

**Applications of Adaptive Controllers Based on  
Nonlinear Parametrization to Level Control of  
Feedwater Heater and Position Control of  
Magnetic Bearing Systems**

by

**Chayakorn Thanomsat**

B.S., Mechanical Engineering

Carnegie Mellon University (1995)

Submitted to the Department of Mechanical Engineering  
in partial fulfillment of the requirements for the degree of


**Master of Science**

at the

**MASSACHUSETTS INSTITUTE OF TECHNOLOGY**

May 1997

© Massachusetts Institute of Technology, 1997. All Rights Reserved.



Author .....

Department of Mechanical Engineering

May 9, 1997

Certified by .....

Anuradha M. Annaswamy

Associate Professor of Mechanical Engineering

Thesis Supervisor

Accepted by .....

Ain A. Sonin

Chairman, Department Graduate Committee

JUL 21 1997

Eng.



# **Applications of Adaptive Controllers Based on Nonlinear Parametrization to Level Control of Feedwater Heater and Position Control of Magnetic Bearing Systems**

by

**Chayakorn Thanomsat**

Submitted to the Department of Mechanical Engineering on  
May 9, 1997, in partial fulfillment of the requirements for  
the degree of Master of Science in Mechanical Engineering

## **Abstract**

In several applications, accurate regulation and control is a difficult task due to inherent nonlinearities as well as parametric uncertainties in the underlying dynamic system. A necessary assumption for most of the parameter estimation schemes available today is that the parameter must occur linearly. However, an increasing demand for accurate models of complex nonlinear systems have prompted researchers and scientists to develop adaptation schemes for parameters which are nonlinear.

In this thesis, a new adaptive control approach based on nonlinear parametrization (Annaswamy *et al.*, 1996) is applied to control problems in two different dynamic systems, (i) level control in feedwater heater system and (ii) position control in magnetic bearing system, both of which contain nonlinear parameters. In the level control problem, a feedwater heater in a 200MW power plant was used as a test platform. The feedwater heater contains several nonlinearities with the dominant ones due to the heater drain valve system as well as the heater cross-sectional area. Dynamic response tests were performed using a full-scale simulation model of the power plant. It is shown that an order of magnitude improvement in settling time and percent overshoot is achievable with the new nonlinear controller. In the position control of magnetic bearing, the performance of an adaptive controller based on nonlinear parametrization is compared to a number of linearly-parametrized controllers. Both the air gap of the bearing and air permeability are uncertain parameters with the former being nonlinear. It is shown through simulations that the nonlinear controller successfully tracks the reference trajectory across the entire operation range while linear controllers perform poorly as they deviate from the nominal operating point.

Thesis Supervisor: Anuradha M. Annaswamy  
Title: Associate Professor of Mechanical Engineering



# Acknowledgments

First, I would like to thank Professor Anuradha M. Annaswamy for her guidance throughout this last two years (1995-1997). Her advice and insights have been invaluable to me and my research at MIT. I am deeply grateful and honored to have crossed path with such a truly inspiring individual. My research experience at MIT would not have been completed without her.

Second, I would like to thank EPRI, my sponsor, for the financial support during my study at MIT. Without your support, it would not be possible for me to be part of the place considered as one of the finest research institutes in the world. Your support is greatly appreciated. I would like to thank Cyrus W. Taft, Chief Engineer at EPRI I&C Center - Tennessee, for his resourcefulness and time he has generously given us throughout the feedwater heater level control research project. This thesis could not have been completed without you. I have great admiration in you both professionally and personally. Thank you.

Third, many people at MIT have contributed to my research work over the years. I thank my best friend, Nick Narisaranukul, for all your supports both at Carnegie-Mellon University and MIT. I will definitely remember endless days/nights we have spent working in the ME computer cluster. I thank all my friends in Adaptive Control Laboratory who have provided me with warm friendships and fruitful discussions. I thank Thai students at MIT and friends in Boston for making my stay such a pleasant one. The two years at MIT was definitely the best years in my life and the memory will always be cherished.

Most importantly, I thank my family, Dad, Mom, my sister-Nui, and my brother-Nick. Your love has always been my inspiration. Dad and Mom, thank for all your endless love and supports you have always had for this son. I love you all. Thank you, Khwan, the love of my life, for your love and encouragement. You have always been the force behind all the things I have accomplished. I love you, always.



# Table of Contents

<b>1</b>	<b>Introduction</b> .....	13
1.1	Motivation and Contribution of Thesis.....	13
1.2	Synopsis of Thesis .....	14
<b>2</b>	<b>Adaptive Controller Based on Nonlinear Parametrization</b> .....	17
2.1	Introduction.....	17
2.2	The Adaptive Controller .....	18
	Problem Statement .....	18
	The Controller .....	18
	Extensions .....	21
<b>3</b>	<b>Level Control in Feedwater Heater System</b> .....	23
3.1	Preliminaries .....	23
	Introduction .....	23
	Theoretical Development .....	26
	Kingston Unit 9 System Description .....	37
	Transient Response Data Acquisition .....	42
	Summary and Remarks .....	51
3.2	Controller Based on Feedback Linearization.....	52
	Introduction .....	52
	Problem Statement .....	52
	The Linear PI Controller .....	54
	Controller Based on Feedback Linearization .....	55
	Simulation Results .....	59
	Summary and Remarks .....	86
3.3	Adaptive Controller Based on Nonlinear Parametrization .....	89
	Introduction .....	89
	Problem Statement .....	89
	Adaptive Controller Based on Nonlinear Parametrization .....	90
	Simulation Results .....	94
	Summary and Remarks .....	102
<b>4</b>	<b>Position Control in Magnetic Bearing System</b> .....	105
4.1	Introduction.....	105
4.2	Dynamic System Modeling.....	106
4.3	The Control Objective - Rotor Position Control.....	107
4.4	Adaptive Controller Based on Nonlinear Parametrization .....	108
4.5	Adaptive Controller Based on Linearized Dynamics .....	116
4.6	Adaptive Controller Based on Linear Parametrization.....	123
4.7	Summary and Remarks .....	129
<b>Appendix A</b> Level Control in Feedwater Heater System .....		131
A.1	Controller Based on Feedback Linearization.....	131
A.2	Adaptive Controller Based on Nonlinear Parametrization .....	149
<b>Appendix B</b> Position Control in Magnetic Bearing System .....		151
B.1	Adaptive Controller Based on Nonlinear Parametrization .....	151
B.2	Adaptive Controller Based on Linearized Dynamics .....	154

B.3 Adaptive Controller Based on Linear Parametrization .....	156
<b>Appendix C Kingston Unit 9 Simulator .....</b>	<b>159</b>
C.1 Controller Based on Feedback Linearization.....	159
<b>Bibliography .....</b>	<b>161</b>



## List of Figures

Figure 3.1: A simple schematic diagram of Rankine cycle .....	24
Figure 3.2: Regenerative cycle .....	25
Figure 3.3: Feedwater heater.....	26
Figure 3.4: The control volume of the feedwater heater.....	26
Figure 3.5: The schematic diagram of the drain flow system.....	33
Figure 3.6: Kingston Unit 9 feedwater heater system schematic diagram .....	39
Figure 3.7: Simulator hardware block diagram .....	41
Figure 3.8: Kingston Unit 9 closed-loop test 1 .....	44
Figure 3.9: Kingston Unit 9 closed-loop test 2 .....	45
Figure 3.10: Kingston Unit 9 closed-loop test 3 .....	46
Figure 3.11: Simulator closed-loop test 1 .....	48
Figure 3.12: Simulator closed-loop test 2 .....	49
Figure 3.13: Simulator closed-loop test 3 .....	50
Figure 3.14: Heater level response on Kingston Unit 9 simulator.....	84
Figure 3.15: Valve command signals.....	85
Figure 3.16: The plot of $f$ as a function of $dP$ where $h = 0.70$ ft. ....	90
Figure 4.1: Rotor position using adaptive controller based on nonlinear parametrization where initial position = 200 microns .....	112
Figure 4.2: Adaptation errors using adaptive controller based on nonlinear parametrization where initial position = 200 microns .....	113
Figure 4.3: Rotor position using adaptive controller based on nonlinear parametrization where initial rotor position = 200 microns and initial reference position = 100 microns	114
Figure 4.4: Adaptation errors using adaptive controller based on nonlinear parametrization where initial rotor position = 200 microns and initial reference position = 100 microns	115
Figure 4.5: Rotor position using adaptive controller based on linearized dynamics with $z_0 = 120$ microns and $bz = 11.5$ .....	119
Figure 4.6: Adaptation parameters using adaptive controller based on linearized dynamics with $z_0 = 120$ microns and $bz = 11.5$ .....	120
Figure 4.7: Rotor position using adaptive controller based on linearized dynamics with $z_0 = 130$ microns and $bz = 11.5$ .....	121
Figure 4.8: Adaptation parameters using adaptive controller based on linearized dynamics with $z_0 = 130$ microns and $bz = 11.5$ .....	122
Figure 4.9: Rotor position using adaptive controller based on linear parametrization: $z_0 = 10$ microns.....	124
Figure 4.10: Rotor velocity using adaptive controller based on linear parametrization: $z_0 = 10$ microns.....	125
Figure 4.11: Adaptation parameters using adaptive controller based on linear parametrization: $z_0 = 10$ microns .....	126
Figure 4.12: Adaptation parameters using adaptive controller based on linear parametrization: $z_0 = 10$ microns .....	127
Figure 4.13: Adaptation parameters using adaptive controller based on linear parametrization: $z_0 = 10$ microns .....	128



## List of Tables

TABLE 1. Values of B for smooth pipes.....	36
TABLE 2. Kingston Unit 9 closed-loop response tests .....	43
TABLE 3. Simulator closed-loop response tests .....	47
TABLE 4. Transient response results from Kingston Unit 9 plant.....	51
TABLE 5. Transient results from Kingston Unit 9 simulator .....	51
TABLE 6. Conventional PI-controller.....	59
TABLE 7. Feedback linearizing P-controller .....	59
TABLE 8. Feedback linearizing PI-controller.....	59
TABLE 9. Conventional PI-controller.....	63
TABLE 10. Feedback linearizing P-controller .....	63
TABLE 11. Feedback linearizing PI-controller.....	63
TABLE 12. Conventional PI-controller.....	67
TABLE 13. Feedback linearizing P-controller .....	67
TABLE 14. Feedback linearizing PI-controller.....	67
TABLE 15. Conventional PI-controller.....	71
TABLE 16. Feedback linearizing P-controller .....	71
TABLE 17. Feedback linearizing PI-controller.....	71
TABLE 18. Conventional PI-controller.....	75
TABLE 19. Feedback linearizing P-controller .....	75
TABLE 20. Feedback linearizing PI-controller.....	75
TABLE 21. Conventional PI-controller.....	79
TABLE 22. Feedback linearizing P-controller .....	79
TABLE 23. Feedback linearizing PI-controller.....	79
TABLE 24. Control parameters and setpoint changes for conventional PI controller	83
TABLE 25. Control parameters and setpoint changes for nonlinear controller .....	83
TABLE 26. Control Parameters.....	94
TABLE 27. Control Parameters.....	95
TABLE 28. Control Parameters.....	96
TABLE 29. Control Parameters.....	97
TABLE 30. Control Parameters.....	98
TABLE 31. Control Parameters.....	99
TABLE 32. Control Parameters.....	100
TABLE 33. Control Parameters.....	101
TABLE 34. Properties of $f_1$ , $f_{2u}$ , and $f_{3u2}$ as a function of $h_i$ .....	108
TABLE 35. Parameters: Adaptive controller based on nonlinear parametrization ...	111
TABLE 36. Selected Parameters: Adaptive controller based on linearized dynamics	117



# Chapter 1

## Introduction

### 1.1 Motivation and Contribution of Thesis

Adaptive control has been in the center of attention of many researchers and scientists in recent years. One of many attractive properties of the adaptive controllers is the ability to conform to changes in the system operating conditions. In the real processes, the actual parameters are rarely known. The characteristics of the process can change with time due to a variety of factors both internal and external. Classical linear control theory sometimes is not adequate for the system to perform satisfactorily over the entire operating range. This gave rise to the adaptive control theory which allows monitoring and adjusting the parameters towards better performance.

A vast majority of adaptive control theory is based on the common assumption that the parametric uncertainty occurs linearly. In some systems, the simplification can be made to meet that criterion. However, there are certain classes of nonlinear systems which do not lend themselves to that assumption. It becomes apparent that a new approach which deals with the nonlinear parametric uncertainty is inevitably required. Among the current approaches includes the nonlinear least-squares algorithm [6] and the extended Kalman filter method [9]. Parameter convergence usually depends on the underlying nonlinearity and the initial estimates. When these algorithms are implemented on the nonlinear systems, the stability property is usually unknown.

Recently, a new approach has been introduced by ([1],[10]). Their algorithm is applicable to dynamic systems where the underlying parameters occur nonlinearly. It has been shown that an adaptive controller based on nonlinear parametrization can be realized

which ensures global stability. In this thesis, this approach will be illustrated through a level control problem in the feedwater heater system and a position control problem in the magnetic bearing system in comparison with other controllers.

The contribution of the thesis consists of the following:

1. Establish the dynamic models for both the feedwater heater and the magnetic bearing systems.
2. Realize the adaptive controller based on nonlinear parametrization which will lead to global stability for the respective systems.
3. Compare the performance of the different controllers on the systems.

## **1.2 Synopsis of Thesis**

This thesis is organized into four chapters as follows:

Chapter 2 provides readers with the background on adaptive controller based on nonlinear parametrization as proposed by [1]. Here, the adaptation algorithms are shown to result in the controller which leads to global stability using Lyapunov stability analysis. However, the reader will not be provided with the detailed proof but are encouraged to seek further information from the reference sources.

A detail case study of level control in feedwater heater system is illustrated through Chapter 3. In this chapter, the dynamic model of the system was derived based on the actual feedwater heater system at Kingston Unit 9, Tennessee. The closed-loop transient tests were also performed at the plant and are presented here. The Kingston Unit 9 simulator system was used as the test bed for the new controller. The advantage of the simulator is that it allows the controller to be tested before the actual implementation. MATLAB is used extensively to simulate the closed-loop system with the derived controllers. The summary and remarks are given at the end of the chapter.

Chapter 4 illustrates another example which the parameter occurs nonlinearly. The objective is to control the position of the rotor in the magnetic bearing system. In comparison, the controllers based on linear control theory are shown to perform poorly when deviated from the operating point.

Finally, the source codes for both the MATLAB programs and the FORTRAN program used on Kingston Unit 9 simulator are included in the appendices.





# Chapter 2

## Adaptive Controller Based on Nonlinear Parametrization

### 2.1 Introduction

Adaptive Control has received considerable interest among the researchers and scientists in recent years. A majority of the adaptive control theories have centered around the assumption that the unknown parameters occur linearly. Even in the adaptive control of nonlinear systems, many estimation algorithms have been restricted to the systems which parameters are linear. In many control problems, it becomes apparent that the nonlinear models which are able to replicate complex dynamics require nonlinear parametrization.

We present here an approach proposed by [1] which we will illustrate through feedwater heater level control and magnetic bearing position control applications that the controller results in better overall performance and stable systems. It is also shown that a stable adaptive algorithm can be developed for a system which has both linear and nonlinear parametrization.

## 2.2 The Adaptive Controller

### 2.2.1 Problem Statement

The dynamic system under consideration is of the form

$$x^{(n)} = f(\phi, \theta) + \phi^T \alpha + u \quad (2.1)$$

where  $\theta \in R$  and  $\alpha \in R^m$  are unknown parameters,  $u$  is a scalar control input,  $\phi(t) \in R^p$  and  $\varphi(t) \in R^m$  are known functions of the system variables, and  $f$  is a scalar function that is non-linear both in  $\phi$  and  $\theta$ . The desired trajectory  $x_m$  is chosen as the output of the model whose dynamics is governed by the differential equation

$$D(s)[x_m] = r \quad (2.2)$$

where  $D(s)$  is a Hurwitz polynomial and  $r$  is a bounded reference input. If the scalar output error is defined as  $e = x - x_m$ , the goal is to choose the control input so that the error  $e$  converges to zero asymptotically while ensuring that all system variables remain bounded.

The following assumptions are made regarding the system:

1. All state variables are accessible for measurement.
2.  $\phi(t)$  and  $\varphi(t)$  are measurable functions, and are bounded functions of the state variables.
3.  $\theta \in [\theta_{min}, \theta_{max}]$ , and  $\theta_{min}$  and  $\theta_{max}$  are known.
4. For all  $\theta$  and any  $\phi(t)$ ,  $f(\phi(t), \theta)$  is either concave, or convex.
5.  $f$  is a known bounded function of its arguments.

### 2.2.2 The Controller

The structure of the dynamic system clearly suggests that when the parameters  $\theta$  and  $\alpha$  are known, a feedback-linearizing controller can be realized which stabilizes the system and ensures output tracking. One choice of such a controller is described below. A composite error  $e_s$ , which is a scalar measure of the state error is defined as

$$e_s = D(s) \left[ \int_0^t e d\tau \right] \quad (2.3)$$

The control input is then chosen as

$$u = -ke_s - D_1(s)[x] + r - \varphi^T \alpha - f(\phi, \theta) \quad (2.4)$$

where  $k > 0$ , and  $D(s) = s^n + D_1(s)$ . This leads to the error equation  $D(s)[e(t)] = -ke_s$  which can be written using Eq. (2.3) as

$$\dot{e}_s = -ke_s \quad (2.5)$$

and hence  $e_s$  is bounded and  $e_s(t) \rightarrow 0$  asymptotically. From Eq. (2.3), it follows that for  $i = 0, \dots, n-1$ ,  $x^{(i)} - x_m^{(m)}$  is bounded and tends to zero asymptotically.

The problem is to find a certainty equivalence controller using (2.3) and adaptive laws for estimating  $\theta$  and  $\alpha$  when the latter are unknown so as to achieve global boundedness and asymptotic tracking. Suppose the following controller and adaptive laws are chosen in the presence of unknown parameters:

$$u = -ke_s - D_1(s)[x] + r - \varphi^T \hat{\alpha} - f(\phi, \hat{\theta}) - u_a(t) \quad (2.6)$$

$$\dot{\hat{\alpha}} = e_s \Gamma_\alpha \varphi \quad (2.7)$$

$$\dot{\hat{\theta}} = e_s \gamma_\theta w \quad (2.8)$$

where  $w(t)$  and  $u_a(t)$  are time-varying signals to be chosen later and  $\Gamma_\alpha$  and  $\gamma_\theta$  are adaptive gains used to alter the transient behavior of the parameter estimates during the adaptation.

If the parameter errors are defined as  $\tilde{\alpha} = \hat{\alpha} - \alpha$ ,  $\tilde{\theta} = \hat{\theta} - \theta$ , the error equation becomes

$$\dot{e}_s = -ke_s - \varphi^T \tilde{\alpha} + f - \hat{f} - u_a(t) \quad (2.9)$$

where  $\hat{f} = f(\phi, \hat{\theta})$ . This suggests the commonly used Lyapunov function candidate

$$V = \frac{1}{2}(e_s^2 + \tilde{\alpha}^T \Gamma_\alpha^{-1} \tilde{\alpha} + \gamma_\theta^{-1} \tilde{\theta}^2) \quad (2.10)$$

with a time derivative

$$\dot{V} = -ke_s^2 + e_s[f - \hat{f} + w\tilde{\theta} - u_a(t)]. \quad (2.11)$$

It is easy to see that when  $f$  is concave, since

$$f - \hat{f} \leq \left( \frac{\partial f}{\partial \theta} \right)_{\hat{\theta}} (\theta - \hat{\theta}) \quad (2.12)$$

for all  $\theta$  (Bertsekas, 1989), if we choose

$$w(t) = \left( \frac{\partial f}{\partial \theta} \right)_{\theta = \hat{\theta}} , u_a(t) \equiv 0 \quad (2.13)$$

It ensures that  $\dot{v} \leq 0$  provides  $e_s \geq 0$ . However, when  $e_s$  is negative, Eq. (2.13) is not adequate for ensuring that  $v$  is a Lyapunov function. Similarly, since the inequality in (2.12) is reversed when  $f$  is convex, it is obvious that when  $e_s$  is positive, our choice of the signals in (2.13) would not suffice for achieving a negative semidefinite  $v$ . This implies that a fairly distinctive strategy needs to be adopted for the case when  $e_s$  is negative (or positive) when  $f$  is concave (or convex) with respect to  $\theta$ .

It will now be shown that whether  $f$  is concave or convex, the following adaptive controller ensures global boundedness and asymptotic tracking.

$$u = -k\varepsilon_s - D_1(s)[x] + r - \varphi^T \hat{\alpha} - f(\phi, \hat{\theta}) - u_a(t) \quad (2.14)$$

$$\dot{\hat{\alpha}} = \varepsilon_s \Gamma_\alpha \varphi \quad (2.15)$$

$$\dot{\hat{\theta}} = \varepsilon_s \gamma_\theta w \quad (2.16)$$

$$\varepsilon_s = e_s - \varepsilon \text{sat}\left(\frac{e_s}{\varepsilon}\right) \quad (2.17)$$

and  $u_a$  and  $w$  are time functions which are chosen as follows:

1. When  $f(\phi, \theta)$  is concave in  $\theta$  for all  $\phi$ ,

$$u_a(t) = 0 \text{ if } e_s \geq 0 \quad (2.18)$$

$$u_a(t) = \text{sat}\left(\frac{e_s}{\varepsilon}\right) \left[ \hat{f} - f_{\min} - \left( \frac{f_{\max} - f_{\min}}{\theta_{\max} - \theta_{\min}} \right) (\hat{\theta} - \theta_{\min}) \right] \text{ otherwise.} \quad (2.19)$$

$$w(t) = \left. \frac{\partial}{\partial \theta} f(\phi, \theta) \right|_{\theta = \hat{\theta}} \text{ if } e_s \geq 0 \quad (2.20)$$

$$w(t) = \frac{f_{\max} - f_{\min}}{\theta_{\max} - \theta_{\min}} \text{ otherwise.} \quad (2.21)$$

2. When  $f(\phi, \theta)$  is convex in  $\theta$  for all  $\phi$ ,

$$u_a(t) = \text{sat}\left(\frac{e_s}{\varepsilon}\right) \left[ \hat{f} - f_{min} - \left( \frac{f_{max} - f_{min}}{\theta_{max} - \theta_{min}} \right) (\hat{\theta} - \theta_{min}) \right] \text{ if } e_s \geq 0 \quad (2.22)$$

$$u_a(t) = 0 \text{ otherwise.} \quad (2.23)$$

$$w(t) = \frac{f_{max} - f_{min}}{\theta_{max} - \theta_{min}} \text{ if } e_s \geq 0 \quad (2.24)$$

$$w(t) = \left. \frac{\partial}{\partial \theta} f(\phi, \theta) \right|_{\theta = \hat{\theta}_i} \text{ otherwise.} \quad (2.25)$$

where

$$f_{max} = f(\phi, \theta_{max}) \text{ and } f_{min} = f(\phi, \theta_{min}) \quad (2.26)$$

The detail discussion of global stability is not included here. However, interested readers are encouraged to obtain more information in ([1],[10]).

### 2.2.3 Extensions

In [10], it has been shown that the results discussed above can be extended to systems of the following form:

$$\dot{x} = A(p)x + b_p x + \sum \sigma_i f_i(\phi, \theta_i) + \psi^T \alpha \quad (2.27)$$

where  $x(t) \in R^n$ ,  $(A(p), b_p)$  is controllable,  $p$  and  $\theta_i$  are unknown scalar parameters,  $\alpha \in R^m$  is unknown,  $A$  and  $f_i$  are nonlinear in  $p$  and  $\theta_i$ , respectively.



# Chapter 3

## Level Control in Feedwater Heater System

### 3.1 Preliminaries

#### 3.1.1 Introduction

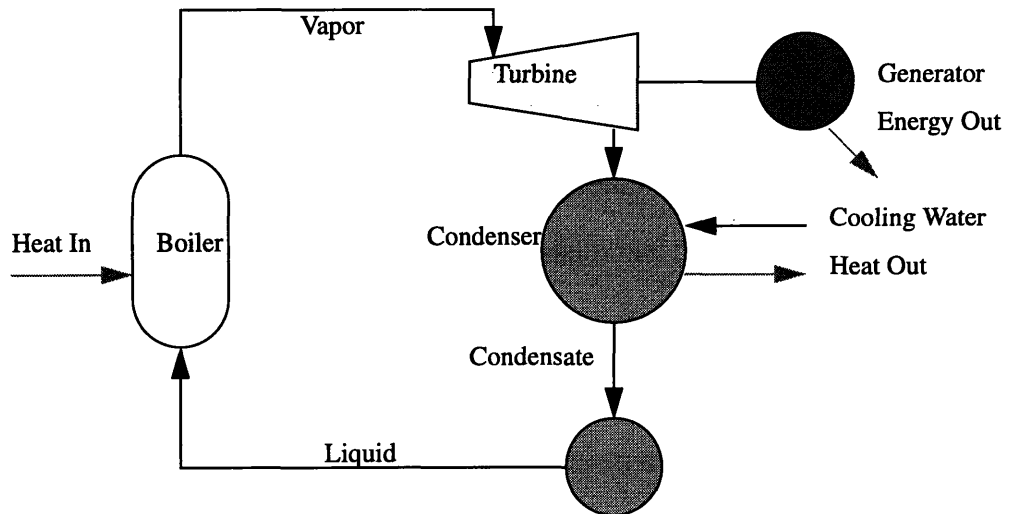
A power cycle is defined as thermodynamic process in which the working fluid can be made to undergo changes involving energy transitions and subsequently return to its original state. The main objective of any cycle is to convert one form of energy to another useful form. For example, the energy stored in fossil fuel is released in the combustion process which, in turn, used to heat the liquid water to the state of vapor that is useful in driving the turbine blade and ultimately generate electricity. We are concerned that natural energy resource is limited and it has to be utilize in the most efficient way possible. In the following sections, we will discuss a vapor power cycle of interest, the Rankine cycle, in which our main goal will be to provide readers the necessary elements to understand the significance of the feedwater heater system.

#### The Rankine Cycle

This cycle can be described in the diagram shown in Figure 3.1. It consists of four distinct processes. Start with the feed pump, the liquid supplied to the boiler is brought to the boiler pressure. In the ideal cycle, the liquid supplied to the pump is assumed to be saturated at the lowest pressure of the cycle. In an actual cycle, the liquid is slightly subcooled to prevent vapor bubbles from forming in the pump (which causes a process known as *cavitation*, which will subsequently damage the pump). For the ideal cycle, the compression process is taken to be isentropic, and the final state of the liquid supplied to the boiler is subcooled at the boiler pressure. This subcooled liquid is heated to the saturation state in

the boiler, and it is subsequently vaporized to yield the steam for the prime mover in the cycle. The energy for the heating and vaporizing process of the liquid is provided by the combustion of the fuel in the boiler. If the superheating of the vapor is desired, it is also accomplished in the boiler (also called a steam generator). The vapor leaves the steam generator and is expanded isentropically in a prime mover (such as a turbine or steam engine) to provide the work output of the cycle. After the expansion process is completed, the working substance is piped to the condenser, where it rejects heat to the cooling water.

**Figure 3.1:** A simple schematic diagram of Rankine cycle



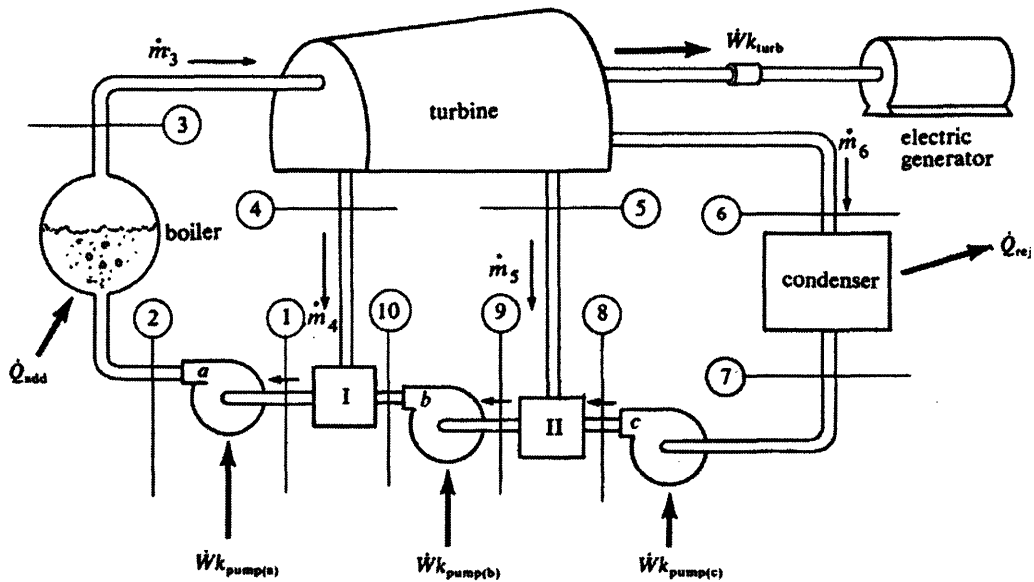
### The Regenerative Cycle

A method to increase steam cycle efficiency without increasing the superheated steam pressure and temperature is the regenerative heating process which is, essentially, a method of adding heat at a higher temperature. Regenerative heating, in practice, is the process of expansion of steam in the turbine well into the phase-change region. Moisture is withdrawn mechanically from the turbine to reduce the effects of wear and corrosion. In Figure 3.2, we show the physical features of the regenerative steam turbine cycle. The saturated liquid from the condenser is fed to a mixing chamber by a pump. The chamber allows the liquid to be heated by mixing with the steam bled from the turbine; then this



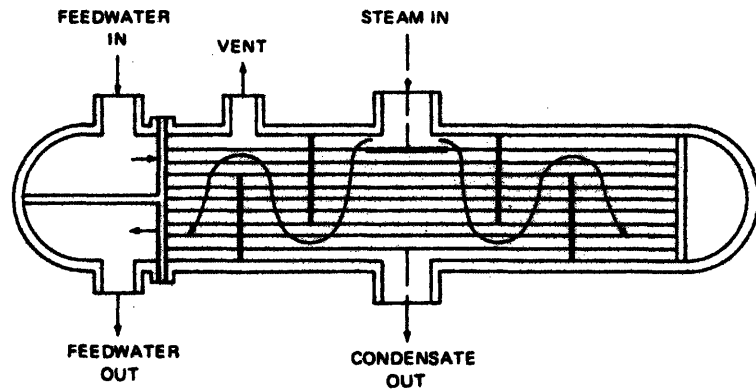
mixture is fed to the next chamber and ultimately back to the boiler for recirculation. In the cycle, two mixing chamber are utilized, but more could be added. These mixing chambers are called feedwater heaters.

**Figure 3.2: Regenerative cycle**



In large central station installation, from one up to a dozen of feedwater heaters are often used. These can attain length of over 60 ft., diameters up to 7 ft., and have over 30,000 ft.<sup>2</sup> of surface. Figure 3.3 shows a straight condensing type of feedwater heater with steam entering at the center and flowing longitudinally, in a baffled path, on the outside of the tubes. Vents are provided to prevent the buildup noncondensable gases in the heater, and are especially important where the pressures are less than atmospheric pressure. Also, deaerators are often used in conjunction with the feedwater heaters or as a separate units to reduce the quantity of oxygen and other noncondensable gases in the feedwater heater to acceptable levels.

**Figure 3.3: Feedwater heater**

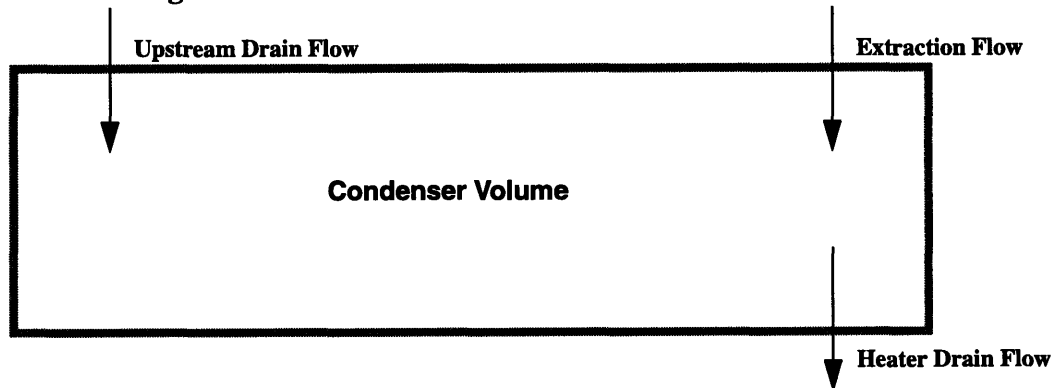


### 3.1.2 Theoretical Development

#### The Extraction Steam Flow

The amount of the extraction steam flow is directly influenced by the thermodynamic process within the heater. Two important regulating mechanisms are the heat transfer process and the thermodynamic process. The heat transfer process occurs within the heater as well as between the heater and the environment. The thermodynamic process takes place within the control volume where upstream drain flow, heater drain flow, and extraction flow interact. The model used in predicting the extraction steam flow is based on the conservation of energy, the conservation of mass, and the thermodynamic relations. The model suggested by Davis is presented here. [5]

**Figure 3.4: The control volume of the feedwater heater**



## 1. Definitions of Variables

$V_c$  = total volume of the condensing region

$M_c$  = total mass of the water in the condenser

$P$  = heater pressure

$h_f$  = saturated liquid specific enthalpy at  $P$

$h_g$  = saturated vapor specific enthalpy at  $P$

$h_{fg} = h_g - h_f$

$h_c$  = specific enthalpy of the condenser mixture

$v_f$  = saturated liquid specific volume at  $P$

$v_g$  = saturated vapor specific volume at  $P$

$v_{fg} = v_g - v_f$

$v_c$  = specific volume of condenser mixture

$x$  = steam quality in the condenser

$m_1$  = extraction flow

$h_3$  = specific enthalpy of flow from the desuperheater

$m_2$  = upstream drain flow

$h_2$  = upstream drain specific enthalpy

$m_5$  = heater drain flow

$h_5$  = heater drain specific enthalpy (assumed equal to  $h_f$ )

$UA_{co} = UA$  for the heat transfer from condenser to tubes

$LMTD_{co} =$  condenser log mean temperature difference

$UA_{hs} = UA$  for the heat transfer from condenser to heat sink

$T_{hs} =$  temperature of the heat sink

$T_{sat} =$  saturation temperature at  $P$

## 2. The Extraction Steam Flow Model

The conservation of mass states that

$$\frac{dM_c}{dt} = m_1 + m_2 - m_5 \quad \text{where } M_c = \frac{V_c}{v_c}. \quad (3.1)$$

Therefore,

$$\frac{dM_c}{dt} = \frac{1}{v_c} \frac{dV_c}{dt} - \frac{V_c}{v_c^2} \frac{dv_c}{dt} \quad \text{where } \frac{dV_c}{dt} = 0 \quad (3.2)$$

since the condenser volume is constant.

Now,

$$\frac{dM_c}{dt} = -\frac{V_c}{v_c^2} \frac{dv_c}{dt} = m_1 + m_2 - m_5. \quad (3.3)$$

Consider  $v_c$  to be a function of the pressure and the enthalpy inside the condenser,

then

$$\frac{dv_c}{dt} = \left( \frac{\partial v_c}{\partial h_c} \right)_P \frac{dh_c}{dt} + \left( \frac{\partial v_c}{\partial P} \right)_{h_c} \frac{dP}{dt}. \quad (3.4)$$

**Derivation of  $\frac{dh_c}{dt}$**

The energy balance relation states that

$$\frac{d}{dt} M_c u_c = m_1 h_3 + m_2 h_2 - m_5 h_f - UA_{co} LMTD_{co} + UA_{hs} (T_{hs} - T_{sat}). \quad (3.5)$$

By definition,

$$u_c = h_c - P v_c. \quad (3.6)$$

Then,

$$\frac{d}{dt} M_c u_c = \frac{d}{dt} M_c (h_c - P v_c) = (h_c - P v_c) \frac{dM_c}{dt} + M_c \frac{d}{dt} (h_c - P v_c) \quad (3.7)$$

$$\frac{d}{dt} M_c u_c = h_c \frac{dM_c}{dt} - P v_c \frac{dM_c}{dt} + M_c \frac{dh_c}{dt} - M_c v_c \frac{dP}{dt} - M_c P \frac{dv_c}{dt} \quad (3.8)$$

Since  $M_c v_c = V_c$ ,

$$\frac{d}{dt} M_c u_c = h_c \frac{dM_c}{dt} + M_c \frac{dh_c}{dt} - V_c \frac{dP}{dt} - P \left( v_c \frac{dM_c}{dt} + M_c \frac{dv_c}{dt} \right). \quad (3.9)$$

However, since the condenser volume is constant,

$$v_c \frac{dM_c}{dt} + M_c \frac{dv_c}{dt} = \frac{d}{dt} M_c v_c = \frac{dV_c}{dt} = 0. \quad (3.10)$$

Hence,

$$\frac{d}{dt} M_c u_c = h_c \frac{dM_c}{dt} + M_c \frac{dh_c}{dt} - V_c \frac{dP}{dt} \quad \text{or} \quad M_c \frac{dh_c}{dt} = \frac{d}{dt} M_c u_c - h_c \frac{dM_c}{dt} + V_c \frac{dP}{dt}. \quad (3.11)$$

Substitute the expression  $\frac{d}{dt} M_c u_c$  in Eq. (3.5), we then have

$$M_c \frac{dh_c}{dt} = m_1 h_3 + m_2 h_2 - m_5 h_f - UA_{co} LMTD_{co} + UA_{hs} (T_{hs} - T_{sat}) - h_c \frac{dM_c}{dt} + V_c \frac{dP}{dt}. \quad (3.12)$$

Again, since

$$\frac{dM_c}{dt} = m_1 + m_2 - m_5, \quad (3.13)$$

$$\frac{dh_c}{dt} = \frac{1}{M_c} \left[ m_1 (h_3 - h_c) + m_2 (h_2 - h_c) - m_5 (h_f - h_c) - UA_{co} LMTD_{co} + UA_{hs} (T_{hs} - T_{sat}) + V_c \frac{dP}{dt} \right]. \quad (3.14)$$

### Derivation of $M_c \frac{h_{fg}}{v_{fg}} \left( \frac{\partial v_c}{\partial P} \right)_{h_c}$

Using the fact that

$$v_c = \frac{h_c}{h_{fg}} (v_g - v_f) + \frac{1}{h_{fg}} (h_g v_f - h_f v_g) \quad \text{and} \quad v_{fg} = v_g - v_f, \quad (3.15)$$

$$v_c = h_c \frac{v_{fg}}{h_{fg}} + v_f \frac{h_g}{h_{fg}} - v_g \frac{h_f}{h_{fg}}. \quad (3.16)$$

By differentiating term by term,

$$\frac{\partial}{\partial P} \left( h_c \frac{v_{fg}}{h_{fg}} \right) = - \frac{h_c v_{fg}}{h_{fg}^2} \left( \frac{\partial h_{fg}}{\partial P} \right) + \frac{1}{h_{fg}} \left( \frac{\partial}{\partial P} h_c v_{fg} \right) = - \frac{h_c v_{fg}}{h_{fg}^2} \left( \frac{\partial h_{fg}}{\partial P} \right) + \frac{1}{h_{fg}} \left( h_c \frac{\partial v_{fg}}{\partial P} + v_{fg} \frac{\partial h_c}{\partial P} \right), \quad (3.17)$$

$$\frac{\partial}{\partial P} \left( v_f \frac{h_g}{h_{fg}} \right) = -\frac{h_g v_f}{h_{fg}^2} \left( \frac{\partial h_{fg}}{\partial P} \right) + \frac{1}{h_{fg}} \left( h_g \frac{\partial v_f}{\partial P} + v_f \frac{\partial h_g}{\partial P} \right), \text{ and } \frac{\partial}{\partial P} \left( v_g \frac{h_f}{h_{fg}} \right) = -\frac{h_f v_g}{h_{fg}^2} \left( \frac{\partial h_{fg}}{\partial P} \right) + \frac{1}{h_{fg}} \left( h_f \frac{\partial v_g}{\partial P} + v_g \frac{\partial h_f}{\partial P} \right).$$

Therefore,

$$\left( \frac{\partial v_c}{\partial P} \right)_{h_c} = -\frac{h_c v_{fg}}{h_{fg}^2} \left( \frac{\partial h_{fg}}{\partial P} \right) + \frac{1}{h_{fg}} \left( h_c \frac{\partial v_{fg}}{\partial P} \right) - \frac{h_g v_f}{h_{fg}^2} \left( \frac{\partial h_{fg}}{\partial P} \right) + \frac{1}{h_{fg}} \left( h_g \frac{\partial v_f}{\partial P} + v_f \frac{\partial h_g}{\partial P} \right) + \frac{h_f v_g}{h_{fg}^2} \left( \frac{\partial h_{fg}}{\partial P} \right) - \frac{1}{h_{fg}} \left( h_f \frac{\partial v_g}{\partial P} + v_g \frac{\partial h_f}{\partial P} \right).$$

By rearranging,

$$\left( \frac{\partial v_c}{\partial P} \right)_{h_c} = h_c \left[ \frac{1}{h_{fg}} \left( \frac{\partial v_{fg}}{\partial P} \right) - \frac{v_{fg}}{h_{fg}^2} \left( \frac{\partial h_{fg}}{\partial P} \right) \right] + \left[ \left( \frac{h_f v_g}{h_{fg}^2} - \frac{h_g v_f}{h_{fg}^2} \right) \frac{\partial h_{fg}}{\partial P} + \frac{h_g}{h_{fg}} \left( \frac{\partial v_f}{\partial P} \right) - \frac{h_f}{h_{fg}} \left( \frac{\partial v_g}{\partial P} \right) - \frac{v_g}{h_{fg}} \left( \frac{\partial h_f}{\partial P} \right) + \frac{v_f}{h_{fg}} \left( \frac{\partial h_g}{\partial P} \right) \right].$$

Multiply by  $M_c$ ,

$$M_c \left( \frac{\partial v_c}{\partial P} \right)_{h_c} = M_c h_c \left[ \frac{1}{h_{fg}} \left( \frac{\partial v_{fg}}{\partial P} \right) - \frac{v_{fg}}{h_{fg}^2} \left( \frac{\partial h_{fg}}{\partial P} \right) \right] + M_c \left[ \left( \frac{h_f v_g}{h_{fg}^2} - \frac{h_g v_f}{h_{fg}^2} \right) \frac{\partial h_{fg}}{\partial P} + \frac{h_g}{h_{fg}} \left( \frac{\partial v_f}{\partial P} \right) - \frac{h_f}{h_{fg}} \left( \frac{\partial v_g}{\partial P} \right) - \frac{v_g}{h_{fg}} \left( \frac{\partial h_f}{\partial P} \right) + \frac{v_f}{h_{fg}} \left( \frac{\partial h_g}{\partial P} \right) \right].$$

Note that the expressions in the bracket are single-valued functions of  $P$ .

The quality of the mixture  $x = \frac{v_c - v_f}{v_{fg}}$ .

The enthalpy of the water in the condenser  $h_c = h_f + \left( \frac{v_c - v_f}{v_{fg}} \right) h_{fg}$ .

Multiplying by  $M_c$ , we now have

$$M_c h_c = M_c h_f + \frac{M_c v_c}{v_{fg}} h_{fg} - \frac{M_c v_f}{v_{fg}} h_{fg} = M_c \left( h_f - \frac{v_f h_{fg}}{v_{fg}} \right) + v_c \frac{h_{fg}}{v_{fg}}. \quad (3.18)$$

Thus,

$$M_c \left( \frac{\partial v_c}{\partial P} \right)_{h_c} = \left[ M_c \left( h_f - \frac{v_f h_{fg}}{v_{fg}} \right) + v_c \frac{h_{fg}}{v_{fg}} \right] \left[ \frac{1}{h_{fg}} \left( \frac{\partial v_{fg}}{\partial P} \right) - \frac{v_{fg}}{h_{fg}^2} \left( \frac{\partial h_{fg}}{\partial P} \right) \right] + M_c \left[ \left( \frac{h_f v_g}{h_{fg}^2} - \frac{h_g v_f}{h_{fg}^2} \right) \frac{\partial h_{fg}}{\partial P} + \frac{h_g}{h_{fg}} \left( \frac{\partial v_f}{\partial P} \right) - \frac{h_f}{h_{fg}} \left( \frac{\partial v_g}{\partial P} \right) - \frac{v_g}{h_{fg}} \left( \frac{\partial h_f}{\partial P} \right) + \frac{v_f}{h_{fg}} \left( \frac{\partial h_g}{\partial P} \right) \right].$$

Further expanding the expression gives,

$$\begin{aligned}
M_c \left( \frac{\partial v_c}{\partial P} \right)_{h_c} &= M_c \left[ \frac{h_f}{h_{fg}} \left( \frac{\partial v_{fg}}{\partial P} \right) - \frac{v_f}{v_{fg}} \left( \frac{\partial v_{fg}}{\partial P} \right) - \frac{h_f v_{fg}}{h_{fg}^2} \left( \frac{\partial h_{fg}}{\partial P} \right) + \frac{v_f}{h_{fg}} \left( \frac{\partial h_{fg}}{\partial P} \right) \right] \\
&+ M_c \left[ \frac{h_f v_{fg}}{h_{fg}^2} \left( \frac{\partial h_{fg}}{\partial P} \right) - \frac{h_g v_f}{h_{fg}^2} \left( \frac{\partial h_{fg}}{\partial P} \right) + \frac{h_g}{h_{fg}} \left( \frac{\partial v_f}{\partial P} \right) - \frac{h_f}{h_{fg}} \left( \frac{\partial v_g}{\partial P} \right) - \frac{v_g}{h_{fg}} \left( \frac{\partial h_f}{\partial P} \right) + \frac{v_f}{h_{fg}} \left( \frac{\partial h_g}{\partial P} \right) \right] + V_c \left[ \frac{1}{v_{fg}} \left( \frac{\partial v_{fg}}{\partial P} \right) - \frac{1}{h_{fg}} \left( \frac{\partial h_{fg}}{\partial P} \right) \right]. \\
M_c \left( \frac{\partial v_c}{\partial P} \right)_{h_c} &= M_c \left[ \left( \frac{h_f}{h_{fg}} - \frac{v_f}{v_{fg}} \right) \left( \frac{\partial v_{fg}}{\partial P} \right) + \frac{h_g}{h_{fg}} \left( \frac{\partial v_f}{\partial P} \right) - \frac{h_f}{h_{fg}} \left( \frac{\partial v_g}{\partial P} \right) - \frac{v_g}{h_{fg}} \left( \frac{\partial h_f}{\partial P} \right) + \frac{v_f}{h_{fg}} \left( \frac{\partial h_g}{\partial P} \right) \right] \\
&+ M_c \left[ \left( \frac{h_f v_{fg} - h_g v_f - h_f v_{fg} + v_f h_{fg}}{h_{fg}^2} \right) \left( \frac{\partial h_{fg}}{\partial P} \right) \right] + V_c \left[ \frac{1}{v_{fg}} \left( \frac{\partial v_{fg}}{\partial P} \right) - \frac{1}{h_{fg}} \left( \frac{\partial h_{fg}}{\partial P} \right) \right].
\end{aligned}$$

Note that  $h_f v_g - h_g v_f - h_f v_{fg} + v_f h_{fg} = h_f v_g - h_g v_f - h_f (v_g - v_f) + v_f (h_g - h_f) = 0$ .

Therefore,

$$M_c \left( \frac{\partial v_c}{\partial P} \right)_{h_c} = M_c \left[ \left( \frac{h_f}{h_{fg}} - \frac{v_f}{v_{fg}} \right) \left( \frac{\partial v_{fg}}{\partial P} \right) + \frac{h_g}{h_{fg}} \left( \frac{\partial v_f}{\partial P} \right) - \frac{h_f}{h_{fg}} \left( \frac{\partial v_g}{\partial P} \right) - \frac{v_g}{h_{fg}} \left( \frac{\partial h_f}{\partial P} \right) + \frac{v_f}{h_{fg}} \left( \frac{\partial h_g}{\partial P} \right) \right] + V_c \left[ \frac{1}{v_{fg}} \left( \frac{\partial v_{fg}}{\partial P} \right) - \frac{1}{h_{fg}} \left( \frac{\partial h_{fg}}{\partial P} \right) \right].$$

Now, multiply the expression by  $\frac{h_{fg}}{v_{fg}}$ ,

$$\begin{aligned}
M_c \frac{h_{fg}}{v_{fg}} \left( \frac{\partial v_c}{\partial P} \right)_{h_c} &= M_c \left[ \frac{h_{fg}}{v_{fg}} \left( \frac{h_f}{h_{fg}} - \frac{v_f}{v_{fg}} \right) \left( \frac{\partial v_{fg}}{\partial P} \right) + \frac{h_g}{v_{fg}} \left( \frac{\partial v_f}{\partial P} \right) - \frac{h_f}{v_{fg}} \left( \frac{\partial v_g}{\partial P} \right) - \frac{v_g}{v_{fg}} \left( \frac{\partial h_f}{\partial P} \right) + \frac{v_f}{v_{fg}} \left( \frac{\partial h_g}{\partial P} \right) \right] \\
&+ V_c \left[ \frac{h_{fg}}{v_{fg}^2} \left( \frac{\partial v_{fg}}{\partial P} \right) - \frac{1}{v_{fg}} \left( \frac{\partial h_{fg}}{\partial P} \right) \right].
\end{aligned}$$

This expression can alternatively be written in form of,  $M_c \frac{h_{fg}}{v_{fg}} \left( \frac{\partial v_c}{\partial P} \right)_{h_c} = M_c A + V_c B$ ,

where  $A$  and  $B$  are strictly functions of pressure.

From the original conservation of mass equation,

$$m_1 + m_2 - m_5 = -\frac{V_c}{v_c} \left[ \left( \frac{\partial v_c}{\partial h_c} \right)_P \frac{dh_c}{dt} + \left( \frac{\partial v_c}{\partial P} \right)_{h_c} \frac{dP}{dt} \right],$$

$$\begin{aligned}
m_1 + m_2 - m_5 &= -\frac{V_c}{v_c} \left\{ \frac{v_{fg}}{h_{fg}} \frac{1}{M_c} [m_1 (h_3 - h_c) + m_2 (h_2 - h_c) - m_5 (h_f - h_c) - UA_{co} LMTD_{co} + UA_{hs} (T_{hs} - T_{sat})] \right\} \\
&- \frac{V_c}{v_c} \left\{ \left( \frac{v_{fg}}{h_{fg}} \frac{1}{M_c} \right) v_c \frac{dP}{dt} + \left( \frac{\partial v_c}{\partial P} \right)_{h_c} \frac{dP}{dt} \right\}.
\end{aligned}$$

Since  $v_c = \frac{V_c}{M_c}$ ,

$$m_1 + m_2 - m_5 = -\frac{v_{fg}}{v_c h_{fg}} [m_1(h_3 - h_c) + m_2(h_2 - h_c) - m_5(h_f - h_c) - UA_{co}LMTD_{co} + UA_{hs}(T_{hs} - T_{sat})]$$

$$- M_c \frac{v_{fg} dP}{h_{fg} dt} - \frac{V_c}{v_c^2} \left[ \left( \frac{\partial v_c}{\partial P} \right)_{h_c} \frac{dP}{dt} \right],$$

$$\frac{v_c h_{fg}}{v_{fg}} (-m_1 - m_2 + m_5) = m_1(h_3 - h_c) + m_2(h_2 - h_c) - m_5(h_f - h_c) - UA_{co}LMTD_{co} + UA_{hs}(T_{hs} - T_{sat})$$

$$+ V_c \frac{dP}{dt} + \frac{M_c h_{fg}}{v_{fg}} \left( \left( \frac{\partial v_c}{\partial P} \right)_{h_c} \frac{dP}{dt} \right).$$

Ultimately, the extraction steam flow can be expressed in the following form,

$$m_1 = \left( \frac{1}{h_3 - h_c + \frac{v_c h_{fg}}{v_{fg}}} \right) \left\{ m_2 \left( h_2 - h_c - \frac{v_c h_{fg}}{v_{fg}} \right) - m_5 \left( h_c - h_f - \frac{v_c h_{fg}}{v_{fg}} \right) - UA_{co}LMTD_{co} + UA_{hs}(T_{hs} - T_{sat}) - V_c \right\}$$

$$- \left( \frac{1}{h_3 - h_c + \frac{v_c h_{fg}}{v_{fg}}} \right) \left\{ M_c \left[ \frac{h_{fg}}{v_{fg}} \left( \frac{h_f}{h_{fg}} - \frac{v_f}{v_{fg}} \right) \left( \frac{\partial v_{fg}}{\partial P} \right) + \frac{h_g}{v_{fg}} \left( \frac{\partial v_f}{\partial P} \right) - \frac{h_f}{v_{fg}} \left( \frac{\partial v_g}{\partial P} \right) - \frac{v_g}{v_{fg}} \left( \frac{\partial h_f}{\partial P} \right) + \frac{v_f}{v_{fg}} \left( \frac{\partial h_g}{\partial P} \right) \right] \frac{dP}{dt} \right\}$$

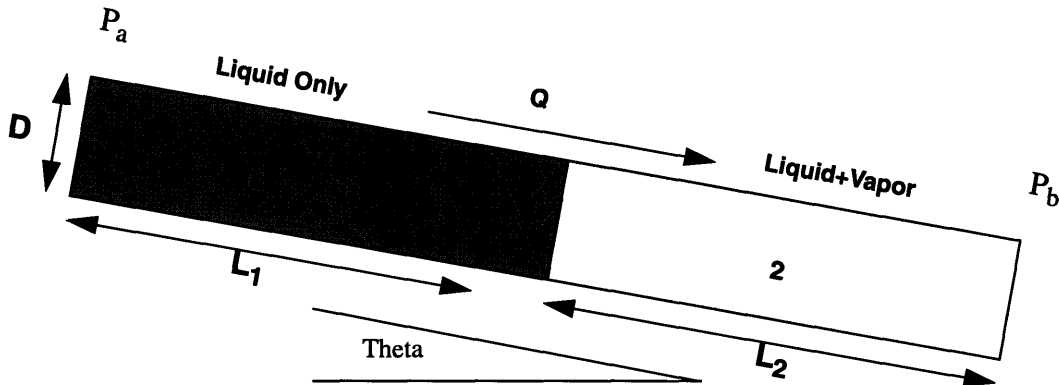
$$- \left( \frac{1}{h_3 - h_c + \frac{v_c h_{fg}}{v_{fg}}} \right) \left\{ V_c \left[ \frac{h_{fg}}{v_{fg}^2} \left( \frac{\partial v_{fg}}{\partial P} \right) - \frac{1}{v_{fg}} \left( \frac{\partial h_{fg}}{\partial P} \right) \right] \frac{dP}{dt} \right\}$$



## Fluid Flow Resistances

In this section, we will derive the dynamic model of the flow system from the drain exit of the heater to the cascaded flow inlet of the downstream heater. Figure 3.5 shows the schematic diagram of the system that our model is based on.

**Figure 3.5:** The schematic diagram of the drain flow system



We will investigate both the case where the flow is assumed to be entirely single phase and the case where the flow downstream is in liquid-vapor phase. The flow through the drain valve is always in liquid phase. The pipe system is assumed to be well insulated therefore the heat transfer to the environment is negligible. The momentum flux changes due to the friction in the pipe is usually very small and is neglected in our model.

### 1. Single-Phase Flow

The pressure drop in the drain system occurs at two places. We will now consider the case which the flow is assumed to be entirely liquid. The pressure drop increases as the fluid passes further downstream through the drain pipe and the control valve.

#### Pressure Drop Across the Pipe

The equation which describe the pressure drop for a flow with velocity  $v$  is given by

$$\Delta P = \frac{1}{2} K \rho v^2 \quad (3.19)$$

where the friction coefficient is defined as

$$K = f \frac{L}{D}. \quad (3.20)$$

The friction factor,  $f$ , is a function of the relative roughness of the pipe and the Reynold's number. In the case which the flow is turbulence ( $Re > 2000$ ),

$$f = \frac{0.316}{Re^{0.25}} \quad (3.21)$$

By substituting the relation in Eq. (3.20)-(3.21) in Eq. (3.19) and solve for the pressure drop as a function of the volumetric flow rate  $Q$ ,

$$Q = Av \quad (3.22)$$

$$\Delta P_{pipe} = \frac{0.316}{2A} \left[ \frac{\mu A}{\rho D} \right]^{0.25} \left[ \frac{L \rho}{D} \right] Q^2 \Big|_{pipe} \quad (3.23)$$

### Pressure Drop Across the Control Valve

The conventional equation for describing the relationship between the pressure drop and the volumetric flow rate is

$$Q = C_v \sqrt{\frac{\Delta P}{\gamma}} \quad (3.24)$$

The valve coefficient,  $C_v$ , is a function of the valve opening and is usually supplied by the valve manufacturer. We can then solve for the pressure drop across the valve as

$$\Delta P_{valve} = 3225.42 \frac{\rho}{C_v^2} Q^2 \quad (3.25)$$

## 2. Two-Phase Flow

We are interested in modeling the effects of the two-phase flow particularly the flow mixture of liquid and vapor. Two-phase flow occurs quite frequently in the feedwater heater system. We are concerned with the downstream section of the drain pipe where the flow exits the valve and the pressure drops resulting in partial phase transformation from liquid to vapor. In this section, we will describe the method of characterizing the two-phase flow. However, it must be noted that despite a large number of studies related in this

area, there are many situations where the uncertainty can be as high as 50%. Most of the two-phase correlations are entirely empirical or semi-empirical. We will discuss the prediction of the static pressure gradient approximation along the straight pipe during the two-phase flow.

### Frictional Pressure Gradient

In deriving the correlations, it is assumed that the homogeneous theory is applicable to the system of interest. The liquid-vapor mixture is treated as homogeneous with a density based on the assumption that both phases flow at the same velocity. This theory gives a reasonable result even in the case where the actual vapor velocity is as much as five times faster than that of the liquid. [4] Furthermore, it is assumed that both phases are in turbulent flow condition in a smooth pipe.

The two-phase component of the pressure gradient due to friction is described by

$$-DP_F = \phi_{FLO}^2 DP_{FLO} \quad (3.26)$$

where  $\phi_{FLO}^2$  is the two-phase multiplier and  $DP_{FLO}$  is the pressure drop due to friction if the flow were all liquid.

The Martinelli-Nelson correlation gives the following approximation for the two-phase multiplier

$$\phi_{FLO}^2 = 1 + (\Gamma^2 - 1) \left[ Bx^{\frac{(2-n)}{2}} (1-x)^{\frac{(2-n)}{2}} + x^{(2-n)} \right] \quad (3.27)$$

where the Blasius exponent  $n$  and physical property parameter  $\Gamma^2$  are defined by

$$n = \frac{\log\left(\frac{\lambda_{LO}}{\lambda_{GO}}\right)}{\log\left(\frac{Re_{GO}}{Re_{LO}}\right)} \quad (3.28)$$

$$\Gamma^2 = \left(\frac{\mu_G}{\mu_L}\right)^{nv_G} \frac{v_L}{v_G} \quad (3.29)$$

and the coefficient  $B$  is defined in the following table.

TABLE 1. Values of  $B$  for smooth pipes

$\Gamma$	$G(\text{kg}/\text{m}^2\text{s})$	$B$
$\Gamma \leq 9.5$	$G \leq 500$	4.8
	$500 \leq G \leq 1900$	$2400/G$
	$G \geq 1900$	$55/G^{0.5}$
$9.5 \leq \Gamma \leq 28$	$G \leq 600$	$520/(\Gamma G^{0.5})$
	$G > 600$	$21/\Gamma$
$\Gamma \geq 28$		$15000/(\Gamma^2 G^{0.5})$

Friedel has shown that this table gives reasonable agreement with an extensive data bank [4]

$\lambda_{LO}$  is the friction factor when the total mixture flows as liquid

$\lambda_{GO}$  is the friction factor when the total mixture flows as vapor

$Re$  is the Reynold's number

$\mu_G$  is the dynamic viscosity of the vapor phase

$\mu_L$  is the dynamic viscosity of the liquid phase

$v$  is the specific volume

$G$  is the mass velocity

### Pressure Gradient due to Changes in the Elevation

Let the mixture density be  $\rho$  and  $\theta$  be the angle of the pipe to the horizontal line,

$$-Dp_g = g\rho_m \sin\theta \quad (3.30)$$

$$\rho_m = \alpha\rho_G + (1-\alpha)\rho_L \quad (3.31)$$

### Pressure Gradient due to Changes in Momentum Flux

The following equation describes the pressure drop in the straight pipe due to the change in momentum flux.

$$-Dp_M = G^2 D(v_e) \quad (3.32)$$

where

$$\frac{v_e}{v_L} = 1 + \left(\frac{v_G}{v_L} - 1\right) [Bx(1-x) + x^2] \quad (3.33)$$

The derivative of  $v_e$  can be expanded as

$$D(v_e) = \left[\frac{\partial v_e}{\partial p}\right]_s Dp + \left[\frac{\partial v_e}{\partial s}\right]_p Ds \quad (3.34)$$

$$Ds = \frac{D(q+F)}{T} = \frac{Dq}{T} + v_H \frac{Dp_F}{T} \quad (3.35)$$

where  $q$  is the heat transferred per unit mass and  $F$  is the frictional energy dissipation per unit mass. If we assume that the frictional energy and heat transfer is negligible in an insulated smooth pipe, the expression becomes

$$-Dp_M = G^2 \left[\frac{\partial v_e}{\partial p}\right]_s Dp = -\frac{G^2}{G_C} Dp \quad (3.36)$$

where  $G_C$  is the mass velocity at maximum flow condition (choked flow).

### 3.1.3 Kingston Unit 9 System Description

#### Kingston Unit 9 Feedwater Heater System

Kingston Unit 9 is a shell and tube heat exchanger in which the feedwater flows through the tube and interacts with the steam extracted from the turbine. As the heat from the extraction steam is transferred to the feedwater, some of it condenses and is collected at the bottom of the heater. The level of the water in the heater is critical to the efficiency of the heat transfer and must be controlled quite closely. If the level is too high, the feedwater tubes are submerged and the heat transfer decreases significantly. If the level is too low, the extraction steam in the shell could flow through the drain cooler and, because of its high velocity, can damage the tubes in that area. Moreover, the drain pipe is sized to

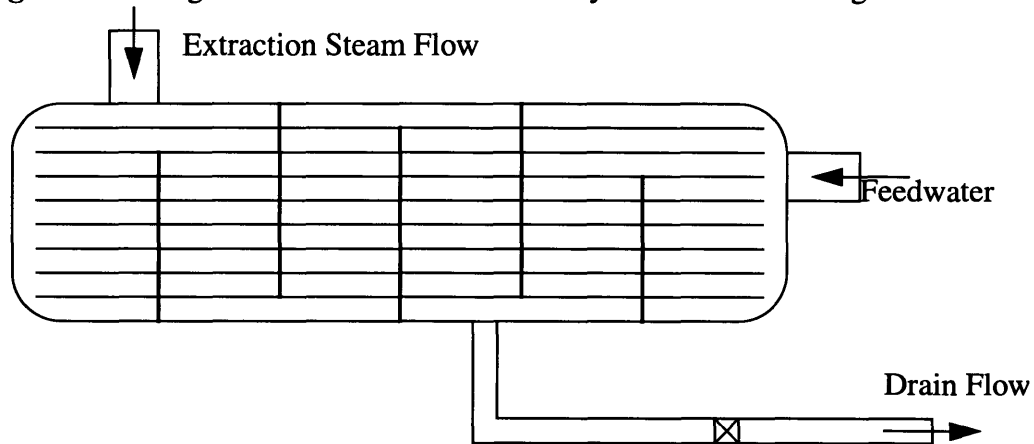
handle fluid not steam, so it will not pass adequate flow if the steam were to enter instead of water.

The level of the water in the shell is controlled by a control valve in the drain pipe. The water from the heater can flow to two different places. In normal condition, it will be passed to the next heater downstream where it is used to augment the temperature of the feedwater before it reaches the current heater. In emergency condition, the water will be passed through the emergency drain valve directly to the condenser of the main turbine. This is not desirable since it is not as efficient.

The heater level is currently controlled by a simple PI controller. There are a couple of system characteristics that can complicate the controller tuning. Ideally, the installed valve characteristic of the drain would be linear over the load range of the plant and the optimum controller gains would also be constant. However, this is not the case therefore the PI controller needs different gains over the load range for optimum response. The inherent valve characteristics should be selected to give the closest to linear response possible without any tweaking in the control system. The sizing of the drain pipe can also affect the flow characteristics quite significantly. These are the two factors which make it difficult to get a linear installed valve characteristic (flow rate versus valve lift over the load range of the plant at actual differential pressure).

Hence, the current PI controller is tuned conservatively so that the stability is maintained as the system characteristics change over the load range of the plant. If the mechanical system is well-designed, the performance is usually acceptable. The schematic of the feedwater heater diagram is given in Figure 3.6.

**Figure 3.6:** Kingston Unit 9 feedwater heater system schematic diagram



### **Kingston Unit 9 System Simulator**

The Tennessee Valley Authority (TVA) Kingston Unit 9 simulator, constructed by Foxboro and ESSCOR, inc., is primarily intended as a training device for engineers and operators of Unit 9. The simulator teaches the trainees how to use the Foxboro I/A system under a variety of plant conditions such as start-ups, shut-down, and emergency situations. The simulator features an Instructor Station Package from which a supervisor can monitor a trainee's performance. Since the simulator exposes a trainee to a wide variety of operating conditions and circumstances, the trainee may gather a lifetime of unit experience before ever operating the actual unit.

Another use of the simulator which is concerned with our operation is to test the effects of controls and/or plant modifications before implementing them on the actual unit allowing tuning optimization and early detection of flaws.

We will now define the terms *simulator*, *model*, and *controls*. The simulator is a complete combination of the Foxboro workstations, the master simulation computer loaded with all model and controls software, the instructor station, and all associated peripheral equipment. The *model* is the set of compiled and linked subroutines which simulates the behavior of the Kingston Unit 9 plant. *Controls* refers to the set of Foxboro unit 9 control

compounds and downloaded onto the master simulation computer from the Foxboro Applications Workstation.

### **1. Simulator Hardware**

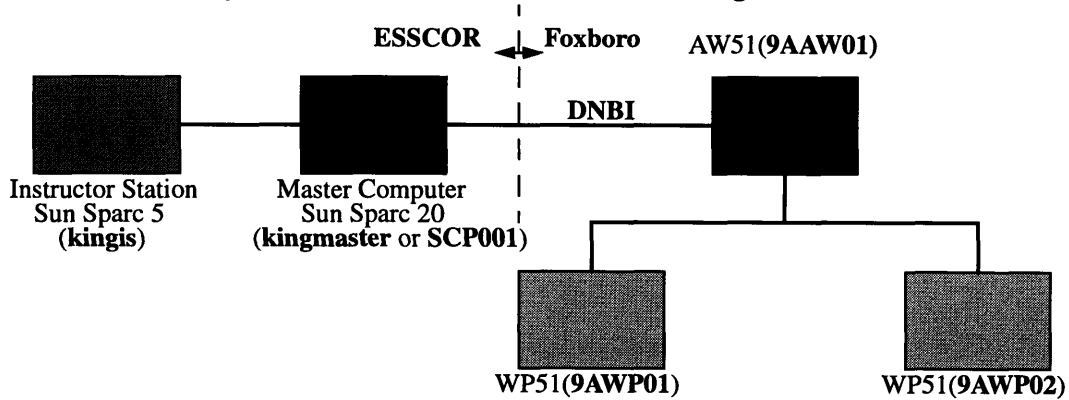
The control hardware portion of the simulator consists of three Foxboro consoles comprised of one Applications Workstation (AW) and two Workstation Processors (WPs). The AW is the center console, flanked by the two WPs. Each workstation has a Sun Sparc LX as its processor.

The “plant” portion of the simulator consists of two Sun Microsystems workstations (a Sparc 20 and a Sparc 5). The Sparc 20 has two machine names, **kingmaster** and **SCP001**. It has two names since it must communicate not only with the Foxboro AW (to which it is recognized by **SCP001**) but also the Sparc 5 (by **kingmaster**). These two names may be used interchangeably throughout this document. Kingmaster is the simulator “master” computer on which the simulator model runs; it passes information to Sparc 5, which serves as the Instructor Station. The machine name of the Sparc 5 is **kingis**. Figure 3.7 shows a hardware block diagram of the simulator hardware. Machine names are in parenthesis and in bold.

SCP001(kingmaster) is connected to the Foxboro hardware through a Dual Node Bus interface (DNBI), and through a serial cable. The DNBI handles all communication between the CP and Foxboro controls, the serial cable passes CP “letterbug” information to the Hardware Connections portion of the Instructor Station.



**Figure 3.7: Simulator hardware block diagram**



## 2. Simulator Software

The two key software elements of the Kingston unit 9 simulator are *SYSL* and *FSIM*. *SYSL* stands for System Simulation Language, and *FSIM* stands for Foxboro Simulation Language. The major functions of each are described below.

The Kingston Unit 9 plant is modeled using the combination of model subroutines and a *SYSL* input file. Each major plant sub-system (such as feedwater, air, fuel, etc.) is modeled in a FORTRAN subroutine. The *SYSL* input file forms the “backbone” of the model by declaring all model variables used, and by making calls to the model subroutines. The steps required to make an executable model are *translation*, *compilation*, and *linking*. When the model is translated, the *SYSL* input file is sorted so that the model subroutines are called in proper order. The translation also converts the input file into compilable FORTRAN source code. This source code is then compiled and linked with the model subroutines, *SYSL* libraries, *FSIM* libraries, and engineering tool libraries. The result of this step is an executable model file. When the model is run, *SYSL* loops through the sorted list of model subroutines, updates variables accordingly, and increments the model timer to the next time step. The rate at which this is “marching” forward occurs can be controlled to make the model go faster or slower than the real (wall clock) time.

The SYSL modeling approach uses a combination of ordinary variables and *state* variables. The ordinary variables are updated in the order in which they are called by the sorted SYSL input file. For example, if model variable *a* calculated in subroutine *A* is a function of model variable *b* calculated in subroutine *B*, SYSL will sort the input file such that subroutine *B* is called before *A*. State variables on the other hand are considered constant throughout the time step being calculated. These variables's derivatives are computed in the various model subroutines, but the state variables will only be updated (i.e. integrated) at the end of the time step, before proceeding to the next time step. The state variable approach to the modeling allows the greatest amount of modeling fidelity with the least consumption of computer processing time.

For the simulator to be useful, it must communicate with controls hardware in such a manner as to make the controls think it is “seeing” the real plant. FSIM accomplishes this task in two ways. First, running FSIM on the master simulation computer turns that computer into a “Soft Compound Processor” (hence, the kingmaster is also named SCP001). Second, FSIM provides the interface between the SYSL plant model and the controls software. This is accomplished through the use of an *I/O Cross-Reference Table*, in which controls *compound:block.parameters* are tied to SYSL model variables. Essentially, this cross reference table replaces the *I/O cabinet* and associated *Field Bus Module* hardware of a real plant installation. FSIM has a process called *cio\_cp*, which synchronizes the controls processing with model processing. Since the model and controls processing occur simultaneously, this process ensures that the model and controls run in “lock-step”.

### **3.1.4 Transient Response Data Acquisition**

#### **Closed-loop Tests From Kingston Unit 9 Plant**

The purpose of conducting the field tests is to use the result to compare with the simulator's model predictions. Heater #1 was subjected to level setpoint changes around the

nominal value of 8 inches with the magnitude of 1 inch or greater. Only the closed-loop tests were conducted since the plant operators suggested that the heater emergency dump valve might be triggered if the control loop were opened. Such condition will be undesirable since the actual plant efficiency will decrease and the level recorded will be the result of the combined effect of the drain valve and the unmodelled emergency dump valve.

Eight signals from heater #1 were recorded. However, we are most concerned with the heater #1 level and the heater #1 valve command. As we have mentioned earlier in the thesis, the level controller used in the feedwater heater at Kingston Unit 9 is proportional plus integral type where both gains can be varied on-line. Three sets of gains were used in the tests with [PB = 35 ; INT = 1.7] being the set which is commonly used.

TABLE 2. Kingston Unit 9 closed-loop response tests

Test Number	Proportional Band	Integral Time	Setpoint Command (in)
1	35	1.7	8.0-9.0-8.0
2	25	1.7	8.0-9.0-8.0
3	18	1.7	8.0-9.0-8.0

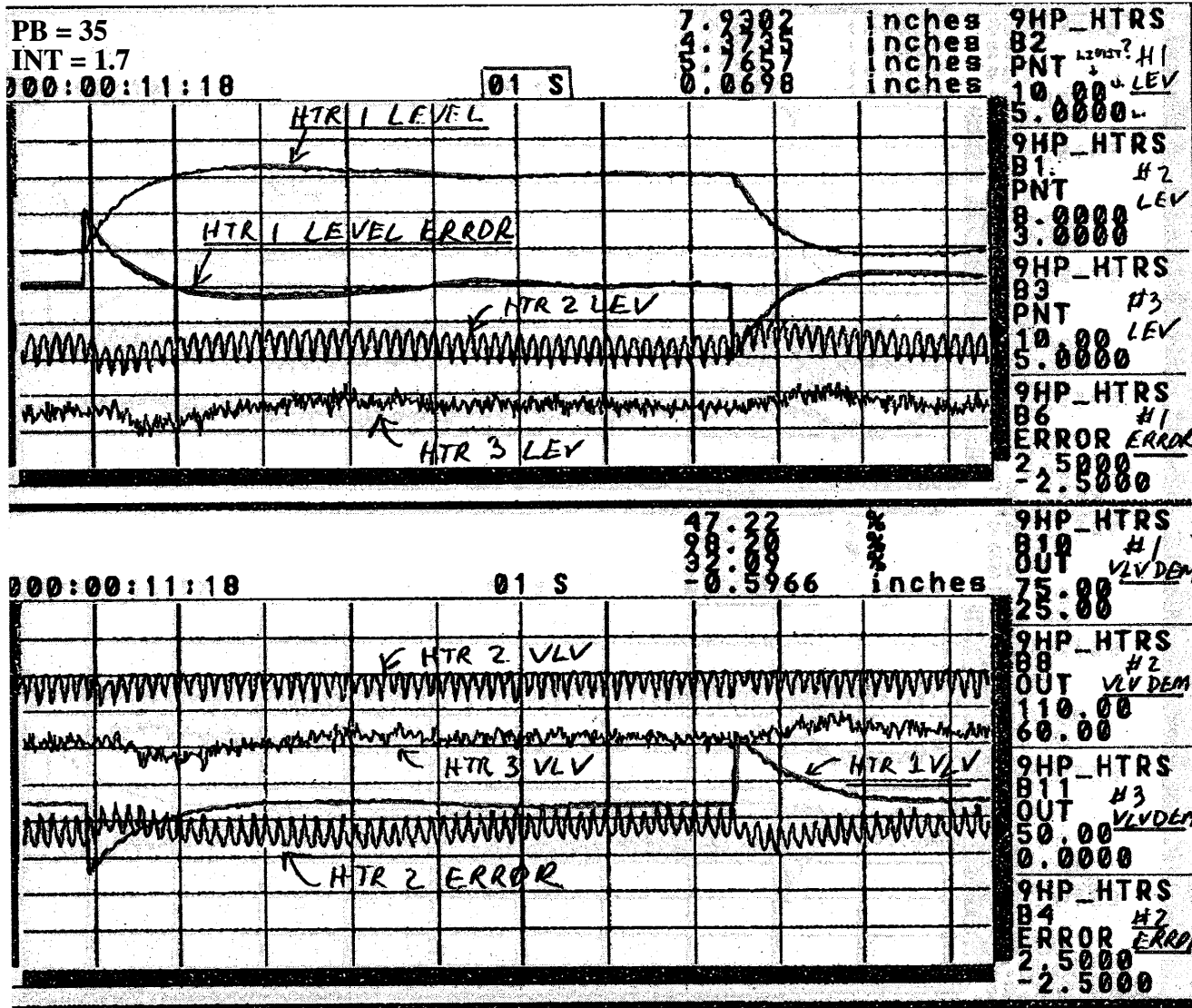


Figure 3.8: Kingston Unit 9 closed-loop test 1

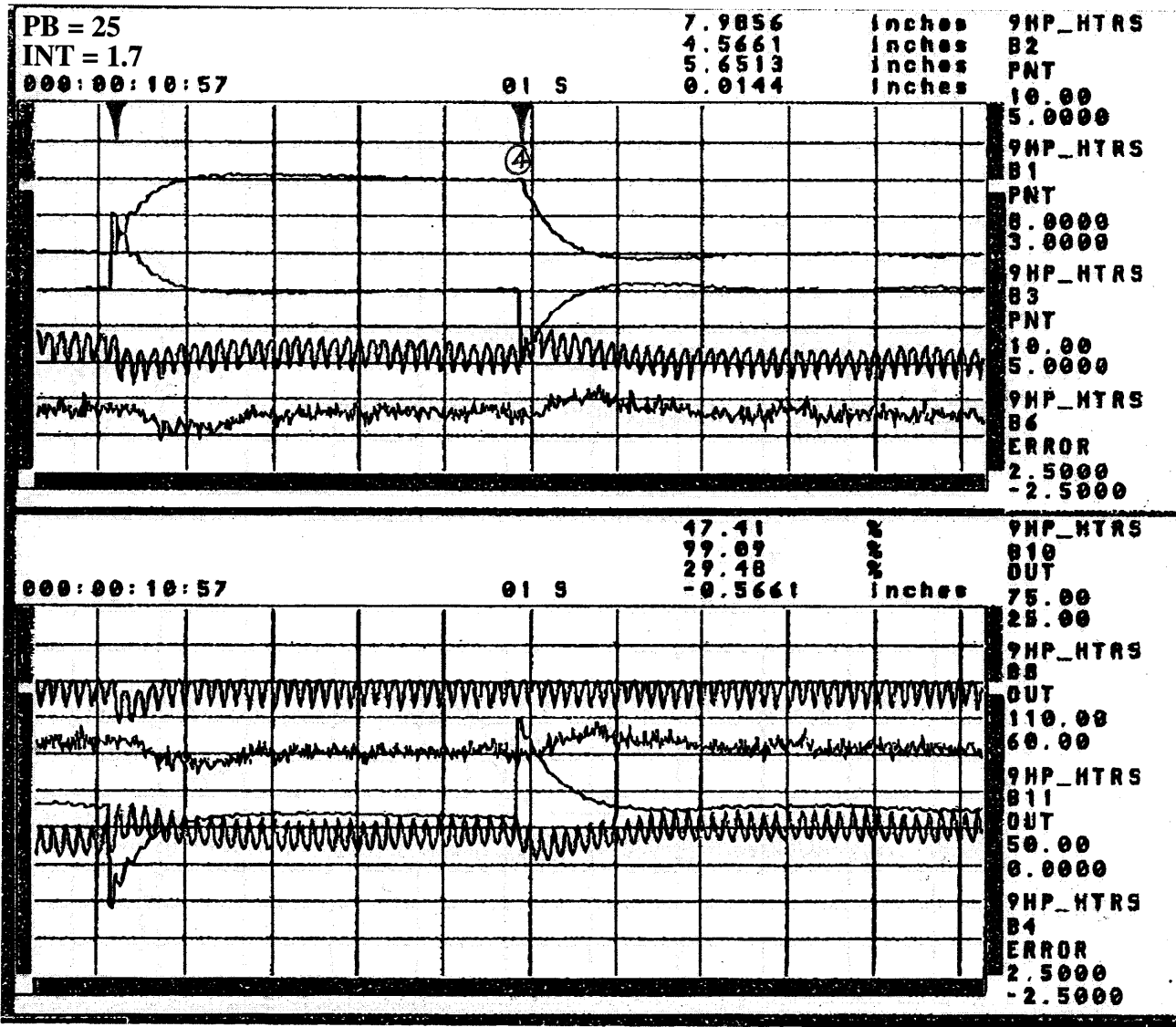


Figure 3.9: Kingston Unit 9 closed-loop test 2

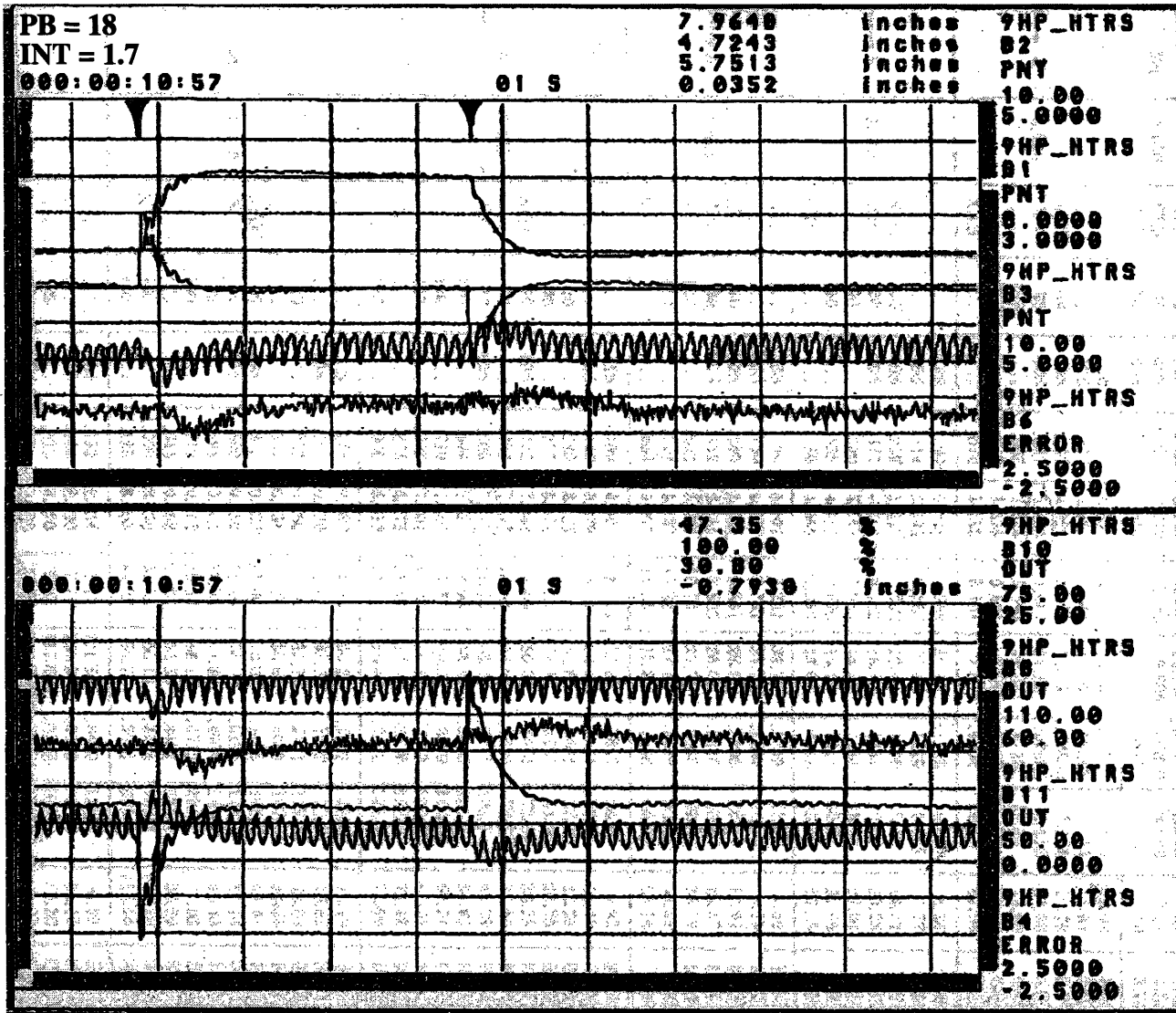


Figure 3.10: Kingston Unit 9 closed-loop test 3

## Closed-loop Tests From Kingston Unit 9 Simulator System

A number of similar transient response tests were also conducted on the plant simulator to verify that the simulator model gives reasonably accurate predictions of Kingston Unit 9 feedwater heater system. Throughout the tests, the same proportional plus integral controller was used on the simulator. The gains were changed at the simulator's operator station each time a new test was conducted. Table 3 described the conditions in which each test was carried out.

TABLE 3. Simulator closed-loop response tests

Test Number	Proportional Band	Integral Time	Setpoint Command (in)
1	35	1.7	8.0-9.0-8.0
2	25	1.7	8.0-9.0-8.0
3	18	1.7	8.0-9.0-8.0

**Figure 3.11: Simulator closed-loop test 1**

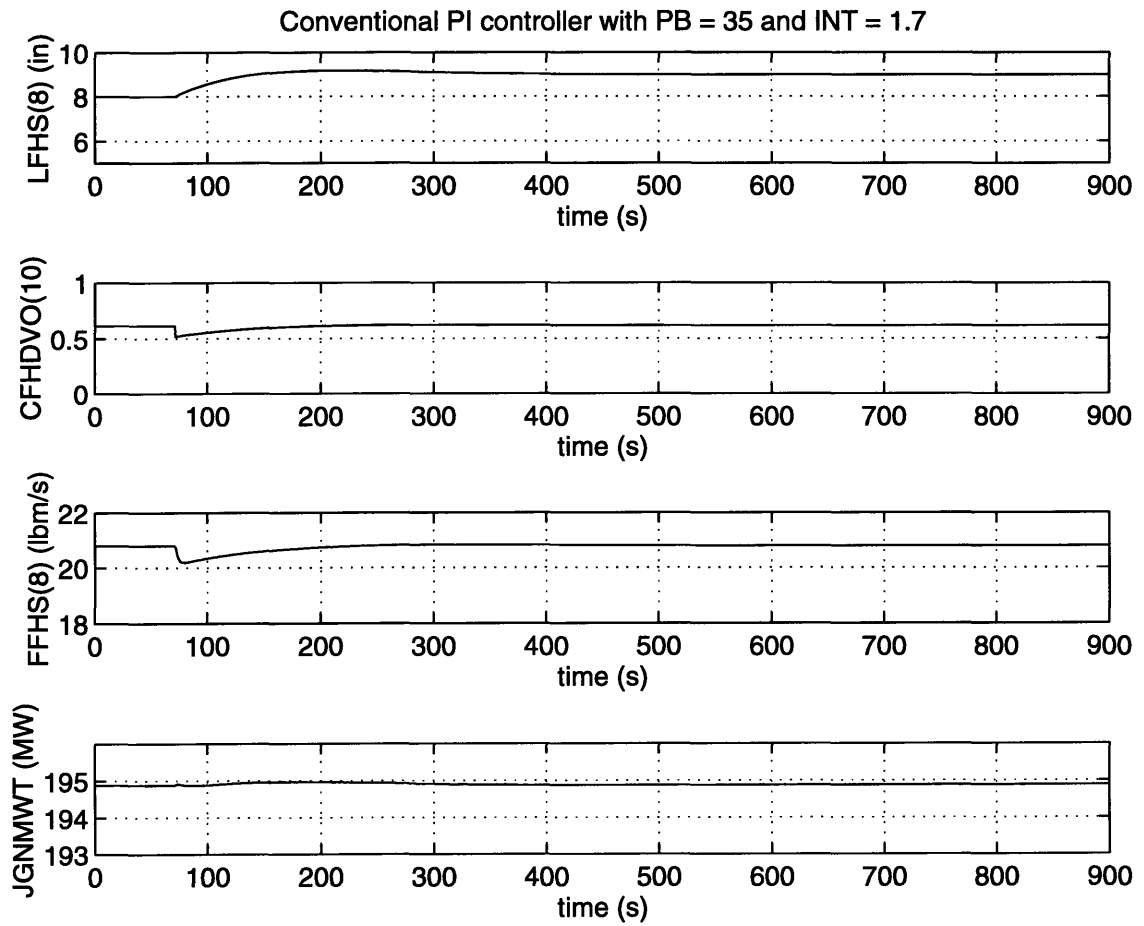
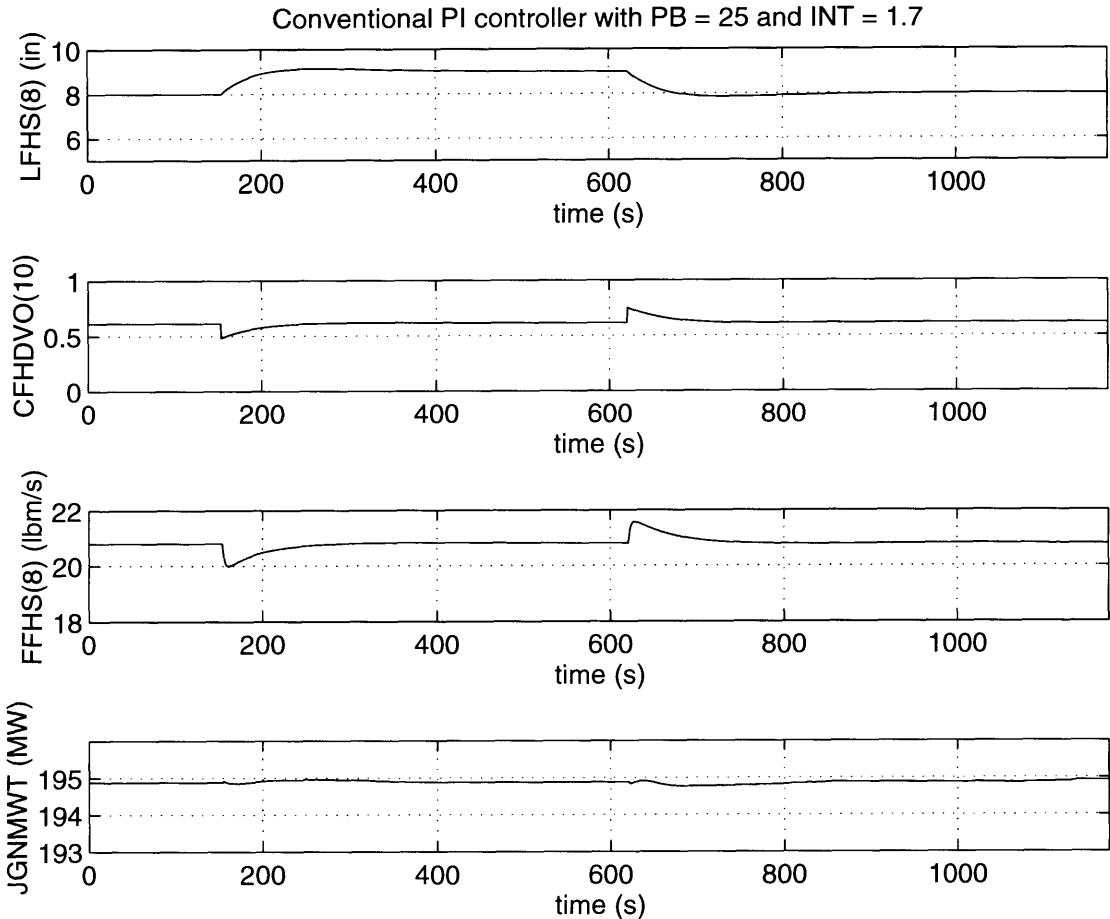
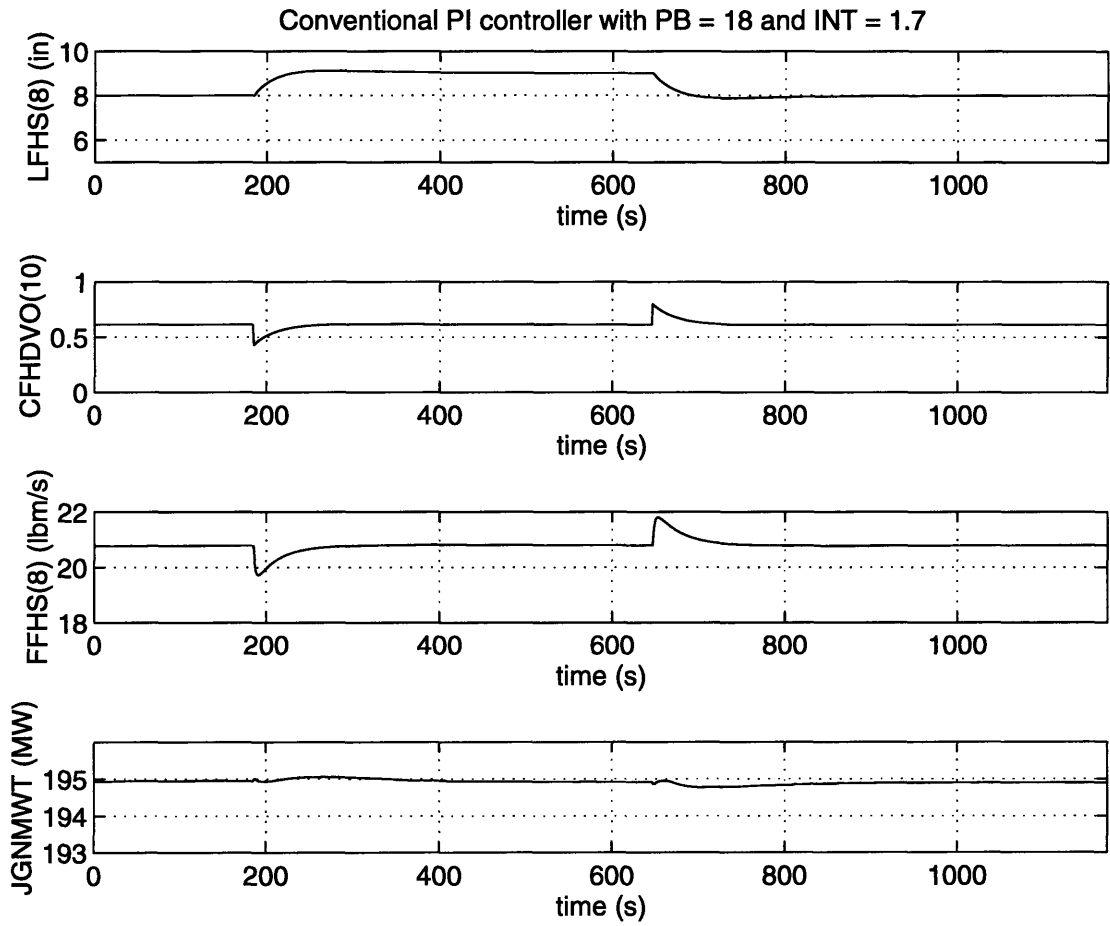




Figure 3.12: Simulator closed-loop test 2



**Figure 3.13: Simulator closed-loop test 3**



### 3.1.5 Summary and Remarks

TABLE 4. Transient response results from Kingston Unit 9 plant

Test	PB	INT	Setpoint	Overshoot	Settling Time
1	35	1.7	8.0-9.0-8.0	~ 10%	~40s
2	25	1.7	8.0-9.0-8.0	~ 5%	~30s
3	18	1.7	8.0-9.0-8.0	~ 5%	~25s

TABLE 5. Transient results from Kingston Unit 9 simulator

Test	PB	INT	Setpoint	Overshoot	Settling Time
1	35	1.7	8.0-9.0-8.0	< 2%	48s
2	25	1.7	8.0-9.0-8.0	< 2%	37s
3	18	1.7	8.0-9.0-8.0	< 2%	28s

The tables above summarize the closed-loop transient response tests result for both the actual feedwater heater system and the simulator. It is noted that due to the graphical nature of the field information obtained from the actual system, only approximates of the overshoot and the settling time can be realized. It is not uncommon that there is a subtle difference in the overshoot characteristics since the simulator cannot capture all the prevailing heater dynamics especially those of higher orders. However, the magnitude of the settling time in both cases are quite similar.

The tests result states that we have the error interval of about 20 percents for the settling time and slightly higher for the percent overshoot. With careful considerations, we can then use the simulator as the test bed for our new controllers and assume that the transient response result will be within reasonable range of the actual system.

## 3.2 Controller Based on Feedback Linearization

### 3.2.1 Introduction

The problem considered here is the feedwater heater level control system. We develop in this section a systematic dynamic model of the system so as to include the nonlinear effects from the control valve and the pipe. In an existing power plant, the controller used to regulate the water level in the feedwater heater is a PI controller. We introduce a new nonlinear controller for the level regulation.

The performance of our nonlinear controllers is investigated and compared with the conventional PI-controller. It is first shown that the nonlinear controller results in a stable system. The simulation was first done assuming a single homogeneous phase model and then the effect of the two phase flow downstream of the valve was included. The effects of the system uncertainties were also examined. The simulation studies show that an order of magnitude improvement in the settling time can be obtained using our nonlinear controller while delivering uniformly better performance under a variety of operating conditions and modeling uncertainties.

### 3.2.2 Problem Statement

#### Assumptions

##### 1. The Dynamic Equation

The dynamic of the heater level can be represented in the simplest form as following,

$$A(h)\frac{dh}{dt} = Q_0 - Q(\theta, \Delta P) \quad (3.37)$$

where  $Q_0$  is the extraction flow rate,  $Q$  is the controlled flow rate,  $A(h)$  is the horizontal cross-sectional area of the heater,  $R$  is the heater radius, and  $L$  is the heater length.

## 2. The Nonlinearities

### Valve Nonlinearity

The relationship between the pressure drop and the flow rate is governed by the following equation.

$$\Delta P_{valve} = 3225.42 \frac{\rho}{C_v^2} Q^2 \quad (3.38)$$

### Pipe Nonlinearity

The pressure drop due to the turbulent flow through pipes can be described as followed.

$$\Delta P_{pipe} = \frac{0.316}{2A} \left[ \frac{\mu A}{\rho D} \right]^{0.25} \left[ \frac{L \rho}{D} \right] Q^2 \Big|_{pipe} \quad (3.39)$$

### Heater Cross-sectional Area

The area of the heater in the horizontal plane is a function of the heater level,  $h$ .

$$A(h) = 2\sqrt{2Rh - h^2}L \quad (3.40)$$

## The Control Flow Equation

$$\Delta P_{12} = \Delta P_{pipe1} + \Delta P_{valve} + \Delta P_{pipe2} \quad (3.41)$$

$$\text{where } \Delta P_{pipe1} = \frac{0.316}{2A} \left[ \frac{\mu A}{\rho D} \right]^{0.25} \left[ \frac{L \rho}{D} \right] Q^2 \Big|_{pipe1}, \quad (3.42)$$

$$\Delta P_{valve} = 3225.42 \frac{\rho}{C_v^2} Q^2, \quad (3.43)$$

$$\text{and } \Delta P_{pipe2} = \frac{0.316}{2A} \left[ \frac{\mu A}{\rho D} \right]^{0.25} \left[ \frac{L \rho}{D} \right] Q^2 \Big|_{pipe2}. \quad (3.44)$$

Based on the assumption that the pressure drop between heater 1 and heater 2 is a known constant at any instant and continuity law, the flow rate as a function of the valve opening can be obtained. The results at various pressure drops is then used to generate the data map.

## The Downstream Pressure Drop (Liquid-Vapor Phase)

$$-Dp = -DP_{FLO}\Phi^2_{FLO} + \frac{g \sin \theta}{v_m} + G^2 D(v_e) \quad (3.45)$$

$$\text{where } G^2 D(v_e) = -\frac{G^2}{G_C} Dp + G^2 \frac{(v_G - v_L)}{(h_G - h_L)} \Omega D(q + F) \quad (3.46)$$

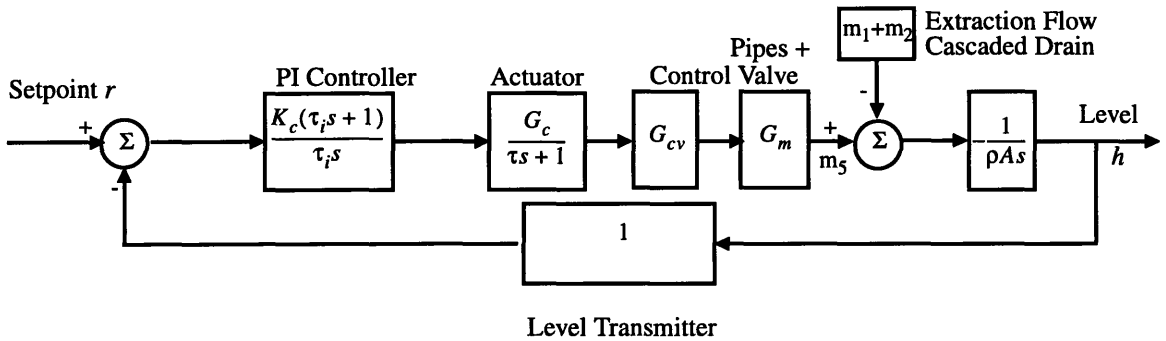
If the heat loss to the environment and the frictional energy dissipated are insignificant then  $D(q + F) = 0$ .

$$\text{And } -Dp = \frac{-DP_{FLO}\Phi^2_{FLO} + \frac{g \sin \theta}{v_m}}{1 - \frac{G^2}{G_C}} \quad (3.47)$$

$$\text{where } \Phi^2_{FLO} = 1 + (\Gamma^2 - 1) \left[ Bx^{\frac{(2-n)}{2}} (1-x)^{\frac{(2-n)}{2}} + x^{(2-n)} \right], \quad (3.48)$$

$$\Gamma^2 = \left( \frac{\mu_G}{\mu_L} \right)^{n v_G} \frac{v_L}{v_G}, \quad n = \frac{\log \left( \frac{\lambda_{LO}}{\lambda_{GO}} \right)}{\log \left( \frac{Re_{GO}}{Re_{LO}} \right)}, \quad B \text{ defined in Table 1, and } x \text{ is the mass dryness fraction}$$

### 3.2.3 The Linear PI Controller



Since the valve characteristic is nonlinear over the plant load range, the optimum controller gains would not be constant. The relationship between the flow rate and the valve opening is selected to give the closest to linear response. PI controller has to be tuned conservatively to cover the load range of the plant.

Determine the transfer function of the plant.

$$\frac{R(s)}{H(s)} = \frac{-K_c G_c G_{cv} G_m (\tau_i s + 1)}{(\tau_i s)(\tau s + 1)(\rho A s) + K_c G_c G_{cv} G_m (\tau_i s + 1)} \quad (3.49)$$

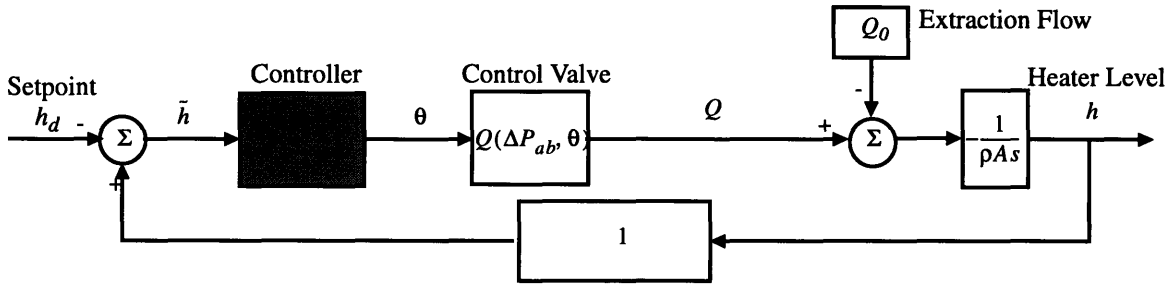
Let  $K_c G_c G_{cv} G_m = C$ .

$$\frac{R(s)}{H(s)} = \frac{C \tau_i s + C}{\rho A \tau_i \tau s^3 + \rho A \tau_i s^2 + C \tau_i s + C} \quad (3.50)$$

Using ITAE criterion,  $K_c$  and  $\tau_i$  can be found.

The gains found are only suitable sufficiently near the operating point. In order to cope with the uncertainties, the gains are selected so as to maintain the system stability. Typical proportional band is 35 and integral time is 1.7.

### 3.2.4 Controller Based on Feedback Linearization



The dynamic model:

$$A(h) \frac{dh}{dt} = Q_0 - Q(\theta, \Delta P)$$

where  $A(h)$ ,  $Q_0$ , and  $Q$  are the area, extraction flow, and drain valve flow, respectively.

#### Known Extraction Flow

By Choosing the control input as  $Q(\theta, \Delta P) = Q_0 - A(h)v = Q_0 + (\alpha)A(h)\bar{h} + \beta A(h) \int \bar{h} dt$ ,

the equivalent dynamic equation is now  $\dot{h} + \alpha \bar{h} + \beta \int \bar{h} dt = 0$  where  $\bar{h} = h - h_d$  and  $\dot{h} = \dot{\bar{h}}$ .

That is

$$\dot{\bar{h}} + \alpha \bar{h} + \beta \int \bar{h} dt = 0$$

which implies that as  $t \rightarrow \infty$ ,  $\bar{h} \rightarrow 0$  if  $\alpha, \beta$  are strictly positive.

#### Unknown Extraction Flow

By Choosing the control input as  $Q(\theta, \Delta P) = -A(h)v = (\alpha)A(h)\bar{h} + \beta A(h) \int \bar{h} dt$ ,

the equivalent dynamic equation is now  $\dot{h} + \alpha h + \beta \int \dot{h} dt = \frac{Q_0}{A(h)}$  where  $A(h) = 2L\sqrt{2Rh - h^2}$

### 1. Constant Cross-sectional Area

Let cross-sectional area be  $A$ .

The dynamic equation is

$$\dot{h} + \alpha \bar{h} + \beta \int \dot{h} dt = \frac{Q_0}{A}.$$

By differentiating,

$$\ddot{h} + \alpha \dot{h} + \beta \dot{h} = 0$$

which implies that as  $t \rightarrow \infty$ ,  $\bar{h} \rightarrow 0$  if  $\alpha, \beta$  are strictly positive.

### 2. Time-varying Cross-sectional Area

The cross-sectional area can be represented as  $A(\bar{h}) = 2L\sqrt{2R(\bar{h} + h_d) - (\bar{h} + h_d)^2}$ .

The dynamic equation is now

$$\dot{\bar{h}} + \alpha \bar{h} + \beta \int \dot{\bar{h}} dt = \frac{Q_0}{A(\bar{h})}.$$

$$\text{By differentiating, } \ddot{\bar{h}} + \alpha \dot{\bar{h}} + \beta \dot{\bar{h}} = - \left( \frac{Q_0}{A^2(\bar{h})} \right) \left( \frac{d}{d\bar{h}} A(\bar{h}) \right) \dot{\bar{h}} = - \left( \frac{2Q_0(R - \bar{h} - h_d)}{4L(2R(\bar{h} + h_d) - (\bar{h} + h_d)^2)^{\frac{3}{2}}} \right) \dot{\bar{h}}. \quad (3.51)$$

$$\text{Let } \left( \frac{2Q_0(R - \bar{h} - h_d)}{4L(2R(\bar{h} + h_d) - (\bar{h} + h_d)^2)^{\frac{3}{2}}} \right) = f(\bar{h}), \quad (3.52)$$

$$\text{and } \ddot{\bar{h}} + [\alpha + f(\bar{h})] \dot{\bar{h}} + \beta \dot{\bar{h}} = 0. \quad (3.53)$$

The expression,  $f(\bar{h})$ , is strictly positive when  $R > (\bar{h} + h_d)$  or  $R > h$  which is the case for this control problem. By choosing the appropriate  $\alpha$  and strictly positive  $\beta$ , we now have a closed loop system which guarantees that  $\bar{h} \rightarrow 0$  as  $t \rightarrow \infty$ .

It can also be shown using Lyapunov theory.

For this system, the Lyapunov function is  $V = \frac{1}{2}\dot{x}^2 + \int_0^x \beta y dy = \frac{1}{2}\dot{x}^2 + \frac{1}{2}\beta x^2$ .

The minimum of this function is at  $x = 0$ ;  $\dot{x} = 0$

The time-derivative of  $V$  is

$$\dot{V} = -(\alpha + f(\bar{h}))\dot{x}^2 \leq 0$$

which can be thought of as representing the power dissipated in the system. By hypothesis,  $\dot{V} = 0$  only if  $\dot{x} = 0$  implies that  $\ddot{h} = -\beta \dot{h}$  which is nonzero as long as  $\dot{h} \neq 0$ . Thus the



system cannot get “stuck” anywhere other than  $h = 0$ . Using the Local Invariant Set theorem, the origin is proved to be a locally asymptotically stable point. Furthermore,  $\int_0^x \beta y dy$  is unbounded as  $|\tilde{h}| \rightarrow \infty$ , and  $V$  is a radially unbounded function and the equilibrium point at the origin is globally asymptotically stable according to the Global Invariant Set theorem.

The valve opening can be determined by incorporating the required flow rate into our valve characteristic model. Assuming that the instantaneous pressure drop between heater 1 and 2 is known, the flow rate can now be represented by an  $n$ -polynomial function of valve opening described below. The valve opening is obtained by solving the function. In the simulation, we assumed that the nominal pressure drop between heater 1 and 2 is 300 psi. And the flow rate can be represented as a function of the valve opening as followed:

$$Q = a\theta^5 + b\theta^4 + c\theta^3 + d\theta^2 + e\theta + f \quad (3.54)$$

where  $a = 0.00000000468362$

$b = 0.00000058202359$

$c = 0.00001816170982$

$d = 0.00366774086313$

$e = 0.00447504781796$

$f = 0.01500049486870$

In the case where the pressure drop is accessible on-line, the appropriate polynomial function can be developed to determine the valve opening accordingly.

## **The Plant Uncertainties**

### **1. Pressure Drop**

The pressure drop between heater 1 and heater 2 is dependent of the plant operating condition. The pressure drop is used to determine the valve-pipe flow characteristic surface (the flow rate at different valve opening and pressure drop). In the simulations, the nominal pressure drop is 300psi.

### **2. Extraction Flow**

The extraction flow is also dependent of the plant load. The nominal value is 0.4444 ft<sup>3</sup>/s equivalent of the liquid.

### **3. Valve Flow Coefficient**

The valve flow coefficient obtained experimentally is considered to be more accurate and applicable to the plant than the data supplied by the manufacturer. This constitutes a source of uncertainties in the valve model.

### 3.2.5 Simulation Results

#### Simulation Case#1: Exact Knowledge of the System

TABLE 6. Conventional PI-controller

Proportional gain (Kp)	Integral gain (Ki)	Modeling error	Valve characteristic	Design extraction flow	Actual extraction flow	Design pressure drop	Actual pressure drop
100/35	(100/35)/(1.7*60)	No.	Linear Approximation.	0.4444 ft <sup>3</sup> /s	0.4444 ft <sup>3</sup> /s	300 psi	300psi

TABLE 7. Feedback linearizing P-controller

proportional gain (alpha)	Modeling error	Valve characteristic	Design extraction flow	Actual extraction flow	Design pressure drop	Actual pressure drop
0.1	No.	5th-order least square approximation.	0.4444 ft <sup>3</sup> /s	0.4444 ft <sup>3</sup> /s	300 psi	300psi

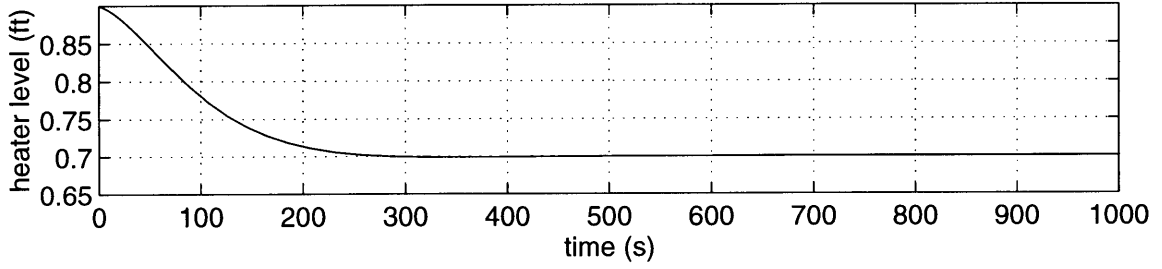
TABLE 8. Feedback linearizing PI-controller

Proportional gain (alpha)	Integral gain (beta)	Modeling error	Valve characteristic	Design extraction flow	Actual extraction flow	Design pressure drop	Actual pressure drop
0.1	0.01	No.	5th-order least square approximation	0.4444 ft <sup>3</sup> /s	0.4444 ft <sup>3</sup> /s	300 psi	300psi

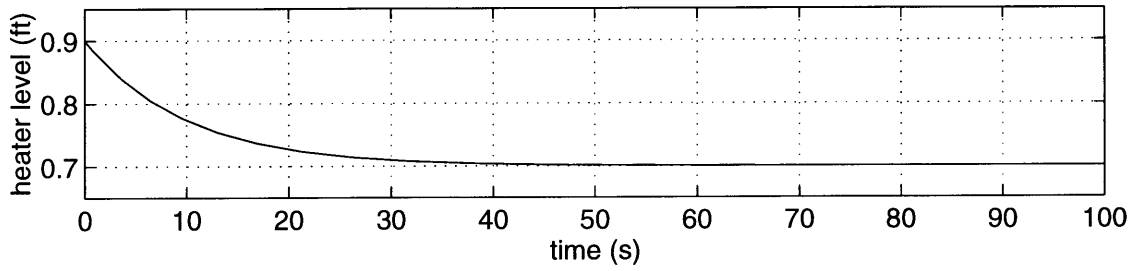
#### Table of Results

Controller	Settling Time (seconds)	Overshoot	Steady-State Error
Conventional PI-controller	207.17	Yes	0
Nonlinear P-controller	31.85	No	0
Nonlinear PI-controller	42.62	Yes	0

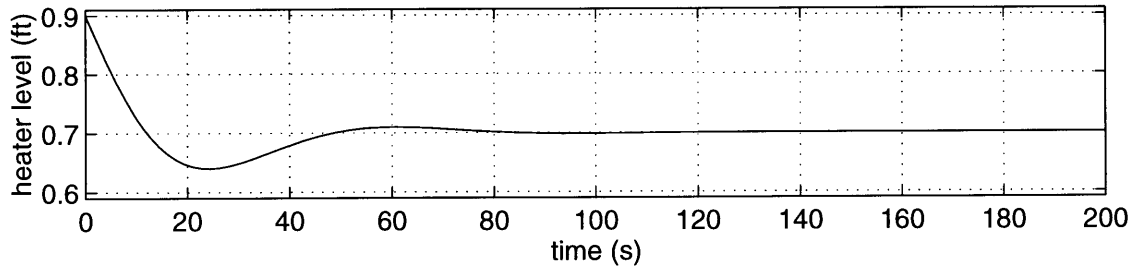
Conventional PI controller:

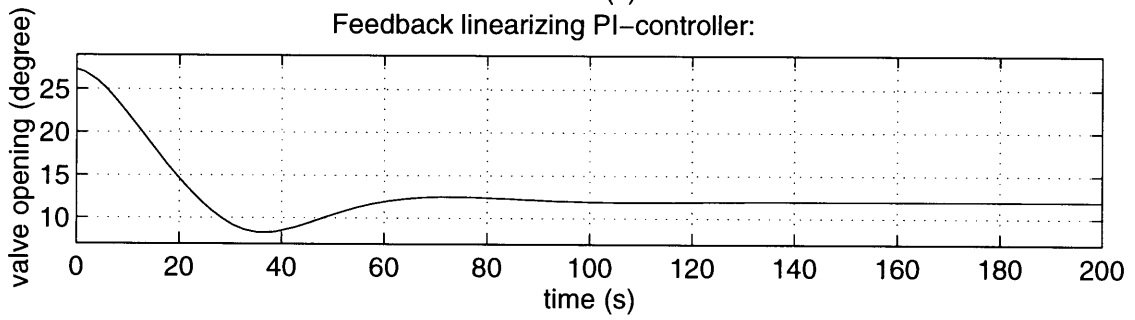
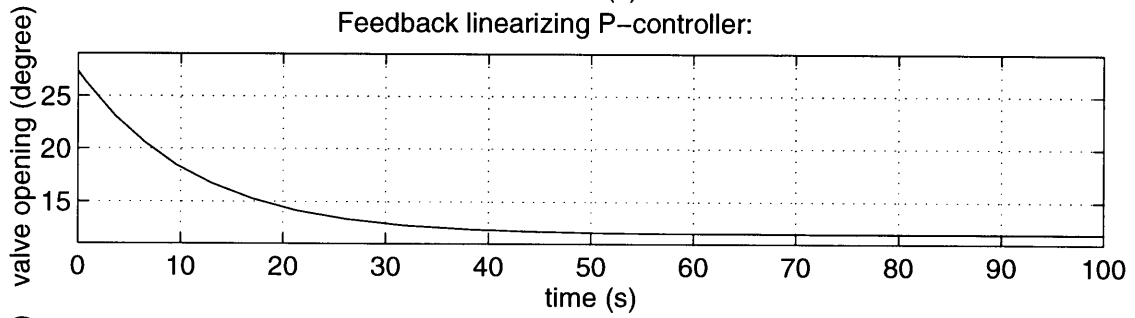
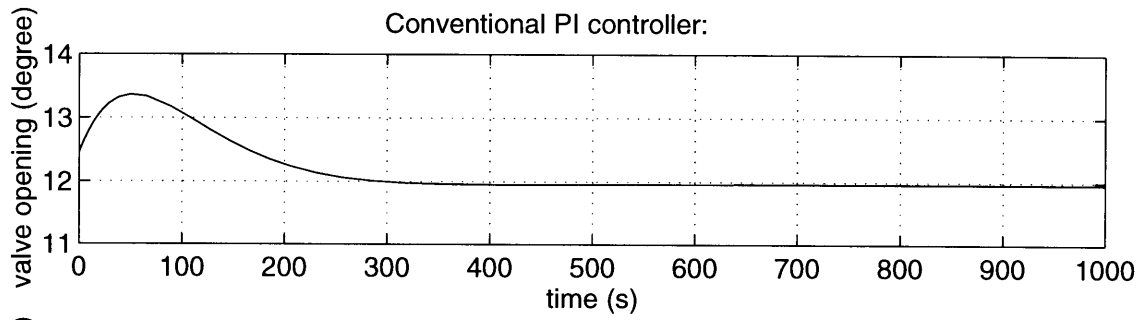


Feedback linearizing P-controller:

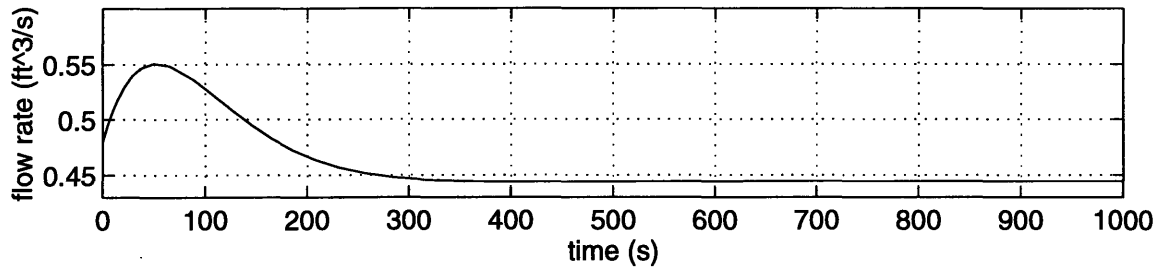


Feedback linearizing PI-controller:

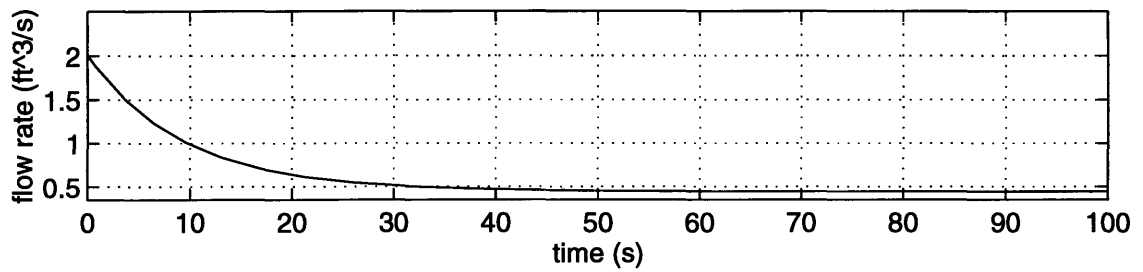




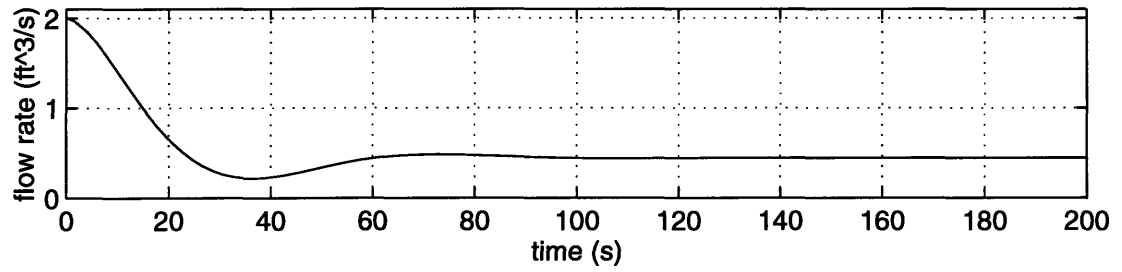
Conventional PI controller:



Feedback linearizing P-controller:



Feedback linearizing PI-controller:



## Simulation Case#2: Unknown extraction flow rate.

TABLE 9. Conventional PI-controller

Proportional gain (Kp)	Integral gain (Ki)	Modeling error	Valve characteristic	Design extraction flow	Actual extraction flow	Design pressure drop	Actual pressure drop
100/35	(100/35)/ (1.7*60)	No.	Linear Approximation.	0.0 ft <sup>3</sup> /s	11.5x0.4444 ft <sup>3</sup> /s	300 psi	300psi

TABLE 10. Feedback linearizing P-controller

proportional gain (alpha)	Modeling error	Valve characteristic	Design extraction flow	Actual extraction flow	Design pressure drop	Actual pressure drop
0.1	No.	5th-order least square approximation.	0.0 ft <sup>3</sup> /s	11.5x0.4444 ft <sup>3</sup> /s	300 psi	300psi

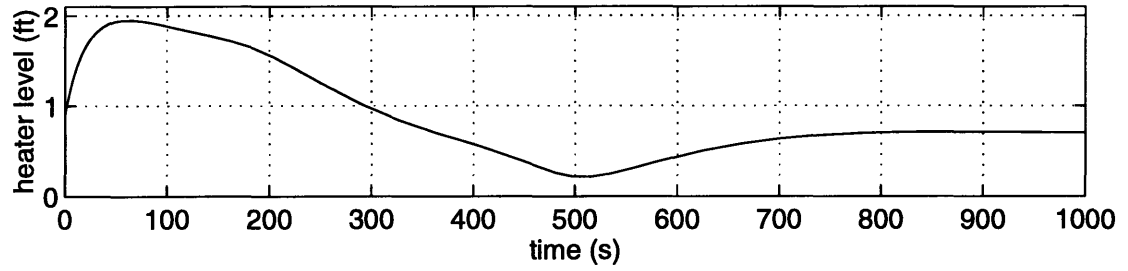
TABLE 11. Feedback linearizing PI-controller

Proportional gain (alpha)	Integral gain (beta)	Modeling error	Valve characteristic	Design extraction flow	Actual extraction flow	Design pressure drop	Actual pressure drop
0.1	0.01	No.	5th-order least square approximation	0.0 ft <sup>3</sup> /s	11.5x0.4444 ft <sup>3</sup> /s	300 psi	300psi

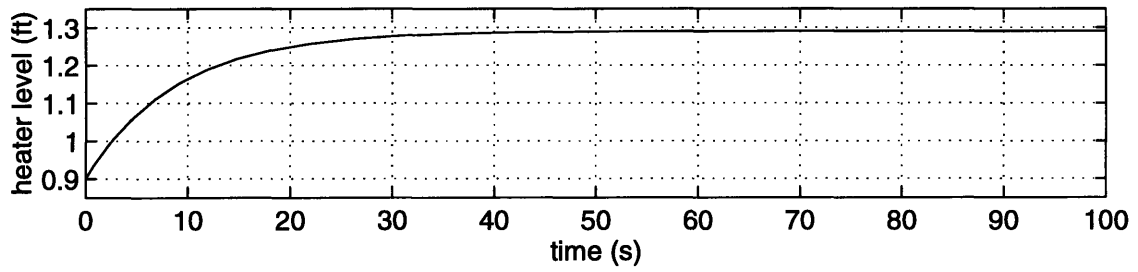
### Table of Results

Controller	Settling Time (seconds)	Overshoot	Steady-State Error
Conventional PI-controller	930.31	Yes	0
Nonlinear P-controller	N/A	No	Yes
Nonlinear PI-controller	62.26	Yes	0

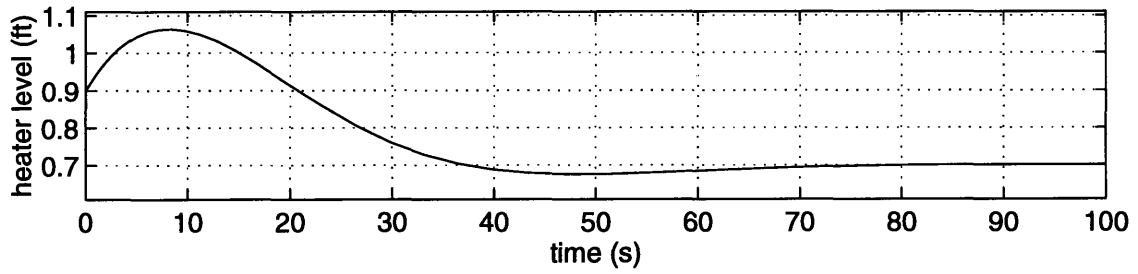
Conventional PI controller:



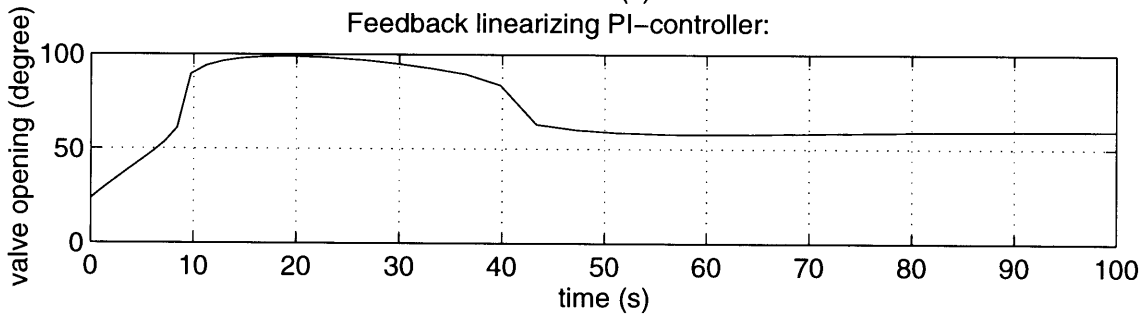
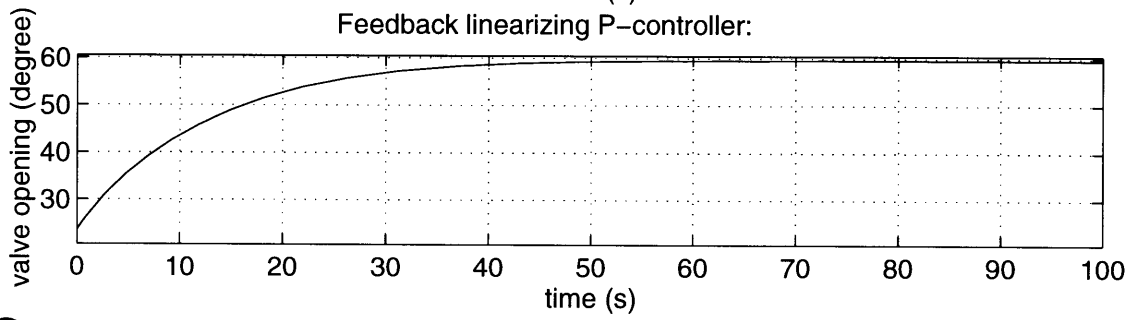
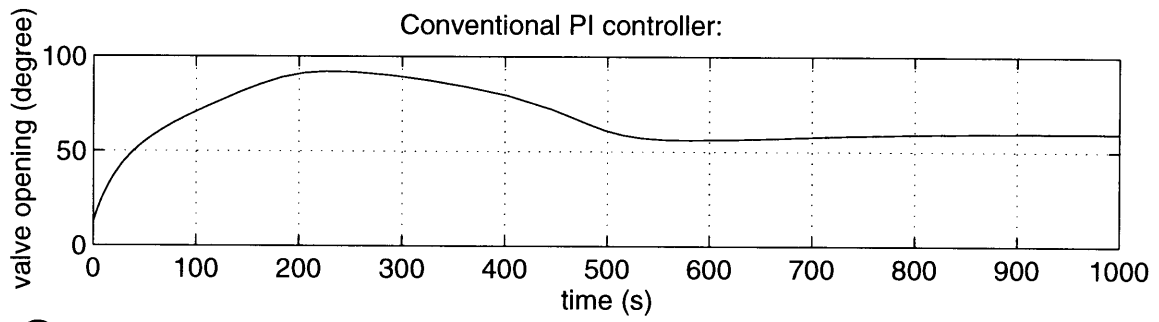
Feedback linearizing P-controller:



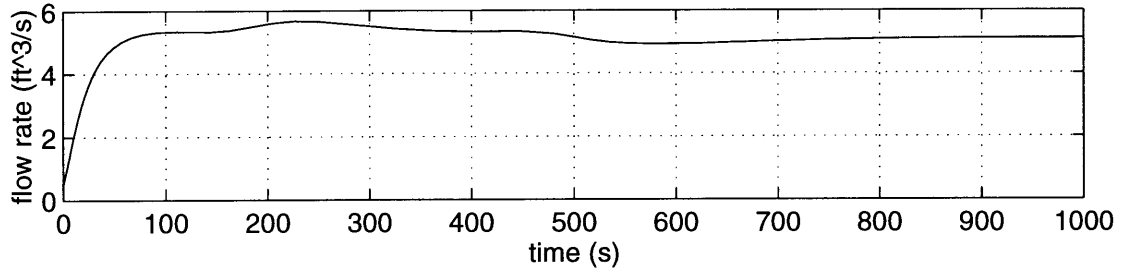
Feedback linearizing PI-controller:



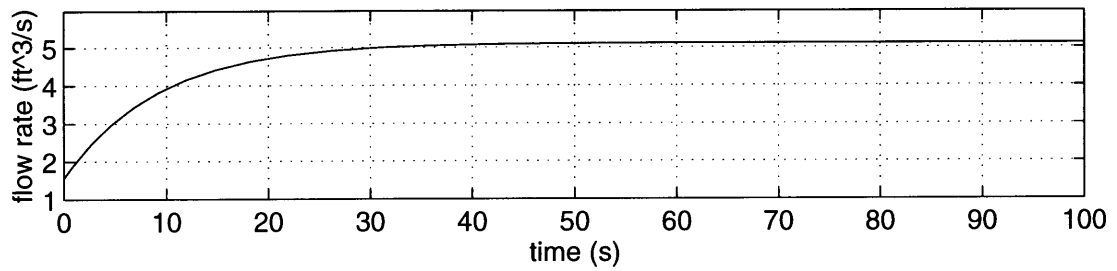




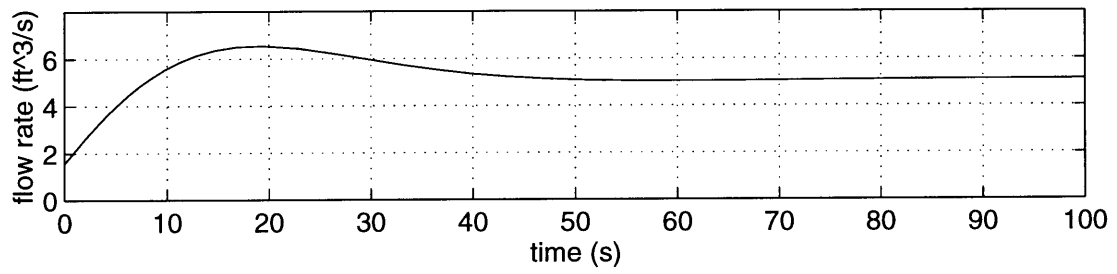
Conventional PI controller:



Feedback linearizing P-controller:



Feedback linearizing PI-controller:



### Simulation Case#3: With Model uncertainty

TABLE 12. Conventional PI-controller

Proportional gain (Kp)	Integral gain (Ki)	Modeling error	Valve characteristic	Design extraction flow	Actual extraction flow	Design pressure drop	Actual pressure drop
100/35	(100/35)/ (1.7*60)	Yes.	Linear Approximation.	0.4444 ft <sup>3</sup> /s	0.4444 ft <sup>3</sup> /s	300 psi	300psi

TABLE 13. Feedback linearizing P-controller

proportional gain (alpha)	Modeling error	Valve characteristic	Design extraction flow	Actual extraction flow	Design pressure drop	Actual pressure drop
0.1	Yes.	5th-order least square approximation.	0.4444 ft <sup>3</sup> /s	0.4444 ft <sup>3</sup> /s	300 psi	300psi

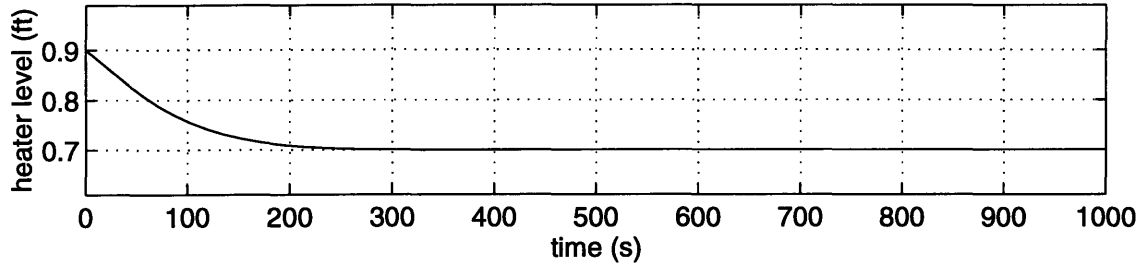
TABLE 14. Feedback linearizing PI-controller

Proportional gain (alpha)	Integral gain (beta)	Modeling error	Valve characteristic	Design extraction flow	Actual extraction flow	Design pressure drop	Actual pressure drop
0.1	0.01	Yes.	5th-order least square approximation	0.4444 ft <sup>3</sup> /s	0.4444 ft <sup>3</sup> /s	300 psi	300psi

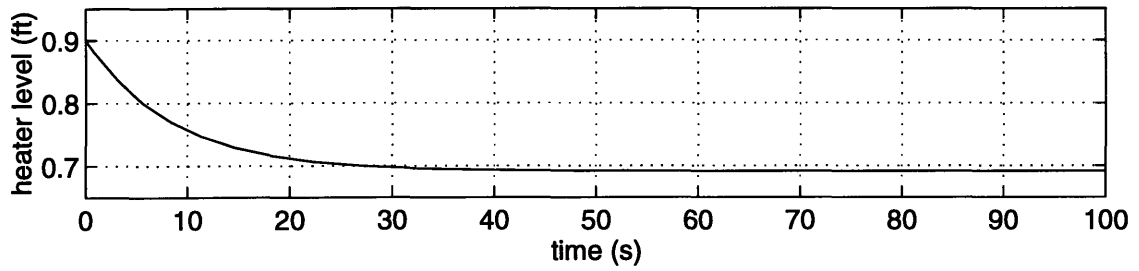
#### Table of Results

Controller	Settling Time (seconds)	Overshoot	Steady-State Error
Conventional PI-controller	193.89	No	0
Nonlinear P-controller	22.38	No	Yes
Nonlinear PI-controller	41.36	Yes	0

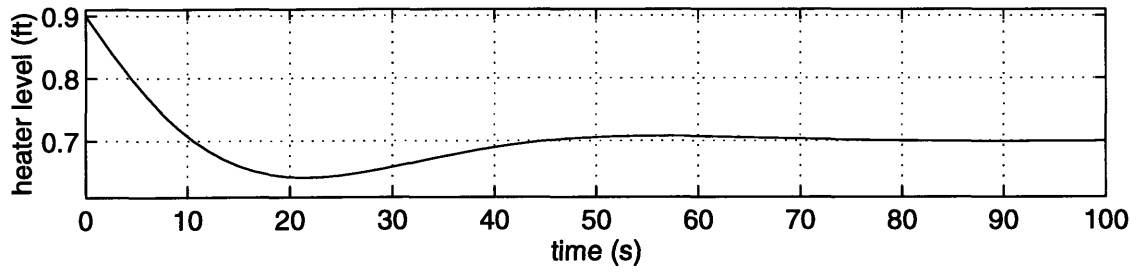
Conventional PI controller:

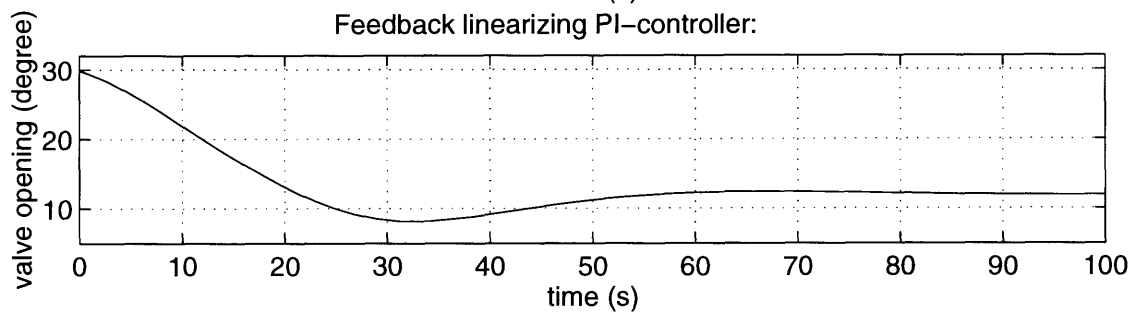
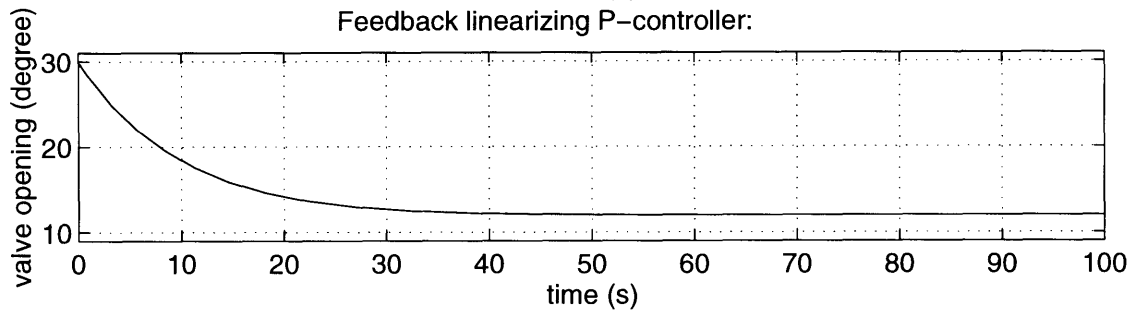
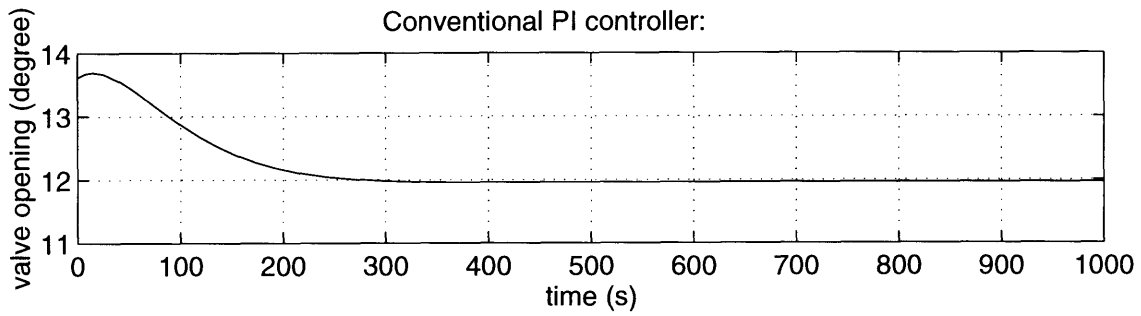


Feedback linearizing P-controller:

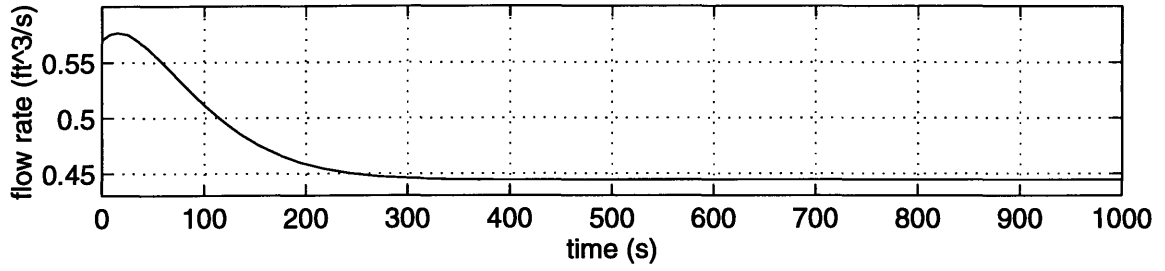


Feedback linearizing PI-controller:

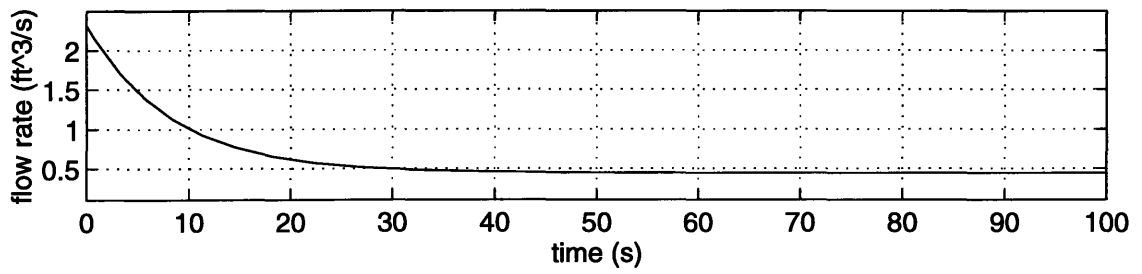




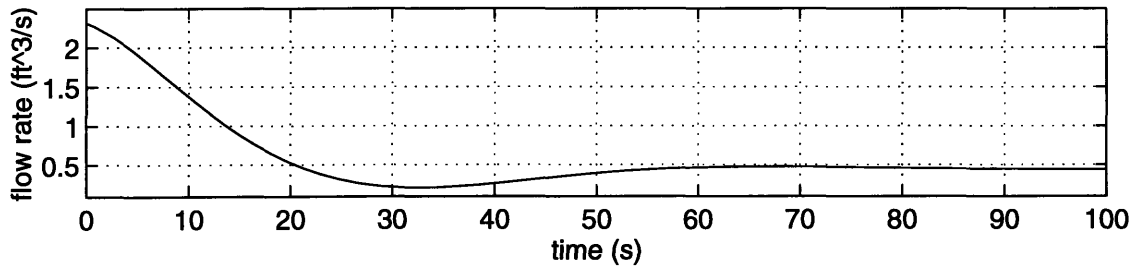
Conventional PI controller:



Feedback linearizing P-controller:



Feedback linearizing PI-controller:



## Simulation Case#4: Unknown pressure drop.

TABLE 15. Conventional PI-controller

Proportional gain (Kp)	Integral gain (Ki)	Modeling error	Valve characteristic	Design extraction flow	Actual extraction flow	Design pressure drop	Actual pressure drop
100/35	(100/35)/ (1.7*60)	No.	Linear Approximation.	0.4444 ft <sup>3</sup> /s	0.4444 ft <sup>3</sup> /s	300 psi	1000psi

TABLE 16. Feedback linearizing P-controller

proportional gain (alpha)	Modeling error	Valve characteristic	Design extraction flow	Actual extraction flow	Design pressure drop	Actual pressure drop
0.1	No.	5th-order least square approximation.	0.4444 ft <sup>3</sup> /s	0.4444 ft <sup>3</sup> /s	300 psi	1000psi

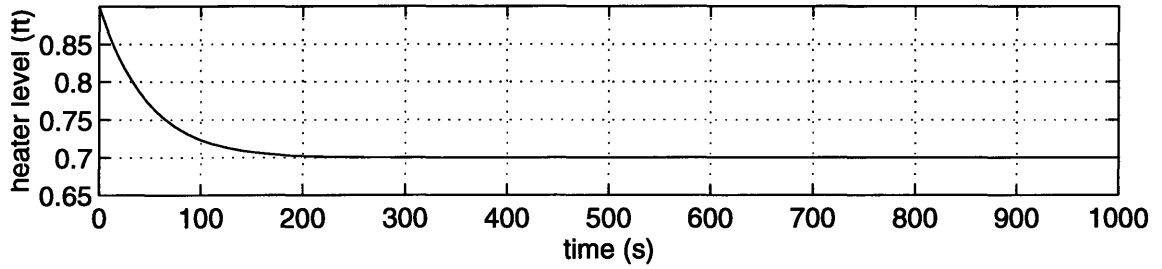
TABLE 17. Feedback linearizing PI-controller

Proportional gain (alpha)	Integral gain (beta)	Modeling error	Valve characteristic	Design extraction flow	Actual extraction flow	Design pressure drop	Actual pressure drop
0.1	0.01	No.	5th-order least square approximation	0.4444 ft <sup>3</sup> /s	0.4444 ft <sup>3</sup> /s	300 psi	1000psi

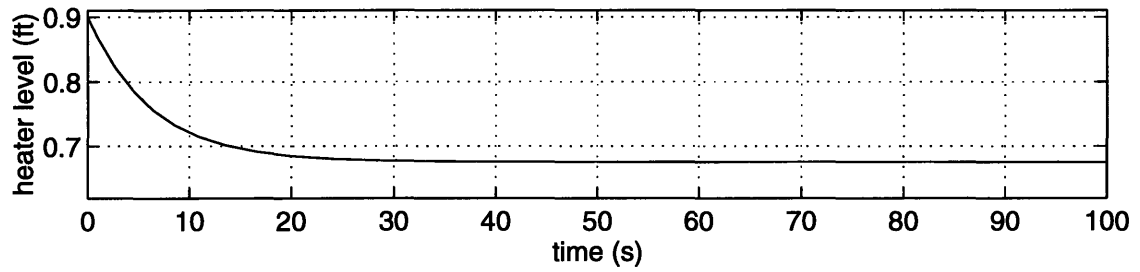
### Table of Results

Controller	Settling Time (seconds)	Overshoot	Steady-State Error
Conventional PI-controller	125.36	No	0
Nonlinear P-controller	N/A	No	Yes
Nonlinear PI-controller	33.11	Yes	0

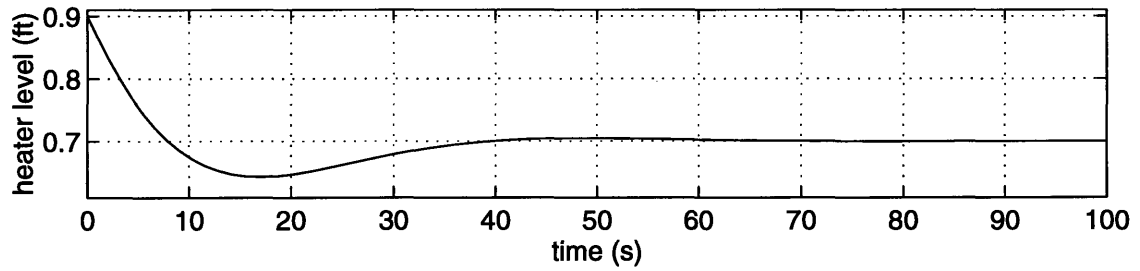
Conventional PI controller:



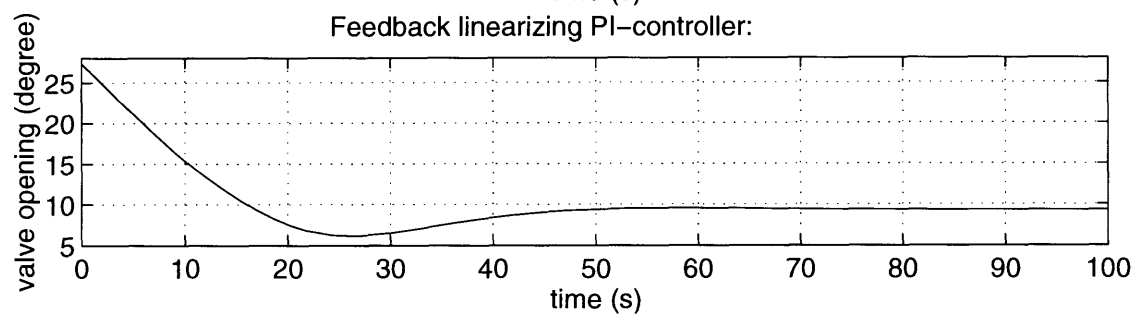
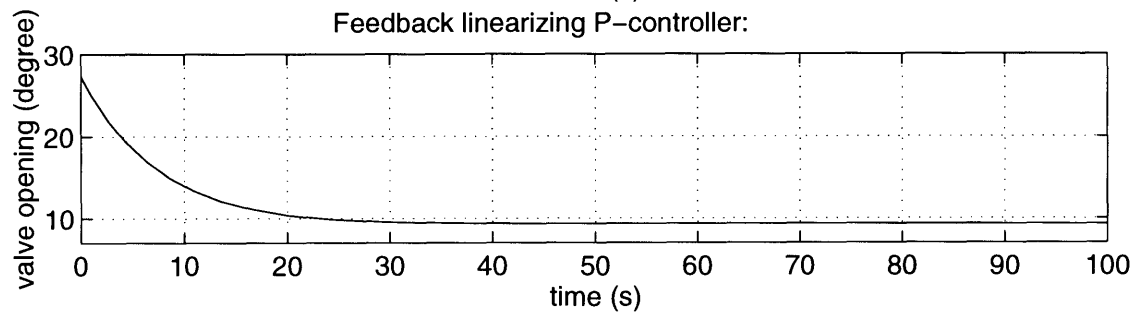
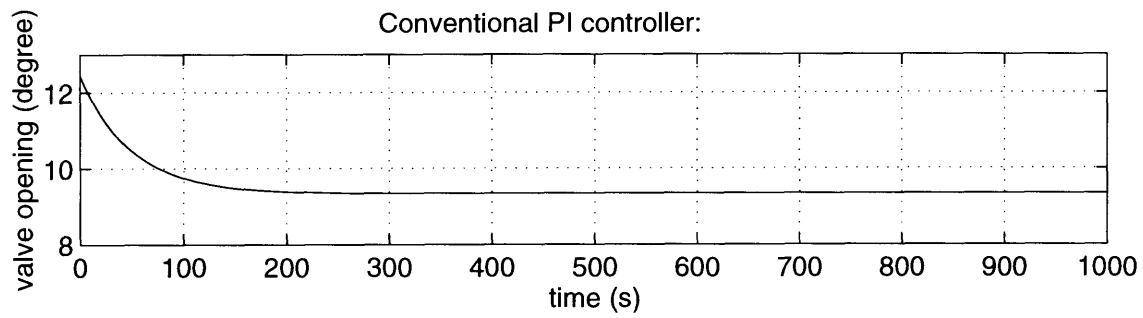
Feedback linearizing P-controller:



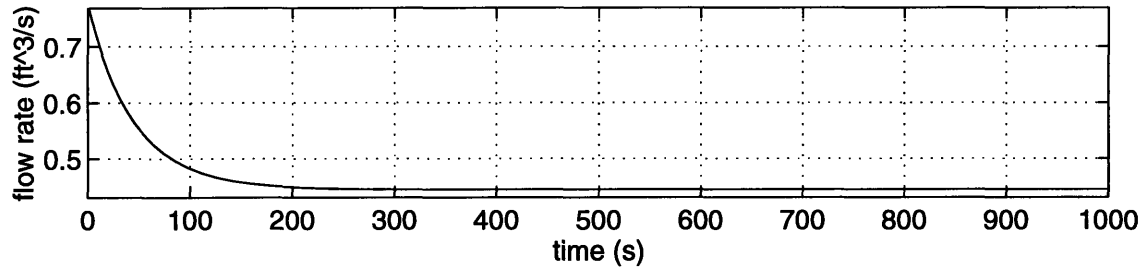
Feedback linearizing PI-controller:



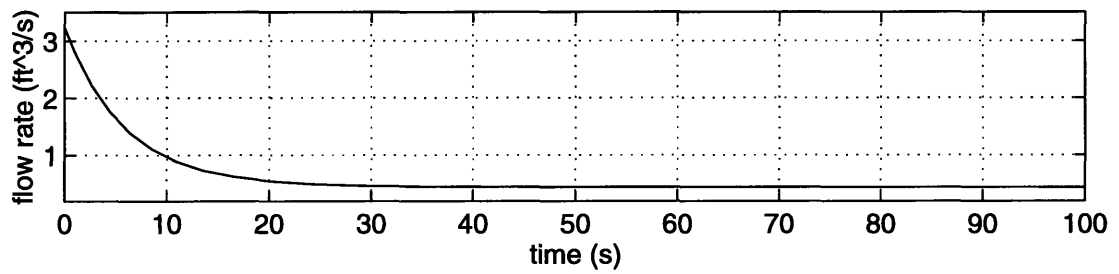




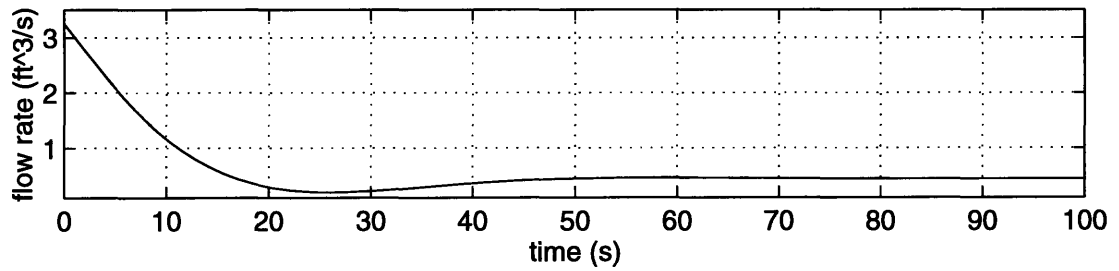
Conventional PI controller:



Feedback linearizing P-controller:



Feedback linearizing PI-controller:



## Simulation Case#5: Unknown pressure drop, extraction flow. With model uncertainty.

TABLE 18. Conventional PI-controller

Proportional gain (Kp)	Integral gain (Ki)	Modeling error	Valve characteristic	Design extraction flow	Actual extraction flow	Design pressure drop	Actual pressure drop
100/35	(100/35)/(1.7*60)	Yes.	Linear Approximation.	0.0 ft <sup>3</sup> /s	0.4444 ft <sup>3</sup> /s	300 psi	1000psi

TABLE 19. Feedback linearizing P-controller

proportional gain (alpha)	Modeling error	Valve characteristic	Design extraction flow	Actual extraction flow	Design pressure drop	Actual pressure drop
0.1	Yes.	5th-order least square approximation.	0.0 ft <sup>3</sup> /s	0.4444 ft <sup>3</sup> /s	300 psi	1000psi

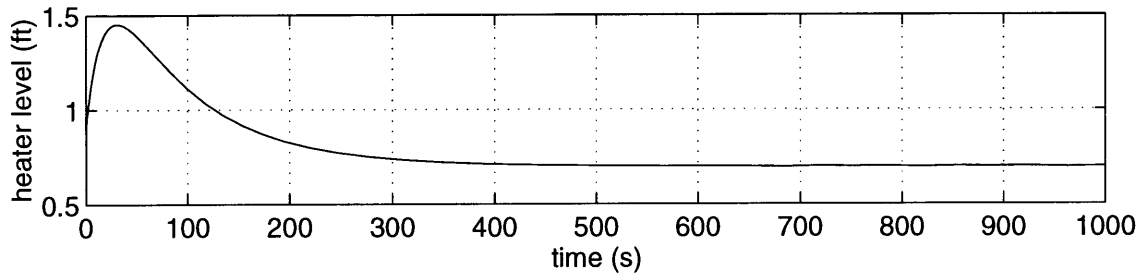
TABLE 20. Feedback linearizing PI-controller

Proportional gain (alpha)	Integral gain (beta)	Modeling error	Valve characteristic	Design extraction flow	Actual extraction flow	Design pressure drop	Actual pressure drop
0.1	0.01	Yes.	5th-order least square approximation	0.0 ft <sup>3</sup> /s	0.4444 ft <sup>3</sup> /s	300 psi	1000psi

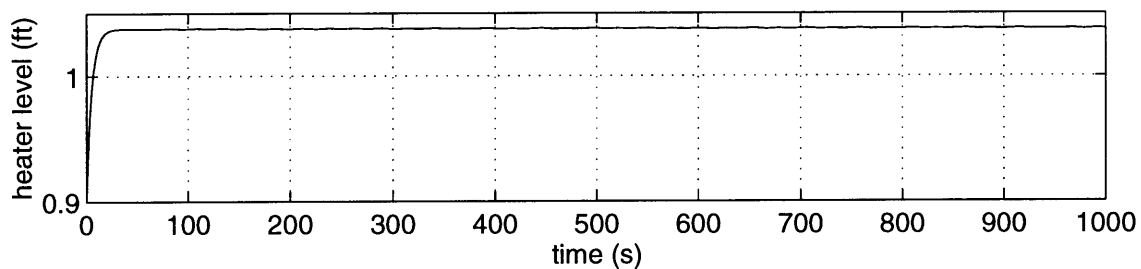
### Table of Results

Controller	Settling Time (seconds)	Overshoot	Steady-State Error
Conventional PI-controller	392.98	Yes	0
Nonlinear P-controller	N/A	No	Yes
Nonlinear PI-controller	30.69	Yes	0

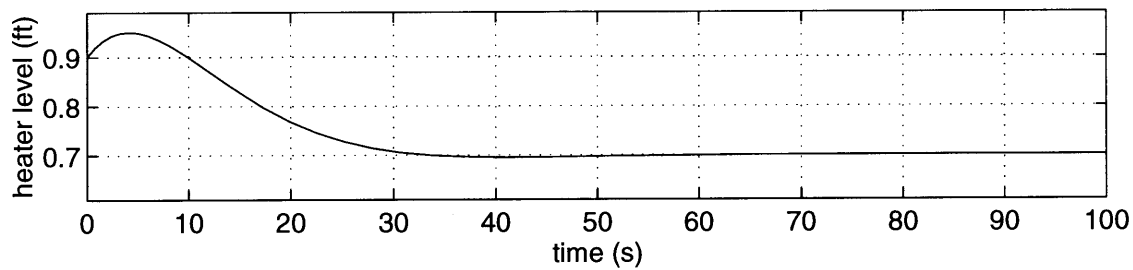
Conventional PI controller:

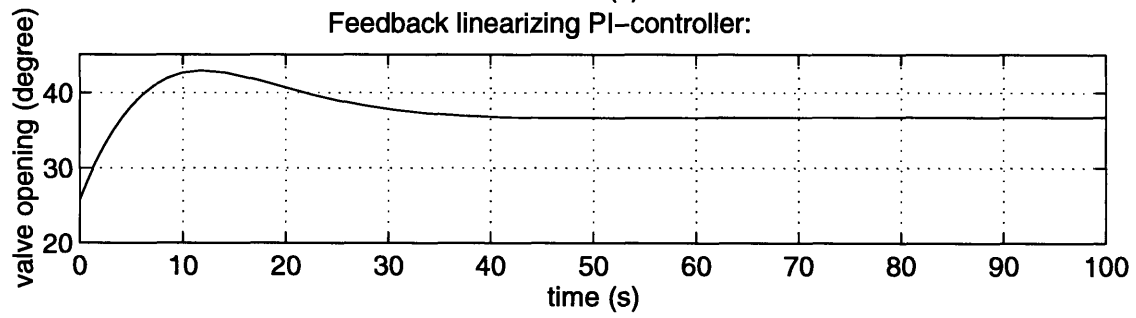
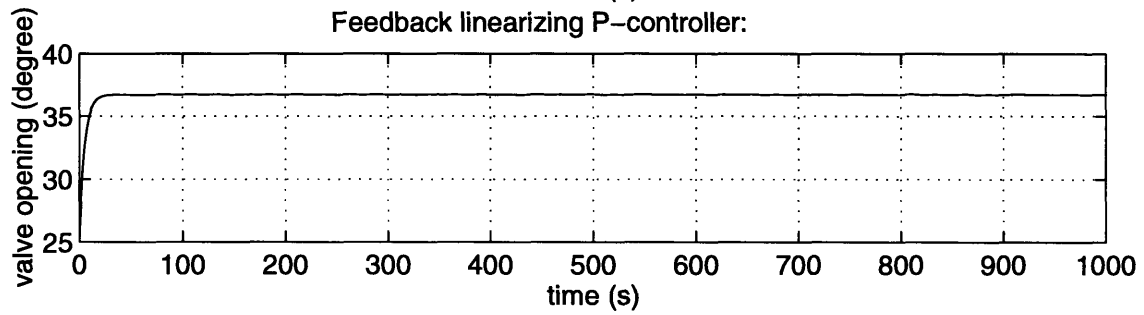
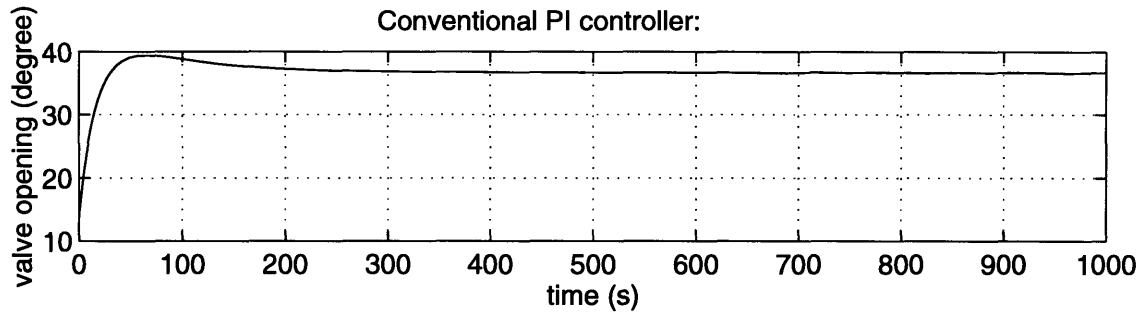


Feedback linearizing P-controller:

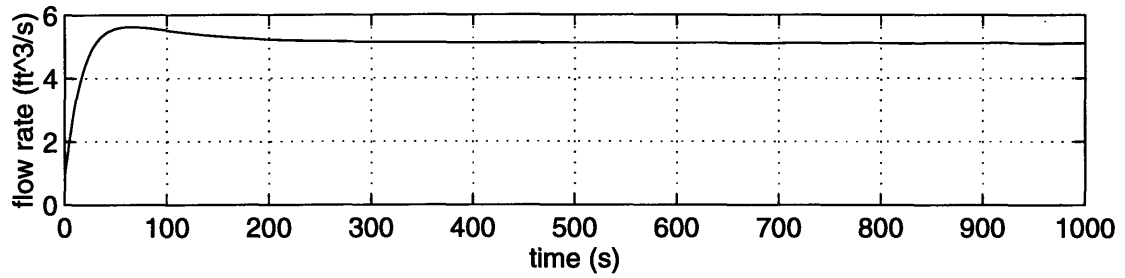


Feedback linearizing PI-controller:

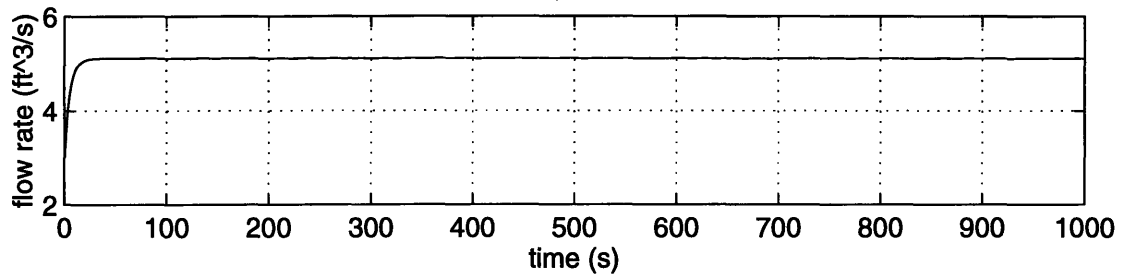




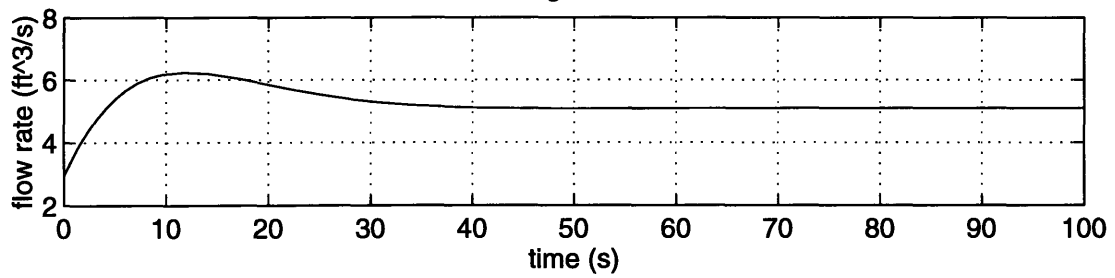
Conventional PI controller:



Feedback linearizing P-controller:



Feedback linearizing PI-controller:



**Simulation Case#6 [Using two phase flow model for the downstream section of the pipe]: Unknown pressure drop, extraction flow. With model uncertainty.**

TABLE 21. Conventional PI-controller

Proportional gain (Kp)	Integral gain (Ki)	Modeling error	Valve characteristic	Design extraction flow	Actual extraction flow	Design pressure drop	Actual pressure drop
100/35	(100/35)/ (1.7*60)	Yes.	Linear Approximation.	0.0 ft <sup>3</sup> /s	11.5*0.444 4 ft <sup>3</sup> /s	300 psi	1000psi

TABLE 22. Feedback linearizing P-controller

proportional gain (alpha)	Modeling error	Valve characteristic	Design extraction flow	Actual extraction flow	Design pressure drop	Actual pressure drop
0.1	Yes.	5th-order least square approximation.	0.0 ft <sup>3</sup> /s	11.5*0.444 4 ft <sup>3</sup> /s	300 psi	1000psi

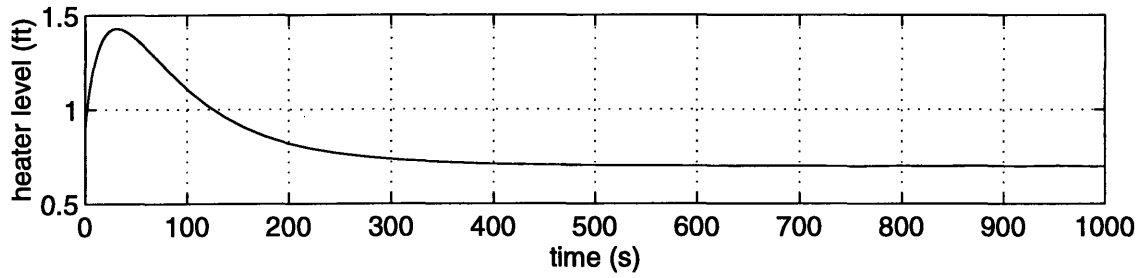
TABLE 23. Feedback linearizing PI-controller

Proportional gain (alpha)	Integral gain (beta)	Modeling error	Valve characteristic	Design extraction flow	Actual extraction flow	Design pressure drop	Actual pressure drop
0.1	0.01	Yes.	5th-order least square approximation	0.0 ft <sup>3</sup> /s	11.5*0.444 4 ft <sup>3</sup> /s	300 psi	1000psi

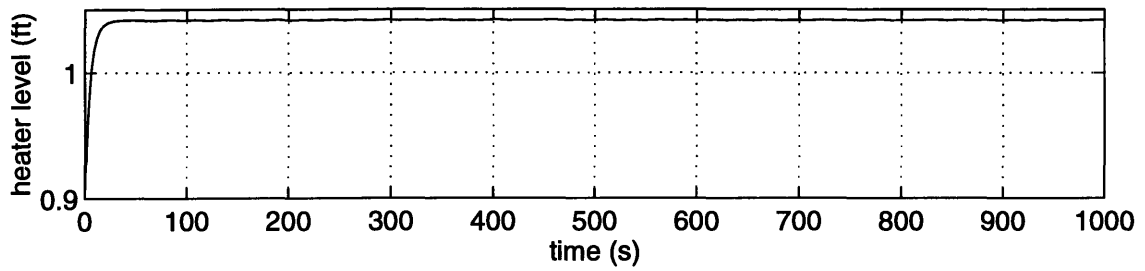
**Table of Results**

Controller	Settling Time (seconds)	Percent Overshoot	Steady-State Error
Conventional PI-controller	395.76	Yes	0
Nonlinear P-controller	N/A	No	Yes
Nonlinear PI-controller	28.30	Yes	0

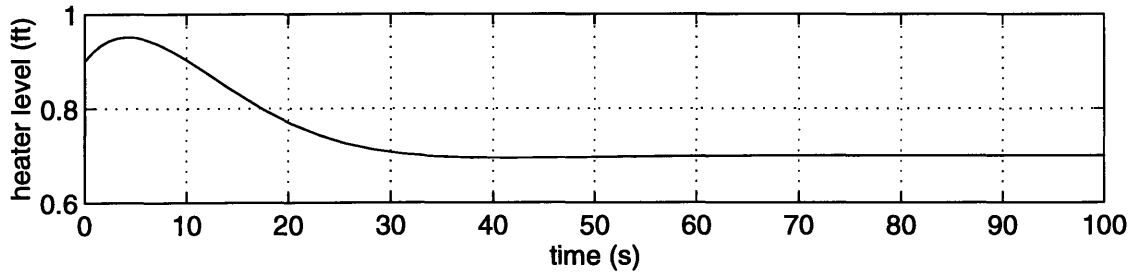
Conventional PI-controller[two phase flow]:



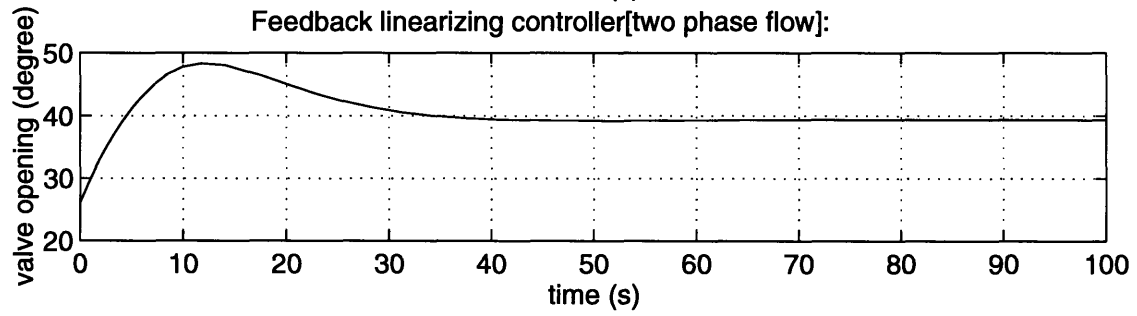
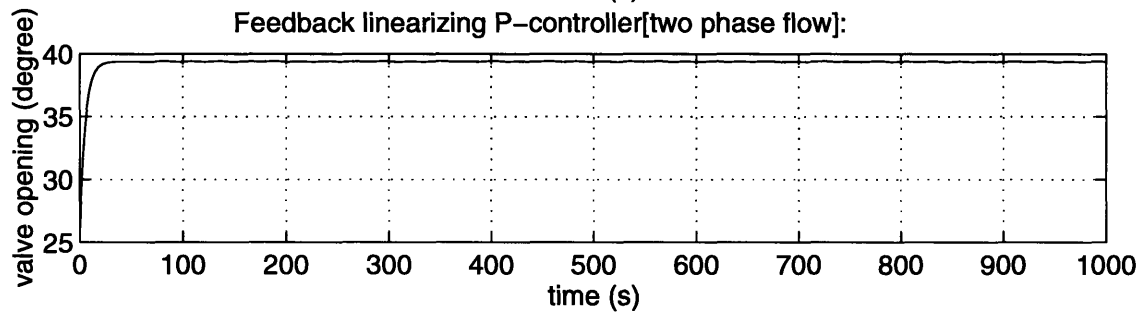
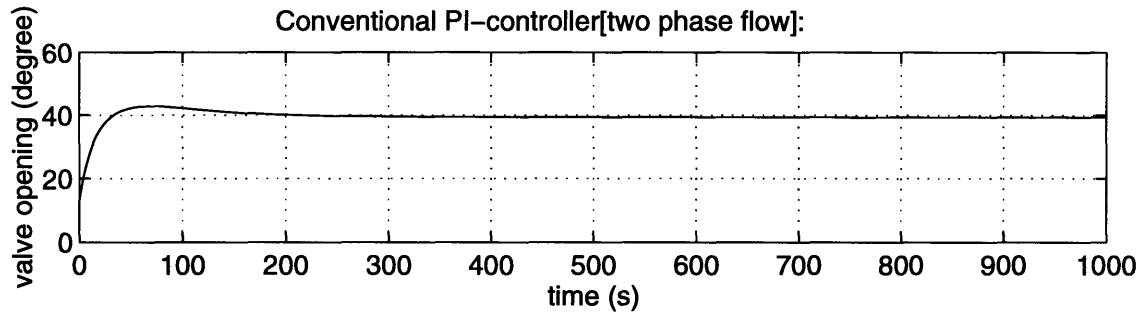
Feedback linearizing P-controller[two phase flow]:



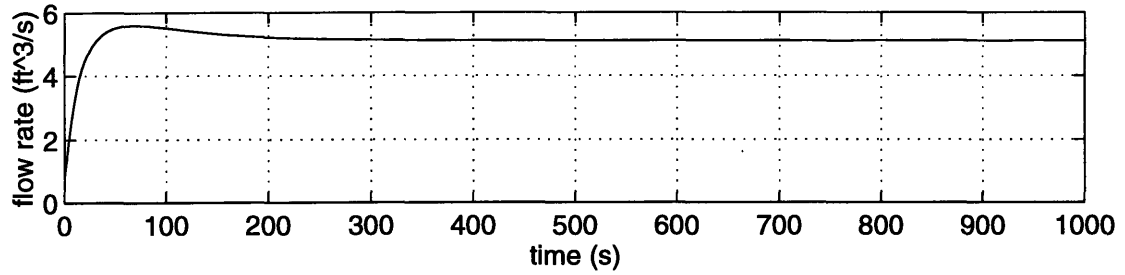
Feedback linearizing controller[two phase flow]:



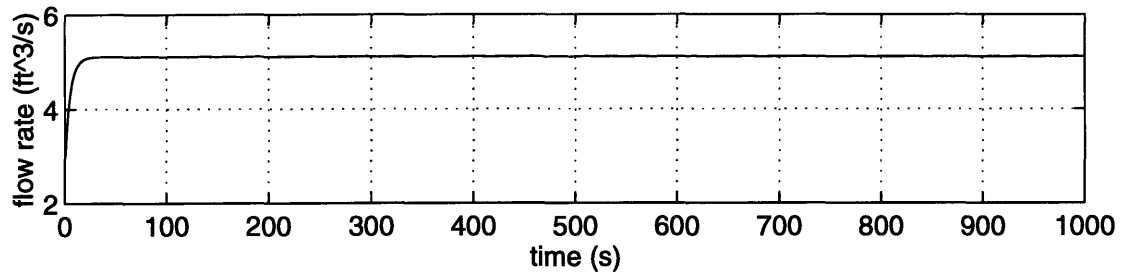




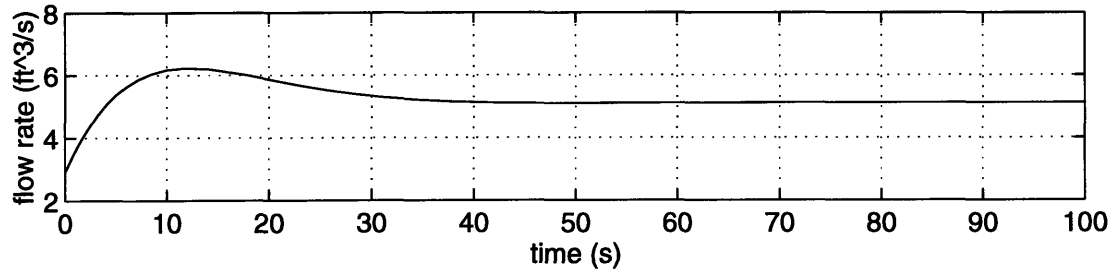
Conventional PI-controller[two phase flow]:



Feedback linearizing P-controller[two phase flow]:



Feedback linearizing controller[two phase flow]:



## Implementation Results on Kingston Unit 9 Simulator

The nonlinear controller was implemented on the Kingston Unit 9 simulator to verify the performance in comparison with the conventional PI controller. The transient response tests was carried out according to the condition prescribed in Table 24 and 25.

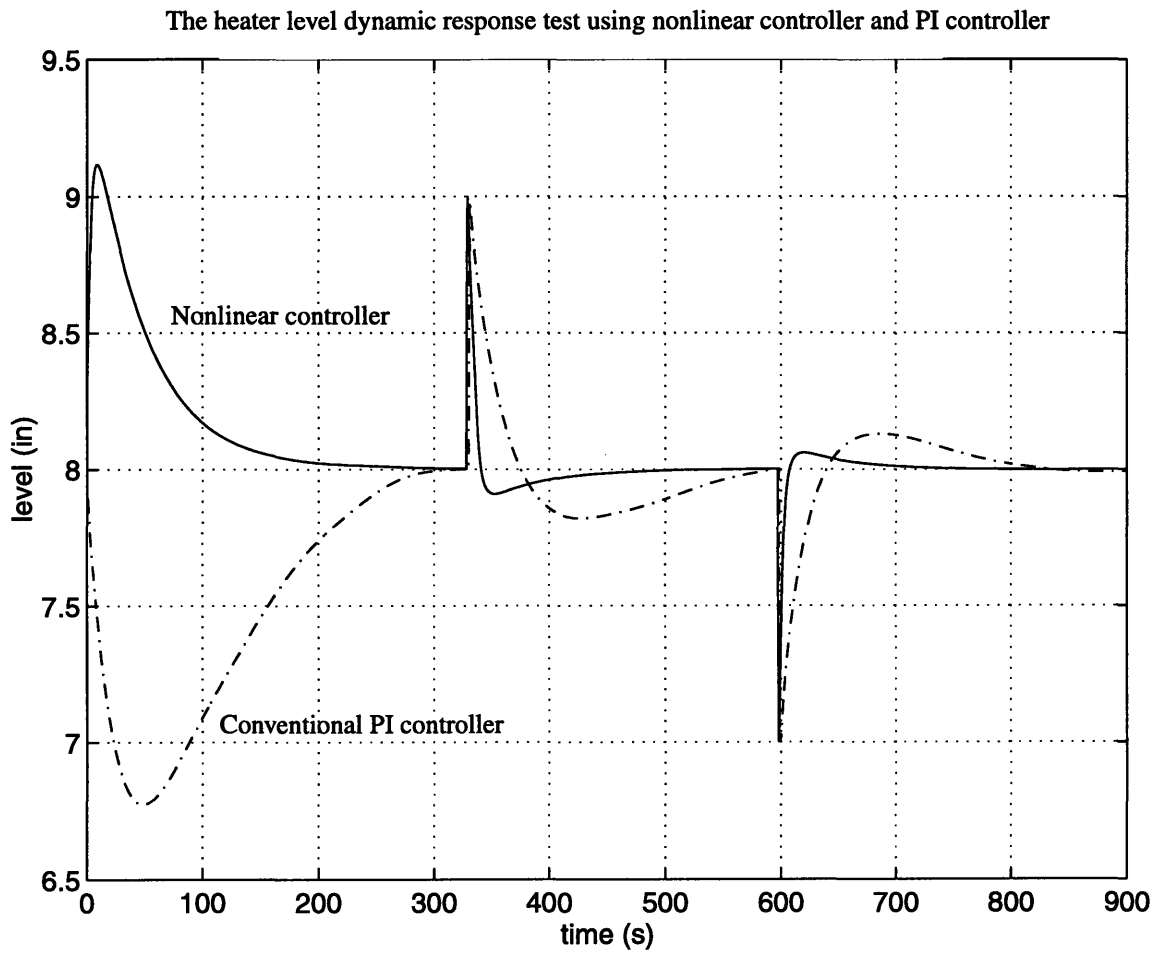
TABLE 24. Control parameters and setpoint changes for conventional PI controller

Proportional Band	Integral Time	Setpoint Command
0.50	0.01	9-8 and 7-8

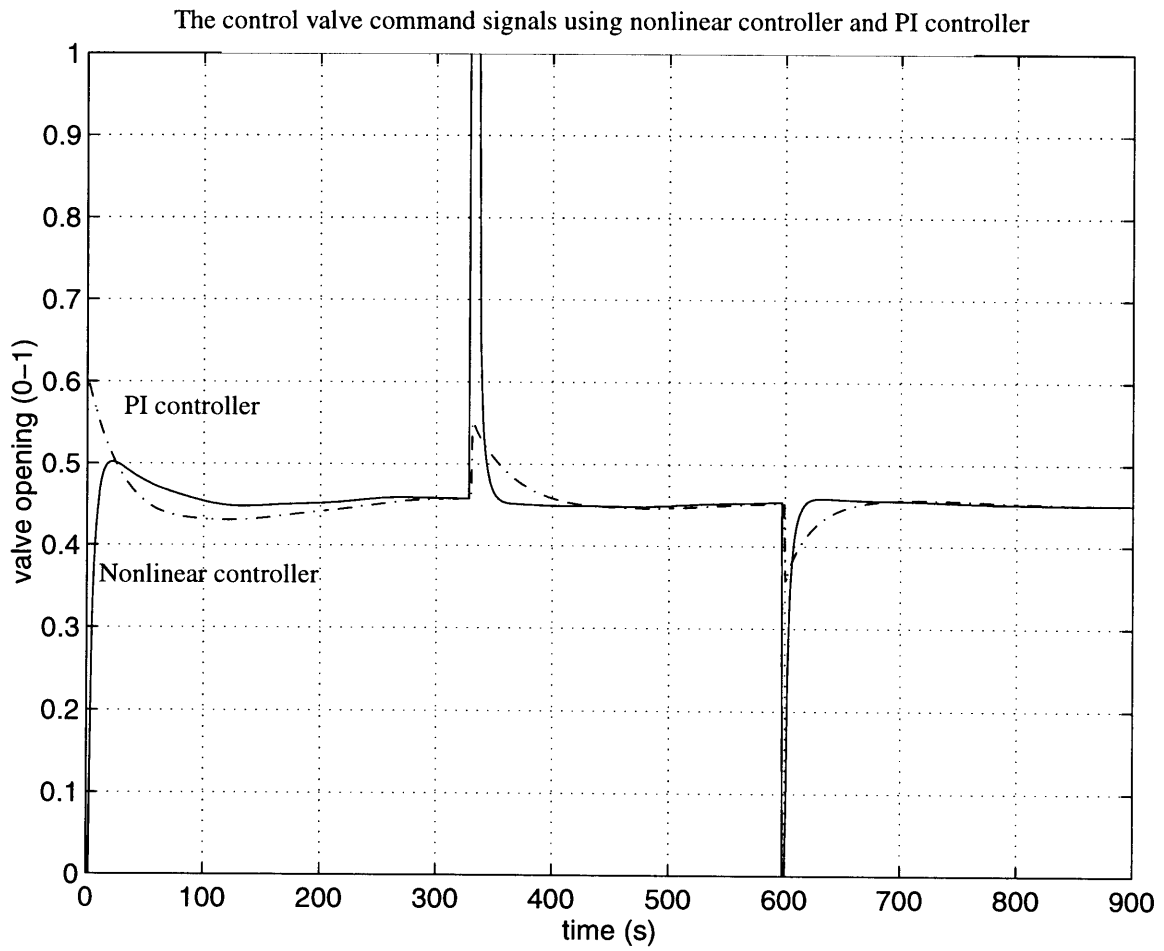
TABLE 25. Control parameters and setpoint changes for nonlinear controller

Gain $\alpha$	Gain $\beta$	Setpoint Command
0.50	0.01	9-8 and 7-8

**Figure 3.14:** Heater level response on Kingston Unit 9 simulator



**Figure 3.15: Valve command signals**



### 3.2.6 Summary and Remarks

The simulations were carried out for both the conventional PI-controller and our non-linear controllers. The purpose of the simulations is to determine the characteristic of each controller and its performance by changing the heater level set point from 0.90 ft. to 0.70 ft. Due to changing nature of the plant, the simulation also considered the effects of uncertainties in the pressure drop between heater#1 and heater#2, the extraction flow and the model. The uncertainty in the pressure drop was simulated by assuming no knowledge in the pressure drop and designing the controllers based only on the nominal pressure drop. The effect of the extraction flow uncertainty was included in a similar manner. The modeling uncertainty was introduced by using the manufacturers valve data to design the controllers. The model using the experimental valve data was assumed to be the most perfect model available for this system. The condition in which each simulation was done is summarized in the preceding table.

In the conventional PI-controller, the proportional and integral gains were chosen according to the dynamic response tests performed on the real plant. In feedback linearizing controllers, the gains are chosen such that the simulations give stable and feasible (no valve saturations) responses.

The simulations in case#1 through case#5 were based on the assumption that the fluid flow in the pipe and valve system is in liquid phase only. In case#1, we assumed perfect information on the system (no model uncertainty, known extraction flow, and pressure drop.). In case#2, the effects of unknown extraction flow are shown. In case#3, we considered the effect of the model uncertainty. Unknown pressure drop was assumed in case#4. Then we combined the effects of model uncertainty with unknown extraction flow and pressure drop in case#5. The simulation case#6 is similar to case#5 but includes the effect of two-phase nature of the flow downstream of the control valve.

It is evident that nonlinear controller has an advantage over the linear PI controller. Through the MATLAB simulations, the following conclusions are drawn.

- At least an order of magnitude decrease in settling time(2%) was obtained using our nonlinear controllers.
  - The zero steady-state error was maintained in the nonlinear PI-controller.
- Further reduction in the settling time was obtained using the nonlinear P-controller but at the expense of a non-zero steady-state error.

Earlier in this section, we have introduced two nonlinear controllers (P-controller, PI-controller) to the feedwater heater control system. Feedback linearization guarantees the stability of the system. The simulations were performed to compare the conventional PI-controller with nonlinear P-controller, and nonlinear PI-controller. Among the plant uncertainties that were considered in the simulations are extraction flow, pressure, and valve flow coefficient. The simulation studies show that an order of magnitude improvement in the settling time can be obtained using our nonlinear controller while delivering uniformly better performance under a variety of operating conditions and modeling uncertainties. The nonlinear P-controller gives zero steady-state error in the case where there is no uncertainties in the model. When the uncertainties are introduced to the system, the P-controller results in a steady state error. The conventional PI-controller works conservatively but the performance is rapidly degraded when the system deviates from the designated operating point. The nonlinear PI-controller has one significant advantage over the conventional PI-controller in that the settling time is much smaller. Also it is robust to the change in the operating condition since it uses the 5th-order least square approximation of the drain flow rate as a function of pressure and valve opening which covers the entire range of operations.

The effect of the two phase flow was also investigated in case#6. Due to the pressure drop after the valve and high drain flow temperature, a fraction of the liquid vaporizes.

Subsequently, the pressure drop in the downstream section of the pipe increases. It is clearly seen by comparing case#5 and case#6 that the increase in valve opening is required to compensate for the flow of vapor which is much lower in density.

MATLAB simulations have shown that the nonlinear controller using feedback linearization works based on the assumption that the system is properly modeled. The performance of the nonlinear PI-controller relies on the accuracy of the drain flow rate model and the availability of the information such as extraction flow and pressure drop. Small deviations from the established system model can be compensated since the controller structure is similar to PI-controller. But large departures from the system model may require additional adaptive schemes which deals directly with those parameters.

Furthermore, the performance comparison between the existing linear PI controller and our nonlinear controller is illustrated through the transient tests conducted on the simulator which is shown in Figure 3.14-3.15. In these tests, both controllers were subjected to the same dynamic system and initial conditions. The drain valve in the simulator is characterized by a nonlinear trim. It is clear from Figure 3.14 that our controller has better settling time and percent overshoot. Our controller quickly achieved the reference level of 8 inches while the PI controller slowly approached the set point. The PI controller also has significantly greater percent overshoot. Figure 3.15 shows that only a fraction of the drain valve capacity was used in the case of PI controller as opposed to the other case. However, these findings should not be too surprising since our nonlinear controller was designed to incorporate both the nonlinearities in the control valve and the heater crosssectional area while the PI controller was tuned conservatively to preserve the stability of the system. Both the simulations and transient tests suggest the use of our nonlinear controller in this level control system which information on nonlinearities prevail.



## 3.3 Adaptive Controller Based on Nonlinear Parametrization

### 3.3.1 Introduction

The problem considered here is again the feedwater heater level control system. We develop in this section a systematic dynamic model of the system so as to include the nonlinear effects from the control valve and the pipe. In an existing power plant, the controller used to regulate the water level in the feedwater heater is a conventional PI controller. Due to inherent nonlinearities, the controller gains may not be optimally selected across the operating range. Conservative gains are used to ensure stability of the system resulting in poor controller performance. We introduce a new nonlinear parametrization controller for the level regulation [1]. With this approach which tackles the nonlinear parametrization directly, we are able to develop a stable nonlinear controller. In the simulations, the performance of our nonlinear controller is investigated. It is confirmed through a series of simulations that the nonlinear controller results in a stable system.

### 3.3.2 Problem Statement

#### Theoretical Development

The dynamic equation of the system can be described by a first order differential equation  $\frac{dh}{dt} = \frac{1}{\rho A(h)}(Q_0 - Q)$  where  $Q$  is the drain flow rate. The drain flow rate is described by the constitutive relation of the flow valve as  $Q = \left[ \frac{C_v^2(\theta)}{3225.42\rho} \right]^{\frac{1}{2}} \sqrt{\Delta P}$  where  $C_v(\theta)$  is the flow coefficient of the valve as a function of the valve opening. Let  $u = \left[ \frac{C_v^2(\theta)}{3225.42\rho} \right]^{\frac{1}{2}}$  and we then have  $Q = u\sqrt{\Delta P}$ .

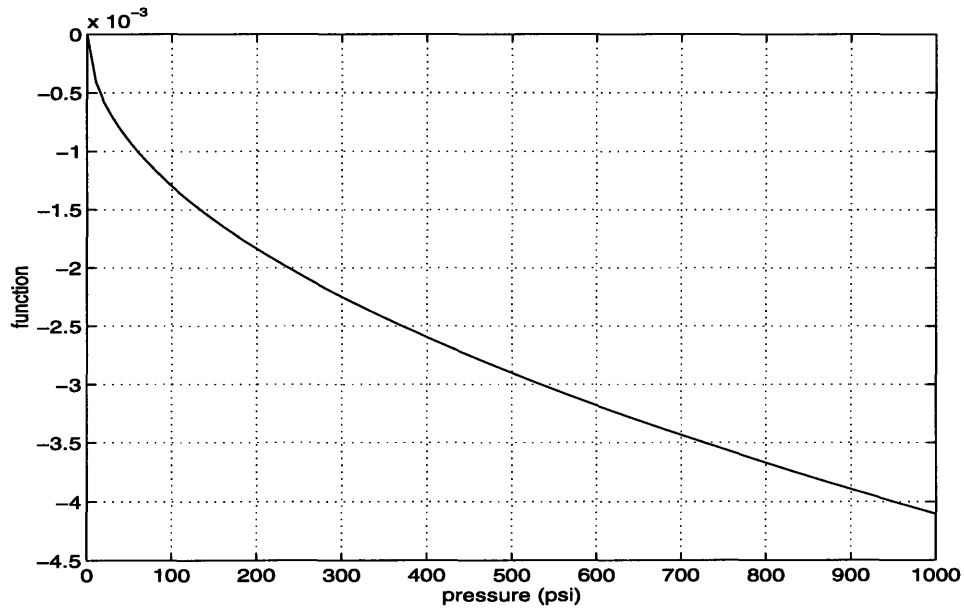
The dynamic equation of the system is now  $\frac{dh}{dt} = \frac{1}{\rho A(h)}Q_0 - \frac{\sqrt{\Delta P}}{\rho A(h)}u$ .

In other words,  $\dot{h} = \psi(h)\alpha + f(\Delta P, h)u$  where  $\psi(h) = \frac{1}{\rho A(h)}$ ,  $\alpha = Q_0$ , and  $f(\Delta P, h) = -\frac{\sqrt{\Delta P}}{\rho A(h)}$ .

Both  $\alpha$  and  $\Delta P$  are unknown parameters with known bounds. It is clear that  $\alpha$  occurs linearly while  $\Delta P$  are nonlinear in  $f(\Delta P, h)$ . The adaptation law for  $\alpha$  can be obtained using

established method in adaptive control. However, the similar approach is not applicable in the latter parameter. Due to the fact that  $f(\Delta P, h)u$  is a convex and monotonically decreasing function of  $\Delta P$  when  $u$  is a positive quantity, a stable adaptation law can be established. And it turns out that  $u$  is always positive since  $C_v(\theta) \geq 0$  results in  $\left( u = \left[ \frac{C_v^2(\theta)}{3225.42p} \right]^{\frac{1}{2}} \right) \geq 0$ . Figure 3.16 illustrates such characteristics of  $f(\Delta P, h)$

**Figure 3.16:** The plot of  $f$  as a function of  $dP$  where  $h = 0.70$  ft.



### 3.3.3 Adaptive Controller Based on Nonlinear Parametrization

The dynamic equation can be written as  $\dot{h} = \psi\alpha + (f - \hat{f})u + \hat{f}u$ .

If  $\hat{f}u = -k\epsilon_s - D_1(s)[h] + r - u_a - \psi\hat{\alpha}$  or  $u = \frac{1}{\hat{f}}(-k\epsilon_s - D_1(s)[h] + r - u_a - \psi\hat{\alpha})$  is chosen,

then  $\dot{h} + D_1(s)[h] = -k\epsilon_s + r - u_a + \psi(\alpha - \hat{\alpha}) + (f - \hat{f})u$  where  $s + D_1(s) = D(s)$  is a Hurwitz polynomial defined in the reference model,  $e_s = D(s) \left[ \int_0^t e d\tau \right]$  is a composite error which is a scalar measure of the state error, and  $\epsilon_s = e_s - \text{sat}\left(\frac{e_s}{\epsilon}\right)$ .

Let the desired trajectory  $h_m$  be chosen as the output of the reference model whose dynamics is governed by the differential equation  $D(s)[h_m] = r$ .

Since  $\dot{e}_s = D(s)[h - h_m]$ ,

the error equation is then  $\dot{e}_s = -k\varepsilon_s - u_a + \psi(\alpha - \hat{\alpha}) + (f - \hat{f})u = -k\varepsilon_s - u_a - \psi\tilde{\alpha} + (f - \hat{f})u$

where  $\tilde{\alpha} = \hat{\alpha} - \alpha$  and  $\hat{f} = f(\Delta\hat{P}, h)$ .

The adaptation law for  $\alpha$  can be obtained using the error model as  $\dot{\hat{\alpha}} = \dot{\alpha} = \Gamma_\alpha \varepsilon_s \psi$ . The adaptation law for  $\Delta P$  cannot be established in the same manner since it is nonlinear in  $\hat{f}$ .

However, we assume that it can be written in the form of  $\dot{\Delta P} = \dot{\hat{\Delta P}} = \Lambda_{\Delta P} \varepsilon_s w$ .

The commonly used Lyapunov function candidate :

$$V = \frac{1}{2}(\varepsilon_s^2 + \tilde{\alpha}^T \Gamma_\alpha^{-1} \tilde{\alpha} + \tilde{\Delta P}^T \Lambda_{\Delta P}^{-1} \tilde{\Delta P})$$

$$\dot{V} = \varepsilon_s \dot{e}_s + \Gamma_\alpha^{-1} \tilde{\alpha} \dot{\tilde{\alpha}} + \Lambda_{\Delta P}^{-1} \tilde{\Delta P} \dot{\tilde{\Delta P}}, \text{ noting that } \frac{d}{dt} \left( \frac{1}{2} \varepsilon_s^2 \right) = \varepsilon_s \dot{e}_s.$$

$$\dot{V} = -k\varepsilon_s^2 + \varepsilon_s[-u_a - \psi\tilde{\alpha} + (f - \hat{f})u] + \varepsilon_s(\psi\tilde{\alpha}) + \varepsilon_s w \tilde{\Delta P}$$

$$\dot{V} = -k\varepsilon_s^2 + \varepsilon_s[(f - \hat{f})u + w\tilde{\Delta P} - u_a], \text{ where the tuning function } u_a = a(t) \text{sat}\left(\frac{e_s}{\varepsilon}\right).$$

$$\dot{V} = -k\varepsilon_s^2 + \varepsilon_s \left[ (f - \hat{f})u + w\tilde{\Delta P} - a(t) \text{sat}\left(\frac{e_s}{\varepsilon}\right) \right]$$

Three distinct cases are considered according to the sign and magnitude of  $\varepsilon_s$ ,

i.)  $|e_s(t)| \leq \varepsilon$

ii.)  $e_s(t) < -\varepsilon$

iii.)  $e_s(t) > \varepsilon$

i.) If  $|e_s| \leq 0$ , then  $\varepsilon_s = 0$  and thus  $\dot{V} = 0$ . Therefore the stability is assured for any choices of  $w$  and  $u_a(t)$ .

ii.) If  $\varepsilon_s < 0$ ,

To make  $\dot{V}$  negative semi-definite, it is required that  $(f - \hat{f})u + w(\hat{\Delta P} - \Delta P) + a(t) \geq 0$ . Since  $f(\Delta P, h)u$  is convex and monotonically decreasing in  $\Delta P$ , we have

$$(f - \hat{f})u \geq \frac{\partial}{\partial \Delta P} f u \Big|_{\hat{\Delta P}} (\Delta P - \hat{\Delta P}).$$

$$\Rightarrow a = 0$$

$$\Rightarrow w = \frac{\partial}{\partial \Delta P} f u \Big|_{\hat{\Delta P}}$$

iii.) If  $\varepsilon_s \geq 0$ ,

It is required that  $(f - \hat{f})u + w(\hat{\Delta P} - \Delta P) - a(t) \leq 0$  to make  $\dot{V}$  negative semi-definite. It is also desired that the quantity  $a(t)$  be minimized. The solution for  $a(t)$  and  $w$  can be obtained by solving the following optimization problem.

$$a(t) \geq (f_{max} - \hat{f})u + w(\hat{\Delta P} - \Delta P_{min})$$

$$a(t) \geq (f_{min} - \hat{f})u + w(\hat{\Delta P} - \Delta P_{max})$$

The solution of the problem is the same as in

$$(f_{max} - \hat{f})u + w(\hat{\Delta P} - \Delta P_{min}) = (f_{min} - \hat{f})u + w(\hat{\Delta P} - \Delta P_{max}).$$

$$\Rightarrow w = -\frac{f_{max} - f_{min}}{\Delta P_{max} - \Delta P_{min}} u$$

$$\Rightarrow a = (f_{max} - \hat{f})u - \left( \frac{f_{max} - f_{min}}{\Delta P_{max} - \Delta P_{min}} \right) u (\hat{\Delta P} - \Delta P_{min})$$

Hence, the laws for choosing the quantities  $a(t)$  and  $w$  are:

$$a = (f_{max} - \hat{f})u - \left( \frac{f_{max} - f_{min}}{\Delta P_{max} - \Delta P_{min}} \right) u (\hat{\Delta P} - \Delta P_{min}) ; \text{ if } \varepsilon_s \geq 0$$

$a = 0$  ; otherwise

and

$$w = -sat\left(\frac{e_s}{\varepsilon}\right) \frac{f_{max} - f_{min}}{\Delta P_{max} - \Delta P_{min}} u \quad ; \text{ if } \varepsilon_s \geq 0$$

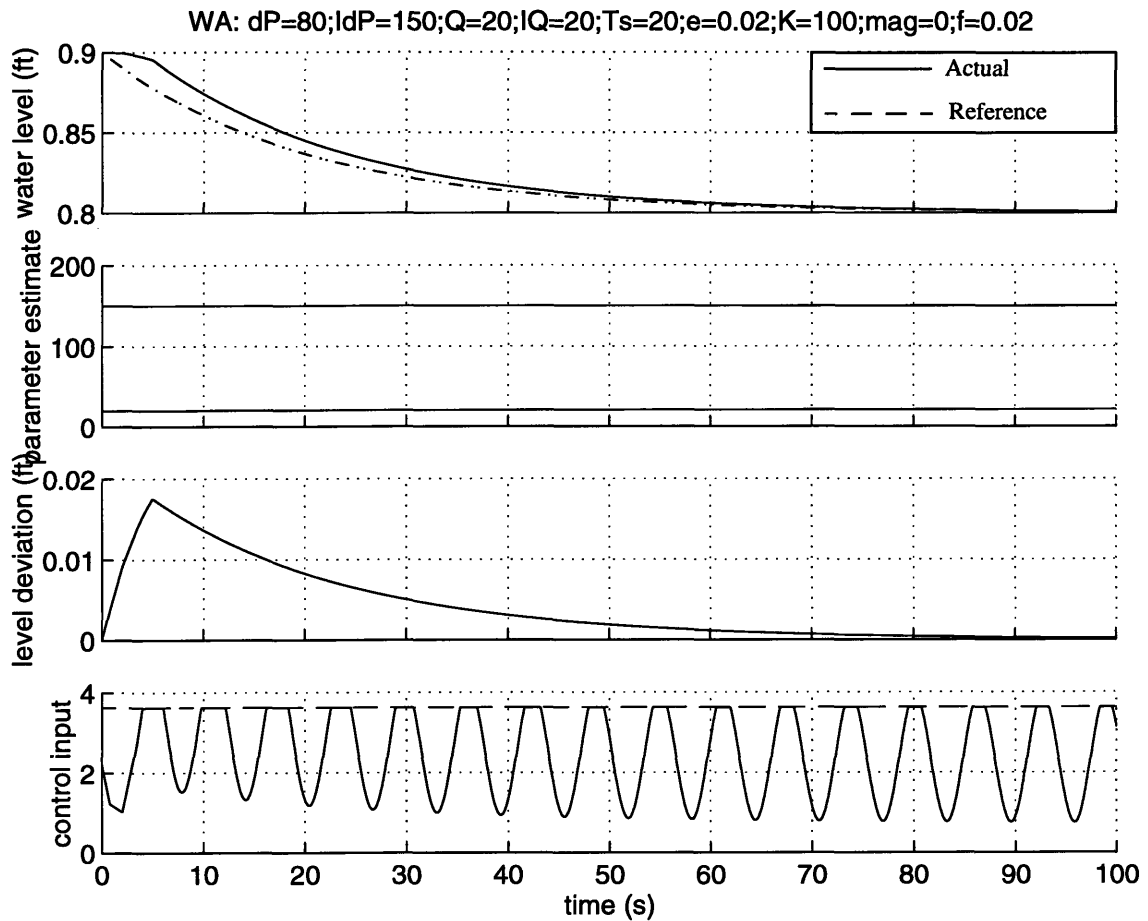
$$w = -sat\left(\frac{e_s}{\varepsilon}\right) \frac{\partial}{\partial \Delta P} f u \Big|_{\hat{\Delta P}} \quad ; \text{ otherwise}$$

### 3.3.4 Simulation Results

#### Simulation Case #1: With Adaptation, Over-predicted Pressure Drop, and $T_s = 20s$

TABLE 26. Control Parameters

Predicted Pressure Drop (psi)	Actual Pressure Drop (psi)	Predicted Extraction Flow Rate (lbs/s)	Actual Extraction Flow Rate (lbs/s)	Model Time Constant (s)	Dead Band	Control Gain K	Sinusoid Magnitude	Sinusoid Frequency (rad/s)
150	80	20	20	20	0.02	100	0	0

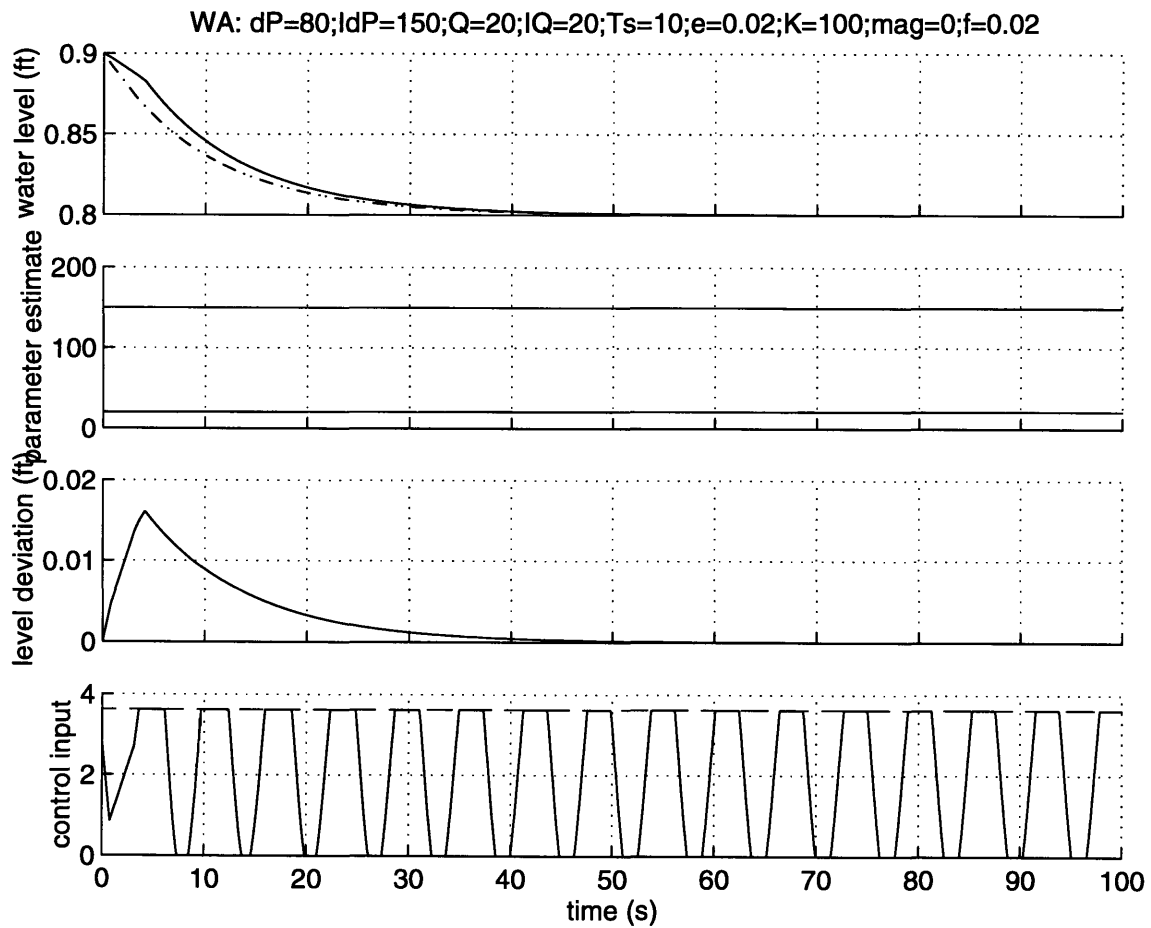


## Simulation Case #2: With Adaptation, Overpredicted Pressure Drop, and $T_s = 10s$

TABLE 27.

Control Parameters

Predicted Pressure Drop (psi)	Actual Pressure Drop (psi)	Predicted Extraction Flow Rate (lbs/s)	Actual Extraction Flow Rate (lbs/s)	Model Time Constant (s)	Dead Band	Control Gain K	Sinusoid Magnitude	Sinusoid Frequency (rad/s)
150	80	20	20	10	0.02	100	0	0

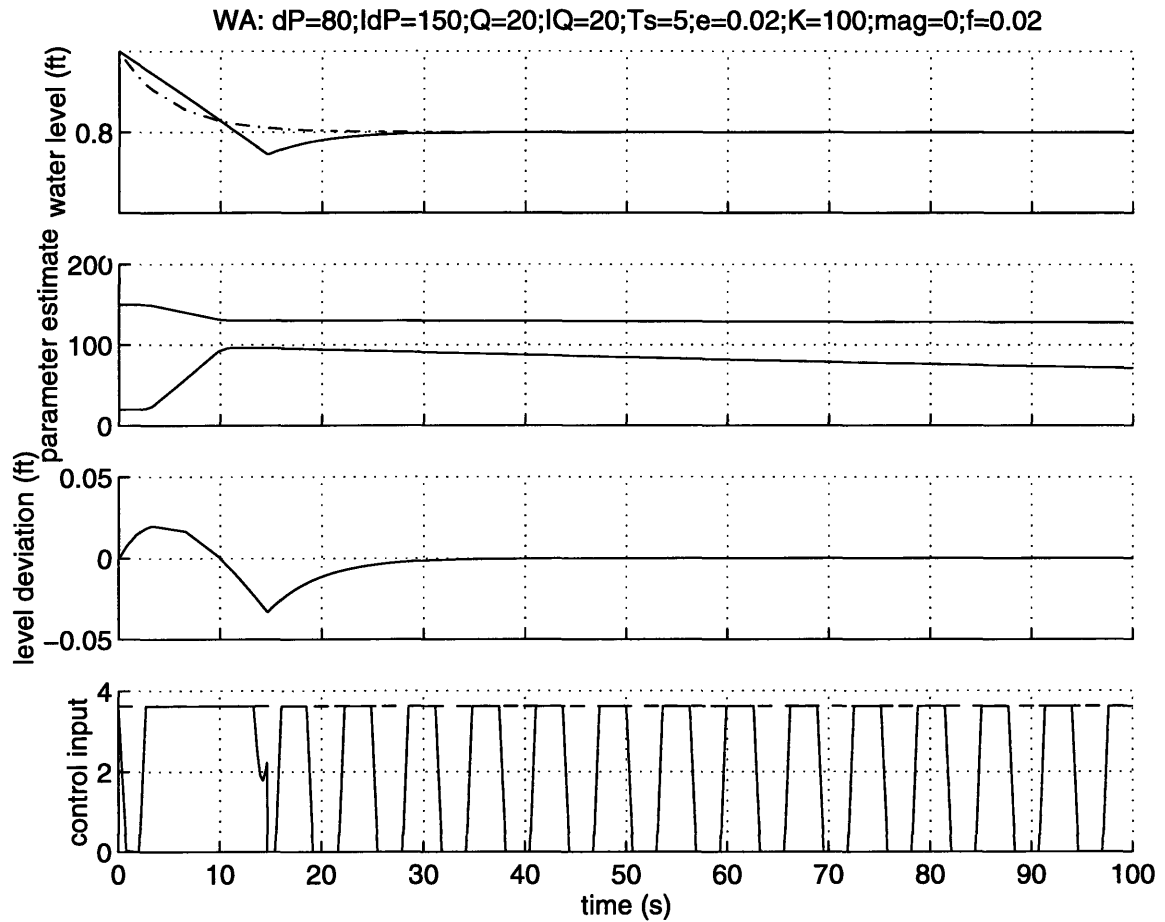


### Simulation Case #3: With Adaptation, Overpredicted Pressure Drop, and $T_s = 5s$

TABLE 28.

Control Parameters

Predicted Pressure Drop (psi)	Actual Pressure Drop (psi)	Predicted Extraction Flow Rate (lbs/s)	Actual Extraction Flow Rate (lbs/s)	Model Time Constant (s)	Dead Band	Control Gain K	Sinusoid Magnitude	Sinusoid Frequency (rad/s)
150	80	20	20	5	0.02	100	0	0



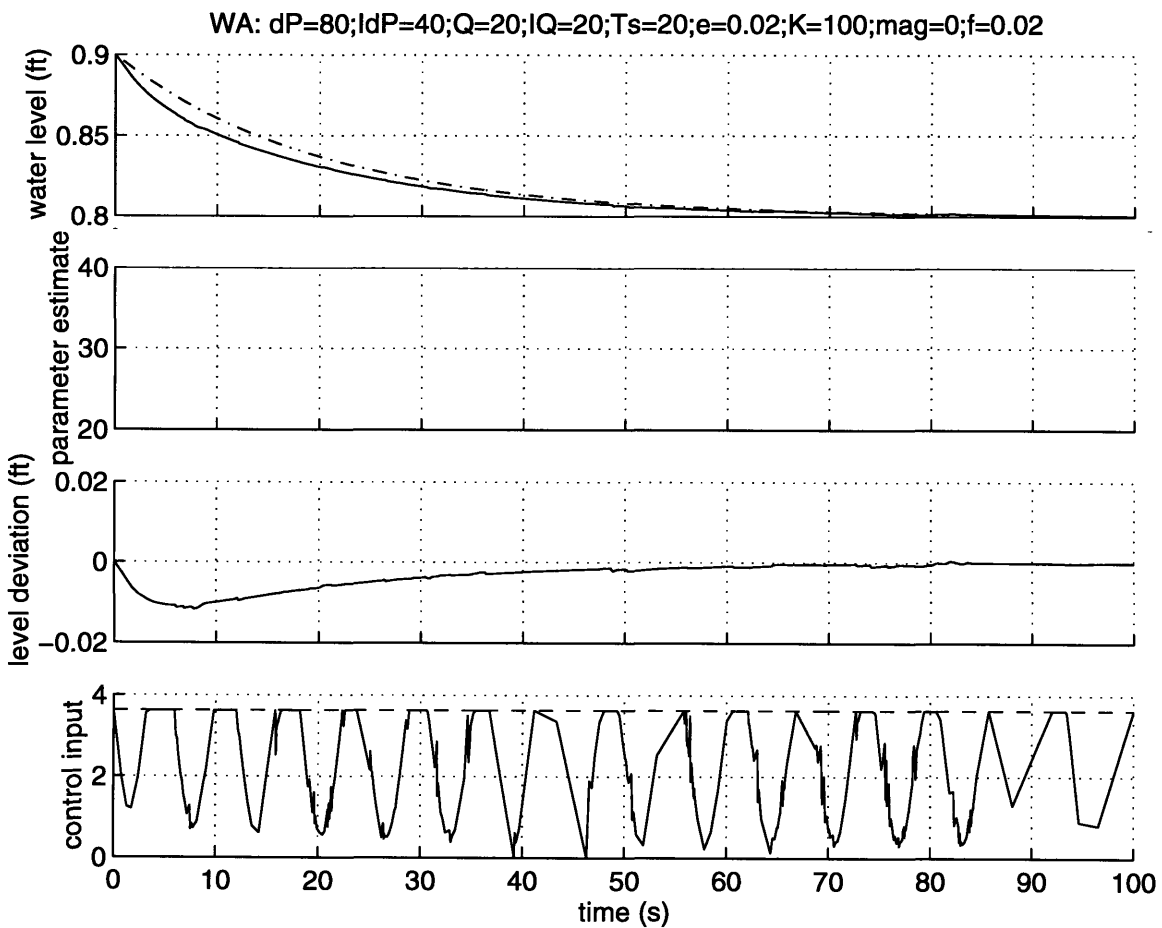


### Simulation Case #4: With Adaptation, Underpredicted Pressure Drop, and $T_s = 20s$

TABLE 29.

Control Parameters

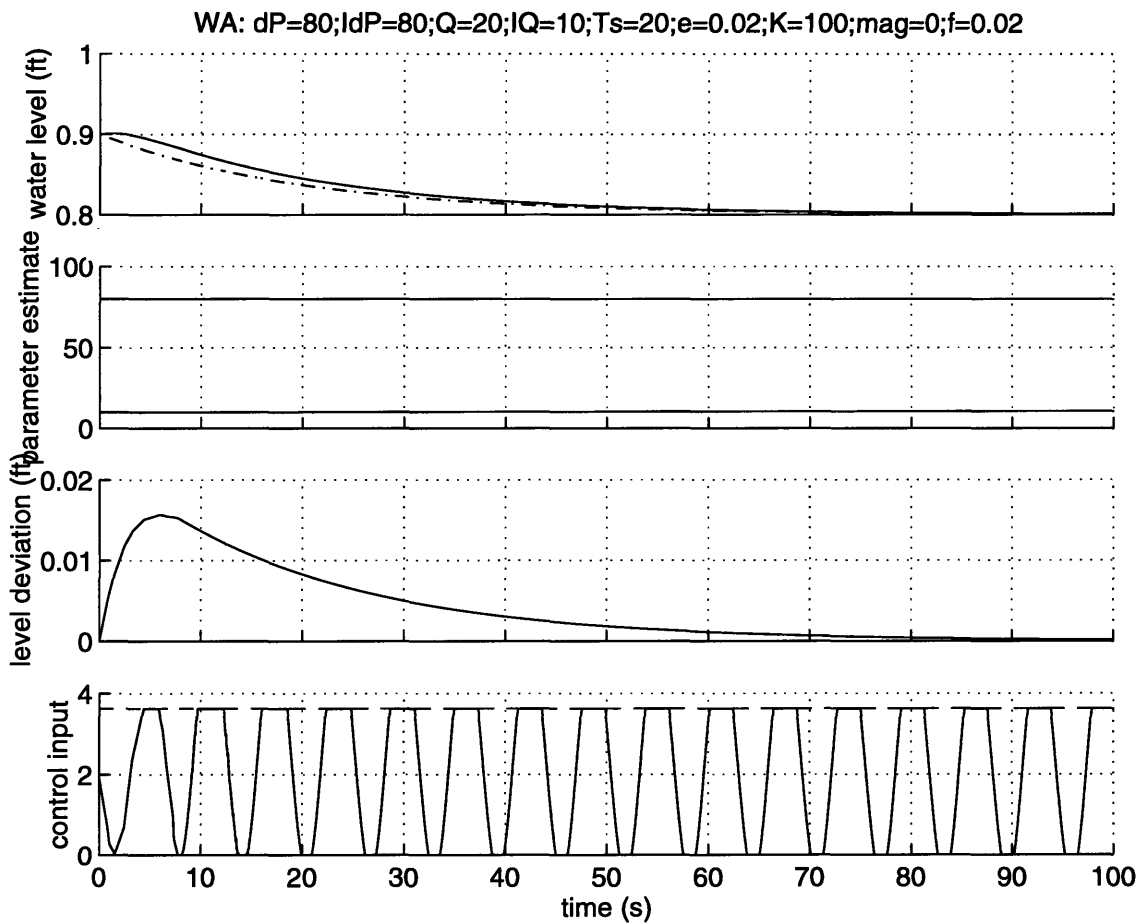
Predicted Pressure Drop (psi)	Actual Pressure Drop (psi)	Predicted Extraction Flow Rate (lbs/s)	Actual Extraction Flow Rate (lbs/s)	Model Time Constant (s)	Dead Band	Control Gain K	Sinusoid Magnitude	Sinusoid Frequency (rad/s)
40	80	20	20	20	0.02	100	0	0



# Simulation Case #5: With Adaptation, Underpredicted Flow, and $T_s = 20s$

TABLE 30. Control Parameters

Predicted Pressure Drop (psi)	Actual Pressure Drop (psi)	Predicted Extraction Flow Rate (lbs/s)	Actual Extraction Flow Rate (lbs/s)	Model Time Constant (s)	Dead Band	Control Gain K	Sinusoid Magnitude	Sinusoid Frequency (rad/s)
80	80	20	10	20	0.02	100	0	0

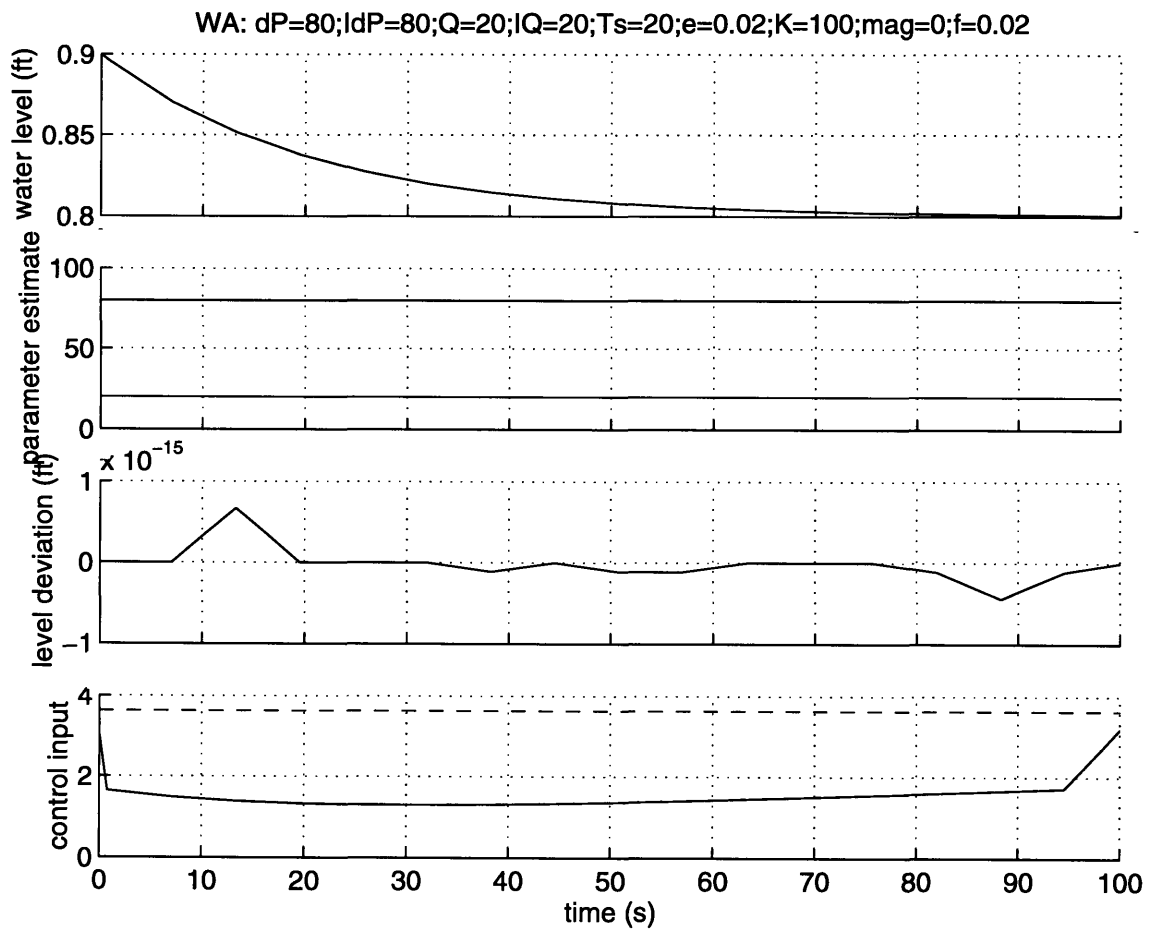


# Simulation Case #6: With Adaptation, Exact Knowledge of Pressure and Flow

TABLE 31.

Control Parameters

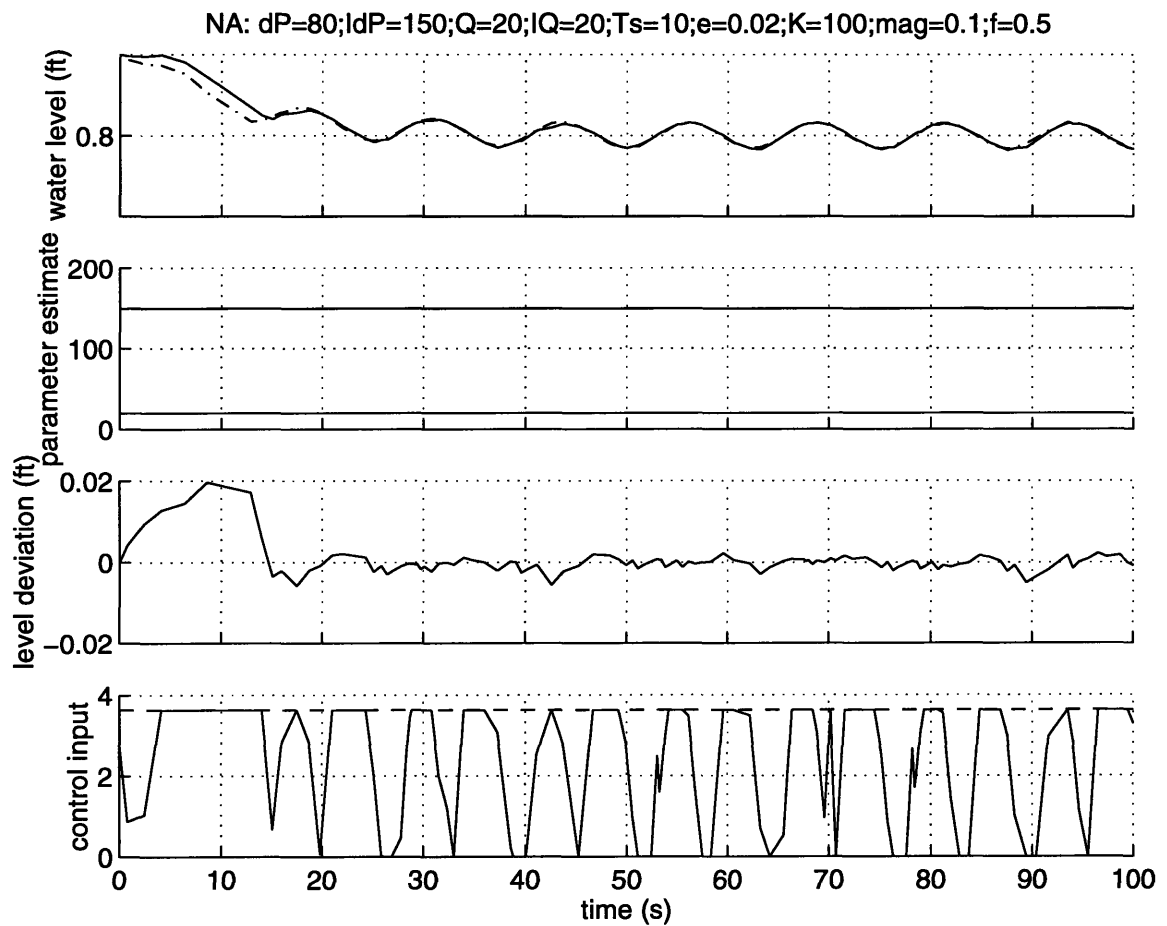
Predicted Pressure Drop (psi)	Actual Pressure Drop (psi)	Predicted Extraction Flow Rate (lbs/s)	Actual Extraction Flow Rate (lbs/s)	Model Time Constant (s)	Dead Band	Control Gain K	Sinusoid Magnitude	Sinusoid Frequency (rad/s)
80	80	20	20	20	0.02	100	0	0



### Simulation Case #7: Without Adaptation, Sinusoid Model

TABLE 32. Control Parameters

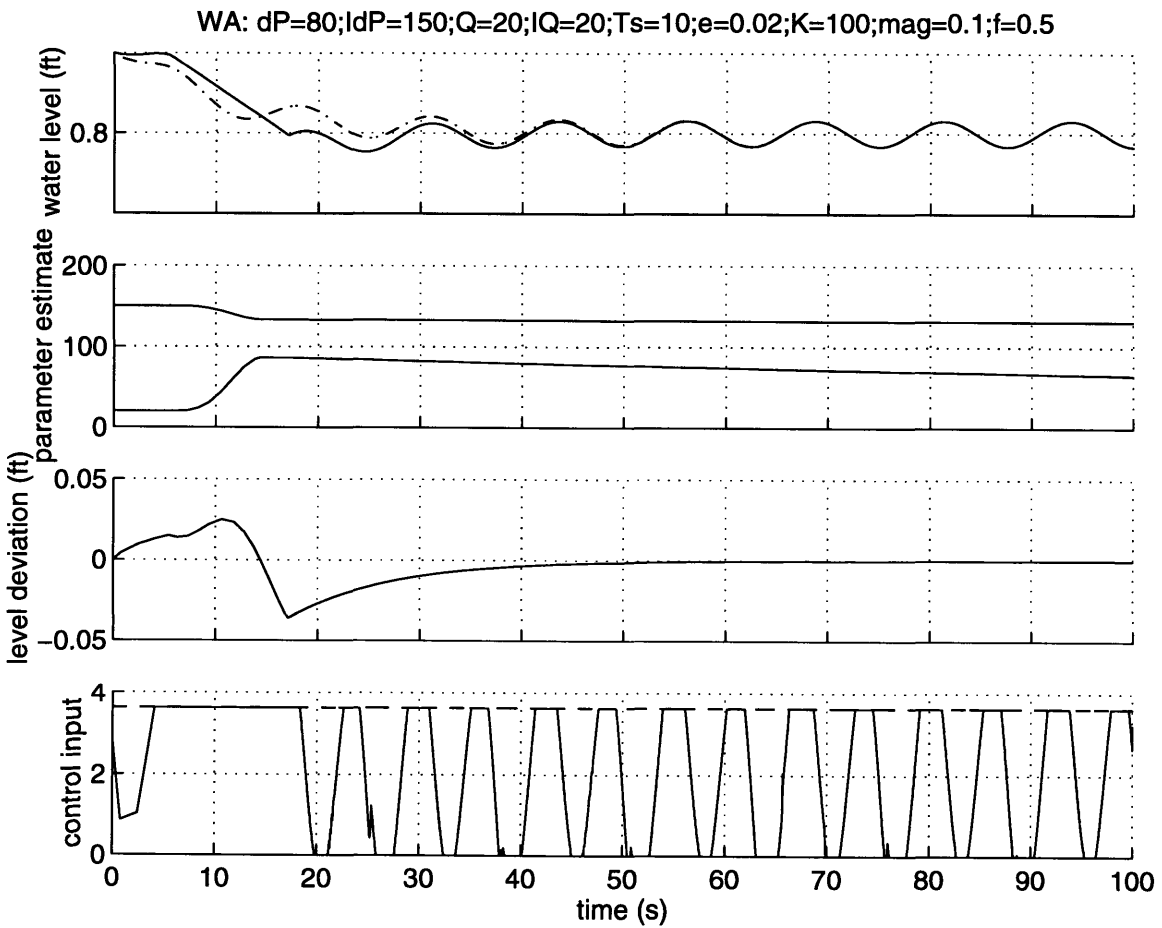
Predicted Pressure Drop (psi)	Actual Pressure Drop (psi)	Predicted Extraction Flow Rate (lbs/s)	Actual Extraction Flow Rate (lbs/s)	Model Time Constant (s)	Dead Band	Control Gain K	Sinusoid Magnitude	Sinusoid Frequency (rad/s)
150	80	20	20	10	0.02	100	0.1	0.5



# Simulation Case #8: With Adaptation, Sinusoid Model

TABLE 33. Control Parameters

Predicted Pressure Drop (psi)	Actual Pressure Drop (psi)	Predicted Extraction Flow Rate (lbs/s)	Actual Extraction Flow Rate (lbs/s)	Model Time Constant (s)	Dead Band	Control Gain K	Sinusoid Magnitude	Sinusoid Frequency (rad/s)
150	80	20	20	10	0.02	100	0.1	0.5



### 3.3.5 Summary and Remarks

In this Chapter we examine the performance of our adaptive controller which is based on nonlinear parametrization through MATLAB simulations. In simulation case #1 through #3, we study the effect of the time constant of the reference model on the performance of the controller. While other controller parameters were held constant, the time constant was reduced from  $20s$  to  $5s$ . We found that the controller can only respond to the model down to  $5s$ -time constant after which the tracking error in the initial period became high. We realize at this point that we have reached a physical limitation of the control valve. The control valve has a limit of the maximum flow it can pass given a certain pressure drop. This is illustrated by the heater level response in simulation case #3 where the slope of the level reached a maximum value which is still not sufficient to follow the reference model. However, the controller was able to track the reference model as soon as the slope became feasible. It is noted also that the change in parameter estimates is significant only in case #3. This is typical of an adaptive controller when operated in absence of persistent excitation.

Simulation case #4 shows the condition where the pressure drop across the control valve is underpredicted. In this case, the control valve command is overvalued since the controller is based on the knowledge that the pressure drop is equal to  $40psi$ . The fact that the heater level response stays below that of the reference model confirms this conclusion. Simulation case #5 illustrates the opposite effect where the controller expects the extraction flow of  $10lbs/s$  and therefore the control valve command is undervalued. This is again confirmed by the heater level response that stays above the reference. Naturally, when the knowledge of the pressure and the extraction flow is available to the controller, the heater level follows the reference exactly as shown in case #6.

We will now illustrate that the adaptation feature of the controller results in a better performance in heater level control. In addition to the constant input, we modulate the Sinusoid signal of small magnitude in the reference model. This is done to provide persistent excitation condition for the adaptation. In simulation case #7, the adaptation was turned off and the controller cannot follow the reference model quite exactly. The heater level error exists and does not tend to zero. On the other hand, the adaptation was turned on in case #8 in which the heater level follow the reference model exactly and the error tends to zero. This is the merit of two additional degrees of freedom provided by the adaptive controller which helps in fine-tuning the control input.

In conclusion, the adaptive controller based on nonlinear parametrization has been shown to perform well despite the absence of the knowledge on the pressure drop across the valve and the extraction flow rate. The adaptation is a necessary and important feature of the controller should an optimum performance is to be obtained.





# Chapter 4

## Position Control in Magnetic Bearing System

### 4.1 Introduction

Magnetic Bearings are currently used in various applications such as machine tool spindle, turbo machinery, robotic devices, and many other contact-free actuators. Such bearings have been observed to be considerably superior to mechanical bearings in many of these applications. Two important reasons for this are total elimination of the friction and the active control nature of magnetic bearing. As early as 1842, Earnshaw introduced the concept of levitating a spinning body using magnetic forces. He suggested that the stability of the body cannot be achieved by means of passive permanent magnets alone but that at least one of the three axes must be actively controlled. A suspended body makes no physical contacts with the magnetic bearings, thus resulting in extremely low friction. Not only do the magnetic bearings have longer life cycle than the conventional bearings but also there is no need for lubrication.

In many of these applications, disturbances and the dynamic changes due to varying operating conditions and loads are present. The fact that the underlying electromagnetic fields are highly nonlinear and open-loop unstable poses a challenging problem in dynamic modeling, analysis, and control. As a result, controllers based on linearized dynamic models may not be suitable for applications where high rotational speed during the operation is desired. Yet another feature in magnetic bearings is the fact that the air gap, which is an underlying physical parameter appears nonlinearly in the dynamic model. Due to thermal expansion effects, there are uncertainties associated with this parameter. The fact that dynamic models of magnetic bearings include nonlinear dynamics as well as

nonlinear parametrization suggests that an adaptive controller is needed which employs prior knowledge about these nonlinearities and uses an appropriate estimation scheme for the unknown nonlinear parameters. We show in this section that such an adaptive controller can be established which ensures that the system will behave in a stable manner and in addition, leads to better performance.

## 4.2 Dynamic System Modeling

We focus on a specific system which employs magnetic bearings which is a magnetically-levitated turbo pump [15]. The spinning motion of the rotor is induced by an electric motor. There are two circular-shaped bearings used to control the horizontal motion of the rotor. The thrust bearings are used to support the weight and vertical load of the rotor. We will evaluate the performance of an adaptive controller in controlling the vertical motion of the rotor.

If  $F_u$  denotes the attractive force exerted by the upper bearing,  $M_u$  the magnetomotive force generated by the magnetic flux,  $n$  the number of coils,  $\mu_0$  the air permeability,  $A$  the pole face area,  $i_u$  the coil current,  $h_0$  the nominal air gap,  $z$  the rotor position, using the fact that the magnetic energy  $\epsilon_u$  is given by

$$\epsilon_u = \frac{\phi_u^2}{2C_a} \quad (4.1)$$

where

$$C_a = \frac{\mu_0 A}{2(h_0 - z)} \quad (4.2)$$

and the relations

$$F_u = \frac{\partial \epsilon_u}{\partial z} = \frac{\phi_u^2}{\mu_0 A} \quad (4.3)$$

$$M_u = \frac{\partial \epsilon_u}{\partial \phi} = \frac{2(h_0 - z)\phi_u}{\mu_0 A} \quad (4.4)$$

$$M_u = ni \quad (4.5)$$

we obtain that

$$F_u = \frac{n^2 \mu_0 A i_u^2}{4(h_0 - z)^2} \quad (4.6)$$

Similarly, the expression for the lower bearing can be derived as follows:

$$F_l = -\frac{n^2 \mu_0 A i_l^2}{4(h_0 + z)^2} \quad (4.7)$$

Hence, the resultant magnetic force exerted by the thrust bearing system is

$$F = \frac{n^2 \mu_0 A i_u^2}{4(h_0 - z)^2} - \frac{n^2 \mu_0 A i_l^2}{4(h_0 + z)^2} \quad (4.8)$$

This equation indicates that the magnetic force is highly nonlinear in both the rotor position and coil current. Also, it shows that the parameter  $h_0$  appears nonlinearly.

### 4.3 The Control Objective - Rotor Position Control

To actively position the rotor, a bias current  $i_0$  is applied to both upper and lower magnets and an input  $u$  is to be determined by the control strategy. A bias current scheme consists of choosing

$$i_u = i_0 + u \quad (4.9)$$

$$i_l = i_0 - u \quad (4.10)$$

One can rewrite Eq. (4.8) as

$$\ddot{z} - g = f_1(h_0, \alpha, z) + f_2(h_0, \alpha, z)u + f_3(h_0, \alpha, z)u^2 \quad (4.11)$$

where

$$\alpha = \frac{n^2 \mu_0 A}{4M} \quad (4.12)$$

$$f_1(h_0, \alpha, z) = \frac{4\alpha h_0 z i_0^2}{(h_0 - z)^2 (h_0 + z)^2} = \alpha z i_0^2 \gamma_1(h_0, z) \quad (4.13)$$

$$f_2(h_0, \alpha, z) = \frac{2\alpha(h_0^2 + z^2)i_0}{(h_0 - z)^2 (h_0 + z)^2} = \alpha i_0 \gamma_2(h_0, z) \quad (4.14)$$

$$f_3(h_0, \alpha, z) = \frac{\alpha h_0 z}{(h_0 - z)^2 (h_0 + z)^2} = \alpha z \gamma_3(h_0, z) \quad (4.15)$$

The control objective is to track the rotor position with a stable second-order model as represented by the following differential equation.

$$\ddot{z}_m + c_1 \dot{z}_m + c_2 z_m = r \quad (4.16)$$

## 4.4 Adaptive Controller Based on Nonlinear Parametrization

By examining Eq. (4.11), it is apparent that the parameter  $h_0$  occurs nonlinearly while  $\alpha$  occurs linearly. An examination of the functions  $f_1$ ,  $f_2 u$ , and  $f_3 u^2$  reveals their concavity/convexity property and are summarized in Table 34. We will now show that the following adaptive controller can be realized:

TABLE 34. Properties of  $f_1$ ,  $f_2 u$ , and  $f_3 u^2$  as a function of  $h_i$

Function	Concavity/ Convexity	Monotonic Property	Prerequisite
$F_1 = f_1$	convex concave	decreasing increasing	$0 < z < h_{min}$ $-h_{min} < z < 0$
$F_2 = f_2 u$	convex concave	decreasing increasing	$u > 0$ $u < 0$
$F_3 = f_3 u^2$	convex concave	decreasing increasing	$0 < z < h_{min}$ $-h_{min} < z < 0$

Following the approach in Chapter 2, we choose the controller as

$$u = \frac{1}{f_2} \{-k\varepsilon_s - D_1(s) + r + u_a(t) + g - \hat{f}_1 - \hat{f}_3 u^2\} \quad (4.17)$$

where  $\varepsilon$  is the dead zone,  $c_1$  and  $c_2$  are positive constants and

$$\begin{aligned} \varepsilon_s &= e_s - \varepsilon \text{sat}\left(\frac{e_s}{\varepsilon}\right) ; e_s = D(s) \left[ \int_0^t (z - z_m) d\tau \right] \\ D(s) &= s^2 + c_1 s + c_2 ; D_1(s) = c_1 s + c_2 \\ u_a(t) &= -\text{sat}\left(\frac{e_s}{\varepsilon}\right) \sum_{i=1}^3 a_i(t) \end{aligned} \quad (4.18)$$

Since  $\alpha$  occurs linearly in  $f_i$ , the adaptation laws can be established using the standard method in adaptive control theory [11] as

$$\hat{\alpha}_1 = \varepsilon_s \Lambda_1 \hat{\gamma}_1 z i_0^2 ; \hat{\alpha}_2 = \varepsilon_s \Lambda_2 \hat{\gamma}_2 i_0 u ; \hat{\alpha}_3 = \varepsilon_s \Lambda_3 \hat{\gamma}_3 z u^2$$

where  $\Lambda_i$  are positive.

Since  $\alpha$  appears in conjunction with three different functions, we need to generate three distinct estimates,  $\hat{\alpha}_i$ ,  $i = 1, 2, 3$  as above. Special treatment must be made to obtain the adaptation law for parameter  $h_i$  using the method outlined in Chapter 2. Since both  $\alpha_i$  and  $i_0$  are positive quantities, it is sufficient to evaluate the property of each function in absence of these information. The approach described in Chapter 2 allows us to establish adaptation laws for  $h_i$  as follows:

$$\hat{h}_1 = \varepsilon_s w_1 ; \hat{h}_2 = \varepsilon_s w_2 ; \hat{h}_3 = \varepsilon_s w_3$$

Since the functions  $F_i$  are either convex or concave,  $a_i$  and  $w_i$  are chosen as follows:

a.)  $F_i$  is convex

If  $\varepsilon_s \geq 0$ ;

$$a_i = \text{sat}(e_s/\varepsilon)\alpha_{\max}\left\{F_{i_{\max}} - \hat{F}_i - \frac{F_{i_{\max}} - F_{i_{\min}}}{h_{\max} - h_{\min}}(\hat{h} - h_{\min})\right\} \quad (4.19)$$

$$w_i(t) = -\text{sat}(\varepsilon_s/\varepsilon)\frac{F_{i_{\max}} - F_{i_{\min}}}{h_{\max} - h_{\min}} \quad (4.20)$$

Otherwise;

$$a_i = 0 \quad (4.21)$$

$$w_i(t) = -\text{sat}(e_s/\varepsilon)\frac{\partial F_i}{\partial h}\bigg|_{\hat{h}} \quad (4.22)$$

b.)  $F_i$  is concave

If  $\varepsilon_s \geq 0$ ;

$$a_i = 0 \quad (4.23)$$

$$w_i(t) = \text{sat}(e_s/\varepsilon)\frac{\partial F_i}{\partial h}\bigg|_{\hat{h}} \quad (4.24)$$

Otherwise;

$$a_i = -\text{sat}(e_s/\varepsilon)\alpha_{\max}\left\{\hat{F}_i - F_{i_{\min}} - \frac{F_{i_{\max}} - F_{i_{\min}}}{h_{\max} - h_{\min}}(\hat{h} - h_{\min})\right\} \quad (4.25)$$

$$w_i(t) = -\text{sat}(\varepsilon_s/\varepsilon)\frac{F_{i_{\max}} - F_{i_{\min}}}{h_{\max} - h_{\min}} \quad (4.26)$$

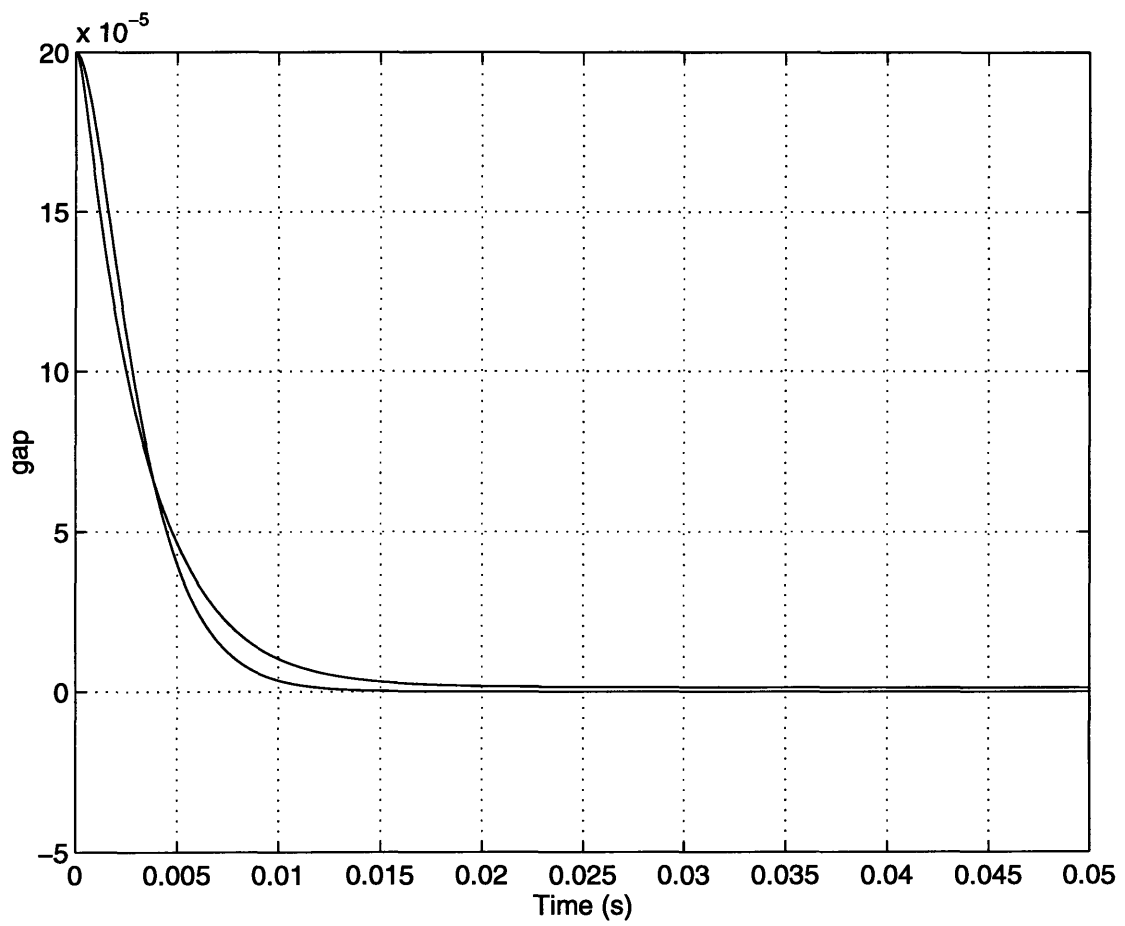
TABLE 35.

Parameters: Adaptive controller based on nonlinear parametrization

Parameters	Values	Parameters	Values
Initial Position	$200\mu m$	$h_0$	$4e^{-4}$
Desired Position	0	$n$	133
$c_1$	360000	$A$	$7e^{-4}$
$c_2$	1200	$i_0$	0.50
$k$	5	$M$	2.2

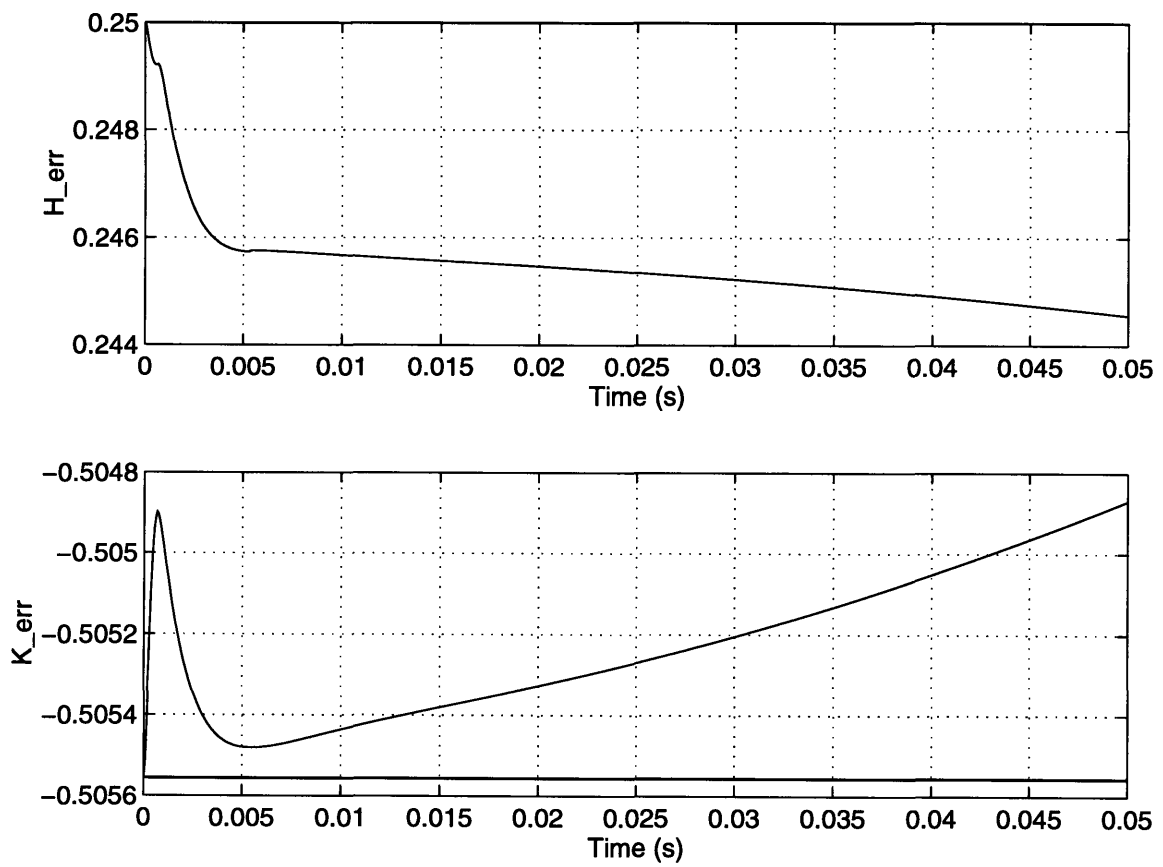
The adaptive controller defined by Eq. (4.17) - (4.26) guarantees the stability of the magnetic bearing system. The control task is to levitate the rotor from the initial position ( $200\mu m$ ) which is the maximum available distance up to the equilibrium position ( $0\mu m$ ) by following a second order reference model. Figure 4.1 shows that the controller achieves excellent tracking accuracy when both system and model originate at  $z = 200\mu m$ . The steady state error is the result of the dead band which can be further reduced by decreasing  $\epsilon$ . Figure 4.2 also reveals that the adaptation parameters  $h_i$  and  $\alpha_i$  do not converge to their actual values which is expected from an adaptive system without persistent excitation. We found the controller to exhibit a similar performance for all initial positions of the rotor and reference model output.

**Figure 4.1:** Rotor position using adaptive controller based on nonlinear parametrization where initial position = 200 microns

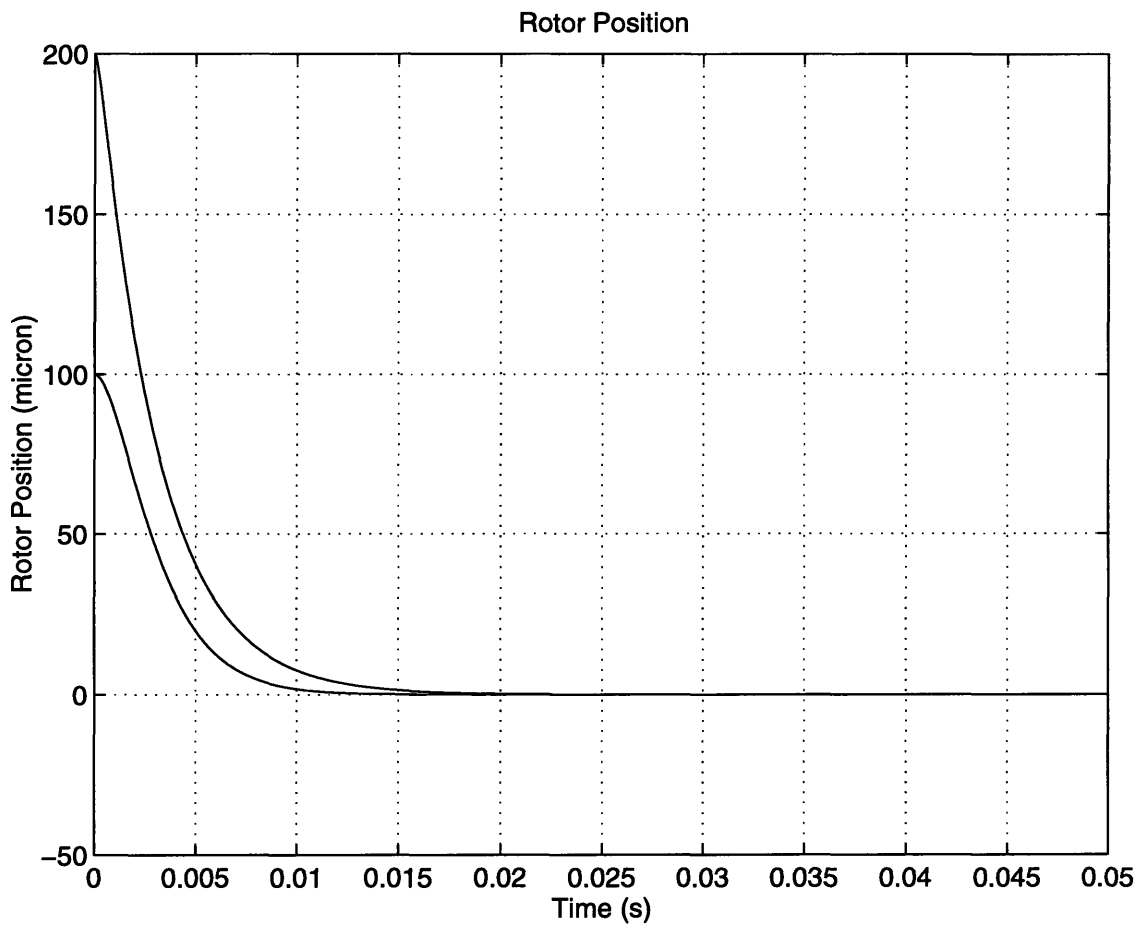




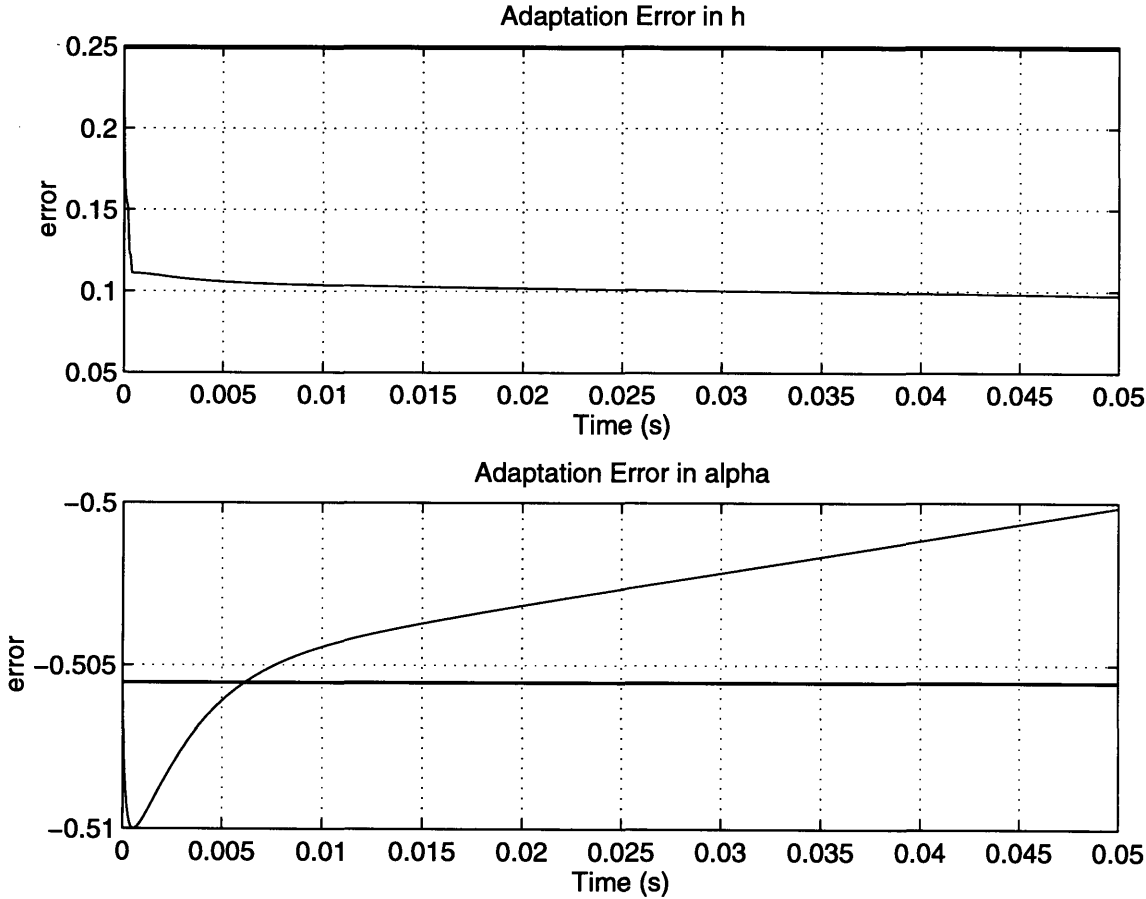
**Figure 4.2:** Adaptation errors using adaptive controller based on nonlinear parametrization where initial position = 200 microns



**Figure 4.3:** Rotor position using adaptive controller based on nonlinear parametrization where initial rotor position = 200 microns and initial reference position = 100 microns



**Figure 4.4:** Adaptation errors using adaptive controller based on nonlinear parametrization where initial rotor position = 200 microns and initial reference position = 100 microns



## 4.5 Adaptive Controller Based on Linearized Dynamics

We will now apply the adaptive control approach based on linearized bearing dynamics to the system in Eq. (4.11) [15]. In this approach, the nonlinear magnetic force represented in Eq. (4.8) is approximated in the following manner:

$$F(u, z) = f(z - z_i) + b_z u \quad (4.27)$$

$$f(z - z_i) = a_0 + a_1(z - z_i) \quad (4.28)$$

where the coefficients  $a_0$  and  $a_1$  depend on  $(z - z_i)$  and  $b_z$  is a known constant. The objective is to track the second order model defined in Eq. (4.16). The control law is chosen as:

$$u = \frac{1}{b_z}(Kz + u_{ad}) \quad (4.29)$$

where  $Kz$  is the full-state feedback component and  $u_{ad}$  is an adaptive control signal used to estimate and cancel  $f(z - z_i)$ .

We define a two-dimensional moving sphere centered at  $z_m$  in state space of radius  $\rho$  and assume that the system state is initially within the sphere. The full-state feedback term is always part of the control signal while the adaptive component is used only when the system trajectory lies outside the sphere as

$$u_{ad} = -\hat{a}_0 - \hat{a}_1(z - z_i) \quad (4.30)$$

$$\hat{a}_0 = \Gamma e^T P [0, 1]^T \quad (4.31)$$

$$\hat{a}_1 = \Gamma (z - z_i) e^T P [0, 1]^T \quad (4.32)$$

with  $z_i$  defined as the state which the system first penetrates the sphere and  $P$  is the matrix that satisfies  $A_m^T P + P A_m = -I$ .

TABLE 36.

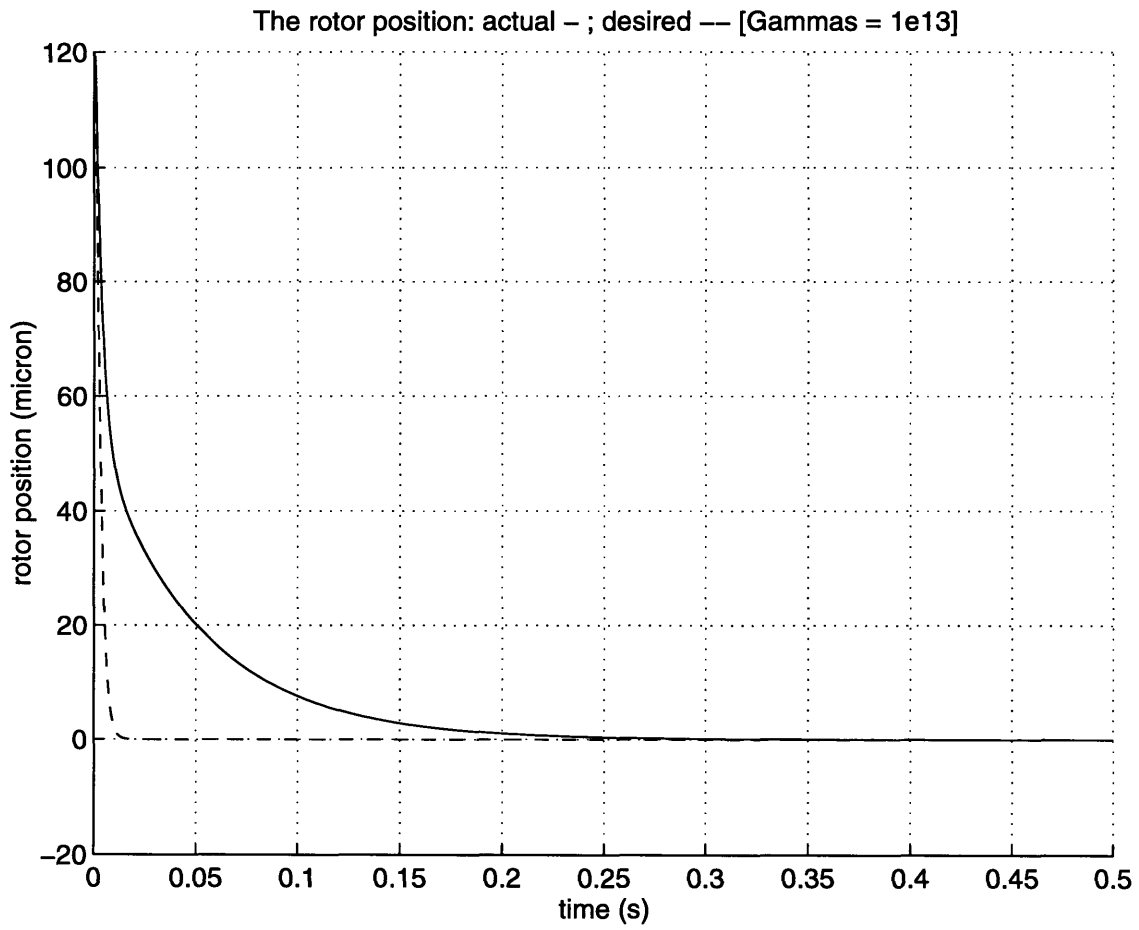
Selected Parameters: Adaptive controller based on linearized dynamics

Parameters	Values
Initial Rotor Position	120, 130 $\mu m$
Desired Rotor Position	0 $\mu m$
Feedback Gains	$[-360000 \ -1200]$
Adaptation Gains	$1e^{13}$
$\rho$	$0.5e^{-9}$
$a_0$	0
$a_1$	0
$b_z$ (Nominal)	11.5A/N

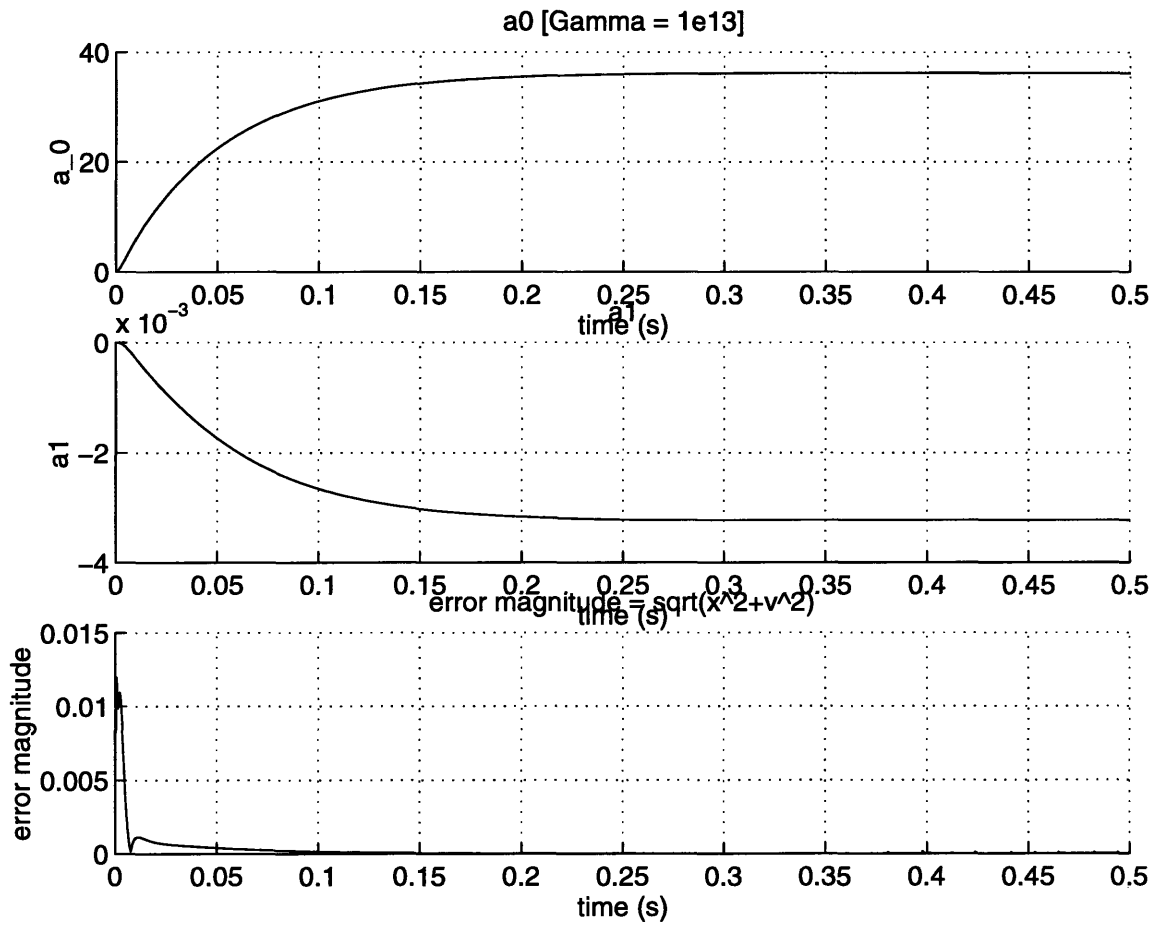
We simulated the behavior of the closed-loop system using such a controller and the results are shown in Figure 4.5 to 4.8 for the parameter values shown in the Table 36. The results from the simulations show that the controller is able to track the reference model for small initial positions of the rotor. We note that the adaptation gains in this case are rather high for the controller to successfully cancel  $f(z-z_i)$ . If these gains are reduced, we observed that steady state error in the rotor position results. Also, we found that the performance of the controller critically depends on the choice of  $b_z$ . As the deviation in  $b_z$  from its nominal value increased, the settling time increased as well, and for values outside  $[10.9, 80]$ , the controller resulted in divergence. Finally, the simulation results shows that when the initial position of the rotor exceeds half of the maximum available distance of  $200\mu m$ , the controller once again leads to a divergent behavior. This is not surprising since the assumption that the magnetic bearing force represented by Eq. (4.27) is only valid in the region sufficiently closed to the equilibrium position. It should be noted that, in con-

trast, our proposed adaptive controller accomplishes stable tracking for all initial rotor positions.

**Figure 4.5:** Rotor position using adaptive controller based on linearized dynamics with  $z_0 = 120$  microns and  $b_z = 11.5$

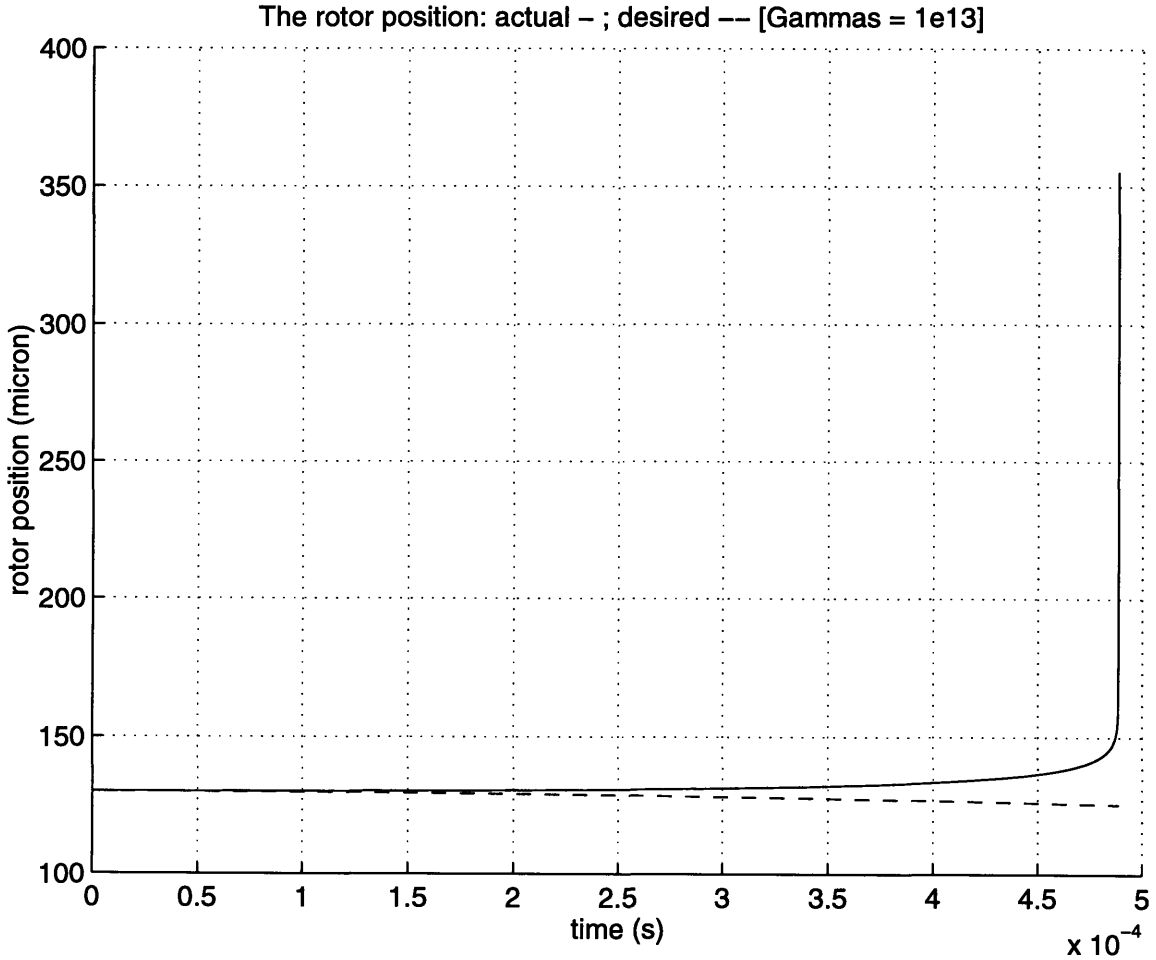


**Figure 4.6:** Adaptation parameters using adaptive controller based on linearized dynamics with  $z_0 = 120$  microns and  $b_z = 11.5$

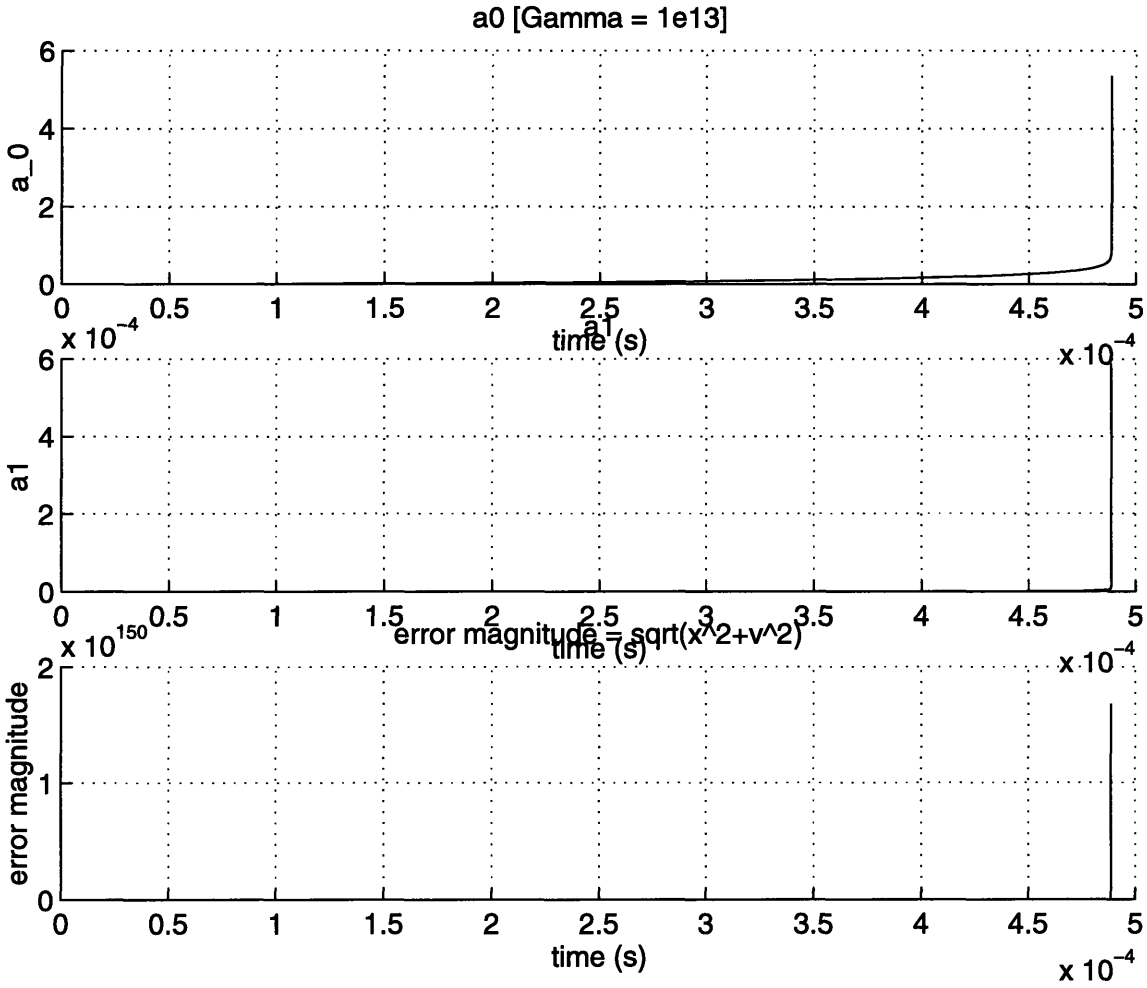




**Figure 4.7:** Rotor position using adaptive controller based on linearized dynamics with  $z_0 = 130$  microns and  $b_z = 11.5$



**Figure 4.8:** Adaptation parameters using adaptive controller based on linearized dynamics with  $z_0 = 130$  microns and  $b_z = 11.5$



## 4.6 Adaptive Controller Based on Linear Parametrization

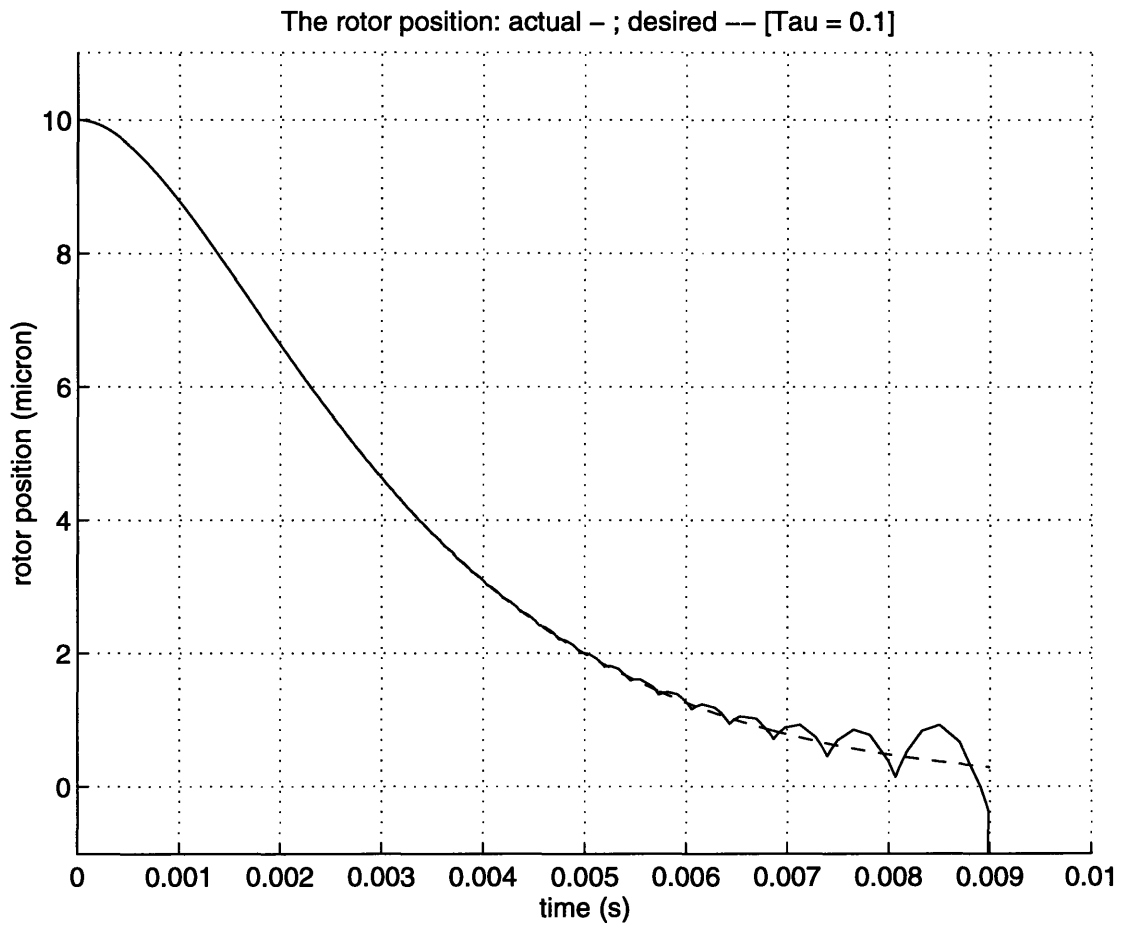
To further emphasize the necessity of nonlinear parametrization for the magnetic bearing system, an adaptive controller using linear approximation of functions  $f_i$  with respect to the parameters  $h$  and  $\alpha$  was used for comparison. We choose the control law as follows:

$$u = \frac{1}{\hat{f}_2} \{-k\varepsilon_s + r + g - \hat{f}_1 - \hat{f}_3 u^2\} \quad (4.33)$$

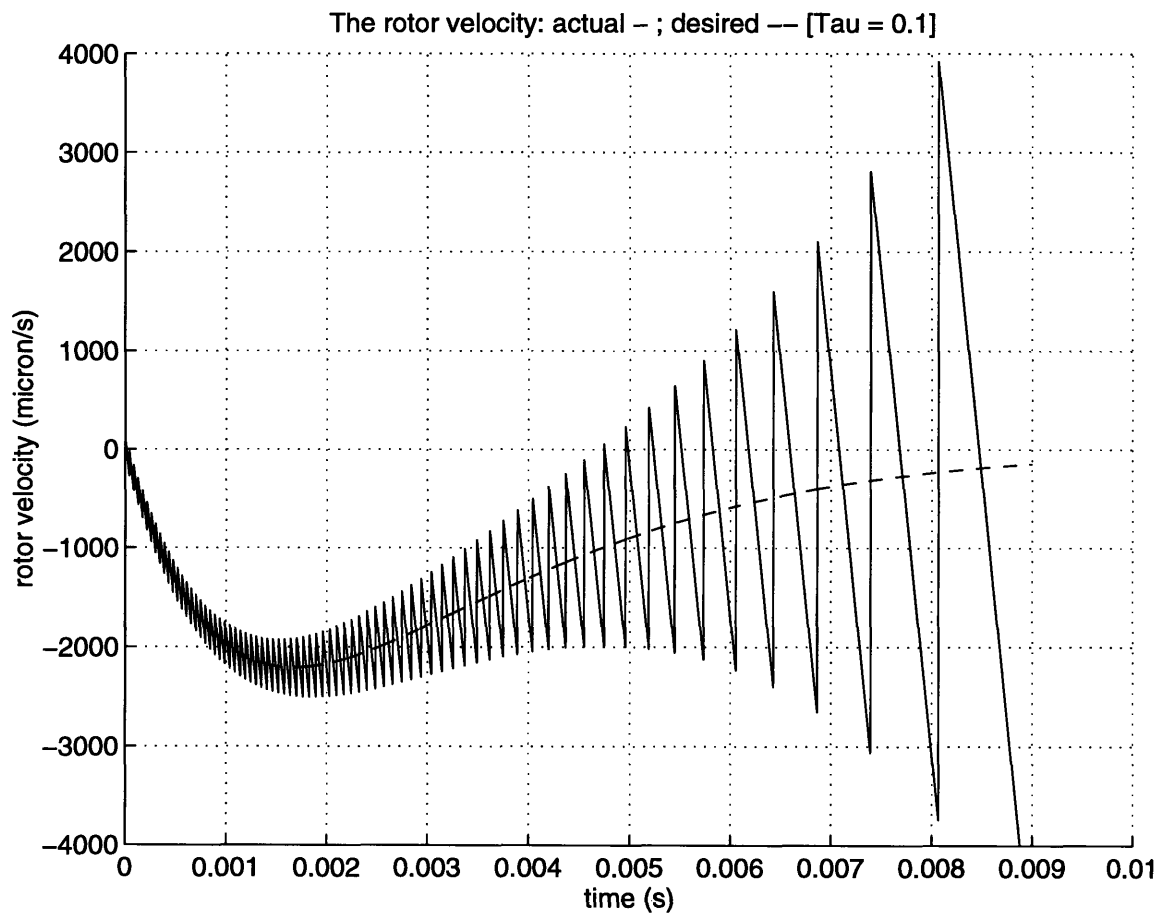
where  $\hat{f}_i$  are approximated at an operating point where  $h_i = h_0$  and  $\alpha_i = \alpha_0$ . Similarly, the adaptation law for each function is established [2].

Figure 4.9 reveals that the controller performs satisfactorily up to about 9ms beyond which the performance quickly degrades. This results from the unboundedness of the rotor velocity (see Figure 4.10 which grows exponentially over time). The parameter estimates were observed to be unbounded as well, suggesting that a nonlinear parametrization approach is warranted in this problem.

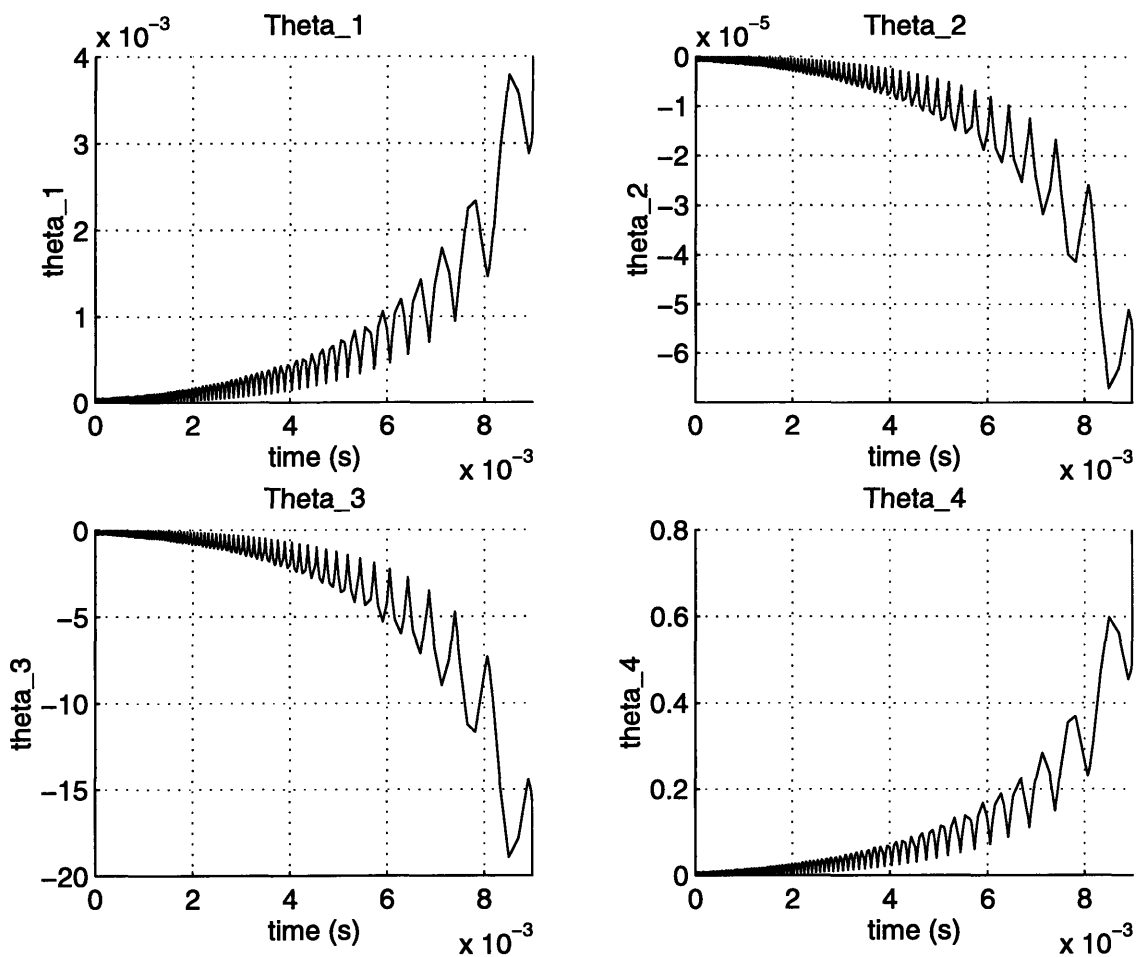
**Figure 4.9:** Rotor position using adaptive controller based on linear parametrization:  $z_0 = 10$  microns



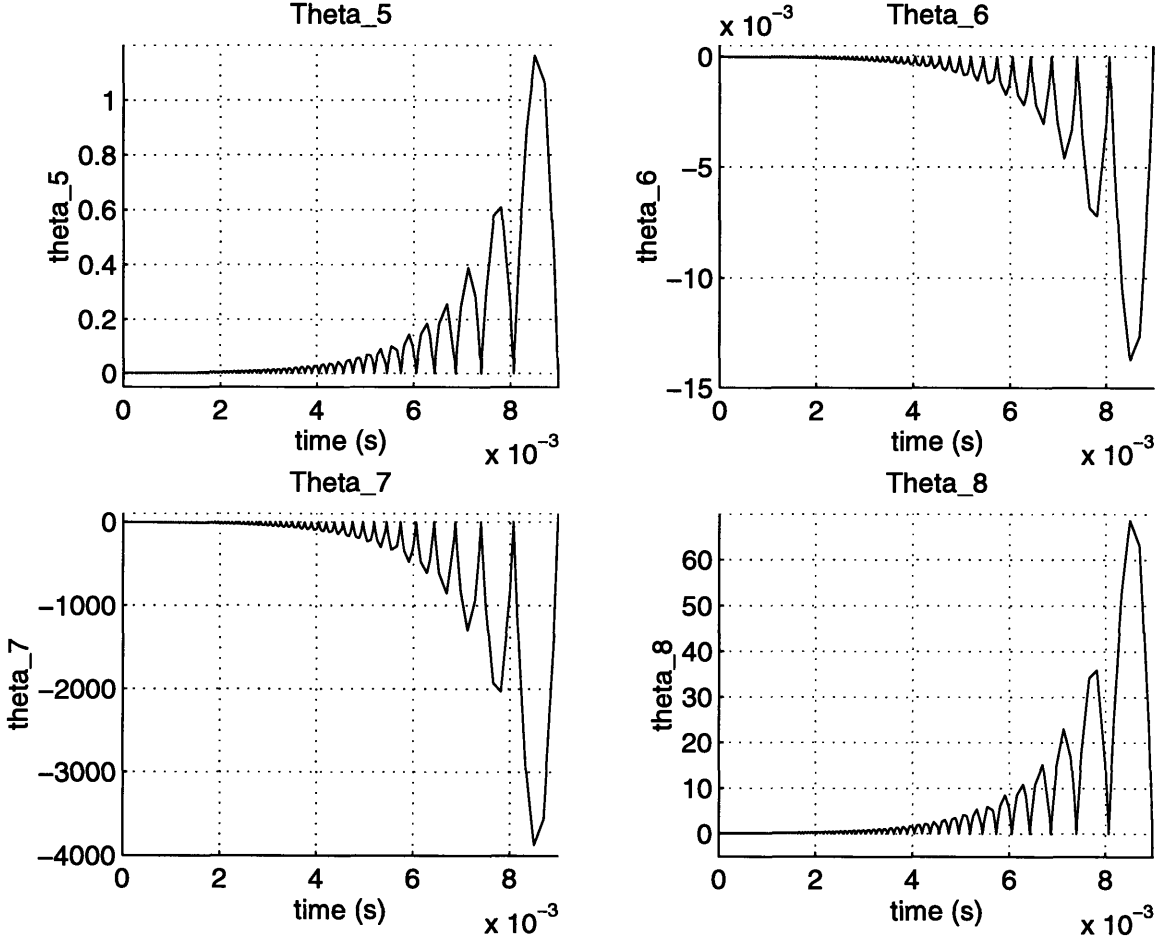
**Figure 4.10:** Rotor velocity using adaptive controller based on linear parametrization:  $z_0 = 10$  microns



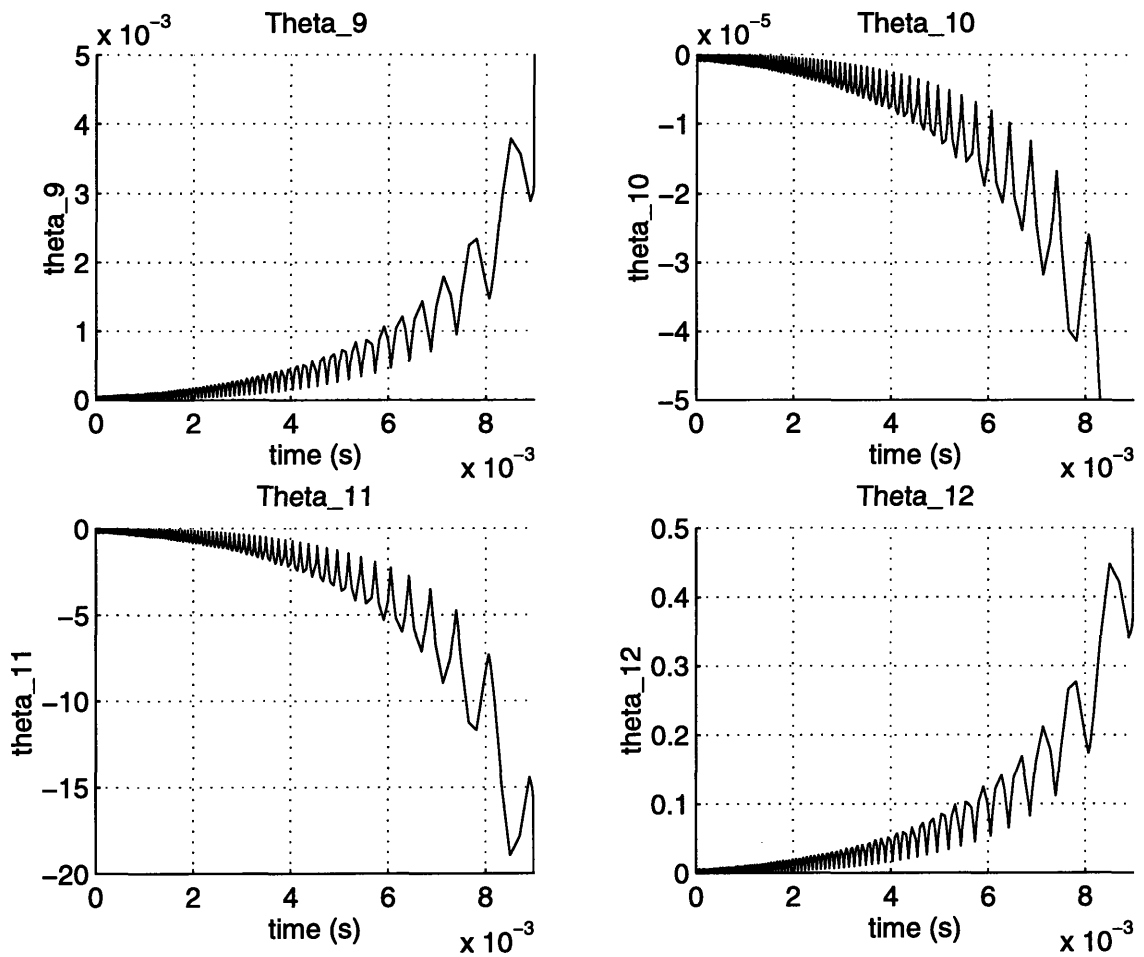
**Figure 4.11:** Adaptation parameters using adaptive controller based on linear parametrization:  $z_0 = 10$  microns



**Figure 4.12:** Adaptation parameters using adaptive controller based on linear parametrization:  $z_0 = 10$  microns



**Figure 4.13:** Adaptation parameters using adaptive controller based on linear parametrization:  $z_0 = 10$  microns





## 4.7 Summary and Remarks

We have presented in this chapter a new adaptive algorithm that is capable of stable estimation and control in dynamic systems which include nonlinear parametrization. It is shown that this algorithm is useful in magnetic bearings which is another example of such dynamic systems. In the case of magnetic bearings, the thrust force is a nonlinear function of the air-gap and as a result leads to yet another nonlinear parametrization. The use of the proposed algorithm results in better tracking error and allows a large excursion in the initial position of the rotor. In contrast, adaptive control based on either linearized dynamics or linear parametrization is seen to lead to inaccurate tracking or to instabilities when the magnitude of the initial rotor position becomes large. Given that complex dynamic systems have a strong likelihood of such nonlinear parametrization, control using the new algorithm has the potential for leading to better performance in a number of applications.



# Appendix A : MATLAB Simulation Programs

## Level Control in Feedwater Heater System

### A.1 Controller Based on Feedback Linearization

#### A.1.1 Simulation Case#1

```
%% This PL.m file is used to integrate the func_9.m file.
%% This file also uses the model_1.m file as an input.

global alpha beta xf Qc

alpha=100/35;
beta=(100/35)/(1.7*60);

t0=0;
tf=1000;
x0=[0.9 0];
xf=0.7;

[t,x]=ode23('func_9',t0,tf,x0);

n=size(x(:,1));
N=n(1,1);
theta=ones(N,1);
Q=ones(N,1);

for l = 1:N,
    Q(l,1)=alpha*(x(l,1)-xf)+beta*x(l,2);
    Qc=Q(l,1);
    theta(l,1)=fzero('model_1',20);
end

Q=(0.0000000468362*theta.^5-0.0000058202359*theta.^4+...
    0.0001816170982*theta.^3+0.00366774086313*theta.^2+...
    0.00447504781796*theta+0.01500049486870);

%% This section is used to find the settling time.
for l = 1:N,
    if abs(x(l,1)-xf) < (0.02*xf)
        t(l,1)
        x(l,1)
    else
        xxxxxx = 9999999999
    end
end

%% Finish finding the settling time

figure(1)
subplot(3,1,1), plot(t,x(:,1))
axis([0 1000 0.65 0.9])
grid
title('Conventional PI controller: Table 1')
xlabel('time (s)')
ylabel('heater level (ft)')

figure(2)
subplot(3,1,1), plot(t,theta)
axis([0 1000 11 14])
grid
title('Conventional PI controller: Table 1')
xlabel('time (s)')
ylabel('valve opening (degree)')

figure(3)
subplot(3,1,1), plot(t,Q)
axis([0 1000 0.43 0.6])
grid
title('Conventional PI controller: Table 1')
xlabel('time (s)')
ylabel('flow rate (ft^3/s)')

*****
%% This func_9.m file serves as an input to the PL.m file.
function xdot=func_9(t,x)

global alpha beta xf Qc

%% the PI-controller algorithm returns theta
Qc=alpha*(x(1)-xf)+beta*x(2);

theta=fzero('model_1',20);

%% the dynamic equation
xtract=0.4444;
area=2*sqrt(2*43.375/12*x(1)-x(1).^2)*25;

xdot(1)=1/area*(xtract-(0.0000000468362*theta.^5-0.0000058202359*...
    theta.^4-0.0001816170982*theta.^3+0.00366774086313*theta.^2+...
    2-0.00447504781796*theta+0.01500049486870));

xdot(2)=x(1)-xf;

*****
%% This model_1.m file contains the function to be evaluated
%% for zero at a particular flow rate and pressure and
%% returns the value of theta. [linear model, experimental, 300psi]
function [foo]=model_1(theta)

global Qc

foo=(0.10030772006691*theta-0.67798183834717)-Qc;
```

```

%% This fbl_p.m file is used to integrate the func_1.m file.
%% This file also uses the model.m file as an input.

global alpha xf

alpha=0.1;

t0=0;
tf=100;
x0=0.9;
xf=0.7;

[t,x]=ode23('func_1',t0,tf,x0);

n=size(x);
N=n(1,1);
theta=ones(N,1);
Q=ones(N,1);

for l = 1:N,
    xtract=0.4444;
    area=2*sqrt(2^43.375/2/12*x(l,1)-x(l,1).^2)*25;
    Q(l,1)=xtract+alpha*area*(x(l,1)-xf);
    Qc=Q(l,1);
    theta(l,1)=fzero('model',20);
end

Q=(0.00000000468362*theta.^5-0.0000058202359*theta.^4+...
    0.00001816170982*theta.^3+0.00366774086313*theta.^2-...
    0.00447504781796*theta+0.01500049486870);

%% This section is used to find the settling time.

more on
for l = 1:N,
    if abs(x(l,1)-xf) < (0.02*xf)
        t(l,1)
        x(l,1)
    else
        xxxxx = 9999999999
    end
end

more off

figure(1)
subplot(3,1,2), plot(t,x)
axis([0 100 0.65 0.95])
grid
title('Feedback linearizing P-controller: Table 2')
xlabel('time (s)')
ylabel('heater level (ft)')

figure(2)
subplot(3,1,2), plot(t,theta)
axis([0 100 11 29])
grid
title('Feedback linearizing P-controller: Table 2')
xlabel('time (s)')
ylabel('valve opening (degree)')

figure(3)
subplot(3,1,2), plot(t,Q)
axis([0 100 0.35 2.5])
grid
title('Feedback linearizing P-controller: Table 2')
xlabel('time (s)')
ylabel('flow rate (ft^3/s)')

*****
%% This func_1.m file serves as an input to the fbl_p.m file.
function xdot=func_1(t,x)

global alpha xf Qc

%% the P-controller algorithm returns theta
xtract=0.4444;
area=2*sqrt(2^43.375/2/12*x(1)-x(1).^2)*25;
Qc=xtract+alpha*area*(x(1)-xf);

theta=fzero('model',20);

%% the dynamic equation
xdot(1)=1/area*(xtract-...
(0.00000000468362*theta.^5-0.0000058202359*theta.^4+...
0.00001816170982*theta.^3+0.00366774086313*theta.^2-...
0.00447504781796*theta+0.01500049486870));

*****
%% This model.m file contains the function to be evaluated
%% for zero at a particular flow rate and pressure and
%% returns the value of theta. [5th-order model]
%% [Using the experimental data, 300psi]

function [foo]=model(theta)

global Qc

foo=0.1*(0.00000000468362*theta.^5-0.0000058202359*theta.^4-...
0.00001816170982*theta.^3+0.00366774086313*theta.^2-...
0.00447504781796*theta+0.01500049486870)-Qc;

*****

```

```

%% This fbl_pi.m file is used to integrate the func_2.m file.
%% This file also uses the model.m file as an input.

global alpha beta xf Qc

alpha=0.1;
beta=0.01;

t0=0;
tf=300;
x0=[0.9 0];
xf=0.7;

[t,x]=ode23('func_2',t0,tf,x0);

n=size(x(:,1));
N=n(1,1);
theta=ones(N,1);
Q=ones(N,1);

for l = 1:N,
    xtract=0.4444;
    area=2*sqrt(2*43.375/2/12*x(l,1)-x(l,1).^2)*25;
    Q(l,1)=xtract+alpha*area*(x(l,1)-xf)+beta*area*x(l,2);
    Qc=Q(l,1);
    theta(l,1)=fzero('model',20);
end

Q=(0.00000000468362*theta.^5-0.0000058202359*theta.^4-...
    0.00001816170982*theta.^3+0.00366774086313*theta.^2-...
    0.00447504781796*theta+0.01500049486870);

%% This section is used to find the settling time.
more on
for l = 1:N,
    if abs(x(l,1)-xf) < (0.02*xf)
        t(l,1)
        x(l,1)
    else
        xxxxxx = 9999999999
    end
end
more off

figure(1)
subplot(3,1,3), plot(t,x(:,1))
axis([0 200 0.59 0.91])
grid
title('Feedback linearizing PI-controller: Table 3')
xlabel('time (s)')
ylabel('heater level (ft)')

figure(2)
subplot(3,1,3), plot(t,theta)
axis([0 200 7 29])
grid
title('Feedback linearizing PI-controller: Table 3')
xlabel('time (s)')
ylabel('valve opening (degree)')

figure(3)
subplot(3,1,3), plot(t,Q)
axis([0 200 0 2.1])
grid
title('Feedback linearizing PI-controller: Table 3')
xlabel('time (s)')
ylabel('flow rate (ft^3/s)')

*****
%% This func_2.m file serves as an input to the fbl_pi.m file.
function xdot=func_2(t,x)

global alpha beta xf Qc

%% the PI-controller algorithm returns theta
xtract=0.4444;
area=2*sqrt(2*43.375/2/12*x(1)-x(1).^2)*25;
Qc=xtract+alpha*area*(x(1)-xf)+beta*area*x(2);

theta=fzero('model',20);

%% the dynamic equation
xdot(1)=1/area*(xtract-...
    (0.00000000468362*theta.^5-0.0000058202359*theta.^4-...
    0.00001816170982*theta.^3+0.00366774086313*theta.^2-...
    0.00447504781796*theta+0.01500049486870));

xdot(2)=x(1)-xf;

*****
%% This model.m file contains the function to be evaluated
%% for zero at a particular flow rate and pressure and
%% returns the value of theta. [5th-order model]
%% [Using the experimental data, 300psi]
function [foo]=model(theta)

global Qc

foo=0.1*(0.00000000468362*theta.^5-0.0000058202359*theta.^4-...
    0.00001816170982*theta.^3+0.00366774086313*theta.^2-...
    0.00447504781796*theta+0.01500049486870)-Qc;

*****

```

## A.1.2 Simulation Case#2

```

%% This PI_1.m file is used to integrate the func_13.m file.
%% This file also uses the model_1.m file as an input.

global alpha beta xf Qc

alpha=100/35;
beta=(100/35)/(1.7*60);

t0=0;
tf=2000;
x0=(0.9 0);
xf=0.7;

[t,x]=ode23('func_13',t0,tf,x0);

n=size(x(:,1));
N=n(1,1);
theta=ones(N,1);
Q=ones(N,1);

for l = 1:N,
    Q(l,1)=alpha*(x(l,1)-xf)+beta*x(l,2);
    Qc=Q(l,1);
    theta(l,1)=fzero('model_1',20);
end

Q=(0.0000000468362*theta.^5-0.0000058202359*theta.^4-...
    0.0001816170982*theta.^3+0.00366774086313*theta.^2-...
    0.00447504781796*theta+0.01500049486870);

%% This section is used to find the settling time.

more on
for l = 1:N,
    if abs(x(l,1)-xf) < (0.02*xf)
        t(l,1)
        x(l,1)
    else
        xxxxxx = 9999999999
    end
end
more off

%% Finish finding the settling time

figure(1)
subplot(3,1,1), plot(t,x(:,1))
axis([0 1000 0 2.1])
grid
title('Conventional PI controller: Table 4')
xlabel('time (s)')
ylabel('heater level (t)')

figure(2)
subplot(3,1,1), plot(t,theta)
%axis([0 1000 11 14])
grid
title('Conventional PI controller: Table 4')
xlabel('time (s)')
ylabel('valve opening (degree)')

figure(3)
subplot(3,1,1), plot(t,Q)
%axis([0 1000 0.43 0.6])
grid
title('Conventional PI controller: Table 4')
xlabel('time (s)')
ylabel('flow rate (t^3/s)')

*****
%% This func_13.m file serves as an input to the PI_1.m file.

function xdot=func_13(t,x)

global alpha beta xf Qc

%% the PI-controller algorithm returns theta
Qc=alpha*(x(1)-xf)+beta*x(2);

theta=fzero('model_1',20);

%% the dynamic equation

xtract=11.5*0.4444;
area=2*sqrt(2*43.375/2/12*x(1)-x(1).^2)*25;

xdot(1)=1/area*(xtract-(0.0000000468362*theta.^5-0.0000058202359*...
    theta.^4-0.0001816170982*theta.^3+0.00366774086313*theta.^...
    2-0.00447504781796*theta+0.01500049486870));

xdot(2)=x(1)-xf;

*****
%% This model_1.m file contains the function to be evaluated
%% for zero at a particular flow rate and pressure and
%% returns the value of theta. [linear model, experimental, 300psi]

function [foo]=model_1(theta)

global Qc

foo=(0.10030772006691*theta-0.67798183834717)-Qc;

*****

```

```

%%% This fbl_p_0.m file is used to integrate the func_4.m file. %%%
%%% This file also uses the model.m file as an input. %%%

global alpha xf
alpha=0.1;

t0=0;
tf=100;
x0=0.9;
xf=0.7;

[t,x]=ode23('func_4',t0,tf,x0);

n=size(x);
N=n(1,1);
theta=ones(N,1);
Q=ones(N,1);

for l = 1:N,

xtract=0;
area=2*sqrt(2*43.375/2/12*x(l,1)-x(l,1).^2)*25;
Q(l,1)=xtract+alpha*area*(x(l,1)-xf);

Qc=Q(l,1);
theta(l,1)=fzero('model',50);

end

Q=(0.0000000468362*theta.^5-0.0000058202359*theta.^4-...
0.00001816170982*theta.^3+0.00366774086313*theta.^2-...
0.00447504781796*theta+0.01500049486870);

%%% This section is used to find the settling time. %%%

more on
for l = 1:N,

ifabs(x(l,1)-xf) < (0.02*xf)
t(l,1)
x(l,1)
else
xxxxxx = 9999999999
end

end
more off

%%% Finish finding the settling time %%%

figure(1)
subplot(3,1,2), plot(t,x)
axis([0 100 0.85 1.35])
grid
title('Feedback linearizing P-controller: Table 5')
xlabel('time (s)')
ylabel('heater level (ft)')

figure(2)
subplot(3,1,2), plot(t,theta)
axis([0 100 20.5 60.5])
grid
title('Feedback linearizing P-controller: Table 5')
xlabel('time (s)')
ylabel('valve opening (degree)')

figure(3)
subplot(3,1,2), plot(t,Q)
axis([0 100 1 5.95])
grid
title('Feedback linearizing P-controller: Table 5')
xlabel('time (s)')
ylabel('flow rate (ft^3/s)')

*****
%%% This func_4.m file serves as an input to the fbl_p_0.m file. %%%

function xdot=func_4(t,x)

global alpha xf Qc

%%% the P-controller algorithm returns theta %%%
xtract=0;
area=2*sqrt(2*43.375/2/12*x(1)-x(1).^2)*25;
Qc=xtract+alpha*area*(x(1)-xf);

theta=fzero('model',50);

%%% the dynamic equation %%%
xtract=11.5*0.4444;
xdot(1)=1/area*(xtract-...
(0.0000000468362*theta.^5-0.0000058202359*theta.^4-...
0.00001816170982*theta.^3+0.00366774086313*theta.^2-...
0.00447504781796*theta+0.01500049486870));

*****
%%% This model.m file contains the function to be evaluated %%%
%%% for zero at a particular flow rate and pressure %%%
%%% returns the value of theta. [5th-order mode] %%%
%%% [Using the experimental data, 300psi] %%%

function [foo]=model(theta)

global Qc

foo=0.1*(0.0000000468362*theta.^5-0.0000058202359*theta.^4-...
0.00001816170982*theta.^3+0.00366774086313*theta.^2-...
0.00447504781796*theta+0.01500049486870)-Qc;

*****

```

```

%% This fbl_pi_0.m file is used to integrate the func_3.m file.
%% This file also uses the model.m file as an input.

global alpha beta xf Qc

alpha=0.1;
beta=0.01;

t0=0;
tf=100;
x0=[0.9 0];
xf=0.7;

[t,x]=ode23('func_3',t0,tf,x0);

n=size(x(:,1));
N=ceil(n/10);
theta=ones(N,1);
Q=ones(N,1);

for l = 1:N,
    xtract=0.0;
    area=2*sqrt(2*43.375/12*x(l,1)-x(l,1).^2)+25;
    Q(l,1)=xtract+alpha*area*(x(l,1)-xf)+beta*area*x(l,2);
    Qc=Q(l,1);
    theta(l,1)=fzero('model',80);
end

Q=(0.00000000468362*theta.^5-0.00000058202359*theta.^4-...
    0.0001816170982*theta.^3+0.00366774086313*theta.^2-...
    0.00447504781796*theta+0.01500049486870);

%% This section is used to find the settling time.
more on
for l = 1:N,
    if abs(x(l,1)-xf) < (0.02*xf)
        t(l,1)
        x(l,1)
    else
        xxxxxx = 9999999999
    end
end
more off

%% Finish finding the settling time

figure(1)
subplot(3,1,3), plot(t,x(:,1))
title('Feedback linearizing PI-controller: Table 6')
axis([0 100 0.61 1.11])
grid
xlabel('time (s)')
ylabel('beater level (ft)')

figure(2)
subplot(3,1,3), plot(t,theta)
%axis([0 100 8 28])
grid
title('Feedback linearizing PI-controller: Table 6')
xlabel('time (s)')
ylabel('valve opening (degree)')

figure(3)
subplot(3,1,3), plot(t,Q)
axis([0 100 0 8])
grid
title('Feedback linearizing PI-controller: Table 6')
xlabel('time (s)')
ylabel('flow rate (ft^3/s)')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% This func_3.m file serves as an input to the fbl_pi_0.m file.
function xdot=func_3(t,x)

global alpha beta xf Qc

%% the PI-controller algorithm returns theta
xtract=0.0;
area=2*sqrt(2*43.375/12*x(1)-x(1).^2)+25;
Qc=xtract+alpha*area*(x(1)-xf)+beta*area*x(2);
theta=fzero('model',80);

%% the dynamic equation
xtract=11.5*0.4444;

xdot(1)=1/area*(xtract-...
    (0.00000000468362*theta.^5-0.00000058202359*theta.^4-...
    0.0001816170982*theta.^3+0.00366774086313*theta.^2-...
    0.00447504781796*theta+0.01500049486870));

xdot(2)=x(1)-xf;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% This model.m file contains the function to be evaluated
%% for zero at a particular flow rate and pressure and
%% returns the value of theta. [5th-order model]
%% [Using the experimental data, 300psi]
function [foo]=model(theta)

global Qc

foo=0.1*(0.00000000468362*theta.^5-0.00000058202359*theta.^4-...
    0.0001816170982*theta.^3+0.00366774086313*theta.^2-...
    0.00447504781796*theta+0.01500049486870)-Qc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```



### A.1.3 Simulation Case#3

```

%% This P1_1.m file is used to integrate the func_14.m file.
%% This file also uses the model_2.m file as an input.

global alpha beta xf Qc

alpha=100/35;
beta=(100/35)/(1.7*60);

t0=0;
tf=1000;
x0=[0.9 0];
xf=0.7;

[t,x]=ode23('func_14',t0,tf,x0);

n=size(x(:,1));
N=n(1,1);
theta=ones(N,1);
Q=ones(N,1);

for l = 1:N,
Q(l,1)=alpha*(x(l,1)-xf)+beta*x(l,2);
Qc=Q(l,1);
theta(l,1)=fzero('model_2',20);
end

Q=(0.0000000468362*theta.^5-0.0000058202359*theta.^4-...
0.00001816170982*theta.^3+0.00366774086313*theta.^2-...
0.00447504781796*theta+0.01500049486870);

%% This section is used to find the settling time.
more on
for l = 1:N,
if abs(x(l,1)-xf) < (0.02*xf)
t(l,1)
x(l,1)
else
xxxxx = 9999999999
end
end
more off

%% Finish finding the settling time

figure(1)
subplot(3,1,1), plot(t,x(:,1))
axis([0 1000 0.61 0.99])
grid
title('Conventional PI controller: Table 7')
xlabel('time (s)')
ylabel('heater level (ft)')

figure(2)
subplot(3,1,1), plot(t,theta)
axis([0 1000 11 14])
grid
title('Conventional PI controller: Table 7')
xlabel('time (s)')
ylabel('valve opening (degree)')

figure(3)
subplot(3,1,1), plot(t,Q)
axis([0 1000 0.43 0.6])
grid
title('Conventional PI controller: Table 7')
xlabel('time (s)')
ylabel('flow rate (ft^3/s)')

*****
%% This func_14.m file serves as an input to the P1_2.m file.
function xdot=func_14(t,x)

global alpha beta xf Qc

%% the PI-controller algorithm returns theta
Qc=alpha*(x(1)-xf)+beta*x(2);

theta=fzero('model_2',20);

%% the dynamic equation
area=0.4444;
area=2*sqrt(2*43.375/12*x(1)-x(1).^2)*25;

xdot(1)=1/area*(xtract-(0.0000000468362*theta.^5-0.0000058202359*...
theta.^4-0.00001816170982*theta.^3+0.00366774086313*theta.^2-...
2*0.00447504781796*theta+0.01500049486870));

xdot(2)=x(1)-xf;

*****
%% This model_2.m file contains the function to be evaluated
%% for zero at a particular flow rate and pressure and
%% returns the value of theta. [linear model, Man, 300psi]
function [foo]=model_2(theta)

global Qc

foo=(0.09062981695704*theta-0.66225439803445)-Qc;

*****

```

```

%% This fbl_p_1.m file is used to integrate the func_5.m file.
%% This file also uses the model_0.m file as an input.

global alpha xf

alpha=0.1;

t0=0;
tf=100;
x0=0.9;
xf=0.7;

[t,x]=ode23('func_5',t0,tf,x0);

n=size(x);
N=n(1,1);
theta=ones(N,1);
Q=ones(N,1);

for l = 1:N,
    xtract=0.4444;
    area=2*sqrt(2*43.375/2/12*x(l,1)-x(l,1).^2)*25;
    Q(l,1)=xtract-alpha*area*(x(l,1)-xf);

    Qc=Q(l,1);
    theta(l,1)=fzero('model_0',20);
end

Q=(0.0000000468362*theta.^5-0.0000058202359*...
theta.^4-0.00001816170982*theta.^3+0.00366774086313*theta.^...
2-0.00447504781796*theta+0.01500049486870);

%% This section is used to find the settling time.

more on
for l = 1:N,
    if abs(x(l,1)-xf) < (0.02*xf)
        t(l,1)
        x(l,1)
    else
        xxxxxx = 9999999999
    end
end
more off

%% Finish finding the settling time

figure(1)
subplot(3,1,2), plot(t,x)
axis([0 100 0.65 0.95])
grid
title('Feedback linearizing P-controller: Table 8')
xlabel('time (s)')
ylabel('heater level (ft)')

figure(2)
subplot(3,1,2), plot(t,theta)
axis([0 100 9 31])
grid
title('Feedback linearizing P-controller: Table 8')
xlabel('time (s)')
ylabel('valve opening (degree)')

figure(3)
subplot(3,1,2), plot(t,Q)
axis([0 100 0.1 2.5])
grid
title('Feedback linearizing P-controller: Table 8')
xlabel('time (s)')
ylabel('flow rate (ft^3/s)')

*****
%% This func_5.m file serves as an input to the fbl_p_1.m file.

function xdot=func_5(L,x)

global alpha xf Qc

%% the P-controller algorithm returns theta
xtract=0.4444;
area=2*sqrt(2*43.375/2/12*x(1)-x(1).^2)*25;
Qc=xtract-alpha*area*(x(1)-xf);

theta=fzero('model_0',20);

%% the dynamic equation
xdot(1)=1/area*(xtract-...
(0.0000000468362*theta.^5-0.0000058202359*...
theta.^4-0.00001816170982*theta.^3+0.00366774086313*theta.^...
2-0.00447504781796*theta+0.01500049486870));

*****
%% This model_0.m file contains the function to be evaluated
%% for zero at a particular flow rate and pressure and
%% returns the value of theta. [5th-order model, manufacturer's]
%% At 300psi.

function [foo]=model_0(theta)

global Qc

foo=(0.0000000533730*theta.^5-0.0000079813121*theta.^4+...
0.00000662324144*theta.^3+0.00259478501535*theta.^2+...
0.00045531029014*theta+0.01186735930655)-Qc;

*****

```

```

%% This fbl_pi_1.m file is used to integrate the func_7.m file
%% This file also uses the model_0.m file as an input.

global alpha beta xf Qc

alpha=0.1;
beta=0.01;

t0=0;
tf=100;
x0=[0.9 0];
xf=0.7;

[t,x]=ode23('func_7',t0,tf,x0);

n=size(x(:,1));
N=n(1,1);
theta=ones(N,1);
Q=ones(N,1);

for l = 1:N,
xtract=0.4444;
area=2*sqrt(2*43.375/12*x(l,1)-x(l,1).^2)*25;
Q(l,1)=xtract+alpha*area*(x(l,1)-xf)+beta*area*x(l,2);

Qc=Q(l,1);
theta(l,1)=fzero('model_0',20);
end

Q=(0.0000000468362*theta.^5-0.0000058202359*...
theta.^4-0.00001816170982*theta.^3+0.00366774086313*theta.^...
2-0.00447504781796*theta+0.01500049486870);

%% This section is used to find the settling time.
more on
for l = 1:N,
ifabs(x(l,1)-xf) < (0.02*xl)
t(l,1)
x(l,1)
else
xxxxxx = 9999999999
end
end
more off

%% Finish finding the settling time

figure(1)
subplot(3,1,3), plot(t,x(:,1))
axis([0 100 0.6 0.91])
grid
title('Feedback linearizing PI-controller: Table 9')
xlabel('time (s)')
ylabel('heater level (ff)')

figure(2)
subplot(3,1,3), plot(t,theta)
axis([0 100 5 32])
grid
title('Feedback linearizing PI-controller: Table 9')
xlabel('time (s)')
ylabel('valve opening (degree)')

figure(3)
subplot(3,1,3), plot(t,Q)
axis([0 100 0.1 2.5])
grid
title('Feedback linearizing PI-controller: Table 9')
xlabel('time (s)')
ylabel('flow rate (ft^3/s)')

*****
%% This func_7.m file serves as an input to the fbl_pi_1.m file.
function xdot=func_7(t,x)
global alpha beta xf Qc

%% the PI-controller algorithm returns theta
xtract=0.4444;
area=2*sqrt(2*43.375/12*x(1)-x(1).^2)*25;
Qc=xtract+alpha*area*(x(1)-xf)+beta*area*x(2);

theta=fzero('model_0',20);

%% the dynamic equation
xdot(1)=1/area*(xtract-...
(0.0000000468362*theta.^5-0.0000058202359*...
theta.^4-0.00001816170982*theta.^3+0.00366774086313*theta.^...
2-0.00447504781796*theta+0.01500049486870));

xdot(2)=x(1)-xf;

*****
%% This model_0.m file contains the function to be evaluated
%% for zero at a particular flow rate and pressure and
%% returns the value of theta. [5th-order model, manufacturer's]
%% A1 300psi.
function [foo]=model_0(theta)
global Qc
foo=(0.0000000533730*theta.^5-0.0000079813121*theta.^4+...
0.00000662324144*theta.^3+0.00259478501535*theta.^2+...
0.00045531029014*theta+0.01186735930655)-Qc;
*****

```

## A.1.4 Simulation Case#4

```

%% This PI_3.m file is used to integrate the func_15.m file.
%% This file also uses the model_1.m file as an input.

global alpha beta xf Qc

alpha=100/35;
beta=(100/35)/(1.7*60);

t0=0;
tf=1000;
x0=[0.9 0];
xf=0.7;

[t,x]=ode23('func_15',t0,tf,x0);

n=size(x(:,1));
N=n(1,1);
theta=ones(N,1);
Q=ones(N,1);

for l = 1:N,
    Q(l,1)=alpha*(x(l,1)-xf)+beta*x(l,2);
end

Qc=Q(1,1);
theta(1,1)=fzero('model_1',20);

Q=(0.0000000781871*theta.^5-0.0000099606931*theta.^4-...
    0.00002536546426*theta.^3+0.00583612263799*theta.^2-...
    0.00658991522416*theta+0.02412600207374);

%% This section is used to find the settling time.
more on
for l = 1:N,
    if abs(x(l,1)-xf) < (0.02*xf)
        t(l,1)
        x(l,1)
    else
        xxxxxx = 9999999999
    end
end
more off

%% Finish finding the settling time
figure(1)
subplot(3,1,1), plot(Lx(:,1))
axis([0 1000 0.65 0.9])
grid
title('Conventional PI controller: Table 10')
xlabel('time (s)')
ylabel('heater level (ft)')

figure(2)
subplot(3,1,1), plot(Ltheta)
axis([0 1000 8 13])
grid
title('Conventional PI controller: Table 10')
xlabel('time (s)')
ylabel('valve opening (degree)')

figure(3)
subplot(3,1,1), plot(LQ)
axis([0 1000 0.43 0.77])
grid
title('Conventional PI controller: Table 10')
xlabel('time (s)')
ylabel('flow rate (ft^3/s)')

*****
%% This func_15.m file serves as an input to the PI_3.m file.
function xdot=func_15(Lx)

global alpha beta xf Qc

%% the PI-controller algorithm returns theta
Qc=alpha*(x(1)-xf)+beta*x(2);

theta=fzero('model_1',20);

%% the dynamic equation
xtract=0.4444;
area=2*sqrt(2*43.375/2/12*x(1)-x(1).^2)*25;

xdot(1)=1/area*(xtract-...
    (0.0000000781871*theta.^5-0.0000099606931*theta.^4-...
    0.00002536546426*theta.^3+0.00583612263799*theta.^2-...
    0.00658991522416*theta+0.02412600207374));

xdot(2)=x(1)-xf;

*****
%% This model_1.m file contains the function to be evaluated
%% for zero at a particular flow rate and pressure and
%% returns the value of theta. [linear model, experimental, 300psi]
function [foo]=model_1(theta)

global Qc

foo=(0.10030772006691*theta-0.67798183834717)-Qc;

*****

```

```

%% This fbl_p_2.m file is used to integrate the func_6.m file.
%% This file also uses the model.m file as an input.

global alpha xf

alpha=0.1;

t=0;
tf=1000;
x0=0.9;
xf=0.7;

[t,x]=ode23('func_6',t0,tf,x0);

n=size(x);
N=n(1,1);
theta=ones(N,1);
Q=ones(N,1);

for l = 1:N,
    xtract=0.4444;
    area=2*sqrt(2*43.375/12*x(l,1)-x(l,1)^2)*25;
    Q(l,1)=xtract+alpha*area*(x(l,1)-xf);

    Qc=Q(l,1);
    theta(l,1)=fzero('model',30);
end

Q=(0.0000000781871*theta.^5-0.0000099606931*theta.^4-...
    0.00002536546426*theta.^3+0.00583612263799*theta.^2-...
    0.00658991522416*theta+0.02412600207374);

%% This section is used to find the settling time.
more on
for l = 1:N,
    if abs(x(l,1)-xf) < (0.02*xf)
        t(l,1)
        x(l,1)
    else
        xxxxxx = 9999999999
    end
end
more off

%% Finish finding the settling time
figure(1)
subplot(3,1,2), plot(t,x)
axis([0 100 0.62 0.91])
grid
title('Feedback linearizing P-controller; Table 11')
xlabel('time (s)')
ylabel('heater level (ft)')

figure(2)
subplot(3,1,2), plot(t,theta)
axis([0 100 7 30])
grid
title('Feedback linearizing P-controller; Table 11')
xlabel('time (s)')
ylabel('valve opening (degree)')

figure(3)
subplot(3,1,2), plot(t,Q)
axis([0 100 0.2 3.5])
grid
title('Feedback linearizing P-controller; Table 11')
xlabel('time (s)')
ylabel('flow rate (ft^3/s)')

*****
%% This func_6.m file serves as an input to the fbl_p_2.m file.
function xdot=func_6(t,x)

global alpha xf Qc

%% the P-controller algorithm returns theta
xtract=0.4444;
area=2*sqrt(2*43.375/12*x(1)-x(1)^2)*25;
Qc=xtract+alpha*area*(x(1)-xf);

theta=fzero('model',30);

%% the dynamic equation
xtract=0.4444;
xdot(1)=1/area*(xtract-...
(0.0000000781871*theta.^5-0.0000099606931*theta.^4-...
0.00002536546426*theta.^3+0.00583612263799*theta.^2-...
0.00658991522416*theta+0.02412600207374));

*****
%% This model.m file contains the function to be evaluated
%% for zero at a particular flow rate and pressure
%% returns the value of theta. [5th-order model]
%% Using the experimental data, 300psi
function [foo]=model(theta)

global Qc

foo=0.1*(0.0000000468362*theta.^5-0.0000058202359*theta.^4-...
0.00001816170982*theta.^3+0.00366774086313*theta.^2-...
0.00447504781796*theta+0.01500049486870)-Qc;

*****

```

```

%% This fbl_pi_4.m file is used to integrate the func_11.m file.
%% This file also uses the model.m file as an input.

global alpha beta xf Qc;
alpha=0.1;
beta=0.01;

t0=0;
tf=100;
x0=[0.9 0];
xf=0.7;

[t,x]=ode23('func_11',t0,tf,x0);

n=size(x(:,1));
N=n(1,1);
theta=ones(N,1);
Q=ones(N,1);

for l = 1:N,
xtract=0.4444;
area=2*sqrt(2*43.375/2/12*x(l,1)-x(l,1).^2)*25;
Q(l,1)=xtract+alpha*area*(x(l,1)-xf)+beta*area*x(l,2);

Qc=Q(l,1);
theta(l,1)=fzero('model',20);
end

Q=(0.0000000781871*theta.^5-0.0000099606931*theta.^4-...
0.00002536546426*theta.^3+0.00583612263799*theta.^2-...
0.00658991522416*theta+0.02412600207374);

%% This section is used to find the settling time.
more on
for l = 1:N,
if abs(x(l,1)-xf) < (0.02*xf)
t(l,1)
x(l,1)
else
xxxxxx = 9999999999
end
end
more off

%% Finish finding the settling time
figure(1)
subplot(3,1,3), plot(t,x(:,1))
axis([0 100 0.61 0.91])
grid
title('Feedback linearizing PI-controller: Table 12')
xlabel('time (s)')
ylabel('heater level (ft)')

figure(2)
subplot(3,1,3), plot(t,theta)
axis([0 100 5 28])
grid
title('Feedback linearizing PI-controller: Table 12')
xlabel('time (s)')
ylabel('valve opening (degree)')

figure(3)
subplot(3,1,3), plot(t,Q)
axis([0 100 0.1 3.5])
grid
title('Feedback linearizing PI-controller: Table 12')
xlabel('time (s)')
ylabel('flow rate (ft^3/s)')

*****
%% This func_11.m file serves as an input to the fbl_pi_4.m file.
function xdot=func_11(t,x)

global alpha beta xf Qc;

%% the PI-controller algorithm returns theta
xtract=0.4444;
area=2*sqrt(2*43.375/2/12*x(1)-x(1).^2)*25;
Qc=xtract+alpha*area*(x(1)-xf)+beta*area*x(2);

theta=fzero('model',20);

%% the dynamic equation
xtract=0.4444;

xdot(1)=1/area*(xtract-...
(0.0000000781871*theta.^5-0.0000099606931*theta.^4-...
0.00002536546426*theta.^3+0.00583612263799*theta.^2-...
0.00658991522416*theta+0.02412600207374));

xdot(2)=x(1)-xf;

*****
%% This model.m file contains the function to be evaluated
%% for zero at a particular flow rate and pressure and
%% returns the value of theta. [5th-order model]
%% [Using the experimental data, 300psi]

function [foo]=model(theta)

global Qc

foo=0.1*(0.0000000468362*theta.^5-0.0000058202359*theta.^4-...
0.00001816170982*theta.^3+0.00366774086313*theta.^2-...
0.00447504781796*theta+0.01500049486870)-Qc;

*****

```

## A.1.5 Simulation Case#5

```

%% This PI_4.m file is used to integrate the func_16.m file.
%% This file also uses the model_2.m file as an input.

global alpha beta xf Qc

alpha=100/35;
beta=(100/35)/(1.7*60);

t0=0;
tf=1000;
x0=[0.9 0];
xf=0.7;

[t,x]=ode23('func_16',t0,tf,x0);

n=size(x(:,1));
N=n(1,1);
theta=ones(N,1);
Q=ones(N,1);

for l = 1:N,
    Q(l,1)=alpha*(x(l,1)-xf)+beta*x(l,2);
    theta(l,1)=fzero('model_2',20);
end

Q=(0.0000000781871*theta.^5-0.0000099606931*theta.^4-...
    0.00002536546426*theta.^3+0.00583612263799*theta.^2-...
    0.00658991522416*theta+0.02412600207374);

%% This section is used to find the settling time.
more on
for l = 1:N,
    if abs(x(l,1)-xf) < (0.02*xf)
        t(l,1)
        x(l,1)
    else
        xxxxxx = 9999999999
    end
end
more off

%% Finish finding the settling time
figure(1)
subplot(3,1,1), plot(L,x(:,1))
%axis([0 1000 0.65 0.9])
grid
title('Conventional PI controller: Table 13')
xlabel('time (s)')
ylabel('heater level (ft)')

figure(2)
subplot(3,1,1), plot(L,theta)
%axis([0 1000 8 13])
grid
title('Conventional PI controller: Table 13')
xlabel('time (s)')
ylabel('valve opening (degree)')

figure(3)
subplot(3,1,1), plot(L,Q)
%axis([0 1000 0.43 0.77])
grid
title('Conventional PI controller: Table 13')
xlabel('time (s)')
ylabel('flow rate (ft^3/s)')

*****
%% This func_16.m file serves as an input to the PI_4.m file.
function xdot=func_16(L,x)

global alpha beta xf Qc

%% the PI-controller algorithm returns theta
Qc=alpha*(x(1)-xf)+beta*x(2);
theta=fzero('model_2',20);

%% the dynamic equation
xtract=11.5*0.4444;
area=2*sqrt(2*43.375/2/12*x(1)-x(1).^2)*25;

xdot(1)=1/area*(xtract-...
    (0.0000000781871*theta.^5-0.0000099606931*theta.^4-...
    0.00002536546426*theta.^3+0.00583612263799*theta.^2-...
    0.00658991522416*theta+0.02412600207374));

xdot(2)=x(1)-xf;

*****
%% This model_2.m file contains the function to be evaluated
%% for zero at a particular flow rate and pressure and
%% returns the value of theta. [linear model, Man, 300psi]
function [foo]=model_2(theta)

global Qc

foo=(0.09062981695704*theta-0.66225439803445)-Qc;

*****

```

```

%% This fbl_p_3.m file is used to integrate the func_17.m file.
%% This file also uses the model_0.m file as an input.

global alpha xf

alpha=0.1;

t0=0;
tf=1000;
x0=0.9;
xf=0.7;

[t,x]=ode23('func_17',t0,tf,x0);

n=size(x);
N=n(1,1);
theta=ones(N,1);
Q=ones(N,1);

for l = 1:N;

xtract=0;
area=2*sqrt(2*43.375/2/12*x(l,1)-x(l,1).^2)*25;

Q(l,1)=xtract+alpha*area*(x(l,1)-xf);

Qc=Q(l,1);
theta(l,1)=fzero('model_0',30);

end

Q=(0.0000000781871*theta.^5-0.0000099606931*theta.^4+...
0.0002536546426*theta.^3+0.00583612263799*theta.^2+...
0.00658991522416*theta+0.02412600207374);

%% This section is used to find the settling time.

more on
for l = 1:N;

ifabs(x(l,1)-xf) < (0.02*xf)
t(l,1)
x(l,1)
else
xxxxxx = 9999999999
end

end
more off

%% Finish finding the settling time

figure(1)
subplot(3,1,2), plot(t,x)
%axis([0 100 0.62 0.91])
grid
title('Feedback linearizing P-controller: Table 14')
xlabel('time (s)')
ylabel('heater level (ft)')

figure(2)
subplot(3,1,2), plot(t,theta)
%axis([0 100 7 30])
grid
title('Feedback linearizing P-controller: Table 14')
xlabel('time (s)')
ylabel('valve opening (degree)')

figure(3)
subplot(3,1,2), plot(t,Q)
%axis([0 100 0.2 3.5])
grid
title('Feedback linearizing P-controller: Table 14')
xlabel('time (s)')
ylabel('flow rate (ft^3/s)')

*****
%% This func_17.m file serves as an input to the fbl_p_3.m file.

function xdot=func_17(t,x)

global alpha xf Qc

%% the P-controller algorithm returns theta
xtract=0;
area=2*sqrt(2*43.375/2/12*x(1)-x(1).^2)*25;
Qc=xtract+alpha*area*(x(1)-xf);

theta=fzero('model_0',30);

%% the dynamic equation
xtract=11.5*0.4444;
xdot(1)=1/area*(xtract-...
(0.0000000781871*theta.^5-0.0000099606931*theta.^4+...
0.0002536546426*theta.^3+0.00583612263799*theta.^2+...
0.00658991522416*theta+0.02412600207374));

*****
%% This model_0.m file contains the function to be evaluated
%% for zero at a particular flow rate and pressure and
%% returns the value of theta. [5th-order model, manufacturer's]
%% At 300psi.

function [foo]=model_0(theta)

global Qc

foo=(0.0000000533730*theta.^5-0.0000079813121*theta.^4+...
0.0000662324144*theta.^3+0.00259478501535*theta.^2+...
0.0004531029014*theta+0.01186735930655)-Qc;

*****

```



```

%%% This fbl_pi_5.m file is used to integrate the func_18.m file. %%%
%%% This file also uses the model_0.m file as an input. %%%

global alpha beta xf Qc

alpha=0.1;
beta=0.01;

t0=0;
tf=100;
x0=[0.9 0];
xf=0.7;

[t,x]=ode23('func_18',t0,tf,x0);

n=size(x(:,1));
N=n(1,1);
theta=ones(N,1);
Q=ones(N,1);

for l = 1:N,

xtract=0.0;
area=2*sqrt(2*43.375/12*x(l,1)-x(l,1).^2)*25;

Q(l,1)=xtract+alpha*area*(x(l,1)-xf)+beta*area*x(l,2);

Qc=Q(l,1);
theta(l,1)=fzero('model_0',40);

end

Q=(0.0000000781871*theta.^5-0.0000099606931*theta.^4+...
0.00002536546426*theta.^3+0.00583612263799*theta.^2-...
0.00658991522416*theta+0.02412600207374);

%%% This section is used to find the settling time. %%%

more on
for l = 1:N,

ifabs(x(l,1)-xf) < (0.02*xf)
t(l,1)
x(l,1)
else
xxxxx = 9999999999
end

end
more off

%%% Finish finding the settling time %%%

figure(1)
subplot(3,1,3), plot(t,x(:,1))
axis([0 100 0.6 1 0.99])
grid
title('Feedback linearizing PI-controller: Table 15')
xlabel('time (s)')
ylabel('heater level (ft)')

figure(2)
subplot(3,1,3), plot(t,theta)
axis([0 100 20 45])
grid
title('Feedback linearizing PI-controller: Table 15')
xlabel('time (s)')
ylabel('valve opening (degree)')

figure(3)
subplot(3,1,3), plot(t,Q)
axis([0 100 2 8])
grid
title('Feedback linearizing PI-controller: Table 15')
xlabel('time (s)')
ylabel('flow rate (ft^3/s)')

*****
%%% This func_18.m file serves as an input to the fbl_pi_5.m file. %%%

function xdot=func_18(t,x)

global alpha beta xf Qc

%%% the PI-controller algorithm returns theta %%%
xtract=0.0;
area=2*sqrt(2*43.375/12*x(1)-x(1).^2)*25;
Qc=xtract+alpha*area*(x(1)-xf)+beta*area*x(2);

theta=fzero('model_0',20);

%%% the dynamic equation %%%
xtract=11.5*0.4444;

xdot(1)=1/area*(xtract-...
(0.0000000781871*theta.^5-0.0000099606931*theta.^4+...
0.00002536546426*theta.^3+0.00583612263799*theta.^2-...
0.00658991522416*theta+0.02412600207374));

xdot(2)=x(1)-xf;

*****
%%% This model_0.m file contains the function to be evaluated %%%
%%% for zero at a particular flow rate and pressure and %%%
%%% returns the value of theta. [5th-order model, manufacturer's] %%%
%%% At 300psi. %%%

function [foo]=model_0(theta)

global Qc

foo=(0.0000000533730*theta.^5-0.0000079813121*theta.^4+...
0.00000662324144*theta.^3+0.00259478501535*theta.^2+...
0.0004531029014*theta+0.01186735930655)-Qc;

*****

```

## A.1.6 Simulation Case#6

```

%% This bi_PI_0.m file is used to integrate the bi_func_12.m file.
%% This file also uses the bi_model_3.m file as an input.

global alpha beta xf Qc

alpha=100/35;
beta=(100/35)/(1.7*60);

t0=0;
tf=1000;
x0=[0.9 0];
xf=0.7;

[t,x]=ode23('bi_func_12',t0,tf,x0);

n=size(x(:,1));
N=n(1,1);
theta=ones(N,1);
Q=ones(N,1);

for l = 1:N,

Q(l,1)=alpha*(x(l,1)-xf)+beta*x(l,2);

Qc=Q(l,1);
theta(l,1)=fzero('bi_model_3',20);

end

Q=(0.0000000119801*theta.^5+0.00000045506570*...
theta.^4-0.00012713682332*theta.^3+0.00803317114257*theta.^...
2-0.02072164365028*theta+0.02523712251737);

%% This section is used to find the settling time.

more on
for l = 1:N,

if abs(x(l,1)-xf) < (0.02*xf)
t(l,1)
x(l,1)
else
xxxxxx = 9999999999
end

end
more off

%% Finish finding the settling time

figure(1)
subplot(3,1,1), plot(t,x(:,1))
%axis([0 1000 0.65 0.90])
grid
title('Conventional PI-controller[two phase flow]: Table 16')
xlabel('time (s)')
ylabel('heater level (ft)')

figure(2)
subplot(3,1,1), plot(t,theta)
%axis([0 1000 9 13])
grid
title('Conventional PI-controller[two phase flow]: Table 16')
xlabel('time (s)')
ylabel('valve opening (degree)')

figure(3)
subplot(3,1,1), plot(t,Q)
%axis([0 1000 0.43 0.70])
grid
title('Conventional PI-controller[two phase flow]: Table 16')
xlabel('time (s)')
ylabel('flow rate (ft^3/s)')

*****
%% This bi_func_12.m file serves as an input to the bi_PI_0.m file.
function xdot=bi_func_12(t,x)

global alpha beta xf Qc

%% the PI-controller algorithm returns theta
Qc=alpha*(x(1)-xf)+beta*x(2);

theta=fzero('bi_model_3',20);

%% the dynamic equation
xtract=11.5*0.4444;
area=2*sqrt(2*43.375/2/12*x(1)-x(1).^2)*25;

xdot(1)=1/area*(xtract-(0.0000000119801*theta.^5+0.00000045506570*...
theta.^4-0.00012713682332*theta.^3+0.00803317114257*theta.^...
2-0.02072164365028*theta+0.02523712251737));

xdot(2)=x(1)-xf;

*****
%% This bi_model_3.m file contains the function to be evaluated
%% for zero at a particular flow rate and pressure and
%% returns the value of theta. [linear model, Man, 300psi]
function [foo]=bi_model_3(theta)

global Qc

foo=(0.07669891094450*theta-0.42215029667971)-Qc;

*****

```

```

%% This bi_fbl_p_2.m file is used to integrate the bi_func_6.m file.
%% This file also uses the bi_model_0.m file as an input.[dummy]

global alpha xf
alpha=0.1;

t0=0;
tf=1000;
x0=0;
xf=0.7;

[L,x]=ode23('bi_func_6',t0,tf,x0);

n=size(x);
N=n(1,1);
theta=ones(N,1);
Q=ones(N,1);

for l = 1:N,
xtract=0.0;
area=2*sqrt(2*43.375/2/12*x(l,1)-x(l,1).^2)*25;
Q(l,1)=xtract+alpha*area*(x(l,1)-xf);

Qc=Q(l,1);
theta(l,1)=fzero('bi_model_0',60);
end

Q=(0.0000000119801*theta.^5+0.00000045506570*...
theta.^4-0.00012713682332*theta.^3+0.00803317114257*theta.^...
2-0.02072164365028*theta+0.02523712251737);

%% This section is used to find the settling time.
more on
for l = 1:N,
ifabs(x(l,1)-xf) < (0.02*xf)
t(l,1)
x(l,1)
else
xxxxxx = 9999999999
end
end
more off

%% Finish finding the settling time
figure(1)
subplot(3,1,2), plot(L,x)
%axis([0 100 0.62 0.91])
grid
title('Feedback linearizing P-controller[two phase flow]; Table 17')
xlabel('time (s)')
ylabel('heater level (ft)')

figure(2)
subplot(3,1,2), plot(L,theta)
%axis([0 100 7 25])
grid
title('Feedback linearizing P-controller[two phase flow]; Table 17')
xlabel('time (s)')
ylabel('valve opening (degree)')

figure(3)
subplot(3,1,2), plot(L,Q)
%axis([0 100 0.2 3.5])
grid
title('Feedback linearizing P-controller[two phase flow]; Table 17')
xlabel('time (s)')
ylabel('flow rate (ft^3/s)')

*****
%% This bi_func_6.m file serves as an input to the bi_fbl_p_2.m file.
function xdot=bi_func_6(L,x)

global alpha xf Qc

%% the P-controller algorithm returns theta
xtract=0.0;
area=2*sqrt(2*43.375/2/12*x(1)-x(1).^2)*25;
Qc=xtract+alpha*area*(x(1)-xf);

theta=fzero('bi_model_0',60);

%% the dynamic equation
xtract=11.5*0.4444;
xdot(1)=1/area*(xtract-(0.0000000119801*theta.^5+0.00000045506570*...
theta.^4-0.00012713682332*theta.^3+0.00803317114257*theta.^...
2-0.02072164365028*theta+0.02523712251737));

*****
%% This bi_model_0.m file contains the function to be evaluated
%% for zero at a particular flow rate and pressure and
%% returns the value of theta. [5th-order model, manufacturer's]
function [foo]=bi_model_0(theta)

global Qc

foo=(0.0000000176289*theta.^5-0.0000000574736*theta.^4-...
0.00005006103011*theta.^3+0.00387922677815*theta.^2-...
0.00835796763611*theta+0.01328852435868)-Qc;

*****

```

```

%% This bi_fbl_pi_4.m file is used to integrate the bi_func_11.m file.
%% This file also uses the bi_model_0.m file as an input.(dummy)

global alpha beta xf Qc

alpha=0.1;
beta=0.01;

t0=0;
tf=100;
x0=[0.9 0];
xf=0.7;

[t,x]=ode23('bi_func_11',t0,tf,x0);

n=size(x(:,1));
N=n(1,1);
theta=ones(N,1);
Q=ones(N,1);

for l = 1:N,
    xtract=0.0;
    area=2*sqrt(2*43.375/2/2*x(l,1)-x(l,1).^2)*25;
    Q(l,1)=xtract+alpha*area*(x(l,1)-xf)+beta*area*x(l,2);
end

Qc=Q(1,1);
theta(1,1)=fzero('bi_model_0',20);

end

Q=(0.0000000119801*theta.^5+0.00000045506570*...
    theta.^4-0.00012713682332*theta.^3+0.00803317114257*theta.^...
    2-0.02072164365028*theta+0.02523712251737);

%% This section is used to find the settling time.
more on
for l = 1:N,
    if abs(x(l,1)-xf) < (0.02*xf)
        t(l,1)
        x(l,1)
    else
        xxxxxx = 9999999999
    end
end
more off

%% Finish finding the settling time

figure(1)
subplot(3,1,3), plot(Lx(:,1))
%axis([0 100 0.61 0.91])
grid
title('Feedback linearizing controller[two phase flow]: Table 18')
xlabel('time (s)')
ylabel('heater level (ft)')

figure(2)
subplot(3,1,3), plot(L,theta)
%axis([0 100 5 28])
grid
title('Feedback linearizing controller[two phase flow]: Table 18')
xlabel('time (s)')
ylabel('valve opening (degree)')

figure(3)
subplot(3,1,3), plot(L,Q)
%axis([0 100 0.1 3.5])
grid
title('Feedback linearizing controller[two phase flow]: Table 18')
xlabel('time (s)')
ylabel('flow rate (ft^3/s)')

*****
%% This bi_model_0.m file contains the function to be evaluated
%% for zero at a particular flow rate and pressure and
%% returns the value of theta. [5th-order model, manufacturer's]
function [foo]=bi_model_0(theta)

global Qc

foo=(0.0000000176289*theta.^5-0.0000000574736*theta.^4-...
    0.0005006103011*theta.^3+0.00387922677815*theta.^2-...
    0.00835796763611*theta+0.01328852435868)-Qc;

*****
%% This bi_model_0.m file contains the function to be evaluated
%% for zero at a particular flow rate and pressure and
%% returns the value of theta. [5th-order model, manufacturer's]
function [foo]=bi_model_0(theta)

global Qc

foo=(0.0000000176289*theta.^5-0.0000000574736*theta.^4-...
    0.0005006103011*theta.^3+0.00387922677815*theta.^2-...
    0.00835796763611*theta+0.01328852435868)-Qc;

*****

```

## A.2 Adaptive Controller Based on Nonlinear Parametrization

```

%%% This plant_sim.m file is used to simulate and generate plots of the post processing results.
clear
clear global
global r ts ref Qo delP Tau Gamma Kes epsilon ...
R L rho delP_min delP_max Qo_min Qo_max

%%% Define the reference model
ts=10; % time constant
r=0.8; % desired water level
ref=r/ts; % equivalent input

%%% Define extraction flow and pressure
%%% normally unknown
Qo=20; % extraction mass flow rate
I_Qo=30; % initial extraction flow rate
delP=100; % pressure drop across the valve
I_delP=40; % initial pressure drop

%%% Define control gains
Tau=1e6; % associate with alpha_hat
Gamma=1e6; % associate with delP_hat
Kes=100; % initially 100
epsilon=0.02; % dead band

%%% Initial control input
init_u=2.5;
u=init_u;

%%% Initial conditions for all states
%%% Initially, x(4) = 37.5 was used.
x=[0.9 0.9 I_Qo I_delP 0];

%%% Simulation
[t,y]=ode45('plant',0,100,x);

%%% Reconstruct the control input vector
s=y;
area=2*(2*R*x(:,1)-x(:,1).^2).^0.5*L*0.368;
psi=1/(rho*area);
err=x(:,1)-x(:,2);
e_s=(x(:,1)-x(:,2))+1/ts*x(:,5);
for N=1:length(e_s)
    if abs(e_s(N,1)/epsilon) > 1,
        saturate(N,1)=sign(e_s(N,1)/epsilon);
    else
        saturate(N,1)=e_s(N,1)/epsilon;
    end
end
epsilon_s = e_s - epsilon*saturate;
f_max=(delP_min)^0.5/(rho*area);
f_min=(delP_max)^0.5/(rho*area);
f_hat=x(:,4).^0.5/(rho*area);
df=-x(:,4).^(-0.5)/(2*rho*area);
%%% Initial control input
Eu(1,1)=init_u;
for M=1:length(epsilon_s)
    if epsilon_s(M,1) >= 0,
        a(M,1) = (f_max(M,1)-f_hat(M,1))*Eu(M,1)-...
            (f_max(M,1)-f_min(M,1))*Eu(M,1)/...
            (delP_max-delP_min);
        w(M,1) = saturate(M,1)*Eu(M,1)*(f_max(M,1)-f_min(M,1))/...
            (delP_max-delP_min);
    else
        a(M,1) = 0;
        w(M,1) = -saturate(M,1)*df(M,1)*Eu(M,1);
    end
    Ua(M,1)=saturate(M,1)*a(M,1);
    Eu(M+1,1)=1/(-x(M,4)^0.5/(rho*area(M,1))) *...
        (-Kes*epsilon_s(M,1)-1/ts*x(M,1)+...
        ref*(1+0.3*sin(1*(M,1))))...
        Ua(M,1)-psi(M,1)*x(M,3);
    if Eu(M+1,1) > 3.623,
        Eu(M+1,1) = 3.623;
    elseif Eu(M+1,1) < 0,
        Eu(M+1,1) = 0;
    end
end
%%% Lyapunov Function
V=1/2*(epsilon_s^2+...
1/Tau*(x(:,3)-I_Qo)^2+...
1/Gamma*(x(:,4)-I_delP)^2);
vdot=Kes*epsilon_s^2+...
epsilon_s*((-80*0.5/(rho*area))-(f_hat)).*...
Eu(2:length(Eu),1).*saturate+...
w.*(x(:,4)-I_delP)-Ua);
figure(20)
clf
subplot(4,1,4), hold
plot(t,Eu(2:length(Eu),1),'g-','t,3.63*ones(size(t))','r-')
title('The equivalent control input')
xlabel('time (s)')
ylabel('control input')
grid

subplot(4,1,1), hold
plot(t,y(:,1),'r-');
plot(t,y(:,2),'g-');
title('N.P.C: delP=100; InitP=40; Tau_s=10; epsilon=0.02; Kes=100')
xlabel('time (s)')
ylabel('water level (ft)')
grid

subplot(4,1,2), hold
plot(t,y(:,3:4));
title('The parameter estimates')
xlabel('time (s)')
ylabel('parameter estimate')
grid

subplot(4,1,3), hold
plot(t,y(:,1)-r);

```

```

title('The plant output error')
xlabel('time (s)')
ylabel('level deviation (ft)')
grid

figure(30)
plot(L,V,'g-')
title('The lyapunov function')
xlabel('time (s)')
ylabel('V')
grid

*****
%%%This plant.m file contains the differential equations%%%
%%%used to described the plant as well as the controller.%%%
%%%The adaptation laws are also included in this file.%%%

funcionxdot=plant(t,x)

global r is ref Qo delP Tau Gamma Kes epsilon ...
R L rho delP_min delP_max Qo_min Qo_max

%%%Define plant constants%%%
R=1.85;%%%radius of the heater
L=25;%%%length of the heater
rho=50;%%%density of the liquid

%%%Define the bounds of the parameters%%%
delP_min=0;%%%pressure drop across the valve
delP_max=200;
Qo_min=0;%%%extraction mass flow rate
Qo_max=40;
%u_max=;%%%equivalent control input
%u_min=;
%%%Define area which is a function of the water level%%%
area=2*(2*R*x(1)-x(1)^2)*0.5*L*0.368;

%%%Define the function psi%%%
psi=1/(rho*area);

%%%Prediction error%%%
err=x(1)-x(2);
e_s=(x(1)-x(2))+1/5*x(size(x,1),5);

ifabs(e_s/epsilon) > 1,
    saturate=-sign(e_s/epsilon);
else
    saturate= e_s/epsilon;
end

epsilon_s = e_s - epsilon*saturate;

%%%Tuning function%%%
f_max=-(delP_min)^0.5/(rho*area);
f_min=-(delP_max)^0.5/(rho*area);
f_hat=x(4)^0.5/(rho*area);
df=-x(4)^(-0.5)/(2*rho*area);

ifepsilon_s >= 0,
    a = (f_max-f_hat)*u-(f_max-f_min)*u/(delP_max-delP_min)*...
        (x(4)-delP_min);
else
    w = -saturate*u*(f_max-f_min)/(delP_max-delP_min);
end
a = 0;

w = saturate*df*u;
end

Ua=saturate*a;

%%%The control law%%%
%a = -Kes*epsilon_s
%b = -1/5*x(1)-Ua-psi*x(3)
%e = ref
%d = -1/5*x(1)

u=1/(-(x(4))^0.5/(rho*area))*...
(-Kes*epsilon_s-1/5*x(1)+...
ref*(1+0.3*sin(1*t))-...
Ua-psi*x(3));

%%%Test if the control input saturates and set it within bounds%%%
ifu > 3.623,
    u = 3.623;
elseif u < 0,
    u = 0;
end

%subplot(4,1,4)
%plot(t,u,'g')
%axis([0 500 0 0.10])

%V_dot= -Kes*epsilon_s^2+epsilon_s*((-300*0.5/(rho*area)+...
%(-x(4)^0.5/(rho*area))*u+w*(x(4)-300)-Ua)

%%%x=[ h, hm, alpha_hat, delP_hat, err ]%%%
xdot(1)=1/(rho*area)*(Qo-delP*0.5*u);
xdot(2)= ref*(1+0.3*sin(1*t))-...
1/5*x(2);
xdot(3)=epsilon_s*Tau*psi;
xdot(4)=epsilon_s*Gamma*w;
xdot(5)=err;
*****

```

# Appendix B : MATLAB Simulation Programs

## Position Control in Magnetic Bearing System

### B.1 Adaptive Controller Based on Nonlinear Parametrization

```
%% This file is used to simulate the bearing control system %%%  
clear  
clear global  
  
% Define time vector  
nplot = 5;  
tfinal = 0.05;  
dT = 1/(15*1000);  
t = 10:dT:tfinal;  
nosamp = tfinal/(dT*nplot)  
  
% Plant you want to simulate (normally unknown ...)  
H = 3*(10^(-4));  
n = 133;  
A = 7*(10^(-4));  
i0 = 0.5;  
mu0 = 1.26*(10^(-6));  
m = 2.2;  
g = 9.8;  
h0 = 4.0*(10^(-4));  
Bsat = n*mu0*i0/h0;  
  
% Normalized variables  
wn = ((Bsat^2)*A/(m*h0*mu0))^0.5;  
Hn = H/h0;  
gn = g/(h0*(wn^2));  
K = 1.7*(n*mu0*i0/(2*H*Bsat))^2;  
  
epsilon = 0.000000001; % deadzone  
  
Kmax = 2*(n*mu0*i0/(2*h0*Bsat))^2;  
h_min = 0.7;  
h_max = 1.5;  
  
% Control gain  
c1 = 360000;  
c2 = 1200;  
kes = 5;  
  
u = 0;  
ref = 0;  
  
% Initial conditions for all states (normalized)  
xp = [2*(10^(-4)), 0, 0];  
xm = [1*(10^(-4)), 0];  
Hh = 0.75*[1, 1, 1]; % originally all ones  
Kh = 1.0*(n*mu0*i0/(2*h0*Bsat))^2*ones(1,3); % originally 1*  
e0 = 0;  
  
x = [xp, xm, Kh, Hh, e0];  
  
% Prediction error  
e_s = (xp(2)-xm(2)) + c2*(xp(1)-xm(1)) + c1*x(size(x),1,1);  
saturate = sat(e_s/epsilon);  
epsilon_s = e_s - epsilon*saturate;  
  
r_hat = [4*x(8)/((x(8)-x(1))^2)*x(8)+x(1))^2;  
2*x(9)^2+x(1)^2)/((x(9)-x(1))^2)*x(9)+x(1))^2);  
x(10)/((x(10)-x(1))^2)*x(10)+x(1))^2);  
r_hat = [x(5)*x(1)*r_hat(1);  
x(6)*r_hat(2);  
x(7)*x(1)*r_hat(3)];  
  
r_min = [4*h_max/((h_max-x(1))^2)*(h_max+x(1))^2);  
2*(h_max^2+x(1)^2)/((h_max-x(1))^2)*(h_max+x(1))^2);  
h_max/((h_max-x(1))^2)*(h_max+x(1))^2);  
r_max = [4*h_min/((h_min-x(1))^2)*(h_min+x(1))^2);  
2*(h_min^2+x(1)^2)/((h_min-x(1))^2)*(h_min+x(1))^2);  
h_min/((h_min-x(1))^2)*(h_min+x(1))^2);  
  
% determine initial control input  
[at,omega] = ids(x(1), [x(8), x(9), x(10)], u, r_hat, saturate, epsilon_s,...  
r_min, r_max, h_min, h_max, Kmax);  
u = bearing_ctl(f_hat,x(1),x(2)), epsilon_s, at, saturate, gn, ref, c1,...  
c2, kes);  
  
% Simulate  
global Hn gn K epsilon c1 c2 kes ref u disturbance Kmax r_min r_max...  
h_min h_max Kmax  
  
%options = odeset('rtol',0.5e-2,'atol',1e-4);  
  
pl = 1:nplot,  
for inc = (pl-1)*nosamp+1:(pl*nosamp);  
inc  
%r = (10^(-4))*(2*rand-1)*c1;  
ref = 2*(10^(-4))*(2*rand-1);  
disturbance = 0*(10^(-4))*(2*rand-1)*c1;  
[T,x] = ode15s('mag_bearing',[0, dT],x);  
  
x = x(size(x),:);  
e_s = (x(2)-x(4)) + c2*(x(1)-x(3)) + c1*x(1);  
saturate = sat(e_s/epsilon);  
epsilon_s(inc+1,1) = e_s - epsilon*saturate;  
xp(inc+1,:) = x(1:2);  
xm(inc+1,:) = x(3:4);  
Kh(inc+1,:) = x(5:7);  
Hh(inc+1,:) = x(8:10);  
end  
  
% Lyapunov function  
V = 0.5*(epsilon_s.^2 + (sum(((Hh-Hn).^2)))' + (sum(((Kh-K).^2)))');  
  
% Plot  
figure(1)  
clg
```

```

plot(t(1:length(Kh),1),xp(:,1));
ylabel('gap')
xlabel('Time (s)')
grid

figure(2)
clf
subplot(211)
plot(t(1:length(Kh),1),Hh-Hn)
ylabel('H_err')
xlabel('Time (s)')
grid
subplot(212)
plot(t(1:length(Kh),1),Kh-K)
ylabel('K_err')
xlabel('Time (s)')
grid
%legend('-', 'Estimate', ':', 'Actual')

figure(3)
clf
plot(t(1:length(Kh),1),V)
ylabel('V')
xlabel('Time (s)')
grid

pause(2)
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% This file contains the description of the magnetic bearing model %%%
function xdot = mag_bearing(t, x)

global Hn gn K epsilon c1 c2 kes ref u disturbance Knmax r_min r_max...
h_min h_max Knmax

err = x(1) - x(3);
e_s = (x(2)-x(4)) + c2*(x(1)-x(3)) + c1*x(11);
saturate = sat(e_s/epsilon);
epsilon_s = e_s - epsilon*saturate;

% f = [4*K*Hn*x(1)/((Hn-x(1))^2)*(Hn+x(1))^2;
%      2*K*(Hn^2+x(1)^2)/((Hn-x(1))^2)*(Hn+x(1))^2;
%      K*Hn*x(1)/((Hn-x(1))^2)*(Hn+x(1))^2];

r_hat = [4*x(8)/((x(8)-x(1))^2)*(x(8)+x(1))^2;
         2*(x(9)^2+x(1)^2)/((x(9)-x(1))^2)*(x(9)+x(1))^2;
         x(10)/((x(10)-x(1))^2)*(x(10)+x(1))^2];

f_hat = [x(5)*x(1)*r_hat(1);
         x(6)*r_hat(2);
         x(7)*x(1)*r_hat(3)];

r_min = [4*h_max/((h_max-x(1))^2)*(h_max+x(1))^2;
         2*(h_max^2+x(1)^2)/((h_max-x(1))^2)*(h_max+x(1))^2;
         h_max/((h_max-x(1))^2)*(h_max+x(1))^2];

r_max = [4*h_min/((h_min-x(1))^2)*(h_min+x(1))^2;
         2*(h_min^2+x(1)^2)/((h_min-x(1))^2)*(h_min+x(1))^2;
         h_min/((h_min-x(1))^2)*(h_min+x(1))^2];

[at,omega] = ids(x(1), [x(8), x(9), x(10)], u, r_hat, saturate, epsilon_s,...
r_min, r_max, h_min, h_max, Knmax);

% at = zeros(1,3); omega = zeros(1,3);
u = bearing_cti(f_hat,[x(1),x(2)], epsilon_s, saturate, gn, ref, c1,...
c2, kes);

% x = [ x, dx_dt, xm, dxm_dt, K1_hat, K2_hat, K3_hat, h1_hat, h2_hat, h3_hat, err]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Turn the adaptation on/off depending on what xdot is used %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% xdot = [ x(2);
%         K*((1+0.5*u)/(Hn-x(1)))^2 - ((1-0.5*u)/(Hn+x(1)))^2 - gn + disturbance;
%         x(4);
%         ref-c1*x(3)-c2*x(4);
%         epsilon_s*r_hat .* [x(1); u ; x(1)*u^2];
%         epsilon_s*omega;
%         err];

xdot = [ x(2);
         K*((1+0.5*u)/(Hn-x(1)))^2 - ((1-0.5*u)/(Hn+x(1)))^2 - gn + disturbance;
         x(4);
         ref-c1*x(3)-c2*x(4);
         0*[x(1); u ; x(1)*u^2];
         0*omega;
         0];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% This file performs the identification of the variable at and omega %%%
function [at,omega] = ids(x, h_hat, u, r_hat, saturate, epsilon_s, r_min,...
r_max, h_min, h_max, Knmax);

omega = zeros(1,3);

if x(1) >= 0,
if epsilon_s >= 0,
at(1) = Knmax*(x(1)*r_max(1) - r_hat(1)) - ...
x(1)*(r_max(1)-r_min(1))*h_hat(1)-(h_min)/(h_max-h_min));
omega(1) = -saturate*(x(1)*(r_max(1)-r_min(1))/(h_max-h_min));
else
at(1) = 0;
omega(1) = -saturate*x(1)*df1(x,h_hat(1));
end
else
if epsilon_s >= 0,
at(1) = 0;
omega(1) = saturate*x(1)*df1(x,h_hat(1));
else
at(1) = Knmax*(x(1)*r_hat(1) - r_max(1)) - ...
x(1)*(r_min(1)-r_max(1))*h_hat(1)-(h_min)/(h_max-h_min);
omega(1) = -saturate*(x(1)*(r_min(1)-r_max(1))/(h_max-h_min));
end
end

if u >= 0,
if epsilon_s >= 0,
at(2) = Knmax*(u*(r_max(2) - r_hat(2)) - ...
u*(r_max(2)-r_min(2))*h_hat(2)-(h_min)/(h_max-h_min));
omega(2) = -saturate*u*(r_max(2)-r_min(2))/(h_max-h_min);
else
at(2) = 0;

```



```

    omega(2) = -saturate*u*df2(x,h_hat(2));
end
else
    if epsilon_s >= 0,
        at(2) = 0;
        omega(2) = saturate*u*df2(x,h_hat(2));
    else
        at(2) = Knmax*(u*(r_hat(2) - r_max(2)) - ...
            u*(r_min(2)-r_max(2))*(h_hat(2)-h_min)/(h_max-h_min));
        omega(2) = -saturate*u*(r_min(2)-r_max(2))/(h_max-h_min);
    end
end

if x(1) >= 0,
    if epsilon_s >= 0,
        at(3) = Knmax*(x(1)*(u^2)*(r_max(3) - r_hat(3)) - ...
            x(1)*(u^2)*(r_max(3)-r_min(3))*(h_hat(3)-h_min)/...
            (h_max-h_min));
        omega(3) = -saturate*x(1)*(u^2)*(r_max(3)-r_min(3))/...
            (h_max-h_min);
    else
        at(3) = 0;
        omega(3) = -saturate*x(1)*(u^2)*df3(x,h_hat(3));
    end
end
else
    if epsilon_s >= 0,
        at(3) = 0;
        omega(3) = saturate*x(1)*(u^2)*df3(x,h_hat(3));
    else
        at(3) = Knmax*(x(1)*(u^2)*(r_hat(3) - r_max(3)) - ...
            x(1)*(u^2)*(r_min(3)-r_max(3))*(h_hat(3)-h_min)/...
            (h_max-h_min));
        omega(3) = -saturate*x(1)*(u^2)*(r_min(3)-r_max(3))/...
            (h_max-h_min);
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% This file contains the control law %%%
function u = bearing_ctl(f_hat, z, epsilon_s, at, saturate, gn, r, c1, c2, kes)

rhs = gn + r - c1*z(1) - c2*z(2) - kes*epsilon_s - sum(at)*saturate;
u = (1/(2*f_hat(3)))*(-f_hat(2) + ...
    (f_hat(2)^2-4*f_hat(3)*(f_hat(1)-rhs))^0.5);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% This file describes the first derivative of f1 %%%
function dfdh = df1(x,theta)

y = x - theta;
z = x + theta;

dfdh = 4/((y^2)*(z^2)) - 16*(theta^2)/((y^3)*(z^3));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% This file describes the first derivative of f2 %%%
function dfdh = df2(x,theta)

y = x - theta;
z = x + theta;

dfdh = 4*theta/((y^2)*(z^2)) - 8*(theta^2+x^2)*theta/((y^3)*(z^3));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% This file describes the first derivative of f3 %%%
function dfdh = df3(x,theta)

y = x - theta;
z = x + theta;

dfdh = 1/((y^2)*(z^2)) - 4*(theta^2)/((y^3)*(z^3));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

## B.2 Adaptive Controller Based on Linearized Dynamics

```

%%%This program is used to simulate the adaptive%%%
%%%control system proposed by TJ in his thesis.%%%

clear
clear global

%%%The plant you want to simulate (normally unknown)%%%

h=4*(10^(-4));
n=133;
A=7*(10^(-4));
i0=0.50;
mu0=1.26*(10^(-6));
M=2.;
g=9.81;
epsilon=0.5e-6;

%%%The control gains%%%
%%%c1 and c2 are selected according to the rotor%%%
%%%achievable dynamic. [ wn = 600 rad/s; zeta = 1]%%%

c1=360000;
c2=1200;

%%%The adaptation gains%%%
%%%are set in plant.m%%%

Gamma=1e13;

%%%Constants unique to this system%%%

bz=11.5;
bz=80;

%P = [1200*1/2*1/360000+360000*(1/2400*(1+1/360000)) 1/2*1/360000;
%      1/2*1/360000 1/2400*(1+1/360000)];

P=[ 0 1.3888888888888889e-06;
    0 2.314814814814815e-09];

%%%Reference position%%%

r=0;

%%%Initial conditions for all states%%%

xp0=[120*(10^(-6)), 0.0];
xm0=[120*(10^(-6)), 0.0];
a0=0;
a1=0;

x0=[xp0, xm0, a0, a1];

%%%Simulate%%%

global h n A i0 mu0 M g c1 c2 r epsilon bz P Gamma xout

t0=0;
tf=0.5;
tol=1.e-6;

%%%We have to define xout here for the first time%%%
%%%to enable the adaptation law for a1 to be valid%%%
%%%even while the system is within the sphere.%%%

xout=[0,0];

[t,x]=ode45('plant',t0,tf,x0,tol,1);

figure(20)
c1g
hold
title('The rotor position: actual - ; desired -- [Gamma = 1e13]')
xlabel('time (s)')
ylabel('rotor position (micron)')
plot(t,x(:,1)*1e6,'r-')
plot(t,x(:,3)*1e6,'g--')
%axis([0 0.01 -1 1])
grid
%print -deps P120B80_0

figure(21)
c1g
hold
title('The rotor position error [Gamma = 1e13]')
xlabel('time (s)')
ylabel('rotor position error (micron)')
plot(t,(x(:,3)-x(:,1))*1e6,'m-')
%axis([0 0.01 -6 1])
grid
%print -deps PE120B80_0

figure(22)
c1g
hold
title('The rotor velocity: actual - ; desired -- [Gamma = 1e13]')
xlabel('time (s)')
ylabel('rotor velocity (micron/s)')
plot(t,(x(:,2)*1e6,'r-');t,x(:,4)*1e6,'g--')
%axis([0 0.01 -0.4e4 0.4e4])
grid
%print -deps V120B80_0

figure(23)
c1g
hold
title('The system states [Gamma = 1e13]')
ylabel('velocity (micron/s)')
xlabel('position (micron)')
plot(x(:,1)*1e6,x(:,2)*1e6,'r-';x(:,1)*1e6,x(:,2)*1e6,'go')
%axis([0 0.01 -0.4e4 0.4e4])
grid
%print -deps S120B80_0

figure(24)
c1g
subplot(3,1,1), hold
title('a0 [Gamma = 1e13]')
xlabel('time (s)')
ylabel('a_0')
plot(t,x(:,5)*c-',')
%axis([0 0.009 -0.0 0.004])
grid

```

```

subplot(3,1,2), hold
title('a1')
xlabel('time (s)')
ylabel('a1')
plot(t,x(:,6),'r-')
%axis([0 0.009 -0.00007 0.00000])
grid
subplot(3,1,3), hold
title('error magnitude = sqrt(x^2+v^2)')
xlabel('time (s)')
ylabel('error magnitude')
plot((sqrt((x(:,1)-x(:,3)).^2+(x(:,2)-x(:,4)).^2)), 'm-')
%axis([0 0.009 -0.00007 0.00000])
grid
%print -deps TH120B80_0

figure(25)
clf
subplot(2,1,1), hold
title('Rotor position')
xlabel('time (s)')
ylabel('rotor position (micron)')
plot(x(:,1)*1e6,'r-')
plot(x(:,3)*1e6,'g-')
legend('Actual', 'Reference')
grid
subplot(2,1,2), hold
title('Tracking error')
xlabel('time (s)')
ylabel('error (micron)')
plot((x(:,1)*1e6-x(:,3)*1e6), 'r-')
grid

figure(26)
clf
subplot(2,1,1), hold
title('Parameter a0')
xlabel('time (s)')
ylabel('a_0')
plot(x(:,5),'c-')
grid
subplot(2,1,2), hold
title('Parameter a1')
xlabel('time (s)')
ylabel('a1')
plot(x(:,6),'r-')
grid

*****
%%% This file contains the information needed to simulate %%%
%%% the magnetic bearing system according to TJ's adaptive %%%
%%% algorithm which employs full-state feedback and adaptation %%%
%%% on-demand approach. %%%

function xdot=plant(t,x)

global h n A i0 mu0 M g c1 c2 r epsilon bz P index Gamma xout

e=[(x(1)-x(3));
(x(2)-x(4))];

%%% If the system penetrates the sphere of radius epsilon, %%%
%%% then set xout to these values only for the first time. %%%

if sqrt(sum(e.^2)) >= epsilon & index == 1
xout=[x(1);
x(2)];
index=0;
elseif sqrt(sum(e.^2)) < epsilon
index=1;
end

%%% Turn on the adaptation if the error is larger than %%%
%%% epsilon which is defined as the radius of the sphere. %%%

if sqrt(sum(e.^2)) >= epsilon
uad=(-x(5)-x(6))*x(1)-xout(1));
Gamma_1=Gamma;
Gamma_2=Gamma;
else
uad=0;
Gamma_1=0;
Gamma_2=0;
end

%%% Now, let's calculate the control input %%%

ulin=[-c1 -c2]*x(1)-0;
x(2)-0];

%u=1*(1/bz*(ulin+uad)+M/bz*g);
u=1/bz*(ulin+1/M*uad)+M/bz*g);

%%% x = [x, dxdt, xm, dxm/dt, a0_1, a1_1, a0_2, a1_2] %%%
xdot = [x(2);
(n^2*mu0*A*(i0+u)^2)/(4*M*(h-x(1))^2)-(n^2*mu0*A*(i0-u)^2)/(4*M*(h+x(1))^2)-g;
x(4);
r-c1*x(3)-c2*x(4);
Gamma_1*e.*P*[0;
1];
Gamma_2*(x(1)-xout(1))*e.*P*[0;
1]];

*****
%%% This file describes the property of saturation function %%%

function y=sat(x)

if abs(x)>1
y=sign(x);
else
y=x;
end

*****

```

## B.3 Adaptive Controller Based on Linear Parametrization

```

%% This program is used to simulate the adaptive control
%% of the magnetic bearing system based on Taylor series
%% linearization technique using unnormalized notations.

clear
clear global

%% The plant you want to simulate (normally unknown)

h=3*(10^(-4));
n=133;
A=7*(10^(-4));
i0=0.50;
mu0=1.26*(10^(-6));
M=2.2;
g=9.81;
alpha=(n^2*mu0*A)/(4*M);

epsilon=1e-9;

%% The control gains
%% c1 and c2 are selected according to the rotor
%% achievable dynamic. [ wn = 600 rad/s; zeta = 1 ]

c1=360000;
c2=1200;
k=1000;

%% The adaptation gains

Tau_1=1e13;
Tau_2=1e13;
Tau_3=1e13;

%% Initial control input and reference position

u=0;
r=0;

%% Initial conditions for all states

xp0=[1*(10^(-5)), 0, 0];
xm0=[1*(10^(-5)), 0];
e0=0;
THETA_10= zeros(1,4);
THETA_20= zeros(1,4);
THETA_30= zeros(1,4);

x0=[xp0, xm0, e0, THETA_10, THETA_20, THETA_30];

%% Simulate

global h n A i0 mu0 alpha M g c1 c2 Tau_1 Tau_2 Tau_3 u r k epsilon

t0=0;
tf=0.02;
tol=1.e-6;

[t,x]=ode45('plant',t0,tf,x0,tol,1);

figure(40)
cig
hold
title('The rotor position: actual - ; desired -- [Tau = 0.1]')
xlabel('time (s)')
ylabel('rotor position (micron)')
plot(t,x(:,1)*1e6,'r-')
plot(t,x(:,3)*1e6,'g--')
axis([0 0.01 -1 11])
grid

figure(41)
cig
hold
title('The rotor position error [Tau = 0.1]')
xlabel('time (s)')
ylabel('rotor position error (micron)')
plot(t,x(:,3)-x(:,1))*1e6,'m-')
axis([0 0.01 -6 1])
grid

figure(42)
cig
hold
title('The rotor velocity: actual - ; desired -- [Tau = 0.1]')
xlabel('time (s)')
ylabel('rotor velocity (micron/s)')
plot(t,x(:,2)*1e6,'r-'); plot(t,x(:,4)*1e6,'g--')
axis([0 0.01 -0.4e4 0.4e4])
grid

figure(43)
cig
hold
title('The integral error [Tau = 0.1]')
xlabel('time (s)')
ylabel('integral error (micron s)')
plot(t,x(:,5)*1e6,'m-')
grid

figure(44)
cig
subplot(2,2,1), hold
title('Theta_1')
xlabel('time (s)')
ylabel('theta_1')
plot(t,x(:,6), 'r-')
axis([0 0.009 -0.0 0.004])
grid
subplot(2,2,2), hold
title('Theta_2')
xlabel('time (s)')
ylabel('theta_2')
plot(t,x(:,7), 'r-')
axis([0 0.009 -0.00007 0.00000])
grid
subplot(2,2,3), hold
title('Theta_3')
xlabel('time (s)')
ylabel('theta_3')
plot(t,x(:,8), 'm-')
axis([0 0.009 -20 0.0])
grid

```

```

subplot(2,2,4), hold
title('Theta_4')
xlabel('time (s)')
ylabel('theta_4')
plot(xf,:9,'w-')
axis([0 0.009 0.0 0.8])
grid

figure(45)
clf
subplot(2,2,1), hold
title('Theta_5')
xlabel('time (s)')
ylabel('theta_5')
plot(xf,:10,'c-')
axis([0 0.009 -0.05 1.2])
grid
subplot(2,2,2), hold
title('Theta_6')
xlabel('time (s)')
ylabel('theta_6')
plot(xf,:11,'r-')
axis([0 0.009 -0.015 0.0005])
grid
subplot(2,2,3), hold
title('Theta_7')
xlabel('time (s)')
ylabel('theta_7')
plot(xf,:12,'m-')
axis([0 0.009 -4000 100])
grid
subplot(2,2,4), hold
title('Theta_8')
xlabel('time (s)')
ylabel('theta_8')
plot(xf,:13,'w-')
axis([0 0.009 -5 70])
grid

figure(46)
clf
subplot(2,2,1), hold
title('Theta_9')
xlabel('time (s)')
ylabel('theta_9')
plot(xf,:14,'c-')
axis([0 0.009 -0.00001 0.005])
grid
subplot(2,2,2), hold
title('Theta_10')
xlabel('time (s)')
ylabel('theta_10')
plot(xf,:15,'r-')
axis([0 0.009 -0.00005 0.0])
grid
subplot(2,2,3), hold
title('Theta_11')
xlabel('time (s)')
ylabel('theta_11')
plot(xf,:16,'m-')
axis([0 0.009 -20 0.0])
grid
subplot(2,2,4), hold
title('Theta_12')
xlabel('time (s)')
ylabel('theta_12')
plot(xf,:17,'w-')
axis([0 0.009 -0.0 0.5])
grid

*****
%%% This program contains the description of the adaptive control system based on Taylor series linearization.
%%% The method adopted here is in unnormalized form.
function xdot=plant(t,x)

global h n A i0 mu0 alpha M g c1 c2 Tau_1 Tau_2 Tau_3 u r k epsilon

err = x(1) - x(3);
e = (x(2)-x(4)) + c2*(x(1)-x(3)) + c1*x(5);
saturate = sat(e/epsilon);
e_s = e - epsilon*saturate;

[f1,fhat_1]=fhat1(x(1),h,alpha,i0,[x(6) x(7) x(8) x(9)]);
[f2,fhat_2]=fhat2(x(1),h,alpha,i0,[x(10) x(11) x(12) x(13)]);
[f3,fhat_3]=fhat3(x(1),h,alpha,i0,[x(14) x(15) x(16) x(17)]);

fhat=[fhat_1 fhat_2 fhat_3];

u = bearing_ctl(fhat, [x(1), x(2)], g, r, c1, c2, k, e_s);

%%% x = [x, dx/dt, xm, dxm/dt, intgr_err, THETA_1, THETA_2, THETA_3]
THETA_1= Tau_1*e_s*f1;
THETA_2= Tau_2*e_s*f2;
THETA_3= Tau_3*e_s*f3;

xdot = [x(2);
(n^2*mu0^A*(i0+u)^2)/(4*M*(h-x(1))^2)-(n^2*mu0^A*(i0-u)^2)/(4*M*(h+x(1))^2)-g;
x(4);
r-c1*x(3)-c2*x(4);
err;
THETA_1;
THETA_2;
THETA_3];

*****
%%% This program calculate the estimation of f_1 using the Taylor series linearization method around the nominal operating point.
function [f1,fhat_1]=fhat1(z,h,alpha,i0,THETA_1)

a=(h-z);
b=(h+z);

f_1o=(4*alpha*h*z*i0^2)/...
(a^2*b^2);

dfda=(4*h*z*i0^2)/...
(a^2*b^2);

dfdh=((4*alpha*z)/(a^2*b^2)-...
(4*alpha*h*z*(2*b*a^2+2*b^2*a))/(a^4*b^4))*i0^2;

dfdz=((4*z)/(a^2*b^2)-...

```

```

(4*h*z*(2*b*a^2+2*b^2*a))/(a^4*b^4)*i0^2;
dfdh2=(-(8*alpha*z)/(a^2*b^3)+(8*alpha*z)/(a^3*b^2)...
(8*alpha*z)/(a^2*b^3)-(24*alpha*h*z)/(a^2*b^4)+(16*alpha*h*z)/(a^3*b^3)...
(8*alpha*z)/(a^3*b^2)+(16*alpha*h*z)/(a^3*b^3)+(24*alpha*h*z)/(a^4*b^2))*i0^2;
f1=(dfda dfdh 1/2*dfdah 1/2*dfdh2);
fhat_1=f_1o+...
[dfda dfdh 1/2*dfdah 1/2*dfdh2]*THETA_1';
*****
%%%%This program calculate the estimation of f_2 using%%
%%the Taylor series linearization method around the %%
%%nominal operating point.%%
function[f2,fhat_2]=fhat2(z,h,alpha,i0,THETA_2)
a=(h-z);
b=(h+z);
f_2o=(2*alpha*(h^2+z^2)*i0)/...
(a^2*b^2);
dfda=(2*(h^2+z^2)*i0)/...
(a^2*b^2);
dfdh=(-(4*alpha*h)/(a^2*b^2)-...
(4*alpha*(b^2+z^2))/(a^2*b^3)-(4*alpha*(h^2+z^2))/(a^3*b^2))*i0;
dfdah=(4*h)/(a^2*b^2)-...
(4*(h^2+z^2))/(a^2*b^3)-(4*(h^2+z^2))/(a^3*b^2))*i0;
dfdh2=(-(4*alpha)/(a^2*b^2)+(8*alpha*h)/(a^2*b^3)-(8*alpha*h)/(a^3*b^2)-...
(8*alpha*h)/(a^2*b^3)-(12*alpha*(h^2+z^2))/(a^2*b^4)+(8*alpha*(h^2+z^2))/(a^3*b^3)-...
(8*alpha*h)/(a^3*b^2)+(8*alpha*(h^2+z^2))/(a^3*b^3)+(12*alpha*(h^2+z^2))/(a^4*b^2))*i0;
f2=(dfda dfdh 1/2*dfdah 1/2*dfdh2);
fhat_2=f_2o+...
[dfda dfdh 1/2*dfdah 1/2*dfdh2]*THETA_2';
*****
%%%%This program calculate the estimation of f_3 using%%
%%the Taylor series linearization method around the %%
%%nominal operating point.%%
function[f3,fhat_3]=fhat3(z,h,alpha,i0,THETA_3)
a=(h-z);
b=(h+z);
f_3o=(alpha*h*z)/...
(a^2*b^2);
dfda=(h*z)/...
(a^2*b^2);
dfdh=(-alpha*z)/(a^2*b^2)-...
(2*alpha*h*z)/(a^2*b^3)-(2*alpha*h*z)/(a^3*b^2);
dfdah=(z)/(a^2*b^3)-...
(2*h*z)/(a^2*b^3)-(2*h*z)/(a^3*b^2);
dfdh2=(-(2*alpha*z)/(a^2*b^3)-(2*alpha*z)/(a^3*b^2)-...
(2*alpha*z)/(a^2*b^3)-(6*alpha*h*z)/(a^2*b^4)+(4*alpha*h*z)/(a^3*b^3)-...
(2*alpha*z)/(a^3*b^2)+(4*alpha*h*z)/(a^3*b^3)+(6*alpha*h*z)/(a^4*b^2));
f3=(dfda dfdh 1/2*dfdah 1/2*dfdh2);
fhat_3=f_3o+...
[dfda dfdh 1/2*dfdah 1/2*dfdh2]*THETA_3';
*****
%%%% This file describes the property of saturation function %%%
function y=sat(x)
ifabs(x)>1
y=sign(x);
else
y=x;
end
*****

```

# Appendix C : FORTRAN Unit 9 Simulator Program

## Kingston Unit 9 Simulator

### C.1 Controller Based on Feedback Linearization

```
c234567890123456789012345678901234567890123456789012345678901234567890
      Subroutine gonctr(Flhs ,Llhs ,Lvls , Alpha , Beta ,
& Cfhvdo)
c *****
c      gonctr.f
c *****
c      ABSTRACT
c
c      This subroutine is designed for modeling a nonlinear
c controller used in controlling the level of the feedwater
c heater. The heater considered here is the shell and tube
c type in horizontal configuration.
c
c *****
c      NOTES
c *****
c
c      In order to implement this controller in the actual
c plant, another routine must be added to this program to
c enable the transition from manual to automatic mode. One
c possibility is to ask the controller to track a first order
c model and allow sufficient time during transition.
c
c      The inner diameter of the heater is also changed to
c to 44.375 as opposed to 43.375 in.
c
c      The effective area is the liquid surface area in the
c horizontal plane of the heater at current heater level.
c *****
c
      Implicit none
c
c ===== Variable Definitions =====
c =====
c
c ----- Inputs -----
c -----
      Real*4 Alpha ! Nonlinear proportional gain
      Real*4 Beta ! Nonlinear integral gain
      Real*4 Flhs ! Extraction (s+s) flow rate (lbm/s)
c Real*4 Plhs ! Shell total pressure (psia);(optional)
      Real*4 Llhs ! Feedwater heater level (in)
      Real*4 Lvls ! Feedwater heater level setpoint (in)
c -----
c ----- Outputs -----
c -----
      Real*4 Cfhvdo ! Drain valve lift (scale 0 to 1)
c -----
c ----- Internal Variables -----
c -----
      Real*4 a_ftr ! X-sectional effective area factor
      Real*4 a_htr ! X-sectional effective area (ft^2)
      Real*4 htr_jgth ! Heater length (ft)
      Real*4 lvl_er ! Heater level error (in)
      Real*4 int_lvl_er ! Integration of level error
      Real*4 lvl_old ! Level measured at the previous step
      Real*4 vol_drain ! Volumetric drain flow (ft^3/s)
      Real*4 inc ! Time increment
      Real*4 rho ! The density of the water (lbm/ft^3)
c -----
c ----- Define heater constants -----
c -----
      data int_lvl_er /0.0/ ! Initial integral error
      data htr_jgth /25./
      data inc /0.25/
      data rho /50/
c -----
c ----- Begin The Program -----
c -----
c ----- Check to see if the value of the heater level -----
c ----- less than zero. Set it equal to zero if it is -----
      If (Llhs .LT. 0.0) Llhs = 0.0
c -----
c ----- First, the cross-sectional area is calculated -----
c ----- The area used is in the horizontal plane -----
      If (Llhs .GE. 0.0 .AND. Llhs .LT. 4.1) a_ftr = 1.0
c ----- Nothing is in the first 4.1 in region -----
      If (Llhs .GE. 0.0 .AND. Llhs .LT. 21.4) a_ftr = 0.368
c ----- 20%+43.2% is taken by the drain cooler and tubes -----
      If (Llhs .GE. 0.0 .AND. Llhs .LT. 43.375) a_ftr = 0.46
c ----- 54% is taken by the tubes -----
c ----- Now, compute the actual area of the heater -----
      a_htr = 2*SQRT(2*(44.375/12)*(Llhs/12)-(Llhs/12)**2)*
& htr_jgth*a_ftr
```

```

c ----- Then, the desired control flow is obtained -----
c
c if (int_lv_ler.EQ. 0) vl_old = Lfhs
c ----- Set vl_old = Lfhs for the first integration -----
c
c   vl_er   = (Lfhs-Lvls)/12
c   int_lv_ler = int_lv_ler+0.5*inc*((vl_old-lvls)/12+vl_er)
c   vl_old   = Lfhs
c ----- Save the value of heater level for next integration -----
c
c   vol_drain = Ffhs/rho+
c   & Alpha*a_ht*vl_er+
c   & Beta*a_ht*int_lv_ler
c
c ----- Call a function that contains the valve flow -----
c ----- data in order to determine the corresponding -----
c ----- drain valve opening (%) -----
c -----
c ----- In the preliminary test, the data points are -----
c ----- used to provide the approximation of the valve -----
c ----- opening. The routine is included below -----
c ----- (Using the experimental data at del_P = 300psi) -----
c -----
c ----- Check if the vol_drain is less than zero. -----
c ----- If it is, set the valve closed. -----
c
c   if (vol_drain.LT. 0.0)
c   & Cfdvo = 0.0
c
c ----- Between 0 and 10 degree valve opening -----
c   if (vol_drain.GE. 0.0 .AND. vol_drain.LT. 0.3726)
c   & Cfdvo = (10/(0.3726-0))*
c   &   vol_drain/90
c
c ----- Between 10 and 20 degree valve opening -----
c   if (vol_drain.GE. 0.3726 .AND. vol_drain.LT. 1.0998)
c   & Cfdvo = (10/(1.0998-0.3726))*
c   &   vol_drain+10*(1.0998-0.3726)*
c   &   0.3726/90
c
c ----- Between 20 and 30 degree valve opening -----
c   if (vol_drain.GE. 1.0998 .AND. vol_drain.LT. 2.3381)
c   & Cfdvo = (10/(2.3381-1.0998))*
c   &   vol_drain+20*(1.0998-0.3726)*
c   &   1.0998/90
c
c ----- Between 30 and 40 degree valve opening -----
c   if (vol_drain.GE. 2.3381 .AND. vol_drain.LT. 3.5430)
c   & Cfdvo = (10/(3.5430-2.3381))*
c   &   vol_drain+30*(1.0998-0.3726)*
c   &   2.3381/90
c
c ----- Between 40 and 50 degree valve opening -----
c   if (vol_drain.GE. 3.5430 .AND. vol_drain.LT. 4.5754)
c   & Cfdvo = (10/(4.5754-3.5430))*
c   &   vol_drain+40*(1.0998-0.3726)*
c   &   3.5430/90
c
c ----- Between 50 and 60 degree valve opening -----
c   if (vol_drain.GE. 4.5754 .AND. vol_drain.LT. 5.0678)
c   & Cfdvo = (10/(5.0678-4.5754))*
c   &   vol_drain+50*(1.0998-0.3726)*
c   &   4.5754/90
c
c ----- Between 60 and 70 degree valve opening -----
c   if (vol_drain.GE. 5.0678 .AND. vol_drain.LT. 5.3260)
c   & Cfdvo = (10/(5.3260-5.0678))*
c   &   vol_drain+60*(1.0998-0.3726)*
c   &   5.0678/90
c
c ----- Between 70 and 80 degree valve opening -----
c   if (vol_drain.GE. 5.3260 .AND. vol_drain.LT. 5.3750)
c   & Cfdvo = (10/(5.3750-5.3260))*
c   &   vol_drain+70*(1.0998-0.3726)*
c   &   5.3260/90
c
c ----- Between 80 and 90 degree valve opening -----
c   if (vol_drain.GE. 5.3750 .AND. vol_drain.LT. 5.5400)
c   & Cfdvo = (10/(5.5400-5.3750))*
c   &   vol_drain+80*(1.0998-0.3726)*
c   &   5.3750/90
c
c ----- If vol_drain is greater than 5.54 ft^3/s -----
c ----- then fully open the valve. -----
c
c   if (vol_drain.GT. 5.5400)
c   & Cfdvo = 1.0
c
c   return
c   end
c -----

```



## References

- [1] A. M. Annaswamy, A.P. Loh, and F.P. Skantze. *Adaptive control of dynamic systems with nonlinear parametrization*. In *Proceedings of the Ninth Yale Workshop on Adaptive and Learning Systems*, Yale University, New Haven, CT, 1996.
- [2] A. M. Annaswamy, C. Thanomsat, N. R. Mehta, and A-P. Loh. *An Adaptation Algorithm for Estimation of Nonlinear Parametrization in Dynamic Systems and its Applications*. [?]
- [3] D. P. Bertsekas. *Nonlinear Programming*. MIT Press, Cambridge, MA, 1989.
- [4] D. Chisholm. *Two-phase Flow in Pipelines and Heat Exchangers*. Longman, Inc., New York, NY, 1983.
- [5] T. C. Davis. *Feedwater Heater Level Control Analysis*. S.B and S.M. Thesis, Massachusetts Institute of Technology, MA, 1985.
- [6] G. C. Goodwin and K. S. Sin. *Adaptive Filtering Prediction and Control*. Prentice-Hall, Inc., 1984.
- [7] I. Granet. *Thermodynamics and Heat Power*. Prentice Hall, Inc., Englewood Cliffs, NJ, 1996.
- [8] E. J. Hoffman. *Power Cycle and Energy Efficiency*. Academic Press Limited, London, UK. 1996.
- [9] A. H. Jazwinski. *Stochastic Processes and Filtering Theory*. Academic Press Limited, London, UK. 1970.
- [10] A-P. Loh, A. M. Annaswamy, and F. P. Skantze. *Adaptation in the Presence of a General Nonlinear Parametrization: An Error Model Approach*. [?]
- [11] K. S. Narendra and A.M. Annaswamy. *Stable Adaptive Systems*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1989.
- [12] K. C. Rolle. *Thermodynamics and Heat Power*. Macmillan Publishing Company, New York, NY, 1994.
- [13] J. P. Tullis. *Hydraulics of Pipelines: Pumps, Valves, Cavitation, Transients*. John Wiley & Sons, Inc., Toronto, CANADA, 1989.
- [14] G. Z. Watters. *Analysis and Control of Unsteady Flow in Pipelines*. Butterworth Publishers, Stoneham, MA, 1984.
- [15] T. Yeh. *Modeling, Analysis and Control of Magnetically Levitated Rotating Machines*. Ph.D. thesis, Massachusetts Institute of Technology, MA, 1995.