To the Graduate Council:

I am submitting herewith a dissertation written by Bipin Mavji Shah entitled "Intelligent Control System Design." I have examined the final copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Chemical Engineering.

Charles F. Moore, Major Professor

We have read this dissertation
and recommend its acceptance:

Accepted for the Council:

Associate Vice Chancellor
and Dean of The Graduate School

INTELLIGENT CONTROL SYSTEM DESIGN

A Dissertation

Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Bipin Mavji Shah

May 1995

This dissertation is dedicated to my parents

Mavjibhai Virji Shah

and

Maniben Mavji Shah

Acknowledgements

I would like to deeply thank my advisor Dr. Charles F. Moore for his guidance, support and patience. I would also like to thank the other committee members, Dr. John Birdwell, Dr. Eugene Stansbury and Dr. Duane Bruns for their support.

I would like to thank the Measurement and Control Engineering Center for providing support for this research and to SIEMENS (formerly TEXAS INSTRUMENTS) for providing equipment for this research.

I would also like to thank Dr. John Arnold for his help and support. Finally I would like to thank my brother Vinod, his wife Shilpa, and my family for their support.

ABSTRACT

Process control is implemented at various levels in the manufacturing process industries. At the *micro* level it deals with implementing safety interlocks, tuning controllers, positioning actuators. etc. At the textitmacro level it deals with plant wide optimization and control, supervisory control and scheduling. The *micro* level implementation of process control uses programmable logic controllers (PLC) and distributed control systems (DCS) The *macro* level implementation uses workstations and main frame computers with communication networks that talk to the controllers at the micro level. Closed analytic representation is unavailable at the macro level, and sometimes at the micro level. Artificial Intelligence plays an important role in implementing process control at the *micro* and *macro* level. This research looks at implementing Intelligent Control on an instrumented shell and tube heat exchanger in the Unit Operations laboratory. The heat exchanger is controlled by an industrial controller (PLC) and an operator interface. System Cultivation tools are developed to monitor and classify process variable trends. These shallow reasoning tools help determine if a process is at steady state, under transition or oscillating. Information from these tools is used by a expert system to provide intelligent supervisory control and intelligent system identification. A framework has been developed to provide an easy way to implement new control strategies and system identification techniques which when combined with the intelligent features of the controller provide a versatile and fail safe controller.

Contents

List of Figures

List of Tables

LIST OF SYMBOLS

$\dot{m}$	=	mass flow rate in lb/hr
C_p	=	fluid heat capacity in $Btu/lb -^\circ F$
U_o	=	overall heat transfer co-efficient $Btu/ft^2 - hr -^\circ F$
A	=	area available for heat transfer ft^2
Q_{steam}	=	energy transfered by steam Btu/hr
ΔT	=	temperature difference $^\circ F$
F_s	=	steam flow rate lb/hr
ΔH_v	=	latent heat of vaporization for steam Btu/lb
ΔT_{lm}	=	log mean temperature difference
PV	=	process Variable
SP	=	controller set-point
PV_s	=	smoothed process variable
$e(t)$	=	error at time t
$e_c(t)$	=	controller error at time t
e_+	=	positive error term
e_-	=	negative error term
m_n	=	controller output
m_x	=	controller bias
K_c	=	controller gain
T_i	=	contrller reset time
T_d	=	controller rate time
T_s	=	sampling time
τ	=	time constant
K_u	=	ultimate gain
T_u	=	ultimate period
ω_u	=	ultimate frequency

h = relay feedback height

a = relay amplitude

ϵ = relay feedback hysterisis width

$d_e(t)$ = deviation decay term at time t

$\dot{d}_e(t)$ = deviation decay rate term at time t

δ_1 = decay ratio for the deviation decay term

δ_2 = decay ratio for the deviation decay rate term

d_{e+} = positive error decay term

d_{e-} = negative error decay term

δ = decay ratio for the error decay terms

$\underline{e}$ = error vector $= [e_1(t) \cdots e_p(t)]'$

$\underline{D}_n(t)$ = vector of normalized disturbance variables $[D_{1,n}(t) \cdots D_{p,n}(t)]'$

$\underline{D}_b$ = vector of normalized disturbance variables at base case
$= [D_{1,b} \cdots D_{p,b}]'$

τ_{Di} = time constant for disturbance variable i

LIST OF ABBREVIATIONS

APT	Application Productivity Tool
ATV	Autotune Variation
CFB	Continous Function Block
DCS	Distributed Control System
DMC	Dynamic Matrix Control
IPID	Intelligent Proportional Integral Derivative
MIS	Management Information Systems
PID	Proportional Integral Derivative
PLC	Programmable Logic Controller
PRBS	Pseudo Random Binary Sequence
SFC	Sequential Function Chart
dde	Deviation decay term
ddr	Deviation ddecay rate term
de_plus	Positive error decay term
de_minus	Negative error decay term

Chapter 1

Introduction

Process Control is implemented at various levels in the manufacturing process industries. At the *micro* level it is concerned with tuning controllers, positioning actuators, reading sensors, etc. At the *macro* level that is concerned with supervisory control, scheduling, optimization and plant wide control. Closed analytic representation of the process at the micro level is not always available. Artificial Intelligence is widely used for problem solving when closed analytic solutions are not available. Artificial Intelligence techniques provide a wide spectrum of tools for problem solving ranging from expert systems, fuzzy logic, neural networks etc. Each of these techniques has its strong and weak features. However no single technique provides a complete solution. Much better performance can be obtained if hybrid systems are built. These hybrid systems provide a collection of tools, which are called upon for services when desired by a supervisory system to provide intelligent problem solving and control. The face of process control is changing rapidly. Supervisory and plant wide control often extend their boundary to include a large number of processes. Distributed Control Systems (DCS) and Programable Logic Controllers (PLC), Management Information Systems (MIS) network to provide plant wide information systems. Intelligent systems can play a crucial role in using this information and improving plant efficiency. An issue that intelligent control design faces is, how much control and intelligence is

to be provided by the local controller vs. supervisory controller.

Process control at the micro level is implemented on PLC's or a DCS system that have limited computing power. These control systems are capable of communicating to supervisory systems or other controllers via plant-wide networking or peer to peer networking systems. Whereas process control at the macro level is implemented on workstations or mainframe type computers. For a workstation class computer or a supervisory computer to monitor all or most of its controllers intelligently can be an overwhelming task.

This research focuses on using Artificial Intelligence on an instrumented heat exchanger controlled by a programmable logic controller (PLC). A real process is chosen as simulations fail to capture the important aspects of reality. The process is equipped with an industrial operator interface. Continual advances in the development of new control algorithms and system identification techniques make it desirable to have a programming environment where new control strategies can be added with minimal work. This research looks at building an interface to the PLC whereby new control algorithms can be easily implemented.

Industrial control systems have *historians* that log important process information which is often used to determine the performance of the process. Techniques like System Cultivation [1, 2, 3] have been developed to perform off-line analysis to detect subtle changes in process operation. However there are no qualitative tools available to determine (on-line) if a process is at steady-state, or is undergoing a disturbance or has one of its control loops oscillating. This research focuses on developing qualitative reasoning tools that can help characterize process variable trends qualitatively. These tools can help determine if a process is at steady state or is undergoing oscillations. Poor controller performance at one unit operation can propagate its variation to down stream processes. Intelligent controllers need to be able to evaluate their own performance and adjust their operation accordingly. Intelligent alarming plays a crucial role in reducing the amount of information presented to the operator.

The percentage of advanced control loops in operation is small compared to conventional control loops[4]. Advanced control requires a high degree of performance from the process. Improperly sized control valves, sensors, poor hydraulic design can result in poor dynamic behavior.

This work is divided into six chapters. Chapter 2 provides a brief literature review of artificial intelligence and its use in process control and some related topics used in this research. Chapter 3 describes the heat exchanger system and its characterizations. Chapter 4 describes the Intelligent SISO controller, the different control algorithms and the system identification techniques used. Chapter 5 presents the Results and Discussion for the intelligent controller. Finally, chapter 6 gives the Conclusions and Recommendations.

Chapter 2

Background

In this chapter we give a brief overview of some of the artificial intelligence techniques used in process control. We will also give a brief overview of some of the advanced control, system identification and system cultivation techniques used in this research. The key areas of AI that will be discussed are expert systems and fuzzy logic. A good reference to AI is provided by Barr, Cohen, and Fiegenbaum [5]. Artificial Intelligence spans a very wide area of literature. Significant work has been done on process monitoring and diagnostics. All of it is not going to be discussed here. A mention will be made of *Qualitative Simulation* [6, 7, 8]. Qualitative simulation deals with generating possible behaviors of dynamic systems using qualitative (sometimes incomplete) models. Neural networks have gained popularity in recent years, however little work has been done on its use in process control [9, 10, 11, 12, 13] compared to the other techniques. They are more widely used for pattern recognition, classification and fault diagnosis. Machine Learning is an important field of artificial intelligence. It deals with attempting to build machines that can learn from past experiences. Michalski [14] provides an excellent collection of articles on machine learning.

2.1 Expert Systems

Expert systems represent one of the the most widely used tool of AI. The important aspects of concern in expert systems are knowledge representation and the inference engine. Knowledge representation can be divided into two groups:*declarative* and *procedural* methods [15]. The first type represents knowledge as a group of facts and a few procedures to process the facts. The second type uses procedures to represent knowledge and how to use it. Description of commonly occurring patterns of structures and events are often called *schemas. Frames, scripts* and *rule models* are some of the schemas used in A.I. Rule models or *Production systems* are widely used in expert systems and have features suitable for expert control [16]. An expert system has three basic components: (a) database, (b) rule base and (c) inference engine. The first two components provide the knowledge representation. The database contains process information, operational information, alarming information, tuning information, etc. The production rules are represented in the rule-base. These are the collection of the *IF-THEN-ELSE* rules. The expert system uses the inference engine (employing some sort of search strategy) to use the current facts to select a production rule.

A review of the problems of the real-time knowledge based systems is given by Laffey *et al.* [17]. Significant literature exists in the use of expert systems for real-time control [18, 19, 20, 21, 22, 23, 24]. Most of the techniques used can be generally classified into two categories. Systems that use "deep knowledge" and systems that use "shallow knowledge" [25, 26].(Often *knowledge* and *reasoning* are used interchangeably in A.I. literature). Deep knowledge refers to knowledge that can be represented by mathematics [27]. Shallow knowledge is knowledge learned from mentors, by experience, by analogy and includes commonly used terms like *expertise, heuristics,* and *rules of thumb.* Intelligent systems utilizing deep knowledge or reasoning require significant computing and memory resources to provide real-time performance.

Arzen gives an overview of the important aspects of real-time expert systems. These include: *Non-monotonic reasoning*, where information from the process and

inferred facts are dynamically updated; *temporal reasoning*, where the expert system is capable of explaining its observations and actions over time; *asynchronous events*, where events with higher priority can interrupt processing of events with lower priority and *reasoning under time constraints*, where the expert system must provide the best possible answer within a given time. Building an expert controller that can provide all or most of the above features requires significant computational power.

Krijgsman *et al.* [21], have developed a knowledge-based expert controller that uses *progressive reasoning*. The knowledge is divided into several layers and decision are propagated through the layers to gain deeper and deeper levels of conclusions. One form of the expert controller uses phase plane heuristics with just the controller error and the change in the controller error to set the controller output. They also build a Supervisory expert controller that uses adaptation by heuristic reasoning and pattern recognition. Expert PI and PID tuners that adapt tuning based on an under-damped response to a set-point change have been developed by several people [28, 29].

2.2 Fuzzy Logic

Fuzzy controllers represent the most widely used A.I. controllers in industry. Fuzzy controllers have been successful on process where a analytic process model cannot be easily obtained and on processes where the operator could control the process better than conventional control systems. Fuzzy logic is based on linguistic representation of variables. Excellent sources on fuzzy logic and mathematics are available [30, 31, 32, 33]. Dubious and Padre [34] give a comprehensive collection of references in fuzzy set theory and its applications. Extension of fuzzy logic to fuzzy controllers is described by Mamdani [35] and others [36, 37].

Fuzzy control has been widely applied to chemical processes. In most cases fuzzy control has provided as good a performance, or better than conventional PID control [38]. Mamdani [39] used a fuzzy logic controller to run a laboratory steam engine.

Umbers and King [40] successfully modelled operators behavior in a fuzzy logic controller to run a cement kiln. Kickert and van Nauta Lekme [41] applied fuzzy control to a warm water plant. Morita et. al. [42] reported the use of fuzzy logic to control glutamic acid fermentation. Østergaard [43] has applied fuzzy control to a counter current heat exchanger. Kandel and Langholz [44] provide a good collection of papers on fuzzy control.

Recently some work has been done on self-tuning and self-organizing fuzzy control systems. Siiao [45] has developed a real-time self-organizing controller that modifies the fuzzy control decision table to improve it performance. The auto-tuning of a fuzzy controller is much more complex than that of a conventional controller as the number of parameters to be adjusted is large.

2.3 System Identification

System identification plays a very important role in process control. System identification deals with attempting to obtain a dynamic model, with minimum effort, that will satisfy performance specifications. Several good sources review the different tools and techniques available for system identification[46, 47, 48, 49, 50, 51].

Experimental design is very crucial in system identification [52]. Various inputs can be used to identify the process model. A persistently exciting signal is required to obtain a good process model [53]. Usually impulse input is inadequate and step responses provide limited information. White noise and Pseudo Random Binary Signal (PRBS) provide a sufficient order of excitation to get good process models.

Process models can be represented using non-parametric or parametric models. Non-parametric models do not require any prior knowledge of the system, however can lead to over-parameterization. Impulse response, step response, corellation analysis and frequency response techniques provide non-parametric analysis. Parametric models require prior knowledge of the process. Input-output descriptions can be used

to obtain parametric models.

2.4 Adaptive Control

An adaptive controller is a controller that can adjust its behavior to compensate for changes in process dynamics and disturbances. Several good sources are available on adaptive control [54, 55, 56, 57]. Unbehaun [58] gives a good review of the application of adaptive control in the cement, chemical, metallurgical and paper industries. Adaptive control now provides several techniques that can be implemented for different conditions. Gain scheduling, auto-tuning, model-reference adaptive systems (MRAS), self-tuning regulators (STR) represent most of the adaptive control techniques available. If the process dynamics are fairly stable use an auto-tuner to adapt when the controller performance becomes poor. If the process dynamics are varying with the variations being unpredictable use a self-tuning controller. However if the variations are predictable use auto-tuning or gain scheduling controllers.

Gain scheduling is widely used when different operating conditions can provide good controller performance by just adjusting the controller gain. It is widely used in the chemical industry when grade changes are made or throughput is changed during manufacturing.

Auto-tuning is used mostly with PID controllers. Special test signals are injected in the process to determine the tuning parameters. These are simple to operate and several commercial controllers are available that provide a "auto-tune" button.

Model-reference adaptive controllers (MRAS) utilize a process model to predict the estimated process output. Based on the error between the true and the estimated output determine the controller parameters are adjusted to reduce the error to zero.

There are two types of self-tuning regulators, *direct* and *indirect*. Direct self-tuning regulators adjust the controller parameters directly. Indirect self-tuning regulators first estimate a process model and then adapt the controller parameters based on the

estimated model. Self-tuning regulators often use pattern recognition techniques to determine tuning constants. One commercial product by FOXBORO called the EXACT [29] adaptive controller uses pattern recognition to detect performance, in terms of damping and overshoot, during set-point changes and adjust tuning constants.

Recursive Least squares (RLS) and Recursive Instrumental Variables (RIV) method are often used in adaptive control [46, 54]. Adaptive control requires significant computing power and is usually implemented commercially in a box with embedded microprocessors. Industrial PLC's are often not suited for adaptive control unless the controller is implemented on separate hardware on the backplane.

Chapter 3

The Heat Exchanger System

In this chapter we give a brief overview of the experimental setup and its operational characteristics. The physical description of the heat exchanger is given first, followed by a description of the control system hardware and software. The heat exchanger provides a simple system for implementing intelligent control and has some peculiar characteristics that makes it an interesting process to study.

3.1 Experimental Setup

3.1.1 The Heat Exchanger

A shell and tube heat exchanger was used to study the intelligent controller. Water flowing through seven 5/8" OD, 57" long BWG 20 brass tubes is heated by steam on the shell side. A schematic of the setup is shown in Figure 3-1

Steam supply going through a block valve is regulated to a pressure of 70 *psig*. It then goes through a vortex flow-meter followed by a control valve into the shell. The heat exchanger is equipped with two types of steam traps selectable by solenoid valves. One is a bucket steam trap that operates by flushing the bucket when it is filled with water, and the other is a float type steam trap that maintains a constant flow of condensate through it. The heat exchanger is also equipped with a vacuum

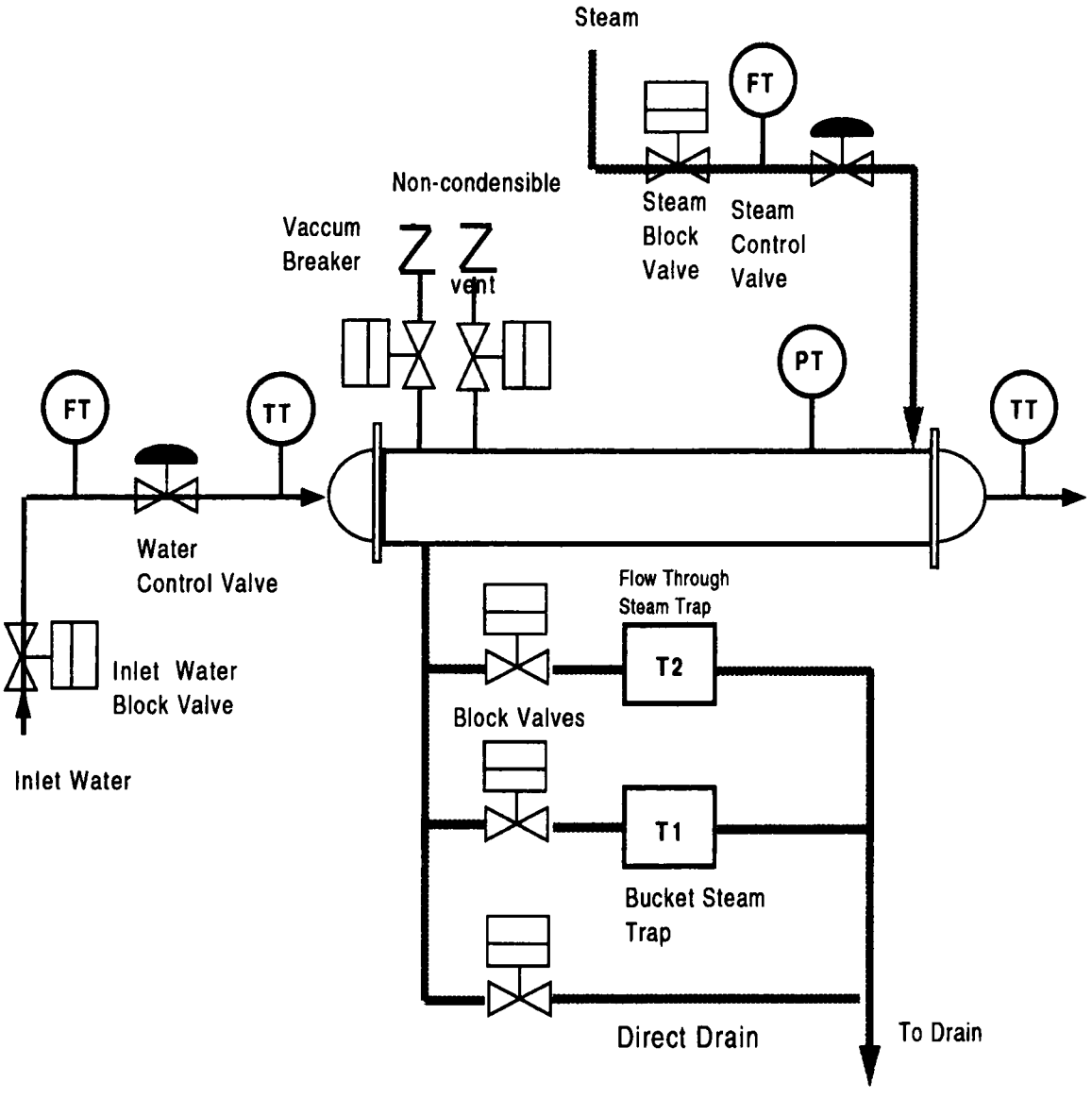


Figure 3-1: The heat exchanger system.

Table 3.1: Table of Sensor information.

Measured Variable	Sensor Type	Range
Inlet Water Temperature	Platinum RTD	50.0-100.0 $°F$
Outlet Water Temperature	Platinum RTD	50.0-100.0 $°F$
Water Flow Rate	Orifice Plate Meter	0.0-50.0 gpm
Steam Flow Rate	Vortex Flow Sensor	0.0-1050.0 lbs/hr
Shell Steam Pressure	Electronic Gauge	-5.0-25.0 $psig$

breaker and a non-condensible vent. If the demand for steam suddenly increases and cannot be met by the control system or the supply, a partial vacuum may be created which causes the condensate to collect in the shell, immersing some of the tubes in water. As a result the area of heat transfer changes causing poor operation of the heat exchanger. The vacuum breaker would open allowing air to enter the shell and flush the condensate. Repeated startup and shutdown of the heat exchanger and the use of the vacuum breaker can cause air to be trapped in the shell and reduce the partial pressure of steam. The non-condensible vent operates automatically, when the temperature of steam goes below 180 $°F$, letting the air out. A blow-down line can be operated by a solenoid valve to remove condensate from the steam line. The heat exchanger is instrumented with inlet and outlet water temperature sensors, inlet water and steam flow sensors and a shell side steam pressure sensor. All sensors provide a 4-20 MA output. Table 3.1 summarizes the sensor types and their ranges.

Two control valves are available to regulate the water flow rate and the steam flow rate. The instruments and the solenoid valves are wired into an auto/manual control station as shown in figure 3-2

Seven RTD sensors are also available. These are only used to perform an overall energy balance on the heat exchanger.

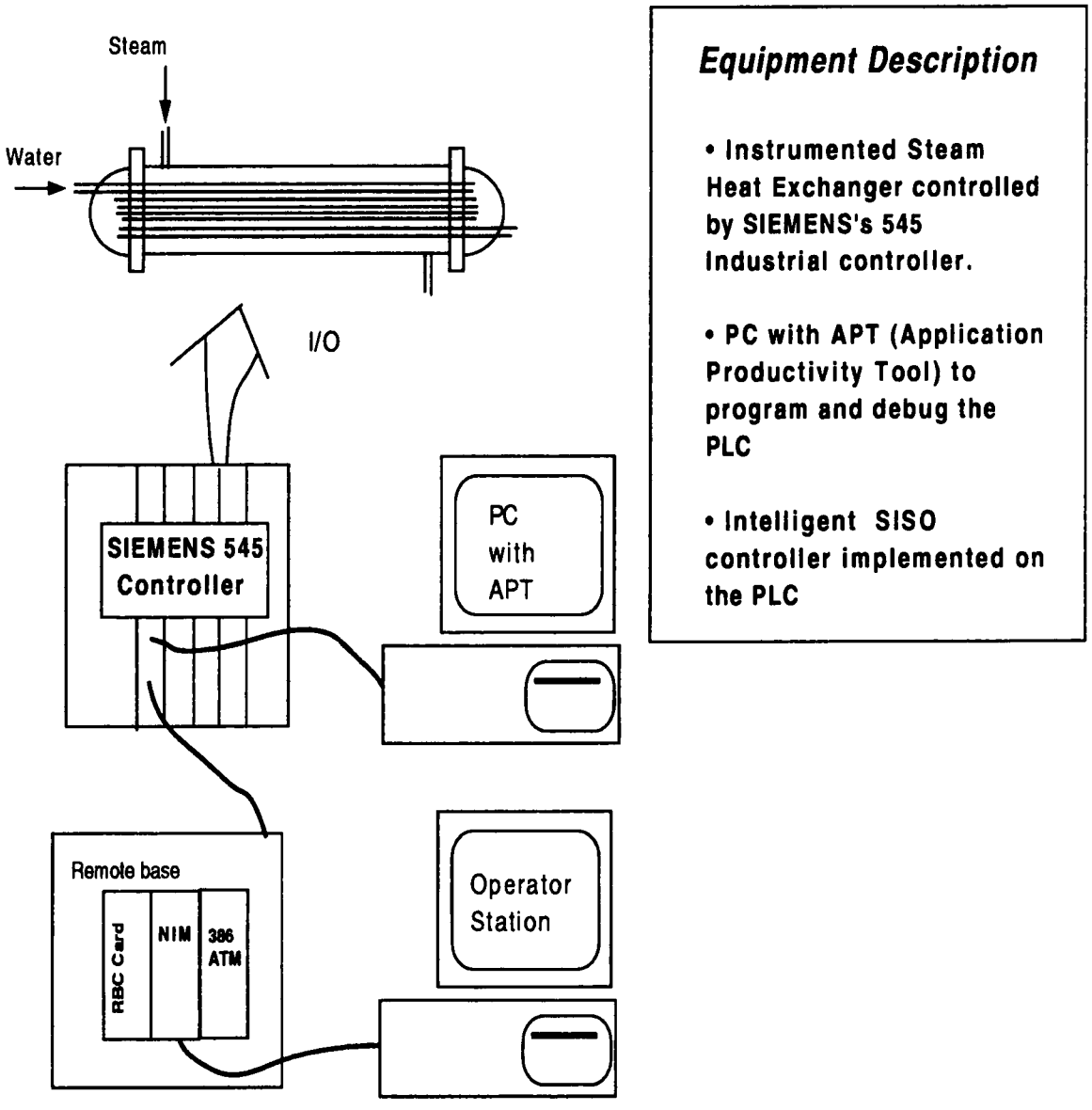


Figure 3-2: The Heat Exchanger Control System.

3.1.2 The Auto-Manual Control Station

The heat exchanger can be completely operated manually. Switches allow the operator to operate the devices by hand or by the PLC. The operator always has the ability to override the PLC by switching to manual operation. Meters permit the sensors to be read directly. Under automatic operation the heat exchanger is tied to a Siemens SIMATIC TI545 PLC. A remote base with a network interface module and a 386/ATM module provides operator interface connections and the ability to implement advanced control strategies, respectively. The operator interface is programmed on a Siemens TISTAR Model 20.

The PLC is programmed using APT (Application Productivity Tool). APT is a high level object-oriented programming tool for the PLC that does not require the knowledge of relay ladder logic. Process I/O can be represented by symbolic names. Devices can be represented by objects that can be acted upon by "verbs". APT permits the process and the code to be partitioned into units making programming and maintenance of the code easy. The two key features of APT are the CFB's (Continuous Function Blocks) and SFC's (Sequential Function charts). CFB's represent objects that can be executed either continuously, at fixed sampling intervals or based on events. SFC's represent procedural actions that are programmed using flowcharts. Devices, PID loops and several other CFB's are objects that have unique structures. Internal properties of these objects are accessible as variables using **Objectname.extension**. The objects have methods defined for them that can be operated by "verbs".

Startup and shutdown procedures for the heat exchanger are implemented using SFC's. A State safe SFC is implemented to permit immediate shutdown of the process by the operator if desired. CFB's (Continuous Function Blocks) are used to implement the remainder of the control algorithms. Interlocks are implemented to permit safe operation of the heat exchanger. The status of the manual control station can be monitored by the PLC and interlocks are implemented to disable PID loops

if the control valve is under manual operation. The rest of the control algorithms are implemented on the PLC in CFB's and SFC's except for the process model estimation using the PRBS (Pseudo Random Binary Sequence) technique. These are implemented on the 386/ATM module. The 386/ATM, a Series 505 module, is a 386SX computer with 4mB RAM and a 40mb hard disk and is capable of high speed backplane communications with the PLC. A Matlab/PLC interface was developed in C on the 386/ATM module. This interface provides Matlab functions to read and write PLC memory locations, and check the controller status. Thus any control strategy or system identification technique that can be implemented on Matlab can be used on the PLC. APT provides key startup parameters to be entered using recipes. This feature is used to store and update the intelligent controller database.

3.1.3 The Operator Interface

The operator interface for the heat exchanger is set up on TISTAR, a commercial operator interface package, that permits APT objects to be animated graphically on a screen. The operator interface runs on a 386/33 Mhz computer running SCO Unix. The operator interface is programmed to reflect the Auto/Manual status of the solenoid valves and the control valves. The package provides screens for real-time and historical trending. It also has a reports package that permits the data logged to be output to a spreadsheet. The package provides extensive alarming capabilities. The package has some limitations. Data cannot be logged for a sampling time of less than 5 seconds. TISTAR does not permit any user defined procedures to be executed. Later a Windows based MMI(man machine interface) package called LOOKOUT was acquired and used. LOOKOUT permits the use of rapid data sampling (less than 1 second). It does not have the ability to represent objects. Thus translating APT objects onto LOOKOUT is very time consuming. It provides DDE (Dynamic Data Exchange) support. DDE is a way of communicating between applications in Windows. This makes it easy to implement any procedure in C, FORTRAN or

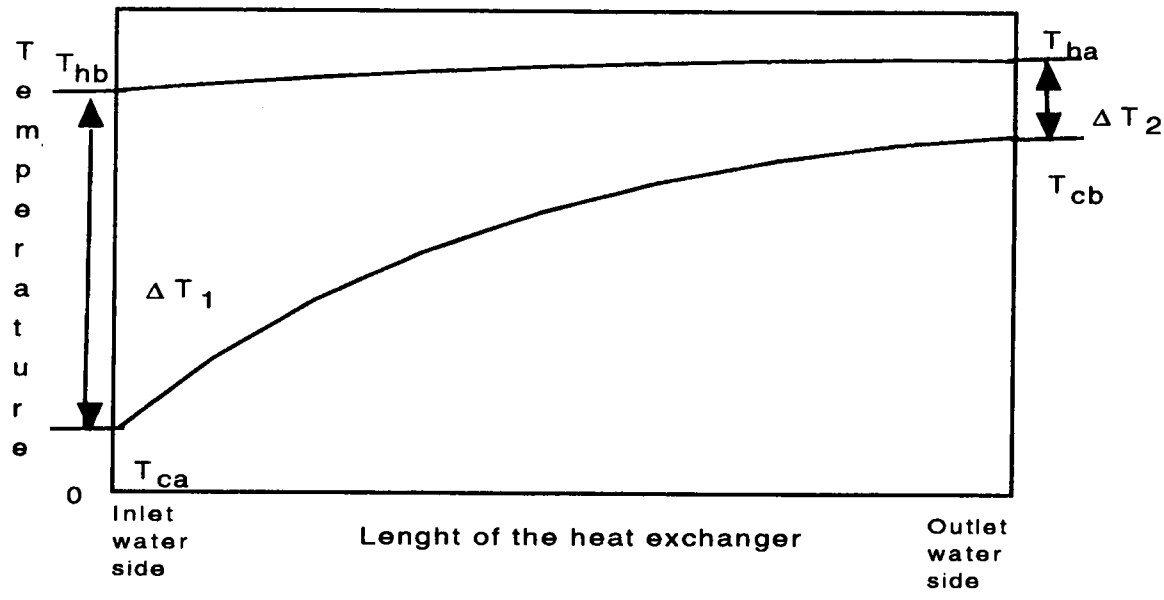


Figure 3-3: Typical temperature profiles in a heat exchanger.

VISUAL BASIC and communicate with the controller. It is desirable to have the combined features of both the MMI packages in one.

3.2 The Heat Exchanger Model

The heat exchanger was operated with the water flow rates being limited to 25 *gpm* as the steam supply cannot provide adequate steam to heat the water significantly above that flow rate. At a water flow rate of 25 *gpm* and the steam control valve completely open the maximum temperature rise obtained is about 40 $^{\circ}F$. The overall energy equation for the heat exchanger is given in eqn. 3.1 where the temperatures are shown in figure 3-3

$$\dot{m}C_p\Delta T = U_o A \Delta T_{lm} = Q_{steam} = \Delta H_v F_s \tag{3.1}$$

where

$\dot{m}$ = Mass flow rate inlb/hr

C_p = fluid heat capacity in $Btu/lb -^\circ F$

ΔT = change in fluid temperature $^\circ F$

U_o = Overall heat transfer co-efficientBtu/ft^2hr-°

A = Area available for heat transferft^2=5.44

ΔT_{lm} = log mean temperature difference $=\frac{\Delta T_2 - \Delta T_1}{\ln(\frac{\Delta T_2}{\Delta T_1})}$

ΔT_1 = temperature difference between the heating and the process fluid at the exit

ΔT_2 = temperature difference between the heating and the process fluid at the inlet

Q_{steam} = energy transfered by steamBtu/lb

ΔH_v = Latent heat of vaporization of steam Btu/lb

F_s = Mass flow of steam lb/hr

Based on an overall energy balance the overall heat transfer co-efficient for the heat exchanger is determined to be in the range of 800-1200 $\frac{Btu}{lb-ft^2-^\circ F}$ which is around the range of 1000-3000$\frac{Btu}{lb-ft^2-^\circ F}$ presented in literature [59] for a shell and tube heat exchanger with film type condensation.

3.2.1 Operation of the Heat Exchanger

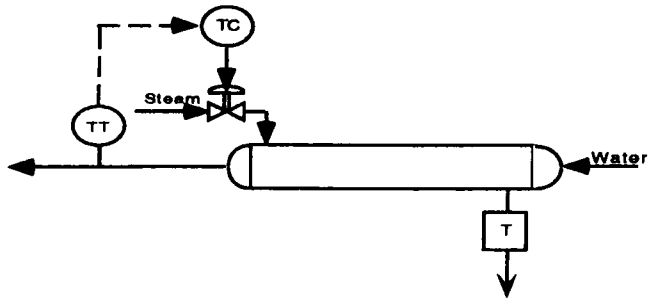
The startup procedure for the heat exchanger includes interlocking the steam block valve and opening the blowdown valve to remove any condensate from the steam line. After blowdown, the water block valve is opened and then the steam block valve. One of the two block valves to the steam traps must be opened. The bucket steam trap may not operate properly at startup because of its design. On cooling the trap collects air which prevents it from operating normally. However if the float trap is used at startup the hot condensate will warm the bucket trap and expel the air permitting it operate normally later. Shutdown involves putting all the controllers in the manual

mode and closing the steam block valve and then the water block valve.

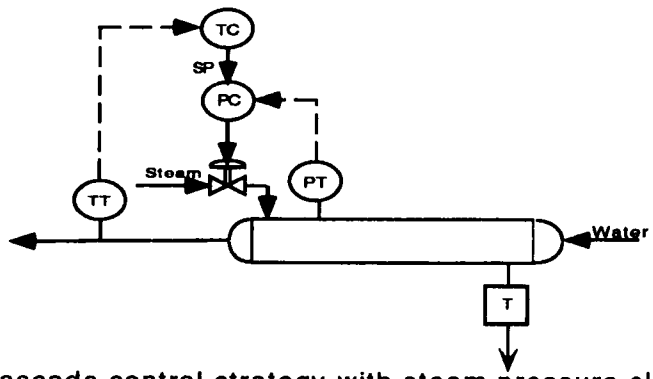
Three conventional control strategies can be implemented as shown in figure 3-4. One is direct control of the outlet temperature using the steam control valve. The two other are cascade control strategies. First uses a inner pressure slave loop the other a inner steam flow slave loop.

The direct control strategy (a) was not chosen as the process exhibits different process models for a step up and step down change in the steam control valve position, as shown in figure 3-5. The step responses are taken at a water flow rate of 13 *gpm*, for a change in the steam control valve of 10%. Lack of a postioner on the control valve can cause this. The cascade control strategies provide repeatable process behaviour for step up and step down changes on the slave controller setpoint. Strategy (b) was chosen for this research as it is more widely used. Heat exchangers are very common in industry and pressure sensors are relatively inexpensive compared to steam flow sensors. Strategy (b) also exhibits some peculier behaviour which makes it an interesting strategy.

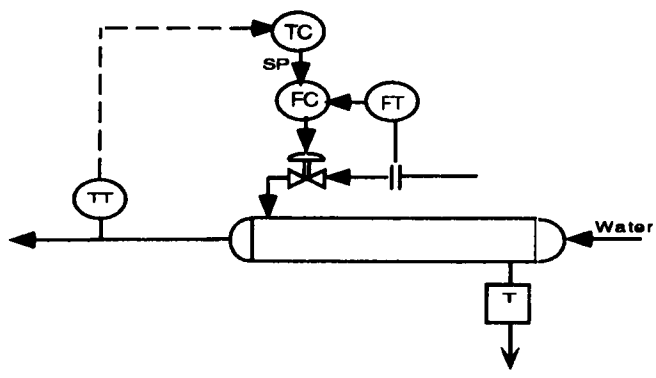
Strategy (b) results in certain undesirable modes of operation. The condensate from the heat exchanger drops 41 inches to the network of steam traps. The flow of condensate due to gravity can lead the heat exchanger to operate in a mode where the shell pressure is less than atmospheric. Condensation of steam in the shell involves a significant change in the volume and needs a constant supply of steam to maintain the pressure constant. If the steam available is inadequate, this can cause water logging of the condensate in the shell and prevent normal operation of the heat exchanger. However, lowering the setpoint on the temperature controller, where significant steam is being condensed, can result in the mode of operation where gravity flow of the condensate creates a partial vacuum in the shell, but the pressure is kept constant by sucking steam into the shell from the supply. The pressure slave controller senses a shell pressure of 0 and provides no change in the controller output. The pressure sensor was re-calibrated to a range of -5.0 to 25.0 *psig*. In this case it is observed that



(a) Direct control strategy



(b) Cascade control strategy with steam pressure slave loop



(c) Cascade control strategy with steam flow slave loop

Figure 3-4: Different temperature control strategies for the heat exchanger.

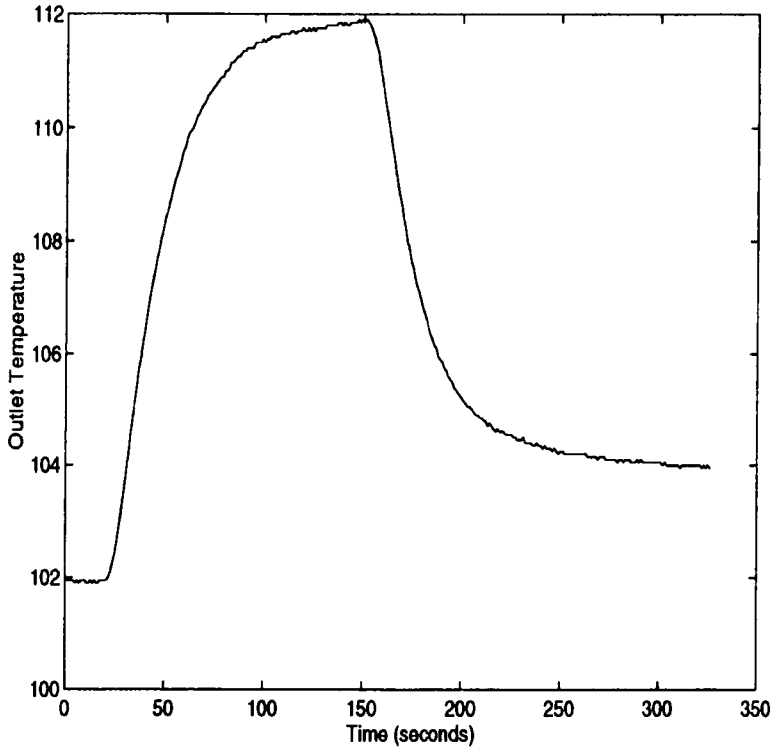


Figure 3-5: Step up and Step down response for direct control.

the shell pressure can go down to -2 *psig* before the vacuum breaker lets air into the shell. Figure 3-6 shows the below atmospheric shell pressure operation of the heat exchanger and figure 3-7 shows the operation of the vacuum breaker with the steam pressure sensor re-calibrated.

This can be remedied by making changes in the process. If the steam traps are moved closer to the exchanger the flow of condensate due to gravity would be reduced. It was later found that the float type steam trap was incorrectly installed.

Inability of large volumes of condensate to drain rapidly causes water hammer in the heat exchanger. Condensate being formed rapidly is pushed by the incoming steam to one end of the shell causing water hammer. This is indicated by a large banging sound and has already caused damage to the heat exchanger causing the water to leak into the shell side through the tube seams. This has been repaired once

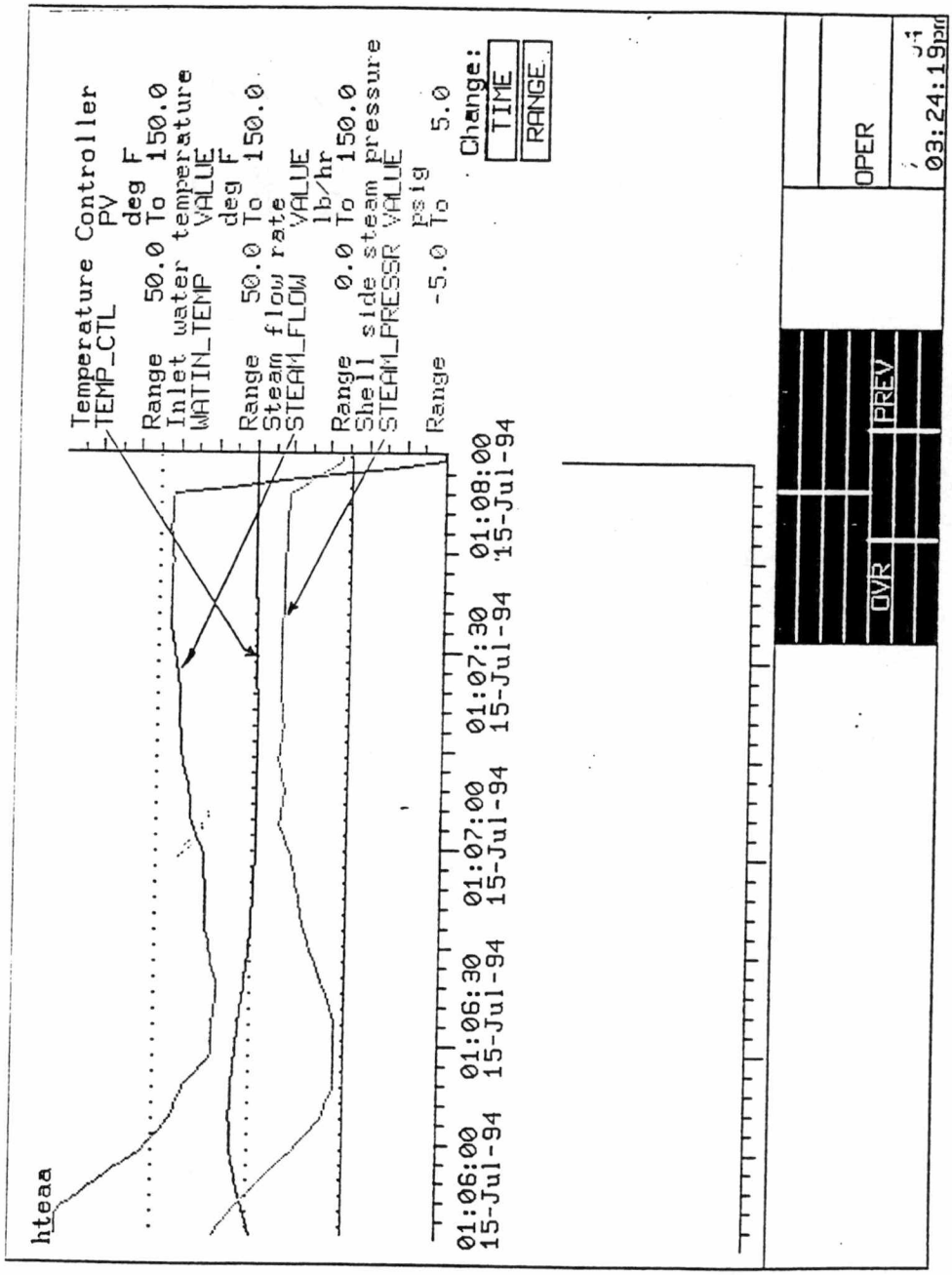


Figure 3-6: Below atmospheric Operation of the Heat Exchanger.

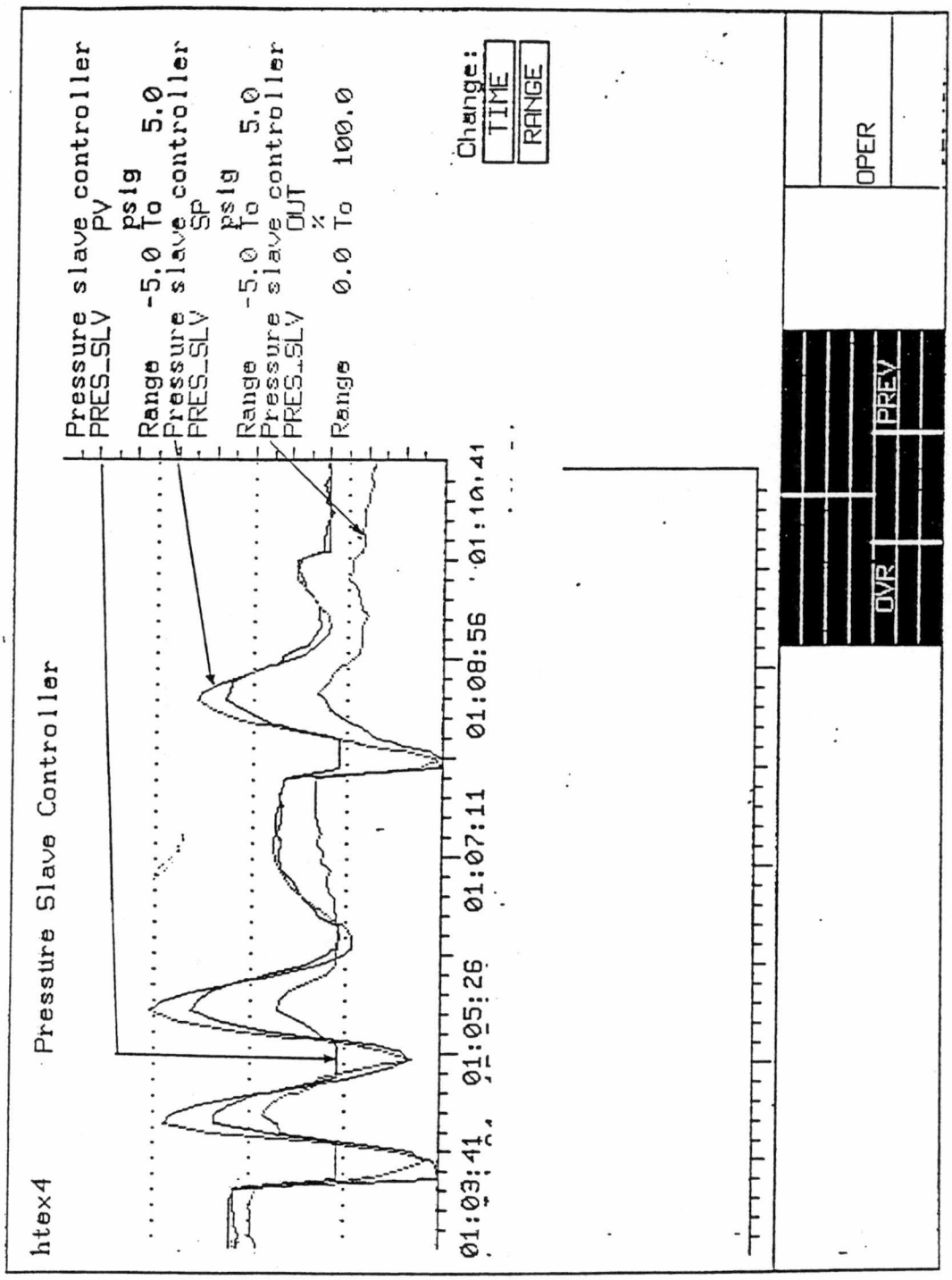


Figure 3-7: Vaccum Breaker Operation of the Heat Exchanger.

Table 3.2: Typical process models for the cascade control strategy.

Water Flow Rate	Process Model
12 *gpm*	$\frac{0.75e^{-5s}}{1+32s}$
18 *gpm*	$\frac{0.53e^{-5s}}{1+40s}$

but is likely to recurr again. The solution to the problem is by properly sizing the condensate lines and reinstalling the float type steam trap.

If the water flow rate is changed within the range of 9 *gpm* to 25 *gpm* the process model varies significantly. Some typical process model values are shown in table 3.2

The water flow sensor provides very inaccurate readings below 12 *gpm*. Thus the operation of the heat exchanger was limited to 9 to 25 *gpm* flows of water. With the steam control valve completely open the maximum change in the water temperature obtained was 40 $^{\circ}F$ at 25 *gpm*. Thus at higher water flow rates the process model becomes very non-linear because of limited steam supply. Figure 3-8 shows the performance of the cascade temperature control strategy at different flow rates. At a water flow rate of 16 *gpm* it provides stable operation. However when the flow rate is reduced to about 10 *gpm*, the control loop becomes unstable to setpoint changes. This led to the development of the intelligent PID (IPID) controller.

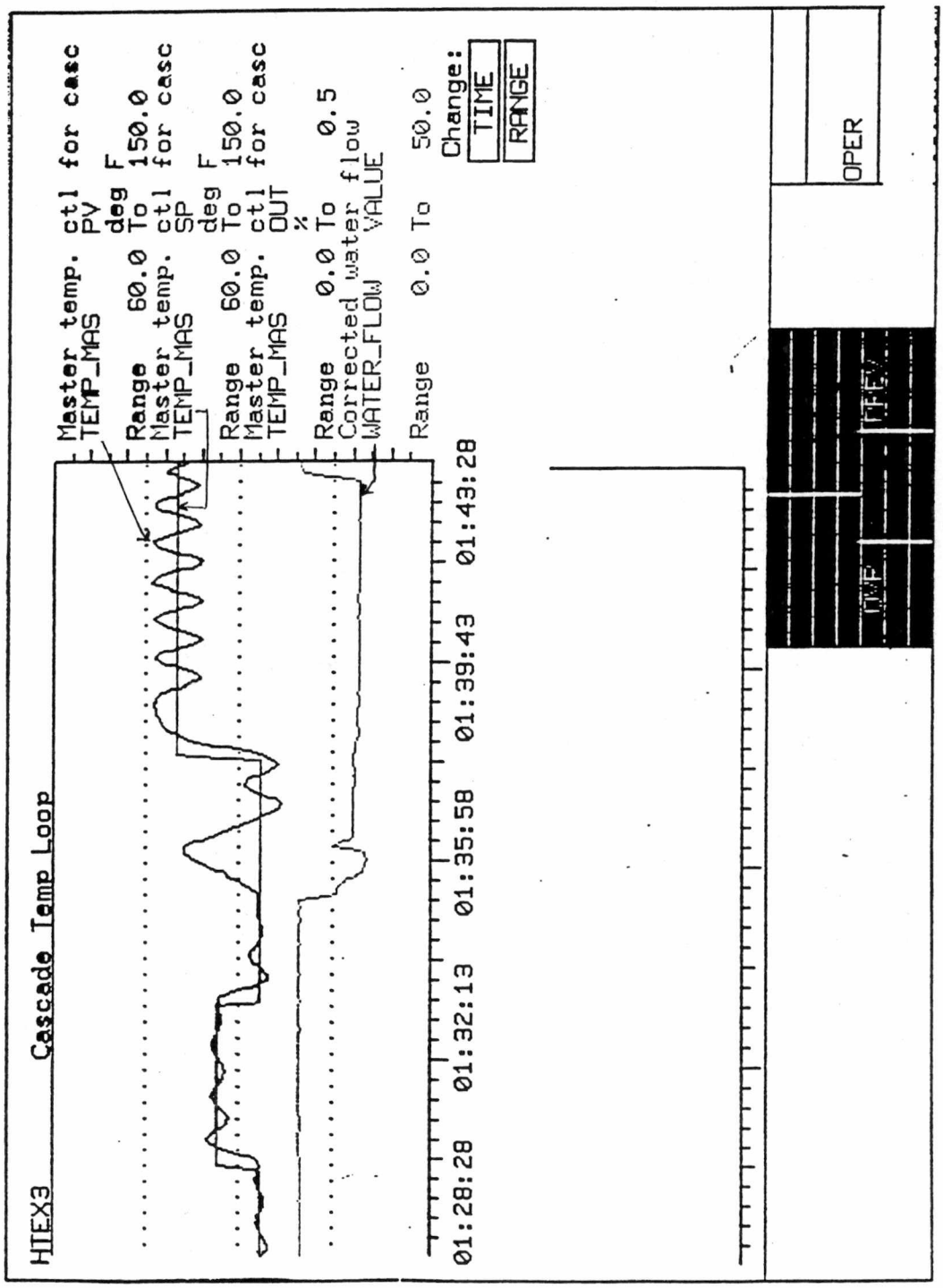


Figure 3-8: Performance of the cascade controller at water flow rates of 16 *gpm* and 10 *gpm*.

Chapter 4

The Intelligent SISO Controller

4.1 Introduction

This chapter describes the basis behind the intelligent controller, a brief description of the theory of the control algorithms and the system identification tools used. The experimental setup used provided limited computing power and memory. This led to the development of some qualitative reasoning tools that demanded a low computational overhead.

4.2 Detecting Trends in Process Variables

Characterization of process trends is desirable in analyzing the performance of the control system. Conventional mathematics provides regression techniques, Fast Fourier transform, power spectral densities, etc. to extract information from process variables. Some work done using neural networks is summarized by Maren [60]. Recent work done by Stephanopoulos and Bakshi [61] provides ways to determine the nature of process trends. However, most of these techniques are computationally intense. Most of the time the results from these analyses indicate that nothing much has happened. Often a lot of computing effort is spent to gain little information. A hierarchical

scheme for detecting process trends is desirable. At the top level simple qualitative tools detect changes in process variables and if necessary invoke the computationally intense tools for further analysis.

Differentiating between transient and permanent changes in disturbance variables can he helpful. An intelligent controller should be able to evaluate its own performance. The ability to detect oscillations can be used to determine a tightly tuned controller. Oscillations in other process variables can induce oscillations in the controlled variable. This does not indicate that the controller is poorly tuned. Two sets of tools have been developed for this research. One permits the detection of permanent and transient changes in process variables, the other detects oscillatory behavior.

4.2.1 Detecting permanent and transient changes in process variables

Shallow reasoning qualitative tools, the deviation decay term and the deviation decay rate term have been developed for this research. Consider the process model shown in figure 4-1. Several disturbance variables affect the process output which is the controlled variable. We wish to know in a qualitative sense how changes in this process variable affect the controlled variable. Consider a process variable, at steady state its value is fixed and can be taken as a reference to form the base case. Let x_b be the base case value for a process variable. Let $e(t)$ be the absolute value of the deviation of the process variable from the base case. As the process drifts from steady state or is reacting to disturbances we wish to classify the nature of the drift. Form a dead-band around $e(t)$ so that we do not respond to noise. Compute the deviation decay term (dde) and the deviation rate decay term (ddr) for the process variable using equations 4.1 and 4.2 respectively.

$$d_e(t) = \delta_1 * d_e(t-1) + e(t) \tag{4.1}$$

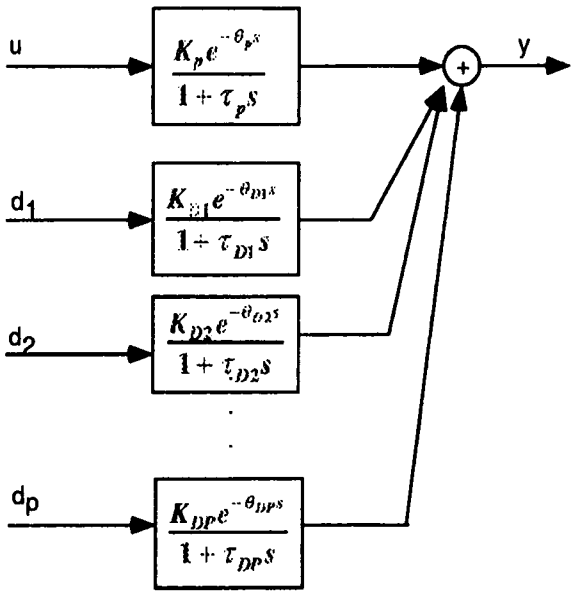


Figure 4-1: A general process model.

where

$d_e(t)$ = deviation decay term at time t

$d_e(t-1)$ = deviation decay term at time $t-1$

$e(t)$ = is the error term at time t
(absolute value of the deviation from the base case)

δ_1 = is the decay rate $[0 < \delta_1 < 1]$

T_s = is the sampling time

τ = is the leading process time constant

$$\dot{d}_e(t) = \delta_2 * \dot{d}_e(t-1) + |e(t) - e(t-1)| \tag{4.2}$$

where

$\dot{d}_e(t)$ = deviation decay term at time t

$\dot{d}_e(t-1)$ = deviation decay term at time $t-1$

$e_c(t)$ = is the error term at time t
(absolute value of the deviation from the base case)

$e(t-1)$ = is the error term at time $t-1$

δ_2 = is the decay rate $[0 < \delta_2 < 1]$

T_s = is the sampling time

τ = is the leading process time constant

All the variables are normalized to the sensor span or a heuristic span (selectable by the user). The sampling time T_s needs to be less than $\tau/6$ to get meaningful results using this technique. The selection of δ_1 and δ_2 is heuristic. We are interested in a time period of 2 to 3 times the time constant (τ) of the process. During this time the process is reacting to the disturbance. The behavior of these terms for a permanent and transient change is shown in figures 4-2 and 4-3. The effect of different values of δ_1 and δ_2 are shown in figure 4-4.

The logic in determining the nature of change is as follows:

- Define two thresholds th_1 and th_2 for each of the terms respectively. The choice is heuristic, typically between 1-10% of the sensor span.

- If the $\dot{d}_e(t)$ returns within the threshold th_2 and $d_e(t)$ is above its threshold th_1, then the change is permanent. The process variable has moved to a new steady state. Update the base case and reset the *dde* term and the *ddr* term.

- If $d_e(t)$ comes within the thresholds th_1, then the change is transient and the process variable has returned to its original value and the effects of this disturbance on the controlled variable have passed.

Proper use of these terms provides a boolean disturbance indicator. When $d_e(t)$ or $\dot{d}_e(t)$ exceed the threshold, the boolean indicator turns true and remains true till

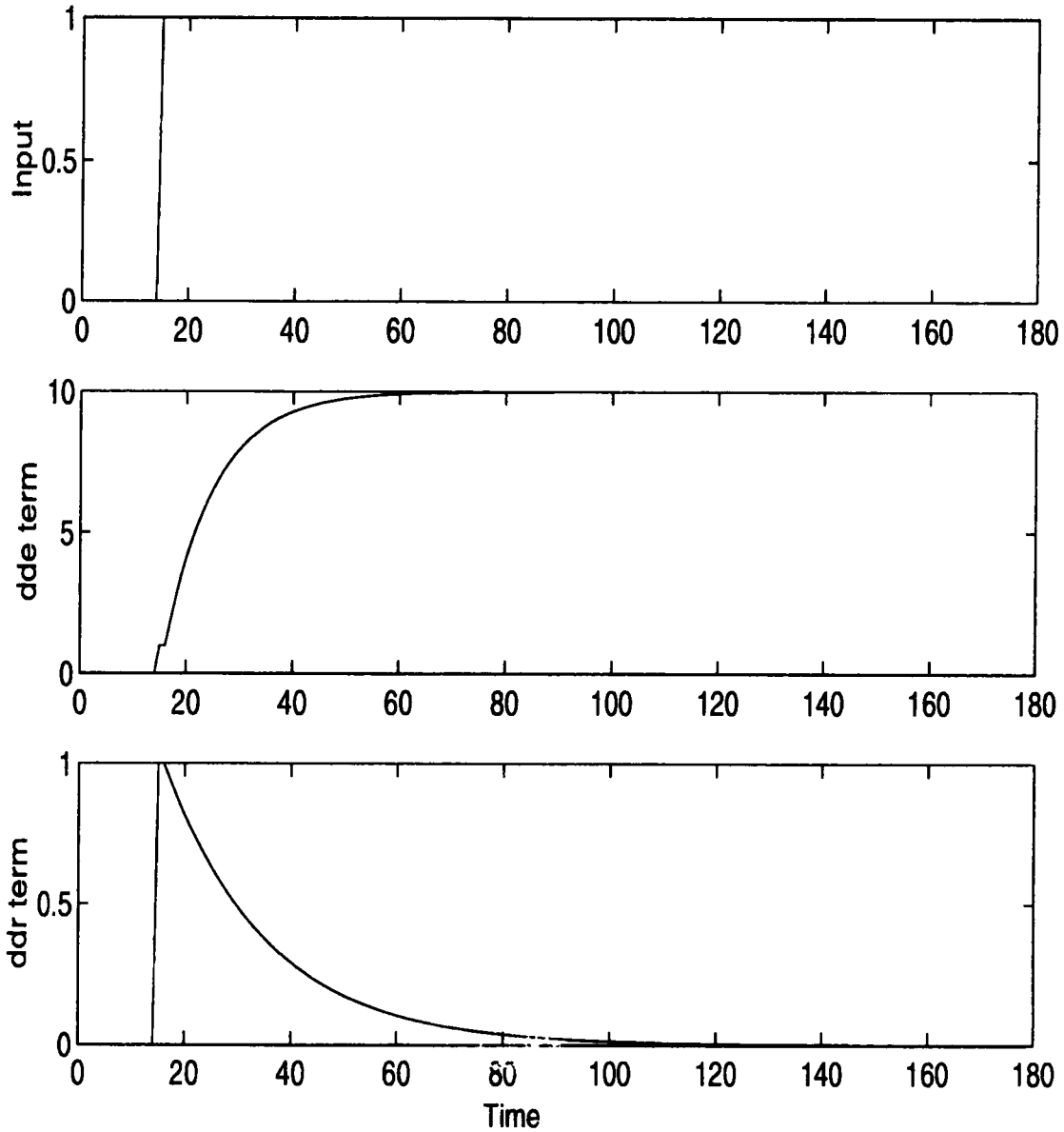


Figure 4-2: Behaviour of the deviation decay terms for a permanent disturbance.

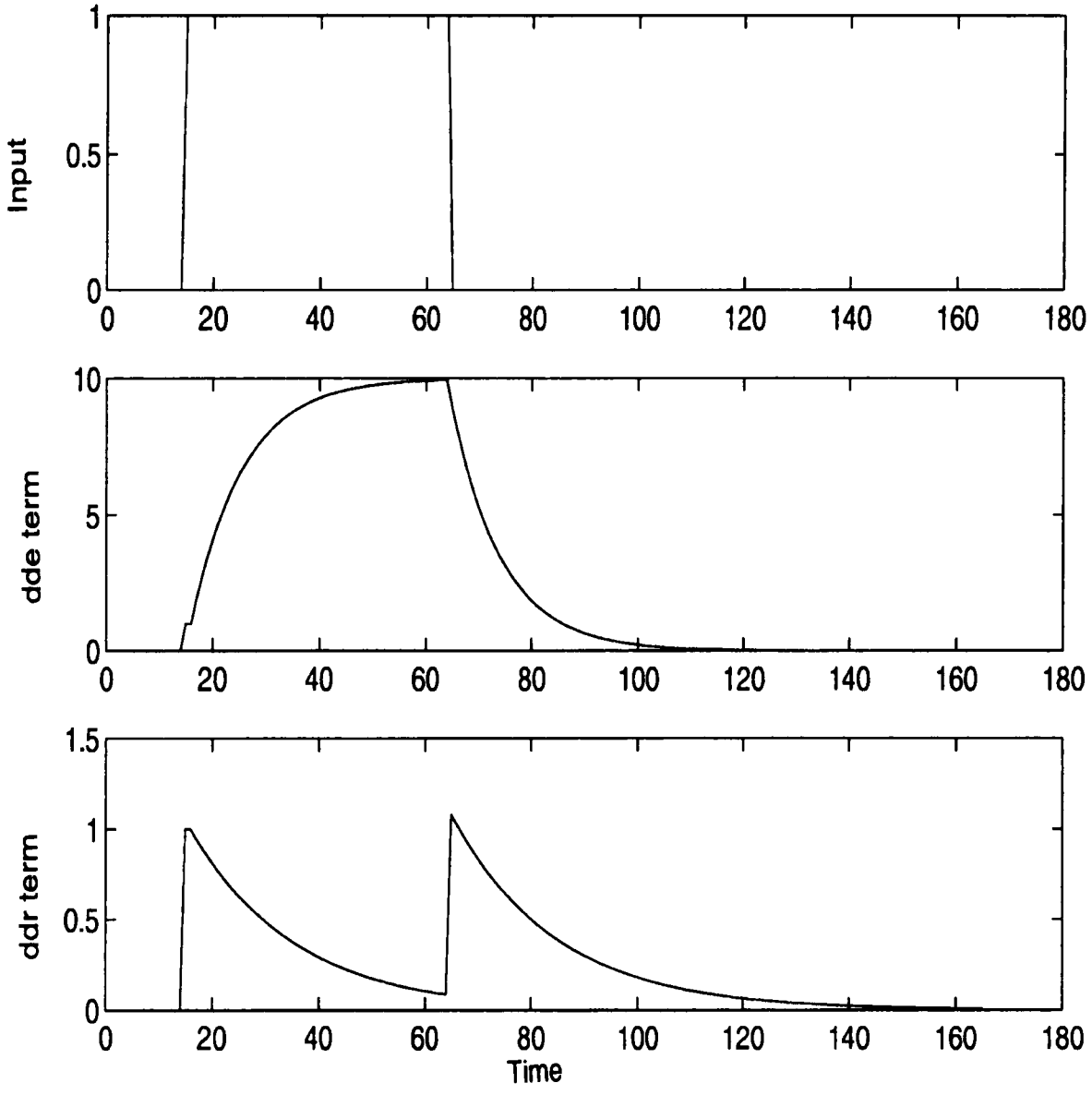


Figure 4-3: Behaviour of the deviation decay terms for a transient disturbance.

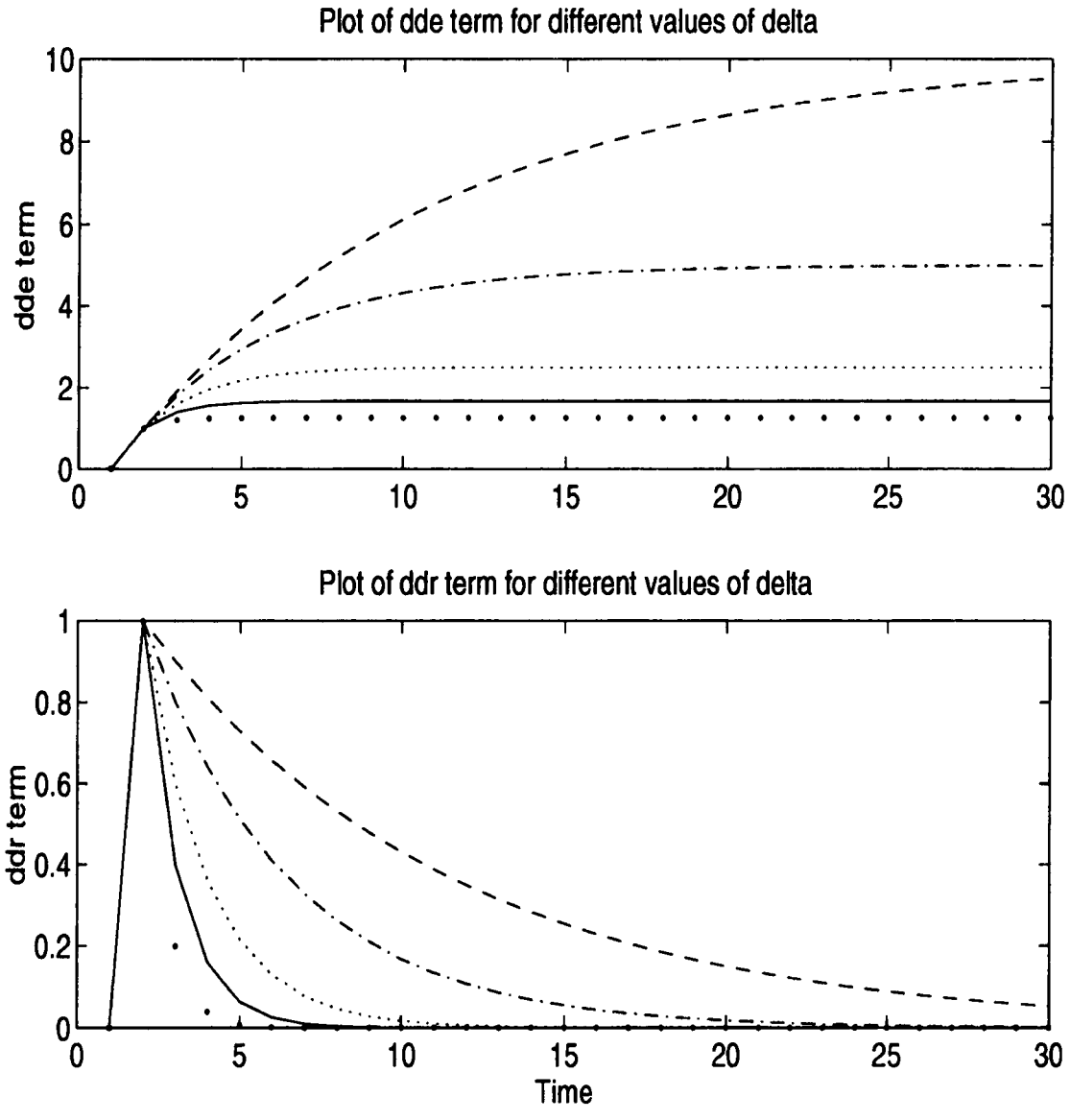


Figure 4-4: The effect of different values δ_1 and δ_2 on the *dde* and *ddr* term for a step change in the process variable. The (. .) line corresponds to δ_i of 0.2, the solid (-) line to a δ_i of 0.4, the dotted (..) line to a δ_i of 0.6, the dash-dot (-.) line to a δ_i of 0.8 and the dashed (– –) line to a δ_i of 0.9

one of the above criteria is met.

Selection of the decay rates

The $\delta_i's$ are chosen such that for a temporary disturbance the $d_e(t)$ returns within the threshold before the $\dot{d}_e(t)$ for a transient that lasts less than 2 times τ. A starting choice for δ_i would be $e^{-Ts/\tau}$.

Properties of the deviation decay terms

- For a very slow drift in a process variable, (e.g. cyclic variations of ambient temperature) the *ddr* terms do not indicate any thing. However when the error from the base case gets large enough such that the *dde* term goes above the threshold, the base case is updated.
- For a sine wave input with a time period comparable the the process time constant, the *dde* and *ddr* terms indicate a disturbance is affecting the process.

4.2.2 Extension to the Multivariate Case

The concept deviation decay terms can be extended to the multivariable case. Consider the process shown in figure 4-1. Several disturbance variables affect the controlled variable. There are two ways the deviation decay terms may be computed.

1. Compute $d_e(t)$ and $\dot{d}_e(t)$ based on $e(t)$ that is the euclidean norm of the error vector $\underline{e}(t)$ described in equation 4.3.

$$\underline{e}(t) = \underline{D}_n(t) - \underline{D}_b \tag{4.3}$$

where

$\underline{e}(t)$ = error vector = $[e_1(t) \cdots e_p(t)]$

$\underline{D}_n(t)$ = vector of normalized disturbance variables = $[D_{1,n}(t) \cdots D_{p,n}(t)]'$

$\underline{D}_b$ = vector of normalized disturbance variables at base case = $[D_{1,b} \cdots D_{p,b}]'$

the $\delta_i's$ are based on the largest τ_{Di} that affects the controlled variable. This technique does not require all the τ_{Di} to be known, but has the disadvantage that a disturbance variable that affects the controlled variable with a small time constant will cause the disturbance flag to stay on long after its effects have passed. Use the criteria mentioned above to determine the nature of the disturbance.

2. Compute $\underline{d}_e(t)$and $\dot{\underline{d}}_e(t)$as shown in equations 4.4 and 4.5 respectively. Then compute $d_e(t)$ and $\dot{d}_e(t)$ using equations 4.6 and 4.7 respectively.

$$\underline{d}_e(t) = \delta_1 \underline{d}_e(t-1) + \underline{e}(t) \tag{4.4}$$

where

$\underline{d}_e(t)$ = $[d_{i,e(t)} \cdots d_{p,e(t)}]'$

$$\delta_1 = \begin{bmatrix} \delta_{1,1} & 0 & \cdots & 0 & 0 \\ 0 & \delta_{1,2} & & & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & & & \delta_{1,p-1} & 0 \\ 0 & 0 & \cdots & 0 & \delta_{1,p} \end{bmatrix}$$

$\underline{e}(t)$ = $[e_1(t) \cdots e_p(t)]'$

$$\dot{\underline{d}}_e(t) = \delta_2 \theta \dot{\underline{d}}_e(t-1) + \underline{e}_{abs}(t) \tag{4.5}$$

where

$$
\begin{aligned}
\underline{\dot{d}}_e(t) &= [\dot{d}_{1,e}(t) \cdots \dot{d}_{p,e}(t)]' \\
\delta_2 &= \begin{bmatrix}
\delta_{2,1} & 0 & \cdots & 0 & 0 \\
0 & \delta_{2,2} & & & 0 \\
\vdots & & \ddots & & \vdots \\
0 & & & \delta_{2,p-1} & 0 \\
0 & 0 & \cdots & 0 & \delta_{2,p}
\end{bmatrix} \\
\theta &= \begin{bmatrix}
\theta_1 & 0 & \cdots & 0 & 0 \\
0 & \theta_2 & & & 0 \\
\vdots & & \ddots & & \vdots \\
0 & & & \theta_{p-1} & 0 \\
0 & 0 & \cdots & 0 & \theta_p
\end{bmatrix} \\
\underline{e}_{abs}(t) &= [|e_{1(t)} - e_1(t-1)| \cdots |e_p(t) - e_p(t-1)]'
\end{aligned}
$$

$$
d_e(t) = \|\underline{d}_e(t)\| \tag{4.6}
$$

$$
\dot{d}_e(t) = \left\|\underline{\dot{d}}_e(t)\right\| \tag{4.7}
$$

The $\theta_i's$ represent scaling factors. Certain variables have a weak effect on the controlled variable. Scaling permits the user to differentiate between the effects of strong and weak variables.

The deviation decay terms determine qualitative changes in process variables. The computational overhead is low and these terms for a large number of process variables can be computed. However for processes with a very large number of process variables, or for plant wide analysis the number of disturbance flags can be very large and confusing. Computing these terms for the multivariate case can reduce the number of disturbance flags.

The second method allows the inclusion of the effects of the different time constants that affect the system and the disturbance detection system would no longer depend on the dominant time constant of the system. In addition it provides an

hierarchical system for disturbance detection. If the disturbance flag is true and the user wishes to know what are the variables that are undergoing the disturbance the user just looks at the individual terms. We now look at the development of tools that can identify oscillations.

4.2.3 Detecting oscillations in process variable trends

The detection of oscillations in process variables and especially in controlled variables is important. Time shifted distribution(TSD) developed by Hackney [62] can help determine oscillations from past performance. It is an off-line technique that requires significant historic data for analysis. We wish to detect oscillations within a few cycles. For an intelligent controller to be able to evaluate its performance it needs to know when the controlled variable is oscillating. These oscillations can be caused by poor controller tuning or by oscillations in other process variables that transmit the variation to the controlled variable. The positive and negative error decay terms have been developed for this research to permit the detection of oscillations in a process variable and are computed as follows:

- First compute a exponentially smoothed term PVs using equation 4.8

$$PV_s(k) = \alpha * PV_s(k-1) + (1-\alpha) * PV(k) \tag{4.8}$$

- Then compute the error e using equation 4.9

$$e = PV(k) - PV_s(k) \tag{4.9}$$

Note: For the controlled variable let e be the controller error $(SP - PV)$.

- Compute the positive and negative error terms using equations 4.10 and 4.11 respectively.

$$\begin{aligned} e_+ &= |PV - PV_s| \quad \text{if } e > 0 \\ &= 0 \qquad\qquad otherwise \end{aligned} \tag{4.10}$$

$$\begin{aligned} e_- &= |PV - PV_s| \quad \text{if } e < 0 \\ &= 0 \qquad\qquad otherwise \end{aligned} \tag{4.11}$$

- Compute the positive and negative error decay terms using equations 4.12 and 4.13 respectively.

$$d_{e+} = \delta * d_{e+}(t-1) + e_+(t) \tag{4.12}$$

$$d_{e-} = \delta * d_{e-}(t-1) + e_-(t) \tag{4.13}$$

where

- Define a threshold, th_{osc}
- Check the condition when either d_{e+} or d_{e-} is above the threshold th_e.
- If d_{e+} is above the threshold and d_{e-} is not, initialize the d_{e-} counter. Next check for the d_{e+} term to go below the threshold. When d_{e+} goes below the threshold, check to see if d_{e-} is above the threshold. If it is, increment the d_{e+} counter by one. Next check for the d_{e-} term going below the threshold, and if the d_{e+} term is above the threshold increment its counter by one. Thus count the number of crossings below the threshold for both the error decay terms. If both the counts are greater than or equal to two then the process variable is oscillating. If both counts are less than two and both the error decay terms are below the threshold th_e then the oscillations have decayed.

Once oscillations are detected. Reset the counters and restart checking for oscillations. Simulation studies have shown that that the sampling time, T_s has to be at least 1/5 the period of the frequency that needs to be detected.

It should be noted that any time a process variable is oscillating, the *disturbance* flag is active. Thus when the process is at steady state the computation of the oscillation indicators is inhibited. This provides for an hierarchical scheme for reasoning.

Selection of the decay rate δ

The selection of δ is heuristic. Simulation studies have shown that for a control loop that has an ultimate period, T_u, selecting δ to be between $e^{-1.1\frac{2\pi T_s}{T_u}}$ and $e^{-1.5\frac{2\pi T_s}{T_u}}$ would permit the detection of oscillations with periods ranging from $0.5T_u$ to $10T_u$ (or a frequency range of $2\omega_u$ to $0.1\omega_u$, where ω_u is the ultimate frequency $= 2\pi/T_u$).

This scheme permits the detection of oscillatory trends with or without a drifting mean. The behavior of the system cultivation tools for detecting oscillation is shown in figure 4-5.

4.3 Control Algorithms

The intelligent controller is capable of implementing several control algorithms. It is designed so that additional algorithms can be easily added.

4.3.1 Standard PID algorithm

The SIMATIC TI545 controller implements the standard PID algorithm as shown in equation 4.14.

$$m_n = K_c \left[e_{cn} + \frac{T_s}{T_i} \sum e_{cn} - \frac{T_d}{T_s}(PV_n - PV_{n-1}) \right] + m_x \qquad (4.14)$$

where

m_n	=	output at time n	K_c	=	controller gain
m_x	=	bias	T_i	=	reset time (min)
e_{cn}	=	error	T_d	=	rate time (min)
PV	=	process variable	T_s	=	sample time

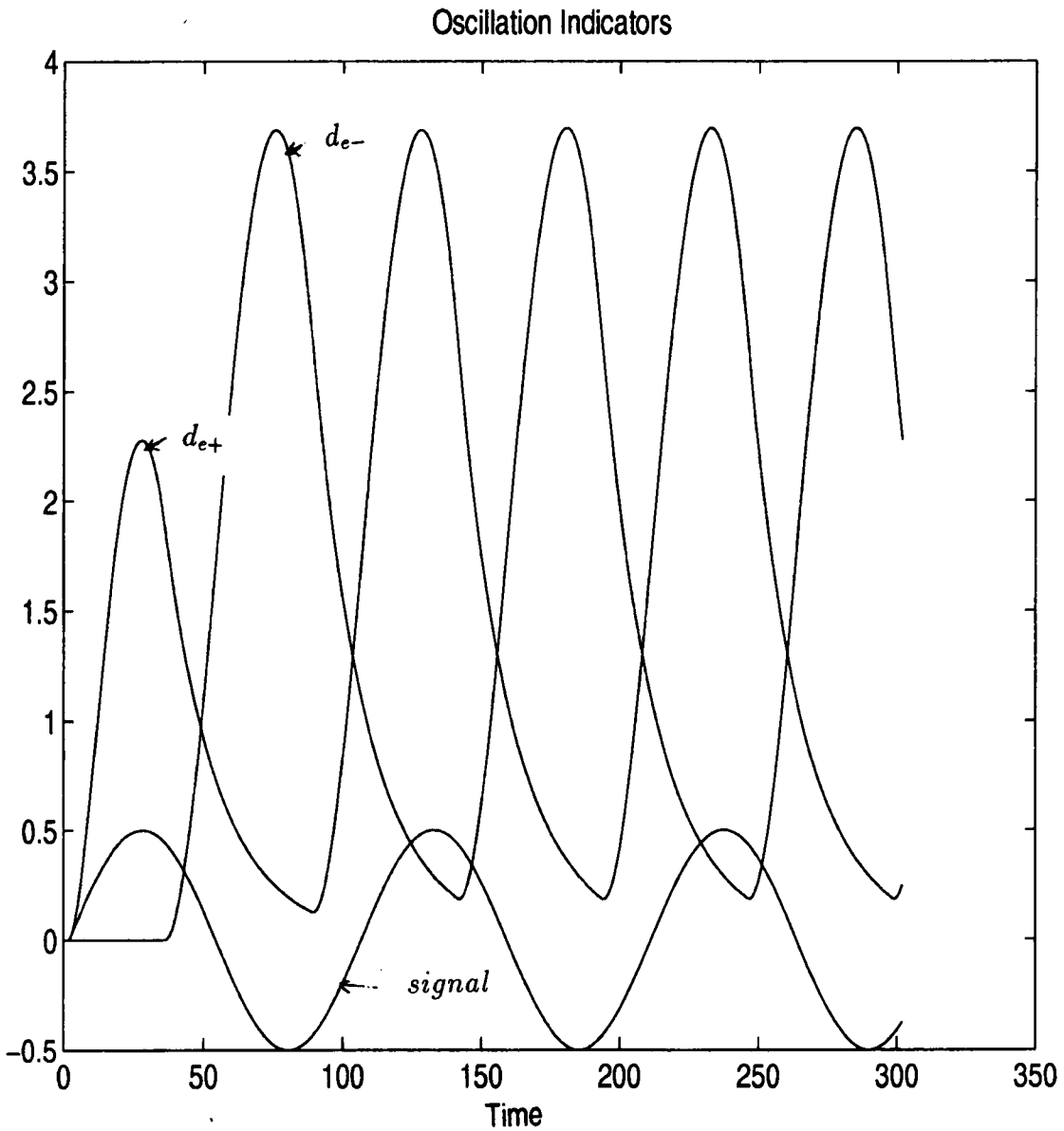


Figure 4-5: Behavior of the d_{e+} and d_{e-} terms for a sine wave input.

The PID controller is implemented using object oriented programming and provides some useful features. The PID loop provides in its structure, variables for *orange and yellow deviation alarms, rate of change alarms, high, high high, low and low low alarms.* The PID controller can operate in one of three *modes.* These are *Manual, Automatic or Cascade.* The PID block allows the user to associate math code (user written code) to be executed on during the following conditions.

- *PV cycle*: Every time the controller reads the process variable.
- *Auto*: Every sample time the controller is in the automatic mode.
- *Cascade:* Every sample time the controller is in the cascade mode.
- *Output*: Every time the output is computed irrespective of what *mode* the controller is in.

Thus code was implemented in the option *Associate math on auto:* to compute the integral square error and the integral absolute error to evaluate controller performance. The operator interface utilizes these properties to animate the controller and implement alarms. The control system is designed such that the operator always has control over the control loops, i.e. he can always disable the controller by putting it in the manual mode at the operator interface.

On setting another variable, *objectname*.**AWS**, to 3, setting Kc to 0, any value generated by user written code, written to the *objectname*.**LMN** variable sets the output of the controller while the controller is in automatic mode. All other control and tuning algorithms are implemented through the PID controller using this scheme. They are able to utilize the object properties of the PID controller. Performance evaluation for the controller is computed in the *Associate math on auto block* will work for any external controller. Disabling the PID controller would disable any other advanced controller or tuning algorithm.The PID controller provides

a *objectname*.**NRDY** read/write boolean variable, which when set to true, puts the controller in manual mode.

APT does not support user defined objects. However a general object-oriented style framework is provided for the other control and tuning algorithms to make implementation and code maintenance easy. Every advanced algorithm has the following variables:

- *Objectname*_START. When set to true will start the algorithm with the initialization procedure.
- *Objectname*_ACTIVE. After successful initialization, this bit is set to true while the algorithm is active.
- *Objectname*_END. When set to true will gracefully terminate the algorithm.
- *Objectname*_NRDY. When set to true indicates that the system is not ready or willing to implement the algorithm or if the algorithm is active will terminate it.

4.3.2 The Intelligent PID (IPID) Controller

The varying process model for the heat exchanger provids sluggish operation at high water flow rates as the conventional PID controller had to be tuned around the low water flow rates to provide stable operation. This led to the development of the intelligent PID (IPID) controller. The intelligent PID controller is implemented on the PLC and does not require the use of an external computer. It partitions the operating space of the heat exchanger into four regions as shown in figure 4-6.

A system cultivation daemon maintains a database of nominal tuning parameters for direct control and cascade control. The system cultivation daemon uses the deviation decay terms to detect variations in the water flow rate and the inlet temperature

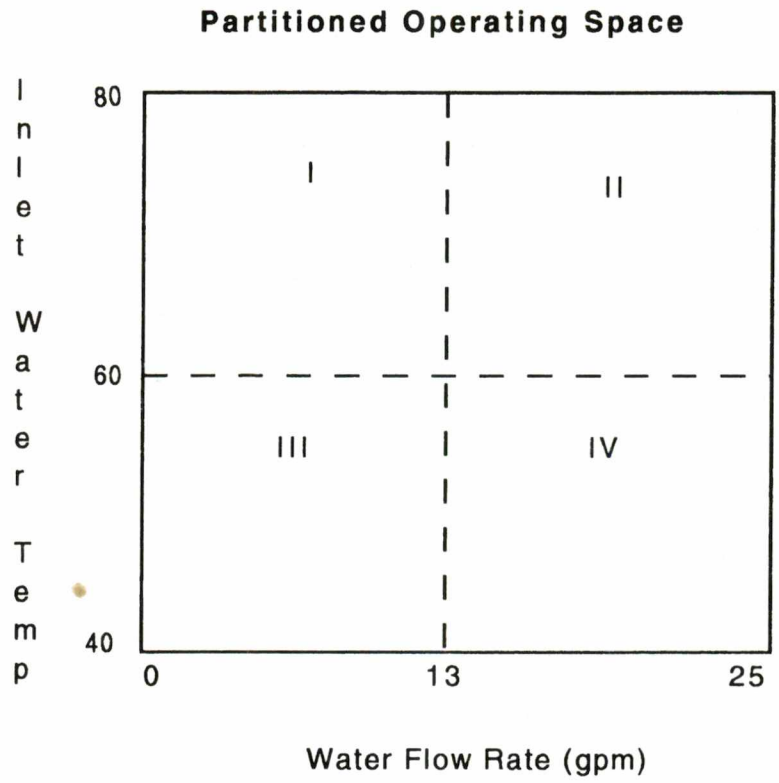


Figure 4-6: Patritioned Operating Space for the heat exchanger.

around a base case. This reduces the interference due to noise. A typical trajectory is shown in figure 4-7

The process is originally at the base case B0. It moves to point 1 via path A. Since the change is permanent the base case is updated to B1. The process now moves along path B across the operating region boundary and returns back to point 1. The tuning constants are not updated. The process new moves to operating point 2 and since the change is permanent the base case is updated to point B2. The tuning parameters applicable for the new operating region are applied. If an auto-tuner is activated for system identification when the IPID controller is active, the tuning parameters for the appropriate operating region are updated.

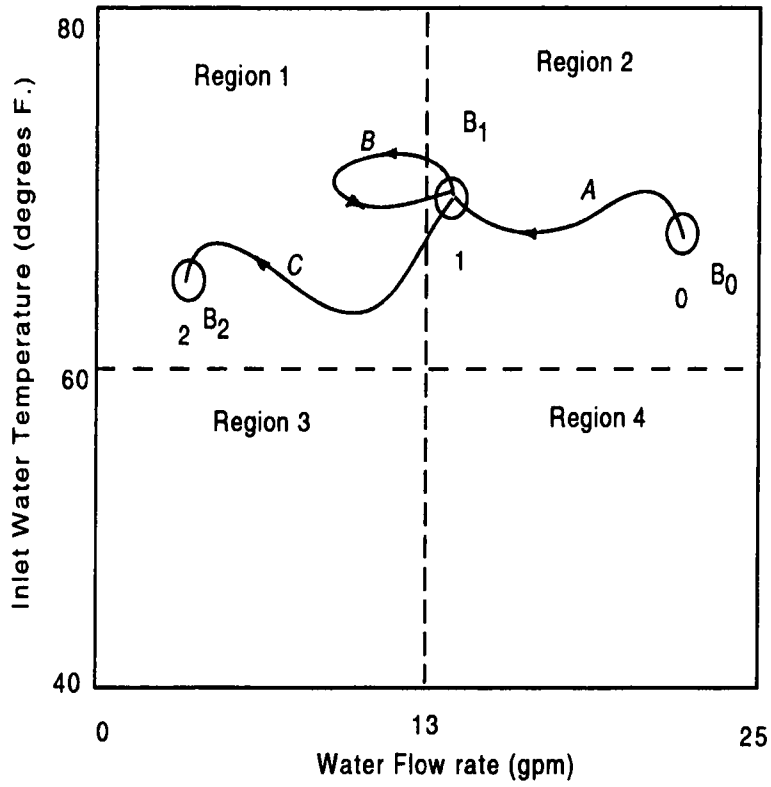


Figure 4-7: Movement of the process variables in the operating regions.

4.3.3 Dynamic Matrix Control

Dynamic Matrix Control (DMC) was implemented on the PLC. DMC is a model based control algorithm that utilizes a step response of the process. The theory behind DMC is described in several references [63, 64]. The formulation of DMC is based on the step response of the process shown in equation 4.15.

$$\delta x = \underline{a}\Delta u \tag{4.15}$$

where $\underline{a}$ is the vector of coefficients for the step response at sampling times $T, 2T, 3T, \cdots NT$. Based on the assumption that the model is linear, if we know the previous set of inputs we can determine the output for M steps using equation 4.16.

$$\underline{x} = \mathbf{A}\underline{u} \tag{4.16}$$

where

$$\underline{x} = \begin{bmatrix} \delta x_1 \\ \delta x_2 \\ \delta x_3 \\ \vdots \\ \delta x_N \\ \delta x_{N+1} \\ \vdots \\ \delta x_M \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} a_1 & 0 & \cdots & 0 \\ a_2 & a_1 & & 0 \\ a_3 & a_2 & & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_N & a_{N-1} & & a_1 \\ a_{N+1} & a_N & & a_2 \\ \vdots & \vdots & & \\ a_M & a_{M-1} & \cdots & a_{M-N-1} \end{bmatrix}$$

and

$$\underline{u} = \begin{bmatrix} \Delta u_1 \\ \Delta u_2 \\ \Delta u_3 \\ \vdots \\ \Delta u_N \end{bmatrix}$$

Given a reference trajectory R we can calculate the error for the future M steps using eqn. 4.17.

$$\underline{e} = R - X = \underline{x} \tag{4.17}$$

where X is the combined output of all the previous inputs. We wish to find the best input vector u that minimizes the criteria

$$\|\mathbf{A}\underline{u} - \underline{e}\|_2^2 \tag{4.18}$$

The least-squares solution to the problem is

$$\underline{u} = \mathbf{A}^{+}\underline{e} \tag{4.19}$$

Often the DMC controller is too aggressive and can be de-tuned using move suppression APT supports array variables making implementation of DMC easy. For implementation purposes we just wish to determine the next output from the controller. We just need the first row of $\mathbf{A}^{+}$ Thus given the step response we can calculate $\mathbf{A}^{+}$ and store the first row in the controller.

4.4 Auto Tuning

The intelligent SISO controller is capable to implementing two auto tuning methods. The ATV auto tuner [56] and the PRBS input-output system identification method [54]. The PRBS technique is used to determine a step response for DMC.

The Relay Feedback Auto-Tuner

Conventional PID tuning techniques used the Ziegler-Nichols (Z-N) ultimate tuning procedure [65]. The proportional only controller is used, whose gain is raised till the process undergoes sustained oscillations. The controller gain at which this occurs is the ultimate gain K_u and the period of oscillations obtained is the ultimate period, T_u. This technique induces large perturbations in the process. Åström and Hagglund [66] suggested a ATV (auto-tune variation) procedure that permits the determination of the ultimate gain and the ultimate period without introducing major perturbations in the process. A block diagram of the relay feedback tuner is shown in figure 4-8.

The ATV auto-tuner introduces a relay feedback of height h. The controller output m is raised h above the steady state value. Once the process output crosses the set-point the controller output is reduced to a value h below the steady state value. This introduces a limit cycle in the process. The ultimate gain of the process is given by equation 4.20 and the ultimate period is the period of the limit cycle. A

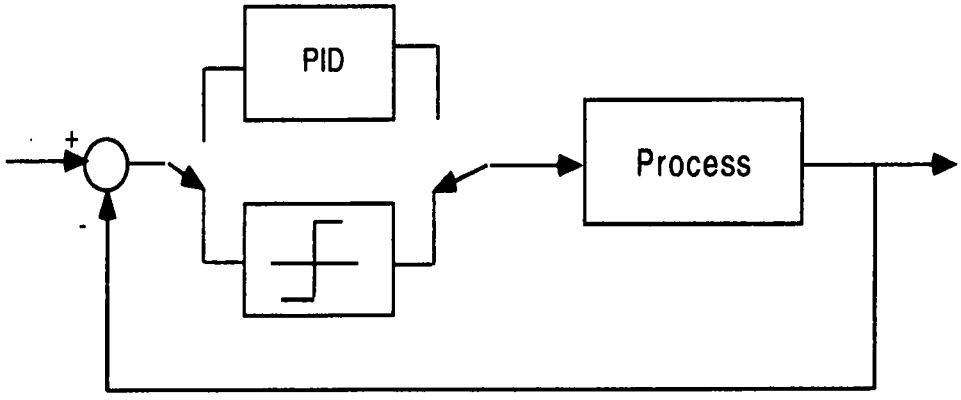


Figure 4-8: Block diagram of the ATV auto-tuner.

typical ATV run is shown in figure 4-9.

$$K_u = \frac{4h}{\pi a} \tag{4.20}$$

Traditional Ziegler-Nichols or modified Z-N tuning rules can now be applied to determine tuning constants. In the presence of noise a relay feedback with a hysteresis width ε is used.

The ultimate gain K_u' can now be calculated by equation 4.21.

$$K_u' = \frac{4h}{\pi\sqrt{a^2 - \varepsilon^2}} \tag{4.21}$$

The ATV auto-tuner is often used as a pre-tuner for more sophisticated tuning algorithms.

4.4.1 The PRBS Input-Output Modeler

Control algorithms like DMC and QDMC require a step response model for the process. The heat exchanger under direct control provides a different model for step up and step down responses about some steady state operating point. Often step responses in the same direction are not repeatable. Correlation based input-output

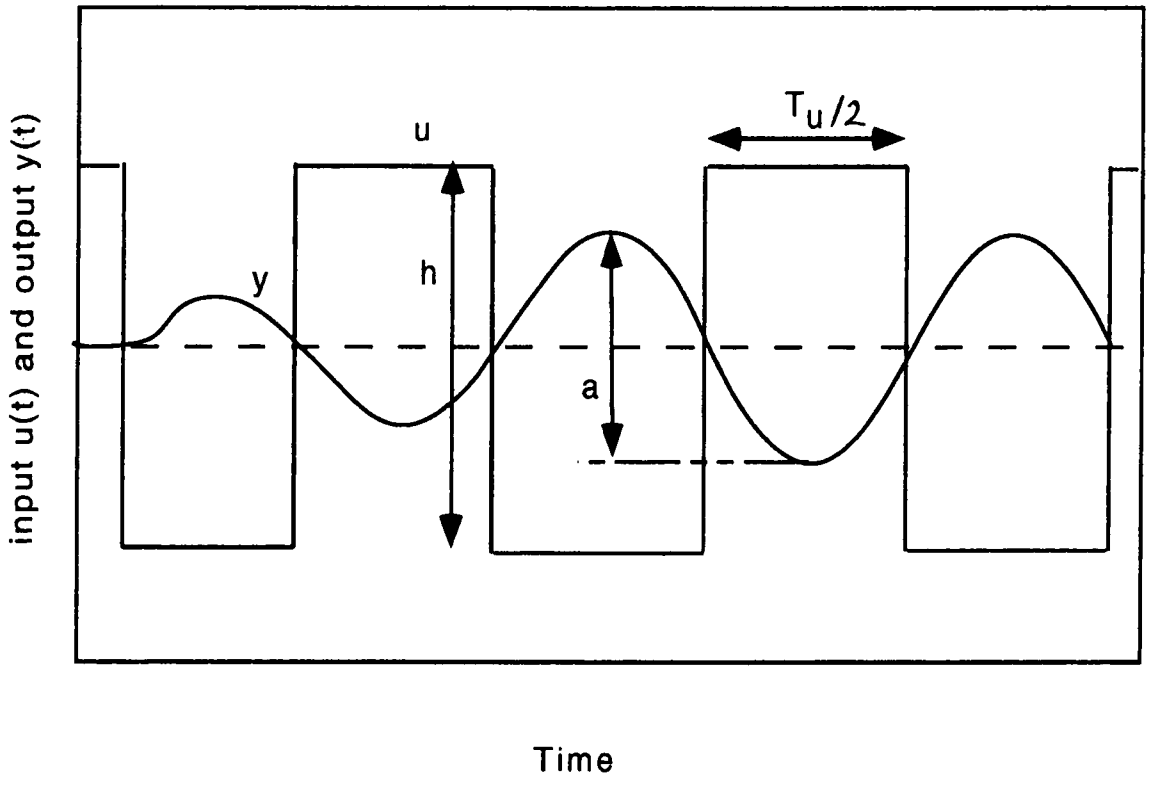


Figure 4-9: Typical ATV run.

modeling techniques can provide a best fit model that may be used to generate a step response for DMC and QDMC. The Pseudo Random binary sequence (PRBS) has a auto-correlation function that is very close to the delta function of ideal white noise [46].

The PRBS signal has two levels with the shortest change occurring at Δt, the PRBS bandwidth. The PRBS sequence can be generated by a n-stage shift register. The sequence has the largest period of $N = (2^n - 1)$ after which it repeats itself. For the analysis to be meaningful it is necessary that the excitation period for the test $\tau = N\Delta t$ be greater than the settling time of the system. Hang [54] provides with some suitable values for N for different signal to noise ratios.

The process model can be extracted in several ways. If a difference equation model of the form in equation 4.22 is assumed.

$$y_i' = b_1 u_{i-1-N} + b_2 u_{i-2-N} + \cdots + b_p u_{i-P-N} - (a_i y_{i-1} + a_2 y_{i-2} + \cdots + a_Q y_{i-Q}) \quad (4.22)$$

where y_i' is a estimate of the output of the process. There are $P + Q$ unknowns If we have NR input-output samples and $NR > (P + Q)$. Then we can determine the coefficients $\underline{\theta} = [b_1\ b_2\ \cdots b_p\ a_1\ a_2\ a_3 \cdots a_Q]$, using least squares, that minimize equation 4.23.

$$\left\| \mathbf{A}\underline{\theta} - \underline{y} \right\|_2^2 \quad (4.23)$$

where

$$\mathbf{A} = \begin{bmatrix} y_{1,n-1} & y_{1,n-2} & \cdots & y_{1,n-Q} & u_{1,n-N} & \cdots & u_{1,n-N-P} \\ y_{2,n-1} & y_{2,n-2} & & y_{2,n-Q} & u_{2,n-N} & & u_{2,n-N-P} \\ \vdots & \vdots & & & & & \\ y_{NR,n-1} & y_{NR,n-2} & \cdots & y_{NR,n-Q} & u_{NR,n-N} & \cdots & u_{NR,n-N-P} \end{bmatrix}$$

and

$$\underline{y} = \begin{bmatrix} y_{1,n} \\ y_{2,n} \\ \vdots \\ y_{NR,n} \end{bmatrix}$$

Although the PRBS sequence and its implementation are done on the PLC, the data logging and model estimation is performed on the 386/ATM module using Matlab.

4.4.2 Intelligent Autotuning and System-Identification

Measured and unmeasured disturbances affecting the system during autotuning can lead to modeling errors. Detecting unmeasured disturbances can be difficult during auto-tuning using commercial auto-tuners as they only monitor the controlled variable. However on a PLC, information on other measured process variables that affect

the system is often available. At a primary level, if a measured disturbance is detected during auto-tuning, the auto-tuning can be disabled. If auto-tuning is successful then the PID parameters are updated in the database for the active controller. If the PRBS signal length, N, is large and the duration of disturbance is small the effects of the disturbance can be reduced or neglected. Some commercial controllers account for the disturbance[6]. The intelligent SISO controller uses the system cultivation tools to determine if a measured disturbance has affected the system during auto-tuning and disables the auto-tuner. Alarms are set up on the operator interface to inform the operator of the actions taken. The disturbance flag prevents the use of the auto-tuner till the effects of the disturbance have passed.

4.5 Evaluating Controller Performance

An intelligent controller must be capable of evaluating its own performance. Typical performance measures for SISO controllers are Integral absolute error (IAE), and Integral square error (ISE) shown in equations 4.24 and 4.25, for setpoint changes and disturbance rejection. Setpoint changes can be used to evaluate the controller performance, as it represents a permanent change in the controller setpoint. Poorly tuned controllers can either lead to oscillatory trends or take a long time to reach the setpoint. The system cultivation tools developed are used to determine controller performance and set alarms or take automatic actions.

$$IAE = \int_0^\infty |e(t)|\, dt \tag{4.24}$$

$$ISE = \int_0^\infty e^2 dt \tag{4.25}$$

Deviation decay terms are evaluated for the set-point and the controller error (PV-SP). Closed loop response for set-point changes can be characterized in several ways

like, percent overshoot and settling time, decay ratio, etc. A set-point change will initiate performance evaluation. The system cultivation tools check for oscillations. If oscillations exist the controller is tightly tuned. Controller performance is not evaluated when the process is undergoing measured disturbances.

If the controller detects oscillations in the controlled variable it checks to see if any of the process variables that affect the controlled variables are undergoing oscillations. If not, then the controller determines that the oscillations are caused by its performance. If an advanced controller like DMC is operational an is unstable it is disabled, returning control to the PID controller. If the PID or the IPID algorithm is active the process gain is reduced by 33 percent and the oscillation detector is reset. If the oscillations persist the procedure is repeated. Whenever the change is made an alarm is set and the operator is informed of the change. During the operation of the IPID controller if the operating region has changed, the oscillator detector is reset. The current implementation does not make any changes to the integral time constant of the controller.

Most of the implementation of the intelligent controller is done on the PLC. A few procedures are implemented on the 386/ATM module. Whenever the services of the external computer are required a handshake procedure is employed. The PLC sets a boolean variable, which is reset by the 386 computer, every sample time. If the PLC detects that the boolean flag has not been reset for more than 3 sample times it assumes that the external computer is inoperative and disables the functional procedures provided by the external computer. Next we describe the results of the intelligent controller.

Chapter 5

Results and Discussion

In this chapter we present the performance of the intelligent controller. The process that the intelligent controller sees is the heat exchanger with the cascade pressure loop. The inner loop is manually tuned. The intelligent controller is capable of implementing several control algorithms, two system identification techniques and provide stable operation. The controller is capable of evaluating its own performance (in a qualitative sense) and punish itself. Unpredictable results from the PRBS system identification technique lead to the detection of a improperly installed steam trap.

5.1 The Intelligent SISO controller

A block diagram overview of the intelligent SISO controller is show in figure 5-1. The controller can be functionally divided into the System Cultivation Daemon block, the database, the system identification block and the controller block.

5.1.1 The Database

The database for the intelligent SISO controller is implemented as a recipe on the controller. The recipe stores the following information:

- The overall and the partitioned process models and the partition information

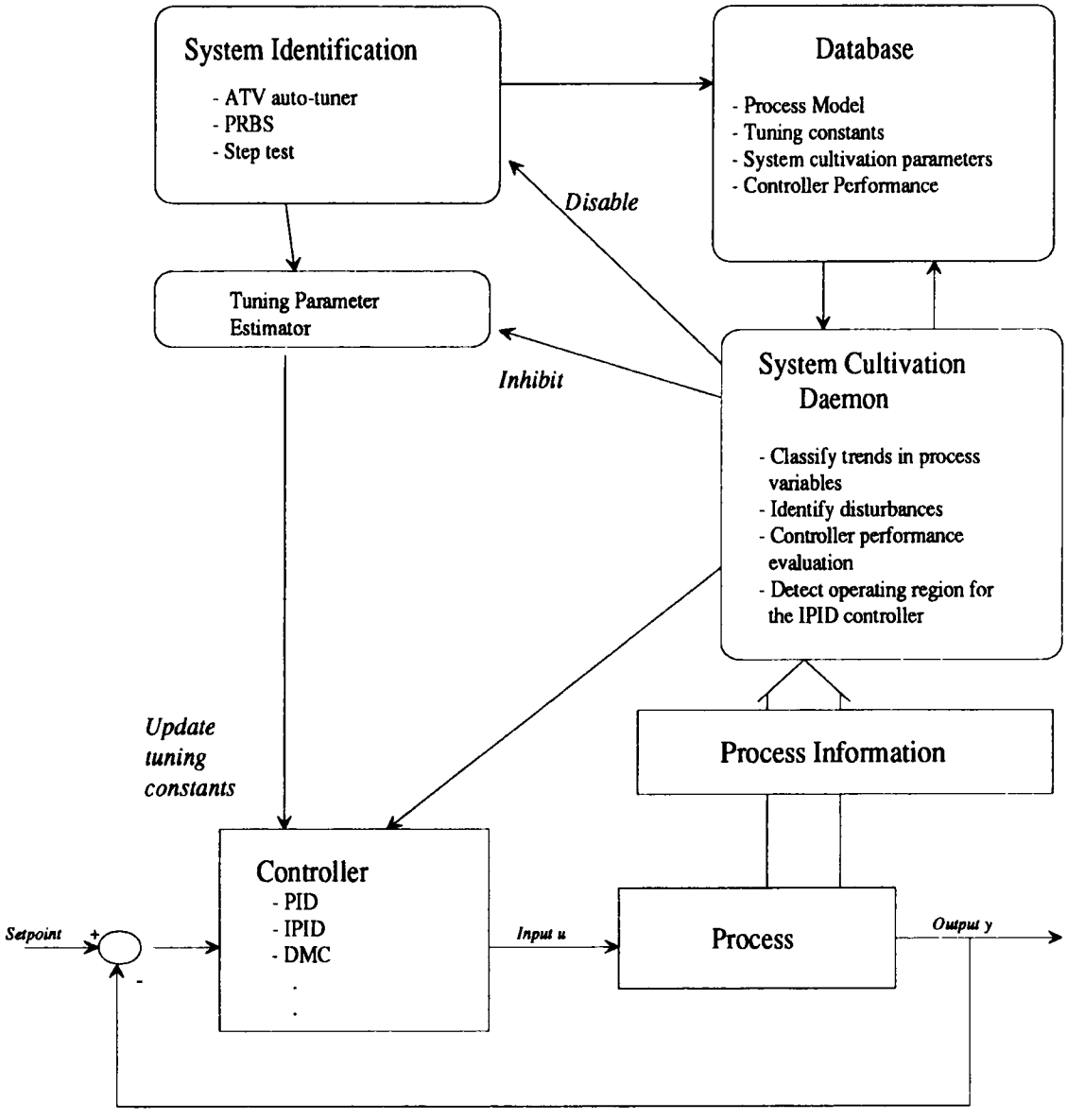


Figure 5-1: Block diagram of the ISISO controller.

- Tuning constants for all the PID controllers.
- Tuning constants for the system cultivation tools.
- A time tagged history of the last five tuning constants implemented for each of the PID controllers.
- A time tagged history of valid ISE and IAE values for Set-point Changes.

At startup, the recipe is loaded from the database on the operator interface and displayed to the user who can changes any parameters. Upon approval the values are downloaded to the controller. The recipe is designed to contain all the relevant information for the intelligent controller. Every time the intelligent controller updates the tuning parameters automatically it is logged in the recipe with a time and date tag and the tuning method used. Only the previous five values are stored. This permits the user to manually select values and if necessary go back to a historical database to verify the controllers performance. Every time the intelligent controller performs a performance evaluation a time tagged log is kept with the ISE, IAE, the control algorithm being used and its tuning parameters.

5.1.2 The Controller Block

Different control algorithms can implemented through the PID controller available on the PLC. The PID, IPID and DMC algorithms are available. The PID controller on the PLC has to be in the auto mode for any control algorithm to adjust the manipulated variable. The DMC controller requires information from an external source, and is disabled at start-up and only available on user verification. The operator always has priority on disabling the controller from the operator interface. A mechanism is provided to add additional control algorithms with little programming effort.

5.1.3 System Identification Block

The system identification block provides the use of the ATV auto-tuner and the PRBS signal. A step response can be obtained from the operator interface through the standard controller display page. The identification blocks are implemented in a form whereby they can be activated or deactivated by from the operator interface.

5.1.4 The System Cultivation Daemon

This block provides the "intelligence" for the ISISO controller. This daemon has a collection of *CFB's* that implement the system cultivation tools. This block manages process information and the information generated by the system cultivation tools as an expert system to provide intelligent control.

Deviation decay terms and the error decay terms are evaluate for the following:

- Water flow rate.
- Inlet water temperature
- Outlet water temperature
- The controller error *(SP-PV)*.
- The steam pressure (this is used only when steam pressure starts to go below atmospheric).

The last two are used in evaluating controller performance and are disabled when the controller is in manual. The others are computed all the time. Information from these tools is used by an *IF-THEN-ELSE* type rule base. Information from the water flow sensor and the inlet water temperature sensor form the *disturbance* flag.

Table 5.1: Performance of the conventional cascade PID controller.

Event	ISE	IAE
Step up change in SP (100°F to 110° F)	93.2	135.0
Step down change in SP (100°F to 110°F)	102	145.9
Disturbance rejection for ΔF_w=3 gpm	45.8	78.0

5.2 Controller Algorithms

5.2.1 Conventional PID controller

Conventional direct PID control gives poor behavior as the process exhibits different responses for step up and step down on the manipulated variable.Cascade control provides a slightly more sluggish but stable process model. The cascade strategy used controls the steam pressure in the inner loop, with the master providing the steam pressure set-point to the slave controller. The ISE and IAE values are shown in table 5.1.

5.2.2 Intelligent PID controller

The intelligent PID (IPID) controller partitions the operating space into 4 regions as shown in figure 4-6. Only the cascade control strategy is used for the IPID controller. The process models for the different operating regions are shown in table 5.2. The disturbance models at water flow rates of 12 gpm and 18 gpm are shown in table 5.3.

Table 5.2: Table of process models for the different operating regions.

Operating Region	Process Model
I	$\frac{0.79e^{-5s}}{1+30s}$
II	$\frac{0.52e^{-5s}}{1+37s}$
III	$\frac{0.72e^{-5s}}{1+32s}$
IV	$\frac{0.50e^{-5s}}{1+36s}$

Table 5.3: Table of disturbance models for different water flow rates.

Water Flow	
12 gpm	18 gpm
$\frac{0.5e^{-5s}}{1+31s}$	$\frac{0.42e^{-5s}}{1+38s}$

The inner pressure loops are tuned manually as the process response is too fast. The tuning parameters for the inner loops are shown in table 5.4. The outer loop tuning constants are determined by operating the process around the center of each operating region and minimizing the ISE values for a step change in the set point. They are listed in table 5.5. The IPID controller performs well in the individual operating regions. A summary of its performance is presented in table 5.6.

The system cultivation tools determine the selection and switching of the operating region and the PID controller tuning parameters. The deviation decay terms, the deviation decay rate terms and the error decay terms are tuned using heuristics and

Table 5.4: Table of tuning constants for inner pressure loop.

Operating Region	$\mathbf{K}_c$	$\mathbf{T}_i$ (min)
I	2.5	0.1
II	1.75	0.05
III	2.4	0.1
IV	1.70	0.05

Table 5.5: Table of tuning constants for outer temperature loop

Operating Region	$\mathbf{K}_c$	$\mathbf{T}_i$ (min)
I	6.0	0.3
II	8.5	0.3
III	6.1	0.3
IV	8.5	0.3

Table 5.6: Performance of the IPID controller in the different operating regions.

Event	**I**		**II**		**III**		**IV**	
	ISE	IAE	ISE	IAE	ISE	IAE	ISE	IAE
$\Delta SP = +10°F$	82.6	120.0	85.7	117.2	83.0	113.4	85.8	116.1
$\Delta SP = -10°F$	83.5	121.8	82.3	121.2	83.8	120.3	89.0	123.2
$\Delta W_f = 4\ gpm$	19.7	114.8	11.26	92.7	23.6	123.3	11.8	96.3

Table 5.7: Table of System Cultivation parameters for the different process variables.

Process Variable	Parameter	Value	Threshold
Water flow rate	Deviation deay term (δ_1)	0.9	2.0
	Deviation decay rate term (δ_2)	0.95	0.2
	Error decay term (δ)	0.83	0.15
Inlet water temp.	Deviation deay term (δ_1)	0.9	0.5
	Deviation decay rate term (δ_2)	0.95	0.1
	Error decay term (δ)	0.82	0.10
Outlet Water temp.	Deviation decay term (δ_1)	0.9	0.5
	Deviation decay rate term (δ_2)	0.96	0.1
	Error decay term (δ)	0.815	0.12

typical values used for this research are shown in table 5.7.

Figure 5-2 shows the performance of the IPID controller as a change in the water flow rate moves the operation from region II to I. The tuning parameters used in operating region II provide a tightly tuned controller for operating region I. The outlet temperature starts oscillating as the system cultivation daemon has still not determined that the transition is permanent. The dde_term and ddr_term plotted are for the water flow rate. The $\dot{d}_e(t)$ term goes below the threshold of 0.3 around 100 seconds and the system cultivation daemon updates the operating region, updates the tuning constants and resets the deviation decay term and the deviation decay rate term. The outlet temperature quickly stabilizes with the new tuning constants.

Figure 5-3 shows the transition form operating region I to II. As this transition is from a process gain of high to low the transition is stable. Note the jump in the outlet water temperature as the tuning constants are updated. Figure 5-4 shows the response to a transient change in the water flow rate. Note the selection of the tuning parameters for the system cultivation tools is such that the $d_e(t)$ term recovers quickly than the $\dot{d}_e(t)$ term for a transient that lasts no longer than 1.5 times the

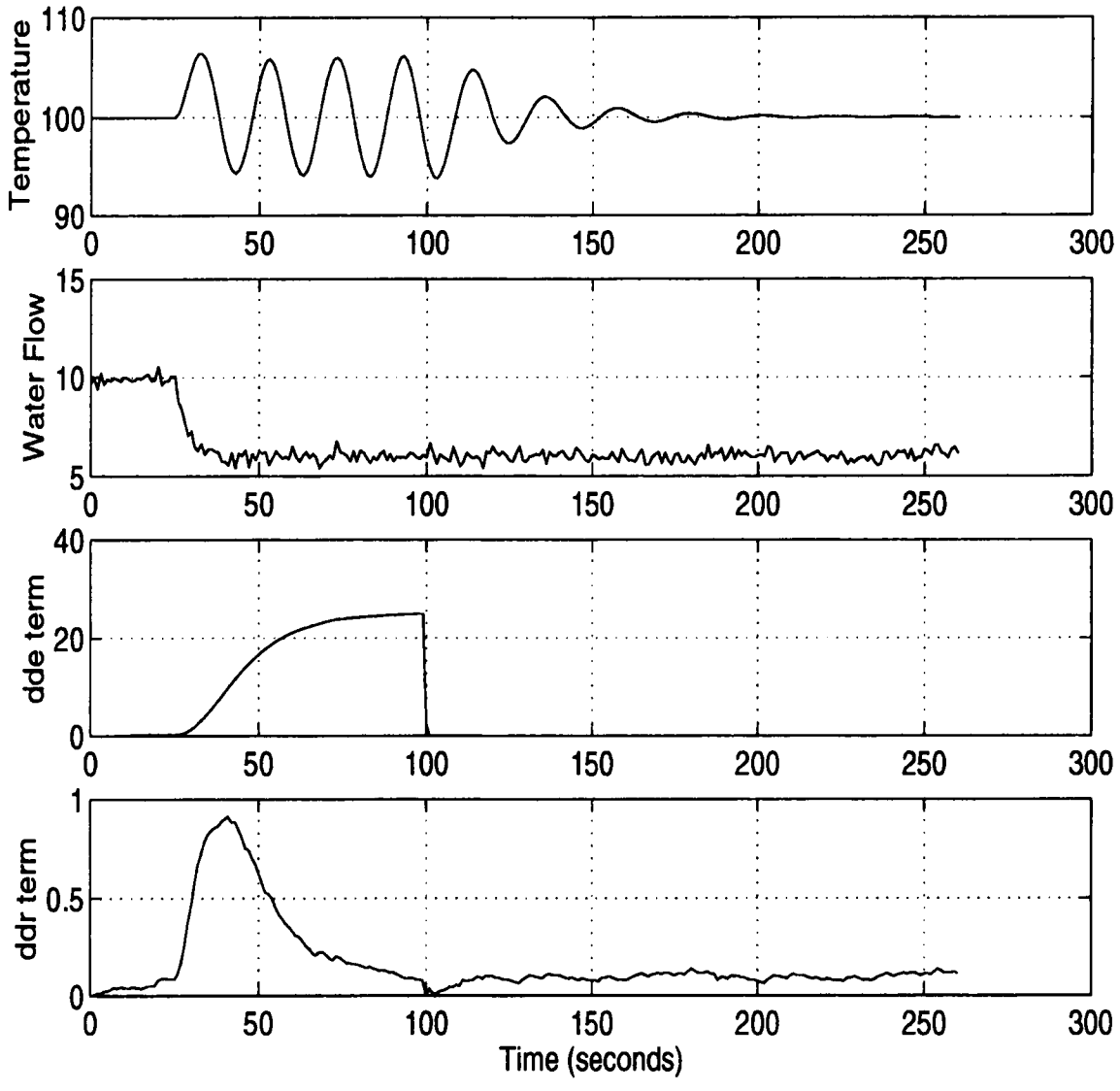


Figure 5-2: Performance of the IPID controller for a change in the operating region from II to I. The *dde* and *ddr* terms plotted are for the water flow rate.

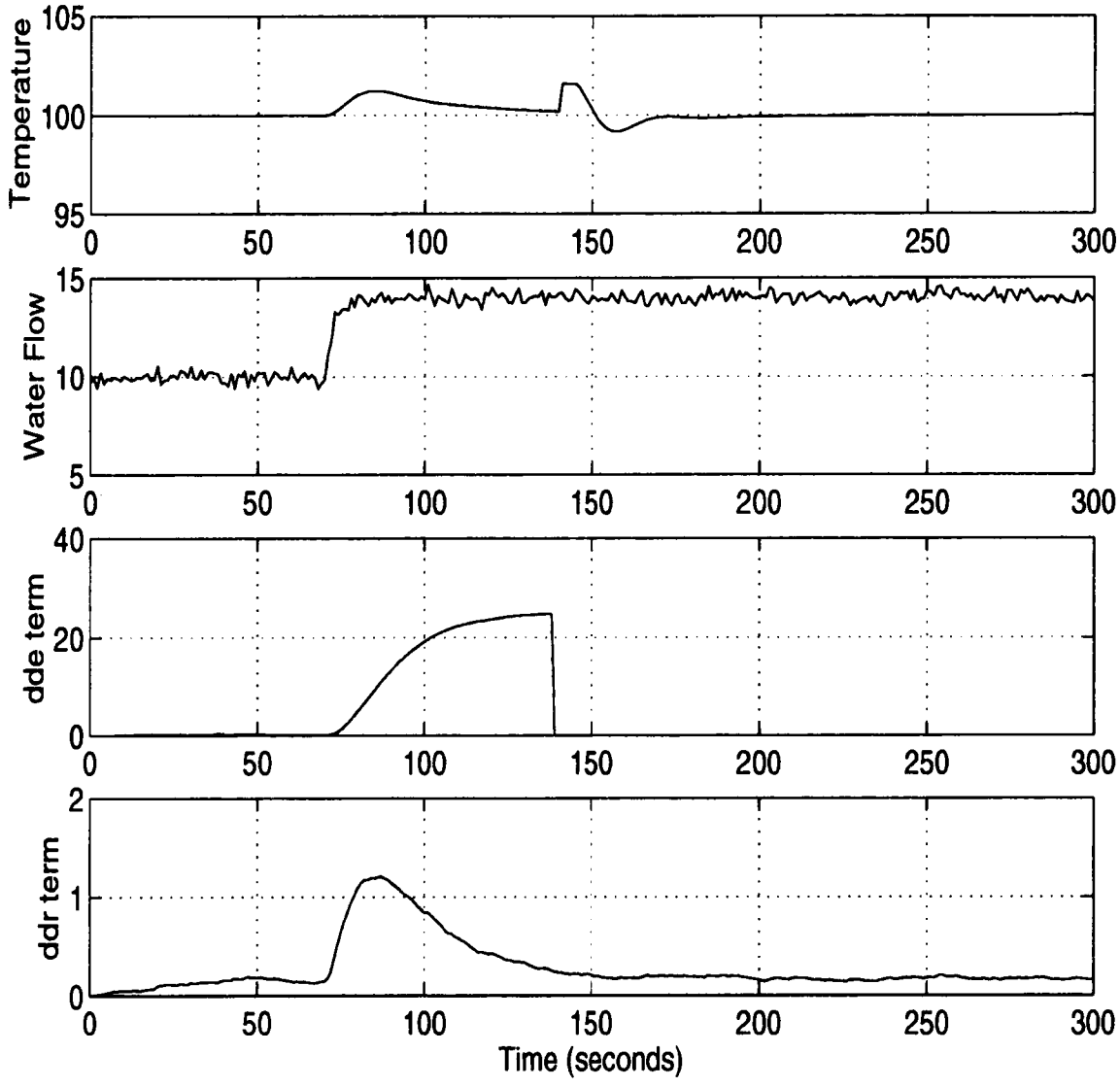


Figure 5-3: Performance of the IPID controller for a change in the operating region from I to II. The *dde* and *ddr* terms plotted are for the water flow rate.

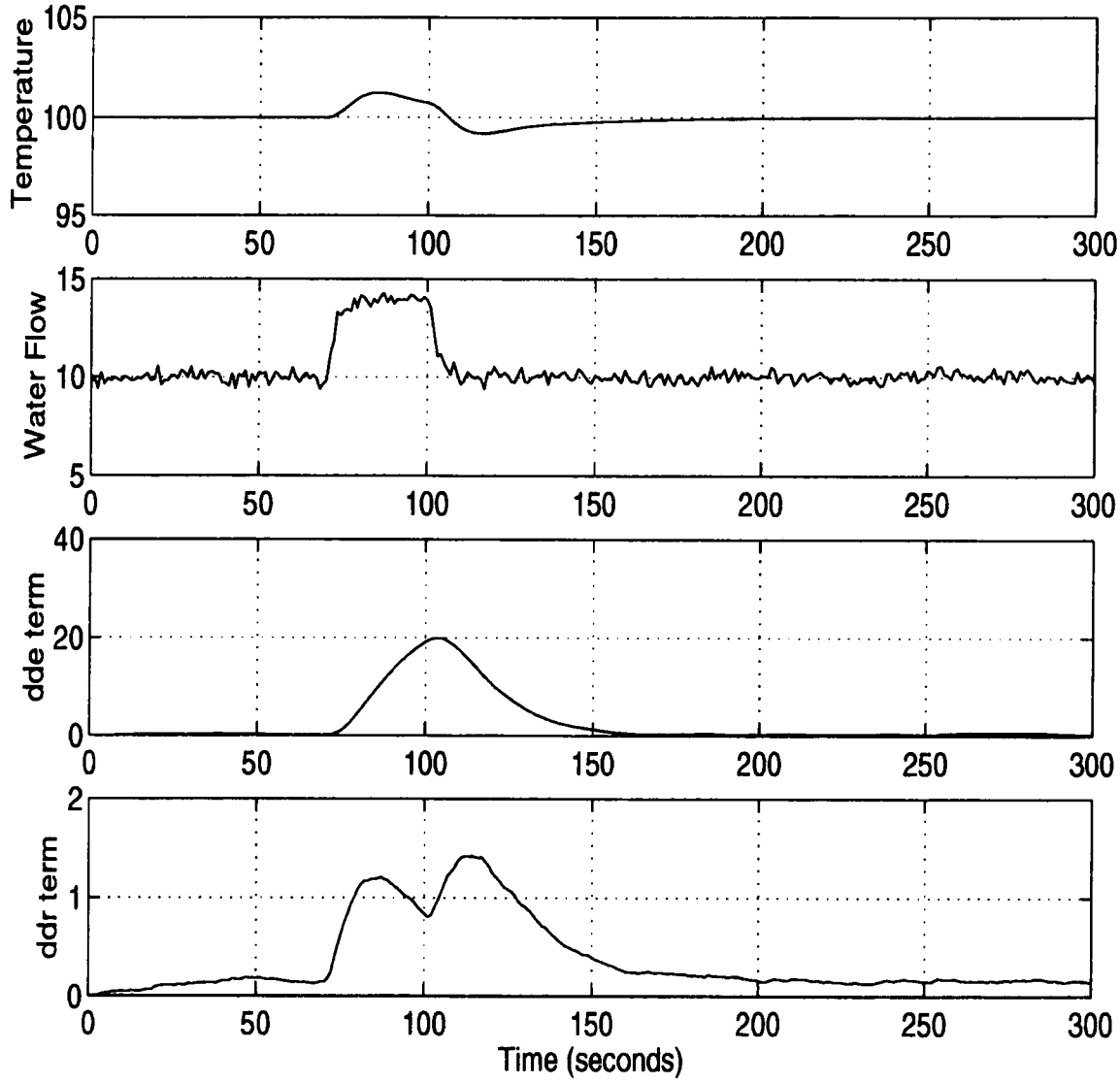


Figure 5-4: Performance of the IPID controller for a transient change in the water flow rate. The *dde* and *ddr* terms plotted are for the water flow rate.

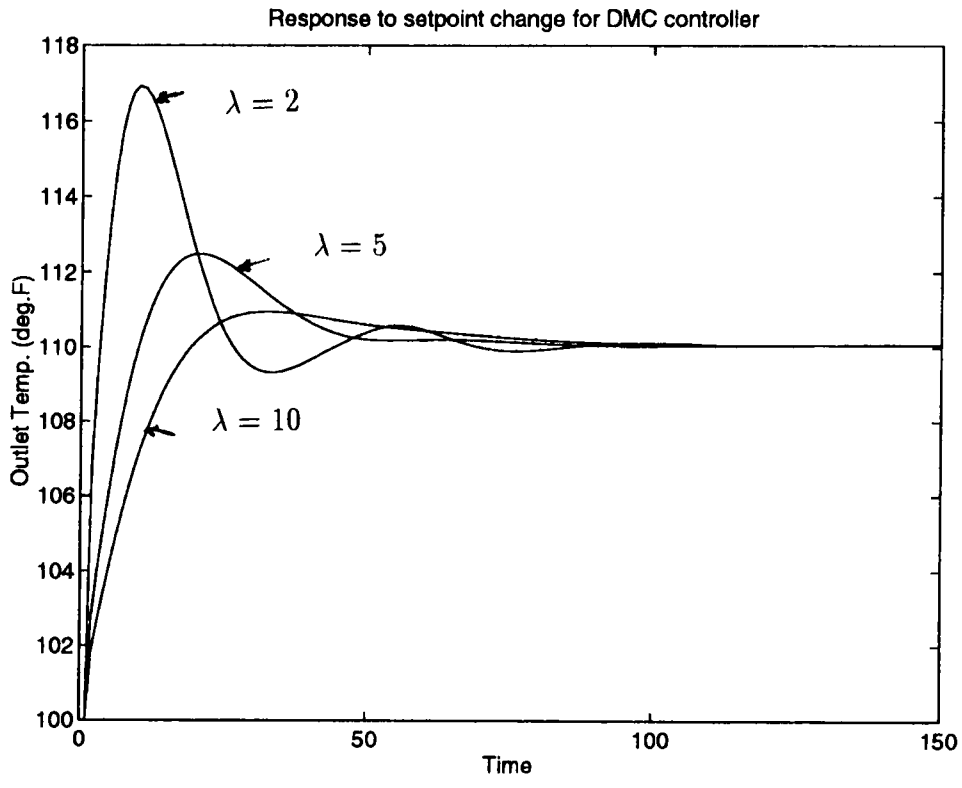


Figure 5-5: Performance of the DMC controller for set-point change for different move suppressions (λ =2, 5, and 10).

time constant of the system. This choice is heuristic This signals a transient change and no updating of the base case or the operating region is performed.

Boundary operation of the heat exchanger under IPID control depends on the previous path to the boundary. If the process has moved from II towards I then there may be some poor performance. However, if it transitioned from I to II it may have a slightly sluggish response. If the operation at the boundary tends to be oscillatory the controller performance evaluation techniques used in the system cultivation daemon will correct for it.

5.2.3 DMC Controller

The DMC controller is based on a linear step response model and performs poorly on the heat exchanger. Figure 5-5 shows the performance of the DMC controller for

a set-point change from 100 to 110 $^{\circ}F$ for different values of move suppression. The DMC controller is unstable without any move suppression. The DMC controller is implemented as an exercise in implementing a control algorithm that is computed externally to the PID controller on the PLC. Its poor performance is utilized later to show the fault tolerant performance of the intelligent controller. The ISE and IAE values for the DMC controller are is summarized in table 5.8.

Table 5.8: Table of ISE and IAE values for set-point change from 100 $^{\circ}F$ to 110 $^{\circ}F$ for DMC controller for various values of move suppression.

Move Suppression	**ISE**	**IAE**
$\lambda = 2$	596.6	141.4
$\lambda = 5$	340.0	117.3
$\lambda = 10$	422.0	122.8

5.3 Intelligent Auto-tuning

5.3.1 Relay Feedback Auto-Tuner

The relay feedback auto-tuner can be implemented any time on the process. It is implemented through the PID controller. The PID controller disables its PID calculations during the tuning run. Figure 5-6 shows the operation of the ATV auto-tuner at a water flow rate of 13 gpm. Before starting the tuning procedure the user is asked to select the relay height an the number of limit cycles to span. Usually the first cycle is ignored. Timers on the PLC are used to determine the limit cycle. The maximum and the minimum values of the process variables are stored and a quick check on the variation of these values is done. If they lie within 15% of the mean maximum and minimum, respectively, then they are accepted. If not, a alarm is signaled to the user. If the data collected is acceptable then the tuning parameters are computed using Zeigler-Nichols rules and implemented after

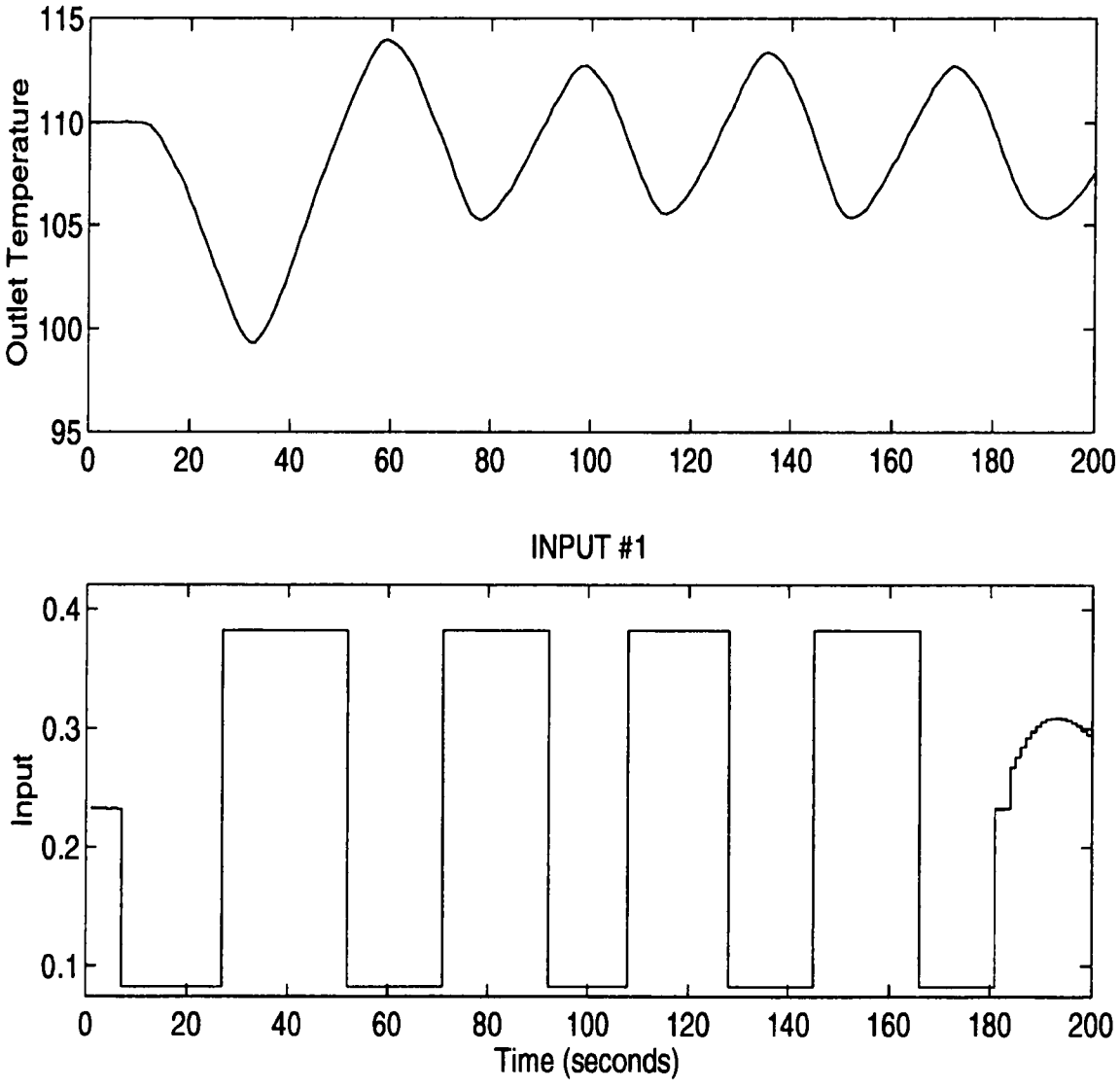


Figure 5-6: Operation of the relay feedback (ATV) auto tuner.

Table 5.9: Results obtained from ATV tuning.

Opr.Rgn.	K_u	T_u (min)	K_c	T_i (min)
I	15.9	0.667	6.4	0.533
II	20.5	0.7	8.2	0.56
III	16.2	0.583	6.5	0.467
IV	22.0	0.633	8.8	0.51

user verification. If the tuning rules are implemented the knowledge database is updated for the appropriate controller type and the operating region. Table 5.9 gives the ultimate gain K_u, ultimate period T_u, and the calculated tuning constants, K_c for the controller. The ATV auto-tuner provides pre-tuning information for the PRBS technique of system identification.

Disturbances affecting the process during auto-tuning can lead to poor modeling and thus poor controller performance. The system cultivation daemon provides intelligent auto-tuning capabilities by disabling the auto-tuner. This is shown in figure 5-7. The disturb flag turns true and disables the ATV tuner and keeps it disabled till the system cultivation tools tell it that the effects of the disturbance have passed. An informational alarm is activated on the operator interface indicating the action taken.

5.3.2 PRBS Tuner

The PRBS tuner is implemented through the operator interface. The user is asked to choose the step size, the sample time and the number of shift registers to be used to generate the PRBS signal. If the sample time chosen is less than 5 seconds a warning is issued indicating that TISTAR cannot log the data faster than 5 seconds and that the user use the 386/ATM module to collect the data. A sample PRBS run on the heat exchanger is shown in figure 5-8.

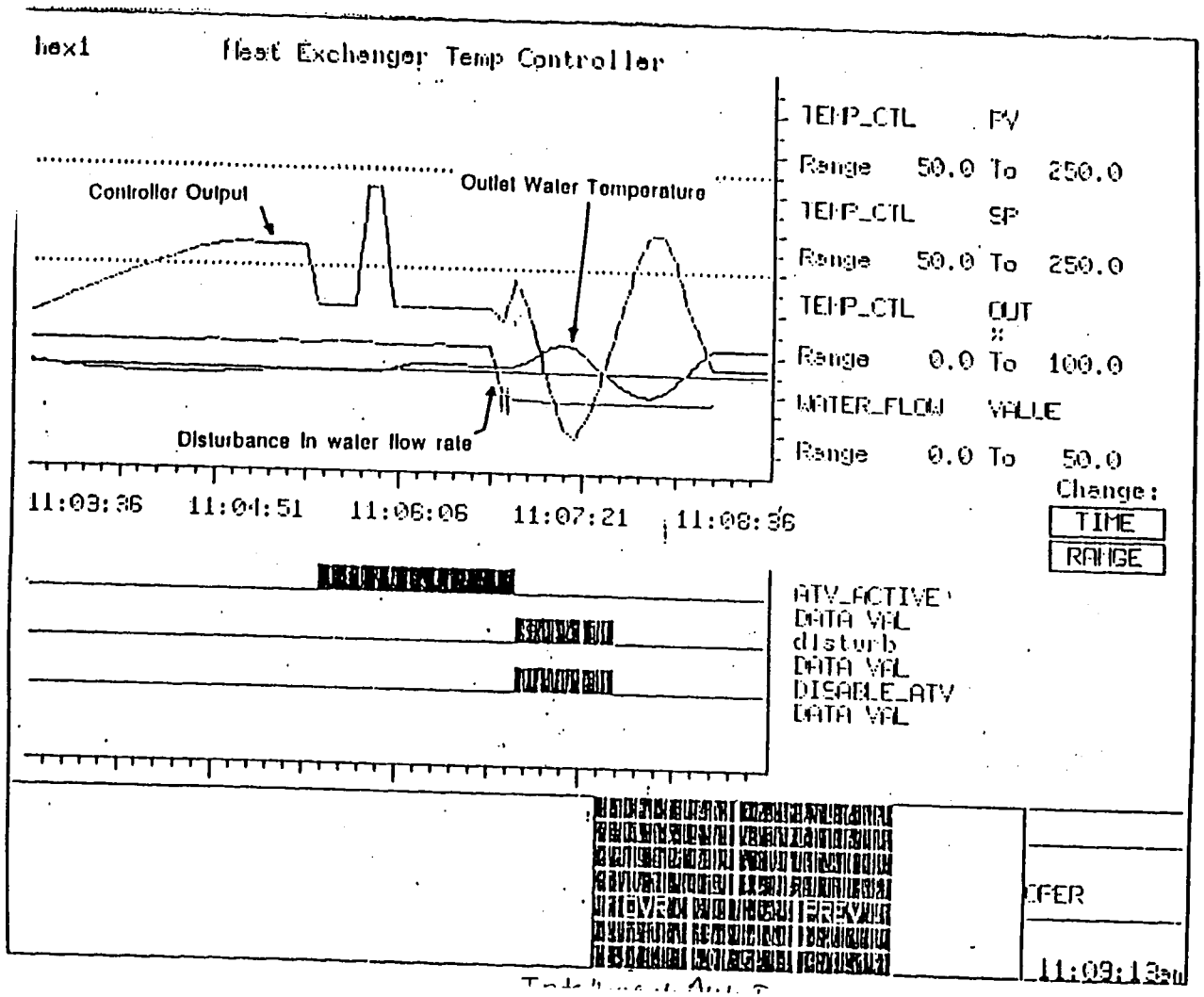


Figure 5-7: Intelligent Auto-tuning

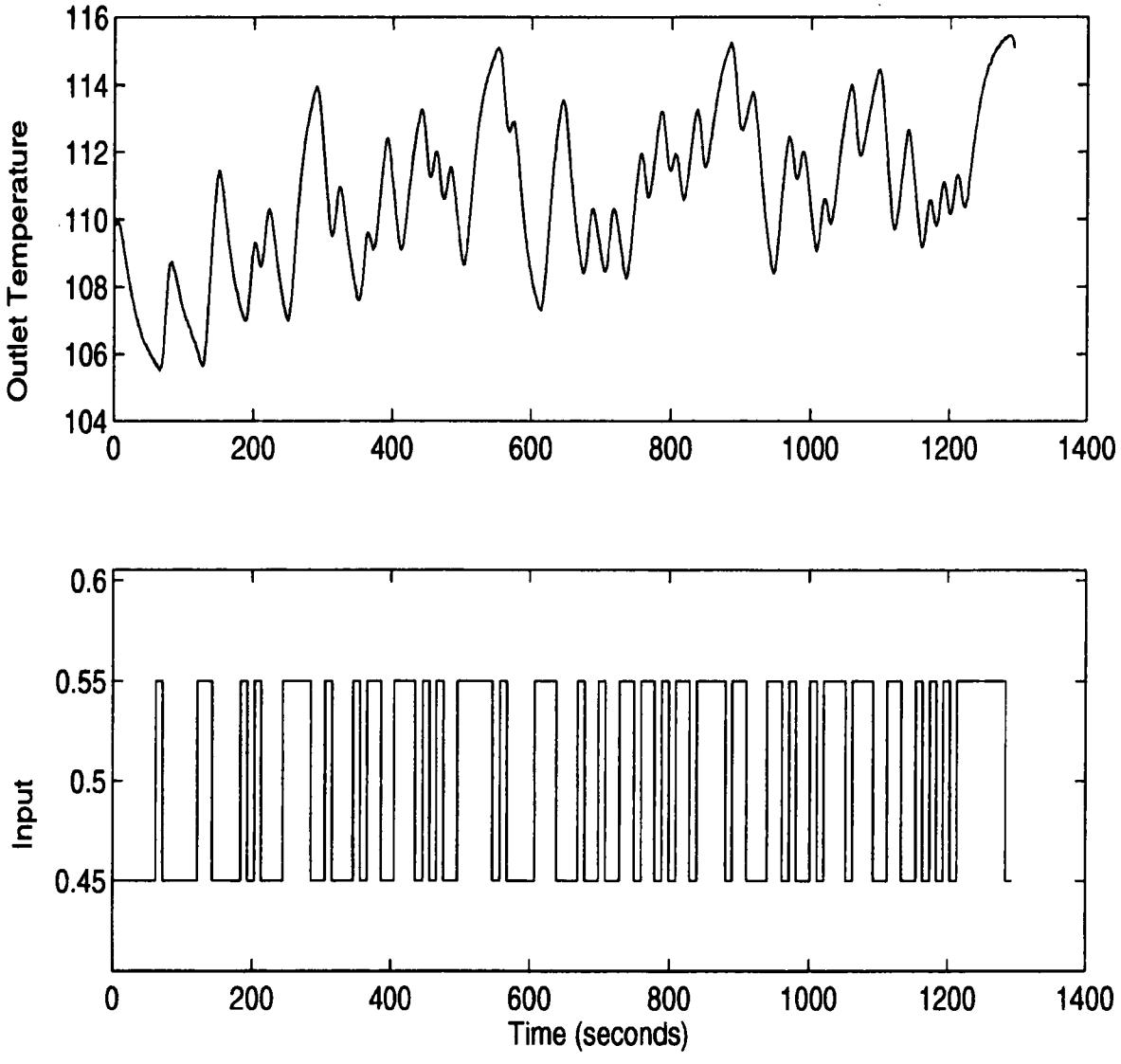


Figure 5-8: Typical PRBS Run at water flow rate of 16 *gpm*.

The data collected from the PRBS run is analyzed off-line on a computer using MATLAB. The sample time for the PRBS signal, Δt,was chosen to be 10 seconds with a PRBS signal using 7 shift registers. This allowed both, the impulse response, and the difference equation model techniques to be used for analysis. Before analyzing the data a Bode plot is generated to check that the frequency response does not break up before the frequency of interest (τ_p). The bode plot for the data shown in figure 5-8 is shown in figure 5-9. The impulse response technique gave poor results as Δt required was below 5 seconds and the response time of the control valve is a few seconds. The analysis showed that the impulse responses gave a larger gain for the process model. This is due to the fact that the step response is obtained by getting the cumulative sum of the impulse response and as the estimated impulse response does not go to zero, it adds to the cumulative sum.

Different orders of the difference equation model were evaluated. The difference equation model can be represented in the form shown in eqn 5.1.

$$A(q)y(t) = B(q)u(t - nk) + e(t) \tag{5.1}$$

We can now select different orders for the polynomials A and B and different values for the delay n. Different models were tried. The estimated model is then simulated with the PRBS input and compared with the actual output. The model providing the lowest sum of square errors is chosen. The selected difference equation is given in eqn. 5.2. The comparison of the process output and the model output is shown in figure 5-10. A step response generated by the model is shown in figure 5-11.

$$y_i = 1.7438y_{i-1} - 0.7515y_{i-2} + 0.0006u_{i-5} - 0.0037u_{i-6} \tag{5.2}$$

The results from the PRBS system identification using the difference equation model are shown in table 5.10. Using the difference equation model to perform system identification provides better results, except at the low water flow rates. Unrepeatable

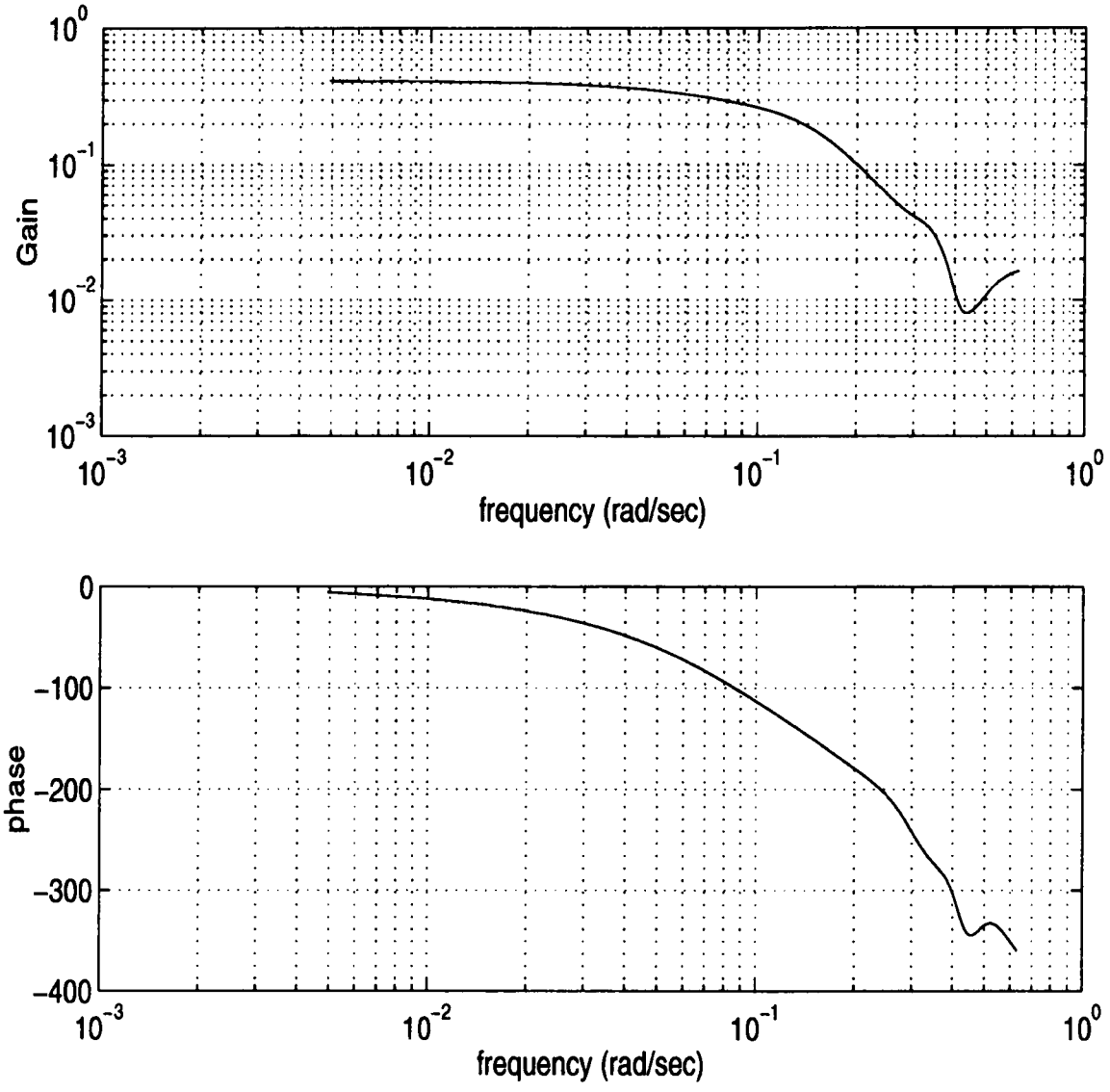


Figure 5-9: Bode Plot for PRBS run in shown in the previous figure.

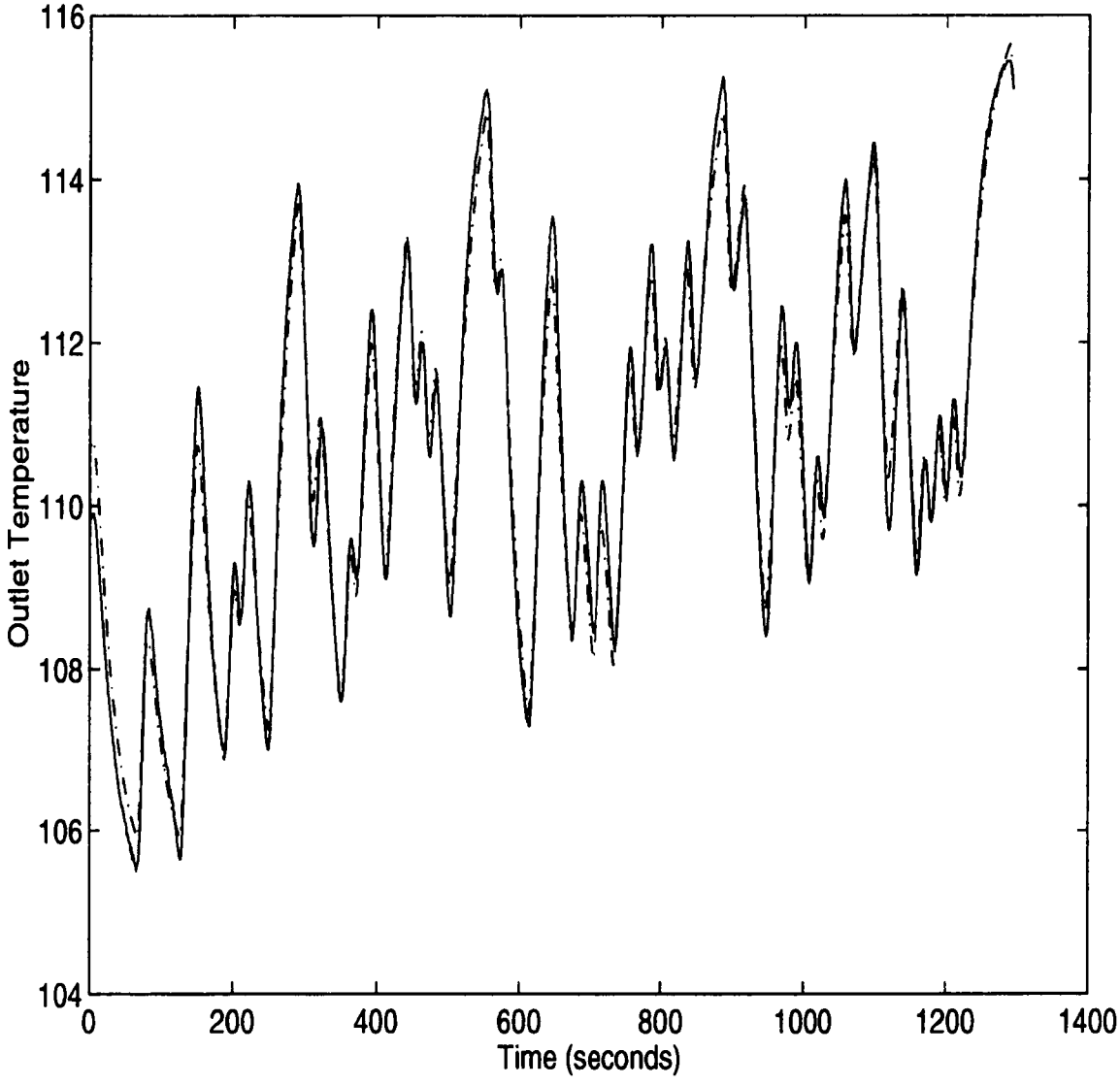


Figure 5-10: Comparison of PRBS model with actual process run. Dashed line is the model and the solid line is the actual process output.

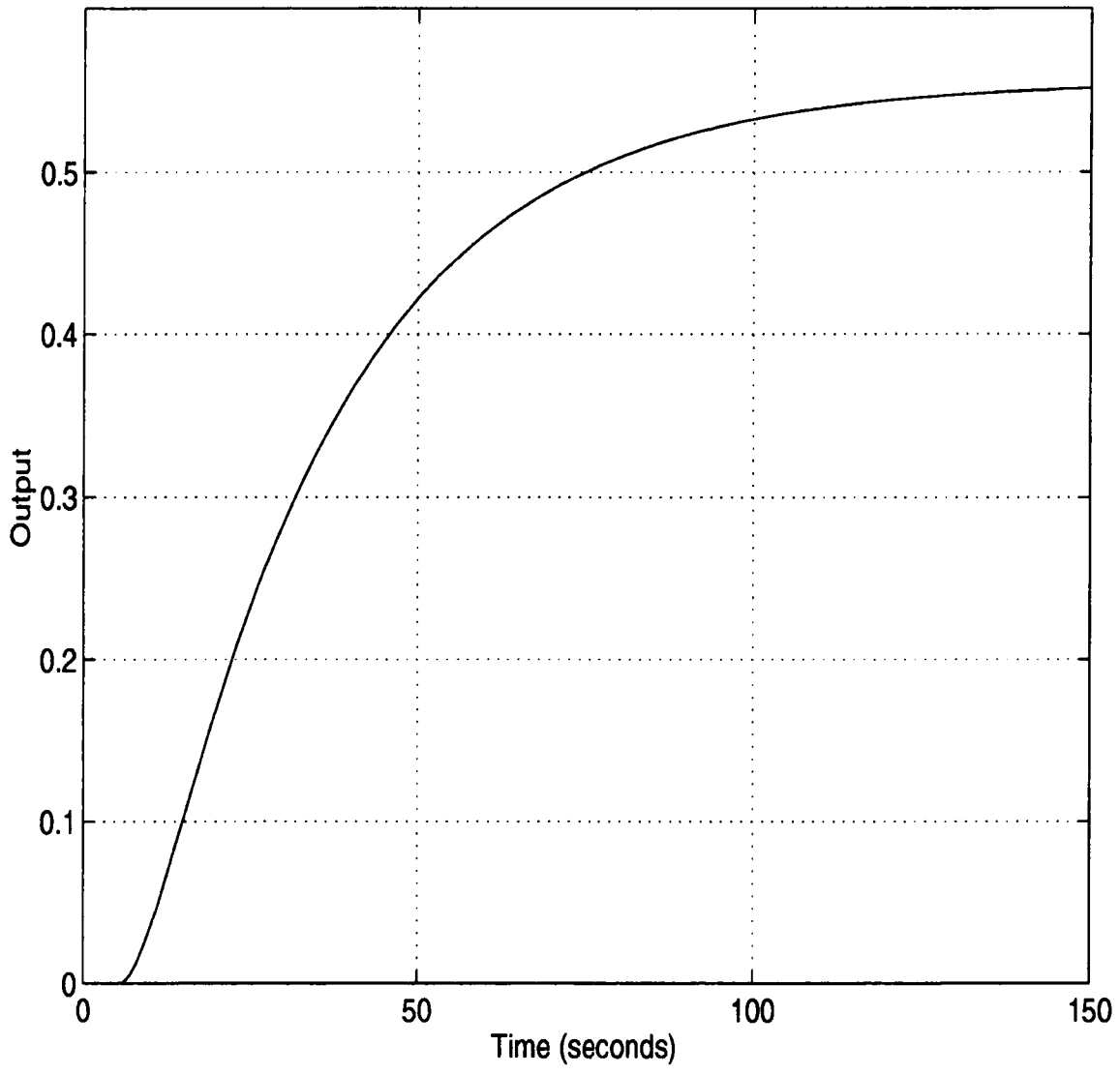


Figure 5-11: Step Response generated from model estimated using PRBS input. This is used for the DMC controller.

Table 5.10: Table of different order models and their performance

Model Order			
Numerator	Denominator	Delay	SSE
1	2	5	32.40
1	2	6	42.42
2	2	5	10.3
2	3	5	12.21
3	2	5	12.05

results obtained which could not be initially explained. This lead to the detection of an improperly installed steam trap. The analysis is provided later in this chapter. A typical run at a water flow rate of 12 *gpm* is shown in figure 5-12. The bode plot shown in figure 5-13 shows the gain plot starts breaking up at lower frequencies. For this process the crossover frequency is around 0.2 *radians/sec.*

The results from the PRBS analysis must be used carefully, especially if impulse response technique is being used. The intelligent controller provides intelligent auto-tuning with the aid of the system cultivation tools. The PRBS techniques can minimize the effects of small disturbances during tuning. If the PRBS sequence length is large and a transient disturbance affects the system only informational messages are displayed. However if a permanent disturbance is detected the auto-tuner is turned off and appropriate alarms set.

5.4 Controller Performance Evaluation

An intelligent controller needs to be able to evaluate its own performance. In the absence of significant computing power it has to use qualitative means to attain the goal. It also has to be able to deal with the credit assignment problem. Poorly tuned

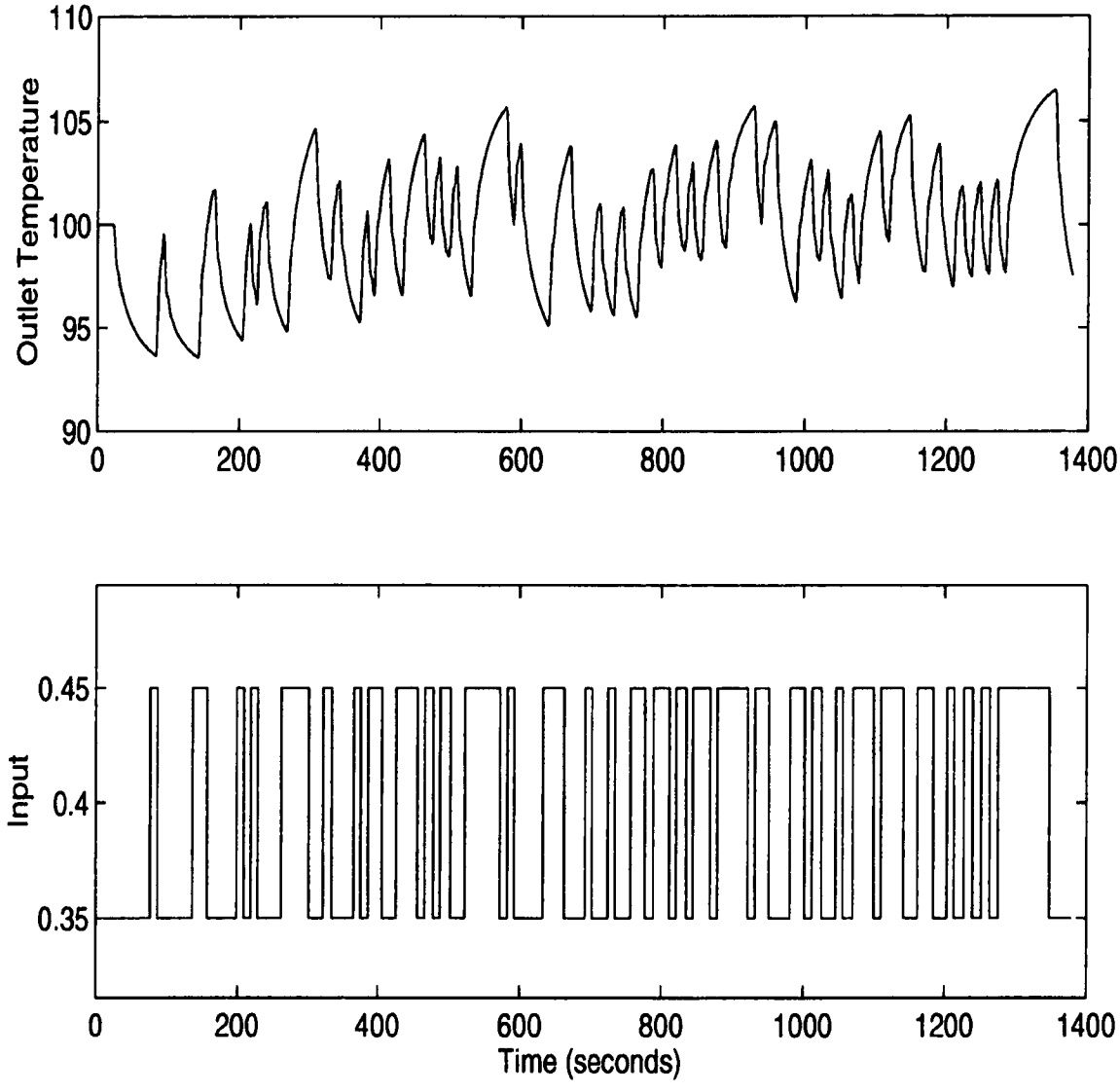


Figure 5-12: PRBS run that gives poor modeling results at a water flow rate of 12 *gpm*.

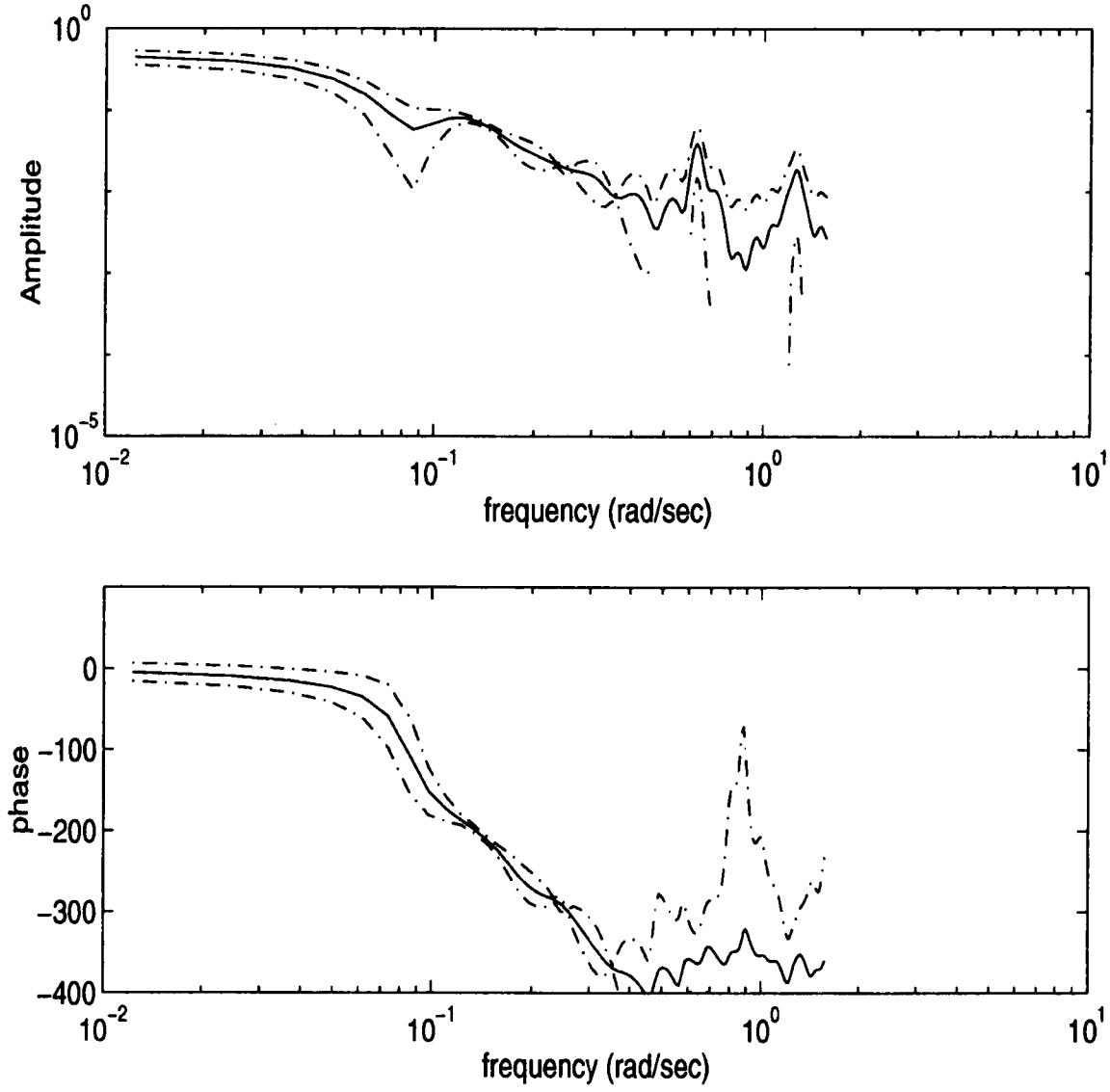


Figure 5-13: Bode Plot for the PRBS run shown in the previous figure. The dashed lines represent the confidence intervals on the respective plots.

controllers have two modes of operation. They are either under tuned or over-tuned. The first leads to sluggish response. The other could lead to an oscillatory trend.

5.4.1 Controller performance evaluation for set-point change

The controller performance can be evaluated only when a set-point change is made or the controller is responding to a disturbance. Performance evaluations are not done for disturbance rejections. Here is the algorithm used to do performance evaluations.

The system cultivation tools are used to monitor the controlled variable, the controller error and changes in the set-point. This information is used along with the information on the other process variables to provide a rule based expert system that provides intelligent control.

- On detecting an edge on the set-point, the following actions are taken:

  1. Check to see if the outlet temperature is at steady state, if not abort evaluation.
  2. A CFB is activated to start independent ISE and IAE evaluations and Oscillation indications.
  3. A software timer is activated on the PLC. It is preset to 120 seconds (selectable by the user).

- If the disturbance flag is activated during the 90 seconds the timer is active. The performance evaluation is aborted as it is likely that the disturbance is going to affect the results.

- At the end of the timing operation, check to see if the error has reached set-point (within the yellow deviation indicator). This is also indicated by the fact that $d_e(t)$ *and* $\dot{d}_e(t)$ have returned to zero within some threshold.

- It it has, then record the $ISE, IAE, \Delta S_p$,date, time, control algorithm active and its tuning constants (if available), the number of times the error crossed zero (this is indicated by the sum of the threshold crossing counters for d_{e+} and d_{e-}) into the database.

- If it has not, then check the following:

  1. How many times has the error crossed zero?
  2. If more than once the controller is likely to be providing oscillatory performance. Record all information into database set a flag to indicate process did not stabilize in 120 seconds.
  3. If zero than the controller is poorly tuned. Set an alarm and inform the user. No corrective action is taken.

- Reset the CFB blocks, oscillation indicators and wait for the next set-point change.

This procedure gives a crude evaluation for the performance of the controller.

5.4.2 Too tightly tuned controller

The other case, an over-tuned controller can induce oscillatory trends in the controlled variable. These can be transmitted to downstream processes and can affect its performance. The positive and negative error decay terms are used to detect oscillations in process variables. The system cultivation daemon helps determine the cause of the oscillations and take corrective action, if possible, to stabilize the process.

Figure 5-14 and figure 5-15 show the response to a step change in the set-point from 100-105 $°F$, and disturbance rejection respectively. Note that the operation in region II for both the events. The performance is poor as the controller is tuned too aggressively. Figure 5-16 and figure 5-17 show the performance of the intelligent

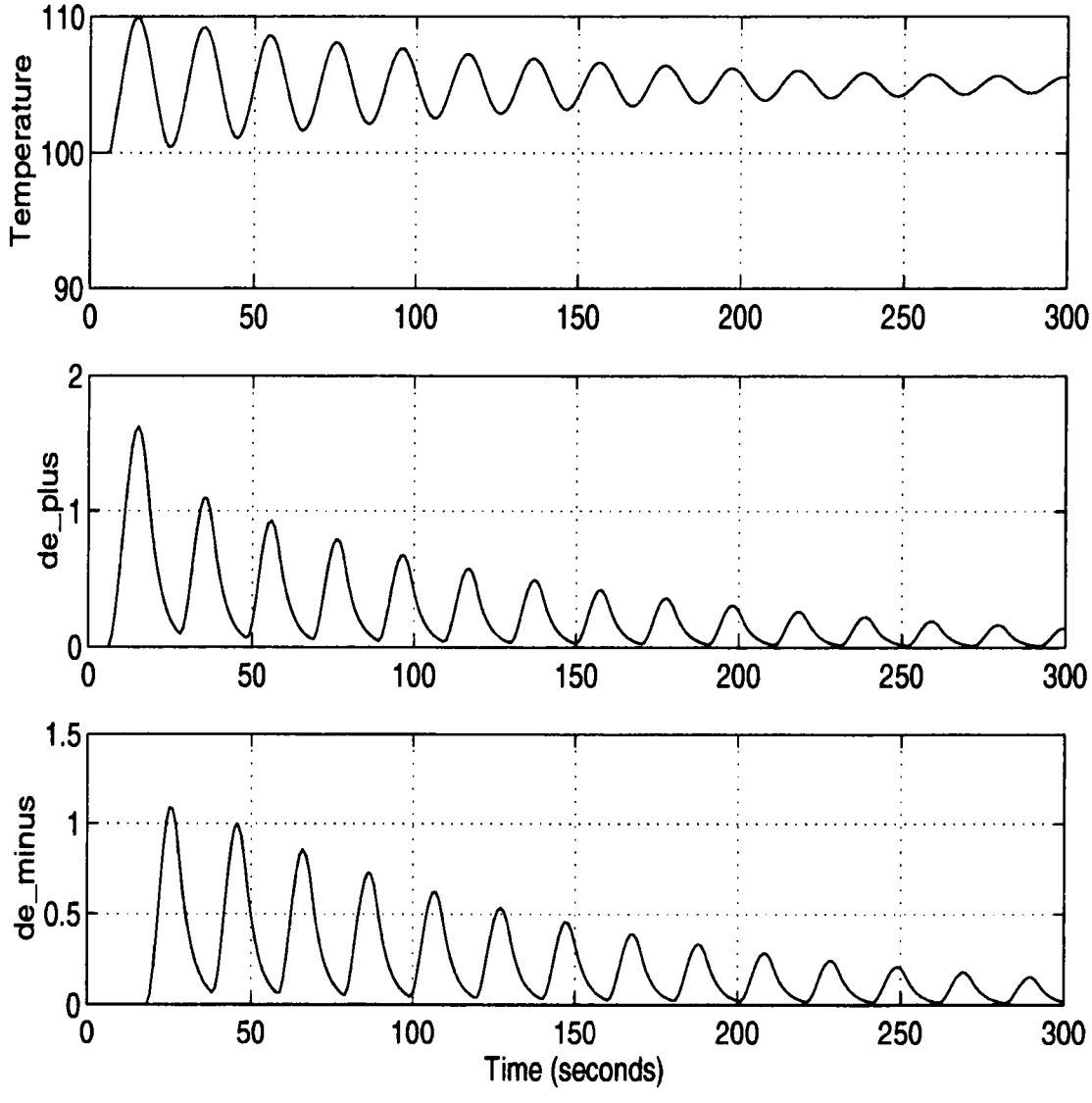


Figure 5-14: Response to a step change in set-point for a poorly tuned controller. The *de_plus* and *de_minus* plotted are based on the controller error.

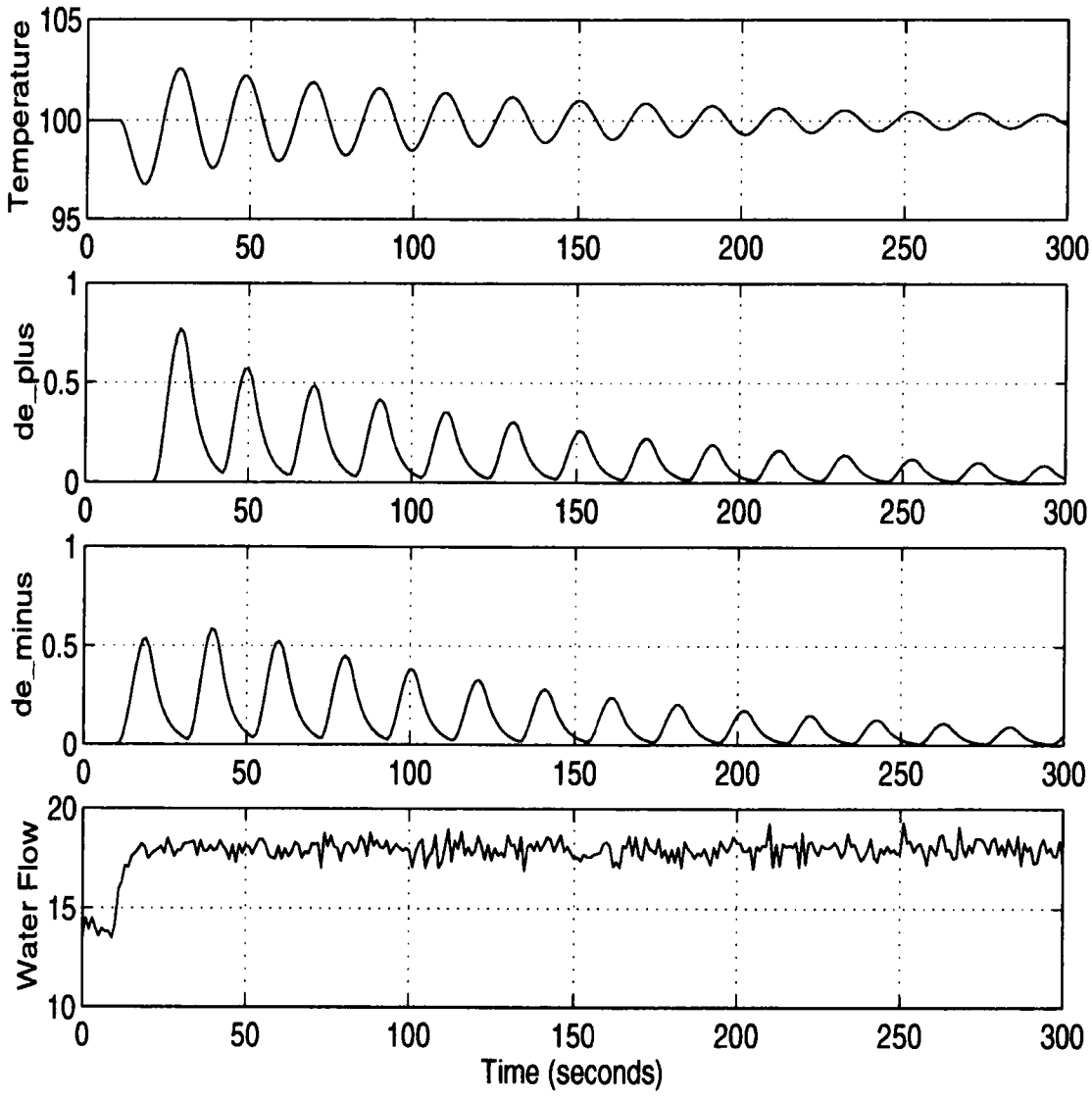


Figure 5-15: Disturbance rejection of the poorly tuned controller. The *de_plus* and *de_minus* plotted are based on the controller error.

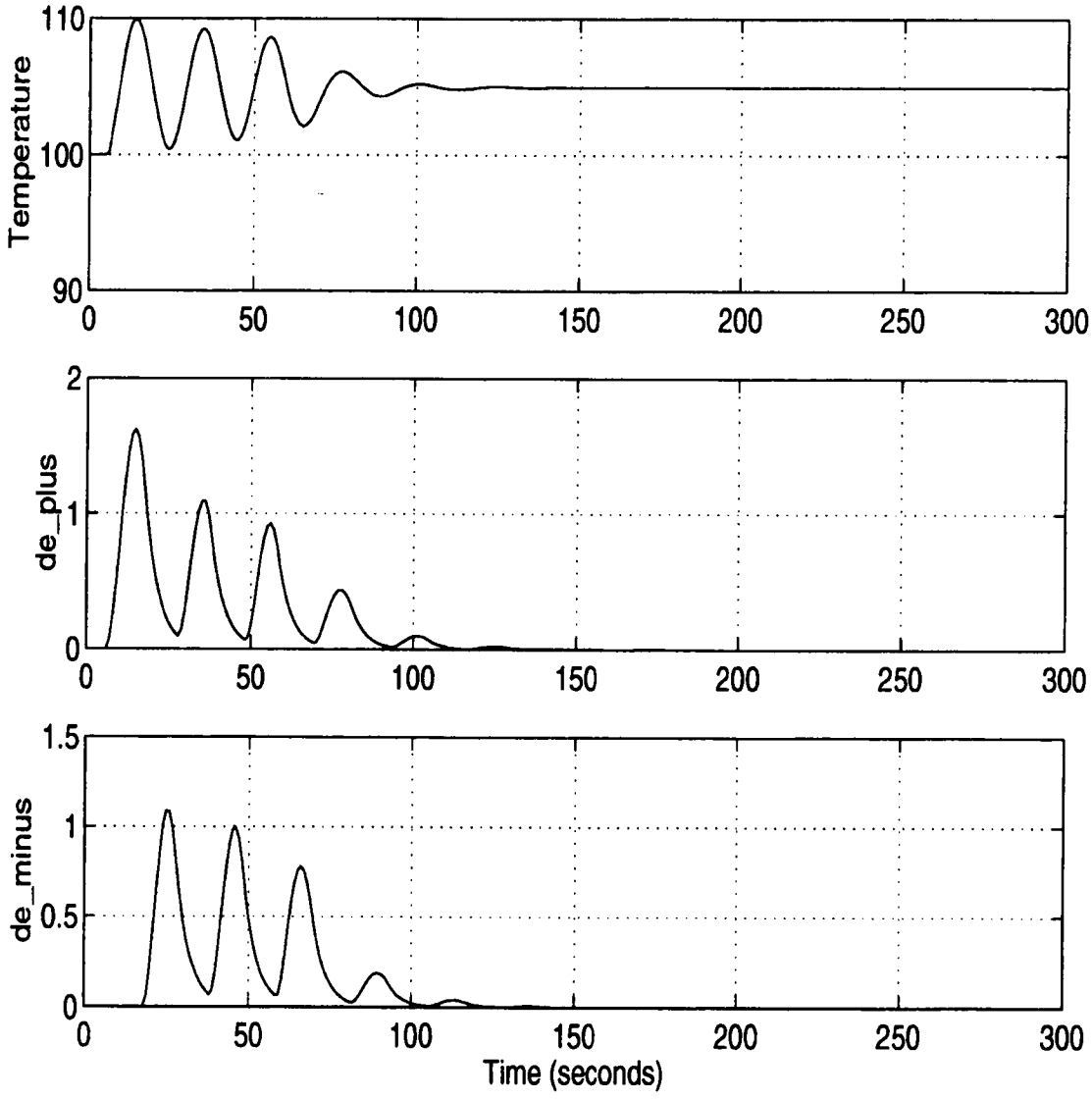


Figure 5-16: Performance of the intelligent controller for a set-point change. The *de_plus* and *de_minus* plotted are based on the controller error

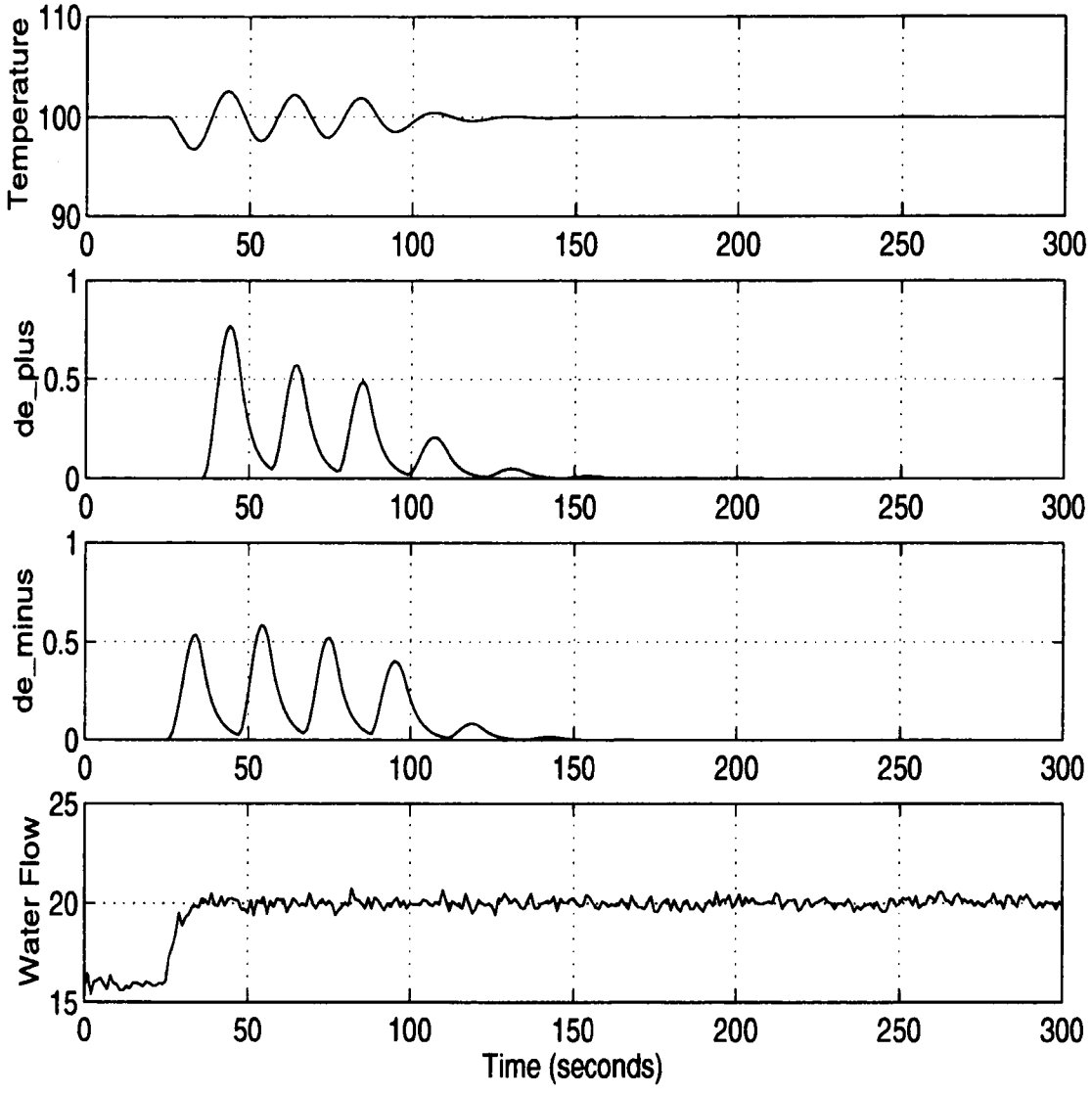


Figure 5-17: Performance of the intelligent controller for disturbance rejection. The *de_plus* and *de_minus* plotted are based on the controller error

controller. The system cultivation daemon is set to detect oscillations for d_{e+} and d_{e-} threshold crossing counts of 2 with a threshold of 0.1. The system cultivation daemon, on detecting oscillations reduces the controller gain by 30%. Upon making the corrective action the threshold crossing counts are set to zero. The oscillation indicators still keep computing. If for some reason the controlled variable is still oscillating further corrective action will be taken.

Figure 5-18 shows the performance of the intelligent controller with a poorly tuned water flow controller. The controlled variable detects oscillations however it checks to see if any of the other process variables that affect the process are undergoing oscillations. It finds that the water flow rate is undergoing oscillations and decides not to take any corrective action.

The performance of the positive and negative error decay terms is also shown in the figures. The steps taken by the system cultivation daemon are explained here:

1. Oscillations are detected on a controlled variable.

2. Are any of the other process variables that affect this process under oscillations.?
   If yes: reset counters and goto 1

3. If the controller is responsible do the following:

   - Is an external controller active (e.g. DMC).? If yes disable external controller and let the PID controller take over. Reset counters and goto 1
   - If the PID controller is active set K_c to 70% of its current value, reset counters and goto 1.

Figure 5-19 shows the expert supervisory performance of the intelligent controller. The DMC controller starts providing oscillatory behavior. The system cultivation daemon detects oscillations and finds that an external control algorithm is active. It disables the external control algorithm and returns control to the PID controller.

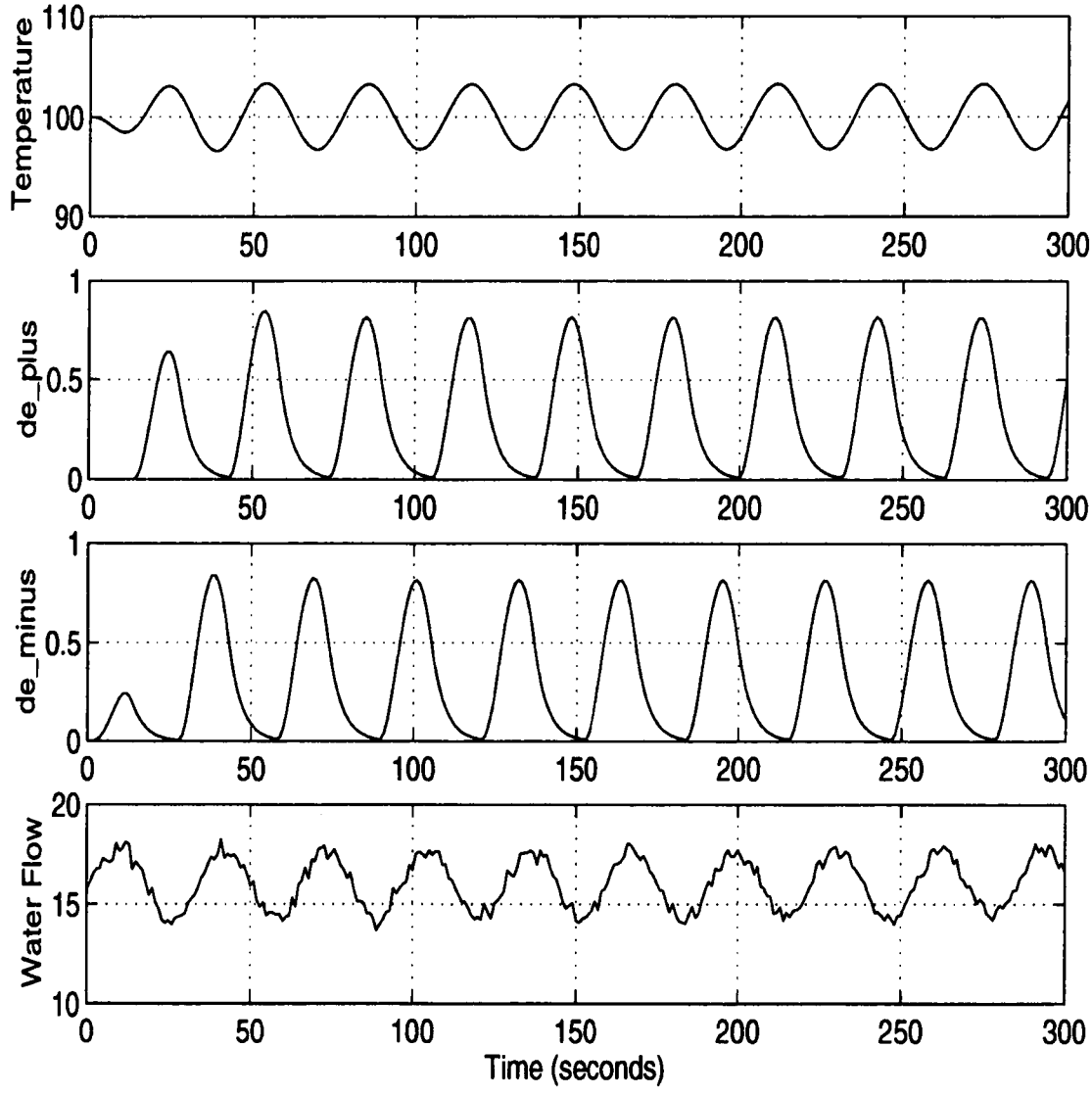


Figure 5-18: Performance of the intelligent controller with a poorly tuned flow loop.

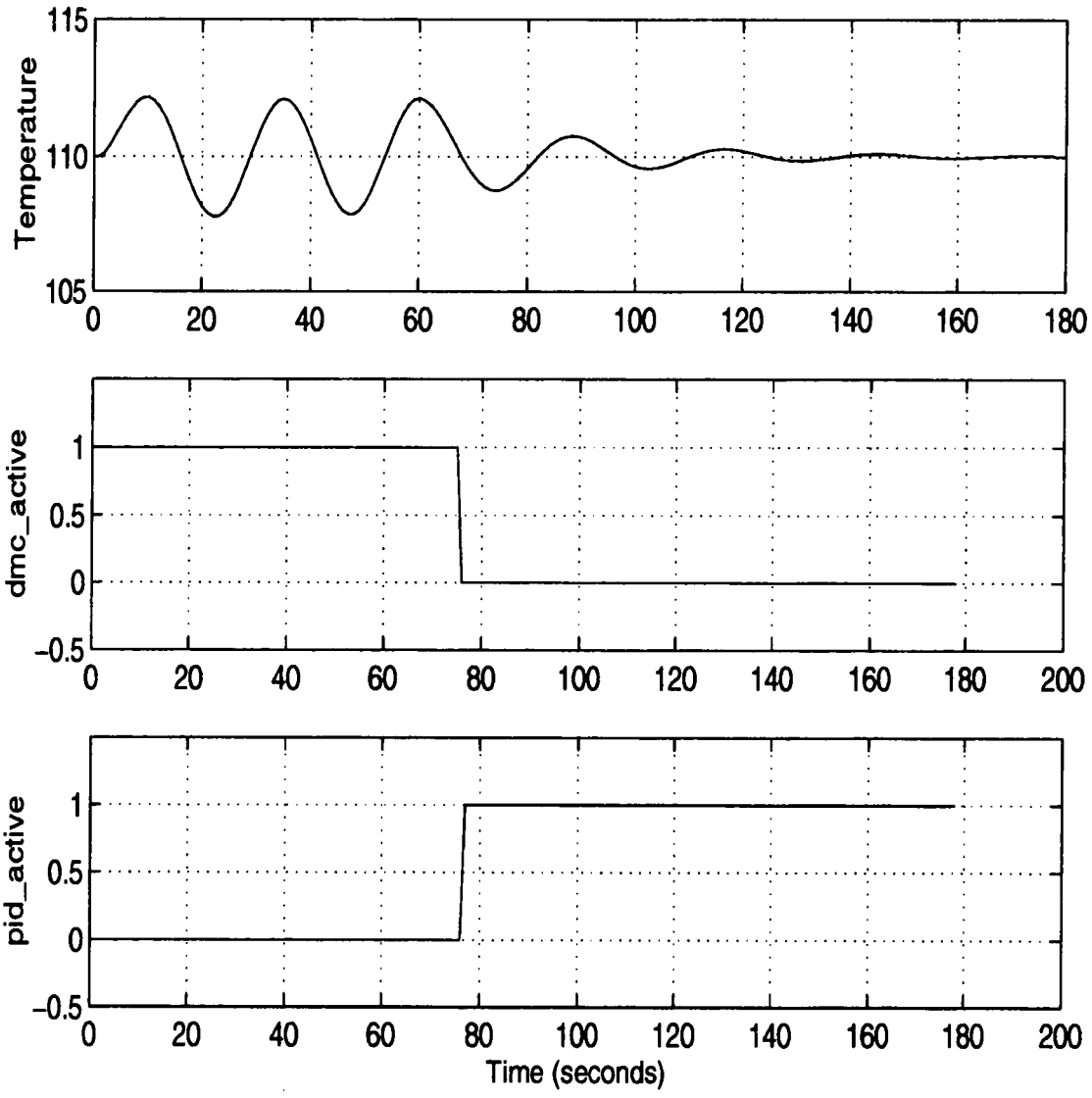


Figure 5-19: Expert supervisory operation of the intelligent controller. *dmc_active* is a flag that indicates that the DMC controller is active. *pid_active* is a flag that indicates that the PID controller is active.

ATV tuner or PRBS tuning induces oscillations in the controlled variables. During tuning the oscillation indicators for the controlled variable are disabled.

5.4.3 Detection of the improperly installed steam trap

About 50 PRBS runs were made on the heat exchanger, 25 at 12 gpm water flow rate and 25 at 18 gpm flow rate. The results from the data collected at 18 gpm yielded reasonable models with 4 bad data sets, that failed the Bode plot test. However the data collected at 12 gpm yielded inconsistent models with 10 bad data sets. The heat exchanger was then operated for 6 hours with the IPID controller. The operating point was allowed to span the operating space. Several ATV and PRBS tests were performed. The data collected was then analyzed using system cultivation to determine what was responsible.

The phenomena occurs at low water flow rates. This led the suspicion towards the steam traps. There are two ways to compute an energy balance for the heat exchanger. One is on the water side, the other on the steam side as shown in eqn 3.1. Standard analysis of the data does not reveal anything. However, using system cultivation, the data was sub-grouped into classes using the water flow rate. Thus if one was to plot Q_{steam}/Q_{water} vs. the change in water flow rate it should be equal to one with a little variation to account for the process dynamics.

Figure 5-20 gives the plot of Q_{steam}/Q_{water} for water flow rates of 10,15 and 20 gpm. The plot show that there is more loss of energy from steam at lower water flow rates. The water flow sensor is not very accurate at low flows. The steam flow sensor is also not very accurate at low steam flow rates. Amidst this uncertainty it is difficult to decide from the data about what is happening. the detection of the improperly. After talking to the manufacturer, who suggested that the trap should be leaking steam, a test was performed. With the water flow shut down, and the condensate collection valve open, steam was allowed to enter the heat exchanger. There was live steam coming out of the condensate collection port. With no condensate being

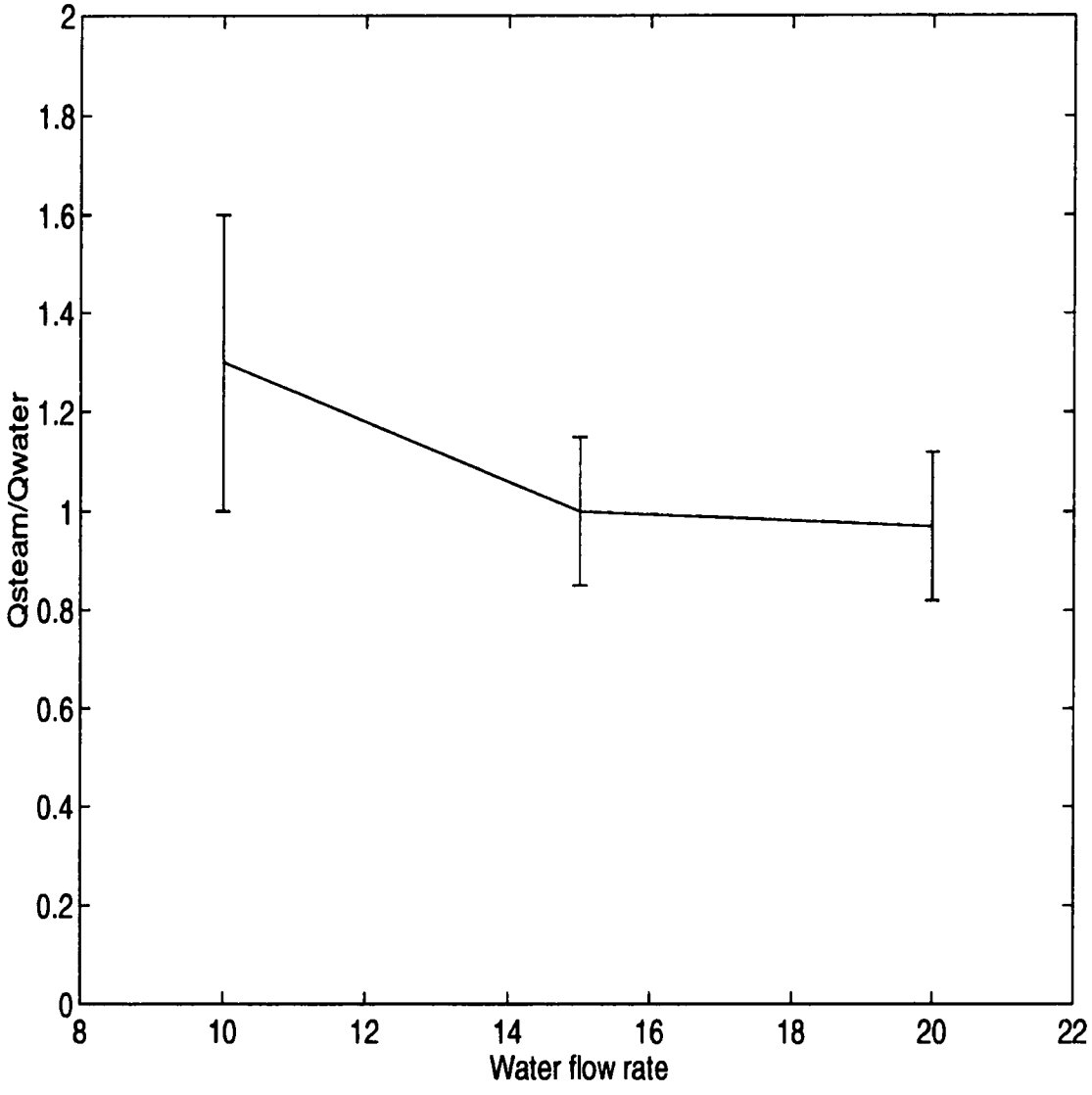


Figure 5-20: Plot of Q_{steam}/Q_{water} vs. Water flow in gpm. The high value at the low flow rate indicates the steam loss due to the steam trap.

formed, no steam should have escaped the trap. This verified that the steam trap was leaking.

Chapter 6

Conclusion and Recommendations

6.1 Conclusions

This work focused on using Artificial Intelligence to improve the performance of a control system on a shell and tube heat exchanger instrumented with an industrial PLC and an Operator Interface. The process provided unique challenges in its operation and implementing the intelligent controller. Lack of computing power, due to hardware limitations, forced the development of some qualitative reasoning tools to provide intelligent control.

- In the absence of significant computing power, qualitative tools, which provide shallow reasoning, can provide intelligent behavior.
- System Cultivation tools have been developed to identify trends in process variables. These tools can act as *disturbance* indicators and *oscillation* indicators. They are used to track and update the operating region of the controller. The tuning constants for these system cultivation tolls are determined by heuristics.
- These tools also provide the means to determine a aggressively tuned controller and de-tune it.

- Intelligent system identification techniques have been developed that use conventional system identification tools to detect disturbances that can affect the system during auto-tuning and disable the auto-tuner.

- Information from the system cultivation tools is used by a simple expert system to enhance the controller performance evaluation and represent process information in a heiarchial manner.

- A general framework is developed to implement advanced control strategies. These are implemented through the PID controller and if they provide unstable performance they are disabled.

- Conventional PID control is very "forgiving", in the sense that it can tolerate flaws in process design. Advanced control and system identification techniques demand a high degree of performance from the process and would require a lot of process problems to be fixed before they can be implemented.

6.2 Recommendations

Current technological advances in process control are leading towards plant wide control and optimization, the use of artificial intelligence to improve process operation. However the intelligent and supervisory decisions are made by a central computer. This can overwhelm the supervisory control system as we expand the boundaries on our plants to perform plant wide control and optimization. There is a strong need for distributed intelligence to take the work load of the main computers. The industry has already responded to this by establishing the *FIELDBUS* standard which permits field devices to have simple intelligence and communicate at high speeds to the controller. These devices are capable of running self diagnostics and will be able to provide multivariable information from one field device.

- Intelligent Control requires considerable computing power and memory. Better hardware needs to be used to implement intelligent control.

- Work needs to be done in building intelligent systems that between field devices and the supervisory control system that can monitor process performance, controller performance and provide local intelligent control. These intelligent systems should be able to store information in a hicarcheal system which can be recalled on demand.

- The system cultivation tools developed need to be refined and implemented on a more complex process like a distillation column. Its impact on plant wide control should be studied.

BIBLIOGRAPHY

Bibliography

[1] Moore B. C, "Analysis of process dynamics using computer memory and controlled resolution," in *American Control Conference*, 1988.

[2] Moore B. C, "An acedemic spy in the real world of industrial computer control: A candid report," tech. rep., 'T Lab, 1986.

[3] Moore B. C, "Hypothesis feedback models for multivariable systems," *IEEE Trans. of Automatic Control*, pp. 1–10, 1986.

[4] Prett D. M. and Morari M., *The Shell Process Control Workshop.* Boston: Butterworths, 1987.

[5] Feigenbaum E. A. and Barr A., *The Handbook of Artificial Intelligence, Vol I-IV.* New York: Addision Wisely Publishing Co., 1989.

[6] Kuipers B., "Commonsense reasoning about causality: Deriving behaviour from structure," *AI*, vol. 24, pp. 169–204, 1984.

[7] Forbus K. D., "Qualitative process theory," *AI*, vol. 24, pp. 85–164, 1984.

[8] Borrow D. G., "Qualitative reasoning about physical systems: An introduction," *AI*, vol. 24, pp. 1–6, 1984.

[9] Maren A. J., *Handbook of Neural Computing Applications*, ch. 22, pp. 345–380. Acedemic Press, 1990.

[10] Gupta M. M. and Rao D. H., *Neuro-Control Systems, Theory and Applications*, ch. Neuro-Control Systems: A Tutorial, pp. 1–43. IEEE Press, 1994.

[11] Antsaklis P. J., Pasino K. M., and Wang S. J., "An introduction to autonomus control systems," *IEEE Control Systems Magazine*, pp. 5–13, 1991.

[12] Narendra K. S. and Mukhopadhyay S., "Intelligent control using neural networks," *IEEE Control Systems Magazine*, pp. 11–18, 1992.

[13] Wu Q. H., Hogg B. W., and Irwin G. W., "A neural network regulator for turbogenerators," *IEEE Transactions on Neural Networks*, pp. 95–100, 1992.

[14] Michalski R. S., Carbonell J. G., and Mitchell T. M., *Machine Learning.* Morgan Kaufmann Publishers Inc., 1988.

[15] Rich E., *Artificial Intelligence.* New York: McGraw Hill Book Co., 1983.

[16] Åström K. J., Anton J. J., and Arzen K. E, "Expert control," *Automatica*, vol. 22, no. 3, pp. 277–286, 1986.

[17] Laffey T. J., Cox P. A., Schmidt J. L., Kao S. M., and Read Y. J., "Real-time knowledge-based systems," *AI Magazine*, vol. 9, no. 1, pp. 27–45, 1994.

[18] Chantler M. J., *Real-time aspects of expert systems in process control.* Savoy Place, London, UK.: IEE, 1988.

[19] Arzen K. E., "Expert systems for process control," *Proc. First Int. Conf. on Applications of Artificial Intelligence in Engineering Practice*, pp. 1127–1138, 1986.

[20] Arzen K. E., "Use of expert system in closed loop feedback control," *Proceedings of the American Control Conference*, pp. 140–145, 1986.

[21] Krijgsman A. J., Verbruggen H. B., and Bruijn P. M., "Knowledge-based real-time control," *IFAC Proc. on Artificial Intelligence in Real-Time Control*, pp. 13–19, 1988.

[22] Jiang J. and Doraiswami R., "Performance monitoring in expert control systems," *Preprints 10th World Congress on Automatic Control*, vol. 6, pp. 303–307, 1987.

[23] Lattimer W. M. *et. al.*, "An expert system for real-time control," *IEEE Software*, pp. 16–24, March 1986.

[24] Moore R. L. *et. al.*, "Expert system methodology for real-time process control," *Preprints 10th World Congress on Automatic Control*, vol. 6, pp. 387–392, 1987.

[25] Fink P. A., "Control and integration of diverse knowledge in a diagnostic expert system," *Proceedings of the 9th IJCAI*, pp. 426–431, 1985.

[26] Moore R. L. and Kramer M. A., "Expert systems in on-line process control," *Expert Systems in Process Control*, pp. 839–867, 1987.

[27] Mavrovouniotis M. L., *Arificial Intelligence in Process Engineering.*, ch. 6, pp. 189–221. Academic Press, Inc., 1990.

[28] Porter B., Jones A. H., and McKeown C. B., "Real-time expert tuners for pi controllers," *IEE Proceedings-D*, vol. 134, no. 4, pp. 260–263, 1987.

[29] Kraus T. W. and Myron T. J., "Self-tuning pid controller uses pattern recognition approach," *Control Engineering Magazine*, June 1984.

[30] Zadeh L. A., "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Trans. on Systems, Man and Cybernetics*, vol. SMC-3, no. 1, pp. 28–44, 1973.

[31] Kaufmann A. and Gupta M. M., *Introduction to Fuzzy Arithmetic.* New York: Van Nostrand Reinhold Co., 1985.

[32] Dubois D. and Prade H., *Fuzzy Sets and Systems.* New York: Acedemic Press, Inc., 1980.

[33] Zimmermann H. J., *Fuzzy Set Theory- and Its Applications.* Boston: Kluwer-Nijhoff Publishing, 1985.

[34] Dubois D. and Prade H., "Recent literature," *Fuzzy Sets and Systems*, vol. 26, pp. 225–242, 1988.

[35] Mamdani E. H., "Advances in linguistic synthesis of fuzzy controllers," *Int. J. Man-Machine Studies*, vol. 8, pp. 669–678, 1976.

[36] Braae M. and Rutherford D.A., "Theoritical and linguistic aspects of the fuzzy logic controller," *Automatica*, vol. 15, pp. 553–577, 1979.

[37] Lee C. C., "Fuzzy logic in control systems: Fuzzy logic controller-part i and ii," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 20, no. 2, pp. 404–435, 1990.

[38] Li Y. F. and Lau C. C., "Development of fuzzy algorithms for servo systems," *IEEE Control Systems Magazine*, pp. 65–71, April 1989.

[39] Mamdani E. H., "Application of fuzzy algorithms for control of a simple dynamic plant.," *Proc. IEEE*, vol. 121, pp. 1585–1588, 1974.

[40] Umbers D. G and King P. J., "An analysis of human decision-making in cement kiln control and the implications for automation," *International Journal of Man-Machine Studies*, pp. 11–23, 1980.

[41] Kickert W. J. M. and van Nauta Lemke H. R., "Application of a fuzzy controller in a warm water plant," *Automatica*, vol. 12, pp. 301–308, 1976.

[42] Morita Y., Nakamura T., and Kuratani T., "Fuzzy control application to glutamic acid fermentation," *IFAC Modeling and Control of Biotechnological Processes*, pp. 231–235, 1985.

[43] Østergaard J. J., "Fuzzy logic control of a heat exchanger process," *Internal Report 7601. Electric Power Engineering Dept.*, 1976.

[44] Kandel A. and Langholz G., *Fuzzy Control Systems.* Boca Raton: CRC Press, 1994.

[45] Siiao S., "Fuzzy self-organizing controller and its application for dynamic processes.," *Fuzzy Sets and Systems*, vol. 26, pp. 151–164, 1988.

[46] Hsia T. C., *System Identification.* Toronto: Lexington Books, 1977.

[47] Prett D. M., Skrovanek T. A., and Pollard J. F., *Process Identification – Past, Present, Future*, pp. 79–104. Boston: Butterworths, 1987.

[48] Åström K. J. and Eykhoff P., "System identification – a survey," *Automatica*, vol. 7, pp. 123–162, 1971.

[49] Åström K. J., "State of the art and needs in process identification," *AIChE Symposium Series*, vol. 72, no. 159, pp. 184–194, 1976.

[50] Anderson H. W., Rasmussen K. H., and Jorgensen S. B., *Advances in Process Identification*, pp. 237–269. New York: AIChE, 1991.

[51] Eykhoff P., *System Identification – Parameter and State Estimation.* New York: John Wiley and Sons, Inc., 1974.

[52] Ljung L., *System Identification. Theory for the User.* Prentice Hall, 1987.

[53] Andersen H. W., Rasmussen K. H., and Jorgensen S. B., "Advances in process identification," in *Chemical Process Control CPCIV*, 1991.

[54] Hang C. C., Lee T. H., and Ho W. K., *Adaptive Control.* Triangle Research Park: Instrument Society Of America, 1993.

[55] Åström K. J. and Hagglund T., *Automatic Tuning of PID Controllers.* Research Triangel Park: Instrument Society of America, 1988.

[56] Åström K. J. and Wittenmark B., *Adaptive Control.* New York: Addison-Wesley Publishing Co., 1989.

[57] Seborg D., Edgar T., and Shah S. L., "Adaptive control strategies for process control: A survey," *AICHE Journal*, vol. 32, no. 6, pp. 881–913, 1986.

[58] Narendra K. S. and Monopoli R. V., *Applications of Adaptive Control*, ch. Application of Adaptive Systems in Adaptive Control, pp. 509–554. Academic Press, 1980.

[59] Kern D. Q., *Process Heat Transfer.* New York: McGraw Hill Book Co., 1961.

[60] Maren A. J., *Handbook of Neural Computing Applications*, ch. 18, pp. 295–308. Acedemic Press, 1990.

[61] Bakshi B. R. and Stephanopoulos G., "Representation of process trends–iii. multiscale extraction of trends from process data," *Computers in Chemical Engineering*, vol. 18, no. 4, pp. 267–302, 1994.

[62] Hackney J. E., *Cultivation of Model Predective Control Systems.* PhD thesis, University of Tennessee, Knoxville, TN, 1989.

[63] Cutler C. R. and Ramaker B. L., "Dynamic matrix control– a computer control algorithm," *Proc. JACC*, 1980.

[64] Garcia C. E., Morari M., and Prett D.M., "Model predictive control: Theory and practice," *IFAC Workshop on Model based Pocess Control*, June 1988.

[65] Stephanopoulos G., *Chemical Process Control.* Prentice-Hall, 1984.

[66] Åström K.J., *Automatic Tuning of PID Controllers.* Instrument Society of America, 1988.

Vita

Bipin Mavji Shah was born on 19th July 1960 in Bombay, India. He graduated with a B.S. in Chemistry from Bombay University in May 1980. He then joined the Bombay University Department of Chemical Technology. He came to the U.S. in August of 1982 and joined the University of Tennessee. He received his B.S. in Chemical Engineering in June of 1984. He then received his M.S. in Chemical Engineering in March of 1988. In May 1994 he received his Ph.D. in Chemical Engineering.