PSEUDO-LINEAR IDENTIFICATION AND ITS APPLICATION TO ADAPTIVE CONTROL

by

Russell Stephen Kemp

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
in
Electrical Engineering

APPROVED:

H. F. VanLandingham, Chairman

L. G. Kraige

A. A. Beex

March, 1987
Blacksburg, Virginia

PSEUDO-LINEAR IDENTIFICATION AND ITS APPLICATION TO ADAPTIVE CONTROL

by

Russell Stephen Kemp

Committee Chairman: Hugh F. VanLandingham
Electrical Engineering

(ABSTRACT)

The method of Pseudo-Linear Identification (PLID) is presented as an explicitly linear approach to the joint state and parameter estimation problem. The convergence properties of the algorithm in the stochastic case are investigated through simulation and comparisons with popular methods are made. The method is then extended to allow the tracking of time-varying system parameters.

An adaptive control structure based upon this Tracking-PLID algorithm is proposed which utilizes pole-placement state variable feedback via Ackermann's formula. The capabilities of this adaptive control scheme are demonstrated by application to a full-order nonlinear aircraft simulation model where significant improvement is shown over fixed-gain controllers in the face of rapid plant changes.

ACKNOWLEDGMENTS

I wish to thank my family for their loving support of all my endeavors and especially my parents for emphasizing the value of education and honest effort throughout my life. This work I proudly dedicate to them. I consider myself very fortunate to have had such fine grandparents whom I thank with all my heart. I am certainly a better man for having had the example of my grandfather whose loss this year has reminded me even more of the importance of family. I also wish to thank for her love and support over the past four years.

I am especially grateful to Dr. Hugh VanLandingham for his patient guidance and for bringing his development of the PLID algorithm to my attention in the first place. I thank Dr. Louis Beex and Dr. Glenn Kraige for their helpful suggestions and service on my graduate committee. Sincere gratitude is expressed to Dr. Brian L. Stevens of the Lockheed-Georgia Company for his thoughtful direction of this effort, his tireless efforts in providing the contractual support and his friendship. The criticisms and suggestions of while not always welcomed with cheer, are greatly appreciated. Thanks also to for his preparation of the diagrams for this thesis.

This work was supported in full by the Advanced Research Organization of the Lockheed-Georgia Company (Sponsor #RC 79865) under proposal #86-1094-02 for which I express my sincere appreciation.

TABLE OF CONTENTS

TABLE OF CONTENTS

1.0 INTRODUCTION

The mathematical modeling of systems and processes forms the foundation of modern engineering analysis and design. Once a structure, vehicle, or process is adequately described by a set of mathematical relationships, it can be studied directly in terms of the mathematics to estimate its physical properties. The equations relating applied loads and stresses in a bridge, thrust and speed in an aircraft, and pressure and flow rate in a pipeline are all examples of mathematical models. Typically, the physical laws governing a system or process can be exploited in deriving its mathematical model. A direct application of Newton's laws of motion, for example, will yield the differential equation relating the thrust of a rocket engine to the rocket's velocity. Once the equations of the model are set forth, one may view the system as simply a "black box" between system inputs and outputs. In our examples, thrust, load, and pressure are inputs and stress, velocity, and flow rate may be viewed as the resulting outputs. The engineer may utilize the black box viewpoint to predict the output of the physical system in response to expected combinations of inputs. Using the insight gained by such predictions, the engineer may choose to modify the system or the inputs to achieve the desired output goals. This process is the essence of engineering design. The accuracy of the mathematical model in representing the true physical system is therefore crucial to the reliability of the entire design process.

1.1 System Identification

In many complex systems, involving the interaction of many interrelated components, modeling by direct application of physical laws would prove very difficult, if not impossible. A thoughtful reflection on the "black box" concept, however, suggests an alternative method for developing mathematical models of systems. If one were to experimentally apply a sufficient variety of inputs to the system and record the resulting outputs he would perhaps be able to identify a mathematical relationship linking the two. The field referred to as System Identification is built upon this fundamental principle of exciting a system with known inputs and using the corresponding output information to determine the system model.

Karl Friedrich Gauss is generally credited with the first systematic approach to an identification problem. In his "Theory of Motion of Heavenly Bodies" [1] from the early nineteenth century, Gauss applied the technique of least squares criterion minimization to estimate orbital parameters in a mathematical model of planetary motions. Least Squares Identification, and its recursive extension, remain among the most popular identification techniques in use today. Gauss' work is illustrative of the approach still taken in identification problems. By formulating his celestial model in terms of a number of variable parameters, Gauss recast his problem into one of "parameter estimation." In fact, the terms "system identification" and "parameter estimation" are often used interchangeably since one

typically estimates the parameters of a model structure chosen to represent the system to be identified.

Gauss' work is an example of batch identification. The parameter estimation was performed using an extensive set of planetary data previously collected over many years. In batch identification, one collects input and output data over a long time period and processes the data all at once to arrive at a single estimate of the system parameters. Recursive identification techniques, on the other hand, produce an updated parameter estimate as each input-output data pair becomes available during the experiment. Many recursive algorithms are derived as extensions of various classical batch identification methods. The advantages of recursive methods are that the entire data record need not be stored, since the data is processed immediately, and that parameter estimates, albeit preliminary approximations, are made available for use very early in the experiment.

One may not always be completely free to choose arbitrary inputs for identification purposes. Clearly, safety considerations limit the range of test inputs allowable on bridges, airplanes, and nuclear power plant systems for example. For these types of systems it is desirable to identify the model using input-output data obtained while the system is in its normal operating condition. Identifying systems from such data is referred to as "on-line" identification and usually relies on recursive algorithms. Thus, a recursive, on-line identification algorithm would periodically sample the inputs and outputs of an operating system to give running estimates of the system parameters as

illustrated in Figure 1.1. Such algorithms could, for instance, be used to monitor a critical chemical process to ensure that the system parameters remain within safe limits during operation.

1.2 Adaptive Control

The field of dynamic system control is concerned with manipulating the inputs to a system in such a way that the outputs satisfy a specified objective [2]. The rule which associates the current system output, the control objectives, and the manipulated input is referred to as a "control law." In designing the control law for a given process or "plant," the control engineer relies on a mathematical model of the plant for insight. Most control law design techniques require knowledge of either a transfer function in a transform domain or a state variable model relating the input to the plant output. The overall concept of a controlled system is illustrated in Figure 1.2. Clearly, an accurate model of the plant is critical to the proper design of an effective control law.

We have seen that an on-line recursive identification algorithm can give a running estimate of the system parameters. In the case of a dynamic system, these parameters would be the coefficients of a transfer function or entries in the state model matrices. In adaptive control, one computerizes the control law synthesis procedure such that as parameter estimate updates become available, the control law is "redesigned" to achieve the control objective for the new plant model. In Figure 1.3 an adaptive control structure is diagrammed to illustrate

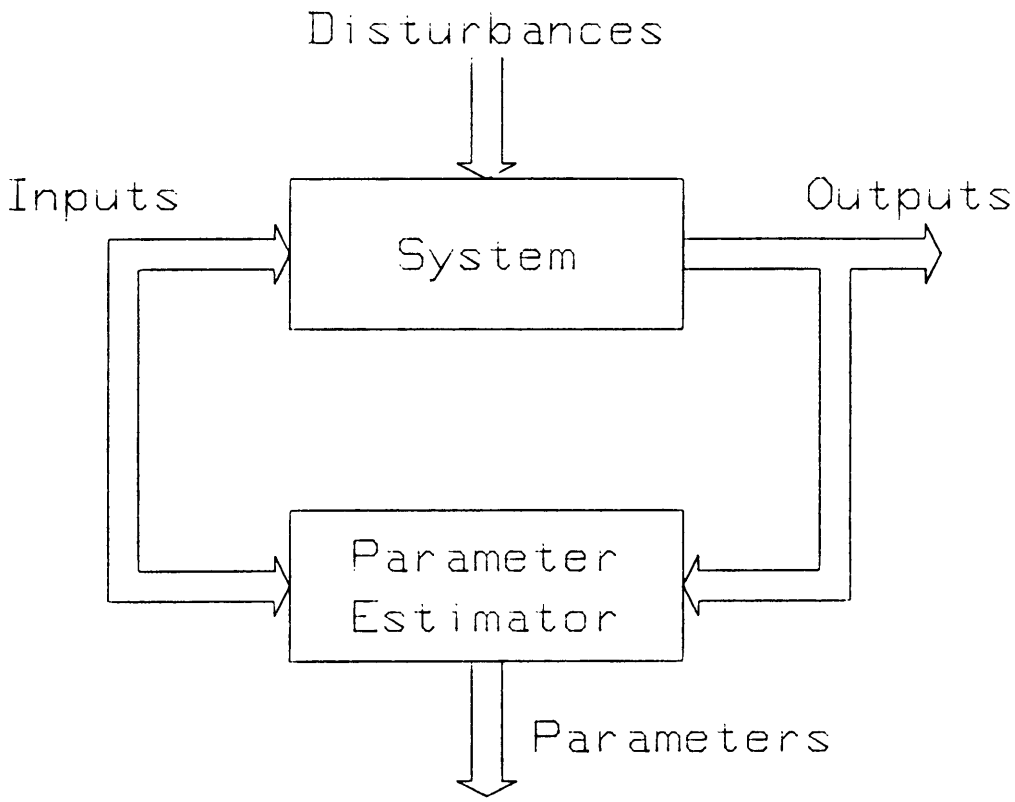

Figure 1.1 On-Line Recursive Identification

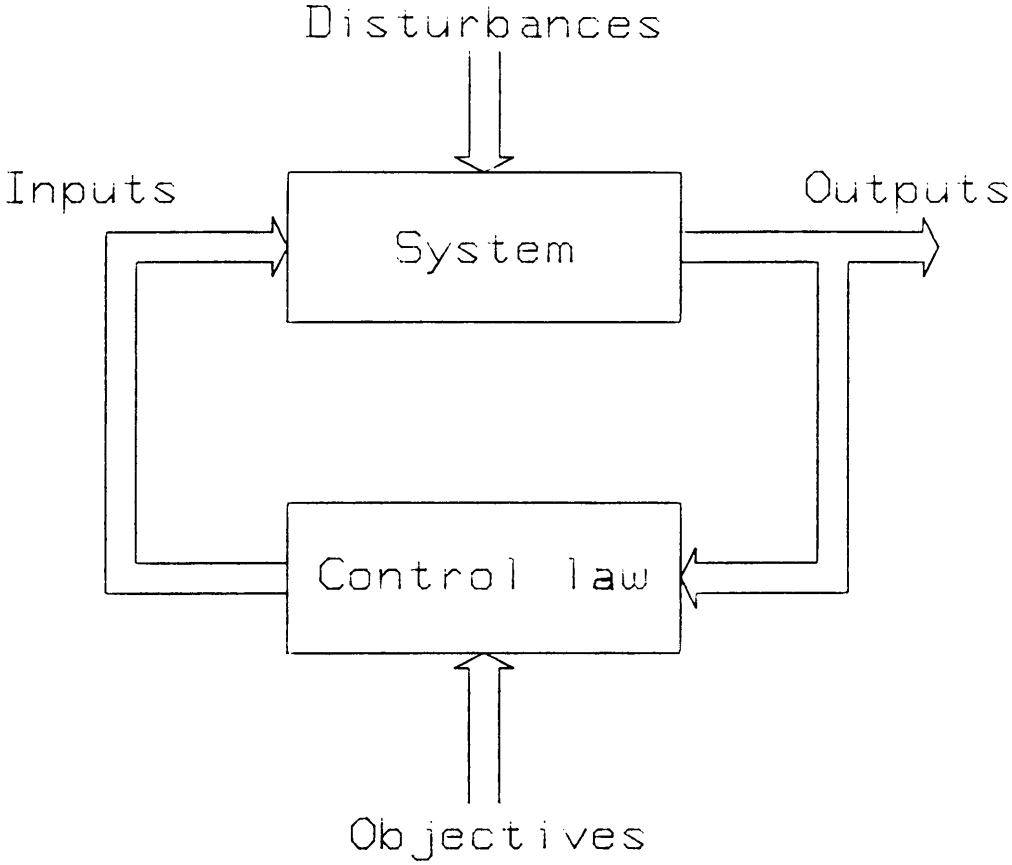

Figure 1.2 Closed-Loop Control

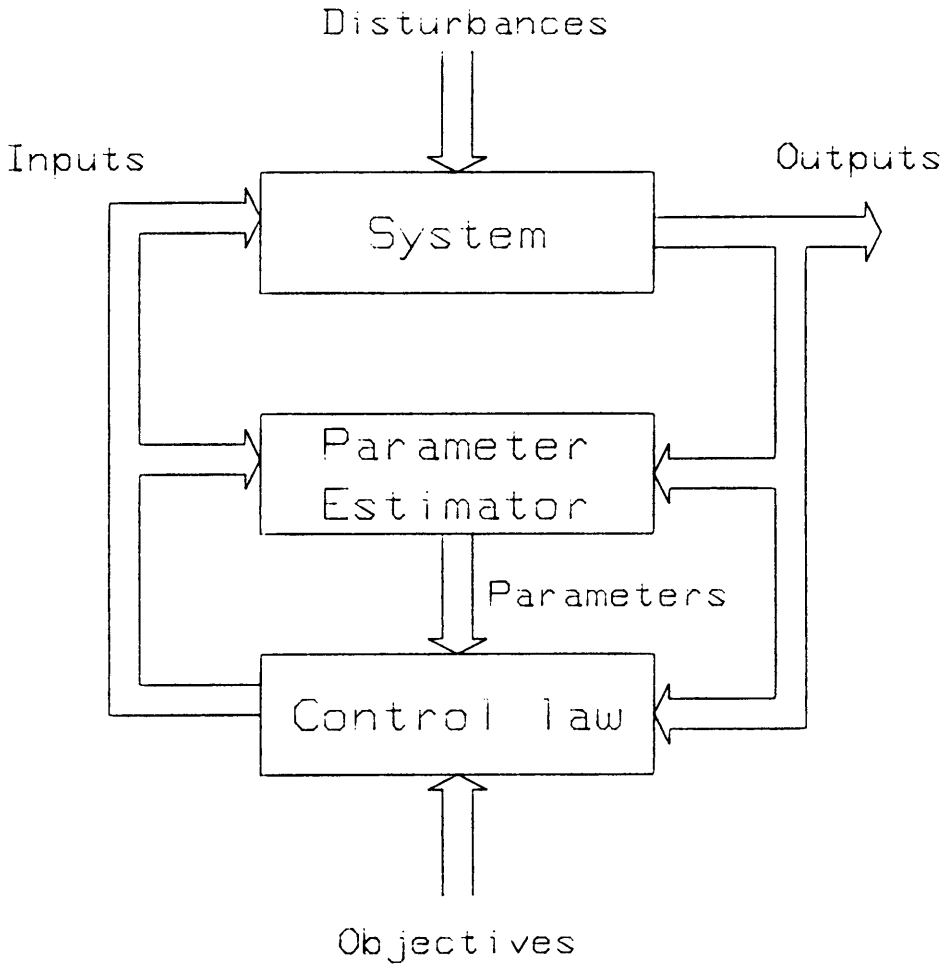

Figure 1.3 Adaptive Closed-Loop Control

the interrelationships between the system, its inputs and outputs, the identifier, and the control law synthesis procedure.

The attraction of adaptive control is its ability to compensate for unforeseen changes in the plant dynamics. A conventional control law is designed using the engineer's best guess of the plant dynamics and the assumption that these dynamic relationships will not change throughout its life. As plants age or undergo unexpected environmental changes, they may not respond to the control inputs in the predicted fashion and will therefore not meet the original control objectives. In critical applications such as aircraft and nuclear power plant control, such a failure to meet control objectives could well be catastrophic. Such plant changes within an adaptive control structure would merely lead to a change in the parameter estimate of the recursive identifier which would result in an improved control law redesigned to compensate for the new dynamics.

The success of the adaptive control approach clearly depends upon the ability of the identification algorithm to provide accurate model parameters to the control law design algorithm. Any identification algorithm used in adaptive control must also be able to respond quickly with new parameter estimates following any change in the plant dynamics.

The following chapters present a recursive identification technique which meets the requirements of fast identification crucial to adaptive control applications. This method is based on a linear Kalman filter algorithm and is called Pseudo-Linear Identification (PLID). The PLID method is first fully developed and its properties in the presence of

process and measurement noise are compared with the Recursive Maximum Likelihood method as a classical yardstick. An extension of the method to enable the tracking of time-varying parameters is then presented along with further examples for this case.

The treatment continues with the development of an adaptive control procedure which uses Ackermann's pole placement formula to compute state and integral feedback gains from the PLID parameter estimates. Examples illustrating the effectiveness of the PLID based adaptive controller in the presence of noise and time-varying plant dynamics are then presented. As a final example, the adaptive controller is used to control the longitudinal dynamics of a modern high performance aircraft. The aircraft examples illustrate the ability of the PLID based adaptive controller to "learn" the aircraft dynamics and quickly adapt to abrupt changes in the aircraft flight conditions. The adaptive controller responses are compared with those of a gain-scheduled fixed controller design to demonstrate the improvement provided by the adaptation.

The final chapter addresses the issues of numerical stability of the Kalman filter used in PLID. Sensitivity to roundoff error is reduced by reformulating the basic PLID algorithm in terms of a factorized Kalman filter. The UDU^T algorithms of Bierman [26] are used to greatly improve the numerical integrity of PLID identification and the resulting adaptive controller. A discussion of final results and suggestions for further study are then made in the conclusion.

2.0 PSEUDO-LINEAR IDENTIFICATION

Pseudo-Linear Identification (PLID) is a recursive method to simultaneously estimate the states and parameters of the discrete-time state model representing a given system. A wide variety of techniques for recursive parameter estimation have been proposed in the past. A popular survey paper by Åström and Eykhoff [3] highlights the common linear system identification approaches ranging from the basic Least Squares through Generalized Least Squares, Instrumental Variable, and Maximum Likelihood formulations. Practical details of these approaches are presented in a user's "cookbook" form in the book by Sinha and Kuszta [4]. The recursive approximation of the Maximum Likelihood method which will be used as a basis for comparison with PLID is conveniently formulated in Goodwin and Payne [5] along with an overview of the popular classical methods.

Previous attempts to treat the problem of simultaneous state and parameter estimation led to the Extended Kalman Filter [6], an outgrowth of the classical Kalman filter state estimation technique which accepted nonlinear models. The Extended Kalman Filter is also treated in the text by Ljung and Söderström [7]. This book lays a broad theoretical foundation for the entire concept of system identification. It may be noted that the Extended Kalman Filter has met with only limited success in application; the reason being that explicit products of state variables in the formulation lead to filter nonlinearities which severely degrade its performance.

In the remainder of this chapter it will be shown how proper formulation of the state model will permit the use of a linear Kalman filter algorithm for joint state and parameter estimation. Use of the well-known linear Kalman filter allows one to apply a wide variety of available filtering literature in examining and even improving the properties of PLID. Some of these key properties will then be discussed before presenting test case simulations and comparisons in the following chapter.

2.1 System State Model

The system to be identified will be assumed to have a single input signal resulting in a single output signal (SISO), though the method to be developed can be extended to multi-input, multi-output (MIMO) cases with some reformulation [8]. Such a system, when representable by a set of linear differential equations, may be modeled by a set of first-order linear differential equations of the form

$$\begin{aligned} \dot{\underline{v}}(t) &= A_c \underline{v}(t) + B_c u(t) \\ y(t) &= C_c \underline{v}(t) \end{aligned} \tag{2.1}$$

where $u(t)$ is the input and signal and $\underline{v}(t)$ is the state vector of dimension n defined as the order of the system. The output $y(t)$ is a linear combination of the state variables in $\underline{v}(t)$ implying that the system behavior is completely described by the specification of $\underline{v}(t)$.

For now, A_c, B_c, and C_c are assumed to be appropriately dimensioned matrices of constant coefficients. The continuous-time system, or plant, is thus completely modeled by an appropriate choice of the coefficients of these matrices.

Our method of identification, however, relies only on discrete-time samples of the inputs and outputs which are processed on a digital computer. We thus find it convenient to remodel the plant as an equivalent discrete-time system and identify the parameters of this discrete-time model. The relationship between the discrete-time and continuous-time models is clearly specified once a sample interval T is chosen. In particular, if we assume ideal sampling of the continuous output signal and zero-order-hold (ZOH) digital-to-analog conversion at the input, we may define [9]

$$A_d = e^{A_c T}$$

$$B_d = \int_0^T e^{A_c T} B_c dt \qquad (2.2)$$

$$C_d = C_c$$

where the matrix exponential is defined through the infinite series representation of the exponential function. Such definitions guarantee the step-input equivalence of

$$\underline{x}_d(kT+T) = A_d\underline{x}d(kT) + B_d u(kT)$$
$$y(kT) = C_d\underline{x}_d(kT) \tag{2.3}$$

to the continuous time system, i.e., for

$$u(t) = u(kT) \quad \text{over} \quad t \in [kT, kT+T) \tag{2.4}$$

one will have

$$y(kT) = y(t) \quad \text{for} \quad k = 0,1,2.... \tag{2.5}$$

Thus the response y(kT) of the discrete-time system (2.3) matches that of the continuous-time system exactly at the sample instants. In addition, the discrete-time states $\underline{x}_d$ (kT) of (2.3) correspond equivalently to samples of the states $\underline{v}(t)$ of (2.1).

We may indeed think strictly in the discrete-time domain of Equations (2.3) and consider u(k) and y(k) as sequences of numbers defined only at discrete time instants. In the strict discrete-time framework we drop the T in the notation and allow the possible inclusion of the feedthrough matrix D_d in the formulation

$$\underline{x}_d(k+1) = A_d\underline{x}_d(k) + B_d u(k)$$

$$y(k) = C_d\underline{x}_d(k) + D_d u(k) \qquad (2.6)$$

We will later use the fact that any physical dynamic system will have $D_d = 0$ since a nonzero D_d implies instantaneous response to inputs. D_d is included for generality in the derivations to follow so that certain nonphysical discrete-time processes may be treated by PLID.

The model of Equations (2.6) is assumed to represent fully the system to be identified and serves as the starting point in the PLID development to follow.

2.2 Observable-Canonical Form and Extended State

Consider an alternative description of the discrete-time system (2.6). In the z-transform domain, the transforms of u(k) and y(k) are related by

$$\frac{Y(z)}{U(z)} = \frac{b_n+b_{n-1}z^{-1} + \ldots b_1 z^{-n+1} + b_0 z^{-n}}{1 - a_{n-1}z^{-1} \ldots -a_1 z^{-n+1} - a_0 z^{-n}} \qquad (2.7)$$

which is recognized as the z domain transfer function for an n-th order system. Inverse transformation results in the equation for the output

$$y(k) = a_{n-1}y(k-1) + \ldots a_o y(k-n) + b_n u(k) +$$

$$b_{n-1}u(k-1) + \ldots + b_o u(k-n) \qquad (2.8)$$

which may be represented by the signal flow graph of Figure 2.1. Clearly, the system is described completely by n parameters (a_i) and n+1 parameters (b_i) or 2n+1 total parameters. In its full form, however, the state model (2.6) has n^2 A_d parameters, plus n B_d, and n C_d parameters for a total of $n^2 + 2n + 1$ parameters. When the two representations are in fact equivalent as required, n^2 of the parameters of (2.6) are redundant. Any attempt to estimate the parameters of Equations (2.6) directly will therefore be lost in trying to determine the redundant parameters which have no influence on the input-output data available. It is possible to reformulate the state model to eliminate this redundancy of parameters. By considering Figure 2.1 as a state diagram and treating the marked nodes as states, one may write a new state model as

$$\underline{x}(k+1) = \begin{bmatrix} 0 & 0 & \ldots & 0 & 0 & a_o \\ 1 & 0 & \ldots & 0 & 0 & a_1 \\ & & \vdots & & & \\ 0 & 0 & \ldots & 0 & 1 & a_{n-1} \end{bmatrix} \underline{x}(k) + \begin{bmatrix} b_o + a_o b_n \\ b_1 + a_1 b_n \\ \vdots \\ b_{n-1} + a_{n-1} b_n \end{bmatrix} u(k)$$

$$y(k) = [0 \quad 0 \ldots 0 \; 1] \; \underline{x}(k) + [b_n] \, u(k) \qquad (2.9)$$

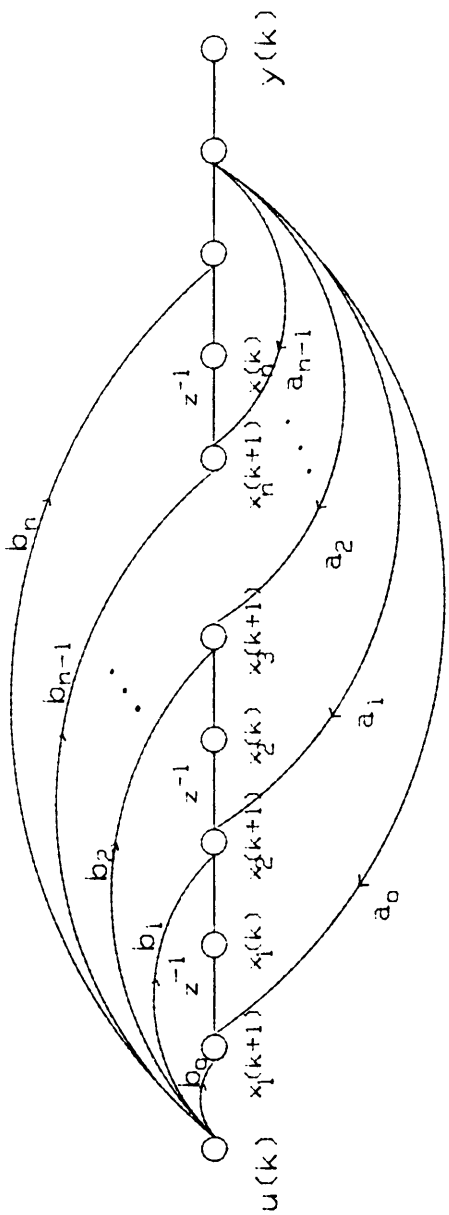

Figure 2.1 System Signal Flow Graph

This is recognized as an observable-canonical form state model of the discrete-time system. This state model has 2n+1 parameters and thus no redundant parameters.

It can be shown that any observable state model A_d, B_d, C_d, and D_d as in (2.6) may be transformed into the form of (2.9) by appropriate state transformation [10]. Observability heuristically implies that all states have an influence on the output y(k). This is not a great restriction for our purposes since our only knowledge of the system to be identified is through its measured output. Clearly, if some state does not influence y(k), the parameters related to that state cannot be identified making our task impossible. If the plant is observable, we are guaranteed the existence of a nonsingular state transformation matrix V such that for

$$\begin{aligned} \underline{x}(k) &= V\,\underline{x}_d(k) \\ A &= V^{-1}A_d\,V \\ B &= V^{-1}B_d \\ C &= C_d\,V \\ D &= D_d \end{aligned} \tag{2.10}$$

we obtain the model

$$\begin{aligned} x(k+1) &= A\underline{x}(k) + Bu(k) \\ y(k) &= C\underline{x}(k) + Du(k) \end{aligned} \tag{2.11}$$

which has the observable-canonical form of Equations (2.9). Under the transformation V, Equations (2.11), (2.9) and (2.6) are equivalent representations of the same system.

The observable-canonical form can be restructured by exploiting the correspondence between $y(k)$ and the last state variable $x_n(k)$ as

$$x(k+1) = \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ & & \vdots & & \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix} \underline{x}(k) + \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{bmatrix} y(k) + \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{n-1} \end{bmatrix} u(k) \qquad (2.12)$$

$$y(k) = [0\ 0\ \dots\ 0\ 1]\ \underline{x}(k) + [b_n]u(k)$$

Now defining the extended state,

$$\underline{s}(k) = [\underline{x}(k)^T \mid a_0\ a_1\ \dots\ a_{n-1} \mid b_0\ b_1\ \dots\ b_{n-1} \mid b_n]^T \qquad (2.13)$$

one can utilize Equations (2.12) to derive the extended state model [8]

$$\begin{aligned} \underline{s}(k+1) &= F(k)\underline{s}(k) \\ y(k) &= H(k)\underline{s}(k) \end{aligned} \qquad (2.14)$$

where

$$F(k) = \begin{bmatrix} J_n & \vdots & y(k)I_n & \vdots & u(k)I_n & \vdots & 0 \\ \dots & \vdots & \dots & \vdots & \dots & \vdots & \\ 0 & \vdots & & I_{2n+1} & & & \end{bmatrix}, \quad J_n = \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ 1 & 0 & \dots & 0 & 0 \\ & & \vdots & & \\ 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix}_{nxn}$$

and

$$H(k) = [\ 0\ 0\ \dots\ 0\ 1 \mid 0\ \dots\ 0 \mid 0\ \dots\ 0 \mid u(k)]$$

In developing the extended state model we are looking ahead to the identification problem whence the parameters become unknown variables. Our constant coefficient n-th order state model has been transformed to a time-varying autonomous state model of order 3n + 1. The key point to be made regarding this model is that it is explicitly linear with respect to the extended state. Any nonlinearity is implicitly buried in the fact that the $y(k)$ entries in $F(k)$, which are measured at each time step, are linear combinations of the states $\underline{x}(k)$ which in turn depend on the parameters of the true system. This leads to the name of Pseudo-Linear Identification since the extended state model used is only linear in an explicit sense.

2.3 Noise Structures

The discussion to this point has assumed that all measurements of the output are without error. We now extend our thinking to address the issue of imperfect or noisy measurement data as well as the case of process noise. Consider a modification of the discrete-time model (2.11) to include independent random processes $\underline{w}(k)$ and $v(k)$ as

$$\begin{aligned} \underline{x}(k+1) &= A\underline{x}(k) + Bu(k) + \underline{w}(k) \\ y(k) &= C\underline{x}(k) + v(k) \end{aligned} \tag{2.15}$$

an observable-canonical stochastic state model. Here $\underline{w}(k)$ and $v(k)$ represent process and measurement noise respectively and are included to model uncertainties in Equations (2.11). Physically, one considers $v(k)$

to arise from inaccurate output sensor data or simply quantization error from a digital transducer. $\underline{w}(k)$ is some reflection of our uncertainty in the validity of the model itself which might arise from non-linearities or neglected and unmodeled dynamics. $\underline{w}(k)$ is assumed to be a vector stochastic process of zero-mean white noise uncorrelated with the zero-mean scalar white-noise process $v(k)$. $Q_1(k)$ and $r(k)$, the covariance matrix of $\underline{w}(k)$ and variance of $v(k)$ respectively, are defined as

$$Q_1(k) = E\,[\underline{w}(k)\underline{w}(k)^T]$$

$$r(k) = E\,[v(k)v(k)] \qquad (2.16)$$

where E[ ] denotes the statistical expectation operator.

We may use Equations (2.15) to aid in refining the extended state model to include these effects. A direct and simple substitution leads to

$$\underline{s}(k+1) = F(k)\underline{s}(k) + \underline{n}(k)$$

$$y(k) = H(k)\underline{s}(k) + v(k) \qquad (2.17)$$

where

$$\underline{n}(k) = [\underline{w}(k)^T \mid \underline{o}^T]^T \qquad (2.18)$$

Clearly, $\underline{n}(k)$ then has covariance $Q(k)$ given by

$$Q(k) = \begin{bmatrix} Q_1(k) & \vdots & 0 \\ \cdots\cdots & \vdots & \cdots\cdots \\ 0 & \vdots & 0 \end{bmatrix} \tag{2.19}$$

Equations (2.17) are a direct application of the model of (2.15) which considers the $\underline{s}(k+1)$ process equation and the $y(k)$ observation equation to be separate and independent operations. In (2.17) the process and measurement noise can still be taken to be uncorrelated processes which greatly simplifies further developments. Closer inspection, of course, reveals that the "noisy" $y(k)$ observations appear explicitly within $F(k)$. Expansion of $y(k)$ into its deterministic and noise components leads to an additional process noise component in (2.17) which is correlated with the measurement noise $v(k)$ and depends upon the system parameters [8].

Reference [8] treats this case in detail and suggests that inclusion of the additional process noise and its correlation with $v(k)$ only improves PLID results after a nearly correct parameter estimate has been achieved. Treatment of this additional noise is therefore only recommended in cases of high measurement noise and as an estimate refining technique after the identified parameters have achieved some steady-state plateau. Clearly, the future application of PLID to the adaptive control problem suggests that fast parameter tracking in the transient phase of the algorithm should be the overriding concern in the method development. The presence of slight parameter bias in low noise

cases typically has only minor effects on the performance of a closed-loop adaptive system [11]. For these reasons, the form of Equations (2.17) will be taken as an adequate system description for the development of the PLID algorithm to follow. Further research might examine any added accuracy to be gained by combining the method of [8] with the Tracking-PLID extension developed in Chapter 4.

2.4 Kalman Filter Estimation

The stochastic extended state model of (2.17) is still time-varying and autonomous even with the noise vectors appended. One final modification is necessary to make the model a causal structure. Note that in (2.17) F(k) depends on y(k) which has yet to be measured. We thus shift the order of the equations by taking the extended state model to equivalently be

$$\begin{aligned} \underline{s}(k) &= F(k-1)\underline{s}(k-1) + \underline{n}(k-1) \\ y(k) &= H(k)\underline{s}(k) + v(k) \end{aligned} \tag{2.20}$$

This modification is made in anticipation of the calculation requirements of the estimator and ensures that no information will be needed before it is available.

Our goal, then, is to estimate effectively the extended state of (2.20) using input and output data samples. Clearly, a good estimate of $\underline{s}$ implies that the parameters of the model (2.9) are identified since they are elements of $\underline{s}$. It is necessary now to quantify our criterion

for a "good" estimate of $\underline{s}$. Suppose that the system of (2.9) has true parameters $\underline{p}_0$ and that the true state of the system at time k is $\underline{x}_0(k)$. The true extended state at k is then $\underline{s}_0(k)$ such that

$$\underline{s}_0(k) = [\underline{x}_0(k)^T \mid \underline{p}_0^T]^T \tag{2.21}$$

A good estimate $\underline{s}(k)$ must be in some sense "close" to $\underline{s}_0(k)$. A reasonable definition of "close" in terms of the Euclidean norm of the vector difference between $\underline{s}_0(k)$ and $\underline{s}(k)$ is given by

$$\varepsilon = E\,[(\underline{s}_0(k) - \underline{s}(k))^T\,(\underline{s}_0(k) - \underline{s}(k))] \tag{2.22}$$

A small value of ε guarantees that the expected value of the mean-square error of the estimate is small. It is thus our goal to develop an estimation approach which minimizes the criterion function ε at any time k.

Fortunately, a review of the literature reveals that such an estimator has already been developed and studied extensively. The discrete-time Kalman filter algorithm is the optimal recursive estimator of the state of (2.20) in the sense that it produces estimates which yield a minimum value of ε [6]. This linear filter is given by

$$M(k) = F(k-1)P(k-1)F^T(k-1) + Q(k-1) \tag{2.23a}$$

$$K(k) = M(k)H^T(k)/[H(k)M(k)H^T(k) + r(k)] \tag{2.23b}$$

$$P(k) = [I - K(k)H(k)]\, M(k) \tag{2.23c}$$

$$\underline{s}(k) = F(k-1)\underline{s}(k-1) + K(k)[y(k) - H(k)F(k-1)\underline{s}(k-1)] \tag{2.23d}$$

The algorithm requires initial values of $P(0)$ and $\underline{s}(0)$ for startup. Ljung and Söderström [7] suggest that in lieu of any better information one should initialize the algorithm with

$$\underline{s}(k) = \underline{0}$$

and (2.24)

$$P(o) = cI_{3n+1}$$

where c is a large positive number and I_{3n+1} is a 3n+1 x 3n+1 identity matrix. One may view the covariance matrix P as a measure of uncertainty in the estimated state. The use of a large number c in (2.24), therefore, is a reflection of our uncertainty in setting $\underline{s}(0)=\underline{0}$.

Of course, any previous knowledge of parameter and state values should be included in the initializations. Prior knowledge of the system poles, for instance, would allow one to properly set the (a_i) parameters within $\underline{s}(0)$. Our increased certainty in these particular elements of $\underline{s}(0)$ should then be reflected by modifying $P(0)$ to

$$P(o) = \begin{bmatrix} cI_n & 0 & 0 \\ 0 & c'I_n & 0 \\ 0 & 0 & cI_{n+1} \end{bmatrix} \tag{2.25}$$

where $c' << c$. It is generally not recommended, however, to set $c' = 0$ even when one is completely certain of the corresponding estimated element. The resulting loss of positive definiteness of P could lead to a loss of filter stability.

The Equations (2.23) with the definitions of (2.14) and (2.16) form the Pseudo-Linear Identification method. While Equations (2.23) form a basic linear Kalman filter algorithm, there are some implicit non-linearities embedded within the $F(K-1)\ \underline{s}(k-1)$ product. The explicit linear properties of the filter do, however, allow the application of the wide array of Kalman filter literature in investigating and extending this basic PLID form. The PLID formulation is thus a very attractive alternative to the Extended Kalman Filter since it provides simultaneous state and parameter estimates within a linear framework.

2.5 Basic PLID Properties

An obviously desirable property of PLID is that as the recursion of (2.23) progresses, the estimates of the extended state should converge on their true values. Such a convergence of the parameter estimates is equivalent to the identification of the original observable-canonical form state model of the system. The conditions under which such a convergence is guaranteed will now be established.

The unique form of the extended state model (2.14) leads to a simple condition on the identifiability of the original system. In particular consider [8]

$$\begin{bmatrix} y(o) \\ y(1) \\ \vdots \\ y(k) \end{bmatrix} = \begin{bmatrix} H(o) \\ H(1)F(o) \\ \vdots \\ H(k)F(k-1)\ldots F(o) \end{bmatrix} \quad \underline{s}_o(o) \triangleq \theta_k \, \underline{s}_o(o) \tag{2.26}$$

The matrix θ_k is recognized as the k-th observability matrix of the extended state model (ESM). The fact that the θ_k has 3n+1 columns suggests that whenever $\text{rank}(\theta_k) = 3n+1$ one could solve equation (2.26) for $\underline{s}_o(0)$ by a suitable generalized matrix inversion. A solution for $\underline{s}_o(0)$ would clearly imply identification of the system parameters. Since $\text{rank}(\theta_k) = 3n+1$ is the condition for observability of the ESM, we may claim that the identifiability of the system is contingent upon the observability of the ESM.

It therefore requires at least 3n+1 observation steps before identification is possible. This follows again from the fact that θ_k has 3n+1 columns and could not possibly have rank=3n+1 before it has 3n+1 rows also. In addition, anytime θ_k has 3n+1 independent rows, the system is identifiable. To investigate the sources of row independence we further examine θ_k as

$$\theta_k = \begin{bmatrix}
0\ 0\ \ldots\ 0\ 0\ 1 & 0\ 0\ \ldots\ 0 & 0 & 0\ 0\ \ldots\ 0 & 0 & u(o) \\
0\ 0\ \ldots\ 0\ 1\ 0 & 0\ 0\ \ldots\ 0 & y(o) & 0\ 0\ \ldots\ 0 & u(o) & u(1) \\
 & & \vdots & & & \\
1\ 0\ \ldots\ 0\ 0\ 0 & 0\ y(o)\ \ldots & y(n-3)\ y(n-2) & 0\ (o)\ \ldots & u(n-3)\ u(n-2) & u(n-1) \\
0\ 0\ \ldots\ 0\ 0\ 0 & y(0)\ \ldots & y(2n-3)\ y(2n-2) & (n-1)\ \ldots & u(n-2)\ u(n-1) & u(n) \\
 & & \vdots & & & \\
0\ 0\ \ldots\ 0\ 0\ 0 & y(n-1)\ \ldots & y(2n-3)\ y(2n-2) & (n)\ \ldots & u(2n-3)\ u(2n-2) & u(2n-1) \\
0\ 0\ \ldots\ 0\ 0\ 0 & y(n)\ \ldots & y(2n-2)\ y(2n-1) & (n)\ \ldots & u(2n-2)\ u(2n-1) & u(2n) \\
 & & \vdots & & & \\
0\ 0\ \ldots\ 0\ 0\ 0 & y(2n-1)\ldots & y(3n-3)\ y(3n-2) & (2n-1)\ldots & u(3n-3)\ u(3n-2) & u(3n-1) \\
0\ 0\ \ldots\ 0\ 0\ 0 & y(2n)\ \ldots & y(3n-2)\ (3n-1) & (2n)\ \ldots & u(3n-2)\ u(3n-1) & u(3n)
\end{bmatrix} \tag{2.27}$$

To insure that θ_k has the highest possible rank, Equation (2.27) suggests we should choose the sequence u(k) to vary strongly from instant to instant. It is recommended in the literature that u(k) should be a random sequence of numbers, thus approximating white noise [4]. This is not surprising since white noise has equal power at all frequencies and would certainly excite all modes of the system. It is clear that any unexcited mode would not appear in the outputs y(k) and could not be identified. Such "persistent excitation" requirements are common to all identification methods [5]. We will take a persistently exciting input to be one that insures θ_k achieving full rank in 3n+1 samples. If such an input condition is assumed, we will have identifiability guaranteed in 3n+1 steps exactly.

It has been shown [8], in fact, that the PLID algorithm can converge to the true parameter values in 3n+1 steps. Such rapid convergence requires a persistently exciting input, no modeling errors and deterministic (noise free) measurements with no process noise. Related deterministic results have shown that if lack of excitation delays θ_k from achieving full rank until sample 3n+1+p, convergence will occur on, but not before the (3n+1+p)-th iteration of the PLID algorithm. This result allows PLID to be easily applied in on-line identification applications because persistent excitation cannot be insured when sampling data from a plant in its normal operating condition. Loss of persistent excitation will not "turn off" PLID as it might with other algorithms; it merely delays convergence until enough independent rows of θ_k have been generated. Once the excitation becomes

insufficient, PLID merely holds its last parameter estimate while continuing to estimate the system states appropriately.

PLID in the deterministic case is thus deadbeat, meaning that it identifies in a minimum number of steps (equal to the filter order). Convergence of the algorithm is contingent upon a simple observability criterion and loss of excitation will not result in an algorithm divergence. Many of these properties hold in the case of low process and measurement noise but, as yet, no theoretical convergence proof exists for the stochastic case. High measurement noise cases do lead to bias (steady state error) in the parameters which can be removed using the extended noise models of [8] referred to in Section 2.3. Convergence in low noise cases using the basic algorithm of (2.23) can be demonstrated by simulation in lieu of a theoretical proof. The next chapter compares the performance of PLID in low noise to that of the popular Recursive Maximum Likelihood method.

3.0 PLID PERFORMANCE IN SIMULATION

The lack of a theoretical stochastic convergence proof for the PLID algorithm necessitates the use of computer simulation in investigating the stochastic properties of PLID. The following section will demonstrate the effectiveness of PLID for a variety of plants and noise environments as well as establish limitations of the basic formulation. A comparison with the Recursive Maximum Likelihood (RML) method will then establish the relative merits of PLID against a popular standard.

3.1 PLID Simulation Structure

Input and output data from four distinctly different discrete-time plant models under a variety of noise conditions are generated in the following manner.

$$\underline{x}_0(k+1) = \begin{bmatrix} 0 & 0 & a_0 \\ 1 & 0 & a_1 \\ 0 & 1 & a_2 \end{bmatrix} \underline{x}_0(k) + \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} u(k) + \begin{bmatrix} w_1(k) \\ w_2(k) \\ w_3(k) \end{bmatrix} \tag{3.1}$$

$$y(k) = [\ 0\ 0\ 1\ ]\ \underline{x}_0(k) + v\ (k)$$

The discrete-time Equations (3.1) are implemented in the subroutine PLANT, listed in the Software Appendix, where (a_i) and (b_i) are the true parameters of the system to be identified. The input u(k) applied is a zero-mean white noise sequence having variance 1, i.e.,

$$E\,[u(k)] = 0$$

$$E\,[u(k)u(l)] = \delta(k-l) \tag{3.2}$$

The process noise vector $\underline{w}(k)$ is also zero-mean white noise with all $w_i(k)$ independent and of variance q, i.e.,

$$E\,[\underline{w}(k)] = \underline{0}$$

$$E\,[\underline{w}(k)\underline{w}(l)^T] = \begin{bmatrix} q & 0 & 0 \\ 0 & q & 0 \\ 0 & 0 & q \end{bmatrix} \delta(k-l) \tag{3.3}$$

Similarly v(k) is zero-mean with variance r. A Guassian random number generation subroutine [12] is called at each k to calculate appropriate random values $\underline{w}(k)$ and v(k) and thus simulate the noise environment.

One will also note the absence of the b_3 or b_n term in Equations (3.1). This is in keeping with the ultimate goal of applying PLID in adaptive control situations for physical plants. Such plants do not have the b_n feedthrough term whose presence indicates instantaneously responding plant dynamics. The PLID algorithm used in the simulations is correspondingly modified to reflect the absence of this term. The extended state vector is simply

$$\underline{s}(k) = [\underline{x}(k)^T \mid a_o \;\; a_1 \;\; a_2 \mid b_o \;\; b_1 \;\; b_2\,]^T \tag{3.4}$$

and the Kalman filter is reduced to order 3n. Matrices F(k) and H(k) are simply the first 3n rows and columns of the matrices in Equations (2.14). A FORTRAN implementation of the PLID algorithm of order 3n is presented as Subroutine PLID in the Software Appendix. The interested reader will recognize the major elements of equations (2.23) in the code to aid in proper use and interpretation of the subroutine in applications.

A main executive program IDENT presented in the Software Appendix interactively ties the PLANT and PLID routines together to demonstrate PLID performance using the input and output samples of the simulated plant model. IDENT performs the necessary initializations in response to user prompts, gives screen output of the recursion results and manages file structured result storage. The simulations were run on an I.B.M. Personal Computer in FORTRAN [13] and utilizing an Intel 8087 floating-point math coprocessor for added computational speed.

3.2 Test Case Descriptions

The transfer functions simulated to test the PLID algorithm were chosen to present a wide variety of pole-zero configurations as follows:

Plant	Transfer Function	Pole-Zero Set	Categorization
A	$\frac{-1.8z^2+1.3z - 0.4}{z^3-1.8z^2+1.3z-0.4}$	$\frac{0.36\pm0.3j}{0.8,\ 0.5 \pm 0.5j}$	Minimum phase zeros Stable poles
B	$\frac{-3z^2 + 2z-1}{z^3-z^2 + 2z-3}$	$\frac{0.33\pm0.47j}{1.28,-0.14\pm1.52j}$	Minimum phase zeros Highly unstable poles

C	$\frac{1.5z^2-2z-0.5}{z^3-1.8z^2+1.3z-0.4}$	$\frac{1.0 \pm 1.23j}{0.8,\ 0.5 \pm 0.5j}$	Non-minimum phase zeros Stable poles
D	$\frac{-z^2-2.5z+3}{z^3-2.5z^2+3z-1}$	$\frac{1.25 \pm 1.2j}{0.5,\ 1.0 \pm 1.0j}$	Non-minimum phase zeros Unstable poles

Input-output data for each of the four plants was generated under the following seven noise conditions:

Deterministic	r = 0.	q = 0.
Low measurement noise	r = 0.005	q = 0.
Moderate measurement noise	r = 0.01	q = 0.
Moderate process noise	r = 0.	q = 0.01
High process noise	r = 0.	q = 0.01
Low measurement and process noise	r = 0.005	q = 0.005
Moderate measurement and process noise	r = 0.01	q = 0.01

The PLID algorithm applied in the twenty-eight cases was initialized uniformly with

$$\begin{aligned} \underline{s}(o) &= \underline{0} \\ P(o) &= \text{diag}(200, 200 \ldots. 200) \\ Q_1(k) &= \text{diag}(q \ldots\ldots q) \qquad \forall k \\ r(k) &= r \qquad \forall k \end{aligned} \tag{3.5}$$

and the recursion carried out until a near steady state parameter estimate was established in the output traces.

3.3 Test Case Results and Conclusions

Time history traces for PLID in the twenty-eight test cases are graphically presented in Figures 3.1-3.28. The plots are labeled to

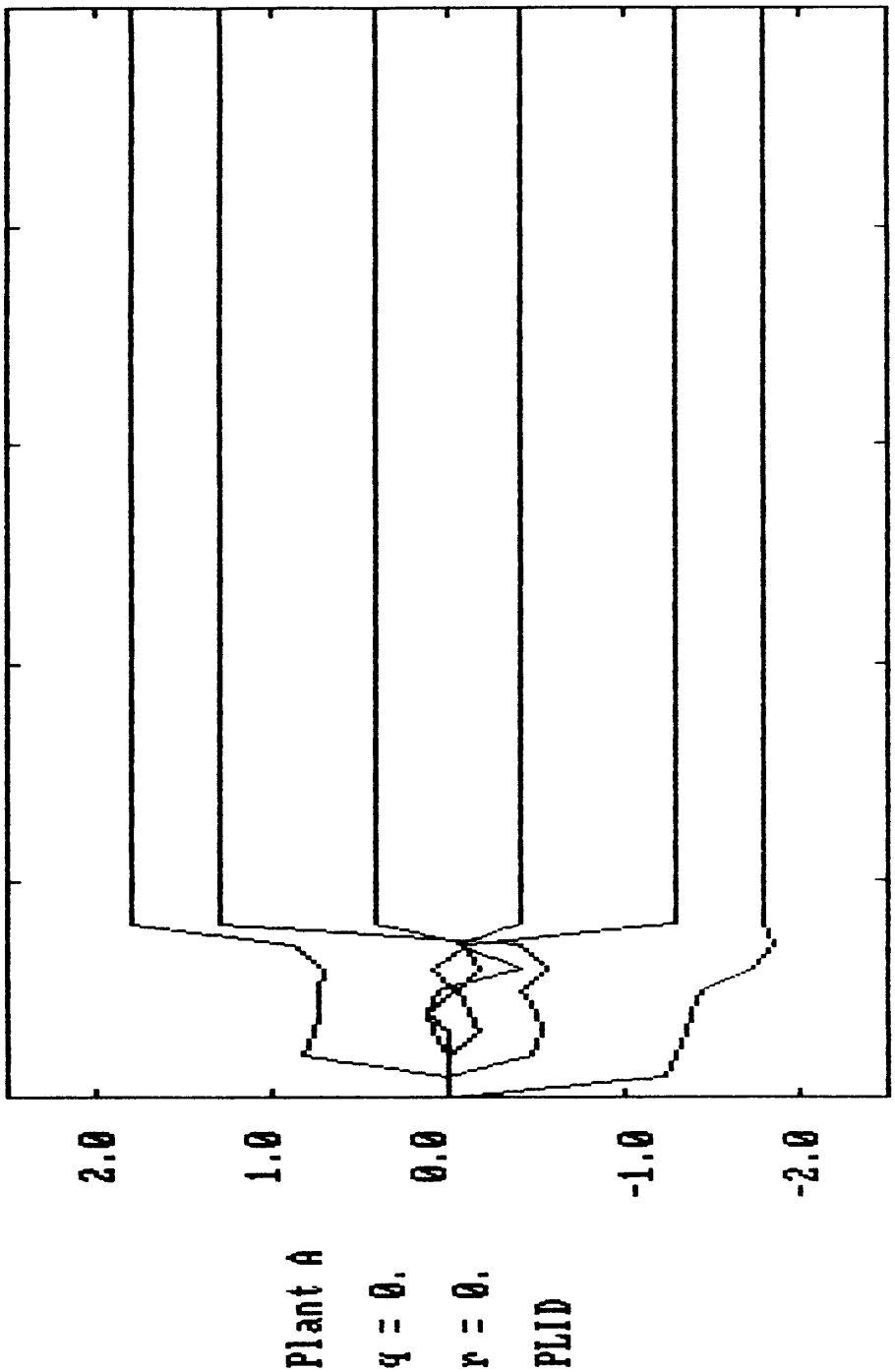

Figure 3.1 PLID Simulation, Plant A, Case 1

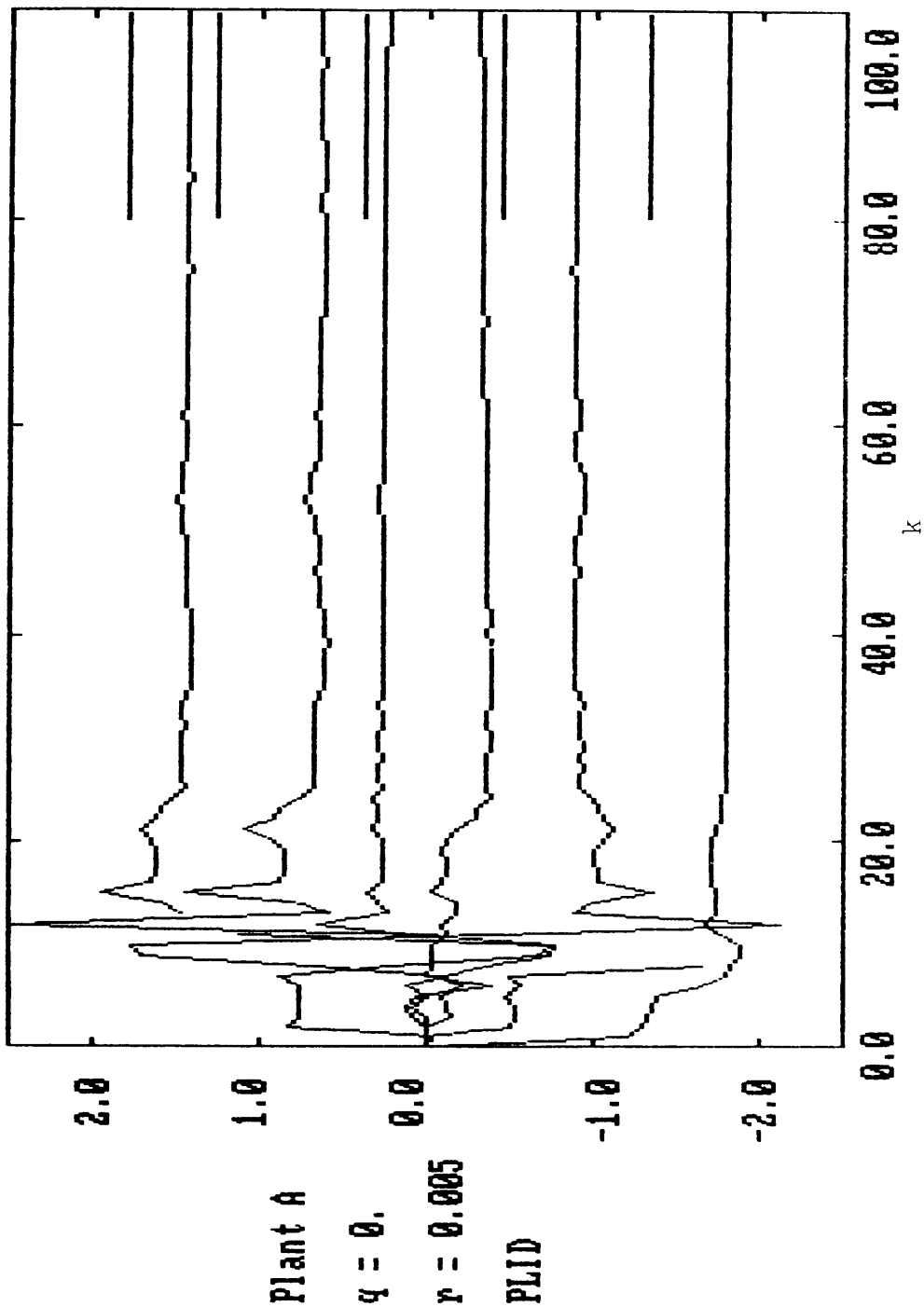

Figure 3.2 PLID Simulation, Plant A, Case 2

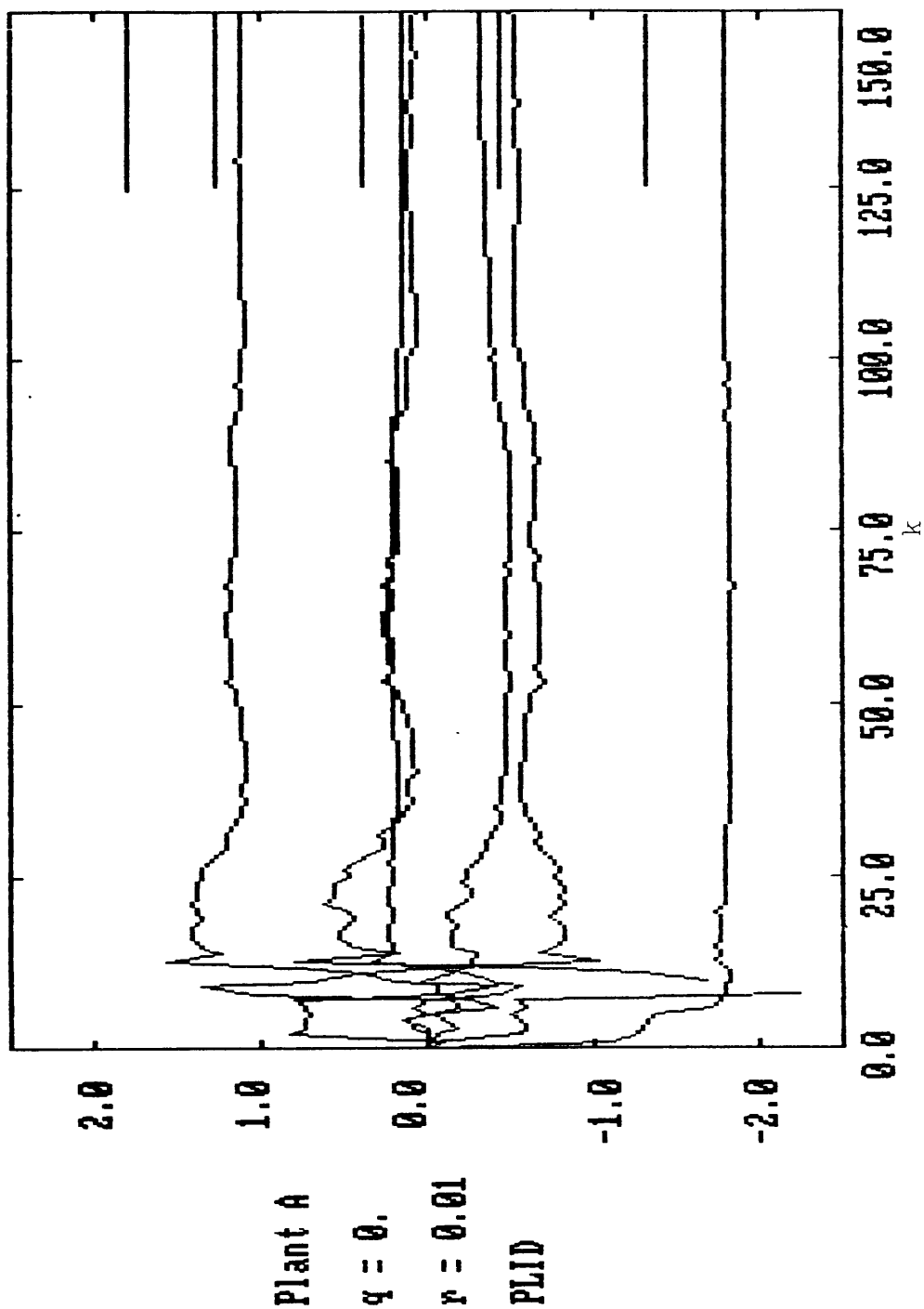

Figure 3.3 PLID Simulation, Plant A, Case 3

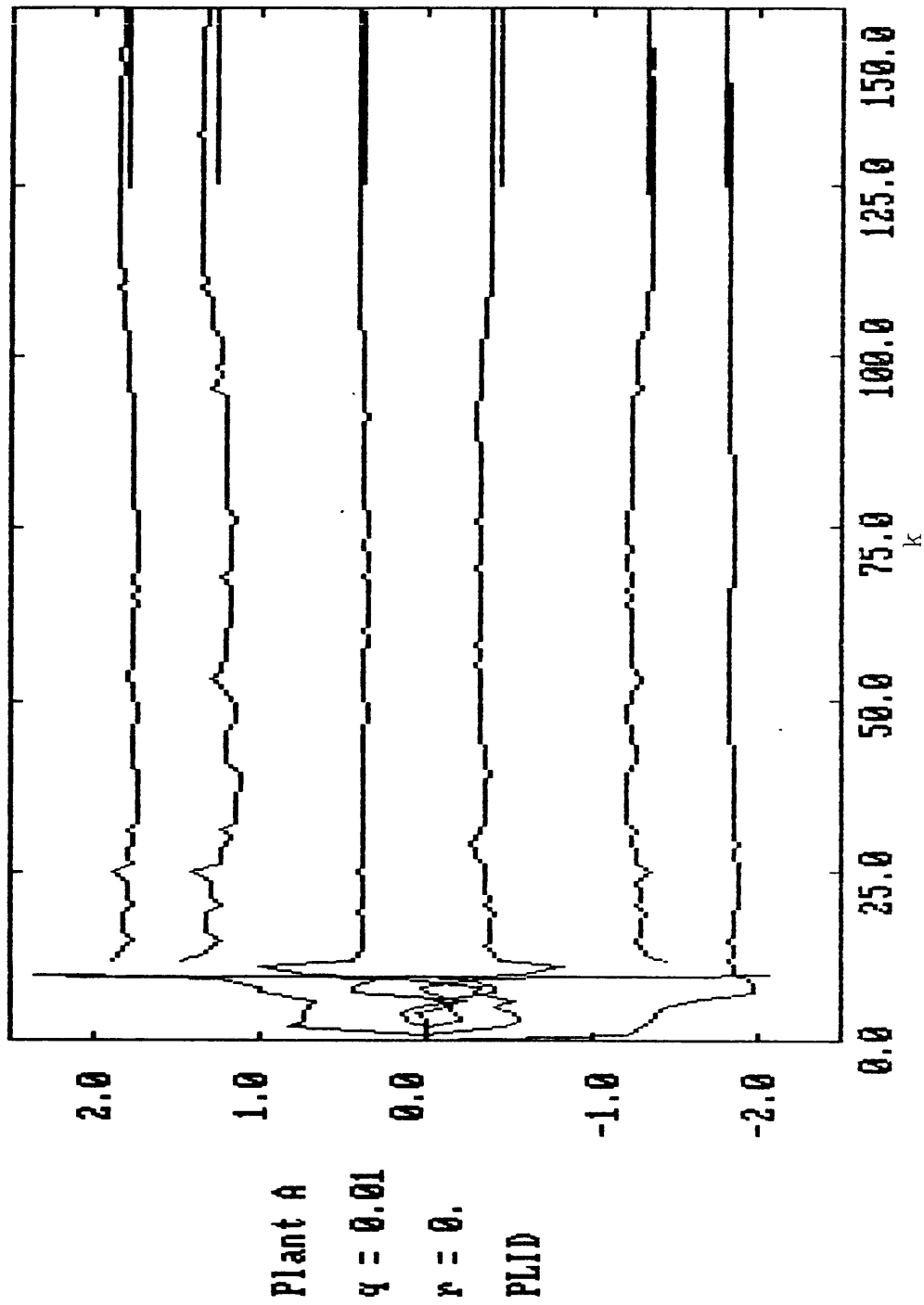

Figure 3.4 PLID Simulation, Plant A, Case 4

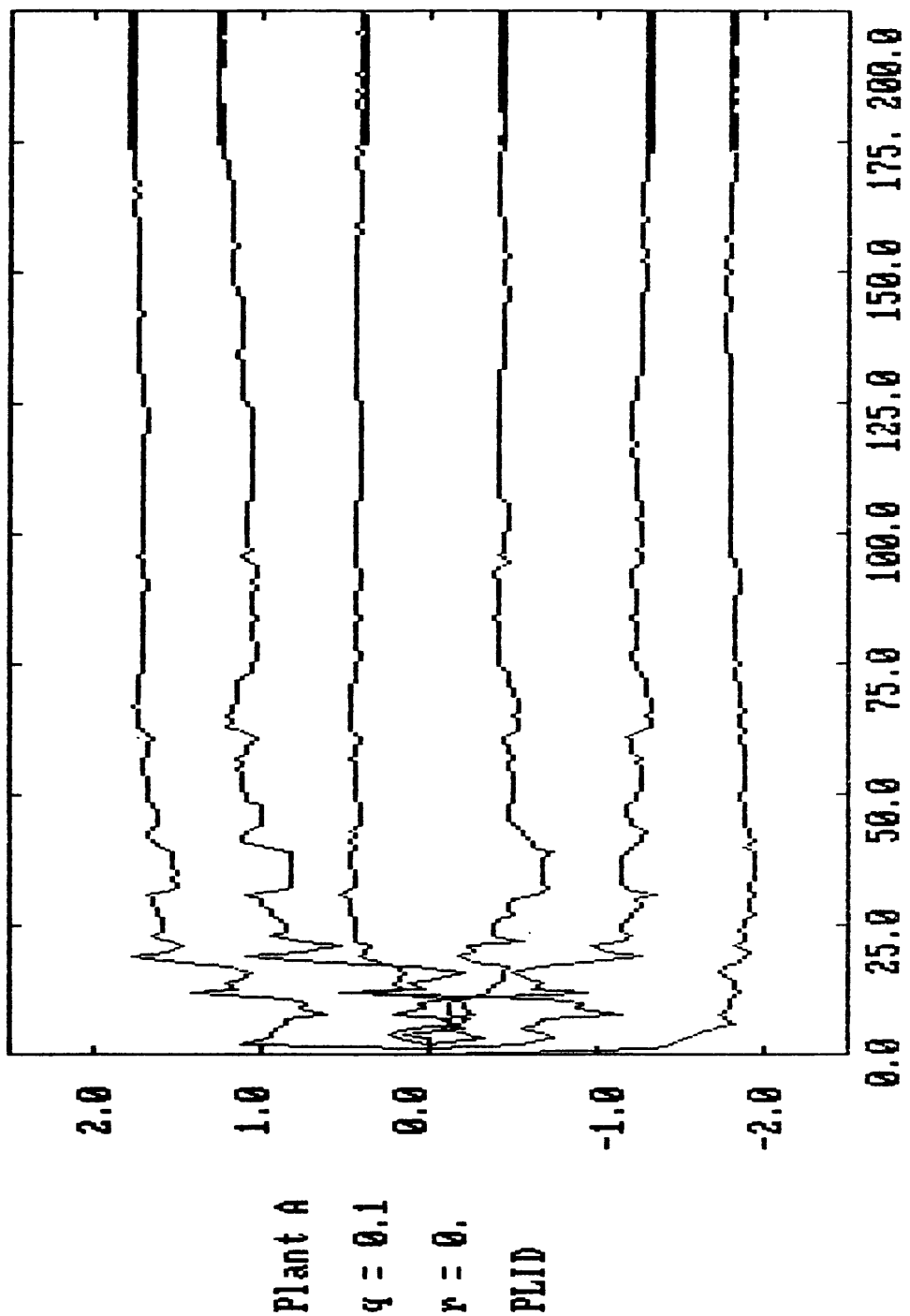

Figure 3.5 PLID Simulation, Plant A, Case 5

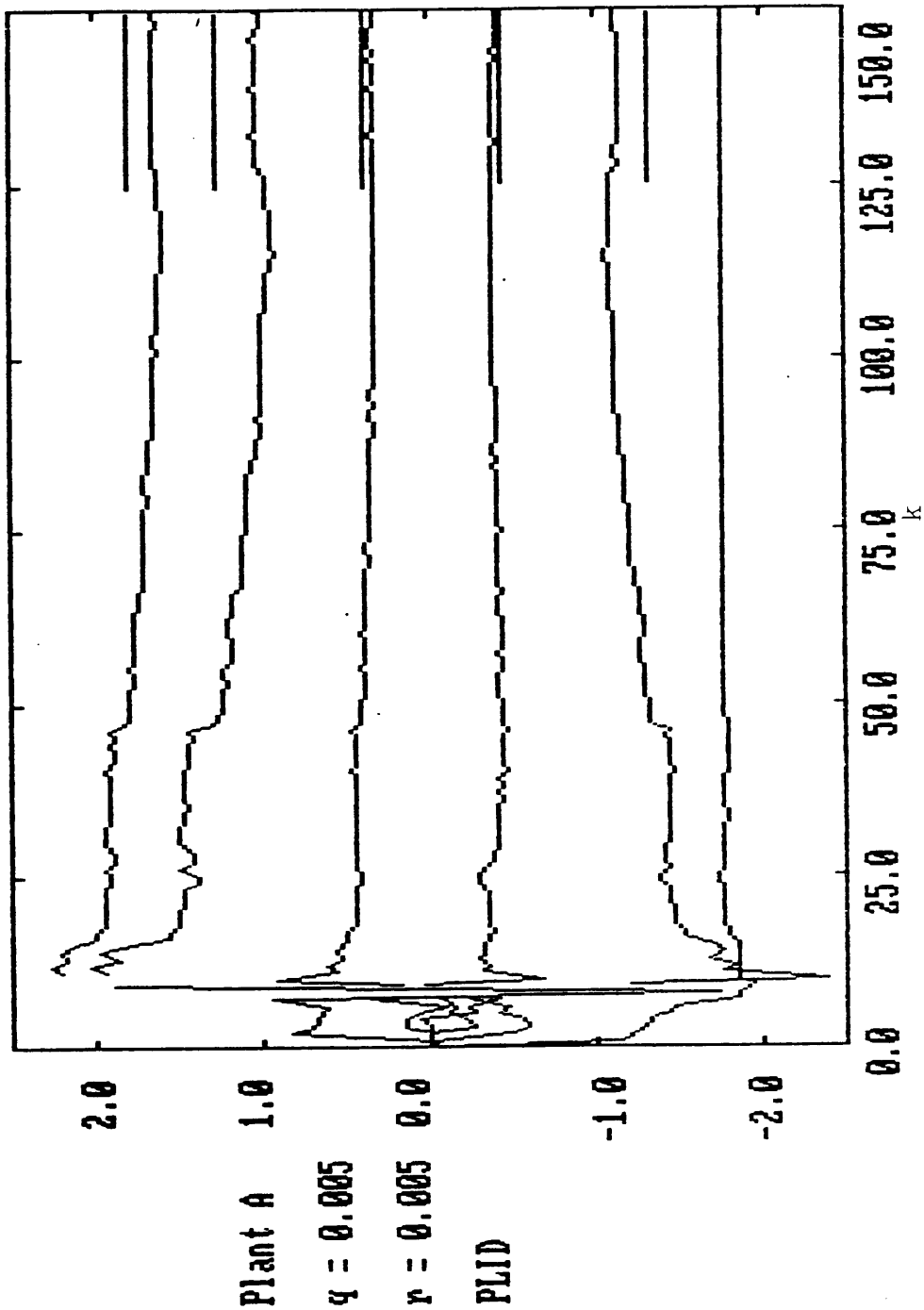

Figure 3.6 PLID Simulation, Plant A, Case 6

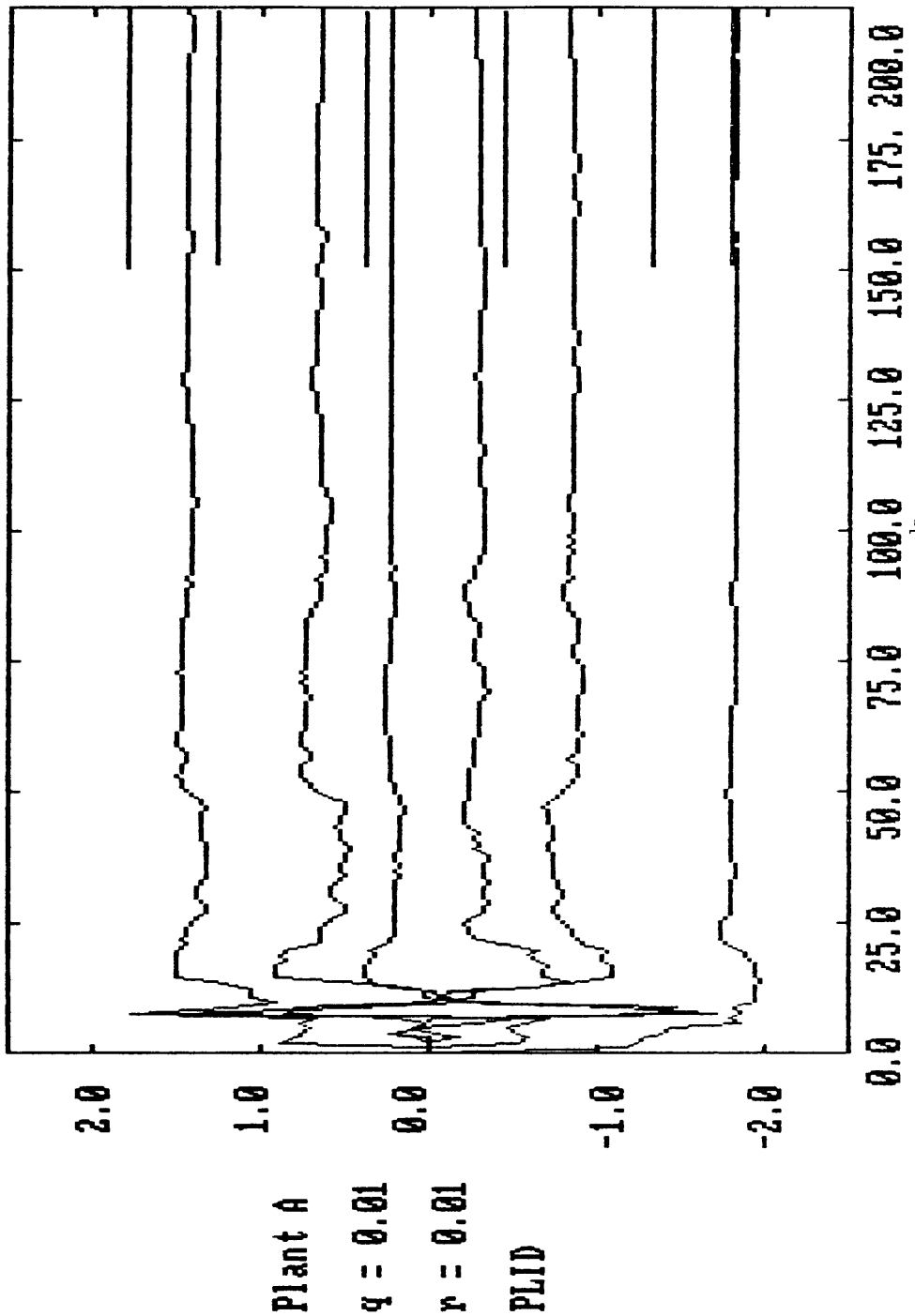

Figure 3.7 PLID Simulation, Plant A, Case 7

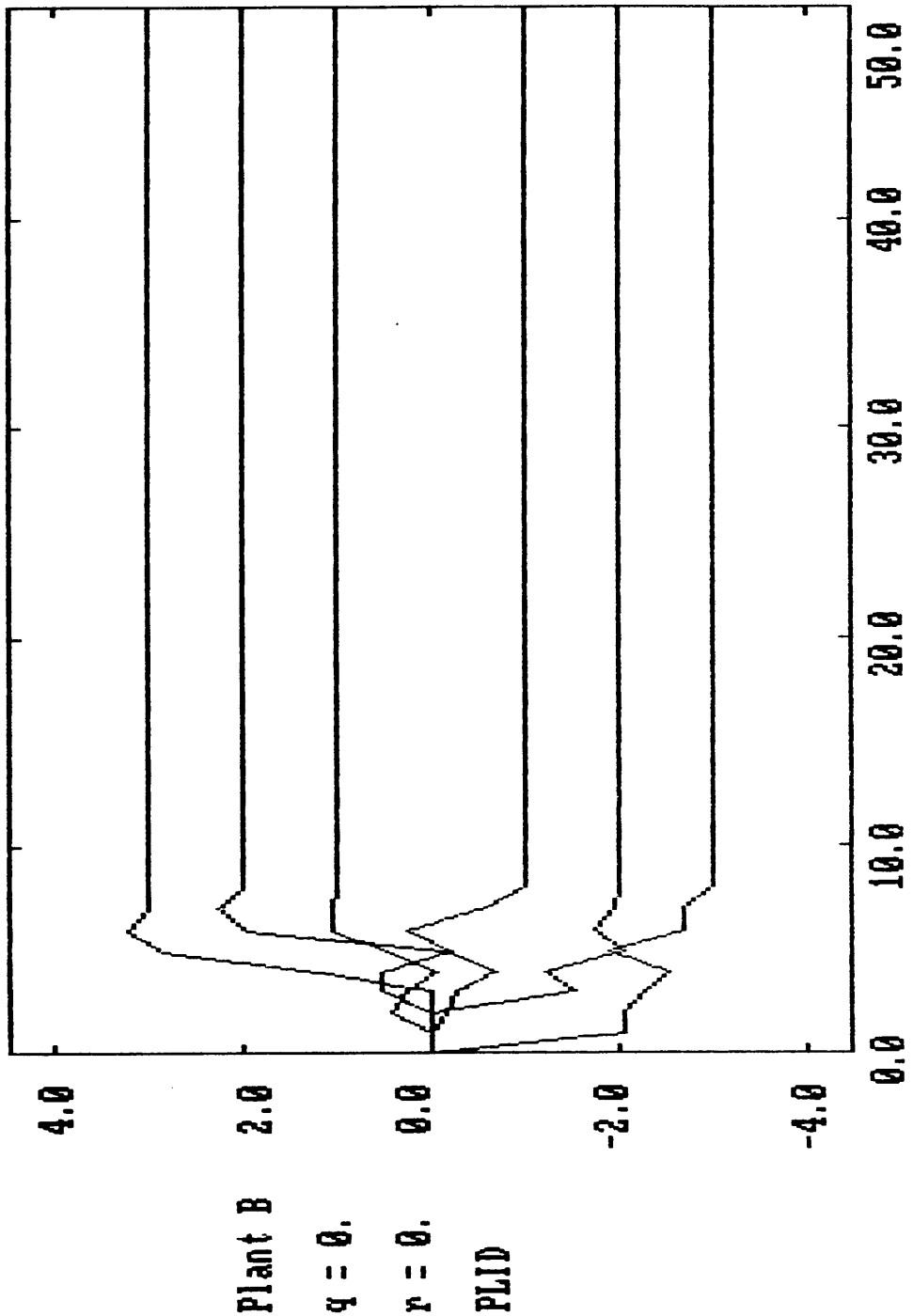

Figure 3.8 PLID Simulation, Plant B, Case 1

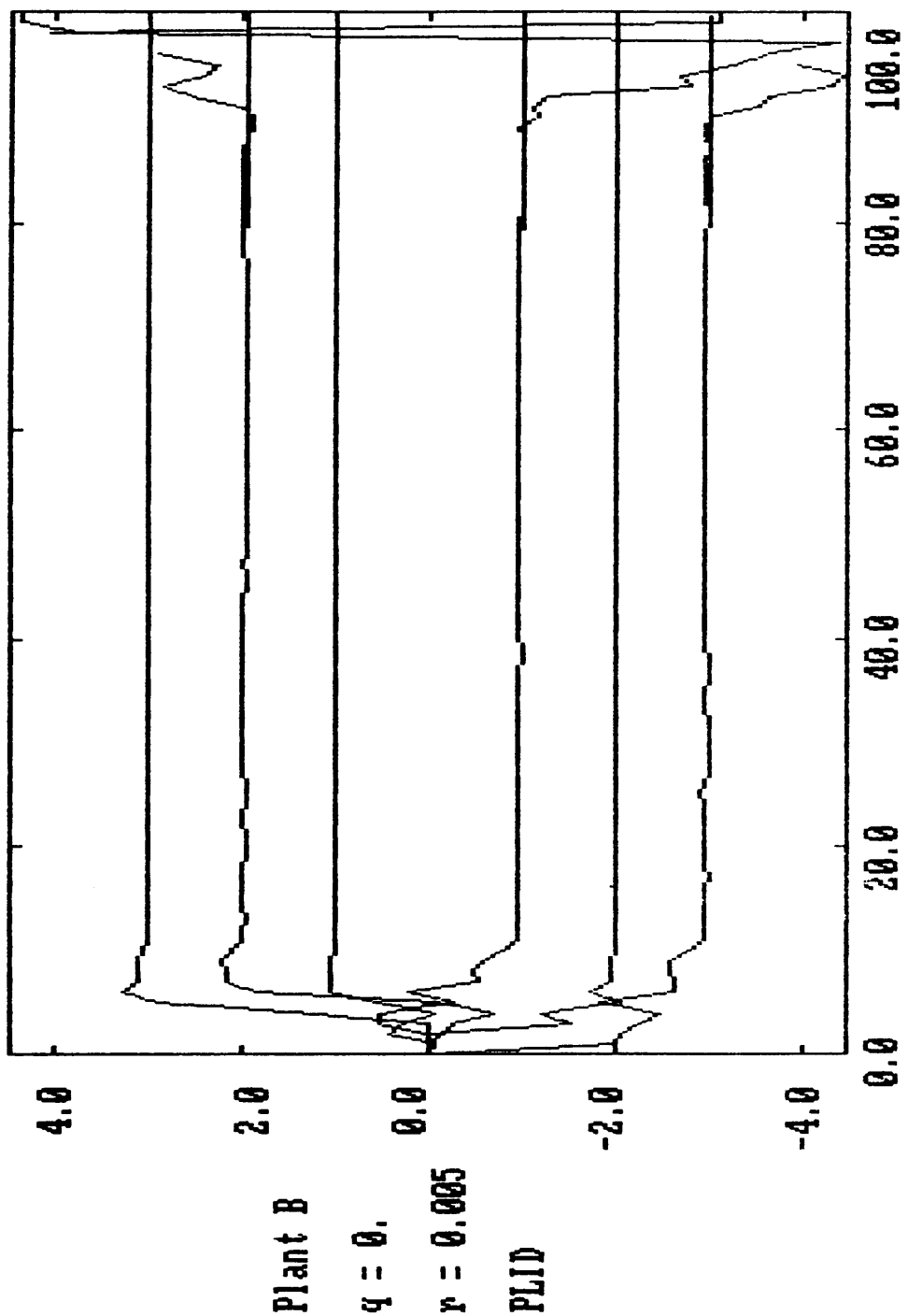

Figure 3.9 PLID Simulation, Plant B, Case 2

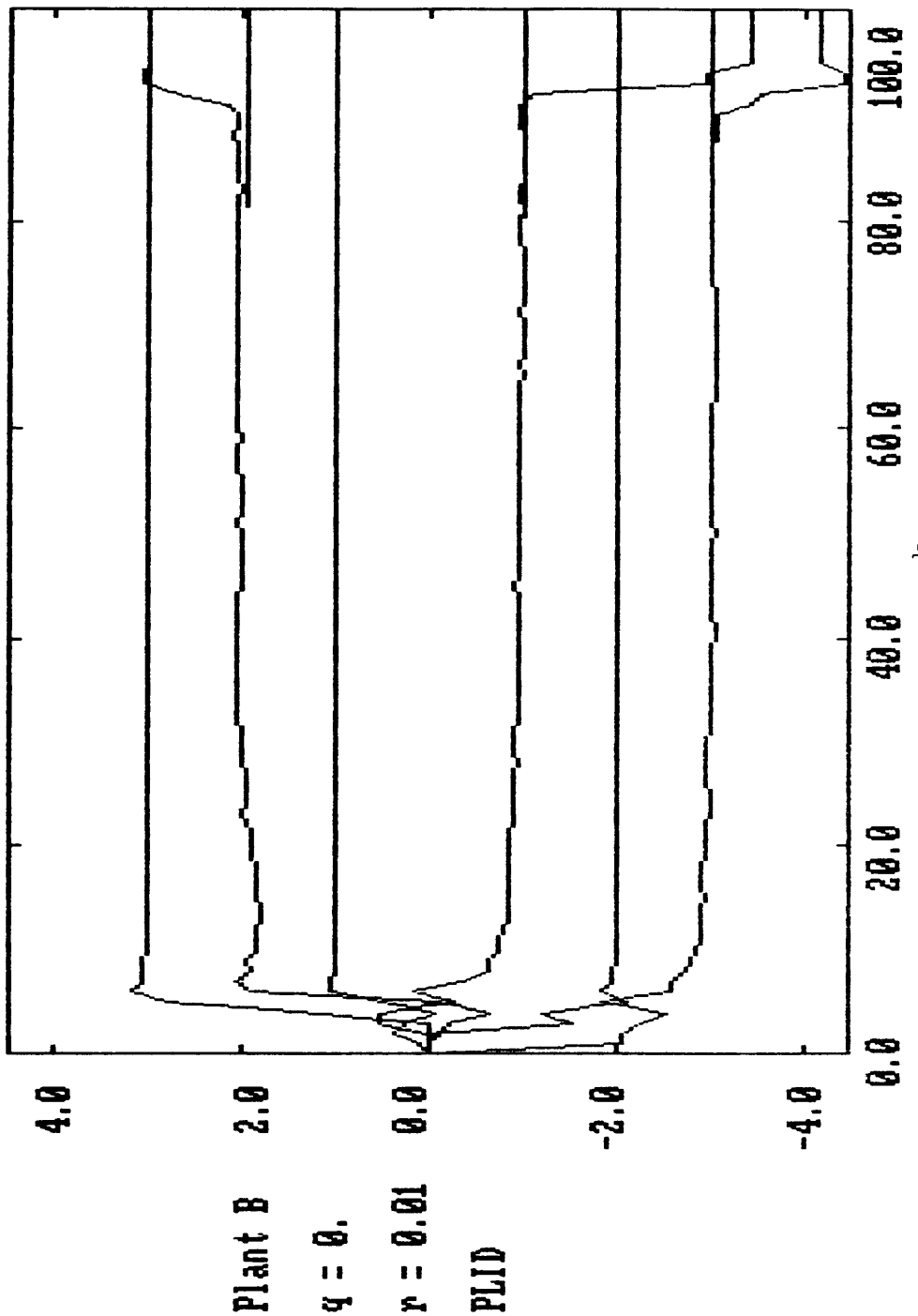

Figure 3.10 PLID Simulation, Plant B, Case 3

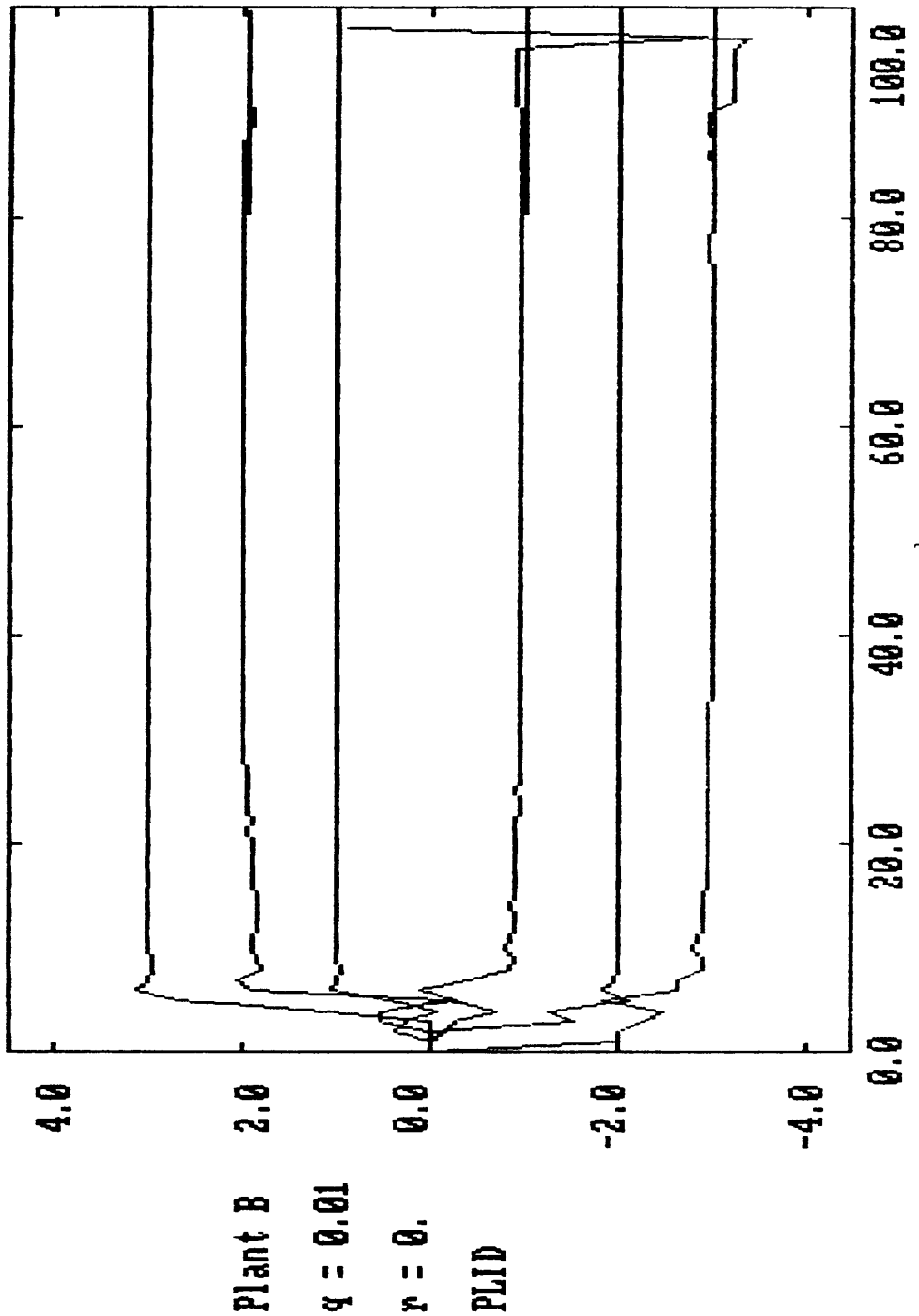

Figure 3.11 PLID Simulation, Plant B, Case 4

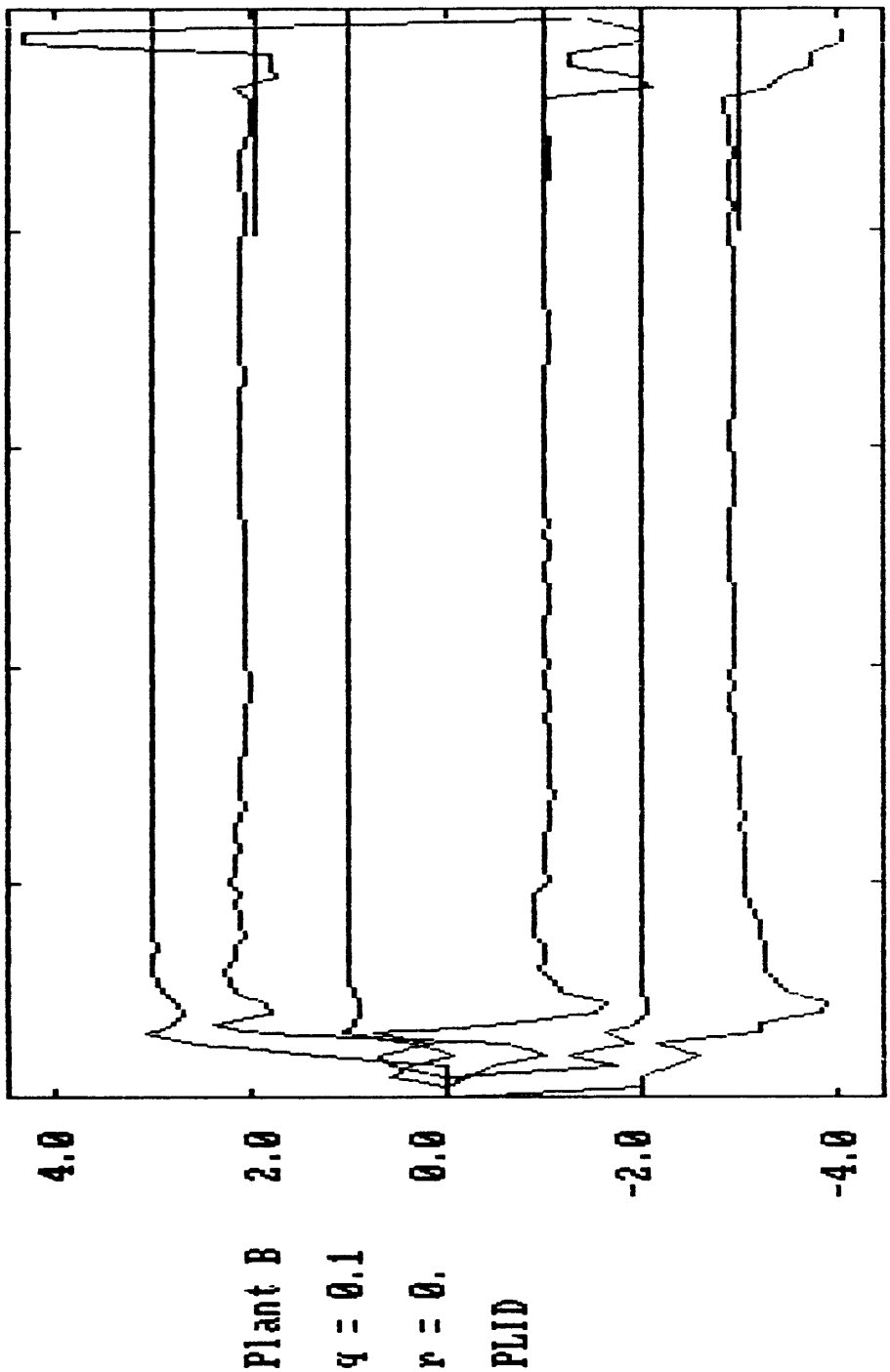

Figure 3.12 PLID Simulation, Plant B, Case 5

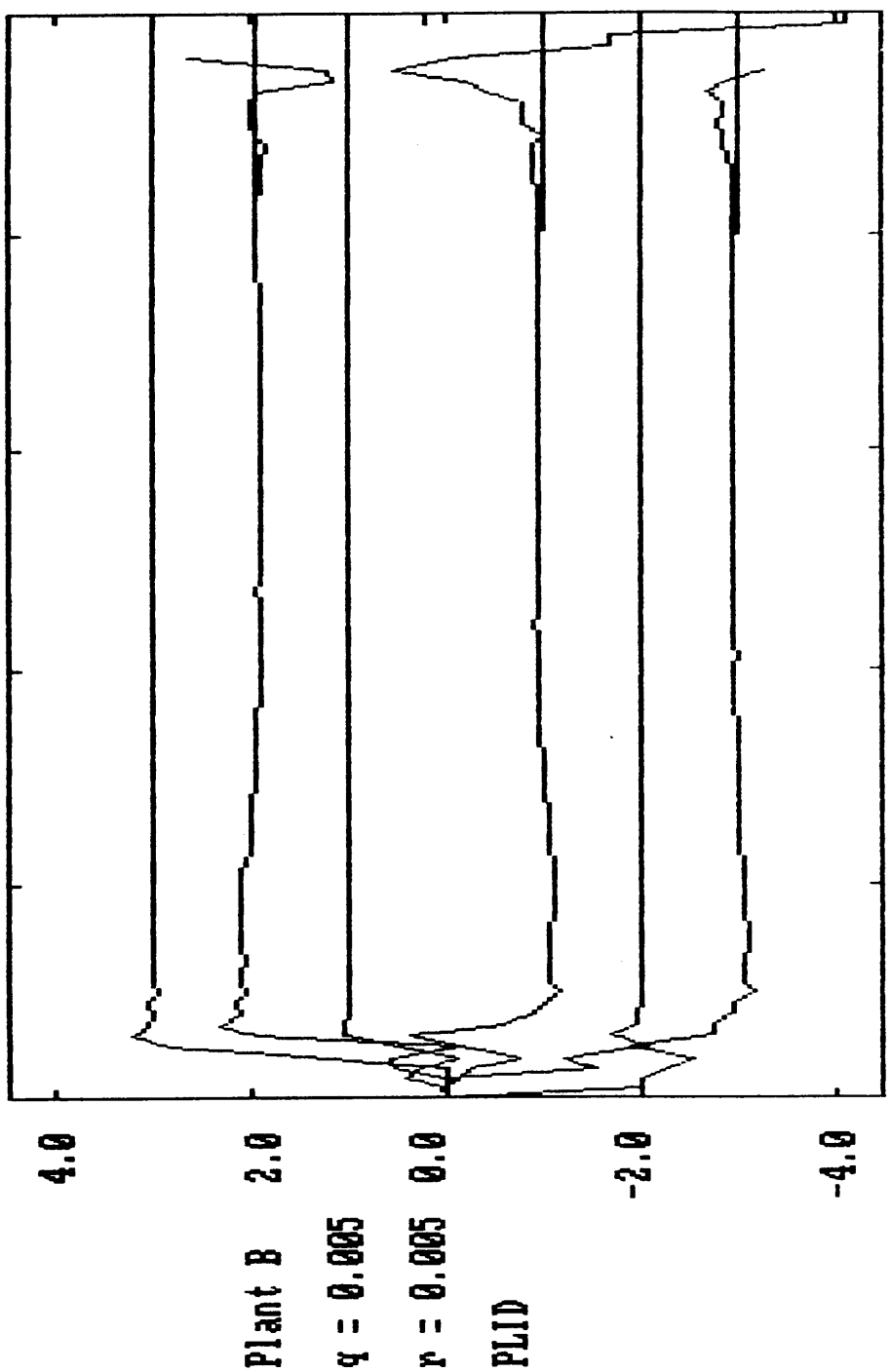

Figure 3.13 PLID Simulation, Plant B, Case 6

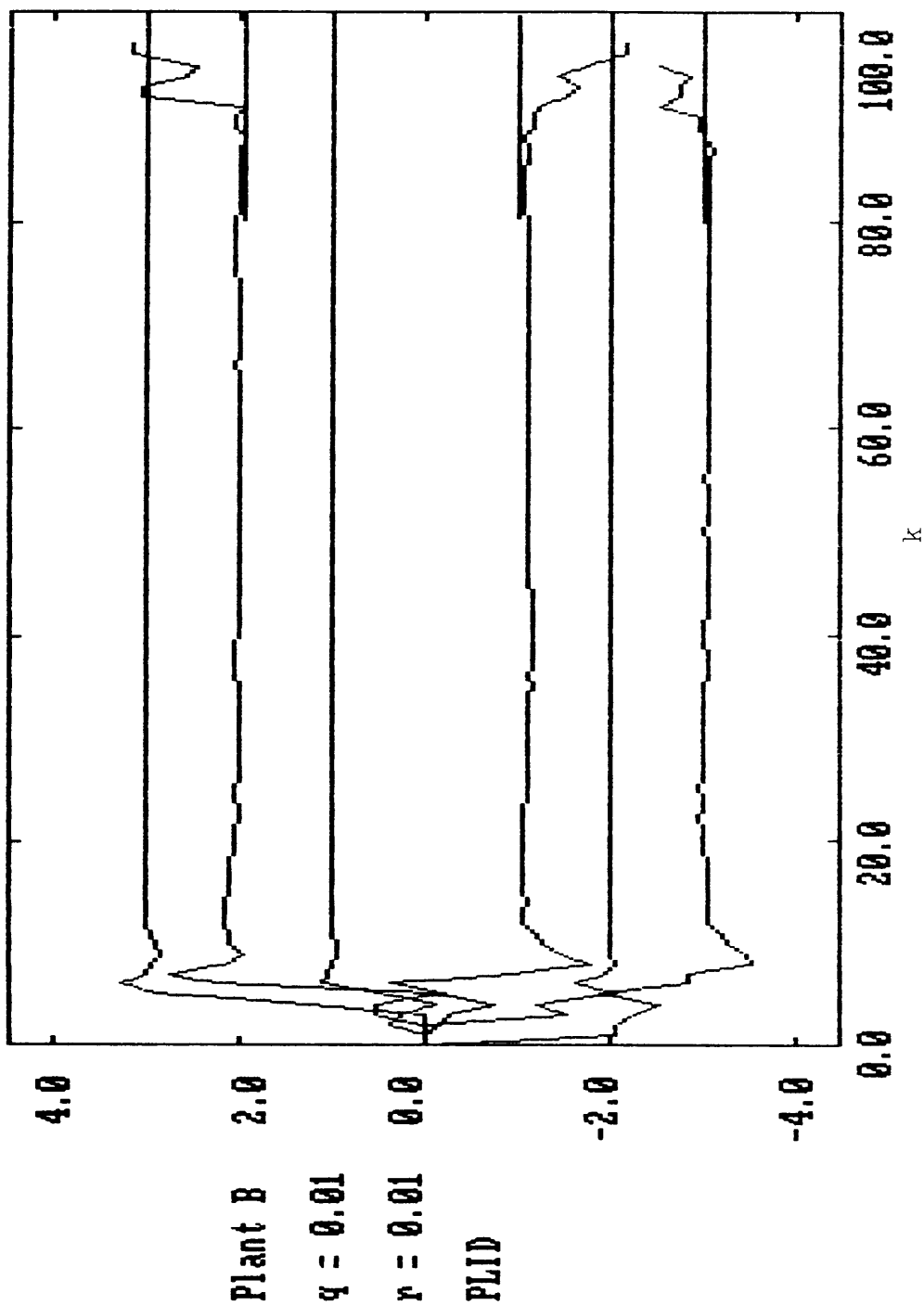

Figure 3.14 PLID Simulation, Plant B, Case 7

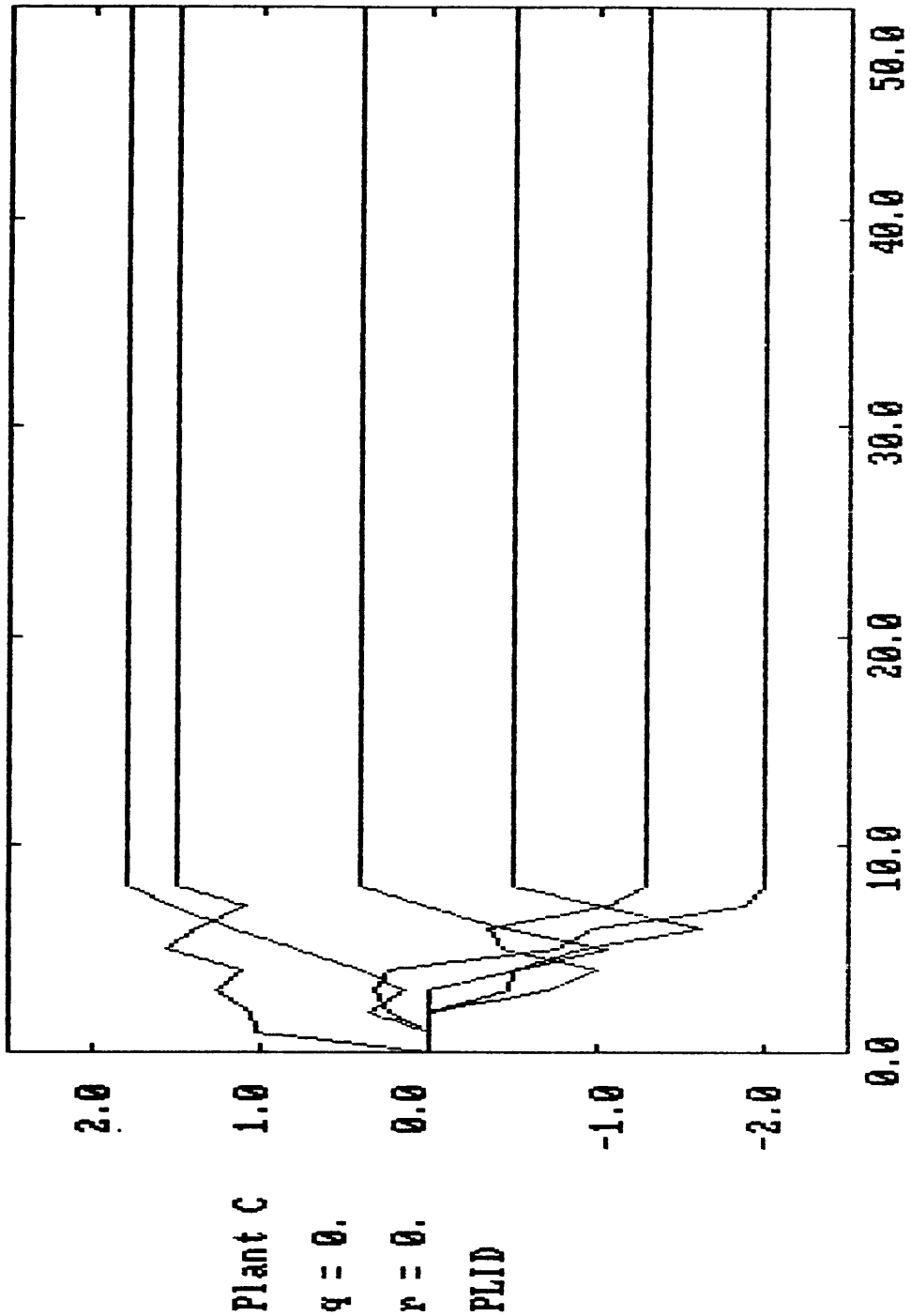

Figure 3.15 PLID Simulation, Plant C, Case 1

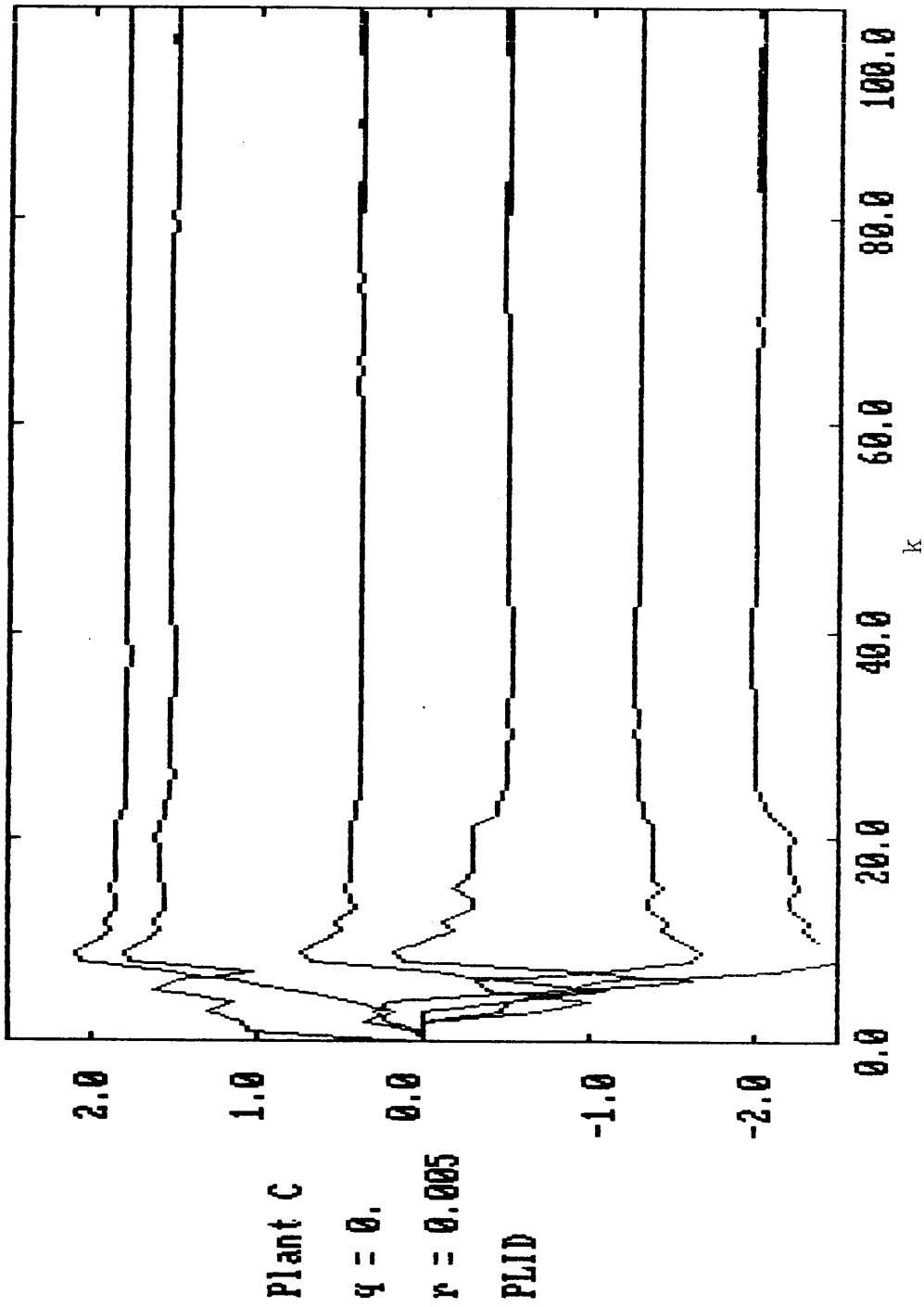

Figure 3.16 PLID Simulation, Plant C, Case 2

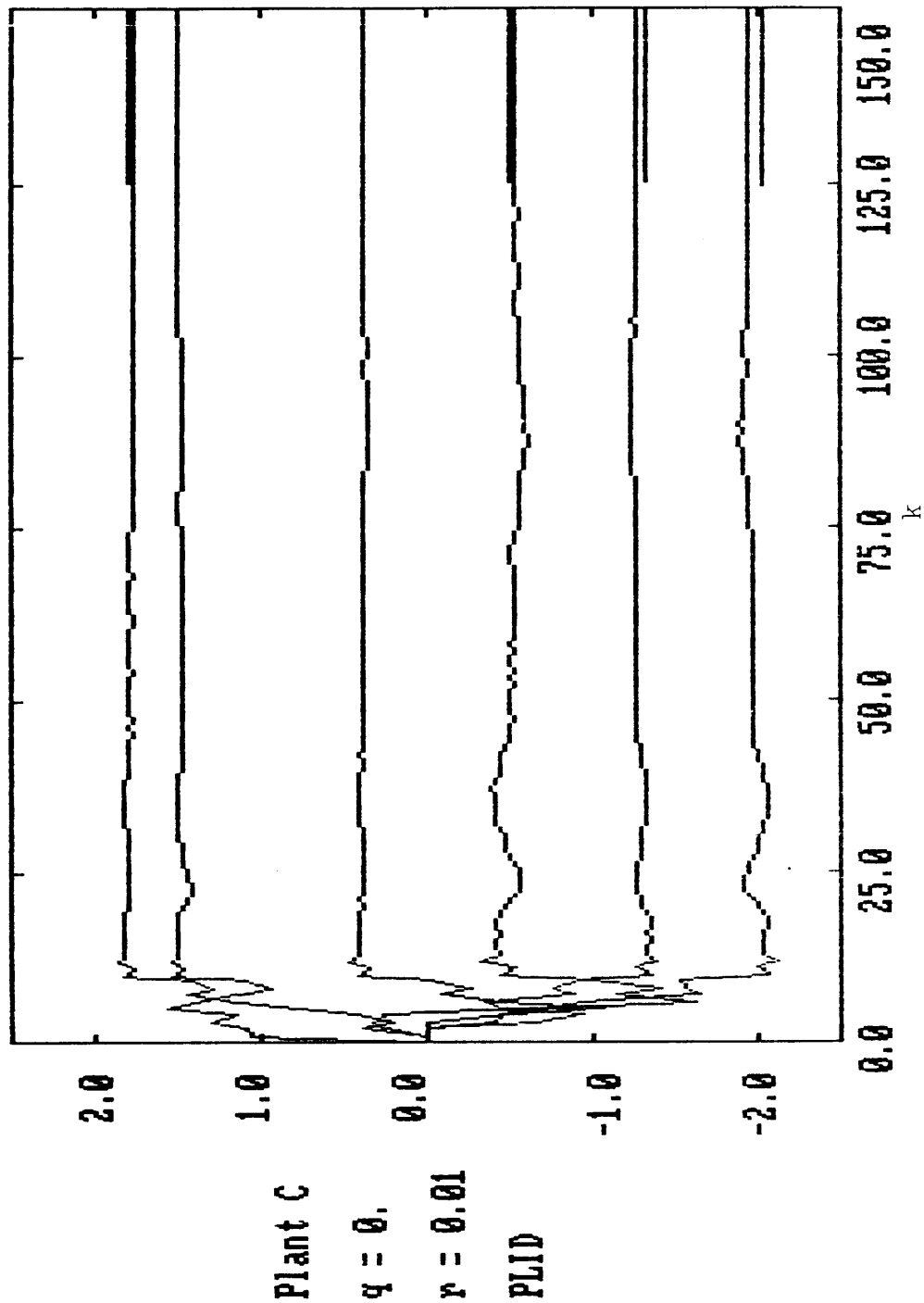

Figure 3.17 PLID Simulation, Plant C, Case 3

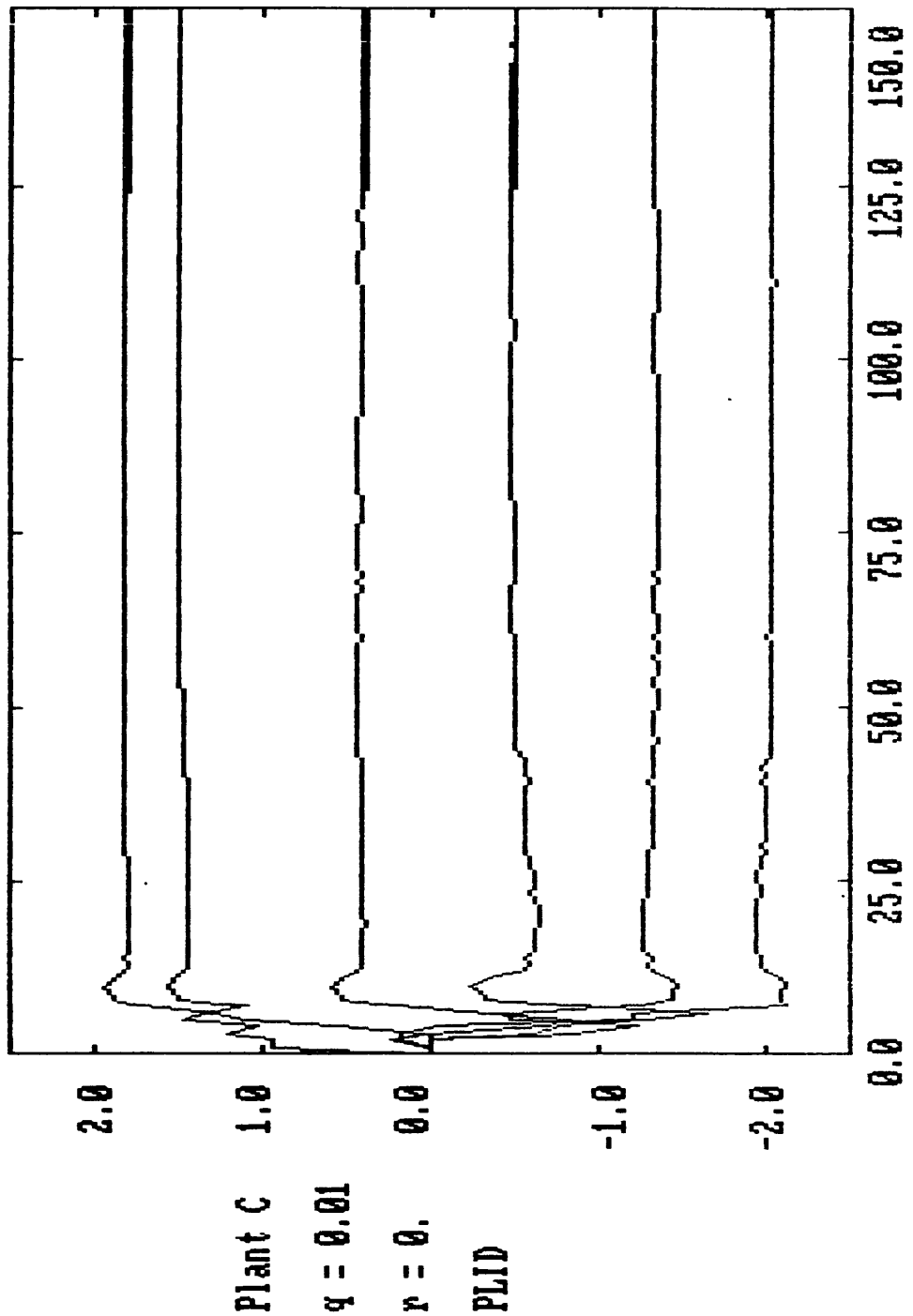

Figure 3.18 PLID Simulation, Plant C, Case 4

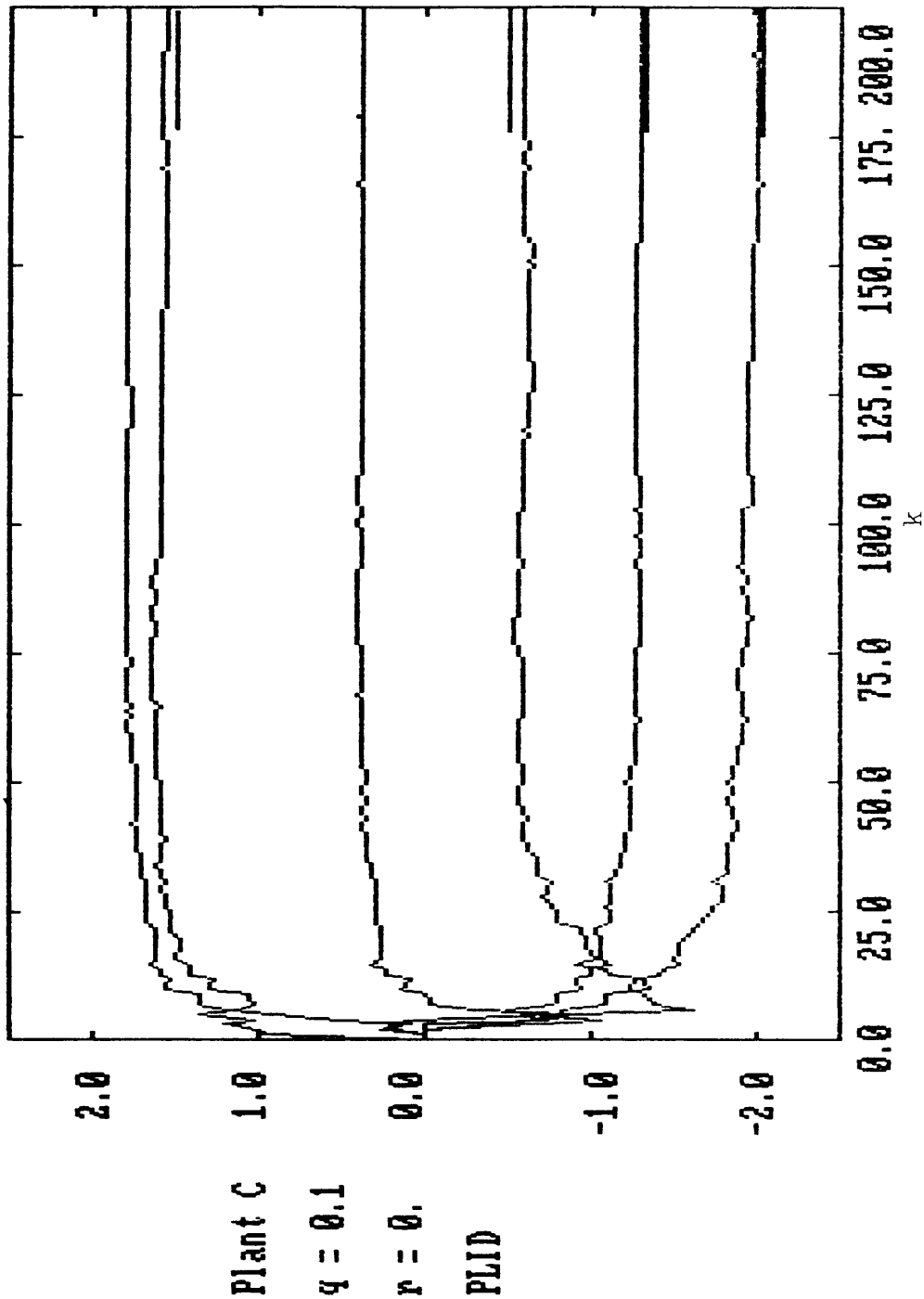

Figure 3.19 PLID Simulation, Plant C, Case 5

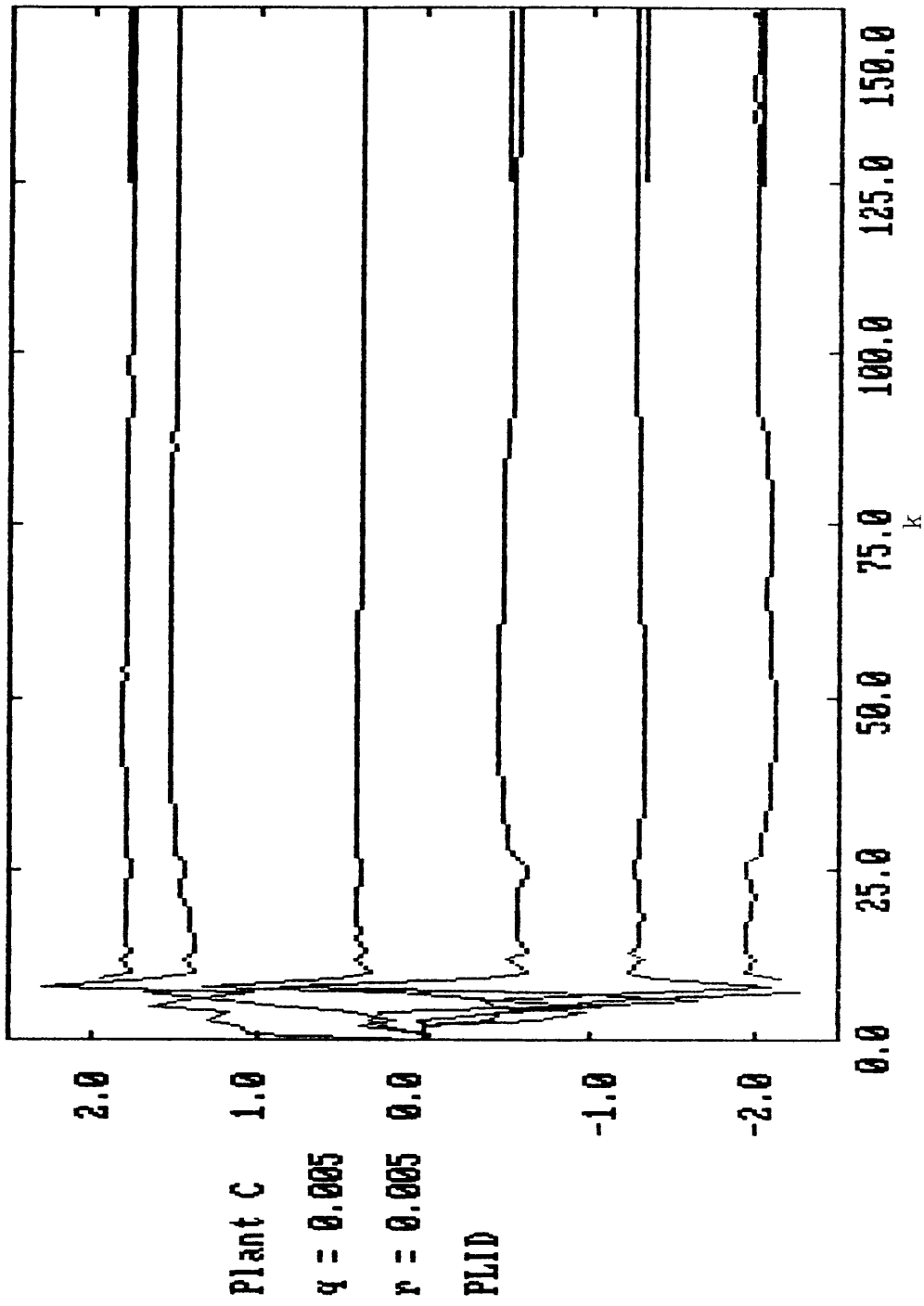

Figure 3.20 PLID Simulation, Plant C, Case 6

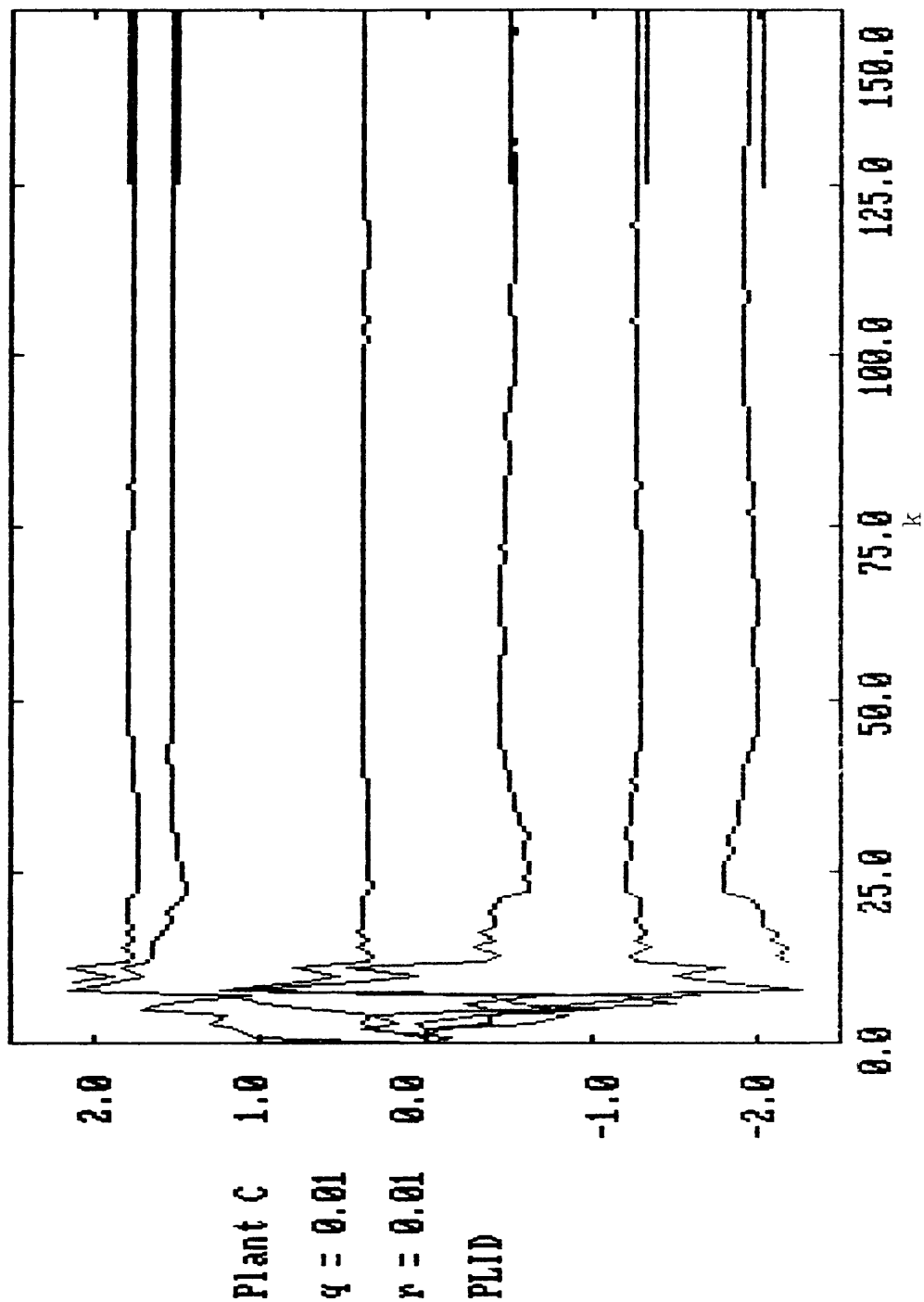

Figure 3.21 PLID Simulation, Plant C, Case 7

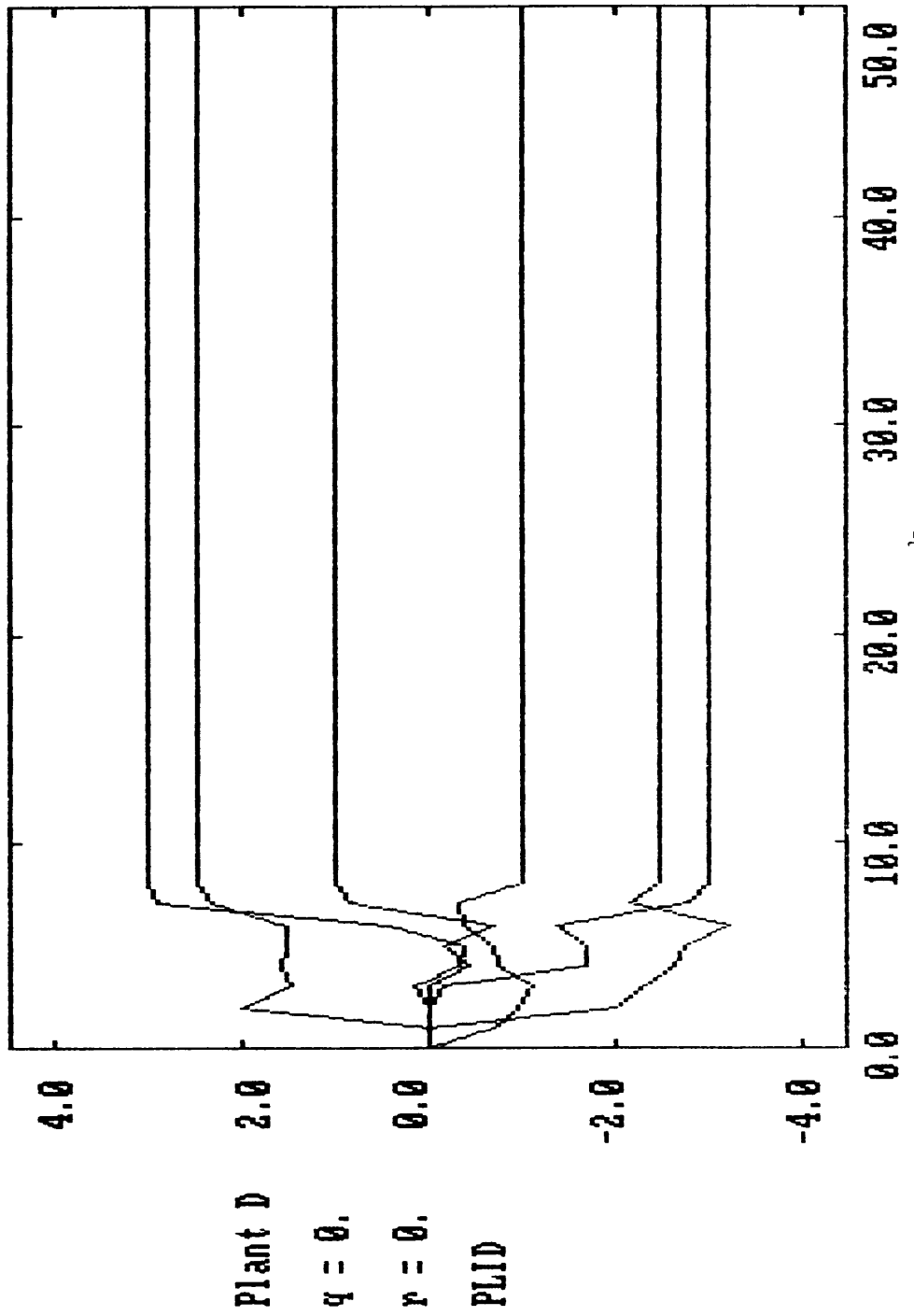

Figure 3.22 PLID Simulation, Plant D, Case 1

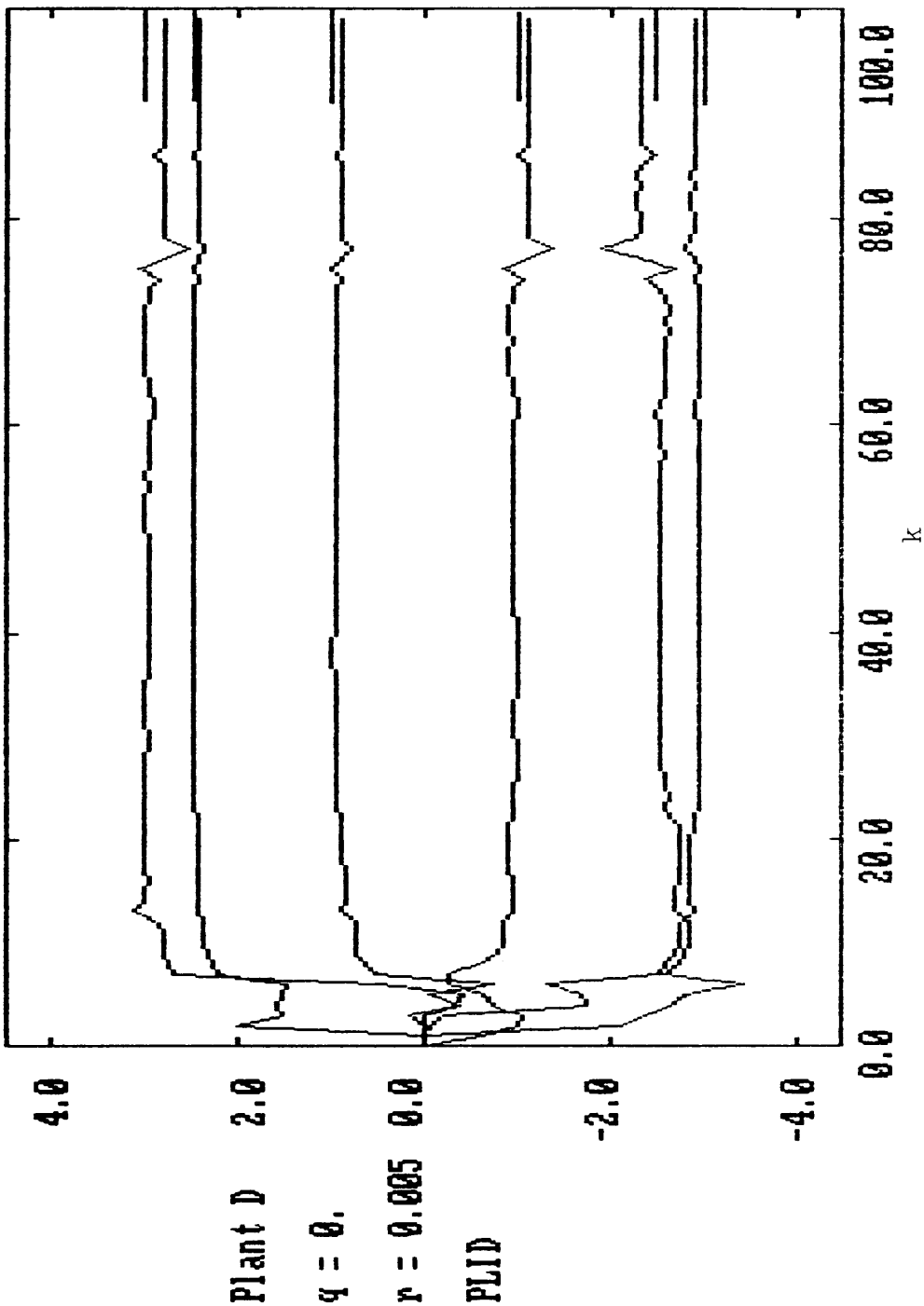

Figure 3.23 PLID Simulation, Plant D, Case 2

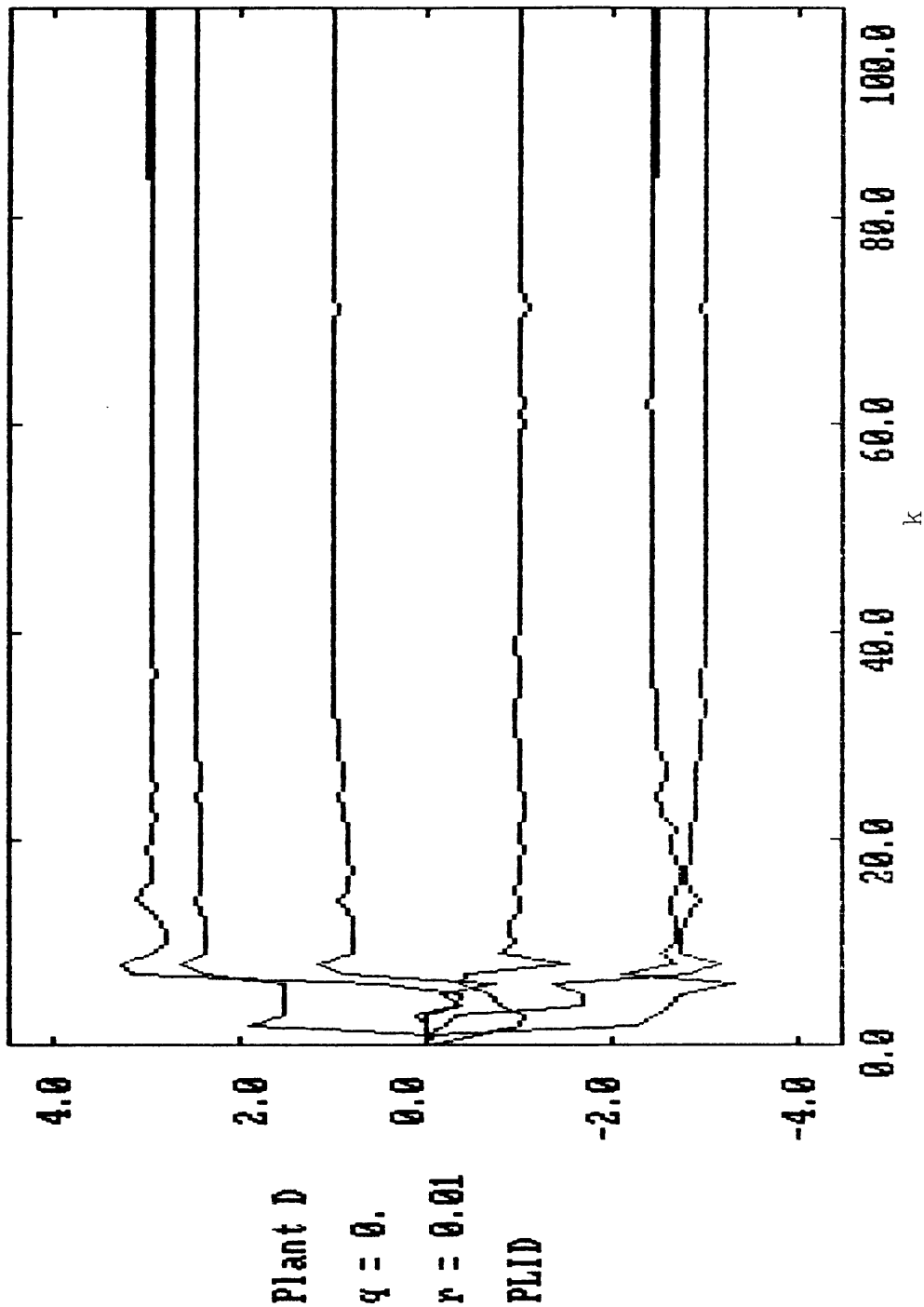

Figure 3.24 PLID Simulation, Plant D, Case 3

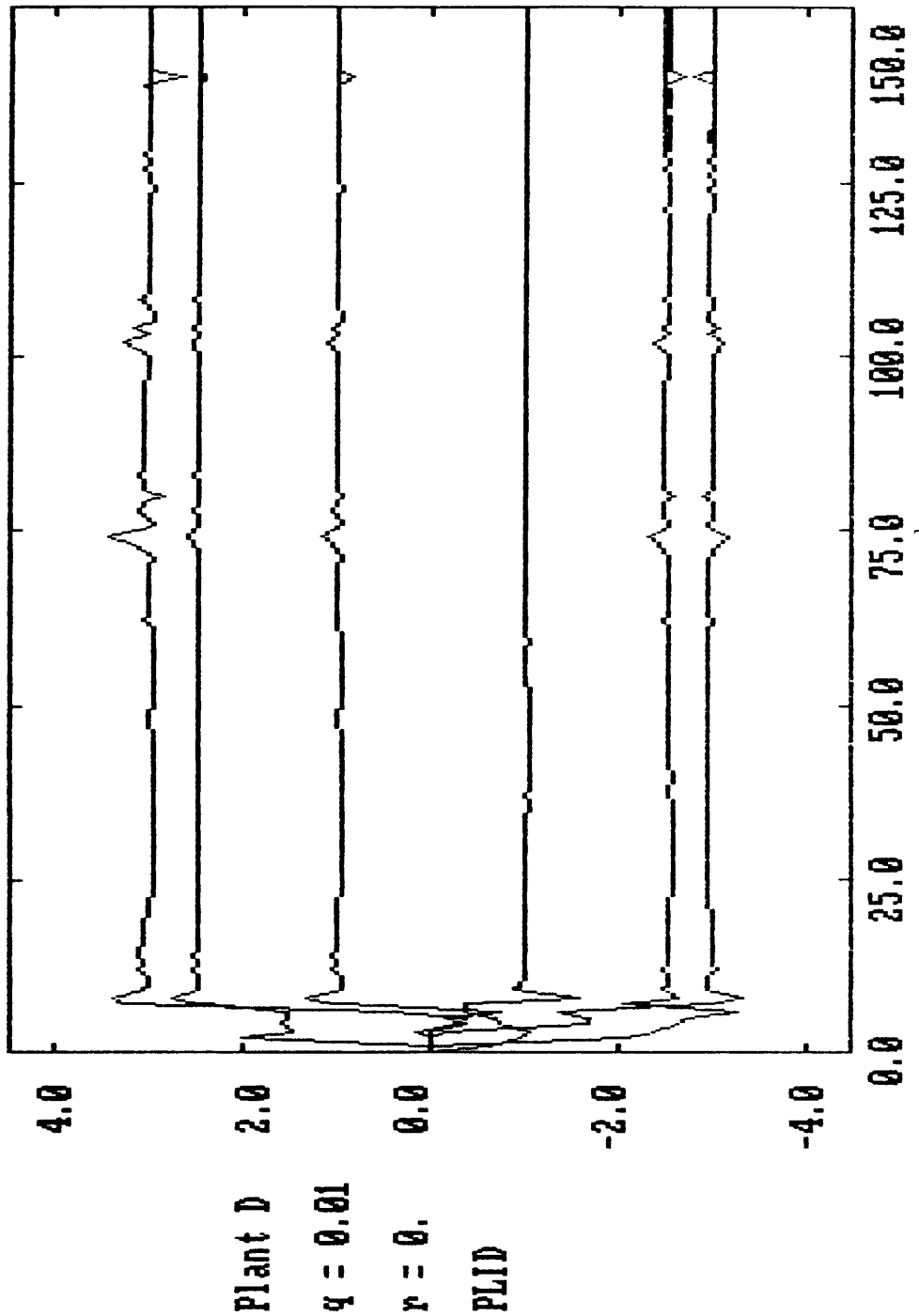

Figure 3.25 PLID Simulation, Plant D, Case 4

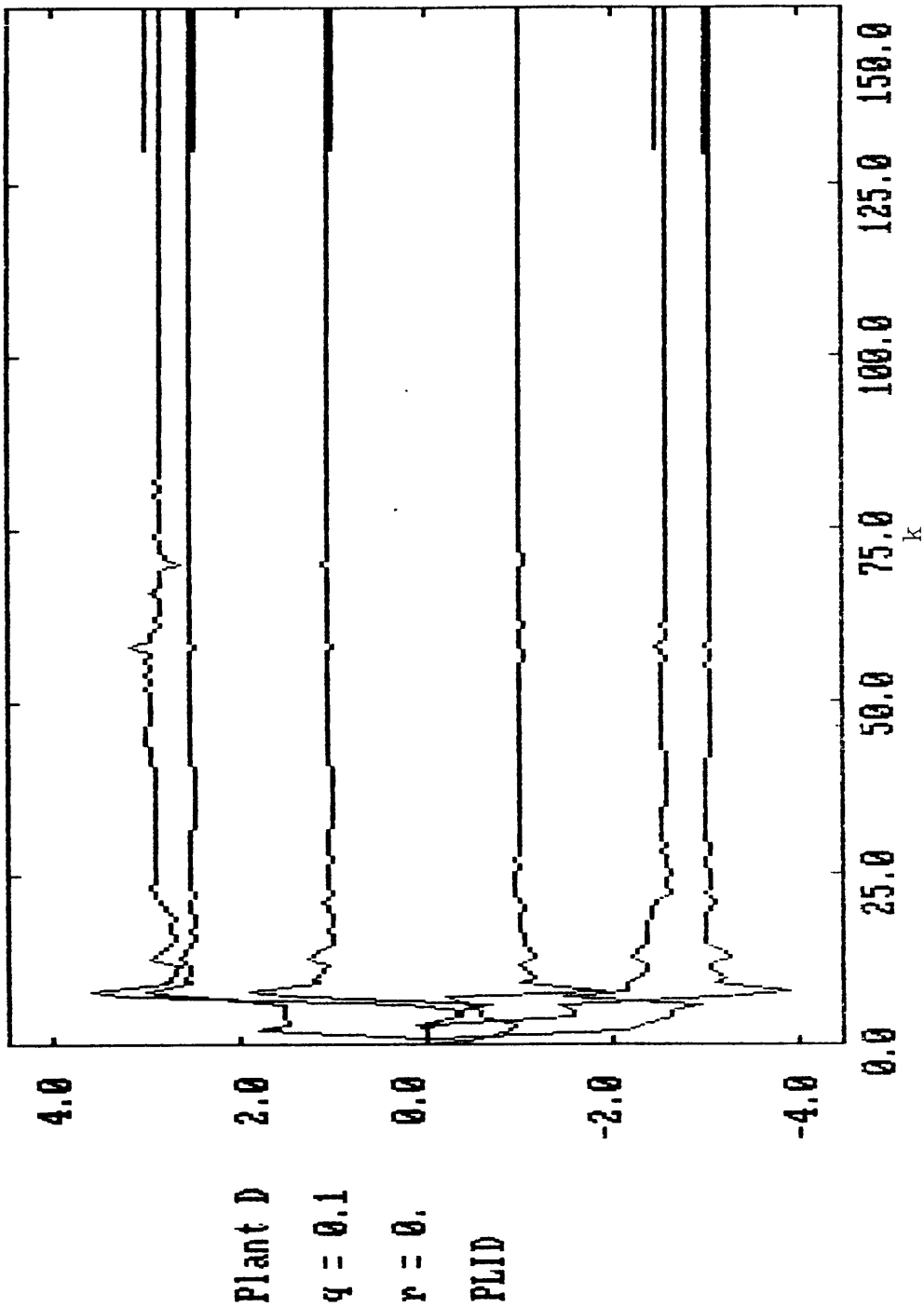

Figure 3.26 PLID Simulation, Plant D, Case 5

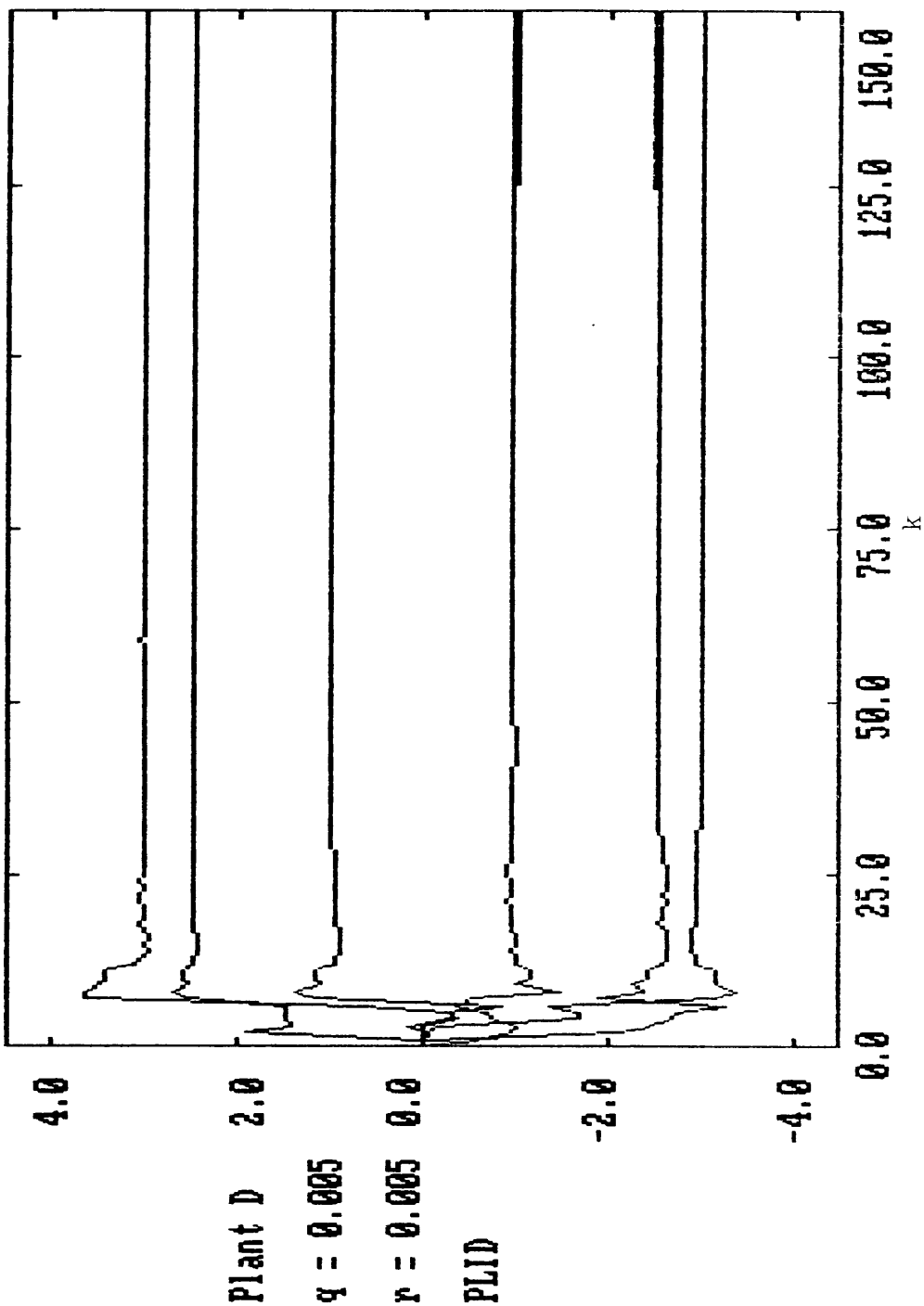

Figure 3.27 PLID Simulation, Plant D, Case 6

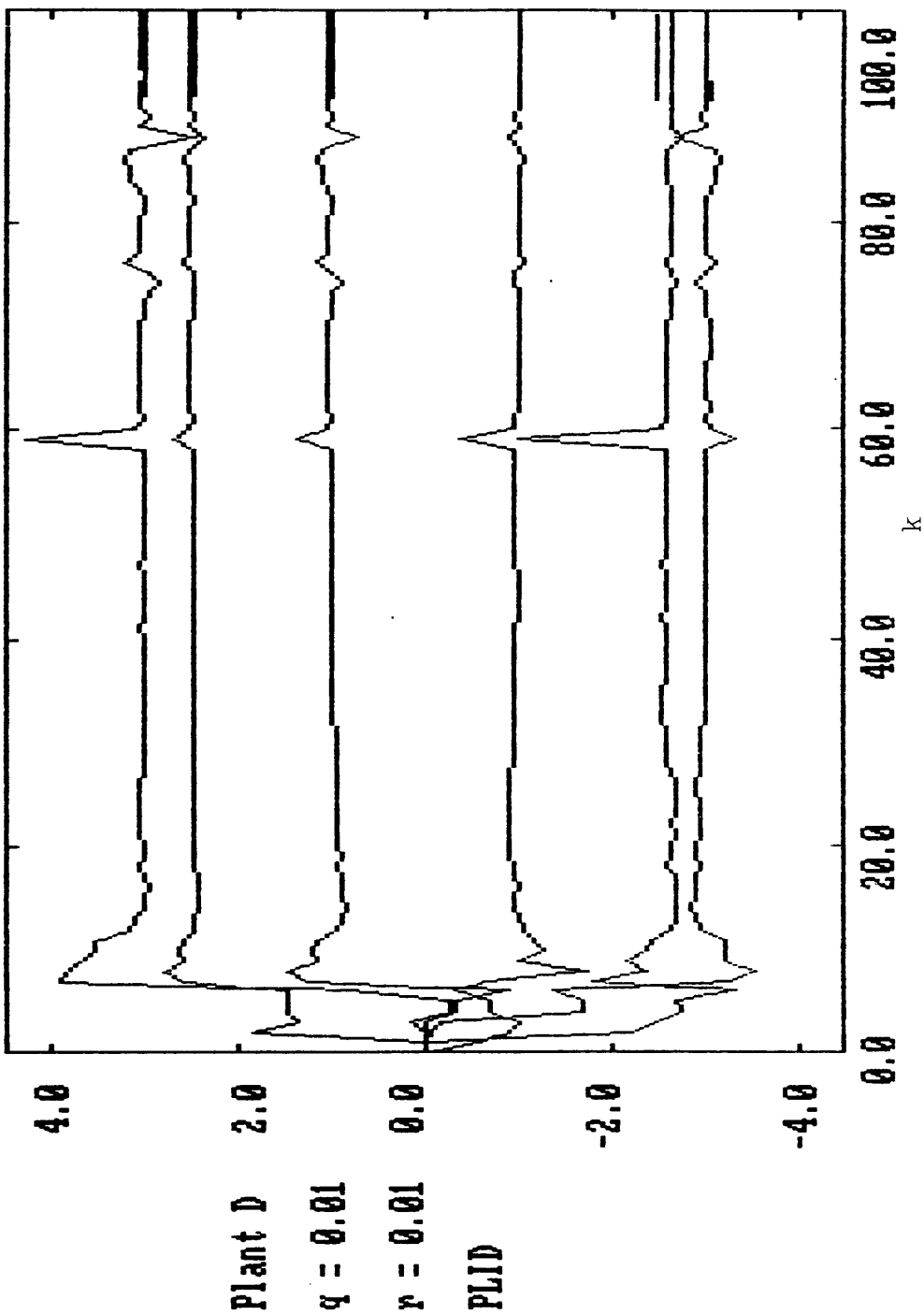

Figure 3.28 PLID Simulation, Plant D, Case 7

completely describe the test situation by listing the plant designation with q and r values. The nominal true values of the parameters are, for reference, drawn in as horizontal lines. The performance of the PLID algorithm is thoroughly described by this set of plots and a variety of conclusions may be drawn from careful study of the traces. We will review the results for Plant A in detail and then highlight those examples for Plants B, C, and D which yield additional insight.

The deterministic case for Plant A, as seen in Figure 3.1, demonstrates the deadbeat response described in Section 2.5. The white noise input u(k) meets our persistent excitation requirements and we see convergence exactly on the eighth iteration as expected. Counting the initial estimate (k=0), k=8 corresponds to the 3n=9-th estimate of the parameters. Figures 3.2 and 3.3 illustrate the steady state error or bias introduced by measurement noise. The bias clearly increases with increasing measurement noise but is within what might be considered acceptable bounds even in Figure 3.3 where the measurement noise power is 1% of the input noise power. Final convergence is seen to be delayed beyond the k=8 step, however the estimates have completed any large amplitude changes within this time frame.

The process noise cases of Figures 3.4 and 3.5 are a different matter entirely. Even when the process noise is a full 10% of the input power there is no bias in the parameters! While convergence is even further slowed, we see again that the "gross acquisition" or high amplitude change portion of the identification is relatively short. The

relative speed of this convergence will be further quantified when comparisons with the RML method are made later in this chapter.

The mixed process and measurement noise cases of Figures 3.6 and 3.7 further reinforce the conclusions made in the separate noise cases. No more bias is introduced than in the corresponding pure measurement noise cases and the convergence rate is slowed by the presence of the process noise. The ability of the algorithm to cope with high process noise is inherent in our modeling assumptions presented in Section 2.3. Such noise is explicitly and correctly included in our formation of the extended-state Kalman filter without approximation. Our formulation does not, however, completely consider the correlated noise effects of noisy measurements $y(k)$ within $F(k)$ which results in the demonstrated bias in these cases.

Figures 3.8-3.28 extend the capable range of PLID to include plants of any non-cancelling pole-zero configuration about the unit circle. Non-minimum phase and unstable plants are easily treated by PLID and the conclusions drawn from the "normal" Plant A results are seen to extend directly. The reader is cautioned against misinterpreting Figures 3.9-3.14 as reflecting an inherent instability in the algorithm when dealing with an unstable plant. The divergence seen at the end of the Plant B simulations is caused by unavoidable roundoff errors due to the sheer size of the output and state estimates. By the 90-th sample the output, and hence the state estimates, are wildly oscillating between values exceeding 10^{20} and -10^{20}. There is thus a wide range separating these values from those of the parameter portion of the extended state.

This and the presence of the large y(k)'s in F(k) lead to numerical ill-conditioning of the entire recursion. Such a situation would not arise during on-line identification where the plant would be regulated within a closed loop to prevent such large output excursions. Numerical considerations for the algorithm itself will be addressed in Chapter 7.

The results for Plant D, where the unstable output values did not go much above 10^{15}, show no divergence even out to 150 samples. Here the results are consistent with the conclusions already drawn, and convergence may be even faster. This is due to the small relative influence of the noises compared to the large output and state values. The similar results for the stable non-minimum phase Plant C will become quite important when the aircraft adaptive control problem is treated in Chapter 6. By examining all these combinations of pole-zero locations relative to the unit circle we have clearly demonstrated the ability of PLID to identify any observable, prime (no pole-zero cancellation) state model with a near deadbeat convergence rate in most cases.

Upon careful reflection, the measurement noise bias situation does not seem to limit our extension of PLID to adaptive control applications. In a closed-loop control application, one should be able to expect sensor precision well within the 1% range established above as a bound within which the PLID bias is acceptable. Process model error and noise disturbances arising from nonlinearities, reduced model-order assumptions, and physical random disturbances of the type which adaptive control is designed to overcome do not bias the PLID results. Thus,

PLID, when provided with reasonably reliable measurements is a strong candidate for the type of fast identification algorithm necessary for adaptive control applications.

3.4 The Recursive Maximum Likelihood Method

The choice of observable-canonical state models to represent our identified systems is certainly not unique. Dynamic models take on many forms in the literature and different choices of dynamic models often motivate different algorithms for their identification [7]. A popular model for representing dynamic models which includes the potential for colored measurement noise is the so called ARMAX model [7].

$$y(k) + a_1 y(k-1) + \ldots + a_n y(k-n) = b_1 u(k-1) + \ldots$$
$$+ b_m u(k-m) + w(k) + c_1 w(k-1) + \ldots + c_r w(k-r) \qquad (3.6)$$

or introducing the backward shift operator q^{-1},

$$q^{-1}\, y(k) \triangleq y(k-1) \qquad (3.7)$$

we have

$$A(q^{-1})y(k) = B(q^{-1})u(k) + C(q^{-1})w(k) \qquad (3.8)$$

where $$A(q^{-1}) = 1 + a_1 q^{-1} + \ldots + a_n q^{-n}$$

$$B(q^{-1}) = b_1 q^{-1} + b_2 q^{-2} + \ldots + b_m q^{-m}$$

$$C(q^{-1}) = 1 + c_1 q^{-1} + \ldots + c_r q^{-r}$$

and u(k) and y(k) are the input and output respectively and w(k) is again a zero-mean white-noise sequence. Fortunately there is a direct relationship between the ARMAX model and the observable-canonical form for the case when n=m=r in Equation (3.6)

$$\underline{v}(k+1) = \begin{bmatrix} 0 & \ldots & 0 & -a_n \\ 1 & \ldots & 0 & -a_{n-1} \\ \vdots & & \vdots & \vdots \\ 0 & \ldots & 1 & -a_1 \end{bmatrix} \underline{v}(k) + \begin{bmatrix} b_n \\ \vdots \\ b_1 \end{bmatrix} u(k) + \begin{bmatrix} c_n - a_n \\ \vdots \\ c_2 - a_2 \\ c_1 - a_1 \end{bmatrix} w(k) \qquad (3.9)$$

$$y(k) = (0 \ldots\ldots 0 \ 1)\ \underline{v}(k) + w(k)$$

There is thus a one-to-one correspondence between the (a_i), (b_i) parameters of the ARMAX model and those of the observable-canonical form of order n from Chapter 2. Namely

ARMAX	corresponds to	OBSERVABLE-CANONICAL
$-a_n$		a_o
$\vdots$		$\vdots$
$-a_2$		a_{n-2}
$-a_1$		a_{n-1}
b_n		b_o
$\vdots$		$\vdots$
b_1		b_{n-1}

(3.10)

The (c_i) parameters of (3.6) have no analog in the observable-canonical form since we have not treated the colored noise case there. The modeling of the noise structure is the inherent difference between the ARMAX model and the observable-canonical form presented in equation (2.12). The reason for considering such a radically different noise model is that the identification method we will develop for (3.9) is reputed to have excellent unbiased noise characteristics [14] and will serve as a challenging yardstick to PLID.

If we define the parameter vector

$$\underline{\theta} = [a_1 \ldots a_n \; b_1 \ldots b_n \; c_1 \ldots c_n]^T \tag{3.11}$$

and the observation vector

$$\underline{x}(k) = [-y(k-1) \ldots -y(k-n) \; u(k-1) \ldots u(k-n) \; w(k-1) \ldots w(k-n)]^T \tag{3.12}$$

then (3.8) can be rewritten as

$$y(k) = \underline{x}(k)^T \underline{\theta} + w(k) \tag{3.13}$$

from which Goodwin and Payne [5] develop what they refer to as the Approximate Maximum Likelihood method. Ljung and Söderström [7] refer

to this method as the Recursive Prediction Error method and Goodwin and Sin [2] the RML2 method. In particular, the method applies the recursion

$$P(k+1) = P(k)\left[I - \frac{\phi(k+1)\phi^T(k+1)P(k)}{(1+\phi^T(k+1)P(k)\phi(k+1)}\right] \tag{3.14a}$$

$$K(k+1) = P(k)\phi(k+1) \,/\, [1 + \phi^T(k+1)P(k)\phi(k+1)] \tag{3.14b}$$

$$\underline{\hat{\theta}}(k+1) = \underline{\hat{\theta}}(k) + K(k+1)\,\hat{w}\,(k+1) \tag{3.14c}$$

where

$$\hat{w}(k) \simeq \hat{\varepsilon}(k) = y(k) - \underline{\hat{x}}^T(k)\underline{\hat{\theta}}(k-1) \tag{3.14d}$$

$$\underline{\hat{x}}^T(k) = [-y(k-1) \ \ldots \ -y(k-n)\,u(k-1)\ldots u(k-n)\hat{\varepsilon}(k-1)\ldots \hat{\varepsilon}(k-n)] \tag{3.14e}$$

and

$$\phi(k+1) = \begin{bmatrix} -\hat{c}_1 & \cdots & -\hat{c}_r & & 0 & & 0 \\ 1 & \cdots & 0 & 0 & & & \\ \vdots & & & \vdots & & & \\ 0 & \cdots & 1 & 0 & & & \\ & & & & & & \\ & & -\hat{c}_1 & \cdots & -\hat{c}_r & & \\ 0 & & 1 & \cdots & 0 & 0 & 0 \\ & & \vdots & & & \vdots & \\ & & 0 & \cdots & 1 & 0 & \\ & & & & & & \\ & & & & -\hat{c}_1 & \cdots & -\hat{c}_r \\ 0 & & 0 & & 1 & \cdots & 0 \quad 0 \\ & & & & \vdots & & \vdots \\ & & & & 0 & & 1 \quad 0 \end{bmatrix} \phi(k) + \begin{bmatrix} -y(k) \\ 0 \\ \vdots \\ 0 \\ u(k) \\ 0 \\ \vdots \\ 0 \\ -e(k) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{3.14f}$$

Initialization of this method, which we will refer to as RML by combining the above names, is in a similar vein as PLID in that P(0) is taken as a diagonal matrix with large diagonal entries and $\underline{\hat{\theta}}(o) = 0$. We also apply a suggestion from Ljung and Söderström [7] that once $\underline{\hat{\theta}}$ has been updated in (3.14c), this new information be taken into account to refine the $\hat{\varepsilon}(k)$ estimate which will be used in the successive step. This modification to Equations (3.14) is clearly denoted as "epsilon refinement" in the Subroutine RML coding in the Software Appendix.

The choice of this method as our yardstick is a compromise between several competing goals. We know that PLID is biased in the white measurement noise case yet unbiased under process noise. The simple and classical Recursive Least Squares method is known to be sensitive (biased) to any noise more complex than white pure measurement noise under which it is unbiased [7]. The extension to the colored noise model for RML is an attempt to investigate a method which has enough degrees of freedom to possibly compensate for the presence of process noise as well. In addition, RML has the same degree, 3n, as PLID which implies a measure of fairness in comparing the rate of convergence which would not be possible in the order 2n RLS arrangement.

3.5 RML Test Case Results and Comparative Conclusions

The RML subroutine links with the main IDENT program in exactly the same manner as PLID to allow easy comparisons of performance. To further ease comparison of the result traces, the sign discrepancy of the (a_i) parameters in (3.10) is rectified on output to the plotting

files. The same twenty-eight test cases were also run with the RML method to examine its performance in the various noise environments. Our purpose, however, is not to fully quantify RML performance; therefore, only selected cases which highlight relative differences with PLID will be presented. The initial conditions used were:

$$P(o) = \text{diag}\ (200, 200 \ldots. 200, 10, 10, 10)$$
$$\underline{\hat{\theta}}\ (o) = \underline{0} \qquad (3.15)$$

The lowering of the covariances associated with the (c_i) parameters helps to maintain the $C(q^{-1})$ polynomial within the necessary stability region without the need for a complicated stability region projection algorithm [2]. This modification was not seen to slow the algorithm or affect the results in any way other than stability assurance.

Figure 3.29 illustrates the common complaint with the RML method from an applications standpoint. While the final estimates are indeed unbiased, even in this deterministic case for Plant A convergence is rather slow. This same slow convergence rate is evident in the moderate measurement noise case of Figure 3.30, however, the results are slightly biased. The bias shown in the quickly attained PLID results of Figure 3.4 seems a small price to pay for its speed. Of course, the slight bias result for PLID requires that the measurement noise be similarly slight. The bias seen in Figure 3.30 is not permanent and would be removed eventually by RML. The extreme slowness is due to the close

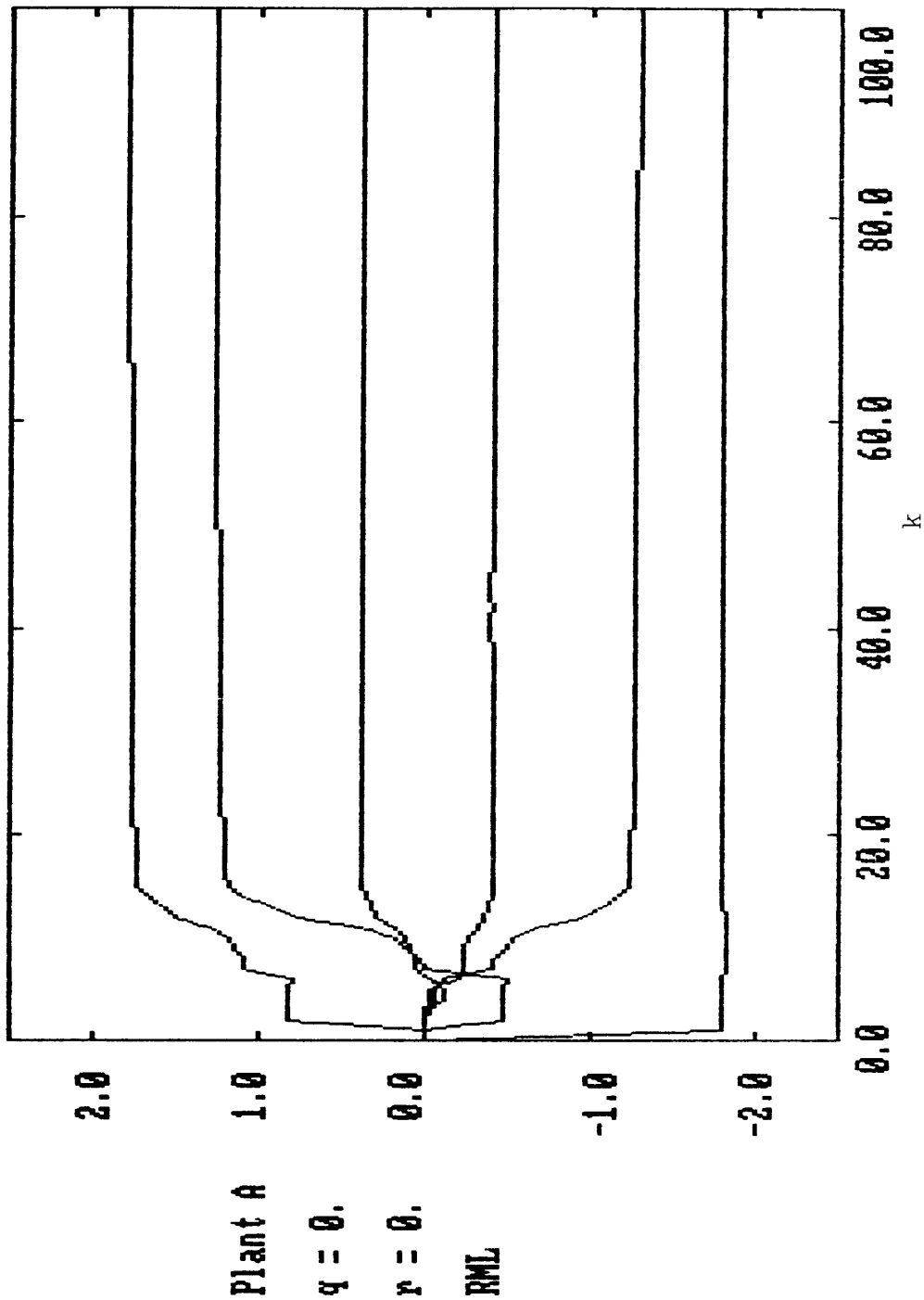

Figure 3.29 RML Simulation, Plant A, Case 1

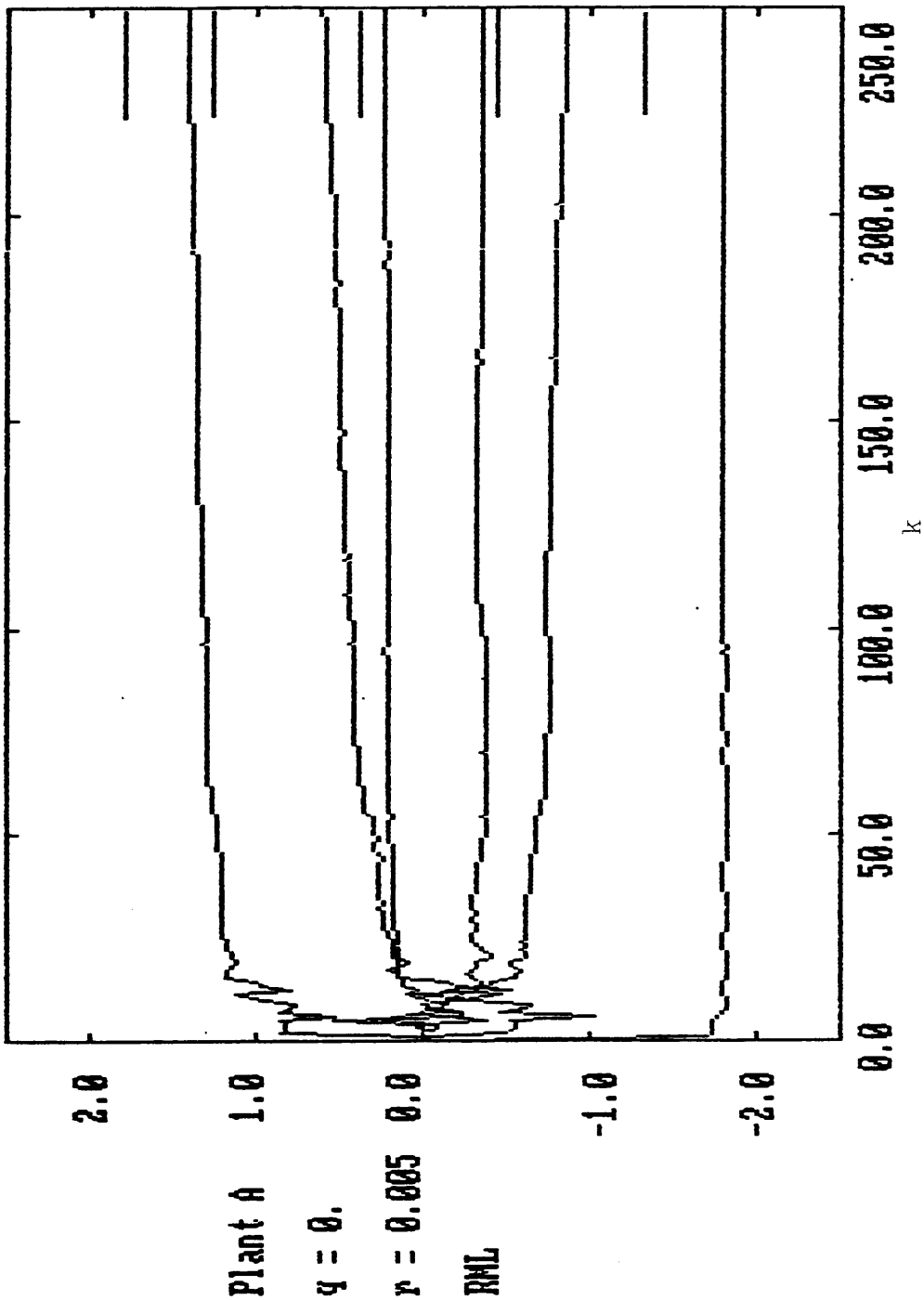

Figure 3.30 RWL Simulation, Plant A, Case 2

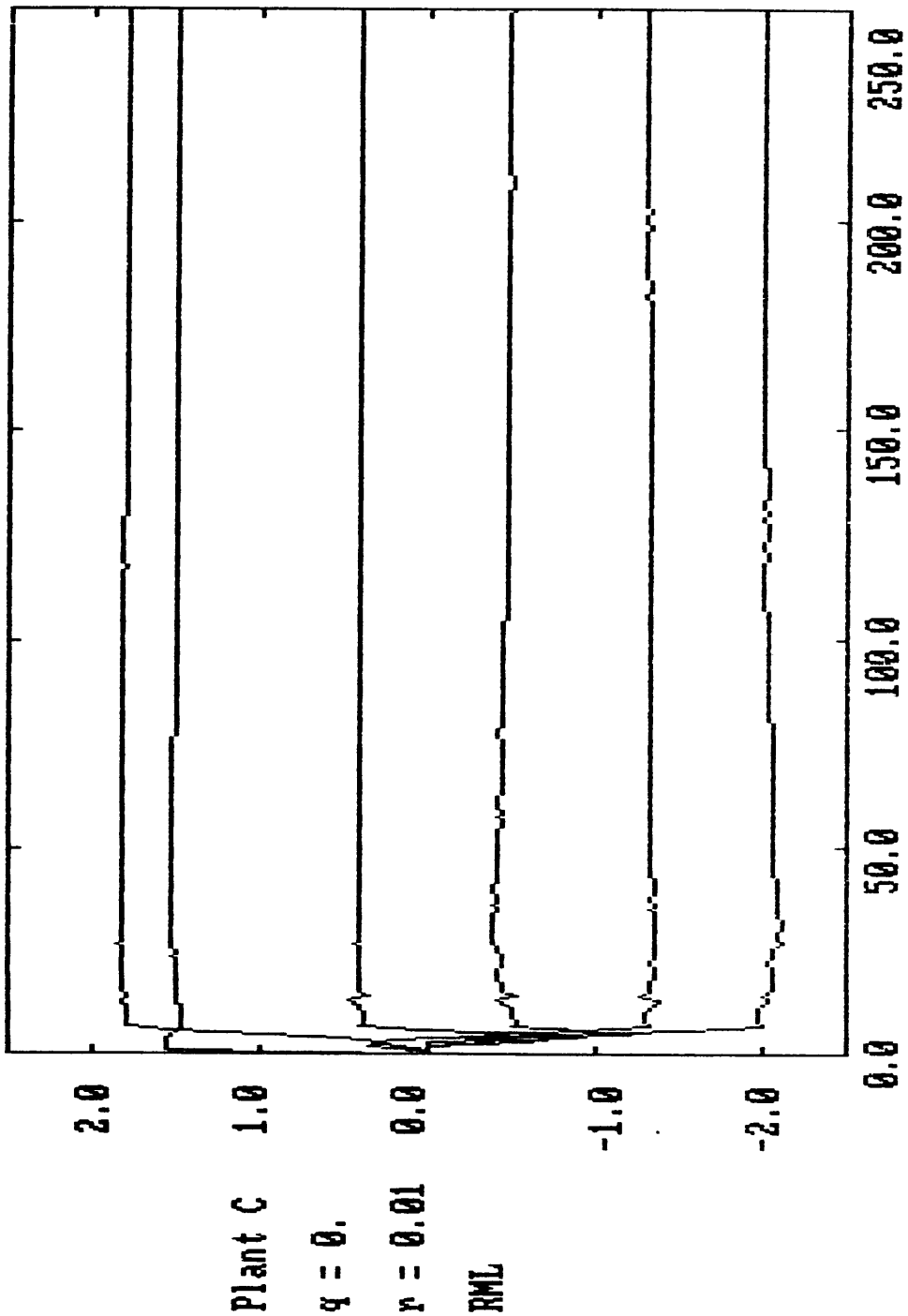

Figure 3.31 RML Simulation, Plant C, Case 3

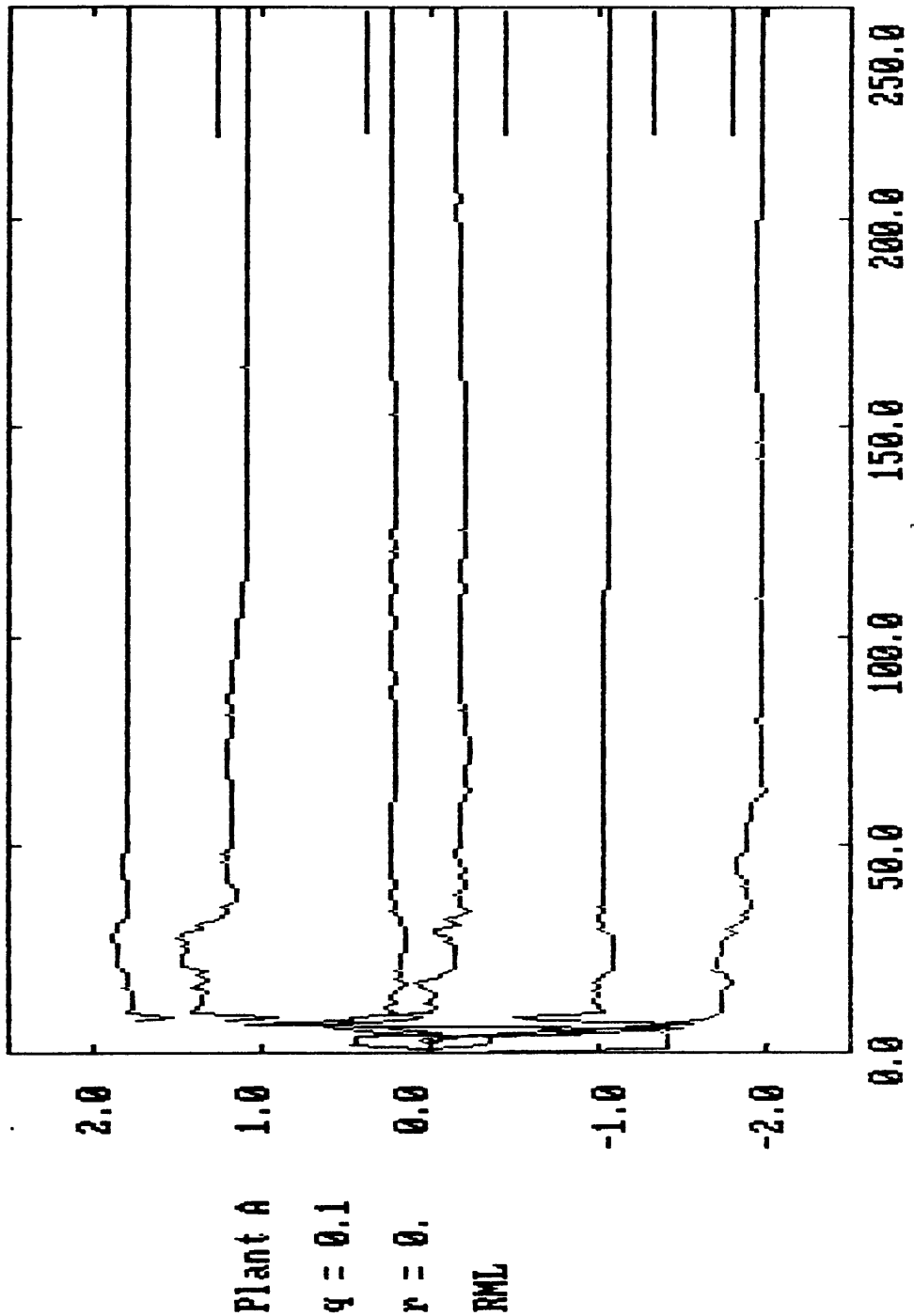

Figure 3.32 RML Simulation, Plant A, Case 5

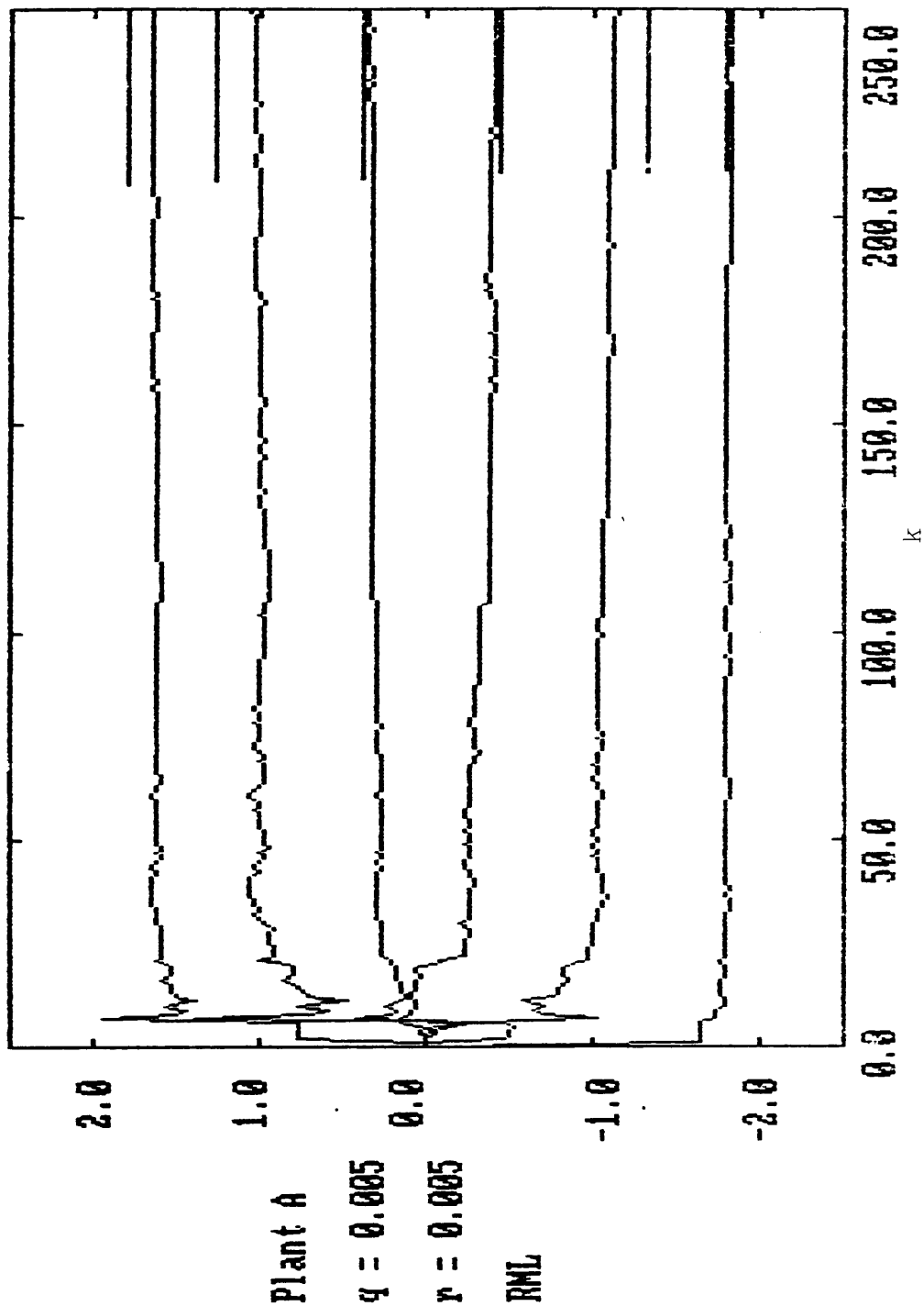

Figure 3.33 RML Simulation, Plant A, Case 6

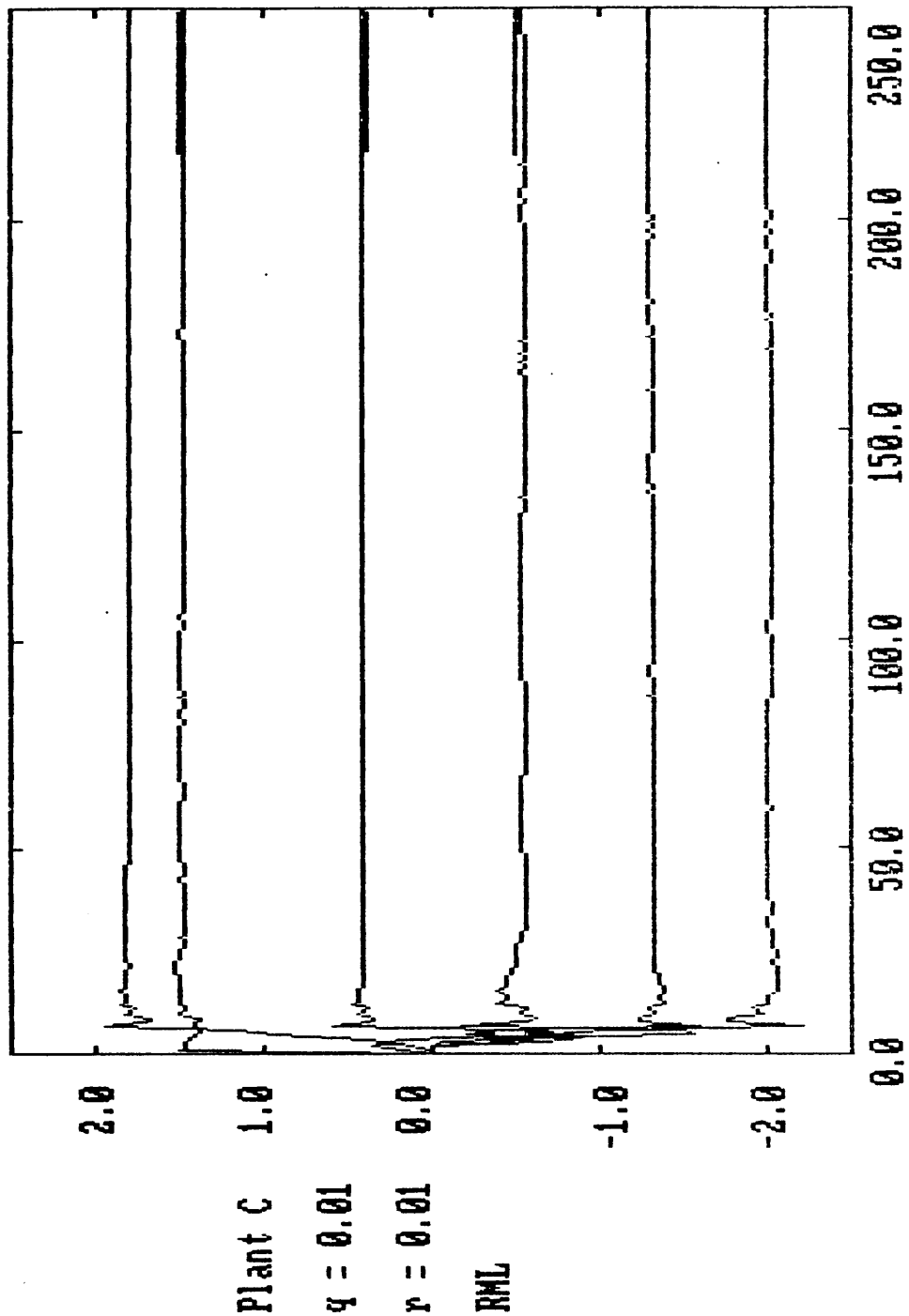

Figure 3.34 RML Simulation, Plant C, Case 7

nature of the poles and zeros of Plant A. In Figure 3.31, Plant C with moderate measurement noise displays no bias, thus verifying the unbiased nature of RML in measurement noise.

In stark contrast with Figure 3.7, which presents unbiased PLID results in high process noise, we see in Figure 3.32 how high process noise leads to a distinct bias even after a slow RML convergence rate. Comparison of Figures 3.33 and 3.34 shows how the combination process and measurement noise cases compare, for Plants A and C with RML. We see that the inclusion of some measurement noise modeling seems to allow RML more flexibility in dealing with the process noise. A check of Equations (3.9) reveals that RML expects any process noise introduction to arise from and be accompanied by correlated measurement noise. Even though our generated process and measurement noises are independent, their appearing together in this instance more closely resembles the framework of the model (3.9) and is more easily compensated for in the RML estimates. It is important to note that the bias is not eliminated in this instance, merely reduced somewhat.

We have thus demonstrated a number of key properties of the PLID method. The near deadbeat response is clearly attractive for the adaptive control application where fast identification is the overriding concern. As long as measurement noises are kept small, the PLID estimates are almost as good as those of RML and certainly converge much faster. Process noise arising from a number of possible sources does not bias PLID results, but clearly does bias RML results and the results of other popular identification methods. We may thus conclude that, in

environments conducive to high precision measurements of input and output data, the PLID method is highly recommendable whenever rapid parameter acquisition and/or process noise rejection are overriding concerns. While the convergence of the parameter estimates to their true values has been the performance aspect examined in this chapter, it should be noted that the state estimates achieve optimal values once parameter convergence occurs. This is because PLID reduces to the basic discrete-time Kalman filter state estimator in the case of perfect parameter estimates, thereby providing the optimal, least mean-square error state estimates in the presence of noise.

4.0 EXTENSION TO PARAMETER TRACKING

While the speed of PLID convergence demonstrated in the previous chapter suggests the applicability of the method for adaptive control, only fixed parameter plants were considered. Unfortunately, the basic PLID algorithm will not always respond quickly or smoothly to changes in the plant parameters once it has converged to the original plant parameters. We may heuristically consider this phenomenon as a complacency in the algorithm once it feels its job has been done. This shutoff or "going to sleep" characteristic of this type of recursive algorithm has been widely documented in the Kalman filtering literature [6], [15] and in the identification literature [2], [7], and [16]. Accordingly, the same sources offer a number of suggestions for overcoming this shortcoming and keeping the algorithms awake and alert to changing parameters.

4.1 Covariance Resetting and Burst Identification

The filter shutoff phenomenon arises from the fact that upon initial convergence the elements of the P matrix approach zero, and consequently the filter gains K driving the estimate update also approach zero. This trend is difficult to reverse even when the new incoming data is entirely inconsistent with the identified model insisted upon by the quiescent algorithm. The simplest correction for the problem is called "covariance resetting" [2] which is equivalent to periodically reinitializing the P matrix diagonal elements to nonzero

values to indicate heightened skepticism in the current estimates. The algorithm is thus restarted using the latest estimate as an initial condition on the extended state estimate. The difficulty with this approach is deciding when to perform the reinitialization; e.g., resetting at fixed predefined intervals or "adaptively" resetting whenever the Kalman gains are deemed too low by some standard. Attempts to employ both of the above resetting approaches to PLID were found ineffective in providing the type of tracking improvement desired.

The resetting approach is only slightly more refined than the concept of "burst" identification where a completely new identification is performed periodically to give parameter estimates only at rather widely separated points in time. Applying burst identification in an adaptive control application clearly assumes that "updating" the controller only at predefined intervals is sufficient to deal with the plant changes which might be encountered. Burst identification and periodic resetting based adaptive control are thus only useful in cases where the plant is assumed to change very slowly. We, on the other hand, seek to extend PLID to effect a continuous track of changing parameters in real time. For example, a sinusoidal variation in a parameter value should lead to a sinusoidal variation in its estimate. In this way, new information about the plant condition can be supplied to the adaptive controller as soon as it is available, not at the next identification "burst" which might be hundreds of samples away.

4.2 Pseudonoise and Fade Factors

The Kalman filter literature suggests methods for tracking which fall into two basic categories: the pseudonoise and fading memory approaches. The pseudonoise approach [7] modifies the covariance time update Equation (2.23a) to include an assumption of white noise on the parameters as

$$M(K) = F(k-1)P(k-1)F^{T}(k-1) + Q'(k-1) \tag{4.1}$$

where

$$Q'(k) = \begin{bmatrix} Q_1(k) & \vdots & 0 \\ \cdots\cdots & \vdots & \cdots\cdots \\ & \vdots & \gamma \quad \gamma \quad 0 \\ 0 & \vdots & 0 \quad \ddots \quad \gamma \end{bmatrix}$$

The γ values may be adjusted to reflect our expectation of how much the parameters are likely to vary. Clearly, this technique boosts the diagonal values of M associated with the parameters and, in fact, insures that they never fall below γ. We have thus assumed that the parameter values are "disturbed" by a white-noise sequence of variance γ. Unfortunately, the parameter variations of physical interest are more likely to be strongly time-correlated, smooth variations, not accurately modeled by white noise.

The concept of a fading memory filter also involves modification of the covariance time update equation (2.23a). With this approach, the

M(k) values are boosted multiplicatively rather than additively as in the pseudonoise case; namely;

$$M(k) = \alpha F(k-1)P(k-1)F^{T}(k-1)+Q(k-1) \tag{4.2}$$

where

$$\alpha \geq 1$$

Gelb [6] points out that this modification has the effect of exponentially weighting the importance of the measurement data to the filter. That is, the effect of old measurements decays exponentially in time causing the filter to put more emphasis on the most recent data. The algorithm is thus prevented from becoming complacent by forcing it to forget the past success and concentrate on the present. This concept seems more in line with the goals of a parameter tracker than the fictitious white noise assumption. Further reasons to favor a fading memory are presented by Maybeck [17] who suggests that pseudonoise may be more appropriate in cases where one is unsure of the filter model F(k) and a fading memory when a linear model such as F(k) is valid, but only for a limited time. If the measurement noise meets our criterion of smallness, there is no reason to question the validity of the filter model itself by assuming psuedonoise. Our only concern is adequately considering time variations in the parameter portion of the extended state. Implementation of the pseudonoise approach has reinforced the conclusion that modeling the parameter variation as white noise is not

valid for our purposes and has directed the study towards the fading memory approach.

4.3 The Block Fading Factor, A Tracking PLID

Direct implementation of the Gelb fade factor as in Equation (4.2) did not prove effective in the case of a time varying plant, and noise sensitivity was heightened. A common sense reconsideration of the goals of parameter tracking suggested a modified Gelb fade factor approach which does achieve the desired result. In the previous chapter the point was made that upon parameter convergence, the state estimate portion of the PLID reduces to the basic Kalman filter. The direct Gelb fade factor, however, "heightens the awareness" of all portions of M to achieve parameter tracking. It is useless to drive the state estimate task harder when decent state estimates are impossible without near parameter convergence. If we can boost performance in just the parameter portion of the extended state, and achieve solid estimates of the time varying parameters, then accurate state estimates will necessarily follow of their own accord. Thus consider

$$M(k) = \begin{bmatrix} \underbrace{\Phi_{11}(k)}_{n} & \vdots & \underbrace{\Phi_{12}(k)}_{2n} \\ \cdots & \cdots & \cdots \\ \underbrace{\Phi_{21}(k)}_{n} & \vdots & \underbrace{\alpha\Phi_{22}(k)}_{2n} \end{bmatrix} \begin{matrix} \} n \\ \\ \} 2n \end{matrix} + Q(k-1) \qquad (4.3)$$

where $\alpha \geq 1$

and

$$\begin{bmatrix} \Phi_{11}(k) & \vdots & \Phi_{12}(k) \\ \cdots\cdots & \vdots & \cdots\cdots \\ \Phi_{21}(k) & \vdots & \Phi_{22}(k) \end{bmatrix} = F(k-1)P(k-1)F^{T}(k-1) \qquad (4.4)$$

In Equation (4.3) we are multiplying the lower right 2n x 2n block of F(k-1)P(k-1)F(k-1) by α instead of the entire matrix. Therefore, only those covariance elements relating parameters to parameters are being boosted. This arrangement effectively tells the algorithm to place more emphasis on achieving good parameter estimates than state estimates, knowing that state estimate accuracy is automatic once the parameters are properly identified.

We will next demonstrate the effectiveness of the parameter block fading factor in allowing the PLID algorithm to track time-varying plant parameters. The only modification to the PLID algorithm software is the multiplication of Equation (4.3) which is clearly specified in the PLID subroutine listing of the Software Appendix. Hereafter, the use of the parameter block fading factor will be referred to as the Tracking-Pseudo-Linear Identification algorithm.

4.4 Tracking-PLID Test Cases and Conclusions

The following test plant will demonstrate the ability of the Tracking-PLID to follow the variation in the parameters with a lag slightly greater than the deadbeat convergence time of 3n samples.

Plant E: $$\frac{1.5\,z^2 - \beta z - 0.5}{z^3 - \alpha z^2 + 1.5z - 0.5}$$

for $k < 50$ $\alpha = \beta = 2$

$50 \leq k < 100$ $\alpha = 2 - (k-50)/100$, $\beta = 2 - (k-50)/50$

$100 \leq k < 150$ $\alpha = 1.5$, $\beta = 1.0$

$150 \leq k \leq 200$ $\alpha = 2.0$, $\beta = 2.5$

Plant E exhibits a linear variation of parameters as well as an abrupt jump in parameters with both the numerator and denominator polynominals affected. The simulations are similar in style to those of Chapter 3 with the same initial conditions and noise parameter settings. The fade factor α is set at a constant 1.001 which has been found to yield excellent results on all plants investigated.

Figure 4.1 shows the deterministic case in which we see that the fading memory of the filter only slightly slows the initial convergence to around sample 12. This seems a small price to pay for the guarantee that successive changes in the parameters will be identified at the same rate. Of particular interest are the convergence rates after the plant change at sample 150 and the linear trend to the parameter estimates between samples 50 and 100. Figures 4.2 and 4.3 demonstrate that, unfortunately, while the Tracking-PLID has noise characteristics similar to those found for the fixed plant cases in terms of initial bias, the tracking feature is much slower. This is an unavoidable result stemming from the tradeoff between heightened algorithm sensitivity for tracking, and a subsequent inability to distinguish between the effect of noisy measurements and parameter variation.

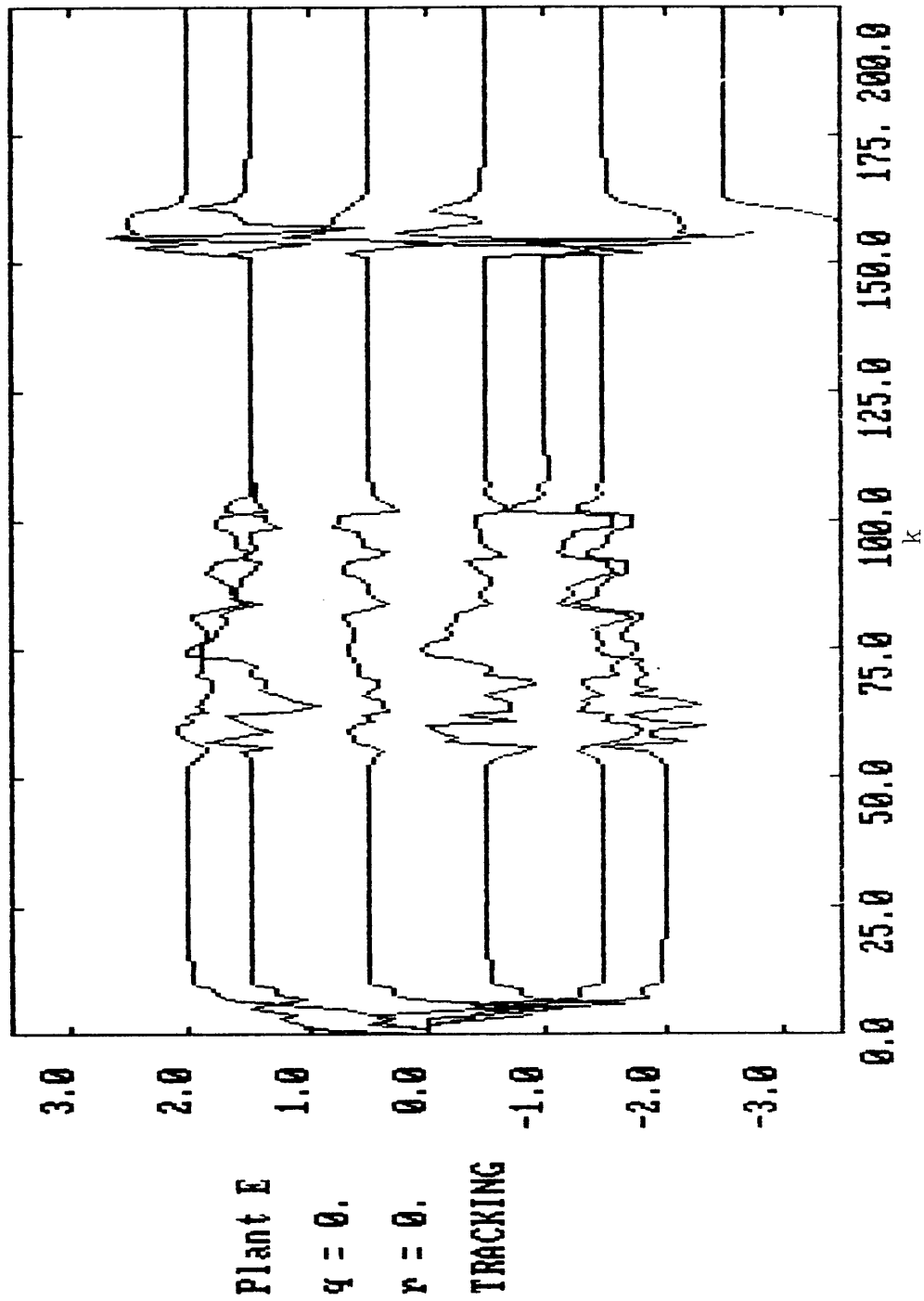

Figure 4.1 Tracking-PLID Simulation , Plant E, Case 1

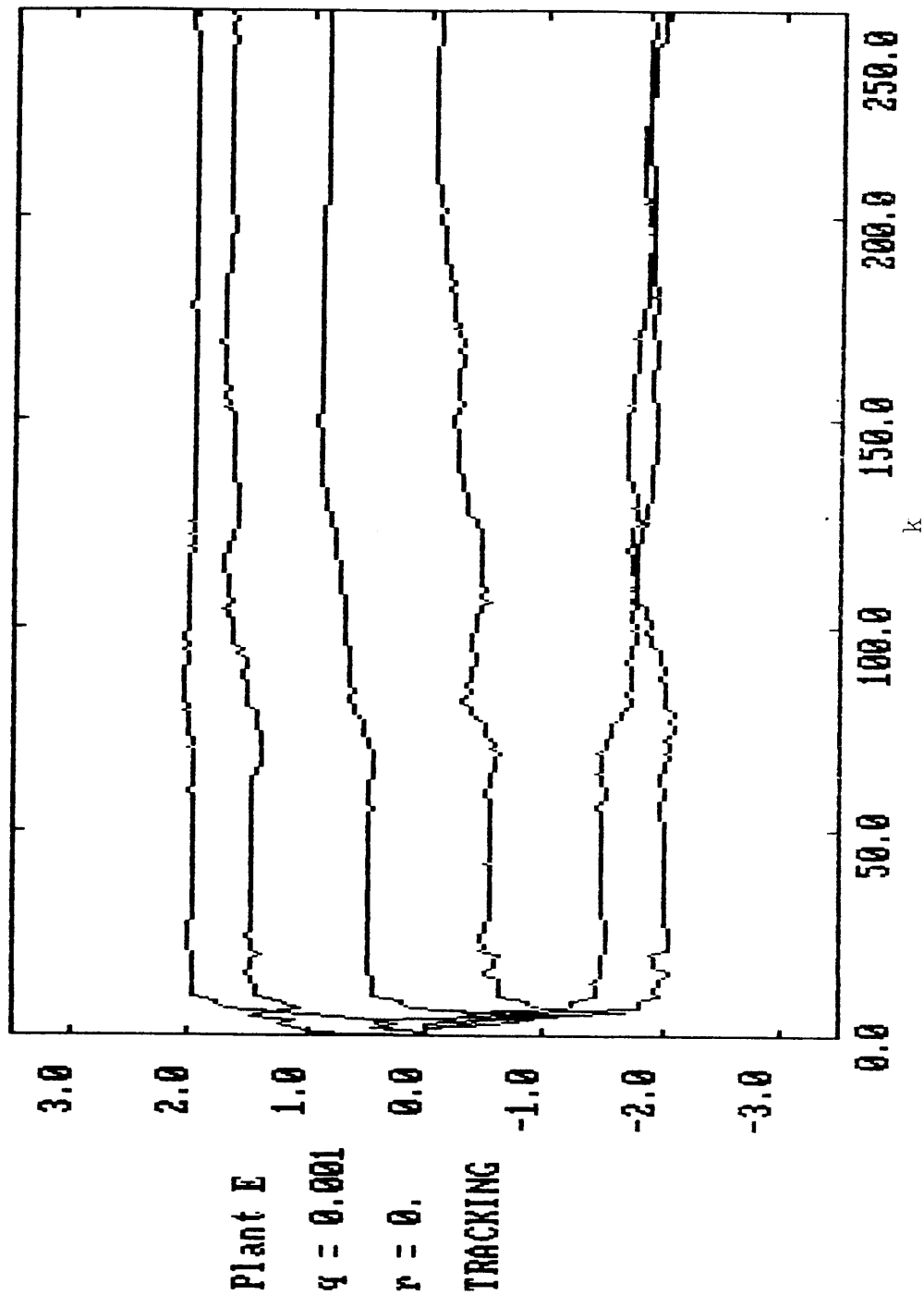

Figure 4.2 Tracking-PLID Simulation, Plant E, Case 2

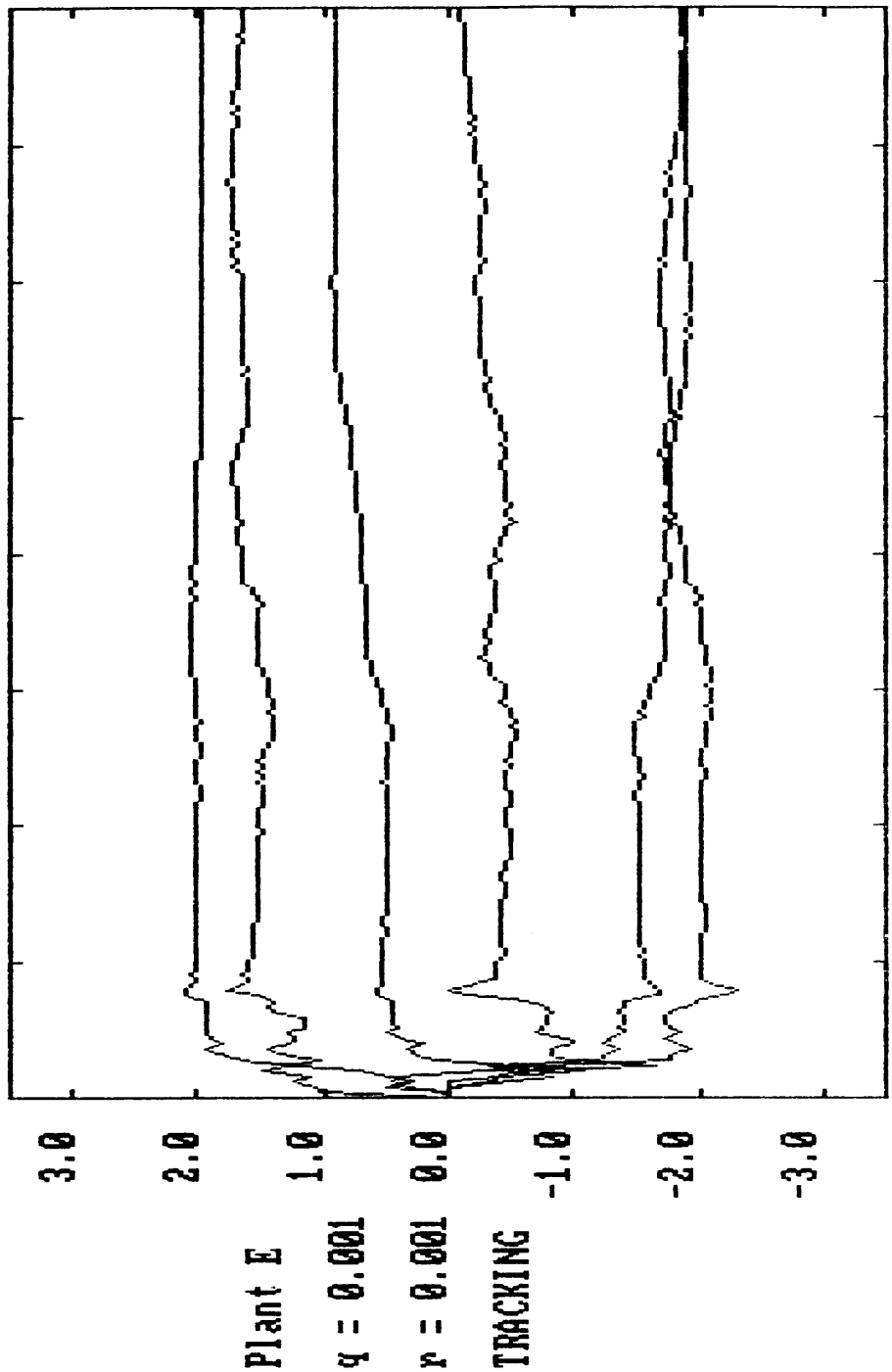

Figure 4.3 Tracking-PLID Simulation, Plant E, Case 3

Quite simply, the parameter block fading factor extends the PLID method to react just as quickly to plant changes as it reacts in its initial parameter acquisition. The presence of the noise parameters Q and r in the algorithm serves to make the algorithm cautious in noisy cases in contradiction of the goal of heightened sensitivity in tracking. As a result, the algorithm cautiousness makes parameter tracking less effective in the presence of noise. This is, however, a common problem with most other identification methods as well, where the goals of insensitivity to noise and sensitivity to parameter variation are in constant conflict [7].

The only real cost seen for this tracking capability is a slight slowing from the deadbeat convergence characteristic to a time of approximately 4n samples. Fortunately, the delayed convergence property in the face of non-persistent excitation is preserved by this tracking modified PLID. The key feature, however, and one that seems unique among identification methods in the literature, is that this 4n sample initial-convergence rate also defines the lag in the parameter estimates behind the changing true parameter values. The consistency of this convergence time is clear from Figure 4.4 where Plant E is modified to change abruptly at k=50 rather than linearly. Examples run using the fading memory RML suggested by Goodwin and Payne [5] exhibit even slower convergence than basic RML and a response to changing parameters that may be described as sluggish at best, even in the noise free case.

With the development of Tracking-PLID, all the groundwork has been laid to formulate an adaptive controller revolving around the PLID

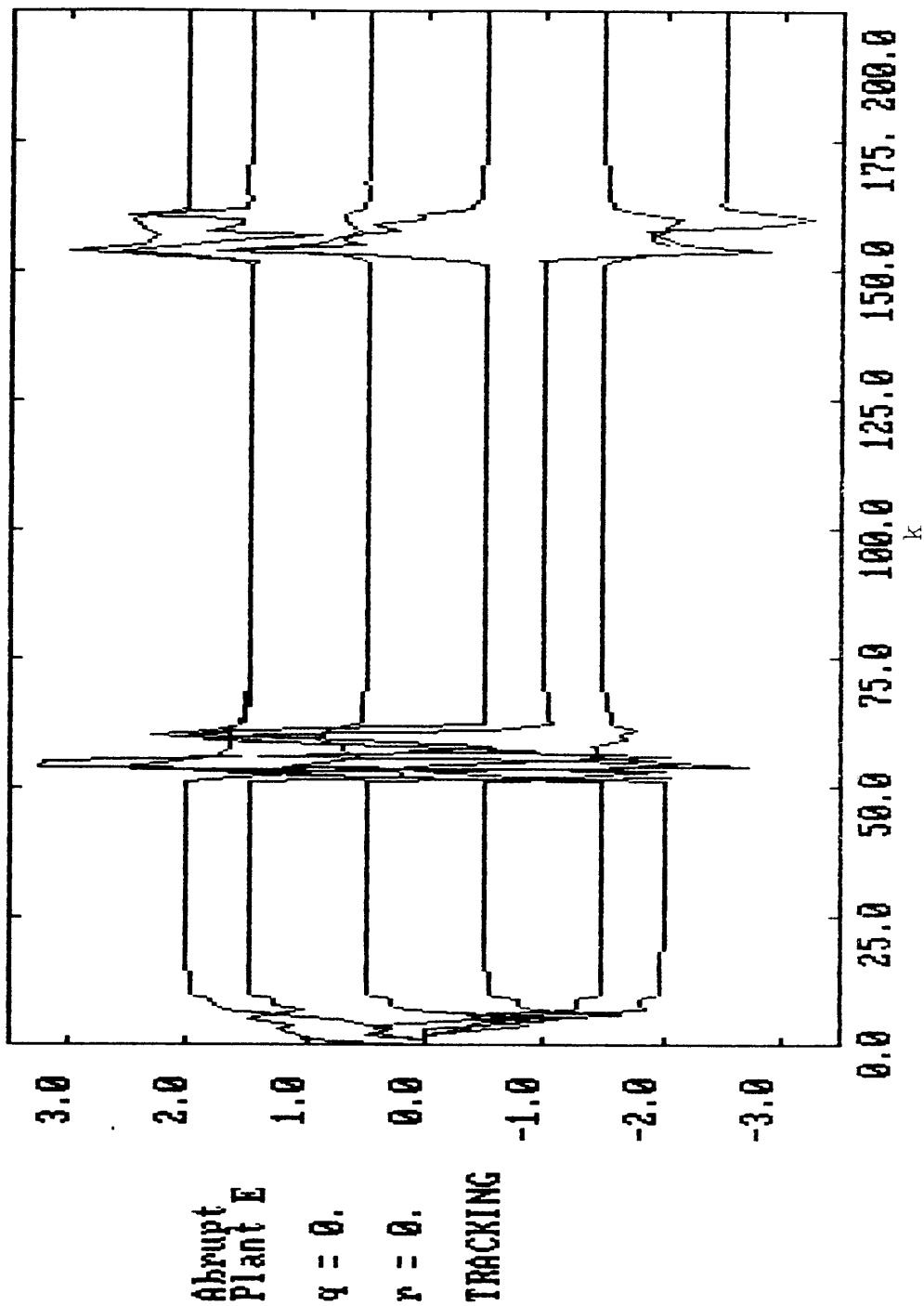

Figure 4.4 Tracking-PLID Simulation, Plant E, Case 4

algorithm. Any indirect adaptive control scheme can only be as effective as the identifier it is based upon. Conversely, the strong capabilities demonstrated by Tracking-PLID suggest that a PLID based adaptive controller would give very quick adaptation to plant changes. This would make the adaptive controller applicable to plants with faster plant variations than the slowly-varying process control type plants seen frequently in the adaptive control literature [2], [11]. In the next chapter, we will build upon this groundwork and propose an adaptive control scheme which relies on the Tracking-PLID algorithm and utilizes the available state estimates as feedback variables.

5.0 APPLICATION TO ADAPTIVE CONTROL

We now turn our attention to the problem of using the PLID algorithm as the basis for an adaptive control algorithm. Once the plant parameter estimates are available, one is free to choose from a wide variety of control synthesis techniques for controlling the plant. Techniques such as Model Reference Adaptive Control, Minimum Variance Control, One-Step Ahead Adaptive Control and the Self-Tuning Regulator [2] have been developed primarily with adaptive applications in mind. An alternative approach to the problem is to automate some common control law design method and carry out these calculations to redesign the controller at each update instant. The parameter estimates provided by the PLID algorithm, which represent those of an observable-canonical state model, suggest that some form of state variable feedback control could be conveniently implemented. In taking such an approach, the entire body of Modern Control Theory can be applied to develop an adaptive controller with properties predictable from fixed design results. The development to follow will lead to a control law synthesis algorithm which will fit into the adaptive control structure of Figure 1.2 and be implemented with a minimum of computational effort.

5.1 The Pole-Placement Approach

Given a linear SISO state model representation of the plant to be controlled, it can be shown [18] that there exists a row vector K such that setting

$$u(k) = -K\underline{x}(k) \tag{5.1}$$

ensures that the closed-loop system matrix

$$A - BK \tag{5.2}$$

has any desired set of eigenvalues. The eigenvalues of A-BK correspond exactly to the poles of the closed-loop system transfer function. We may therefore place the closed-loop poles at any desired location by appropriately defining K, the feedback gain vector. Since the control input is synthesized as a linear combination of the state variables, as in (5.1), this method of control is referred to as pole placement by state variable feedback.

The desired closed-loop poles are chosen to achieve some design criterion. Certainly, the closed-loop poles must be chosen to lie within the unit circle in the z-plane for stability of the closed-loop system. The location within the unit circle may be chosen to maintain a specified degree of damping and rate of decay as dictated by the overall objectives of the control [9].

The choice of the desired closed-loop poles in turn uniquely defines a polynomial in z called the "desired characteristic polynomial" which is the denominator of the closed-loop transfer function to be achieved. In particular, for a system of order n with a set of desired closed-loop poles (p_i),

$$P_{CL}(z) = (z-P_1)(z-P_2) \ldots (z-Pn) \tag{5.3}$$

or alternatively,

$$P_{CL}(z) = z^n + c_{n-1}z^{n-1} + \ldots + c_1 z + c_o \tag{5.4}$$

where $P_{CL}(z)$ is the desired closed-loop polynomial and each of the coefficients c_i is a function of the desired poles p_i.

Given the state model of the plant, denoted by A, B, C, and assuming a regulator input of the form of (5.1)

$$\underline{x}(k+1) = A\underline{x}(k) - BK\underline{x}(k) \tag{5.5}$$

from which

$$\underline{x}(k+1) = (A-BK)\underline{x}(k) \tag{5.6}$$

In Equation (5.6) we may assume the row vector K to be made up of n unknown elements k_i. Eigenvalues of this matrix A-BK define the actual closed-loop poles of the system under the feedback (5.1). Thus

$$A_{CL}(\lambda) = \det(\lambda I - A+BK) \tag{5.7}$$

$$A_{CL}(\lambda) = \lambda^n + a_{n-1}\lambda^{n-1} + \ldots + a_1\lambda + a_o \tag{5.8}$$

$$\therefore\ A_{CL}(z) = z^n + a_{n-1}z^{n-1} + \ldots + a_1 z + a_o \tag{5.9}$$

defines the actual closed-loop polynomial, where each of the a_i is a linear combination of our unknown gains k_i in the SISO case. Equating (5.4) and (5.9) implies that if

$$\begin{aligned} a_o &= c_o \\ a_1 &= c_1 \\ &\vdots \quad\quad \vdots \\ a_{n-1} &= c_{n-1} \end{aligned} \tag{5.10}$$

the actual closed-loop poles equal the desired closed-loop poles. We may therefore solve the system of n linear equations (5.10) for the unknown components of K to determine the necessary feedback to achieve the desired closed-loop poles. A solution to (5.10) is guaranteed to exist if and only if the pair A, B is controllable from the scalar input u(k) [18]. We therefore must insist that our SISO plant be observable for PLID identification and controllable for pole placement in order for our adaptive controller to function.

The state variable feedback approach, of course, assumes that the states of the plant x(k) are available for feedback. Either they must correspond to some physically measurable entity or they must be able to be synthesized from the output data. It is highly unlikely that the states of the observable-canonical form will directly correspond to any physical output which could be measured by a sensor. Fortunately, many standard techniques exist for estimating the state of a system from its

inputs and outputs [19]. By basing our adaptive controller on the PLID algorithm, however, state estimates are automatically available for feedback. The PLID algorithm simultaneously performs the tasks of identifier and adaptive observer. State variable feedback is therefore the most natural form of adaptive control for fully utilizing the power of the PLID algorithm.

A compact, direct computational method exists for the solution of the pole placement problem. Direct solution of Equations (5.10) is only practical when A and B form a controllable-canonical pair. For our observable-canonical state model we may more efficiently solve for K by applying Ackermann's formula [20]

$$K = [0\ 0\ \ldots\ 0\ 1]\ [B\ \ AB\ \ldots\ A^{n-2}B\ A^{n-1}B]^{-1} P_{CL}(A) \tag{5.11}$$

where

$P_{CL}(A)$ is $P_{CL}(z)$ from (5.4), i.e.

$$P_{CL}(A) = A^n + c_{n-1}A^{n-1} + \ldots + c_1 A + c_o I \tag{5.12}$$

The controllability requirement guarantees that the matrix inverse in Equation (5.11) exists.

While solution of (5.11) is relatively computationally intensive, it does provide a direct transformation from the identified parameters in A and B to the feedback gains K which meet the design objectives reflected in $P_{CL}(z)$. A computer subroutine to execute (5.11) thus constitutes our automated design procedure for adjusting the

characteristics of the adaptive controller. The computational requirements of this state feedback approach are certainly less than using Linear Quadratic Regulator (LQR) synthesis for example. The LQR design procedure, which requires the iterative solution of the discrete-time Riccati equation, is less practical for on-line computation. Having developed an automated pole placement approach, we will next extend pole placement to achieve the output tracking of reference inputs.

5.2 State Feedback with Integral Action

The control structure defined by Equation (5.1) falls into the broader class of linear regulators. Equation (5.5) shows no provision for the introduction of a reference input. This is typical of regulator structures which are used simply to maintain the plant at the steady operating condition $\underline{x} = \underline{0}$. Any deviation from $\underline{x} = \underline{0}$ is regulated back to zero with response shapes dictated by the assigned closed-loop poles. While this is an important class of problems in its own right, the performance goals clearly lack the flexibility to drive the system from one desired state to another. We now seek to expand upon the state variable feedback regulator to achieve an input following controller. That is, we want the system output $y(k)$ to follow a reference input $r(k)$ while maintaining a specified set of closed-loop poles.

This goal can be simply achieved by the introduction of "integral action" on a signal representing the error between the reference and the output. Such integral action corresponds to the classical domain Type 1

closed-loop system which guarantees zero steady-state error to step inputs [10]. The relative speed with which the steady state is achieved may then be increased by moving the desired closed-loop poles closer to the z-plane origin. The integral action controller design problem can fortunately be cast in a form easily solved by Ackermann's formula in a way similar to the regulator case.

To formulate the input tracking controller we introduce an auxiliary state z(k) [2] where

$$z(k) = \sum_{j=0}^{k-1} [r(j) - y(j)] \tag{5.13}$$

or

$$z(k+1) = z(k) + r(k) - y(k) \ , \ z(o) = 0 \tag{5.14}$$

For design purposes we temporarily assume r(k)=0 and combine Equation (5.13) with the observable-canonical form of the plant as

$$\begin{bmatrix} x(k+1) \\ \\ z(k+1) \end{bmatrix} = \left[\begin{array}{c:c} A & 0 \\ \hdashline 0 \ \ldots \ -1 & 1 \end{array}\right] \begin{bmatrix} \underline{x}(k) \\ \\ z(k) \end{bmatrix} + \begin{bmatrix} B \\ \\ 0 \end{bmatrix} u(k) \tag{5.15}$$

$$y(k) = [\ 0 \ \ldots \ 1 \ 0\ ] \begin{bmatrix} \underline{x}(k) \\ \\ z(k) \end{bmatrix}$$

From the augmented state model (5.15), we impose linear state variable feedback of the augmented state

$$u(k) = -\begin{bmatrix} K & Ke \end{bmatrix} \begin{bmatrix} \underline{x}(k) \\ z(k) \end{bmatrix} \tag{5.16}$$

With some thought it becomes clear that the gains $[K \quad K_e]$ which will guarantee a desired set of n+1 closed-loop poles may be obtained by directly performing Ackermann's formula for the system (5.15)

$$\left[\begin{array}{ccc:c} & A & & 0 \\ \hdashline 0 & \cdots & -1 & 1 \end{array}\right] \quad \begin{bmatrix} B \\ 0 \end{bmatrix} \xrightarrow[\text{Eqn. (5.11)}]{} \begin{bmatrix} K & Ke \end{bmatrix} \tag{5.17}$$

Once this is done, the reference input r(k) is reintroduced and the state variable feedback with integral control is implemented as shown in Figure 5.1. Study of the system diagram of Figure 5.1 indicates that the closed-loop system can only arrive at a steady state when y(k)=r(k) leading to a constant z(k). Use of the pole placement approach in setting the gains $[K \quad K_e]$ in turn guarantees that the desired steady state will be reached with a rate, damping and frequency defined by the desired stable closed-loop pole locations.

This approach through Ackermann's formula provides a direct transformation between the PLID identified parameters, estimated states and state feedback gains needed to place the poles of the augmented system of Equation (5.15). The combining of this method with the PLID

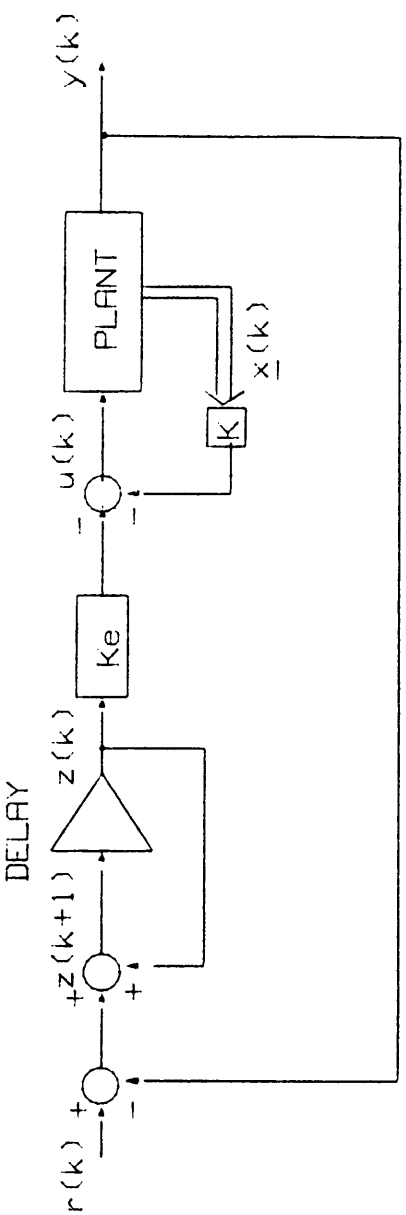

Figure 5.1 State Variable Feedback with Integral Action Control

algorithm completely defines the form of the adaptive controller which is discussed in the following section.

5.3 Implementation of the Adaptive Control Algorithm

Figure 5.2 illustrates the interaction of the various components of the adaptive control algorithm based on PLID. At each sample instant, the input and output measurements are used by the PLID algorithm to update the parameter and state estimates. The parameter estimates are, in turn, processed by Ackermann's formula, subject to preset desired closed-loop poles, to adjust the feedback gains appropriately. The control algorithm then uses the measured output and reference input to compute the running error state z(k) which, when combined with the PLID state estimates and the feedback gains, is used to determine the next controlled plant input u(k).

All the processing necessary to synthesize u(k) from the PLID output $\underline{s}(k)$ (containing parameter and state estimates) is performed by the subroutine CA listed in the Software Appendix. This subroutine uses a matrix inversion routine employing Gaussian elimination with partial pivoting [21] to execute Ackermann's formula. Subroutine CA also performs the updating of z(k) as well as the linear feedback product which computes the input u(k). The adaptive control subroutine also contains several logic function protections which may be customized by the user to improve the system performance in instances of high parameter uncertainty or transition. This logic is used to reject any feedback update deemed unreasonable based on prior knowledge of physical

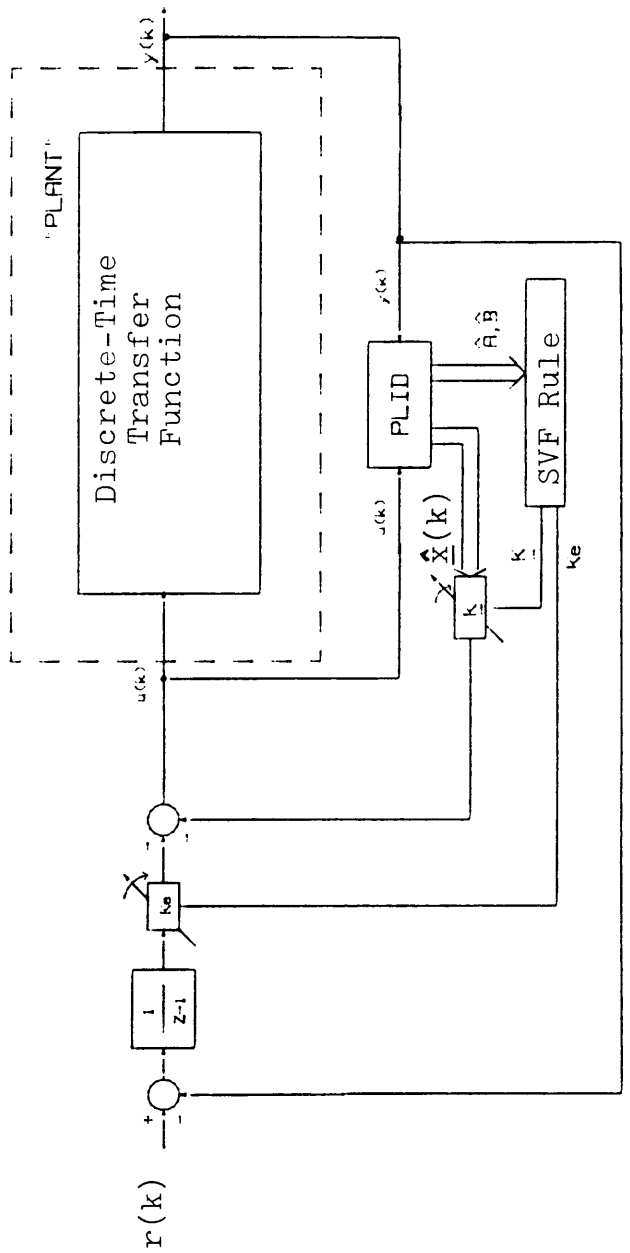

Figure 5.2 Adaptive Control Based on PLID

properties of the plant. The proper "tuning" of these logic parameters and a reasonable initialization of the feedbacks and PLID estimates can greatly enhance the performance of the system under adaptive control.

The first logic safeguard lies within the matrix inversion of Ackermann's formula. Assuming the physical plant does, in fact, satisfy the condition of controllability, no correct parameter estimate would lead to the inversion of a singular matrix. We may therefore assume that any attempt to invert a singular matrix is the result of a questionable parameter estimate. The matrix inversion routine includes a test for singularity which leads to an error flag rather an algorithm failure. Thus, anytime the singularity flag is activated, Subroutine CA simply keeps the previous feedback values in place until a nonsingular, and assumed more correct, parameter estimate becomes available. This safeguard is especially useful in starting the adaptive controller from an initial zero-parameter estimate which, clearly, is not controllable. In this case, the initial feedback "guess" supplied by the user is simply held until the PLID has progressed in its convergence to a controllable parameter estimate.

Quite frequently, the user has an intuitive feel for the "correct" sign of the feedbacks. This is especially true for the error signal feedback K_e. By physical reasoning, with an aircraft in pitch for example, we know that a reference input r(k) for positive (pull-up) g-force should lead to a positive (tip up) deflection of the elevator surface u(k). Since the influence on u(k) from the r(k) input arises from $-K_e$ z(k) it is clear that K_e should always be negative in this

example. To address this issue, Subroutine CA incorporates additional logic which rejects any set of feedback gains not satisfying the user's sign restrictions. Thus, in the above example, a positive update of K_e would be rejected in favor of keeping the previous negative value of K_e. This feature is also especially useful during the initial startup "learning" phase if PLID is initialized at zero.

Application of the above ideas makes it possible to start the adaptive controller from a "no knowledge", $\underline{s}(0) = \underline{0}$, condition while still maintaining a reasonable hold on the resulting outputs. To do this it is simply necessary to supply a small nonzero initial value on the K_e feedback, which of course must satisfy any sign restriction programmed into CA. The identification process would then begin with the first nonzero reference input which would excite the plant through u(k). The PLID algorithm uses measurements of the resulting outputs to refine its estimate of the parameters. The rather loose integral control established by the initial K_e setting is improved steadily beginning with the first nonsingular parameter set to result in a feedback set of proper sign. This improvement continues until final parameter convergence fixes the feedback gains at their "ideal" values. Of course, the length of this learning process is significantly shortened if a good initial estimate of the parameters is supplied.

The PLID adaptive controller just described has been demonstrated to be very effective in learning and adapting to provide uniform control characteristics in the face of changing plant dynamics. A theoretical proof of stability is not available for this control algorithm and would

be particularly challenging in the stochastic case. One can argue, however, that if the PLID converges to correct parameter estimates, stability of the closed loop is guaranteed by the pole placement theory [20]. Since convergence of PLID in the deterministic case has been proven [8], stability of the PLID-based adaptive controller in this no-noise case is assured. Long term stability of the system, of course, is contingent upon stability (non-divergence) of the PLID Kalman filter, an issue which will be addressed in Chapter 7. We will now demonstrate further properties of the PLID adaptive controller by simulation of a variety of control situations.

5.4 Properties of the Adaptive Controller

Simulations of the adaptive controller in operation have demonstrated its effectiveness even in the face of measurement and process noises. While measurement noises have been shown to bias the PLID parameter estimates, these small biases do not significantly degrade the properties of the closed-loop system performance. This is consistent with the literature [11] which suggests that relatively rough parameter estimates may still lead to effective closed-loop adaptive control. The primary demonstrated result is, however, the ability of the PLID adaptive controller to adjust in the environment of a radically changing plant to provide consistent control where fixed gain controllers would fail.

The plant model in the examples to follow represents a preview of the application of PLID adaptive control to aircraft dynamics. In

particular

		Poles/Zeros
PLANT F	$\frac{-0.1025z + 0.15}{z^2-1.7692z+0.7826}$	$\frac{1.46}{\approx 0.89,\ 0.89}$

is the result of sampling (at 0.01 sec.) the SISO transfer function between elevator deflection (positive for tip up) and normal acceleration of a high performance fighter aircraft with inner-loop pitchrate feedback. A second-order approximation to a dominantly fourth-order process has been made and the non minimum-phase zero has been increased to 1.46 from its natural 1.14 to emphasize further the PLID capabilities. The application of the adaptive controller to the full eighth order nonlinear model will be the subject of the following chapter.

The cases to follow were initialized from a "zero knowledge" condition of

$$\underline{s}(o) = \underline{0}$$

$$P(o) = \text{diag}\ (200, \ldots, 200) \tag{5.17}$$

The block fading factor was set to 1.001 and the initial feedback gains were $K_e = -0.01$ and $K = \underline{0}^T$. Within CA, the only logic constraint was $K_e > 0$ based upon our previously discussed physical intuition regarding the input-output relationship of this plant. The state feedback matrix K was allowed to assume any sign.

Figures 5.3 and 5.4 illustrate the application of PLID adaptive control to Plant F. The closed-loop poles were all placed at 0.5 to provide a fast, critically-damped response to the reference input sequence shown. At k=150 the open-loop plant poles are shifted to an unstable 1.3 and 0.6, and at k=250 they are shifted to $0.88\pm0.13j$ by varying one of the denominator parameters. The parameter traces point out one of the common characteristics of identification within a closed feedback loop. Convergence is clearly slower than the six-step deadbeat rate expected in this deterministic case. The reason for the slower convergence is that the excitation of the input is much less than that of the white-noise sequence used in our open-loop experiments. The fact that the input $u(k)$ is dependent upon $y(k)$ (through feedback) increases the likelihood of dependence among rows of the observability matrix (2.26) which delays convergence. Despite this, the algorithm does quickly "learn" the plant model and brings the system under control with the desired critically-damped response.

The non minimum-phase behavior is clear in the output trace as well. Following steps 150 and 250, the output does depart drastically from the reference value due to the abrupt change in the plant dynamics; this is unavoidable since the controller cannot instantaneously adapt to such a change. More important is the ability of the algorithm to restabilize the closed-loop system and return quickly to meeting the design criteria after the change. Figure 5.5, on the other hand, illustrates a divergence and total loss of control resulting from the same plant change under feedback gains which are fixed at their initial

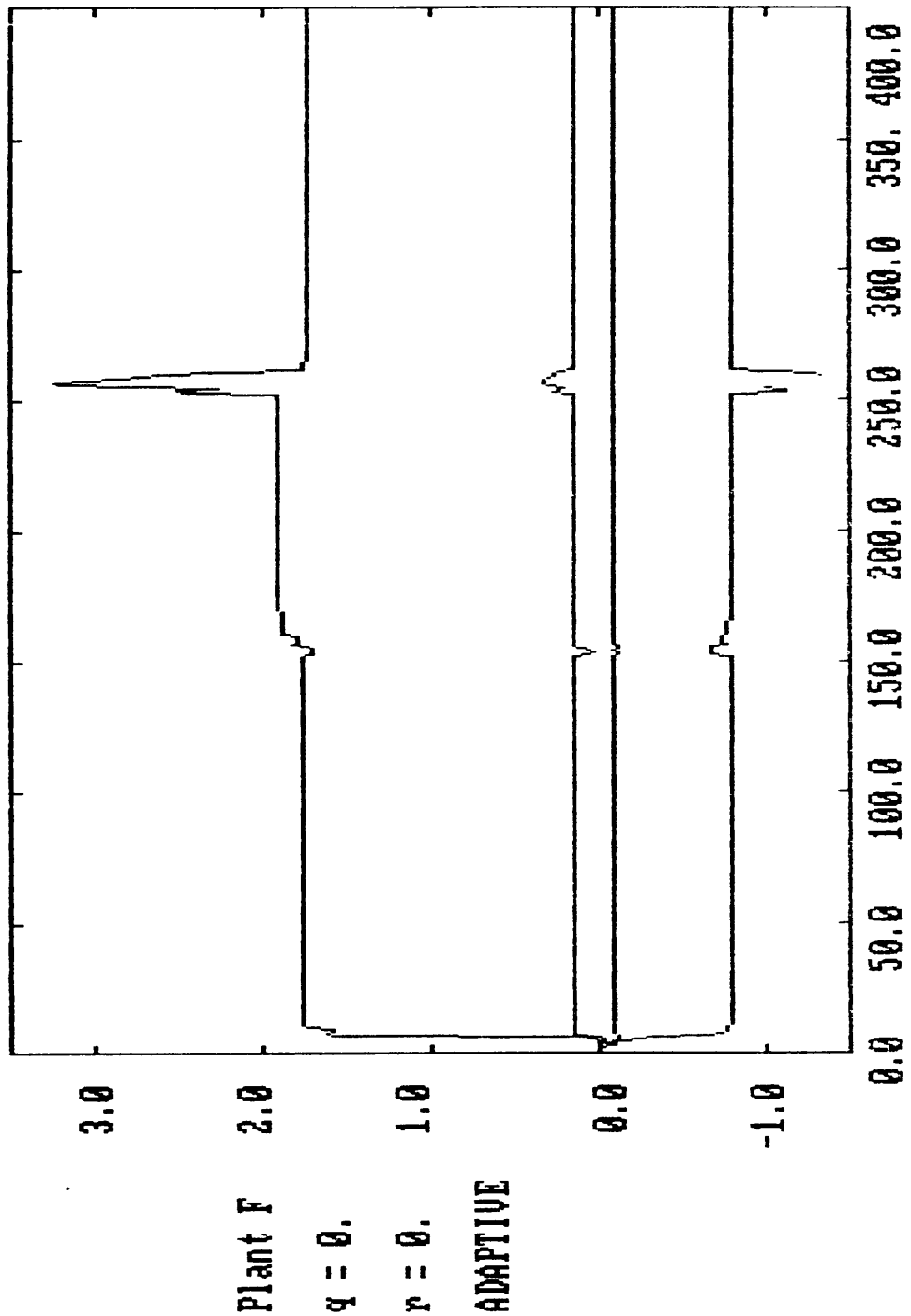

Figure 5.3 Identified Parameters, Case 1

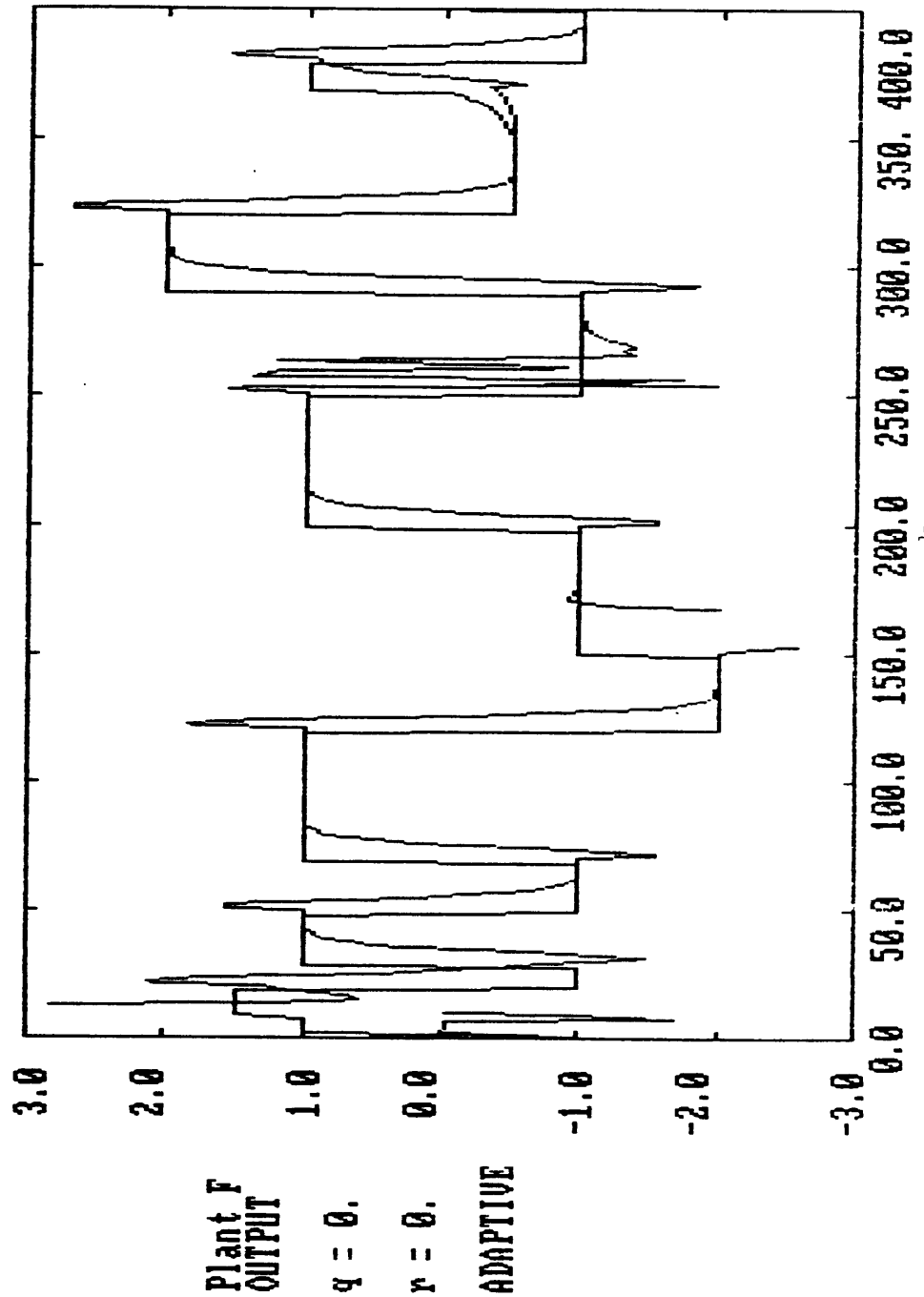

Figure 5.4 Output under Adaptive Control, Case 1

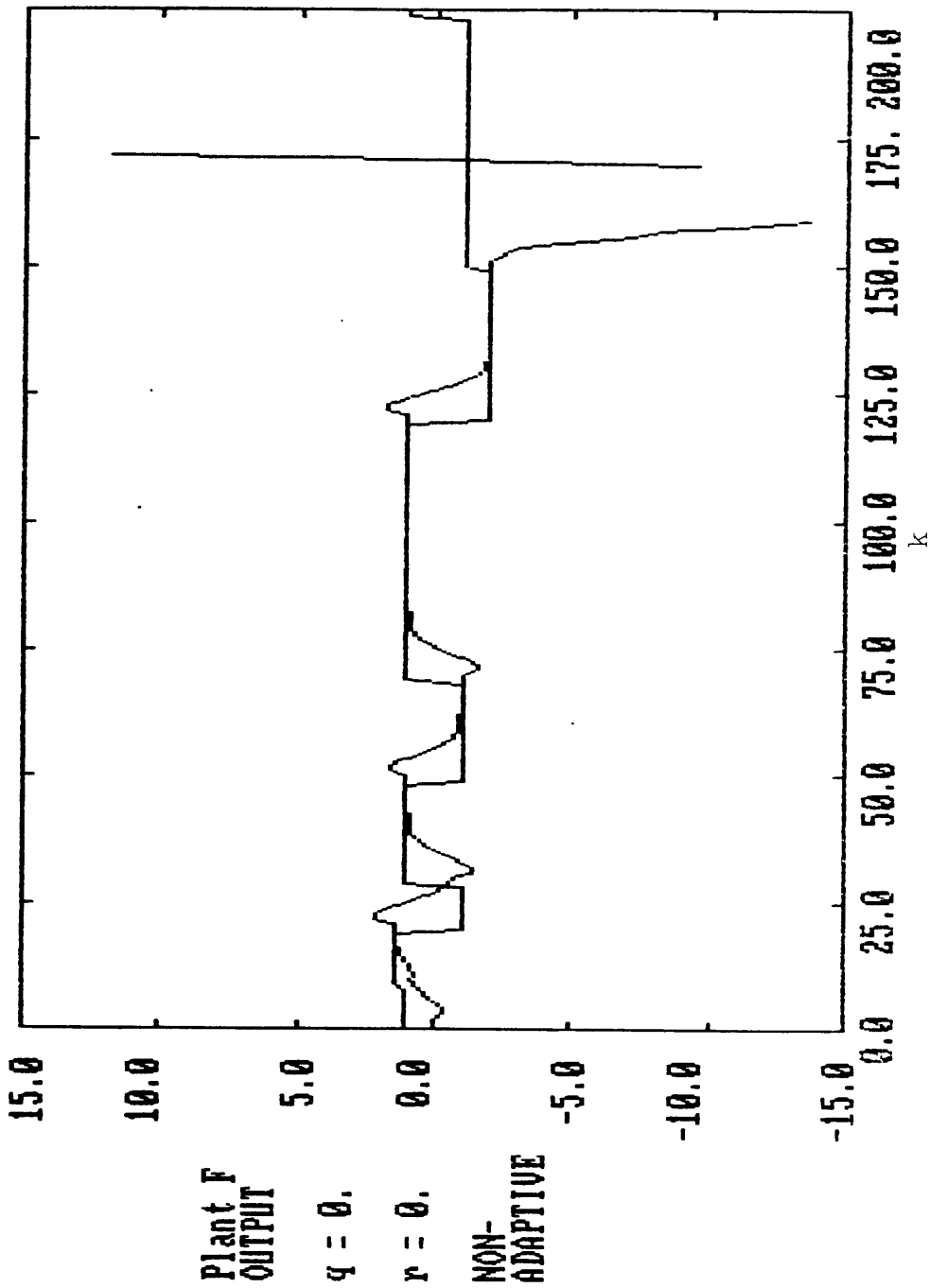

Figure 5.5 Output without Adaptive Control, Case 1

values. The advantage of adaptive control is clear in this situation, as is the capability of the PLID adaptive controller in coping with it.

Figures 5.6-5.13 illustrate the effect that a different set of closed-loop poles can have on the adaptive controller. Here we have set the closed-loop poles to a critically-damped value of z=0.8, which results in predictably slower responses, and exhibiting less of a non-minimum-phase characteristic. The plant is only changed once at the 200th sample to an unstable configuration with poles at z=1.3 and 0.6. The main advantage of this desired closed-loop pole set is that the feedback gains are significantly lower than in the previous example. This leads to improved performance in noise of the closed-loop response (which is a property of pole placement itself and not a PLID sensitivity). All control schemes must trade off fast response and its associated high gains with the resulting increased sensitivity to noisy measurements.

Figures 5.6 and 5.7 demonstrate the effect that measurement noise with a variance r=0.005 has on the controlled system. Despite the clear bias introduced in the parameters, we see that the output responses achieve the design goals. As pointed out when the fading-memory PLID was developed, the adaptation to plant change is slowed by the presence of noise. Stability is, however, rapidly regained after step 200, further emphasizing the forgiving nature of closed-loop properties. The higher variance in the output seen after step 200 is the result of the higher gains required after the adaptation to the new dynamics. Similar

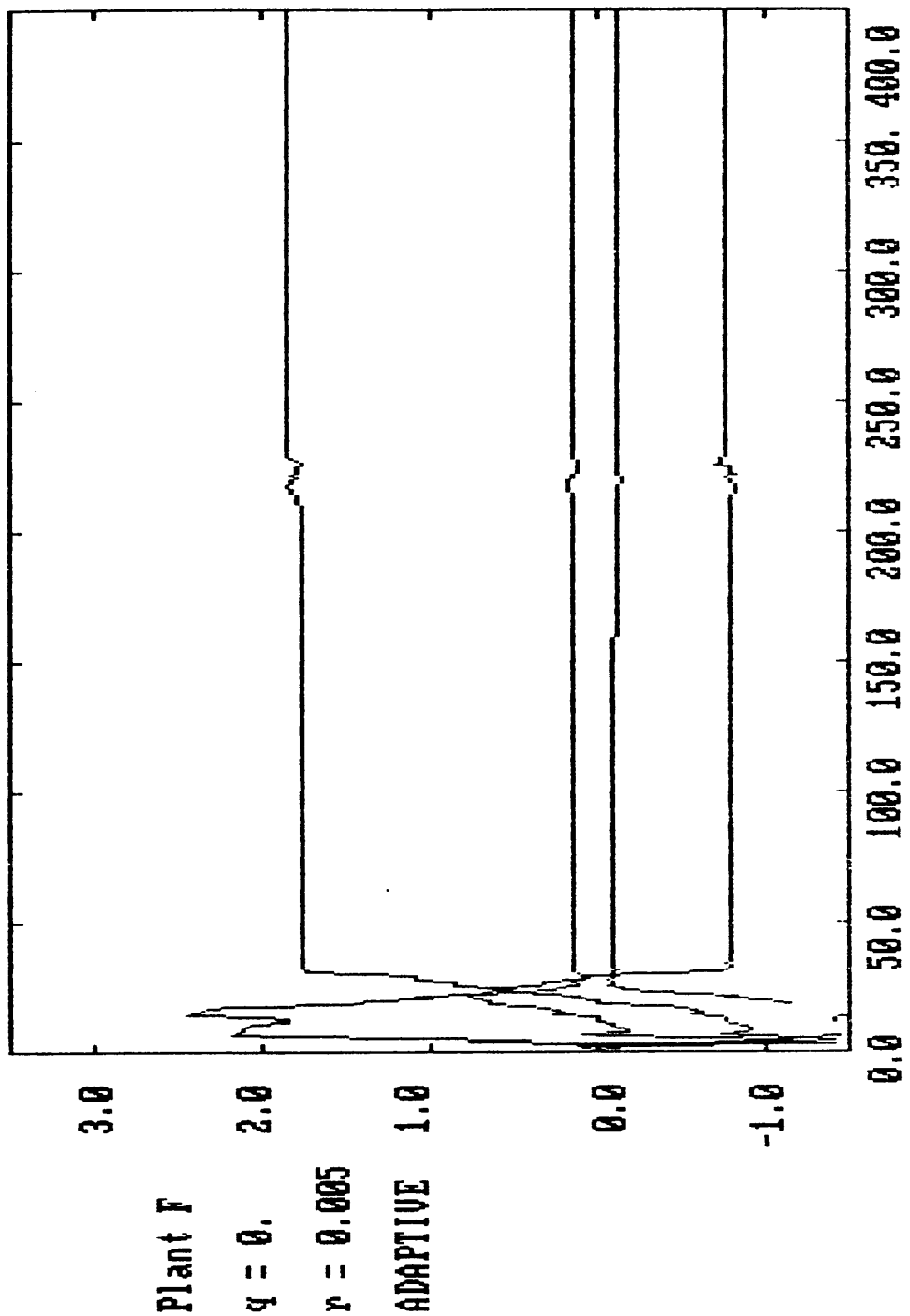

Figure 5.6 Identified Parameters, Case 2

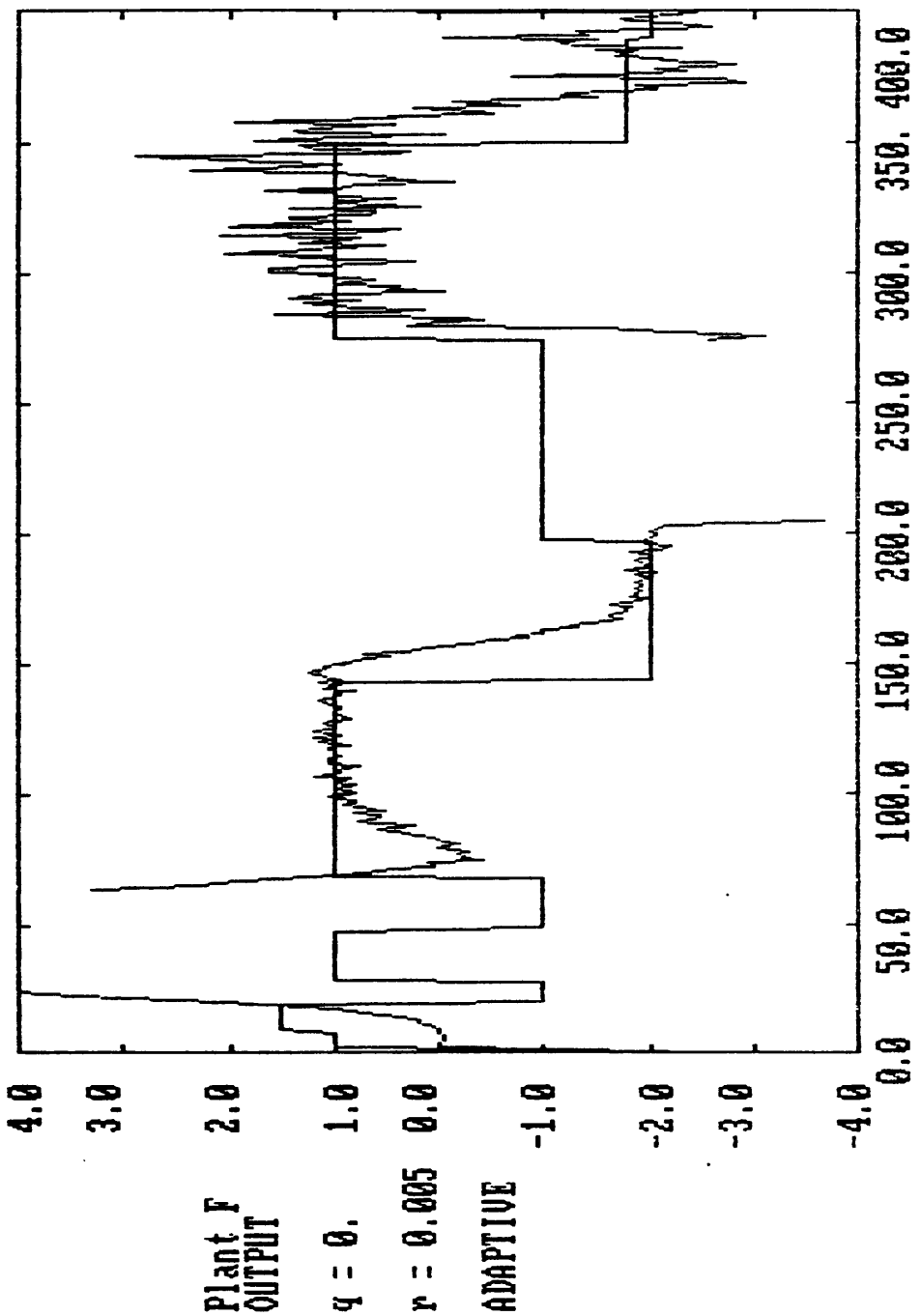

Figure 5.7 Output under Adaptive Control , Case 2

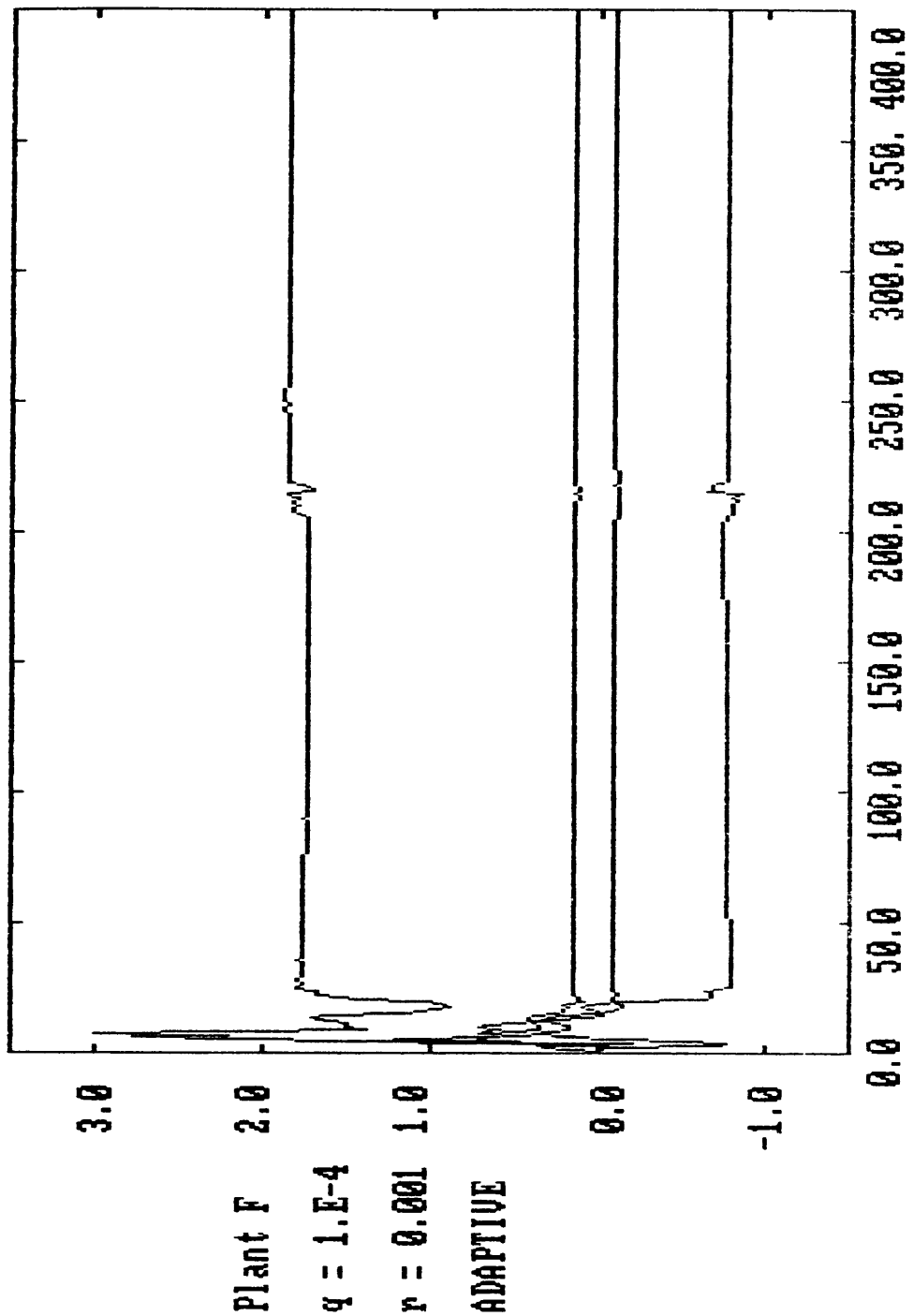

Figure 5.8 Identified Parameters, Case 3

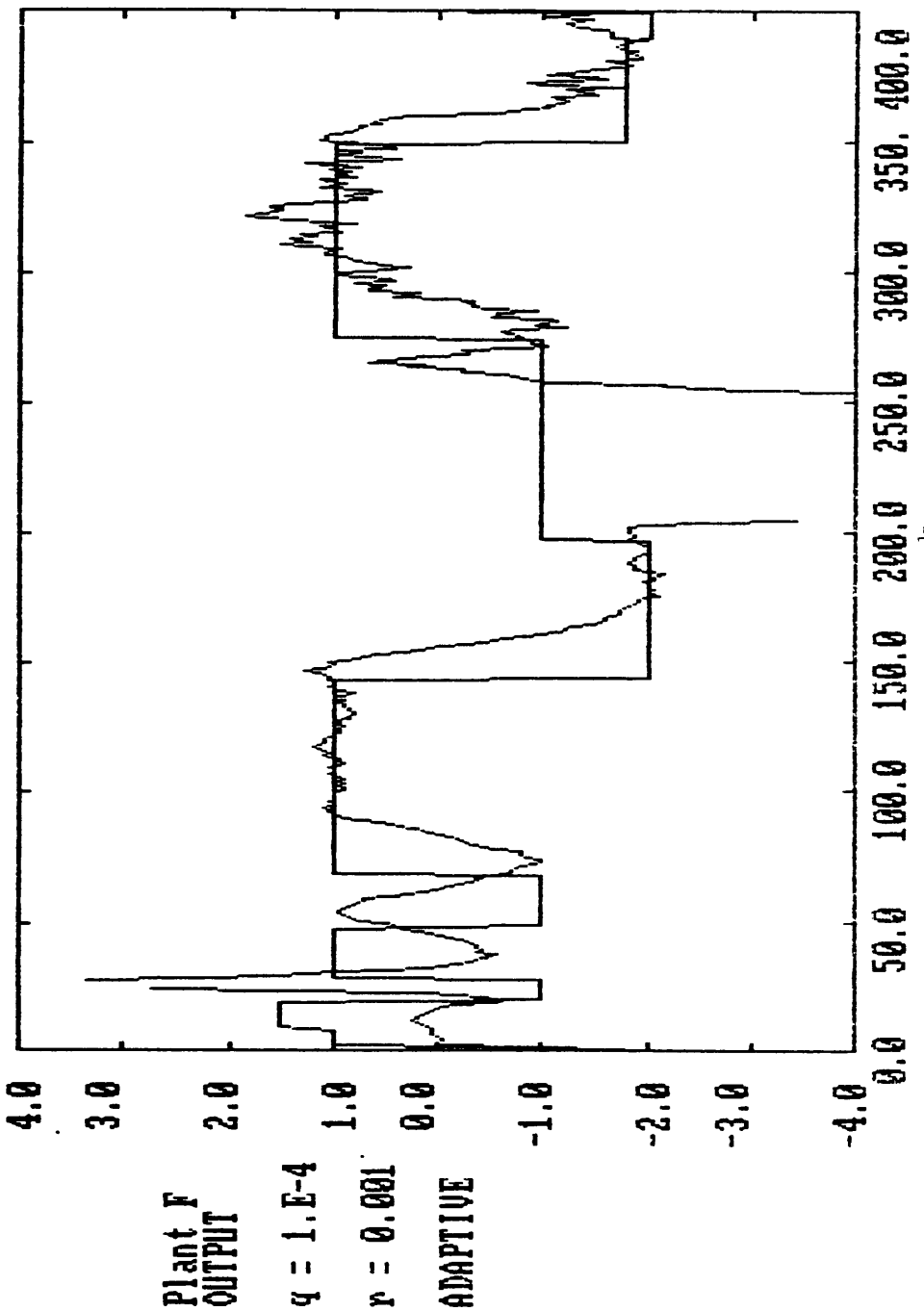

Figure 5.9 Output under Adaptive Control, Case 3

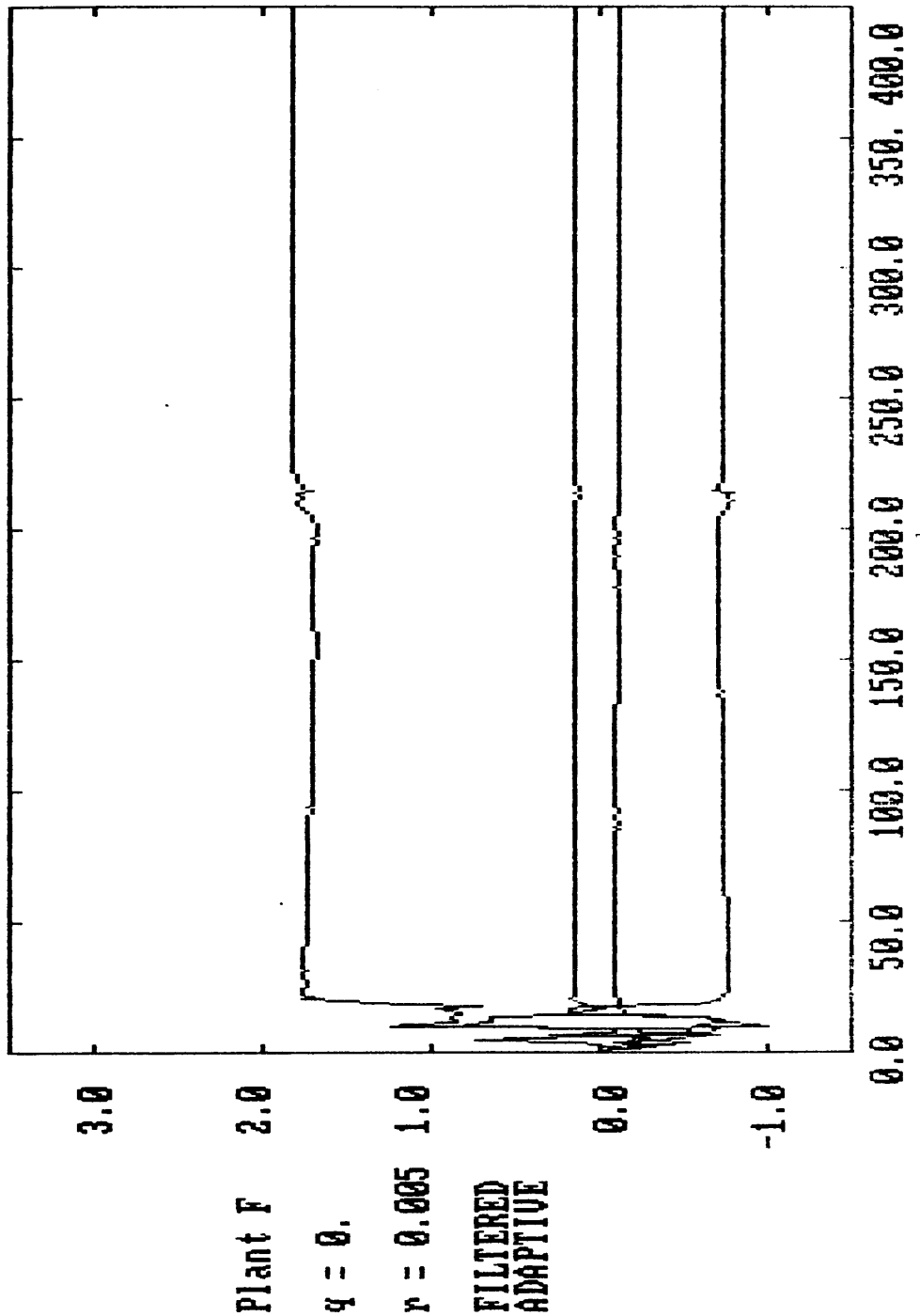

Figure 5.10 Identified Parameters, Case 4

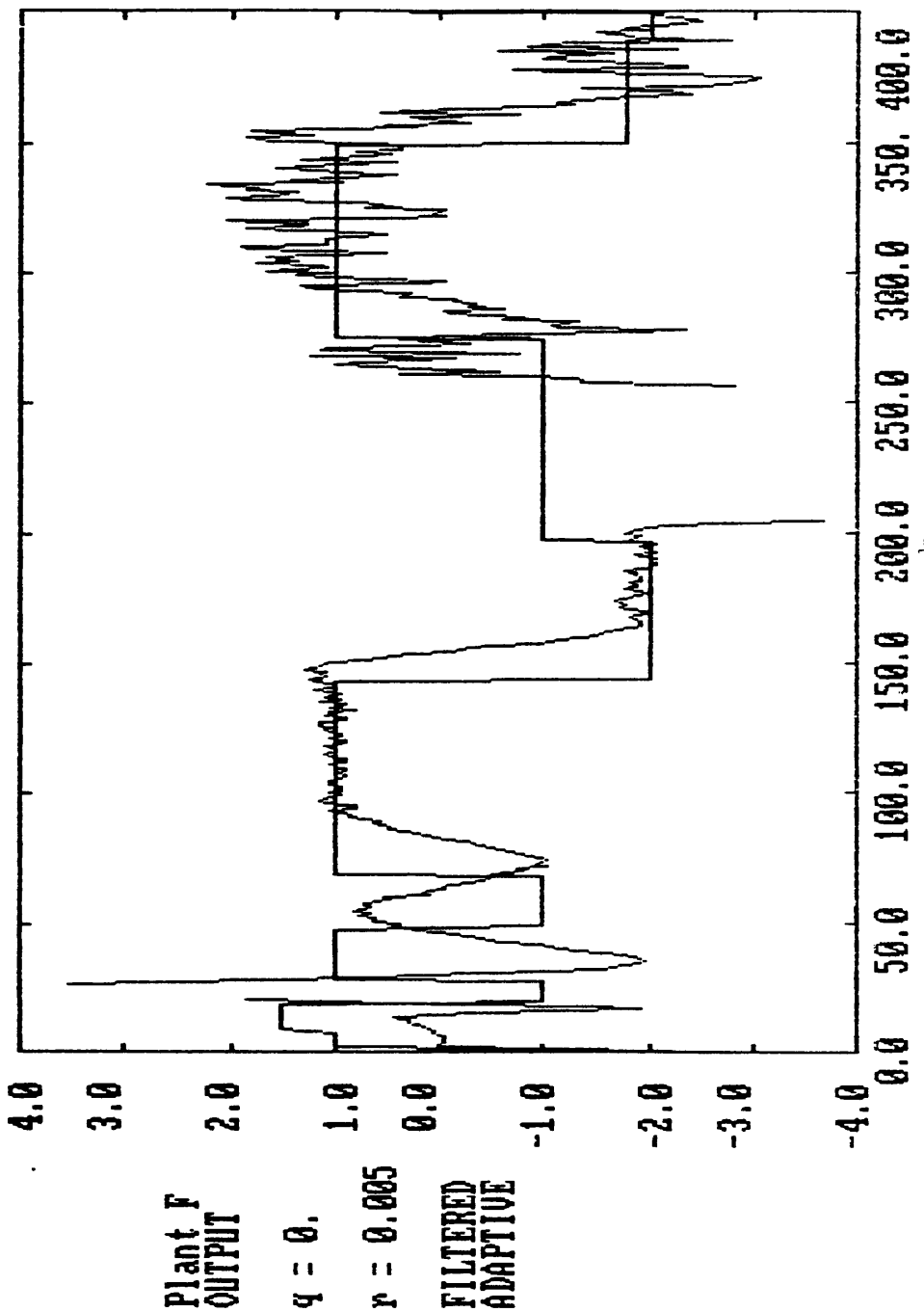

Figure 5.11 Output using Filtered Output Feedback, Case 4

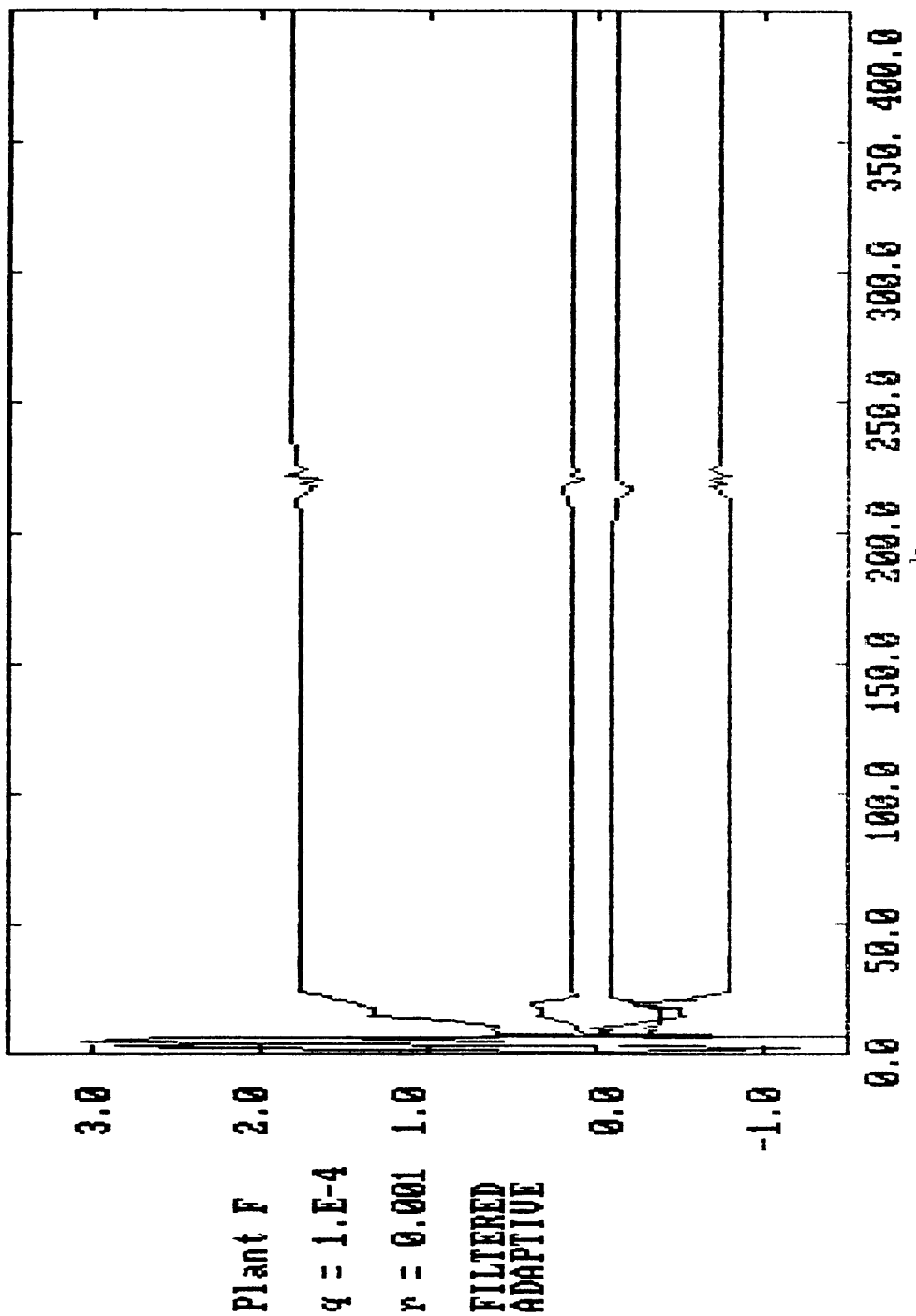

Figure 5.12 Identified Parameters, Case 5

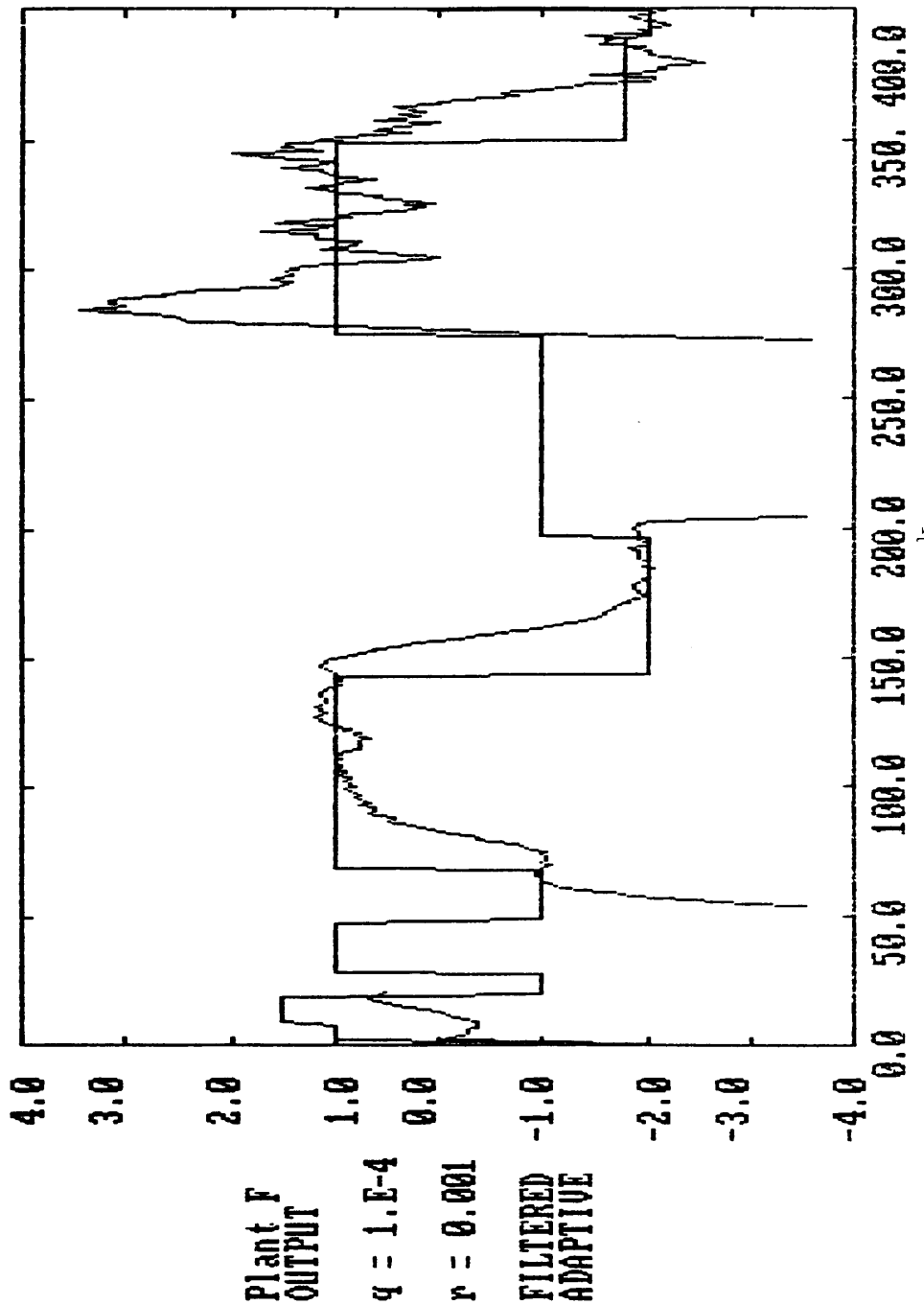

Figure 5.13 Output using Filtered Output Feedback, Case 5

effects are seen in Figures 5.8 and 5.9 where the measurement noise variance is r=0.001 and the process noise variance is q=0.0001. The adjustment to the changed parameter is clearly slowed, but it is sufficient to restabilize the system.

In some cases it may be desirable to use the information provided in the PLID state estimates to further advantage. With the observable-canonical form imposed by PLID it is clear that the last state variable x_n is the output y. Thus, the PLID estimate $\hat{x}_n$ is the Kalman filtered estimate of y (which minimizes the mean-square error). Suppose, therefore, that $\hat{x}_n$ is used in the error channel feedback to z(k), instead of the noisy direct measurement of y; Figures 5.10-5.13 illustrate the result of this modification in the above noisy cases. What is seen is an improvement in the adaptation time of the algorithm. The variance in the true output is essentially unchanged, though some improvement might be expected in the case of significant measurement noise (of course our simplified PLID is highly biased in this case and the more exact algorithm of [8] would have to be employed to examine this). The improvement in the adaptation time results from a reduction in the correlation between the output measurement y(k) and the input u(k). u(k) no longer depends directly on y(k) but on its estimate $\hat{x}_n(k)$ in feedback, thus strengthening the independence of the rows of the observability matrix (2.26).

Care should be exercised in using this filtered feedback in cases where the validity of the linear state model is questionable. If, for example, the model being identified is a reduced-order approximation to

a plant possibly containing unmodeled nonlinearities, it would be more prudent to apply true y(k)-feedback than to rely on the accuracy of the filtered estimate, $\hat{x}_n(k)$, which would only be as good as the model approximation. In the aircraft example of the next chapter, for instance, true output feedback was chosen due to the highly nonlinear nature of the plant and because the measurement error was assumed insignificant.

The above examples have illustrated the advantage to be gained by adaptively controlling plants with unpredictably changing dynamics. While some undesirable transients might occur from an abrupt plant change, the restabilization afforded by the adaptive controller is clearly superior to the complete loss of the system, as in the fixed controller situation. It is hoped that the above development and examples have also demonstrated the need for good judgement in the implementation of adaptive control. The proper choice of feedback constraint logic, initial feedback and parameter settings, the relation between desired closed-loop poles and noise sensitivity, and the use of filtered versus true output feedback are all crucial in ensuring the success of the adaptive controller operation.

Lest this warning make one overly wary of adaptive control, keep in mind that the consideration of these issues is also critical in the development of effective fixed-gain controllers. The combination of the PLID adaptive control scheme with appropriate implementational common sense does, therefore, give the control system designer the ability to cope with unforeseen situations; thus extending the utility of active

control beyond the capabilities of fixed-gain control. The conceptual advance of adaptive control is analogous to the advantage of feedback controllers over open-loop control. Fixed-gain controller design is essentially an open loop endeavor unable to deal with unplanned deviations in the plant, much as open-loop control is unable to cope with any disturbances. Adaptive control, on the other hand, closes this loop to provide a control which uses input/output information to cancel out unpredicted plant changes. The potential payoffs of this capability in terms of reliability and safety clearly justify the added computational complexity associated with the method.

6.0 AIRCRAFT ADAPTIVE CONTROL EXAMPLE

In this chapter the PLID adaptive controller will be applied to a more realistic set of example dynamics. In the examples of Chapter 5, the plant was truly linear and of the same order as the assumed observable-canonical form representation within PLID. There was no constraint on the input magnitude u(k), which would clearly be limited in any practical application to a physical system, be it any input such as applied trust, torque, voltage, or actuator deflection. While the properties learned from Chapter 5 are important and form a necessary step in justifying PLID adaptive control; the ability of the algorithm to perform in less than these ideal conditions must be examined as well. The task of controlling a highly maneuverable aircraft presents a definite challenge to the control engineer even in the fixed-gain case. To apply the PLID adaptive controller to this problem will serve to demonstrate not only the practical capabilities of the algorithm, but also the importance of considering real world implementation issues in controlling physical systems.

6.1 The Aircraft Model and Baseline Controller

A continuous-time state model for a highly maneuverable production aircraft was developed using aerodynamic data from a NASA Technical Paper [22] during the author's employment with the Lockheed-Georgia Company in the Summer of 1985. The model is based on both static and dynamic wind tunnel data look-up tables valid for angle-of-attack and

sideslip angle in the ranges of $-20° < \alpha < 90°$ and $-30° < \beta < 30°$, respectively and speeds up to 0.6 Mach. The state equations of the model are the full nonlinear, six degree-of-freedom rigid-body aircraft equations derived from aerodynamic component buildup. They are complete enough to include even engine turbine gyroscopic moments, an engine dynamic model, actuator models with deflection and rate limits, automated leading-edge flap dynamics, and an altitude-air density environmental model. In all, the longitudinal, or pitch-axis model, is represented as a time-varying, nonlinear, eighth-order state model between the commanded elevator input and normal acceleration output. Here we have taken normal acceleration a_n to be

$$a_n = \frac{-\dot{w} + qu - pv}{g} \tag{6.1}$$

where u, v, w, and p, q, r are the velocities and angular rates associated with the standard X, Y, Z aircraft body-axis set and g=32.18 ft/sec^2. Such a definition implies that 0g flight (a_n=0) is a steady-state or trimmed flight condition, whereas $a_n < 0$ corresponds to a pull-up maneuver.

A very thorough set of software developed at Lockheed allows the trimming and linearization of the nonlinear state model to determine the small perturbation dynamics of the aircraft from steady-state flight conditions. From the linearized longitudinal dynamics, a linear control law was developed involving feedback of angle-of-attack, α, pitchrate, q, normal acceleration error and integrated normal acceleration error

using commanded normal acceleration as the reference input. The inclusion of the integrated error implies a zero steady-state error to step commands and the automatic retrimming (return to 0g) of the aircraft upon command release.

Feedback gains for this design were obtained from linearizations made at 11 points in the dynamic pressure range 25-550 lb/ft^2 and the results stored in linearly interpolated lookup tables as gain schedules. The design linearizations were made assuming a nominal center-of-gravity location at 0.35 of the mean aerodynamic chord of the wing. The controller functions in continuous time and time response simulations are generated using an adaptive predictor-corrector numerical integration technique. Simulations of the aircraft are run on a Digital Equipment Corporation VAX 11/780 using extensive modeling software tying the controller, gain schedules, aircraft equations, and aerodynamic lookups together.

6.2 Implementation of the Adaptive Controller

The baseline controller design described above is of a form that can loosely be termed "open-loop" adaptive. The feedback gains are varied based on the measured dynamic pressure to provide proper control throughout the speed-altitude envelope. The manner in which the gains are related to dynamic pressure is fixed in the gain schedules even if the gains themselves are not. Thus, the baseline controller is a fixed design which is not capable of adjusting to any change other than dynamic pressure, and even then, only in a preset fashion.

The application of true adaptive control to replace the gain scheduled design is done with several complementary goals in mind. Clearly, an adaptive controller would have the ability to deal with unpredicted changes in the aircraft dynamics, which might result from major center-of-gravity shifts, loss of actuator surface effectiveness (e.g., from battle damage), frozen flaps, and the like. The adaptive controller could also replace the gain scheduling function by adaptively compensating for the effect of changing dynamic pressure, thus reducing the complexity of the controller. This would clearly eliminate the reliance upon a dynamic pressure sensor which is susceptable to blockage by debris or ice.

The unique structure of the PLID adaptive controller requires only the measurement of normal acceleration which can be reliably measured with an accelerometer, thus eliminating the need to measure angle-of-attack for feedback. Accurate measurement of angle-of-attack, like dynamic pressure, requires the use of pitot tube based measurements of the airflow about the aircraft. This is objectionable from a radar observability standpoint in military applications and from a reliability standpoint in the case of tube blockage. Previous non-adaptive attempts to eliminate angle-of-attack feedback using dynamic output feedback principles resulted in complex gain scheduling in this application [23]. Application of the PLID adaptive controller to this problem, thus, has the potential benefits of adaptation to unpredicted aircraft changes, gain-schedule elimination, and a reduction in the number of necessary sensors.

The first step in the implementation of the adaptive controller is to specify the input and output points completely and to outline the architecture of the overall closed-loop system. The block diagram of Figure 6.1 illustrates the relationship of the system components to one another. The plant is considered to be the continuous-time dynamic model between the incremental commanded elevator deflection and normal acceleration. The term "incremental" refers to the fact that an inner pitchrate feedback loop is added to the adaptive control input to compute the elevator deflection command signal. This inner loop is maintained to stabilize the open-loop plant by adding a damping effect. This is especially important in holding the aircraft within a linear range of operation during the initial learning process; the natural unstable dynamics quickly diverge into nonlinear oscillation during the learning interval making identification impossible. The gain, G, on this loop is scheduled with dynamic pressure in the examples using the gain schedule for this feedback from the nominal baseline controller. It is believed that later refinements of the design could eliminate the scheduling and hold the q feedback gain at a fixed value quite easily.

The remainder of the block diagram, outside the plant boundary, is identical to the adaptive control structure of the previous chapter. Through the sampling and zero-order-hold operations at the 0.01 second sample interval, the continuous dynamics of the plant are identified and controlled in the discrete-time domain.

We are, therefore, seeking to identify a linear discrete-time model for the eighth-order, nonlinear, continuous-time plant. Clearly, the

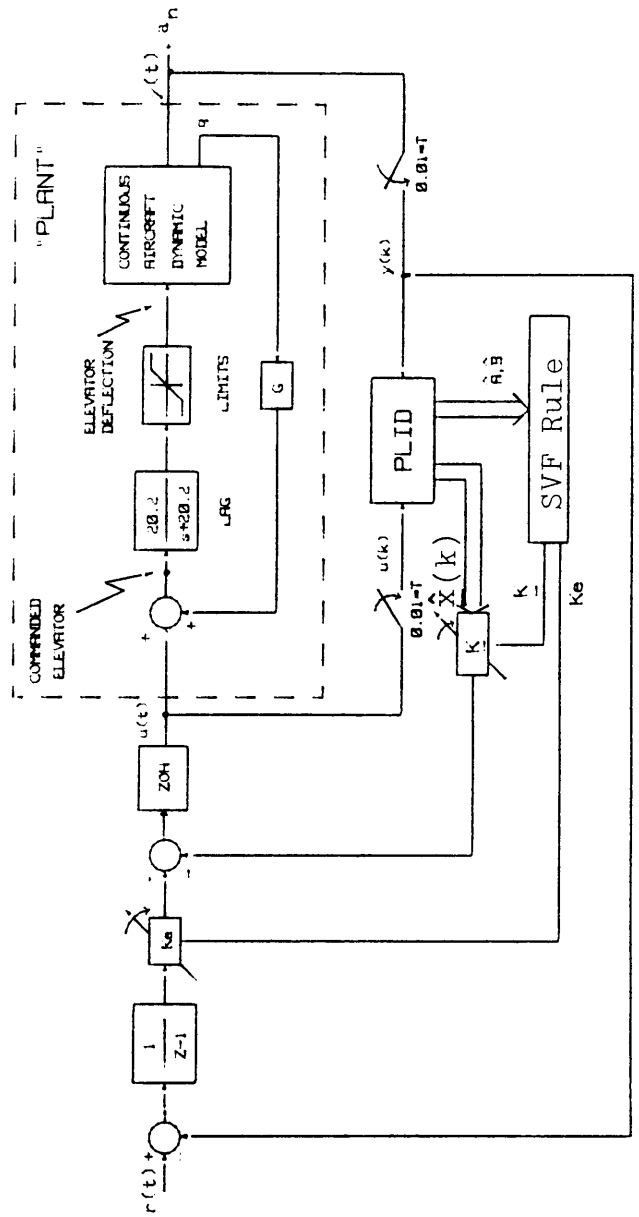

Figure 6.1 Adaptive Aircraft Control

proper choice of order for the identified model will be crucial to the performance of the identification algorithm and the resulting controller [7]. For insight into the approximate system order, a linearization of the airframe dynamics (with only the q feedback engaged) is made about a trim condition of 502 ft/sec airspeed at sea level, corresponding to a dynamic pressure of 30 lb/ft^2. This condition is roughly at the middle of the operational envelope of the model. The poles and zeros of the linearization contained four strong cancellations, leaving the following fourth-order continuous-time transfer function between commanded elevator and normal acceleration:

$$H(s) \approx \frac{-0.1025(s+0.00394)(s+10.1224)(s-12.7113)}{(s-0.01411)(s+12.253\pm1j)(s+0.8488)} \qquad (6.2)$$

A $z=e^{sT}$ mapping leads to the discrete time transfer function:

$$H(z) = \frac{-0.1025(z-099996)(z-0.9037)(z-1.1336)}{(z-1.0001)(z-0.8846\pm0.00885j)(z-0.99155)} \qquad (6.3)$$

which can be reduced to either

$$H(z) \approx \frac{-0.1025(z-1.1356)}{(z-0.88454)(z-0.99155)} \quad \text{or} \quad \frac{-0.1025(z-1.1356)}{(z-0.88454\pm0.0089j)} \qquad (6.4)$$

depending upon the cancellation choice for the zero at 0.9037. The second option is recognized as the model upon which Plant F in Chapter 5 was based. In either case, it is clear that the system is dominantly second-order with a strong non-minimum-phase zero. Thus the model

assumed within PLID for adaptively controlling the aircraft will be of order two. Any higher order assumption faces the task of trying to identify nearly cancelling pole-zero pairs which severely slows convergence of typical identification algorithms [24] (including PLID).

The choice of desired closed-loop poles is made to satisfy the Military Specifications for Flying Qualities of Piloted Airplanes [25] bounds on longitudinal natural frequency and damping. We take our second-order approximation to represent the so called short-period longitudinal approximation and therefore seek to place two of our closed-loop poles at $s=-4\pm 2j$ and the third pole in our overall third-order closed-loop system at $s=-10$. Mapping into the z-plane, and subsequent multiplication, results in the desired closed-loop polynomial

$$P_{CL}(z) = z^3-2.826033z^2 + 2.661487z-0.835271 \qquad (6.5)$$

which will be employed in our adaptive aircraft controller.

Within subroutine CA (listed in the Software Appendix), the feedback logic is set to require positive gain on the state feedbacks and negative gain on the z feedback to provide an overall positive error gain with negative state feedback after the implied negation seen in Figure 6.1. An additional subroutine D listed in the Software Appendix provides the link between the continuous-time simulation software and the discrete-time adaptive control procedure. Subroutine D also initializes the controller parameters upon the first call at time 0.0 to

q= 0.01, r=1.E-8, P-diag(1.), a fade factor of 1.001, and initial parameter guesses reflecting poles near the unit circle and the expected non-minimum-phase zero. It is also necessary to intialize z to a value which will lead to the proper trim elevator setting based on the computed feedbacks for the initial parameter guess.

Subroutine D is called by the continuous-time simulation routine every 0.01 seconds to provide the discrete-time control input based on the adaptive control operations. Upon each call, Subroutine D samples the normal acceleration output, updates the parameter and state estimates by PLID, and then applies Subroutine CA to compute the new control input. Logic contained in Subroutine D then performs input limiting to insure that the commanded elevator input from both the inner loop q-feedback and adaptive control input remains within the allowable range of ±25°. Should the command exceed these limits, the adaptive control input is reduced accordingly and error integration is prevented from overaccumulating during the saturation. The resulting command input is then passed to the continuous-time simulation where it is held constant in ZOH fashion until the next sample 0.01 seconds later. The commanded elevator input is processed through an actuator model in the continuous-time plant which realistically models the lag of the elevator to commands, as well as deflection and rate-of-travel limiting. The control simulations are thus produced from continuous-time modeling software which calls the discrete-time control algorithm through Subroutine D at the appropriate sampling intervals in much the same way as the algorithm would be implemented in practice on an actual aircraft.

This section has served to demonstrate many of the practical aspects that must be considered in applying adaptive control to a physical, continuous-time plant. It is hoped that the reader will appreciate the problem-specific nature of these considerations and develop a feel for how other systems might be similarly approached. We have addressed the issues of order choice, input limiting, initialization, and parameter settings which serve to bridge the gap between the rather academic examples of Chapter 5 and more realistic situations.

6.3 Simulation Conditions and Results

The simulations to follow are started from wings-level forward flight at a dynamic pressure of 300 lb/ft^2, as described earlier. The full six degree-of-freedom, nonlinear model is used throughout with the lateral axes held invariant by a baseline, continuous-time, lateral control system. It is important to emphasize that reduced-order linearizations are not being used to produce the examples as is often done in the literature, but rather the controller is being applied to a simulation model as accurate as possible in faithfully representing the true aircraft.

For direct comparison, the prescribed maneuvers are first flown (in simulation) using the baseline gain-scheduled controller operating in continuous time. This controller is then replaced by the discrete-time PLID adaptive controller and the maneuver reflown from the same reference input sequence. A review of the simulation output traces will demonstrate the ability of the PLID adaptive controller to adapt to

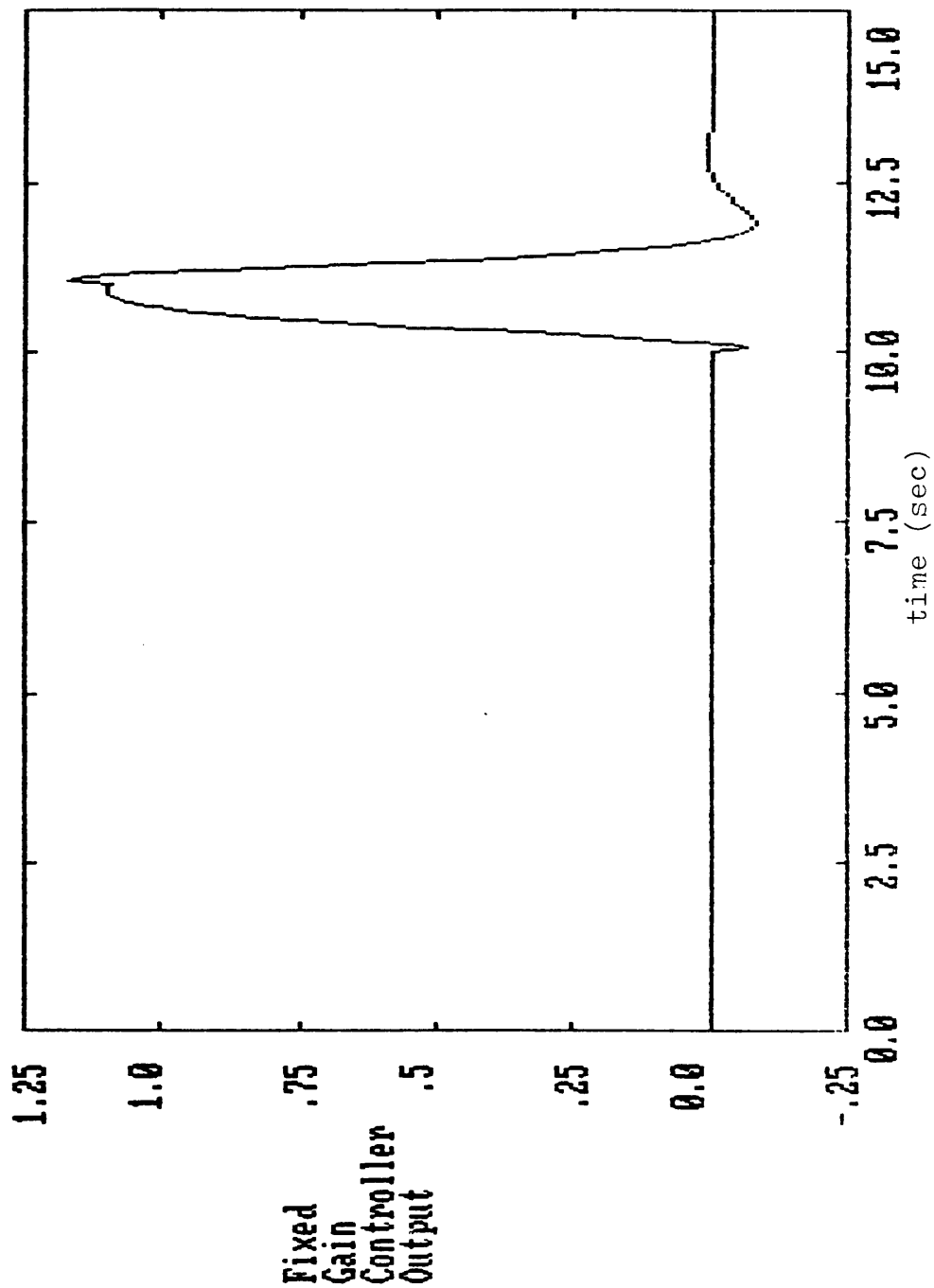

Figure 6.2 Fixed Aircraft Controller Output, Case 1

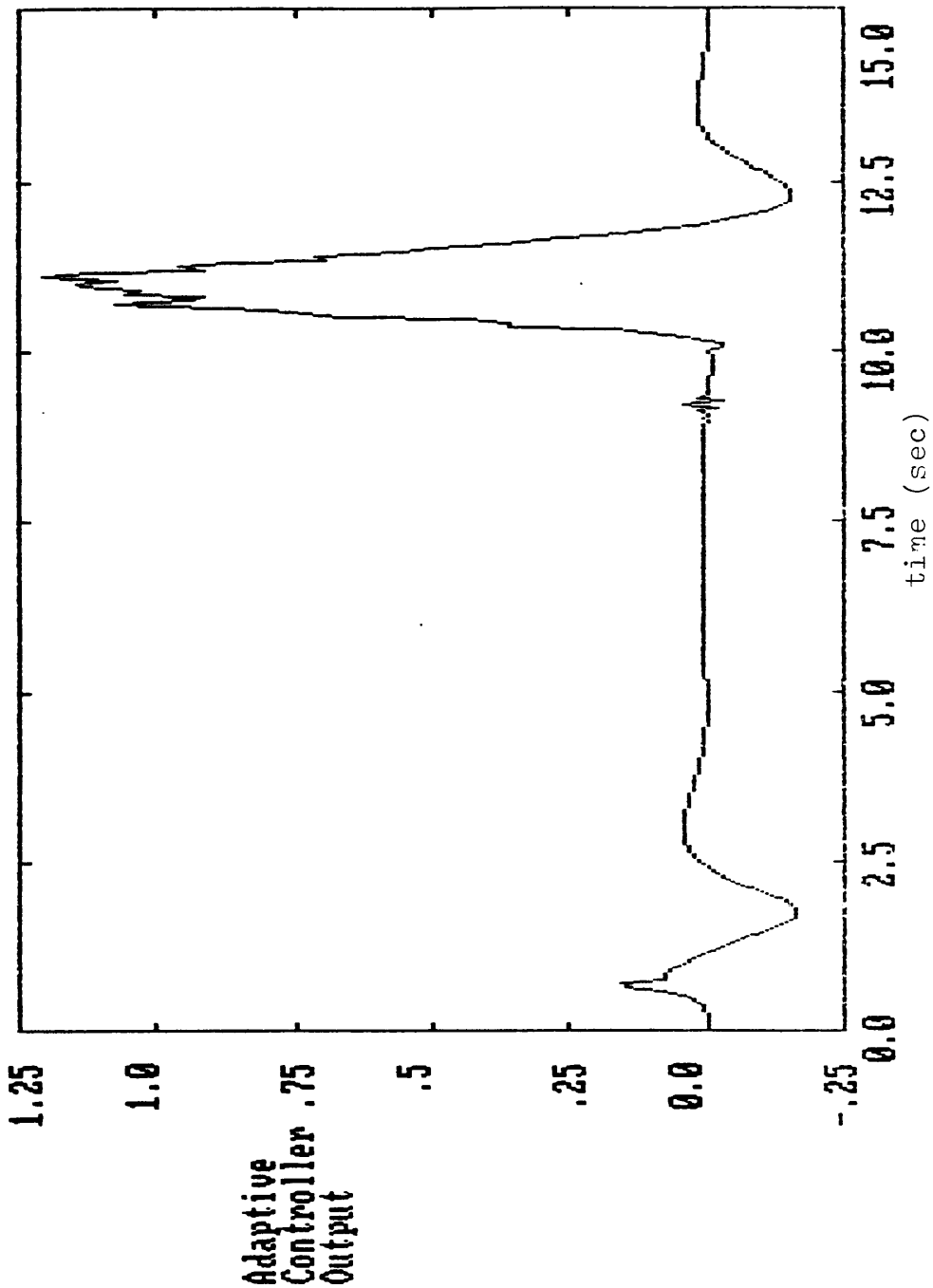

Figure 6.3 Adaptive Aircraft Controller Output, Case 1

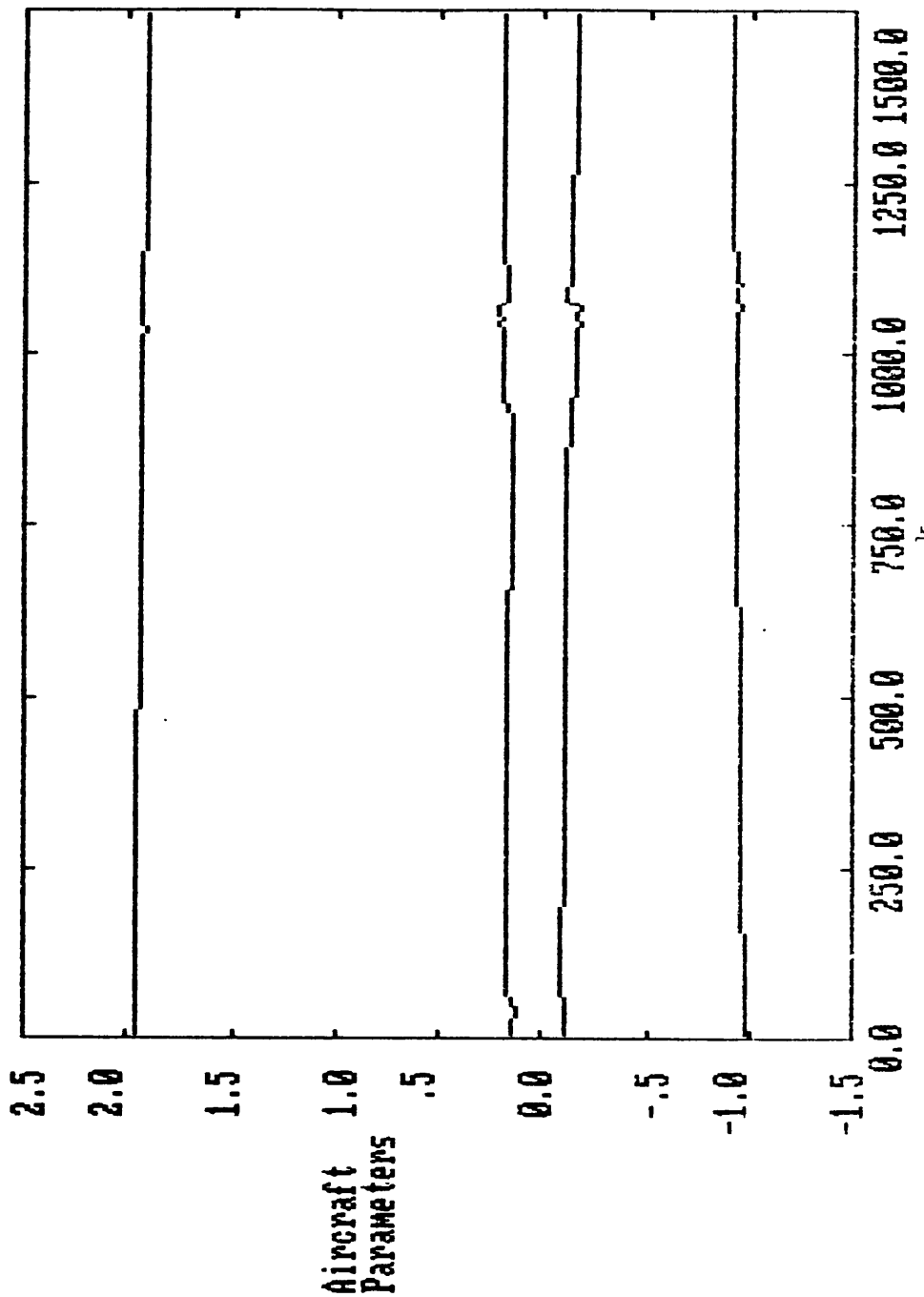

Figure 6.4 Identified Parameters, Aircraft Case 1

realistic plant changes and to improve upon fixed-gain controllers, even when limited to a reduced-order approximate model.

The first maneuver is a simple 1g pull-up of one second duration and a return to trimmed, but now climbing, flight. The climb entry will be accompanied by a significant loss of airspeed unless additional thrust is provided. In recognition of this effect a simple "bang-bang" autothrottle is employed to maintain an airspeed near the initial 502 ft/sec value. This autothrottle will remain in effect throughout all the examples to follow.

Figure 6.2 shows the normal acceleration response to a 1g reference input initiated at 10 sec. and released at 11 sec. using the baseline gain-scheduled controller. For comparison, Figure 6.3 shows the response of the adaptive controller to the same reference input pulse. The transient condition in the initial 5 seconds indicates the learning interval during which the parameter estimates of Figure 6.4 are seen to reach a steady state. The aircraft emerges from this transient in a slight climb, unfortunately accompanied by a loss of airspeed. Since the adaptive controller is not gain scheduled to compensate for this effect, a small disturbance at 9 seconds arises as the controller adapts to the changing flight condition. The transient response to the reference input pulse is quite similar to that of the baseline controller. More undershoot at the 12 second mark is evident, as is a high frequency component in the peak of the response.

This high frequency component is a result of a difficult identification situation. Any pull-up transient is in itself a change in the

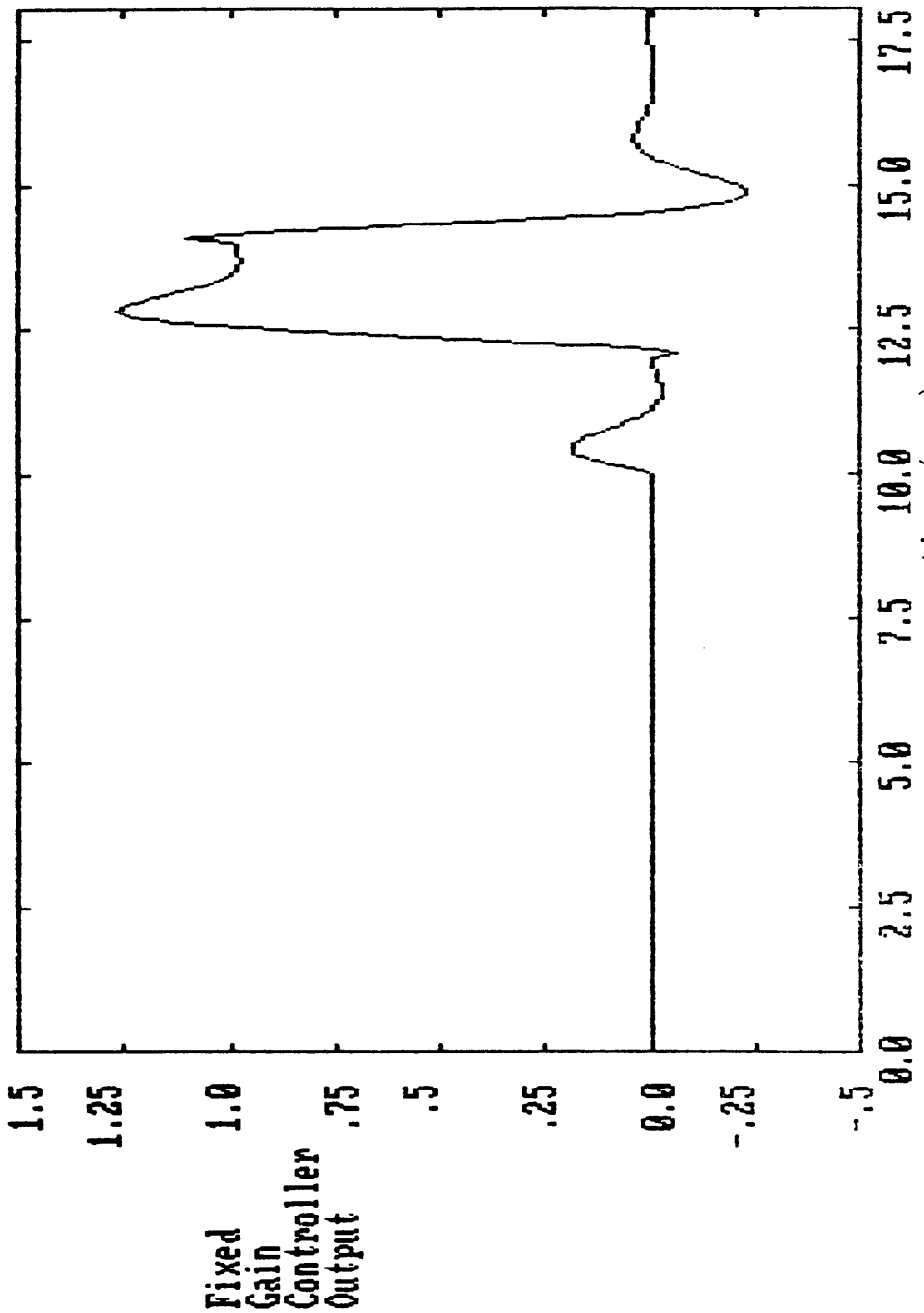

Figure 6.5 Fixed Aircraft Controller Output, Case 2

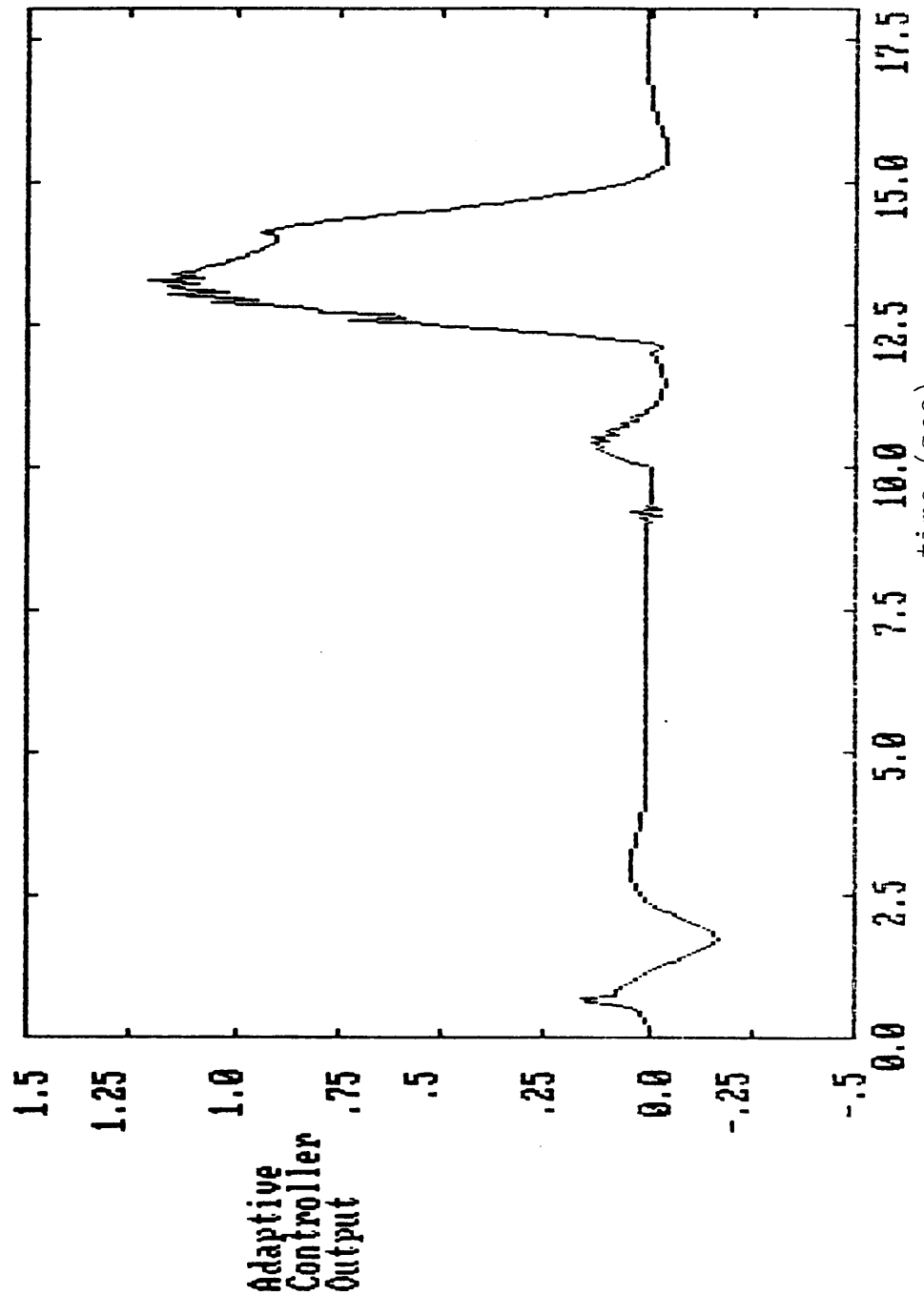

Figure 6.6 Adaptive Aircraft Controller Output, Case 2

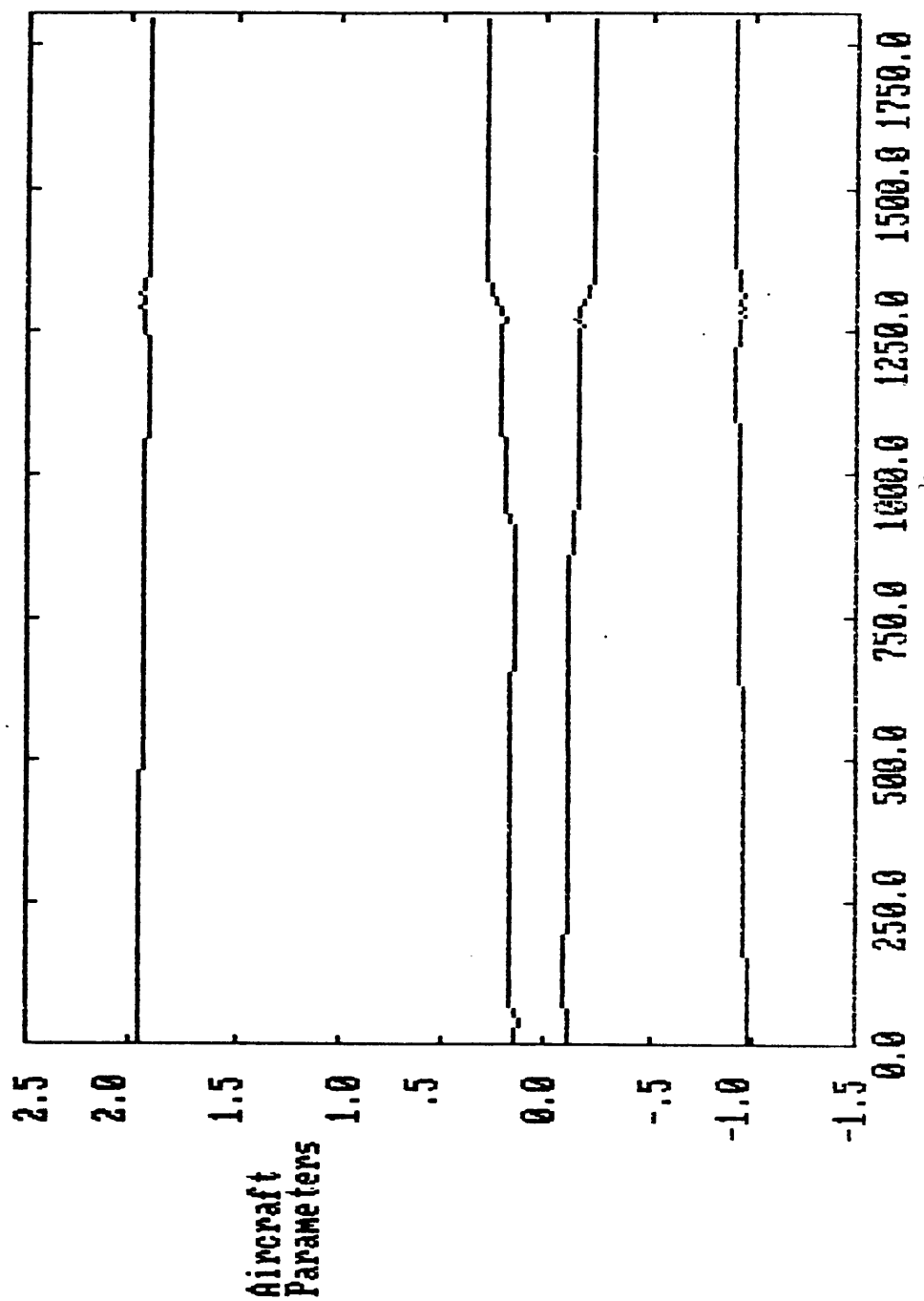

Figure 6.7 Identified Parameters, Aircraft Case 2

plant dynamics as seen in Figure 6.4. Thus, the controller is not only managing the pull up, but learning and changing gains simultaneously. The high frequency "noise" on the normal acceleration response is generated by abnormally high actuator activity (flutter) induced by the rapidly changing feedback gains. On a more positive note, however, we see that the aircraft returns to a trimmed flight condition and the parameters reach a definite steady state soon after the maneuver.

The second maneuver is designed to demonstrate the ability of the adaptive controller to adapt to unpredicted changes in the aircraft dynamics. At the 10 second mark, the center-of-gravity (c.g.) is abruptly shifted from 0.35 to 0.4 of the mean aerodynamic chord of the wing. This more aft center-of-gravity location corresponds to a more unstable aircraft; this could arise from a weapons release or a shifting of cargo. This change is followed by a 1g pull-up initiated at 12 seconds and released at 14 seconds. In Figure 6.5 we see how the baseline controller (designed for 0.35 c.g. location) response is significantly degraded by the shift. The shift is immediately followed by a 0.2g disturbance which is satisfactorily cancelled by the controller, but the pull-up response shows significantly more overshoot and undershoot than found in the first maneuver.

In contrast, we see significant improvement in the adaptive control responses of Figure 6.6. For this maneuver, the disturbance after the c.g. shift is smaller, the overshoot of the pull up is less and the undershoot after the maneuver is insignificant. The adaptation to the c.g. shift seen in the parameter traces of Figure 6.7 is sufficient to

provide a pull-up response having almost the same overall shape as the response to the pull up in the previous example with no c.g. shift. In addition, we note that midway through the pull-up the controller has learned enough about the new conditions to give a smooth response exhibiting no flutter for the remainder of the maneuver. The return to trimmed flight at 15 seconds exhibits almost no undershoot and its smoothness indicates that the parameter estimates have reached a comfortable steady state. The overshoots found in the nominal controller responses would be highly objectionable to a pilot, especially during the high precision maneuvers required during air-to-air missile targeting or refueling.

The pilot would almost certainly prefer the slight flutter in the pull-up of the adaptive controller to the relatively sloppy responses of the baseline controller in the shifted c.g. situation. The flutter in the trace is of such high frequency and of such magnitude (on the order of 0.05g) that it is very doubtful that the pilot would even feel its effect. Any objection to this phenomenon would arise out of concern that the high actuator activity would wear out the actuator servos or excite high frequency structural modes that would accelerate material fatigue. The scope of this work, however, is to demonstrate the application of PLID adaptive control to a realistic system and point out the many practical considerations for proper application of the method. Completely optimizing the design to accommodate this problem-specific characteristic would be tangential to our purposes. It is, however, likely that with proper filtering of the actuator command signal, or

additional feedback constraint logic, this phenomenon could be avoided entirely.

In the final example, we will demonstrate the ability of the adaptive controller to replace the dynamic pressure gain-schedule function. In Figure 6.8, we see the baseline controller response to a 2g pull up from 14-16 seconds following the c.g. shift to 0.4 at 12 seconds and a full-throttle command at 10 seconds. We see again the significant overshoot stemming from the c.g. shift, but gain scheduling has compensated for the increase in thrust and airspeed.

Considering the rapidly changing nature of the plant due to the airspeed and c.g. changes, the adaptive control response of Figures 6.9 and 6.10 again closely resembles the original shape seen in Figure 6.3. Slightly more undershoot at 17 seconds is evident, but the flutter in this region indicates that complete learning has not yet occurred. There seems to be more difficulty in dealing with the c.g. shift since it follows so closely behind the airspeed increase, but overall, the adaptive controller is able to meet its goals. The airspeed change is compensated for without the cumbersome gain schedules. Further, the fact that the c.g. shift is accommodated at all is an improvement over the baseline controller.

Additional comparisons pushing the aircraft performance to its limits under adaptive control have demonstrated its ability to cope with very wide parameter swings. Pull-ups of 4-5g held through complete inside and outside vertical loops, and vertical climbs to high speed, high altitude flight that effectively span the entire operational

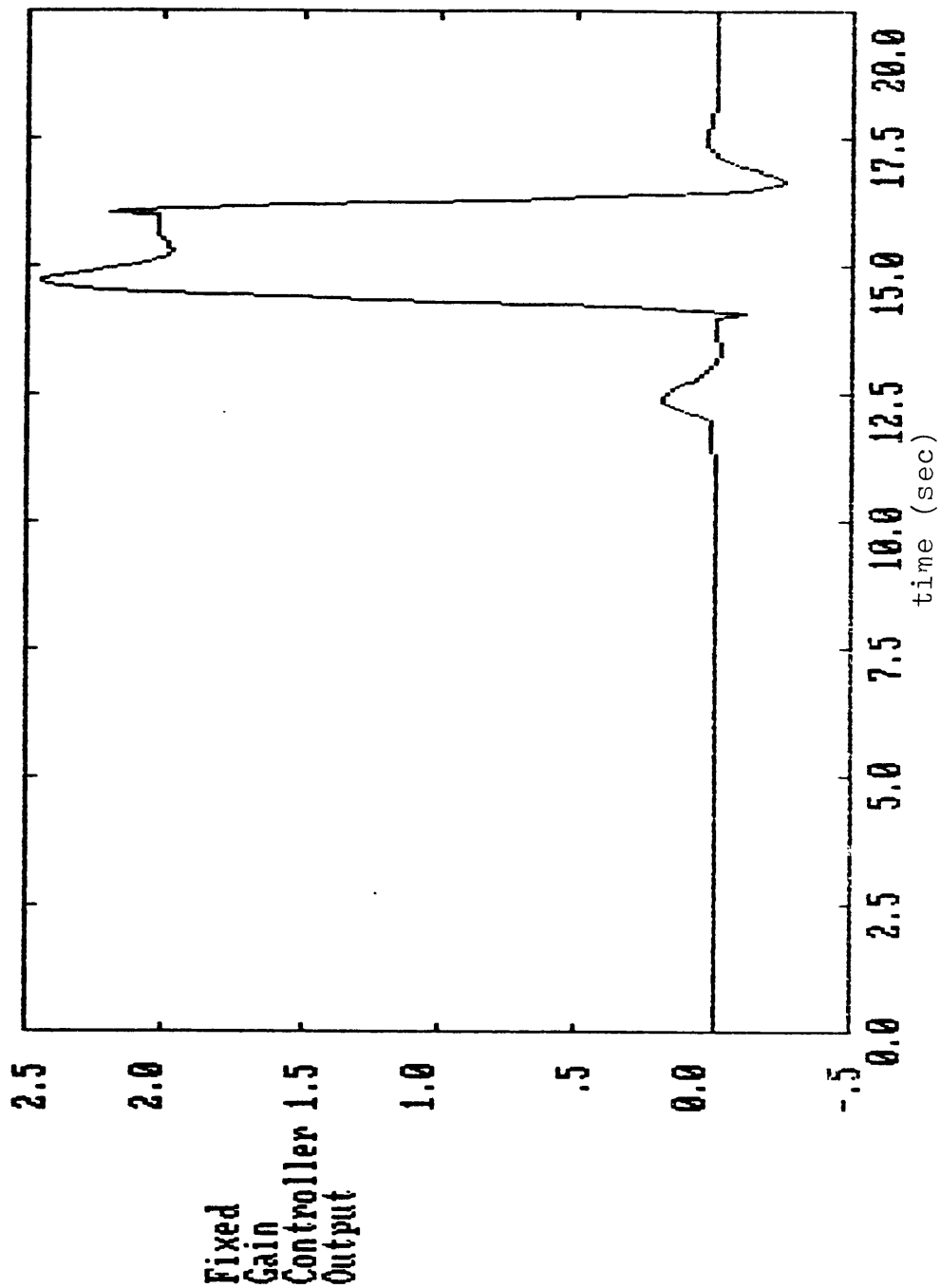

Figure 6.8 Fixed Aircraft Controller Output, Case 3

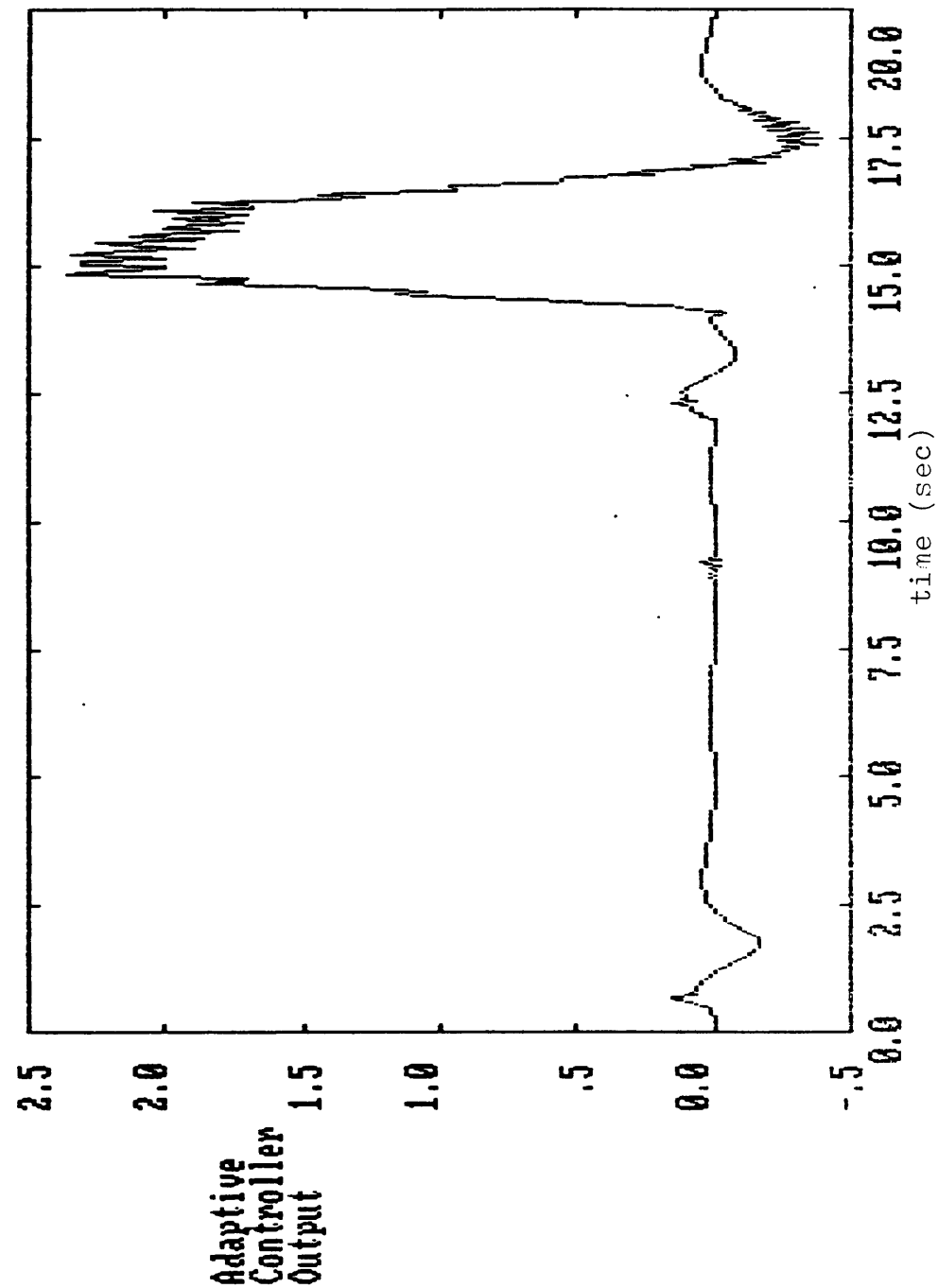

Figure 6.9 Adaptive Aircraft Controller Output, Case 3

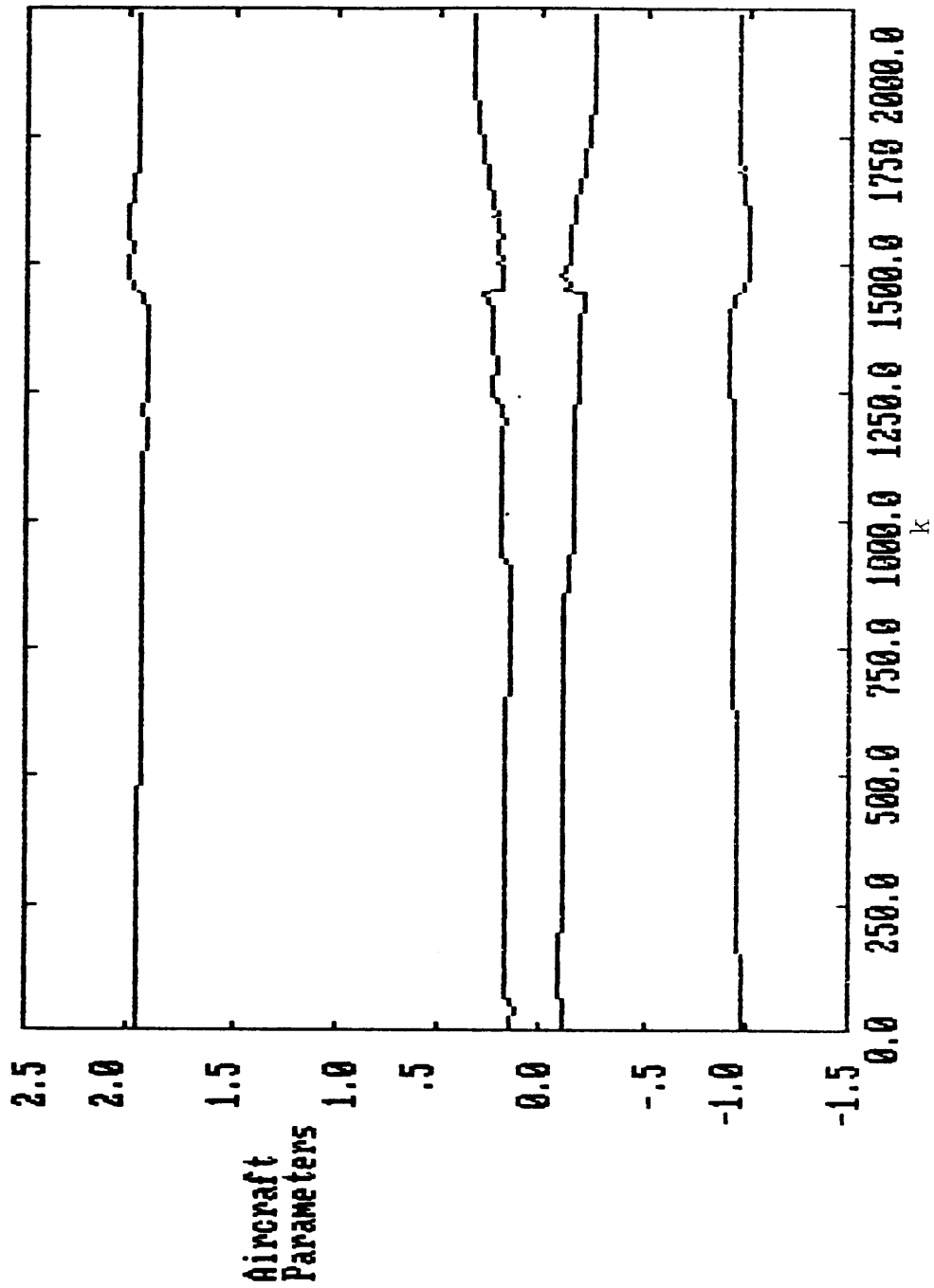

Figure 6.10 Identified Parameters, Aircraft Case 3

envelope have been simulated using the adaptive controller with no degradation in performance compared with the baseline controller. This demonstration of long-term stability in the face of rapid parameter changes, when combined with the improvement over the baseline seen in the c.g. shift example, clearly makes a strong case in favor of adaptive aircraft control. More importantly, for the purpose of this effort, these examples have demonstrated the successful and effective application of the PLID adaptive controller to a realistic set of physical dynamics. It is also hoped that an appreciation for the problems of practical implementation has been gained by the reader as well.

7.0 NUMERICAL CONSIDERATIONS FOR ALGORITHM STABILITY

This chapter will again apply techniques found in the Kalman filtering literature to improve the stability properties of the PLID algorithm. While none of the simulations made with the PLID method have exhibited any stability problems, the conventional Kalman filter formulation used is known to be sensitive to many sources of instability (divergence). Among these are unmodeled dynamics, the presence of nonlinearities, and the effects of computer roundoff [26]. While unmodeled dynamics and nonlinearities are generally unavoidable, as seen in the aircraft example, reformulation of the algorithm can greatly improve its sensitivity to roundoff errors.

A number of ad-hoc strategies for dealing with roundoff have been proposed involving forced symmetry of the covariance matrix, covariance resetting, or the use of artificially high process and measurement noise parameters. All of these methods seek to maintain symmetry and positive definiteness of the covariance matrix since numerical divergence of the algorithm most often manifests itself in the loss of these properties [26].

Gerald J. Bierman's work in matrix factorization suggests that methods employing square-root matrices to preserve symmetry and positive definiteness are "far superior" [26] to the methods listed above. The potential payoffs of the square-root reformulation are the stability afforded by reducing roundoff sensitivity, and the subsequent reduction in computational precision (word length) required to maintain a given

degree of accuracy. The following development will apply Bierman's factorization techniques to the PLID algorithm in order to improve its stability characteristics.

7.1 The UDU^T Kalman Filter

Under the assumption that the covariance matrix P in the Kalman filter is to be symmetric and positive definite (positive eigenvalues), matrix theory guarantees that P can be expressed as

$$P = SS^T \tag{7.1}$$

where S is an upper-triangular matrix. Conversely, the product SS^T where S is an upper-triangular matrix is always symmetric and will be positive definite if S has no zeros on its diagonal. S may be further subdivided as

$$S = UD^{\frac{1}{2}} \tag{7.2}$$

where U is unit upper-triangular (defined as having 1's on the main diagonal) and $D^{\frac{1}{2}}$ is a diagonal matrix. Clearly, then [26]

$$P = UDU^T \tag{7.3}$$

where D is a diagonal matrix of the eigenvalues of P. If D has all its

elements greater than zero, P is guaranteed to be symmetric and positive definite by Equation (7.3).

The reformulation of the Kalman filter takes advantage of the relationship of (7.3) by directly updating and processing U and D throughout. Direct operations upon U and D to accomplish the Kalman filter updates guarantee that the represented P matrix maintains the symmetry and positive definiteness necessary for stability. To overview the reformulation, recall the basic Kalman filter equations

$$M(k) = F(k-1)P(k-1)F^T(k-1)+Q(k-1) \tag{7.4a}$$

$$K(k) = M(k)H^T(k)/[H(k)M(k)H^T(k)+r(k)] \tag{7.4b}$$

$$P(k) = [I-K(k)H(k)]M(k) \tag{7.4c}$$

$$\underline{s}(k) = F(k-1)\underline{s}(k-1) + K(k)[y(k) - H(k)F(k-1)\underline{s}(k-1)] \tag{7.4d}$$

The update of the covariance matrix from P(k-1) clearly takes place in two stages. First, M(K) is found from P(k-1) in Equation (7.4a); this is referred to as the "time update" since it depends on the state transition matrix of the system, F(k-1). M(k) is then used in the "measurement update" Equations (7.4b and c) to compute P(k). Bierman has developed two separate algorithms to perform these different updates directly using the U-D factors of the respective matrices. The theoretical derivation of these update techniques will not be reproduced here; only a heuristic discussion of the application of the methods to our problem will be presented.

The U-D time-update algorithm takes the factors U(k-1) and D(k-1) corresponding to

$$P(k-1) = U(k-1)D(k-1)U^T(k-1) \tag{7.5}$$

and by direct operation produces $\tilde{U}$ and $\tilde{D}$, corresponding to

$$M(K) = \tilde{U}\,\tilde{D}\,\tilde{U}^T \tag{7.6}$$

By direct operation it is meant that at no point in the computation is the product of Equation (7.5) computed, thus leaving the relationship and its desirable properties implicit. The FORTRAN mechanization of the time-update algorithm [26] also leaves the "ones" on the U diagonal implicit by storing the elements of D on the diagonal of U. Unfortunately, the FORTRAN code presented by Bierman in [26] contains several typographical errors. Use of the supplemental reference [27] of Bierman and Nead (and some rederivation) led to the correct coding found in Subroutine UDU in the Software Appendix to this work.

The measurement update mechanization similarly uses the intermediate $\tilde{U}$-$\tilde{D}$ factors from the time update to directly determine the U(k)-D(k) factors of P(k). Unfortunately, this mechanization does not permit r to be zero. Thus, to maintain compatibility with our previous work when perfect measurements are assumed, a specified r of zero is replaced in Subroutine UDU by 1.E-99. On initialization of the new PLID algorithm, the large diagonal elements of P are simply the elements of

D, and U is taken to be an identity matrix. Since the D elements are assumed stored on the U diagonals by Bierman's mechanization, the storage elements denoted by P in the PLID subroutine are occupied by U and D in the UDU subroutine. In short, the UDU subroutine is completely interchangeable with the PLID subroutine in all the simulation and adaptive control software with no changes necessary in the initialization or storage procedures.

7.2 Fading Memory UDU^T Filter Modification

Inclusion of the fading memory properties of Chapter 4 into the UDU structure is somewhat complicated. Recall that the following equations were used to implement the block fading factor:

$$M(k) = \begin{bmatrix} \Phi_{11}(k) & \vdots & \Phi_{12}(k) \\ \cdots & \vdots & \cdots \\ \Phi_{21}(k) & \vdots & \alpha\Phi_{22}(k) \end{bmatrix} + Q(k), \quad \alpha \geq 1 \tag{7.7}$$

where

$$\begin{bmatrix} \Phi_{11}(k) & \vdots & \Phi_{12}(k) \\ \cdots & \vdots & \cdots \\ \Phi_{21}(k) & \vdots & \Phi_{22}(k) \end{bmatrix} = F(k-1)P(k-1)F^T(k-1) \tag{7.8}$$

Since Q(k) is only nonzero in the upper left n x n block, we may equivalently consider, dropping some k arguments as well,

$$M(k) = \begin{bmatrix} \Phi_{11} & \vdots & \Phi_{12} \\ \cdots & \vdots & \cdots \\ \Phi_{21} & \vdots & \alpha\Phi_{22} \end{bmatrix} \tag{7.9}$$

where

$$\Phi = \left[\begin{array}{c:c} \Phi_{11} & \Phi_{12} \\ \hdashline \Phi_{21} & \Phi_{22} \end{array}\right] = F\ P\ F^T + Q \tag{7.10}$$

Now, the time update of the U(k-1)-D(k-1) factors yield $\tilde{U}$ and $\tilde{D}$ such that

$$\Phi = \tilde{U}\ \tilde{D}\ \tilde{U}^T \tag{7.11}$$

where Φ is defined in Equation (7.10)

The difficulty lies in determining how to modify $\tilde{U}$ and $\tilde{D}$ to accomplish the effect of block fading. One might naively compute the product of Equation (7.11) to obtain Φ, perform the block α modification and then determine the U-D factors of the result by a standard factorization algorithm. However, forming the product of Equation (7.11) defeats the direct computation goal. We therefore, seek direct manipulation of $\tilde{U}$ and $\tilde{D}$ to accomplish the block fade.

Consider that

$$\Phi = \tilde{U}\ \tilde{D}\ \tilde{U}^T = \left[\begin{array}{c:c} U_1 & S \\ \hdashline 0 & V_2 \end{array}\right] \left[\begin{array}{c:c} D_1 & 0 \\ \hdashline 0 & D_2 \end{array}\right] \left[\begin{array}{c:c} U_1^T & 0 \\ \hdashline S^T & V_2^T \end{array}\right] \tag{7.12}$$

where the matrices U_1 and U_2 are unit upper-triangular, D_1 and D_2 are diagonal, and S is rectangular such that U_1 is n x n, U_2 is 2n x 2n, and S is n x 2n. Multiplication then implies that

$$\begin{bmatrix} U_1 D_1 U_1^T + S D_2 S^T & \vdots & S D_2 V_2^T \\ \cdots\cdots & \vdots & \cdots\cdots \\ V_2 D_2 S^T & \vdots & U_2 D_2 U_2^T \end{bmatrix} = \begin{bmatrix} \Phi_{11} & \vdots & \Phi_{12} \\ \cdots & \vdots & \cdots \\ \Phi_{21} & \vdots & \Phi_{22} \end{bmatrix} \tag{7.13}$$

To achieve the fading factor effect, we would like the following to hold, see (7.9):

$$\begin{aligned} M'_{22} &= \alpha \ \Phi_{22} \\ M'_{12} &= \Phi_{12} \\ M'_{11} &= \Phi_{11} \end{aligned} \tag{7.14}$$

Such that

$$M'(k) = \begin{bmatrix} M'_{11} & \vdots & M'_{12} \\ \cdots\cdots & \vdots & \cdots\cdots \\ (M'_{12})^T & \vdots & M'_{22} \end{bmatrix} \tag{7.15}$$

We therefore seek S', D', D_2' such that

$$\begin{bmatrix} U_1 & \vdots & S' \\ \cdots & \vdots & \cdots \\ 0 & \vdots & V_2 \end{bmatrix} \begin{bmatrix} D'_1 & \vdots & 0 \\ \cdots & \vdots & \cdots \\ 0 & \vdots & D'_2 \end{bmatrix} \begin{bmatrix} U_1^T & \vdots & 0 \\ \cdots & \vdots & \cdots \\ (S')^T & \vdots & U_2^T \end{bmatrix} = M'(k) \tag{7.16}$$

Equating Equations (7.16) and (7.13), subject to (7.14), implies the following:

$$U_2 D_2' U_2^T = \alpha \ U_2 D_2 U_2^T \tag{7.17a}$$

$$S' D'_2 U_2 T = S \ D_2 U_2^T \tag{7.17b}$$

$$U_1 D'_1 U_1^T + S' D'_2 (S')^T = U_1 D_1 U_1^T + S D_2 S^T \tag{7.17c}$$

Solution of (7.17) leads to the modification equations

$$D'_2 = \alpha D_2 \tag{7.18a}$$

$$S' = S/\alpha \tag{7.18b}$$

$$D'_1 = D_1 + [U_1^{-1}] S [D_2 - D_2/\alpha] S^T [U_1^{-1}]^T \tag{7.18c}$$

which define D_1', D_2' and S' in terms of the blocks of $\tilde{U}$ and $\tilde{D}$ from the time update and α. Clearly, and as required, $\alpha = 1$ implies that $\tilde{U}$ and $\tilde{D}$ are unchanged.

Unfortunately, D_1' is not necessarily diagonal, thereby requiring us to factorize it using Bierman's U-D factorization algorithm [26] such that U and $\bar{D}$ are found to satisfy

$$U \bar{D} U^T = \begin{bmatrix} D'_1 & \vdots & 0 \\ \cdots & \vdots & \cdots \\ 0 & \vdots & D'_2 \end{bmatrix} \tag{7.19}$$

Then defining

$$\bar{U} = \begin{bmatrix} U_1 & \vdots & S' \\ \cdots & \vdots & \cdots \\ 0 & \vdots & U_2 \end{bmatrix} U \tag{7.20}$$

yields the desired result

$$M'(k) = \overline{U}\overline{D}\overline{U}^T$$

where $\overline{U}$ and $\overline{D}$ are the block-faded modifications of the update factors $\tilde{U}$-$\tilde{D}$.

The overall progression of Subroutine UDU is thus as follows:

U(k-1), D(k-1)	⟶	$\tilde{U}$, $\tilde{D}$	Bierman Time Update
$\tilde{U}$, $\tilde{D}$	⟶	$\overline{U}$, $\overline{D}$	Eqns. (7.12, 18, 19, 20)
$\overline{U}$, $\overline{D}$	⟶	U(k),D(k)	Bierman Measurement Update

The listing of this subroutine in the Software Appendix designates these three main portions. Note that if α=1 the fading-memory steps are simply bypassed, since in this case $\tilde{U} = \overline{U}$ and $\tilde{D} = \overline{D}$.

It is arguable whether the complexity of the block fade as carried out above is indeed preferable to the naive multiplication-fade-factorization approach, especially since it requires inversion of the matrix U_1. The fact that U_1 is only n x n and unit upper-triangular does guarantee that the inversion is well conditioned and of expected high accuracy. In any event, we have made every possible effort through this approach to avoid forming explicit UDU^T products in the computations to insure results of the highest possible integrity. Certainly, in a situation where such integrity is not deemed worth the extra effort, the simpler approach could be applied with confidence.

7.3 Simulation and Results

In this section we will simply replace the PLID subroutine with Subroutine UDU and reproduce a previous simulation to demonstrate equivalence. In particular, Figure 7.1 repeats the experiment of Figure 4.4, using the UDU^T algorithm instead. Recall that this is a deterministic case of a plant with two step-changes in the parameters. We see that the UDU^T algorithm gives identical results to the PLID algorithm, even in this difficult case of parameter tracking.

All of the simulations presented in this work were performed using FORTRAN double-precision arithmetic. As mentioned before, no stability problems with the Kalman filter of PLID have been encountered. Bierman suggests that a rule-of-thumb for his UDU^T based filter is that the algorithm provides the same accuracy as the basic Kalman filter using twice as large a word length [26]. Thus, relying on the truth of this statement, we could expect to use the UDU^T based filter in single precision and obtain just as much accuracy and stability as our standard PLID algorithm running in double precision. The savings in memory and computational speed afforded by this change would certainly be valuable in implementing PLID adaptive control on physical systems. The memory and speed limitations of most on-board flight control computers, for example, would likely preclude the use of the double-precision PLID adaptive controller on aircraft. Applying Bierman's UDU^T based algorithm in single precision is feasible and much more practical in such realistic situations.

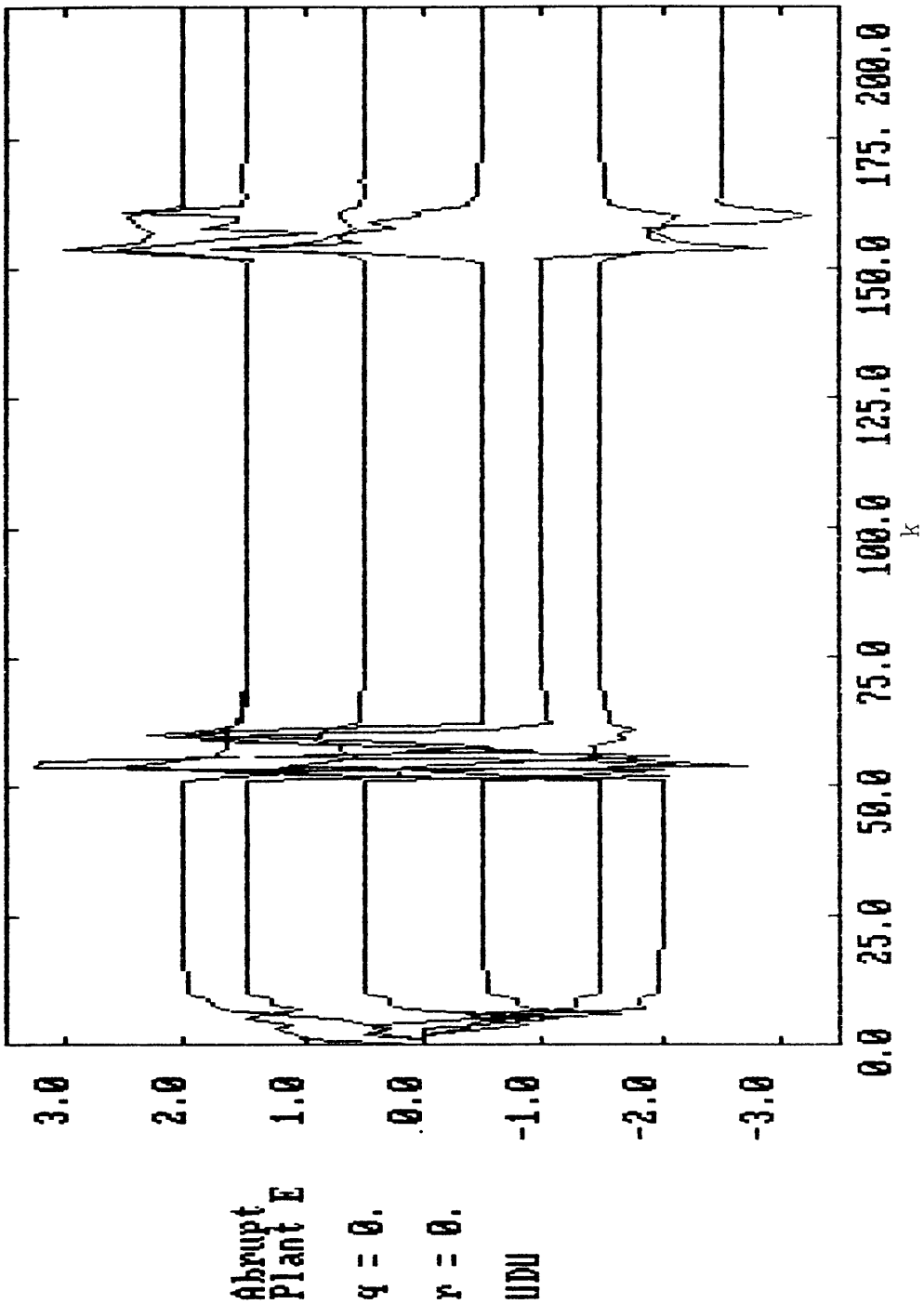

Figure 7.1 UDU^T Simulation, Plant E

For the combined purposes of improved algorithm stability and potential reduction of computational precision requirements, we have developed this factorized UDU^T approach. Its equivalence with the PLID algorithm has been established and its superior numerical properties have been pointed out. It is therefore recommended that this form of the PLID algorithm be employed during practical application. The decision to use single-precision arithmetic is left to the user and should be made only if dictated by computational hardware limitations. Clearly, since Kalman filtering is, in itself, sensitive to many other sources of divergence, one should use the highest possible numerical precision allowed by the hardware and timing limitations of the specific problem. The UDU^T formulation does, however, provide the highest possible numerical integrity once the numerical precision choice has been made and for this reason is a valuable improvement to the PLID approach.

8.0 CONCLUSION

The effort previously described has examined the properties of the Pseudo-Linear Identification (PLID) method and extended the basic form of this method to allow tracking of time-varying parameters. This extension leads directly to the viability of PLID-based adaptive control, which is easily developed by the application of Modern Control Theory. The observable-canonical state model representation of the identified system and the simultaneous state variable estimates make the PLID algorithm together with online state variable feedback a natural combination for adaptive control development. The resulting pole-placement PLID adaptive control scheme has been demonstrated to be effective not only for rather sterile linear plants but also in controlling a higher-order, nonlinear, time-varying aircraft model. It is sincerely hoped that this start-to-finish approach of developing the basic identifier, studying its properties, extending its capabilities and applying it to the adaptive control problem has not only led to a practical adaptive control algorithm, but has also provided a fundamental appreciation of the underlying goals and principles involved.

By first discussing the very concept of a mathematical model and classifying the various types of identification algorithms, we have shown that PLID is certainly not the only approach to the problem. The PLID derivation follows directly from the specification of an observable-canonical form state model to represent the system in

question. Clearly, the choice of a different model structure could lead to an entirely new method of identification, suggesting that an almost infinite variety exists in this field. The highly interdisciplinary nature of System Identification is evident in the extensive application of Kalman filtering results, not only in deriving the basic PLID algorithm, but also in extending its capabilities. The interdisciplinary nature of adaptive control is further emphasized in its special combination of control theory, state estimation, identification, statistics, and numerical analysis.

We have shown that the PLID algorithm has deadbeat convergence properties in the deterministic case and that this rapidity is carried over into cases involving measurement and process noise. The method is entirely unbiased in cases of white process noise with convergence only slightly slowed, while measurement noises do lead to biases in the parameter estimates. We have referred the reader to an alternative derivation of the algorithm able to eliminate such biases, but maintain that in the adaptive control application, where reasonably accurate sensors are likely, the small introduced biases are preferable to a slower convergence.

The algorithm is seen to lie somewhere between the Recursive Least Squares (RLS) and RML algorithms in character. PLID exhibits the speed of RLS while providing accuracy in the face of high process noise. The ability of the RML method to eliminate measurement noise bias is accompanied by much slower convergence than the slightly biased PLID algorithm. Unlike RML, however, PLID is not designed to deal with the

colored measurement noise case. Modification of PLID to identify the necessary noise model for treating colored noise is a suggested topic for further study. In the low measurement noise cases expected in closed control loop operation, the fast convergence rate of the PLID algorithm indicates promise in its subsequent application to adaptive control.

A significant portion of the contribution of this effort is the extension of the basic PLID algorithm to allow tracking of time-varying parameters. The discovery of the block-fading factor approach is the result of combining Kalman filtering ideas with practical consideration of the goals and capabilities of the basic PLID algorithm. The block fading-memory PLID algorithm responds as quickly to parameter changes as it responds initially and is capable of tracking the parameter variation shape with a lag only slightly greater than the deadbeat convergence time. The degradation of the tracking characteristics in noise is common to all tracking identification algorithms. Heuristically, this stems from the problem of determining which "unexpected" outputs are the result of a noisy measurement and which stem from changes in the plant itself. Simulations within the adaptive control structure have, however, demonstrated that the ability of PLID to track time-varying parameters in noise is sufficient to maintain effective closed-loop control.

The ready availability of state estimates from the PLID algorithm points directly to the use of state variable feedback for adaptive control. By augmenting the identified state model with an integrated

error state, a type-1 controller exhibiting zero steady-state error to step commands can be developed. Ackermann's formula for closed-loop pole assignment provides an automated link between the identified parameters and the necessary feedback gains. The resulting adaptive controller, tying these concepts together, has been demonstrated to be effective in restabilizing systems after drastic plant changes and restoring the desired closed-loop response characteristics.

The application of the PLID adaptive controller to a full nonlinear aircraft was attempted with two goals in mind. Primarily, the aircraft examples demonstrate the utility and effectiveness of the PLID adaptive controller in a realistic environment. The plant being controlled exhibits strong nonlinearities, time variation, and limited actuator effectiveness; and, thus, provides a great challenge to the adaptive controller. The ability of this approach to cope with these difficulties shows the practicality and utility of PLID adaptive control for physical systems.

The aircraft adaptive control examples also provide an appreciation for the issues which must be addressed when reaching beyond simple contrived applications to develop a truly implementable adaptive controller. Input actuator limits, error overaccumulation (integral windup) prevention, inner-loop stabilization, feedback logic constraints, initial conditions, and actuator activity are all examples of issues not usually treated in the adaptive control literature. We have also developed an improved PLID algorithm based upon factorization techniques which addresses the problem of numerically induced algorithm

divergence, another very practical issue in itself. It should be clear from our development that the treatment of these problems is often just as important to the success of the adaptive control as the power of the algorithm itself. While adaptive control does provide continually "automatic" redesign, it does so only within a specified structure which must be carefully constructed to deal with physical constraints.

The power and potential for adaptive control to improve the safety and reliability of a wide variety of systems should not be underestimated, however. While we point out that attention to detail cannot be ignored in applying this technology, we have also shown how a thoughtful implementation of adaptive control can provide a measure of safety unthinkable with fixed-design controllers. The potential for sensor and gain schedule elimination is also attractive from both a cost and control complexity standpoint. It is felt that the application of adaptive techniques should be considered as a viable option as automatic control is relied on more heavily in designs for systems of ever increasing complexity.

The PLID adaptive controller is the culmination of the effort described herein but is certainly not intended as any final solution in this field. Throughout the work we have pointed out possible avenues for extension and improvement, especially in the areas of bias reduction and the lack of a theoretical analysis of stochastic convergence. The feasibility and effectiveness of a fast adaptive controller have nonetheless been clearly demonstrated, and the favorable characteristics of PLID identification crucial in achieving this goal are clear. The

ability to simultaneously estimate parameters and states in a linear fashion represents a significant contribution to the field of system identification, and it is sincerely hoped that this work motivates others to recognize and more fully utilize the capabilities of the PLID method.

REFERENCES

[1] Gauss, K. F., "Theory of Motion of Heavenly Bodies," New York: Dover, 1963. (As discussed in Goodwin and Payne [5].)

[2] Goodwin, G. C., and K. S. Sin, Adaptive Filtering, Prediction, and Control, New Jersey: Prentice-Hall, 1984.

[3] Åström, K. J., and P. Eykoff, "System Identification-A Survey," Automatica, Vol. 7, pp. 123-162, March 1971.

[4] Sinha, N. K., and B. Kuszta, Modeling and Identification of Dynamic Systems, Van Nostrand Reinhold, 1983.

[5] Goodwin, G. C., and R. L. Payne, Dynamic System Identification: Experiment Design and Data Analysis, New York: Academic Press, 1977.

[6] Gelb, A., Applied Optimal Estimation, Cambridge, MA: MIT Press, 1974.

[7] Ljung, L., and T. Söderström, Theory and Practice of Recrusive Identification, Cambridge, MA: MIT Press, 1983.

[8] Hopkins, M. S., and H. F. VanLandingham, "Joint State and Parameter Estimation Using Psuedo-Linear Identification," under review for IEEE Trans. on Automatic Control.

[9] VanLandingham, H. F., Introduction to Digital Control, New York: Macmillan Pub. Co., 1985.

[10] Kuo, B. C., Automatic Control Systems, New Jersey: Prentice-Hall, 1982.

[11] Eykhoff, P., Editor, Trends and Progress in System Identification, Oxford: Pergamon Press, 1981.

[12] Abaci, C., "The Scientific Desk" FORTRAN Subroutine Package, 1984.

[13] Microsoft FORTRAN Compiler User's Guide, Microsoft Coroporation, 1984.

[14] Gertler, J., and C. Bányász, "A Recursive (On-Line) Maximum Likelihood Identification Method," IEEE Trans. on Automatic Control, Vol. AC-19, No. 6, pp. 816-820, December 1974.

[15] McGarty, T. P., Stochastic Systems and State Estimation, New York: John Wiley and Sons, 1974.

[16] Mendel, J. M., Discrete Techniques of Parameter Estimation, New York: Marcel Dekker, 1973.

[17] Maybeck, P. S., Stochastic Models, Estimation, and Control Volume 2, New York, Academic Press, 1982.

[18] Brogan, W. L., Modern Control Theory, New Jersey: Prentice-Hall, 1985.

[19] Luenberger, D. G., "Observing the State of a Linear System," IEEE Trans. Military Electronics, pp. 74-80, April 1964.

[20] Phillips, C. L., and M. T. Nagle, Jr., Digital Control System Analysis and Design, New Jersey: Prentice-Hall. 1984.

[21] Atkinson, K., Elementary Numerical Analysis, New York: John Wiley and Sons, 1985.

[22] NASA Technical Paper 1538, Simulator Study of Stall/Post-Stall Characteristics of a Fighter Airplane with Relaxed Longitudinal Static Stability, Springfield, VA: National Tecnical Information Service, 1979.

[23] Lewis, F. L., B. L. Stevens, R. S. Kemp, and R. D. Marshall, "Eigenstructure Assighment by Dynamic Output Feedback," American Control Conference Proceedings, Seattle, June 1986.

[24] Stevens, B. L., R. D. Marshall, and F. L. Lewis, "Identification of Nonprime Systems," American Control Conference Proceedings, Seattle, June 1986.

[25] MIL-F-8785C - Military Specification, Flying Qualities of Piloted Airplanes, Flight Dynamics Laboratory, Air Force Wright Aeronautical Labs, AF Systems Command, Wright Patterson AFB, Ohio.

[26] Bierman, G. J., Factorization Methods for Discrete Sequential Estimation, New York: Academic Press, 1977.

[27] Bierman, G. J., and M. W. Nead, A Parameter Estimation Subroutine Package, Pasadena, CA: Jet Propulsion Laboratory (NASA) Publication 77-26, 1977.

SOFTWARE APPENDIX

This appendix contains listings of the FORTRAN subroutines referred to in the text. In order, these subroutines are:

PLANT

PLID

IDENT

RML

CA

D

UDU

```
$NOFLOATCALLS
C
C************************************** SUBROUTINE PLANT
C
      SUBROUTINE PLANT(X,XP,U,Y,DSEED)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON Q1,R,NL
      REAL*8 X(6),XP(6),Q1(6,6),A(3),B(4)
      REAL*4 W(10)
C
      DATA A/.4,-1.3,1.8/,B/-.4,1.3,-1.8,0./
C
      IF(NL.NE.3)RETURN
C
      NLP=NL+1
C
      ISEED=IFIX(SNGL(DSEED))
      CALL L6A14T(ISEED,W,NLP)
      DSEED=DBLE(FLOAT(ISEED))
C
      DO 10 I=1,NL
   10 W(I)=SNGL(DSQRT(Q1(I,I)))*W(I)
      V=DSQRT(R)*DBLE(W(NLP))
C
      DO 20 I=1,NL
      IF(I.EQ.1)THEN
      XP(1)=A(1)*X(NL) + (B(1)+A(1)*B(NLP))*U +DBLE(W(1))
      ELSE
      XP(I)=X(I-1) + A(I)*X(NL)+(B(I)+A(I)*B(NLP))*U+DBLE(W(I))
      ENDIF
   20 CONTINUE
      Y=X(NL) + B(NLP)*U + V
      RETURN
      END
```

```
$NOFLOATCALLS
C
C************************************ SUBROUTINE PLID
C
      SUBROUTINE PLID(S,P,U,Y,YPL,SP,PP)
C
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON Q1,R,NL
      COMMON/OUTPUT/K,FACT
      REAL*8 S(20),P(20,20),SP(20),PP(20,20),Q1(6,6),M(20,20),
     &F(20,20),H(20),DUM1(20,20),DUM2(20),FT(20,20),K(20)
C
      N=3*NL
C
C  BUILD F AND H FROM U AND Y
C

      DO 10 I=1,N
      H(I)=0.
      DO 10 J=1,N
   10 F(I,J)=0.
      DO 20 I=2,NL
   20 F(I,I-1)=1.
      DO 40 I=1,NL
      F(I,I+NL)=Y
   40 F(I,I+2*NL)=U
      DO 50 I=NL+1,N
   50 F(I,I)=1.
      H(NL)=1.
C
C
C  KALMAN ALGORITHM
C
C  M(K)
C
      DO 70 I=1,N
      DO 70 J=1,N
   70 FT(I,J)=F(J,I)
      CALL MATMULT(P,FT,DUM1,N)
      CALL MATMULT(F,DUM1,M,N)
C
C  FADING MEMORY SEGMENT
C
      DO 80 I=NL+1,N
      DO 80 J=NL+1,N
   80 M(I,J)=FACT*M(I,J)
      DO 85 I=1,NL
   85 M(I,I)=M(I,I)+Q1(I,I)
C
C  K(K)
C
      CALL MV(M,H,DUM2,N)
      SUM=0.
      DO 90 I=1,N
   90 SUM = SUM + H(I)*DUM2(I)
      DENOM =SUM + R
      IF(DENOM.NE.0.)THEN
      DO 100 I=1,N
  100 K(I)=DUM2(I)/DENOM
      ELSE
```

```
      DO 105 I=1,N
  105 K(I)=0.
      ENDIF
C
C  P(K)
C
      DO 110 I=1,N
      DO 110 J=1,N
      DUM1(I,J)=-1.*K(I)*H(J)
  110 IF(I.EQ.J)DUM1(I,J)=DUM1(I,J)+1.
      CALL MATMULT(DUM1,M,PP,N)
C
C  S(K)
C
      CALL MV(F,S,DUM2,N)
      SUM=0.
      DO 120 I=1,N
  120 SUM=SUM + H(I)*DUM2(I)
      SUM = YPL-SUM
      DO 130 I=1,N
  130 SP(I) = DUM2(I) + K(I)*SUM
C
      RETURN
      END
C
C  UTILITY SUBROUTINES FOR MATRIX AND VECTOR MULTIPLICATION
C
      SUBROUTINE MATMULT(A,B,C,N)
      REAL*8 A(20,20),B(20,20),C(20,20),CD(20,20)
      DO 10 I=1,N
      DO 10 J=1,N
      CD(I,J)=0.
      DO 10 K=1,N
   10 CD(I,J)=CD(I,J) + A(I,K)*B(K,J)
      DO 20 I=1,N
      DO 20 J=1,N
   20 C(I,J)=CD(I,J)
      RETURN
      END
C
      SUBROUTINE MV(A,V,C,N)
      REAL*8 V(20),A(20,20),C(20),CD(20)
      DO 10 I=1,N
      CD(I)=0.
      DO 10 K=1,N
   10 CD(I)=CD(I) + A(I,K)*V(K)
      DO 20 J=1,N
   20 C(J)=CD(J)
      RETURN
      END
```

```
$NOFLOATCALLS
C
C***************************************** PROGRAM IDENT
C
      PROGRAM IDENT
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON Q1,R,NL
      COMMON/CNTRL/FB,IFLAG
      COMMON/OUTPUT/G,FACT
      COMMON/RML/XH,PHI,EPS
      COMMON/TIME/K
      COMMON/NPLANT/Q1P,RP
      REAL*8 S(20),P(20,20),SP(20),PP(20,20),G(20),Q1(6,6),
     &X(6),XP(6),U(400),Y(400),Q1P(6,6),YT(400)
      REAL*8 XH(20),PHI(20)
      REAL*8 YR(400),C(7),FB(7)
      CHARACTER*12 OBSFILE,INFILE,OUTFILE,OUT2
      CHARACTER*1 ANS,ANS2,ANS3,ANS4,ANS5,ANS6
C
      WRITE(*,2)
    2 FORMAT(1X,'POS. REAL SEED >=1. :')
      READ(*,*)DSEED
C
C  DIALOG
C
    5 WRITE(*,10)
   10 FORMAT(/15X,'SYSTEM IDENTIFICATION PROGRAM'//
     &10X,'INPUT SYSTEM ORDER:',\)
      READ(*,*)NL
      N=3*NL
      DO 15 I=1,NL
   15 X(I)=0.
      DO 16 I=1,N
      G(I)=0.
      XH(I)=0.
      PHI(I)=0.
      DO 16 J=1,N
   16 P(I,J)=0.
      EPS=0.
      K=0
      WRITE(*,20)
   20 FORMAT(/10X,'NUMBER OF RECURSION STEPS (KMAX):',\)
      READ(*,*)KMAX
C
      WRITE(*,21)
   21 FORMAT(/10X,'OPEN OR CLOSED LOOP (O OR C)?:',\)
      READ(*,29)ANS5
      IF(ANS5.EQ.'C')THEN
      ANS2='O'
      WRITE(*,22)
   22 FORMAT(/10X,'REFERENCE INPUT FILE NAME:',\)
      READ(*,36)INFILE
      OPEN(UNIT=3,FILE=INFILE,STATUS='OLD')
      DO 23 I=1,KMAX+1
   23 READ(3,*)YR(I)
      CLOSE(UNIT=3,STATUS='KEEP')
      WRITE(*,25)
   25 FORMAT(/10X,'DESIRED C-L POLYNOMIAL COEFFS:'/)
      DO 26 I=1,NL+1
      IM1=I-1
```

```
      WRITE(*,27)IM1
   27 FORMAT(10X,'C(',I2,'):',\)
   26 READ(*,*)C(I)
      DO 928 I=1,NL+1
  928 FB(I)=0.D0
      FB(NL+1)=-1.D-1
      Z=0.D0
      WRITE(*,55)
   55 FORMAT(/10X,'FILTERED OR TRUE FEEDBACK (F OR T)',\)
      READ(*,29)ANS6
      ELSE
      WRITE(*,30)
   30 FORMAT(/10X,'STORED OBSERVATIONS OR ONLINE PLANT (S OR O)?',\)
      READ(*,29)ANS2
   29 FORMAT(A1)
      IF(ANS2.EQ.'S')THEN
      WRITE(*,31)
   31 FORMAT(/10X,'OBSERVATION FILE NAME:',\)
      READ(*,28)OBSFILE
   28 FORMAT(A12)
      OPEN(UNIT=1,FILE=OBSFILE,STATUS='OLD')
      DO 32 I=1,KMAX+1
   32 READ(1,*)KTRIPE,U(I),Y(I)
      CLOSE(UNIT=1,STATUS='KEEP')
      ELSE
      WRITE(*,33)
   33 FORMAT(/10X,'INPUT SEQUENCE FILE NAME:',\)
      READ(*,36)INFILE
   36 FORMAT(A12)
      OPEN(UNIT=3,FILE=INFILE,STATUS='OLD')
      DO 34 I=1,KMAX+1
   34 READ(3,*)U(I)
      CLOSE(UNIT=3,STATUS='KEEP')
      ENDIF
      ENDIF
C
      WRITE(*,50)
   50 FORMAT(/10X,'INITIAL ESTIMATES (S):'/)
      DO 60 I=1,NL
      WRITE(*,70)I
   70 FORMAT(10X,'X(',I2,'):',\)
   60 READ(*,*)S(I)
      WRITE(*,*)
      DO 80 I=1,NL
      IM1=I-1
      WRITE(*,90)IM1
   90 FORMAT(10X,'A(',I2,'):',\)
   80 READ(*,*)S(NL+I)
      WRITE(*,*)
      DO 100 I=1,NL
      IM1=I-1
      WRITE(*,110)IM1
  110 FORMAT(10X,'B(',I2,'):',\)
  100 READ(*,*)S(2*NL+I)
      DO 150 I=1,N
      DO 150 J=1,N
  150 P(I,J)=0.
      WRITE(*,120)
  120 FORMAT(/10X,'INITIAL ERROR COVARIANCE MATRIX'/15X,'TYPE 1
     & FOR SIMPLE P*I FORM, 2 OTHERWISE'//)
```

```
      READ(*,*) IANS
      IF(IANS.EQ.1)THEN
      WRITE(*,130)
  130 FORMAT(10X,'INPUT P:',\)
      READ(*,*)PA
      DO 140 I=1,N
  140 P(I,I)=PA
      ELSE
      DO 160 I=1,N
      WRITE(*,170)I,I
  170 FORMAT(10X,'P(',I2,',',I2,'):',\)
  160 READ(*,*)P(I,I)
      ENDIF
      WRITE(*,180)
  180 FORMAT(/10X,' ALGORITHM MEASUREMENT ERROR COVARIANCE, R:',\)
      READ(*,*)R
      WRITE(*,185)
  185 FORMAT(/10X,'PLANT MEASUREMENT ERROR COVARIANCE, RP:',\)
      READ(*,*)RP
      WRITE(*,190)
  190 FORMAT(/10X,'ALGORITHM PROCESS NOISE COVARIANCE, Q1*I:',\)
      READ(*,*)QA
      DO 210 I=1,NL
  210 Q1(I,I)=QA
      WRITE(*,195)
  195 FORMAT(/10X,'PLANT PROCESS NOISE COVARIANCE, Q1P*I:',\)
      READ(*,*)QA
      DO 215 I=1,NL
  215 Q1P(I,I)=QA
      WRITE(*,231)
  231 FORMAT(10X,'FADE/NOISE FACTOR:',\)
      READ(*,*)FACT
      WRITE(*,240)
  240 FORMAT(//10X,'IS FILE OUTPUT DESIRED?',\)
      READ(*,241)ANS
  241 FORMAT(A1)
      IF(ANS.EQ.'Y')THEN
      WRITE(*,250)
  250 FORMAT(/10X,'OUTPUT FILE NAME:',\)
      READ(*,251)OUTFILE
  251 FORMAT(A12)
      OPEN(UNIT=2,FILE=OUTFILE,STATUS='NEW')
      ENDIF
      WRITE(*,252)
  252 FORMAT(/10X,'TRACK STATS?',\)
      READ(*,241)ANS3
      IF(ANS3.EQ.'N')GOTO 259
      WRITE(*,253)
  253 FORMAT(/10X,'SAVE STATS?',\)
      READ(*,241)ANS4
      IF(ANS4.EQ.'N')GOTO 259
      WRITE(*,254)
  254 FORMAT(/10X,'STATS FILE:',\)
      READ(*,251)OUT2
      OPEN(UNIT=3,FILE=OUT2,STATUS='NEW')
C
C  BEGIN SIMULATION LOOPING
C
      K=0
  259 WRITE(*,260)
```

```
  260 FORMAT(//10X,'EXTENDED STATE ESTIMATES:'//)
      IF(ANS2.EQ.'O')THEN
      IF(ANS5.EQ.'C')U(1)=0.D0
      CALL PLANT(X,XP,U(1),Y(1),DSEED)
      DO 265 I=1,NL
  265 X(I)=XP(I)
      ENDIF
      DO 290 K=0,KMAX
      WRITE(*,280)K,(S(I),I=1,N)
  280 FORMAT(1X,I3,3X/(7(1PE11.3)))
      IF(ANS.EQ.'Y')WRITE(2,280)K,(S(I),I=1,N)
      IF(ANS3.EQ.'Y')THEN
      WRITE(*,284)(P(I,I),I=1,N)
  284 FORMAT(7(1PE11.3))
      SUM=0.
      DO 282 J=1,N
  282 SUM=SUM+G(J)*G(J)
      SUM=DSQRT(SUM)
      WRITE(*,283)SUM
  283 FORMAT(1X,'KALMAN GAIN NORM:',1PE11.3)
      ENDIF
      IF(ANS4.EQ.'Y')THEN
      WRITE(3,280)K,(P(I,I),I=1,N)
      WRITE(3,286)SUM
  286 FORMAT(1PE11.3)
      ENDIF
C
      IF(ANS2.EQ.'O')CALL PLANT(X,XP,U(K+1),Y(K+2),DSEED)
      IF(ANS5.EQ.'C')THEN
      YT(K+2)=X(NL)
  400 FORMAT(1X,'REFERENCE, TRUE OUTPUT, NOISY OUT:',3(1PE11.3)/)
      ENDIF
C
      IF(K.EQ.KMAX)GOTO 291
      UK=U(K+1)
      YK=Y(K+1)
      YKP=Y(K+2)
      CALL PLID(S,P,UK,YK,YKP,SP,PP)
      DO 285 I=1,N
      S(I)=SP(I)
      DO 285 J=1,N
  285 P(I,J)=PP(I,J)
      IF(ANS2.EQ.'O')THEN
C
      IF(ANS5.EQ.'C')THEN
      CALL CONTROL(S,C,Z,UNEW)
      U(K+2)=UNEW
      IF(ANS6.EQ.'T')THEN
      Z=Z+YR(K+2)-Y(K+2)
      ELSE
      Z=Z+YR(K+2)-S(NL)
      ENDIF
      NLP=NL+1
      WRITE(*,401)(FB(I),I=1,NLP)
      WRITE(*,400)YR(K+2),YT(K+2),Y(K+2)
  401 FORMAT(1X,'FEEDBACK GAINS:',4(1PE11.3))
      ENDIF
C
      CALL PLANT(X,XP,U(K+2),YDUM,DSEED)
      DO 266 I=1,NL
```

```
  266 X(I)=XP(I)
      ENDIF
  290 CONTINUE
  291 IF(ANS2.EQ.'O')THEN
      WRITE(*,292)
  292 FORMAT(/10X,'SAVE I/O SEQUENCE?',\)
      READ(*,294)ANS
  294 FORMAT(A1)
      IF(ANS.EQ.'Y')THEN
      WRITE(*,296)
  296 FORMAT(/10X,'I/O FILENAME:',\)
      READ(*,295)OBSFILE
  295 FORMAT(A12)
      OPEN(UNIT=1,FILE=OBSFILE,STATUS='NEW')
      DO 293 K=1,KMAX+1
      KM1=K-1
  293 WRITE(1,*)KM1,U(K),Y(K)
      CLOSE(UNIT=1)
      CLOSE(UNIT=2)
      CLOSE(UNIT=3)
      ENDIF
      ENDIF
C
      IF(ANS5.EQ.'C')THEN
      WRITE(*,305)
  305 FORMAT(/10X,'SAVE REF/OUTPUTS SEQUENCE?:',\)
      READ(*,294)ANS
      IF(ANS.EQ.'Y')THEN
      WRITE(*,296)
      READ(*,295)OBSFILE
      OPEN(UNIT=1,FILE=OBSFILE,STATUS='NEW')
      DO 310 K=1,KMAX+1
      KM1=K-1
  310 WRITE(1,*)KM1,YR(K),YT(K),Y(K)
      CLOSE(UNIT=1,STATUS='KEEP')
      ENDIF
      ENDIF
C
      WRITE(*,300)
  300 FORMAT(//5X,'ANOTHER GO?',\)
      READ(*,301)ANS
  301 FORMAT(A1)
      IF(ANS.EQ.'Y')GOTO 5
      STOP
      END
```

```
$NOFLOATCALLS
C
C***************************************** SUBROUTINE RML
C
      SUBROUTINE PLID(S,P,U,Y,YPL,SP,PP)
C
C      RECURSIVE MAXIMUM LIKELIHOOD IDENTIFICATION
C
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON Q1,R,NL
      COMMON/OUTPUT/K,FACT
      COMMON/RML/XH,PHI,EPS
      REAL*8 Q1(6,6),S(20),P(20,20),SP(20),PP(20,20),M(20,20),
     &DUM3(20,20),DUM1(20,20),DUM2(20),K(20),XH(20),PHI(20),
     &REL(2),IM(2),XM(2)
C
      N=3*NL
C
C  SWITCH SIGNS ON A(Z) FOR COMPATIBILITY IN PLOTS
C
      DO 10 I=1,NL
   10 S(I)=-S(I)
C
C   BUILD XH
C
      DO 11 I=1,NL-1
      J=NL-I+1
   11 XH(J)=XH(J-1)
      XH(1)=-Y
      DO 12 I=1,NL-1
      J=2*NL-I+1
   12 XH(J)=XH(J-1)
      XH(NL+1)=U
      DO 13 I=1,NL-1
      J=3*NL-I+1
   13 XH(J)=XH(J-1)
      XH(2*NL+1)=EPS
C
C  COMPUTE W
C
      SUM=0.
      DO 14 I=1,N
   14 SUM=SUM+XH(I)*S(I)
      W=YPL-SUM
C
C  CONSTRUCT M
C
      DO 20 I=1,N
      DO 20 J=1,N
   20 M(I,J)=0.
      DO 30 JK=1,3
      I=(JK-1)*NL+1
      DO 25 L=1,NL-1
   25 M(I+L,I+L-1)=1.
      DO 30 J=I,JK*NL
   30 M(I,J)=-1.*S(2*NL+J-I+1)
C
C  PROPOGATE PHI
C
      CALL MV(M,PHI,DUM2,N)
```

```
      DO 40 I=1,N
   40 PHI(I)=DUM2(I)
      PHI(1)=PHI(1)-Y
      PHI(NL+1)=PHI(NL+1)+U
      PHI(2*NL+1)=PHI(2*NL+1)+XH(2*NL+1)
C
C    GAIN K
C
      CALL MV(P,PHI,DUM2,N)
      SUM=0.
      DO 50 I=1,N
   50 SUM=SUM+PHI(I)*DUM2(I)
      DENOM=1./FACT+SUM
      DO 60 I=1,N
   60 K(I)=DUM2(I)/DENOM
C
C    PROPOGATE P
C
      DO 70 I=1,N
      DO 70 J=1,N
   70 DUM1(I,J)=PHI(I)*PHI(J)
      CALL MATMULT(DUM1,P,DUM3,N)
      DO 80 I=1,N
      DO 80 J=1,N
      DUM3(I,J)=-DUM3(I,J)/DENOM
   80 IF(I.EQ.J)DUM3(I,J)=1.+DUM3(I,J)
      CALL MATMULT(P,DUM3,PP,N)
      DO 85 I=1,N
      DO 85 J=1,N
   85 PP(I,J)=FACT*PP(I,J)
C
C    UPDATE S
C
      DO 90 I=1,N
   90 SP(I)=S(I)+K(I)*W
C
C    STABILITY TRAP FOR C(Z)  ... VALID FOR 2ND ORDER PLANTS ONLY!!
C
      IF(NL.EQ.2)THEN
      C1=SP(5)
      C2=SP(6)
      REL(1)=-C1/2.D0
      REL(2)=-C1/2.D0
      IF((C1*C1).GE.4.D0*C2)THEN
       T=DSQRT(C1*C1-4.D0*C2)
       REL(1)=REL(1)+T/2.D0
       REL(2)=REL(2)-T/2.D0
       IM(1)=0.D0
       IM(2)=0.D0
      ELSE
       T=DSQRT(4.*C2-C1*C1)
       IM(1)=T/2.D0
       IM(2)=-T/2.D0
      ENDIF
      DO 110 I=1,2
      XM(I)=DSQRT(REL(I)*REL(I)+IM(I)*IM(I))
      IF(XM(I).GT.1.)THEN
       REL(I)=REL(I)/XM(I)
       IM(I)=IM(I)/XM(I)
      ENDIF
```

```
  110 CONTINUE
      SP(5)=-1.*(REL(1)+REL(2))
      SP(6)=REL(1)*REL(2)+IM(1)*IM(1)
      ENDIF
C
C UPDATE EPS  (EPSILON REFINEMENT)
C
      SUM=0.
      DO 95 I=1,N
   95 SUM=SUM+XH(I)*SP(I)
      EPS=YPL-SUM
C
C  SIGN SWITCHING FOR PLOT COMPATIBILITY
C
      DO 100 I=1,NL
  100 SP(I)=-1.*SP(I)
C
      RETURN
      END
C
C  UTILITY SUBROUTINES FOR MATRIX AND VECTOR MULTIPLICATION
C
      SUBROUTINE MATMULT(A,B,C,N)
      REAL*8 A(20,20),B(20,20),C(20,20),CD(20,20)
      DO 10 I=1,N
      DO 10 J=1,N
      CD(I,J)=0.
      DO 10 K=1,N
   10 CD(I,J)=CD(I,J) + A(I,K)*B(K,J)
      DO 20 I=1,N
      DO 20 J=1,N
   20 C(I,J)=CD(I,J)
      RETURN
      END
C
      SUBROUTINE MV(A,V,C,N)
      REAL*8 V(20),A(20,20),C(20),CD(20)
      DO 10 I=1,N
      CD(I)=0.
      DO 10 K=1,N
   10 CD(I)=CD(I) + A(I,K)*V(K)
      DO 20 J=1,N
   20 C(J)=CD(J)
      RETURN
      END
```

```
$NOFLOATCALLS
C
C****************************************** SUBROUTINE CA
C
      SUBROUTINE CONTROL(S,C,Z,UNEW)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON Q1,R,NL
      COMMON/CNTRL/FB,IFLAG
      REAL*8 S(20),C(7),A(20,20),B(20),FB(7),ACA(20,20),
     &DUMM(20,20),DUMV(20),CB(20,20),Q1(6,6),WK(40),FBN(7)
C
      REAL*8 X(20)
      NC=NL+1
C
C   BUILD A AND B MATRICES FROM PLID ESTIMATES
C
      DO 10 I=1,NC
      B(I)=0.
      DO 10 J=1,NC
   10 A(I,J)=0.
C
      DO 20 I=1,NL
   20 A(I,NL)=S(NL+I)
      DO 30 I=2,NL
   30 A(I,I-1)=1.
      A(NC,NL)=-1.
      A(NC,NC)=1.
C
      DO 40 I=1,NL
   40 B(I)=S(2*NL+I)
      B(NC)=0.
C
C   BUILD ACA AND CB
C
      DO 50 I=1,NC
      DO 50 J=1,NC
      DUMM(I,J)=0.
      ACA(I,J)=0.
   50 CB(I,J)=0.
C
      DO 60 I=1,NC
      ACA(I,I)=C(1)
      CB(I,1)=B(I)
   60 DUMM(I,I)=1.
C
      DO 70 K=1,NL
      CALL MATMULT(A,DUMM,DUMM,NC)
      CALL MV(DUMM,B,DUMV,NC)
      DO 70 I=1,NC
      CB(I,K+1)=DUMV(I)
      DO 70 J=1,NC
   70 ACA(I,J)=ACA(I,J)+DUMM(I,J)*C(K+1)
C
      CALL MATMULT(A,DUMM,DUMM,NC)
      DO 80 I=1,NC
      DO 80 J=1,NC
   80 ACA(I,J)=ACA(I,J)+DUMM(I,J)
C
C   ACKERMAN'S FORMULA
C
```

```
      CALL MATINV(CB,NC,IER)
C
C   TEST IER FLAG FOR MATRIX SINGULARITY
C
      IF(IER.NE.1)THEN
C
C   FEEDBACK GAIN UPDATING LOGIC WITH CONSTRAINTS
C
      CALL MATMULT(CB,ACA,DUMM,NC)
      DO 90 I=1,NC
   90 FBN(I)=DUMM(NC,I)
      IF(FBN(NC).LT.0.)THEN
       DO 94 I=1,NL
   94  IF(FBN(I).LT.0.)GO TO 93
      DO 92 I=1,NC
   92 FB(I)=FBN(I)
      ENDIF
   93 CONTINUE
      ENDIF
C
      DO 95 I=1,NL
   95 X(I)=S(I)
C
C  INPUT CALCULATION
C
      UNEW=0.
      DO 100 I=1,NL
  100 UNEW=UNEW-FB(I)*X(I)
      UNEW=UNEW-FB(NC)*Z
C
      RETURN
      END
C
C UTILITY SUBROUTINE FOR IN-PLACE MATRIX INVERSION
C
      SUBROUTINE MATINV(A,N,IER)
      IMPLICIT REAL*8 (A-H,O-Z)
      INTEGER PIVOT(25)
      REAL*8 A(20,20),B(20,20),MULT
C
      DO 5 I=1,N
      DO 5 J=1,N
      B(I,J)=0.D0
    5 IF(I.EQ.J)B(I,I)=1.D0
C
      DO 40 K=1,N-1
      PIVOT(K)=K
      AMAX=DABS(A(K,K))
      DO 10 I=K+1,N
      ABSA=DABS(A(I,K))
      IF(ABSA.GT.AMAX)THEN
       PIVOT(K)=I
       AMAX=ABSA
      ENDIF
   10 CONTINUE
      IF(AMAX.EQ.0.D0)THEN
      IER=1
      RETURN
      ENDIF
      IF(PIVOT(K).NE.K)THEN
```

```
      I=PIVOT(K)
      DO 15 J=1,N
      TEMP=B(K,J)
      B(K,J)=B(I,J)
15    B(I,J)=TEMP
      DO 20 J=K,N
      TEMP=A(K,J)
      A(K,J)=A(I,J)
20    A(I,J)=TEMP
      ENDIF
      DO 30 I=K+1,N
      MULT=A(I,K)/A(K,K)
      A(I,K)=MULT
      DO 35 J=1,N
35    B(I,J)=B(I,J)-MULT*B(K,J)
      DO 30 J=K+1,N
30    A(I,J)=A(I,J)-MULT*A(K,J)
40    CONTINUE
      IF(A(N,N).EQ.0.D0)THEN
      IER=1
      RETURN
      ENDIF
      DO 60 K=1,N
      DO 60 II=1,N
      I=N+1-II
      SUM=0.D0
      DO 50 J=I+1,N
50    SUM=SUM+A(I,J)*B(J,K)
60    B(I,K)=(B(I,K)-SUM)/A(I,I)
      IER=0
      DO 70 I=1,N
      DO 70 J=1,N
70    A(I,J)=B(I,J)
      RETURN
      END
```

```
C********************************************** SUBROUTINE D

      SUBROUTINE D(TIME,DT,X,XP)
      REAL X(*),AN,AY,QBAR,MACH,THRTL,SB,DE,YSTICK,PEDAL,
     &TIME,XSTICK
      SAVE
      COMMON Q1,R,NL
      COMMON/CNTRL/FB,IFLAG
      COMMON/PARAM/XCG
      COMMON/OUT/G,FACT
      COMMON/OUTPUT/AN,AY,QBAR,MACH
      COMMON/CONTROLS/THTL,SPB,ELEV,AIL,RDR
      COMMON/COMMANDS/THRTL,SB,XSTICK,YSTICK,PEDAL,SDELTA
      REAL*8 S(20),P(20,20),SP(20),PP(20,20),G(20),Q1(6,6),
     &C(7),FB(7),R,FACT,GX,ULIM,U0,U1,Y0,Y1,DELTA,Z,UNEW
      DATA C/-.835271,2.661487,-2.826033,0.,0.,0.,0./
      CHARACTER*12 OFL
      CHARACTER*1 ANS
      ICALL=ICALL+1
      EDLIM=.436332

  FLIGHT COMMAND SEQUENCE

      V=502.
      XSTICK=0.D0
      IF(TIME.GE.10.)V=680.
      IF(TIME.GE.12.)XCG=0.4
      IF(TIME.GE.14.0)THEN
        XSTICK=2.
      ENDIF
      IF(TIME.GE.16.)THEN
        XSTICK=0.
      ENDIF

      IF(X(8).LT.V)THRTL=1.
      IF(X(8).GT.V)THRTL=.25

  ADAPTIVE LONGITUDINAL CONTROLLER

      NL=2
      N=3*NL
      FACT=1.001

 INITIALIZATION

      IF(ICALL.EQ.1)THEN
      SDELTA=X(1)
      RETURN
      ENDIF
      IF(ICALL.EQ.2)THEN
      WRITE(*,5)
 5    FORMAT(/10X,'FILENAME FOR PARAM. OUTPUT:',$)
      READ(*,6)OFL
 6    FORMAT(A12)
      OPEN(UNIT=21,FILE=OFL,STATUS='NEW')
      DO 7 I=1,NL
 7    FB(I)=0.D0
      FB(NL+1)=-.05
      DO 10 I=1,N
```

```
 10 S(I)=0.D0
    S(3)=-.9856
    S(4)=1.953
    S(5)=.1416
    S(6)=-.1025
    U0=DBLE(X(1))
    Y0=0.D0
    DO 20 I=1,N
    DO 20 J=1,N
    P(I,J)=0.0
 20 IF(I.EQ.J)P(I,I)=200.
    DO 21 I=NL,N
 21 P(I,I)=0.D0
    DO 30 I=1,NL
    DO 30 J=1,NL
 30 Q1(I,J)=0.0
    DO 40 I=1,NL
 40 Q1(I,I)=.01
    R=1.D-8

    GET STATE ESTIMATES RIGHT

    FACT=1.0
    CALL PLID(S,P,U0,Y0,Y0,SP,PP)
    CALL PLID(SP,PP,U0,Y0,Y0,S,P)
    DO 32 I=1,N
    DO 32 J=1,N
    P(I,J)=0.D0
 32 IF(I.EQ.J)P(I,J)=1.D0

    INITIALIZE FEEDBACKS

    Z=0.D0
    CALL CONTROL(S,C,Z,UNEW)

    Z=0.D0
    DO 33 I=1,NL
 33 Z=Z+FB(I)*S(I)
    Z=Z+DBLE(X(1))
    Z=-Z/FB(NL+1)

    SDELTA=X(1)
    RETURN
    ENDIF

    Y1=DBLE(AN)

    CALL PLID(S,P,U0,Y0,Y1,SP,PP)
    K=IFIX(TIME/0.01)
    DO 285 I=1,N
    S(I)=SP(I)
    DO 285 J=1,N
285 P(I,J)=PP(I,J)

    CALL CONTROL(S,C,Z,DELTA)

    DZ=DBLE(XSTICK-AN)

    Q=X(15)
    QLOOP=QF(QBAR)*Q
```

```
INPUT LIMITING

    DECMAX=EDLIM-QLOOP
    DECMIN=-EDLIM-QLOOP
    DEC=DELTA
    IF(DEC.GT.DECMAX)THEN
     DELTA=DECMAX
     IF(DZ*FB(NL+1).LT.0.)DZ=0.
    ENDIF
    IF(DEC.LT.DECMIN)THEN
     DELTA=DECMIN
     IF(DZ*FB(NL+1).GT.0.)DZ=0.
    ENDIF
    U1=DELTA
    SDELTA=SNGL(U1)
    Z=Z+DZ
280 FORMAT(1X,I4,3X/(7(1PE11.3)))
    IF(TIME/.05.EQ.SNGL(INT(TIME/.05)))WRITE(21,280)K,(S(I),I=1,N)

    U0=U1
    Y0=Y1

    RETURN
    END
```

```
$NOFLOATCALLS
C
C***************************************** SUBROUTINE UDU
C
      SUBROUTINE PLID(S,P,U,Y,YPL,SP,PP)
C
C      BIERMAN'S UDU COVARIANCE KALMAN FILTER WITH
C         GELB'S FADE FACTOR IMPLEMENTATION
C
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON Q1,RA,NL
      COMMON/OUTPUT/K,FACT
      REAL*8 S(20),P(20,20),SP(20),Q1(6,6),
     &F(20,20),H(20),DUM2(20),K(20),
     &UU(20,20),D(20),X(20),PP(20,20),
     &A(20),V(20),LAMBDA
      REAL*8 U1(20,20),SQ(20,20)
C
      N=3*NL
C
C  PREVENT R=0.
C
      R=RA
      IF(RA.EQ.0.D0)R=1.D-99

C  BUILD F AND H FROM U AND Y
C
      DO 1010 I=1,N
      H(I)=0.
      DO 1010 J=1,N
 1010 F(I,J)=0.
      DO 1020 I=2,NL
 1020 F(I,I-1)=1.
      DO 1040 I=1,NL
      F(I,I+NL)=Y
 1040 F(I,I+2*NL)=U
      DO 1050 I=NL+1,N
 1050 F(I,I)=1.
      H(NL)=1.
C
C  KALMAN ALGORITHM
C
C
C   TIME UPDATE OF UDU, ALGORITHM OF BIERMAN PP 132-133
C
      NCAP=N+NL
      DO 1001 I=1,NCAP
      D(I)=0.D0
      DO 1001 J=1,NCAP
 1001 UU(I,J)=0.D0
C
      DO 1 I=1,N
      DO 1 J=1,N
    1 UU(I,J)=P(I,J)
      DO 2 I=1,NL
    2 UU(I,N+I)=1.D0
      DO 3 I=1,NL
    3 D(N+I)=Q1(I,I)
C
      V(1)=0.D0
```

```
      DO 20 JJ=2,N
      J=N+2-JJ
      DO 5 I=1,J
    5 D(I)=UU(I,J)
      DO 10 I=1,N
      UU(I,J)=F(I,J)
      DO 10 KK=1,J-1
   10 UU(I,J)=UU(I,J)+F(I,KK)*D(KK)
      V(J)=0.D0
      DO 15 KK=1,N
   15 V(J)=V(J)+F(J,KK)*S(KK)
   20 V(1)=V(1)+F(1,J)*S(J)
      V(1)=V(1)+F(1,1)*S(1)
      D(1)=UU(1,1)
      DO 30 J=1,N
      X(J)=V(J)
   30 UU(J,1)=F(J,1)
C
      DO 100 JJ=1,N
      J=N+1-JJ
      SIG=0.D0
      DO 40 KK=1,NCAP
      V(KK)=UU(J,KK)
      A(KK)=D(KK)*V(KK)
   40 SIG=SIG+V(KK)*A(KK)
      UU(J,J)=SIG
      IF(J.EQ.1)GOTO 100
      DINV=0.D0
      IF(SIG.GT.0.D0)DINV=1./SIG
      JM1=J-1
      DO 70 KK=1,JM1
      SIG=0.
      DO 50 I=1,NCAP
   50 SIG=SIG+UU(KK,I)*A(I)
      SIG=SIG*DINV
      DO 60 I=1,NCAP
   60 UU(KK,I)=UU(KK,I)-SIG*V(I)
   70 UU(J,KK)=SIG
  100 CONTINUE
  101 DO 105 J=2,N
      DO 105 I=1,J-1
      UU(I,J)=UU(J,I)
      UU(J,I)=0.D0
  105 CONTINUE
C
C***FADING MEMORY SEGMENT
C
      IF(FACT.GT.1.D0)THEN
      DO 205 I=N+1,NCAP
      DO 205 J=1,N
      UU(I,J)=0.D0
  205 UU(J,I)=0.D0
      DO 206 I=1,N
      DO 206 J=1,N
      P(I,J)=0.D0
  206 PP(I,J)=0.D0
      DO 200 I=1,N
  200 D(I)=UU(I,I)
      DO 210 I=1,N
  210 UU(I,I)=1.D0
```

```
C
      DO 220 I=1,NL
      DO 220 J=1,NL
  220 U1(I,J)=UU(I,J)
C
      DO 240 I=1,NL
      DO 240 J=NL+1,N
  240 SQ(I,J-NL)=UU(I,J)
C
      NR=N-NL
      CALL MATINV(U1,NL,IER)
      CALL MATMULT(U1,SQ,U1,NR)
C
      DO 250 I=1,NR
  250 PP(I,I)=D(NL+I)-D(NL+I)/FACT
      DO 255 I=1,NR
      DO 255 J=1,NR
  255 P(I,J)=U1(J,I)
C
      CALL MATMULT(U1,PP,PP,NR)
      CALL MATMULT(PP,P,P,NR)
C
      DO 260 I=1,NL
  260 P(I,I)=P(I,I)+D(I)
C
      DO 270 I=NL+1,N
  270 P(I,I)=FACT*D(I)
C
      DO 275 JJ=2,N
      J=2-JJ+N
      D(J)=P(J,J)
      ALPHA=1.D0/D(J)
      DO 275 KK=1,J-1
      BETA=P(KK,J)
      U1(KK,J)=ALPHA*BETA
      DO 275 I=1,KK
  275 P(I,KK)=P(I,KK)-BETA*U1(I,J)
      D(1)=P(1,1)
C
      DO 276 I=1,N
  276 U1(I,I)=1.D0
C
      DO 277 I=2,N
      DO 277 J=1,I-1
  277 U1(I,J)=0.D0
C
      DO 280 I=1,NL
      DO 280 J=NL+1,N
  280 UU(I,J)=UU(I,J)/FACT
      CALL MATMULT(UU,U1,UU,N)
C
      DO 290 I=1,N
  290 UU(I,I)=D(I)
      ENDIF
C
C  ALGORITHM OF BIERMAN P. 102... MEASUREMENT UPDATE
C
      DO 110 J=1,N
  110 YPL=YPL-H(J)*X(J)
      DO 120 JK=2,N
```

```
      J=N+2-JK
      DO 130 KK=1,J-1
  130 H(J)=H(J)+UU(KK,J)*H(KK)
  120 K(J)=UU(J,J)*H(J)
      K(1)=UU(1,1)*H(1)
      ALPHA=R+K(1)*H(1)
      GAMMA=1./ALPHA
      UU(1,1)=R*GAMMA*UU(1,1)
  125 DO 140 J=2,N
      BETA=ALPHA
      ALPHA=ALPHA+K(J)*H(J)
      LAMBDA=-H(J)*GAMMA
      GAMMA=1./ALPHA
      UU(J,J)=BETA*GAMMA*UU(J,J)
  126 DO 140 I=1,J-1
      BETA=UU(I,J)
      UU(I,J)=BETA+K(I)*LAMBDA
  140 K(I)=K(I)+K(J)*BETA
      YPL=YPL*GAMMA
      DO 150 J=1,N
  150 SP(J)=X(J)+K(J)*YPL
C
C  ZERO BOTTOM OF UU
C
      DO 160 I=1,N
      DO 160 J=1,N
      IF(I.GT.J)UU(I,J)=0.D0
  160 PP(I,J)=UU(I,J)
C
      RETURN
      END
C
C UTILITY SUBROUTINE FOR MATRIX MULTIPLICATION
C
      SUBROUTINE MATMULT(A,B,C,N)
      REAL*8 A(20,20),B(20,20),C(20,20),CD(20,20)
      DO 10 I=1,N
      DO 10 J=1,N
      CD(I,J)=0.
      DO 10 K=1,N
   10 CD(I,J)=CD(I,J) + A(I,K)*B(K,J)
      DO 20 I=1,N
      DO 20 J=1,N
   20 C(I,J)=CD(I,J)
      RETURN
      END
C
      SUBROUTINE MV(A,V,C,N)
      REAL*8 V(20),A(20,20),C(20),CD(20)
      DO 10 I=1,N
      CD(I)=0.
      DO 10 K=1,N
   10 CD(I)=CD(I) + A(I,K)*V(K)
      DO 20 J=1,N
   20 C(J)=CD(J)
      RETURN
      END
```

The vita has been removed from the scanned document