The adaptive control of a coal-fired fluidized bed combustor

by

Kenneth W. Junk

A Thesis Submitted to the
Graduate Faculty in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
Major: Mechanical Engineering

Iowa State University
Ames, Iowa
1989

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

d	=	Deviation digital command signal
D_s	=	Digital command signal
e	=	Error signal
I	=	Identity matrix
J	=	Performance index
K	=	Observer feedback gain vector
K_i	=	Integral gain
K_p	=	Proportional gain
L	=	LQG feedback gain vector
P	=	Covariance matrix
Q	=	Air flow rate, scfm
Q^*	=	Air flow rate set point, scfm
q	=	Forward shift operator
q^{-1}	=	Backward shift operator
R	=	Reaction rate, $^\circ$F/min
R_N	=	Weighting matrix (terminal state)
R_u	=	Weighting matrix (system input)
R_x	=	Weighting matrix (system output)

T	=	Temperature, °F
t	=	Discrete time index
u	=	System command signal
v	=	Gaussian random variate
X	=	Augmented state vector
x	=	State vector
Y	=	Augmented output vector
y	=	System output signal
y^*	=	System set point
z	=	Integrator variable

Greek Symbols

δ	=	Kronecker delta
λ	=	Exponential weighting factor
Φ	=	Model matrix
ϕ	=	Coefficient vector
σ^2	=	Variance
τ	=	Apparent dead time, min
θ	=	Variable vector

Usage: The diacritical ($\wedge$) indicates an estimate.

LIST OF ABBREVIATIONS AND ACRONYMS

A/D	Analog-to-Digital
ANOVA	Analysis of Variance
ARMA	Autoregressive Moving-Average
CFB	Circulating Fluidized Bed
CPU	Central Processing Unit
D/A	Digital-to-Analog
DARMA	Deterministic Autoregressive Moving-Average
DF	Degrees of Freedom
E/P	EMF-to-Pneumatic
FBC	Fluidized Bed Combustor
GLM	General Linear Models
LPG	Liquefied Petroleum Gas
LQG	Linear Quadratic Gaussian
LVDT	Linear Variable Displacement Transformer
MIMO	Multiple Input, Multiple Output
MRAC	Model-Reference Adaptive Control
MRE	Matrix Riccati Equation
P	Proportional

PE	Persistently Exciting
PI	Proportional-Integral
PID	Proportional-Integral-Derivative
RMS	Root-Mean-Square
SAS	Statistical Analysis System
SISO	Single Input, Single Output
SOAS	Self-Oscillating Adaptive Systems
SS	Sums of Squares
STR	Self-Tuning Regulators
SVR	State Variable Representation
Z/N	Ziegler-Nichols

1. INTRODUCTION

In recent years, fluidized bed combustors (FBCs) have become an attractive means for meeting future power generation and process heat requirements in an environmentally acceptable manner. Among many favorable characteristics, fluidized bed combustors are capable of burning low-grade, variable quality fuels at low combustion temperatures. Coal-fired FBCs, for example, typically operate in the range of 1400 °F to 1800 °F, to be compared with temperatures of 3500 °F for boilers that utilize cyclone furnaces [20]. The lower combustion temperatures result in a reduction of NO_x emissions, elimination of ash clinkering, and elimination of slag fouling. In addition, an acceptor stone, such as dolomite or limestone, is often used in the bed material to reduce the sulfur content of the product gases. However, because of inherent nonlinear dynamics arising from fluidization, radiation, and chemical kinetics, classical linear control schemes may be inappropriate. Consequently, new algorithms to control temperature in FBCs should be examined.

Traditionally, many dynamic systems are regulated using simple linear feedback controllers. From step inputs, a linear model can be formulated, a root locus constructed, and a proportional-integral-derivative (PID) controller designed. Alternatively, PID control may be designed through heuristic techniques, such as the Ziegler-Nichols method [9]. Although predicated on linear system theory and design, PID

controllers have a robust nature such that the closed-loop system will usually remain stable even though weakly nonlinear dynamics may be present. Fluidized bed combustors, on the other hand, are highly nonlinear systems [27,32] and have transient natures that are not well understood. When highly nonlinear systems are controlled using linear techniques, they may be poorly tuned in the desired operating range as nonlinear behavior is often compensated through excessively conservative design [17]. To improve system performance, several strategies have been developed which attempt to contend with nonlinear system dynamics. One particularly successful approach has been the use of adaptive control design techniques. Controllers built upon these techniques not only promise to improve system performance, but also lend themselves well to applications in process control.

This investigation explores the temperature control of a two-bed fluidized combustor using an adaptive optimal-control algorithm to vary air flow rate. The controller consists of a recursive least-squares parameter estimator, an observer, and a linear quadratic Gaussian (LQG) control design procedure. This combination enables the controller to estimate system parameters and update feedback gains when necessary. Further, this study addresses the tracking form of optimal-control, accomplished by augmenting the state vector with an integrator. Finally, as a reference, the adaptive control algorithm is compared with a PI controller tuned by the Ziegler-Nichols method.

2. FLUIDIZED BED COMBUSTOR OPERATIONS

This chapter reviews the fundamentals of fluidization as related to fluidized bed combustor operations. Included are discussions of conventional temperature control techniques and their corresponding limitations. The chapter concludes with a discussion of temperature control using a two-bed combustor design.

2.1 The Nature of Fluidization

Fluidization is the process by which a gas or liquid passes vertically through a bed of fine solids, setting the particles in motion. If agitated sufficiently, the aggregate motion of solids resembles that of a turbulent fluid. The term fluidization, then, is simply an appellation, intended to draw parallels with turbulent fluid flow. However, aside from some visual similarities, the analogy with turbulent fluid flow is weak as fluidization exhibits many unusual characteristics associated with gas-solid and liquid-solid interactions.

Fluidization progresses through many stages, each dependent on the flow velocity of the fluidizing medium. At low flow velocities, the bed remains quiescent (a "dead bed") as fluid merely percolates through voids between particles. A slight rise in the flow rate will expand the bed, and localized packets of particles will begin to vibrate.

As the flow velocity increases, the bed passes through a state of incipient flu-

idization. At this stage, particle weight is counterbalanced by frictional forces from the fluidizing medium, resulting in the suspension of solids. A further increase in the flow rate above the point of incipient fluidization produces a heterogeneously fluidized bed, characterized by formation of bubbles as the fluidizing medium travels upward through the bed material.

At very high flow velocities, solid particles become entrained in the fluidizing medium and are elutriated from the bed. This loss of bed material is undesirable in most bubbling bed designs. However, characteristics associated with elutriation can be advantageous when utilized in circulating fluidized bed (CFB) designs. CFB designs capture and recirculate entrained bed material and, more importantly, recirculate unburned (solid) fuel back into the combustion chamber.

Most fluidized beds and fluidized bed systems are of bubbling bed design, circulating bed design, or hybrids of circulating and bubbling bed designs. Of the numerous design possibilities, this study investigates adaptive control as applied to a coal-fired, bubbling bed combustor with a fluidizing medium of air and bed material of sand.

2.2 Fluidized Bed Operations

References to fluidized beds are scattered throughout history; however, the first important commercial development was by Fritz Winkler for powdered coal gasification. Patented in 1922 and operational by 1926, the first unit stood 39 ft tall and had a cross sectional area of 108 ft^2 [21]. A few years later, a (spouting) fluidized bed was developed for combustion of coal. Designed by J.F.O. Stratton, this unit was installed at a U.S. Gypsum Company paper mill in 1928 and had a capacity of 5000

lb coal/hr [28].

Through the 1940s and 1950s, substantial progress in unit designs and efficiencies were made by many firms in the process industries, viz., Badische Anilin und Soda-Fabrik (BASF), Dorr-Oliver, Sumitomo Chemical Manufacturing Company (Japan), Union Carbide, Combustion Engineering, and Standard Oil Company (New Jersey) [21,28]. Developments in fluidized bed designs led to applications as diverse as catalytic cracking, drying and sizing of powdery materials, coating of metals, ore roasting, microencapsulation, coal gasification, and coal combustion.

Of particular interest is the application of coal-fired bubbling beds for steam generation. Bubbling beds have many distinct advantages over the conventional mechanical stoker, pulverizer-burner, and cyclone furnace combustion systems. For example, the turbulent nature of fluidized beds encourages mixing of oxygen with carbon, leading to an efficient combustion process. If a sulfur sorbent, such as limestone or dolomite, is fed into the bed, the turbulent environment allows for effective in situ desulfurization of the product gases [30]. In addition to favorable mixing characteristics, the large surface area associated with the bed material provides for improved heat transfer rates between combustion gases and the bed. Furthermore, high heat transfer rates, coupled with efficient combustion and vigorous mixing, create an isothermal combustion chamber, which is ideal for uniform temperature control.

2.3 Temperature Control and the Two-Bed Concept

Nonlinear and nonstationary behavior is inherent in virtually all aspects of fluidized bed combustors, and nowhere is this more apparent than in the temperature of

the bed. Wide variations in temperature can be attributed to nonlinear heat transfer coefficients as well as to the highly variable (and unpredictable) composition of coal. However, slight fluctuations in temperature significantly reduce sorbent effectiveness, adversely affect NO_x emissions, and diminish combustion efficiency [10]. Therefore, constant and uniform bed temperature is essential for optimal fluidized bed performance.

Control of bed temperature must be accomplished without extensive equipment modifications during combustor operation. For a fixed coal feed rate, bed temperature can be adjusted by one or a combination of the following [10,28]:

- recirculating flue gases
- discharging bed material
- varying air flow rates
- immersing heat exchanger tubes in the bed
- slumping sections of the bed

This list is not comprehensive, although it does present some of the more effective means by which bed temperature can be controlled. Most of these methods influence the bed temperature by changing the surface-to-bed heat transfer coefficient. However, the first two methods — recirculating flue gases and discharging bed material — are notable exceptions. The first of these alters bed temperature by recirculating flue gases back into the combustion chamber. Since flue gases are predominantly composed of nitrogen and carbon dioxide, they do not react with the fuel, but they do absorb energy released from the combustion process. The result is a decrease in bed temperature with a corresponding increase in gas temperature at the combustor exit.

Instead of recirculating flue gases, temperature control can be accomplished by discharging and accumulating hot bed material, allowing it to cool, and then reinjecting the cooler material back into the bed. However, this process requires considerable capital with accompanying technical, reliability, and maintenance difficulties.

A more effective approach to temperature control involves regulation of the fluidizing air velocity. As illustrated in Fig. 2.1, velocity of the fluidizing medium substantially alters the magnitude of the surface-to-bed heat transfer coefficient. At low flow velocities, the bed is fixed and changes in the heat transfer coefficient are slight. However, at the onset of fluidization, the heat transfer coefficient increases markedly, rising by one or two orders of magnitude for a fluidizing medium of gas and two to four orders of magnitude for a fluidizing medium of liquid [12,21]. The rapid rise in the heat transfer coefficient at the onset of fluidization can be attributed to an increase in particulate circulation. At a certain point, the heat transfer coefficient reaches a maximum and then decreases as the fluidization velocity increases. Decreases beyond the maximum can be explained by lower solid material concentrations associated with vigorous bubbling within the bed [12,21]. Therefore, changing fluidization velocities will change the rate heat is absorbed by the combustor wall, especially if the heat transfer process is augmented with a water jacket or water wall.

Heat can also be removed from the system by immersing heat exchanger tubes in the bed material. Comprehensive analyses of the heat transfer process with tubes immersed in fluidized beds are presented by Gelperin and Einstein [12] and by Kunii and Levenspiel [21]. In addition to heat transfer characteristics, studies have been conducted which investigate the integrity of tubes located within the abrasive bed en-

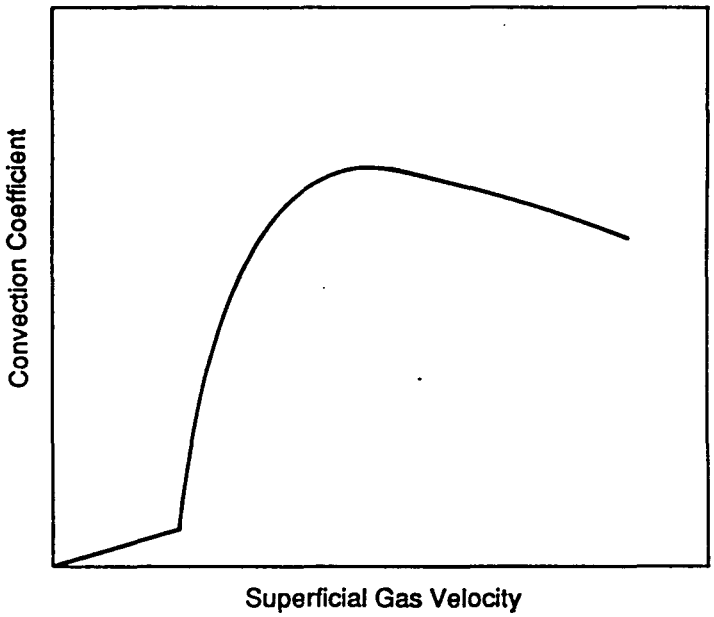


Figure 2.1: Convection heat transfer coefficient in a fluidized bed

vironment. For example, Nack et al. [28] reviewed a study conducted by the National Coal Board of England (1971) regarding tube corrosion, erosion, and ash deposition in coal-fired, bubbling bed combustors. In brief, the study did not find significant tube degradation due to ash penetration, indicating that the bed environment was not sufficiently erosive to remove the protective oxide layer and expose clean metal. In fact, erosion rates diminished with time, suggesting the formation of a stable protective oxide layer. Low erosion rates were to be expected as fluidization velocities ($<$11 ft/sec) were much less than flow velocities of 100 ft/sec encountered in conventional furnaces where impacting particles just begin to erode tubes. However, the study did find that tubes located in the freeboard region were more susceptible to erosion, although erosion rates were under the acceptable criterion of 1.5 μin/hr.

In addition, data concerning ash deposition, surface attack, and sulfide penetration suggested that few material problems were present for tubes located within bubbling beds. A notable exception was AISI 321 steel at 1500 $^{\circ}$F, which experienced severe pitting and intergranular sulfide penetration.

With or without immersed tubes, adjustments of fluidization velocity are limited to a lower value prescribed by stoichiometric air flow requirements, and an upper limit prescribed by elutriation characteristics. Furthermore, temperature control is restricted to manipulation of the fluidizing medium as heat transfer coefficients tend to be fire-side limited. That is, adjusting flow velocity in a water jacket, water wall, or immersed tube bundle has little effect on heat transfer from the combustion chamber. Control of bed temperature through regulation of fluidization velocity is further complicated by highly nonlinear radiation effects, nonuniform coal properties, and heat generation/air flow interactions.

Heat transfer rates and bed temperature can also be altered by slumping sections of the bed [5,10,28]. Slumping, or defluidization, is accomplished by partitioning the air plenum and selectively discontinuing air flow to regions of the bed. When defluidized, the higher heat transfer rates associated with solids circulation collapse to the lower heat transfer rates associated with conduction. The result is a reduction in the overall heat transfer rate from the combustion chamber. Although the rate at which heat can be removed from the bed can be altered, the slumping process has a formidable drawback; namely, the defluidized regions allow fuel to smolder, producing localized hot-spots that tend to sinter ash particles [5,10].

These procedures have had limited success in controlling bed temperature, the

effectiveness of which may be characterized by the load turndown ratio. Defined as the ratio of maximum to minimum energy output of the combustor, load turndown ratios in fluidized beds are modest at best, with most processes attaining ratios under four [6,7].

An alternative approach — where load turndown ratios greater than ten have been obtained [6,7] — utilizes a two-bed fluidized combustor. A schematic of a two-bed combustor is shown in Fig. 2.2. The combustor is comprised of a central, fluidized combustion bed and an annular, fluidized heat transfer bed. Each bed has an independent air plenum by which air flow rates can be adjusted separately so as to decouple combustion and heat transfer processes.

Temperature control is achieved by changing the air velocity in the annular bed. When no air is supplied to the annular bed, heat transfer from the central bed to the water jacket is by conduction through the bed material. As the air velocity increases to a point just before incipient fluidization, heat transfer is augmented by gas convection. A further increase in air velocity will fluidize the annular bed, adding particulate convection to the entire heat transfer process. The result is a dramatic increase in the rate at which heat is removed from the central bed and a corresponding decrease in the temperature of the central bed.

Heat transfer from particulate convection in the annular bed can be modified by changing the particle size distribution [29]. Heat transfer is enhanced with a smaller particle distribution due to an increased surface-area-to-volume ratio, but the advantage is bounded: a point is reached where the heat transfer coefficient on the central bed side becomes rate limiting [10,11]. Hence, large reductions in the

size distributions have little effect on the overall heat transfer characteristics of the system.

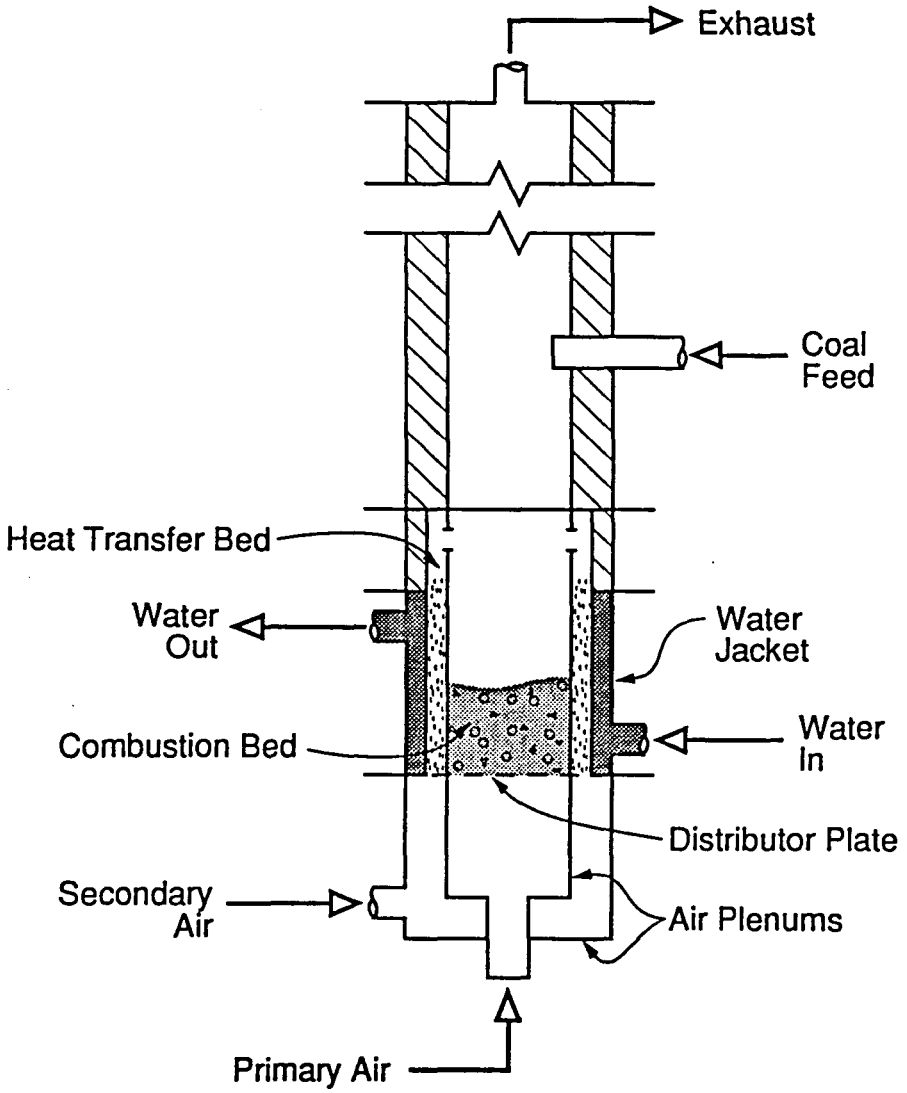


Figure 2.2: Schematic of a two-bed fluidized combustor

3. REVIEW OF CONTROL THEORY

Materials presented in this chapter review some of the fundamental concepts in both classical and modern control theory. Section 3.1 briefly examines classical control in the continuous time domain with an emphasis on Ziegler-Nichols tuning rules. Section 3.2 examines modern control theory in the discrete-time domain as applied to self-tuning controllers. Topics discussed in this section include system identification, state reconstruction, and optimal-control design procedures.

3.1 Classical Control Theory

Perhaps the most effective means by which systems can be regulated, guided, or otherwise commanded is through application of feedback control. A feedback controller generates a system command signal based on a function of an error signal, where error is defined as the difference between a reference signal (desired quantity) and a system output signal (actual quantity).

Feedback control can be traced as far back as Hellenistic period to Ktesibios. A mechanician who lived in Alexandria during the first half of the third century B.C. [26], Ktesibios is credited with developing the first feedback device in recorded history: a water clock where near-constant flow rates were obtained through use of a float valve controller.

Throughout history, various water clocks, oil lamps, and furnaces incorporating feedback controllers in their designs were developed, many of which remain obscure. However, the importance of feedback control was to come to light with the development of the flyweight governor. Although the flyweight governor was originally patented by Thomas Mead (1787) for speed control of windmills [4,26], Matthew Boulton and James Watt used the governor to control the speed of steam engines. Their success brought feedback control to the forefront of technology [4].

After the introduction of the flyweight governor, theory and design of feedback control grew enormously. As the field grew, proportional-integral (PI) control emerged as one of the most widely accepted methods of feedback control. From an error signal, $e(t)$, PI control generates an input signal, $u(t)$, which is comprised of components proportional to the error and proportional to the integral of the error. In the Laplace domain, with zero initial conditions, PI control may be described through the transfer function

$$\frac{u(s)}{e(s)} = K_p + \frac{K_i}{s} \tag{3.1}$$

wherein K_p and K_i are proportional and integral gains, respectively. If the dynamics of the plant are well known, K_p and K_i may be specified through root-locus, Nyquist or Bode design techniques. On the other hand, if the system dynamics are too complicated or not well known, as is the case with fluidized beds, data from an open-loop step response can be used to establish coefficients of an approximate system model. This model may be used in conjunction with heuristic design techniques, such as the Ziegler-Nichols (Z/N) method, to specify controller parameters. For PI control, the Z/N method specifies controller coefficients based on an assumed process model

consisting of a first-order system with a time lag, i.e.,

$$\frac{y(s)}{u(s)} = \frac{e^{\tau s}}{s+a} \tag{3.2}$$

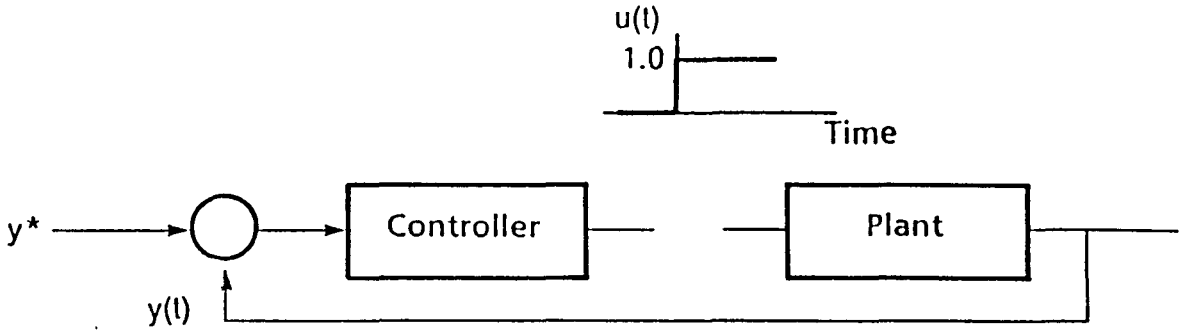
From an open-loop step response, estimates for the apparent dead time, τ, and the reaction rate, R, can be assessed (see Fig. 3.1). These two coefficients characterize the system whereby the Z/N method suggests a PI controller of the form [9]

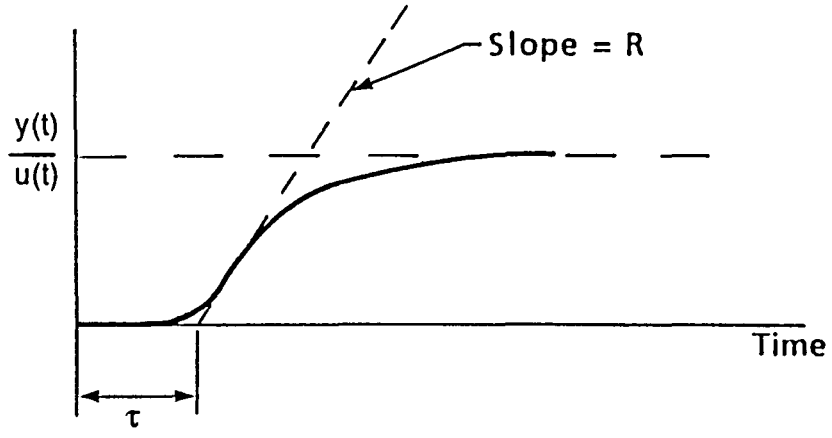
$$\frac{u(s)}{e(s)} = \frac{100}{110\tau R}\left[1 + \frac{1}{3.3\tau s}\right] \tag{3.3}$$

3.2 Adaptive Control Theory

The designation "Adaptive Control" is an umbrella for an array of algorithms including Self-Oscillating Adaptive Systems (SOASs), Gain Scheduling, Auto Tuning, Model-Reference Adaptive Control (MRAC) and Self-Tuning Regulators (STRs) [2], to name but a few. Although the adaptive schemes are different, all adjust the controller in an attempt to conform with changing system parameters.

Interest in adaptive control theory began in the early 1950s with the need to improve the response of autopilots in high performance aircraft [2,17]. As the flight envelope for aircraft expanded, conventional, constant-gain control systems that might work well in one region would be unsatisfactory in other regions. The need for a more flexible control scheme provided an impetus for adaptive control; however, early applications were not successful. Failure occurred for two reasons: (1) a lack of progress in hardware required to implement the algorithms and (2) a lack of a comprehensive theory in adaptive control [2,17].





F'gure 3.1: Step response test for Ziegler-Nichols tuning

In the 1960s, space programs in both the United States and the Soviet Union focused many resources on the development of control systems for tracking and guiding spacecraft. Many contributions to control theory, including advances in system identification, parameter estimation, state-space theory, and stability theory, provided the basis for subsequent successful adaptive algorithms.

Interest in adaptive control surfaced again in the 1970s, although early applications were limited in scope. In the late 1970s and early 1980s, progress in stability theory [e.g., 1] and a microelectronics revolution allowed for several successful applications of adaptive control [2]. Today, the field of adaptive control is by no means a complete or even a well developed discipline. Although much progress has been made, many advances are currently being made in both the commercial and university sectors.

Over the past 30 years, many adaptive controllers have been developed and many have lent themselves well to the field of process control; indeed, most successful applications of adaptive control have come about in this area [17]. Of the wide variety of adaptive algorithms, gain scheduling, model-reference adaptive control, and self-tuning regulators (controllers) have received the most attention [17]. To date, though, self-tuning control methods have far outstripped both gain scheduling and model reference methods in process control [17].

This study examines the performance of a self-tuning controller on a fluidized bed combustor. The controller, as described by Åström and Wittenmark [2] and Goodwin and Sin [13], consists of three components: (1) a recursive least-squares parameter identification procedure, (2) an observer, and (3) a linear quadratic Gaussian (LQG)

optimal-control design procedure. LQG control design was chosen because closed loop stability is guaranteed, provided the system is either (a) uniformly completely controllable and uniformly completely reconstructible or (b) exponentially stable [24]. A schematic of a self-tuning controller is shown in Fig. 3.2. Note that the controller consists of two loops: an outer feedback loop and an inner controller adjustment loop. An overview of the self-tuning controller algorithm is presented for a second-order model; extensions to higher-order models are straightforward.

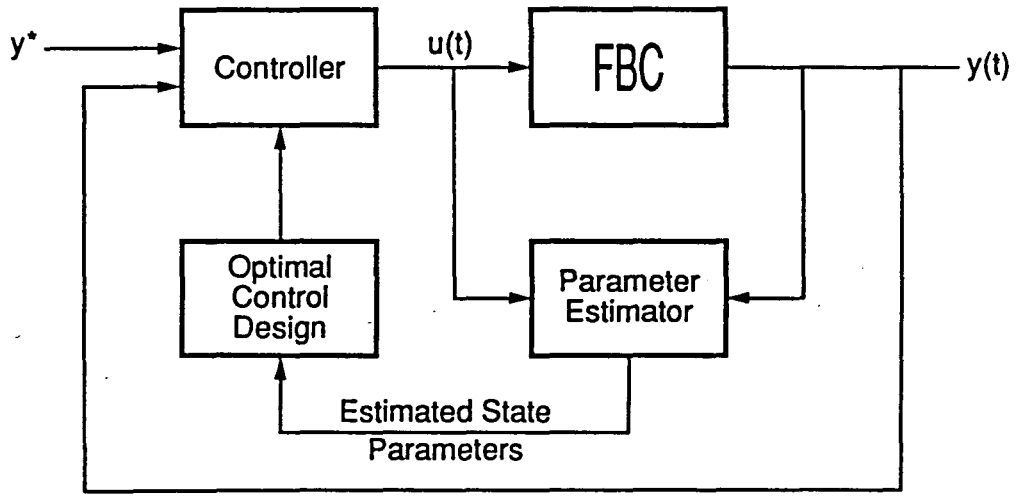


Figure 3.2: Block diagram of an adaptive controller

3.2.1 System Model

In the discrete-time domain, systems are described through a difference operator representation. The forward shift operator and backward shift operator are written as q and q^{-1}, respectively. For a function $y(t)$, where t is a sequential time index, $qy(t)$ references the function y at time $(t+1)$; similarly, $q^{-1}y(t)$ references the function y

at time $(t-1)$. In general,

$$\begin{aligned} q^i y(t) &= y(t+i), \qquad t \geq 0 \\ q^{-i} y(t) &= y(t-i), \qquad t \geq i \end{aligned} \tag{3.4}$$

and

$$q^{-i} y(t) = 0 \qquad\qquad 0 < t < i$$

In terms of addition and multiplication, the shift operator, with constant coefficients, satisfies all the algebraic laws of polynomials.

Using left difference operator representation, systems are modeled through linear combinations of past outputs, $y(t)$, and past inputs, $u(t)$. Following the notation of Goodwin and Sin [13], systems are expressed in the discrete-time domain through a deterministic autoregressive moving-average (DARMA) model of the form

$$\begin{aligned} A(q^{-1})y(t) &= B(q^{-1})u(t) + d \\ t &\in [0,1,2,3,\ldots] \end{aligned} \tag{3.5}$$

where A and B are the scalar polynomials

$$\begin{aligned} A(q^{-1}) &= 1 + a_1 q^{-1} + \ldots + a_n q^{-n} \\ B(q^{-1}) &= b_1 q^{-1} + \ldots + b_m q^{-m} \end{aligned}$$

and d is an offset parameter. Previous values of y are autoregressive components whereas previous values of u are moving-average components. The qualifier "deterministic" has been introduced to suggest an input signal that is not a white noise

process. For a second-order system, Eq. (3.5) may be expanded and rearranged to obtain

$$\begin{aligned} y(t) &= -a_1 y(t-1) - a_2 y(t-2) + b_1 u(t-1) + b_2 u(t-2) + d \qquad (3.6) \\ t &\in [0,1,2,3,\ldots] \end{aligned}$$

where, for the FBC, $y(t)$ denotes a deviation central bed temperature and $u(t)$ denotes a deviation annular bed air flow rate. Deviations were taken about initial conditions, e.g., $y(t) = \dot{T}(t) - T(t_o)$, to enforce zero initial conditions. The offset parameter d has been incorporated in the DARMA model to account for nonsteady-state initial conditions.

Alternatively, the DARMA model may be written in state-space form. Using left difference operator representation, the DARMA model may be described through the set of linear difference equations

$$\begin{bmatrix} x_1(t+1) \\ x_2(t+1) \end{bmatrix} = \begin{bmatrix} -a_1 & 1 \\ -a_2 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} u(t) + \begin{bmatrix} d \\ 0 \end{bmatrix} \qquad (3.7)$$

in which $y(t)$ is described through an output equation, a linear combination of state variables $x_1(t)$ and $x_2(t)$. For this case, the output equation takes the form

$$y(t) = \begin{bmatrix} 1, & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} \qquad (3.8)$$

This particular state variable representation (SVR) is in an observer form since the system is uniformly completely observable [13]. In matrix notation, Eqs. (3.7) and (3.8) can be abbreviated as

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) + D \qquad (3.9) \\ y(t) &= Cx(t) \end{aligned}$$

where

$$A = \begin{bmatrix} -a_1 & 1 \\ -a_2 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \qquad D = \begin{bmatrix} d \\ 0 \end{bmatrix} \tag{3.10}$$

$$C = \begin{bmatrix} 1, & 0 \end{bmatrix}$$

$$x(t) = \begin{bmatrix} x_1(t), & x_2(t) \end{bmatrix}^T$$

Additional considerations concerning SVRs, including discussions pertaining to necessary and sufficient conditions for system controllability and observability, are presented by Goodwin and Sin [13] and Kwakernaak and Sivan [24].

3.2.2 System Identification

The coefficients of the DARMA model — a_1, a_2, b_1, b_2, and d — may be estimated through off-line identification procedures, such as least-squares [2]. However, for a fluidized bed combustor, the coefficients of the model tend to drift since the model is a linear approximation of the nonlinear system about the current operating point. In other words, as the temperature changes, the coefficients of the DARMA model also change. In addition, coefficients from one run may not match the coefficients of another run, especially if different types of fuel or bed material are used between runs. Hence, better coefficient estimates may be obtained through on-line procedures, such as a recursive least-squares algorithm with exponential data weighting.

The method of least-squares is built upon dividing the DARMA model between variates and coefficients in the manner

$$y(t) = \phi(t-1)^T \theta \tag{3.11}$$

where

$$\phi(t-1) = \begin{bmatrix} -y(t-1), & -y(t-2), & u(t-1), & u(t-2), & 1 \end{bmatrix}^T \tag{3.12}$$

and

$$\theta = \begin{bmatrix} a_1, & a_2, & b_1, & b_2, & d \end{bmatrix}^T \tag{3.13}$$

In keeping with the notation presented by Åström and Wittenmark [2] and Goodwin and Sin [13], ϕ has been indexed with respect to the most recent component of the variate vector, $y(t-1)$. Accordingly, for t observations, the DARMA model may be collectively written as

$$\begin{bmatrix} y(1) \\ y(2) \\ y(3) \\ \vdots \\ y(t) \end{bmatrix} = \begin{bmatrix} \phi(0)^T \\ \phi(1)^T \\ \phi(2)^T \\ \vdots \\ \phi(t-1)^T \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ b_1 \\ b_2 \\ d \end{bmatrix} + \begin{bmatrix} e(1) \\ e(2) \\ e(3) \\ \vdots \\ e(t) \end{bmatrix} \tag{3.14}$$

$$Y = \Phi \times \theta + e$$

wherein Φ is the "model matrix" [8], and e is a column vector of residuals, defined as the difference between the actual output Y and the predicted output $\hat{Y}$. By minimizing the residual sum of squares, J, where

$$\begin{aligned} J &= \sum_{i=1}^{t} e(i)^2 \\ &= \sum_{i=1}^{t} (y(i) - \phi(i-1)^T \hat{\theta})^2 \end{aligned} \tag{3.15}$$

best estimates of the partial regression coefficients can be established. The minimal of J is found through an elementary application of matrix calculus [3] from which the

normal equations

$$\Phi^T \Phi \hat{\theta} = \Phi^T Y \tag{3.16}$$

are obtained. Provided the nonnegative, symmetric matrix $\Phi^T \Phi$ is nonsingular, the solution of Eq. (3.16) is simply

$$\hat{\theta} = [\Phi^T \Phi]^{-1} \Phi^T Y \tag{3.17}$$

Of particular interest is the relationship between $[\Phi^T \Phi]^{-1}$ and partial regression coefficient variances and covariances. For the regression vector $\theta = [\theta_1, \theta_2, \theta_3, \ldots]^T$, covariance of θ_i and θ_j is defined as

$$CV(\theta_i, \theta_j) = E\{(\hat{\theta}_i - \theta_i)(\hat{\theta}_j - \theta_j)\} \qquad \forall i \neq j \tag{3.18}$$

When i = j, Eq. (3.18) is simply an expression for the variance of θ_i. On the assumption that residuals are normally and independently distributed about a mean of zero and variance of σ^2, variances and covariances of the partial regression coefficients can be calculated from

$$E\{[\hat{\theta} - \theta][\hat{\theta} - \theta]^T\} = \sigma^2 [\Phi^T \Phi]^{-1} \tag{3.19}$$

Additional considerations concerning the covariance matrix, along with a derivation of Eq. (3.19), are presented by Cox [8].

Because large quantities of data are recorded, calculations for $[\Phi^T \Phi]^{-1}$ become prohibitively cumbersome. Consequently, to reduce computation time, standard least-squares estimates are expressed recursively so that new results for θ and $[\Phi^T \Phi]^{-1}$ are obtained from old results, corrected for new data. For the single input, single output (SISO) case, least-squares estimates are calculated through the sequential

equations [17,25]

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \frac{P(t-2)\phi(t-1)}{1+\phi(t-1)^T P(t-2)\phi(t-1)}[y(t) - \phi(t-1)^T\hat{\theta}(t-1)] \quad (3.20)$$

in which

$$P(t-1) = P(t-2) - \frac{P(t-2)\phi(t-1)\phi(t-1)^T P(t-2)}{1+\phi(t-1)^T P(t-2)\phi(t-1)} \quad (3.21)$$

For brevity, $[\Phi^T\Phi]^{-1}$ is denoted by P, where P has been indexed with respect to the most recent set of data, $\phi(t-1)$. With these equations, recursive least-squares gives equal weight to all sampled data. However, since coefficients may change over time, old data, and corresponding old coefficients, diminish the ability of the algorithm to track new, changing coefficients. Therefore, on the assumption that the most recent data are the most informative, old data are discarded through an exponential "forgetting" factor. Such exponential discarding is accomplished by assigning the most recent data a unit weight and assigning data received n samples ago a weight proportional to λ^n for $0.00 \ll \lambda \leq 1.00$ [2,17,25]. With an exponential forgetting factor, Eqs. (3.20) and (3.21) are modified slightly [17,25]:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \frac{P(t-2)\phi(t-1)}{\lambda(t-1)+\phi(t-1)^T P(t-2)\phi(t-1)}[y(t) - \phi(t-1)^T\hat{\theta}(t-1)] \quad (3.22)$$

in which

$$P(t-1) = \frac{1}{\lambda(t-1)}\left[P(t-2) - \frac{P(t-2)\phi(t-1)\phi(t-1)^T P(t-2)}{\lambda(t-1)+\phi(t-1)^T P(t-2)\phi(t-1)}\right] \quad (3.23)$$

Typically, values for λ range from 0.95 to 1.00, where a value of 1.00 retains all data. Caution must be exercised when specifying λ. If λ is much less than 1.00, old data are discarded too quickly, and system noise becomes a problem. In addition, exponential

forgetting may cause difficulties in parameter estimation if the system reaches steady-state. Since steady-state data are deficient in information and old, information-rich data are discarded, P diverges as a function of λ^{-1}. If P has grown sufficiently large and new information or observation noise is suddenly introduced, a "burst phenomenon" results, which produces deceptive parameter estimates. The increase in P due to a lack of persistently exiting (PE) conditions is known as estimator wind-up.

Several attempts have been made to keep P bounded and prevent deterioration of the parameter estimation algorithm. Ljung and Söderström [25] propose that λ be allowed to vary as a function of time so that it asymptotically approaches 1.00 from some specified initial value. For example, a time dependent λ described by the function

$$\lambda(t) = \lambda_o \lambda(t-1) + (1-\lambda_o) \tag{3.24}$$

$$n.b., \qquad \lim_{t\to\infty} \lambda(t) = 1.00$$

imposes exponential data weighting during the initial tuning period. However, adjusting λ does not guarantee a bounded P. Other methods, which do guarantee a bounded P, include termination of the parameter estimation algorithm after initial convergence, enforcing a constant trace of P, or forgetting information only when new information is available [2].

This study takes cognizance of limitations associated with exponential forgetting, but the dynamics of an FBC are such that the least-squares algorithm is nearly always sufficiently excited, making a constant forgetting factor appropriate. For the combustor used in this investigation, a constant forgetting factor of 0.99 was found to be satisfactory.

In addition to weighting data, consideration must be given to initial values for θ and P. Initial values for θ were obtained through an off-line analysis of an early test run. Initial parameter coefficients were specified as

$$\hat{\theta}(0) = \begin{bmatrix} -1.19, & 0.228, & -0.05, & -0.30, & 1.4 \end{bmatrix}^T \tag{3.25}$$

The P matrix, on the other hand, was initially set to

$$P(0) = \begin{bmatrix} 15 & 0 & 0 & 0 & 0 \\ 0 & 15 & 0 & 0 & 0 \\ 0 & 0 & 15 & 0 & 0 \\ 0 & 0 & 0 & 15 & 0 \\ 0 & 0 & 0 & 0 & 15 \end{bmatrix} \tag{3.26}$$

Large values were specified along the diagonal to reflect the uncertainty of the off-line estimates. Because the initial P matrix was set so large, initial convergence dynamics often resulted in large, spurious variations in parameter estimates. Difficulties stemmed from convergence problems associated with the characteristic equation

$$\begin{aligned} A(q^{-1})y(t) &= 0; \\ A(q^{-1}) &= (1 + a_1 q^{-1} + a_2 q^{-2}) \end{aligned} \tag{3.27}$$

Occasionally, $A(q^{-1})$ was estimated as unstable (i.e., roots outside the complex unit disk). To prevent such results, Eq. (3.27) was analyzed with Jury's stability test (Appendix A). If the roots were estimated as unstable, θ was not updated. By using Jury's stability test, parameters converged quickly and smoothly as absurd estimates were discarded. Typically, $A(q^{-1})$ was estimated as unstable two to fifteen times during initial parameter convergence. Further discussions concerning system

stability, including a statistical analysis of the temperature response, are presented in Section 5.1.

3.2.3 System Reconstruction

The underlying assumption regarding the state-space model of Subsection 2.3.1 is that the state vector $x(t)$ is completely known or measurable. In this case, the state is partially measurable insofar as the central bed temperature, $x_1(t)$, is measurable. In contrast, the state variable $x_2(t)$ is a dummy parameter — created for convenience — and cannot be directly measured. Hence, the state vector must be approximated (i.e., reconstructed) through a function of the observed variable, $x_1(t)$.

The state vector may be approximated through a full-order observer, a dynamical system whose output approaches the state to be reconstructed. Illustrated in Fig. 3.3, a full-order observer may be expressed through the equation [13,14,18,22,24]

$$\begin{aligned} \hat{x}(t+1) &= A\hat{x}(t) + Bu(t) + D + K[y(t) - C\hat{x}(t)] \\ K &= \begin{bmatrix} k_1, & k_2 \end{bmatrix}^T \end{aligned} \tag{3.28}$$

For adaptive control, SVR observer coefficients are replaced by estimates obtained from the recursive least-squares algorithm. That is,

$$A = \begin{bmatrix} -\hat{a}_1 & 1 \\ -\hat{a}_2 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} \hat{b}_1 \\ \hat{b}_2 \end{bmatrix} \qquad D = \begin{bmatrix} \hat{d} \\ 0 \end{bmatrix} \tag{3.29}$$

Dynamics of the observer are characterized by a difference equation of the reconstruction error [24]

$$\begin{aligned} e(t+1) &= (A - KC)e(t); \\ e(t) &= x(t) - \hat{x}(t) \end{aligned} \tag{3.30}$$

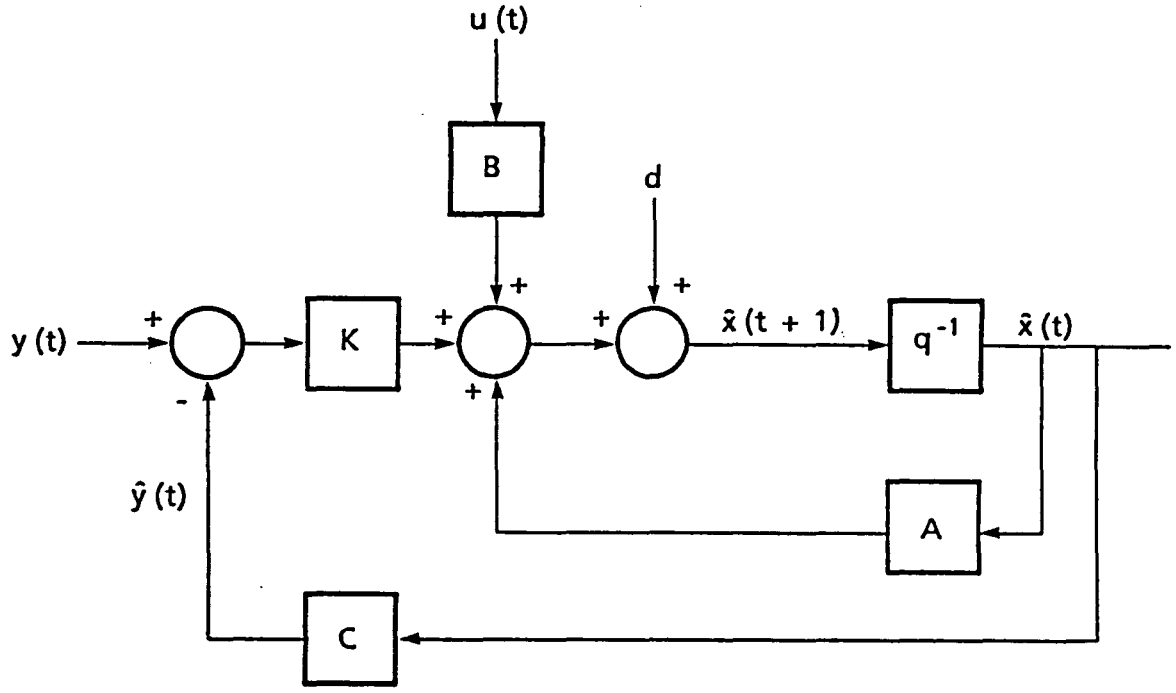


Figure 3.3: Block diagram of a full-order observer

Observer eigenvalues (poles) are specified through an appropriate choice of values for the gain vector K. The gain vector may be specified using a pole assignment technique in which the difference equation is first transformed to a canonical form [13]. The gain vector is then specified such that the coefficients of the canonical characteristic equation match the coefficients of a desired characteristic equation. However, for low-order systems, no transformation is required; the gain vector can be directly specified by matching appropriate coefficients of the system characteristic equation with coefficients of a desired characteristic equation. For observer design, eigenvalues are placed such that the reconstruction error is asymptotically stable and approaches zero for large time [24]. The characteristic equation of the reconstruction

error can be found by evaluating

$$det[qI - A + KC] = 0 \tag{3.31}$$

Collecting like coefficients, the determinant for a second-order system reduces to the polynomial

$$q^2 + (\hat{a}_1 + k_1)q + (\hat{a}_2 + k_2) = 0 \tag{3.32}$$

For stability, poles must be placed within the interior of the unit disk on the complex plane. To enable the observer to track system performance, observer dynamics should be faster than closed-loop system dynamics but not so fast that system noise becomes a problem. Therefore, observer poles should be placed just to the left of closed-loop system poles. Off-line analysis and simulation of adaptive control performance on the FBC suggested that both observer poles placed at 0.3 would effect adequate observer response. With these eigenvalues, the desired characteristic equation of the observer is simply

$$q^2 - 0.6q + 0.09 = 0 \tag{3.33}$$

Comparison of Eq. (3.32) with Eq. (3.33) requires that observer gains satisfy the set of equations

$$\begin{aligned} \hat{a}_1 + k_1 &= -0.6 \\ \hat{a}_2 + k_2 &= 0.09 \end{aligned} \tag{3.34}$$

As a final note, for systems with an order much greater than two, calculations pertaining to full-order observers become burdensome. In such cases, the state vector can be reconstructed through a reduced-order observer. Theory and dynamics of

reduced-order observers are discussed by Kwakernaak and Sivan [24] and are mentioned here for completeness.

3.2.4 Linear Quadratic Gaussian Optimal Control

The estimated state matrix, along with the corresponding state observer, may be incorporated directly into an LQG control design, which is a regulator form of optimal-control. Since a variable set point is often required, the regulator problem is converted to a tracking problem through an integral action. Integral action was chosen over precompensation to enforce zero steady-state error conditions. The conversion to a tracking problem is accomplished by augmenting the SVR with a function $z(t)$, which integrates the difference between the set point $y^*(t)$ and the system output $y(t)$. The integral (i.e., sum) of the differences

$$z(t) = \sum_{i=0}^{t-1} (y^*(i) - y(i)) \tag{3.35}$$

can be expressed recursively as

$$\begin{aligned} z(t+1) &= z(t) + y^*(t) - y(t); \\ z(0) &= 0 \end{aligned} \tag{3.36}$$

By augmenting Eqs. (3.7) and (3.8) with Eq. (3.36), the new, third-order state can be written as [13,34]

$$\begin{bmatrix} x(t+1) \\ z(t+1) \end{bmatrix} = \begin{bmatrix} A & 0 \\ -C & I \end{bmatrix} \begin{bmatrix} x(t) \\ z(t) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u(t) + \begin{bmatrix} D \\ y^* \end{bmatrix} \tag{3.37}$$

$$\begin{bmatrix} y(t) \\ z(t) \end{bmatrix} = \begin{bmatrix} C & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} x(t) \\ z(t) \end{bmatrix} \tag{3.38}$$

In matrix notation,

$$X(t+1) = A_{aug}X(t) + B_{aug}u(t) + D_{aug} \tag{3.39}$$

$$Y(t) = C_{aug}X(t) \tag{3.40}$$

wherein $X(t)$ is the column vector $[x_1(t), x_2(t), z(t)]^T$ and $Y(t)$ is the column vector $[y(t), z(t)]^T$. A block diagram of the augmented state-space model is shown in Fig. 3.4.

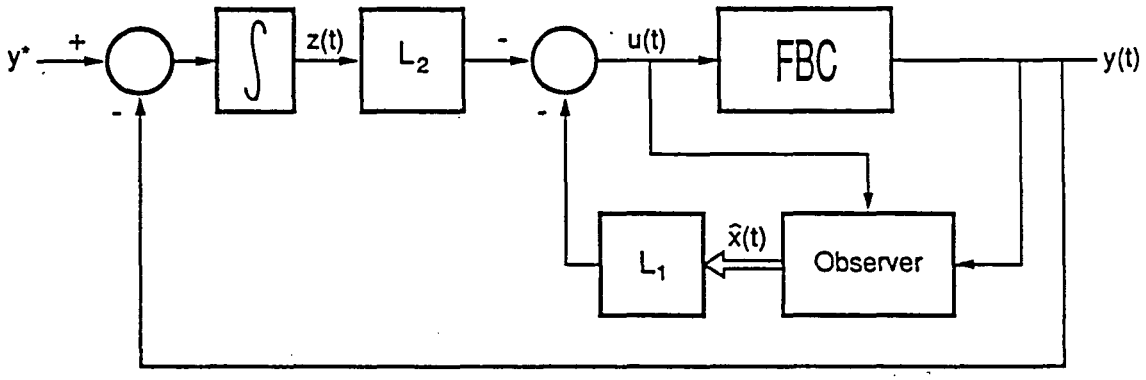


Figure 3.4: Block diagram of augmented state feedback control

Feedback gains for the augmented state, $L(t)$, are obtained from a third-order LQG algorithm, which minimizes the performance index

$$J = E\left\{X(N)^T R_N X(N) + \sum_{t=0}^{N-1}\left(X(t)^T R_x X(t) + u(t)^T R_u u(t)\right)\right\} \tag{3.41}$$

subject to

$$X(t+1) = A_{aug}X(t) + B_{aug}u(t) + D_{aug} + v_1$$

$$Y(t) = C_{aug}X(t) + v_2$$

$$u(t) = -L(t)X(t)$$

$$X(t_o) = X_o$$

where v_1 and v_2 are system and measurement disturbances, assumed to be normally and independently distributed about a mean of zero and covariance of

$$E\left\{\begin{bmatrix} v_1(t) \\ v_2(t) \end{bmatrix}\begin{bmatrix} v_1^T(r) & v_2^T(r) \end{bmatrix}\right\} = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}\delta(t-r) \tag{3.42}$$

The feedback gain vector, $L(t)$, is optimized by minimizing Eq. (3.41), which is weighted by the symmetric, positive definite matrices R_x, R_u, and R_N. Solution of the LQG optimal-control problem requires [13]

$$L(t) = [R_u + B_{aug}^T S(t+1) B_{aug}]^{-1} B_{aug}^T S(t+1) A_{aug} \tag{3.43}$$

where $S(t)$ satisfies the matrix Riccati equation

$$S(t) = R_x + L(t)^T R_u L(t) + \left(A_{aug} - B_{aug}L(t)\right)^T S(t+1)\left(A_{aug} - B_{aug}L(t)\right) \tag{3.44}$$

A steady-state solution of the feedback gain vector, $\bar{L}$, is obtained by iterating Eqs. (3.43) and (3.44) backwards in time. Subject to weak assumptions [13,24], $L(t)$ converges to $\bar{L}$; convergence is usually achieved within five to ten iterations. Note that, because the optimization horizon is made sufficiently large, steady-state feedback gains are independent of the terminal condition $X(N)^T R_N X(N)$. Finally, by replacing the state vector X(t) with state estimates obtained from the observer, the secondary air flow rate can be calculated from the equation

$$u(t) = -\bar{L}\begin{bmatrix} \hat{x}_1, & \hat{x}_2, & z(t) \end{bmatrix}^T \tag{3.45}$$

4. EXPERIMENTAL PROCESS DESCRIPTION

The purpose of this investigation was to use an adaptive control algorithm to regulate the central bed temperature of a two-bed fluidized combustor. As a reference, the performance of the adaptive control algorithm was compared with that of a classical PI controller tuned by the Ziegler-Nichols method. In both cases, bed temperatures were measured with thermocouples and secondary air flow rates were adjusted accordingly. Because a SISO control objective was specified, primary air flow and coal feed rates were maintained at constant values during combustor operation.

This chapter describes the fluidized bed system and accompanying computer hardware, start-up procedures, sampling time specifications as well as statistical procedures used to determine DARMA model adequacy. The chapter concludes with a presentation of data used to determine Z/N coefficients for the PI control algorithm.

4.1 Combustor Design

A process diagram of the fluidized bed system is shown in Fig. 4.1. The central and the annular fluidized beds are separated by a $\frac{1}{8}$-in thick stainless steel insert, which is welded to a common distributor plate. The gas distributor is a simple multiorifice plate constructed from $\frac{1}{2}$-in thick stainless steel stock. The plate has 250 evenly spaced orifices, each with a diameter of $\frac{3}{32}$-in. A large number of orifices

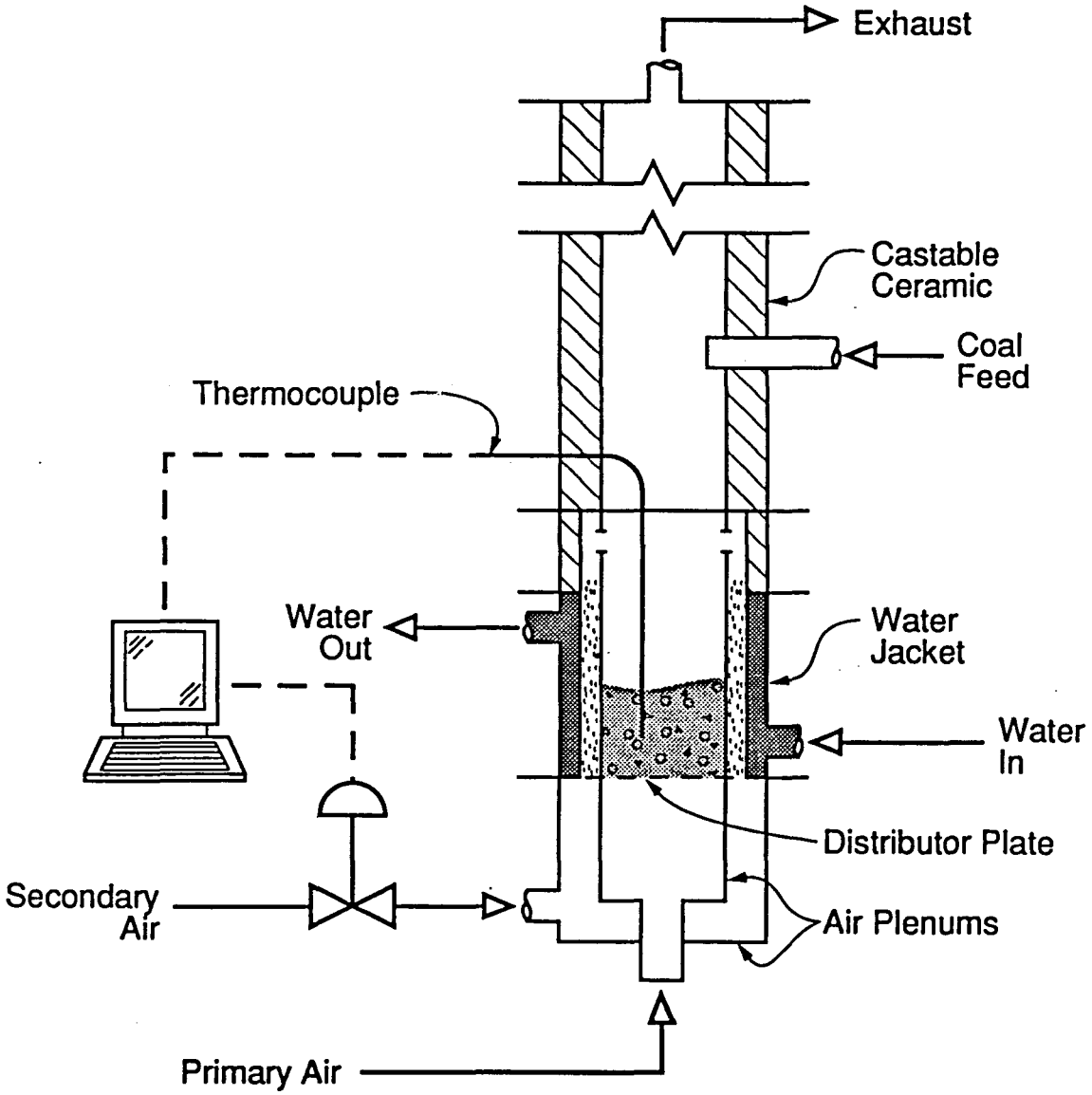


Figure 4.1: Piping and instrumentation diagram of test rig

prevent gas channeling and provide for uniform fluidizing characteristics within the bed. An 80 mesh stainless steel screen is spot-welded to the upper surface of the distributor plate to restrict backflow of sand into the air plenums. In addition to retaining sand, the screen functions as flame arrestor [23], preventing unaided flame propagation of liquefied petroleum gas (LPG) into the primary air supply line during the start-up procedure.

Air supplied to the annular bed is exhausted through nozzles to the combustor freeboard. Four $\frac{1}{2}$-in nozzles are evenly spaced on the dividing wall circumference, and a stainless steel screen is placed across each nozzle to prohibit effusion of annular bed sand into the central bed. These nozzles are located in a manner such that annular bed air is directed into the freeboard region, thereby promoting combustion of unreacted gases and coal fines released from the central bed [10].

A freeboard constructed of mild steel is placed above the combustion and heat transfer beds. The inner wall of the freeboard is insulated with a 1-in layer of Kao-cast RFT castable refractory. From the freeboard, flue gases pass through a high-efficiency cyclone where entrained particles are removed. Finally, combustion gases are exhausted from the fluidized bed system through a roof mounted, induced draft fan.

4.2 Fuel Supply

Experiments were performed using bituminous coals of variable quality and composition, including Illinois No. 6 and Iowa "Cherokee" seam coals as well as various weathered Iowa coals. Neither proximate nor ultimate analyses were performed to

characterize the coals as such information was not required for proper controller performance.

Coal was crushed and screened to a top size of $\frac{3}{8}$-in, meeting feeder clearance requirements, and screened to a bottom size of $\frac{1}{8}$-in, thereby reducing the quantity of elutriable fines in the fuel supply. The crushed coal was metered into the fluidized bed by an AccuRate dry chemical feeder (Model 602), equipped with a $\frac{3}{4}$-in diameter, variable speed helix and a 5 gallon vinyl hopper. The feeder held approximately 30 lb of coal and could accurately meter fuel into the bed at 1 to 50 lb/hr. To prevent hot combustion gases from circulating through the fuel supply and into the laboratory, the hopper was sealed during test runs. However, a 30 lb sealed fuel supply restricted combustor operation to a two hour maximum.

Occasionally, the coal supply — notably, the weathered Iowa coal — had a high moisture content and would create metering difficulties. Water drawn out of the coal by the hot helix formed a sticky amalgamate that could not be feed into the bed. Consequently, to remove water from the coal, high moisture coals were allowed to dry in the open for 24 hours.

4.3 Bed Material

Experiments were performed on a two-bed combustor using fluidized beds of sand. The central bed had a static height of 6 inches and contained a general purpose sand, sieved to a 16×20 mesh particle distribution. Heat was removed from the central bed by an annular, fluidized heat transfer bed with a static height of 9 inches. The outer bed contained a fine silica sand, sieved to a 50×70 mesh particle distribution.

To ensure effective heat transfer, the annular bed sand height was kept above the central bed sand height. The converse — a central bed with a sand height greater than that of the annular bed — reduces the effectiveness of the heat transfer bed, making temperature control of the central bed difficult.

4.4 Data Acquisition and Digital Control Hardware

Temperature and flow rate data were acquired and digital control was executed with a Zenith Z-158 microcomputer. The Z-158 is based on an Intel 8088 CPU at 4.77 Mhz. The unit was configured with an Intel 8087 math coprocessor, 640 K of addressable memory, and a 20 Mb hard drive. The data interface consisted of a Metrabyte DAS-8 A/D converter and a Metrabyte DDA-06 D/A converter. The DAS-8 is an 8-channel, 12-bit, successive approximation A/D converter with conversion times of 25 microseconds. The DDA-06 is a 6-channel, 12-bit analog output interface with 24 parallel digital I/O lines. Both data acquisition and digital control codes were written and compiled with Microsoft QuickBASIC 4.0.

Temperature data were obtained from three type-K (chromel/alumel) thermocouple probes in the central bed, and one type-K thermocouple probe in the annular bed. To protect them from the abrasive bed environment, the thermocouples were enshrouded in a 304 stainless steel casing. The probes were connected to a Metrabyte sub-multiplexer board (Model EXP-16), which amplified thermoelectric voltages and provided cold junction compensation. Analog signals from the sub-multiplexer board were sent to the DAS-8 A/D interface board.

Both primary and secondary air flow rates were calculated from pressure drops

across orifice flow meters. Pressure drops across the meters were measured with a Schaevitz LVDT pressure transducer (Model P3061) and were calibrated against a laminar flow meter. Analog signals from the pressure transducer were sent to the DAS-8 A/D converter, whereupon air flow rates were interpolated from the calibrated pressure drop data.

Air flow rates into the fluidized bed were adjusted by a Fisher Design GS valve ($\frac{3}{8}$-in port) with a Fisher Type 513R reversible diaphragm actuator. Plumbed to a 60 psig air supply line, the Design GS valve could admit a maximum air flow rate of 50 scfm into the annular bed. However, a 27 scfm saturation limit was set within the digital control programs as high flow rates and concomitant high fluidization velocities promoted elutriation of bed material and fuel from the combustor. The Fisher valve assembly was regulated by a Bellofram Type 1000 E/P Transducer, which was connected to the DDA-06 analog output interface. Considerable difficulties were encountered with the Fisher valve assembly due to packing friction and flow forces on the valve plug. To improve valve performance, a position feedback control loop was installed on the system. Details of controller hardware and valve performance are presented in Appendix B.

4.5 System Start-Up

Primary air flow was set and maintained at 20 scfm throughout the entire test run. During start-up, secondary air flow was set and maintained at 4 scfm. A nominal secondary air flow rate was maintained during system start-up to fluidize the annular bed and allow the stainless steel insert to expand in the transverse direction. After

start-up, a lower limit of 4 scfm was enforced on secondary air as flow rates below 4 scfm were neither accurate nor repeatable (see Appendix B).

Before coal could be metered into the combustor, the bed temperature had to be raised, by external means, to a temperature where coal combustion was autogeneous. To do this, the bed was preheated to 1200 °F by burning LPG. The gas was introduced into the primary air plenum and fed into the bed along with combustion air. To ignite the LPG, energized electrodes were placed above the combustion bed. Although LPG flames are self-propagating, the electrodes were charged during the entire start-up procedure to re-ignite flames which may have been blown off, thereby prohibiting gas build-up in the combustor. When the combustor temperature reached 1200 °F, LPG was progressively decreased as coal was metered into the bed. Continuous and spontaneous combustion of coal was verified by visual inspection, at which time LPG into the unit was discontinued.

Coal feed rates were set and maintained at 12 lb/hr. Primary and secondary air flow rates were maintained at 20 and 4 scfm, respectively. Under these conditions, the bed was allowed to reach an arbitrary temperature, whereupon an adaptive or classical controller algorithm was invoked.

4.6 Sampling Period Specifications

Goodwin and Sin [13] suggest a control update time equal to one-fifth of the fastest time constant. The combustor used in this investigation had a dominant time constant of 200 seconds; however, a sampling interval of 20 seconds — or one-tenth of the dominant time constant — was specified. Sampling intervals much smaller

than 20 seconds resulted in parameter identification difficulties. Specifically, to avoid estimating system parameters on measurements predominated by noise, a sufficient change in temperature had to occur between data samples. On the other hand, sampling intervals much larger than 20 seconds resulted in poor regulation as bed temperatures tended to stray between control updates.

A separate sampling interval was specified for the inner digital control loop, which was used to adjust the Fisher control valve assembly. Within this loop, flow rate data were acquired from the pressure transducer and command signals were issued to the Bellofram pressure regulator at 150 millisecond intervals. In addition to pressure data, temperature data were acquired at the 150 millisecond intervals. To prevent spurious measurements from corrupting the parameter estimation procedure, temperature data were filtered. Filtering consisted of a comparison between two successive temperature measurements. If the difference between the measurements exceeded 5 $^{\circ}$F, the procedure was repeated. Averaging techniques were ineffective as false measurements were inclined to be several hundred degrees from the actual bed temperature. Filtered temperature data were continuously updated on the computer display so that system performance could be directly monitored; however, only the temperature data sent to the adaptive control or classical control algorithms at 20 second intervals were recorded.

4.7 Statistical Procedures

To determine a suitable system order for the DARMA model, data from sample runs were analyzed with the General Linear Models (GLM) procedure on SAS

(Statistical Analysis System). Statistical procedures included a multiple regression ANOVA (analysis of variance) in which F-tests were computed for both sequential (Type I) and partial (Type III) sums of squares. In the GLM procedure, a Type I analysis is a forward sequential examination of variances associated with the individual predictor variables. For this analysis, DARMA model variables were arranged by increasing model order, i.e., $y(t-1)$, $u(t-1)$, ... , $y(t-6)$, $u(t-6)$. A Type III variance partition is included in the ANOVA to assess the importance of including each variable in the statistical model [8,31]. The Type III variance partition tabulates sums of squares associated with individual predictor variables if those variables had been placed in the final position of the Type I sequence.

Parameter estimates were also calculated with the GLM procedure and significance was determined through t-tests. Finally, using the multiple regression procedure PROC REG on SAS, covariance and partial regression matrices were calculated and tabulated for the DARMA model estimates.

4.8 PI Control Design

A PI controller with gains specified by the Z/N method was used to reference the performance of adaptive control. To obtain Z/N coefficients, a step response was performed on the two-bed combustor. An Illinois No. 6 coal, crushed and screened to $\frac{3}{8} \times 8$ mesh, was fed into the combustor at a constant rate of 15 lb/hr; combustion air into the unit was held constant at 20 scfm. Annular bed air was stepped from 4.0 scfm to 10.0 scfm and then from 10.0 scfm to 15.0 scfm. The unfiltered temperature response is graphed in Fig. 4.2, and the corresponding air flow rate is graphed in

Fig. 4.3. From the step response data, the control law

$$\begin{aligned} u(t) &= -0.078\left[e(t) + \frac{1}{114}\int_0^t e(t)dt\right] \\ e(t) &= y^* - y(t) \end{aligned} \tag{4.1}$$

was implemented in the discrete-time domain where integration was approximated by the Trapezoidal Rule. Performance tests were conducted with sampling intervals set at 5 seconds and at 20 seconds. Temperature responses of the FBC under PI control are presented in Sections 5.3 and 5.4.

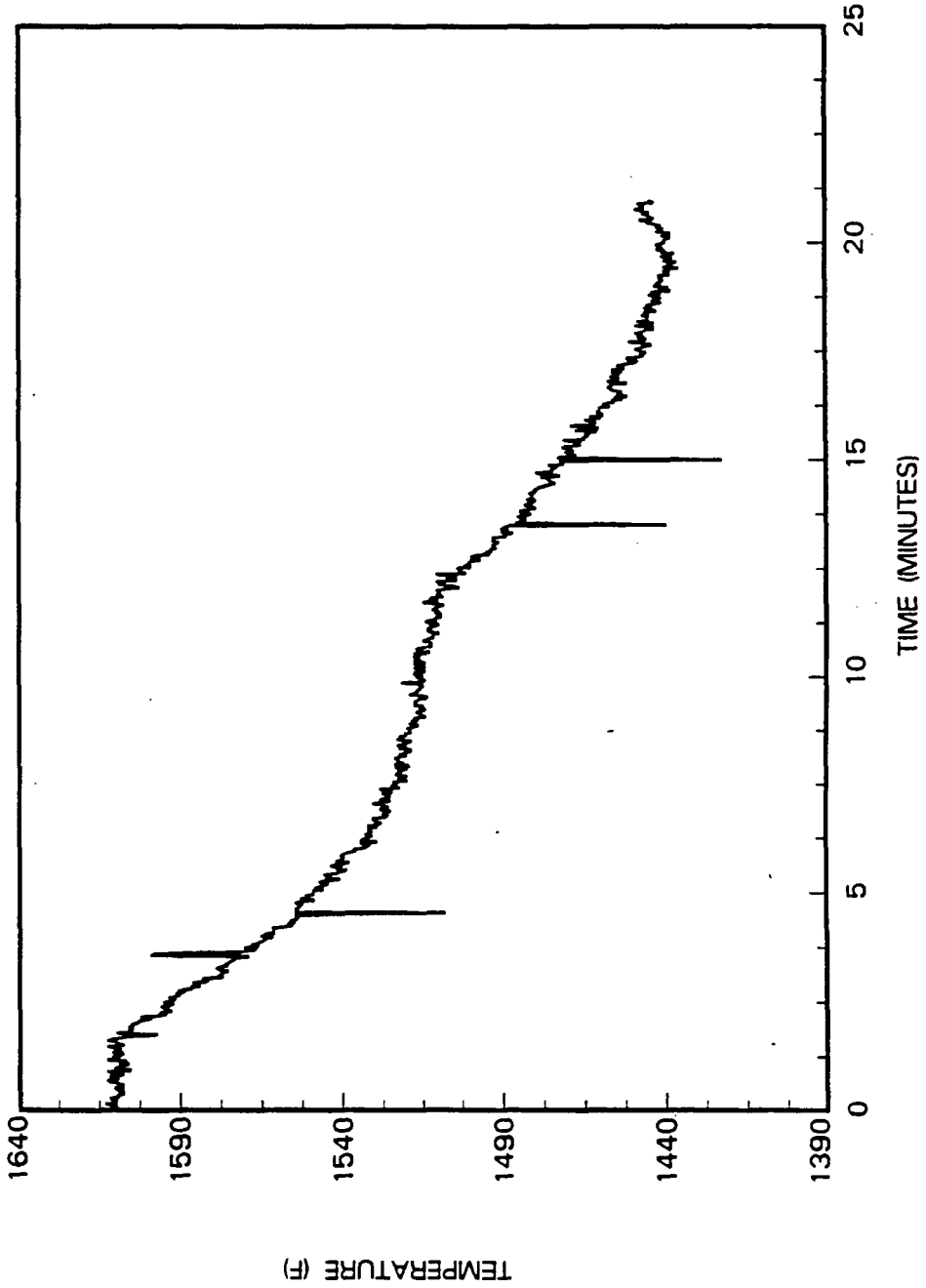


Figure 4.2: Temperature response of a two-bed fluidized combustor

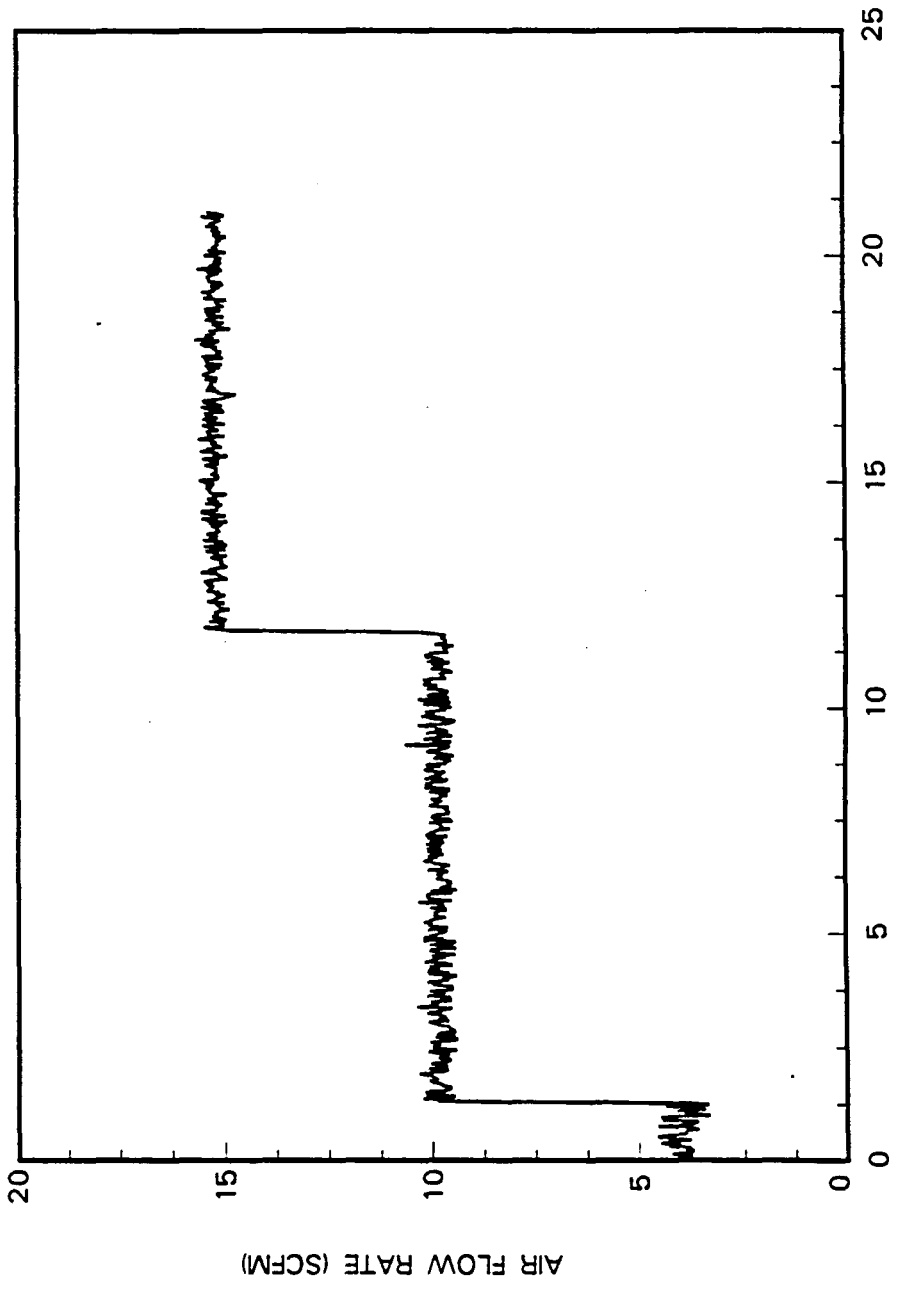


Figure 4.3: Step input of secondary air

5. RESULTS AND DISCUSSION

5.1 Off-Line Statistical Analysis of the DARMA Model

Temperature and air flow rate data used for the statistical analysis are graphed in Fig. 5.1 and Fig. 5.2, respectively. Deviation parameters were taken about an initial temperature of 1543 °F and an initial air flow rate of 3.46 scfm. A multiple regression ANOVA was performed with the data using a sixth-order DARMA model of the form

$$\begin{aligned} y(t) = \; & - \; a_1 y(t-1) - a_2 y(t-2) - \ldots - a_6 y(t-6) \\ & + \; b_1 u(t-1) + b_2 u(t-2) + \ldots + b_6 u(t-6) + d \end{aligned} \tag{5.1}$$

where

y denotes a deviation bed temperature

u denotes a deviation air flow rate

d is an offset parameter

The ANOVA and subsequent Type I and Type III variance partitions are presented in Table 5.1; complete results from the SAS GLM procedure are listed in Appendix C.

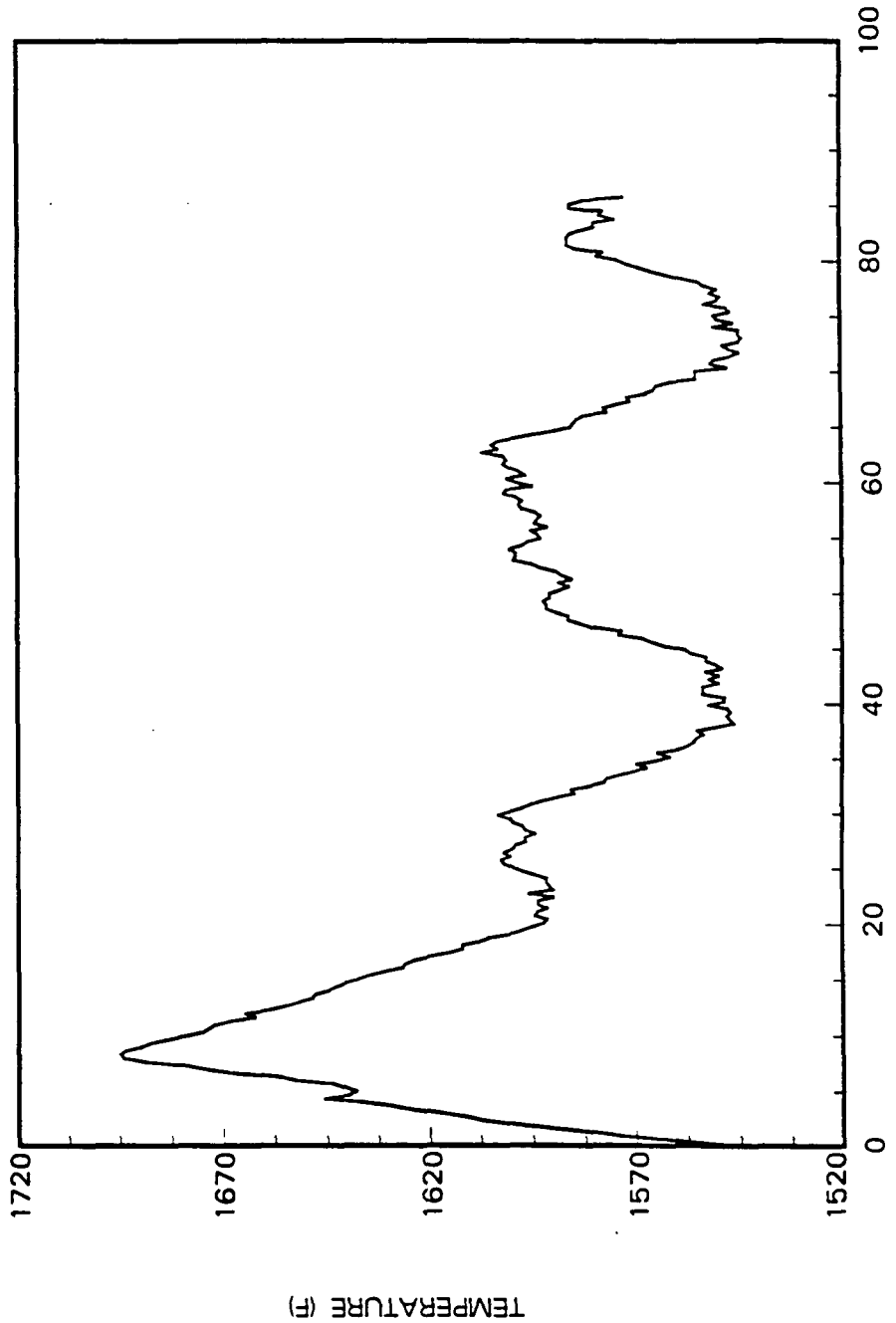


Figure 5.1: Temperature data for the ANOVA

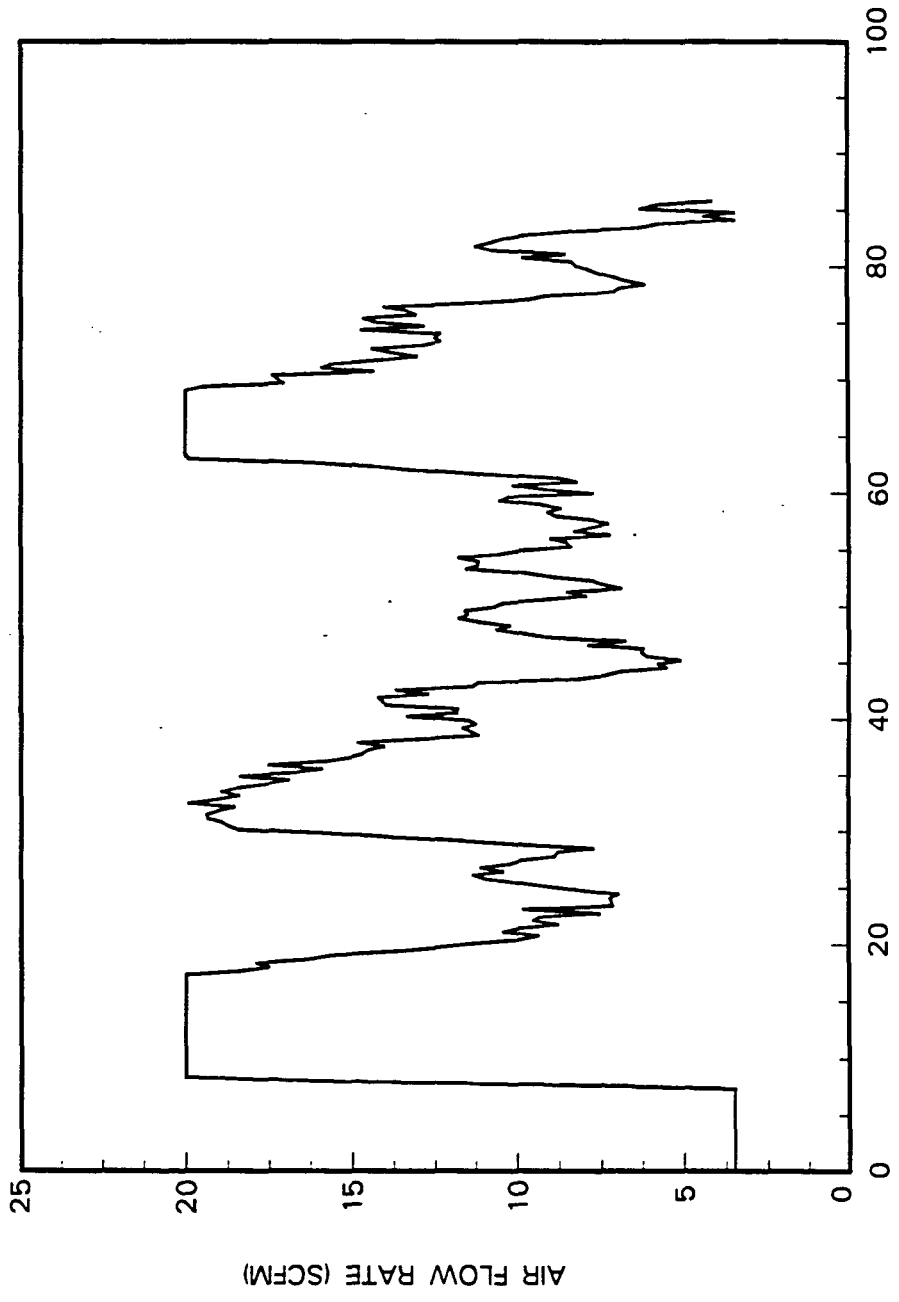


Figure 5.2: Secondary air flow rate data for the ANOVA

Table 5.1: Regression analysis, 6th-order DARMA model

DEPENDENT VARIABLE: y(t)

SOURCE	DF	SUM OF SQUARES	MEAN SQUARE	F VALUE	PR > F	R-SQUARE	C.V.
MODEL	12	307520.6765	25626.7230	3566.68	0.0	0.994539	5.4777
ERROR	235	1688.4840	7.1850	ROOT MSE	Y MEAN		
C. TOTAL	247	309209.1605		2.6804	48.9344		

SOURCE	DF	TYPE I SS	F VALUE	PR > F	DF	TYPE III SS	F VALUE	PR > F
$y(t-1)$	1	306329.7318	42634.39	0.0	1	1167.6077	162.51	0.0001
$u(t-1)$	1	914.0153	127.21	0.0001	1	12.4912	1.74	0.1886
$y(t-2)$	1	6.5827	0.92	0.3395	1	2.4967	0.35	0.5561
$u(t-2)$	1	92.0353	12.81	0.0004	1	46.7205	6.50	0.0114
$y(t-3)$	1	0.0737	0.01	0.9194	1	54.3870	7.57	0.0064
$u(t-3)$	1	20.3431	2.83	0.0938	1	2.3604	0.33	0.5671
$y(t-4)$	1	66.6105	9.27	0.0026	1	18.4149	2.56	0.1107
$u(t-4)$	1	0.2178	0.03	0.8619	1	16.4505	2.29	0.1316
$y(t-5)$	1	5.4187	0.75	0.3860	1	0.0198	0.00	0.9581
$u(t-5)$	1	56.6472	7.88	0.0054	1	11.8872	1.65	0.1996
$y(t-6)$	1	9.4652	1.32	0.2522	1	9.2639	1.29	0.2573
$u(t-6)$	1	19.5346	2.72	0.1005	1	19.5346	2.72	0.1005

Results from the ANOVA suggest that the most significant predictor variable is $y(t-1)$. Significance is evident in both the Type I and Type III variance partitions; in each case, $p < 0.0001$. A high level of significance is expected as the system response is not a completely random process, i.e., the temperature at time (t) is highly dependent on the temperature at time $(t-1)$. A more surprising result, though, is the lack of significance with regard to the parameter $y(t-2)$. The lack of significance has been interpreted as an anomaly and cannot be construed as representative of all system operations.

Interpretation of the ANOVA for the first two system input variables, $u(t-1)$ and $u(t-2)$, is a little more perplexing and requires a little more care. The Type I partition indicates that $u(t-1)$ is significant $(p < 0.0001)$ whereas the Type III partition indicates that $u(t-1)$ is marginally significant $(p < 0.19)$. The difference in significance can be attributed to the nature of the analyses when several extraneous predictor variables are included in the statistical model. However, since parameters were sequenced in descending order of importance, the Type I sums of squares analysis is more indicative of actual parameter significance. Accordingly, $u(t-1)$ may be considered as an important factor in characterizing the bed temperature response. Further discussions concerning the interpretation of model adequacy are presented by Cox [8].

In addition to significant air flow rates at time $(t-1)$, the Type I variance partition suggests that temperature response is strongly influenced by air flow rates at time $(t-2)$. That $u(t-2)$ is an important parameter is further demonstrated by the Type III variance partition. Consequently, a DARMA model of minimum order

two is required to characterize the dynamics of the fluidized bed combustor. However, along with second-order predictor variables, the Type I variance partition suggests that FBC response is affected by the parameters $y(t-4)$, $u(t-5)$, and $u(t-6)$. Hence, a second-order DARMA model is not a complete model of combustor operations, and cognizance must be taken with regard to both temperature and air flow rate influences from the distant past.

A multiple regression ANOVA of bed dynamics was performed with a second-order DARMA model. The adjusted correlation coefficient for the second-order DARMA model is 0.9939, to be compared with an adjusted correlation coefficient of 0.9943 for the sixth-order DARMA model. Partial regression coefficient estimates of the second-order model are presented in Table 5.2. Significant coefficients include a_1, b_2, and d, but coefficients b_1 and a_2 can be taken as zero. However, neither b_1 nor a_2 was specified as zero in the RLS procedure so as to account for possible significance during combustor operation.

Table 5.2: Coefficient estimates, 2nd-order DARMA model

COEFFICIENT	ESTIMATE	T FOR Ho: COEFFICIENT=0	PR > |T|	STD ERROR OF ESTIMATE
d	3.5942	7.79	0.0001	0.4617
a_1	-0.9272	14.13	0.0001	0.0656
b_1	0.0717	0.51	0.6126	0.1415
a_2	-0.0717	1.08	0.2794	0.0662
b_2	-0.4811	-3.46	0.0006	0.1390

For a sampling interval of 20 seconds, the GLM procedure estimated roots of the $A(q^{-1})$ polynomial near unity on the complex unit circle. Specifically, roots were

estimated at -0.0718 and 0.9990, suggesting a stable open-loop system. This may also be verified using Jury's stability test in which the second-order polynomial

$$\begin{aligned} F(q) &= (a_o q^2 + a_1 q + a_2) y(t); \qquad (5.2) \\ a_o &= 1 \end{aligned}$$

has roots inside the unit circle if the criteria

$$\begin{aligned} F(1) &> 0 \\ F(-1) &> 0 \\ |a2| &< a_o \end{aligned}$$

are satisfied. Using values listed in Table 5.2,

$$\begin{aligned} F(1) &= 0.0011 \\ F(-1) &= 0.1445 \\ |-0.07167| &< 1 \end{aligned}$$

Although Jury's stability test suggests a stable open-loop system, coefficient variance is sufficiently large such that $F(1)$ cannot be taken as strictly positive with a high degree of confidence. That is,

$$\begin{aligned} V(F(1)) &= V(a_o + a_1 + a_2) \qquad (5.3) \\ &= V(a_1) + V(a_2) - 2CV(a_1, a_2) \\ n.b., \qquad V(a_o) &= 0; CV(a_o, a_1) = 0; CV(a_o, a_1) = 0 \end{aligned}$$

Using variance and covariance estimates from the covariance matrix in Appendix C,

$$V(F(1)) = 0.0173$$

For this data set, variance of $F(1)$ is an order of magnitude greater than the expected value of $F(1)$. Hence, roots can easily be estimated outside the unit circle, which would contradict observed open-loop system characteristics. Therefore, during combustor operation, coefficient estimates from the RLS procedure had to be constrained so that roots of $A(q^{-1})$ remained within the unit circle.

5.2 Adaptive Control

Several test runs were made with the self-tuning controller, but, for brevity, only three tests will be examined in detail. Although many tests were performed, results from these runs sufficiently characterize the performance of the adaptive controller.

For all three tests, the time interval between control signal updates was set to 20 seconds, one-tenth of the dominant time constant. LQG weighting matrices R_x and R_u were specified as

$$\begin{aligned} R_x &= \begin{bmatrix} 100 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ R_u &= [8000] \end{aligned} \tag{5.4}$$

The first test was performed to examine features of estimate trajectories during the initial tuning period. Temperature data from the first run are illustrated in Fig. 5.3 and corresponding annular bed air flow rate data are illustrated in Fig. 5.4. RLS coefficient estimates and feedback gains are graphed in Figs. 5.5 though 5.8.

The initial transient temperature response of this run was typical of most adaptive control tests on the fluidized bed. Initial coefficient estimates and initial feedback

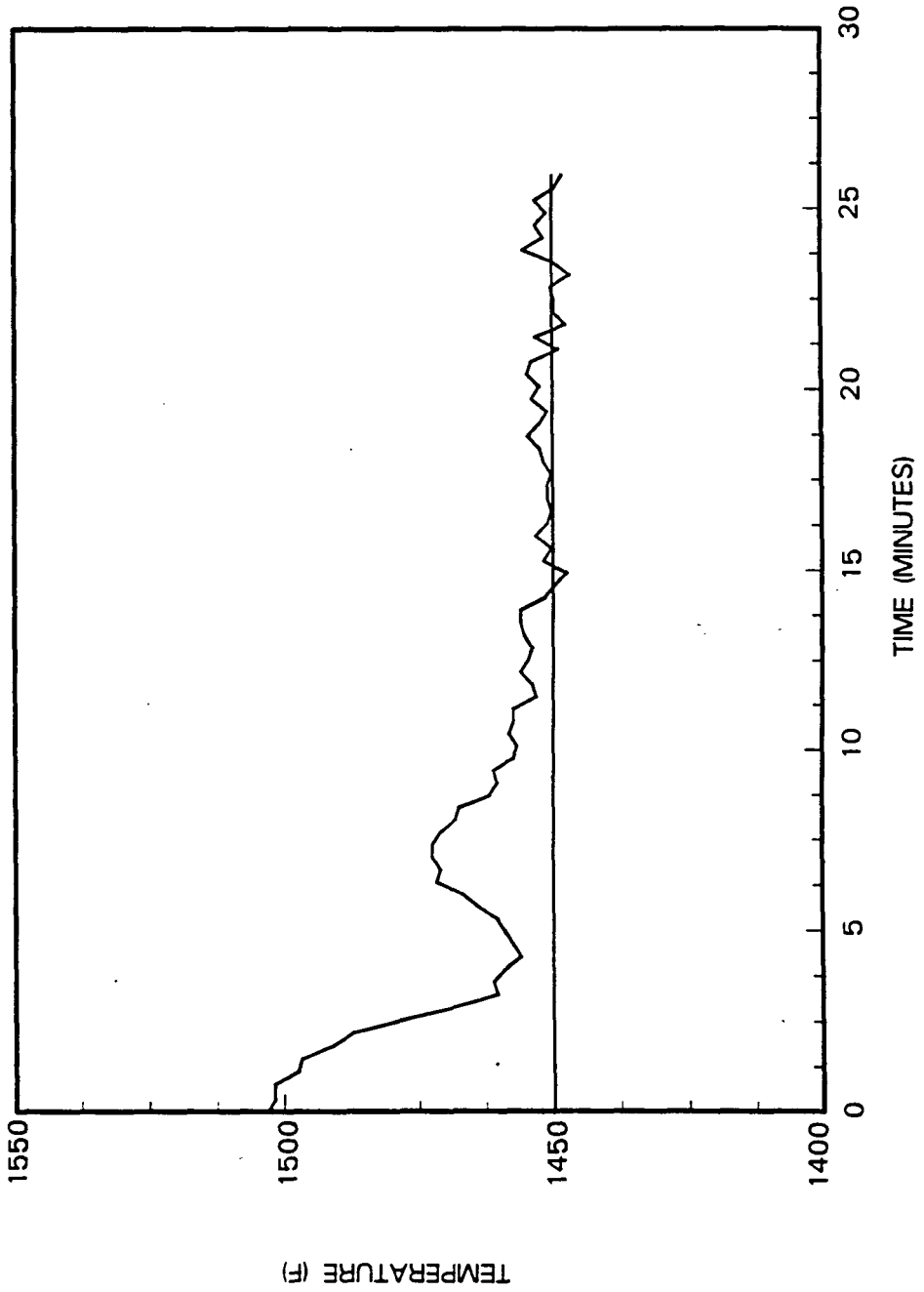


Figure 5.3: Test 1 temperature response data

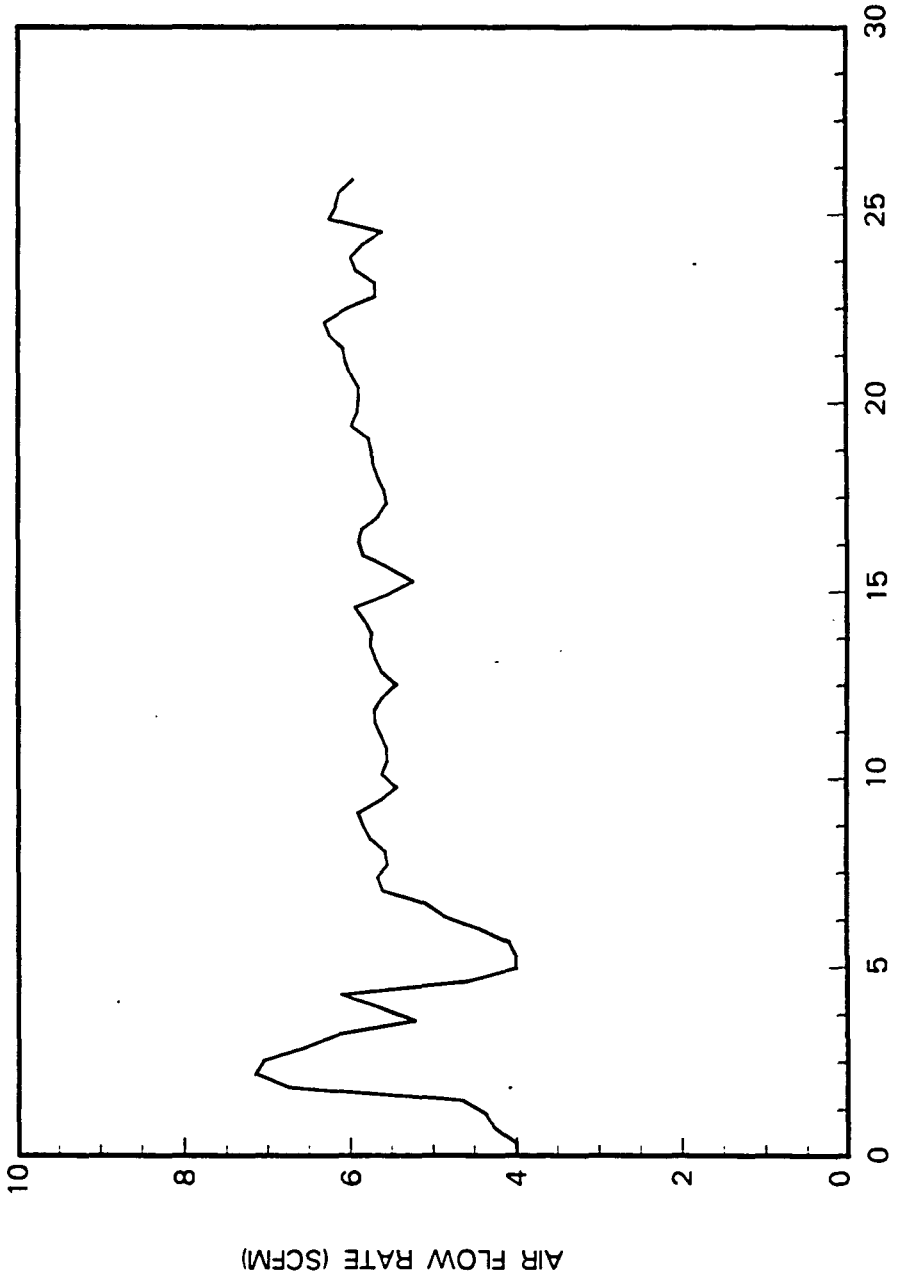


Figure 5.4: Test 1 secondary air flow rate data

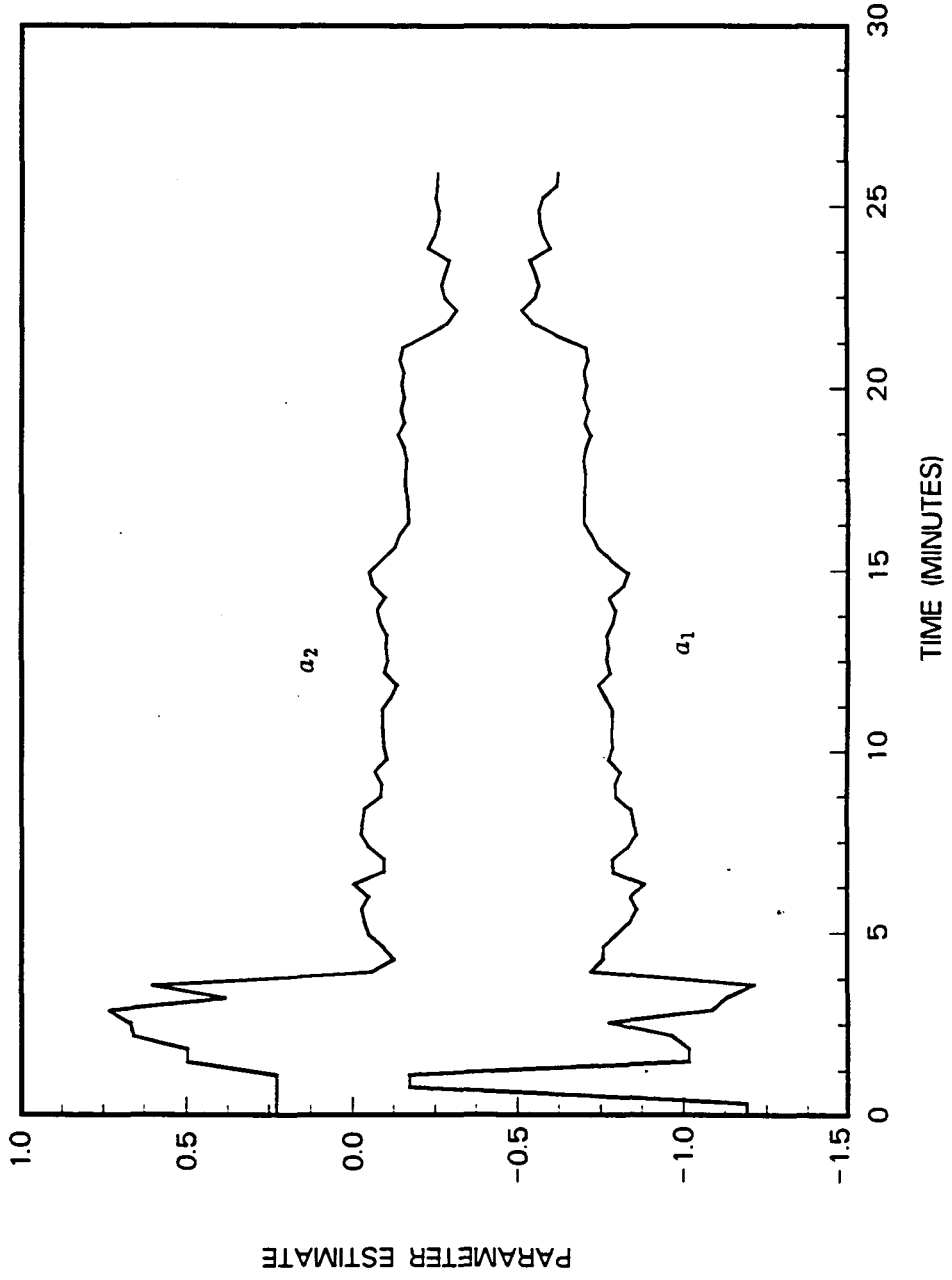


Figure 5.5: Test 1 parameter estimates of a_1 and a_2

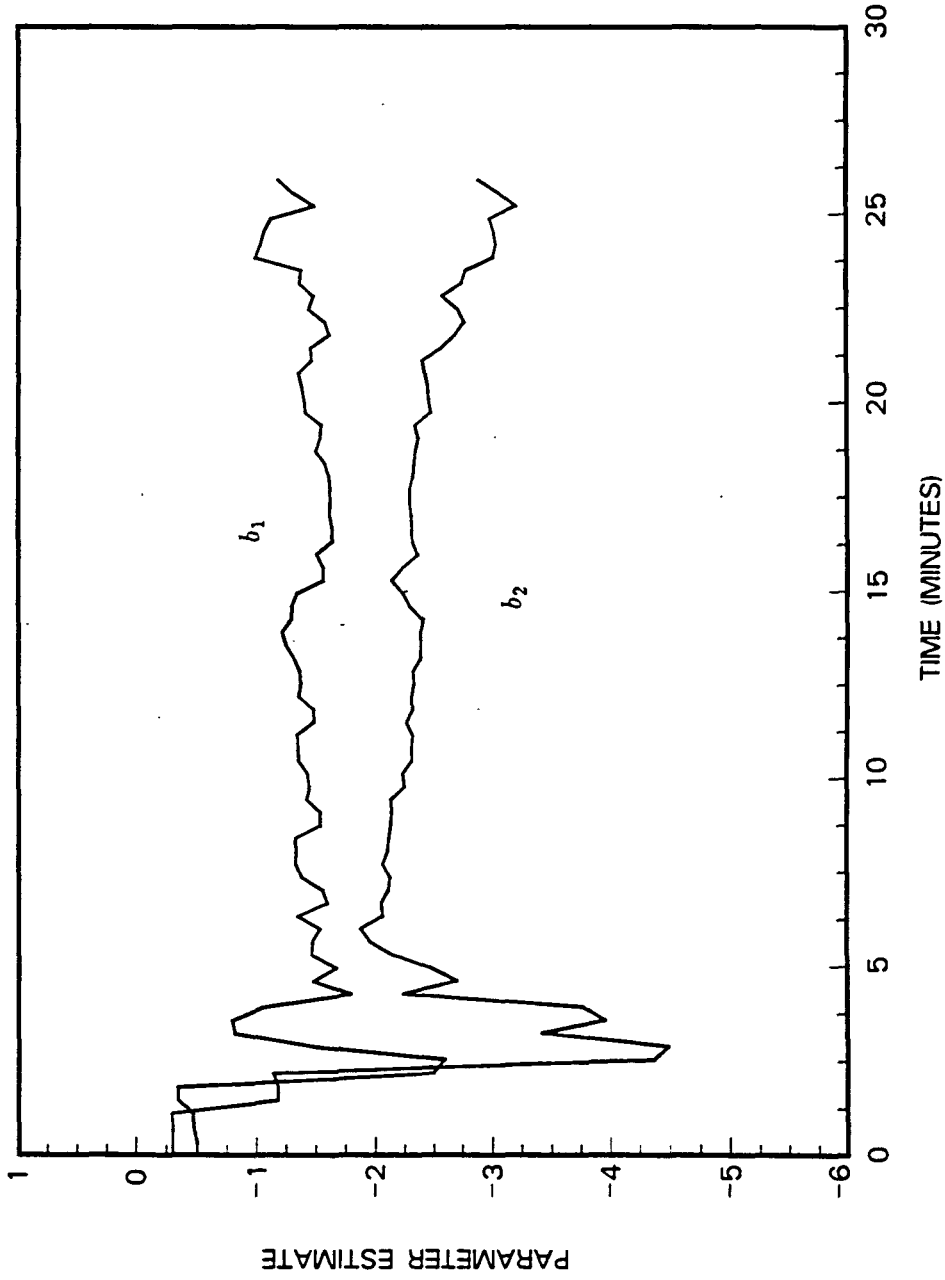


Figure 5.6: Test 1 parameter estimates of b_1 and b_2

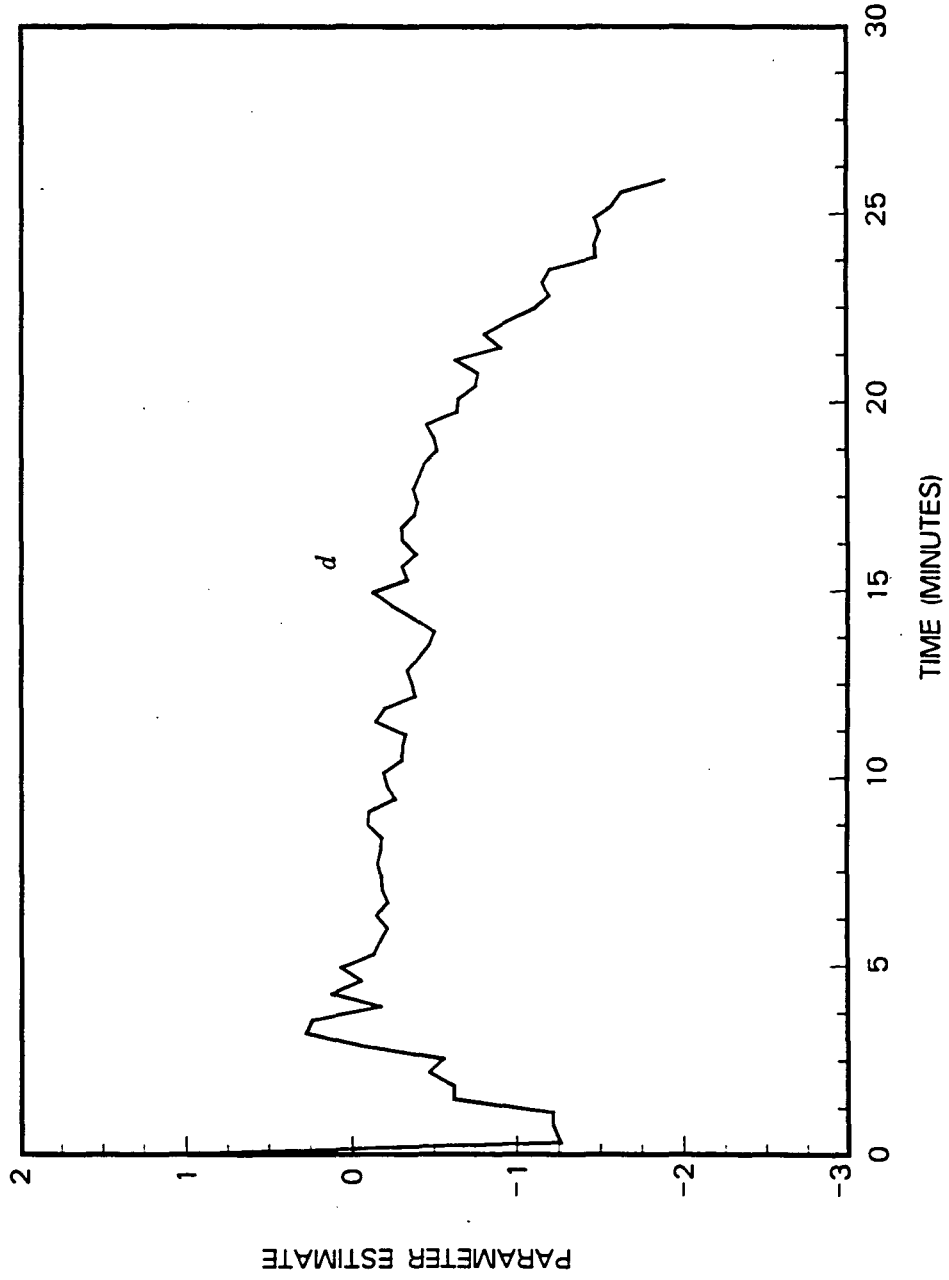


Figure 5.7: Test 1 parameter estimate of d

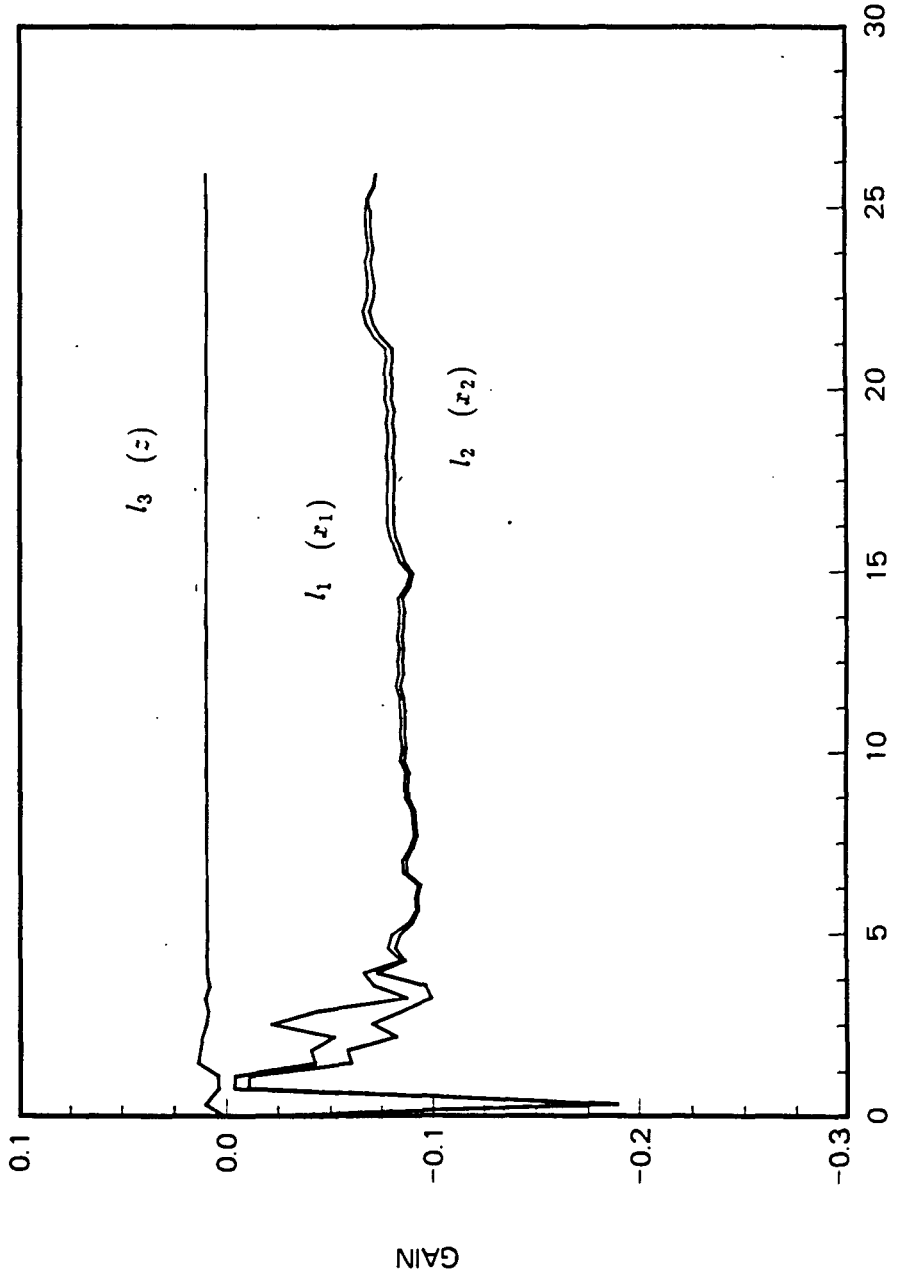


Figure 5.8: Test 1 LQG feedback gains

gains fluctuated as both the covariance matrix and the partial regression coefficient vector converged. Parameter convergence during the initial tuning process is manifested by the temperature response at 7 minutes into the run. After coefficients and feedback gains converged, the bed temperature asymptotically approached the set point temperature of 1450 °F. At steady-state conditions, the bed temperature had an rms error of 1.97 °F and variance of 4.46 $^{\circ}F^2$ (35 df).

Coefficient trajectories of a_1 and a_2 are graphed in Fig. 5.5. Both coefficients converged quickly although the a_1 estimate briefly crossed over the abscissa into the positive-real domain before converging to a negative value. During the initial tuning period, a few noticeable flats were present on both the a_1 and a_2 trajectories. At these places, Jury's stability test detected roots outside the unit disk, and plateaus were created as RLS estimates were not revised.

Similar patterns are present in the b_1 and b_2 trajectories, which are illustrated in Fig. 5.6. As with the a_1 and a_2 coefficients, initial convergence of b_1 and b_2 was tumultuous. Other similarities exist to the extent that both estimates of b_1 and b_2 were nonzero and converged after 5.7 minutes of controller operation. In contrast to the a and b trajectories, the offset parameter, d, (Fig. 5.7) converged in a relatively quick and smooth manner. Originally, the offset parameter was intended to account for nonsteady initial conditions; however, during combustor operation, the RLS procedure incorporated nonlinear system dynamics in the estimate of d. Therefore, as the fluidized bed evolved over time, the estimate of d changed accordingly. For this particular run, the offset parameter initially converged to a constant value of -0.8 but later strayed from this value. Usually, the estimate of d deviated from a constant

value at the onset of a disturbance. Regrettably, the test was terminated prematurely.

In this run as well as all other runs, parameter estimates varied about mean values. Some concern arose over whether or not estimate "noise" would adversely affect feedback gains and system performance. For the most part, though, slight deviations present in the parameter estimates were not mapped onto the feedback gains, as illustrated in Fig. 5.8. In other words, after the tuning transient, coefficient estimates fluctuated slightly, but feedback gains l_1, l_2, and l_3 (corresponding to observer variables $\hat{x}_1(t)$, $\hat{x}_2(t)$, and $z(t)$, respectively) remained approximately constant.

The second of the three tests examined controller response to a step change in the set point temperature. Results from this run demonstrate and characterize tracking abilities of the self-tuning controller. Temperature data from this run are graphed in Fig. 5.9 and corresponding air flow rates are graphed in Fig. 5.10. Coefficient estimates and feedback gains are illustrated in Figs. 5.11 through 5.14.

Progression of bed temperature during the tuning stage was similar to that of the first run. Perhaps the only noticeable difference was a spike in the air flow rate at 6 minutes into the run. The spike resulted from poor coefficient estimates. Even though roots were estimated as stable, estimates were so poor that anomalous air flow rates occurred. However, the sudden surge in the secondary air flow rate did not present any difficulties; in fact, the surge enhanced system performance by reducing settling time. Improved system performance was indeed fortuitous as sudden changes in either air flow or temperature magnify nonlinear system behavior and introduce potential parameter identification difficulties. However, since the surge in the air flow rate was unique to this run, a detailed investigation into the cause and solution of

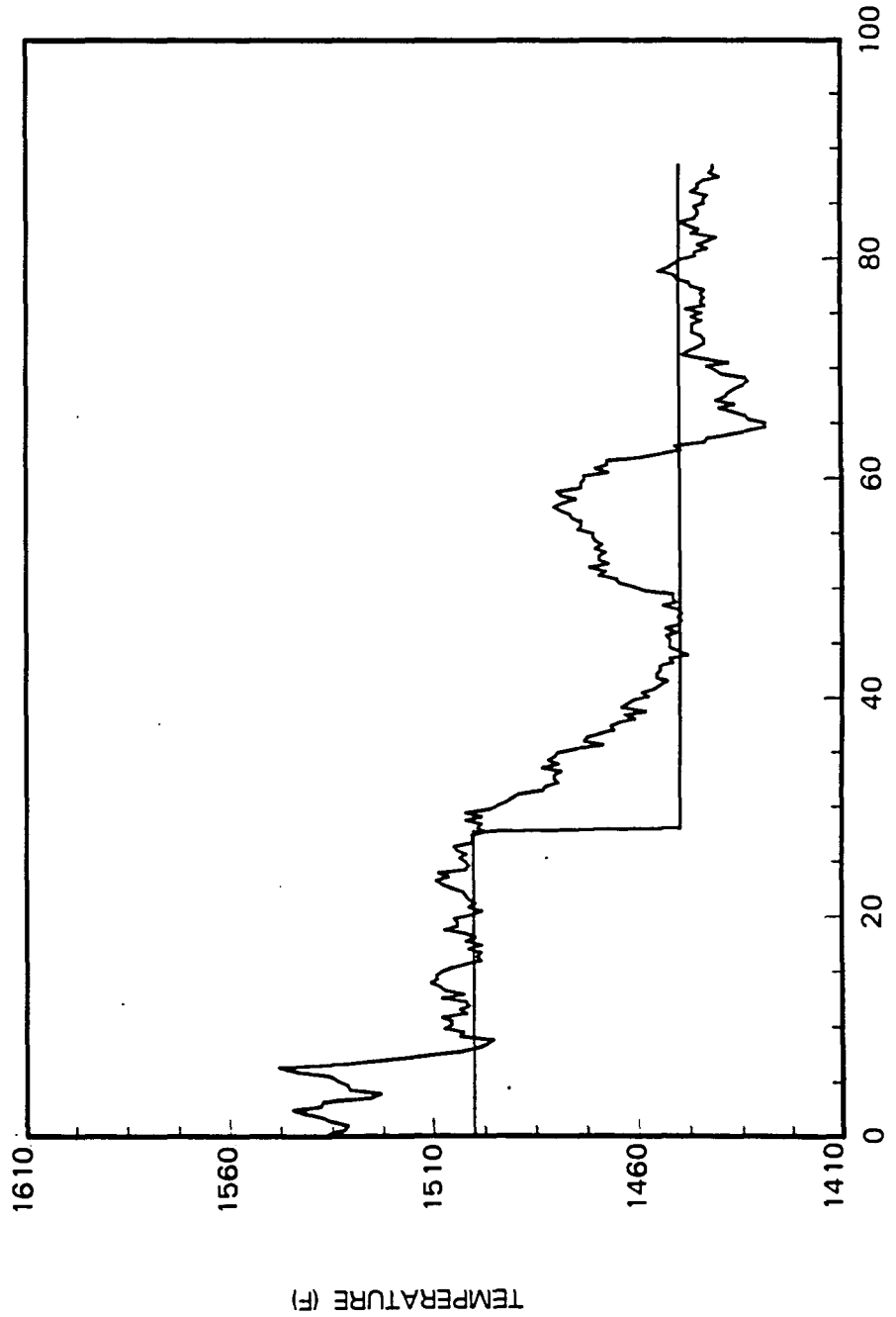


Figure 5.9: Test 2 temperature response data

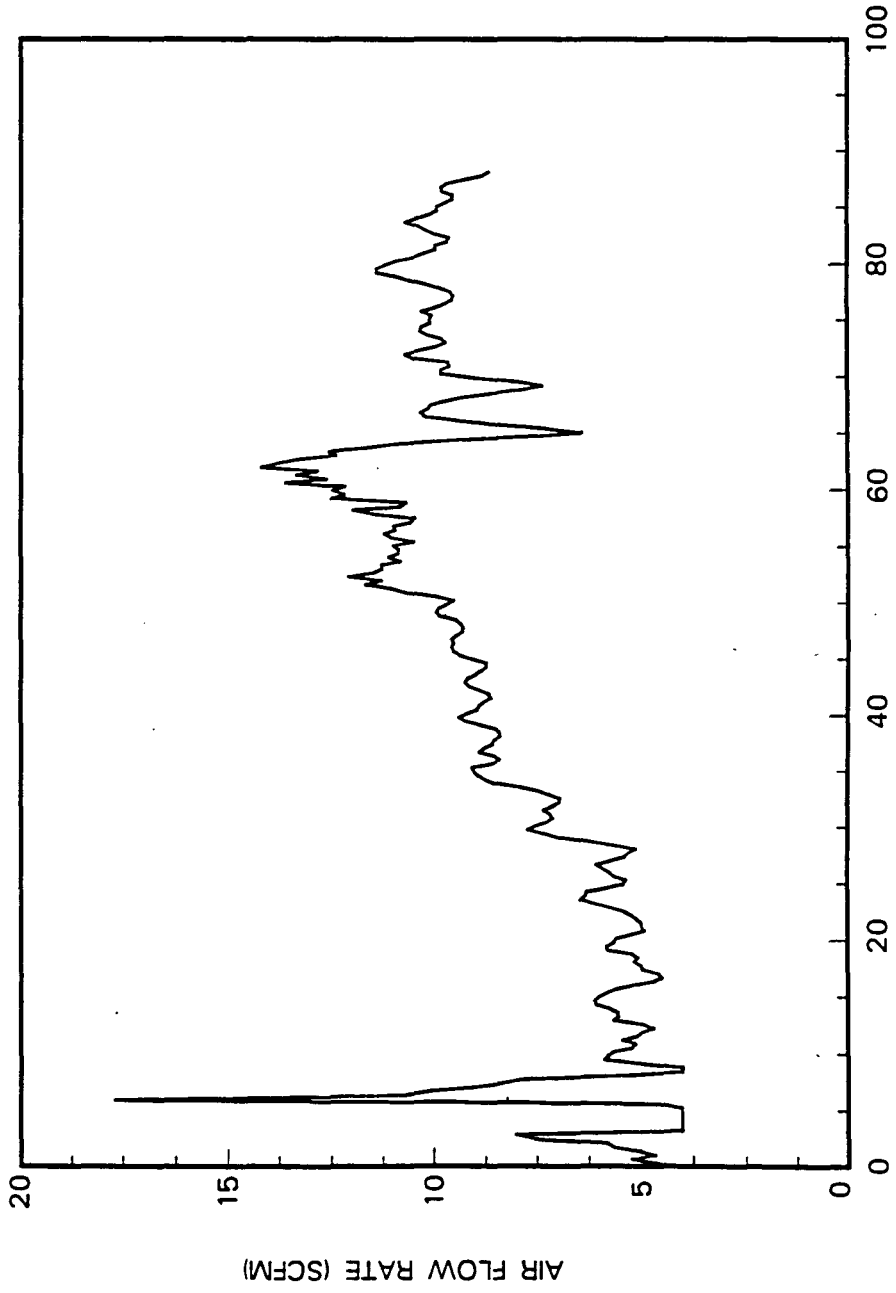


Figure 5.10: Test 2 secondary air flow rate data

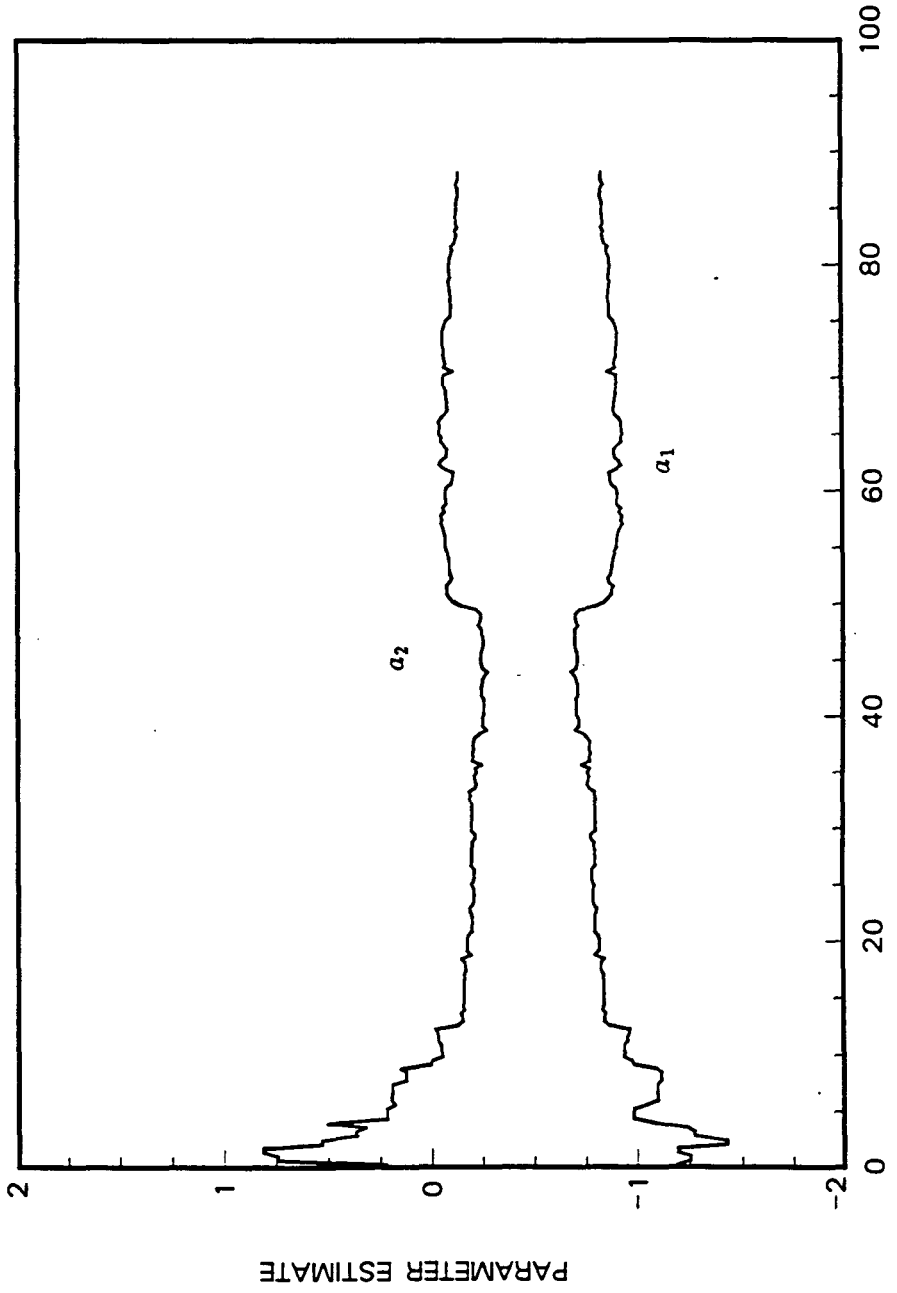


Figure 5.11: Test 2 parameter estimates of a_1 and a_2

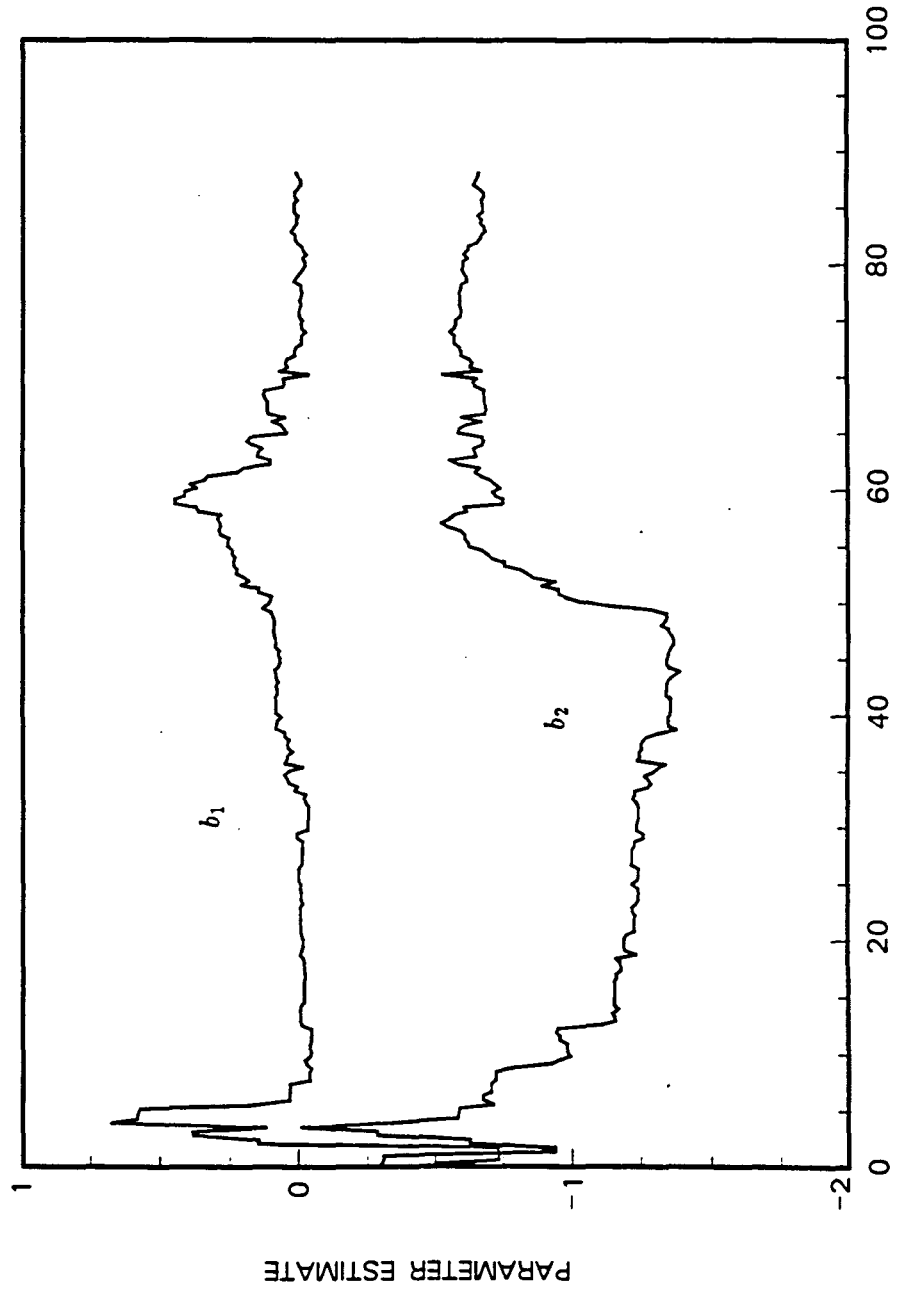


Figure 5.12: Test 2 parameter estimates of b_1 and b_2

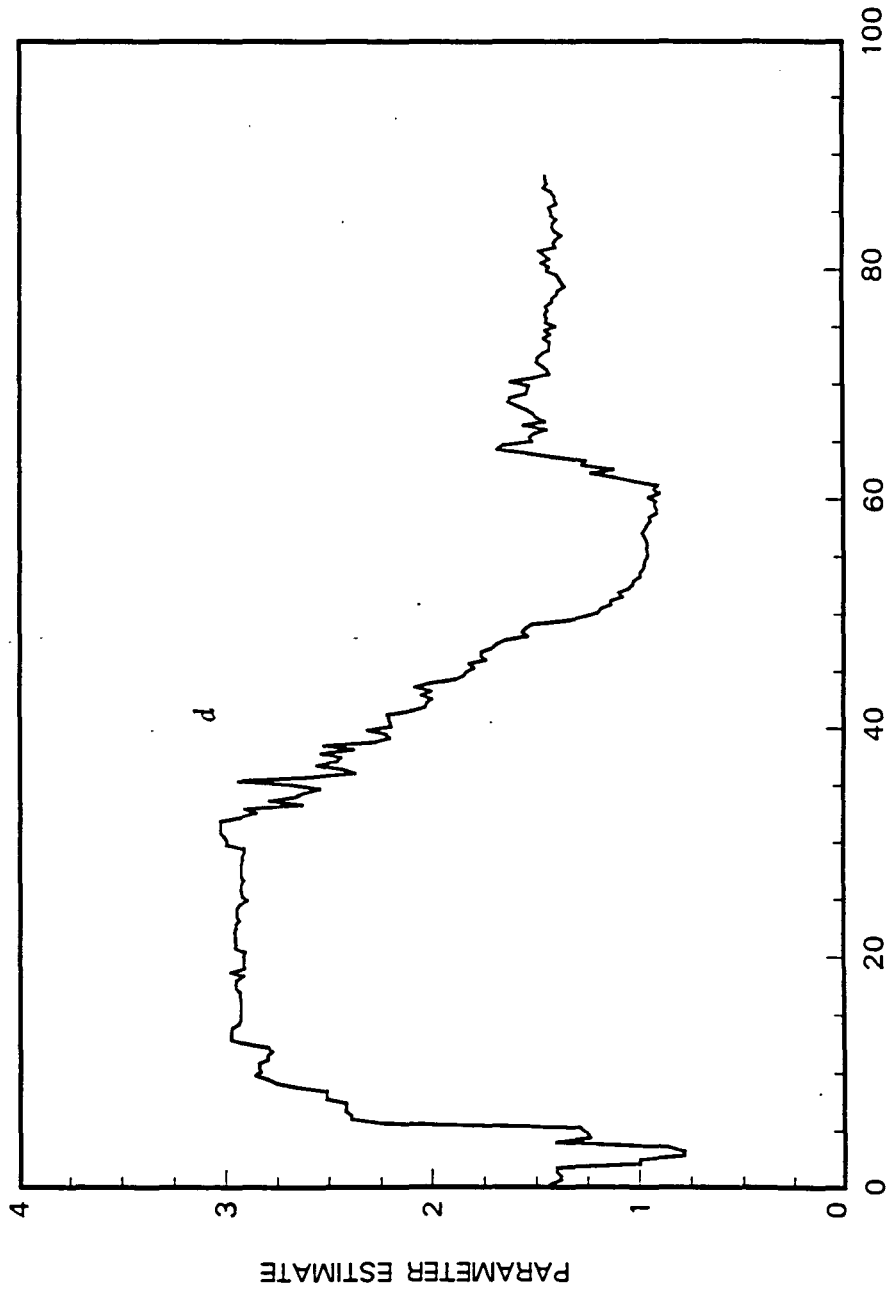


Figure 5.13: Test 2 parameter estimate of d

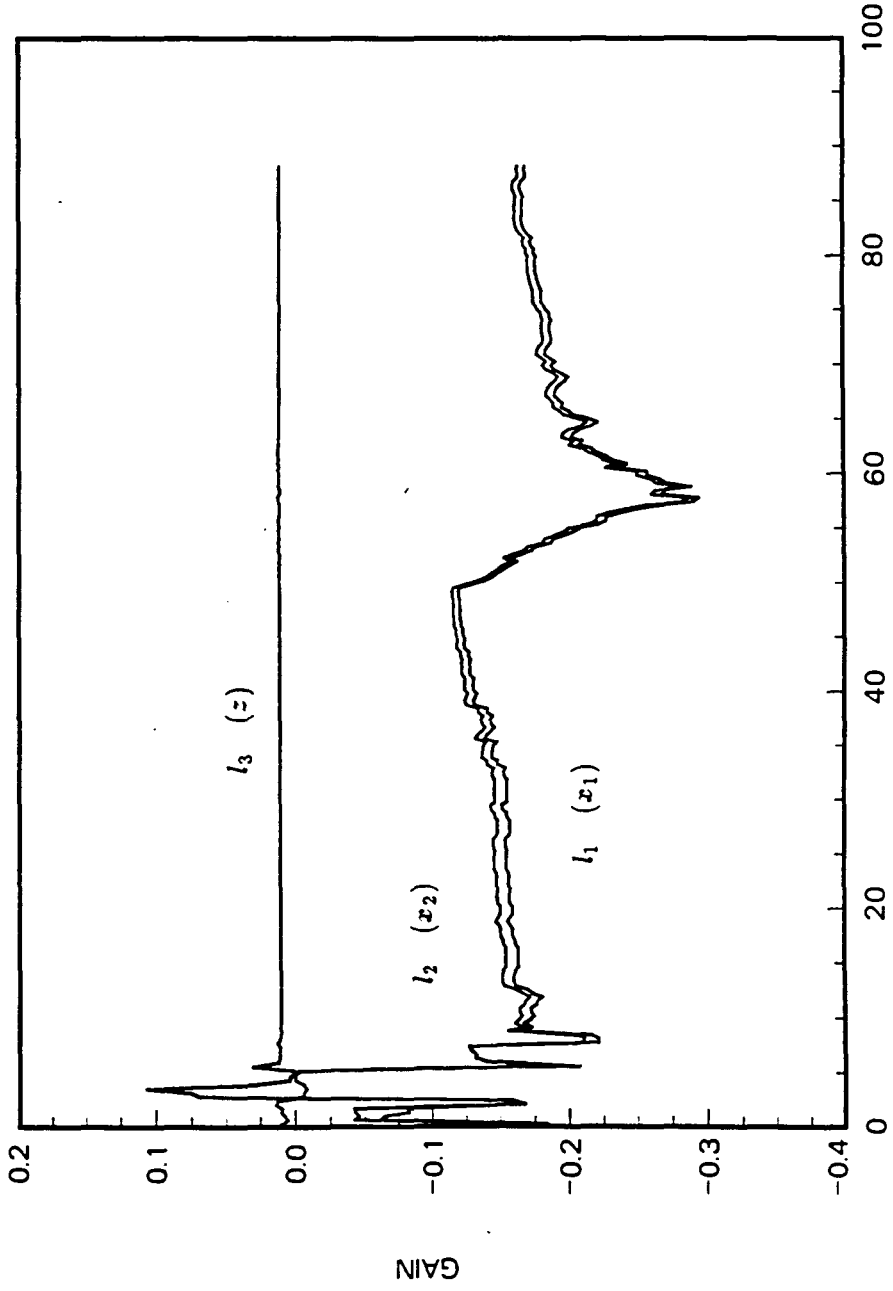


Figure 5.14: Test 2 LQG feedback gains

this problem was not warranted.

At 33 minutes into the run, the temperature set point was changed from 1500 °F to 1450 °F. The adaptive controller was able to track the change, although the response was a little sluggish. Attention is drawn to the disturbance at 50 minutes. The nature and magnitude of the disturbance are presently unknown although variable properties of the coal are suspected. Note that, even for this disturbance, the controller was capable of bringing the system back to the set point with little overshoot.

For the second run, the steady-state bed temperature had an rms error of 2.35 °F and a pooled variance of 10.36 $°F^2$ (115 df). The successful regulation of temperature was dependent on sufficiently accurate estimates of a_1, a_2, b_1, b_2 and d. Convergence of these coefficients occurred within ten minutes of operation. Coefficients a_1 and a_2 (Fig. 5.11) closed in upon constant values, which changed slightly at the beginning of the disturbance. Similarly, parameters b_1 and b_2 (Fig. 5.12) closed in upon constant values. In this case, consistent with the ANOVA, coefficient b_1 initially approached zero. At the disturbance, though, b_1 become significant and b_2 concomitantly decreased in magnitude. After the disturbance, b_1 asymptotically approached zero again whereas b_2 remained approximately constant but at a magnitude appreciably smaller.

Similar patterns occurred with the estimate of the offset parameter d (Fig 5.13). At first, d converged rapidly to a value of 2.95. At 33 minutes into the run, a new set point was specified, and as the controller tracked the change, the estimate of d was updated to account for nonlinear system dynamics. At the beginning of the disturbance, the rate of change in the offset parameter increased markedly, then leveled off. After the disturbance, system dynamics changed once again, and the

offset parameter was updated accordingly.

Feedback gains for this run remained fairly constant after initial parameter convergence but changed appreciably at the disturbance. After the disturbance, the rate of change in the feedback gains diminished as they settled to slightly higher values. In all, the controller could identify and conform to system changes but was slow to compensate for the disturbance. The sluggish response was due, in part, to a slow forgetting factor coupled with a highly conservative command signal weight, R_u.

The last run demonstrates that temperature performance was repeatable even though estimate trajectories differed considerably. Temperature and air flow data from this run are graphed in Figs. 5.15 and 5.16, respectively. RLS coefficient estimates and feedback gains are graphed in Figs. 5.17 though 5.20.

Temperature response of this run was similar to those of the previous runs; steady-state temperature had an rms error of 1.88 °F. Typical steady-state temperature data are listed in Table 5.3. Variance of the tabulated data is 8.86 $°F^2$ (29 df), to be compared with a pooled, temperature variance of 9.36 $°F^2$ (120 df).

Two unknown disturbances were introduced into the system: one at 18 minutes and another at 50 minutes. The latter disturbance appears to be of shorter duration and larger in magnitude than the former.

In contrast to the previous tests, a_1 converged positively and a_2 converged negatively, with each varying only slightly during the entire test run. Similarly, both coefficients b_1 and b_2 closed in on nearly constant values but changed considerably as the controller tracked the change in the set point. Initially, b_1 approached zero, but became positive when the set point was changed. Note that, even though b_1

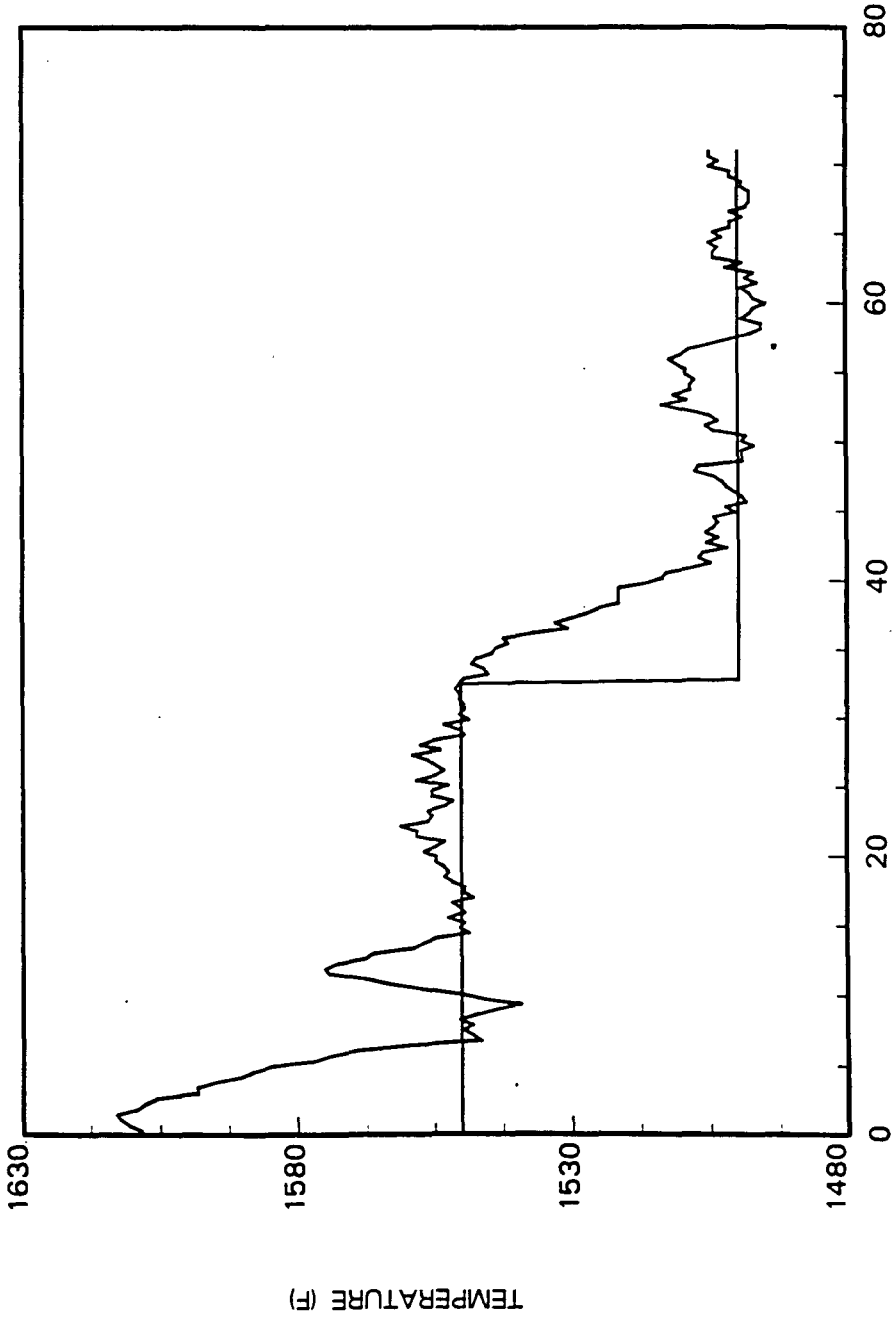


Figure 5.15: Test 3 temperature response data

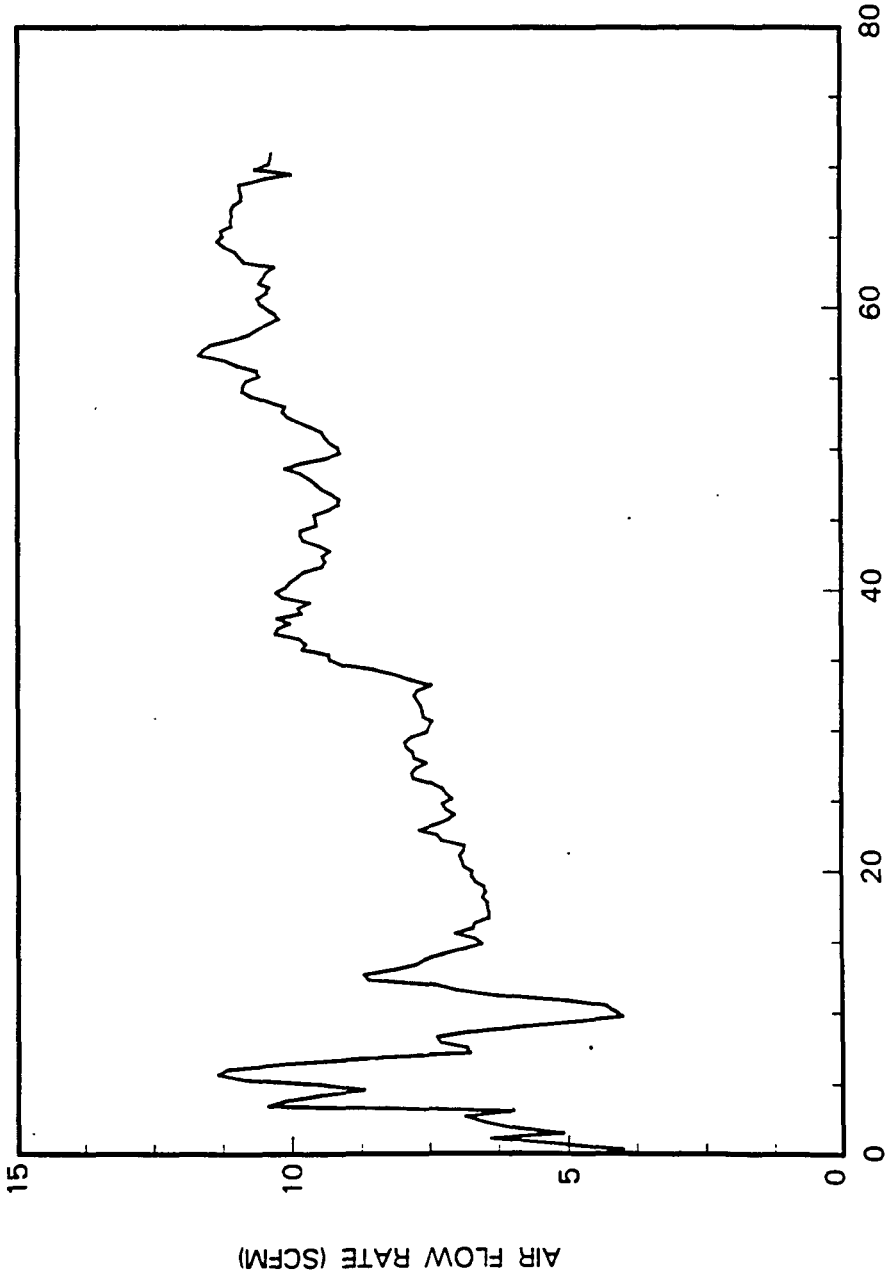


Figure 5.16: Test 3 secondary air flow rate data

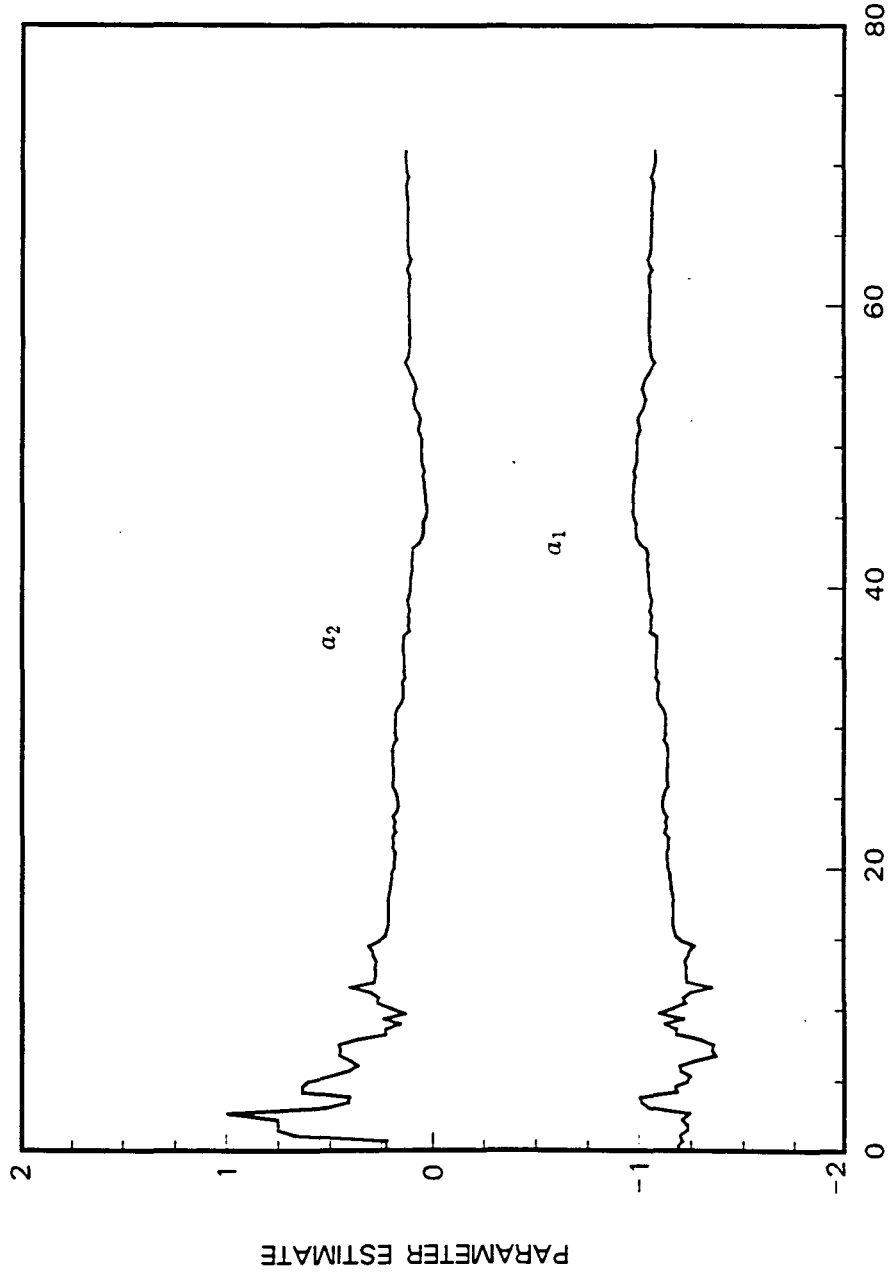


Figure 5.17: Test 3 parameter estimates of a_1 and a_2

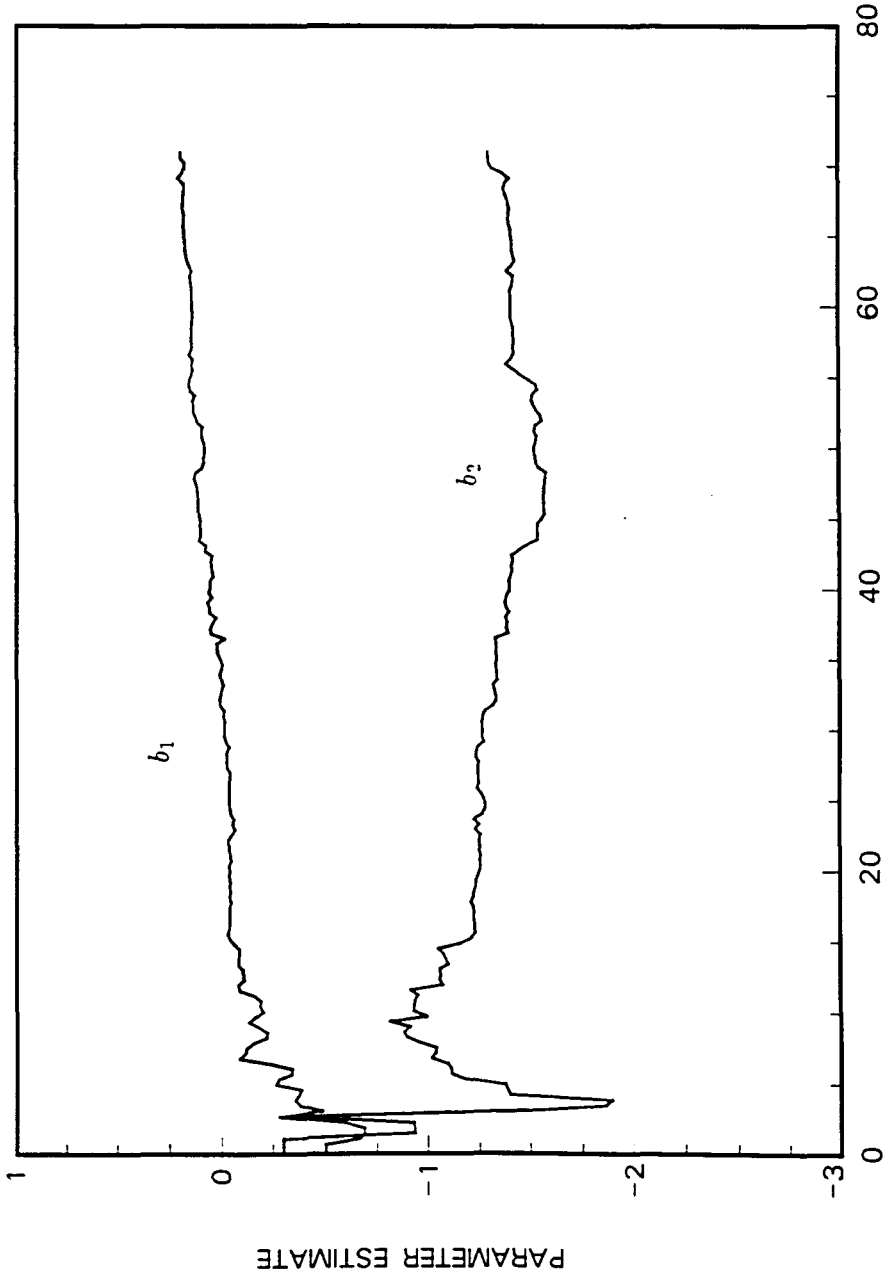


Figure 5.18: Test 3 parameter estimates of b_1 and b_2

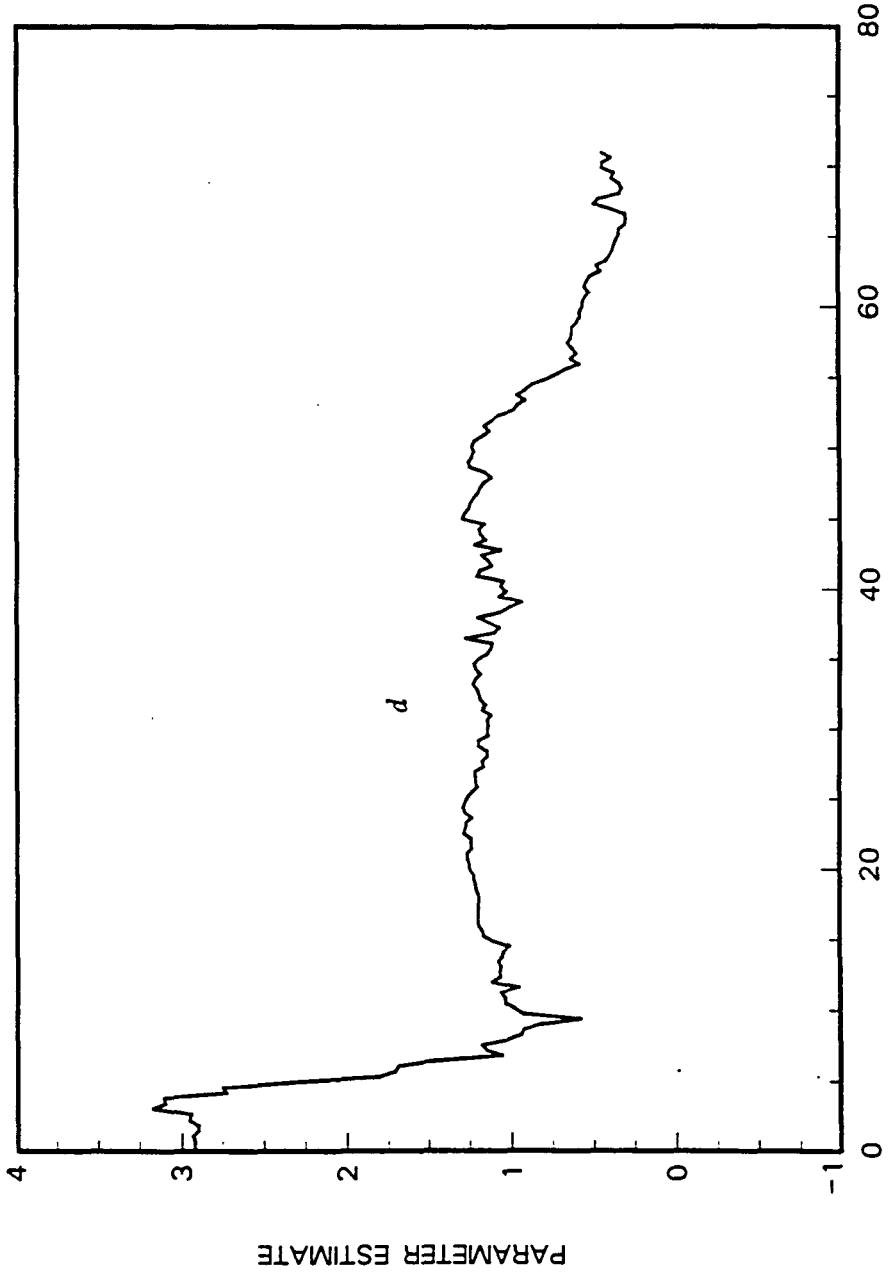


Figure 5.19: Test 3 parameter estimate of d

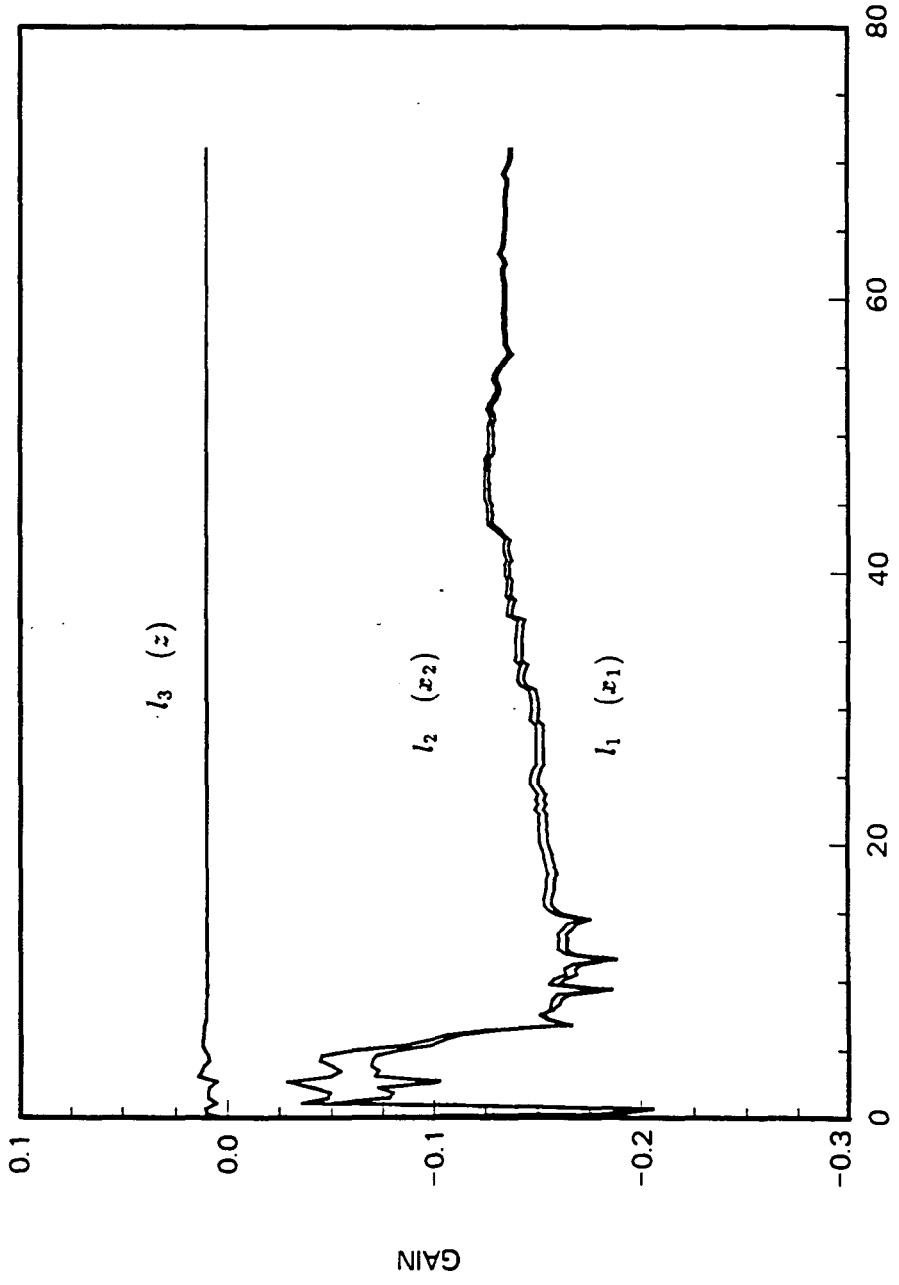


Figure 5.20: Test 3 LQG feedback gains

Table 5.3: FBC response with adaptive control

Time (min)	Set Point (°F)	Measured Temp. (°F)	Time (min)	Set Point (°F)	Measured Temp. (°F)
77.119	1500	1504.98	82.614	1500	1502.11
77.484	1500	1501.39	82.979	1500	1499.24
77.848	1500	1497.80	83.347	1500	1504.27
78.212	1500	1495.65	83.713	1500	1504.27
78.581	1500	1495.65	84.083	1500	1503.55
78.947	1500	1499.24	84.450	1500	1504.98
79.322	1500	1497.80	84.817	1500	1502.83
79.686	1500	1497.09	85.182	1500	1504.27
80.055	1500	1494.93	85.548	1500	1501.39
80.421	1500	1497.09	85.915	1500	1501.39
80.787	1500	1497.80	86.284	1500	1499.24
81.151	1500	1499.24	86.653	1500	1501.39
81.518	1500	1496.37	87.018	1500	1498.52
81.882	1500	1498.52	87.383	1500	1497.80
82.246	1500	1497.09	87.752	1500	1498.80

became positive, b_2 was sufficiently negative to accurately describe the fluidized bed system. In other words, after initial tuning transients, the RLS algorithm consistently estimated a system in which an increase in the air flow rate would result in a corresponding decrease in the bed temperature.

Unlike the second run, feedback gains remained approximately constant even through the disturbances. This indicates that the disturbances were less severe during this run than for the pervious run.

Two points of interest arise from these test runs. The first point pertains to the offset parameter. Originally, the offset parameter was included in the DARMA model to account for nonsteady-state initial conditions. However, as the combustor evolved over time, nonlinear system dynamics — especially dynamics associated with system

disturbances — were incorporated in the estimate of the offset parameter. Clearly, then, a constant offset parameter cannot adequately describe the nonlinear dynamics of disturbances within a fluidized bed combustor. To be sure, further study regarding model adequacy of unexpected disturbances is warranted.

The second point of interest pertains to the robustness of the adaptive control algorithm. The self-tuning controller used in this investigation combines a robust parameter estimation procedure with a robust control law. Questions arise as to whether or not a combination of robust components will necessarily result in a robust aggregate [1]. The results obtained in this study tend to support the notion of a robust whole though controller robustness is tempered slightly by the lack of adequate disturbance identification and compensation.

5.3 PI Control I

As a reference, the self-tuning algorithm was compared with a PI controller tuned by the Ziegler-Nichols method. Just as with the self-tuning controller, sampling time was set to 20 seconds.

System response under PI control is graphed in Fig. 5.21 and corresponding annular air flow rates are graphed in Fig. 5.22. Note that both transient and steady-state behavior are considerably worse with PI control than with adaptive control. For example, substantial overshoot was present at 10 minutes into the run. Poor performance is further evidenced by a steady-state rms error of 7.87 °F and a pooled, steady-state temperature variance of 62.2 $°F^2$ (155 df). Although system performance was not extraordinary, no system instabilities were observed during controller

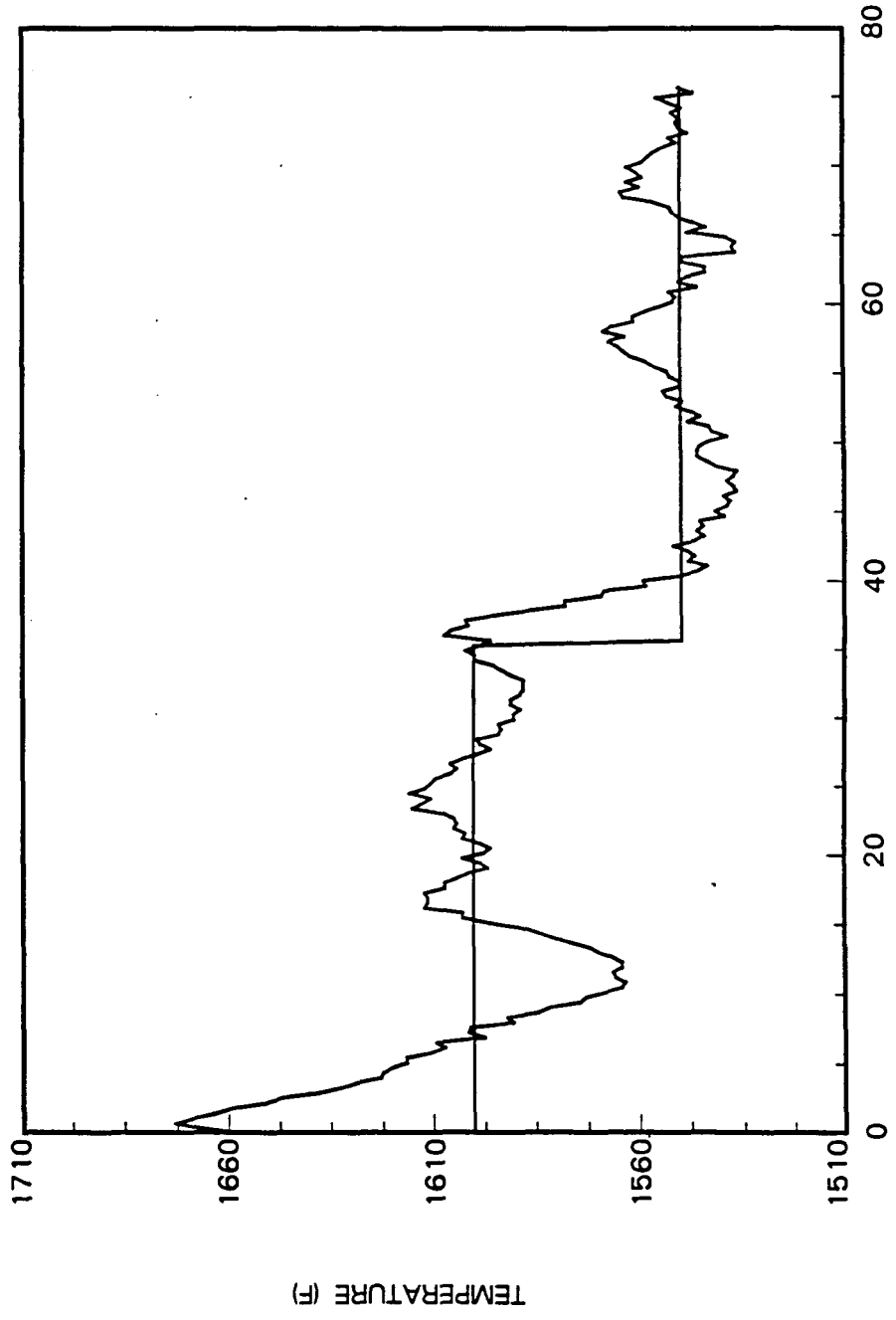


Figure 5.21: Temperature response data with PI control (20 second sampling interval)

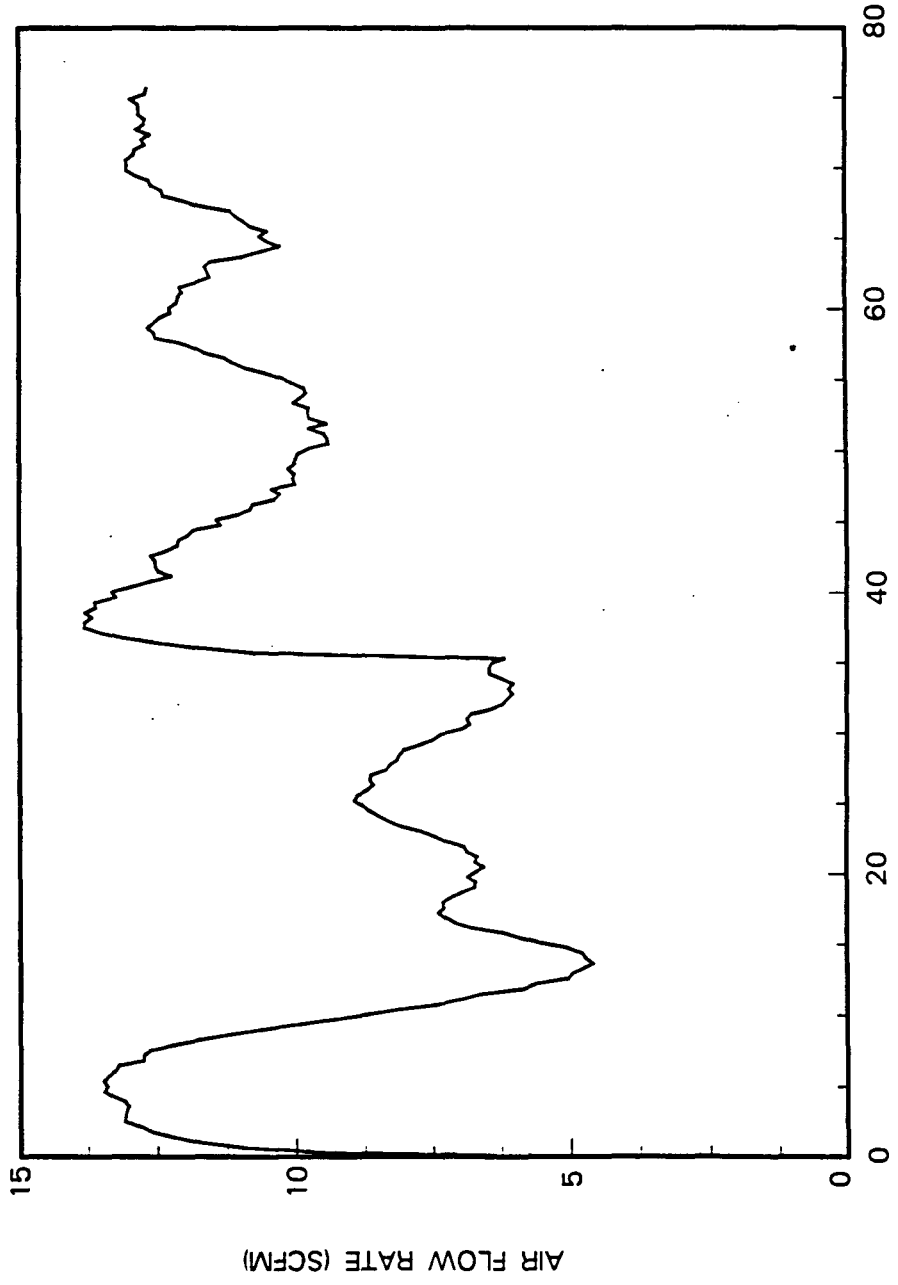


Figure 5.22: Secondary air flow rate with PI control (20 second sampling interval)

operation.

Poor controller performance was due, in part, to an off-line parameter estimation procedure. With variable characteristics of coal and bed sand, system estimation from a step input did not accurately describe future bed operations. In itself, the adaptive control algorithm is a PI controller, but, for a fluidized bed, on-line estimation allows for better tuning of the controller than does off-line estimation.

5.4 PI Control II

Difficulties were encountered with both adaptive and classical control techniques in minimizing variance at the set point temperature. Problems can be traced to the relatively large sampling interval. With a 20 second interval between control updates, bed temperatures tended to stray and could not be corrected by controller action. Hence, decreasing the sampling time from 20 seconds to 5 seconds should reduce steady-state temperature variance. Despite potential improvements in system performance, a decrease in the sampling interval would create potential parameter identification problems in the adaptive control algorithm. For a 5 second sampling time, changes in bed temperature would be dominated by system noise. and parameter identification would be based on the dynamics of the noise. This would produce deceptive parameter estimates that could ultimately lead to system instabilities. To solve this problem, system parameters would have to be estimated with a 20 second sampling interval and then projected onto a 5 second control update interval.

A projection scheme was not implemented with the adaptive control algorithm; rather, potential effectiveness was demonstrated with a PI controller in which the

sampling interval was set at 5 seconds. As a cautionary measure, the upper saturation limit on annular bed air was reduced to 20 scfm.

System response is graphed in Fig. 5.23 and annular air flow rates are graphed in Fig. 5.24. Without a doubt, steady-state temperature regulation was exceptional; rms error of the steady-state temperature response was 1.66 °F. Representative steady-state temperature data are presented in Table 5.4. For the most part, the steady-state temperature response had a small variance since temperature measurements were within the resolution of the thermocouple. Variance of the tabulated data is 3.1 $°F^2$ (37 df), to be compared with a pooled, steady-state temperature variance of 4.8 $°F^2$ (214 df).

Some overshoot was present in the transient system response although this can be corrected by adjusting controller coefficients. In addition, a hint of integrator wind-up was present at 58 minutes into the run but does appear to have degraded system performance. Also, because of influences from nonlinear system characteristics, temperature response was highly dependent on operating conditions. Nevertheless, the robust nature of classical feedback control was sufficient such that system instabilities were avoided.

Despite an otherwise flawless test, annular bed air flow rates became saturated at 41 minutes into the run. Although saturation protection was incorporated into the controller algorithms, substantial changes in air flow requirements during the transient response suggest that controller gains should be reduced. However, reduction in feedback gains would diminish the ability of the controller to effectively respond to system disturbances.

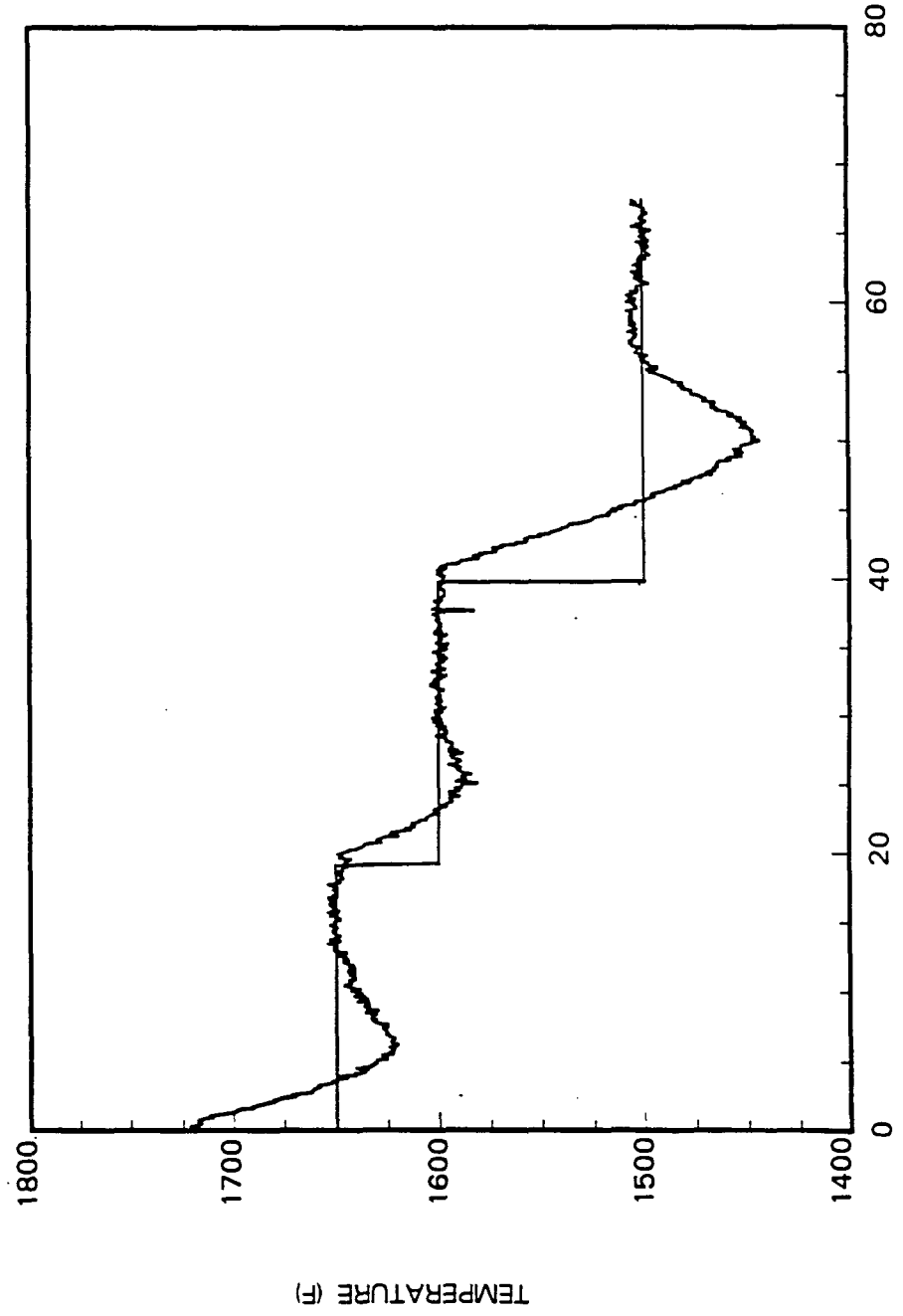


Figure 5.23: Temperature response data with PI control (5 second sampling interval)

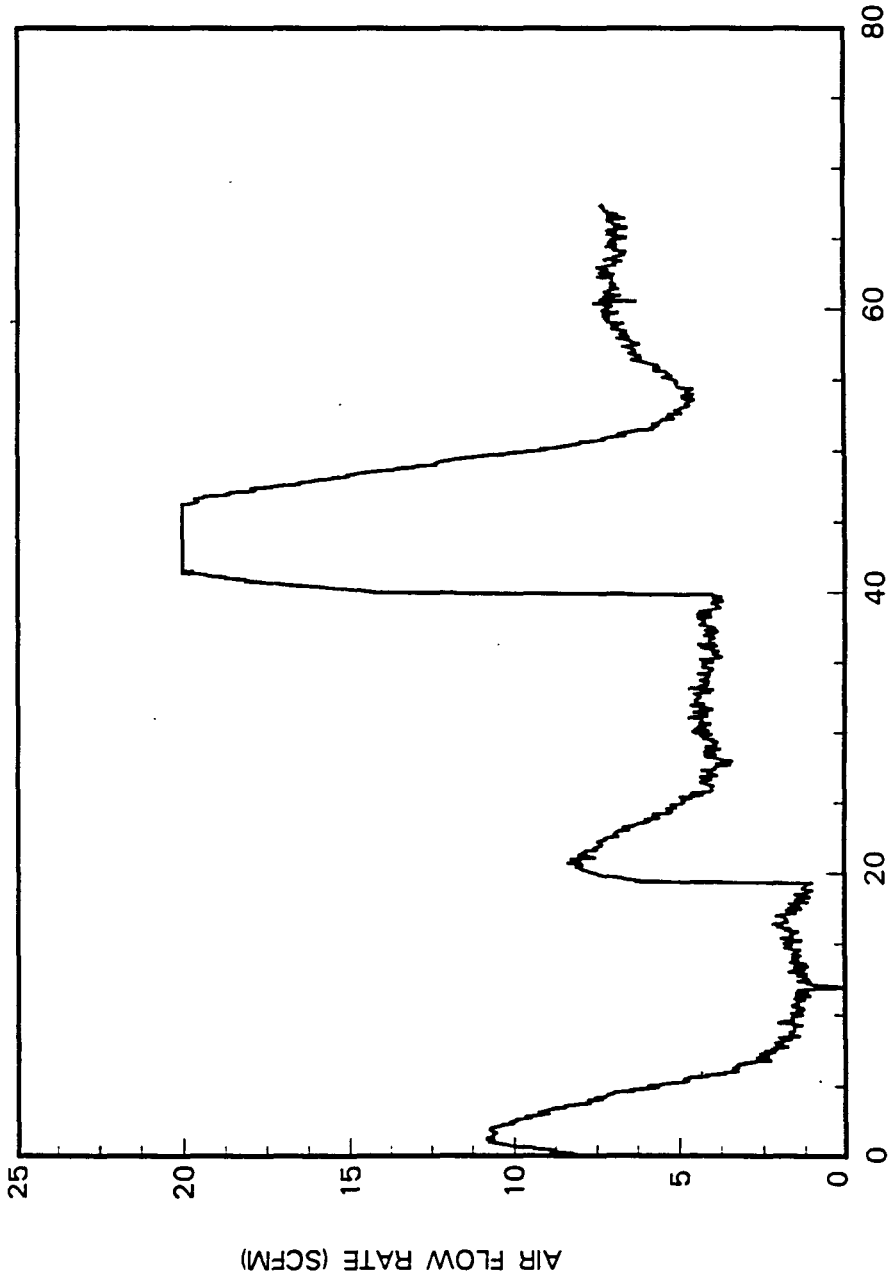


Figure 5.24: Secondary air flow rate with PI control (5 second sampling interval)

Table 5.4: FBC response with PI control

Time (min)	Set Point (°F)	Measured Temp. (°F)	Time (min)	Set Point (°F)	Measured Temp. (°F)
28.446	1600	1596.78	30.237	1600	1601.88
28.541	1600	1601.16	30.331	1600	1598.25
28.634	1600	1600.44	30.425	1600	1601.87
28.729	1600	1596.05	30.519	1600	1599.70
28.823	1600	1596.78	30.613	1600	1596.78
28.917	1600	1596.05	30.707	1600	1599.71
29.012	1600	1597.51	30.801	1600	1598.98
29.106	1600	1598.98	30.896	1600	1601.17
29.199	1600	1596.78	30.990	1600	1599.70
29.294	1600	1601.16	31.084	1600	1598.25
29.388	1600	1598.98	31.178	1600	1600.44
29.482	1600	1598.25	31.272	1600	1598.98
29.577	1600	1602.61	31.366	1600	1601.17
29.671	1600	1600.44	31.460	1600	1599.71
29.765	1600	1598.97	31.555	1600	1598.97
29.859	1600	1603.33	31.649	1600	1599.71
29.954	1600	1599.71	31.743	1600	1599.70
30.048	1600	1601.16	31.838	1600	1599.71
30.142	1600	1600.44	31.932	1600	1597.51

RMS error and variance from this test are compared with those of previous runs in Table 5.5. In general, performance of PI control with a 5 second sampling interval is better than that of adaptive control with a 20 second sampling interval. However, improvements in performance are not as substantial as those between adaptive control and PI control with 20 second sampling intervals.

Table 5.5: Summary of test runs

Controller Type	Run No.	Sampling Interval (sec)	RMS Error (°F)	df	Pooled Variance ($°F^2$)
Adaptive	1	20	1.97	35	4.46
Adaptive	2	20	2.35	115	10.36
Adaptive	3	20	1.88	120	9.36
PI	1	20	7.87	155	62.20
PI	2	5	1.66	214	4.80

6. CONCLUSION

The two-bed fluidized combustor in this study highlights both the benefits and the limitations of adaptive control. Perhaps the greatest advantage of adaptive control is the ability of the controller to identify system parameters, tune itself to optimal settings, and retune itself should process dynamics change. The adaptive tuning process worked well with the two-bed fluidized combustor, and after an initial tuning period, the controller could track the set point temperature and respond to system disturbances. Further, because the controller could estimate parameters on-line, system response was better than that with classical PI control.

Although system performance was improved, the adaptive controller had some restrictions. For example, the controller required supervision logic in the form of Jury's stability test to ensure proper parameter convergence. In addition, the parameter estimation scheme could not effectively separate nonstationary, colored noise from system dynamics; hence, to increase the signal to noise ratio, the sampling rate had to be decreased. A longer sampling interval resulted in a larger steady-state temperature variance as the controller could not immediately correct temperature deviations. Along similar lines, difficulties were encountered in modeling and identifying nonlinear system dynamics associated with disturbances, which resulted in a slow controller response to disturbances.

Clearly, adaptive control is not a panacea; rather, adaptive control is an effective design option that contends with nonlinear system dynamics through a linearized, time varying model. Although adaptive control worked well for the SISO case, a fluidized bed combustor is inherently a multiple input, multiple output (MIMO) system with coupled, nonlinear dynamics. However, the adaptive algorithm used in this study can be generalized to a MIMO control objective. For example, both bed temperature and combustion efficiency can be regulated through simultaneous adjustments of primary and secondary air flow rates. Since adaptive control seems to be particularly well suited for MIMO objectives, future work should address the characteristics and limitations of adaptive MIMO control on fluidized bed combustors using the same principles developed for the SISO case.

7. APPENDIX A: JURY'S STABILITY TEST

In the discrete-time domain, system stability can be decided through application of Jury's stability test, a method for determining whether or not roots of a characteristic equation lie within the complex unit circle.

Using right difference operator notation, Jury's stability test is applied to an nth order polynomial with real coefficients of the form

$$F(q) = a_n q^n + a_n - 1 q^{n-1} + \ldots + a_2 q^2 + a_1 q + a_o = 0 \qquad (7.1)$$

For a_n positive, the following table may be constructed [22].

Table 7.1: Jury's stability test

Row	q^0	q^1	q^2	...	q^{n-k}	...	q^{n-1}	q^n
1	a_o	a_1	a_2	...	a_{n-k}	...	a_{n-1}	a_n
2	a_n	a_{n-1}	a_{n-2}	...	a_k	...	a_1	a_o
3	b_o	b_1	b_2	...	b_{n-k}	...	b_{n-1}	
4	b_{n-1}	b_{n-2}	b_{n-3}	...	b_k	...	b_o	
5	c_o	c_1	c_2	...				
6	c_{n-2}	c_{n-3}	c_{n-4}	...				
$\vdots$	$\vdots$	$\vdots$	$\vdots$	...				
$2n-5$	p_o	p_1	p_2	p_3				
$2n-4$	p_3	p_2	p_1	p_o				
$2n-3$	s_o	s_1	s_2					

Note that elements in even rows are written in reverse order of elements in the pre-

ceding odd rows. Elements a_o, a_1, a_2, ... , a_n are coefficients of the characteristic equation, and elements b, c, ... , s are calculated from the determinants

$$b_k = \begin{vmatrix} a_o & a_{n-k} \\ a_n & a_k \end{vmatrix} \qquad c_k = \begin{vmatrix} b_o & b_{n-1-k} \\ b_{n-1} & b_k \end{vmatrix} \ldots \tag{7.2}$$

$$\ldots s_o = \begin{vmatrix} p_o & p_3 \\ p_3 & p_o \end{vmatrix} \qquad s_2 = \begin{vmatrix} p_o & p_1 \\ p_3 & p_2 \end{vmatrix}$$

All roots of the characteristic equation $F(q)$ lie strictly within the complex unit circle if and only if

$$\begin{aligned} F(1) &> 0 \\ F(-1) &> 0, \qquad n \in [2,4,6,8,\ldots] \\ F(-1) &< 0, \qquad n \in [3,4,5,7,\ldots] \end{aligned} \tag{7.3}$$

and

$$\begin{aligned} |a_o| &< a_n \\ |b_o| &> |b_{n-1}| \\ |c_o| &> |c_{n-2}| \\ |d_o| &> |d_{n-3}| \\ &\vdots \\ |q_o| &> |q_{n-2}| \end{aligned}$$

To determine the stability status of a second-order polynomial, the necessary and sufficient conditions reduce to

$$F(1) > 0$$

$$F(-1) > 0$$

$$|a_O| < a_n$$

Occasionally, some or all of the elements in a row of Table 7.1 are zero, providing indeterminate results. These singular cases are due to marginally stable systems, characterized by roots on the unit circle. The singularity can be removed by increasing or decreasing the radius of the unit circle by an infinitesimal amount ϵ. To change the radius, the right difference operator is replaced by

$$q = (1 + \epsilon)q \tag{7.4}$$

where powers of q can be approximated through the relation

$$(1 + \epsilon^n)q^n \simeq (1 + n\epsilon)q^n \tag{7.5}$$

Hand calculations can be simplified by using Raible's tabular form. In addition to simplified calculations, coefficients in Raible's tabulation can be used to determine the number of roots inside or outside the unit circle. For the singular case, the number of roots on the unit circle is simply the difference between the number of roots within a circle of radius $(1 + \epsilon)$ and the number of roots within a circle of radius $(1 - \epsilon)$. Details of Raible's tabular method are presented by Kuo [22].

8. APPENDIX B: SECONDARY AIR FLOW CONTROL

Of considerable importance to proper temperature control was effective regulation of secondary air flow rates. Actual flow rates had to be accurately maintained, independent of system disturbances, at values prescribed by the adaptive control algorithm. Air flow rate requests were updated every 20 seconds and required prompt valve response so as to minimize transient flow interactions with bed temperature.

Air flow rates into the fluidized bed were adjusted by a Fisher Design GS valve with a Fisher Type 513R reversible diaphragm actuator. The Type 513R actuator has a "fail closed" action, chosen over a "fail open" action so as to avert a pressure surge from entering the combustor in the event of power interruption. When attached to a 60 psig air line, the Design GS valve could admit a maximum volumetric air flow rate of 50 scfm into the combustor. However, a 27 scfm saturation limit was set within the control programs as high flow rates and concomitant high fluidization velocities promoted elutriation of bed material and fuel from the combustor.

Air pressures used to drive the Type 513R actuator were regulated by a Bellofram Type 1000 E/P Transducer. Using a 12-bit resolution D/A converter, digital signals were converted to 0–5 volt electrical signals and sent to the Bellofram transducer. Here, the transducer converted the electrical signals to 3–15 psig valve actuator command signals.

Volumetric air flow rates into the combustor were calibrated against a 12-bit digital signal, an integer ranging from 0 to 4095, corresponding to an actuator stroke of 0% to 100% of full travel, respectively. Air flow rates were recorded at ambient conditions as the 12-bit digital signal was first increased from 0 to 4000 and then decreased from 4000 to 0 by increments of 200. A typical calibration curve is shown in Fig. 8.1. Notice that a substantial hysteresis loop was present during valve operation. For a given input signal, flow rates may differ by an excess of 10 scfm — a significant influence on bed temperature. The large differential in air flow rates at a given input signal can be attributed to a combination of frictional forces within the valve packing and flow forces on the valve plug.

The flow hysteresis was corrected through position feedback control. A schematic and corresponding block diagram of the controller are illustrated in Fig. 8.2. The position feedback control loop is comprised of three main components: (1) a position transducer, (2) a signal conditioner, and (3) a control law.

Referring to Fig. 8.2, position signals were generated by a 50 kΩ (2" stroke) TCI linear potentiometer (1). The potentiometer was attached to the actuator rod and patched onto an EAI TR-20 analog computer. Position signals were amplified, inverted and scaled (2,3). For a 5 volt reference signal, actuator range could be set by adjusting potentiometer No. 3.

The inverted and scaled position signals were added to the microcomputer's reference signal at summing junction No. 4. The resulting error signal was sent through an inverter (5), an integrator (6), and then through another summing junction (7). At this point, the signal was amplified by a factor of ten and scaled by potentiome-

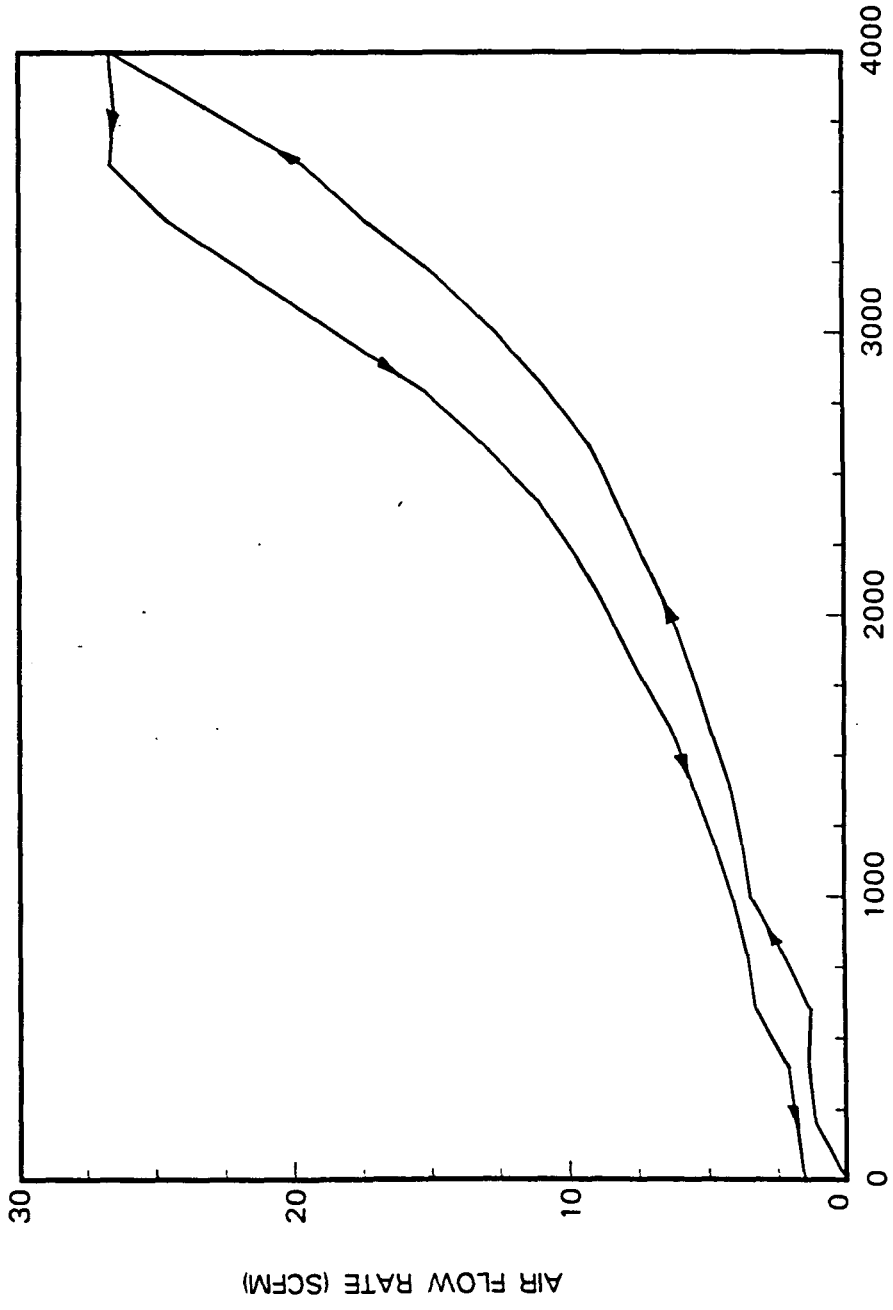


Figure 8.1: Air flow rate and digital command input calibration curve

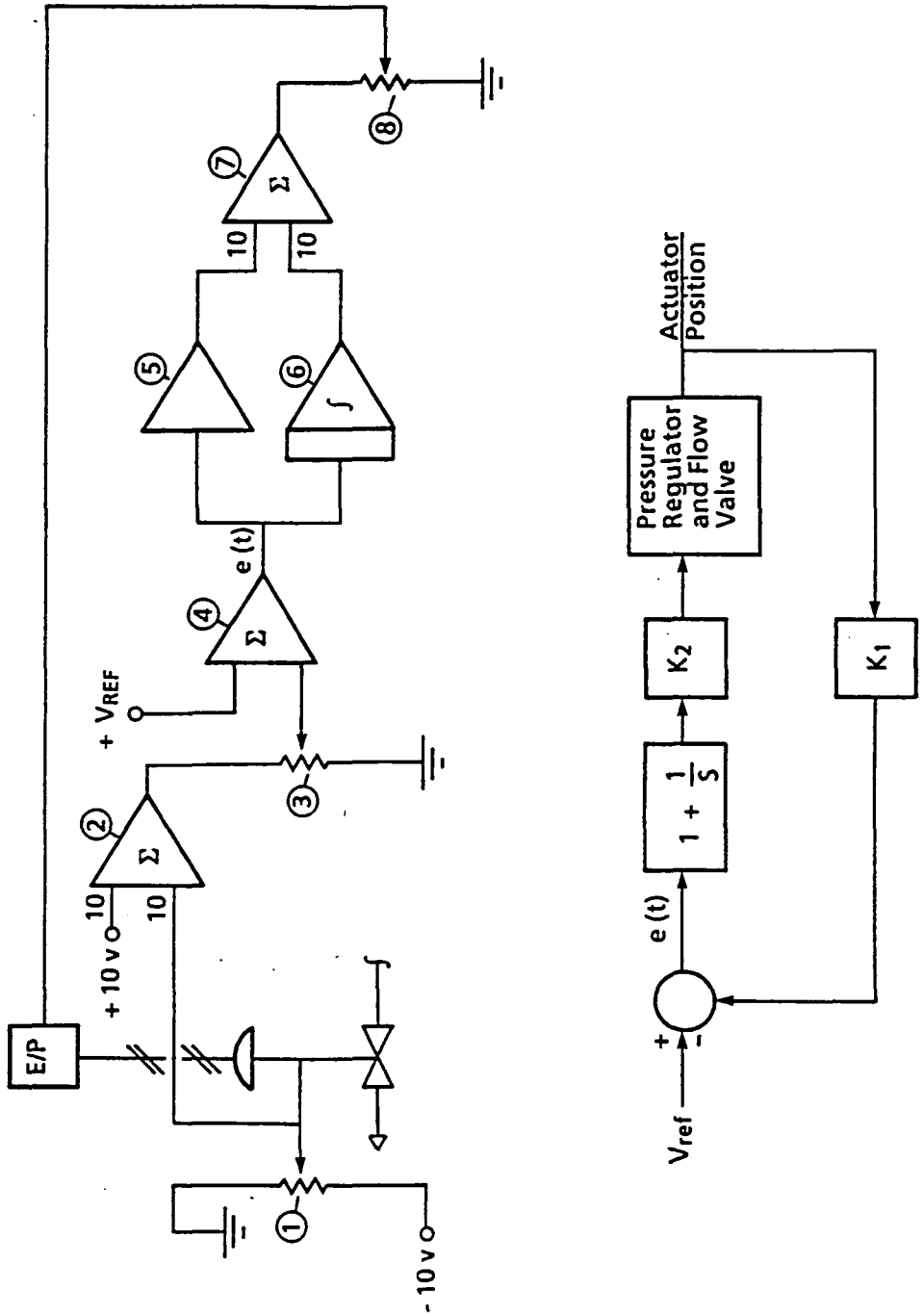


Figure 8.2: Detail of position feedback control

ter No. 8. An appropriate scaling factor was determined though an ultimate period design criterion. To complete the loop, output signals were sent to the Bellofram pressure transducer where they were converted to pneumatic command signals.

A typical calibration curve with position feedback control is shown in Fig. 8.3. Some hysteresis is still present near the lower limit of the operating range and can be attributed to frictional forces coupled with low command signals. Therefore, to avoid the difficulties associated with small flow rates, a lower saturation limit of 4.0 scfm was specified within both the classical and adaptive control algorithms.

Command signals sent to the position feedback controller were generated by an outer, feedback control subroutine. To compensate for nonlinear valve characteristics, the subroutine used both calibration data and PI control rules. Initially, digital command signals for requested air flow rates were interpolated from a calibration curve and sent to the position feedback controller. The valve system was allowed 0.8 seconds to respond before a discrete-time PI controller was invoked [1].

After sending a calibrated command signal, measured air flow rates were usually within 10% of requested air flow rates. In this range, nonlinear valve characteristics were reduced to a point where effective PI control could be implemented. Deviation parameters were taken about conditions at the commencement of PI control and were chosen so as to enforce zero initial conditions. Using Ziegler-Nichols rules to tune the digital loop, command signals were generated by control law

$$d(t) = 80e(t) + 50\int_{t1}^{t} e(t)dt; \tag{8.1}$$

[1] For differences in requested air flow rates less than 0.41 scfm, calibration control was bypassed altogether.

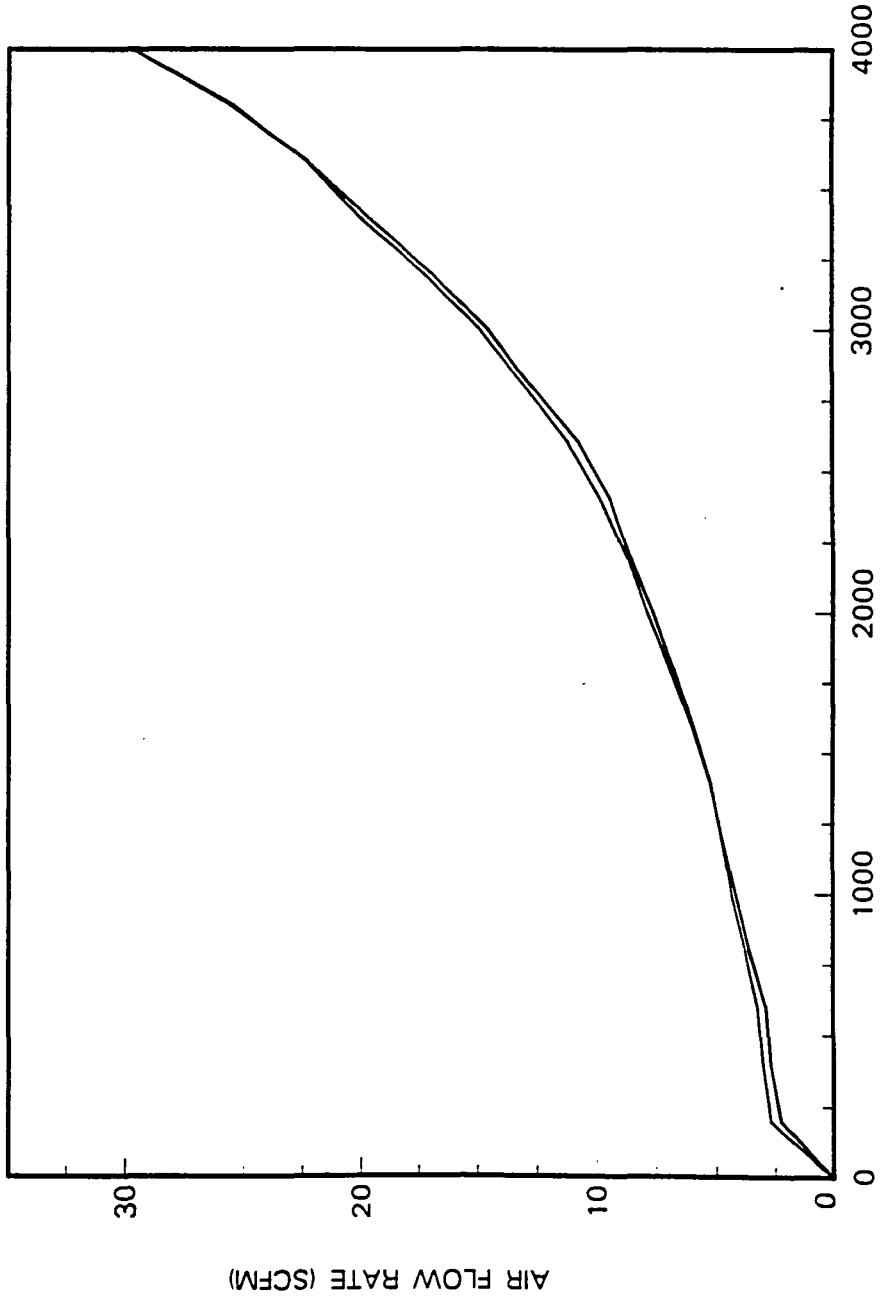


Figure 8.3: Air flow rate and digital command input calibration curve with position feedback control

$$e(t) = Q^*(t) - Q(t)$$

$$D_s(t) = Int[d(t) + D(t_1)], \qquad D_s(t) \in [0, 1, 2, \ldots, 4095]$$

where

$D_s(t)$ = 12-bit digital signal

$d(t)$ = deviation signal parameter

$Q(t)$ = measured air flow rate

$Q^*(t)$ = requested air flow rate

Integration was approximated by the Trapezoidal Rule with sampling intervals of 150 milliseconds. To prevent control deterioration from integral wind-up, integration of the error was terminated when output signals reached saturation limits and was resumed when output signals were no longer saturated.

Step responses of the valve with both the position feedback loop and the digital control loop are shown in Fig. 8.4. Response times were consistently under 2 seconds and steady-state variance was under 0.04 $scfm^2$. Hence, with both control loops, valve performance was more than adequate for the purposes of this investigation.

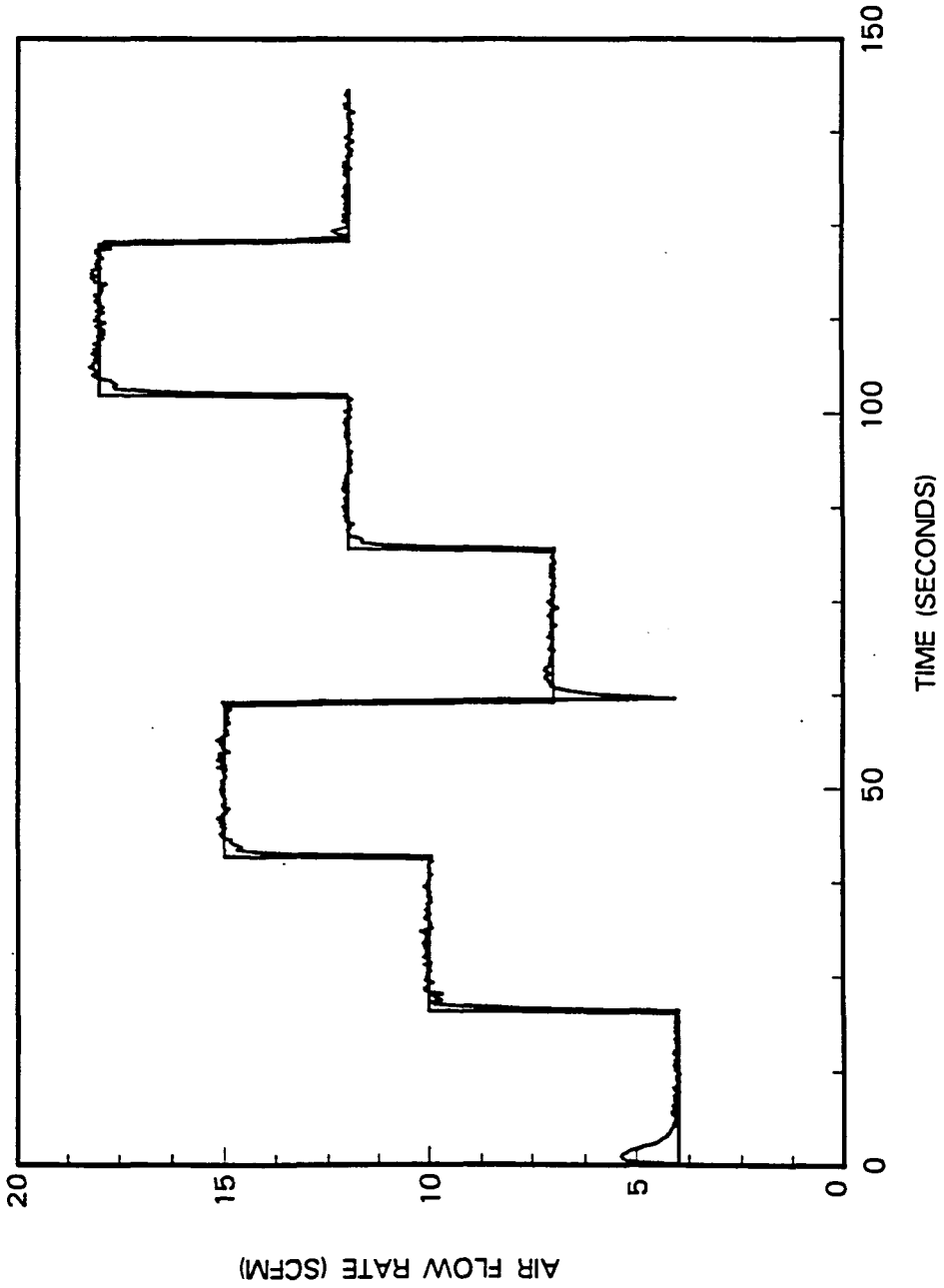


Figure 8.4: Response of flow valve with analog and digital control loops

9. APPENDIX C: SAS ANOVA

Results from the statistical analysis for both second- and sixth-order DARMA models are presented Tables 9.1 through 9.8. Tabulations include multiple regression ANOVAs and parameter estimates from PROC GLM as well as listings of covariance and partial regression matrices from PROC REG. Results are presented for the second-order DARMA model in Tables 9.1 through 9.4 and are followed by results for the sixth-order DARMA model in Tables 9.5 through 9.8.

Table 9.1: Regression analysis, 2nd-order DARMA model

DEPENDENT VARIABLE: y(t)

SOURCE	DF	SUM OF SQUARES	MEAN SQUARE	F VALUE	PR > F	R-SQUARE	C.V.
MODEL	4	307342.3653	76835.5913	10001.66	0.0	0.993963	5.6641
ERROR	243	1866.7952	7.6822	ROOT MSE	Y MEAN		
C. TOTAL	247	309209.1605		2.7717	48.9345		

SOURCE	DF	TYPE I SS	F VALUE	PR > F	DF	TYPE III SS	F VALUE	PR > F
$y(t-1)$	1	306329.7319	39874.82	0.0	1	1534.1901	199.70	0.0001
$u(t-1)$	1	914.0154	118.98	0.0001	1	1.9754	0.26	0.6126
$y(t-2)$	1	6.5827	0.86	0.3555	1	9.0280	1.18	0.2794
$u(t-2)$	1	92.0354	11.98	0.0006	1	92.0354	11.98	0.0006

Table 9.2: Coefficient estimates, 2nd-order DARMA model

PARAMETER	ESTIMATE	T FOR Ho: PARAMETER=0	PR > |T|	STD ERROR OF ESTIMATE
Intercept	3.5942	7.79	0.0001	0.4617
$y(t-1)$	-0.9272	14.13	0.0001	0.0656
$u(t-1)$	0.0717	0.51	0.6126	0.1415
$y(t-2)$	-0.0717	1.08	0.2794	0.0662
$u(t-2)$	-0.4811	-3.46	0.0006	0.1390

Table 9.3: Covariance of estimates, 2nd-order DARMA model

	Intercept	$y(t-1)$	$u(t-1)$	$y(t-2)$	$u(t-2)$
Intercept	0.21315	-0.01399	-0.00180	0.01299	-0.01302
$y(t-1)$	-0.01399	0.00431	0.00112	-0.00433	0.00059
$u(t-1)$	-0.00180	0.00112	0.02001	-0.00132	-0.01874
$y(t-2)$	0.01298	-0.00433	-0.00132	0.00437	-0.00042
$u(t-2)$	-0.01302	0.00059	-0.01874	-0.00042	0.01932

Table 9.4: Correlation of estimates, 2nd-order DARMA model

	Intercept	$y(t-1)$	$u(t-1)$	$y(t-2)$	$u(t-2)$
Intercept	1.0000	-0.4618	-0.0276	0.4253	-0.2028
$y(t-1)$	-0.4618	1.0000	0.1203	-0.9969	0.0642
$u(t-1)$	-0.0276	0.1203	1.0000	-0.1406	-0.9530
$y(t-2)$	0.4253	-0.9969	-0.1406	1.0000	-0.0460
$u(t-2)$	-0.2028	0.0642	-0.9530	-0.0460	1.0000

Table 9.5: Regression analysis, 6th-order DARMA model

DEPENDENT VARIABLE: y(t)

SOURCE	DF	SUM OF SQUARES	MEAN SQUARE	F VALUE	PR > F	R-SQUARE	C.V.
MODEL	12	307520.6765	25626.7230	3566.68	0.0	0.994539	5.4777
ERROR	235	1688.4840	7.1850	ROOT MSE	Y MEAN		
C. TOTAL	247	309209.1605		2.6804	48.9344		

SOURCE	DF	TYPE I SS	F VALUE	PR > F	DF	TYPE III SS	F VALUE	PR > F
$y(t-1)$	1	306329.7318	42634.39	0.0	1	1167.6077	162.51	0.0001
$u(t-1)$	1	914.0153	127.21	0.0001	1	12.4912	1.74	0.1886
$y(t-2)$	1	6.5827	0.92	0.3395	1	2.4967	0.35	0.5561
$u(t-2)$	1	92.0353	12.81	0.0004	1	46.7205	6.50	0.0114
$y(t-3)$	1	0.0737	0.01	0.9194	1	54.3870	7.57	0.0064
$u(t-3)$	1	20.3431	2.83	0.0938	1	2.3604	0.33	0.5671
$y(t-4)$	1	66.6105	9.27	0.0026	1	18.4149	2.56	0.1107
$u(t-4)$	1	0.2178	0.03	0.8619	1	16.4505	2.29	0.1316
$y(t-5)$	1	5.4187	0.75	0.3860	1	0.0198	0.00	0.9581
$u(t-5)$	1	56.6472	7.88	0.0054	1	11.8872	1.65	0.1996
$y(t-6)$	1	9.4652	1.32	0.2522	1	9.2639	1.29	0.2573
$u(t-6)$	1	19.5346	2.72	0.1005	1	19.5346	2.72	0.1005

Table 9.6: Coefficient estimates, 6th-order DARMA model

PARAMETER	ESTIMATE	T FOR Ho: PARAMETER=0	PR > |T|	STD ERROR OF ESTIMATE
Intercept	1.9913	2.79	0.0057	0.7138
$y(t-1)$	−0.8773	12.75	0.0001	0.0688
$u(t-1)$	0.2489	1.32	0.1886	0.1888
$y(t-2)$	−0.0654	0.59	0.5561	0.1110
$u(t-2)$	−0.7739	−2.55	0.0114	0.3035
$y(t-3)$	−0.3665	2.75	0.0064	0.1332
$u(t-3)$	0.1777	0.57	0.5671	0.3101
$y(t-4)$	0.2169	−1.60	0.1107	0.1355
$u(t-4)$	−0.4651	−1.51	0.1316	0.3074
$y(t-5)$	−0.0071	0.05	0.9581	0.1348
$u(t-5)$	0.3308	1.29	0.1996	0.2572
$y(t-6)$	0.0990	−1.14	0.2573	0.0872
$u(t-6)$	0.2414	1.65	0.1005	0.1464

Table 9.7: Covariance of estimates, 6th-order DARMA model

	Intercept	$y(t-1)$	$u(t-1)$	$y(t-2)$	$u(t-2)$	$y(t-3)$	$u(t-3)$
Intercept	0.50950	-0.01368	0.01224	-0.00873	0.00132	-0.00243	-0.00453
$y(t-1)$	-0.01368	0.00474	0.00118	-0.00437	-0.00244	0.00002	0.00359
$u(t-1)$	0.01224	0.00118	0.03563	-0.01239	-0.04502	0.01446	0.00922
$y(t-2)$	-0.00873	-0.00437	-0.01239	0.01233	0.01785	-0.00904	-0.00840
$u(t-2)$	0.00132	-0.00244	-0.04502	0.01785	0.09211	-0.03014	-0.05707
$y(t-3)$	-0.00243	0.00002	0.01446	-0.00904	-0.03014	0.01774	0.02075
$u(t-3)$	-0.00453	0.00359	0.00922	-0.00840	-0.05707	0.02075	0.09613
$y(t-4)$	0.00640	-0.00185	-0.00357	0.00275	0.01889	-0.00996	-0.03181
$u(t-4)$	-0.00393	-0.00071	-0.00646	0.00603	0.01710	-0.01004	-0.05769
$y(t-5)$	0.00252	0.00127	0.00185	-0.00322	-0.00608	0.00298	0.01944
$u(t-5)$	-0.02947	0.00161	0.00560	-0.00404	-0.01047	0.00467	0.01526
$y(t-6)$	0.01442	0.00020	-0.00170	0.00162	0.00196	-0.00176	-0.00360
$u(t-6)$	-0.02102	-0.00168	0.00052	0.00156	0.00298	0.00066	-0.00522

	$y(t-4)$	$u(t-4)$	$y(t-5)$	$u(t-5)$	$y(t-6)$	$u(t-6)$
Intercept	0.00640	-0.00393	0.00252	-0.02947	0.01442	-0.02102
$y(t-1)$	-0.00185	-0.00071	0.00127	0.00161	0.00020	-0.00168
$u(t-1)$	-0.00357	-0.00646	0.00185	0.00560	-0.00170	0.00052
$y(t-2)$	0.00275	0.00603	-0.00322	-0.00404	0.00162	0.00156
$u(t-2)$	0.01889	0.01710	-0.00608	-0.01047	0.00196	0.00298
$y(t-3)$	-0.00996	-0.01004	0.00298	0.00467	-0.00176	0.00066
$u(t-3)$	-0.03181	-0.05769	0.01944	0.01526	-0.00360	-0.00522
$y(t-4)$	0.01836	0.02079	-0.01044	-0.00780	0.00114	0.00279
$u(t-4)$	0.02079	0.09447	-0.03085	-0.05113	0.01483	0.00387
$y(t-5)$	-0.01044	-0.03085	0.01817	0.01881	-0.00879	-0.00330
$u(t-5)$	-0.00780	-0.05113	0.01881	0.06616	-0.01329	-0.02197
$y(t-6)$	0.00114	0.01483	-0.00879	-0.01329	0.00761	0.00009
$u(t-6)$	0.00279	0.00387	-0.00330	-0.02197	0.00009	0.02144

Table 9.8: Correlation of estimates, 6th-order DARMA model

	Intercept	$y(t-1)$	$u(t-1)$	$y(t-2)$	$u(t-2)$	$y(t-3)$	$u(t-3)$
Intercept	1.00000	-0.27850	0.09090	-0.11010	0.00610	-0.02560	-0.02040
$y(t-1)$	-0.27850	1.00000	0.09090	-0.57190	-0.11660	0.00200	0.16820
$u(t-1)$	0.09090	0.09090	1.00000	-0.59110	-0.78600	0.57500	0.15760
$y(t-2)$	-0.11010	-0.57190	-0.59110	1.00000	0.52980	-0.61130	-0.24390
$u(t-2)$	0.00610	-0.11660	-0.78600	0.52980	1.00000	-0.74550	-0.60650
$y(t-3)$	-0.02560	0.00200	0.57500	-0.61130	-0.74550	1.00000	0.50240
$u(t-3)$	-0.02040	0.16820	0.15760	-0.24390	-0.60650	0.50240	1.00000
$y(t-4)$	0.06610	-0.19890	-0.13960	0.18250	0.45940	-0.55190	-0.75710
$u(t-4)$	-0.01790	-0.03340	-0.11130	0.17680	0.18330	-0.24530	-0.60530
$y(t-5)$	0.02610	0.13680	0.07260	-0.21520	-0.14870	0.16610	0.46520
$u(t-5)$	-0.16050	0.09070	0.11540	-0.14140	-0.13410	0.13630	0.19140
$y(t-6)$	0.23170	0.03380	-0.10320	0.16770	0.07410	-0.15190	-0.13300
$u(t-6)$	-0.20120	-0.16680	0.01880	0.09610	0.06700	0.03370	-0.11500

	$y(t-4)$	$u(t-4)$	$y(t-5)$	$u(t-5)$	$y(t-6)$	$u(t-6)$
Intercept	0.06610	-0.01790	0.02610	-0.16050	0.23170	-0.20120
$y(t-1)$	-0.19890	-0.03340	0.13680	0.09070	0.03380	-0.16680
$u(t-1)$	-0.13960	-0.11130	0.07260	0.11540	-0.10320	0.01880
$y(t-2)$	0.18250	0.17680	-0.21520	-0.14140	0.16770	0.09610
$u(t-2)$	0.45940	0.18330	-0.14870	-0.13410	0.07410	0.06700
$y(t-3)$	-0.55190	-0.24530	0.16610	0.13630	-0.15190	0.03370
$u(t-3)$	-0.75710	-0.60530	0.46520	0.19140	-0.13300	-0.11500
$y(t-4)$	1.00000	0.49920	-0.57160	-0.22380	0.09680	0.14080
$u(t-4)$	0.49920	1.00000	-0.74450	-0.64670	0.55310	0.08610
$y(t-5)$	-0.57160	-0.74450	1.00000	0.54260	-0.74770	-0.16720
$u(t-5)$	-0.22380	-0.64670	0.54260	1.00000	-0.59230	-0.58340
$y(t-6)$	0.09680	0.55310	-0.74770	-0.59230	1.00000	0.00740
$u(t-6)$	0.14080	0.08610	-0.16720	-0.58340	0.00740	1.00000

10. APPENDIX D: CONTROLLER CODES

Program codes for both the adaptive and classical control algorithms are presented in this appendix. Code for the adaptive controller is listed in JLSQ2.BAS and is followed by code for the classical PI controller in PI20.BAS.

```
'-----------------------------------------------------------------
'******************** Main Module: JLSQ2.BAS ***********************
'-----------------------------------------------------------------
' Rev 1.0:  25 May 1989
' Rev 7.3:  18 November 1989

' This program uses an adaptive control algorithm to command annular
' bed air flow rate in a coal-fired, two-bed fluidized combustor.
' The program is based on a weighted least-squares parameter
' estimator (SUB WLSq), an observer (SUB Observer), and a linear
' quadratic Gaussian optimal-control design procedure (SUB LQG).
' Codes were written and compiled with Microsoft QuickBASIC 4.0.
' Data are written to output files using Lotus 1-2-3 File Import
' format.

'------------------------ Nomenclature (*) ------------------------

'       Act.flow   = Actual secondary air flow rate, scfm
'       Begin.prog = Time program is initiated
'       Board      = Sub-multiplexer board number
'       Channel    = Sub-multiplexer board channel
'       Caldd      = Digital command signal data from calibration
'                    curve with position feedback control
'       Calflow    = Secondary air flow rate data from calibration
'                    curve with position feedback control, scfm
'       Chk        = System initialization flag
'       Dev.T      = Initial system temperature, F
'       Dev.u      = Initial system air flow rate, scfm
'       DD         = Digital command signal to Metrabyte DDA-06
'       F          = Feedback gain vector
'       Flag       = Error flag
'       Flow       = Interpolated air flow rate
'       G          = Observer gain vector
'       Iter       = Iteration counter
'       Lambda     = Weighting coefficient
'       Ma         = Workspace matrix (nxn)
'       Mb         = Workspace matrix (nxn)
'       Mc         = Workspace vector (nx1)
```

```
'       Md          = Workspace vector (nx1)
'       md%         = DAS-8 mode
'       N           = Number of DARMA model coefficients
'       nk          = Number of data in Type-K thermocouple lookup
'                     table
'       Orif        = Orifice flow meter calibration data
'       Ornum       = Orifice flow meter number
'                         1 -- primary air
'                         2 -- N/A
'                         3 -- secondary air
'       P1          = Covariance matrix, indexed at time (t-1)
'       Phi1        = Variate vector, indexed at time (t-1)
'       Range       = Range settings for gas analyzers
'       Rqd.flow    = Required secondary air flow rate, scfm
'       Rxx         = Weighting matrix, state vector
'       Ru          = Weighting matrix, input variable
'       SetPoint    = Set point temperature, F
'       sik         = Voltage step interval in Type-K thermocouple
'                     lookup table, mV
'       svk         = Starting voltage in Type-K thermocouple lookup
'                     table, mV
'       Temp        = Thermocouple temperature vector
'       TF          = Temperature, F
'       TC          = Temperature, C
'       Theta       = Coefficient vector at time (t)
'       Theta1      = Coefficient vector at time (t-1)
'       Tset        = Deviation set point temperature, F
'       U           = Deviation air flow rate, scfm, indexed
'                     at time (t)
'       U1          = Deviation air flow rate, scfm, indexed
'                     at time (t-1)
'       U2          = Deviation air flow rate, scfm, indexed
'                     at time (t-2)
'       ValveNo     = Valve number
'                         1 -- water
'                         2 -- primary air
'                         3 -- secondary air
'       Xhat        = Observer state vector
'       Xiter       = Iteration counter
```

```
'       Y           = Deviation temperature, F, indexed at time (t)
'       Y1          = Deviation temperature, F, indexed at time (t-1)
'       Y2          = Deviation temperature, F, indexed at time (t-2)
'       Yhat        = RLS best-estimate bed temperature
'       Yout        = Workspace matrix (nxn)
'       Z           = Integration variable
'       Zout        = Workspace matrix (nxn)

' (*) Variables are double precision unless otherwise indicated.

'------------------ Subprogram Definitions ------------------------

'       Add             Adds two (nx1) or (1xn) vectors
'       Addnn           Adds two (nxn) matrices
'       Cntrl           Controls secondary air flow rate.
'       Coldjunct       Finds cold junction compensation voltage
'       DAS8            Metrabyte DAS8 subprograms (library file)
'       Display         Screen headings overlay subprogram
'       Interp          Calculates air flow rate from orifice meter
'                       pressure drop readings
'       Interpolation   Calculates temperature from thermocouple
'                       voltage readings
'       Jury            Examines roots of the A(q) polynomial with
'                       Jury's stability test
'       LQG             Linear Quadratic Gaussian optimal-control
'                       design procedure (Goodwin and Sin, 1984)
'       Mult1nxn1       Multiplies a (1xn) vector with an (nx1) vector
'       Mult1nxnn       Multiplies a (1xn) vector with an (nxn) matrix
'       Multn1x1n       Multiplies an (nx1) vector with a (1xn) vector
'       Multn1xc        Multiplies an (nx1) vector with a constant
'       Multnnxc        Multiplies an (nxn) matrix with a constant
'       Multnnxn1       Multiplies an (nxn) matrix with an (nx1) vector
'       Multnnxnn       Multiplies an (nxn) matrix with an (nxn) matrix
'       Observer        Observer subprogram
'       Pr              Reads the pressure differential across the
'                       orifice flow meter
'       Readcal         Reads air flow data from the calibration
'                       curve with position feedback control
'       Subtract        Subtracts two vectors
```

```
'       Subtractnxn     Subtracts two matrices
'       Temperature     Reads voltages from thermocouples
'       Transnxn        Transposes an (nxn) matrix
'       Valve           Sends digital command signal to Bellofram
'                       Type 1000 E/P transducer
'       WLSq            Recursive, weighted least-squares subprogram
'       Xcntrl          Specifies requested air flow rate and
'                       checks for saturation.

'--------------------- Begin Program ------------------------------

DEFDBL A-Z
DECLARE SUB Jury (N#, Theta1#(), Theta#())
DECLARE SUB Observer (N#, Theta#(), U#, Y#, Xhat#())
DECLARE SUB Cntrl (Rqd.flow#, Act.flow#)
DECLARE SUB Transnxn (N#, Ma#(), Yout#())
DECLARE SUB Subtractnxn (N#, Ma#(), Mb#(), Yout#())
DECLARE SUB Interpolation (TF#, Voltage#(), Board%, Chan%)
DECLARE SUB Display ()
DECLARE SUB Coldjunct ()
DECLARE SUB Valve (ValveNo#, DD#)
DECLARE SUB Temperature (Temp#())
DECLARE SUB Pr (Ornum#, flow#)
DECLARE SUB Readcal ()
DECLARE SUB Interp (DD#, Rqd.flow#)
DECLARE SUB WLSq (N#, Y#, U#, Theta#(), yhat#)
DECLARE SUB Multinxn1 (N#, Mc#(), Md#(), Xout#)
DECLARE SUB Subtract (N#, Mc#(), Md#(), Zout#())
DECLARE SUB Multnnxn1 (N#, Ma#(), Mc#(), Zout#())
DECLARE SUB Multnnxnn (N#, Ma#(), Mb#(), Yout#())
DECLARE SUB Multn1x1n (N#, Mc#(), Md#(), Yout#())
DECLARE SUB Multn1xc (N#, Mc#(), x#, Zout#())
DECLARE SUB Add (N#, Mc#(), Md#(), Zout#())
DECLARE SUB LQG (N#, Theta#(), F#())
DECLARE SUB Multinxnn (N#, Mc#(), Ma#(), Zout#())
DECLARE SUB Addnn (N#, Ma#(), Mb#(), Yout#())
DECLARE SUB Multnnxc (N#, Ma#(), x#, Yout#())
DECLARE SUB Xcntrl (F#(), Xhat#(), Y#, Tset#, U#)
DECLARE SUB Das8 (Md%, BYVAL dummy%, Flag%)
```

```
COMMON SHARED D%(), Digital%()
COMMON SHARED Calflow(), Caldd(), Orif(), pressure, zoff
COMMON SHARED Tk(), Volt(), Range(), Gain()
COMMON SHARED nk, sik, svk, cjc, Begin.prog, Start.control
COMMON SHARED Z
N = 5
'------------------ Data Acquisition Variables --------------------

DIM Calflow(50), Caldd(50), Orif(6, 2)
DIM D%(6), Range(5), Volt(5), Gain(3), Digital%(3, 16)
DIM Tk(308)
DIM SHARED Temp(1, 4)

'----------------------- LQG Variables ----------------------------

DIM SHARED A(N, N), B(N), C(N)
DIM SHARED Rxx(N, N), S(N, N), L(N), F(N)

'--------------------- Workspace Variables -------------------------

DIM SHARED Ma(N, N), Mb(N, N), Mc(N), Md(N), Zout(N), Yout(N, N)
DIM SHARED Temp1(N, N), Temp2(N, N), Temp3(N, N), Temp4(N, N)
DIM SHARED Sto1(N), Sto2(N), Sto3(N)

'----------------------- WLSq Variables ---------------------------

DIM SHARED Theta(N), P2(N, N), Theta1(N)
DIM SHARED Num(N, N), P1(N, N)

'---------------------- Observer Variables ------------------------

DIM SHARED Xhat(N)

'--------------------- System Initialization ----------------------

CLS
PRINT "Initializing System ..."
Begin.prog = TIMER
KEY(1) ON: KEY(2) ON: KEY(10) ON
```

```
ON KEY(1) GOSUB key1
ON KEY(2) GOSUB key2
ON KEY(10) GOSUB key10

'------------ Orifice Flow Meter Calibration Data -----------------

DATA 2.3426,0.51324,1.8196,0.50523,1.2647,0.51488,1.3359,0.5005
DATA 0.8500,0.53392,0.34717,0.49591

FOR III = 1 TO 6
    FOR JJJ = 1 TO 2
        READ Orif(III, JJJ)
    NEXT JJJ
NEXT III

'--------------- Table Lookup Data for Type-K Thermocouple --------------

DATA  309 ,  .2 , -6.6
READ nk, sik, svk

DATA -353.5,-249.3,-224.0,-207.6,-194.3,-182.8,-172.3,-162.8,-153.8,-145.4
DATA -137.3,-129.6,-122.3,-115.2,-108.3,-101.6, -95.1, -88.7, -82.5, -76.4
DATA  -70.4, -64.6, -58.8, -53.1, -47.5, -42.0, -36.6, -31.2, -25.9, -20.6
DATA  -15.4, -10.2,  -5.1,  -0.0,   5.0,  10.1,  15.1,  20.0,  25.0,  29.9
DATA   34.8,  39.7,  44.6,  49.5,  54.3,  59.1,  64.0,  68.8,  73.6,  78.4
DATA   83.2,  88.0,  92.9,  97.7, 102.5, 107.4, 112.2, 117.1, 122.0, 126.9
DATA  131.8, 136.7, 141.7, 146.6, 151.6, 156.5, 161.5, 166.5, 171.5, 176.5
DATA  181.6, 186.6, 191.6, 196.6, 201.6, 206.6, 211.6, 216.6, 221.5, 226.5
DATA  231.5, 236.4, 241.4, 246.3, 251.2, 256.1, 261.0, 265.9, 270.8, 275.6
DATA  280.5, 285.3, 290.2, 295.0, 299.8, 304.6, 309.4, 314.3, 319.1, 323.9
DATA  328.7, 333.4, 338.2, 343.0, 347.8, 352.6, 357.3, 362.1, 366.9, 371.6
DATA  376.4, 381.1, 385.9, 390.6, 395.4, 400.1, 404.8, 409.6, 414.3, 419.0
DATA  423.8, 428.5, 433.2, 437.9, 442.6, 447.3, 452.0, 456.8, 461.5, 466.2
DATA  470.9, 475.6, 480.3, 485.0, 489.7, 494.4, 499.1, 503.8, 508.5, 513.1
DATA  517.8, 522.5, 527.2, 531.9, 536.6, 541.3, 546.0, 550.7, 555.4, 560.0
DATA  564.7, 569.4, 574.1, 578.8, 583.5, 588.2, 592.9, 597.6, 602.3, 607.0
DATA  611.7, 616.4, 621.2, 625.9, 630.6, 635.3, 640.0, 644.8, 649.5, 654.2
DATA  658.9, 663.7, 668.4, 673.2, 677.9, 682.7, 687.4, 692.2, 696.9, 701.7
DATA  706.5, 711.3, 716.1, 720.8, 725.6, 730.4, 735.2, 740.0, 744.8, 749.7
```

```
DATA  754.5, 759.3, 764.1, 769.0, 773.8, 778.7, 783.5, 788.4, 793.3, 798.1
DATA  803.0, 807.9, 812.8, 817.7, 822.6, 827.5, 832.4, 837.3, 842.2, 847.2
DATA  852.1, 857.1, 862.0, 867.0, 872.0, 876.9, 881.9, 886.9, 891.9, 896.9
DATA  901.9, 906.9, 911.9, 916.9, 922.0, 927.0, 932.0, 937.1, 942.2, 947.2
DATA  952.3, 957.4, 962.5, 967.6, 972.7, 977.8, 982.9, 988.0, 993.1, 998.2
DATA 1003.4,1008.5,1013.7,1018.8,1024.0,1029.2,1034.4,1039.6,1044.8,1050.0
DATA 1055.2,1060.4,1065.6,1070.8,1076.1,1081.3,1086.6,1091.9,1097.2,1102.4
DATA 1107.7,1113.0,1118.3,1123.7,1129.0,1134.3,1139.7,1145.0,1150.4,1155.8
DATA 1161.2,1166.6,1172.0,1177.4,1182.9,1188.3,1193.8,1199.2,1204.7,1210.2
DATA 1215.7,1221.2,1226.8,1232.3,1237.9,1243.5,1249.1,1254.7,1260.3,1265.9
DATA 1271.6,1277.3,1282.9,1288.6,1294.3,1300.1,1305.8,1311.5,1317.3,1323.1
DATA 1328.9,1334.7,1340.5,1346.4,1352.2,1358.1,1363.9,1369.8,1375.7

FOR i = 0 TO nk - 1
    READ Tk(i)
NEXT i

'--------------------- Ranges for Gas Analyzers --------------------

Range(1) = 25: Volt(1) = 5                          ' Oxygen
Range(2) = 1.2: Volt(2) = 5                         ' Carbon Monoxide
Range(3) = 30: Volt(3) = 5                          ' Carbon Dioxide
Range(4) = 2000: Volt(4) = 1                        ' Sulfur Dioxide
Range(5) = 1000: Volt(5) = 1 .                      ' NOx

'--------------- Gains for Metrabyte EXP-16's ---------------------

Gain(1) = 50
Gain(2) = 1000
Gain(3) = 50

'---------------------- System Constants -------------------------

8 :
INPUT "Enter Orcal Offset:  ", zoff
PRINT "Edit? (y/[n]):  "
10 : ans$ = INKEY$: IF ans$ = "" GOTO 10
IF ans$ = "y" OR ans$ = "Y" THEN GOTO 8
pressure = 60 + 14.7
```

```
ValveNo = 3
Ornum = 3
Start.control = 0

' Open secondary air transducer.

PRINT "Closing All Valves ..."

DD = 0: v = 1: CALL Valve(v, DD)
DD = 0: v = 2: CALL Valve(v, DD)
DD = 0: v = 3: CALL Valve(v, DD)

PRINT "Opening Pressure Transducer ..."
OUT 773, 2
t1 = TIMER

'Let initial transient settle down

DO: t2 = TIMER: LOOP UNTIL t2 - t1 > 3.5

' Find cold junction temperature.

PRINT "Reading Cold Junction Temperature ..."
CALL Coldjunct

' Open data file

OPEN "c:\Thermo2.prn" FOR OUTPUT AS #1
OPEN "c:\Intern2.prn" FOR OUTPUT AS #2
OPEN "c:\Param2.prn" FOR OUTPUT AS #3
WRITE #1, "--TIMER--/--SetPoint--/--temp1--/--temp2--/--temp3--
          /--temp4--/--rqd.flow--/--act.flow--/"
WRITE #2, "--TIMER--/--SetPoint--/--f(1)--/--f(2)--/--f(3)--/--u--
          /--Tset-- /--z--/"
WRITE #3, "--TIMER--/--theta1--/--theta2--/--theta3--/--theta4--
          /--theta5--/--xhat1--/--xhat2--/--y--/--dev.t--/--Yhat--/"

PRINT "Reading Calibration ..."
CALL Readcal
```

```
'----------------- End of System Initialization --------------------

PRINT " ----> Done <----"
CLS

' Loop here until adaptive controller is invoked.

CALL Display

DO
    Rqd.flow = 4!
    CALL Cntrl(Rqd.flow, Act.flow)
    CALL Temperature(Temp())

    WRITE #1, CSNG(TIMER), SetPoint, CSNG(Temp(1, 1)),
              CSNG(Temp(1, 2)), CSNG(Temp(1, 3)),
              CSNG(Temp(1, 4)), CSNG(Rqd.flow), CSNG(Act.flow)
    WRITE #2, CSNG(TIMER), SetPoint, CSNG(F(1)), CSNG(F(2)),
              CSNG(F(3)), CSNG(U), CSNG(Tset), CSNG(Z)
    WRITE #3, CSNG(TIMER), CSNG(Theta(1)), CSNG(Theta(2)),
              CSNG(Theta(3)), CSNG(Theta(4)), CSNG(Theta(5)),
              CSNG(Xhat(1)), CSNG(Xhat(2))

LOOP UNTIL Start.control <> 0

' ------------------- Begin control program -----------------------
' Initialize and check system

LOCATE 22, 8: PRINT "Initializing Adaptive Controller"
iter = 0

DO
    iter = iter + 1
    CALL Temperature(Temp())
    comp1 = Temp(1, 1)
    comp2 = Temp(1, 2)
LOOP UNTIL ABS(comp1 - comp2) < 5 OR iter > 30
```

```
Dev.T = Temp(1, 1)
Tset = SetPoint - Dev.T
Y = 0
CALL Pr(Ornum, flow)
Dev.u = flow
U = 0
CALL Xcntrl(F(), Xhat(), Y, Tset, U)
CALL WLSq(N, Y, U, Theta(), yhat)
CALL Observer(N, Theta(), U, Y, Xhat())
LOCATE 22, 8: PRINT "                                    "

' End controller initialization

DO
    TT1 = TIMER
    Tset = SetPoint - Dev.T
    Rqd.flow = U + Dev.u
    IF Rqd.flow > 23 THEN Rqd.flow = 23
    IF Rqd.flow < 4 THEN Rqd.flow = 4
    CALL Cntrl(Rqd.flow, Act.flow)

'   Temperature spike filter

    tempsum = 0
    FOR xiter = 1 TO 10
        iter = 0
        DO
            iter = iter + 1
            CALL Temperature(Temp())
            comp1 = Temp(1, 1)
            comp2 = Temp(1, 2)
        LOOP UNTIL ABS(comp1 - comp2) < 5 OR iter > 20
        tempsum = tempsum + Temp(1, 1)
    NEXT xiter

    Y = tempsum / 10 - Dev.T
    U = Rqd.flow - Dev.u
    CALL WLSq(N, Y, U, Theta(), yhat)
    CALL LQG(3, Theta(), F())
```

```
    CALL Observer(N, Theta(), U, Y, Xhat())
    CALL Xcntrl(F(), Xhat(), Y, Tset, U)

    LOCATE 11, 5: PRINT USING "#####.####"; Theta(1)
    LOCATE 12, 5: PRINT USING "#####.####"; Theta(2)
    LOCATE 13, 5: PRINT USING "#####.####"; Theta(3)
    LOCATE 14, 5: PRINT USING "#####.####"; Theta(4)
    LOCATE 15, 5: PRINT USING "#####.####"; Theta(5)
    DO: LOOP UNTIL TIMER - TT1 > 20!
    WRITE #1, CSNG(TIMER), SetPoint, CSNG(Temp(1, 1)),
              CSNG(Temp(1, 2)), CSNG(Temp(1, 3)),
              CSNG(Temp(1, 4)), CSNG(Rqd.flow), CSNG(Act.flow)
    WRITE #2, CSNG(TIMER), SetPoint, CSNG(F(1)), CSNG(F(2)),
              CSNG(F(3)), CSNG(U), CSNG(Tset), CSNG(Z)
    WRITE #3, CSNG(TIMER), CSNG(Theta(1)), CSNG(Theta(2)),
              CSNG(Theta(3)), CSNG(Theta(4)), CSNG(Theta(5)),
              CSNG(Xhat(1)), CSNG(Xhat(2)), CSNG(Y),
              CSNG(Dev.T), CSNG(yhat)
LOOP
STOP

key1:
     CLS
     INPUT "Enter Set Point:  ", SetPoint
     CLS
     CALL Display
     LOCATE 10, 30: PRINT USING "####.##"; SetPoint
     RETURN
key2:
     CLS
     INPUT "Enter Set Point:  ", SetPoint
     CLS
     CALL Display
     LOCATE 10, 30: PRINT USING "####.##"; SetPoint
     Start.control = 1
     RETURN
key10:
     CLOSE #1
     CLOSE #2
```

```
      CLOSE #3
      OUT 773, 0
      DD = 0: v = 1: CALL Valve(v, DD)
      DD = 0: v = 2: CALL Valve(v, DD)
      DD = 0: v = 3: CALL Valve(v, DD)
      STOP
RETURN
STOP
'-------------------------------------------------------------------
'****************** End Main Module: JLSQ2.BAS ********************
'-------------------------------------------------------------------


DEFDBL A-Z
SUB Add (N, Mc(), Md(), Zout())
 FOR i = 1 TO N
     Zout(i) = Mc(i) + Md(i)
 NEXT i
END SUB

DEFDBL A-Z
SUB Addnn (N, Ma(), Mb(), Yout())
 FOR i = 1 TO N
     FOR J = 1 TO N
         Yout(i, J) = Ma(i, J) + Mb(i, J)
     NEXT J
 NEXT i
END SUB

DEFDBL A-Z
SUB Cntrl (Rqd.flow, Act.flow) STATIC
 ts = TIMER
 delay = 18
 Ornum = 3
 ValveNo = 3
 LOCATE 17, 5

'---------------------- Rough Controller ----------------------------

 IF Rqd.flow > 27 THEN
```

```
    Rqd.flow = 27
    DD = 4095
    CALL Valve(ValveNo, DD)
    DO
        CALL Pr(Ornum, flow)
        LOCATE 17, 23: PRINT USING "####.##"; Rqd.flow
        LOCATE 18, 23: PRINT USING "####.##"; flow
        CALL Temperature(Temp())
        LOCATE 7, 10:
        PRINT USING " #####.##   ";Temp(1, 1),Temp(1, 2), Temp(1, 3)
        LOCATE 8, 10:
        PRINT USING " #####.##"; Temp(1, 4)
        t2 = TIMER
        LOCATE 3, 47: PRINT USING "####.###"; (t2 - Begin.prog) / 60
    LOOP UNTIL TIMER - ts > delay
    GOTO 200:
ELSEIF Rqd.flow < 0 THEN Rqd.flow = 0
    Rqd.flow = 0
    DD = 0
    CALL Valve(ValveNo, DD)
    DO
        CALL Pr(Ornum, flow)
        LOCATE 17, 23: PRINT USING "####.##"; Rqd.flow
        LOCATE 18, 23: PRINT USING "####.##"; flow
        CALL Temperature(Temp())
        LOCATE 7, 10:
        PRINT USING " #####.##  ";Temp(1, 1), Temp(1, 2), Temp(1, 3)
        LOCATE 8, 10: PRINT USING " #####.##"; Temp(1, 4)
        t2 = TIMER
        LOCATE 3, 47: PRINT USING "####.###";(t2-Begin.prog)/60
    LOOP UNTIL TIMER - ts > delay
    GOTO 200
END IF

CALL Pr(Ornum, flow)
e = flow - Rqd.flow
IF ABS(e) < .41 THEN
    GOTO 110:
END IF
IF ABS(e) < 3 THEN
```

```
    CALL Interp(DD, Rqd.flow)
    CALL Valve(ValveNo, DD)
    GOTO 100:
END IF
t1 = TIMER
DO
    t2 = TIMER
    CALL Interp(DD, Rqd.flow)
    CALL Valve(ValveNo, DD)
    CALL Pr(Ornum, flow)
    LOCATE 17, 23: PRINT USING "####.##"; Rqd.flow
    LOCATE 18, 23: PRINT USING "####.##"; flow
    CALL Temperature(Temp())
    LOCATE 7, 10:
    PRINT USING " #####.##  "; Temp(1, 1), Temp(1, 2), Temp(1, 3)
    LOCATE 8, 10:
    PRINT USING " #####.##"; Temp(1, 4)
    LOCATE 3, 47: PRINT USING "####.###"; (t2 - Begin.prog) / 60
    IF t2 - ts > delay THEN GOTO 200:
LOOP UNTIL t2 - t1 > .8 AND ABS(flow - Rqd.flow) / Rqd.flow < .1

100 :

'---------------- Perturbation PI Controller ---------------------

CALL Pr(Ornum, flow)
t1 = TIMER
integral = 0
kp = 80
ki = 50
e = Rqd.flow - flow
e.old = e
D.nom = DD

110 :
DO
    t2 = TIMER
    DT = t2 - t1
    CALL Pr(Ornum, flow)
```

```
      e = Rqd.flow - flow
      integral = integral + .5 * (e + e.old) * DT
      d2 = INT(kp * e + ki * integral)
      DD.new = D.nom + d2
      IF DD.new > 4095 THEN DD.new = 4095
      IF DD.new < 0 THEN DD.new = 0

      CALL Valve(ValveNo, DD.new)
      LOCATE 17, 23: PRINT USING "####.##"; Rqd.flow
      LOCATE 18, 23: PRINT USING "####.##"; flow
      CALL Temperature(Temp())
      LOCATE 7, 10:
      PRINT USING " #####.## "; Temp(1, 1), Temp(1, 2), Temp(1, 3)
      LOCATE 8, 10:
      PRINT USING " #####.##"; Temp(1, 4)
      LOCATE 3, 47: PRINT USING "####.###"; (t2 - Begin.prog) / 60
      e.old = e
      t1 = t2
 LOOP UNTIL TIMER - ts > delay

200 :
 CALL Pr(Ornum, flow)
 Act.flow = flow
END SUB

DEFDBL A-Z
SUB Coldjunct
' Find cold junction temperature (24.4 mV/C).

 adr% = 768
 Md% = 0: Flag% = 0
 CALL Das8(Md%, VARPTR(adr%), Flag%)
 IF Flag% <> 0 THEN LOCATE 1, 5: PRINT "ERROR 1.0": STOP
 Md% = 1: D%(0) = 0: D%(1) = 0
 CALL Das8(Md%, VARPTR(D%(0)), Flag%)
 IF Flag% <> 0 THEN LOCATE 1, 5: PRINT " ERROR 1.1": STOP
 CJSUM = 0
 RRR = 10
 FOR J = 1 TO RRR
```

```
    Md% = 4: Num% = 0
    CALL Das8(Md%, VARPTR(Num%), Flag%)
    IF Flag% <> 0 THEN LOCATE 1, 5: PRINT "ERROR 1.2": STOP
    CJSUM = CJSUM + Num%
 NEXT J
 cjc = (CJSUM * 5!) / (2047! * .0244 * RRR)
END SUB

DEFSNG A-Z
SUB Display
 LOCATE 1, 20: PRINT "Lilac2 "
 LOCATE 3, 18
 PRINT "Temperature (F)": LOCATE 3, 40: PRINT "Time:            min."
 LOCATE 5, 1
 PRINT "            Probe No. 1       Probe No. 2       Probe No. 3 "
 LOCATE 6, 1
 PRINT " ------------------------------------------------------------"
 PRINT "Central Bed:"
 PRINT "Annular Bed:"

 LOCATE 10, 1: PRINT "Temperature Set Point (F):"
 LOCATE 11, 1: PRINT "A1:  "
 LOCATE 12, 1: PRINT "A2:"
 LOCATE 13, 1: PRINT "B1:"
 LOCATE 14, 1: PRINT "B2:"
 LOCATE 15, 1: PRINT "D:  "

 LOCATE 17, 1
 PRINT "Air Desired   (scfm):"
 PRINT "Secondary Air (scfm):"
 LOCATE 21, 57: PRINT "F[1] ... Set Point"
 LOCATE 22, 57: PRINT "F[2] ... Begin Control"
 LOCATE 23, 57: PRINT "F[10] .. Exit"
 LOCATE 21, 1
 PRINT "-----------------------------------------------------"
 LOCATE 22, 1: PRINT "|MSGS:": LOCATE 22, 56: PRINT "|"
 LOCATE 23, 1
 PRINT "-----------------------------------------------------"
END SUB
```

```
DEFDBL A-Z
SUB Interp (DD, Rqd.flow)
 IF Rqd.flow > 32 THEN
     Rqd.flow = 32
     DD = 4095
     GOTO xend
     Rqd.flow = 0
     DD = 0
     GOTO xend
 END IF
 compare = 0
 i = 0
 DO
     i = i + 1
     compare = Calflow(i) - Rqd.flow
 LOOP UNTIL compare > 0

 pct = (Rqd.flow - Calflow(i - 1)) / (Calflow(i) - Calflow(i - 1))
 DD = INT(pct * 200 + Caldd(i - 1))
 xend:
END SUB

DEFDBL A-Z
SUB Interpolation (TF, Voltage(), Board%, Chan%)
' Entry variables:
'       CJC  = cold junction compensator temperature in deg. C.
'       VOLT = thermocouple voltage in volts
' Exit variables:
'      TC = temperature in degrees Centigrade
'      TF = temperature in degrees Fahrenheit

 vk = 1000 * Voltage(Board%, Chan%) + 1! + (cjc - 25) * .0405

 EK = INT((vk - svk) / sik)

' Out of bounds, round to lower limit
 IF EK < 0 THEN TC = Tk(0): GOTO 2360
```

```
' Out of bounds, round to upper limit
 IF EK > nk - 2 THEN TC = Tk(nk - 1): GOTO 2360

 TC = Tk(EK) + (Tk(EK + 1) - Tk(EK)) * (vk - EK * sik - svk) / sik

2360 : TF = TC * 9 / 5 + 32
END SUB

DEFDBL A-Z
SUB Jury (N, Theta1(), Theta())
 Flag = 0
 x = 1
 J = x ^ 2 + Theta(1) * x + Theta(2)
 IF J <= 0 THEN Flag = 1
 x = -1
 J = x ^ 2 + Theta(1) * x + Theta(2)
 IF J <= 0 THEN Flag = 1
 IF ABS(Theta(2)) >= 1 THEN Flag = 1
 IF Flag = 1 THEN
     Theta(1) = Theta1(1)
     Theta(2) = Theta1(2)
     Theta(3) = Theta1(3)
     Theta(4) = Theta1(4)
     Theta(5) = Theta1(5)
 END IF
END SUB

DEFDBL A-Z
SUB LQG (Nn, Theta(), F()) STATIC

' ----------------------- Augmented SVR ---------------------------

 A(1, 1) = -Theta(1)
 A(1, 2) = 1
 A(1, 3) = 0
 A(2, 1) = -Theta(2)
 A(2, 2) = 0
 A(2, 3) = 0
 A(3, 1) = -1
```

```
A(3, 2) = 0
A(3, 3) = 1

B(1) = Theta(3)
B(2) = Theta(4)
B(3) = 0

'--------------------- Weighting Matrices -------------------------

IF Chk = 0 THEN
    Rxx(1, 1) = 100
    Rxx(1, 2) = 0
    Rxx(1, 3) = 0
    Rxx(1, 2) = 0
    Rxx(2, 2) = 1
    Rxx(2, 3) = 0
    Rxx(3, 1) = 0
    Rxx(3, 2) = 0
    Rxx(3, 3) = 1

    Ruu = 8000

    Chk = 1
' Initial S matrix corresponding to initial parameter estimates
    S(1, 1) = 2860: S(1, 2) = 2558: S(1, 3) = -141
    S(2, 1) = 2558: S(2, 2) = 2387: S(2, 3) = -119
    S(3, 1) = -141: S(3, 2) = -119: S(3, 3) = 19
END IF

iter = 0
DO
    iter = iter + 1

'-------------- L = (Ruu + B'*S(t+1)B)^-1*B'S(t+1)A ----------------

    CALL Multnnxn1(Nn, S(), B(), Sto1())
    CALL Mult1nxn1(Nn, B(), Sto1(), Xout)
    den = Ruu + Xout
    CALL Multnnxnn(Nn, S(), A(), Temp1())
```

```
      CALL Multinxnn(Nn, B(), Temp1(), Sto1())
      CALL Multnixc(Nn, Sto1(), 1 / den, L())


'-------------------------- L'RuuL -------------------------------

      CALL Multnix1n(Nn, L(), L(), Temp1())
      CALL Multnnxc(Nn, Temp1(), Ruu, Temp4())

'-------------------------- A-BL ---------------------------------

      CALL Multnix1n(Nn, B(), L(), Temp1())
      CALL Subtractnxn(Nn, A(), Temp1(), Temp3())

'--------------------- (A-BL)'S(t+1)(A-BL) -----------------------

      CALL Transnxn(Nn, Temp3(), Temp1())
      CALL Multnnxnn(Nn, Temp1(), S(), Temp2())
      CALL Multnnxnn(Nn, Temp2(), Temp3(), Temp1())

'------------------- S(t) = Rxx + Temp1 + Temp4 -------------------

      CALL Addnn(Nn, Temp1(), Temp4(), Temp3())
      CALL Addnn(Nn, Temp3(), Rxx(), S())

'  Check for convergence.

 xerror = 0
      FOR i = 1 TO Nn
          dn = ABS(L(i))
          IF dn = 0 THEN dn = .00001
          xerror = xerror + ABS(F(i) - L(i)) / dn
      NEXT i

      FOR i = 1 TO Nn
          F(i) = L(i)
      NEXT i
 LOOP UNTIL iter > 5 AND xerror < .05
END SUB
```

```
DEFDBL A-Z
SUB Multinxn1 (N, Mc(), Md(), Xout)
 Xout = 0
 FOR i = 1 TO N
     Xout = Xout + Mc(i) * Md(i)
 NEXT i
END SUB

DEFDBL A-Z
SUB Multinxnn (N, Mc(), Ma(), Zout())
 FOR i = 1 TO N
     Zout(i) = 0
     FOR J = 1 TO N
         Zout(i) = Zout(i) + Mc(J) * Ma(J, i)
     NEXT J
 NEXT i
END SUB

DEFDBL A-Z
SUB Multn1x1n (N, Mc(), Md(), Yout())
 FOR i = 1 TO N
    FOR J = 1 TO N
        Yout(i, J) = Mc(i) * Md(J)
    NEXT J
 NEXT i
END SUB

DEFDBL A-Z
SUB Multn1xc (N, Mc(), x, Zout())
 FOR i = 1 TO N
     Zout(i) = Mc(i) * x
 NEXT i
END SUB

DEFDBL A-Z
SUB Multnnxc (N, Ma(), x, Yout())
 FOR i = 1 TO N
     FOR J = 1 TO N
```

```
            Yout(i, J) = Ma(i, J) * x
        NEXT J
    NEXT i
END SUB


DEFDBL A-Z
SUB Multnnxn1 (N, Ma(), Mc(), Zout())
    FOR i = 1 TO N
        Zout(i) = 0
        FOR J = 1 TO N
            Zout(i) = Zout(i) + Ma(i, J) * Mc(J)
        NEXT J
    NEXT i
END SUB

DEFDBL A-Z
SUB Multnnxnn (N, Ma(), Mb(), Yout())
    FOR i = 1 TO N
        FOR J = 1 TO N
            Temp = 0
            FOR k = 1 TO N
                Temp = Temp + Ma(i, k) * Mb(k, J)
            NEXT k
            Yout(i, J) = Temp
        NEXT J
    NEXT i
END SUB

DEFDBL A-Z
SUB Observer (N, Theta(), U, Y, Xhat()) STATIC
' Start-up Procedure

    IF Chk = 0 THEN
        DIM obsA(N, N), obsB(N), obsC(N), obsD(N), xhat.old(N), G(N)
        FOR i = 1 TO N
            xhat.old(N) = 0
        NEXT i
        U.old = 0
        Y.old = 0
```

```
    Chk = 1
    GOTO uend:
 END IF

 obsA(1, 1) = -Theta(1): obsA(1, 2) = 1
 obsA(2, 1) = -Theta(2): obsA(2, 2) = 0
 obsB(1) = Theta(3)
 obsB(2) = Theta(4)
 obsC(1) = 1
 obsC(2) = 0
 obsD(1) = Theta(5)
 obsD(2) = 0

' Find observer gains.

 G(1) = -.6 - Theta(1)
 G(2) = .09 - Theta(2)

 CALL Multnnxn1(N, obsA(), xhat.old(), Sto1())
 CALL Multn1xc(N, obsB(), U.old, Sto2())
 obserr = Y.old - xhat.old(1)
 CALL Multn1xc(N, G(), obserr, Sto3())

 FOR i = 1 TO N
    Xhat(i) = Sto1(i) + Sto2(i) + Sto3(i) + obsD(i)
 NEXT i

 FOR i = 1 TO N
    xhat.old(i) = Xhat(i)
 NEXT i

 y.old = Y
 U.old = U
uend:
END SUB

DEFDBL A-Z
SUB Pr (Ornum, flow)
 adr% = 768
```

```
 Md% = 0: Flag% = 0
 CALL Das8(Md%, VARPTR(adr%), Flag%)
 IF Flag% <> 0 THEN LOCATE 1, 5: PRINT "ERROR 2.0"

'Read pressure transducer.

 m% = 1: D%(1) = 4: D%(0) = 4: Flag% = 0
 CALL Das8(m%, VARPTR(D%(0)), Flag%)
 IF Flag <> 0 THEN LOCATE 22, 8: PRINT "ERROR 2.1"
     Flag% = 0: pres% = 0
     press.sum = 0
     FOR N = 1 TO 50
         Md% = 4
         CALL Das8(Md%, VARPTR(pres%), Flag%)
         IF Flag% <> 0 THEN LOCATE 22, 8: PRINT "ERROR 2.2"
         press.sum = press.sum + pres% / 2047!
     NEXT N
 inches.air = press.sum / 50 * 10
 inches.air = inches.air - zoff
 IF inches.air < 0 THEN inches.air = 0
 flow=pressure*(Orif(Ornum,1))*((inches.air/pressure)^Orif(Ornum,2))
END SUB

DEFDBL A-Z
SUB Readcal
 OPEN "c:\2.cal" FOR INPUT AS #5
 FOR i = 1 TO 20
     INPUT #5, Caldd(i), Calflow(i)
 NEXT i
 CLOSE #5
END SUB

DEFDBL A-Z
SUB Subtract (N, Mc(), Md(), Zout())
 FOR i = 1 TO N
     Zout(i) = Mc(i) - Md(i)
 NEXT i
END SUB
```

```
DEFDBL A-Z
SUB Subtractnxn (N, Ma(), Mb(), Yout())
 FOR i = 1 TO N
     FOR J = 1 TO N
         Yout(i, J) = Ma(i, J) - Mb(i, J)
     NEXT J
 NEXT i
END SUB

DEFDBL A-Z
SUB Temperature (Temp())
 DIM Tsum(1, 4)
 adr% = 768
 Md% = 0: Flag% = 0
 CALL Das8(Md%, VARPTR(adr%), Flag%)
 IF Flag% <> 0 THEN LOCATE 1, 5: PRINT "ERROR 3.0"

'Read thermocouple voltages.

 NTIMES = 3
 FOR P = 1 TO 1
     FOR C = 1 TO 4
         Tsum(P, C) = 0
     NEXT C
 NEXT P

 FOR JJ = 1 TO NTIMES
     FOR Board% = 1 TO 1
         Md% = 1: D%(0) = Board%: D%(1) = Board%
         CALL Das8(Md%, VARPTR(D%(0)), Flag%)
         IF Flag% <> 0 THEN LOCATE 22, 8: PRINT "ERROR 3.1"
         FOR Chan% = 1 TO 4
             Md% = 14
             CALL Das8(Md%, VARPTR(Chan%), Flag%)
             IF Flag% <> 0 THEN LOCATE 22, 8: PRINT "ERROR 3.2"
             Md% = 4
             CALL Das8(Md%, VARPTR(Digital%(Board%, Chan%)), Flag%)
             IF Flag% <> 0 THEN LOCATE 22, 8: PRINT "ERROR 3.3"
             Voltage(Board%, Chan%) = Digital%(Board%, Chan%) * 5!
```

```
                                    / (Gain(Board%) * 2047!)
            CALL Interpolation(TF, Voltage(), Board%, Chan%)
            T(Board%, Chan%) = TF
            Tsum(Board%, Chan%) = Tsum(Board%,Chan%)+T(Board%,Chan%)
        NEXT Chan%
    NEXT Board%
NEXT JJ

FOR B = 1 TO 1
    FOR C = 1 TO 4
        Temp(B, C) = Tsum(B, C) / NTIMES
    NEXT C
NEXT B

' Read Gas Analyzers (for future use)

'adr% = 832
'md% = 0: flag% = 0
'CALL das8(md%, VARPTR(adr%), flag%)
'IF flag% <> 0 THEN LOCATE 1, 5: PRINT "ERROR 3.4"

'XXX = 20

'FOR chan% = 1 TO 5
'    md% = 1: D%(0) = chan%: D%(1) = chan%
'    CALL das8(md%, VARPTR(D%(0)), flag%)
'    md% = 4: SUM2 = 0
'    FOR MM = 1 TO XXX
'        CALL das8(md%, VARPTR(conc%), flag%)
'        IF flag% <> 0 THEN LOCATE 22, 8: PRINT "ERROR 3.5"
'        SUM2 = SUM2 + conc%
'    NEXT MM
'    gasp(chan%) = range(chan%) * SUM2 * 5 / (XXX*volt(chan%)* 2047)
'    GASUM(chan%) = GASUM(chan%) + gasp(chan%)
' NEXT chan%
END SUB

DEFDBL A-Z
SUB Transnxn (N, Ma(), Yout())
 FOR i = 1 TO N
```

```
    FOR J = 1 TO N
        Yout(i, J) = Ma(J, i)
    NEXT J
 NEXT i
END SUB

DEFDBL A-Z
SUB Valve (ValveNo, DD)
 lochan% = 2 * ValveNo
 hichan% = 1 + 2 * ValveNo

 Xh% = INT(DD / 256)
 XL% = DD - Xh% * 256

 OUT 848 + lochan%, XL%
 OUT 848 + hichan%, Xh%
END SUB

DEFDBL A-Z
SUB WLSq (N, Y, U, Theta(), yhat) STATIC
' Check initial values.

 IF Chk = 0 THEN
    Lambda = .99
    Phi1(1) = -Y
    Phi1(2) = -Y
    Phi1(3) = U
    Phi1(4) = U
    Phi1(5) = 1
    Theta1(1) = -1.19
    Theta1(2) = .228
    Theta1(3) = -.5
    Theta1(4) = -.3
    Theta1(5) = 1.4

    Theta(1) = Theta1(1)
    Theta(2) = Theta1(2)
    Theta(3) = Theta1(3)
    Theta(4) = Theta1(4)
```

```
    Theta(5) = Theta1(5)

    Chk = 1
    FOR i = 1 TO N
        FOR J = 1 TO N
                P2(i, J) = 0
        NEXT J
        P2(i, i) = 15
    NEXT i
    GOTO zend
END IF

'-------- Phi1 = [-Y(t-1), -Y(t-2), u(t-1), u(t-2), 1]' ------------

Phi1(1) = -Y1
Phi1(2) = -Y2
Phi1(3) = U1
Phi1(4) = U2
Phi1(5) = 1

'------------------ Yhat(t) = phi1'*Theta1 ----------------------

CALL Mult1nxn1(N, Phi1(), Theta1(), yhat)

'-------------------- e(t) = Y(t) - yhat --------------------------

e = Y - yhat

'------------------ num = P2*phi1*phi1'*P2 -------------------------

CALL Multn1x1n(N, Phi1(), Phi1(), Temp1())
CALL Multnnxnn(N, P2(), Temp1(), Temp2())
CALL Multnnxnn(N, Temp2(), P2(), Num())

'------------------ den = 1+ Phi1'*P2*Phi1 ------------------------

CALL Multnnxn1(N, P2(), Phi1(), Sto1())
CALL Mult1nxn1(N, Phi1(), Sto1(), Xout)
den = Lambda + Xout
```

```
'------------------ P1 = 1/lamdba(P2-num/den) ----------------------

 CALL Multnnxc(N, Num(), 1 / den, Temp1())
 CALL Subtractnxn(N, P2(), Temp1(), Temp2())
 CALL Multnnxc(N, Temp2(), 1 / Lamda, P1())

'-------------- Theta = Theta1 + P2*phi1/den*e(t) -----------------

 CALL Multnnxn1(N, P2(), Phi1(), Sto1())
 CALL Multn1xc(N, Sto1(), e / den, Sto2())
 CALL Add(N, Theta1(), Sto2(), Theta())

' Update parameters.

 CALL Jury(N, Theta1(), Theta())

 FOR i = 1 TO N
     Theta1(i) = Theta(i)
     FOR J = 1 TO N
         P2(i, J) = P1(i, J)
     NEXT J
 NEXT i

zend:
 Y2 = Y1
 U2 = U1
 Y1 = Y
 U1 = U
END SUB

DEFDBL A-Z
SUB Xcntrl (F(), Xhat(), Y, Tset, U) STATIC
 IF Chk = 0 THEN
     z.old = 0
     Yhat.old = 0
     Y.old = 0
     Chk = 1
     GOTO yend:
```

```
 END IF
 Z = z.old + Tset - y.old
 U = -F(1) * Xhat(1) - F(2) * Xhat(2) - F(3) * Z

' Check for saturation; some integral wind-up allowed to prevent
' spikes.  Saturation based on 4 scfm offset.

 IF U > 23 THEN
     U = 23
     Z = z.old
 ELSEIF U < -1 THEN
     U = -1
     Z = z.old
 END IF

' Update parameters.

 Yhat.old = yhat
 Y.old = Y
 z.old = Z

yend:
END SUB
```

```
'-----------------------------------------------------------------
'******************** Main Module: PI20.BAS ************************
'-----------------------------------------------------------------
' Rev 1.0:  15 August 1989
' Rev 1.2:  18 November 1989

' To reference the performance of the adaptive control algorithm, a
' PI controller, tuned by the Ziegler-Nichols method, was used to
' command annular bed air flow rate.  The time interval between
' control updates was set to 20 seconds.  Codes were written and
' compiled with Microsoft QuickBASIC 4.0.  Data are written to
' output files using Lotus 1-2-3 File Import format.

'---------------------- Nomenclature (*) ---------------------------

'       Act.flow   = Actual secondary air flow rate, scfm
'       Begin.prog = Time program is initiated
'       Board      = Sub-multiplexer board number
'       Channel    = Sub-multiplexer board channel
'       Caldd      = Digital command signal data from calibration
'                    curve with position feedback control
'       Calflow    = Secondary air flow rate data from calibration
'                    curve with position feedback control, scfm
'       Dev.T      = Initial system temperature, F
'       Dev.u      = Initial system air flow rate, scfm
'       DD         = Digital command signal to Metrabyte DDA-06
'       Flag       = Error flag from Metrabyte DAS-8 subprograms
'       Flow       = Interpolated air flow rate
'       Iter       = Iteration counter
'       md%        = DAS-8 mode of operation
'       nk         = Number of data in Type-K thermocouple lookup
'                    table
'       Orif       = Orifice flow meter calibration data
'       Ornum      = Orifice flow meter number
'                        1 -- primary air
'                        2 -- N/A
'                        3 -- secondary air
'       Range      = Range settings for gas analyzers
```

```
'       Rqd.flow    = Required secondary air flow rate, scfm
'       SetPoint    = Setpoint temperature, F
'       sik         = Voltage step interval in Type-K thermocouple
'                     lookup table, mV
'       svk         = Starting voltage in Type-K thermocouple lookup
'                     table, mV
'       Temp        = Thermocouple temperature vector
'       TF          = Temperature, F
'       TC          = Temperature, C
'       Tset        = Deviation set point temperature, F
'       U           = Deviation air flow rate, scfm
'       ValveNo     = Valve number
'                         1 -- water flow into water jacket
'                         2 -- primary air flow rate
'                         3 -- secondary air flow rate
'       Xiter       = Iteration counter

' (*) Variables are double precision unless otherwise indicated.

'------------------- Subprogram Definitions -----------------------

'       Cntrl           Controls secondary air flow rate
'       Coldjunct       Finds cold junction compensation voltage
'       DAS8            Metrabyte DAS8 subprograms (library file)
'       Display         Screen headings overlay subprogram
'       Interp          Calculates air flow rate from orifice meter
'                       pressure drop readings
'       Interpolation   Calculates temperature from thermocouple
'                       voltage readings
'       Pr              Reads the pressure differential across the
'                       orifice flow meter
'       Readcal         Reads air flow data from the calibration
'                       curve with position feedback control
'       Temperature     Reads voltages from thermocouples
'       Valve           Sends digital command signal to Bellofram
'                       Type 1000 E/P transducer

'-------------------- Begin Program ------------------------------
```

```
DEFDBL A-Z
DECLARE SUB cntrl (rqd.flow#, act.flow#)
DECLARE SUB interpolation (tf#, voltage#(), board%, chan%)
DECLARE SUB Display ()
DECLARE SUB coldjunct ()
DECLARE SUB valve (valveNo#, DD#)
DECLARE SUB temperature (temp#())
DECLARE SUB pr (ornum#, flow#)
DECLARE SUB readcal ()
DECLARE SUB interp (DD#, rqd.flow#)
DECLARE SUB xcntrl (f#(), xhat#(), y#, Tset#, u#)
DECLARE SUB das8 (md%, BYVAL dummy%, flag%)

COMMON SHARED D%(), Digital%()
COMMON SHARED calflow(), caldd(), orif(), pressure, zoff
COMMON SHARED Tk(), volt(), range(), gain()
COMMON SHARED nk, sik, svk, cjc, begin.prog, start.control

'---------------- Data Acquisition Variables --------------------

DIM calflow(50), caldd(50), orif(6, 2)
DIM D%(6), range(5), volt(5), gain(3), Digital%(3, 16)
DIM Tk(308)
DIM SHARED temp(1, 4)

CLS
PRINT "Initializing System ..."
begin.prog = TIMER

KEY(1) ON: KEY(2) ON: KEY(10) ON
ON KEY(1) GOSUB key1
ON KEY(2) GOSUB key2
ON KEY(10) GOSUB key10

' Orifice calibration data

DATA 2.3426,0.51324,1.8196,0.50523,1.2647,0.51488,1.3359,0.5005
DATA 0.8500,0.53392,0.34717,0.49591
```

```
FOR III = 1 TO 6
    FOR JJJ = 1 TO 2
        READ orif(III, JJJ)
    NEXT JJJ
NEXT III

' ---------------------- Thermocouple Data ----------------------------

DATA  309 ,  .2 , -6.6
READ nk, sik, svk

'Temperature at -6.6mv, -6.4mV, -6.2mV etc.

DATA -353.5,-249.3,-224.0,-207.6,-194.3,-182.8,-172.3,-162.8,-153.8,-145.4
DATA -137.3,-129.6,-122.3,-115.2,-108.3,-101.6, -95.1, -88.7, -82.5, -76.4
DATA  -70.4, -64.6, -58.8, -53.1, -47.5, -42.0, -36.6, -31.2, -25.9, -20.6
DATA  -15.4, -10.2,  -5.1,  -0.0,   5.0,  10.1,  15.1,  20.0,  25.0,  29.9
DATA   34.8,  39.7,  44.6,  49.5,  54.3,  59.1,  64.0,  68.8,  73.6,  78.4
DATA   83.2,  88.0,  92.9,  97.7, 102.5, 107.4, 112.2, 117.1, 122.0, 126.9
DATA  131.8, 136.7, 141.7, 146.6, 151.6, 156.5, 161.5, 166.5, 171.5, 176.5
DATA  181.6, 186.6, 191.6, 196.6, 201.6, 206.6, 211.6, 216.6, 221.5, 226.5
DATA  231.5, 236.4, 241.4, 246.3, 251.2, 256.1, 261.0, 265.9, 270.8, 275.6
DATA  280.5, 285.3, 290.2, 295.0, 299.8, 304.6, 309.4, 314.3, 319.1, 323.9
DATA  328.7, 333.4, 338.2, 343.0, 347.8, 352.6, 357.3, 362.1, 366.9, 371.6
DATA  376.4, 381.1, 385.9, 390.6, 395.4, 400.1, 404.8, 409.6, 414.3, 419.0
DATA  423.8, 428.5, 433.2, 437.9, 442.6, 447.3, 452.0, 456.8, 461.5, 466.2
DATA  470.9, 475.6, 480.3, 485.0, 489.7, 494.4, 499.1, 503.8, 508.5, 513.1
DATA  517.8, 522.5, 527.2, 531.9, 536.6, 541.3, 546.0, 550.7, 555.4, 560.0
DATA  564.7, 569.4, 574.1, 578.8, 583.5, 588.2, 592.9, 597.6, 602.3, 607.0
DATA  611.7, 616.4, 621.2, 625.9, 630.6, 635.3, 640.0, 644.8, 649.5, 654.2
DATA  658.9, 663.7, 668.4, 673.2, 677.9, 682.7, 687.4, 692.2, 696.9, 701.7
DATA  706.5, 711.3, 716.1, 720.8, 725.6, 730.4, 735.2, 740.0, 744.8, 749.7
DATA  754.5, 759.3, 764.1, 769.0, 773.8, 778.7, 783.5, 788.4, 793.3, 798.1
DATA  803.0, 807.9, 812.8, 817.7, 822.6, 827.5, 832.4, 837.3, 842.2, 847.2
DATA  852.1, 857.1, 862.0, 867.0, 872.0, 876.9, 881.9, 886.9, 891.9, 896.9
DATA  901.9, 906.9, 911.9, 916.9, 922.0, 927.0, 932.0, 937.1, 942.2, 947.2
DATA  952.3, 957.4, 962.5, 967.6, 972.7, 977.8, 982.9, 988.0, 993.1, 998.2
DATA 1003.4,1008.5,1013.7,1018.8,1024.0,1029.2,1034.4,1039.6,1044.8,1050.0
DATA 1055.2,1060.4,1065.6,1070.8,1076.1,1081.3,1086.6,1091.9,1097.2,1102.4
```

```
DATA 1107.7,1113.0,1118.3,1123.7,1129.0,1134.3,1139.7,1145.0,1150.4,1155.8
DATA 1161.2,1166.6,1172.0,1177.4,1182.9,1188.3,1193.8,1199.2,1204.7,1210.2
DATA 1215.7,1221.2,1226.8,1232.3,1237.9,1243.5,1249.1,1254.7,1260.3,1265.9
DATA 1271.6,1277.3,1282.9,1288.6,1294.3,1300.1,1305.8,1311.5,1317.3,1323.1
DATA 1328.9,1334.7,1340.5,1346.4,1352.2,1358.1,1363.9,1369.8,1375.7

FOR i = 0 TO nk - 1
    READ Tk(i)
NEXT i

' Set Ranges for Gas Analyzers

range(1) = 25: volt(1) = 5                    ' Oxygen
range(2) = 1.2: volt(2) = 5                   ' Carbon Monoxide
range(3) = 30: volt(3) = 5                    ' Carbon Dioxide
range(4) = 2000: volt(4) = 1                  ' Sulfur Dioxide
range(5) = 1000: volt(5) = 1                  ' NOx

' Set Gains for the EXP-16'S

gain(1) = 50
gain(2) = 1000
gain(3) = 50

' ------------------- System Constants --------------------------
8 :
INPUT "Enter Orcal Offset:  ", zoff
PRINT "Edit? (y/[n]):  "
10 : ans$ = INKEY$: IF ans$ = "" GOTO 10
IF ans$ = "y" OR ans$ = "Y" THEN GOTO 8
pressure = 60 + 14.7
valveNo = 3
ornum = 3
start.control = 0

' Open Secondary Air Transducer

PRINT "Closing All Valves ..."
```

```
DD = 0: v = 1: CALL valve(v, DD)
DD = 0: v = 2: CALL valve(v, DD)
DD = 0: v = 3: CALL valve(v, DD)

PRINT "Opening Pressure Transducer ..."
OUT 773, 2
t1 = TIMER
DO: t2 = TIMER: LOOP UNTIL t2 - t1 > 3.5

' Find Cold Junction Temperature

PRINT "Reading Cold Junction Temperature ..."
CALL coldjunct

' Open Data File

OPEN "c:\PI20.PRN" FOR OUTPUT AS #1
WRITE #1, "--TIMER--/--SetPoint--/--temp1--/--temp2--/--temp3--
          /--temp4--/--rqd.flow--/--act.flow--/--"

PRINT "Reading Calibration ..."
CALL readcal

'End initialization

PRINT " ----> Done <----"
CLS

' Loop Here until Controller is Invoked

CALL Display

DO
    rqd.flow = 4!
    CALL cntrl(rqd.flow, act.flow)
    CALL temperature(temp())

    WRITE #1, CSNG(TIMER), setpoint, CSNG(temp(1, 1)),
              CSNG(temp(1, 2)), CSNG(temp(1, 3)), CSNG(temp(1, 4)),
```

```
            CSNG(rqd.flow), CSNG(act.flow)

LOOP UNTIL start.control <> 0

'Call Control Program

dev.u = 4!
G = -.078
Ki = 1 / 114
integral = 0
e.old = 0
DO
    TT1 = TIMER
    IF rqd.flow > 20 THEN rqd.flow = 20
    IF rqd.flow < 4 THEN rqd.flow = 4
    CALL cntrl(rqd.flow, act.flow)
tempsum = 0
FOR xiter = 1 TO 10
    iter = 0
    DO
        iter = iter + 1
        CALL temperature(temp())
        comp1 = temp(1, 1)
        comp2 = temp(1, 2)
    LOOP UNTIL ABS(comp1 - comp2) < 5 OR iter > 20
    tempsum = tempsum + temp(1, 1)
NEXT xiter
    y = tempsum / 10
    u = rqd.flow - dev.u
    e = setpoint - y
    xtemp = integral
    integral = integral + .5 * (e + e.old) * 20
    e.old = e
    u = G * (e + Ki * integral)
    rqd.flow = u + dev.u
    IF rqd.flow > 23 THEN
        rqd.flow = 23
        integral = xtemp
    ELSEIF rqd.flow < 4 THEN
```

```
        rqd.flow = 4
        integral = xtemp
    END IF
    WRITE #1, CSNG(TIMER), setpoint, CSNG(temp(1, 1)),
              CSNG(temp(1, 2)), CSNG(temp(1, 3)), CSNG(temp(1, 4)),
              CSNG(rqd.flow), CSNG(act.flow)
LOOP
STOP

key1:
    CLS
    INPUT "Enter Set Point:  ", setpoint
    CLS
    CALL Display
    LOCATE 10, 30: PRINT USING "####.##"; setpoint
    RETURN
key2:
    CLS
    INPUT "Enter Set Point:  ", setpoint
    CLS
    CALL Display
    LOCATE 10, 30: PRINT USING "####.##"; setpoint
    start.control = 1
    RETURN
key10:
    CLOSE #1
    OUT 773, 0
    DD = 0: v = 1: CALL valve(v, DD)
    DD = 0: v = 2: CALL valve(v, DD)
    DD = 0: v = 3: CALL valve(v, DD)
    STOP
RETURN
STOP
'-------------------------------------------------------------------
'****************** End Main Module: PI20.BAS **********************
'-------------------------------------------------------------------


DEFDBL A-Z
SUB cntrl (rqd.flow, act.flow) STATIC
```

```
 ts = TIMER
 delay = 20
 ornum = 3
 valveNo = 3
 LOCATE 17, 5

'------------------- Rough Controller -----------------------------

IF rqd.flow > 27 THEN
    rqd.flow = 27
    DD = 4095
    CALL valve(valveNo, DD)
    DO
        CALL pr(ornum, flow)
        LOCATE 17, 23: PRINT USING "####.##"; rqd.flow
        LOCATE 18, 23: PRINT USING "####.##"; flow
        CALL temperature(temp())
        LOCATE 7, 10:
        PRINT USING " #####.##   ";temp(1, 1),temp(1, 2),temp(1, 3)
        LOCATE 8, 10:
        PRINT USING " #####.##"; temp(1, 4)
        t2 = TIMER
        LOCATE 3, 47: PRINT USING "####.###"; (t2 - begin.prog) / 60

    LOOP UNTIL TIMER - ts > delay
    GOTO 200:
ELSEIF rqd.flow < 0 THEN rqd.flow = 0
    rqd.flow = 0
    DD = 0
    CALL valve(valveNo, DD)
    DO
        CALL pr(ornum, flow)
        LOCATE 17, 23: PRINT USING "####.##"; rqd.flow
        LOCATE 18, 23: PRINT USING "####.##"; flow
        CALL temperature(temp())
        LOCATE 7, 10:
        PRINT USING " #####.##  ";temp(1, 1),temp(1, 2),temp(1, 3)
        LOCATE 8, 10:
        PRINT USING " #####.##"; temp(1, 4)
```

```
        t2 = TIMER
        LOCATE 3, 47: PRINT USING "####.###"; (t2 - begin.prog) / 60

    LOOP UNTIL TIMER - ts > delay
    GOTO 200
END IF

CALL pr(ornum, flow)
e = flow - rqd.flow
IF ABS(e) < .41 THEN
        GOTO 110:
END IF
IF ABS(e) < 3 THEN
        CALL interp(DD, rqd.flow)
        CALL valve(valveNo, DD)
        GOTO 100:
END IF
t1 = TIMER
DO
    t2 = TIMER
    CALL interp(DD, rqd.flow)
    CALL valve(valveNo, DD)
    CALL pr(ornum, flow)
    LOCATE 17, 23: PRINT USING "####.##"; rqd.flow
    LOCATE 18, 23: PRINT USING "####.##"; flow
    CALL temperature(temp())
    LOCATE 7, 10:
    PRINT USING " #####.##  "; temp(1, 1), temp(1, 2), temp(1, 3)
    LOCATE 8, 10:
    PRINT USING " #####.##"; temp(1, 4)
    LOCATE 3, 47: PRINT USING "####.###"; (t2 - begin.prog) / 60
    IF t2 - ts > delay THEN GOTO 200:
LOOP UNTIL t2 - t1 > .8 AND ABS(flow - rqd.flow) / rqd.flow < .1

100 :

'----------------- Perturbation Controller ----------------------

CALL pr(ornum, flow)
```

```
t1 = TIMER
integral = 0
kp = 80
Ki = 50
e = rqd.flow - flow
e.old = e
D.nom = DD

110 :

DO
    t2 = TIMER
    DT = t2 - t1
    CALL pr(ornum, flow)
    e = rqd.flow - flow
    integral = integral + .5 * (e + e.old) * DT
    d2 = INT(kp * e + Ki * integral)
    DD.new = D.nom + d2
    IF DD.new > 4095 THEN DD.new = 4095
    IF DD.new < 0 THEN DD.new = 0

    CALL valve(valveNo, DD.new)
    LOCATE 17, 23: PRINT USING "####.##"; rqd.flow
    LOCATE 18, 23: PRINT USING "####.##"; flow
    CALL temperature(temp())
    LOCATE 7, 10:
    PRINT USING " #####.##      ";temp(1, 1),temp(1, 2),temp(1, 3)
    LOCATE 8, 10:
    PRINT USING " #####.##"; temp(1, 4)
    LOCATE 3, 47: PRINT USING "####.###"; (t2 - begin.prog) / 60

    e.old = e
    t1 = t2
LOOP UNTIL TIMER - ts > delay

200 :
CALL pr(ornum, flow)
act.flow = flow
END SUB
```

```
DEFDBL A-Z
SUB coldjunct

'Find Cold Junction Temperature (24.4 mV/ C)

adr% = 768
md% = 0: flag% = 0
CALL das8(md%, VARPTR(adr%), flag%)
IF flag% <> 0 THEN LOCATE 1, 5: PRINT "ERROR 1.0": STOP

md% = 1: D%(0) = 0: D%(1) = 0
CALL das8(md%, VARPTR(D%(0)), flag%)
IF flag% <> 0 THEN LOCATE 1, 5: PRINT " ERROR 1.1": STOP
CJSUM = 0
RRR = 10
FOR J = 1 TO RRR
    md% = 4: num% = 0
    CALL das8(md%, VARPTR(num%), flag%)
    IF flag% <> 0 THEN LOCATE 1, 5: PRINT " ERROR 1.2": STOP
    CJSUM = CJSUM + num%
NEXT J
cjc = (CJSUM * 5!) / (2047! * .0244 * RRR)
END SUB

DEFSNG A-Z
SUB Display
 LOCATE 1, 20: PRINT "        Lilac3 "

 LOCATE 3, 18: PRINT "Temperature (F)"
 LOCATE 3, 40: PRINT "Time:                min."
 LOCATE 5, 1
 PRINT "            Probe No. 1       Probe No. 2       Probe No. 3 "
 LOCATE 6, 1
 PRINT " ------------------------------------------------------------"
 PRINT "Central Bed:"
 PRINT "Annular Bed:"

 LOCATE 10, 1: PRINT "Temperature Set Point (F):"
```

```
 LOCATE 17, 1
 PRINT "Air Desired   (scfm):"
 PRINT "Secondary Air (scfm):"
 LOCATE 21, 57: PRINT "F[1] ... Set Point"
 LOCATE 22, 57: PRINT "F[2] ... Begin Control"
 LOCATE 23, 57: PRINT "F[10] .. Exit"
 LOCATE 21, 1: PRINT "-------------------------------------------"
 LOCATE 22, 1: PRINT "|MSGS:": LOCATE 22, 56: PRINT "|"
 LOCATE 23, 1: PRINT "-------------------------------------------"
END SUB

DEFDBL A-Z
SUB interp (DD, rqd.flow)
 IF rqd.flow > 32 THEN
     rqd.flow = 32
     DD = 4095
     GOTO xend
 ELSEIF rqd.flow < 0 THEN
     rqd.flow = 0
     DD = 0
     GOTO xend
 END IF
 compare = 0
 i = 0
 DO
     i = i + 1
     compare = calflow(i) - rqd.flow

 LOOP UNTIL compare > 0

 pct = (rqd.flow - calflow(i - 1)) / (calflow(i) - calflow(i - 1))
 DD = INT(pct * 200 + caldd(i - 1))
xend:
END SUB

DEFDBL A-Z
SUB interpolation (tf, voltage(), board%, chan%)

' ---- Interpolation routine to find K thermocouple temperature ----
```

```
' Entry variables:-
'      CJC  = cold junction compensator temperature in deg. C.
'      VOLT = thermocouple voltage in volts
' Exit variables:-
'     TC = temperature in degrees Centigrade
'     TF = temperature in degrees Fahrenheit
' Execution time on std. IBM P.C. = 46 milliseconds
' Perform CJC compensation for K type

 vk = 1000 * voltage(board%, chan%) + 1! + (cjc - 25) * .0405

' Find look up element
 EK = INT((vk - svk) / sik)
 IF EK < 0 THEN Tc = Tk(0): GOTO 2360
 IF EK > nk - 2 THEN Tc = Tk(nk - 1): GOTO 2360
'Do interpolation
 Tc = Tk(EK) + (Tk(EK + 1) - Tk(EK)) * (vk - EK * sik - svk) / sik

2360 tf = Tc * 9 / 5 + 32
END SUB

DEFDBL A-Z
SUB pr (ornum, flow)
 adr% = 768
 md% = 0: flag% = 0
 CALL das8(md%, VARPTR(adr%), flag%)
 IF flag% <> 0 THEN LOCATE 1, 5: PRINT "ERROR 2.0"

'Read Pressure Transducer

 m% = 1: D%(1) = 4: D%(0) = 4: flag% = 0
 CALL das8(m%, VARPTR(D%(0)), flag%)
 IF flag <> 0 THEN LOCATE 22, 8: PRINT "ERROR 2.1"
     flag% = 0: pres% = 0
     press.sum = 0
     FOR N = 1 TO 50
         md% = 4
         CALL das8(md%, VARPTR(pres%), flag%)
         IF flag% <> 0 THEN LOCATE 22, 8: PRINT "ERROR 2.2"
```

```
        press.sum = press.sum + pres% / 2047!
    NEXT N
    inches.air = press.sum / 50 * 10
    inches.air = inches.air - zoff
    IF inches.air < 0 THEN inches.air = 0
flow=pressure*(orif(ornum,1))*((inches.air/pressure)^orif(ornum,2))
END SUB

DEFDBL A-Z
SUB readcal
 OPEN "c:\2.cal" FOR INPUT AS #5
FOR i = 1 TO 20
    INPUT #5, caldd(i), calflow(i)
NEXT i
CLOSE #5
END SUB

DEFDBL A-Z
SUB temperature (temp())
 DIM Tsum(1, 4)

 adr% = 768
 md% = 0: flag% = 0
 CALL das8(md%, VARPTR(adr%), flag%)
 IF flag% <> 0 THEN LOCATE 1, 5: PRINT "ERROR 3.0"

'Read Thermocouples

 NTIMES = 3
 FOR P = 1 TO 1
    FOR c = 1 TO 4
        Tsum(P, c) = 0
    NEXT c
 NEXT P

FOR JJ = 1 TO NTIMES
    FOR board% = 1 TO 1
        md% = 1: D%(0) = board%: D%(1) = board%
        CALL das8(md%, VARPTR(D%(0)), flag%)
```

```
        IF flag% <> 0 THEN LOCATE 22, 8: PRINT "ERROR 3.1"
        FOR chan% = 1 TO 4
            md% = 14
            CALL das8(md%, VARPTR(chan%), flag%)
            IF flag% <> 0 THEN LOCATE 22, 8: PRINT "ERROR 3.2"
            md% = 4
            CALL das8(md%, VARPTR(Digital%(board%, chan%)), flag%)
            IF flag% <> 0 THEN LOCATE 22, 8: PRINT "ERROR 3.3"
            voltage(board%, chan%) = Digital%(board%, chan%) * 5!
                                     / (gain(board%) * 2047!)
            CALL interpolation(tf, voltage(), board%, chan%)
            T(board%, chan%) = tf
            Tsum(board%,chan%) = Tsum(board%,chan%) + T(board%,chan%)
        NEXT chan%
    NEXT board%
NEXT JJ

FOR B = 1 TO 1
    FOR c = 1 TO 4
        temp(B, c) = Tsum(B, c) / NTIMES
    NEXT c
NEXT B


' Read Gas Analyzers (future use)

'adr% = 832
'md% = 0: flag% = 0
'CALL das8(md%, VARPTR(adr%), flag%)
'IF flag% <> 0 THEN LOCATE 1, 5: PRINT "ERROR 3.4"

'XXX = 20

'FOR chan% = 1 TO 5
'        md% = 1: D%(0) = chan%: D%(1) = chan%
'        CALL das8(md%, VARPTR(D%(0)), flag%)
'        md% = 4: SUM2 = 0
'        FOR MM = 1 TO XXX
'             CALL das8(md%, VARPTR(conc%), flag%)
```

```
'             IF flag% <> 0 THEN LOCATE 22, 8: PRINT "ERROR 3.5"
'             SUM2 = SUM2 + conc%
'         NEXT MM
'         gasp(chan%) = range(chan%)*SUM2*5/(XXX*volt(chan%)*2047!)
'         GASUM(chan%) = GASUM(chan%) + gasp(chan%)
'NEXT chan%
END SUB

DEFDBL A-Z
SUB valve (valveNo, DD)
 lochan% = 2 * valveNo
 hichan% = 1 + 2 * valveNo

 Xh% = INT(DD / 256)
 XL% = DD - Xh% * 256

 OUT 848 + lochan%, XL%
 OUT 848 + hichan%, Xh%
END SUB
```

11. BIBLIOGRAPHY

[1] Anderson, B.D.O., *et al.* (1986). *Stability of Adaptive Systems.* MIT Press, Cambridge, Mass.

[2] Åström, K.J., and B. Wittenmark. (1989). *Adaptive Control.* Addison Wesley, New York.

[3] Bellanger, M.G. (1987). *Adaptive Digital Filters and Signal Analysis.* Marcel Dekker, New York.

[4] Bennett, S. (1979). *A History of Control Engineering.* The Institution of Electrical Engineers, New York.

[5] Botterill, J.S.M., A.G. Salway, and Y. Teoman. (1984). "Heat conduction through slumped fluidized beds." In *Fluidization*, ed. K. Kunii and R. Toei. Engineering Foundation, New York.

[6] Brown, R.C., and J.E. Foley. (1988). "Load turndown in a two-bed fluidized combustor." *Industrial and Engineering Chemistry Research*, **27**: 24–30.

[7] Brown, R.C., J.E. Foley, and W. Buttermore. (1989). "Heat transfer in a two-bed fluidized combustor." *American Institute of Chemical Engineers Symposium Series*, **85**(269): 153–158.

[8] Cox, C.P. (1987). *A Handbook of Introductory Statistical Methods.* John Wiley, New York.

[9] Doebelen, E.O. (1985). *Control System Principles and Design.* John Wiley, New York.

[10] Foley, J.E. (1987). "Development of a small-scale fluidized bed combustor fired by coal-water mixtures." M.S. thesis. Iowa State University, Ames, Iowa.

[11] Foley, J.E. (1989). "Heat transfer and combustion in a two-bed fluidized combustor." Ph.D. dissertation. Iowa State University, Ames, Iowa.

[12] Gelperin, N.I., and V.G. Einstein. (1971). "Heat transfer in fluidized beds." In *Fluidization*, ed. J.F. Davidson and D. Harrison. Academic Press, New York.

[13] Goodwin, G.C., and K.S. Sin. (1984). *Adaptive Filtering, Prediction, and Control.* Prentice-Hall, Englewood Cliffs, N.J.

[14] Gopal, M. (1988). *Digital Control Engineering.* John Wiley, New York.

[15] Gregory, J.W. (1989). "Coal-water-mixture combustion mechanisms in fluidized beds." M.S. thesis. Iowa State University, Ames, Iowa.

[16] Hainley, D.C., M.Z. Haji-Sulaiman, S. Yavuzkurt, and A.W. Scaroni. (1987). "Operating experience with a fluidized bed test combustor." *Transactions of the ASME*, **109**: 58–65.

[17] Hunt, K.J. (1989). *Stochastic Optimal Control Theory with Application in Self-Tuning Control.* **117**. Springer-Verlag, New York.

[18] Jacquot, R.G. (1981). *Modern Digital Control Systems.* Marcel Dekker, New York.

[19] Jamaluddin, A.S., and M.N. Islam. (1983). "Heat transfer from a burning fluidized bed to coil surfaces." *Journal of Powder & Bulk Solids Technology*, **7**(2): 1–5.

[20] Kam, L.W., and A.P. Priddy. (1985). *Power Plant Design.* John Wiley, New York.

[21] Kunii, D., and O. Levenspiel. (1969). *Fluidization Engineering.* Wiley, New York.

[22] Kuo, B.C. (1980). *Digital Control Systems.* Holt, Rinehart, and Winston, New York.

[23] Kuo, K.K. (1986). *Principles of Combustion.* John Wiley, New York.

[24] Kwakernaak, H., and R. Sivan. (1972). *Linear Optimal Control Systems.* Wiley-Interscience, New York.

[25] Ljung, L., and T. Söderström. (1982). *Theory and Practice of Recursive Identification.* MIT Press, Cambridge, Mass.

[26] Mayr, O. (1970). "The Origins of Feedback Control." MIT Press, Cambridge, Mass.

[27] McFarlane, R.C., T.W. Hoffman, P.A. Taylor, and J.F. MacGregor. (1983). "Control of fluidized bed reactors. 1. Modeling, simulation, and single-loop control studies." *Ind. Eng. Chem. Process Des. Dev.*, **22**(1): 22-30.

[28] Nack, H., *et al.* (1983). "Fluidized-bed combustion review." In *Fluidization Technology*, ed. D.L. Keairns. Hemisphere Publishing, Washington.

[29] Pflum, D. (1989). "Heat transfer in high aspect ratio fluidized beds." M.S. thesis. Iowa State University, Ames, Iowa.

[30] Sarofim, A.F., and J.M. Beer. (1978). "Modelling of Fluidized Bed Combustion." *17th Symp. (Int.) Combustion.* The Combustion Institute, Pittsburgh, Pa, pp. 189–204.

[31] Steel, R.G.D., and J.H. Torrie. (1980). *Principles and Procedures of Statistics.* McGraw-Hill, New York.

[32] van der Post, A.J., O.H. Bosgra, and G. Boelens. (1981). "Modeling the Dynamics of Fluidisation and Combustion in a Coal-Fired FBC." *Journal of Powder & Bulk Solids Technology*, **5**(4): 32–37.

[33] Xavier, A.M., and J.F. Davidson. (1985). "Heat transfer in fluidized beds: convection heat transfer in fluidized beds." *Fluidization*, ed. J.F. Davidson, R. Clift, and D. Harrison. Academic Press, New York.

[34] Young, P.C., and J.C. Willems. (1972). "An approach to the linear multivariable servomechanism problem." *Int. J. Control*, **15**(5): 961–979.