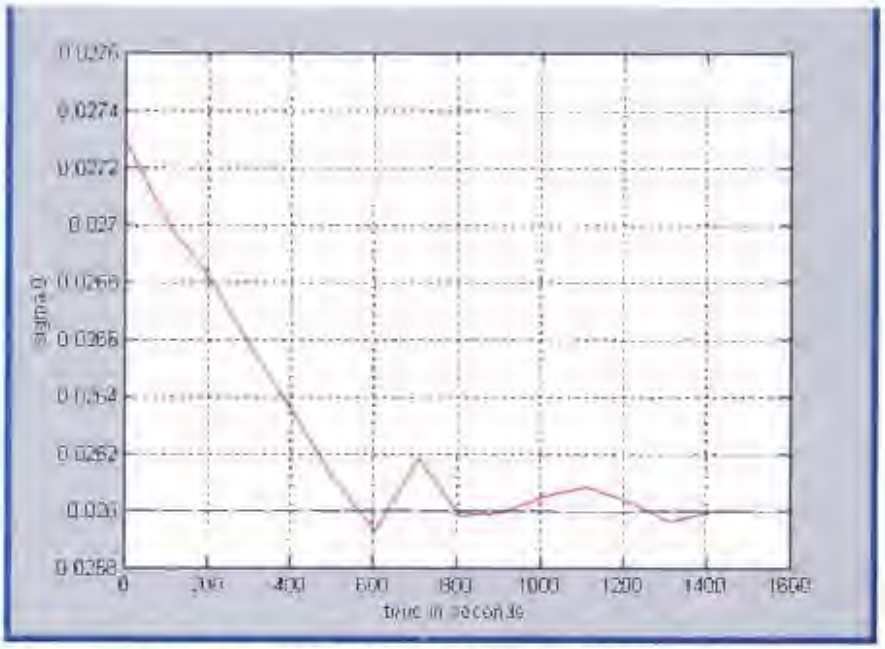

Figure 7.10. The auto tuning of σ_0 with σ_1 kept constant

Figure 7.10 shows the auto tuning of the controller parameter σ_0 with σ_1 kept constant at 0.0399. The starting value for the controller parameter is $\sigma_{0i} = 0.0273$, this value is 5% more than the value of the controller parameter σ_{0} for which the criterion is a minimum. The controller parameter started decreasing steadily toward the expected value of $\sigma_0 = 0.026$ shown by the blue trace in figure 7.8. It took approximately 1500 seconds to settle. The value for the positive scalar is $\gamma = 0.0003$.

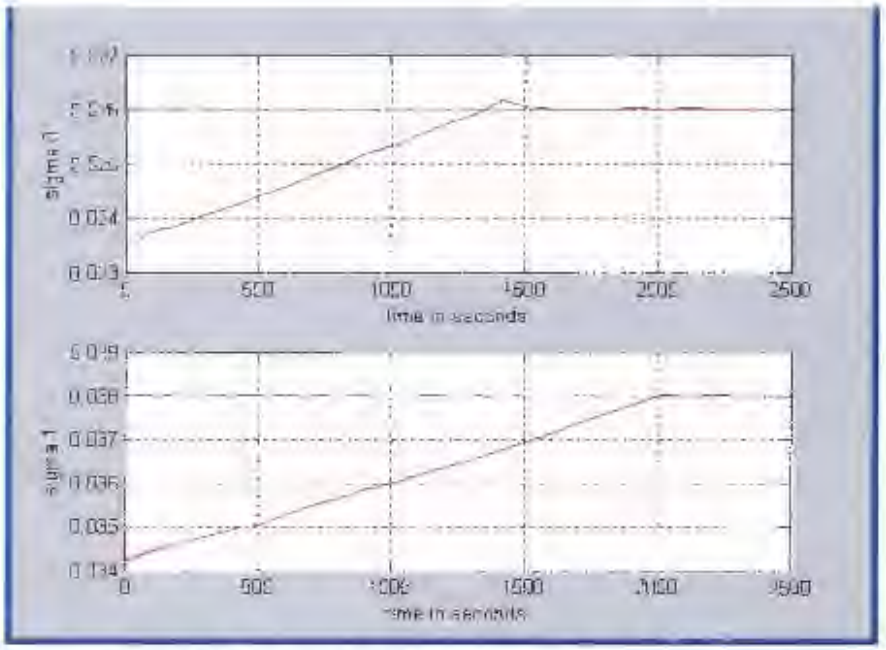

Figure 7.11: The auto tuning of both σ_0 and σ_1 simultaneously

Figure 7.11 shows the automatic tuning of the both controller parameters simultaneously, the starting values for the controller parameters was 10% less than the expected value for which the quadratic criterion is near its minimum. Both controller parameters increased steadily until the expected value, as illustrated in figure 7.11 that the time for the controller parameters to reach their respective expected values is not necessarily the same. It took σ_0 approximately $t = 1500$ seconds to reach the expected value of $\sigma_0 = 0.026$, while σ_1 took approximately $t = 2000$ seconds to reach the expected value of $\sigma_1 = 0.038$. The value for the positive scalar is $\gamma = 0.0003$.

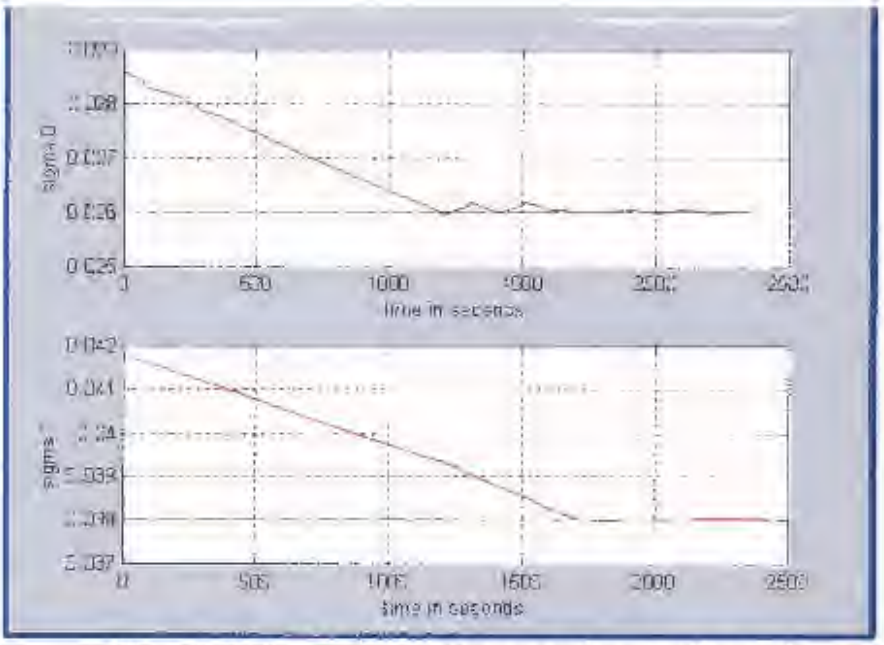

Figure 7.12: The auto tuning of both σ_0 and σ_1 simultaneously

Figure 7.12 shows the automatic tuning of the both controller parameters simultaneously. the starting values for the controller parameters was 10% more than the expected value for which the quadratic criterion is near its minimum. Both controller parameters decreased steadily until the expected value, as illustrated in figure 7.12 that the time for the controller parameters to reach their respective expected values is not necessarily the same. It took σ_0 approximately $t = 1700$ seconds to reach the expected value of $\sigma_0 = 0.026$, while σ_1 took approximately $t = 1900$ seconds to reach the expected value of $\sigma_1 = 0.038$. The value for the positive scalar is $\gamma = 0.0003$.

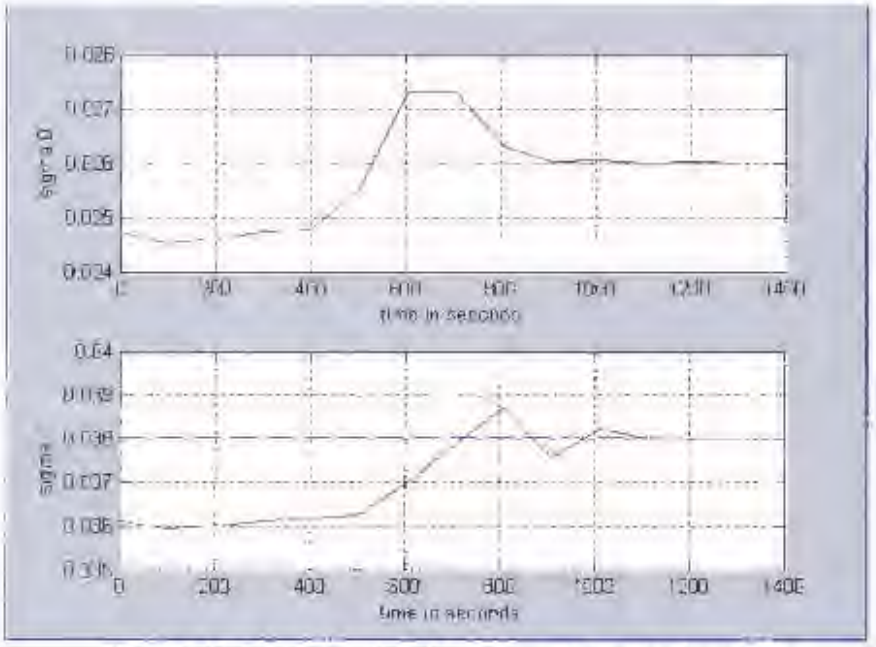

Figure 7.13: The auto tuning of both σ_0 and σ_1 simultaneously

Figure 7.13 shows the automatic tuning of the both controller parameters simultaneously, the starting values for the controller parameters was 5% less than the expected value for which the quadratic criterion is near its minimum. Both controller parameters increased steadily until the expected value, as illustrated in figure 7.13 that the time for the controller parameters to reach their respective expected values is not necessarily the same but it was shorter than the one in figure 7.11 and figure 7.12 for obvious reasons. It took σ_0 approximately $t = 1000$ seconds to reach the expected value of $\sigma_0 = 0.026$, while σ_1 took approximately $t = 1200$ seconds to reach the expected value of $\sigma_1 = 0.038$. The value for the positive scalar is $\gamma = 0.0003$.

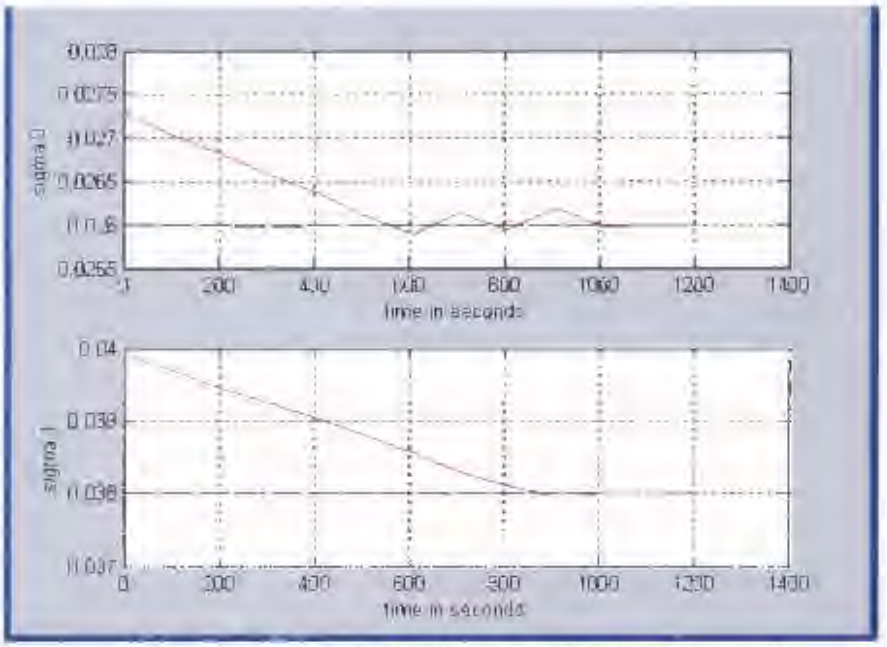

Figure 7.14: The auto tuning of both σ_0 and σ_1 instantaneously

Figure 7.14 shows the automatic tuning of the both controller parameters simultaneously. the starting values for the controller parameters was 5% more than the expected value for which the quadratic criterion is near its minimum. Both controller parameters decreased steadily until the expected value as illustrated in figure 7.14. It took σ_0 approximately $t = 1200$ seconds to reach the expected value of $\sigma_0 = 0.026$, while σ_1 took approximately $t = 1000$ seconds to reach the expected value of $\sigma_1 = 0.038$. The value for the positive scalar is $\gamma = 0.0003$.

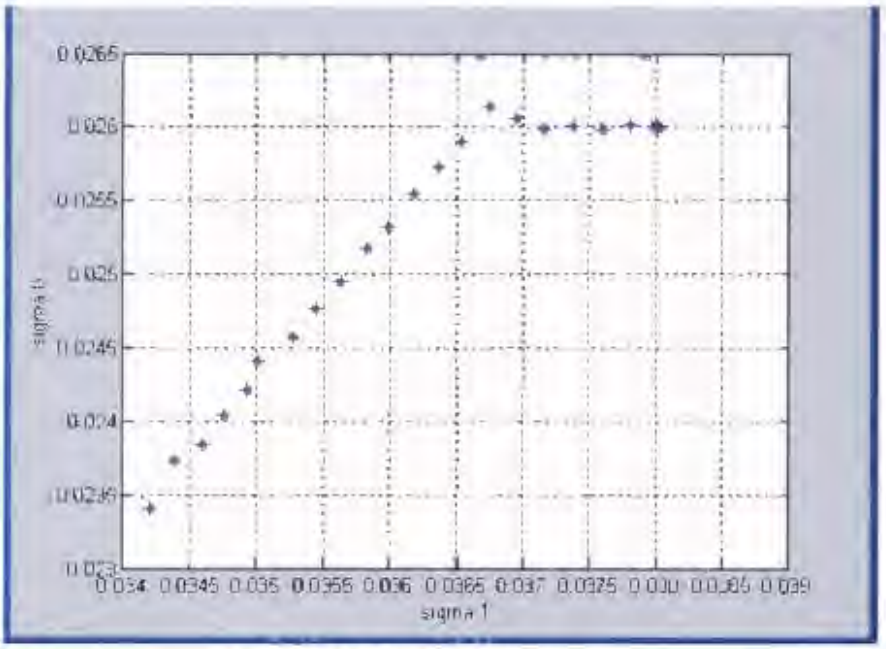

Figure 7.15: The trace of σ_0 vs σ_1

Figure 7.15 shows the trace for the controller parameters σ_0 and σ_1 when their starting values are 10% less than their expected value. The blue dot circled by the red boundary is the expected point where both controller parameters have reached their expected value.

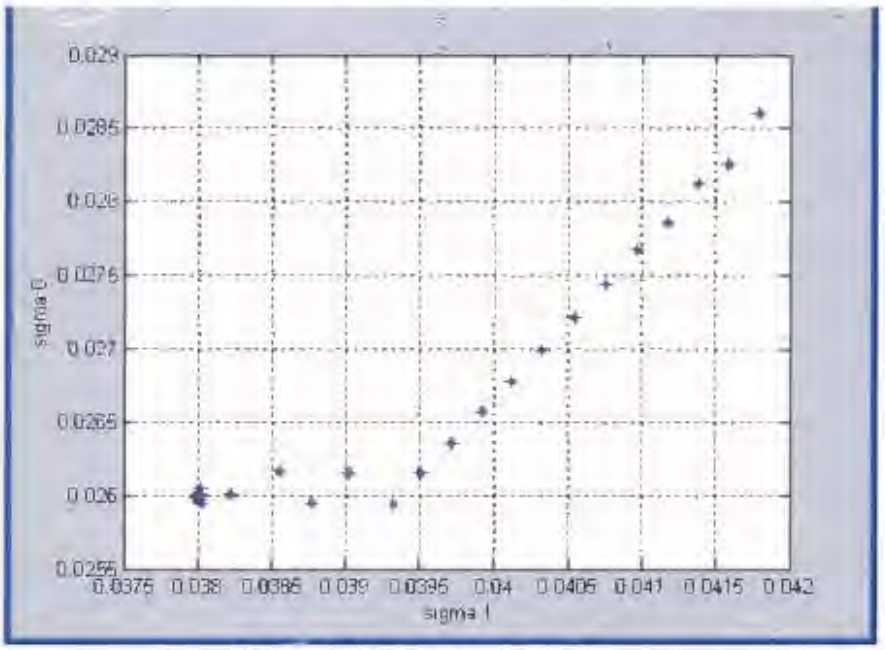

Figure 7.16: The trace of σ_0 vs σ_1

Figure 7.16 shows the trace for the controller parameters σ_0 and σ_1 when their staring values was 10% less than their expected value. The blue dot circled by the red boundary is the expected point where both controller parameters have reached their expected value.

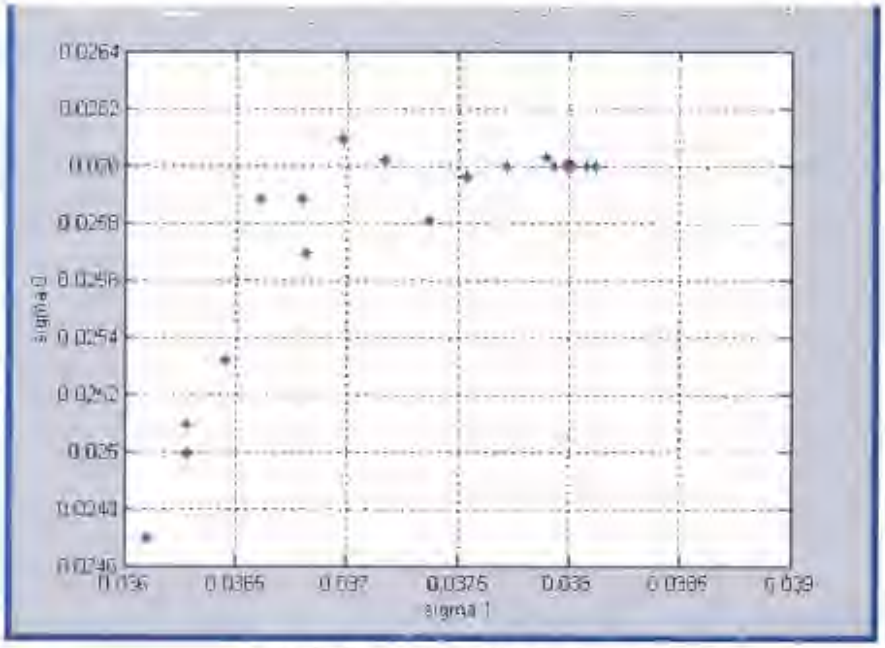

Figure 7.17: The trace of σ_1 vs σ_0

Figure 7.17 shows the trace for the controller parameters σ_0 and σ_1 when their starting values are 5% less than their expected value. The blue dot circled by the red boundary is the expected point where both controller parameters have reached their expected value.

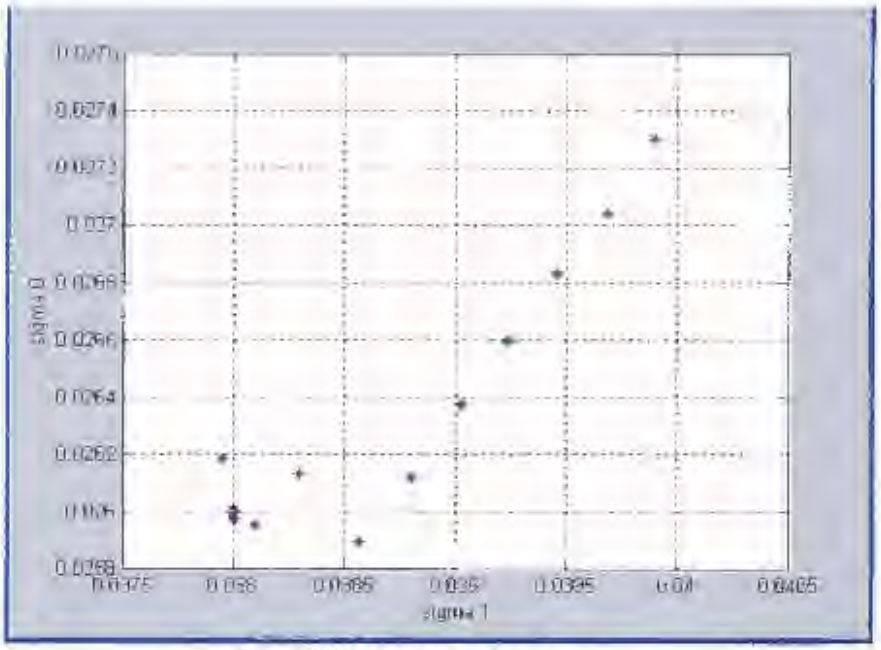

Figure 7.18: The trace of σ_0 vs σ_1

Figure 7.18 shows the trace for the controller parameters σ_0 and σ_1 when their starting values are 5% more than their expected value. The blue dot circled by the red boundary is the expected point where both controller parameters have reached their expected values.

The observation from figure 7.15-7.18 showing the trace of σ_0 and σ_1 is that some trace shows oscillation. this is due to the value of γ as this is the scalar that controls the rate of change.

7.2.3 Auto tuning of the IFT parameters for the DC motor with varying inertia

This section of the dissertation varies the inertia of the load for the DC motor to ensure that the IFT algorithm Visual Basic code works regardless of the varying load, and can be able to find the controller parameters for which any given system for which it has a minimum

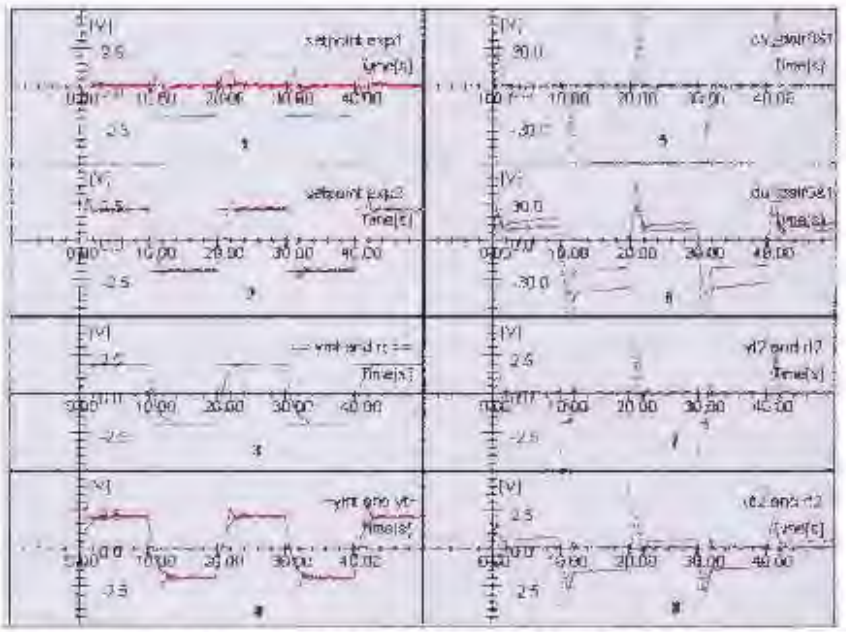

Figure 7.19: The results from both experiment 1 and 2 of the IFT algorithm for the DC motor with lighter load

Figure 7.19 shows the results from IFT algorithm for a case when the load of the DC motor is changed. Graph 4 shows that the overshoot experienced is smaller than for the previous load, hence from the experiment it can be assumed that the overshoot is directly proportional to the load of the motor.

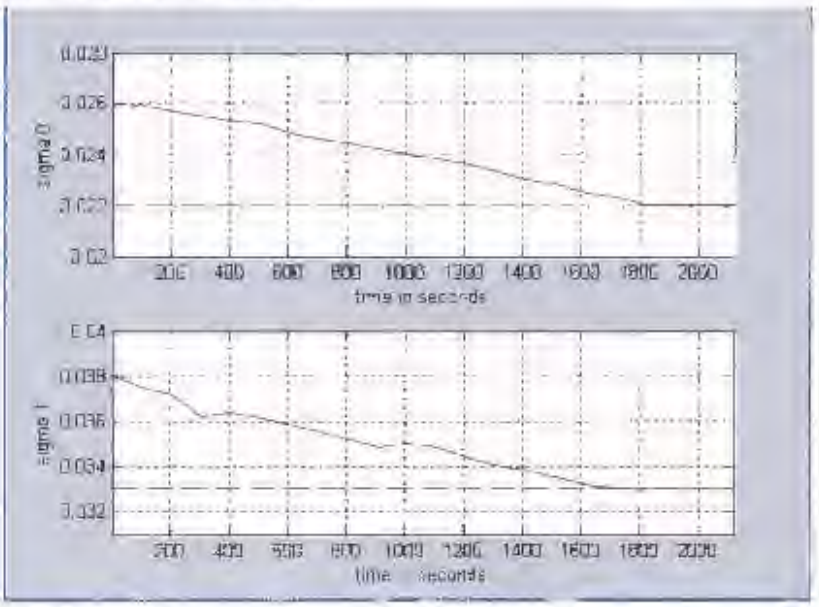

Figure 7.20: The auto tuning of the DC motor with different load

Figure 7.20 shows the auto tuning of both controller parameters σ_0 and σ_1, the chosen starting values for the controller parameters is 0.026 and 0.038 respectively. These

starting values were chosen because they are the values for which the quadratic criterion of the IFT algorithm is near its minimum for a DC motor with heavier load as stated in section 7.2.2. The values for the controller parameters for which the criterion is near its minimum for the lighter load is $\begin{bmatrix} 0.022 & 0.033 \end{bmatrix}$ in the following format $\begin{bmatrix} \sigma_0 & \sigma_1 \end{bmatrix}$.

Chapter 8

Conclusion and future development

8.1 CONCLUSIONS

The following conclusions can be drawn from the results in this thesis:

- It was shown that the MRAC gradient approach method is related very closely to the IFT algorithm. The MRAC algorithm produced the actual output response y with an overshoot of approximately 60%. The overshoot was shown to be due to the real system containing offsets which were not accounted for by the literature in the book Adaptive Control by *KJ Astrom and B Wittenmark (1989).*

- The IFT algorithm was applied to a one degree of freedom controller problem for a DC motor. The algorithm worked well for the controller parameter σ defined within the stable region or domain of the DC motor. The overshoot experience was below 40%. Thus the overshoot produced from the IFT algorithm was very small compared to the MRAC algorithm.

- It was shown that the controller in a IFT algorithm in a closed loop generates the data that contains enough information for the identification of the model suited for a better and improved control design by updating the controller parameters σ_0 and σ_1.

- For the controller to be tuned properly one has to collect the closed loop experimental data so that the estimation of the gradient can be calculated. The advantage of this algorithm is one can improve the controller successively from time to time without breaking the loop.

- The controller parameters for which the quadratic criterion is a minimum for a given domain are $\sigma = [0.026 \quad 0.038]$ in the following format $\sigma = [\sigma_0 \quad \sigma_1]$.

- The IFT algorithm automatically tuned the controller parameters to the desired valued stated above.

- The IFT algorithm was able to produce the correct controller parameters for a DC motor with a different load to ensure that the algorithm compensates for changes in the inertia of the load.

- It was shown that the rate at which the IFT algorithm will convert to the correct controller parameters for which the quadratic criterion is minimum depends entirely on the positive scalar λ, if this scalar is given a value which is too large the chances of missing the minimum increase. While if the scalar's value is too small the algorithm will take a long time to converge.

8.2 FUTURE RESEARCH

The following are the future work that is suggested through the observation duration of this research:

- The IFT algorithm can be implemented for a one degree of freedom controller with a disturbance signal and the overshoot on the output signal reduced to zero percent.

- The effect of large changes in inertia were observed to cause instability in the IFT controller and this could be investigated in detail.

- The effect of the IFT parameter gamaz or γ could be linked to the rate at which the controller parameters converged to the optimum and this rate could be optimized.

- The equations of the IFT parameter optimization could be modified to provide a forgetting factor functionality in which past data is discarded at a controlled rate set by the forgetting factor.

Chapter 8

Conclusion and future development

8.1 CONCLUSIONS

The following conclusions can be drawn from the results in this thesis:

- It was shown that the MRAC gradient approach method is related very closely to the IFT algorithm. The MRAC algorithm produced the actual output response y with an overshoot of approximately 60%. The overshoot was shown to be due to the real system containing offsets which were not accounted for by the literature in the paper by *KJ Astrom and B Wittenmark (1989), Adaptive Control.*

- The IFT algorithm was applied to a one degree of freedom controller problem for a DC motor. The algorithm worked well for the controller parameter σ defined within the stable region or domain of the DC motor. The overshoot experience was below 40%. Thus the overshoot produced from the IFT algorithm was very small compared to the MRAC algorithm.

- It was shown that the controller in a IFT algorithm in a closed loop generates the data that contains enough information for the identification of the model suited for a better and improved control design by updating the controller parameters σ_0 and σ_1.

- For the controller to be tuned properly one has to collect the closed loop experimental data so that the estimation of the gradient can be calculated. The advantage of this algorithm is one can improve the controller successively from time to time without breaking the loop.

- The controller parameters for which the quadratic criterion is a minimum for a given domain are $\sigma = [0.026 \quad 0.038]$ in the following format $\sigma = [\sigma_0 \quad \sigma_1]$.

- The IFT algorithm automatically tuned the controller parameters to the desired valued stated above.

- The IFT algorithm was able to produce the correct controller parameters for a DC motor with a different load to ensure that the algorithm compensates for changes in the inertia of the load.

- It was shown that the rate at which the IFT algorithm will convert to the correct controller parameters for which the quadratic criterion is minimum depends entirely on the positive scalar λ, if this scalar is given a value which is too large the chances of missing the minimum increase. While if the scalar's value is too small the algorithm will take a long time to converge.

8.2 FUTURE RESEARCH

The following are the future work that is suggested through the observation duration of this research:

- The IFT algorithm can be implemented for a one degree of freedom controller with a disturbance signal and the overshoot on the output signal reduced to zero percent.

- The effect of large changes in inertia were observed to cause instability in the IFT controller and this could be investigated in detail.

- The effect of the IFT parameter gamaz or γ could be linked to the rate at which the controller parameters [illegible]

- The equations of the IFT parameter optimization could be modified to provide a forgetting factor functionality in which past data is discarded at a controlled rate set by the forgetting factor.

References

[1] K. Hamamoto, T. Fukuda and T. Sugie (2003), Iterative feedback tuning of controllers for a two-mass-spring system with friction, Control Engineering Practice, Vol 11, pp (1061-1068)

[2] A. Karimi, L. Mišković and D. Bonvin (2003), Iterative correlation-based controller tuning with application to a magnetic suspension system, Control Engineering Practice, Vol 11, pp (1069-1078)

[3] J. Sjöberg, F. De Bruyne, M Agarwal, B.D.O Anderson, M. Gevers, F.J Kraus and N. Linard (2003), Iterative controller optimization for nonlinear systems, Control Engineering Practice, Vol 11, pp (1079-1086)

[4] D.J Murray-Smith, Juš Kocijan and Mingrui Gong (2003), A signal conVolution method for estimation of controller parameter sensitivity functions for tuning of feedback control systems by an iterative process, Control Engineering Practice, Vol 11, pp (1087-1094)

[5] O. Lequin, M. Gevers, M. Mossberg, E. Bosmans and L. Triest (2003), Iterative feedback tuning of PID parameters: comparison with classical tuning rules, Control Engineering Practice, Vol 11, pp (1023-1033)

[6] S. Gunnarsson, V Collignon and O Rousseaux (2003), Tuning of a decoupling controller for a 2×2 system using iterative feedback tuning , Control Engineering Practice, Vol 11, pp (1035-1041)

[7] W. K. Ho, Y. Hong, A. Hansson, H. Hjalmarsson and J. W. Deng (2003), Relay auto-tuning of PID controllers using iterative feedback tuning, Automatica, Vol 31(1), pp (149-157)

[8] H. Hjalmarsson and M. Gevers (2003), Special section on algorithms and applications of Iterative Feedback Tuning, Control Engineering Practice, Vol 11, pp 1021

[9] F. De Bruyne (2003), Iterative feedback tuning for internal model controllers, Control Engineering Practice, Vol 11, pp (1043-1048)

[10] T. Meurers, S. M. Veres and A. C. H. Tan (2003), Model-free frequency domain iterative active sound and vibration control, Control Engineering Practice, Vol 11, pp (1049-1059)

[11] C Leonardo, R Robert and L Peter (1999), Direct iterative tuning via spectral analysis, Automatica ,Vol 36, pp (1301-1307)

[12] Ari G. Partanen and Robert R. Bitmead (1995), The application of an iterative identification and controller design to a sugar cane crushing mill, Automatica, Vol 31, pp (1547-1563)

[13] H Hjalmarsson and T Birkeland (1998), Iterative feedback tuning of linear time-invariant MIMO systems, 37th IEEE Conference, pp (3893-3898)

[14] H Hjalmarsson, M Gevers, S Gunnarsson and O Lequin (1998), Iterative feedback tuning: Theory and application, (pp 26-41)

[15] H Hjalmarsson, S Gunnarsson and M Gevers (1994), A convergent iterative restricted complexity control design scheme, In Proceedings of the 33rd CDC, Lake Buena Vista, (pp. 1735–1740).

[16] M. Gevers, H.L. Trentelman and J.C Willems (1993), "Toward a joint design of identification and control", in Essays on Control: Perspectives in the Theory and its Applications, H.L. Trentelman and J. C. Willems, Eds. Boston, MA: Birkh¨auser, pp (111–151)

[17] P.M.J Van den Hof (1997), "Closed loop issue in system identification", in 11th IFAC, Symposium on System identification, Fukuoka, Japan

[18] W.S. Lee, B.D.O Anderson, R.L Kosut and I.M.Y Mareels (1993), A new approach to adaptive robust control, Int Journal of Adaptive Control and Signal Processing, Vol 7, pp (183-211)

[19] K.J Astrom (1993), Matching criteria for control and identification , in Proc ECC, Groningen, The Netherlands, pp (248-251)

[20] D.G Hulbert (1983), Multivariable control and optimization of the operation of a milling circuit at East Driefontein, Randburg, Counsin for Mineral Technology, Report M98, 20pp

[21] D. Dougherty, and D. Copper (2003), A practical multiple model adaptive strategy for single-loop MPC, Control Engineering Practice, 11(2), pp (141-159)

[22] KJ Astrom and B Wittenmark (1989), Adaptive Control, Addison-Wesley, Massachusetts.

[23] M. Junhong, T. Hiroyuki and Shimokohbe (2003), Double-integrator control for precision positioning in the presence of friction, Precision Engineering, Vol 27, pp (419-428).

[24] T. Orlawska-kowalska, K. Szabat and K. Jaszczak (2001), The influence of parameters and structure of PI-type fuzzy-logic controller on DC drive system dynamics, Fuzzy Sets and Systems, Vol 131, pp (251-264).

[25] Chu Kwong Chak, Gang Feng and T. Hesketh (1999), Multirate adaptive optimal control with application to DC motor, Computers and Electrical Engineering, Vol 23, pp (65-79).

[26] Chao C.L and Neou J (1997), Model reference adaptive control of air-lubricated capstan drive for precision positioning, Precision Engineering, Vol 24, pp (285-290)

[27] R.J.P Schrama and P.M.J Van den Hof (1993), Iterative identification and control design: A three step procedure with robustness analysis, Proc. ECC, pp.(237-241)

[28] R.A. de Callafon and P.M.J Van den Hof (1997), Suboptimal feedback control by a scheme of iterative identification and control design, mathematical modeling of system, Vol 3(1), pp (77-101)

[29] W.S. Lee, B.D.O Anderson, R.L Kosut and I.M.Y Mareels (1993), A new approach to adaptive robust control, Int. J. Adaptive Control and Signal Processing, Vol 7, pp (183-211)

[30] H. Hjalmarsson (1998), Control of nonlinear systems using iterative feedback tuning, American Control Conference, Philadelphia, pp (2083-2087)

[31] Robbins H and Monro S (1951), A stochastic approximation method, Ann. Math. Stat, Vol 22 pp (400-407)

Appendix 1

This section of the dissertation derives and explains equations that were stated and used in the earlier chapters. (The equation numbering used in this appendix follows that of the main text for ease of reference)

A1.1 DERIVATION OF MRAC EQUATIONS

$$y = \frac{a * t0}{1 + a * s0 + s * T} r \quad \text{..... (3.3)}$$

Equation (3.3) is obtained by replacing u in this equation $y = \frac{a}{1 + s * T} u$ with $u = t0 * r - s0 * y$

$$t0 = \frac{a_m * T}{a * T_m} \quad \text{..... (3.7)}$$

$$s0 = \frac{T - T_m}{a * T_m} \quad \text{..... (3.8)}$$

The parameters in equation (3.7) and (3.8) are obtained by comparing the numerator and the denominator of the actual and the desired transfer function given by (3.3) and (3.5) by writing them in such a way that the coefficient of s is one as illustrated in (3.3a) and (3.5a).

$$y = \left(\frac{\frac{a * t0}{T}}{s + \frac{1}{T} + \frac{a}{T} * s0} \right) \quad \text{..... (3.3a)}$$

$$y_m = \left(\frac{\frac{a_m}{T_m}}{\frac{1}{T_m} + s} \right) \quad \text{..... (3.5a)}$$

$$\frac{\partial e}{\partial t0} = \frac{a}{1 + a * s0 + s * T} r \quad \text{..... (3.9)}$$

$$\frac{\partial e}{\partial s0} = \frac{a^2 * t0}{(1 + a * s0 + s * T)^2} r \quad \text{..... (3.10)}$$

Equation (3.9) and (3.10) are obtained by taken the derivative of the following equation

$e_m = \frac{a * t0}{1 + a * s0 + s * T} r - \frac{a_m}{1 + s * T_m} r$ .... (3.6) with respect to $t0$ and $s0$ respectively.

A1.2 DERIVATION OF IFT ALGORITHM EQUATIONS FOR TWO DEGREE OF FREEDOM CONTROLLER

$$e_m(\sigma) = y(\sigma) - y_m = \left(\frac{k(\sigma) * G}{1 + f(\sigma) * G} r - y_m \right) + \frac{1}{1 + f(\sigma) * G} d \quad \text{..... (A2.4)}$$

Equation (A2.4) is the difference between the actual and the desired response

$$\frac{\partial y}{\partial \sigma}(\sigma) = \frac{G}{1 + f(\sigma) * G} \frac{\partial k}{\partial \sigma}(\sigma) * r - \frac{k(\sigma) * G^2}{(1 + f(\sigma) * G)^2} \frac{\partial f}{\partial \sigma} r - \frac{G}{(1 + f(\sigma) * G)^2} \frac{\partial f}{\partial \sigma}(\sigma) * d$$

$$= \frac{1}{k(\sigma)} \frac{\partial k}{\partial \sigma}(\sigma) * T_0(\sigma) * r - \frac{1}{k(\sigma)} \frac{\partial f}{\partial \sigma}(\sigma) * \left([T_0]^2(\sigma) * r + T_0(\sigma) * S_0(\sigma) * d \right) \quad \text{..... (A2.13)}$$

The equation in (A2.13) is obtained by taking the derivative of the actual response y with respect to the controller parameter sigma σ.

A1.3 DERIVATION OF IFT ALGORITHM EQUATIONS FOR ONE DEGREE OF FREEDOM CONTROLLER

$$\frac{\partial y}{\partial \sigma}=\frac{1}{K}*\frac{G*K*r}{(1+G*K)}*\frac{\partial K}{\partial \sigma}-\frac{1}{K}*\frac{G^2*K^2*r}{(1+G*K)^2}*\frac{\partial K}{\partial \sigma} \quad \text{..... (4.9)}$$

The equation (4.9) is obtained by taking the derivative of the actual response y with respect to the controller parameter sigma σ for a one-degree of freedom controller.

$$\frac{\partial y}{\partial \sigma}=\frac{1}{K}*\frac{\partial K}{\partial \sigma}*\left[T*r-T^2*r\right] \quad \text{..... (4.11)}$$

Since $T=\frac{G*K}{1+G*K}$ equation (4.9) can be summarized into (4.11)

$$\frac{\partial y}{\partial \sigma}=\frac{1}{K}*\frac{\partial K}{\partial \sigma}*y^{(2)} \quad \text{..... (4.14)}$$

The term in bracket in equation (4.11) is equal to the actual response $y_i^{(2)}$ from experiment 2, hence equation (4.14)

$$\frac{\partial u}{\partial \sigma}=\frac{1}{K}*\frac{\partial K}{\partial \sigma}*u^{(2)} \quad \text{..... (4.21)}$$

Similarly with the input derivative in equation (4.20) $u^{(2)}$ is the input signal from experiment 2.

$$\frac{\partial u}{\partial \sigma_0}=\frac{1}{(\sigma_1*z+\sigma_0)}*u^{(2)} \quad \text{..... (4.22)}$$

$$\frac{\partial u}{\partial \sigma_1}=\frac{z}{(\sigma_1*z+\sigma_0)}*u^{(2)} \quad \text{..... (4.23)}$$

Equation (4.22) and (4.23) are obtained by replacing the type 1 controller with the equation $K(z)=\frac{\sigma_1*z+\sigma_0}{(z-1)}$ and separating the vector $\sigma=[\sigma_0 \quad \sigma_1]$.

$$\frac{a*\beta_0+a*\beta_1*s}{s*(1+s*T)+a*\beta_0+a*\beta_1*s}=\frac{a_m}{1+s*T_m} \quad \text{..... (5.12)}$$

denominator and numerator of equation (5.12) gives:

$$(a*\beta_0 + a*\beta_1*s)*(1+s*T_m) = a_m*(s*(1+s*T)+a*\beta_0+a*\beta_1*s) \quad \text{..... (5.13)}$$

coefficient comparison of equation (5.13)

$$s^2: \quad a*\beta_1*T_m = a_m*T \quad \text{..... (5.13a)}$$

$$s^1: \quad a*\beta_0*T_m + a*\beta_1 = a_m + a_m*a*\beta_1 \quad \text{..... (5.13b)}$$

$$s^0: \quad a*\beta_0 = a_m*a*\beta_0 \quad \text{..... (5.13c)}$$

But the gain for the desired model $a_m = 1$, therefore from equation (5.13a)

$$\beta_1 = \frac{T}{a*T_m} \quad \text{..... (5.15)}$$

Equation (5.15) into (5.13b) produces (5.16)

$$\beta_0 = \frac{1}{a*T_m} \quad \text{..... (5.16)}$$

A1.4 DETERMINING THE DC MOTOR TRANSFER FUNCTION

The DC motor's transfer function was found to be:

$$g(s) = \frac{1.01}{(1+2.00*s)}[v]/[v] \quad \text{..... (5.2)}$$

This value is not 100% accurate as it is calculate by approximation and its contains the system noise from the apparatus from the servo motor mentioned in chapter 5.

A1.5 THE DC MOTOR APPARATUS

Table A1.1: The table explains the use for the apparatus used for the DC motor in this thesis

Pre – amplifier unit	This is used to reverse the direction on the motor
Attenuator unit	Used to attenuate or vary the amplitude of a signal
Op Amp unit	This is used for unity feedback
Input pot unit	Not used in this thesis
DC motor	The motor drives a gear-box

Output pot unit	Not used in this thesis
Servo Amplifier	This controls the speed of rotation and also the direction of rotation of the servo-motor
Power supply	This provides the various a.c. and d.c. supplies required
Analog/Digital Converter	Convert analog to digital signals
Digital Multimeter	Display the results numerically
Reduction Gear Tacho unit	This produces the speed which is fed to the ADC
PC or Computer	Used to code visual basic and run the algorithms

Appendix 1

This section of the dissertation derives and explains equations that were stated and used in the earlier chapters. (The equation numbering used in this appendix follows that of the main text for ease of reference)

A1.1 DERIVATION OF MRAC EQUATIONS

$$y = \frac{a * t0}{1 + a * s0 + s * T} r \quad \text{.....} \quad (3.3)$$

Equation (3.3) is obtained by replacing u in this equation $y = \frac{a}{1 + s * T} u$ with $u = t0 * r - s0 * y$

$$t0 = \frac{a_m * T}{a * T_m} \quad \text{.....} \quad (3.7)$$

$$s0 = \frac{T - T_m}{a * T_m} \quad \text{.....} \quad (3.8)$$

The parameters in equation (3.7) and (3.8) are obtained by comparing the numerator and the denominator of the actual and the desired transfer function given by (3.3) and (3.5) by writing them in such a way that the coefficient of s is one as illustrated in (3.3a) and (3.5a).

$$y = \left(\frac{\frac{a * t0}{T}}{s + \frac{1}{T} + \frac{a}{T} * s0} \right) \quad \text{.....} \quad (3.3a)$$

$$y_m = \left(\frac{\frac{a_m}{T_m}}{\frac{1}{T_m} + s} \right) \quad \text{..... (3.5a)}$$

$$\frac{\partial e}{\partial t0} = \frac{a}{1 + a * s0 + s * T} r \quad \text{..... (3.9)}$$

$$\frac{\partial e}{\partial s0} = \frac{a^2 * t0}{(1 + a * s0 + s * T)^2} r \quad \text{..... (3.10)}$$

Equation (3.9) and (3.10) are obtained by taken the derivative of the following equation

$e_m = \frac{a * t0}{1 + a * s0 + s * T} r - \frac{a_m}{1 + s * T_m} r$ .... (3.6) with respect to $t0$ and $s0$ respectively.

A1.2 DERIVATION OF IFT ALGORITHM EQUATIONS FOR TWO DEGREE OF FREEDOM CONTROLLER

$$e_m(\sigma) = y(\sigma) - y_m = \left(\frac{k(\sigma) * G}{1 + f(\sigma) * G} r - y_m \right) + \frac{1}{1 + f(\sigma) * G} d \quad \text{..... (A2.4)}$$

Equation (A2.4) is the difference between the actual and the desired response

$$\frac{\partial y}{\partial \sigma}(\sigma) = \frac{G}{1 + f(\sigma) * G} \frac{\partial k}{\partial \sigma}(\sigma) * r - \frac{k(\sigma) * G^2}{(1 + f(\sigma) * G)^2} \frac{\partial f}{\partial \sigma} r - \frac{G}{(1 + f(\sigma) * G)^2} \frac{\partial f}{\partial \sigma}(\sigma) * d$$

$$= \frac{1}{k(\sigma)} \frac{\partial k}{\partial \sigma}(\sigma) * T_0(\sigma) * r - \frac{1}{k(\sigma)} \frac{\partial f}{\partial \sigma}(\sigma) * \left([T_0]^2(\sigma) * r + T_0(\sigma) * S_0(\sigma) * d \right) \text{..... (A2.13)}$$

The equation in (A2.13) is obtained by taking the derivative of the actual response y with respect to the controller parameter sigma σ.

A1.3 DERIVATION OF IFT ALGORITHM EQUATIONS FOR ONE DEGREE OF FREEDOM CONTROLLER

$$\frac{\partial y}{\partial \sigma}=\frac{1}{K}*\frac{G*K*r}{(1+G*K)}*\frac{\partial K}{\partial \sigma}-\frac{1}{K}*\frac{G^2*K^2*r}{(1+G*K)^2}*\frac{\partial K}{\partial \sigma} \quad \text{..... (4.9)}$$

The equation (4.9) is obtained by taking the derivative of the actual response y with respect to the controller parameter sigma σ for a one-degree of freedom controller.

$$\frac{\partial y}{\partial \sigma}=\frac{1}{K}*\frac{\partial K}{\partial \sigma}*\left[T*r-T^2*r\right] \quad \text{..... (4.11)}$$

Since $T=\frac{G*K}{1+G*K}$ equation (4.9) can be summarized into (4.11)

$$\frac{\partial y}{\partial \sigma}=\frac{1}{K}*\frac{\partial K}{\partial \sigma}*y^{(2)} \quad \text{..... (4.14)}$$

The term in bracket in equation (4.11) is equal to the actual response $y_t^{(2)}$ from experiment 2, hence equation (4.14)

$$\frac{\partial u}{\partial \sigma}=\frac{1}{K}*\frac{\partial K}{\partial \sigma}*u^{(2)} \quad \text{..... (4.21)}$$

Similarly with the input derivative in equation (4.20) $u^{(2)}$ is the input signal from experiment 2.

$$\frac{\partial u}{\partial \sigma_0}=\frac{1}{(\sigma_1*z+\sigma_0)}*u^{(2)} \quad \text{..... (4.22)}$$

$$\frac{\partial u}{\partial \sigma_1}=\frac{z}{(\sigma_1*z+\sigma_0)}*u^{(2)} \quad \text{..... (4.23)}$$

Equation (4.22) and (4.23) are obtained by replacing the type 1 controller with the equation $K(z)=\frac{\sigma_1*z+\sigma_0}{(z-1)}$ and separating the vector $\sigma=\left[\sigma_0 \quad \sigma_1\right]$.

$$\frac{a*\beta_0+a*\beta_1*s}{s*(1+s*T)+a*\beta_0+a*\beta_1*s}=\frac{a_m}{1+s*T_m} \quad \text{..... (5.12)}$$

Cross multiplying the denominator and numerator of equation (5.12) gives:

$$(a*\beta_0 + a*\beta_1*s)*(1+s*T_m) = a_m*(s*(1+s*T)+a*\beta_0+a*\beta_1*s) \quad \text{..... (5.13)}$$

coefficient comparison of equation (5.13)

$$s^2: \quad a*\beta_1*T_m = a_m*T \quad \text{..... (5.13a)}$$

$$s^1: \quad a*\beta_0*T_m + a*\beta_1 = a_m + a_m*a*\beta_1 \quad \text{..... (5.13b)}$$

$$s^0: \quad a*\beta_0 = a_m*a*\beta_0 \quad \text{..... (5.13c)}$$

But the gain for the desired model $a_m = 1$, therefore from equation (5.13a)

$$\beta_1 = \frac{T}{a*T_m} \quad \text{..... (5.15)}$$

Equation (5.15) into (5.13b) produces (5.16)

$$\beta_0 = \frac{1}{a*T_m} \quad \text{..... (5.16)}$$

A1.4 DETERMINING THE DC MOTOR TRANSFER FUNCTION

The DC motor's transfer function was found to be:

$$g(s) = \frac{1.01}{(1+2.00*s)}[v]/[v] \quad \text{..... (5.2)}$$

This value is not 100% accurate as it is calculate by approximation and its contains the system noise from the apparatus from the servo motor mentioned in chapter 5.

A1.5 THE DC MOTOR APPARATUS

Table A1.1: The table explains the use for the apparatus used for the DC motor in this thesis

Pre – amplifier unit	This is used to reverse the direction on the motor
Attenuator unit	Used to attenuate or vary the amplitude of a signal
Op Amp unit	This is used for unity feedback
Input pot unit	Not used in this thesis
DC motor	The motor drives a gear-box

Output pot unit	Not used in this thesis
Servo Amplifier	This controls the speed of rotation and also the direction of rotation of the servo-motor
Power supply	This provides the various a.c. and d.c. supplies required
Analog/Digital Converter	Convert analog to digital signals
Digital Multimeter	Display the results numerically
Reduction Gear Tacho unit	This produces the speed which is fed to the ADC
PC or Computer	Used to code visual basic and run the algorithms

Appendix 2

Iterative Feedback Tuning

This appendix chapter is mainly from the paper *Iterative feedback tuning: Theory and Applications* by Hakan Hjalmarsson, Michel Gevers, Svante Gunnarsson, Olivier Lequin (1998) that deals with IFT systems.

A2.1 BACKGROUND

Many of problems experienced in the control industry can be expressed in terms of a criterion or cost function. In order to explicitly find solutions to such optimizations problems, it requires one to know in details the model of the plant and its disturbances. In practice it is often that the plant and the disturbances are not known in great detail and as a result it is hard to achieve the best possible performance with the controller [14].

In order to tune the parameters of a controller for optimizing the performance of the controlled system one requires iterative gradient-based minimization procedures. It is very complicated and cumbersome to compute the gradient of the criterion function with respect to the controller's parameters because of the model for the plant and the disturbances. If one does not know the model for the plant and the disturbance it is not known how one can be able to compute the gradient [14].

In the past all attempts at obtaining the minimum of a control performance criterion always relied on having the model for the plant and the disturbances. A number of methods have been proposed for the direct tuning of the controller parameters [27], [28] and [29], these methods are all based on achieving some certain properties of the closed

loop system of the plant. Until recently different schemes have been proposed to address the problem of model based designing of controller parameters namely [14]:

- Control design
- Iterative identification

The way these schemes work is they iteratively perform the identification of the plant's model and they update the model-based controller and every updated controller is being applied to the plant. The present controller in a closed loop generates the data, and this data contains enough information for the identification of the model suited for a new and improved control design. For every iteration the controller is tuned better and better than the previous one and it is getting closer to the control objective and the performance is improved on the plant [14].

Previous attempts at minimizing the control performance criterion by direct controller parameters tuning has failed because of the difficulty of having to compute the gradient of this cost criterion with respect to the controller parameters [14]. For a controller of low order structure the experimental and industries have reported successes with iterative identification based controller design schemes. The Gauss-Newton based scheme can be used to minimize the criterion. For a controller with a two-degree of freedom, three steps are performed at each step of the iteration process:

- Collect data from the controlled plant under normal operation conditions
- Feeding back at the setpoint r the output measured during normal operation
- Collect data under normal operation conditions

Because of this procedure the method is named Iterative Feedback Tuning (IFT). When the controller has one degree of freedom the third step becomes unnecessary and it is left out when the criterion minimization process takes place, this will be explained later in chapter 4 and 5. The algorithm does not require one to know the model of the system to be controlled, hence the method is growing in popularity in industry because of their large and complicated systems which at times are very difficult to model as mathematical systems see [5].

Iterative feedback tuning also known as IFT is a method based on iterative tuning of the controller's parameters in the direction of decreasing gradient of a given cost function. The gradients are estimated using the closed loop experimental data, and an update of the parameters of the controller is estimated at each iteration. Hjalmarrsson [8] first proposed the IFT method in 1994 after the industries experience problems of always needing to know explicitly the model of the system in order to design any control for it.

People in industries encountered further difficulties, as other systems could not be modeled properly. When using the IFT algorithm one need not know the model of the system to be controlled, and like other numerical optimization routine one can use a step size on the parameters, which enables one to control the rate of change between the new and the previous controller [14].

IFT scheme because of its simplicity is used for a variety of tasks from the simplest like optimal tuning of simple PID controllers to more complex tasks like systematic design of controllers of increasing complexity [14]. The other advantage of using the IFT algorithm is that the controller can be improved successively from time to time without ever breaking the loop [14]. In process control applications one of the objectives in controller designing is to achieve disturbance rejection, while with IFT scheme the tuning of the controller parameters for disturbance rejection is driven by the disturbances themselves [14].

The IFT algorithm is analyzed in the following section, the equations are derived for a general two degree of freedom controller system. This section is mainly based on the following resources: The paper *iterative feedback tuning, Theory and application* by Hakan Hjalmarsson, Michel Gevers, Svante Gunnarsson, Olivier Lequin (1998) that deals with IFT systems, the paper *Iterative feedback tuning of linear time-invariant MIMO system* by Hjalmarsson H, Birkeland T (1998), and the paper *A convergent iterative restricted complexity control design* by Hjalmarsson H, Gunnarsson S (1994).

A2.2 ANALYSIS OF THE IFT METHOD

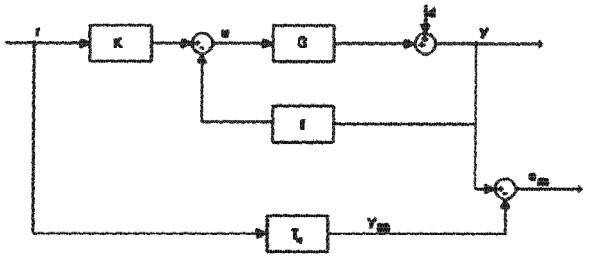

Figure A2.1: Block diagram for a two degree of freedom controller for a closed loop system

The figure above shows the feedback system with the following notation:

- G represents the plant's transfer function
- d is the process unmeasurable disturbances
- T_d is the desired transfer function
- e_m is the error between the actual and the desired output
- y is the actual output response
- y_m is the desired output response
- u is the input signal
- r is the setpoint
- f and k are the controllers

Assuming that the system is controlled by a two degree of freedom controller as illustrated in figure A2.1 the following equation can be derived:

$$u = k(\sigma) * r - f(\sigma) * y \quad \text{..... (A2.1)}$$

In equation (A2.1) σ is the controller parameter

Let the desired output y_m represent the output response from the closed loop system, and it can be defined as:

$$y_m = T_d * r \quad \text{..... (A2.2)}$$

The actual output response y is represented by:

$$y(\sigma)=\frac{k(\sigma)*G}{1+f(\sigma)*G}r+\frac{1}{1+f(\sigma)*G}d \quad \text{..... (A2.3)}$$

The error e_m between the actual and the desired output response can be written as follows:

$$e_m(\sigma)=y(\sigma)-y_m \;=\left(\frac{k(\sigma)*G}{1+f(\sigma)*G}r-y_m\right)+\frac{1}{1+f(\sigma)*G}d \quad \text{..... (A2.4)}$$

The error e_m in equation (A2.4) is caused by disturbances as well as incorrect tracking of the setpoint signal r.

The quadratic criterion in equation (A2.5) will be considered for this section:

$$J(\sigma)=\frac{1}{2N}E\left[\sum_{t=1}^{N}(L_y e_m(\sigma))^2+\lambda\sum_{t=1}^{N}(L_u u(\sigma))^2\right] \quad \text{..... (A2.5)}$$

The parameter lamda λ in this quadratic equation is a scalar and N is the finite length for which the experiment is performed in a given time.

In equation (A2.5), $E\left[\sum_{t=1}^{N}(L_y e_m(\sigma))^2+\lambda\sum_{t=1}^{N}(L_u u(\sigma))^2\right]$ is the expectation with respect to the disturbance d.

The controller parameter (σ) is defined as follows:

$$\sigma^* = \arg\min J(\sigma) \quad \text{..... (A2.6)}$$

Equation (A2.6) means that the correct controller parameters sigma (σ) are those for which the quadratic criterion is a minimum.

The purpose of the quadratic criterion in equation (A2.5) is to tune the process response to a desired response of finite length N in a mean square sense. The first term in equation (A2.5) is an error e_m, frequency weighted by a filter L_y. The second term is the penalty on the control effort that is frequency weighted by a filter L_u. The two filters L_y and L_u can be set to 1, but if one wants to give added flexibility to the design one can set them to other values depending on the problem at hand [14].

Let $T_0(\sigma)$ be the closed loop and $S_0(\sigma)$ be the sensitivity function. Therefore the following equations can be derived:

$$T_0(\sigma) = \frac{k(\sigma) * G}{1 + f(\sigma) * G} \quad \text{(A2.7)}$$

$$S_0(\sigma) = \frac{1}{1 + f(\sigma) * G} \quad \text{(A2.8)}$$

$$T_0(\sigma) + S_0(\sigma) = 1 \quad \text{(A2.9)}$$

Substituting equation (A2.7) and equation (A2.8) into equation (A2.5) gives the following equation:

$$J(\sigma) = \frac{1}{2N} \sum_{t=1}^{N} \left\{ L_y \left(y_m - T_0(\sigma) * r \right) \right\}^2 + \frac{1}{2} E\left[\{ L_y S_0(\sigma) * d \}^2 \right] + \lambda * \frac{1}{2N} E\left[\sum_{t=1}^{N} \left(L_u u(\sigma) \right)^2 \right] \quad \text{A2.10}$$

In equation (A2.10) the first term is the tracking error, while the second term is the disturbance contribution and the third and last term is the penalty on the control effort [14].

A2.3 THE MINIMIZATION OF THE QUADRATIC CRITERION

The gradient of the quadratic criterion $J(\sigma)$ is derived in this section, the purpose for this is to satisfy equation (A2.6), to find the controller parameter sigma (σ) for which the quadratic criterion is a minimum.

A2.3.1 Output signal

A criterion or cost function has been defined as a function of the controller parameters sigma (σ). Its minimum is found by partial differentiation of the cost function $J(\sigma)$ with respect to the controller parameters. The resulting equations are re-written to make them dependent on measured plant variables, first with an independent setpoint and then with a derived setpoint. The resulting data can be used to determine the optimum parameters without bias due to measurement noise. For simplicity the derivation will assume that both filters L_y and L_u are equal to 1.

In order for the correct controller parameters sigma $(\sigma)$to be obtained that satisfies equation (A2.6) , the gradient of $J(\sigma)$ is obtained and at the point where this gradient is a minimum the correct controller parameters are found. In order to achieve that, the gradient is equated to zero as shown in equation (A2.1).

$$\frac{\partial J}{\partial \sigma}(\sigma)=\frac{1}{N}E\left[\sum_{t=1}^{N} e_m(\sigma)\frac{\partial e_m}{\partial \sigma}(\sigma)+\lambda\sum_{t=1}^{N} u(\sigma)\frac{\partial u}{\partial \sigma}(\sigma)\right]=0 \quad \text{..... (A2.11)}$$

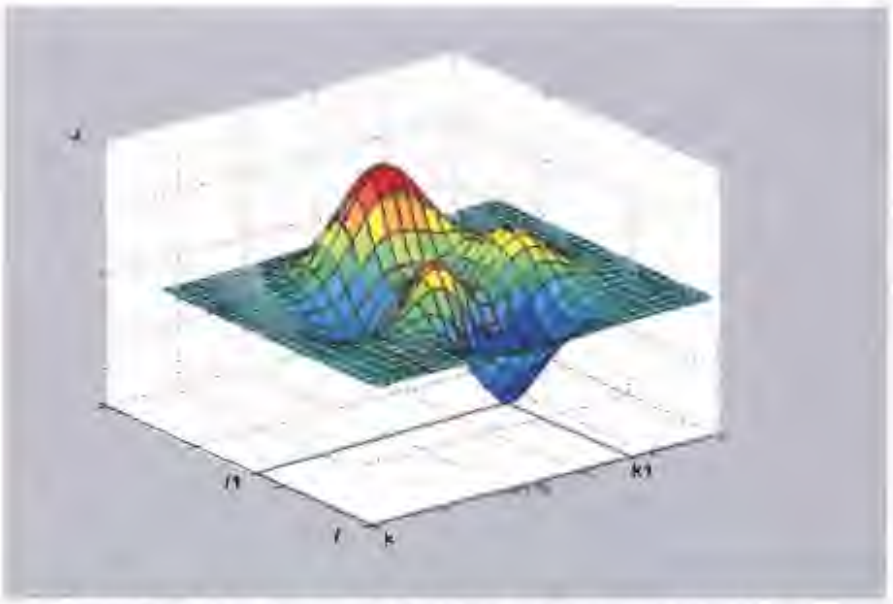

Figure A.2: The illustration of the IFT criterion minimazation in 3-D

Figure A2.2 illustrate the procedure to find the controller parameters to satisfy equation (A2.6) "$\sigma^{*} = \arg\min J(\sigma)$" where by the controller parameter sigma (σ) is a vector made of $[l \quad k]$. The search for the controller parameters of the given function has to be done in both axis to makes the quadratic criterion $J(\sigma)$ a minimum. To illustrate that in figure A2.2 the hypothetical quadratic criterion is a minimum when the controller parameter (σ) has the following indexes $[l1 \quad k1]$.

One cannot solve equation (A2.11) directly as it stands because it has one too many unknowns and it will be cumbersome to compute the gradient $\frac{\partial J}{\partial \sigma}(\sigma)$. If it were possible to compute it the solution will be obtained from the following iterative algorithm:

$$\sigma_{(i+1)} = \sigma_i - \gamma_i R_i^{-1} \frac{\partial J}{\partial \sigma}(\sigma_i) \qquad \text{(A2.12)}$$

In the iterative algorithm equation (A2.12) above γ_i is a positive real scalar that determines the step size, while R_i is a positive matrix [14]. In order to determine the

point "σ" at which the criterion is minimized the following quantities need to be generated:

- The error signal $e_m(\sigma)$ and the input signal $u(\sigma)$
- The gradients $\frac{\partial e_m}{\partial \sigma}(\sigma)$ and $\frac{\partial u}{\partial \sigma}(\sigma)$

While the following quantities needs to be computed:

- The following products $e_m(\sigma)*\frac{\partial e_m}{\partial \sigma}(\sigma)$ and $u(\sigma)*\frac{\partial u}{\partial \sigma}(\sigma)$

Performing the following process can generate these quantities:
Firstly the error e_m is obtained by taking the difference between the actual and the desired response as stated in equation (A2.4), the desired response is independent of the controller parameter sigma (σ) therefore the gradient $\frac{\partial e_m}{\partial \sigma}(\sigma)=\frac{\partial y}{\partial \sigma}(\sigma)$

$$\frac{\partial y}{\partial \sigma}(\sigma)=\frac{G}{1+f(\sigma)*G}\frac{\partial k}{\partial \sigma}(\sigma)*r-\frac{k(\sigma)*G^2}{(1+f(\sigma)*G)^2}\frac{\partial f}{\partial \sigma}r-\frac{G}{(1+f(\sigma)*G)^2}\frac{\partial f}{\partial \sigma}(\sigma)*d$$

$$=\frac{1}{k(\sigma)}\frac{\partial k}{\partial \sigma}(\sigma)*T_0(\sigma)*r-\frac{1}{k(\sigma)}\frac{\partial f}{\partial \sigma}(\sigma)*\left([T_0]^2(\sigma)*r+T_0(\sigma)*S_0(\sigma)*d\right)..\text{ (A2.13)}$$

Equation (A2.13) is derived by taking the derivative of equation (A2.4) with respect to the controller parameter sigma (σ) and using equation (A2.7), (A2.8) and (A2.9) to simplify the resulting derivative equation.

In equation (A2.13) the expressions $k(\sigma),\frac{\partial k}{\partial \sigma}(\sigma)$ and $\frac{\partial f}{\partial \sigma}(\sigma)$ are all known functions which depend on the controller parameter sigma (σ). The closed loop $T_o(\sigma)$ and the sensitivity function $S_o(\sigma)$ depend on the unknown system and as a result they cannot be computed at this stage.

The following can be observed in equation (A2.13): There is a double filtering of the setpoint r and the disturbance d and this can be expressed mathematically as:

$$[T_0]^2 * r + T_0 S_0 * d = T_0 y \quad \text{..... (A2.14)}$$

Equation (A2.14) is derived by using the expression for the actual output response in equation (A2.3) as $y = T_0 * r + S_0 * d$.

Substituting equation (A2.14) into equation (A2.13) gives the simplified version of equation (A2.13)

$$\frac{\partial y}{\partial \sigma}(\sigma) = \frac{1}{k(\sigma)} * \left[\frac{\partial k}{\partial \sigma}(\sigma) * T_0(\sigma) * r - \frac{\partial f}{\partial \sigma}(\sigma) * T_0(\sigma) * y \right]$$

$$= \frac{1}{k(\sigma)} \left[\left(\frac{\partial k}{\partial \sigma}(\sigma) - \frac{\partial f}{\partial \sigma}(\sigma) \right) * T_0(\sigma) * r + \frac{\partial f}{\partial \sigma}(\sigma) T_0(\sigma)(r - y) \right] \quad \text{..... (A2.15)}$$

The last term in equation (A2.15) is the difference between the setpoint signal r and the actual output response y. This term can be obtained by subtracting the actual output response signal from one experiment on the closed loop system from the setpoint signal, and this error signal can be used as the setpoint signal in the new experiment [14].

The IFT algorithm for a two degree of freedom controller's initial stage can be summarized as having the following procedure:

For every iteration j of the controller-tuning algorithm, three experiments will be used and each will be of duration N with a constant controller $c(\sigma) = \{k(\sigma), f(\sigma)\}$ operating on the plant. Experiment one and three consists of collecting the data under normal operating condition, while experiment two is the most interesting one. The setpoint signal

is has N - length $r_j^{(i)}$ where $i = 1,2,3,$ while the corresponding actual output response signal is represented by $y^{(i)}(\sigma_j)$ where $i = 1.2.3.$ hence the following:

$$r_j^{(1)} = r, \quad y^{(1)}(\sigma_j) = T_0(\sigma_j) * r + S_0(\sigma_j) * d_j^{(1)} \quad \text{(A2.16)}$$

$$r_j^{(2)} = (r - y^{(1)}), \quad y^{(2)}(\sigma_j) = T_0(\sigma_j)(r - y^{(1)}(\sigma_j)) + S_0(\sigma_j) * d_j^{(2)} \quad \text{(A2.17)}$$

$$r_j^{(3)} = r, \quad y^{(3)}(\sigma_j) = T_0(\sigma_j) * r + S_0(\sigma_j) * d_j^{(3)} \quad \text{(A2.18)}$$

The last term on the equations are multiplied by a term $d_j^{(i)}$ this represent the disturbance acting on the system during experiment i at iteration j. The disturbances come from different experiments therefore it is a reasonable to assume that they are mutually independent. This assumption can only be satisfied if the length N of one experiment is greater than the time of the disturbances [14]. These experiments yield an exact realization of $e_m(\sigma_j)$:

$$e_m(\sigma_j) = y^{(1)}(\sigma_j) - y_m \quad \text{(A2.19)}$$

Let $est\left[\frac{\partial y}{\partial \sigma}\right]$ denotes the estimation of the gradient $\frac{\partial y}{\partial \sigma}$

$$est\left[\frac{\partial y}{\partial \sigma}(\sigma_j)\right] \cong \frac{1}{k(\sigma_j)}\left[\left(\frac{\partial k}{\partial \sigma}(\sigma_j) - \frac{\partial f}{\partial \sigma}(\sigma_j)\right) * y^{(3)}(\sigma_j) + \frac{\partial f}{\partial \sigma}(\sigma_j) * y^{(2)}(\sigma_j)\right] \quad \text{(A2.20)}$$

This estimation of the gradient in equation (A2.20) can be further rewritten by straight substitution of equation (A2.16), (A2.17) and (A2.18) in (A2.20) hence the following:

$$est\left[\frac{\partial y}{\partial \sigma}(\sigma_j)\right] = \frac{\partial y}{\partial \sigma}(\sigma_j) + \frac{S_0(\sigma_j)}{k(\sigma_j)}\left[\left(\frac{\partial k}{\partial \sigma}(\sigma_j) - \frac{\partial f}{\partial \sigma}(\sigma_j)\right) * d_j^{(3)} + \frac{\partial f}{\partial \sigma}(\sigma_j) * d_j^{(2)}\right] \quad \text{(A2.21)}$$

When comparing the gradient equations (A2.21) with (A2.15) two conclusions can be drawn:

- The disturbance generated in the first experiment is no longer causing any problem
- The output of the first experiment with the disturbance is exactly what is needed as setpoint signal in the second experiment to generate an estimate of $\frac{\partial e_m}{\partial \sigma}$ [14].

A2.3.2 Input signal

One can generate the measurements of the process input by using the setpoint signals from equations (A2.16), (A2.17) and (A2.18). The signals are generated in the following manner:

In the first experiment the setpoint is set to a particular function of time, and then the actual output response $y^{(1)}$ is produced. At the same time the signal $r - y^{(1)}$ is generated and stored and used in experiment 2 as the next setpoint $r^{(2)} = r - y^{(1)}$ that produces the second actual response $y^{(2)}$. By so doing the estimate of the sensitivity function $\frac{\partial u}{\partial \sigma}(\sigma_j)$ can be generated from the following:

$$u(\sigma) = \frac{k(\sigma)}{1 + f(\sigma)G} r - \frac{f(\sigma)}{1 + f(\sigma)G} d = S_0(\sigma)[k(\sigma)r - f(\sigma)d] \quad \text{(A2.22)}$$

$$\frac{\partial S_0}{\partial \sigma}(\sigma) = -\frac{1}{k} T_0(\sigma) S_0(\sigma) \frac{\partial f}{\partial \sigma}(\sigma) \quad \text{(A2.23)}$$

Equation (A2.23) is derived by taking the derivative of the sensitivity function S_0 in (A2.8) and using (A2.7) and (A2.8) to simplify it.

The gradient of the input signal with respect to the controller parameter can be derived equation (A2.22) it follows that

$$\frac{\partial u}{\partial \sigma}(\sigma) = S_0(\sigma)\left[\frac{\partial k}{\partial \sigma} r - \frac{\partial f}{\partial \sigma}(\sigma)\left[T_0(\sigma) r + S_0(\sigma) d\right]\right]$$

$$= S_0(\sigma)\left[\frac{\partial k}{\partial \sigma} r - \frac{\partial f}{\partial \sigma}(\sigma) y\right]$$

$$= S_0(\sigma)\left[\frac{\partial k}{\partial \sigma}(\sigma) - \frac{\partial f}{\partial \sigma}(\sigma) r + \frac{\partial f}{\partial \sigma}(\sigma)(r - y)\right] \quad \text{(A2.24)}$$

When the setpoint signals, which are defined in equation (A2.16), (A2.17) and (A2.18) are applied to the system in the three experiments the corresponding three sets of inputs are produced and can be logged:

$$u^{(1)}(\sigma_j) = S_0(\sigma_j)\left[k(\sigma_j) r - f(\sigma) d_j^{(1)}\right] \quad \text{(A2.25)}$$

$$u^{(2)}(\sigma_j) = S_0(\sigma_j)\left[k(\sigma_j)(r - y^{(1)}(\sigma_j)) - f(\sigma_j) d_j^{(2)}\right] \quad \text{(A2.26)}$$

$$u^{(3)}(\sigma_j) = S_0(\sigma_j)\left[k(\sigma_j) r - f(\sigma_j) d_j^{(3)}\right] \quad \text{(A2.27)}$$

$u^{(1)}(\sigma)$ is a perfect realization of $u(\sigma)$ meaning that the input signal obtained in experiment 1 is equivalent to $u(\sigma)$ the one needed in equation (A2.11) for computation of the criterion minimization process. Therefore

$$u(\sigma) = u^{(1)}(\sigma) \quad \text{(A2.28)}$$

The estimation of $\frac{\partial u}{\partial \sigma}(\sigma_j)$ can be expressed as the following:

$$est\left[\frac{\partial u}{\partial \sigma}(\sigma_j)\right] \cong \frac{1}{k(\sigma_j)}\left[\left(\frac{\partial k}{\partial \sigma}(\sigma_j) - \frac{\partial f}{\partial \sigma}(\sigma_j)\right) u^{(3)}(\sigma_j) + \frac{\partial f}{\partial \sigma}(\sigma_j) u^{(2)}(\sigma_j)\right] \quad \text{(A2.29)}$$

The equation (A2.29) is an approximation of equation (A2.24), and it has the generated experiment input signals $u^{(2)}$ and $u^{(3)}$ from equation (A2.26) and (A2.27) respectively.

When looking at equation (A2.29) and equation (A2.24) and comparing the two, the above equation can be rewritten as shown below:

$$est\left[\frac{\partial u}{\partial \sigma}(\sigma_j)\right] = \frac{\partial u}{\partial \sigma}(\sigma_j) - \frac{f(\sigma_j)S_0(\sigma_j)}{k(\sigma_j)}\left[\frac{\partial f}{\partial \sigma}(\sigma_j)d_j^{(2)} + \left(\frac{\partial k}{\partial \sigma}(\sigma_j) - \frac{\partial f}{\partial \sigma}(\sigma_j)\right)d_j^{(3)}\right] \ldots \text{(A2.30)}$$

A2.3.3 Summary of the IFT algorithm:

With a controller $c(\sigma) = \{k(\sigma), f(\sigma)\}$ operating on the plant, the response signals $y^{(1)}(\sigma), y^{(2)}(\sigma), y^{(3)}(\sigma)$ given by equation (A2.16), (A2.17) and (A2.18) respectively need to be generated. The input signals $u^{(1)}(\sigma), u^{(2)}(\sigma), u^{(3)}(\sigma)$ need to be generated as well. These are given by equation (A2.25), (A2.26) and (A2.27) respectively. The next step is to compute $e_m(\sigma), est\left[\frac{\partial y}{\partial \sigma}(\sigma)\right], u(\sigma)$ and $est\left[\frac{\partial u}{\partial \sigma}(\sigma)\right]$ using equation (A2.4), (A2.20), (A2.28) and (A2.29). Based on these estimates the next controller parameter can be updated by the recursive equation (A2.12) $\sigma_{j+1} = \sigma_j - \gamma_j R_j^{-1}\frac{\partial J}{\partial \sigma}(\sigma_j)$.

A2.4 SELECTION OF MATRIX R_j

There are a number of choices when it comes to this matrix R_j, though it has to be positive definite matrix. The matrix has to ensure that it gives the negative gradient direction, in this research two types are investigated. The first one is the identity matrix while the second one is given by this equation:

$$R_j = \frac{1}{N}\sum_{t=1}^{N}\left(est\left[\frac{\partial y}{\partial \sigma}(\sigma_j)\right]est\left[\frac{\partial y}{\partial \sigma}(\sigma_j)\right]^T + \lambda * est\left[\frac{\partial u}{\partial \sigma}(\sigma_j)\right]est\left[\frac{\partial u}{\partial \sigma}(\sigma_j)\right]^T\right) \ldots\ldots\ldots \text{(A2.31)}$$

where the first term in the bracket is the dot product of the estimation of the gradient of the output response y with itself, while the second term is the dot product of the estimation of the gradient of the input signal u with itself and the latter result is multiplied by a weighting factor lamda λ. The role of lamda λ is to scale the second term in equation (A2.3.1)