DESIGN OF A DEDICATED IFT MICROCONTROLLER

by

GRAYSON HIMUNZOWA

Student number: 214025179

Submitted in fulfilment of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN ENGINEERING (MECHATRONICS)

in the

FACULTY OF ENGINEERING

at the

NELSON MANDELA UNIVERSITY

Supervisor: Professor Farouk Smith

2017

DECLARATION

NAME: **GRAYSON HIMUNZOWA**

STUDENT NUMBER: 214025179

QUALIFICATION: Doctor of Philosophy in Engineering (Mechatronics)

In accordance with rule G5.6.3, I hereby declare that the above-mentioned thesis is my own work and that it has not previously been submitted for assessment to another university or for another qualification.

.....

SIGNATURE

GRAYSON HIMUNZOWA

11/04/18

DATE:

SUMMARY

The design of a Dedicated IFT Microcontroller originated from the successful implementation of the Iterative Feedback Tuning (IFT) technique into the Digital Signal Processor microcontroller (DSP56F807C) at the University of Cape Town in 2006. However, implementation of the IFT technique on a general-purpose microcontroller is neither optimal, nor a cost-effective exercise, as most of the microcontroller peripherals remain unused, and drain energy for doing nothing. In addition, microcontrollers and DSPs are software-driven devices whose nature is sequential in executing algorithms, and hence have a significant effect on the bandwidth of the closed-loop control. To mitigate the said problem, the design of a Dedicated IFT Microcontroller is proposed in this thesis. To accomplish this goal, the preliminary task was to explore the IFT theory and its applications, followed by a review of the literature on FPGA design methodology for industrial control systems, Microcontroller design principles, and FPGA theory and trends. Furthermore, a survey of electronic design automation (EDA) tools and other application software was also conducted.

After the literature review, the IFT was investigated exhaustively by applying it to three types of plants, namely: a DC motor, an oscillatory plant, and an unstable plant. Each of these plants were tested using three types of initial controllers, namely heavily-damped, critically damped and under-damped initial controllers. The plants were also tested by varying the amplitude of the reference signal, followed by using a single-step signal of constant amplitude of one volt. The intention of exploring all of these possibilities was meant to firmly expose the IFT boundaries of applicability, so that the final product would not be vulnerable to unnecessary post-production discoveries.

The design methodology adopted in this research was a popular hierarchical and modular top-down procedure, which is an array of abstraction levels that are detailed as: system level, behavioural level, Register-Transfer Level (RTL) and Gate level. At system level, the Dedicated IFT Microcontroller was defined. Thereafter, at behavioural level, the design was simulated using VHDL, created by porting the LabView IFT code to the Xilinx EDA tool. At the RTL, the synthesisable VHDL code utilising fixed-point number representation was written. The compiled bit file was downloaded onto National Instruments (NI) Digital Electronics FPGA Board featuring

the Spartan 3 series FPGA. This was tested, using a method known as simulation in the hardware.

The key contribution of this thesis is the experimental validation of the IFT technique on FPGA hardware as it has never been published before, the work described in chapter four and five. The other contribution is the analysis of 1DOF IFT technique in terms of limitations of applicability for correct implementation, which is the main work of chapter three.

This work could be used to explore other computational methods, like the use of floating-point number representation for high resolution and accuracy in numerical computations. Another avenue that could be exploited is Xilinx's recent Vivado methodology, which has the capacity for traditional programming languages like C or C++, as these have in-built floating-point number capability.

Finally, out of this work, two papers have already been published by Springer and IEEE Xplore Publishers, and a journal paper has also been written for publication in the *Control Systems Technology* journal.

ACKNOWLEDGEMENTS

- I dedicate this thesis to the almighty God for giving me life, energy and zeal throughout my period of study.
- Thank you to my supervisor (Professor Farouk Smith) for guidance and also financial support in the procurement of research equipment and reviewing the thesis.
- Thank you to Professor EI-Hossein Khaled for welcoming me to NMMU.
- Thank you to Professor Martin Braae, formerly from the University of Cape Town, for assistance with the technical issues of the project.
- Thank you to Professor Mundia Muya and Doctor Ackim Zulu (both from the University of Zambia) for supporting this study programme at home.

LIST OF FIGURES

Page

LIST OF TABLES

Page

LIST OF ACRONYMS

1-DOF, 2-DOF	One Degree of Freedom, Two Degrees of Freedom
BFGS	Broyden - Fletcher - Goldfarb - Shanno
DSP	Digital Signal Processor
PID, PI	Proportional Integral Differential, Proportional Integral
DC, ISE	Direct Current, Integrated System Environment
VLSI	Very Large Scale Integration
EDA	Electronic Design Automation
IC, NI	Integrated Circuit, National Instruments
FSM, ASM	Finite State Machine, Algorithmic State Machine
VHDL	Very High Speed IC Hardware Description Languages
HLS	High Level Synthesis
ADC	Analogue-to-Digital Converter
DAC	Digital-to-Analogue Converter
ZOH	Holding Circuit
FPGA	Field-Programmable Gate Array
RTL	Register Transfer Level
IFT	Iterative Feedback Tuning
PWM	Pulse Width Modulation

TABLE OF CONTENTS

Page

CHAPTER THREE

STUDY AND VALIDATION OF IFT TECHNIQUE

CHAPTER 4

CHAPTER 5

TESTING OF A DEDICATED IFT MICROCONTROLLER

CHAPTER 6

CONTRIBUTION AND FUTURE RESEARCH

CHAPTER ONE

RESEARCH PROPOSAL

1.1 INTRODUCTION

The design of a Dedicated IFT Microcontroller was conceived as the result of the successful implementation of the Iterative Feedback Tuning (IFT) technique into the Digital Signal Processor microcontroller (DSP56F807C) at the University of Cape Town in 2006. However, the implementation of the IFT technique on a general-purpose microcontroller is not optimal, nor a cost-effective exercise, as most of the microcontroller peripherals remain unused, meaning that such extra hardware is a cost in energy. Microcontrollers and DSPs are software-driven devices whose nature is sequential in executing algorithms, and hence have a significant effect on the bandwidth of the closed loop control. These problems can be mitigated by developing a dedicated or hard-wired IFT Microcontroller that contains only necessary peripherals, and having the hardware IFT execute in parallel. To realise this, the IFT Microcontroller was developed on a Field Programmable Gate Array (FPGA). This concept would not only resolve the above mentioned problems, but also improve performance (in terms of power consumption and execution speed).

1.2 MOTIVATION OF THE RESEARCH

A vast number of IFT applications currently existing or described in [1] and other literatures, is one of the motivating factors that triggered this research. For example: tuning of PID controller parameters [2, 3, 4, 7], application to Zang sugar cane process plant [5], application to non-linear systems [6], application to model-free design with guaranteed stability [8], application to DC-servo with backlash [9], application in design of robust controllers [9], to mention a few. The IFT technique is one of the most recent methods that can mitigate the above-mentioned problems efficiently, but no stand-alone hardware or commercial product exists that can implement IFT technique.

Finally, due to rapid progress in very large-scale integration (VLSI) technology and electronic design automation (EDA) techniques in recent years, an opportunity for the development of complex and high-performance controllers for industrial electronic systems has been created. Nowadays, the design engineer is using modern EDA tools to design, simulate, and verify a design before committing to hardware [12].

The development of a Dedicated IFT Microcontroller at the level of an integrated circuit (IC) is important, as this can lead to commercialisation of the IFT technique or development of industrial product with compactness, and excellent control performance at reduced cost.

1.3 PROBLEM STATEMENT

Problems resulting from modelled and external uncertainty always deteriorate the control performance of proportional-integral-differential (PID) controllers that are widely used in industrial control. In the absence of self-tuning, the fixed PID parameters can hardly adapt to uncertainty or time-varying systems [13]. In addition to this, there is a particular case of industrial interest in which tuning of a proportional-integral (PI) or PID controller needs to be adaptive since classical approaches contain a number of fundamental problems [1], such as:

- the amount of offline tuning required;
- the assumption on the plant structure;
- the issue of system stability; and
- the difficulties in dealing with nonlinear, large time-delayed and time-variant plants.

Hence, the main objective of this thesis is to design a Dedicated IFT Microcontroller with compactness and improved performance to resolve the above problems existing in industrial control.

1.4 HYPOTHESIS

Our hypothesis statement is given as follows:

We believe that a dedicated IFT Microcontroller with improved performance can be developed into an FPGA device.

1.5 OBJECTIVES

The main objective of this research is to design a Dedicated IFT Microcontroller with improved controller performance using FPGA hardware, and is achieved through the following specific objectives:

- Study and validate the IFT technique with a view to developing a novel hardware (Dedicated IFT Microcontroller) for tuning proportional-integral (PI) controller parameters. Both IFT technique of one degree of freedom (1DOF) and two degrees of freedom (2DOF) will be studied but only 1DOF will be validated in order to simplify the dedicated IFT Microcontroller hardware.
- Design the architecture (data path) and finite-state machine (FSM) for a Dedicated IFT Microcontroller and develop VHDL code for it. The general purpose architecture is avoided for reasons of high speed and power consumption as compared with dedicated architecture.
- Testing of a Dedicated IFT Microcontroller with a simulated DC motor. A physical DC motor is avoided due to want of keeping the research narrow but detailed.

1.6 RESEARCH METHODOLOGY

Our research method progressed as follows:

- Research proposal development through intensive review of literature in the areas of adaptive control algorithms (with main emphasis on IFT algorithm theory and applications), controllers based on FPGA hardware and FPGA Design Methodology for Industrial Control System.
- A study of IFT technique and FPGA architecture was carried out. Since FPGA was a platform on which controller hardware was developed, a thorough survey across a wide family of FPGAs was conducted in order to select a suitable FPGA that can accommodate a highly complex adaptive control technique (IFT).

- design of a Dedicated IFT Microcontroller and development of VHDL code; simulation and experimentation; analysis and discussion of results.
- optimisation of a Dedicated IFT Microcontroller architecture; simulation and experimentation; analysis and discussion of results.
- testing of the Dedicated IFT Microcontroller to a novel process.
- analysis and discussion of results.
- thesis report write-up.

1.7 PLAN OF DEVELOPMENT

Chapter two discusses the literature review on IFT technique theory and applications, FPGA-based controllers, and FPGA Design Methodologies for the Industrial Control System. Chapter three reports the study and validation of the IFT technique, with a view to creating a Dedicated IFT Microcontroller. Chapter four describes the design of a Dedicated IFT Microcontroller. Chapter five describes the testing of a Dedicated IFT Microcontroller. New contributions, recommendations and Chapter Summarys are provided in Chapter six.

CHAPTER TWO

LITERATURE REVIEW

2.1 SEARCH OBJECTIVE

Since the main objective of the thesis was to design a Dedicated IFT Microcontroller, the preliminary task was to explore the IFT theory and its applications, mainly its recent advances: the improved algorithm, its extension, and the combination with other algorithms. In addition, a great deal of literature on FPGA design methodology for industrial control systems, Microcontroller design principles, and FPGA theory and trends were reviewed and this formed the major part of the project. Furthermore, a survey of electronic design tools and other application software was done. These included Quartus II Web Edition, ModelSim-Altera Web Edition, Xilinx ISE and Vivado HLS.

2.2 IFT TECHNIQUE APPLICATIONS SURVEY

A vast number of IFT applications currently existing or described in literature are a motivating factor in the pursuance of the thesis entitled "Design of a Dedicated IFT Microcontroller". Some of these applications surveyed are outlined below:

- In [2], IFT was applied to optimise the Electronic Throttle Control (ETC) system of an engine. The application showed that the IFT provides very good performance for controller tuning. The system was implemented on a Pentium processor and was carried out experimentally. This has clearly shown the need to develop a dedicated hardware for its implementation, in order to reduce the cost and improve its performance in the long run.
- In [3], IFT was applied to tune PID parameters in applications where the objective is to achieve a fast response, to set point changes, and the performance of IFT-tuned PID controllers was compared to the performance of the classical tuned PID-controllers. It revealed better results (faster settling time) than the later. The work was carried out through simulation examples. Thus, a simple and efficient PID-tuning scheme was developed in this particular work though non-experimental work, however, no experimental verification was conducted.

- In [4], a relay auto-tuning of PID controllers using IFT was applied to a process control problem in which the PID controller was auto-tuned to give specific bandwidth and phase margin. The algorithm was tested in the laboratory on a coupled tank, and the theoretical results were demonstrated to be observed in practice, confirming the viability of the IFT technique.
- In [15], the extension of IFT as a tuning algorithm was presented. informative data was used to improve the convergence properties of the method, and reduce the total number of required plant experiments, especially when tuning for disturbance rejection. This was achieved through application of an external probing signal in the tuning algorithm. The technique was further used to guarantee nominal stability and to improve the parameter update using a line search algorithm for determining iteration step size, through the use of Levenberg-Marquardt optimisation. The proposed algorithm was compared to classical formulation in the simulation study of the disturbance rejection problem. It was found that perturbed IFT is an advantage when tuning for disturbance rejection.
- In [16], tuning of robot joint controllers using IFT was considered, different IFT schemes were validated in simulation, and real experiments on an industrial robot manipulator were conducted. From a practical point of view, the scheme therefore offers several advantages: it is straightforward to apply the direct optimal tuning algorithm, particularly to basic control loops in the process industry, which are typically PID loops. In addition, IFT has high potential for tuning of controllers applied to non-linear systems, which is currently a challenge in industrial control. With favourable results having been obtained, the need has arisen to develop a chip that implements the technique.
- In [17], IFT was applied to tune a second-degree-of-freedom (2DOF) PID-controller to minimise the given quadratic cost function of a system output error and control effort. The tuning effort was divided into two parts. First, the classical 1DOF PID controller was iteratively tuned and the remaining parameters of 2DOF PID were then tuned, independently in the next iterative procedure; and second, the Broyden-Fletcher-Glodfard-Shanno (BFGS) method as a quasi-Newton method was employed. From the numerical simulations and the experiments, the

effectiveness of the proposed method was shown. For experimental purposes, the controller was implemented into the PC, a clear indication that IFT could be feasible to be implemented into a custom device such as a Microcontroller or an FPGA.

In addition to the above surveyed IFT applications, FPGA-based controllers were also surveyed and a sample of them are listed in the next subsection. The purpose of this survey was to ascertain the feasibility and benefits of using an FPGA as a platform for developing the Dedicated IFT Microcontroller.

2.3 FPGA-BASED CONTROLLERS LITERATURE REVIEW

The surveyed literature on FPGA-based controllers and FPGA Design Methodology for Industrial Control Systems is summarised as follows:

- In [12], the state of the art of FPGA based controller design was reviewed. The design methodologies utilised three main design rules given as algorithm refinement, modularity, and systematic search for the best compromise between control performance and architectural constraints. Two complete case studies on the benefits of FPGA implementation, when using the proposed system modelling and design methodology were presented. A control algorithm, when implemented in an FPGA, can have a short execution time due to the high degree of parallelism of its architecture. Another perspective on FPGA design is to use FPGAs with analogue to digital converter (ADC) and digital to analogue converter (DAC) imbedded in them. Only Microsemi have fused an ADC into their FPGA to date.
- In [13], a motion control system, using a radial basis function neural network (RBF NN) self-tuning PID controller for X-Y table was realised in an FPGA, to create a motion control-integrated circuit (IC). This was done by hardware/software co-design (a necessary concept nowadays), and as a result the final product was made more compact, robust, flexible, and costeffective. The work was experimental and revealed favourable results. An IC that implements self-tuning PID was developed into the FPGA device.

- In [14], a novel technique for implementation of an efficient FPGA-based PID-controller for motion control of a permanent magnet DC motor is presented. This implementation technique circumnavigates the problem of interfacing an analogue-to-digital converter in real time. The implementation was done on the Xilinx Spartan 3 FPGA Board from National Instruments. The design showed significant improvement over the present way of implementing digital controllers into microcontrollers.
- In [18], a digital controller for a switching power converter was implemented in an FPGA device. The digital hardware using Very Large Scale Integrated Circuit Hardware Description Language (VHDL) with floating-point arithmetic were both verified by simulation and experiments. The results showed that the designed system meets its specification, and the simulations match the experimental results closely.
- In [19], a modular design of embedded feedback controllers was studied by utilising simulations into the FPGA hardware. To this end, a novel distributed-arithmetic (DA)-based PID-controller algorithm was proposed and integrated into a digital feedback control system. The DA-based PID-controller demonstrated 80% savings in hardware utilisation and 40% savings in power consumption, which is desirable in embedded control applications. Simulations and experiments were tested using the system; it demonstrated good closed-loop stability and performance.
- In [20], a methodology was presented based on the control system L1 or l1 norms for computing the appropriate number of bits to represent each quantity when using fixed-point number representation. The methodology was shown to be effective for designing hardware for both shift-form and delta-form representations of the compensator, and was applied to a magnetic bearing control system. In this example, the delta-form realisation required less hardware than shift-form realisation.
- In [21], a simple auto-tuning technique for digitally controlled dc-dc for synchronous buck converters was proposed; this was an approach based on a relay feedback method. By the use of the iterative procedure, the tuning of PID parameters as a

result was obtained directly by including the controller in the relay feedback. Experimental investigations were conducted using the FPGA platform.

2.4 Chapter Summary

Considering the surveyed literature within the subject of the IFT technique applications based on FPGA technology, only [21] is directly related to the IFT technique, but its application is specific to Buck converters. The authors does not investigate, nor specify, whether the technique could be used for other plants.

CHAPTER THREE

STUDY AND VALIDATION OF IFT TECHNIQUE

3.1 INTRODUCTION

In this chapter, the IFT technique is studied and validated. This is accomplished by carrying out an overview of the technique's basic theory, followed by simulation on NI LabView software, mainly for the purpose of testing and validating it before committing it to hardware. The reason for the choice of LabView software as the platform for simulation, is that programs running on the LabView platform can easily be ported to VHDL with minimal modifications. To ensure an exhaustive investigation, the IFT is tested and validated by applying it to three types of plants, namely: a DC motor, an oscillatory plant, and an unstable plant. It is also tested by varying the amplitude of the reference signal, and also by using the single-step signal. The intention of exploring all of these possibilities is meant to firmly expose the IFT boundaries of applicability so that the final product is not vulnerable to unnecessary post-production discoveries such as operational errors.

3.2 ITERATIVE FEEDBACK TUNING TECHNIQUE OVERVIEW

Since the Dedicated IFT Microcontroller is based on the IFT technique, we consider a comprehensive overview of IFT basic theory in this section.

The IFT technique is purely a data-driven and gradient-based approach for optimising controller parameters, without full knowledge of the plant [9, 25]. It yields an unbiased estimate of the model (meaning that the IFT finds the correct controller for the plant in operation) [25]. Its concept derives from the given controller structure (given in advance), and the specification of a criterion or objective function of a Linear Quadratic Gaussian (LQG). The LQG is formed by data that is collected from closed-loop experiments, in which the number of experiments depend on the degree of freedom of the controller in question. For example, a second-degree-of-freedom (2DOF) controller would require three experiments to be performed at each stage of the iterative design, while a first-

degree-of-freedom (1DOF) controller would require only two batch experiments expected to be run at each stage of the iterative design. Thereafter, either of the gradient-based local minimisation techniques, such as steepest descent, the Hessian, Gauss-Newton, or Quasi-Newton [22], can search a minimum of the criterion LQG.

In this section we describe the IFT technique of a second-degree-of-freedom controller ($C = \{C_r, C_y\}$), so that investigations of implementation issues relevant to the design of the Dedicated IFT Microcontroller are simplified. The IFT technique is implemented on the discrete linear time invariant system G, as shown in figure 1.

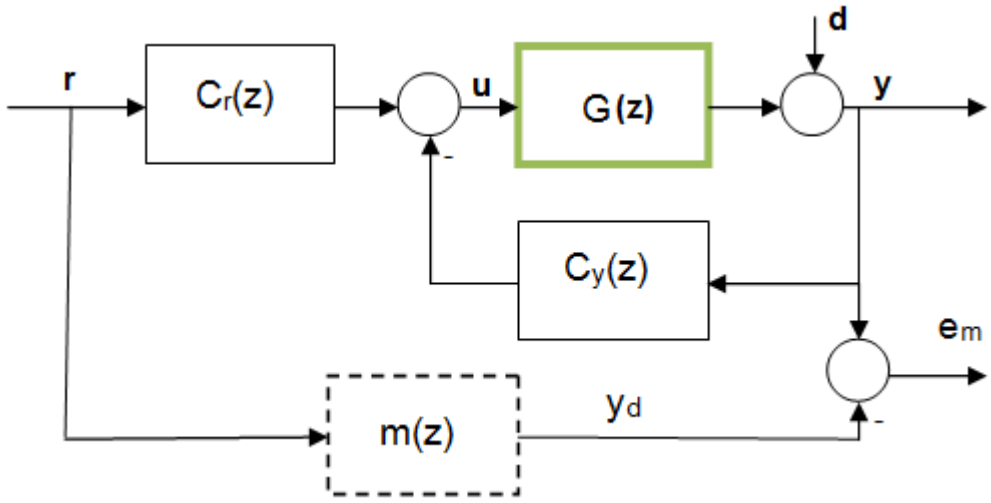

Figure 1: Block diagram of a closed loop system of a two-degree-of-freedom controller

- {r(t)} is the reference signal; the argument t is added to denote the sampling period,
- u(t) and y(t) are control and output signals respectively.
- $C_r(z,\rho)$ and $C_y(z,\rho)$ are linear time-invariant transfer functions;
- ρ is a vector of the PI controller parameters;
- $m(z)$ and e_m are the desired model for the controlled system, and the modelling error respectively.

The internal closed loop signals can be described via sensitivity function S and transfer function T_r, as follows:

$$u(t) = C_r * S * r(t) - C_y * S * d(t) \quad (1)$$

$$y(t) = T_r * r(t) + S * d(t) \quad (2)$$

Where $S = \frac{1}{1 + G * C_y}$ and $T_r = \frac{G * C_r}{1 + G * C_y}$

The output of the desired model is formulated from figure 1, and given in equation (3).

$$y_d = T_d * r(t) \quad (3)$$

Where $T_d = m(z)$

Finally, the modelling error is computed from equation (4).

$$e_m(t) = y(t) - y_d \quad (4)$$

Figure1 can be simplified into an IFT of a 1DOF controller if $C_r \cong C_y = C$. The 1DOF controller runs only two experiments, and we will use it to formulate the hardware architecture for the design of a Dedicated IFT Microcontroller later, due to its simplicity compared to the 2DOF controller. The signals in equations (1), (2), (3), and (4) are measured in experiment#1, however, signals like the closed-loop error ($e(t) = r(t) - y(t)$), and the modelling error $e_m(t)$, are stored, since they are used to generate a reference signal and the criterion function in experiment#2. The reference signal $r(t)$ in experiment#1 is also used as a reference signal in experiment#3 for the 2DOF controller.

The IFT procedure for the 2DOF controller experiments is as follows: At iteration i of the controller tuning algorithm, the controller $C(z, \rho_i) = \{C_r(z, \rho_i), C_y(z, \rho_i)\}$ operates on the actual plant. Thereafter, three experiments are run, with each experiment storing a sequence of N-length data, as depicted in table 1.

Experiment#1 and experiment#3 consist only of gathering data under normal operating conditions, with the closed loop using the same reference signal for the said experiments [9]. Experiment#2 collects data from the closed-loop system with a reference signal($r - y\ (\rho)$) derived from experiment#1. From now on we denote the N-length reference signal and the corresponding output signals by $\{r_i^{\,j}\}$ and $\{y_i^{\,j}(\rho_i)\}$ respectively. Where $i = 0,1,2,...,N$, a sample number or iteration number, and $j = 1,2,3$ is an experiment number.

Table 1: IFT signals for experiment#1, experiment#2 and experiment#3

Experiment# j	Input signal	Measured signals	Stored signals
1.	$r_i^{\,j} = r$	$y^j(\rho_i) = T(\rho_i) * r_i^{\,j} + S(\rho_i) * d_i^{\,j}$ $u^j(\rho_i) = S(\rho_i) * C_r(\rho) * r_i^{\,j} - C_y(\rho_i) d_i^{\,j}$ $e_m = y^j - y_d$ $e^j = r_i^{\,j} - y^j$	e^j, e_m and y^j
2.	$r_i^{\,j} = e^j - y^j(\rho_i)$	$y^j(\rho_i) = T(\rho_i)(r_i^{\,j} - y^j(\rho_i)) + S(\rho_i) * d_i^{\,j}$ $u^j(\rho_i) = S(\rho_i) * C_r(\rho) * (r_i^{\,j} - y^j) - C_y(\rho_i) d_i^{\,j}$ $e^j = r_i^{\,j} - y^j(\rho_i)$	$\frac{\partial e_m}{\partial \rho}(\rho_i)$ and $\frac{\partial u^j}{\partial \rho}(\rho_i)$
3.	$r_i^{\,j} = r_i^{\,j}$	$y^j(\rho_i) = T(\rho_i) * r_i^{\,j} + S(\rho_i) * d_i^{\,j}$ $u^j(\rho_i) = S(\rho_i) * C_r(\rho) * r_i^{\,j} - C_y(\rho_i) d_i^{\,j}$ $e^j = r_i^{\,j} - y^j(\rho_i)$	Same as in experiment#1

All the signals from figure 1 are tabulated in table 1 as $r_i^{\,1}, e_i^{\,1}, u_i^{\,1}, y_i^{\,1}, d_i^{\,1}$, denoting the reference, error, control, plant output, and disturbance signals in experiment#1

respectively; for experiment#2, these signals are denoted by $r_i^2, e_i^2, u_i^2, y_i^2, d_i^2$, and by $r_i^3, e_i^3, u_i^3, y_i^3, d_i^3$ in experiment#3.

After the experiments, the processing stage commences with the gradient calculation, followed by the selection of a positive definite matrix R, and finally, the updating of the controller parameters. We illustrate the sequence, beginning with the formulation of the criterion function.

The modelling error and the controller output signals that are measured from the experiments, are assembled together with filters and constants to form a criterion function, as given in equation (5).

$$J(\rho_i) = \frac{1}{2N} E\{\sum_{i=0}^{N} [(L_e * e_m(\rho_i))^2 + \lambda * (L_u * u^1(\rho_i))^2]\} \quad (5)$$

Where L_e and L_u are frequency-dependent weights or filters that penalise the modelling error e_m and control input u^1, according to the designer's needs. E{…} is the expectation taken with respect to stochastic disturbances that enter the process and thus affect the closed loop.

Each version of the criterion function defines a specific type of IFT. Some types of IFT criterion functions are, for example, where the cost function is obtained from the modelling error e_m, as in equation (5), from the traditional error e, or from the traditional error without the control signal u included in the cost function..

Many specific IFT criterion functions can be formulated as in equation (5), by varying the type of desired model. The IFT criterion function, driven by the traditional error, depends heavily on the model of the plant, since different plants would produce different traditional errors. The use of the traditional error to formulate the cost function is achieved by making the desired model, $m(z) = 1$.

The objective of the IFT is to find an optimal set of parameters ρ, which minimises the criterion function J, as given in equation (6). This is the minimisation of the cost function, also given in equation (7) later, in another form [11].

$$\rho_{i+1} = \arg\min J(\rho_i) \tag{6}$$

The major stumbling block for the solution of this optimal control problem is the computation of the gradient of the criterion function with respect to controller parameters [9]. The gradient of the criterion function is thus given by equation (7), with the frequency weighting filters set to one (thus $L_e = L_u = 1$) for simplicity, but they are important, since they give extra flexibility in optimisation of the criterion function.

$$\frac{\partial J}{\partial \rho}(\rho_i) = \frac{1}{N} * \sum_{i=0}^{N} [e_m(\rho_i) * \frac{\partial e_m}{\partial \rho}(\rho_i) + \lambda * \sum_{i=0}^{N} u^1(\rho_i) * \frac{\partial u^1}{\partial \rho}(\rho_i) \tag{7}$$

The necessary condition for optimality of the parameters is $\frac{\partial J}{\partial \rho}(\rho) = 0$. Hence, to be able to compute such an equation for ρ, the model of the plant, in totality, is required. In this respect, the plant model is generally not known in most industrial applications, and therefore, the required analytical form cannot be obtained. The main contribution of the IFT was to circumvent the said problem, by offering a procedure to calculate the gradient $\frac{\partial J}{\partial \rho}(\rho)$ directly from closed-loop data. Hence, the IFT is entirely a data-based paradigm, implying that it can easily be implemented into a DSP microcontroller (best for algorithms [12]), as opposed to an FPGA device that has parallel attributes.

If the gradient $\frac{\partial J}{\partial \rho}(\rho)$ could be computed, then the solution of equation (7) would be obtained by stochastic approximation of Robins and Monro of 1951 [15], given as

$$\rho_{i+1} = \rho_i - \gamma_i * R_i^{-1} \frac{\partial J}{\partial \rho}(\rho_i) \tag{8}$$

In equation (8), γ (gamma) is a positive real scalar that determines the step size. It can be fixed or established by a line search. The matrix R is some appropriate positive definite matrix that determines the amplitude of the steps (or the step sizes), in the direction of each parameter provided by either the steepest descent, the Gauss Newton method, or the Quasi Newton method. The choice of R as an identity matrix renders a steepest descent gradient that is normally negative and also slow to converge, but would be beneficial for power consumption, because it does not require much resource of the FPGA device. The choice of R as a Hessian matrix is not feasible – even in the literature, the approximation of the Hessian is preferred over the Hessian matrix directly [3, 4, 5, 6, 8]. This can be generated by the Gauss Newton method (as shown in equation (9)), or the Quasi Newton method [25, 26], however, hardware implementation into an FPGA would be complicated and would demand high power consumption since huge FPGA resource shall be required.

$$R_i = \frac{1}{N} * \sum_{i=1}^{N} (E\left[\frac{\partial e_m}{\partial \rho}(\rho_i)\right] E\left[\frac{\partial e_m}{\partial \rho}(\rho_i)\right]^T + \lambda * E\left[\frac{\partial u^1}{\partial \rho}(\rho_i)\right] E\left[\frac{\partial u^1}{\partial \rho}(\rho_i)\right]^T \qquad (9)$$

The Quasi Newton choice of R is covered widely in [11, 17].

3.2.1 Generation of the gradients for modelling error and controller signal

The main difficulty with IFT is the problem with calculation of the gradient of the criterion function J, with respect to controller parameters, since the model of the plant is not known. This problem is resolved through filtering the modelling error and the controller output signals. The output of these filters are the gradients of the respective signals, which are measured and stored to compute the gradient of the criterion function, as shown in equation (7).

Noting that

$\frac{\partial e_m}{\partial \rho}(\rho_i) = \frac{\partial y}{\partial \rho}(\rho_i)$, since it is the plant that causes variation in the modelling error e_m, we state the expressions in equations (10) and (11). These are standard IFT equations, derived in [9, 26].

$$\frac{\partial e_m}{\partial \rho}(\rho_i) = \frac{\partial}{\partial \rho}(y^1) = \frac{\partial}{\partial \rho}(T(\rho_i) * r + S(\rho_i) * d_i)$$

$$= \frac{1}{C_r(\rho)} * \left[(\frac{\partial C_r}{\partial \rho}(\rho_i) - \frac{\partial C_y}{\partial \rho}(\rho_i)) y^3 + \frac{\partial C_y}{\partial \rho}(\rho_i) y^2 \right] \quad (10)$$

$$\frac{\partial u^1}{\partial \rho}(\rho_i) = \frac{\partial}{\partial \rho}(u^1) = \frac{1}{C_r(\rho)} * \left[(\frac{\partial C_r}{\partial \rho}(\rho_i) - \frac{\partial C_y}{\partial \rho}(\rho_i)) u^3 + \frac{\partial C_y}{\partial \rho}(\rho_i) u^2 \right] \quad (11)$$

The first term in equation (10) is a filter, taking the output signal from the plant y^3 as input, and the second term in equation (10), also a filter taking y^2 as input. The output of the two filters are passed via two input adders to produce the gradient of the modelling error. Similarly, the first term of equation (11) is a filter, taking signals from the controller u^3 as input, and the second term also a filter taking an input signal u^2 from the controller. The output of the two filters are also passed via two input adders to produce the gradient of the controller output. It is shown here that gradients of a criterion function are signals measured from the closed loop system, and applied in equation (7) to compute the gradient of J with respect to controller parameters.

3.2.2 IFT technique of a 1DOF controller

Having described the IFT technique for a 2DOF controller, we present here the IFT technique for the 1DOF controller. This is aimed at implementation on the FPGA hardware, because its algorithm is simpler than the IFT technique of the 2DOF controller. As mentioned previously in section 3.2, it runs two experiments compared to three of IFT technique for the 2DOF controller. The choice of degree of freedom depends mainly on

application needs. For example, the IFT of the 2DOF controller is good for disturbance rejection, and the IFT for the 1DOF controller is good for reference signal tracking and regulation [26].

In this section, we develop a deterministic version of Iterative Feedback Tuning for the case of a simple PI controller, so that the design of the Dedicated IFT Microcontroller is simplified. The PI controller is chosen because it is a very common control law, and hence a reasonable starting point. It is also important to industrial applications [22].

We investigate a single input, single output control system of 1DOF control law, as shown in figure 2.

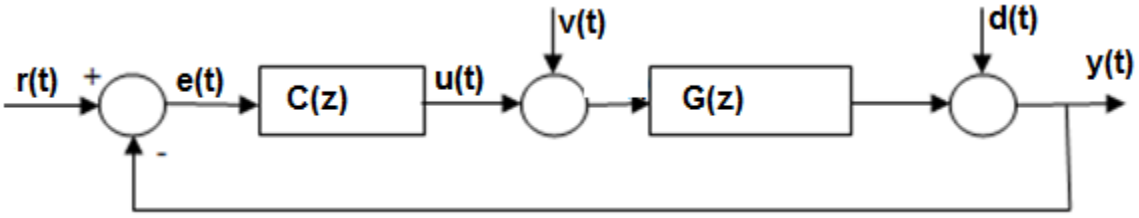

Figure 2: Closed Loop System for 1DOF controller

The reference, error, input disturbance, controller output, output disturbance, and plant output signals are represented by $r(t), e(t), v(t), u(t)$ $d(t)$ and $y(t)$ respectively. The plant $G(z)$ is a step invariant transformation of the DC motor ($\frac{1.01}{2s+1}$) used in [1]. This was chosen to validate the IFT technique, by comparing its performance on the FPGA, with that of [1]. $C(z)$ is the PI controller, which is a step transformation of the s-domain PI controller ($\rho_0 + \frac{\rho_1}{s}$). The internal closed loop signals can be described with sensitivity and transfer functions (considering $d(t) = 0$), as follows:

$$u(t) = C * S * r(t) - C * S * d(t) \quad (12)$$

$$y(t) = T_r * r(t) + S * d(t) \quad (13)$$

Where $S = \frac{1}{1+G*C}$ and $T_r = \frac{G*C}{1+G*C}$

We consider $d(t)$ as zero mean weakly stationary random noise.

Experiment#1 of the 1DOF controller measures and stores signals $e^1(\rho)$, $u^1(\rho)$, and $y^1(\rho)$ for N length of time. Rho is a constant for each set of two experiments (#1 to #2). However, it is changed after every experiment#2, thus rho is used as an argument. Experiment#2 is known as the gradient experiment, where the gradient of the modelling error and control input signals are measured and stored. Thereafter, the criterion function J is specified by utilising the measured signals in experiment#2, and exiperiment#1. After summing over N-length samples, the gradient ($\frac{\partial J}{\partial \rho}$) of the criterion function, needed to optimise or update the PI-controller parameters, is obtained.

3.2.3 Criterion function specification

Controller optimisation requires that a criterion function is defined, usually by expressing it in terms of J, as given in equation (5). We rewrite the criterion function expression, since it utilises the traditional error e^2, and not the modelling error e_m used in equation (5). The desired model is set to unity (desired model, $m(z) = 1$), implying that the cost function, that optimises the controller parameters, is as a result of the summation of the traditional error, and the controller output signals which was obtained from experiment#2, and given in equation (14) below.

$$J = \frac{1}{2N}\sum_{i=0}^{N}[(L_e * e^2(\rho))^2 + \lambda * (L_u * u^2(\rho))^2] \quad (14)$$

e^2 denotes the traditional error from experiment#2, and u^2 the control signal. The first term is the closed loop tracking error, and the second term the penalty on the control

effort from experiment#2. L_e and L_u in the first and second terms are frequency weighting filters that were initialised to one ($L_e = L_u = 1$) to simplify the hardware to be implemented on the FPGA, though it does give extra flexibility in the control action. The advantage of this criterion function is that it presents a good balance between overshoot and settling time [24]**,** and it simplifies the hardware design since the desired model is reduced to unity, It also makes the assumption for a plant structure become insignificant, since the desired model is reduced to a scalar. However, the main drawback of this criterion function, as compared to other minimisation criterion functions, is the difficulty of obtaining an analytical solution to the controller design [24].

After formulating the criterion function J, we minimise it by generating its gradient through summation, as given by equation (15).

$$\frac{\partial J}{\partial \rho}(\rho_i) = \frac{1}{N}\sum_{i=0}^{N}[e^2(\rho_i)\frac{\partial e^2}{\partial \rho}(\rho_i) + \lambda * u^2(\rho_i)\frac{\partial u^2}{\partial \rho}(\rho_i) \quad (15)$$

e^2 denotes a traditional error, and u^2 denotes a control signal . Once the gradient $\frac{\partial J}{\partial \rho}$ is computed, the solution of equation (15) is found in the same way it is done in the IFT of the 2DOF controller.

3.2.4 Algorithm formulation for 1DOF

The algorithm is formulated in two segments, namely experiment#1 and experiment#2.

Experiment#1 comprises equations (16), (17), (18) and (19). Equation (16) compares the reference signal and plant output signal, to produce the traditional error, as shown in figure 2.

$$e^1(t) = r_i^1(t) - y^1(t) \quad (16)$$

The PI-controller, without a holding circuit (ZOH), is required for the formulation of a filter ($\frac{1}{C}*\frac{\partial C}{\partial \rho}$) (in z-domain), for generating gradients needed for computation of the gradient of the cost function, as shown in equation (15). The reason for this is because the IFT loop from the traditional error to parameter update is purely a digital process, hence does not require usage of a DAC or ADC. The PI controller on the other hand, with a holding circuit (ZOH), is required because its output drives a DAC for actuating the plant. For this reason, the transformation of the PI controller transfer function into a digital equation, without a ZOH circuit, and with a ZOH circuit, is given in appendix V, yielding difference equations, as given in equations (17) and (18).

$$u_t = u_{t-1} + (\rho_0 + \rho_1) * e_t - \rho_0 * e_{t-1} \text{ [V]} \quad (17)$$

$$u_t = u_{t-1} + \rho_0 * e_t + (\rho_1 * T - \rho_0) * e_{t-1} \text{ [V]} \quad (18)$$

We also transform the plant into a digital equation with a ZOH circuit, since the output of a plant is connected to an ADC.

Transformation to the Z-domain begins by first setting the sampling time for the plant according to the governing principles of the Nyquist sampling theorem. This theorem states that the "sampling frequency f_s for a given source signal should be at least double the signal frequency". In our simulation, the sampling frequency adopted is 2.5 times the signal frequency. This is to ensure that the digital system mimics the continuous system. Thus

$$f_s = 2.5 * \frac{1}{T}$$

T is the time constant for the DC motor model or plant. In this case, the sampling frequency = 5 Hz, yielding a sampling time of 200 ms.With this sampling time in place, the plant digital equation is derived as shown in equation (19). Derivation details for the plant digital equation is presented in Appendix V.

$$y(t) = 0.904837 * y(t-1) + 0.09516 * u(t) \quad (19)$$

The equations of (20), (21), (22), (23), (24), (25), (26), (27), and (28), as shown below, describe the algorithm of experiment two.

The plant output signal $y^2(t)$, is compared with the reference signal $e^1(t)$ (from experiment#1), as achieved by equation (20).

$$e^2(t) = e^1(t) - y^2(t) \tag{20}$$

The PI-controller is reproduced in experiment#2, to generate a control signal, $u^2(t)$, and is given equation (21).

$$u_t^{\ 2} = u^2{}_{t-1} + \rho_0 * e_t^{\ 2} + (\rho_1 * T - \rho_0) * e^2{}_{t-1} \text{ [V]} \tag{21}$$

The plant output signal, $y^2(t)$ in experiment#2, is given in equation (22).

$$y^2(t) = 0.904837 * y^2(t-1) + 0.048057 * u^2(t) \tag{22}$$

The difference equations for generating the gradient of the error signal, $\frac{\partial e^2}{\partial \rho}$,with respect to controller parameters directly, were developed by Hjarmarsson in 1994 [5], and were an important advance in the development of adaptive controllers. This was achieved by passing the error signal through a filter $\frac{1}{C} * \frac{\partial C}{\partial \rho}$ and measuring the output, which yielded the gradient of the error signal required for specification of the cost function. This was developed in order to circumvent the difficulty in computation of the gradient of the cost function, as some quantities involved are unknown or only partially known [5].

The difference equations for the gradient signals are derived as follows:

The PI-controller, given in equation (17), is used to derive the gradient difference equations, as shown below:

$$\frac{1}{C}*\frac{\partial C}{\partial \rho}=\frac{1}{C}*\begin{bmatrix}\frac{\partial C}{\partial \rho_0}\\ \\ \frac{\partial C}{\partial \rho_1}\end{bmatrix}=\frac{z-1}{(\rho_0+\rho_1)*z-\rho_0}*\begin{bmatrix}1\\ \\ \frac{z}{z-1}\end{bmatrix}=\begin{bmatrix}\frac{z-1}{(\rho_0+\rho_1)*z-\rho_0}\\ \\ \frac{z}{(\rho_0+\rho_1)*z-\rho_0}\end{bmatrix}=\begin{bmatrix}\frac{de^2}{d\rho_0}/e^2\\ \\ \frac{de^2}{d\rho_1}/e^2\end{bmatrix}$$

Additionally, the difference equations are given in equations (23) and (24).

$$\frac{de^2}{d\rho_0}(t)=\frac{\rho_0}{\rho_0+\rho_1}*\frac{de^2}{d\rho_0}(t-1)+\frac{1}{\rho_0+\rho_1}*(e^2(t)-e^2(t-1)) \quad (23)$$

$$\frac{de^2}{d\rho_1}(t)=\frac{\rho_0}{\rho_0+\rho_1}*\frac{de^2}{d\rho_0}(t-1)+\frac{1}{\rho_0+\rho_1}*(e^2(t)) \quad (24)$$

The gradient of the criterion function, which is a summation of measured signals, $e^2(t)$, $u^2(t), \frac{de^2}{d\rho}$, and $\frac{du^2}{d\rho}$, are given in equations (25) and (26).

$$\frac{dJ}{d\rho_0}(t)=\frac{dJ}{d\rho_0}(t-1)+e^2(t)*\frac{de^2}{d\rho_0}(\mathrm{t}) \quad (25)$$

$$\frac{dJ}{d\rho_1}(t)=\frac{dJ}{d\rho_1}(t-1)+e^2*\frac{de^2}{d\rho_1}(t) \quad (26)$$

The parameter update expressions are given in equations (27) and (28).

$$\rho_0(t)=\rho_0(t-1)-\gamma_i*\frac{dJ}{d\rho_0}(\mathrm{t}) \quad (27)$$

$$\rho_1(t)=\rho_1(t-1)-\gamma_i*\frac{dJ}{d\rho_1}(t) \quad (28)$$

The equations from (16) to (28) outline the flow of the IFT technique applied to a controller of 1DOF, and can be mapped onto the Matlab or LabView platforms, for simulation purposes.

In section 3.3, this is done, with LabView, mainly to take advantage of LabView's FPGA add-on software, which can be compiled for downloading into an FPGA device.

3.3 SIMULATION OF IFT TECHNIQUE USING THE LABVIEW PLATFORM

This section is devoted to validating and testing the IFT "to destruction", so that the boundaries of its applicability can be obtained. This is done to avoid unnecessary post-production discoveries in the final product that may seriously impact on its commercial viability. In order to carry out a wide investigation, three different plants were used in the simulation, namely: the DC Motor that was used in [1], and is reproduced here (in equation (29) for ease of reference), the oscillatory plant given in equation (30), and the unstable plant given in equation (31).

$$g(s) = \frac{1.01}{2s+1} \tag{29}$$

$$g(s) = \frac{10}{s^2+3s+10} \tag{30}$$

$$g(s) = \frac{1}{2s-1} \tag{31}$$

Though IFT assumes that knowledge of the plant is not available or only partially known, the initial focus will be on the DC motor (a known plant), controlled by the PI-controller, for validation and testing purposes. The region of controller parameters are determined, which yields a stable closed-loop system in the z-plane by use of the root locus method, to provide preliminary information (especially the initial controller parameters for running the IFT) for the test and validation of the IFT technique, before simulation (on the LabView platform) is commenced. The PI controller and the DC motor in the z-domain with the holding circuits are derived in appendix V, and is stated here for easy of reference: the PI controller represented in the z-domain with a holding circuit is given as:

$$u(z) = \frac{\rho_0 * z + (\rho_1 * T - \rho_0)}{z-1}$$

Similarly, the DC motor model is also given as $\frac{0.09516}{z-0.9048057}$.

The open-loop PI controller applied to the DC motor is given as

$k(z)*gh(z)=\frac{\rho_0*z+(\rho_1*T-\rho_0)}{z-1}*\frac{0.09516}{z-0.9048057}$, yielding open-loop poles at $z=1$ and $z=0.9048057$, with one moving zero ($z=1-\frac{\rho_1}{\rho_0}T$) since it is dependent on the controller parameters which are varying due to the controller tuning, and the sampling time that is chosen by the designer in relation to the sampling time of the plant. That is, the sampling time for the controller should be faster than that of the plant. We determine the $\frac{\rho_1}{\rho_0}$ ratio, since the zero must lie in the region $-1<z<0.9048057$ to force the locus of the closed-loop poles inside the unit circle, in order to help to pull the poles outside, into the unit circle. To expose the stability region for the closed-loop system, the closed-loop system is derived for the PI controller applied to the DC motor. The derivation is given in appendix V for equation (32).

$$\frac{y(z)}{r(z)}=\frac{0.09516*\rho_0*(z+(\frac{\rho_1}{\rho_0}*T-1))}{z^2+(0.095*\rho_0-1.90)*z+(0.90+0.095*\rho_1*T-0.095*\rho_0)}\ [\text{V/V}] \qquad (32)$$

This equation is a closed-loop PI controller applied to the DC motor. It is expressed in such a way that the root locus can easily be applied to tracing the poles of the system. However, to perform the procedure, the controller parameters must first be chosen, because the zero of the open-loop system must lie within the range $-1<z<0.9048057$ on the left-hand side of the two open-loop poles in the z-plane. For each chosen zero, the parameters are computed, followed by running the root locus and verifying the region of stability. To determine the parameters, the zero location is chosen, and to simplify the design, the open-loop system zero should lie on real axis from $z=-0.9$ to $z=0.9$.

From this point on, rho0 and rho1 will be used in the text to denote ρ_0 and ρ_1 respectively. For each zero location chosen, controller parameters are computed using

the expression $1-\frac{\rho_1}{\rho_0}*T$ (the moving zero). rho0 is then chosen with the knowledge that, as it varies from 0 to infinite gain, the closed-loop poles move towards open-loop and infinite zeros. Hence, if the rho0 value is chosen for oscillatory, under-damped, fast-damped, critically damped, slow-damped or over-damped responses, the parameters, for examplea zero at $z=0.0$, are computed as follows:

$$1-\frac{\rho_1}{\rho_0}*T=0.0$$
$$\frac{\rho_1}{\rho_0}=\frac{1}{T}$$
$$\rho_1=10*\rho_0=10$$

$T=0.1$ seconds (chosen for a PI controller model) is two times faster than the sampling time for the plant. As an example, rho0 is set equal to 1.0, and the value of rho1 calculated depending on the zero location. The ratio of rho1 to rho0 determines how far apart the parameters are to each other as they become optimised by the IFT.

Table 2 shows the roots of the characteristic equation for the closed-loop system as per zero location on the z-plane. The root locus for the PI controller applied to the DC motor for different zero locations, shown in table 2, are illustrated in appendix III. According to the captured root locus results for the PI controller applied to the DC motor, the closed-loop poles for the system with zeros from $z=-0.2$ to $z=0.9$ are stable. However, for the system with zeros from $z=-0.9$ to $z=-0.4$, it has part of its loci in an unstable region. Hence, it is prudent to use the system with zeros from $z=-0.2$ to $z=0.9$ that have the loci inside the unit circle. Amongst the zero locations stipulated in table 2, location 0.4 is selected for investigation. It does not mean the choice 'zero location' is niche, but only chosen as an example. This zero is in the region of stability, as can be shown by a plot of root locus, in appendix III, figure AP.3. The rho-space is selected, from which results are obtained and mapped. Thereafter, the rho-space where the closed loop system is stable, is determined.

Table 2: Roots of the DC motor closed-loop control system

Zero location on the z-plane	Rho0	Rho1	Roots
2.00	0.10	0.20	0.98 0.91
0.90	1.00	1.00	0.90 + 0.01i 0.90 + 0.01i
0.40	1.00	6.00	0.90 + 0.22i 0.90 + 0.22i
-0.20	1.00	12.00	0.90 + 0.32i 0.90 + 0.32i
-0.60	1.00	16.00	0.90 + 0.38i 0.90 + 0.38i
-4.2	1.00	52.00	0.90 + 0.70i 0.90 + 0.70i

Having conducted root-locus analysis for the closed-loop system at different zero locations, the PI controller applied to the DC motor is verified for stability by use of a Routh-Hurwitz criterion. This procedure is accomplished by transforming the system's characteristic equation from equation (32), into the s-plane by means of bilinear transformation as follows: the characteristic equation is given as

$$z^2 + (0.09516 * \rho_0 - 1.904837) * z + (0.904837 + 0.09516 * \rho_1 * T - 0.09516 * \rho_0 = 0 \quad (33)$$

In the above given characteristic equation, z is substituted with $\frac{w+1}{w-1}$ to yield the w-plane equation.

$$\left(\frac{w+1}{w-1}\right)^2 + (0.09516 * \rho_0 - 1.904837) * \left(\frac{w+1}{w-1}\right) + (0.904837 + 0.09516 * \rho_1 * T - 0.09516 * \rho_0 = 0$$

$$0.095*\rho_1*T*w^2+(0.1904-0.19*\rho_1*T+0.19*\rho_0)*w+0.095*\rho_1*T-0.095*\rho_0+1.9048=0$$

Hence, the Routh-Hurwitz array is formed, as given in table 3.

Table 3: Routh-Hurwitz for the characteristic equation

w^2	$0.095*\rho_1*T \quad 0.095*\rho_1*T-0.095*\rho_0+1.9048$
w	$(0.1904-0.19*\rho_1*T+0.19*\rho_0) \qquad 0$
w^0	$0.095*\rho_1*T-0.095*\rho_0+1.9048$

To find a range of controller parameters where the closed-loop system is stable, we arrange rho1 in terms of rho0 as follows: for zero location at 0.8 and sampling time at 0.1 seconds, the relationship is computed as:

$$1-\frac{\rho_1}{\rho_0}*T=0.8$$
$$\frac{\rho_1}{\rho_0}=\frac{0.2}{T}$$
$$\rho_1=2*\rho_0$$

Hence, rho1 in table 3 is substituted with 2*rho0, in order to find the range of controller parameters for which the closed-loop system is stable. The range for w^2 is computed as follows:

$$0.095*T*2*\rho_0=0$$
$$\rho_0=0$$

Giving a range of $\rho_0>0$.

Similarly, the range for w is computed as follows:

$$(0.1904 - 0.19 * \rho_1 * T + 0.19 * \rho_0) = 0$$
$$\rho_0 = \frac{0.1904}{-0.1520} = -1.2526$$

Giving a range of $\rho_0 > -1.2526$.

Finally, the range for w^0 is zero, indicated by the relation:

$$0.095 * \rho_1 * T - 0.095 * \rho_0 + 1.9048 = 0$$
$$(0.095 - 0.019)\rho_0 = 1.9048$$
$$\rho_0 = 25.0632$$

Giving a range of $\rho_0 < 25.0632$.

Therefore, the overall range for closed-loop system stability for the PI controller applied to the DC motor (for rho0) is given as $-1.2526 < \rho_0 < 25.0632$. Similarly, the range for rho1 is $-2.5052 < \rho_1 < 50.1264$. Hence, parameter values outside the obtained range for rho0 and rho1 would render the closed-loop system unstable. This information can help to test the tuning action of the IFT technique, by selecting parameters outside the range and verifying if IFT is capable of forcing the parameters within the given range.

3.3.1 Simulation of the IFT applied to the DC motor

Using the rho-space mapped in table 2, we run the PI controller (without the IFT) driving the DC motor, and map the results of output responses in appendix IV. The PI controller (with the IFT) is implemented next, driving the DC motor once again and recording the results of the output responses in figure 3, figure 4 and figure 5. Thereafter, the results are compared to ascertain the validity of the IFT technique as an optimising technique. From this, we can safely select initial parameters which are slow-damped, fast-damped, and oscillatory closed-loop systems. The IFT is implemented 1 000 times, optimising one sample at a time while observing the trend of optimisation in 1000 spaces (N = 1000) or more, where a great amount of tuning information is required. A single IFT cycle composing of {experiment#1...}, {experiment#2...}, and {update parameters} phases

takes 40s and is synchronised with the step changes in the reference signal. The DC motor output signal y, reference signal r, controller parameters ρ_0 and ρ_1, error e and cost function J, are the focus of our investigations.

(I) Slow-damped case

The results for the initial slow-damped closed-loop system is illustrated in figure 3. As depicted, the controller is started with initial parameters of $rho0 = 0.1$ and $rho1 = 0.2$ which results in a slow or heavily damped closed-loop system. This is indicated in figure AP.IV.1 in appendix IV. The response of the DC motor and the reference signals are illustrated in figure 3a, followed by the results for the controller parameters and the cost function in figure 3b. Figure 3c illustrates the results for the controller parameters in a magnified format. Finally, figure 3d shows results for the error signal.

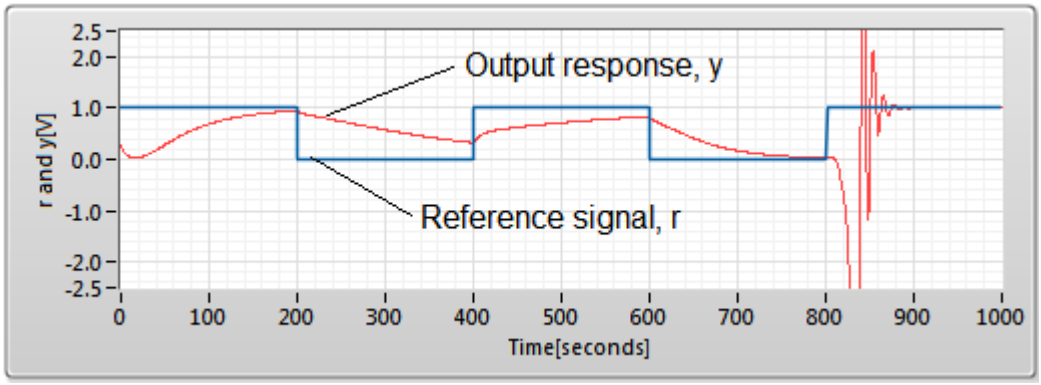

Figure 3(a): Reference signal and DC motor output signal

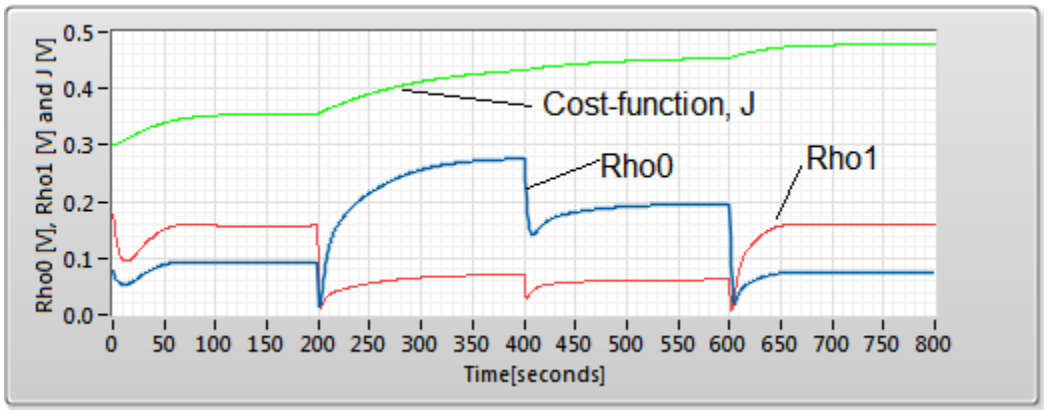

Figure 3(b): Rho0, Rh01 and cost function, J signals

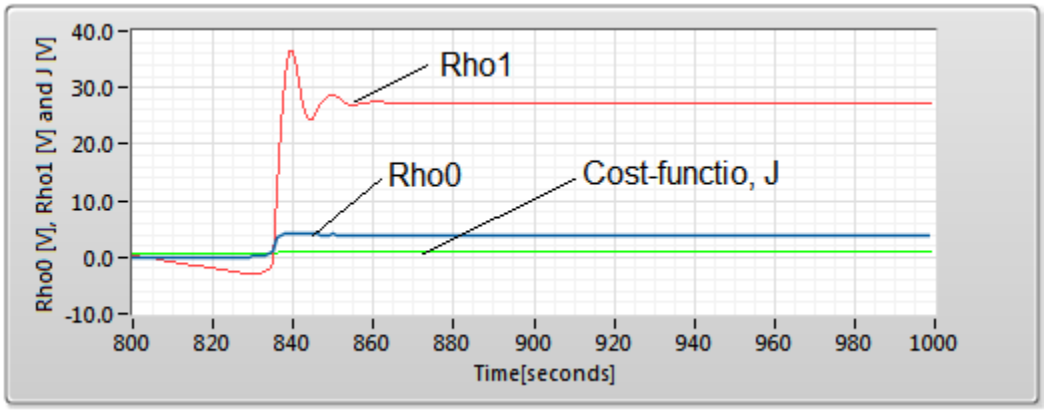

Figure 3(c): Rho0, Rh01 and cost function, J signals

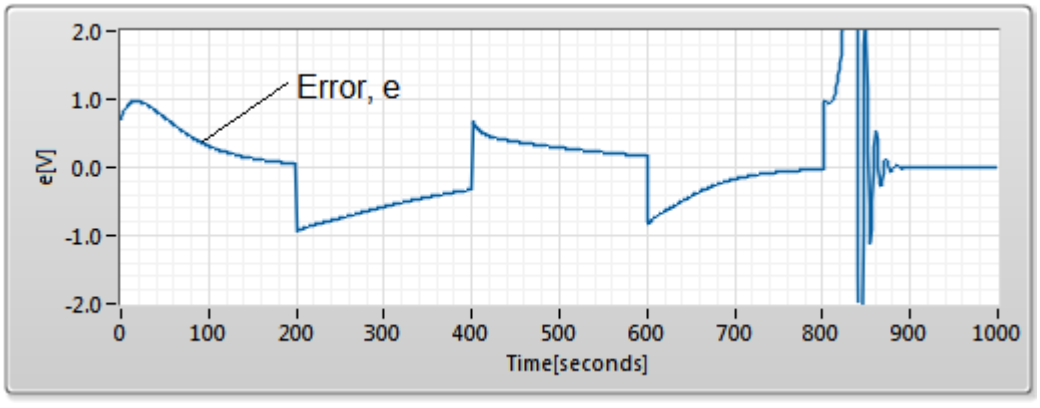

Figure 3(d): Error signal

The DC motor output response is heavily damped from sample number zero to sample number 800, and thereafter it becomes unstable up to sample number 860 where controller parameters converge to rho0 = 4.2 and rho1 = 28.0, as illustrated in figure 3c. This behaviour is readily explained by roots of the characteristic equation for the closed-loop system for the given sample numbers in table 4. At sample number 15 the roots indicate slow stability, which later turns out to be an integrator, and finally converges to an oscillatory stable response. The sample numbers selected are mapped in table 4 with their respective parameter values and roots. This optimisation has multi-minima at samples number 15, 200, 405, and 605, as shown in figure 3b. Optimisation occurs up to sample 860 where the cost function converges to 0.4. The cost function builds up from sample number 1 to sample number 840. This is due to the presence of a huge error in the region, as shown in figure 3d. Clearly, the simulation for the initial slow-damped closed-loop system has validated the optimising capability of the IFT in optimising controller parameters, in that the heavily-damped DC motor response turns into an oscillatory stable response (see figure 3a).

Table 4: Roots of the DC motor closed-loop control system

Sample No.	Rho0	Rho1	Roots
15	0.06	0.10	0.99 0.90
200	0.02	0.02	1.00 0.90
405	0.14	0.03	1.00 0.89
605	0.02	0.01	1.00 0.90
840	4.2	36.00	0.75 + 0.53i 0.75 - 0.53i

900	4.2	28.00	0.75 + 0.45i 0.75 - 0.45i

(II) Fast-damped case

The results for the initial fast-damped closed-loop system are illustrated in figure 4.

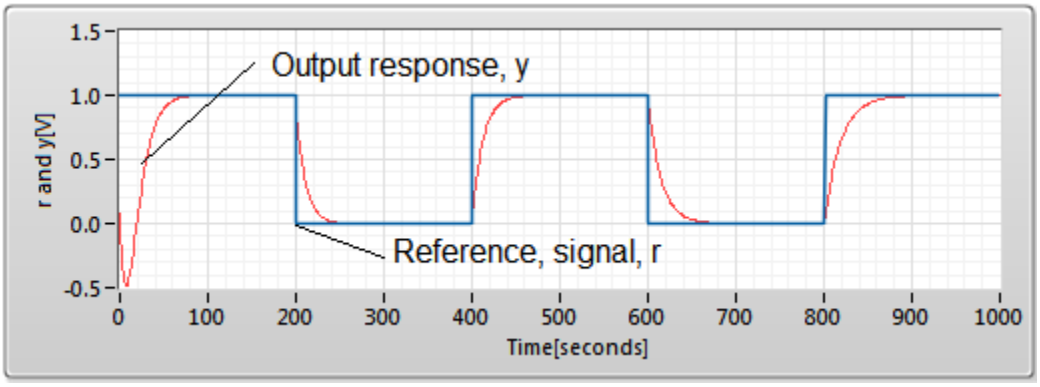

Figure 4(a): Reference and DC motor output signal

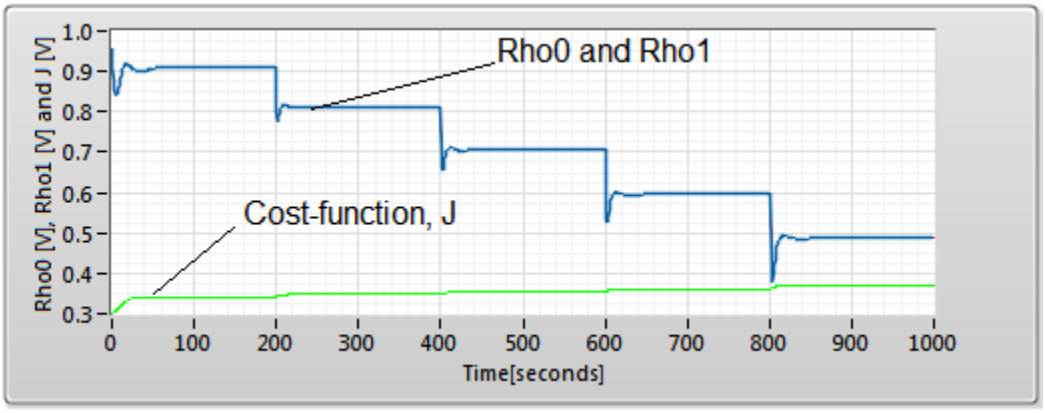

Figure 4(b): PI controller parameter and cost function signals

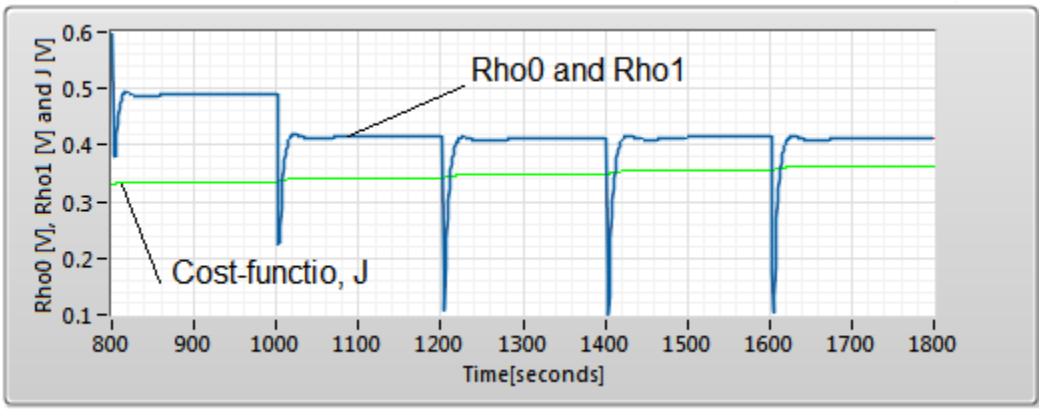

Figure 4(c): Magnified PI controller parameter signals

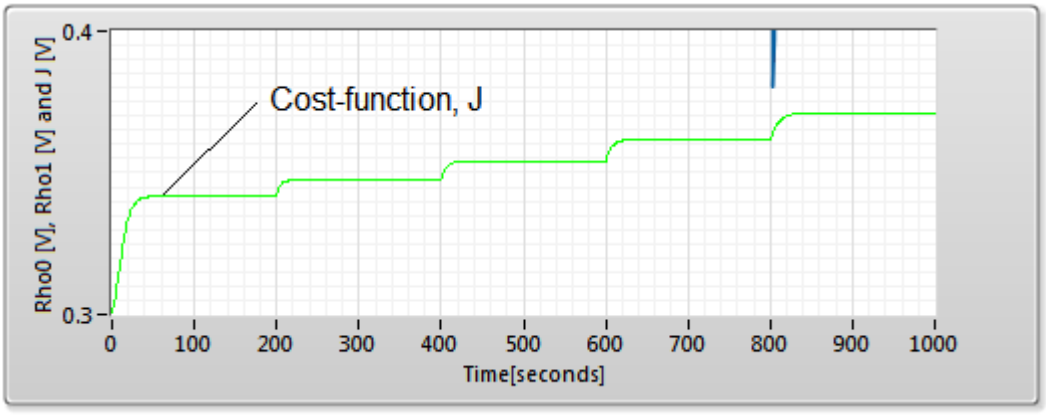

Figure 4(d): Magnified cost function signal

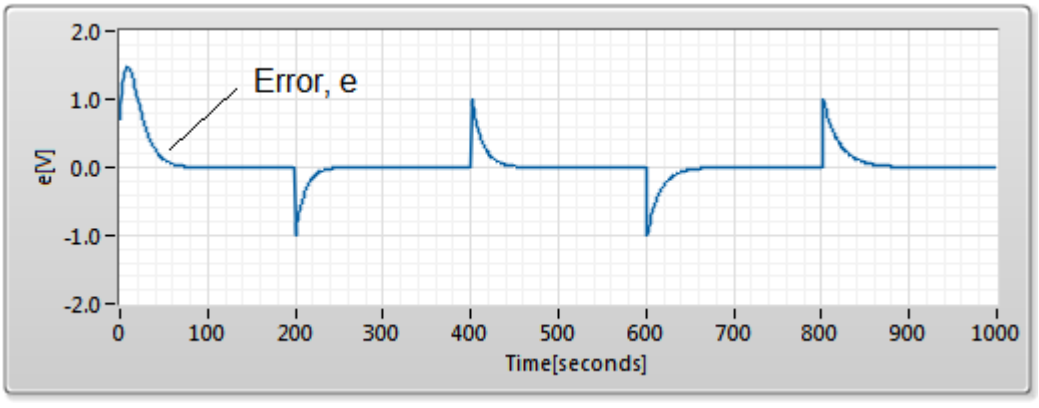

Figure 4(e): Error signal

The IFT is implemented with initial parameters of $rho0 = 1.0$ and $rho1 = 1.0$, which represents a fast-damped or critical-damped closed-loop system, as indicated in figure AP.IV.2 of appendix IV. The response of the DC motor and the reference signals are illustrated in figure 4a, followed by the results for controller parameters and the cost function segmented into figures 4b, 4c, and 4d, in order to magnify significant sections of the graphs. The results for the error signal is shown in figure 4e. The response demonstrates similarity to the response of the PI controller (without the IFT) applied to the DC motor, as demonstrated from sample number 200 to sample number 1000 (see figure 4a and figure AP.IV.2).

From sample number zero to sample number 200, the response is in its transient state, as shown by the roots of the characteristic equation in table 5. The details are obtained by studying figures 4b, 4c, and 4d, which displays a variation of the controller parameters as a result of the controller tuning.

The tuning in this case is of varying multi-minims from sample number zero to sample number 1000, and then converges to a single-minima at sample 1200, occurring at every 200 samples, with parameters converged to rho0 = 0.4 and rho1 = 0.4. The cause of these minimum spikes is as a result of increase in the cost function after every 200 samples (see figure 4d: The increments of the cost function). The minimums are tabulated in table 5.

The cost function builds up after every 200 samples, from sample number zero to sample number 1000, due to the remnant error that is not minimised (see figure 4e).

Table 5: Roots of the DC motor closed-loop control system

Sample No.	Rho0	Rho1	Roots
5	0.85	0.85	0.91 + 0.01i 0.91 - 0.01i
200	0.78	0.78	0.93 0.90

400	0.65	0.65	0.94 0.90
600	0.53	0.53	0.95 0.90
800	0.38	0.38	0.96 0.90
1000	0.23	0.23	0.98 0.90
1200	0.13	0.13	0.99 0.90
1400	0.13	0.13	0.99 0.90
1600	0.13	0.13	0.99 0.90
1800	0.13	0.13	0.99 0.90

(III) Oscillatory case

The results for the initial oscillatory controller parameters are illustrated in figure 5.

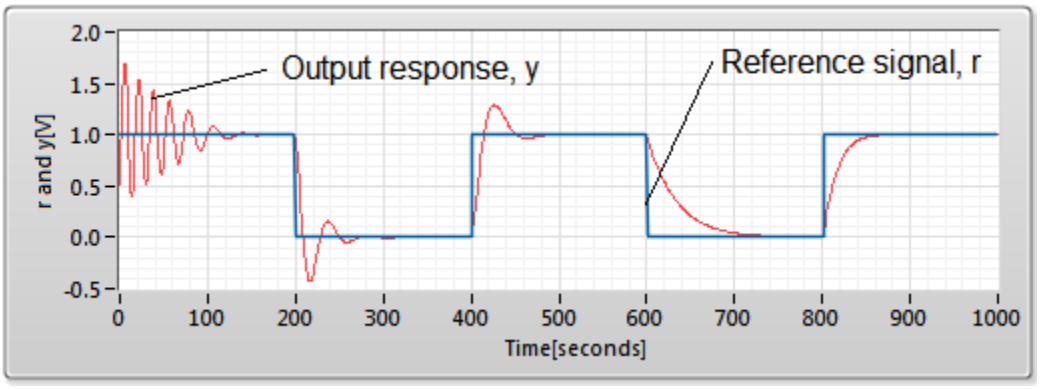

Figure 5(a): Reference signal and DC motor output signal

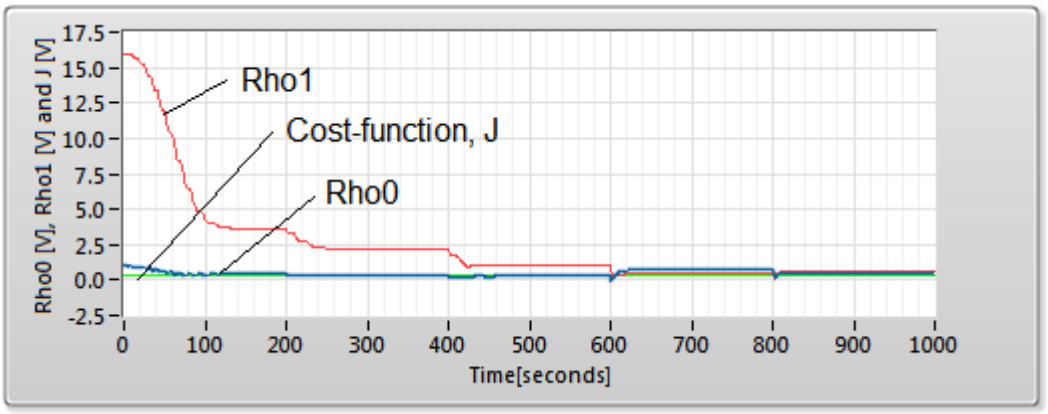

Figure 5(b): PI controller parameter signals and cost function signal

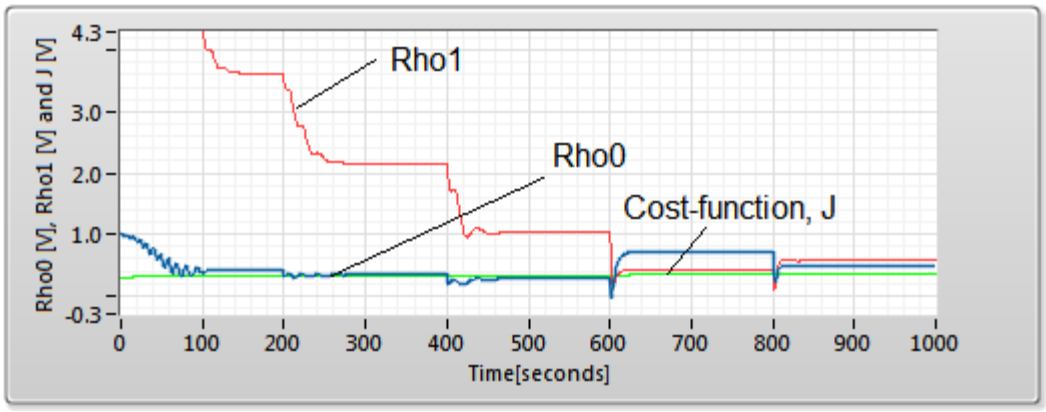

Figure 5(c): PI controller parameter signals and cost function signal

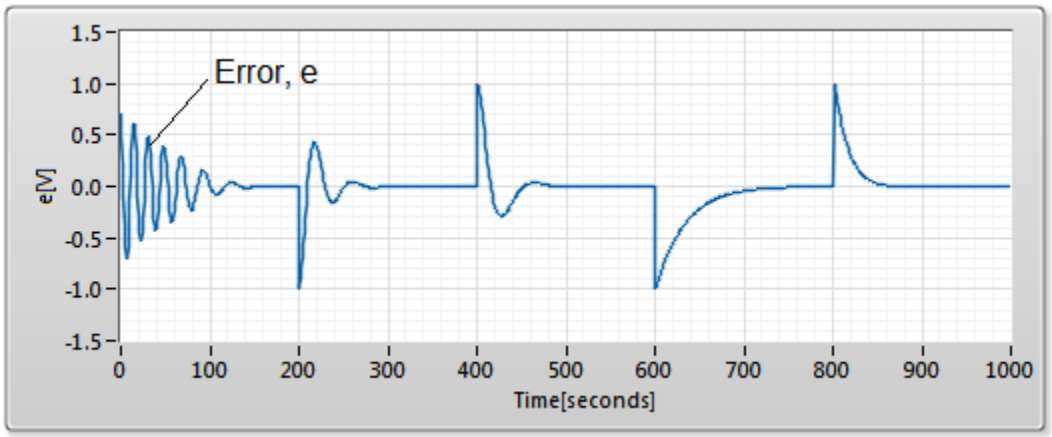

Figure 5(d): Error signal

For the oscillatory case, the controller is implemented utilising initial parameters of $rho0 = 1.0$ and $rho1 = 16.0$. These are oscillatory controller parameters, as indicated in figure AP.IV.2 of appendix IV. The response of the DC motor and the reference signal are illustrated in figure 5a, followed by the results for the controller parameters and the cost function, segmented into figures 5b and 5c, in order to magnify the minima that occurs ahead of convergence. The result for the error signal is in figure 5d. The DC motor output signal and the reference signal are indicated in figure 5a, and the controller parameters, rho0, rho1 and the cost function illustrated in figure 5b and 5c. Finally, figure 5d illustrates the error signal. The response demonstrates a great amount of tuning information, from the oscillatory to the fast-damped closed-loop system. Selected sample numbers with their controller parameters and roots of the characteristic equation, are tabulated in table 6.

Sample numbers from zero to 700 are oscillatory parameters, demonstrated by the roots of the characteristic equation in table 6.

After sample number 700 to sample number 1000, the controller parameters yield real roots inside the unit circle, indicating stability. This again has validated the optimising action of IFT technique controlling the DC motor.

Table 6: Roots of the oscillatory closed-loop control system

Sample No.	Rho0	Rho1	Roots
0	1.00	16.00	0.93 + 0.27i 0.93 - 0.27i
200	0.12	3.18	0.94 + 0.27i 0.94 - 0.27i
400	0.12	2.06	0.94 + 0.13i 0.94 - 0.13i
600	0.0	0.12	0.99 0.91

800	0.06	0.06	0.99 0.90
1000	0.12	0.13	0.99 0.90

3.3.2 The effect of varying the reference signal amplitude on IFT optimisation

Thus far, the test has only used a reference signal amplitude of one volt; hence it is not yet known what the effect on the system will be if the amplitude of the reference signal were to be varied. Here, table 7, table 8 and table 9 are presented to illustrate the effect of varying the amplitude of the reference signal for the case of the initially slow-damped, initially fast-damped, and oscillatory controller parameters, respectively.

The data for all the three cases shows variation in the length of the settling time for the DC motor response, variation in convergence of parameters, and variation in the gamma constant that is initially chosen manually, but as a result of varying the amplitude of the reference signal, needs to be varied as well. However, the degree of variation depends on the specific case (slow damped, fast damped, or oscillatory).

Table 7: Effect of varying amplitude reference signal – a slow-damped case

No.	Amplitude [V]	Settling time [samples]	Overshoot [V]	Rho0	Rho1	Gamma
1	0.10	20.00	0.00	4.00	1.90	2.20
2	0.20	25.00	0.00	1.80	2.40	1.10
3	0.30	40.00	0.00	1.10	1.20	1.20
4	0.40	10.00	0.08	20.00	105.00	1.20

5	0.50	50.00	0.20	10.00	5.50	1.20
6	0.60	0.00	0.00	21.50	15.00	1.20
7	0.70	100.00	0.00	1.50	0.75	1.20
8	0.80	25.00	0.00	9.00	6.50	1.20
9	0.90	10.00	0.00	4.01	3.50	1.10
10	1.00	5.00	0.30	10.00	9.00	1.10
11	1.10	2.00	0.80	18.00	17.00	1.10
12	1.20	1.00	0.00	19.00	18.00	1.10

Table 8: Effect of varying amplitude reference signal – a fast-damped case

No.	Amplitude [V]	Settling time [samples]	Overshoot [V]	Rho0	Rho1	Gamma
1	0.10	20.00	0.00	4.30	4.300	3.20
2	0.20	50.00	0.00	1.44	1.440	2.20
3	0.30	65.00	0.00	0.83	0.825	2.20
4	0.40	5.00	0.00	12.50	12.500	2.20
5	0.50	10.00	0.30	33.00	33.000	2.20
6	0.60	70.00	0.00	0.80	0.800	2.20
7	0.70	105.00	0.00	0.65	0.65	1.20
8	0.80	100.00	0.00	0.80	0.80	1.20

9	0.90	4.00	0.10	24.00	24.00	1.20
10	1.00	75.00	0.00	1.00	1.00	1.20
11	1.10	70.00	0.00	32.00	32.00	1.20
12	1.20	15.00	0.00	5.05	5.05	1.10

Table 9: Effect of varying amplitude reference signal – an oscillatory case

No.	Amplitude [V]	Settling time [samples]	Overshoot [V]	Rho0	Rho1	Gamma
1	0.10	15.00	0.00	2.40	2.50	90.20
2	0.20	25.00	0.00	2.55	2.40	80.20
3	0.30	55.00	0.00	15.13	7.75	80.20
4	0.40	25.00	0.01	10.20	10.20	60.20
5	0.50	25.00	0.05	20.25	20.25	60.20
6	0.60	0.01	0.00	10.40	5.10	60.20
7	0.70	5.00	0.20	20.00	85.00	60.20
8	0.80	5.00	5.00	16.00	2.00	60.20
9	0.90	15.00	0.30	15.00	55.00	20.20
10	1.00	15.00	0.01	7.00	5.50	20.20
11	1.10	15.00	0.10	11.00	13.00	1.20

12	1.20	10.00	1.25	40.00	200.00	1.20

That the degree of variation depends on the specific case is clearly shown by the starting gamma constant in each instance. For example, the starting gamma constant for a slow-damped system is 2.2, while for a fast-damped it is 3.2, and for the oscillatory system 90.2. This is facilitated by the size of error. If error is big, the cost function will rise very fast, implying a high gradient for the cost function, which in turn will require a small gamma constant. However, if the error is small, there is a need for a big gamma constant in order to affect the optimisation process. This implies that for varying reference signal amplitude, the gamma constant should be set by a line search to avoid the adaptive control blow-up of the 1980s [40].

In the slow-damped case, the increase of the reference signal amplitude has an influence on the sensitivity of the gamma constant. For instance, from 0.1V to 0.3V, the gamma constant changed three times, while from 0.3V to 0.8V, there is no change, and again, from 0.9V to 1.2V, it remains constant.

In the case of the fast-damped initial system, the gamma constant is the same (at 2.2) from 0.1V to 0.6V, and changes to 1.2 from 0.7V to 1.1V.

In an oscillatory case, there are five regions of operation with different gamma constants. For instance, we can see from table 8 that at 0.1V, the gamma constant is 90.2, at 0.2V and 0.3V it is 80.2, from 0.4V to 0.8V it is at 60.2, at 0.9V and 1.0V it is at 20.2, and finally, at 1.1V and 1.2V it is at 1.2. Clearly, the gamma constant decreased from 90.2 to 1.2 as a result of increasing amplitude of the reference signal from 0.1V to 1.2V, implying that increasing the amplitude of the reference signal causes an increase in the error signal and the adaptive control blow-up of the 1980s. We can decide to work with a gamma constant of 1.2 throughout to avoid the adaptive control blow-up, however, the only drawback is, tuning would take a long time for lower amplitudes of the reference signal. Higher reference signal amplitudes cause controller parameters to converge outside the stability bound ($-4.95054 < \rho_1 < 88.08494$), implying that such amplitudes can cause

instability in the system. For example, the reference signal of amplitude 1.2V makes parameters converge to rho0 = 40 and rho1 = 200 outside the stability range. The problem of system blow-up also exists, especially when gamma is made constant while the reference signal amplitude is varied upwards, and this, in addition, makes the parameters converge to infinity. This phenomenon is traced to controller parameter update variable overflow when the gradient of the cost function tends to be infinite.

3.3.3 The effect of the single-step reference signal on IFT optimisation

As the case in the previous section, we can show that the tuning scheme of the IFT technique by using a reference signal of a single step. We utilise the initial fast-damped closed-loop system for this demonstration only as an example, not that there is anything unique about it. Figure 6 indicate results for running the IFT using a reference signal of a single step.

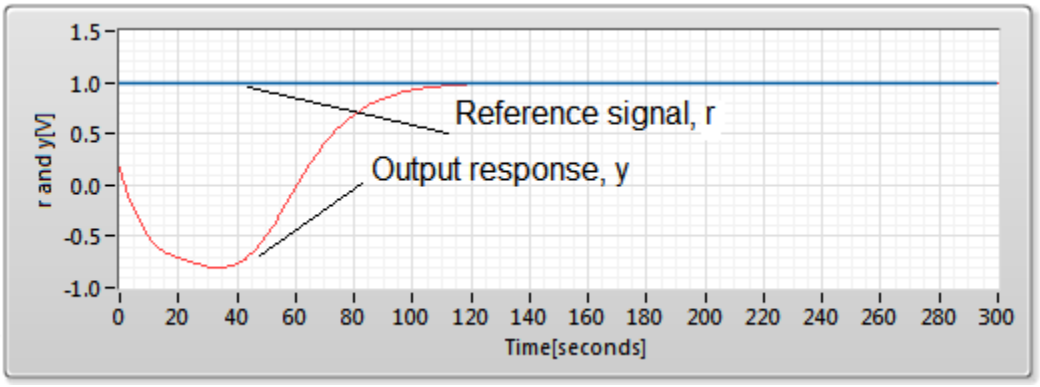

Figure 6(a): Reference and DC motor output signals

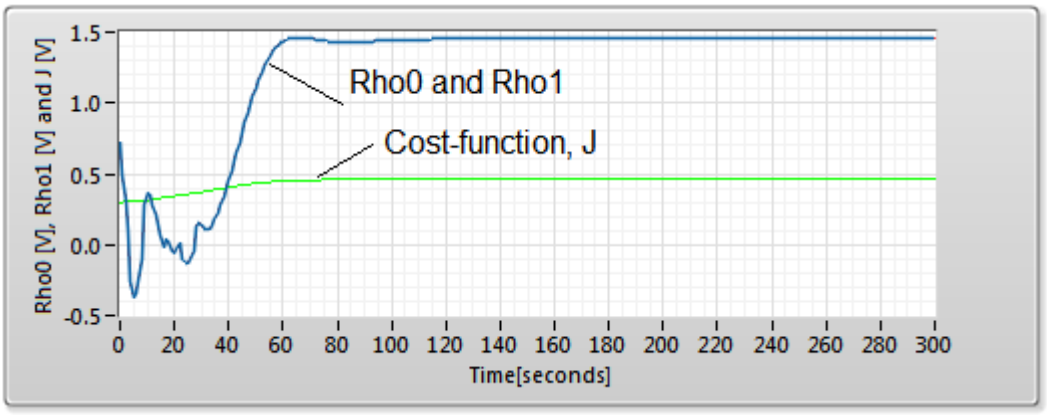

Figure 6(b): PI controller parameter and cost function signals

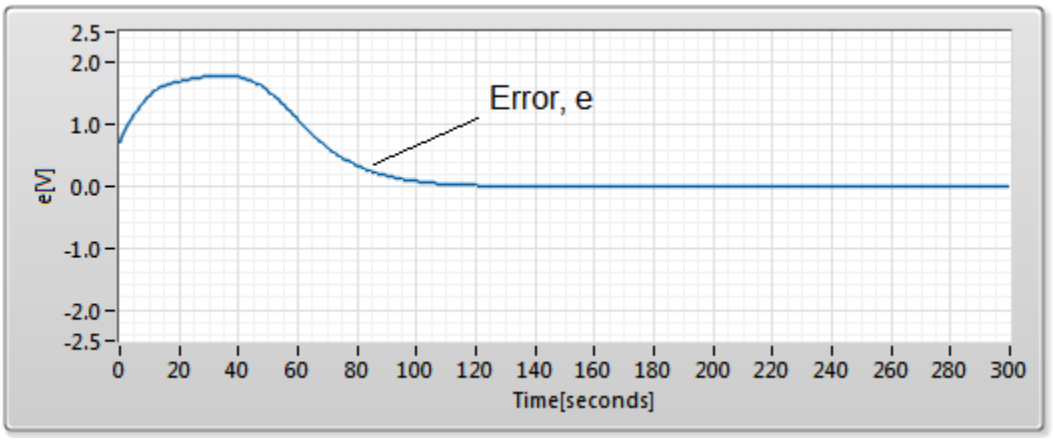

Figure 6(c): Error signal

The DC motor output and the reference signals are indicated in figure 6a. The controller parameters, rho0 and rho1 and the cost function are illustrated in figure 6b. Finally, figure 6c illustrates the error signal. The variation of parameters from sample number zero to sample number 120 is an indication that there was tuning that occurred before convergence. The response also demonstrates convergence to a fast-damped closed-loop system.

In this test, the parameters for both rho0 and rho1 converge to 1.45 (see figure 6b), which is different to the convergence in the test number 10 of table 7, where the series of step reference signals were utilised yielding rho0 and rho1 converging to 1. The result for the

single-step reference signal implies that the presence of the step signal triggers optimisation only for the period of the step signal, and afterwards the system runs in the converged mode for the rest of its operation as long as it is not disturbed. This means that the step signal can also be modelled as a source of disturbance in the closed-loop system, to ensure the adaption gets the information it needs. However, the IFT will only handle a certain percentage of noise before it fails to optimise, as indicated in the test conducted in section 3.3.2. For example, if we took a gamma constant of 1.2, the maximum noise amplitude the IFT is capable of handling is 1.1V, and anything above that, the IFT will not be able to handle unless the gamma constant is reduced, as shown in table 7. Hence, lower gamma constants can enable the IFT to handle higher noise levels though optimisation, but becomes slow for such gamma constants.

3.3.4 Simulation of the IFT using the oscillatory plant

A similar study as before is carried out using the oscillatory plant in equation (30). The plant represented in the Laplace domain is sampled into a digital equation as follows:

A standard transfer function is used to approximate the sampling frequency, thus,

$g(s) = \frac{{\omega_n}^2}{s^2 + 2*\zeta*\omega_n*s + {\omega_n}^2}$, which can be equated to the plant in equation (30) so that we approximate the sampling frequency f_s. Therefore, the natural frequency f_n is computed as follows:

$$\omega_n = \sqrt{10} = 3.1623 = 2*\pi*f_n$$
$$f_n = \frac{3.1623}{2*\pi} = 0.5032$$

The damping ratio ζ for the oscillatory plant, on the other hand, is computed as follows:

$\zeta = \frac{3}{2*\omega_n} = \frac{3}{2*\sqrt{10}} = 0.4743$ and the peak time T_p is

$$T_p = \frac{\pi}{\omega_n*\sqrt{1-\zeta^2}} = \frac{\pi}{\omega} = \frac{\pi}{\sqrt{10}*\sqrt{1-0.4743^2}} = 1.1286s$$

No. of oscillations = $\frac{2*\omega_n\sqrt{1-\zeta^2}}{\pi*\zeta*\omega_n} = \frac{2}{\pi}*\sqrt{(\frac{1}{\zeta}-1)} = 1.667s$

settling time is $t_s = \frac{4}{\zeta*\omega_n} = \frac{4}{0.4743*3.1623} = 2.6669s$, hence the periodic time T is

$T = \frac{2.6669}{1.667} = 1.5998s$ The frequency of oscillation is

$f = \frac{1}{1.5998} = 0.6251$ Hz

and the sampling time T_s is $T_s = \frac{1}{10*f} = \frac{1}{10*0.6251} = 0.1600 \approx 0.2s$

Utilising the computed sampling time, the Matlab code given in appendix V is used to compute equation (34) to generate a transfer function (in the z-domain with the holding circuit).

$$gh = \frac{0.1601*z + 0.1308}{z^2 - 1.252*z + 0.5488} \text{ [V]} \qquad (34)$$

Equation (34) is then converted to a digital equation that is given in equation (35).

$$gh = \frac{0.1601*z^{-1} + 0.1308*z^{-2}}{1 - 1.252*z^{-1} + 0.5488*z^{-2}} = \frac{y}{u}$$

$$y_t = 1.252*y_{t-1} - 0.5488*y_{t-2} + 0.1601*u_{t-1} + 0.1308*u_{t-2} \text{ [V]} \qquad (35)$$

We utilise the PI controller again to control the oscillatory plant sampled above. The open-loop PI controller applied to the oscillatory plant is given as

$k(z)*gh(z) = \frac{\rho_0*z + (\rho_1*T - \rho_0)}{z-1} * \frac{0.1601*z + 0.1308}{z^2 - 1.252*z + 0.5488}$, yielding open-loop poles at $z = 1$, $z = 0.6260 + 0.3961\text{i}$ and $z = 0.6260 - 0.3961\text{i}$ with two zeros $z = 0.8170$ and $z = 1 - \frac{\rho_1}{\rho_0}T$ dependent on the controller parameters that are varying due to the controller tuning. Similarly, as for the case of the DC motor, the sampling time for the controller was made faster than the sampling time for the plant. The stability region for the closed-loop system is determined in order to expose the terrain to operate the IFT. To this end, the closed-loop system transfer function is derived for the PI controller applied to the oscillatory plant. The derivation is given as:

$$= \frac{0.16*\rho_0*z^2+(0.16*\rho_1*T-0.03*\rho_0)*z+0.13*(\rho_1*T-\rho_0)}{z^3-2.09*z^2+(1.80+(0.16*\rho_1*T-0.03*\rho_0)*z+0.13*(\rho_1*T-\rho_0)-0.55} \text{ [V/V] (36)}$$

Equation (36) is a closed-loop PI controller applied to the oscillatory plant. The same initial controller parameters utilised for the DC motor is used, to check the universality of the IFT in tuning controllers when applied to different plants. This is important because the IFT technique assumes that the plant is not known, or only partially known.

(I) Slow-damped case

The IFT with initial parameters $rho0 = 0.1$ and $rho1 = 0.2$ is implemented. This was also used in section 3.3.1 for a slow-damped case of the DC motor, in order to compare the results of the two plants. The response of the oscillatory plant and the reference signals are illustrated in figure 7a, followed by the results for the controller parameters as shown in figure 7b. The results for the error signal is shown in figure 7c.

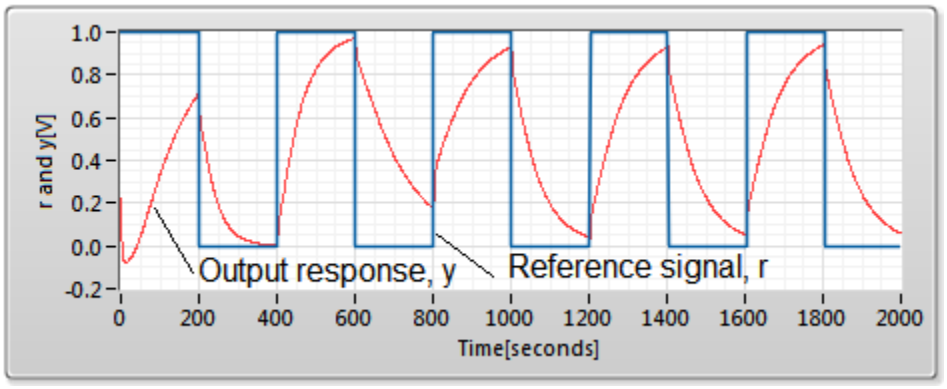

Figure 7(a): IFT response for oscillatory plant

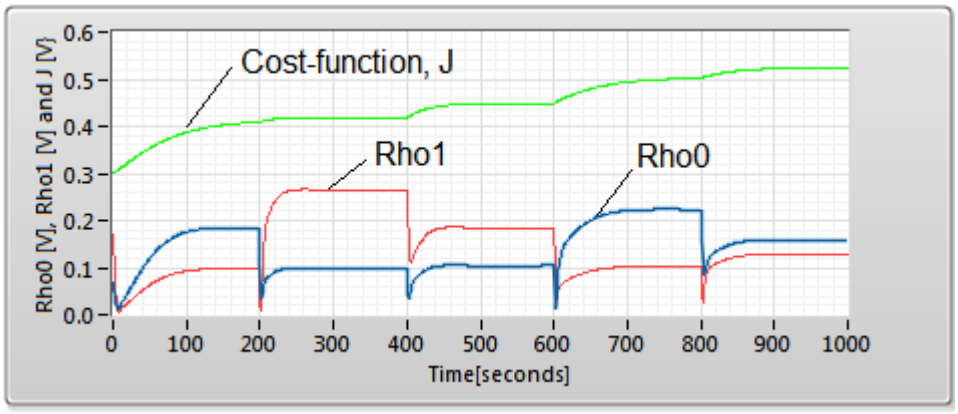

Figure 7(b): Parameter trajectory and cost function

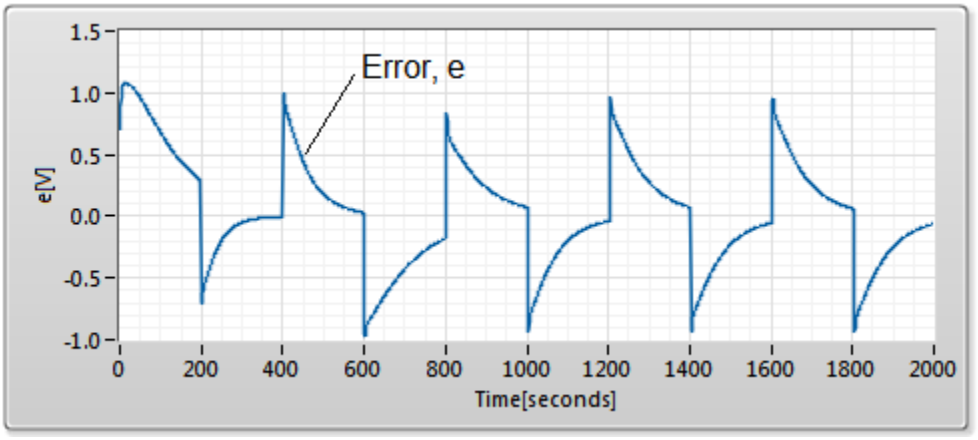

Figure 7(c): Error signal

The oscillatory plant output and the reference signals are indicated in figure 7a. The controller parameters, rho0 and rho1 and the cost function in figure 7b. Finally, figure 7c illustrates the error signal. From the closed-loop transfer function, the roots of the characteristic equation at samples 0, 6, 200, 400, 600, 800, 1000 and 2000 are computed, and tabulate them in table 10. Parameter values are extracted from figure 7b from the tabulated sample numbers.

The time constant τ of the unstable oscillatory pole at sample number zero of table 10 is checked:

$$\tau = \frac{f_s}{1-|R|} = \frac{0.2}{0.1647} = 1.2143s$$

The pole will decay in $4*\tau = 4.8572s$. Where f_s is a sampling frequency and $R \in (0,1)$, is the radius of the pole in the unit circle.

The real pole's 'time constant' at sample number zero is given as

$$\tau = \frac{f_s}{1-|R|} = \frac{0.2}{0.237} = 0.2621s$$

The pole decays in $4*\tau = 1.0484s$, much faster than the oscillatory pole decay. From table 10 can be observed that the gain of the controller ranges between 0.01 and 0.2 (throughout the sample space), making the already heavily damped closed-loop system slower, hence the slow response demonstrated in figure 7a.

Table 10: Roots of the oscillatory plant closed-loop control system

Sample no.	Rho0	Rho1	Roots
0	0.10	0.20	-1.16 + 1.00i -1.1647 - 1.00i 0.24
6	0.01	0.01	-1.16 + 1.00i -1.16 - 1.00i 0.23
200	0.02	0.01	-1.16 + 1.00i -1.16 - 1.00i 0.23
400	0.04	0.10	-1.16 + 1.00i -1.16 - 1.00i 0.24
600	0.01	0.02	-1.16 + 1.00i -1.16 - 1.00i 0.23
800	0.10	0.04	-1.17 + 1.00i -1.17 - 1.00i 0.24
1000	0.15	0.12	-1.17 + 1.00i -1.17 - 1.00i 0.24
2000	0.16	0.16	-1.17 + 1.00i -1.17 - 1.00i 0.24

From the results obtained and illustrated in figure 7a, it can be seen that the response does not converge, due to the heavily damped closed-loop system caused by the low

gain controller. Hence, the settling time is almost infinite and causes the cost function to increase continuously. To resolve the this problem, the reference signal amplitude should be raised to shorten the settling time, as it affects in the increase of the gain of the controller, hence improving the optimisation, as shown in figure 7d and figure 7e. In the same vein, step length should be increased to allow the output signal to settle before the reference signal dies away or changes polarity.

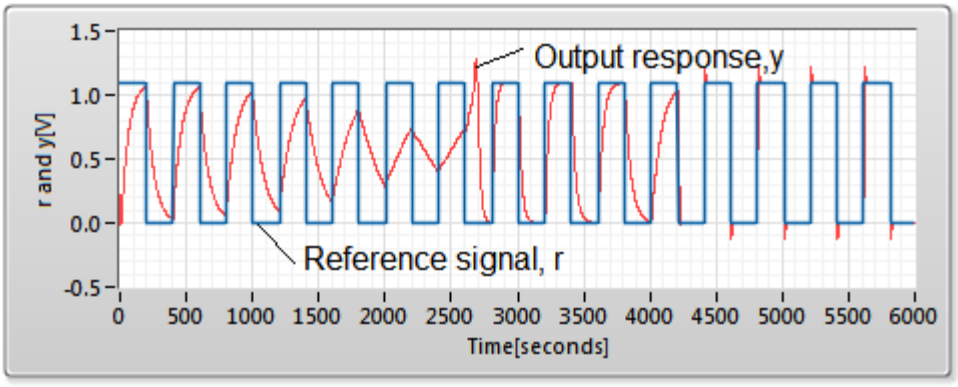

Figure 7(d): IFT response for oscillatory plant

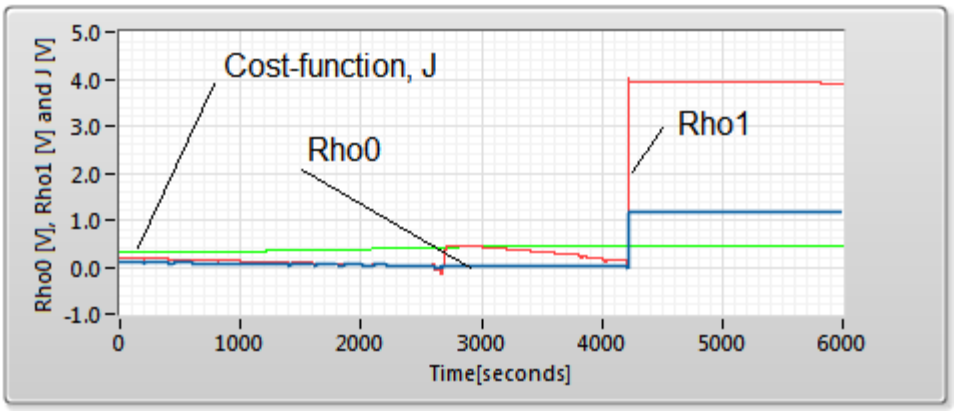

Figure 7(e): Parameter trajectory and cost function

Figure 7d and figure 7e demonstrate some degree of optimisation, though it takes a long time before the parameters converge (in this case convergence occurs after 4200 samples). The heavily damped controller requires higher reference signal amplitudes to converge.

(II) Fast-damped case

For the fast-damped case, the initial parameters are set at $rho0 = 1$ and $rho1 = 1$, similar to section 3.3.1 for a fast-damped case of the DC motor. This was carried out in order to make comparisons to the results of the oscillatory plant and those of the DC motor. The response of the oscillatory plant and the reference signals are illustrated in figure 8a, followed by the results for the controller parameters, as shown in figure 8b. The results for the error signal is shown in figure 8e.

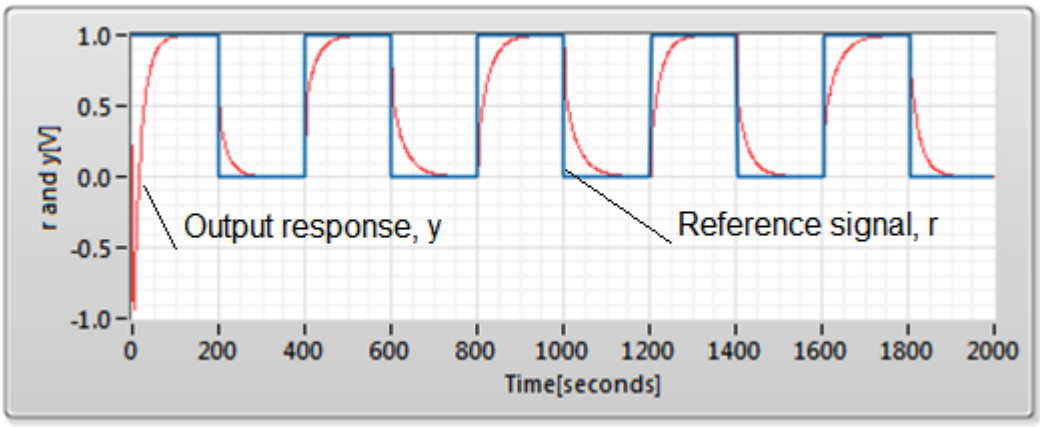

Figure 8(a): IFT response for oscillatory plant

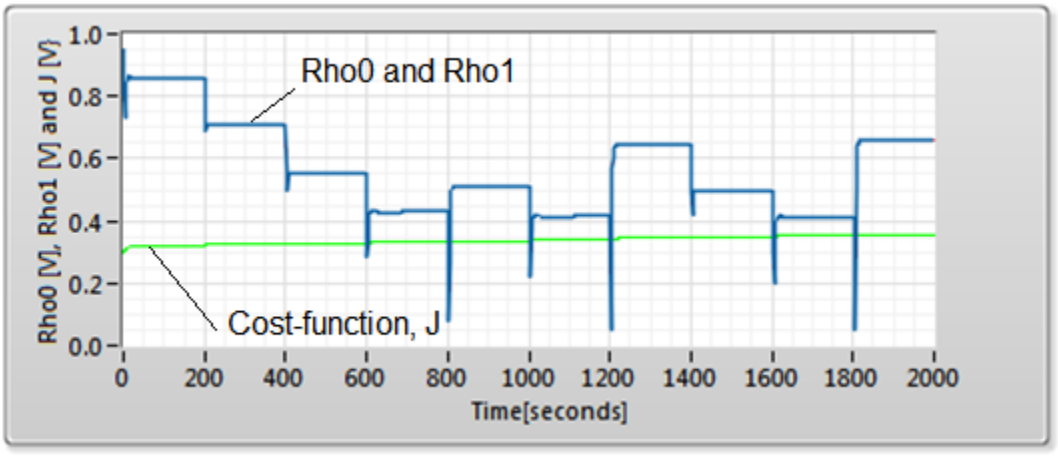

Figure 8(b): Parameter trajectory and cost function

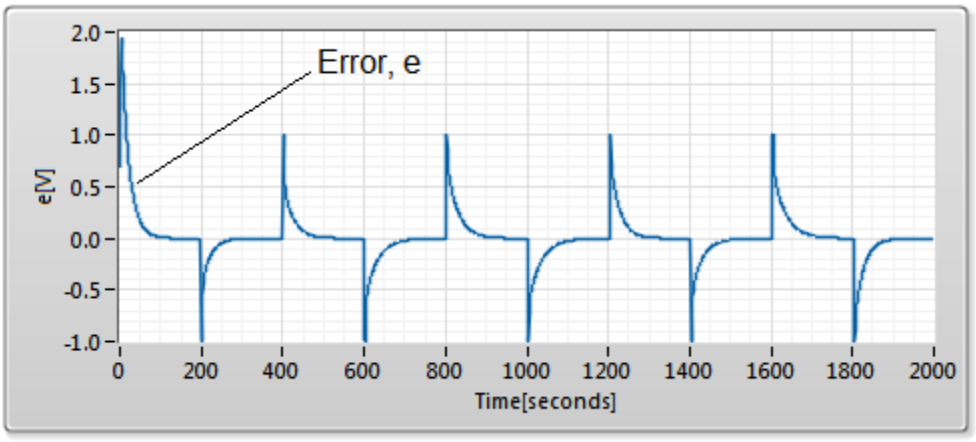

Figure 8(c): Error signal.

Figure 8a demonstrates a stable response throughout the entire sample-space, however, when analysing it in terms of the parameter trajectory, this IFT optimisation has multi-minimums. In comparison to results of the DC motor, we see some similarity in output responses, especially in the settling time. The roots of the system characteristic equation for a given controller parameters are computed at selected sample numbers, as given in table 11. This is to analyse the tuning of controller parameters as a result of the IFT optimisation.

One intriguing observation is the existence of two complex poles outside the unit circle while the response indicates some level of stability. An obvious reason could be owing to the region of convergence (ROC) which is bounded by the complex pole, and the pole inside the unit circle. To elaborate the argument further, sample number 1200 is selected as an example: the range in this case is $-1.1640 < z < 0.2737$, and since the region cuts the unit circle, the system becomes stable in this region.

Table 11: Roots of the fast-damped plant for closed loop control system

Sample No.	Rho0	Rho1	Roots
0	1.00	1.00	-1.18 + 1.018i -1.18 - 1.018i 0.27
5	0.75	0.75	-1.18 + 1.013i

			-1.18 - 1.013i 0.26
600	0.30	0.30	-1.17 + 1.00i -1.17 - 1.00i 0.25
800	0.10	0.10	-1.17 + 1.00i -1.17 - 1.00i 0.24
1000	0.20	0.20	-1.17 + 1.00i -1.17 - 1.00i 0.24
1200	0.05	0.05	-1.16 + 1.00i -1.16 - 1.00i 0.24
1400	0.40	0.40	-1.17 + 1.01i -1.17 - 1.01i 0.25
2000	0.65	0.65	-1.18 + 1.01i -1.18 - 1.01i 0.26

(III) Oscillatory plant Case

For the oscillatory plant case, initial parameters are set at $rho0 = 1$ and $rho1 = 10$. This is different to the parameters applied in section 3.3.1 for the oscillatory case of the DC motor, since the former gave a blow-up error. The response of the oscillatory plant and the reference signals are illustrated in figure 9a, followed by the results for the controller parameters in figure 9b. The results for the error signal is shown in figure 9c.

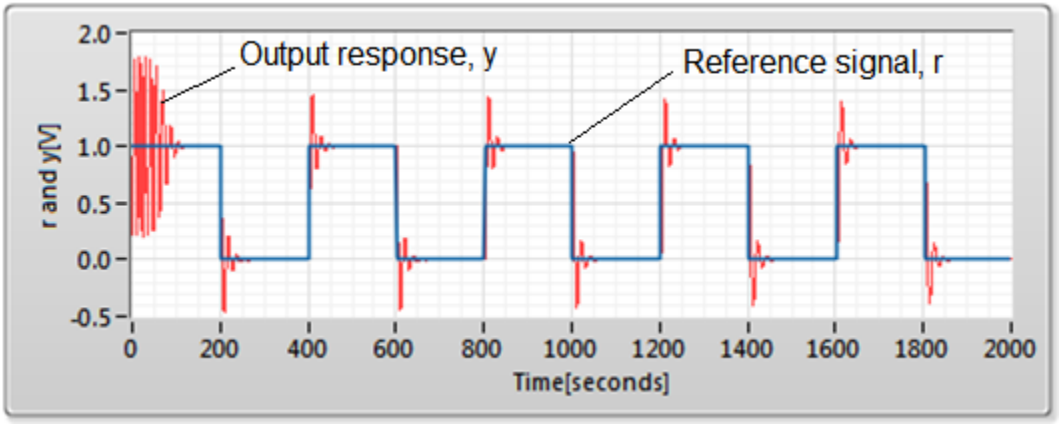

Figure 9(a): IFT response for oscillatory plant

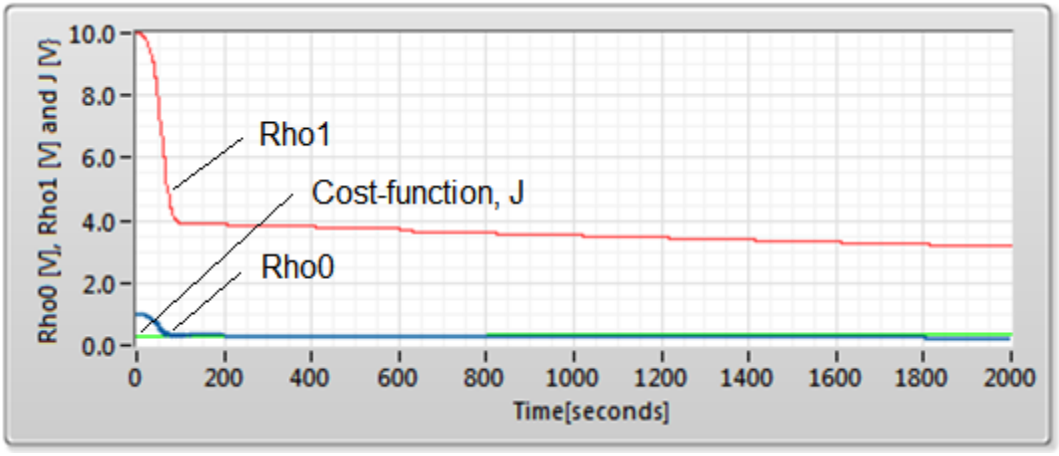

Figure 9(b): Parameter trajectory and cost function

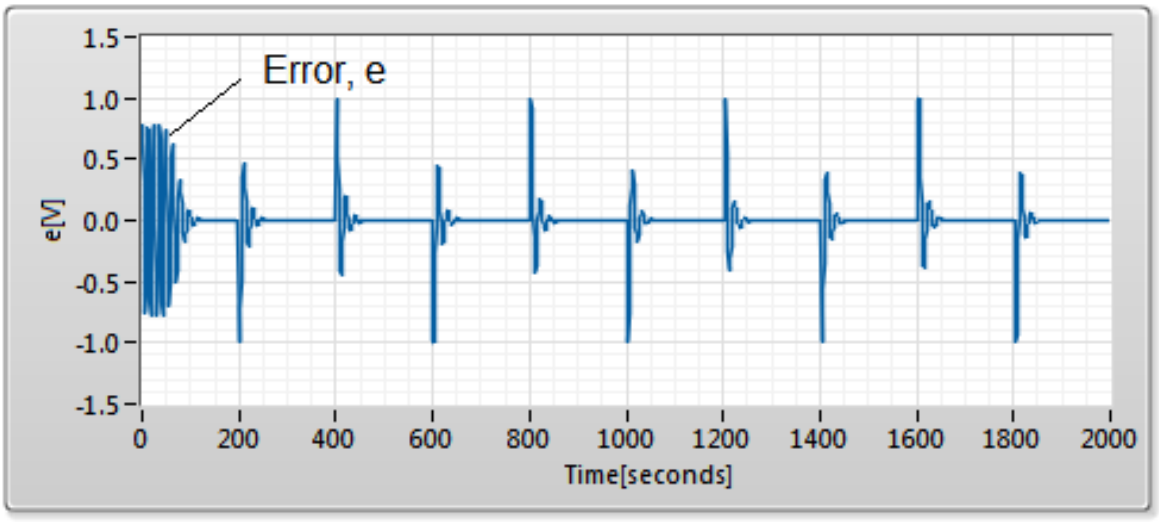

Figure 9(c): Error signal

The IFT is implemented 2000 times, as compared to 1000 times for the oscillatory case in the DC motor application. The reason for this is because the initial oscillatory parameters take a long time to converge when applied to the oscillatory plant. Parameters tune with high gradient from sample zero to sample 200, then they take a gradual gradient, but continue optimising since the cost function gradient has not minimised. Parameters together with their respective roots for selected samples are tabulated in table 12.

Table 12: Roots of the oscillatory plant for the closed-loop control system

Sample No.	Rho0	Rho1	Roots
0	1.00	10.00	-1.16 + 1.05i -1.16 - 1.05i 0.22
50	0.50	8.00	-1.15 + 1.04i -1.15 - 1.04i 0.2121
100	0.50	4.00	-1.16 + 1.02i -1.16 - 1.02i 0.23
1000	0.50	3.50	-1.16 + 1.02i -1.16 - 1.02i 0.24
1600	0.50	3.20	-1.17 + 1.02i -1.17 - 1.02i 0.24
2000	0.50	3.00	-1.17 + 1.02i -1.17 - 1.02i 0.24

The roots of the characteristic equation for parameters of selected sample numbers tabulated in table 11 demonstrate two complex poles outside the unit circle, with only the real pole inside the unit circle – a similar trend to the one observed in the fast-damped case of the oscillatory plant. Hence the same argument regarding stability is also applied here.

3.3.5 Simulation of the IFT using the unstable plant

In this section, we test the IFT by applying it to the unstable plant given in equation (31),

using a similar procedure as in the IFT applied to the oscillatory plant, and the plantsampled into a digital equation. The Matlab code for converting from s-domain to z-domain is given in appendix V under the derivation for equation (39). The obtained equation is given in equation (37).

$$gh = \frac{0.1052}{z - 1.105)} \text{ [V]} \qquad (37)$$

Hence equation (37) is converted to a digital equation as given in equation (38).

$$gh = \frac{0.1052 * z^{-1}}{1 - 1.105 * z^{-1}} = \frac{y}{u}$$

$$y_t = 1.105 * y_{t-1} + 0.1052 * u_{t-1} \qquad (38)$$

The closed-loop control system for the unstable plant is also derived in appendix V under the derivation for equation (39) and the final transfer function is given in equation (39).

$$gh(z) = \frac{0.1052 * \rho_0 * z \ + (0.1052 * \rho_1 * T - 0.1052 * \rho_0)}{z^2 + (0.1052 * \rho_0 - 2.105) * z \ + 1.105 + 0.1052 * (\rho_1 * T - \rho_0)} \text{ [V]} \quad (38)$$

(I) Slow-damped case

The same initial controller parameters ($rho0 = 0.1$ and $rho0 = 0.2$) used for the oscillatory plant and the DC motor are applied here, and the IFT is implemented for 1000 samples. The results are illustrated in figure 10, though only 600 samples are indicated for the response and the error signal, while for parameters and the cost function, only 100 samples are indicated. This is mainly to make the graphs readable.

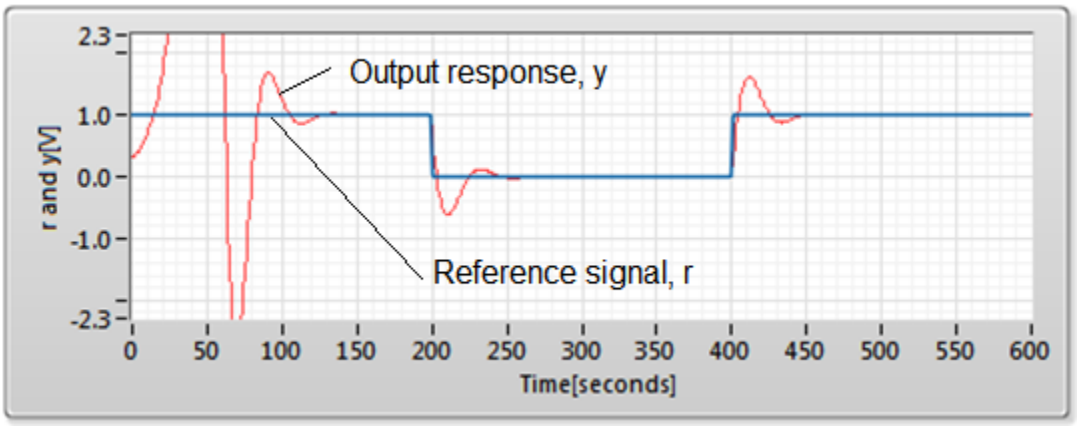

Figure 10(a): IFT response for unstable plant

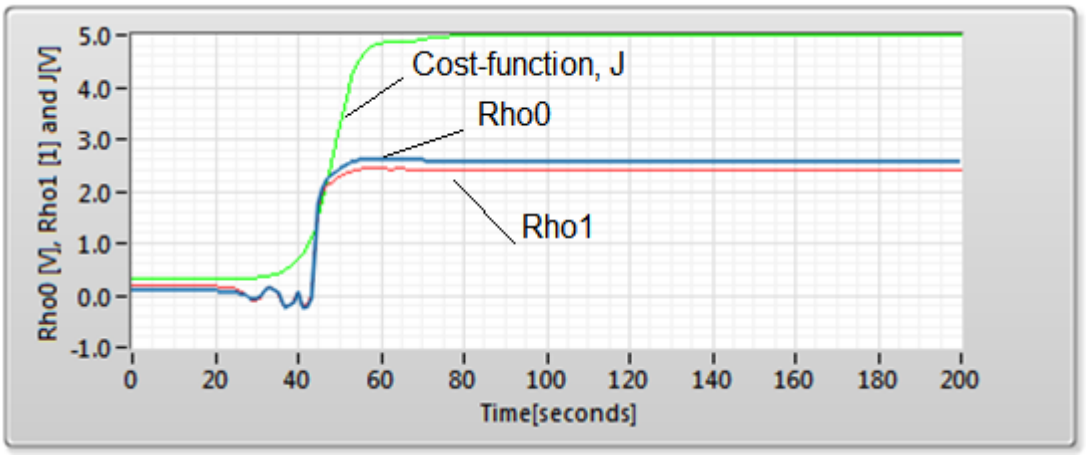

Figure 10(b): Parameter trajectory for unstable plant and cost function

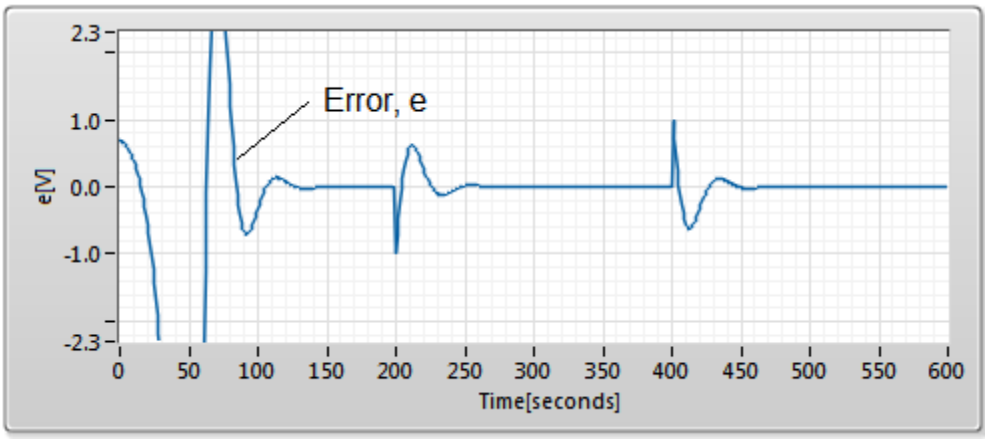

Figure 10(c): Error behaviour

The unstable plant response and reference signal is illustrated in figure 10a, followed by the controller parameters, rho0 and rho1. The cost function is illustrated in figure 10b, and the error signal in figure 10c. The response is unstable from sample number zero to sample number 100, where it converges to the oscillatory closed-loop system up to the last sample (sample number 1000). Figure 10b illustrates how the controller parameters vary, starting with sample number zero to sample number 100, where the gradient of the cost function decreased to zero. and Because of this occurrence, the error signal reduced to a minimum (see figure 10c). The roots of the characteristic equation for the unstable plant at selected sample points are tabulated in table 13. Using the roots at sample number 100, and also the given sampling time (200ms), the IFT is validated by computing the settling time and the period of the response. The settling time to the $\pm 2\%$ band is readily computed to be 9.1s, and the period of oscillation is 7.1s. This agrees with the response shown in figure 10a in that one cycle of the response (for example from sample 200 to sample number 237) takes 37 samples, which works out to be 7.4s. Also, the same response settles in 45 samples, working out to be 9s.

Table 13: Roots of the oscillatory plant for the closed-loop control system

Sample No.	Rho0	Rho1	Roots
0	0.10	0.20	1.06 1.04
30	-0.08	-0.10	1.12 1.00
36	-0.21	-0.21	1.14 1.00
42	-0.21	-0.20	1.14 1.00
100	2.60	2.40	0.92 + 0.13i 0.92 - 0.13i
1000	2.60	2.40	0.92 + 0.13i 0.92 - 0.13i

(II) Fast-damped case

For the fast-damped case, the controller parameters utilised are $rho0 = 1.0$ and $rho0 = 1.0$ – the same as used in the DC motor plant. The results generated are given in figure 11.

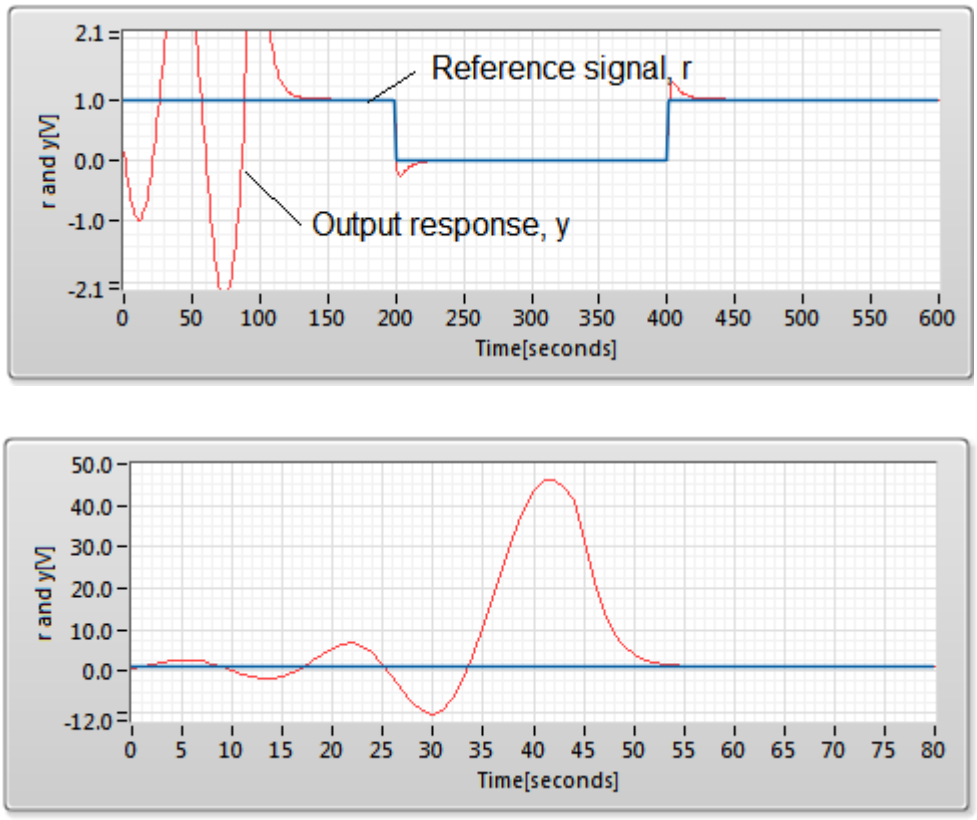

Figure 11(a): IFT response for unstable plant

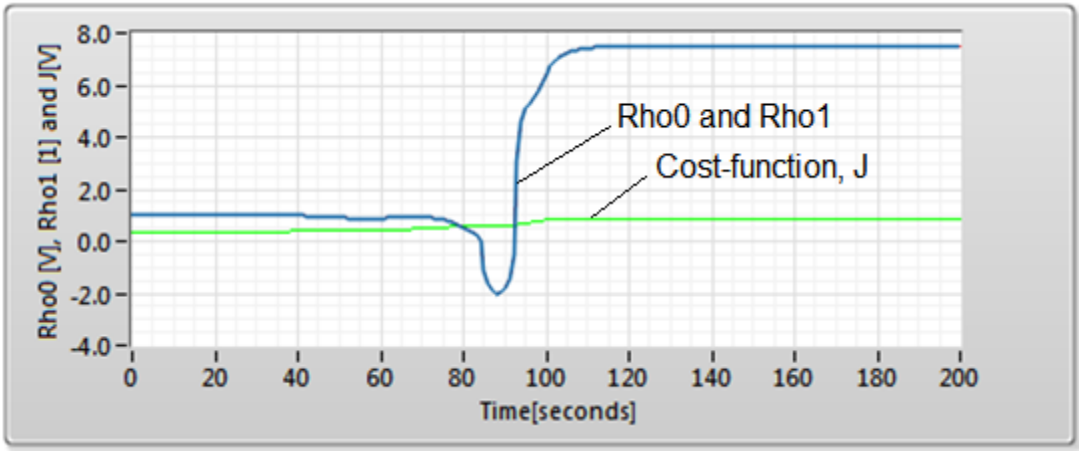

Figure 11(b): Parameter trajectory for unstable plant

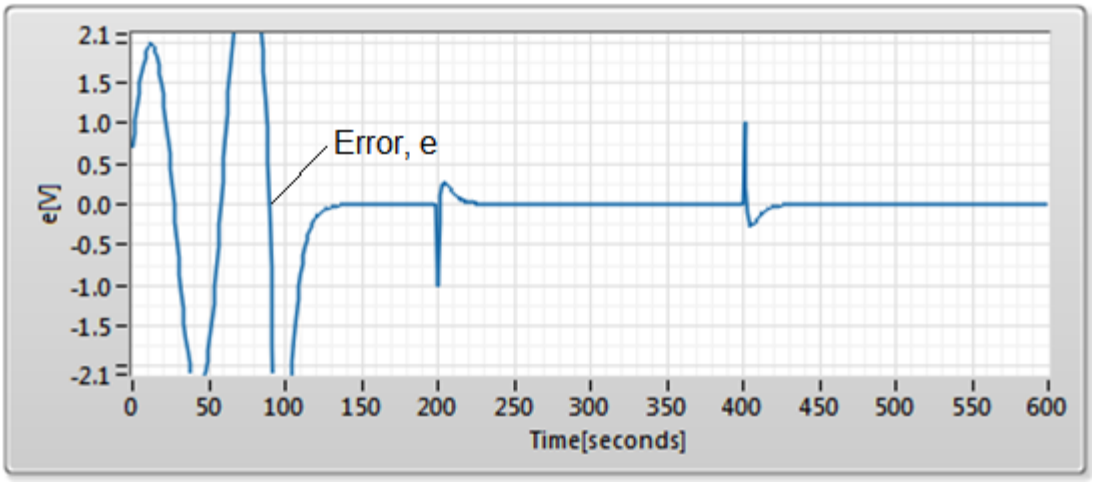

Figure 11(c): Error behaviour

The output response and the reference signal are indicated in figure 11a. The parameters, rho0, rho1 and the cost function are shown in figure 11b, and finally, the error signal is illustrated in figure 11c. The output response is unstable from sample number zero to sample number 130, and takes a longer time than in the case of the slow-damped case. At sample number 130, the controller parameters converge to a very fast response with an overshoot. The roots of the characteristic equation are computed for the selected sample points and tabulated in table 14.

Table 14: Roots of the fast-damped plant for the closed-loop control system

Sample No.	Rho0	Rho1	Roots

0	1.00	1.00	1.00 + 0.10i 1.00 - 0.10i
45	0.90	0.90	1.01 + 0.10i 1.01 - 0.10i
55	0.80	0.80	1.01 + 0.09i 1.01 - 0.09i
87.5	-2.00	-2.00	1.37 0.94
95	6.00	6.00	0.82 0.66
120	**7.50**	**7.50**	**0.85** **0.46**

(III) Oscillatory case

Finally, the initial controller parameters utilised for the oscillatory case are $rho0 = 1$ and $rho0 = 16$ – the same as in the case of the DC motor. The output response and the reference signal are indicated in figure 12a. The parameters, rho0, rho1 and the cost function are shown in figure 12c and the error signal is illustrated in figure 12d. The IFT is implemented for 1000 samples, however, only 600 samples are shown in figure 12. The results obtained are illustrated in figure 12a, and demonstrate oscillations from sample number zero to sample number 60, where the response converges to a stable oscillatory response with an overshoot of 1.1V and a settling time (to $\pm 2\%$ band) of 1.4s according to the root at sample number 200. This is the same as the response at sample number 200 (see magnified figure 12b).

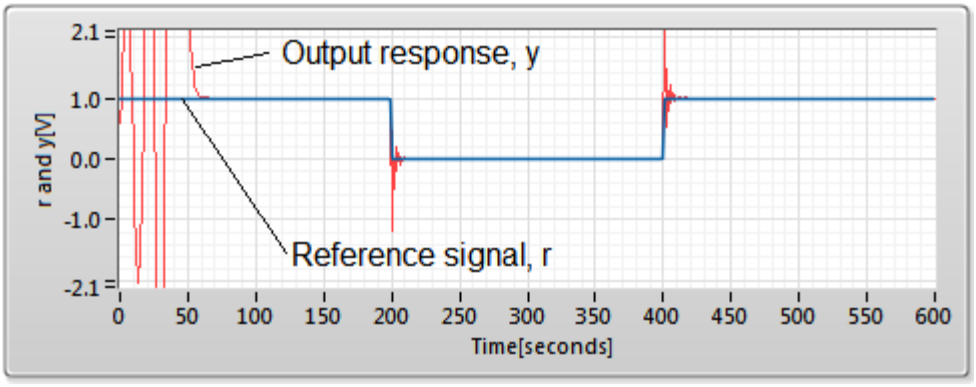

Figure 12(a): IFT response for unstable plant

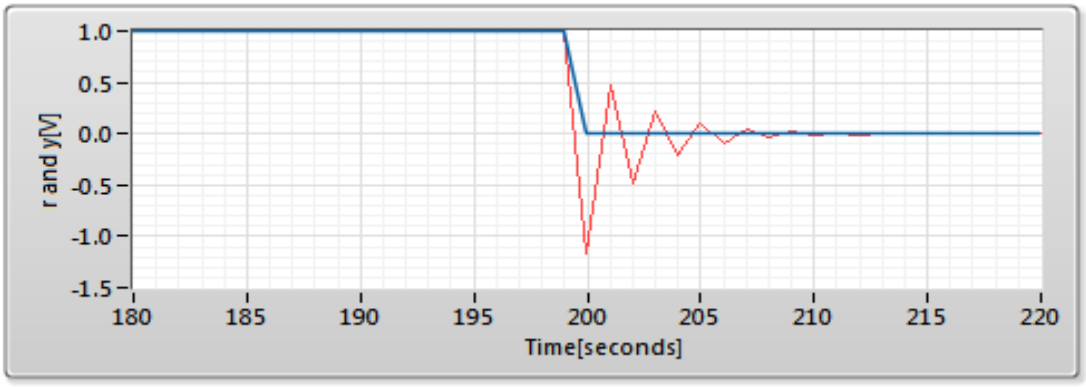

Figure 12(b): Magnified IFT response for unstable plant

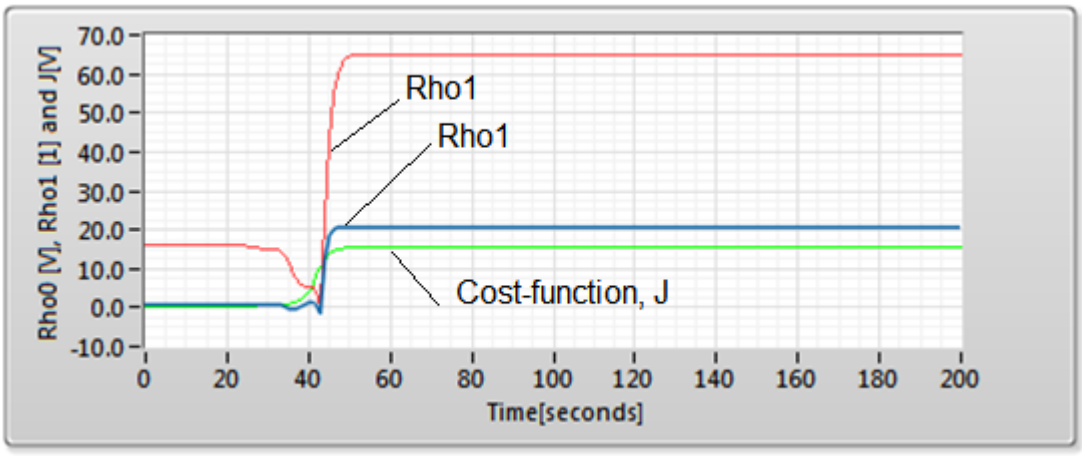

Figure 12(c): Parameter trajectory for unstable plant and cost function

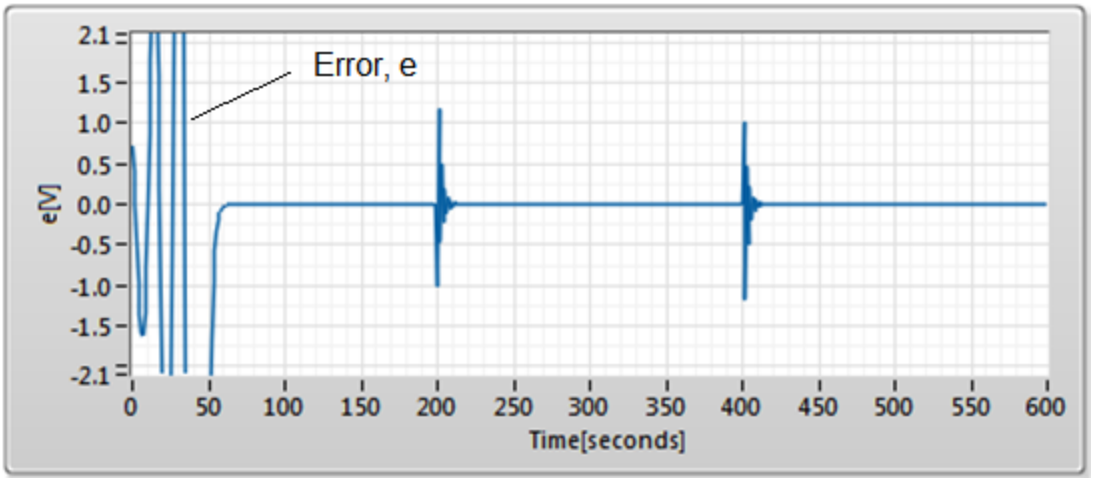

Figure 12(d): Error behaviour

Parameters vary from sample number zero to sample number 60, where the system converges to a stable oscillatory system of rho0 = 20 and rho1 = 65. The roots of the characteristic equation are computed at selected sample points and tabulated in table 15. The instability occurs only for a shorter time as compared to the two former cases (slow damped and fast damped). The observation reveals the response of high frequency initially for the period of 43 samples.

Table 15: Roots of the oscillatory plant for the closed-loop control system

Sample No.	Rho0	Rho1	Roots
0	**1.00**	**16.00**	**1.00 + 0.41i** **1.00 - 0.41i**
36	**-0.50**	**10.00**	**1.08 + 0.31i** **1.08 - 0.31i**
37	**-0.50**	**5.50**	**1.08 + 0.23i** **1.08 - 0.23i**
43	**-1.00**	**0.50**	**1.18** **1.03**
200	**20.00**	**65.00**	**0.56** **-0.56**

3.3.6 CHAPTER SUMMARY

In this chapter, the IFT technique was tested and validated, as demonstrated by the following:

- The initial controller parameters for the IFT should not be picked randomly, but should be determined by fixing the zero of the controller inside the unit circle, so that it helps to pull the poles of the closed-loop system inside during optimisation. This procedure, used for all three plants, confirmed the viability of the IFT technique in optimising badly tuned controllers, to improved ones.
- The Gamma constant needs to be selected in relation to the size of the error. For large error, the gradient of the cost function increases rapidly, and hence, requires a small gamma constant, and when the error is small, the cost function increases gradually, requiring a big gamma constant to tune the parameters quickly. Higher gamma constants may cause system blow-up due to internal variable overflow as a result of the identity matrix becoming asymmetric. The solution to the problem is to choose gamma constant by a line search.

- Reference signal amplitude variation has an effect on the convergence of the controller parameters, and system blow-up for constant gamma. If the amplitude of the reference signal is varied, then gamma should be set by the line search.
- This chapter demonstrated the universality of the IFT in controlling different types of systems – a validation that it can control a wide range of plants, as stated in the literature.

CHAPTER FOUR

DESIGN OF A DEDICATED IFT MICROCONTROLLER

4.1 INTRODUCTION

Having studied and validated the IFT technique in chapter three (a necessary task before the technique is committed to hardware the design, implementation and verification of a Dedicated IFT Microcontroller is now presented in this chapter. To accomplish this, an overview of FPGA technology and Electronics Design Automation (EDA) tools is presented. This is followed by investigating the feasibility of developing the IFT technique into an FPGA device, since it is the device earmarked for development of the Microcontroller. The design methodology adopted in this research is a popular hierarchical and modular top-down procedure, which is an array of abstraction levels (system level, behavioural level, register transfer level (RTL) and physical level). At system level, the hardware (Dedicated IFT Microcontroller) is defined or specified based on the IFT algorithm described in chapter three. Thereafter, at behavioural level, the design is simulated using VHDL, which is created by porting the LabView IFT code (developed in chapter three) to the Xilinx EDA tool. The simulation is meant to verify the feasibility of running the IFT on FPGA hardware. At register transfer level (RTL), a synthesisable VHDL code utilising fixed-point number representation is written. This is preferred to floating number representation for implementation into the FPGA, due to the benefit of performance. The compiled bit file is downloaded onto the target device (NI Digital Electronics FPGA Board featuring Spartan 3). Finally, the developed hardware is tested using a method known as simulation in the hardware.

4.2 OVERVIEW OF FPGA TECHNOLOGY

FPGAs were first introduced in 1984 by Xilinx [27], and since then have seen tremendous growth, and have become a popular implementation media for digital hardware. A few examples can be cited from the literature as follows: a digital controller for a dc-dc

converter using Floating Point Arithmetic [18]; modular FPGA-Based PID controller [19, 20]; an auto-tuned digitally controlled buck converter based on relay feedback [21], etc.

The most recent FPGAs (Xilinx's Virtex ultra scale+ and Kintex ultra scale+) are produced using a 16-nm copper process [30], and their density can reach more than 5540850 logic elements per component, with internal clocking resources of up to 1 GHz [29].

As a result of the advancement in process technology, FPGA's enhanced logic capacity made them a viable implementation platform for large and complex digital designs, which is why it is believed in this study that the IFT technique (despite being a complex technique) can be implemented in an FPGA. FPGAs present a compelling alternative for digital system implementation based on economical access to scalability and performance provided by Moore's law [28]. As Moore's law progresses, it will be difficult to use an application-specific integrated circuit (ASIC) because of difficulties brought about by state-of-the-art deep submicron processes in VLSI [28], a problem that can only be mitigated by using FPGA devices.

The two fundamental technologies that distinguish FPGAs are architecture and CAD tools that a user must employ to create FPGA designs. The goal of this survey is to investigate the existing state of the art in FPGA architecture and CAD tools, so that context can be provided for the design of the Dedicated IFT Microcontroller.

An FPGA is defined as a two-dimensional array of configurable logic blocks (CLBs) embedded into a sea of programmable interconnection networks, which connects the CLBs (according to the hardware requirement to implement) to each other. The final customised FPGA, representing the targeted application, is connected via input/output (I/O) blocks to the external world. A basic example of a generic FPGA architecture is depicted in figure 13.

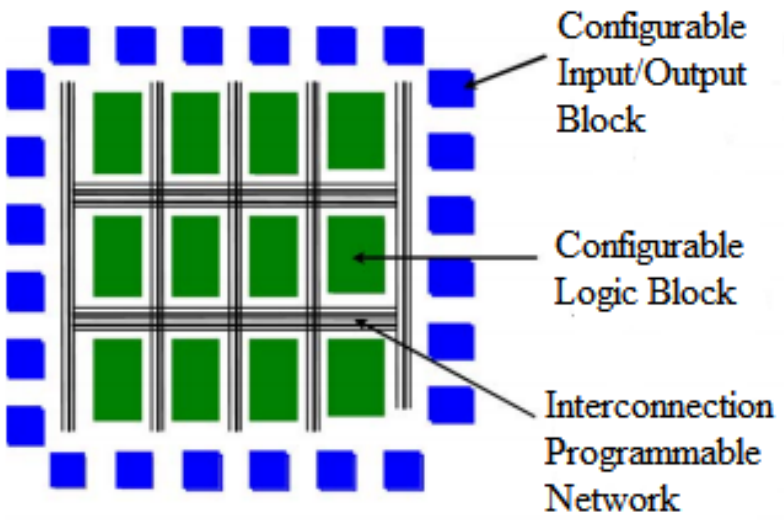

Figure 13: Generic FPGA architecture

In Xilinx terms, a CLB is a cluster of eight logic cells, or a group of four slices [31]. Based on the complexity of a generic CLB, for example the number of logic cells it contains, the CLB can be classified as either a fine grain (one logic cell or gate) or coarse grain (CLB containing several logic cells or gates). Examples of fine-grain FPGAs are Crosspoint, Plessey and Algotronix FPGAs. The main advantage of using fine grain is that CLBs are fully utilised because it is easier to use small logic gates efficiently [38]. On the other hand, fine-grain FPGAs have problems in that they use a large number of routing segments, resulting in a cost in area, since it is the routing network that takes a larger percentage of area in an FPGA (ranging from 80% to 90%), in comparison to CLBs that takes only 10% to 20% [38]. Alternatively, a coarse-grain CLB comprises look-up tables (LUTs), which are very common in Actel, Xilinx and Altera. The structure of a logic cell, which is a fundamental grain of the FPGA, is illustrated in figure 14.

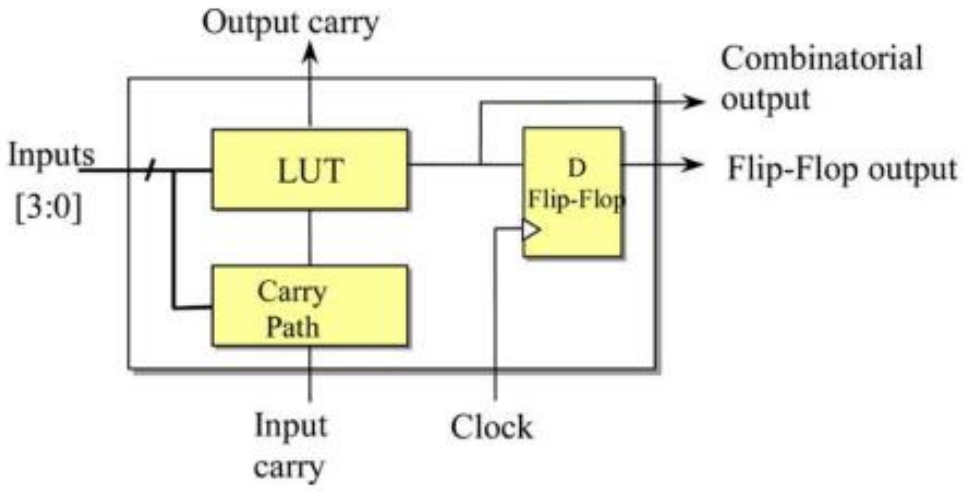

Figure 14: Logic cell structure [11]

It consists of a four-bit look-up table (LUT-4) that can be configured as ROM, RAM or a combinatorial function. A carry-look-ahead data path is also included in the cell to build an efficient arithmetic operator. A D-type flip-flop, with all its control inputs (synchronous or asynchronous set/reset, enable) permits registering the output of the logic cell [12]. Finally, a coarse-grain FPGA does not require a large number of routing segments [38] as is required in fine grain because the density of CLBs is higher. This information is significant in that it helps in selecting a suitable FPGA for use in a desired application. In this research, a coarse-grain FPGA is selected for the design of a Dedicated IFT Microcontroller, because the technique is complex and its implementation requires an FPGA of high logic capacity.

Three types of programming technologies are utilised in FPGAs, and these are outlined as follows: static RAM (SRAM), flash (EEPROM), and anti-fuse. These programming technologies characterise the architecture of an FPG and can also be used as a criterion to select a preferred FPGA for an application in question. In the case of the IFT technique implementation, SRAM technology was selected. This is because SRAM is by far the most widespread technology due to its simple integrated circuit process (it uses standard CMOS), and it has fast re-programmability, which is good for testing applications. The SRAM FPGAs are normally paired with external flash memory on the same printed circuit

board (PCB) to preserve the application program when power is off. However, the flash FPGA is of interest for its non-volatility and area efficiency [28] and can be applied in the final design or perfected application because it preserves the configuration of an FPGA when the power is off without an external flash memory.. Furthermore, the flash-based FPGAs guarantee the configuration against the single-event upset (SEU) radiations [28] – a potential candidate for aerospace application.

More recently, a trend of heterogeneous architectures emerged, with the introduction of some dedicated blocks such as Random Access Memory (RAM) DSP accelerator units (hardwired multipliers with their accumulators, high-speed clock management circuitry, and serial transceivers). The units are embedded into hard processor cores such as the PowerPC, or advanced reduced instruction set computing (RISC) machines [31, 33], and soft-processor cores such as Nios [32], Picoblaze [31], and Microblaze [31]. Moreover, an interesting feature of control applications is the recent integration of an ADC in the fusion component from Actel (now known as Microsemi) [33].

FPGAs are frequently used to implement complex functions, and the implementation of the IFT technique (quite complex also) on an FPGA could be feasible. This claim regarding an FPGA having the capability to accommodate complex treatment is backed by recent advancement in VLSI, and development of design tools such as hardware description languages (HDL) such as VHDL and Verilog [34, 35]. Other developmental tools that are emerging nowadays (from both industry and academia) are high-level synthesis (HLS) and electronic system level (ESL) synthesis tools. These tools have been in the research domain for over three decades, though only recently have found their use due to the need for quality results (in terms of performance and energy efficiency, and strict time-to-market schedules) in embedded applications [36]. Another interesting feature concerning HLS tools is the possession of floating-point number representation, which is currently not easy to use in VHDL or Verilog. HLS tools are based on C/C++ or system C high-level languages, which have floating-point number capability. The design methodologies using the abovementioned tools are briefly described in appendix II.

Finally, the FPGA in use in this research is the National Instruments Digital Electronics FPGA Board. The NI Digital Electronics FPGA Board is a circuit development platform based on the 3S500E Xilinx Spartan- 3E FPGA. Besides the FPGA, the board comprises:

- sliding switches,
- LEDs,
- two-digit, seven-segment display,
- push buttons,
- rotary push-button knob and LEDs for the external clock,
- diligent Pmod terminals for external attachment,
- USB download interface, and
- large breadboard area for digital electronics circuitry.

4.3 DESIGN METHODOLOGY

As already mentioned, our design follows a hierarchical and modular approach and we use the Xilinx ISE EDA tool to accomplish this goal. Going through the levels of abstraction, the design is formulated as follows:

4.3.1 System Level Abstraction

System level is the first step in the design process. It constitutes defining or specifying the Dedicated IFT Microcontroller. However, in our case, the specification was done in chapter three where the IFT technique was formulated and simulated with the help of the LabView software. However, we redefine it here again in the form of its architecture, as illustrated in figure 15, using the equations that were derived in chapter three (to be specific, in subsection 3.2.4). Figure 15 can also be termed as a data path (where operations are performed) for the Microcontroller.

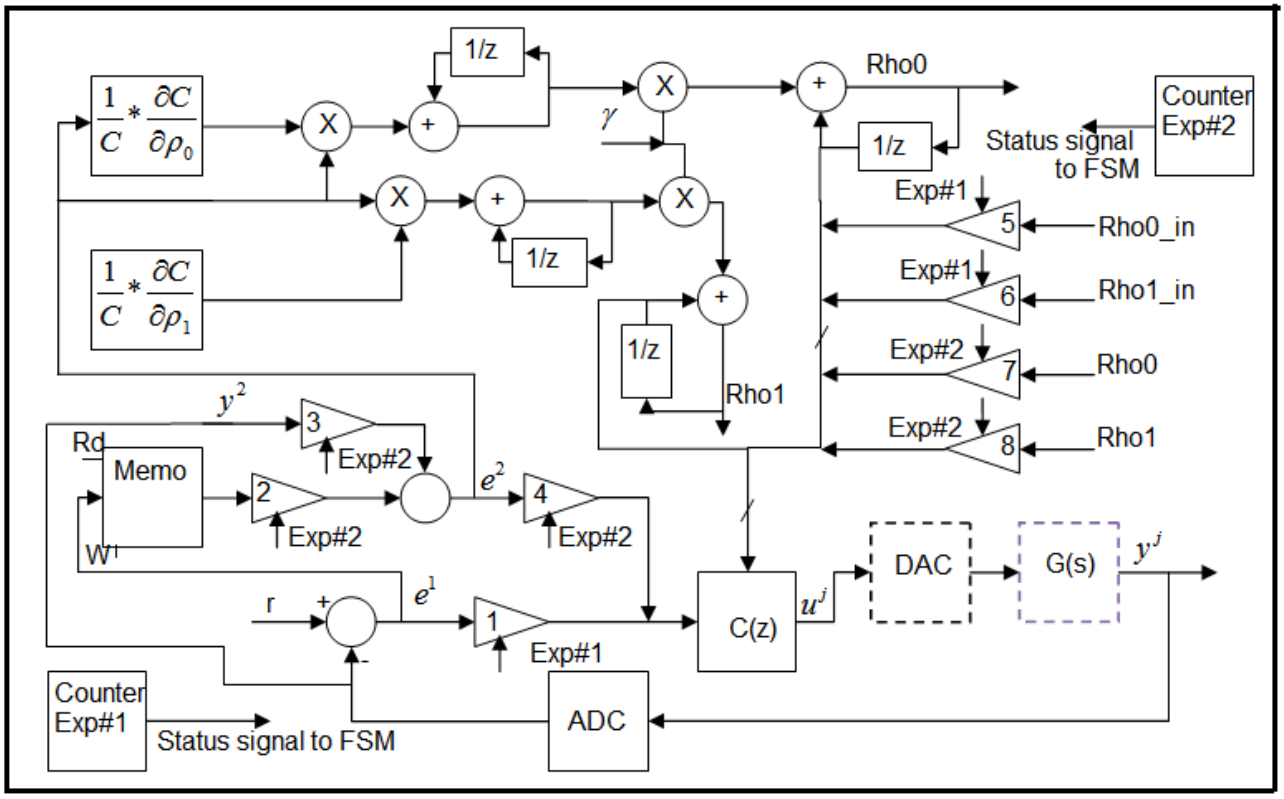

Figure 15: IFT technique architecture

The IFT is formulated using traditional error, thus only the first term of equation (14) is used, as shown in equation (40), mainly for simplifying the hardware.

$$J = \frac{1}{2N}\sum_{i=0}^{N}[(L_e * e^2(\rho))^2 \qquad (40)$$

The inputs to the data path are the reference signal, plant output signal, gamma constant and the parameters designated by $r(i)$, $y^j(i)$, γ, and $rho0\_in$ $rh01\_in$ respectively. The parameters are presented in different symbols to those used in chapter three, mainly to indicate that there are initial parameters, and only used in experiment# 1. The flow of signals is described exactly as in chapter three. The two counters, (the counter for experiment# 1 and the counter for experiment# 2) seemingly not connecting to anything in the architecture, are loop counters for the two experiments. The counter for experiment# 1 counts the number of iterations for an N-length period in experiment# 1.

Similarly, the counter for experiment# 2 counts the number of iterations for an N-length period in experiment# 2. When the iteration count is reached, the counter produces a status signal for a finite-state machine (FSM), which correspondingly sends an instruction to the data path to change the state of the FSM. There are eight tri-state buffers, numbered from 1 to 8, responsible for transitioning of the IFT from one experiment to the other. The control terminal for each tri-state buffer is signified by either exp# 1 or exp# 2, implying the signals buffered by the respective tri-state buffers can only be allowed in a specific experiment, and the dynamics of the transition are coordinated by the FSM, also known as the control unit. The DAC and plant blocks have their outline dashed to illustrate that there are external devices that should not be included in the integrated circuit (IC). However, the ADC block shown in the solid outline implies that it can be part of the IC since some FPGA vendors have managed to fuse an ADC into the FPGA.

The data path cannot function by itself and requires a controller termed a finite-state machine, or control unit to supervise it. The other related name for the FSM is an algorithm state machine (ASM), similar to traditional flowcharts used for programming languages. Hence, the FSM and ASM is also defined in relation to the data path, in order to have a complete specification of the dedicated Microcontroller.

The FSM generates instructional signals to the data path in relation to the clock signal. A block diagram of an FSM is shown in figure 16, with accompanying instructional and status signals. The FSM for the IFT Microcontroller generates two signals to start experiment# 1 and experiment # 2 sequentially, and it also receives status signals to detect whether the cycle has ended in experiment# 1 or experiment # 2. It also takes the clock and error signals as inputs. This FSM is termed as Mealy machine as whose output depends both on current state and inputs. The error signal is checked to detect whether it has gone above the tolerated margin or not, and if this is true and the cycle has also ended, Experiment# 2 must begin. This explanation is easily illustrated by a state diagram, which is indicated in figure 17.

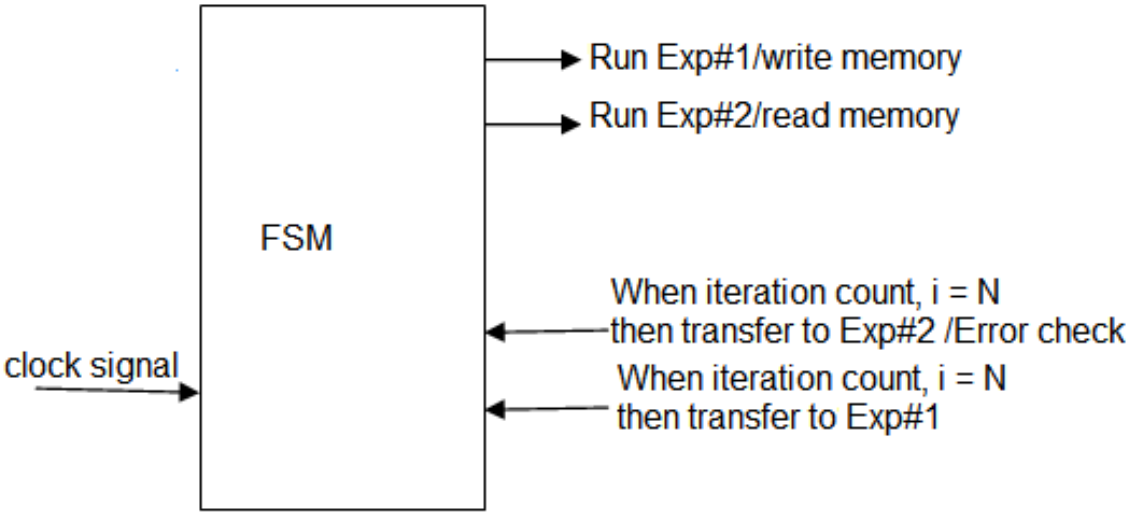

Figure 16: IFT technique finite-state machine block diagram

The inputs are denoted as x for the condition "error e > tolerated error e_t and counter exp# 1 $= N$," and the input for the condition "counter exp#2 $\neq N$" as c. Moore and mealy outputs are also denoted for Experiment# 1 and Experiment# 2 as y_a, y_b, y_1 and y_2 respectively.

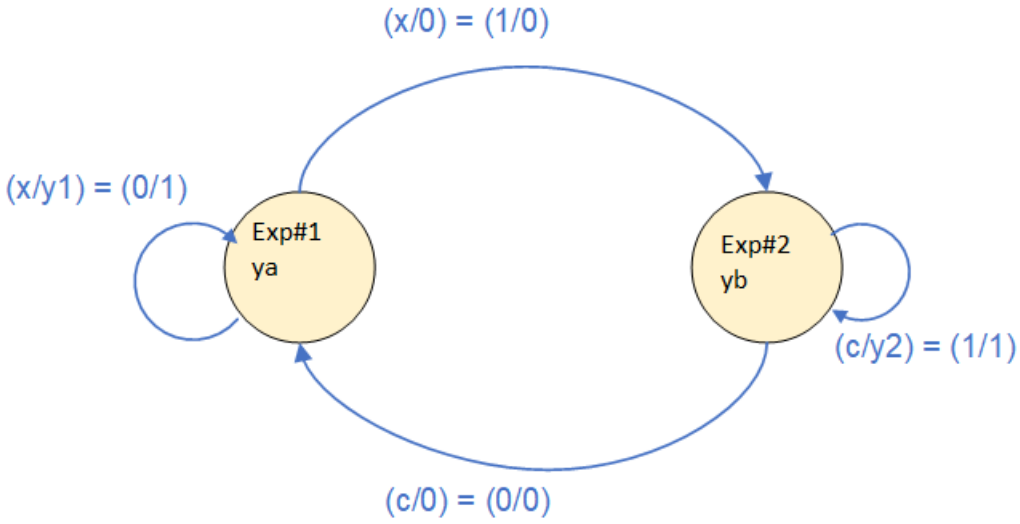

Figure 17: IFT technique FSM state diagram

As shown in figure 17, when the input signal $x = 0$, the transition arrow loops in Experiment# 1 and mealy output $y_1 = 1$ to drive the tri-state buffers in Experiment# 1, but when $x = 1$ the arrow transitions to Experiment# 2 with the mealy output $y_2 = 0$. In Experiment# 2 state, the input $c = 1$ makes mealy output $y_2 = 1$ to drive the tri-state buffers for Experiment# 2, but when $c = 0$ the terminal transitions to Experiment# 1. Finally, the controller can also be expressed in terms of an algorithm state machine (ASM). This is a better representation because it mimics the hardware. Figure 18 illustrates the ASM chart for the IFT technique.

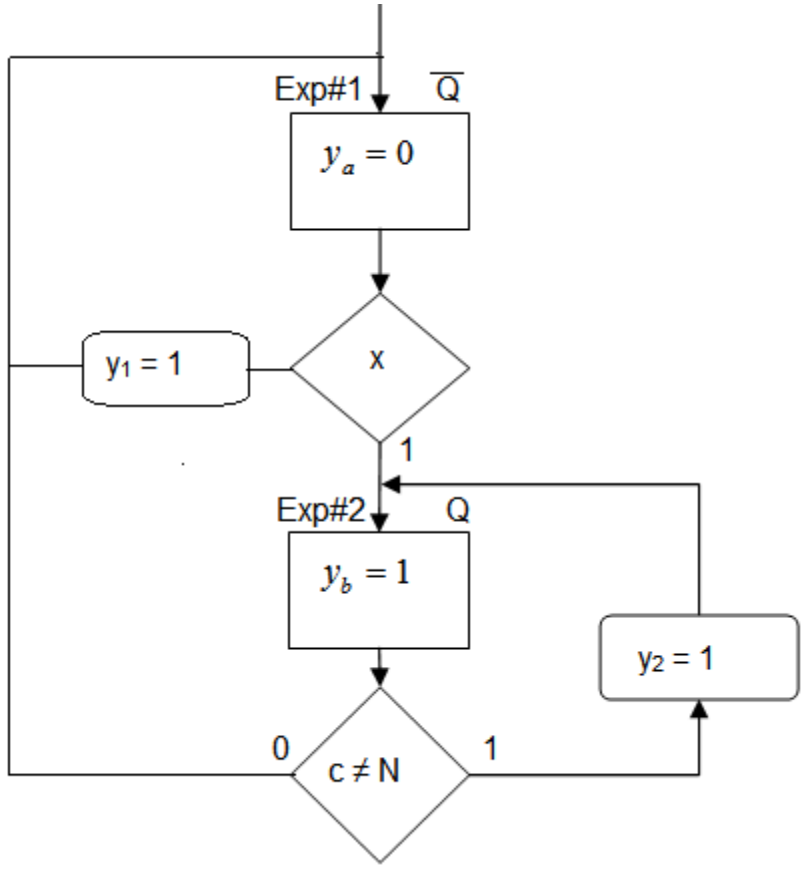

Figure 18: IFT technique ASM chart

From the ASM chart, Boolean expression is thus easily formulated:

Moore output $y_a = \overline{Q}$ and Moore output $y_b = Q$

Mealy output $y_1 = \overline{Q} * \overline{x}$ and Mealy output $y_2 = Q * c$

Next state equation $Q^+ = \overline{Q} * \overline{x} + Q * c$

The derived Boolean expression is easily utilised to design the FSM for controlling the data path given in figure 9.

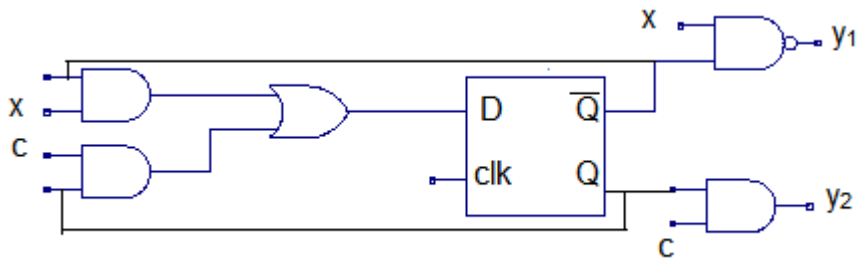

Figure 19: IFT technique FSM

X is the condition for the iteration count, i = N and error, e > tolerated error; c is a condition for iteration count, i ≠ N; y_1 and y_2 are signals to operate tri-state buffers for Experiment# 1 and Experiment# 2 respectively.

In the system level abstraction, the IFT dedicated Microcontroller was specified. The next stage is the behavioural level of abstraction where the code is written to simulate the design. The code is usually written in programming languages such as Matlab, LabView, or C++. Since this task was done in chapter three, it will not be repeated here, but instead the hardware will be describe using a hardware description language and will be simulated on the VHDL platform of ISim Xilinx simulator.

4.3.2 Behavioural Level Abstraction

In this section, the behaviour of the algorithm using VHDL is described. The development of the VHDL code for the IFT Microcontroller is described in the next subheading.

I. Development of VHDL Code for IFT Microcontroller

Following the simulation of the IFT technique on the LabView platform, and its specification in the system level abstraction, the behavioural VHDL code is developed and simulated in this section. Behavioural modelling is chosen because of its sequential attributes, a property suitable for algorithms like the IFT technique. It uses processes to describe behaviour of hardware, similar to programming languages, since program statements in a process are executed sequentially. Hence, any code developed in LabView or the C-language can be ported to a VHDL process block, and would run as expected. With this background, the simple procedure utilised was to port the LabView code into the Xilinx ISE EDA tool, and with minimal modification, a behavioural model VHDL code that describes the IFT hardware was developed. This type of modelling is also known as finite-state machine with data path (FSMD), and is preferred to finite-state machine plus data path (FSM + D), for ease of implementation.

The IFT Microcontroller VHDL code is composed of three processes, namely: a process for clock division, a process for generating the reference signal, and a process for the PI controller combined with the IFT technique. The last process is ported from the LabView IFT code.

In the development of the VHDL code, the LabView code was modified to use fixed-point number representation in order to enhance accuracy of the controller. Floating-point number representation is the best with regard to accuracy, but its implementation is complicated and very demanding in terms of hardware resource.

The VHDL code, formulated for the IFT Microcontroller with fixed-point number representation. To transform the LabView code into the VHDL code, a word length of 12 bits was chosen to match that of the ADC and the DAC, which also have 12-bit resolution.

The fixed-point number representation is denoted by Q(m, -n) format. Where Q is a fixed-point number variable, m is an integer part of a fixed-point number, and n is a fraction part

of a fixed-point number. For 12-bit word length and a 2V maximum input/output signal, the format is given as Q(1, -10), meaning that the fixed-point number has 2 bits for the integer part and 10 bits for the fractional part. Hence, using the specified Q format, all LabView file expressions is written in terms of fixed-point representation. The modified instructions are then converted to VHDL expressions. An example is demonstrated for the following expressions from the LabView code:

- the comparator, e = r - y,
- the PI controller, u[t] = u[t-1] + rho0*e[t] + (rho1*T - rho0)*e[t-1], and
- the plant, y[t] = 0.90483*y[t-1] + 0.09516*u[t].

Taking the instruction e = r – y, it is modelled in the form of fixed-point number representation, thus e(2, -10) = r(1, -10) - y(1, -10).

The result of the subtraction register e, is higher by 1 bit than the operand registers. This is known as an overflow, which can have an impact on the signal level. For example, in this design the reference signal varies between 0V and 2V, meaning that, the result is upper-bounded by 2V, and if an overflow occurs, the signal rises beyond 2V, which is a problem. To resolve this problem, the most significant bit (msb) of the register is dropped (without any loss of accuracy) by scaling or shifting. Hence, register e(2, -10) is shifted to the left by 1 bit yielding e(1, -11), which can still be resized or truncated to e(1, -10) to maintain the 12-bit format.

Taking the next instruction that implements the PI controller gives
u[t] = u[t-1] + rho0*e[t] + (rho1*T - rho0)*e[t-1]
This is modelled in the form of fixed-point number representation as
u[t](5,-21) = u[t-1](1,-10) + rho0(1,-10)*e[t](1,-11) +
(rho1(1,-10)*T(1,-10) - rho0(1,-10))*e[t-1](1,-11)
=u[t-1](1,-10) + rho0*e[t](3,-21) + (rho1*T(3,-20) - rho0(1,-10))*e[t-1](1,-11)
=u[t-1](1,-10) + rho0*e[t](3,-21) + ({rho1*T-rho0}(4,-20))*e[t-1](1,-11)
=u[t-1](1,-10) + rho0*e[t](3,-21) + {rho1*T+rho0}*e[t-1](6,-31)
=u[t-1](1,-10) + (rho0*e[t]+{rho1*T+rho0}*e[t-1])(6,-31)
=(u[t-1]+(rho0*e[t]+{rho1*T+rho0}*e[t-1]))(7,-31)

The result of the control register u[t] has 7 bits for the integer part and 31 bits for the fractional part. This is scaled by shifting 6 times to the left, yielding u[t](1, -37). Finally, the plant expression, given as in equation (19) is reproduced here for convenience.

y[t] = 0.90483*y[t-1] + 0.09516*u[t]

It is also modelled in the form of fixed-point number representation as follows:

y(1, -20) = C(1, -10)*y(1, -10) + D(1,-10)* u(1, -10)
= C(1,-10)*y(1, -10) + D(1,-10)*u(1, -10)
= C*y(3,-20) + D*u(3,-20)
=[C*y+D*u](4,-20)

C and D are the constants of the plant, 0.90483 and 0.09516 respectively.
The result of the output y, has 4 bits for the integer part of the fixed-point number, and 20 bits for the fractional part of the fixed-point number. This is resized or truncated to allow it to fit on the output port, which is a 12-bit word length.

The next procedure was to code in the VHDL language all the modified expressions. The code for the comparator, PI controller and the plant is given in table 16. Similarly, computation for the IFT component is carried out, however, the transformation is given in appendix I. The complete VHDL code is assembled for simulation using the Xilinx IS simulator.

The next level of abstraction is the register transfer level, where synthesisable VHDL code is developed.

Table 16: VHDL code

Variable e: ufixed(2 downto -10);

```
Variable r: ufixed(1 downto -10);
Variable y: ufixed(1 downto -10);
Variable e: ufixed(2 downto -14);
Variable u: ufixed(4 downto -29);
Variable rho1: ufixed(1 downto -14);
Variable rho0: ufixed(1 downto -14);
 Variable gain: ufixed(1 downto -14);
Variable y: ufixed(1 downto -14);
e[t] := r[t] – y[t];

u[t] = u[t-1] + rho0*e[t] + (rho1*T - rho0)*e[t-1] ;
y[t] := 0.9048*y[t] + 0.0962*u[t];
```

4.3.3 RTL abstraction

At the RTL level of abstraction, synthesisable VHDL code is developed and simulated, that implements process blocks that make up a dedicated IFT microcontroller. The blocks are outlined as follows: the clock division block for the PI controller and the IFT; the clock division block for the analogue to digital converter (ADC); the block for the reference signal; the main block for the PI controller and the IFT technique, and the block for six channels of the pulse width modulation DACs. A summary of these blocks is given in table 17. After this, the process blocks are instantiated (connected) into a complete hardware. This kind of modelling is a combination of behavioural and structural modelling. It was adopted because the IFT technique is a highly complex technique to be modelled in the structural level entirely, as structural modelling is a low level approach. Each process block developed is tested before connecting it into the system, a paradigm known as modular design, permitting reuse. In summary, the process blocks are tabulated in table 17. The VHDL codes for the process blocks are illustrated in appendix VI.

Table 17: VHDL code structure for IFT technique

Process Type	Function

Clock Divisiontoggle	Generates clocks for the main process and the r(t) generator process
Clock DivisionADC	Generates clocks for the ADC process
Reference signal process	Generates set-point signal, r(t)
Analogue to Digital converter process (ADC)	Interfaces analogue signals to the IFT technique hardware
Main Process	Implements the PI and IFT technique
Six Pulse Width Modulated Digital to analogue converters (DAC) process	Converts digital signals (set-point, controller parameters, and output signal) to analogue signals for displaying onto the labview oscilloscope

4.3.3.1 Simulation of VHDL code for IFT Microcontroller

The VHDL code developed at RTL level is simulated in this section. This simulation differs from the one in chapter three since its focus is on validating the feasibility of running the IFT technique on the FPGA hardware. A test bench is developed in HDL and perform the RTL simulation. The RTL term is used in this instance to mean the HDL code is formulated at the register transfer level. A test-bench (functions as a virtual lab bench) consisting of the IFT Microcontroller module (described in VHDL language), and a code segment to generate a stimulus, are created. The stimulus is 'reset' signal for resetting and experiment signals for switching from one experiment to another respectively. The results of the simulation are illustrated in figure 20.

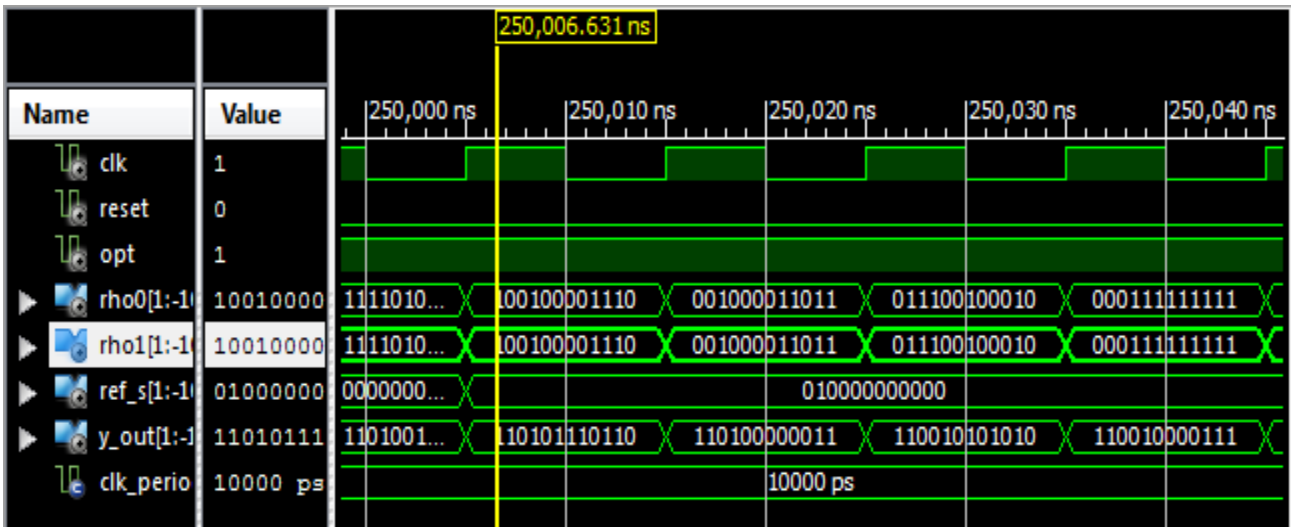

Figure 20: Simulation Results for the IFT VHDL Code

Figure 20 illustrates the optimisation action of the IFT, in that the parameters (rho0 and rho1) show some degree of variation as the tuning process progresses. Even the output response of the plant, y_out, shows some level of change in relation to variation in parameters. This is a demonstration that IFT technique can run on FPGA hardware.

Device utilisation is checked to ascertain the amount of hardware required to develop the Dedicated IFT Microcontroller when modelled using the fixed-point number representation, and the results can be compared with other models. The device utilisation summary is tabulated in table 18.

Table 18: Device Utilisation Summary

	Used	Available	Utilisation	Comment
Number of slice flip flops	919	9,312	9%	ok
Number of 4 input LUTs	1249	9,312	13%	ok
Number of occupied slices	940	4,656	20%	ok

Resource usage is below 50%,indicating that the IFT can be programmed on a smaller FPGA hardware.

4.3.4 Gate level abstraction

At the gate level of abstraction, the Xilinx EDA tool was used to synthesise and implement the VHDL code developed at RTL level. The synthesis process converts the hardware description of the IFT Microcontroller into generic gate level components (logic gates and flip-flop) for implementation into the FPGA. This is where various IFT Microcontroller design files were merged or translated into a single netlist,for technology mapping, placing and routing. To elaborate on the three implementation processes, it can be expanded as follows: at the translation stage, the software merges multiple design files to a single netlist, followed by mapping generic gates to the FPGA's logic cells and input and output pins (I/OBs). After mapping, placement and routing is commenced. At this stage, the cells are placed in physical locations in the FPGA, and determines the routes to connect various signals. Finally, the EDA tool generates a programming file that is downloaded into the FPGA.

4.4 CHAPTER SUMMARY

In this chapter, the Dedicated IFT Microcontroller was designed by formulating a data path (for the PI controller and the optimisation unit) and the FSM for sequencing the operations of the data path, as the IFT is a data based or sequential technique. To the best knowledge of the author, the implementation of IFT into FPGA hardware has never been published in literature before. The design was also simulated on ISim Xilinx simulator to check the validity and resource usage.

CHAPTER FIVE

TESTING OF A DEDICATED IFT MICROCONTROLLER

5.1 INTRODUCTION

In this chapter, a Dedicated IFT Microcontroller is tested. A rig is set up utilising NI Elvis II+, a Digital Electronics FPGA Board, Xilinx ISE Design suite version 12.2 EDA tools, LabView 2015 and a Toshiba laptop running Windows 2007. The model used in the design is the IFT of 1DOF.

The process of experimentation begins by downloading VHDL code, developed in chapter four, into the Digital Electronics FPGA Board. It is conducted using a technique known as simulation in the hardware, since all the components of the microcontroller and the plant are imbedded into the FPGA. Six signals are monitored, namely: the output response, reference signal, two parameters, the error signal and control signal. These signals are analysed and compared to those of theory of chapter three, for validation.

5.2 EXPERIMENTAL SETUP

The procedure adopted utilised the VHDL code developed in the subsection 4.3.2, by embedding it (all the components, including the plant) into the FPGA device. The code was implemented, and monitored the plant output response signal, the reference signal, two parameter signals, the error and control signals for analysis and comparison.

The setup in this experiment included the following: Digital Electronics FPGA Board using Spartan 3 from National Instruments; a Xilinx EDA tools, and an NI Elvis II+ panel. The NI Elvis II+ integrates 12 most commonly used instruments hosted on the LabView platform, making it easy to carry out the experiment. The instruments include an oscilloscope, digital multimeter, function generator, bode analyser and a variable power supply. Figure 21 depicts the picture of the setup used in the research.

The six signals above were measured and analysed through display on the LabView platform. By use of the data acquisition system (DAQ) hosted in the Elvis panel, signals from the FPGA board were acquired for measurement. The setup in the form of a block diagram is illustrated in figure 22.

The spartan 3 FPGA hosts the VHDL code describing a Dedicated IFT Microcontroller, the DC motor indicated as G(z), six channels of PWM, and DACs, making the microcontroller self-contained. Six signals in PWM format are extracted from the FPGA (at run time), and are passed through a block of low pass filters for conversion (averaging) into analogue signals, which are then acquired by the data acquisition system embedded in the NI Elvis II+ panel, for transmission to the LabView display (see figure 21 and 22).

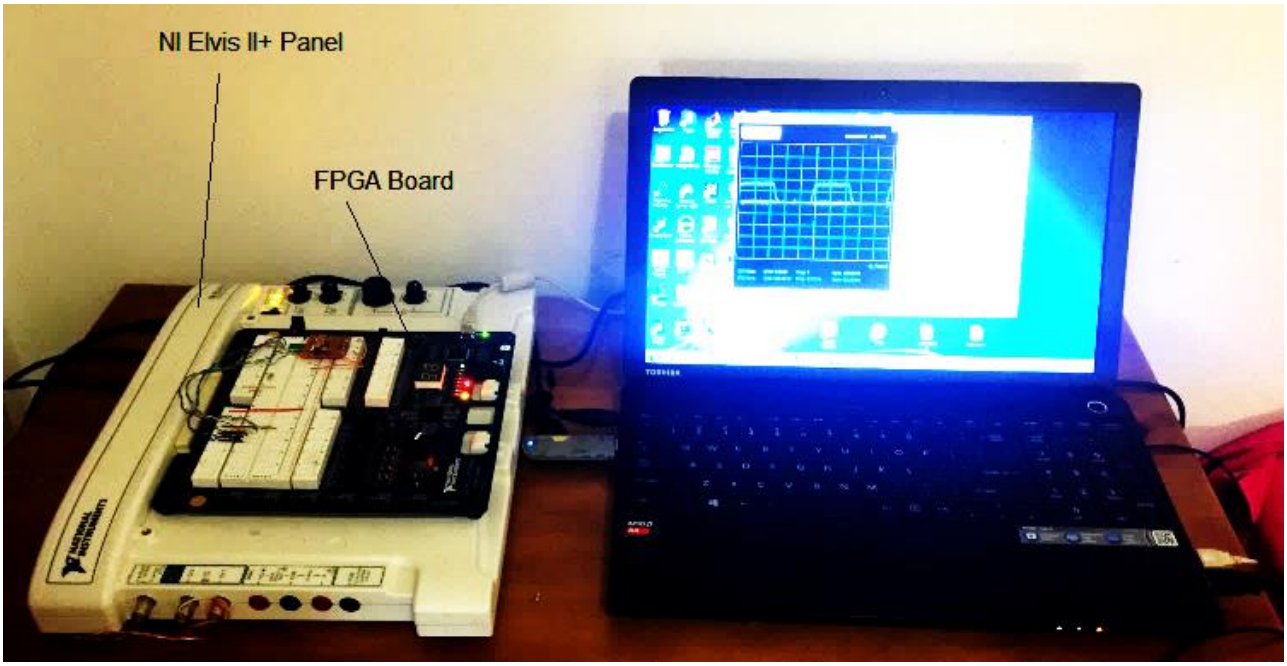

Figure 21: Experimental setup for testing of IFT Microcontroller

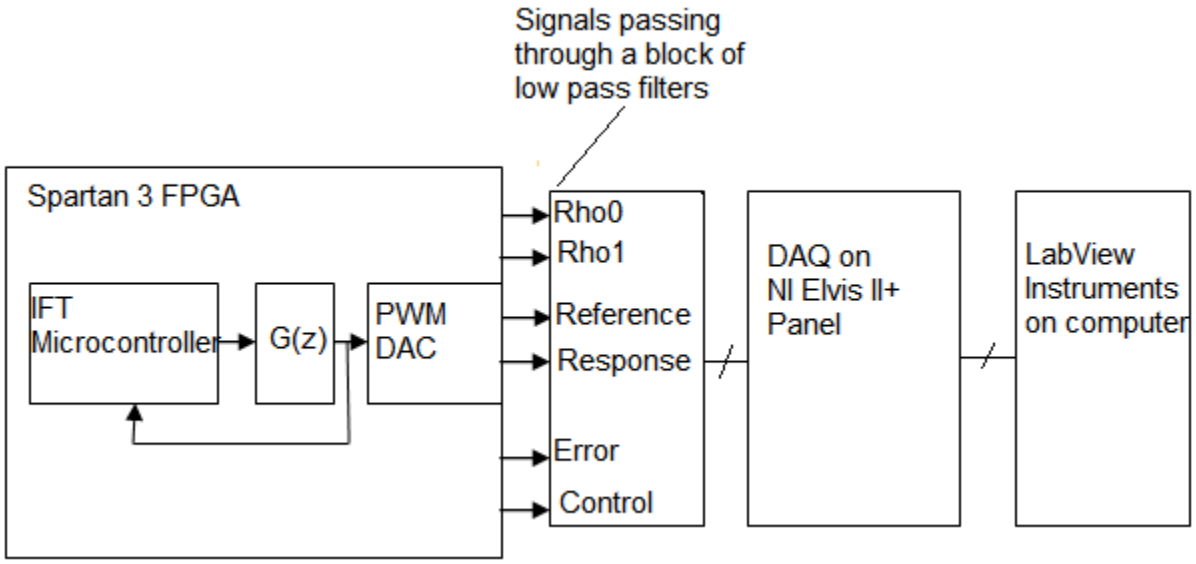

Figure 22: Block diagram of experimental setup for testing a dedicated IFT Microcontroller

The rate of acquisition follows basic rules of sampling theory. The low pass filter is basically a resistor and a capacitor connected as shown in figure 23. The six signals from the Microcontroller modulates the duty cycle of a PWM signal of a respective DAC that generates the signals through the filters, for conversion into analogue voltages via DAQ channels all the way to the LabView display. Hence the DAC output voltage is related to the duty cycle of the PWM signal as given in equation (41).

$$\text{DAC voltage} = amplitude * dutycycle \ \text{[V]} \qquad (41)$$

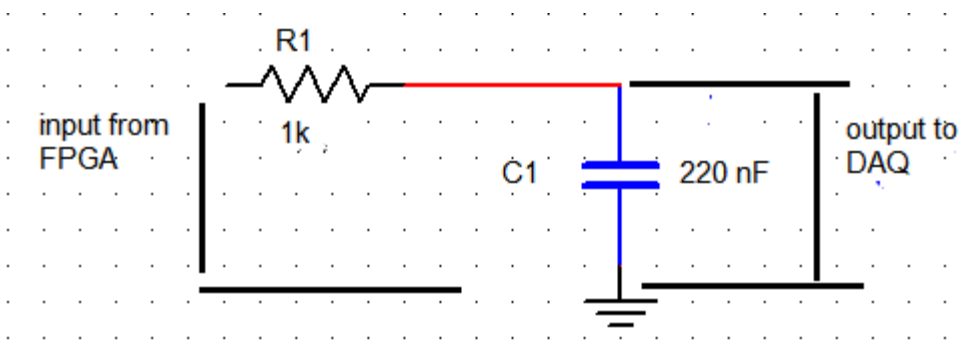

Figure 23: Low pass filter

Duty cycle is obtained by the expression given as:

Duty cycle $= \frac{T_{on}}{T_{on} + T_{off}}$, and the resolution of the DAC is given as

Resolution $= 2^n$

T is the time and n the number of bits, which in the case of the IFT Microcontroller, is 12-bit resolution. From the resolution, the PWM frequency is determined using the expression in equation (42).

$$PWMfreq = \frac{FPGAclockfrequecy}{2^n} = \frac{50*10^6}{4096} = 1.2MHz \quad (42)$$

Then the cut-off frequency of the filter is given by the expression in equation (43).

$$LPf_c = \frac{PWMfreq}{2^{(1+n)}} = \frac{1}{2\pi * R * C} \quad (43)$$

This works out as follows:

$R*C = \frac{2^n}{\pi * PWMfreq} = \frac{4096}{3.142*1.2MHz} = 0.0011$ seconds, yielding a cut-off frequency of 909.0909 Hz. Equation (42) demonstrates decreased PWM frequency at higher resolution systems, however, the problem of high ripple at low frequencies diminishes DAC performance. At high frequency, the generated PWM signals are free from noise, but tend to lower the resolution of the DAC. These two constraints limit the DAC performance, which is well suited for FPGA applications because of its compactness.

From the filters, the signals link to physical pins of the analogue ports of the data acquisition system, via the USB cable, into the computer, to the DAQ assistant, which acts as an interface between the hardware and LabView. The DAQ assistant then distributes the signals to destination graphs. The block diagram of the LabView code is illustrated in figure 24.

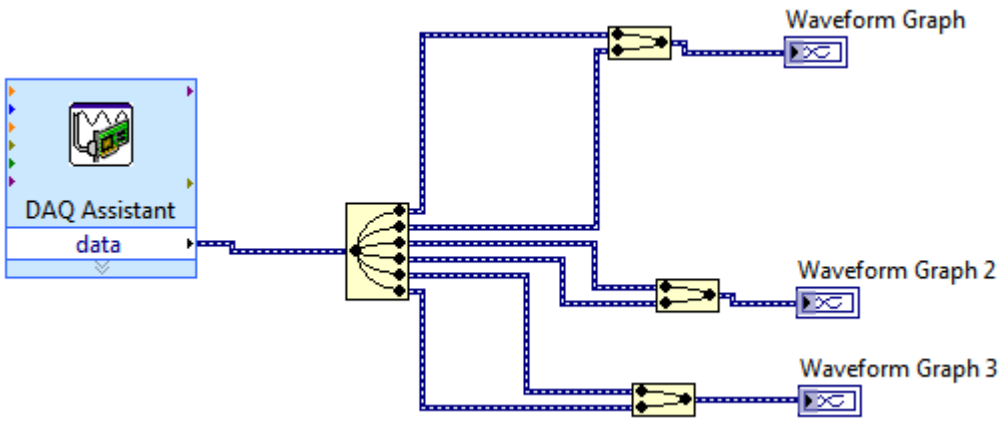

Figure 24: LabView code for displaying signals on graphs

In some cases, a software-based oscilloscope (in LabView) was used specifically to measure minute signals, as will be shown later.

5.2.1 Procedure of the experiment

Having described the experimental setup, the procedure of the experiment is now described. Although the IFT technique assumes no knowledge of the plant, or only partial knowledge thereof, this experiment used a known model of the DC motor (in software format) from chapter three for the purpose of validation and testing (the Dedicated IFT Microcontroller) only.

The experiment was conducted as follows: first, the PI controller was tested alone without the IFT to ensure its proper operation. Thereafter, samples of parameter points derived in chapter three were assembled as a terrain for the IFT to operate. Table 19 shows a record of the samples. The parameters are chosen with the premise that the zero should be inside the unit circle to establish stable terrain for the IFT to operate. Note that the IFT here was used for smoothening only.

Table 19: Parameter pairs in the region of stability

Rho0: ± 0.01, 0.02, 0.03, 0.04, 0.05, ...0.9
Rho1: ± 0.01, 0.02, 0.03, 0.04, 0.05, ...0.9

Second, having defined the stable terrain for the IFT, it is then made to work within the specified region, and in this way the IFT optimises fast. Third, the IFT is implemented using an external button linking to the latch in the Dedicated IFT Microcontroller, which connects to the PI controller. There is also a reset external button for resetting the controller when necessary. When the microcontroller is turned on, it runs the PI controller alone with initial parameters that are randomly set by default, due to the transition period of the microcontroller, which sometimes can exhibit bad responses like the one shown in figure 24a. Figure 24b shows the parameters, and figure 24c shows the error and control signals. To improve the initial response, the controller is tuned by pressing the tuning button, while observing the response trace on the LabView oscilloscope. If the best response was not achieved, the button is touched again. The process of searching can continue until the parameters converge.

Looking again at the response in figure 24a in analytical manner: the output response shown in figure 24a oscillates at a frequency of 150 Hz, and this oscillation could be as a result of positive feedback (at higher values of rho0) caused by the plant lag in the closed-loop system. The oscillation should have been running at the frequency 5 Hz, slightly above the frequency of the DC motor (see chapter three). Such a poor output response is improved by tuning, as already described. The tuning button is pressed to search the best response, as the one shown in figure 25a.

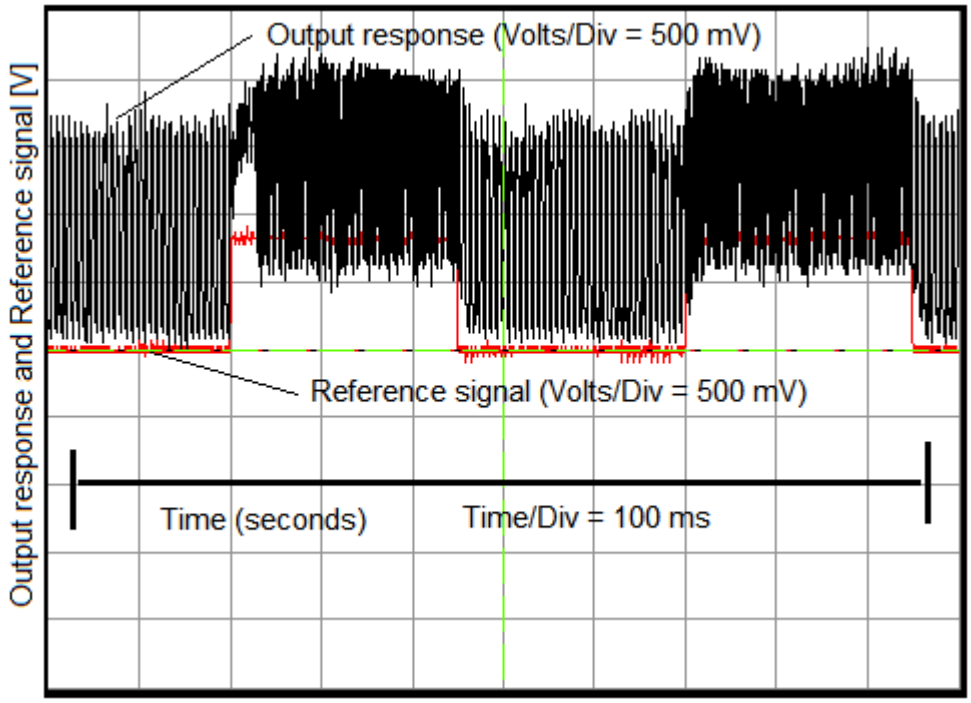

Figure 24(a): Experimental setup for testing of IFT Microcontroller showing the initial response of the controller

The badly tuned controller in this case has average parameters at $rho0 = 0.05$ and $rho1 = 0.05$. Checking the roots of the closed-loop system using equation (32) and the current parameters, yields poles $z = 0.9952$ and $z = 0.9001$ close to the unit circle boundary, this is sensitive to limit cycles. Hence, the oscillations demonstrated in figure 24a result from the oscillatory poles. Also trying maximum parameters at $rho0 = 0.2$ and $rho0 = 0.15$ yields poles $z = 0.9861$ and $z = 0.8948$, and minimum parameters at $rho0 = -0.2$ and $rho0 = -0.15$, yield poles $z = 1.0152$ and $z = 0.9039$, resulting in an unstable system, illustrating a badly tuned controller. However, the IFT was able to tune it to a better response, as shown in figure 25a.

The error and control signals are illustrated in figure 24c. Both signals average at 0.05V, pushing the output response up, as can be seen in figure 24a.

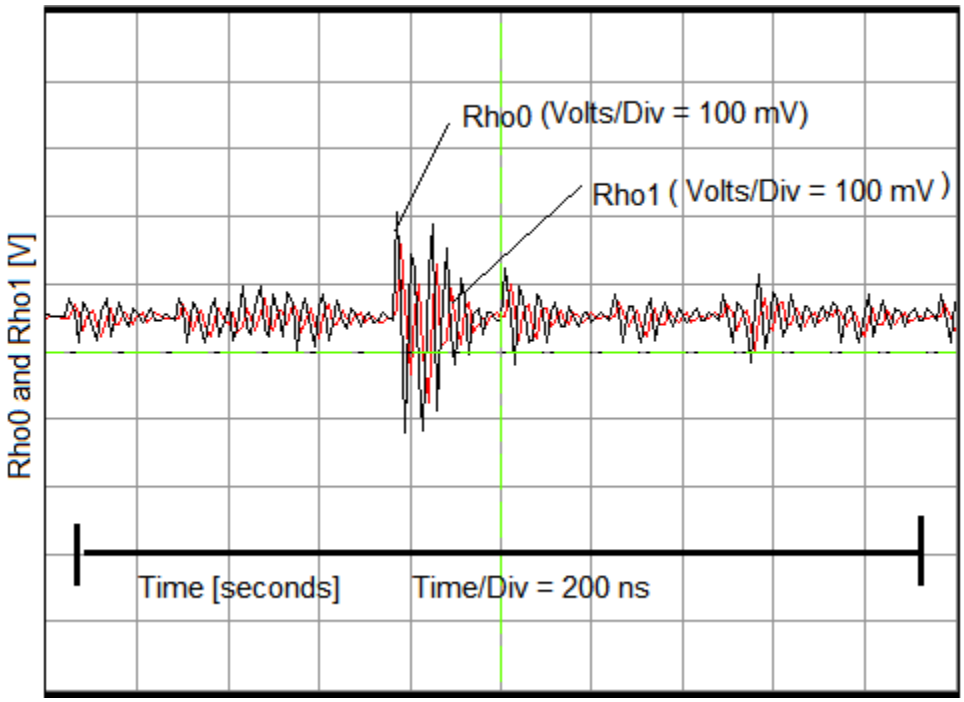

Figure 24(b): Resulting parameters from running the IFT Microcontroller

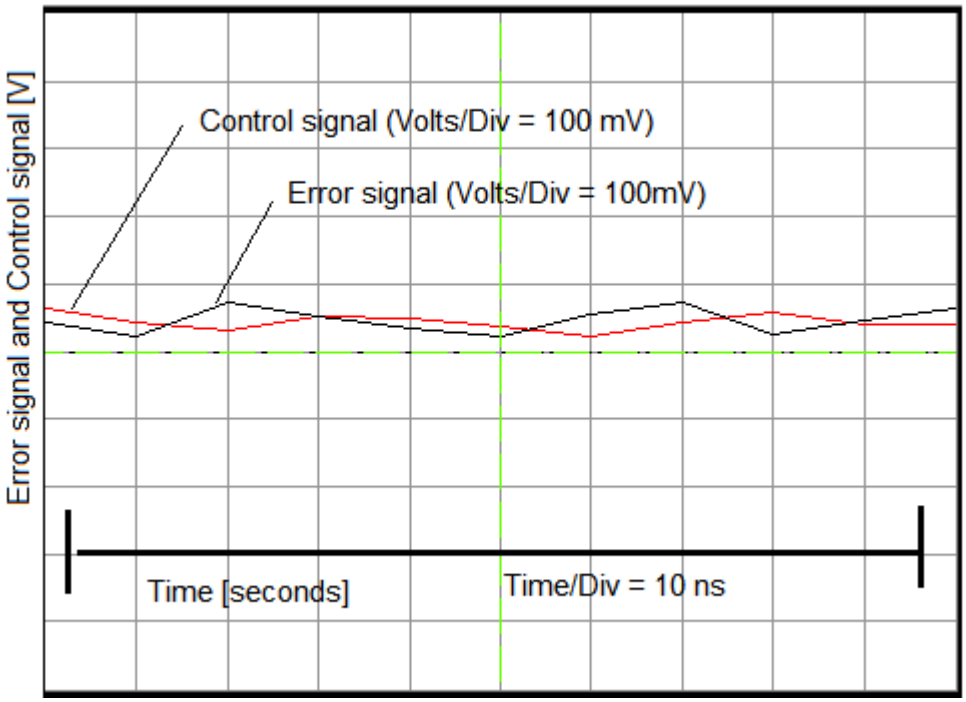

Figure 24(c): Experimental setup for testing of IFT Microcontroller

The results of this experimental setup after convergence are illustrated in figure 25a, verification in figure 25b, followed by the results for controller parameters given in figure 25c and figure 25d, illustrating results for error signal and control signal.

Figure 25a illustrates the best response resulting from controller tuning by the IFT, though, there is an exhibit of noise, which could be as a result of ripple from the PWM DAC at low frequencies. Verification of experimental results for the selected rho-value, from figure 25c, matches those predicted from theory, for example stepping the same system formulated in the Laplace domain as given in equation (44), and is given in figure 25b.

$$y(s) = \frac{1.01*\rho_0 * s + 1.01*\rho_1}{2*s^2 + (1+1.01*\rho_0)*s + 1.01*\rho_1} \text{ [V]} \qquad (44)$$

Both traces indicate a settling time of 20 ms, which is a validation of the developed hardware.

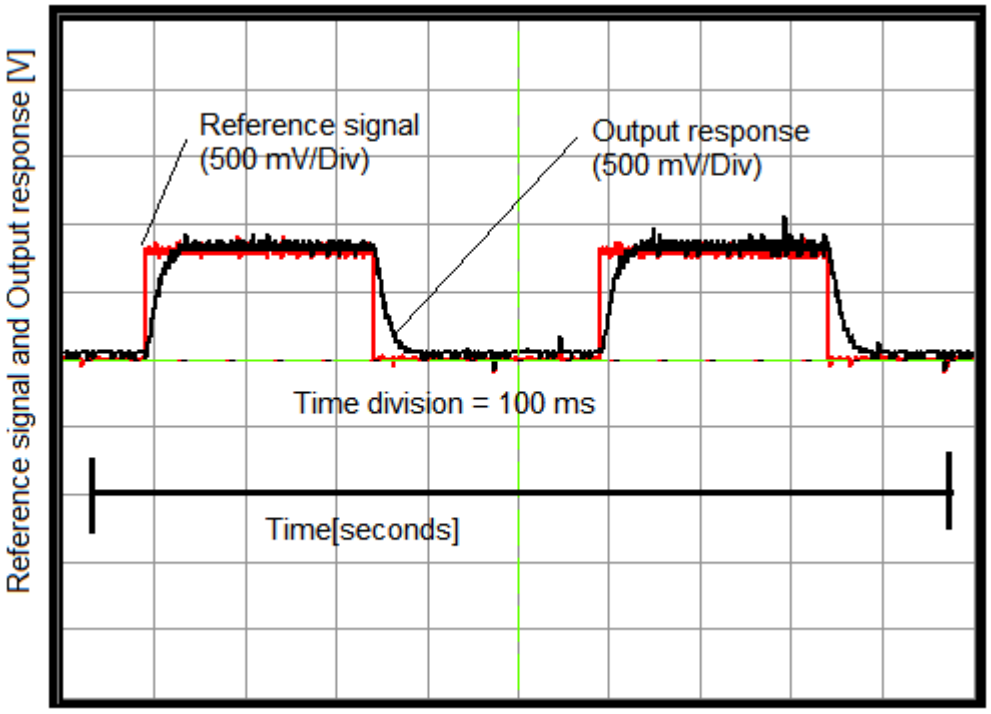

Figure 25(a): Experimental setup for testing of IFT Microcontroller

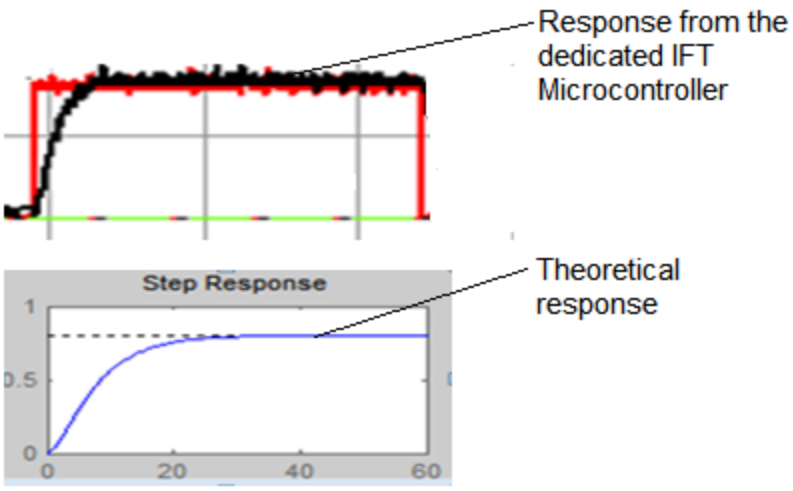

Figure 25(b): Comparison of response from the hardware with the theoretical response

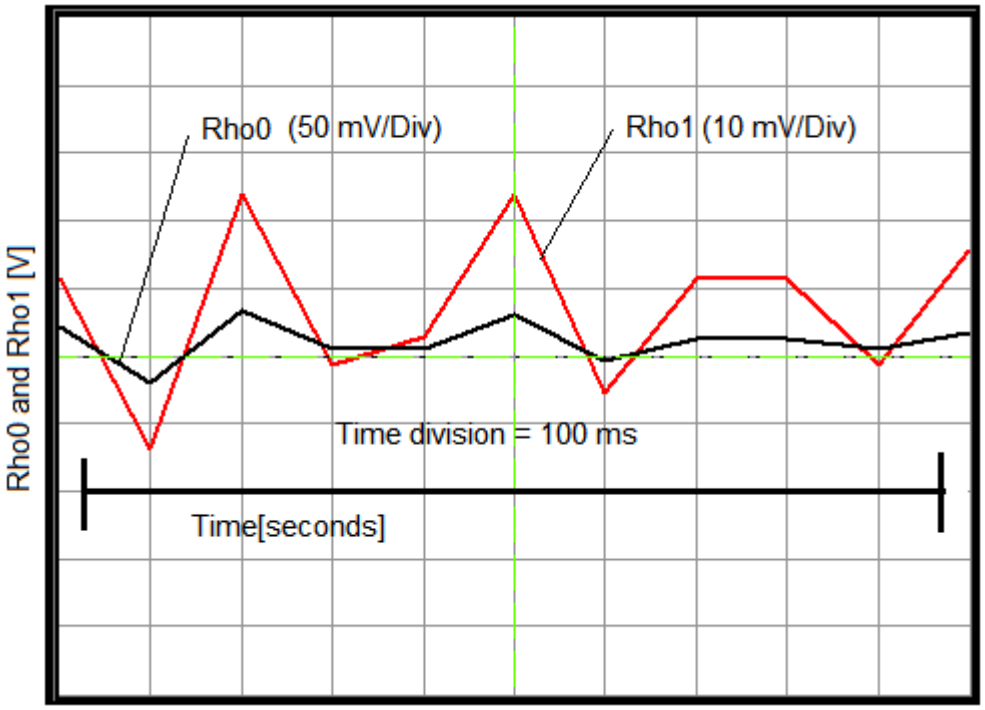

Figure 25(c): Experimental setup for testing of IFT Microcontroller

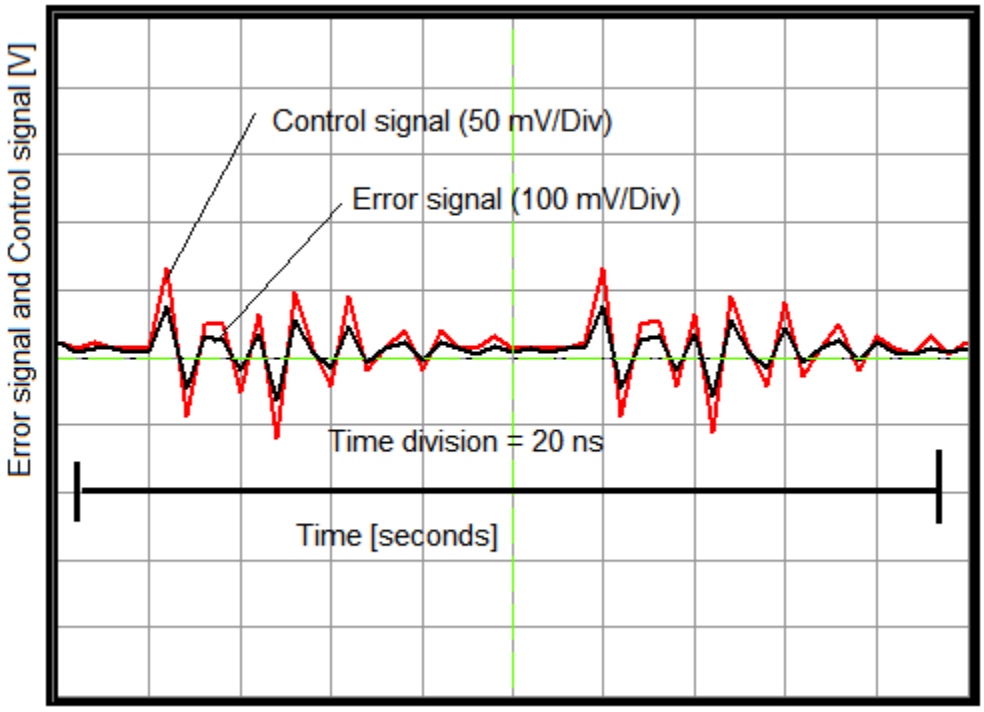

Figure 25(d): Experimental setup for testing of IFT Microcontroller

5.3 CHAPTER SUMMARY

In this chapter, the Dedicated IFT Microcontroller was tested through the use of the method known as simulation in the hardware, where all the components of the Microcontroller (including the plant), are imbedded into the FPGA. Six signals are extracted through the DAC for measurement, and analysis on the LabView display. The captured output response is verified for the given rho-values to match with those predicted from theory. The validation is ascertained, proving the hardware as working.

CHAPTER SIX

CONCLUSION, CONTRIBUTIONS AND FUTURE RESEARCH

6.1 INTRODUCTION

In this chapter, the author gives an outline of the contribution made in the thesis, conclusion and the direction of future research.

6.2 NEW CONTRIBUTIONS

The main contribution of the thesis is the experimental validation of IFT on FPGA hardware given in chapter four and the successful testing of the Dedicated IFT Microcontroller delineated in chapter five. As opposed to the main body of work with IFT where implementations is mainly focused on theory and validated by simulations and experiments where the main computing is run on PCs (desktops, laptops). In this work the possibility of running IFT directly on dedicated IFT Microcontroller built on FPGA technology is proven. This could incentivize the area of dedicated auto tuning commercial applications.

The thesis has also made a contribution in the analysis of 1DOF IFT technique in terms of limitations of applicability for correct implementation, which is the main work of chapter three.

6.3 FUTURE RESEARCH

In view of the design of the Dedicated IFT Microcontroller, the author recommends the following research direction in future:

- Use floating-point number representation to improve the resolution and accuracy in numerical computations by trading off performance in terms of speed and required FPGA resource;

- Use Vivado's High Level Synthesis (HLS) methodology, since its principle is based on traditional programming languages like C++. It is expected that HLS can improve the workability of the hardware as it has the properties of data-based algorithms and has double precision floating point capacity; and
- Implementation of more advanced IFT criterion functions such as 2DOF.

6.4 CONCLUSION

In this thesis a dedicated IFT microcontroller has been developed through the following:

A. exploration of the IFT theory and its applications, followed by the review of literature and a survey of EDA tools.
B. investigation of the IFT by applying it to three different models and each model tested using three types of initial controllers, varlying amplitude of the reference signal and constant amplitude, single-step reference signal.
C. designing the Dedicated IFT Microcontroller by utilisation of hierarchical and modular top down procedure, followed by simulation and validation.

6.5 PUBLICATIONS

From the work carried out in this thesis thus far, two papers have been published to:

- Springer book series of the Automation Control Theory Perspectives in Intelligent Systems, chapter 20, and it is entitled "*FPGA Based Self Tuning PI Controller using IFT Technique*," DIO: 10.1007/978-3-319-33389-2_20,
- IEEE Xplore entitled "*Study of IFT Technique in a View to Create a Novel Hardware*," DOI: 10.1109/CIACT 2017.7977346.
- A journal paper has been written, entitled "*Validation of IFT Technique for Imbedded Applications*," which is in readiness for publication to IEEE Control System Technology.

REFERENCES

[1] G. Himunzowa: Investigations into Implementation of IFT Technique into Microcontroller. Thesis for Degree of Masters Univ Cape Town., no. May, 2008.

[2] S. Jiang, M. H. Smith, J. Kitchen, "Optimisation of PID controller for Engine Electronic Throttle system using IFT technique" SAE International ISSN 0148-7191, 2009.

[3] O. Lequin, M. Gevers, M. Mossberg, E. Bosmans and L. Triest, "Iterative Feedback Tuning of PID parameters: comparison with classical tuning rules", Control Engineering Practice, v11, pp (1023 - 1033), 2003.

[4] W. K. Ho, Y. Hong, A. Hansson, H. Hjalmarson and J. W. Deng, " Relay Auto-tuning of PID controllers using Iterative Feedback Tuning", Automatica, v31, pp (149 - 157), 2003.

[5] Ari. G. Parttanen and Robert R. Bitmead, "The application of an iterative identification and controller design to sugar cane crushing mill", Automatica, v31, pp (1547 - 1563), 1995.

[6] H. Hjalmarsson, “Iterative Feedback Tuning-an Overview,” Adapt. Control Signal Process., vol. 16(5), pp. 373–395, 2002.

[7] O. Arrieta, A. Visioli, R. Vilanova, “Improved PID Autotuning for balanced control operation,” IEEE, 978-1-4244-2728-4/09/2009.

[8] F. De Bruyne and L. C. Kammer, “Iterative feedback tuning with guaranteed stability,” Proc. Am. Control Conf. San Diego, Calif., no. June, pp. 7–11, 1999.

[9] O. L. H. Hjalmarsson, M. Gevers, S. Gunnarsson, “Iterative Feedback Tuning : Theory and Applications,” IEEE Control Systems, Digital Stock1996, pp. 26–41, August 1998.

[10] H. Proch, M. Gevers, B. D. O. Anderson, and C. Ferrera, “Iterative Feedback Tuning for robust controller design and optimization,” Proceedings of the 44th IEEE

Conference on Decision and Control, and the European Control Conference, TuC04.5 pp. 3602–3607, Dec. 12 - 15 2005.

[11] M.B. Radac, “Iterative Techniques for Controller Tuning,” PhD Thesis, Sep. 2011.

[12] E. Monmasson and M. N. Cirstea: "FPGA Design Methodology for Industrial Control Systems- a review," IEEE Trans. Ind. Electron., vol.54. no.4. pp.1824-1842. Aug. 2007.

[13] Y. S. Kung, M. S. Wang, and T. Y. Chuang.: "FPGA-Based Self-Tuning PID Controller using RFB Neural Network and its Application in X-Y Table, IEEE International Symposium on Industrial Electronics. July 5-8, 2009.

[14] S. Ghosh, R. K. Barai, S. Bhattarcharya, P. Bhattacharya, S. Rudra, A. Dutta and R. Pyne, "An FPGA Based Implementation of a Flexible Digital PID Controller For a Motion Control System," International Conference on Computer Communications and Informatics, Jan. 04 - 06, 2013.

[15] J. K. Huusom, P. N. Kjolstad, J. S. Bay “Improving Convergence of IFT,” Journal of Process Control, no. 19(4), 570-578, DOI: 10. 1016/j. jprocont, September, 2008.

[16] A. Bindi “Iterative Feedback Tuning with Application to Robotics,” Masters Thesis, December, 2003.

[17] K. Hamamoto, T. Fukuda, and T. Sugie, "Iterative Feedback Tuning of Controllers for two-mass-spring system with friction," *Control Engineering Practice*, vol.11, pp. 1061-1068, September 2003

[18] I. Urriza, L. A. Barrag'an, J. I. Atigas, D. Navarro and O. Lucia, "FPGA Implementation of a Digital Controller for a dc-dc Converter Using Floating Point Arithmetic," IEEE, 978-1-4244-4649-0/09/ 2009.

[19] Y. F. Chan, M. Moallem, and W. Wang, "Design and Implementation of Modular FPGA-Based PID Controllers, "IEEE Trans. on Industrial Electronics, vol. 5, no. 4, August 2007.

[20] Z. Fang, J. E. Carletta, and R. J. Veillette, "A Methodology for FPGA-Based Control Implementation," IEEE Trans. on Control Systems Technology, vol. 13, no. 6, Nov. 2005.

[21] W. Stefanutti, P. Mattavelli, S. Saggini, and M. Ghioni, "Auto tuning of Digitally Controlled Buck Converters based on Relay Feedback," IEEE, 0-7803-9033-5, May 2005.

[22] W. Weihong, H. Zhongsheng, and J. Shangtai: Overview of the Iterative Feedback Tuning. 2007 Chinese Control Conf., no. 2, pp. 14–18, Jul. 2006.

[23] O. Lequin, H. Hjalmarsson, M. Gevers, S. Gunnarsson, “Iterative Feedback Tuning : Theory and Applications,” IEEE Control Systems, Digital Stock1996, pp. 26–41, August 1998.

[24] R. R. Bitmeadt: Iterative Feedback Tuning via Minimization of the Absolute Error. Proceedings of the 38th IEEE Conference on Decision and Control no. December, 1999.

[25] H. Hjalmarsson, S. Gunnarsson, and M. Gevers: A Convergent Iterative Restricted Complexity Control Design Scheme, Proceedings of the 44th IEEE Conference on Decisionand Control no. Dec, pp. 1735–1740, 1994.

[26] H. Hjalmarsson, “Iterative Feedback Tuning-an Overview,” Adapt. Control Signal Process., vol. 16(5), pp. 373–395, 2002.

[27] W. Carter, K. Duong, R. H. Freeman, H. Hsieh, J. Y. Ja, J. E. Mahoney, L. T. Ngo, and S. L. Sze, "A User Programmable Reconfiguration Gate Array," in proc. of the IEEE Custom Integrated Circuits Conference, pp. 233 - 235, May 1986.

[28] U. Farooq, Z. Marrakchi, and H. Mehrez, "Tree-based Heterogeneous FPGA Architectures," Springer Book Series, www.springer.com/978-1-4614-3593-8, chap. 2, 2012.

[29] E. Monmasson, L. Idkhajine, and M. W. Naouar, "FPGA-Based Controllers," IEEE Industrial Electronics Magazine, Digital Object Identifier 10.1109/ year 2011.

[30] Xilinx, "Xilinx Multi-Node Product Portfolio ," https://www.xilinx.com/products/silicon- devices/fpga.html

[31] (2009). Xilinx Data Book, [Online]. Available: www.xilinx.com

[32] (2009). Altera Data Book, [Online]. Available: www.altera.com

[33] (2009). Actel Data Book, [Online]. Available: www.actel.com

[34] P. J. Ashenden, "The design's Guide to VHDL," Sao Mateo, CA: Morgan Kaufmann, 1995

[35] P. P. Chu, " FPGA Prototyping By VHDL Examples-Xilinx Spartan-3 version," Wiley- interscience, 2008

[36] S. Qin, M. Berekovic, "A Comparison of High-Level Design Tools for SoC-FPGA on Disparity Map calculation Example," 2nd International Workshop of FPGAs for Software Programmers, London, UK, Sep. 1 2015

[37] A. A. Jerraya, H. Ding, P. Kission and M. Rahmoumi, "Behavioural synthesis and Component Reuse with VHDL," Norwell, MA: Kluwer

[38] I. Kuon, R. Tessier, and J. Rose, "FPGA Architecture: Survey and Challenges," Foundations and Trends in EDA, vol. 2, no. 2, pp 135-253, 2007.

[39] P. Coussy, M. Meredith, D. D. Gajski, and A. Takach, "An Introduction to High Level Synthesis," Copublished by the IEEE and the IEEE CASS, 0740-7475, sep. 2009.

[40] Martin Braae: Adaptive Control Engineering, University of Cape Town, South Africa, 2004.

APPENDIX I

Here we transform the IFT component from LabView code into the fixed-point number representation as given below:

de_rho0[t-1] = de_rho0[t]
de_rho0[t-1](1,-10) = de_rho0[t](1,-10)
de_rho1[t-1] = de_rho1[t]
de_rho1[t-1](1,-10) = de_rho1[t](1,-10)
r[t] = e[t]
r[t](1,-10) = e[t](1,-10)
e2[t-1] = e2[t]
e2[t-1](1,-10) = e2[t](1,-10)
e2[t] = r[t] - y2[t]
e2[t](2,-10) = r[t](1,-10) - y2[t](1,-10)
u2[t-1] = u2[t]
u2[t] = u2[t-1] + rho0*e2[t] + (T*rho1 - rho0)*e2[t-1]
u2[t](5,-21) = u2[t-1](1,-10) + rho0(1,-10)*e2[t](1,-11) + (T(1,-10)*rho1(1,-10) - rho0(1,-10))*e2[t-1](1,-11)
y2[t] = 0.9048*y2[t] + 0.0962*u2[t]
y2[t](1,-20) = 0.9048*y2[t](1,-10) + 0.0962*u2[t](1,-10)
de2_rho0[t] = (rho0/(rho0 + rho1))*de2_rho0[t-1] + (1/(rho0 + rho1))*(e2[t]-e2[t-1])
de2_rho0[t](14,-23) = (rho0/(rho0 + rho1))*de2_rho0[t-1] + (1/(rho0 + rho1))*(e2[t]-e2[t-1])
de2_rho1[t] = (rho0/(rho0 + rho1))*de2_rho1[t-1] + (1/(rho0 + rho1))*(e2[t]-e2[t-1])
de2_rho1[t](14,-23) = (rho0/(rho0 + rho1))*de2_rho1[t-1] + (1/(rho0 + rho1))*(e2[t]-e2[t-1])

dJ_rho0[t-1] = dJ_rho0[t]
dJ_rho1[t-1] = dJ_rho1[t]
dJ_rho0[t] = dJ_rho0[t-1] + e2[t]*de2_rho0[t]
dJ_rho0[t](3,-21) = dJ_rho0[t-1] + e2[t]*de2_rho0[t]

```
dJ_rho1[t] = dJ_rho1[t-1] + e2[t]*de2_rho1[t]
dJ_rho1[t](3,-21) = dJ_rho1[t-1] + e2[t]*de2_rho1[t]
dJ_rho0y[t] = dJ_rho0[t] - dJ_rho0[t-1]
dJ_rho0y[t](2,-10) = dJ_rho0[t] - dJ_rho0[t-1]
dJ_rho1y[t] = dJ_rho1[t] - dJ_rho1[t-1]
dJ_rho1y[t](2,-10) = dJ_rho1[t] - dJ_rho1[t-1]
rho0[t-1] = rho0[t]
rho1[t-1] = rho1[t]
rho0[t] = rho0[t-1] - gamma*dJ_rho0y
rho0[t](3,-21) = rho0[t-1] - gamma*dJ_rho0y
rho1[t] = rho1[t-1] - gamma*dJ_rho1y
rho1[t](3,-21) = rho1[t-1] - gamma*dJ_rho1y
end.
```

APPENDIX II

HDL design methodology

A famous design HDL methodology is based on the hierarchical and modular approach defined at different levels of abstraction using design "top-down methodology" [37]. This hierarchical flow of the top-down design method is shown in figure APII.1.

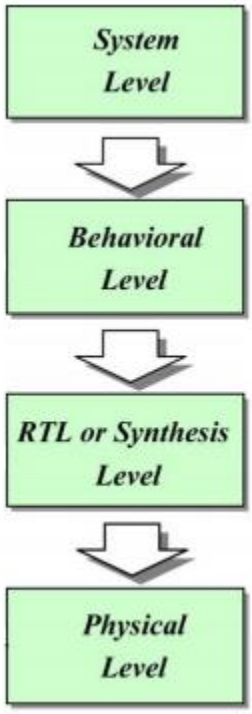

Figure APII.1: Top-down design flow

and its corresponding design flow are presented as follows:

- System level: this is where specifications of the circuit are given, in other words components of the system are delineated. For example, a computer system will comprise the CPU, memory, input/output, and display devices.
- Behavioural level: this is algorithmic description of the system; it is described in terms of hardware description language (HDL).
- Register transfer level (RTL): this is where the circuit is described in terms of its components (how the components are connected); synthesisable HDL is written at this level.

- Physical level: this is where the circuit is physically described by taking into account the target hardware characteristics involved in implementing the hardware description into an FPGA and gives an exact representation of the circuit in terms of desired specifications as given in system level. In order to simulate and validate the digital circuit's functionality, a test bench is written and executed.

HLS design methodology

Unlike hardware description language (HDL), HLS design methodology raises the design abstraction level and allows rapid generation of optimised RTL hardware for performance, area and power consumption [39]. The hierarchical flow of design methodology for HLS is shown in figure APII.2,

and its corresponding design flow are presented as follows:

Compilation and modelling: this is where HLS begins, by transforming (compiling the functional specifications) the input description into a formal representation.

Allocation: this is where the type and number of functional hardware (for example adders, multipliers, multiplexers, etc.) needed to satisfy the design constraints are defined; depending on the HLS tool, some components may be added during scheduling and binding tasks; the components are selected from the RTL component library.

Scheduling: this is where all operations required in the specification model must be scheduled into cycles (in other words, for each operation variables must be read from their sources and brought to the functional unit where they are executed and the result brought to its destination, storage or functional unit).

Binding: this is where each variable that carries values across cycles must be bound to a storage unit.

Generation: this is where an RTL model of the synthesised design is generated.

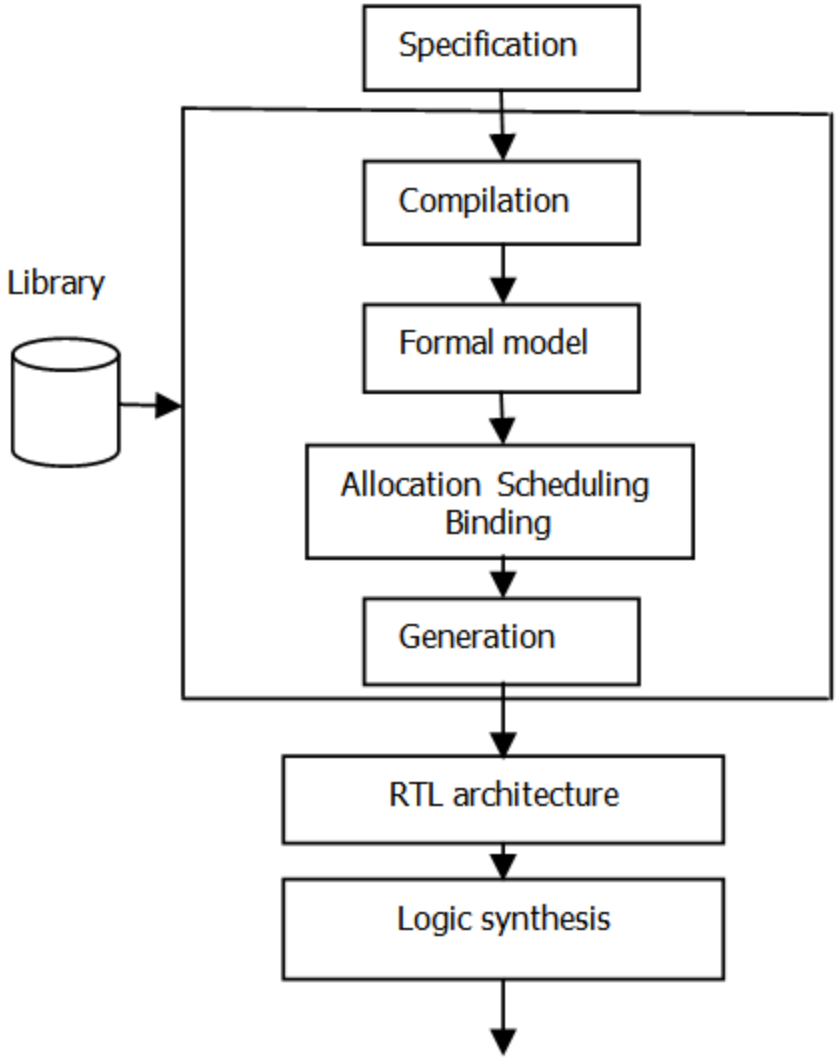

Figure APII.2: HLS design flow

APPENDIX III

Root Locus Results for PI Controller Applied to the DC Motor

In order to test and validate the optimisation action of the IFT, root locus is utilised to investigate the stability region of the PI controller when applied to the DC motor. The transfer function for the closed-loop system is given in (32) and is reproduced here (APIII.1) for convenience.

$$\frac{y(z)}{r(z)} = \frac{0.09516 * \rho_0 * (z + (\frac{\rho_1}{\rho_0} * T - 1))}{z^2 + (0.095 * \rho_0 - 1.90) * z + (0.90 + 0.048 * \rho_1 * T - 0.095 * \rho_0)} \qquad \text{(APIII.1)}$$

The trajectories of characteristic equation roots for a range of zero locations are illustrated in figure APIII.1 to figure APIII.8.

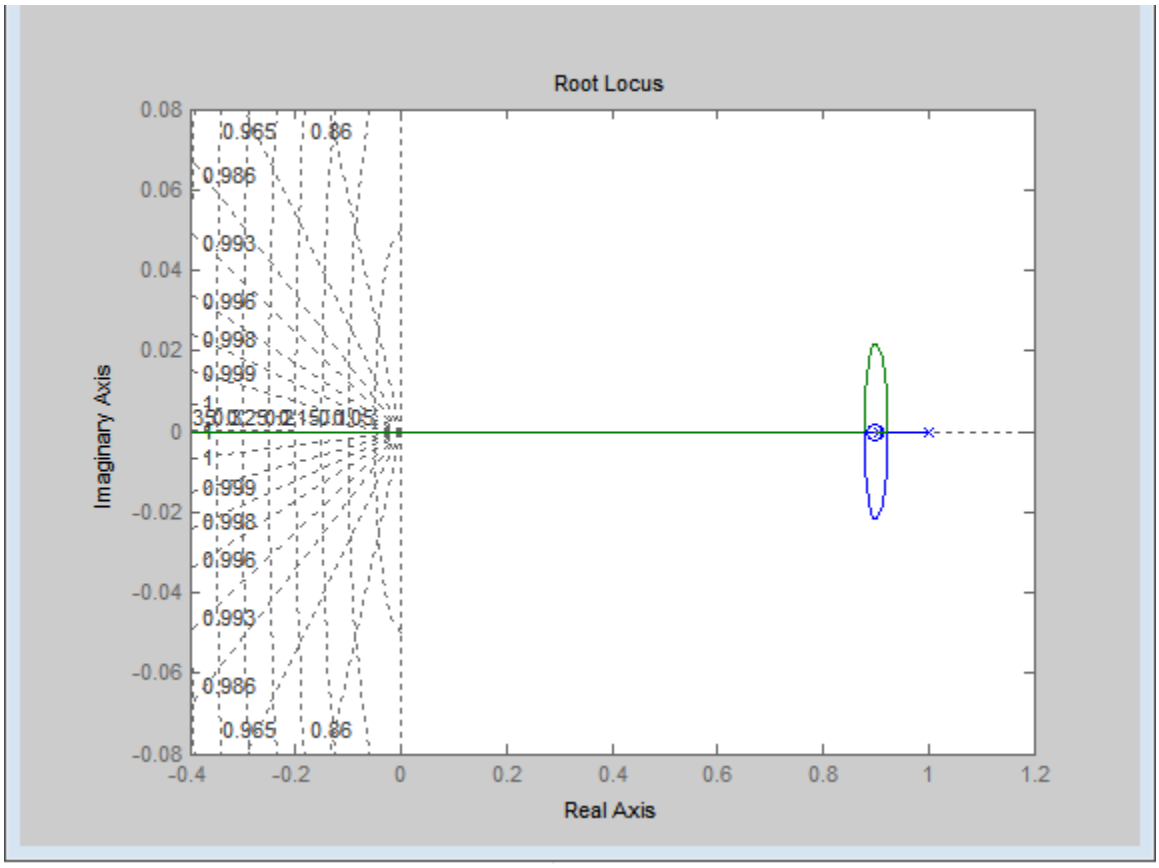

Figure APIII.1. Root locus for PI controller applied to DC motor for zero at 0.9

At location where the zero = 0.9, the roots are inside the unit circle, hence the system is stable.

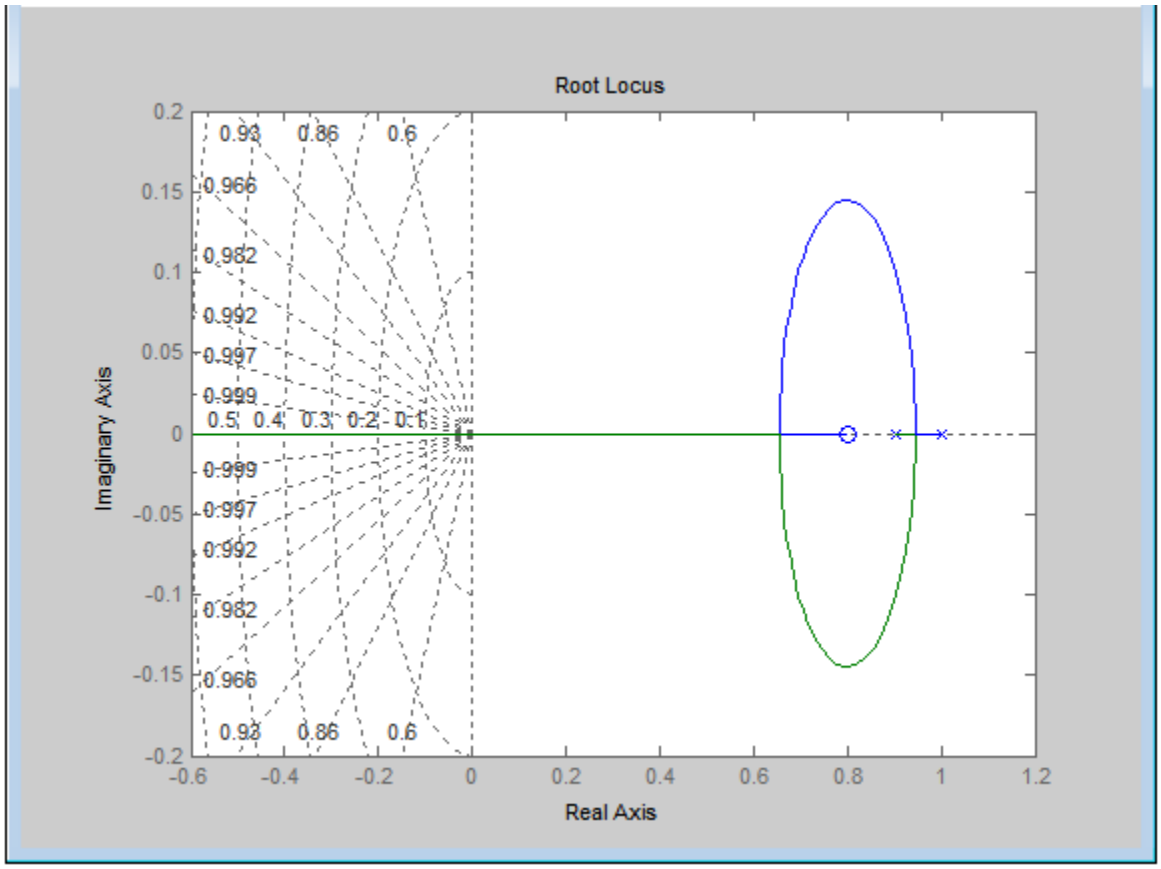

Figure APIII.2. Root locus for PI controller applied to DC motor for zero at 0.8

At location where the zero = 0.8 all the roots are inside the unit circle with larger loci than the loci at zero location of 0.9. This is still a stable system.

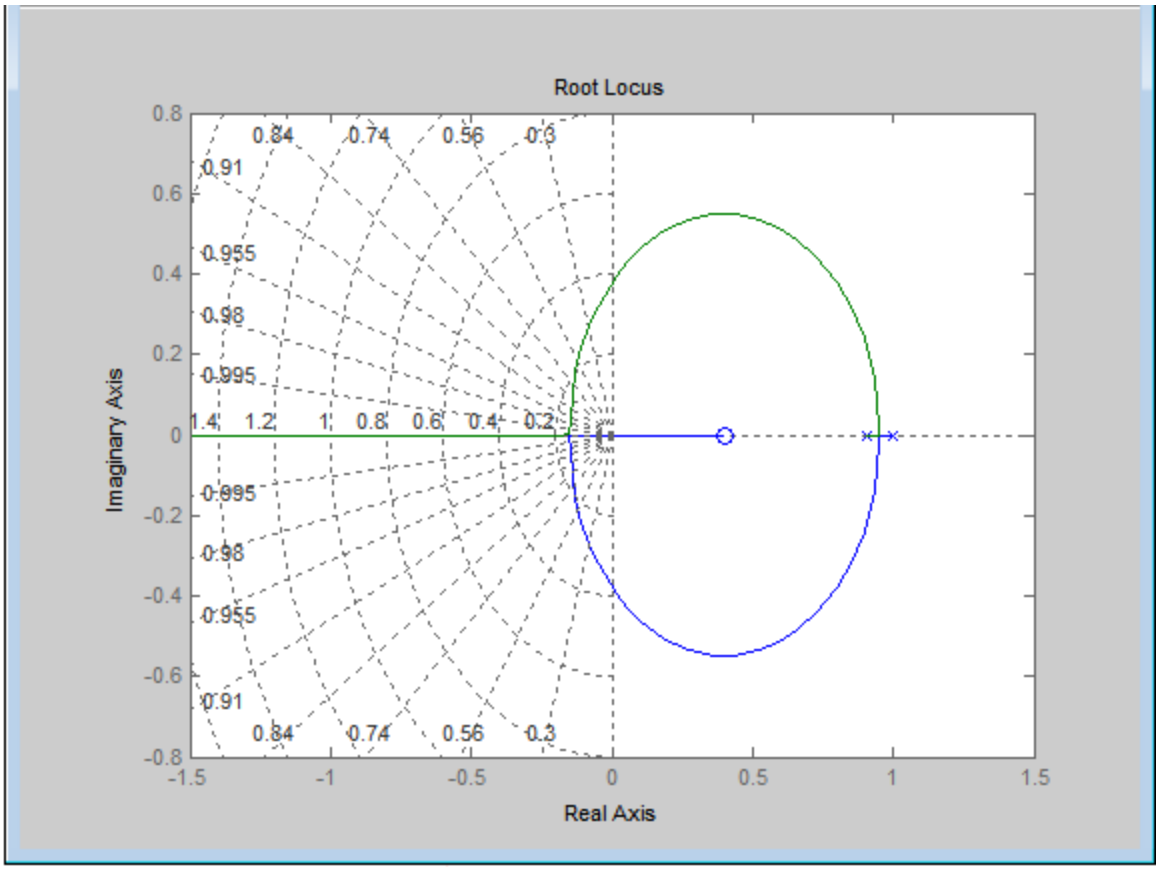

Figure APIII.3. Root locus for PI controller applied to DC motor for zero at 0.4

At location where the zero = 0.4 all the roots are inside the unit circle having loci larger than at zero location of 0.8.

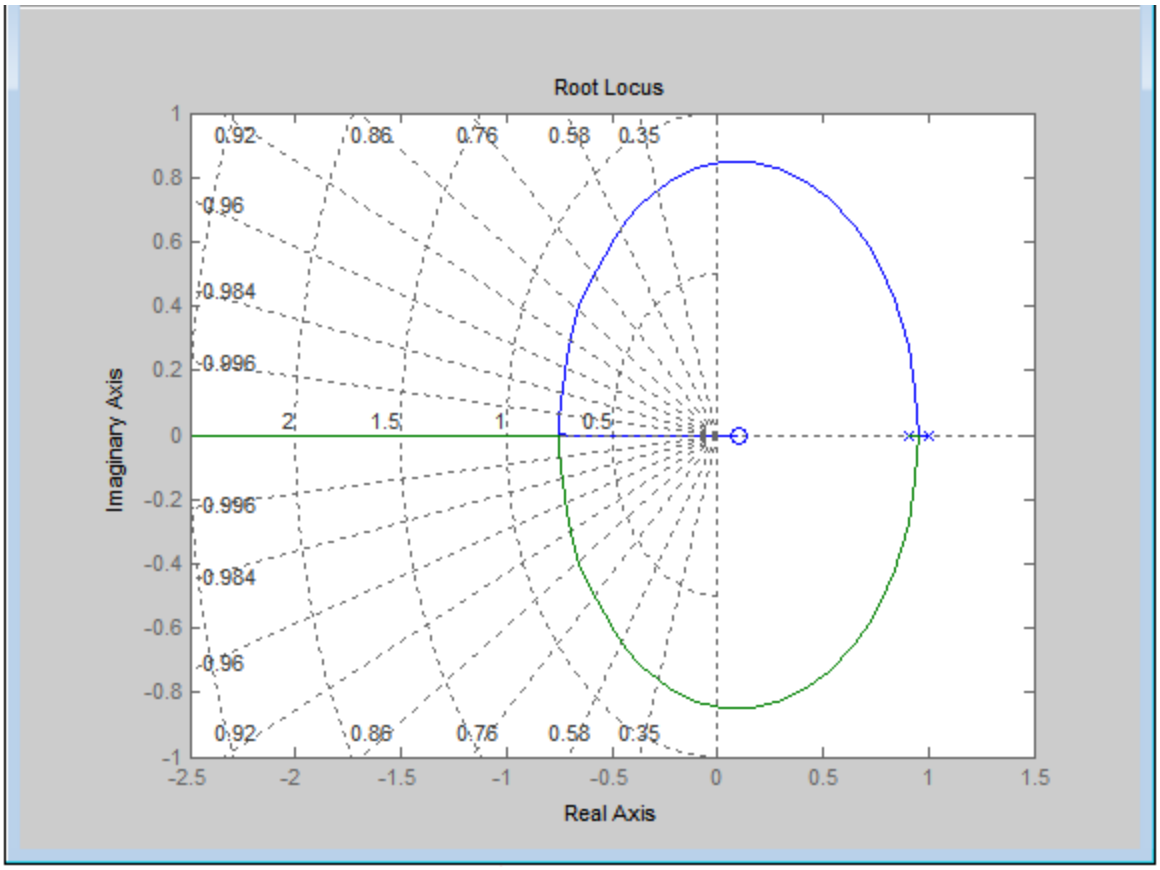

Figure APIII.4. Root locus for PI controller applied to DC motor for zero at 0.1

At location where the zero = 0.1 all the roots are inside the unit circle having loci larger than at zero location of 0.4.

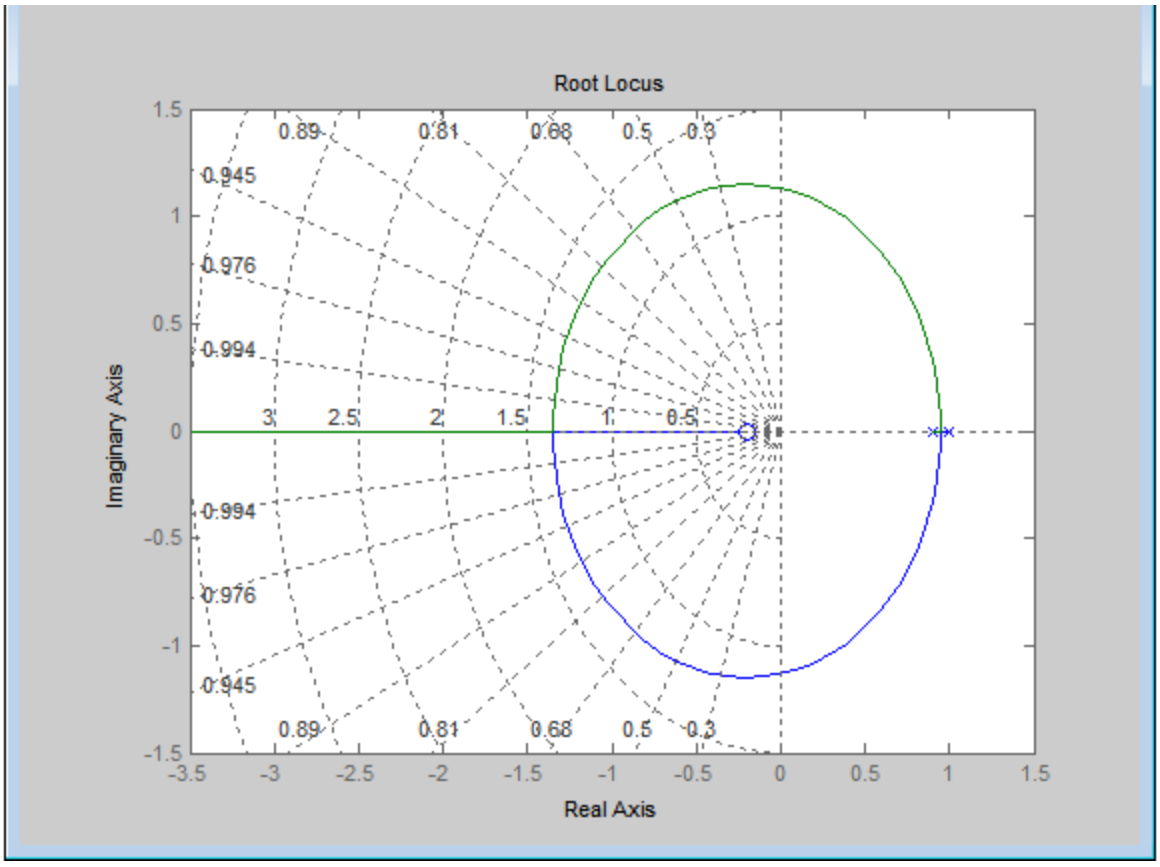

Figure APIII.5. Root locus for PI controller applied to DC motor for zero at -0.2

At location where the zero = -0.2 the loci begins to encircle the unit circle meaning that part of the loci gets into unstable region.

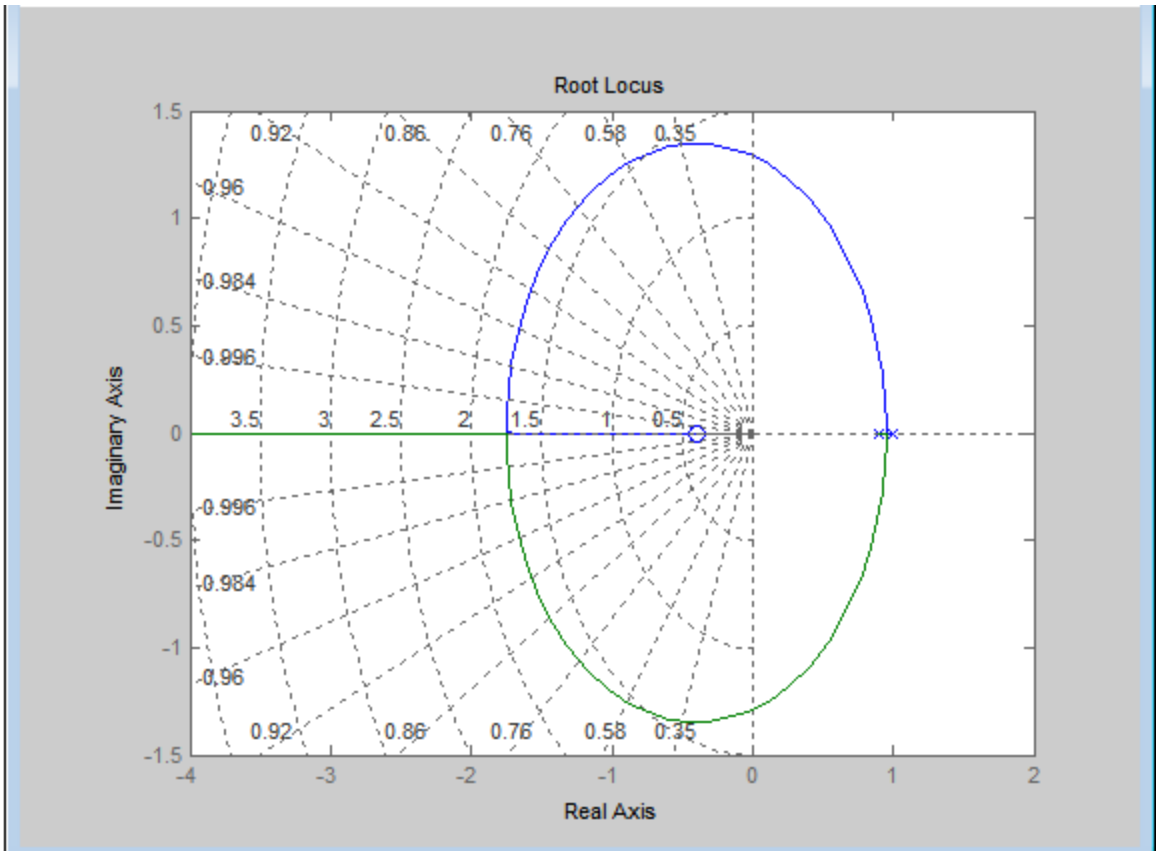

Figure APIII.6. Root locus for PI controller applied to DC motor for zero at -0.4

At location where the zero = -0.4 the part of the loci that encircles the unit circle enlarges.

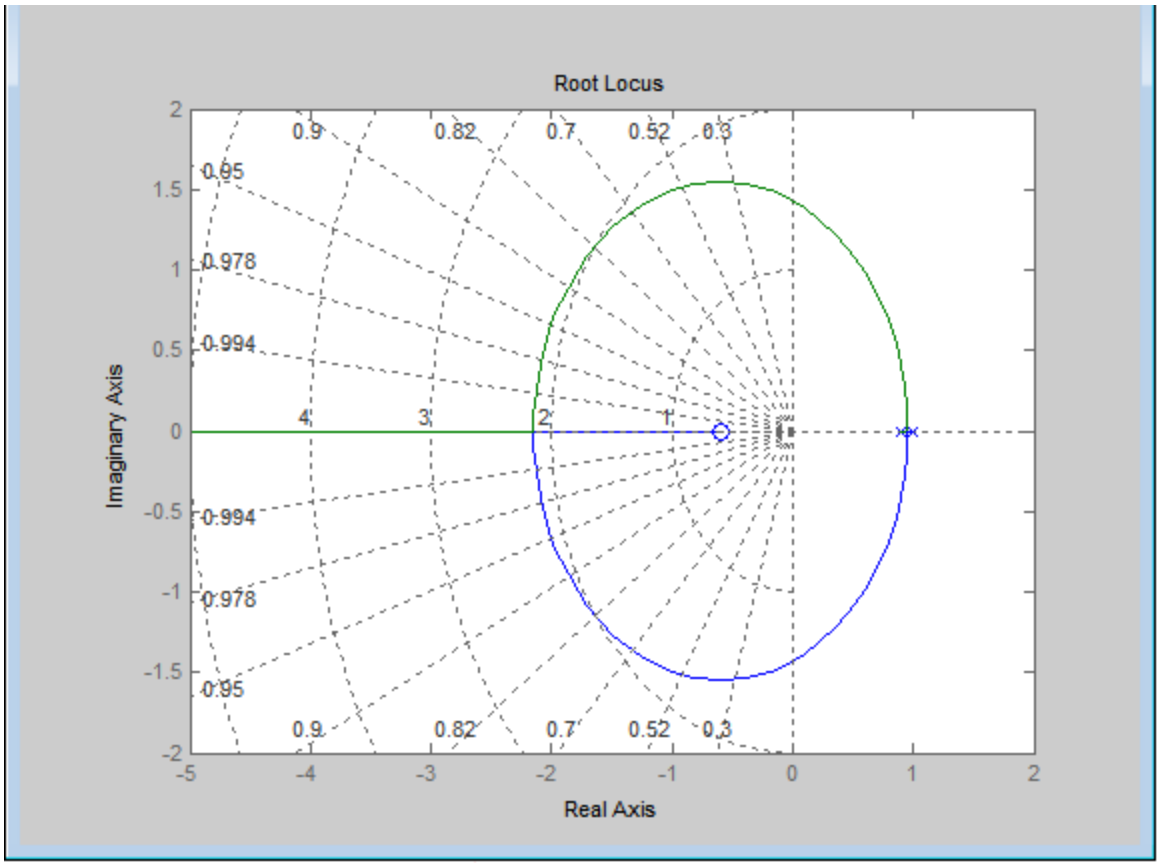

Figure APIII.7. Root locus for PI controller applied to DC motor for zero at -0.6

At location where the zero = -0.6 the part of the loci that encircles the unit circle widens further.

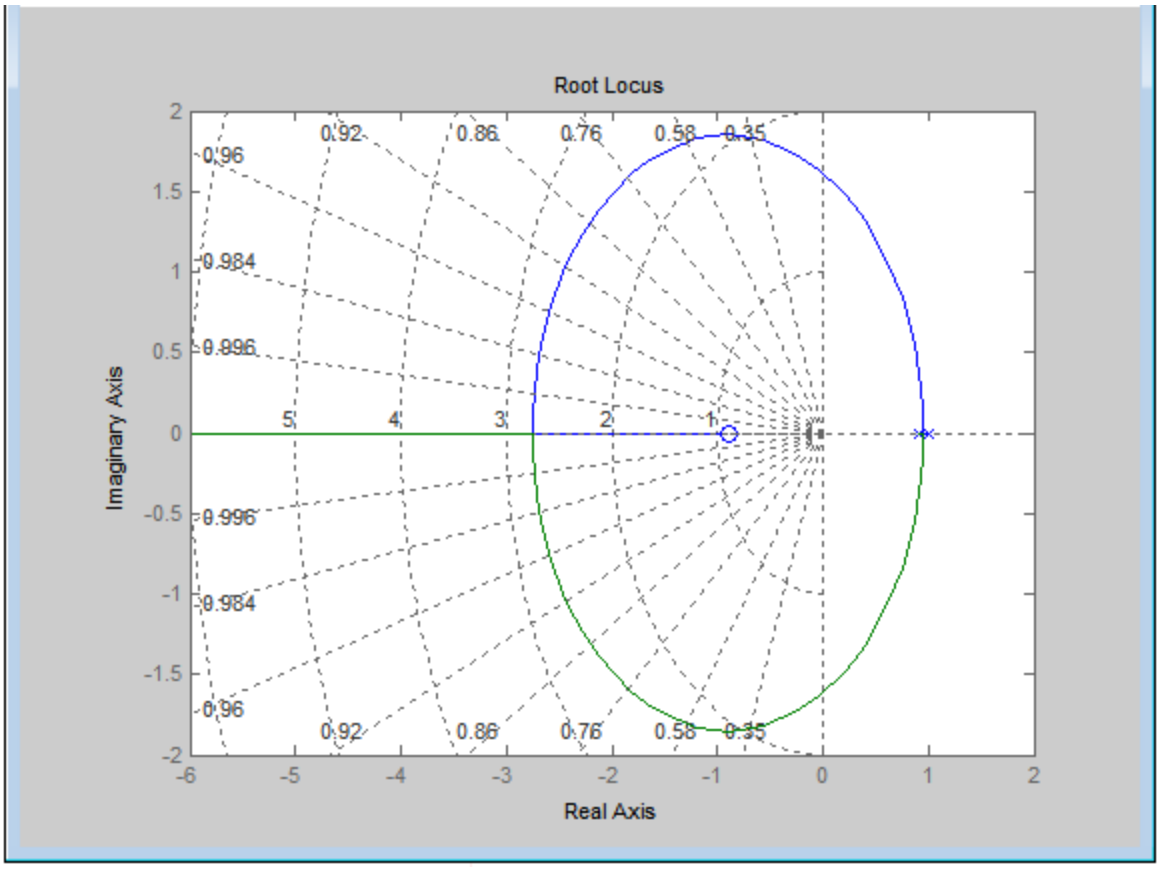

Figure APIII.8. Root locus for PI controller applied to DC motor for zero at -0.9

At location where the zero = -0.9 the part of the loci that encircle the unit circle widen further.

Summary

From the results of the captured root locus traces, the closed-loop poles for the system with zeros from $z = -0.2$ to $z = 0.9$ are stable, but for the system with zeros from $z = -0.9$ to $z = -0.4$ have part of their loci in unstable region. Hence, it is prudent to use the system with zeros from $z = -0.2$ to $z = 0.9$ which have the loci completely inside the unit circle.

APPENDIX IV

In this appendix, we demonstrate five different output responses, obtained from running the PI controller applied to the DC motor in a closed-loop (without the IFT). The controller utilises a range of controller parameters so that we later use them as initial parameters for running the PI controller with the IFT 'on' in order to investigate the tuning action of the IFT technique. The controller parameters are chosen by first assuming rho0 and then computing rho1 by using closed-loop system zero from (32), which is reproduced here for convenience.

$$z = 1 - \frac{\rho_1}{\rho_0} * T$$

Where T

is the sampling frequency of the PI controller.

1. Slow-damped PI controller (rho0 = 0.1 and rho1 = 0.2)

Figure APIV.1 illustrates the output response of the closed-loop system for controller parameters rho0 = 0.1 and rho1 = 0.2. The response has not settled in the given step-length of 200 samples or 40 seconds. This indicates that the slow-damped PI controller requires longer step length than that of 200 samples length to enable the output response settle.

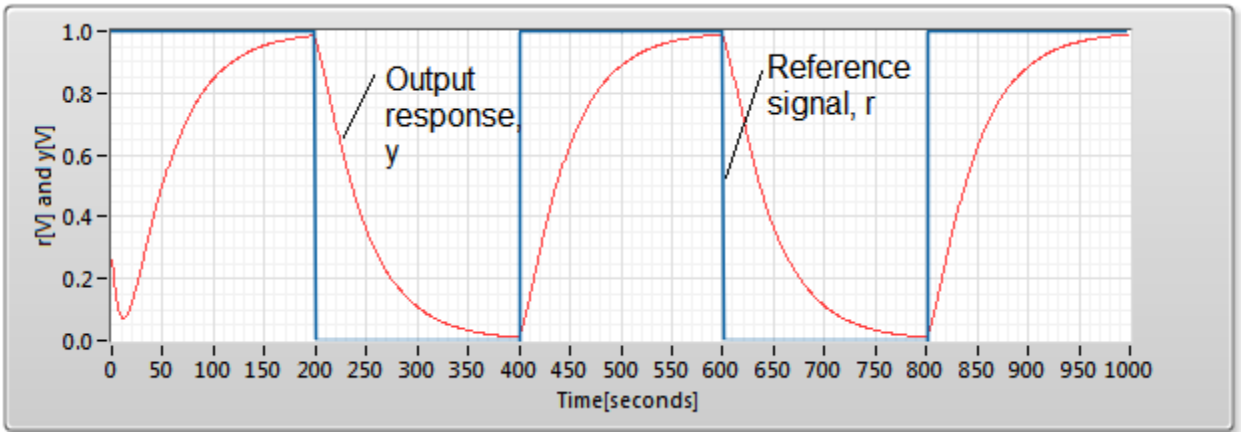

Figure APIV.1 Output response of the DC motor

2. Fast-damped PI controller (rho0 = 1.0 and rho1 = 1.0)

Figure APIV.2 illustrates the output response of the closed-loop system for controller parameters rho0 = 1.0 and rho1 = 1.0. The response settles in 90 samples or 18 seconds of the step length. This demonstration indicates that the fast-damped PI controller does not require longer step-length than the 200 samples length to enable the response settle.

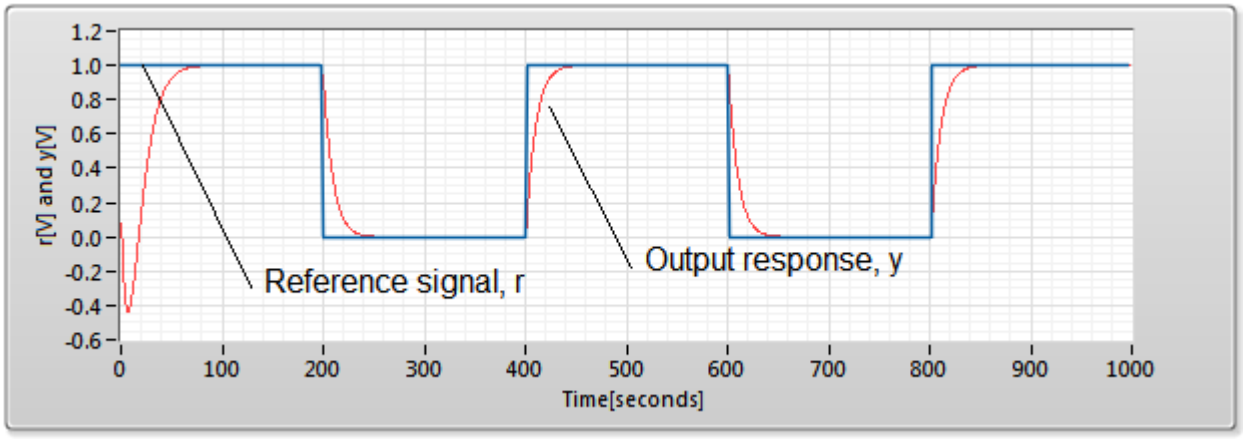

Figure APIV.2 Output response of the DC motor

3. Fast-damped PI controller (rho0 = 1.0 and rho1 = 2.0)

Figure APIV.3 illustrates the output response of the closed-loop system for controller parameters rho0 = 1.0 and rho1 = 2.0. The response settles in 60 samples or 12 seconds of the step length.

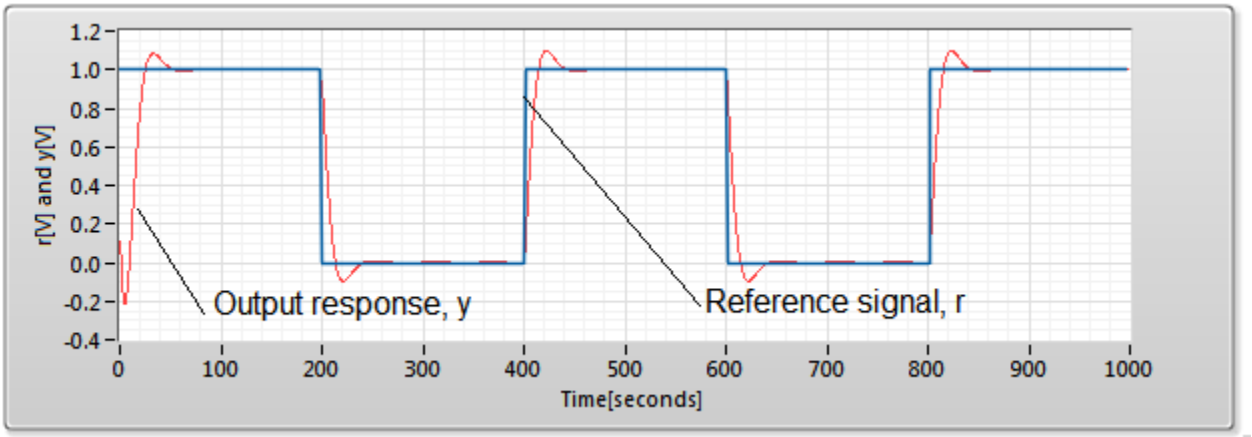

Figure APIV.3 Output response for the DC motor

4. Oscillatory PI controller (rho0 = 1.0 and rho1 = 9.0)

Figure APIV.4 illustrates the output response of the closed-loop system for controller parameters rho0 = 1.0 and rho1 = 9.0. The response settles in 60 samples or 12 seconds of the step length; same as the fast-damped PI controller of rho0 = 1.0 and rho1 = 2.0.

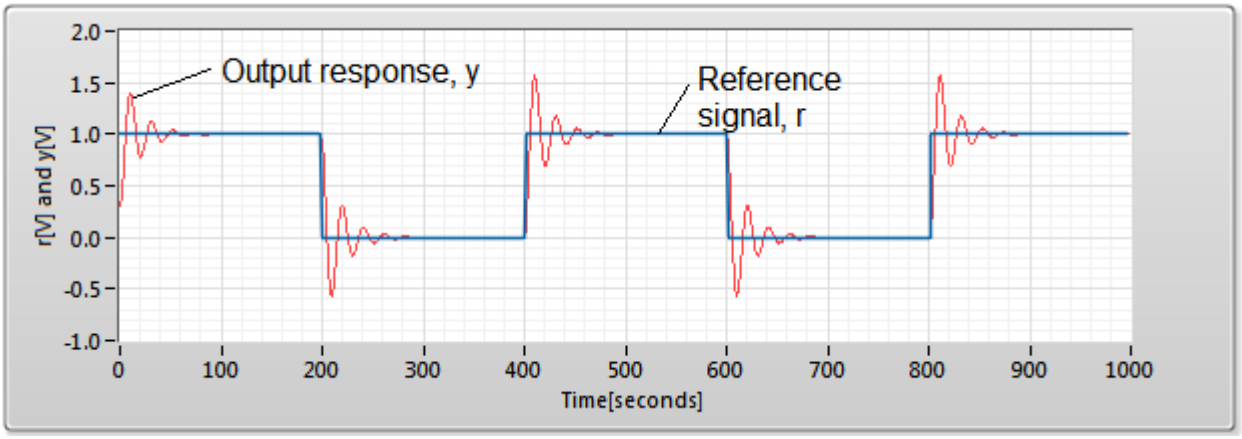

Fig. APIV.4 Output response for the DC

5. Oscillatory PI controller (rho0 = 1.0 and rho1 = 16.0)

Figure APIV.5 illustrates the output response of the closed-loop system for controller parameters (rho0 = 1.0 and rho1 = 16.0). The response settles in 80 samples or 16 seconds of the step length.

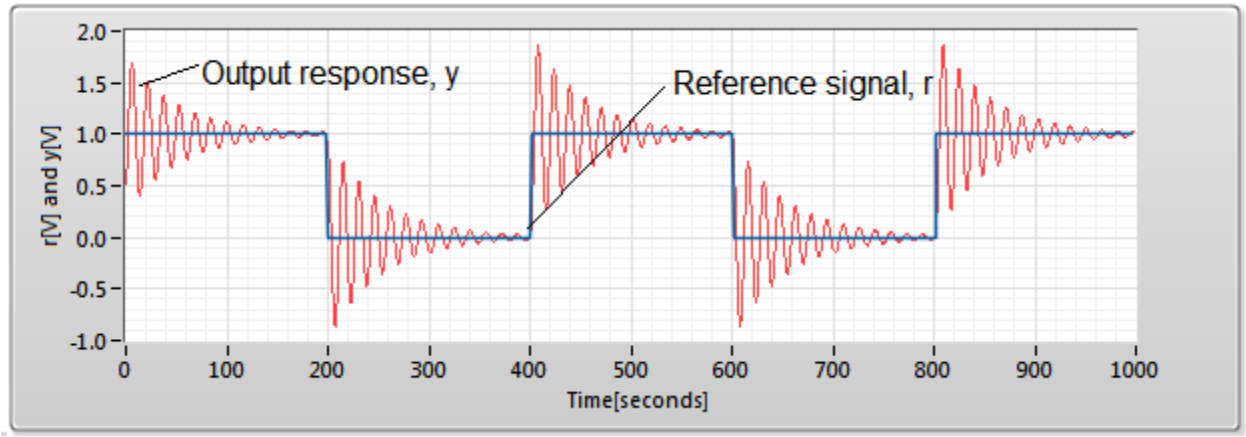

Figure APIV.5 Output response for the DC motor

Three of the above controller parameters are used as initial parameters for the study and testing of IFT technique in section 3.3. These are the slow-damped controller (rho0 = 0.1 and rho1 = 0.1), the fast-damped controller (rho0 = 1.0 and rho1 = 1.0) and the oscillatory controller (rho0 = 1.0 and rho1 = 16)

APPENDIX V

Various derivations for chapter three are done here to avoid choking the flow concept of the thesis. These are presented respective of equation numbers in chapter three as given in the subheadings below.

1. Derivation for (17) and (18) from section 3.2.4

The PI controller without ZOH circuit transfer function is transformed into the digital equation as follows:

$$C(z) = Z(\rho_0 + \frac{\rho_1}{s}) = Z(\rho_0) + Z(\frac{\rho_1}{s}) = \rho_0 * Z(1) + \rho_1 * Z(\frac{1}{s})$$
$$= \rho_0 * 1 + \rho_1 * (\frac{z}{z-1}) = \frac{\rho_0 * (z-1) + \rho_1 * z}{z-1} = \frac{(\rho_0 + \rho_1) * z - \rho_0}{z-1} = \frac{u}{e}$$

Giving the difference equation (AP V.17):

$$z * u\ (z) - u\ (z) = (\rho_0 + \rho_1) * z * e(z) - \rho_0 * e\ (z)$$
$$u_t = u_{t-1} + (\rho_0 + \rho_1) * e_t - \rho_0 * e_{t-1} \qquad \text{(AP V.17)}$$

We now derive the PI controller difference equation with a zero-hold circuit (ZOH) circuit for driving the plant as follows:

$$C(z) = Z(\frac{1-e^{-sT}}{s} * C(s)) = \frac{z-1}{z} * Z\left[\frac{\rho_0}{s} + \frac{\rho_1}{s^2}\right] = (\frac{z-1}{z}) Z\left[\frac{\rho_0}{s}\right] + (\frac{z-1}{z}) Z\left[\frac{\rho_1}{s^2}\right]$$
$$= \frac{z-1}{z} * \frac{\rho_0 * z}{z-1} + \frac{z-1}{z} * \frac{\rho_1 * T * z}{(z-1)^2} = \rho_0 + \frac{\rho_1 * T}{z-1}$$
$$= \frac{\rho_0 * z - \rho_0 + \rho_1 * T}{z-1} = \frac{\rho_0 * z + (\rho_1 * T - \rho_0)}{z-1} = \frac{u}{e}$$

Giving the difference equation:

$$u_t = u_{t-1} + \rho_0 * e_t + (\rho_1 * T - \rho_0) * e_{t-1} \qquad \text{(AP V.18)}$$

2. Derivation for (19) from section 3.2.4

The DC motor s-domain transfer function is converted to z-domain to derive a digital equation as given in (AP V. 19)

$$\frac{y}{u} = Z\left[\frac{1-e^{-sT}}{s} * \frac{0.505}{s+0.5}\right] = Z\left[1-e^{-sT}\right] * Z\left[\frac{0.505}{s*(s+0.5)}\right] = Z\left[1-e^{-sT}\right] * Z\left[\frac{A}{s}+\frac{B}{s+0.5}\right]$$
$$= \left(1-z^{-1}\right) * Z\left[\frac{1.01}{s}-\frac{1.01}{s+0.5}\right] = \left(1-z^{-1}\right) * 1.01 * \left[\frac{z}{z-1}-\frac{z}{z-e^{-aT}}\right] = \left(\frac{z-1}{z}\right) * 1.01 * \left[\frac{z}{z-1}-\frac{z}{z-e^{-aT}}\right]$$
$$= \frac{1.01*(1-e^{-aT})}{z-e^{-aT}} = \frac{0.09516}{z-0.9048057}$$

$$y(t) = 0.904837 * y(t-1) + 0.09516 * u(t) \qquad \text{(AP V.19)}$$

3. Derivation for (32) from section 3.3

Derivation for a closed-loop PI controller applied to the DC motor is illustrated as follows:

$$\frac{y(z)}{r(z)} = \frac{g(z)*k(z)}{1+gh(z)*k(z)}$$

$$= \frac{\left(\frac{\rho_0 * (z+\frac{\rho_1}{\rho_0} * T - 1)}{z-1}\right) * \left(\frac{0.09516}{z-0.904837}\right)}{1+\left(\frac{\rho_0 * (z+\frac{\rho_1}{\rho_0} * T - 1)}{z-1}\right) * \left(\frac{0.09516}{z-0.904837}\right)}$$

$$\frac{y(z)}{r(z)} = \frac{0.09516 * \rho_0 * (z+(\frac{\rho_1}{\rho_0} * T - 1))}{z^2 + (0.095 * \rho_0 - 1.90) * z + (0.90 + 0.095 * \rho_1 * T - 0.095 * \rho_0)} \text{ [V/V]} \qquad \text{(AP V.32)}$$

4. Derivation for (34) from section 3.3.4

We use Matlab command to convert oscillatory plant of (30) from s- domain to z-domain as given below:

>>sampling time = 0.2s;

>>h = tf(10, [1 3 10]);

>>gh = c2d(h, sampling time, 'zoh')

When the code is run, the transfer function (in z-domain with holding circuit) is generated as given in (34) and presented here for ease of reference.

$$gh = \frac{0.1601*z \ +0.1308}{z^2 - 1.252*z + 0.5488} \text{ [V]} \qquad \text{(AP V.34)}$$

Then (34) is converted to a digital equation that is given in (35).

$$gh = \frac{0.1601*z^{-1} + 0.1308*z^{-2}}{1 - 1.252*z^{-1} + 0.5488*z^{-2}} = \frac{y}{u}$$

$$y_t = 1.252*y_{t-1} - 0.5488*y_{t-2} + 0.1601*u_{t-1} + 0.1308*u_{t-2} \text{ [V]} \qquad \text{(AP V.35)}$$

We utilise the PI controller again to control the oscillatory plant discretised above. The open-loop PI controller applied to the oscillatory plant is given as

5. Derivation for (36) from section 3.

Derivation for a closed-loop PI controller applied to the oscillatory plant is done as follows:

$$\frac{y(z)}{r(z)} = \frac{g(z)*k(z)}{1 + gh(z)*k(z)}$$

$$= \frac{\left(\frac{\rho_0 *(z + \frac{\rho_1}{\rho_0} *T - 1)}{z-1} \right) * \left(\frac{0.1601*z \ +0.1308}{z^2 - 1.252*z + 0.5488} \right)}{1 + \left(\frac{\rho_0 *(z + \frac{\rho_1}{\rho_0} *T - 1)}{z-1} \right) * \left(\frac{0.1601*z \ +0.1308}{z^2 - 1.252*z + 0.5488} \right)}$$

$$= \frac{0.16*\rho_0 *z^2 + (0.16*\rho_1 *T - 0.03*\rho_0)*z + 0.13*(\rho_1 *T - \rho_0)}{z^3 - 2.09*z^2 + (1.80 + (0.16*\rho_1 *T - 0.03*\rho_0)*z + 0.13*(\rho_1 *T - \rho_0) - 0.55} \text{ [V/V]} \qquad \text{(AP V.36)}$$

6. Derivation for (37) from section 3.

The Matlab code for converting from s-domain to z-domain.

%start of Matlab code

>>sampling time = 0.2s;

>>h = tf(1, [2 -1]);

>>gh = c2d(h, sampling time, 'zoh')

%end of Matlab code

And after running the code (39) is generated:

$$gh = \frac{0.1052}{z - 1.105)} \text{ [V]} \qquad \text{(AP V.37)}$$

Hence the above equation is converted to a digital equation.

$$gh = \frac{0.1052 * z^{-1}}{1 - 1.105 * z^{-1}} = \frac{y}{u}$$

$$y_t = 1.105 * y_{t-1} + 0.1052 * u_{t-1} \qquad \text{(AP V.38)}$$

7. Derivation for (39) from section 3.

The closed-loop control system for the unstable plant is derived as follows:

$$gh(z) = \frac{g(z) * k(z)}{1 + g(z) * k(z)}$$

$$gh(z) = \frac{(\frac{\rho_0 z + \rho_1 * T - \rho_0}{z - 1}) * (\frac{0.1052}{z - 1.105)})}{1 + (\frac{\rho_0 z + \rho_1 * T - \rho_0}{z - 1}) * (\frac{0.1052}{z - 1.105)})}$$

$$gh(z) = \frac{0.1052 * \rho_0 * z \ + (0.1052 * \rho_1 * T - 0.1052 * \rho_0)}{z^2 + (0.1052 * \rho_0 - 2.105) * z \ + 1.105 + 0.1052 * (\rho_1 * T - \rho_0)} \text{ [V] (AP V.39}$$