Rensselaer Polytechnic Institute
Troy, New York

FINAL REPORT PART A
GRANT no. NGR-33-018-~~04~~ 014
NATIONAL AERONAUTICS
AND SPACE ADMINISTRATION

LEARNING MODELS FOR ADAPTIVE SYSTEM IDENTIFICATION

by

JAMES W. SHERMAN

Submitted on behalf of

Rob Roy
Associate Professor of Electrical Engineering

October 1967

CONTENTS

LIST OF TABLES

LIST OF FIGURES

FOREWORD

The work performed under NASA grant NGR-33-018-014 covered a wide range of subjects which are coupled by the common theme of dual control. Dual control is the problem of optimal control of a process under the condition of incomplete information. Consequently, the problems of identification, adaptation, and sensitivity of optimal control systems were investigated. The final report for this grant was divided into five separate reports. These reports are as follows:

A. Learning Models for Adaptive System Identification

B. Adaptive Simulation Using Mode Identification

C. Sensitivity Design Technique

D. Bending Frequency Identification (Saturn Booster) With a Digital Coherent Memory Filter

E. Pulse Rate Adaptive Threshold Logic Units

CHAPTER I

INTRODUCTION

1.1 Approach

System identification is the process of experimentally determining the variable parameters of a model chosen to represent a physical system. The purpose of this paper is to develop a method of system identification using pattern recognition techniques. This is achieved by examining the relationship between the problems of pattern recognition and system identification. The general mathematical equations of discriminate functions used in pattern recognition are very similar to the general mathematical equations used for representing a system. Thus, a method of system identification can be obtained by taking the techniques that have been developed for solving problems of pattern recognition and applying them in a corresponding manner to the general relationships that are arrived at in system representation.

Consider a physical system with a single input and a single output. The input-output relationship of this system can be described by a functional[21]. This functional represents a transformation from the past system input and output and the present system input to the present or future system output. If the relevant system past is represented by a set of N measurements of the past system input and output, then the functional can be replaced or approximated by a transformation from the N space of such measurements to the real line. This transformation can also be viewed as a hypersurface in a N plus 1 space. Selecting the set of measurements to

be used is only part of the process of choosing a model for the physical system to be identified. The general form of the input-output transformation for the model must be chosen to sufficiently approximate the input-output transformation for the physical system. For a linear system, the hypersurface of the transformation is a hyperplane. Thus, once the model is chosen, system identification is a matter of determining the variable parameters in the general form of the hypersurface.

The basic task in pattern recognition is to classify a pattern on the basis of the attainable measurements. This is accomplished by determining a transformation from the attainable measurements to the classification based on a training set of patterns. The transformation is usually a set of surfaces, or discriminant functions, which separate the patterns of the training set into their correct categories. These surfaces are determined by first assuming a general form for the surfaces and then iteratively adjusting these surfaces after observing their performance on each member of the training set. This is called "nonparametric" training[13].

The analogy between pattern recognition and system identification is clear. Both require the determination of a hypersurface. If measurements taken from the normal operating record of the system output and the past systems input and output are viewed as a sequence of patterns, then the well developed techniques of pattern recognition can be applied. The ϕ-learning machine technique will be used here. The method is very general and can be trained by responding to the error between the system output and the ϕ-learning machine output. Note that it is the viewpoint which is

important because this viewpoint naturally leads to the use of techniques from a seemingly unrelated field.

The goal here is to develop a general method of system identification for execution on a special purpose digital computer. The desired characteristics for this method consist of the following capabilities:

1. Use in an on-line application;

2. Identification from the normal operating record of the system;

3. Control of identification error due to measurement noise;

4. Trade-off between computational complexity and speed of identification;

5. Use of the a priori knowledge of the physical system.

The desired flexibility is achieved by using the ϕ-learning machine which is briefly explained in Section 1.2. The broad class of system models that can be viewed as ϕ-learning machines is examined in some detail in Chapter II. Training procedures are developed in Chapter III for an on-line tracking model identification scheme. These procedures allow a trade-off between computational complexity, speed of identification, and control of identification error due to measurement noise. In Chapter IV simulations for a wide variety of systems give a quantitative indication of the characteristics of the identification scheme.

The description of the simulation programs and the techniques used is given in the appendices.

1.2 ϕ-Learning Machines

The term "ϕ-learning machine" refers to the generic form of a pattern recognition device. The general block diagram of this device is

shown in Figure (1.1). The "pattern" is represented by a d-dimensional vector $\underline{X}$. The first operation is a transformation of the input vector $\underline{X}$ into a vector $\underline{F}$ on ϕ-space. Vector $\underline{F}$ is a set of linearly independent functions $f_i(\underline{X})$. The coordinates in ϕ-space are the elements of the vector $\underline{F}$. Specific examples of ϕ-functions are:

1. Linear functions: $f_i(\underline{X}) = x_i \quad i = 1, \ldots d$

2. Quadric functions: $f_i(\underline{X})$ has the form $x_k^n x_\ell^m$ for $k, \ell = 1,\ldots,d$ and $n, m = 0$ and 1

3. r^{th} order polynomial functions: $f_i(\underline{X})$ has the form $x_{k_1}^{n_1} x_{k_2}^{n_2} \ldots x_{k_r}^{n_r}$ for $k_1, k_2,\ldots,k_r = 1,\ldots,d$ and $n_1, n_2,\ldots,n_r = 0$ and 1.

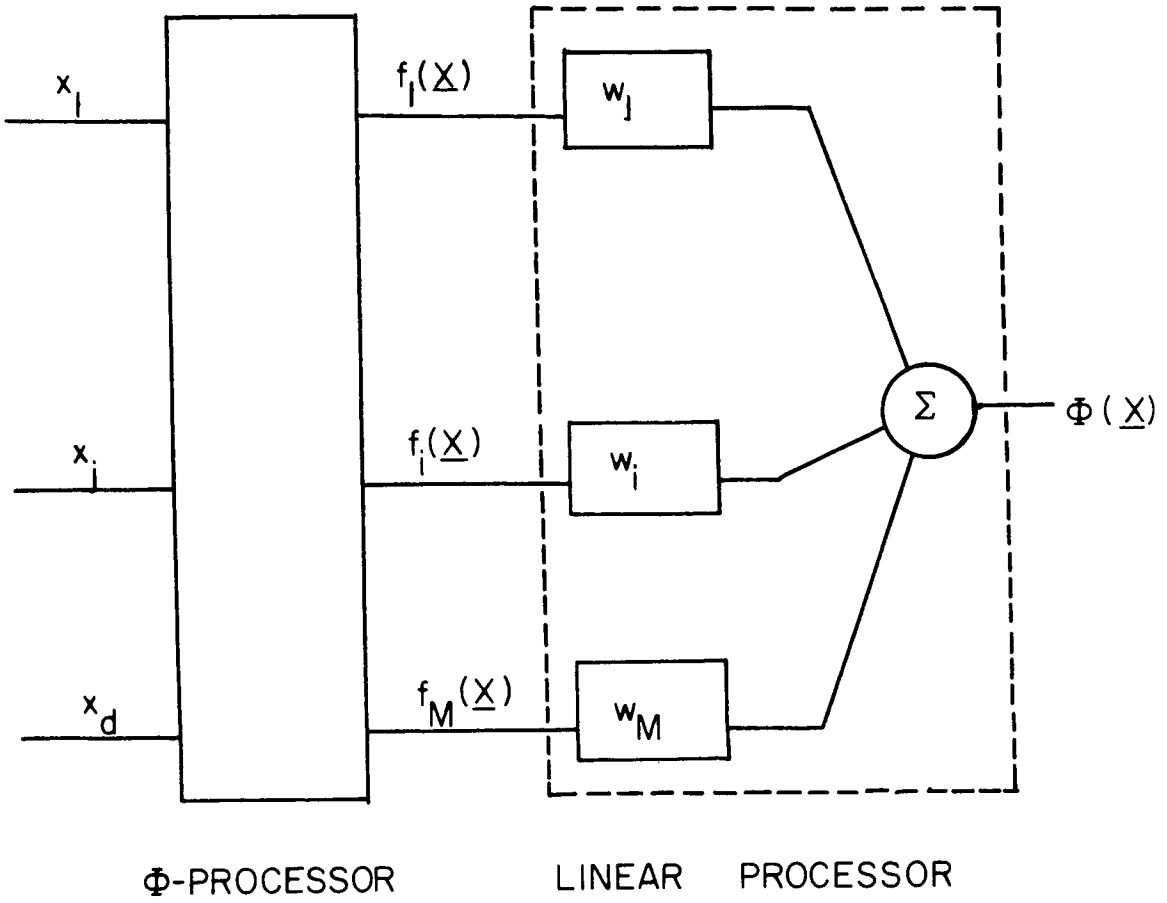
If the original vector $\underline{X}$ was defined in a d-dimensional space for r^{th} order polynomials, the vector $\underline{F}(\underline{X}) = \text{col. } f_1(\underline{X}), f_2(\underline{X}),\ldots,f_M(\underline{X})$ is defined in an M-dimensional space where

$$M = \binom{d+r}{r} - 1 \qquad r = \text{order of the polynomials} \tag{1.2-1}$$

The second operation is a weighted linear summation of the functions $f_i(\underline{X})$. The function

$$\phi(\underline{X}) = \sum_{i=1}^{M} w_i f_i(\underline{X}) \tag{1.2-2}$$

represents a linear transformation from the ϕ-space to the real line and an r^{th} order polynomial transformation from the original $\underline{X}$ pattern space.



Φ-MACHINE

FIGURE I.I

Equivalently, if the ϕ-space and pattern space are augmented by the function value $\phi(\underline{X})$, the function represents a hyperplane in the augmented ϕ-space, and an r^{th} order polynomial surface in the augmented pattern space.

In the transformed space, or ϕ-space, $\phi(\underline{X})$ is adjustable by an iterative error correcting algorithm. Consequently, the use of a transformation to a nonlinear space considerably eases the conceptual and computational difficulties in achieving a given hypersurface in the augmented linear space. The general procedure is quite similar to that of multiple regression[14] where a least squares fit to a given surface is achieved.

1.3 Historical Review

System identification can be divided into two basic problems: development of a mathematical model for a general type of process; and identification of the unknown parameters of the model for a particular process. A wide variety of techniques for system modeling and identification have been published. However, the techniques of interest are those that can identify from the normal operating record. In other words, these techniques do not need special inputs or other contrived situations.

The modeling of linear systems has become a part of the basic knowledge of every engineer. Kalman[8] developed a technique to identify and control a process which he called a self-optimizing control system. This technique used pseudo correlation functions and was based on the theories of linear systems and sampling. Levin[11] has shown that a least squares procedure is the "optimum" method for estimating an impulse-response in the presence of Guassian noise. The precision of impulse-response estimation based on short, normal operating records has been investigated by

Kerr and Surber[9]. This is achieved by computing an "expected error" based on the input and the estimated variance of the measurement noise. Jenkins and Roy[7] have developed an adaptive control system and applied it to the control of a flexible booster. This method combines dynamic programming as applied to solving the control problem and the Kalman filter as applied to state and parameter estimation. A learning method for system identification based on the error correcting training procedure used in learning machines has been investigated by Nagumo and Noda[12]. This method identifies the sampled impulse response of a linear system and is applicable to cases where the random input signal is non-stationary or to the identification of linear quasi-stationary systems such as adaptive systems, learning systems, etc.

The development of models for non-linear systems has been somewhat disorganized. Most of the work has been for individual types of processes. However, a general model for nonlinear systems with two level inputs was developed by Roy and DeRusso[15]. This model was based on the tabular form for a functional. The application of pattern recognition to system identification is not new, however the previous techniques required large memories. Examples of these are the application of decision theory by Roy and Miller[16] and of modal learning machines by Roy and Schley[17].

A great deal of work has been done recently with the series expansion of a functional developed by Volterra[21]. Wiener[2] showed that any nonlinear system with finite settling time could be characterized by a linear network which characterized the input past, followed by a zero memory nonlinearity. This cascade of two operations is essentially a

specific form of the functional approach of Volterra. This approach was also studied by Cameron and Martin[4]. Analysis of nonlinear systems by Volterra series has been treated in general terms by Barrett[1], Brilliant[2], and Smets[19]. Synthesis of nonlinear systems in like manner was the subject of Van Trees[20], Shen[18], and Bush[3]. Bush developed use of multi-dimensional Z-transforms for nonlinear discrete systems. A system identification method for Volterra series models using a stochastic approximation technique has been reported by Kwatny and Shen[10].

A comprehensive statement of the state of the art of system identification in automatic control systems was recently given in survey papers by Cuenod and Sage[5] and Eykhoff[6]. Cuenod and Sage discussed and compared some of the principle computational problems and procedures in system identification. Eykhoff investigated statistical estimation techniques by comparing various methods.

CHAPTER II

SYSTEM REPRESENTATION

2.1 Introduction

Physical systems may be represented mathematically by a variety of methods. The resulting mathematical model is based on a fundamental set of assumptions about the physical system. Some of the assumptions that can be made are: order of the system, settling time, sample data representation, and type of nonlinearity. The model chosen should involve those assumptions which best fit the a priori knowledge of the system and the function for which the model is to be used. Whether the system is to be treated as linear or nonlinear is the first consideration. Assuming linearity greatly simplifies the model but may be inaccurate for other than a small operating range. The type of nonlinearity postulated will govern how complex the model must be for a given accuracy. In addition, a sampled data representation is utilized if the system is to be controlled digitally. Physical systems are usually of a low-pass nature making the sample data representation very reasonable. The sampling rate must be fast enough to preserve the characteristics of the physical system and to allow the physical system to be controlled. The settling time of a system is defined as the amount of past input history required to generate the output to a desired degree of accuracy. Adopting a settling time requires knowledge only of the transient response of the system, not its structure. However, this approach generates a model that has a fairly large number of parameters, such as the samples of an impulse response. The order of a

sample data system may be viewed as the maximum number of delayed samples of the input or output needed to determine the next output. Assuming the order of the system based on its physical structure yields a compact model. Being of a fixed order, this model will not satisfactorily represent a system of higher order because it is very dependent on the structure of the system.

Based on his a priori knowledge of the system or lack of it, the designer must make the proper assumptions in view of the function for which the model is to be used.

2.2 Volterra Series Expansion for Delay Line Representations

Consider the linear system of Figure (2.1). The output $y(t)$ is given by the convolution integral

$$y(t) = \int_{-\infty}^{\infty} h(\tau)\, x(t - \tau)\, d\tau \tag{2.2-1}$$

Clearly,

$$\left|y(t)\right| = \left|\int_{-\infty}^{\infty} h(\tau)\, x(t - \tau)\, d\tau\right|$$

$$\leq \int_{-\infty}^{\infty}\left|h(\tau)\right| \, d\tau \, \sup_{t} \left|x(t)\right| \tag{2.2-2}$$

If

$$\int_{-\infty}^{\infty}\left|h(t)\right| \, dt < \infty \tag{2.2-3}$$

then a bounded input to the system produces a bounded output. Such a system is called "stable". The systems that will be considered here are those

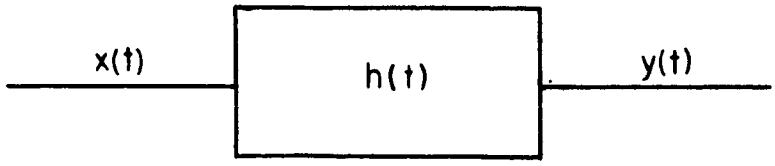


FIGURE 2.1

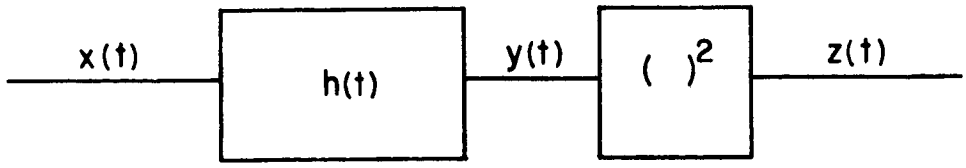


FIGURE 2.2

systems which are called "stable".

Next, examine the nonlinear system of Figure (2.2). Since

$$z(t) = y^2(t) \tag{2.2-4}$$

the output of the system is given by

$$z(t) = \int_{-\infty}^{\infty} h_1(\tau_1)\, x(t - \tau_1)\, d\tau_1 \int_{-\infty}^{\infty} h_2(\tau_2)\, x(t - \tau_2)\, d\tau_2$$

$$= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h_1(\tau_1)\, h_2(\tau_2)\, x(t - \tau_1)\, x(t - \tau_2)\, d\tau_1\, d\tau_2 \tag{2.2-5}$$

If a two dimensional kernel $h_2(\tau_1, \tau_2)$ is defined as

$$h_2(\tau_1, \tau_2) = h_1(\tau_1)\, h_1(\tau_2) \tag{2.2-6}$$

then

$$z(t) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h_2(\tau_1, \tau_2)\, x(t - \tau_1)\, x(t - \tau_2)\, d\tau_1\, d\tau_2 \tag{2.2-7}$$

The two dimensional kernel $h_2(\tau_1, \tau_2)$ is called a "regular homogeneous" functional of second degree. This kernel is "realizable" if

$$h_2(\tau_1, \tau_2) = 0 \qquad \text{for either } \tau_1 \text{ or } \tau_2 < 0 \tag{2.2-8}$$

and "stable" if

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h_2(\tau_1, \tau_2)\, d\tau_1\, d\tau_2 < \infty \tag{2.2-9}$$

Those functionals with realizable kernels are called volterra kernels. Kernels of this type play an important role in the analysis[1,19] and synthesis[18,20] of nonlinear systems.

Next, consider the case where the nonlinear block is an arbitrary continuous function

$$z(t) = f\left[y(t)\right] \tag{2.2-10}$$

The function $f(y)$ can be approximated* by a finite order polynomial.

$$f(y) \approx f_N(y) = \sum_{i=1}^{N} a_i\, y^i \tag{2.2-11}$$

Consequently $z_N(t)$ can be expressed as

$$\begin{aligned} z_N(t) &= a_0 + a_1 \int_{-\infty}^{\infty} h_1(\tau)\, x(t-\tau)\, d\tau \\ &+ \cdots\cdots \qquad\qquad \cdots\cdots \\ &+ a_N \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} h_N(\tau_1,\ldots,\tau_N)\, x(t-\tau_1)\ldots x(t-\tau_N)\, d\tau_1 \ldots d\tau_N \end{aligned} \tag{2.2-12}$$

or

$$\begin{aligned} z_N(t) = a_0 + \sum_{i=1}^{N} a_i \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} h_i(\tau_1,\ldots,\tau_i)\, x(t-\tau_1) \cdots \\ x(t-\tau_i)\, d\tau_1 \ldots d\tau_i \end{aligned} \tag{2.2-13}$$

where

$$h_i(\tau_1,\ldots,\tau_i) = \prod_{j=1}^{i} h_j(\tau_j) \tag{2.2-14}$$

---

*The Weierstrass Theorem assures that a sequence of polynomials exist which converge in a closed interval to $f(y)$. For bounded functions, this implies convergence in the mean. Thus, discontinuous nonlinearities are excluded.

If $f(y)$ is analytic in a given region, then $f(y)$ can be expanded in a power series

$$f(y) = \sum b_i y^i + b_0 \tag{2.2-15}$$

and

$$z(t) = \sum b_i \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} h_i(\tau_1, \ldots, \tau_i)\, x(t-\tau_1) \ldots x(t-\tau_i)\, d\tau_1 \ldots d\tau_i + b_0 \tag{2.2-16}$$

Since the power series (2.2-15) will converge for all $|y(t)| < \epsilon$ the functional power series will converge for all

$$|x(t)| < \frac{\epsilon}{\int_{-\infty}^{\infty} |h(\tau)|\, d\tau} \tag{2.2-17}$$

Systems which can be represented by a functional power series with a nonzero radius of convergence are called "analytic systems".[2] Although the limit of Equations (2.2-13) and (2.2-16) are the same in the region of convergence of the functional power series, Equation (2.2-16) is restricted in its range of validity.

If the system was time varying, then Equation (2.2-13) would be extended to the more general expression

$$z_N(t) = a_0 + \sum_{i=1}^{N} a_i \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} h_i(t, \tau_1, \ldots, \tau_i)\, x(t-\tau_1) \ldots x(t-\tau_i)\, d\tau_1 \ldots d\tau_i \tag{2.2-18}$$

The systems which will be investigated are the class of nonlinear systems whose output depends to an arbitrarily small extent on the remote

past; in other words, finite settling time systems. Wiener[22] showed that any nonlinear system with finite settling time could be characterized by a linear network which characterized the input past, followed by a zero memory nonlinearity. Dr. Wiener used a Laguerre network to produce an orthogonal representation of the input past, then followed this network with a set of Hermite polynomials which represented the zero memory nonlinearity. This cascade of two operations is essentially a specific form of the functional approach of Volterra. In this case, the values of the functionals depend on the values of a real function over a finite interval. The functions are continuous and square integrable over a finite interval. This approach was also studied by Cameron and Martin.[4]

Consider the case where the representation of the input past consists of a set of sample values. Thus

$$\underline{X}(t) = \text{input vector} = \text{col}\left[x_1(t)\ x_2(t)\ \cdots\ x_n(t)\right]$$

$$x_i(t) = x(t - (i-1)T)$$

T = sampling interval

nT = settling time of system (2.2-19)

Furthermore, the input will be assumed to be piecewise constant

$$x(t) = x_i \qquad (i-1)T \leq t < i\,T \qquad |x_i| \leq X \tag{2.2-20}$$

This type of input is inherent in a digital computer controlled system.

Under these assumptions, Equation (2.2-13) becomes[23]

$$z_N(t) = H_0 + \sum_{i=1}^{N} \sum_{k_1=1}^{n} \cdots \sum_{k_i=1}^{n} H_{k_1 \ldots k_i}(t)\, x_{k_1 \ldots k_i} \tag{2.2-21}$$

where

$$H_{k_1 \ldots k_i}(t) = \begin{cases} 0 & t < mT \\ \int_{k_1 T}^{M} \cdots \int_{k_i T}^{M} a_i h_i(t, \tau_1 \cdots \tau_i)\, d\tau_1 \cdots d\tau_i & mT \leq t < (m+1)T \\ \int_{k_1 T}^{(k_1+1)T} \cdots \int_{k_i T}^{(k_i+1)T} a_i h_i(t, \tau_1 \cdots \tau_i)\, d\tau_1 \cdots d\tau_i & (m+1)T \leq t \end{cases} \qquad (2.2\text{-}22)$$

$$m = \max \left[k_1, k_2, \ldots, k_i \right] ; \qquad H_0 = a_0$$

$$M = \begin{cases} (k_i+1)T & \text{for} \quad k_i < m \\ t & \text{for} \quad k_i = m \end{cases}$$

If Equation (2.2-21) is expanded, taking into account the symmetry of the kernels, then the form of the transformation surface is seen.

$$z_N(t) = H_0 + \sum_{i=1}^{N} \sum_{k_1=1}^{n} \cdots \sum_{k_i=k_{i-1}}^{n} H_{k_1 \ldots k_i}(t)\, x_{k_1} \cdots x_{k_i} \qquad (2.2\text{-}23)$$

Note that if $N = 1$ (linear system), the transformation surface is a hyperplane, for $N = 2$ a quadric surface, or in general, an N^{th} order polynomial surface.

Equation (2.2-23) is the general form for the Volterra series expansion for delay line representations and can be used as an adaptive model of a slowly time-varying plant. The parameters of the model can be obtained by the training procedures discussed in Chapter III.

2.3 Volterra Series Expansion for Difference Equation Representations

In Section 2.2, the physical system was modeled by a delay line representation which was found by assuming a finite settling time for the sampled system. In this section, the physical system will be modeled by a difference equation obtained by assuming the order of the sampled system. When considering a single input, single output, N^{th} order system without impulsive components, the general time varying difference equation is

$$y(k) = f(k, x(k-1),\ldots,x(k-N), y(k-1),\ldots,y(k-N)) \qquad (2.3\text{-}1)$$

$x(k)$ = the input at the k^{th} sampling instant

$y(k)$ = the output at the k^{th} sampling instant

For the time invariant system, this reduces to

$$y(k) = f(x(k-1),\ldots,x(k-N), y(k-1),\ldots,y(k-N)). \qquad (2.3\text{-}2)$$

In the case of time-invariant linear systems, the general N^{th} order equation becomes

$$y(k) = \sum_{i=1}^{N} a_i\, y(k-i) + \sum_{i=1}^{N} b_i\, x(k-i) \qquad (2.3\text{-}3)$$

with the corresponding Z-transform for these systems being

$$H(z) = \frac{\sum_{i=1}^{N} b_i\, z^{-i}}{1 - \sum_{i=1}^{N} a_i\, z^{-i}} \qquad (2.3\text{-}4)$$

The system is stable if the zeroes of the denominator of Equation (2.3-4) are within the unit circle in the Z-plane.

The M^{th} order Volterra series approximating Equation (2.3-2) is

$$y(k) = H_0 + \sum_{i=1}^{M} \sum_{k_1=1}^{2N} \cdots \sum_{k_i=k_{i-1}}^{2N} H_{k_1 \cdots k_i} \, c_{k_1} \cdots c_{k_i} \tag{2.3-5}$$

$$c_q = \begin{cases} x(k-q) & \text{for} \quad q \leq N \\ y(k+N-q) & \text{for} \quad q > N \end{cases}$$

This is the form of the input-output transformation surface of the Volterra series expansion of the difference equation. Multi-dimensional Z-transforms exist for those nonlinear systems whose Volterra kernels corresponding to the cross terms in x and y are zero. The multi-dimensional transform for these systems is

$$H(z_1, \ldots, z_M) = \tag{2.3-6}$$

$$\frac{\displaystyle\sum_{i=1}^{M} \sum_{k_1=1}^{N} \cdots \sum_{k_i=k_{i-1}}^{N} H_{k_1 \cdots k_i} \, z_1^{-k_1} \cdots z_i^{-k_i}}{1 - \displaystyle\sum_{i=1}^{M} \sum_{k_1=1}^{N} \cdots \sum_{k_i=k_{i-1}}^{N} H_{k_1+N, \ldots, k_i+N} \, z_1^{-k_1} \cdots z_i^{-k_i}}$$

These systems are stable if the zeroes of the denominator of Equation (2.3-6) are within the unit hypersphere in Z space. Work on the classes of non-linear systems that can be modeled by these techniques has been done by A. M. Bush[3] and others.

Equation (2.3-5) is the general form of this type of system representation and can be used as an adaptive model of a slowly time varying plant. The parameters can be obtained by the learning algorithm to be discussed.

2.4 Other System Representations

Representations put forth in the two previous sections are by no means the only ways to model a system. Other models may be developed by using the a priori knowledge of a particular physical situation to gain special advantages.

The Volterra series expansions used are essentially multivariate series in which the variables are raised only to positive powers. To sufficiently model some physical systems with these expansions, it is necessary to have a very large number of terms. A representation that includes negative powers of the variables can be used to model these more difficult systems. Any multivariate power series containing inverse terms (negative powers of any variables) can be changed into a ratio of two Volterra series by finding a common denominator. This general form is

$$y = \frac{g_0 + \sum_{i=1}^{M} \sum_{k_1=1}^{N} \cdots \sum_{k_i=k_{i-1}}^{N} g_{k_1,\ldots,k_i} x_{k_1} \cdots x_{k_i}}{1 + \sum_{i=1}^{M} \sum_{k_1=1}^{N} \cdots \sum_{k_i=k_{i-1}}^{N} h_{k_1,\ldots,k_i} x_k \cdots x_k} \qquad (2.4\text{-}1)$$

A special type of model is formed from a set of transfer functions followed by a linear or nonlinear combination network. An example is the Laguerre network followed by the Hermite polynomials used by Dr. N. Weiner.

An advantage of this method is that it provides output continuously but can be trained on a sample data basis.

Polynomials are not the only type of nonlinearity that can be used to model a physical system. If the range of the input and output of the system is not the infinite interval, the nonlinear functions used can be periodic or defined on only a finite interval. A multi-dimensional Fourier series is an example of this approach.

2.5 Generalized Model

A generalized model that may be trained as a ϕ-learning machine will now be shown. The model includes all of the system representations mentioned in this chapter as special cases. Let $\underline{X}$ be a vector representation of the measured system past input and output. The generalized model is

$$y(k) = \frac{\underline{U}^T \underline{G}(\underline{X})}{1 + \underline{V}^T \underline{H}(\underline{X})} \tag{2.5-1}$$

For training, the model must be viewed as a ϕ-learning machine. The ϕ-learning machine corresponding to Equation (2.5-1) is

$$y(k) = \phi(x, y) = \underline{W}^T \underline{F}(\underline{X}) \tag{2.5-2}$$

where

$$\underline{W} = \begin{pmatrix} \underline{U} \\ \underline{V} \end{pmatrix}$$

$$\underline{F}(\underline{X}) = \begin{pmatrix} \underline{G}(\underline{X}) \\ y(k)\ \underline{H}(\underline{X}) \end{pmatrix}$$

The methods and properties of training this ϕ-learning machine by error correcting algorithms are the subject of the next chapter.

CHAPTER III

TRAINING PROCEDURES

3.1 Introduction

Training procedures will be developed and studied in this chapter for ϕ-learning machines corresponding to system models with and without inverse terms. The basic training procedure uses an error correcting algorithm to train the ϕ-machine. This algorithm iteratively adjusts the weight vector of the linear portion of the ϕ-machine based upon the normal operating record of the system.[12] Several variations of the training procedure will be considered.

The training procedures use the following nomenclature:

y_i = observed output of the system at the i^{th} iteration

y_i^* = output of "optimum" model at the i^{th} iteration

$\hat{y}_i$ = output of learning model at i^{th} iteration

$\underline{X}_i$ = vector representation of system past input and output at the i^{th} iteration for input to ϕ-processor

$z_i = \phi(\underline{X}_i)$ = output of ϕ-machine at i^{th} iteration

$\underline{U}_i$ = numerator weight vector of model at i^{th} iteration

$\underline{V}_i$ = denominator weight vector of model at i^{th} iteration

$\underline{W}_i = \begin{pmatrix} \underline{U}_i \\ \underline{V}_i \end{pmatrix}$ = weight vector of ϕ-machine at i^{th} iteration

$\underline{W}_*$ = "optimum" weight vector of ϕ-machine

$\underline{F}(\underline{X}) = \text{col.}\left[f_1(\underline{X}),\ldots,f_M(\underline{X})\right]$ = ϕ-transformation

$\underline{G}_i$ = numerator vector of ϕ-functions for model at i^{th} iteration

$\underline{H}_i$ = denominator vector of ϕ-functions for model at i^{th} iteration

$\underline{F}_i = \begin{pmatrix} \underline{G}_i \\ * \\ y_i \underline{H}_i \end{pmatrix} = \underline{F}(\underline{X}_i)$ = output of ϕ-processor at i^{th} iteration

$\left\| \underline{F}_i \right\| = \underline{F}_i^T \underline{F}_i$ (squared Euclidian norm)

a = convergence factor of algorithm

The system model with inverse terms may be written as:

$$\hat{y}_i = \frac{\underline{U}_i^T \underline{G}_i}{1 - \underline{V}_i^T \underline{H}_i} \tag{3.1-1}$$

The corresponding ϕ-machine is

$$z_i = \underline{U}_i^T \underline{G}_i + y_i \underline{V}_i^T \underline{H}_i = \underline{W}_i^T \underline{F}_i \tag{3.1-2}$$

Both the system model without inverse terms and its corresponding ϕ-machine may be written as:

$$\hat{y}_i = z_i = \underline{U}_i^T \underline{G}_i = \underline{W}_i^T \underline{F}_i \tag{3.1-3}$$

The sequence of steps in the basic training procedure is as follows:

1. Set the initial weight vector. A zero weight vector is adequate.
2. Determine y_i and $\underline{X}_i$.
3. Generate $\underline{G}_i$, $\underline{H}_i$, and z_i.

4. Calculate new weight vector using the following error correcting algorithm

$$\underline{W}_{i+1} = \underline{W}_i + \frac{a\,(y_i - z_i)}{\left\| \begin{pmatrix} G_i \\ y_i \underline{H}_i \end{pmatrix} \right\|} \begin{pmatrix} \underline{G}_i \\ y_i \underline{H}_i \end{pmatrix} \qquad (3.1\text{-}4)$$

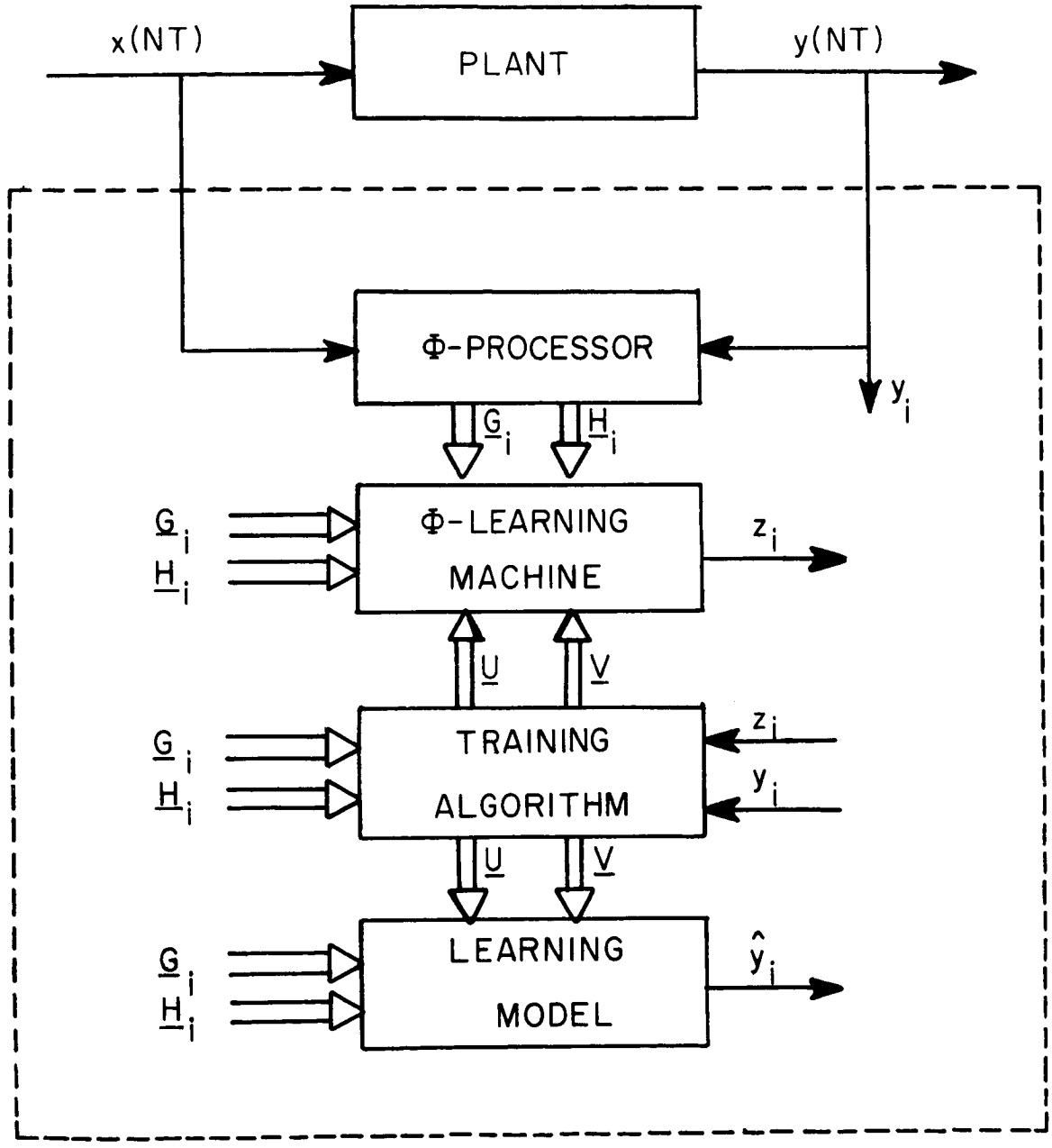
5. Repeat starting with step No. 2 using next value of i.

Figure (3.1) is a diagram of a model tracking system that can use any of the training procedures in this chapter.

3.2 Geometric Interpretation and an Example

The basic training procedure generates a sequence of $\underline{W}_i$'s whose components converge towards an accurate representation of the physical system. The learning process can be viewed geometrically be representing the weights as vectors in an Euclidian "weight" space. If a physical system is adequately represented by a ϕ-machine model, then the weight vector of the ϕ-machine model is the "optimum" weight vector in "weight" space. The weight error vector is the difference between the "optimum" and the current learning model weight vectors.

The output of the system is the scalar product of the "optimum" weight vector and the vector output of the ϕ-processor. Likewise, the output of the model is the scalar product of the current model weight vector and the vector output of the ϕ-processor. The error of the ϕ-learning machine is the difference of the scalar products which is the projection of the weight error vector on the vector output of the ϕ-processor. The training procedure then changes the current weight vector in order to correct the



MODEL TRACKING SYSTEM

FIGURE 3.1

observed error. This is done by adding a multiple of the vector output of the ϕ-processor to the current weight vector such that the difference between the observed model and system scalar products is zero.

The system shown in Figure (3.2) will be used to illustrate this algorithm. The input-output relationship for this system is

$$y\left[NT\right] = 17 \times \left[NT\right] + 17 \times \left[(N-1)T\right] = (17,\ 17) \begin{pmatrix} x\left[NT\right] \\ x\left[(N-1)T\right] \end{pmatrix} \qquad (3.2\text{-}1)$$

The correct weight vector is col. (17, 17). Assume an input sequence such as

N	0	1	2	3	4	5	6	7	8
$x\left[NT\right]$	0	1	1	1	-4	2	-1	0	-1

Table I lists the results at each iteration. The sequence of weight vectors is plotted in Figure (3.3). This sequence converges to the correct weight vector. The weight vector error is the difference between the correct weight vector and the ϕ-machine weight vector. The sequence of weight vector errors is also plotted in Figure (3.3). The difference between two successive weight vectors, $\underline{W}_{i+1} - \underline{W}_i$, is the projection of the weight vector error, $\underline{E}_i$, on the unit vector in the direction of the input vector $\underline{X}_i$. Therefore the weight vector error cannot increase. The weight vector error will remain the same, if two successive input vectors are linearly dependent. The input vectors for $i = 2, 3$ and for $i = 5, 6$ are examples of this.

3.3 Error Bounds for General System Representations

Consider the system of Figure (3.4). The output of the system is corrupted by additive noise N. $\underline{W}_*$ is the weight vector of the ϕ-machine

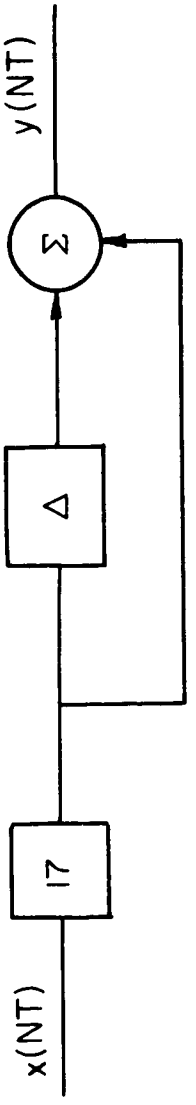
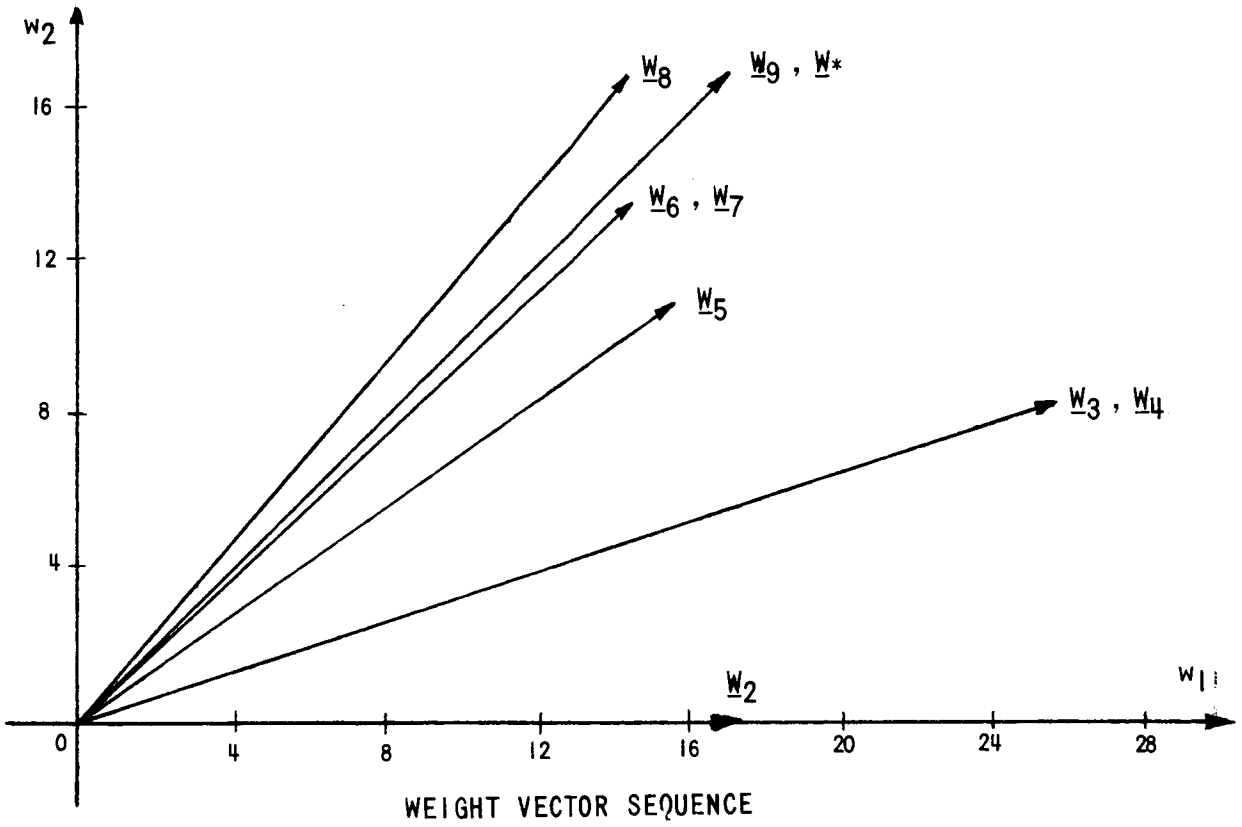


FIGURE 3.2

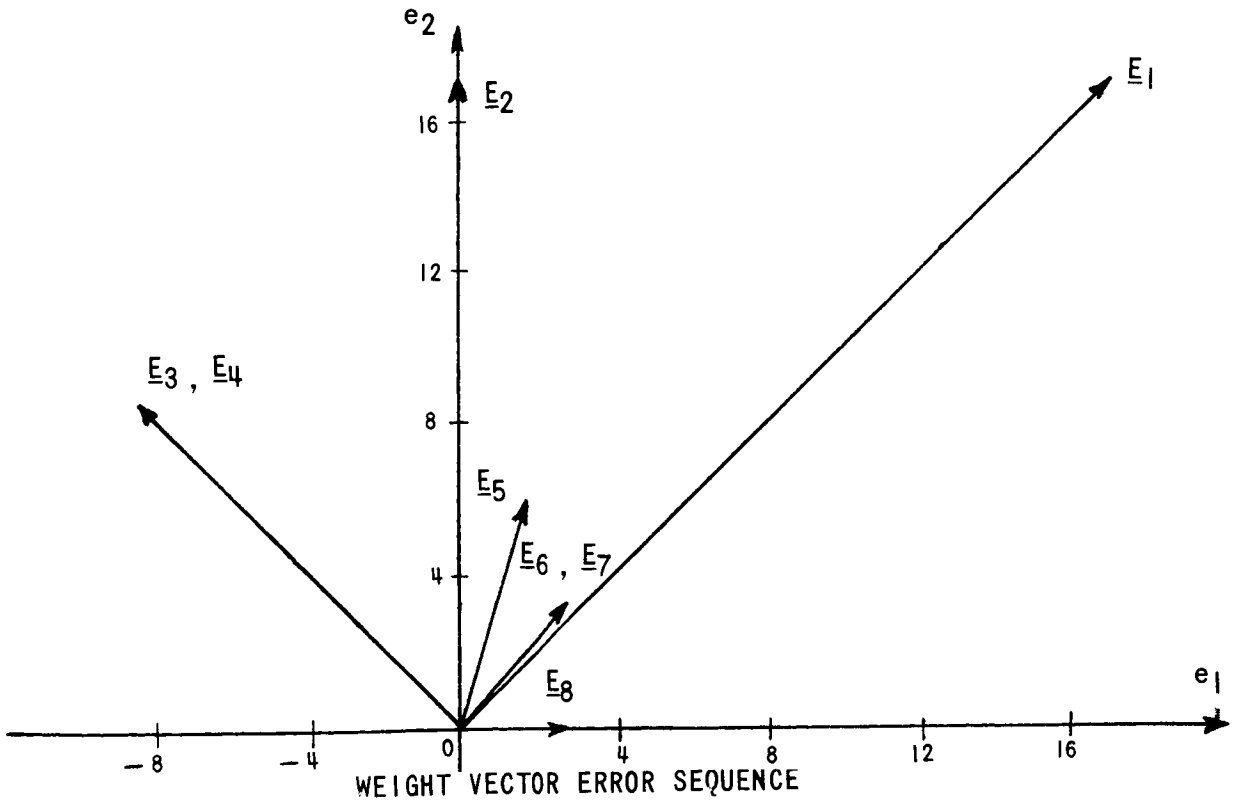
TABLE I

i	1	2	3	4	5	6	7	8
y_i	17	34	34	-51	-34	17	-17	-17
z_i	0	17	34	-93.5	-13	17	-13.6	-14.2
$\underline{X}^T$	1, 0	1, 1	1, 1	-4, 1	2, -4	-1, 2	0, -1	-1, 0
$\underline{W}_i^T$	0, 0	17, 0	25.5, 8.5	25.5, 8.5	15.5, 11	14.2, 13.6	14.2, 13.6	14.2, 17
$c_i\underline{X}_i^T$	17, 0	8.5, 8.5	0, 0	-10, 2.5	-1.3, 2.6	0, 0	0, 3.4	2.8, 0
$\underline{W}_{i+1}^T$	17, 0	25.5, 8.5	25.5, 8.5	15.5, 11	14.2, 13.6	14.2, 13.6	14.2, 17	17, 17

where $\underline{F}_i = \underline{X}_i$, $a = 1$, and $c_i = (y_i - z_i)/\|\underline{X}\|$



WEIGHT VECTOR SEQUENCE



WEIGHT VECTOR ERROR SEQUENCE

FIGURE 3.3

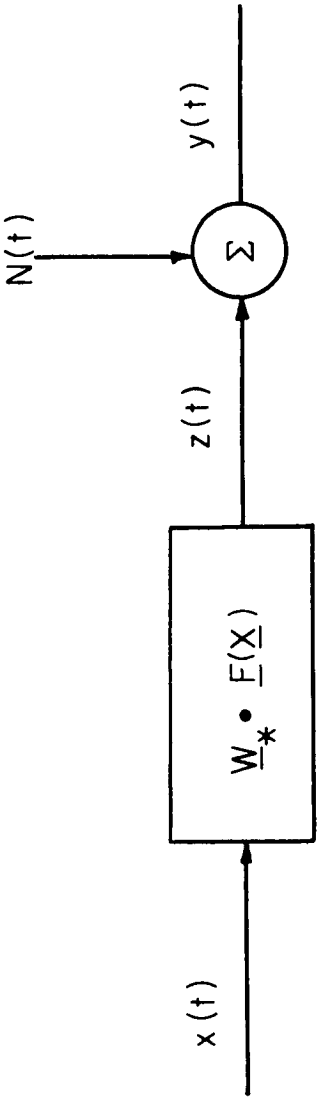


FIGURE 3.4

model which describes the system. The actual system output can be written as:

$$y^* = \underline{W}_*^T \underline{F}(\underline{X}) = \underline{W}_* \cdot \underline{F}(\underline{X}) \tag{3.3-1}$$

Then the observed system output can be written as:

$$y_i = y_i^* + N_i = \underline{W}_*^T \underline{F}_i + N_i \tag{3.3-2}$$

For systems that are approximated by Equation (3.3-1) the truncated terms of their representations are combined with the additive noise N. At the i^{th} iteration the additive noise N_i is the sum of the measurement noise and the representation residual.

Let $\underline{E}_i$ be the weight error vector.

$$\underline{E}_i = \underline{W}_* - \underline{W}_i \tag{3.3-3}$$

Convergence of the ϕ-learning machine will be defined as $\underline{E}_i$ approaching zero as i increases without bound.

A recursive relationship will now be developed for $\underline{E}_i$ by combining Equations (3.3-3) and (3.1-4)

$$\underline{E}_{i+1} = \underline{E}_i - \frac{a\,(y_i - z_i)}{\left\| \begin{pmatrix} \underline{G}_i \\ y_i\, \underline{H}_i \end{pmatrix} \right\|} \begin{pmatrix} \underline{G}_i \\ y_i\, \underline{H}_i \end{pmatrix} \tag{3.3-4}$$

Eliminating y_i and z_i with equations (3.3-2) and (3.1-2) yields

$$\underline{E}_{i+1} = \underline{E}_i - \frac{a}{\|\hat{\underline{F}}_i\|} (\underline{W}_*^T \underline{F}_i + N_i - \underline{W}_i^T \hat{\underline{F}}_i) \hat{\underline{F}}_i \qquad (3.3\text{-}5)$$

$$= \underline{E}_i - \frac{a}{\|\hat{\underline{F}}_i\|} (\underline{W}_*^T \hat{\underline{F}}_i + N_i - \underline{W}_i^T \hat{\underline{F}}_i - N_i \underline{V}_*^T \underline{H}_i) \hat{\underline{F}}_i$$

$$= \underline{E}_i - \frac{a}{\|\hat{\underline{F}}_i\|} \underline{E}_i^T \hat{\underline{F}}_i \hat{\underline{F}}_i - \frac{a}{\|\hat{\underline{F}}_i\|} N_i(1 - \underline{V}_*^T \underline{H}_i) \hat{\underline{F}}_i$$

where

$$\hat{\underline{F}}_i = \begin{pmatrix} \underline{G}_i \\ (y_i^* + N_i)\underline{H}_i \end{pmatrix} = \underline{F}_i + \begin{pmatrix} \underline{0} \\ N_i \underline{H}_i \end{pmatrix} = \underline{F}_i + \triangle \underline{F}_i \qquad (3.3\text{-}6)$$

Let

$$A_i = I - \frac{a}{\|\hat{\underline{F}}_i\|} \hat{\underline{F}}_i \hat{\underline{F}}_i^T$$

and

$$b_i = \frac{a}{\|\hat{\underline{F}}_i\|} N_i(1 - \underline{V}_*^T \underline{H}_i) \qquad (3.3\text{-}7)$$

then

$$\underline{E}_{i+1} = A_i \underline{E}_i - b_i \hat{\underline{F}}_i \qquad (3.3\text{-}8)$$

Consider the case of an exact model with no measurement noise ($N_i = 0$). Equation (3.3-8) reduces to

$$\underline{E}_{i+1} = A_i \underline{E}_i \qquad (3.3\text{-}9)$$

since b_i is zero.

Taking the norm of Equation (3.3-9) noting that $\hat{\underline{F}}_i = \underline{F}_i$ gives

$$\left\| \underline{E}_{i+1} \right\| = \underline{E}_i^T A_i^2 \underline{E}_i \tag{3.3-10}$$

$$= \underline{E}_i^T \left[I - \frac{2a}{\left\| \underline{F}_i \right\|} \underline{F}_i \underline{F}_i^T + \frac{a^2}{\left\| \underline{F}_i \right\|^2} \underline{F}_i \underline{F}_i^T \underline{F}_i \underline{F}_i^T \right] \underline{E}_i$$

$$= \underline{E}_i^T \left[I - \frac{a(2 - a)}{\left\| \underline{F}_i \right\| \left\| \underline{E}_i \right\|} \underline{F}_i \underline{F}_i^T \right] \underline{E}_i$$

$$= \left[1 - \frac{a(2 - a)}{\left\| \underline{F}_i \right\| \left\| \underline{E}_i \right\|} (\underline{F}_i^T \underline{E}_i)^2 \right] \left\| \underline{E}_i \right\|$$

Let

$$c_i = \frac{a(2 - a)}{\left\| \underline{F}_i \right\| \left\| \underline{E}_i \right\|} (\underline{F}_i^T \underline{E}_i)^2 \tag{3.3-11}$$

where $0 \leq c_i \leq 1$ since $0 < a < 2$, then

$$\left\| \underline{E}_{i+1} \right\| = \prod_{j=1}^{i} (1 - c_j) \left\| \underline{E}_1 \right\| \tag{3.3-12}$$

Note that, since $0 \leq c_i \leq 1$,

$$0 \leq \prod_{j=1}^{i} (1 - c_j) \leq 1 \tag{3.3-13}$$

It will be assumed that c_i never equals one because $c_i = 1$ implies immediate convergence, $\left\| \underline{E}_{i+1} \right\| = 0$. A necessary and sufficient condition for the limit of Equation (3.3-12) to converge to zero as i increasesis that the sum of the c_i diverges.

$$\sum_{j=1}^{\infty} c_j = \infty \tag{3.3-14}$$

Therefore, the input vector must probe the input vector space so that an infinity of c_i are nonzero. Also, the c_i cannot approach zero too quickly. By Equations (3.3-11) and (3.3-14)

$$\sum_{j=1}^{\infty} \frac{(\underline{F}_i^T \underline{E}_i)^2}{\|\underline{F}_i\| \ \|\underline{E}_i\|} = \infty \tag{3.3-15}$$

Thus, the sequence of $\underline{F}_i$'s must probe its vector space in all directions infinitely often in the training sequence. Therefore, the sequence of $\underline{X}_i$'s must probe its vector space in both magnitude and direction.

If N_i is not identically zero the ϕ-learning machine may still converge. However, this can happen only if the sequence of N_i's approaches zero.

Consider the case of a model without inverse terms which has measurement noise. Now b_i is not zero, but $\Delta \underline{F}_i$ is zero. Equation (3.3-8) reduces to

$$\underline{E}_{i+1} = A \underline{E}_i + b_i \underline{F}_i \tag{3.3-16}$$

Taking the norm of Equation (3.3-16), noting that $\hat{\underline{F}}_i = \underline{F}_i$ and $\underline{H}_i = 0$ yields

$$\|\underline{E}_{i+1}\| = \underline{E}_i^T A_i^2 \underline{E}_i - 2b_i \underline{F}_i^T A_i \underline{E}_i + b_i^2 \|\underline{F}_i\| \tag{3.3-17}$$

$$= \left[1 - \frac{a(2-a)(\underline{F}_i^T \underline{E}_i)^2}{\|\underline{F}_i\| \ \|\underline{E}_i\|}\right] \|\underline{E}_i\| + \frac{a N_i}{\|\underline{F}_i\|} (a N_i + 2(a-1) \underline{F}_i^T \underline{E}_i)$$

$$= (1 - c_i) \|\underline{E}_i\| + d_i$$

$$= \prod_{j=1}^{i} (1 - c_j) \|\underline{E}_1\| + \sum_{j=1}^{i-1} d_j \prod_{k=j+1}^{i} (1 - c_k) + d_i$$

where

$$d_i = \frac{a\, N_i}{\|\underline{F}_i\|} \left(a\, N_i + 2(a-1)\, \underline{F}_i^T\, \underline{E}_i\right) \tag{3.3-18}$$

Assume that d_i is bounded for all i, that condition (3.3-14) is satisfied, and that there are only a finite number of c_i less than a preset positive constant.

$$d_i \leq d_{max}, \quad \text{all } i \qquad 0 \leq c_i < \epsilon \tag{3.3-19}$$

for $N-1$ values of i. An upper bound on $\|\underline{E}_i\|$ is found by operating on Equation (3.3-17).

$$\begin{aligned}
\|\underline{E}_{i+1}\| &\leq \prod_{j=1}^{i} (1 - c_i)\, \|\underline{E}_1\| + d_{max} \left[1 + \sum_{j=1}^{i-1} \prod_{k=j+1}^{i} (1 - c_k)\right] \\
&\leq \prod_{j=1}^{i} (1 - c_i)\, \|\underline{E}_1\| + d_{max} \left[1 + N - 1 + \sum_{j=1}^{i-N} \prod_{k=j+1}^{i+1-N} (1 - \epsilon)\right] \\
&\leq \prod_{j=1}^{i} (1 - c_i)\, \|\underline{E}_1\| + d_{max} \left[N + \sum_{j=1}^{i-N} \epsilon^j\right]
\end{aligned} \tag{3.3-20}$$

Taking the limit as $i \longrightarrow \infty$ gives

$$\|\underline{E}_\infty\| \leq d_{max} \left[N + \frac{1}{\epsilon}\right] \tag{3.3-21}$$

where

$$d_{max} = \max_i \left[\frac{a\, N_i}{\|\underline{F}_i\|} \left(a\, N_i + 2\,(a-1)\, \underline{F}_i^T\, \underline{E}_i\right)\right]$$

This very conservative bound is dependent on the magnitude of the noise, N_i, the convergence factor, and the weight error vector. The bound decreases as these quantities decrease.

Consider the case of a model with inverse terms which has measurement noise. Now b_i and $\Delta \underline{F}_i$ are not zero. Taking the norm of Equation (3.3-8) gives

$$\left\| \underline{E}_{i+1} \right\| = \underline{E}_i^T A_i^2 \underline{E}_i - 2b_i \hat{\underline{F}}_i^T A_i \underline{E}_i + b_i^2 \left\| \hat{\underline{F}}_i \right\| \qquad (3.3\text{-}22)$$

$$= \left[1 - \frac{a(2-a)(\hat{\underline{F}}_i^T \underline{E}_i)^2}{\left\| \hat{\underline{F}}_i \right\| \left\| \underline{E}_i \right\|} \right] \left\| \underline{E}_i \right\| + \frac{a N_i}{\left\| \hat{\underline{F}}_i \right\|} (1 - \underline{V}_*^T \underline{H}_i)$$

$$(aN_i (1-\underline{V}_*^T \underline{H}_i) - 2(1-a) \hat{\underline{F}}_i^T \underline{E}_i)$$

Let

$$c_i = 1 - \frac{a(2-a)}{\left\| \hat{\underline{F}}_i \right\| \left\| \underline{E}_i \right\|} (\hat{\underline{F}}_i^T \underline{E}_i)^2 \qquad (3.3\text{-}23)$$

where $0 \leq c_i \leq 1$ since $0 < a < 2$. Let

$$d_i = \frac{a N_i}{\left\| \hat{\underline{F}}_i \right\|} (1-\underline{V}_*^T \underline{H}_i)(aN_i(1-\underline{V}_*^T \underline{H}_i) + 2(a-1) \hat{\underline{F}}_i^T \underline{E}_i) \qquad (3.3\text{-}24)$$

Assume that this new c_i still satisfies condition (3.3-14). This is reasonable because the Equation (3.3-23) is Equation (3.3-11) with $\hat{\underline{F}}_i$ replacing $\underline{F}_i$. $\hat{\underline{F}}_i$ should satisfy the same criteria that $\underline{F}_i$ must satisfy for condition (3.3-14) to hold because $\hat{\underline{F}}_i$ is just $\underline{F}_i$ that has been corrupted by N_i, the additive noise. Now

$$\left\| \underline{E}_{i+1} \right\| = \prod_{j=1}^{i} (1-c_j) \left\| \underline{E}_1 \right\| + \sum_{j=1}^{i-1} d_j \prod_{k=j+1}^{i} (1-c_k) + d_i \qquad (3.3\text{-}25)$$

A upper bound on $\left\| \underline{E}_{\infty} \right\|$ is found by operating on Equation (3.3-25) in the same manner as on Equation (3.3-17). Making the assumptions of Equation (3.3-19) on the new c_i and d_i yields an upper bound for $\left\| \underline{E}_{\infty} \right\|$ in this case.

$$\left\| \underline{E}_{\infty} \right\| \leq d_{max} \left[N + \frac{1}{\epsilon} \right] \qquad (3.3\text{-}26)$$

where

$$d_{max} = \max_i \left[\frac{a N_i}{\left\| \hat{\underline{F}}_i \right\|} (1-\underline{V}_*^T \underline{H}_i)(aN_i(1-\underline{V}_*^T \underline{H}_i) + 2(a-1) \hat{\underline{F}}_i^T \underline{E}_i) \right]$$

Only the definition of d_i makes this bound different from Equation (3.3-21). They both have the same properties.

3.4 Error Bounds for Volterra Series System Representations

A useful concept for nonlinear systems is the dynamic approximation of a system about an operating point. The approximation could be a linear or a nonlinear system that is simpler than the physical system. This section investigates how this approximation technique can be accomplished with the ϕ-learning machine for systems that can be represented by a Volterra series.

The Volterra series system representations of Chapter II can be written in the generalized form of Equation (2.4-1). Let $\underline{X}$ be a N-dimensional vector representation of the system past input and output, then

$$y^* = \frac{P(\underline{X})}{Q(\underline{X})} \tag{3.4-1}$$

where

$$P(\underline{X}) = p_0 + \sum_{i=1}^{\infty} \sum_{k_1=1}^{N} \cdots \sum_{k_i=k_{i-1}}^{N} p_{k_1,\ldots,k_i} \; x_{k_1} \cdots x_{k_i}$$

$$Q(\underline{X}) = q_0 + \sum_{i=1}^{\infty} \sum_{k_1=1}^{N} \cdots \sum_{k_i=k_{i-1}}^{N} q_{k_1,\ldots,k_i} \; x_{k_1} \cdots x_{k_i} \tag{3.4-2}$$

$$\underline{X} = \text{col.} \left[x_1, \ldots, x_N \right]$$

$P(\underline{X})$ and $Q(\underline{X})$ are called Volterra series or functional power series. They can be expanded in a Taylor series about an operating point, $\underline{X}^0$. For an r^{th} order expansion the remainder will be insignificant in some sufficiently small region about the operating point.

$$P(\underline{X}) = P(\underline{X}^0) + P_R \tag{3.4-3}$$

$$+ \sum_{i=1}^{r} \sum_{k_1=1}^{N} \cdots \sum_{k_i=k_{i-1}}^{N} \frac{\partial^i P(\underline{X})}{\partial x_{k_1}, \ldots, \partial x_{k_i}} \bigg|_{\underline{X}^0} (x_{k_1} - x_{k_1}^0) \ldots (x_{k_i} - x_{k_i}^0)$$

$$Q(\underline{X}) = Q(\underline{X}^0) + Q_R$$

$$+ \sum_{i=1}^{r} \sum_{k_1=1}^{N} \cdots \sum_{k_i=k_{i-1}}^{N} \frac{\partial^i Q(\underline{X})}{\partial x_{k_1}, \ldots, \partial x_{k_i}} \bigg|_{\underline{X}^0} (x_{k_1} - x_{k_1}^0) \ldots (x_{k_i} - x_{k_i}^0)$$

Now Equation (3.4-1) may be approximated by neglecting the remainder.

$$y^* \approx \frac{P(\underline{X}^0) + \sum_{i=1}^{r} \sum_{k_1=1}^{N} \cdots \sum_{k_i=k_{i-1}}^{N} \left. \frac{\partial^i P(\underline{X})}{\partial x_{k_1}, \ldots, \partial x_{k_i}} \right|_{\underline{X}^0} (x_{k_1} - x_{k_1}^0) \ldots (x_{k_i} - x_{k_i}^0)}{Q(\underline{X}^0) + \sum_{i=1}^{r} \sum_{k_1=1}^{N} \cdots \sum_{k_i=k_{i-1}}^{N} \left. \frac{\partial^i Q(\underline{X})}{\partial x_{k_1}, \ldots, \partial x_{k_i}} \right|_{\underline{X}^0} (x_{k_1} - x_{k_1}^0) \ldots (x_{k_i} - x_{k_i}^0)} \tag{3.4-4}$$

Equation (3.4-4) may be put in the form of a ϕ-machine. Regrouping the terms of Equation (3.4-4) so that the equation is a functional power series in $\underline{X}$ instead of $(\underline{X} - \underline{X}^0)$. Dividing the result by the constant term of the denominator gives

$$y^* \approx \frac{\tilde{p}_0 + \sum_{i=1}^{r} \sum_{k_1=1}^{N} \cdots \sum_{k_i=k_{i-1}}^{N} \tilde{p}_{k_1, \ldots, k_i} x_{k_1} \cdots x_{k_i}}{1 + \sum_{i=1}^{r} \sum_{k_1=1}^{N} \cdots \sum_{k_i=k_{i-1}}^{N} \tilde{q}_{k_1, \ldots, k_i} x_{k_1} \cdots x_{k_i}} \tag{3.4-5}$$

where

$$p_0 = \frac{P(\underline{X}^0) + \sum_{i=1}^{r} \sum_{k_1=1}^{N} \cdots \sum_{k_i=k_{i-1}}^{N} \left. \frac{\partial^i P(\underline{X})}{\partial x_{k_1}, \ldots, \partial x_{k_i}} \right|_{\underline{X}^0} (-x_{k_1}^0) \ldots (-x_{k_i}^0)}{Q(\underline{X}^0) + \sum_{i=1}^{r} \sum_{k_1=1}^{N} \cdots \sum_{k_i=k_{i-1}}^{N} \left. \frac{\partial^i Q(\underline{X})}{\partial x_{k_1}, \ldots \partial x_{k_i}} \right|_{\underline{X}^0} (-x_{k_1}^0) \ldots (-x_{k_i}^0)} \tag{3.4-6}$$

$$\widetilde{p_1} =$$
$$\vdots$$
$$\widetilde{p_{N,\ldots,N}} =$$
$$\widetilde{q_1} =$$
$$\vdots$$
$$\widetilde{q_{N,\ldots,N}} =$$

$$\frac{\left.\dfrac{\partial^r Q(\underline{X})}{\partial^r x_N}\right|_{\underline{X}^O}}{Q(\underline{X}^O) + \sum_{i=1}^{r} \sum_{k_1=1}^{N} \cdots \sum_{k_i=k_{i-1}}^{N} \left.\dfrac{\partial^i Q(\underline{X})}{\partial x_{k_1}, \ldots, \partial x_{k_i}}\right|_{\underline{X}^O} (-x_{k_1}) \ldots (-x_{k_i})}$$

The "optimum" ϕ-machine to model the system given in Equation (3.4-4) is

$$z = \phi(\underline{X}) = \underline{U_*}^T \underline{G}(\underline{X}) + y^* \underline{V_*}^T \underline{H}(\underline{X}) \tag{3.4-7}$$

where the "optimum" weight vectors are

$$M = \binom{N+r}{r} - 1 \tag{3.4-8}$$

$$\underline{U_*} = \begin{pmatrix} u_1 \\ \vdots \\ u_{M+1} \end{pmatrix}, \quad \underline{V_*} = \begin{pmatrix} v_1 \\ \vdots \\ v_M \end{pmatrix}$$

$$u_1 = \widetilde{p}_{0,\ldots,0}, \ldots, u_{M+1} = \widetilde{p}_{N,\ldots,N}$$

$$v_1 = \widetilde{q}_{1,\ldots,1}, \ldots, v_M = \widetilde{q}_{N,\ldots,N}$$

and the ϕ-function vectors are

$$\underline{G}(\underline{X}) = \begin{pmatrix} g_1 \\ \vdots \\ g_{M+1} \end{pmatrix}, \quad \underline{H}(\underline{X}) = \begin{pmatrix} h_1 \\ \vdots \\ h_M \end{pmatrix} \tag{3.4-9}$$

$$g_1 = 1,\ g_2 = x_1,\ g_3 = x_2, \ldots, g_{M+1} = x_N^r$$

$$h_1 = x_1,\ h_2 = x_2, \ldots, h_M = x_N^r$$

Consider those systems approximated by Equation (3.4-1) whose outputs are corrupted by additive noise. The observed output of the system may be represented by

$$y_i = \underline{W}_*^T \underline{F}_i + P_R + y_i^* Q_R + N_i \tag{3.4-10}$$

where N_i includes the measurement noise and the error in approximating the physical system by Equation (3.4-1). The error bounds of Section 3.3 can be applied to this case by substituting $P_R + y_i^* Q_R + N_i$ for N_i in Equations (3.3-21) and (3.3-26).

If P_R and Q_R are identically zero, the error bounds and other results are the same as in Section 3.2. When P_R and Q_R are not zero, the system is being approximated by a lower order Volterra series than that which describes the system exactly. The error bound of Equation (3.3-26) for models with inverse terms becomes

$$\left\| \underline{E}_\infty \right\| \leq d_{max} \left[N + \frac{1}{\epsilon} \right] \tag{3.4-11}$$

where

$$d_{max} = \max_i \left[\frac{a}{\left\| \hat{\underline{F}}_i \right\|} (1-\underline{V}_*^T \underline{H}_i)(N_i+P_R+y_i^* Q_R) \; a(1-\underline{V}_*^T \underline{H}_i) \right.$$
$$\left. (N_i+P_R+y_i^* Q_R) + 2(a-1) \hat{\underline{F}}_i^T \underline{E}_i \right]$$

The error bound of Equation (3.3-21) for models without inverse terms becomes

$$\left\| \underline{E}_\infty \right\| \leq d_{max} \left[N + \frac{1}{\epsilon} \right] \tag{3.4-12}$$

where

$$d_{max} = \max_i \left[\frac{a}{\left\| \underline{F}_i \right\|} (N_i + P_R + y_i^* Q_R) \left[a(N_i + P_R + y_i^* Q_R) \right.\right.$$
$$\left.\left. +2(a-1) \underline{F}_i^T \underline{E}_i \right] \right]$$

Therefore, the desired results of approximation of the physical system by a simpler system can be achieved. The magnitude of the error is controlled by the operating region of the physical system, the additive noise, and the convergence factor.

3.5 Algorithm Variations

Once the model given by Equation (2.5-1) is chosen there are many ways to solve for the weight vector. For an on-line training procedure an incremental adjustment of the weight vector seems best. The new weight vector is given by

$$\underline{W}_{i+1} = \underline{W}_i + a \, \Delta \underline{W}_i \tag{3.5-1}$$

where $\underline{W}_i$, a, $\Delta \underline{W}_i$ are the old weight vector, convergence factor, and weight vector increment respectively. As the convergence factor a is decreased from unity, the adaption process will be slowed. Thus, the weight

vector will be affected less by noisy measurements. This should reduce steady state weight vector error and the sensitivity to noise. However, the rate at which the model can follow a time varying system will also be reduced. The weight vector increment is to be calculated from measurements $(y_i, \underline{X}_i)$ taken from the operating record of the system. These measurements are transformed into $(y_i, \underline{F}_i)$ the measurement output of the system and the ϕ-functions of the model. The error-correcting procedure of Section 3.1 was chosen for its simplicity in using only the most recent measurement for calculating the weight vector increment. This section investigates the factors involved in choosing a more complex learning algorithm.

An algorithm more complex than the error correcting algorithm may use multiple sets of measurements. The choice of such an algorithm will involve a trade-off between computational complexity and accuracy or rate of adaption. The factors of concern in computational complexity are storage, time and processor complexity. The method and equipment used will determine the accuracy with which the system can be modeled. They will also determine how fast the model can track a time varying system.

Consider the problem of finding the weight vector increment based on the stated values of the measurements from N selected data points. Let these measurements be denoted by

$$y_j, \underline{F}_j \qquad j = 1, N \tag{3.5-2}$$

The output of the present model corresponding to the j^{th} system output y_j is given by

$$z_j = \underline{W}_i^T \underline{F}_j \tag{3.5-3}$$

This problem can now be expressed as trying to find the best solutions for $\Delta \underline{W}_i$ to the following set of simultaneous linear equations

$$y_j - z_j = \Delta \underline{W}_i^T \underline{F}_j \qquad j = 1, N \tag{3.5-4}$$

The solution to this problem is not unique unless the $\underline{F}_j$'s form a basis for the ϕ-space. That is, if the determinant of the matrix formed by the vectors $\underline{F}_j$ $(j = 1, N)$ is non-singular. This is true if and only if ϕ-space is an N-space and the $\underline{F}_j$ form a linearly independent set of vectors. The error between the present model output z_j and the system output y_j can be viewed as the projection of the current weight error vector $\underline{E}_i = \underline{W}_* - \underline{W}_i$ onto the ϕ-function vector $\underline{F}_j$.

$$y_j - z_j = \underline{E}_i^T \underline{F}_j \tag{3.5-5}$$

The weight vector increment must be a linear combination of the $\underline{F}_j$'s because the components of $\underline{E}_i$ in directions other than the $\underline{F}_j$'s (if any) are unknown. Therefore, the weight vector increment is given by

$$\Delta \underline{W}_i = \sum_{j=1}^{N} c_j \underline{F}_j \tag{3.5-6}$$

If the $\underline{F}_j$'s are linearly independent and $\Delta \underline{W}_i$ is assumed to be a linear combination of the $\underline{F}_j$'s, then the solution to Equation (3.5-4) becomes unique. If the $\underline{F}_j$'s are not linearly independent, the c_j's may be picked on basis of a minimum square error procedure or a similar method. Such a method can be derived to suit special purposes from techniques developed in the literature for mathematical programming.

The choice of an algorithm depends on the nature of the process and the requirements placed on the identification scheme. There are several factors that can be varied during the execution of the training procedure. These factors are convergence factor, number of data points, and the method for calculating the weight vector increment. "Convergence" for all these variations is guaranteed by simple extensions of Section 3.3. The simplest extension of the error correcting algorithm would be to iterate the error correcting algorithm on several data points between measurement times. The next step would be a projection algorithm to calculate the weight vector increment based on a set of linearly independent $\underline{F}_j$'s. Finally, a weighted least squares procedure is the most complex algorithm. The error correcting algorithm and the projection algorithm have been simulated. The results are given in Chapter IV for a variety of processes and conditions.

3.6 Projection Algorithm

The training procedure when using the projection algorithm will follow the sequence of steps in Section 3.1 with the error correcting algorithm of Step 4 replaced by the projection algorithm. The projection algorithm will take the new $(y_i, \underline{F}_i)$ and add it to the set of stored $(y_i, \underline{F}_i)$'s previously received. The set of vectors will be adjusted so that it is still linearly independent. Then based on the current weight vector $\underline{W}_i$ and the stored $(y_i, \underline{F}_i)$, the projection algorithm will calculate the weight vector increment, $\Delta \underline{W}_i$.

First the projection algorithm must determine if the new $\underline{F}_i$ is linearly independent of the stored $\underline{F}_i$'s. The stored $\underline{F}_i$'s are a linearly independent set by definition and therefore form a basis of a sub-space in ϕ-space. Let this basis be denoted by Ψ.

$$\Psi = \left[\ldots, \underline{F}_i, \ldots\right] \tag{3.6-1}$$

The projection matrix Ψ^* for this sub-space will project any vector in ϕ-space onto the Ψ sub-space.

$$\Psi^* = \Psi(\Psi^T \Psi)^{-1} \Psi^T \tag{3.6-2}$$

$\underline{F}_i$ is linearly independent of the stored $\underline{F}_i$'s only if the projection of $\underline{F}_i$ onto the Ψ sub-space doesn't equal $\underline{F}_i$. However, if the difference between $\underline{F}_i$ and its projection is very small, the $\underline{F}_i$ will be considered linearly dependent in order to avoid numerical difficulties in computation. Therefore, $\underline{F}_i$ will be considered linearly dependent if

$$\|\underline{F}_i - \Psi^* \underline{F}_i\| < \epsilon \|\underline{F}_i\| \quad 0 < \epsilon < 1 \tag{3.6-3}$$

In such a case one or more stored $(y_i, \underline{F}_i)$'s will be discarded so that Equation (3.6-3) is no longer satisfied. Since $\underline{F}_i$ is now linearly independent of the stored $\underline{F}_i$'s, $(y_i, \underline{F}_i)$ will be added to the set of stored $(y_i, \underline{F}_i)$'s. Let $\underline{Y}$ be the vector of stored y_i's in the same order as the corresponding $\underline{F}_i$'s in the Ψ matrix.

The weight vector increment $\Delta \underline{W}_i$ must satisfy Equations (3.5-4) and (3.5-6) which can now be written as:

$$\underline{Y} - \Psi^T \underline{W}_i = \Psi^T \Delta \underline{W}_i \tag{3.6-4}$$

$$\Delta \underline{W}_i = \Psi \underline{c} \tag{3.6-5}$$

These equations can be solved for the vector $\underline{c}$ which indicates the necessary combination of stored $\underline{F}_i$'s in $\Delta \underline{W}_i$.

$$\underline{Y} - \Psi^T \underline{W}_i = \Psi^T \Psi \underline{c}$$

$$\underline{c} = (\Psi^T \Psi)^{-1} (\underline{Y} - \Psi^T \underline{W}_i)$$

The desired $\Delta \underline{W}_i$ is obtained by substituting $\underline{c}$ back into Equation (3.6-5).

$$\Delta \underline{W}_i = \Psi(\Psi^T \Psi)^{-1} (\underline{Y} - \Psi^T \underline{W}_i) \tag{3.6-6}$$

Thus Equation (3.1-4) can be replaced by

$$\underline{W}_{i+1} = \underline{W}_i + a \Psi(\Psi^T \Psi)^{-1} (\underline{Y} - \Psi^T \underline{W}_i) \tag{3.6-7}$$

The vector equation for the observed system output corresponding to Equation (3.3-2) is

$$\underline{Y} = \Psi^T \underline{W}_* + \underline{N} \tag{3.6-8}$$

Equation (3.6-6) becomes

$$\Delta \underline{W}_i = \Psi(\Psi^T \Psi)^{-1} \Psi^T(\underline{W}_* - \underline{W}_i) + \Psi(\Psi^T \Psi)^{-1} \underline{N} \tag{3.6-9}$$

$$= \Psi^* \underline{E}_i + \Psi(\Psi^T \Psi)^{-1} \underline{N}$$

$\Delta \underline{W}_i$ is the projection of the weight error vector onto the sub-space plus an error due to measurement noise and modeling error. Hence, this method is named the projection algorithm.

CHAPTER IV

SIMULATIONS OF MODEL TRACKING IDENTIFICATION

4.1 Introduction

The model tracking system of Figure (3.1) was simulated on an IBM 360/50. The input to the plant in Figure (3.1) was selected in one of two ways for each of the identification runs. A single sample of uncorrelated noise from a digital Gaussian noise generator was used as the input for those runs made to determine the ability of the technique to identify a particular system under given conditions. The effects of identification and control on each other were investigated by computing the input necessary to force the learning model output to follow a desired trajectory. This input was computed at each iteration and applied to the plant. The additive noise in Figure (3.1) was provided by the digital Gaussian noise generator and was independent of the input noise sequence. In order to achieve a desired correlation, the additive noise was passed through a first order filter. After each iteration of the training procedure, the normalized weight vector error $(\|\underline{E}_i\|/\|\underline{W}_*\|)^{1/2}$ and the rms errors between the outputs of the plant, the learning model, and the "optimum" model were computed.

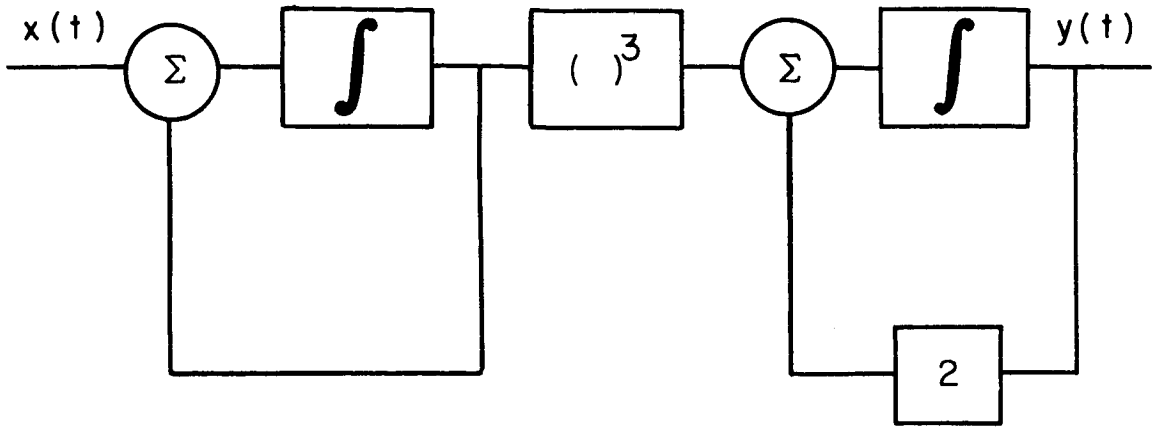
The system identification technique was tried on a linear time invariant system, a linear time varying system, and two nonlinear systems. An underdamped second order linear system was used to study the effects of varying the convergence factor a, the standard deviation σ of the additive noise, and the first order correlation ρ of the additive noise.

The system used had a damping ratio of .02 and a natural frequency of one radian per second and was sampled at one second intervals. A five state model of the Saturn booster was used to test the ability of the technique to follow a time varying plant. A description of the booster model can be found in Appendix B. The discrete plant matrices were computed from the continuous plant matrices at one second intervals for a sampling interval of one tenth of a second. Figure (4.1) shows the cubic and inverse systems which are the two nonlinear plants used to study the ability of the technique to identify nonlinear plants. The cubic system is two first order linear systems separated by a pure cubic nonlinearity. The inverse system is the second order underdamped linear system preceded by a nonlinearity which is the ratio of a fifth order polynomial to a fourth order polynomial which has no real roots. The inverse system was also used to study the dynamic linearization of a nonlinear plant.

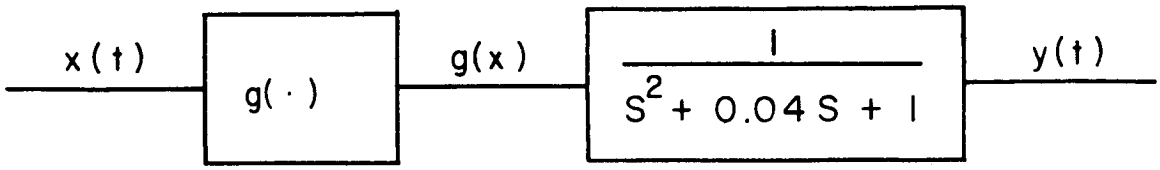
The results of the simulations are divided into two groups: those for which the optimum model is stationary and those for which the optimum model is non-stationary. The normalized weight vector error was plotted on a logarithmic scale. This was done to emphasize small values of the normalized weight vector error and to transform the curves which should be exponentials into straight lines.

4.2 Results for Stationary "Optimum" Models

In order to provide insight into how the technique searches for the correct weight vector, a twenty tap delay line model was used for the second order linear system with the error-correcting algorithm of Section 3.1.



CUBIC SYSTEM



$$g(x) = \frac{0.2x + 1.2x^3 + 0.04x^5}{1.0 + 1.04x^2 + 0.04x^4}$$

INVERSE SYSTEM

FIGURE 4.1

The "optimum" weight vector is the sampled impulse response of the system. The components of the "optimum" weight vector and the actual weight vector after five, ten, twenty, and forty iterations are plotted in Figure (4.2) versus their position on the delay line. This plot shows that the components of the actual weight vector approach the sampled impulse response quite quickly, although not monotonically. Both the plant and the model outputs are plotted in Figure (4.3) versus the number of iterations or elapsed time in seconds. This reveals that the learning model tracks the system to the accuracy of the graph (0.013) after eighty seconds. The normalized weight vector error plotted in Figure (4.4) decreases exponentially in Figure (4.4) until a lower limit is reached at about 0.02. This limit is the result of the error between the plant and the "optimum" model outputs which has a rms value of about .02.

The second order system was identified with three configurations of the model tracking system. First, the error correcting algorithm was used with the twenty tap delay-line model. Then the error correcting algorithm was used with a second order difference equation model. Finally, the projection algorithm was used with the difference equation model. All three situations were simulated for a wide variety of conditions. The convergence factor a, the standard deviation of the additive noise σ, and the correlation of the additive noise ρ were given values of 1.0, 0.5, 0.25; 0.0, 0.01, 0.1, 1.0; 0.0, 0.5, 0.707 respectively. Identification runs were made with the weight vector initialized both to zero and to the "optimum" weight vector for all possible combinations of a, σ, and ρ.

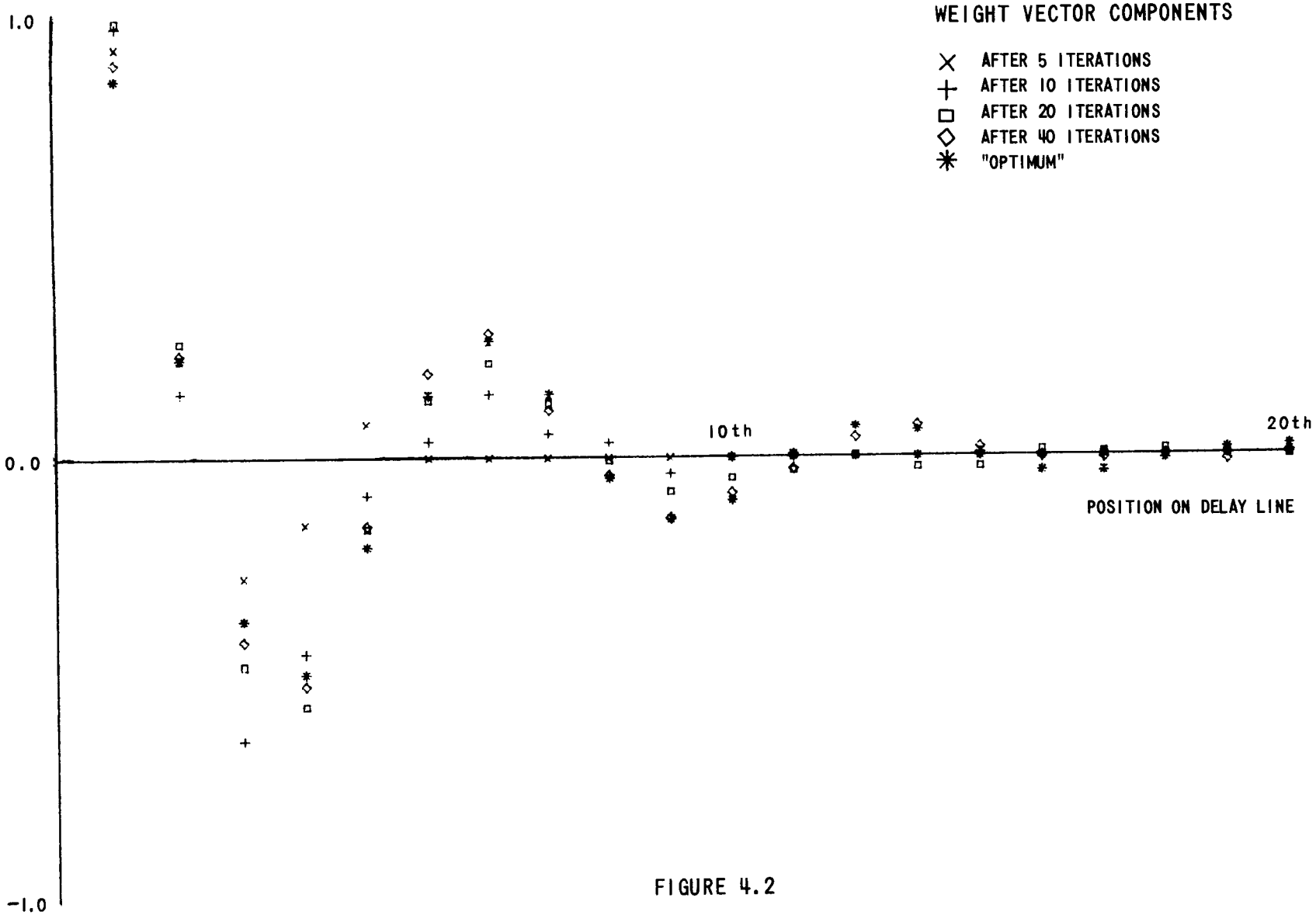
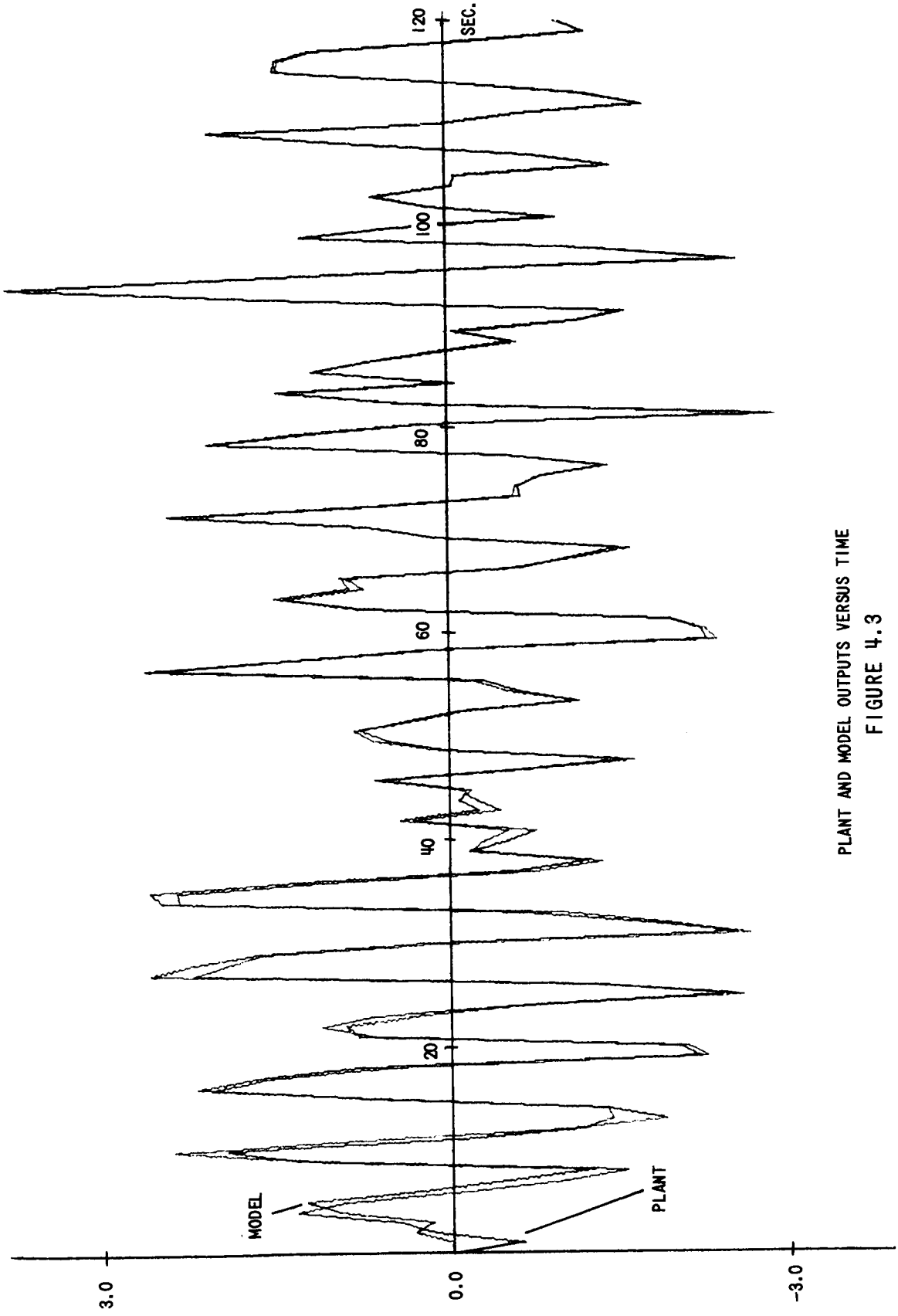

WEIGHT VECTOR COMPONENTS
AFTER 5 ITERATIONS
AFTER 10 ITERATIONS
AFTER 20 ITERATIONS
AFTER 40 ITERATIONS
"OPTIMUM"
1.0
0.0
-1.0
10th
20th
POSITION ON DELAY LINE

FIGURE 4.2



PLANT AND MODEL OUTPUTS VERSUS TIME

FIGURE 4.3

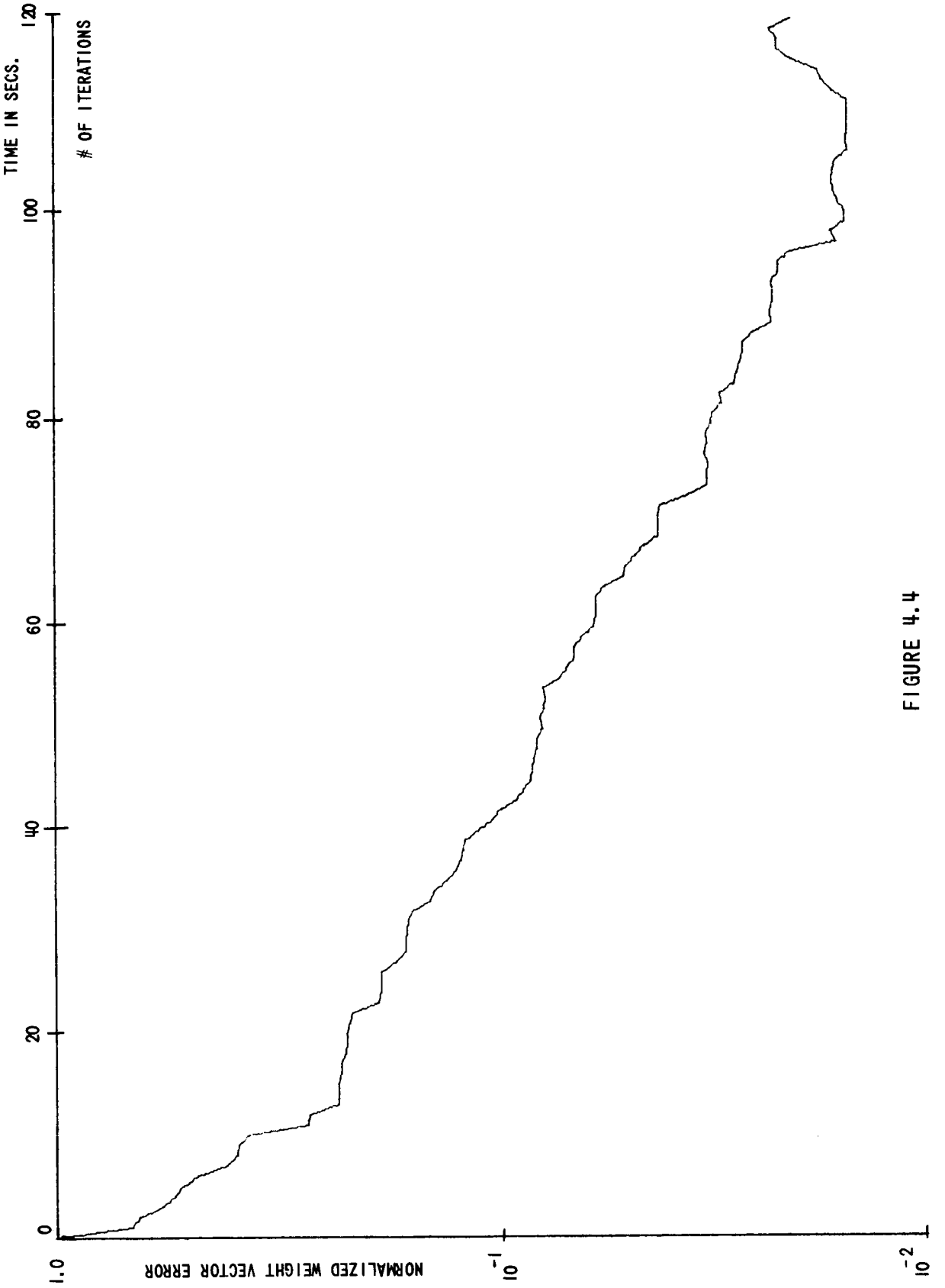

TIME IN SECS.
OF ITERATIONS
0
20
40
60
80
100
120
NORMALIZED WEIGHT VECTOR ERROR
1.0
10^{-1}
10^{-2}

FIGURE 4.4

Figures (4.5) through (4.11) are plots of data selected from these identification runs. Table II contains auxiliary information about these runs and the identification of the curves. Generally, the normalized weight vector error approached a limiting value as the time increased and oscillated about it. The normalized weight vector error curves are plotted in pairs. The first curve is for $\underline{W}_1 = \underline{0}$ and the second curve is for $\underline{W}_1 = \underline{W}_*$. The pairs of curves approach a mutual asymptote. The $\underline{W}_1 = \underline{W}_*$ curves approach the asymptote more quickly than the $\underline{W}_1 = \underline{0}$ curves. Therefore, the mean normalized weight vector error for the $\underline{W}_i = \underline{W}_*$ curves closely approximates the mutual asymptote of the pair of curves.

The effect of increasing ρ was to increase the asymptote of the normalized weight vector error curve. However, this increase was slight in most cases. Figure (4.5) shows the results for the most sensitive case.

Figures (4.6), (4.7) and (4.8) show the effects of increasing the standard deviation σ of the additive noise. The asymptotes of the curves are approximately proportional to the standard deviation σ. The step in curve E of Figure (4.6) is due to the zero initial conditions of the plant. The modeling error caused by having only twenty taps on the delay line is zero until after twenty seconds and then it has a rms value of 0.0146. The asymptotes for a given σ are greatest in Figure (4.8) and least in Figure (4.6). The difference equation model with the error correcting algorithm is only slightly more sensitive than the delay line model with the error correcting algorithm. The difference equation model

TABLE II

AUXILIARY INFORMATION ABOUT IDENTIFICATION OF THE LINEAR, SECOND ORDER UNDERDAMPED PLANT

E_1 = RMS error between y_i and y_i^*

E_2 = RMS error between z_i and y_i

E_3 = RMS error between y_i^* and z_i

E_4 = Mean normalized weight vector error

FIGURE 4.5 Difference Equation Model, Error Correcting Algorithm, a = 1.0, σ = 0.1

Plot	ρ	$\underline{W}_1$	E_1	E_2	E_3	E_4
A	0.0	$\underline{0}$	.1002	.2016	.1809	.2313
B	0.5	$\underline{0}$	.1003	.2071	.1866	.2392
C	0.707	$\underline{0}$	.1008	.2258	.1957	.2552
D	0.0	$\underline{W}_*$	.1002	.1560	.1232	.1125
E	0.5	$\underline{W}_*$	.1003	.1625	.1278	.1241
F	0.707	$\underline{W}_*$	.1008	.1823	.1388	.1532

FIGURE 4.6 Delay Line Model, Error Correcting Algorithm, a = 1.0, ρ = 0.00

Plot	σ	$\underline{W}_1$	E_1	E_2	E_3	E_4
A	0.0	$\underline{0}$	.01459	.1534	.1528	.1864
B	0.01	$\underline{0}$	.01785	.1549	.1532	.1871
C	0.1	$\underline{0}$	.1015	.2184	.1851	.2168
D	1.0	$\underline{0}$	1.003	1.486	1.042	1.065
E	0.0	$\underline{W}_*$	.01459	.02103	.01355	.01277
F	0.01	$\underline{W}_*$	.01785	.02647	.01717	.01607
G	0.1	$\underline{W}_*$	.1015	.1499	.1040	.1043
H	1.0	$\underline{W}_*$	1.003	1.471	1.029	1.041

TABLE II (Cont'd)

FIGURE 4.7 Difference Equation Model, Error Correcting Algorithm, a = 1.0, ρ = 0.0

Plot	σ	$\underline{W}_1$	E_1	E_2	E_3	E_4
A	0.0	$\underline{0}$	9.154E-7	.1409	.1409	.2039
B	0.01	$\underline{0}$	.01002	.1405	.1406	.2040
C	0.1	$\underline{0}$	.1002	.2016	.1809	.2313
D	1.0	$\underline{0}$	1.002	1.555	1.231	1.140
E	0.0	$\underline{W}_*$	9.154E-7	9.235E-7	1.051E-7	6.695E-7
F	0.01	$\underline{W}_*$	.01002	.01560	.01232	.01125
G	0.1	$\underline{W}_*$	.1002	.1560	.1232	.1125
H	1.0	$\underline{W}_*$	1.002	1.560	1.232	1.125

FIGURE 4.8 Difference Equation Model, Projection Algorithm, a = 1.0, ρ = 0.0

Plot	σ	$\underline{W}_1$	E_1	E_2	E_3	E_4
A	0.0	$\underline{0}$	9.154E-7	.06102	.06102	.1395
B	0.01	$\underline{0}$	.01002	.9094	.9091	.8752
C	0.1	$\underline{0}$	.1002	9.073	9.070	8.641
D	0.0	$\underline{W}_*$	9.154E-7	2.289E-5	2.293E-5	2.084E-5
E	0.01	$\underline{W}_*$	.01002	.9072	.9070	.8640
F	0.1	$\underline{W}_*$	.1002	9.073	9.070	8.640

TABLE II (Cont'd)

FIGURE 4.9 Delay Line Model, Error Corrective Algorithm, $\sigma = 0.1$, $\rho = 0.0$

Plot	a	$\underline{W}_1$	E_1	E_2	E_3	E_4
A	1.0	$\underline{0}$	.1015	.2184	.1851	.2168
B	0.5	$\underline{0}$	.1015	.2619	.2415	.2979
C	0.25	$\underline{0}$	.1015	.4093	.4026	.4381
D	1.0	$\underline{W}_*$	.1015	.1499	.1040	.1043
E	0.5	$\underline{W}_*$	.1015	.1185	.0511	.05958
F	0.25	$\underline{W}_*$	.1015	.1084	.03244	.03422

FIGURE 4.10 Difference Equation Model, Error Correcting Algorithm, $\sigma = 0.1$, $\rho = 0.0$

Plot	a	$\underline{W}_1$	E_1	E_2	E_3	E_4
A	1.0	$\underline{0}$	.1002	.2016	.1809	.2313
B	0.5	$\underline{0}$	.1002	.2028	.1866	.2876
C	0.25	$\underline{0}$	.1002	.2679	.2599	.4044
D	1.0	$\underline{W}_*$	.1002	.1560	.1232	.1125
E	0.5	$\underline{W}_*$	.1002	.1199	.07438	.05808
F	0.25	$\underline{W}_*$	.1002	.1109	.05397	.05613

FIGURE 4.11 Difference Equation Model, Projection Algorithm, $\sigma = 0.01$, $\rho = 0.0$

Plot	a	$\underline{W}_1$	E_1	E_2	E_3	E_4
A	1.0	$\underline{0}$	.01002	.9094	.9091	.8752
B	0.5	$\underline{0}$	.01002	.5272	.5272	.5249
C	0.25	$\underline{0}$	.01002	.3284	.2860	.3767
D	1.0	$\underline{W}_*$	.01002	.9072	.9070	.8640
E	0.5	$\underline{W}_*$	.01002	.5242	.5242	.5037
F	0.25	$\underline{W}_*$	.01002	.3197	.3197	.3374

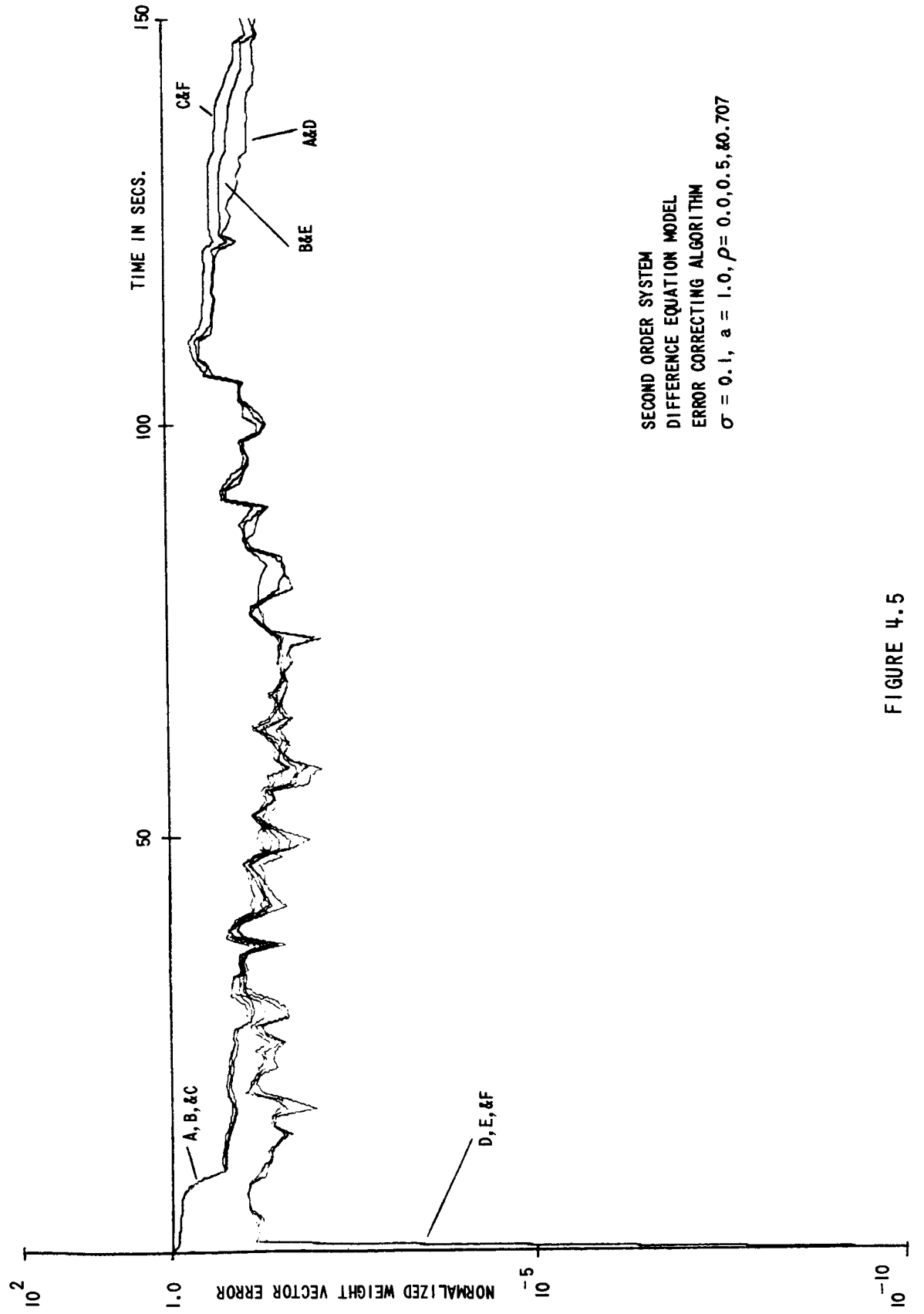

SECOND ORDER SYSTEM
DIFFERENCE EQUATION MODEL
ERROR CORRECTING ALGORITHM
σ = 0.1, a = 1.0, ρ= 0.0,0.5,&0.707
TIME IN SECS.
150
100
50
C&F
A&D
B&E
A, B, &C
D, E, &F
NORMALIZED WEIGHT VECTOR ERROR
10^2
1.0
10^-5
10^-10

FIGURE 4.5

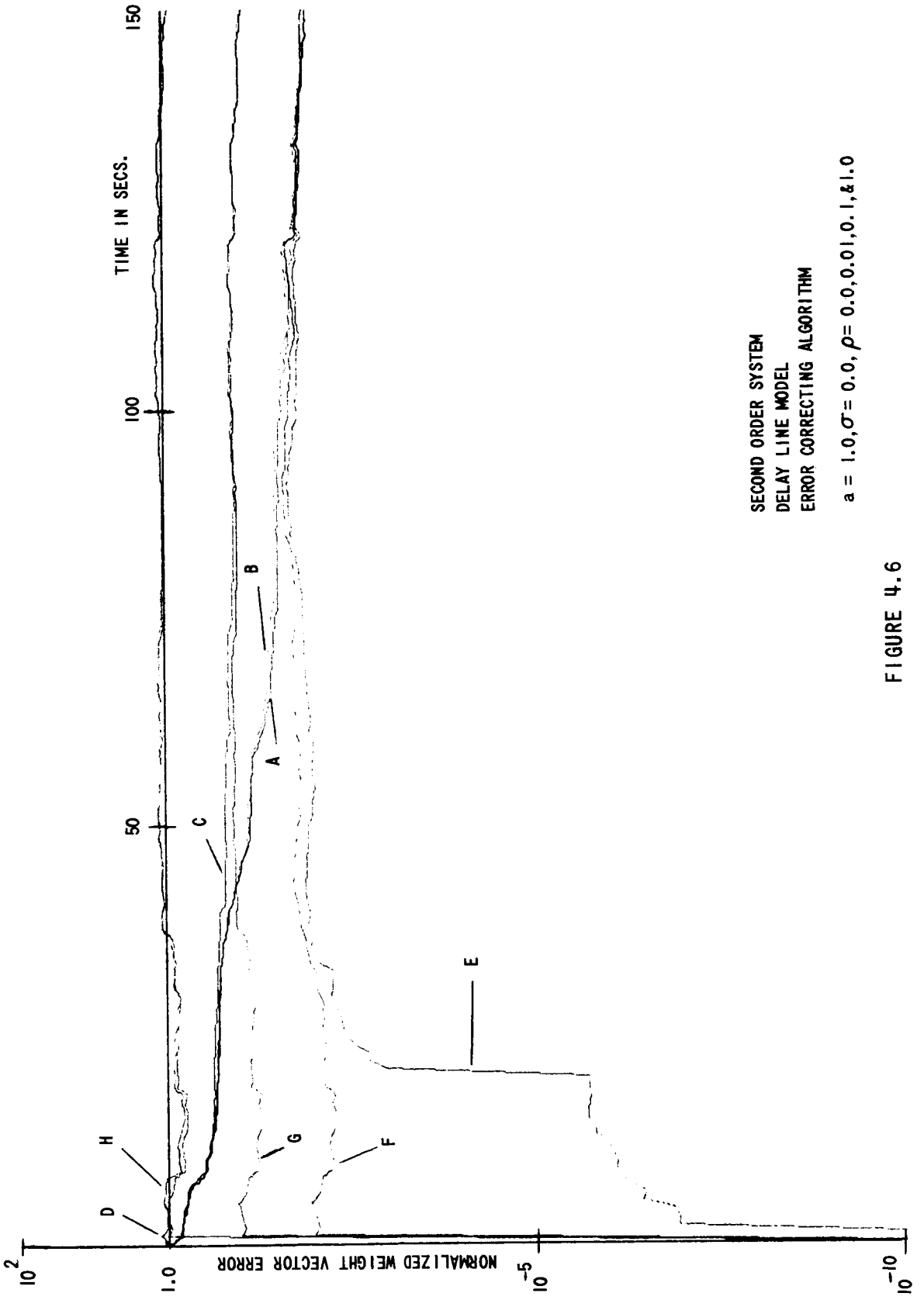

TIME IN SECS.
150
100
50
NORMALIZED WEIGHT VECTOR ERROR
10^{2}
1.0
10^{-5}
10^{-10}
A
B
C
D
E
F
G
H
SECOND ORDER SYSTEM
DELAY LINE MODEL
ERROR CORRECTING ALGORITHM
a = 1.0, σ = 0.0, ρ = 0.0, 0.01, 0.1, & 1.0

FIGURE 4.6

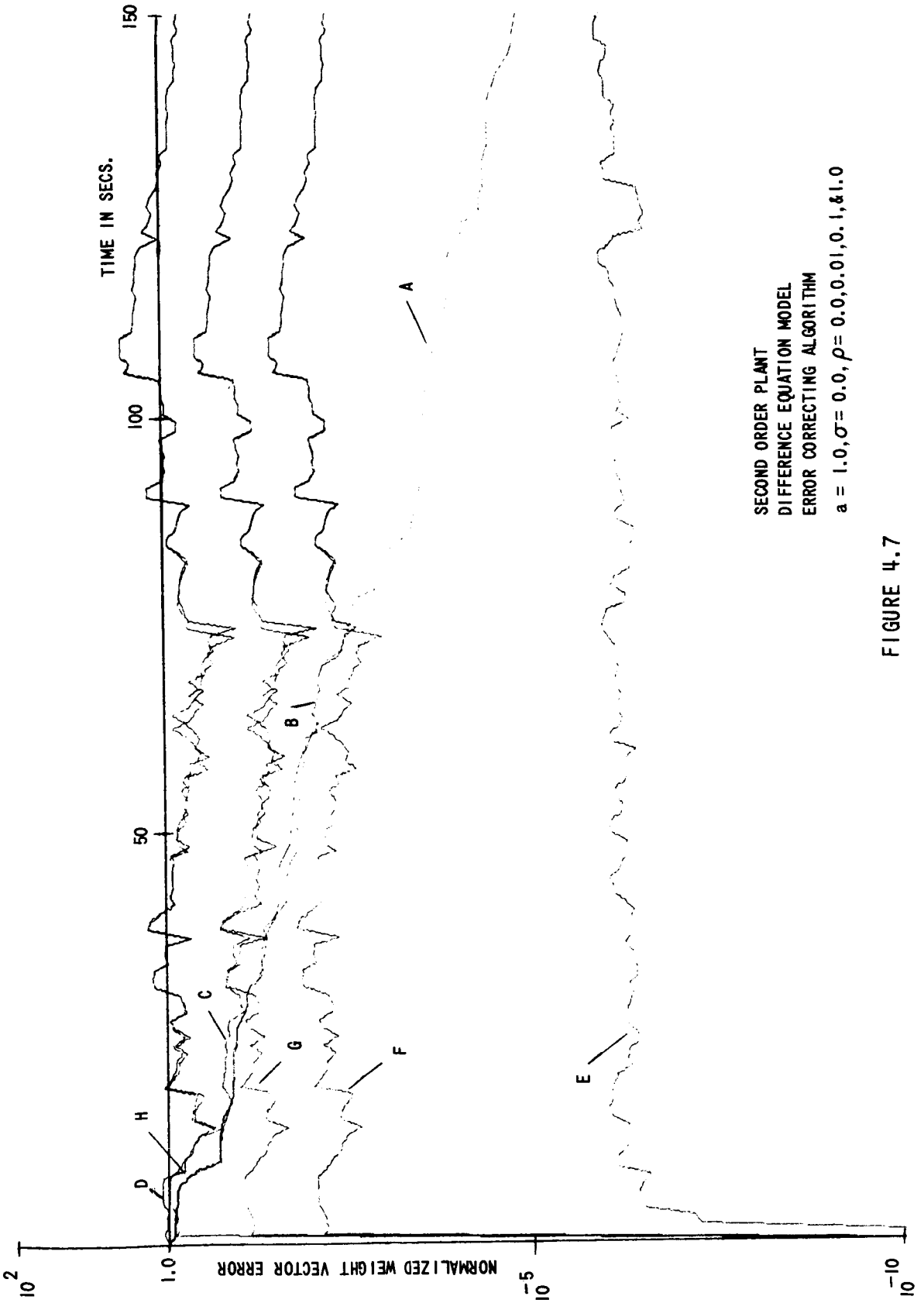

TIME IN SECS.
150
100
50
1.0
NORMALIZED WEIGHT VECTOR ERROR
10^{2}
10^{-5}
10^{-10}
A
B
C
D
E
F
G
H
SECOND ORDER PLANT
DIFFERENCE EQUATION MODEL
ERROR CORRECTING ALGORITHM
a = 1.0, σ = 0.0, ρ = 0.0, 0.01, 0.1, & 1.0

FIGURE 4.7

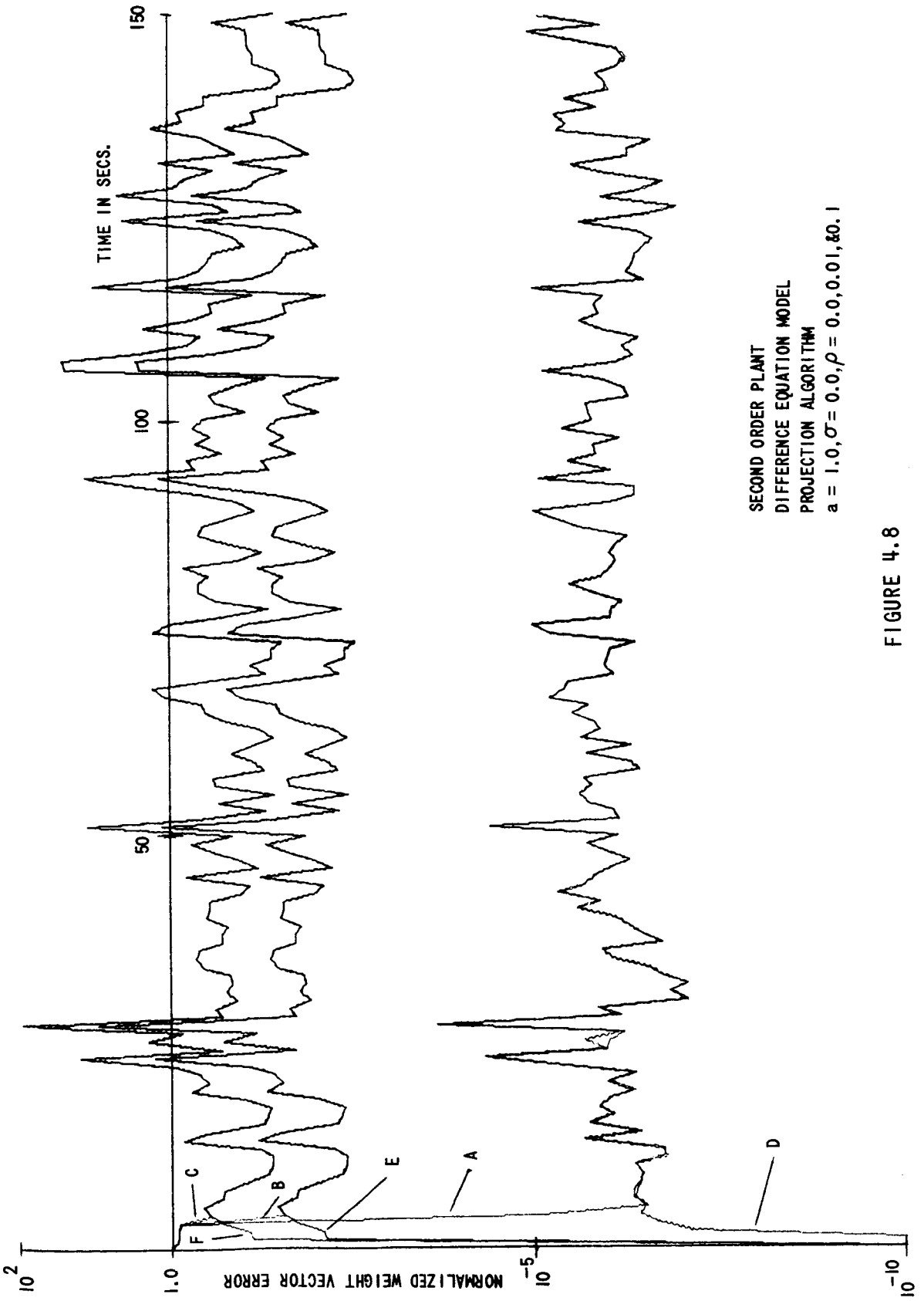

TIME IN SECS.
150
100
50
NORMALIZED WEIGHT VECTOR ERROR
10^{2}
1.0
10^{-5}
10^{-10}
A
B
C
D
E
F
SECOND ORDER PLANT
DIFFERENCE EQUATION MODEL
PROJECTION ALGORITHM
a = 1.0, σ = 0.0, ρ = 0.0, 0.01, &0.1

FIGURE 4.8

with the projection algorithm is much more sensitive than either of the other combinations. The convergence rate is exponential until the asymptote is reached and is nearly independent of σ. The convergence rate for the difference equation model with the error correcting algorithm is twice as great as that for the delay line model with the error correcting algorithm. As expected, the convergence rate for the difference equation model with the projection algorithm is much greater than the other combinations. Thus, a greater sensitivity to noise accompanies the faster convergence.

Varying the convergence factor a gives the results in Figures (4.9), (4.10), and (4.11). The asymptotes and the convergence rate are both approximately proportional to the convergence factor for all cases. Table II reveals that decreasing the convergence factor provides some control over the identification error. For the cases using the error correcting algorithm, the rms error between the outputs of the plant and the learning model approaches the value of σ as a decreases. The rms error between the outputs of the "optimum" model and the learning model becomes less than the value of σ, as a decreases. Thus, the effects of additive noise can be averaged.

The normalized weight vector error plots of Figure (4.12) result from requiring the learning model to follow the random signal used for the input in the previous runs. The convergence rate is slightly less than that for corresponding identification runs. Runs with $\sigma = 0.1$ were made for $a = 1.0$ and $a = 0.5$. The asymptote of the normalized weight vector error curve for $a = 1.0$ was twice as great (0.04) as that for $a = 0.5$ (0.02).

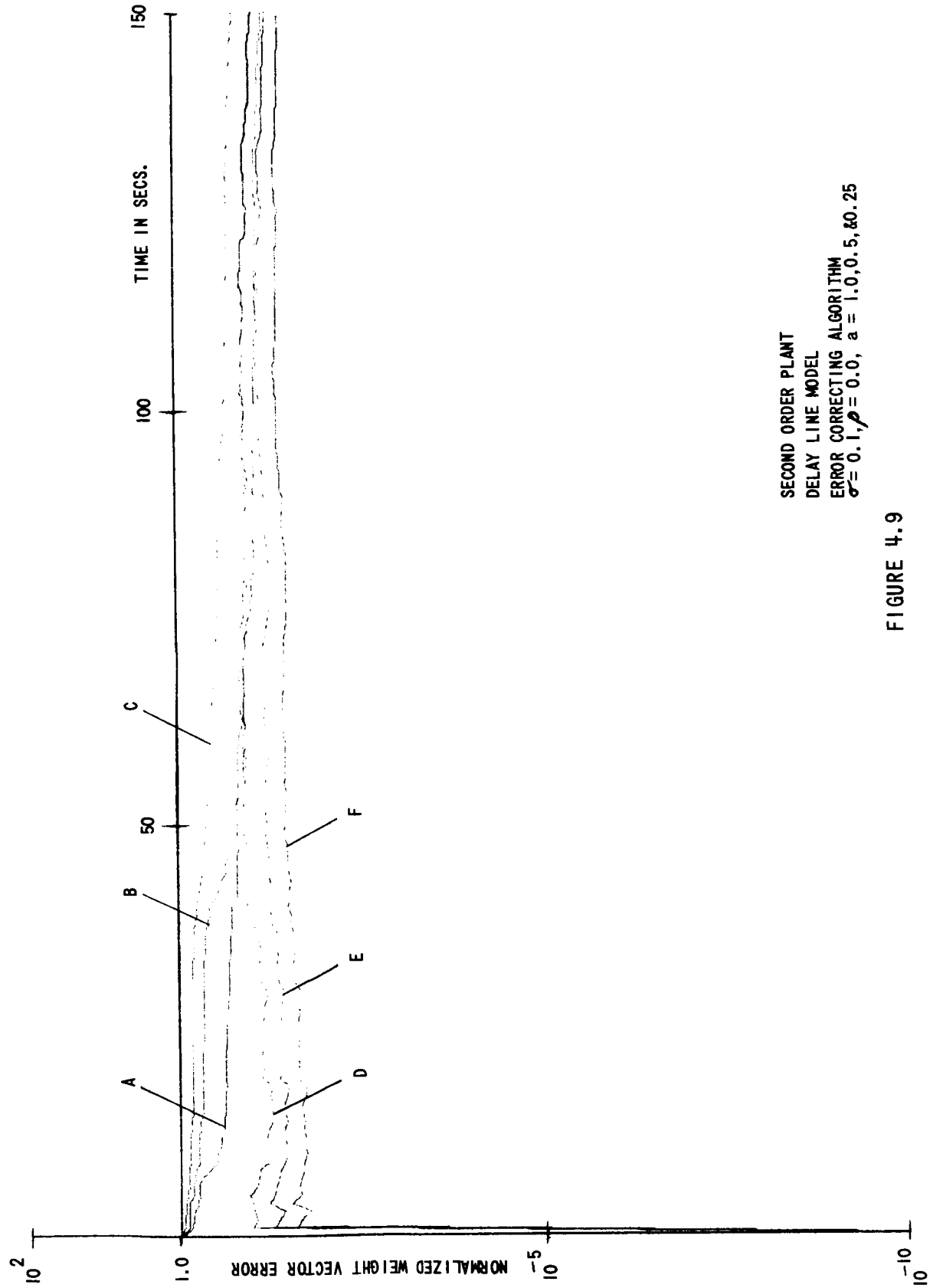
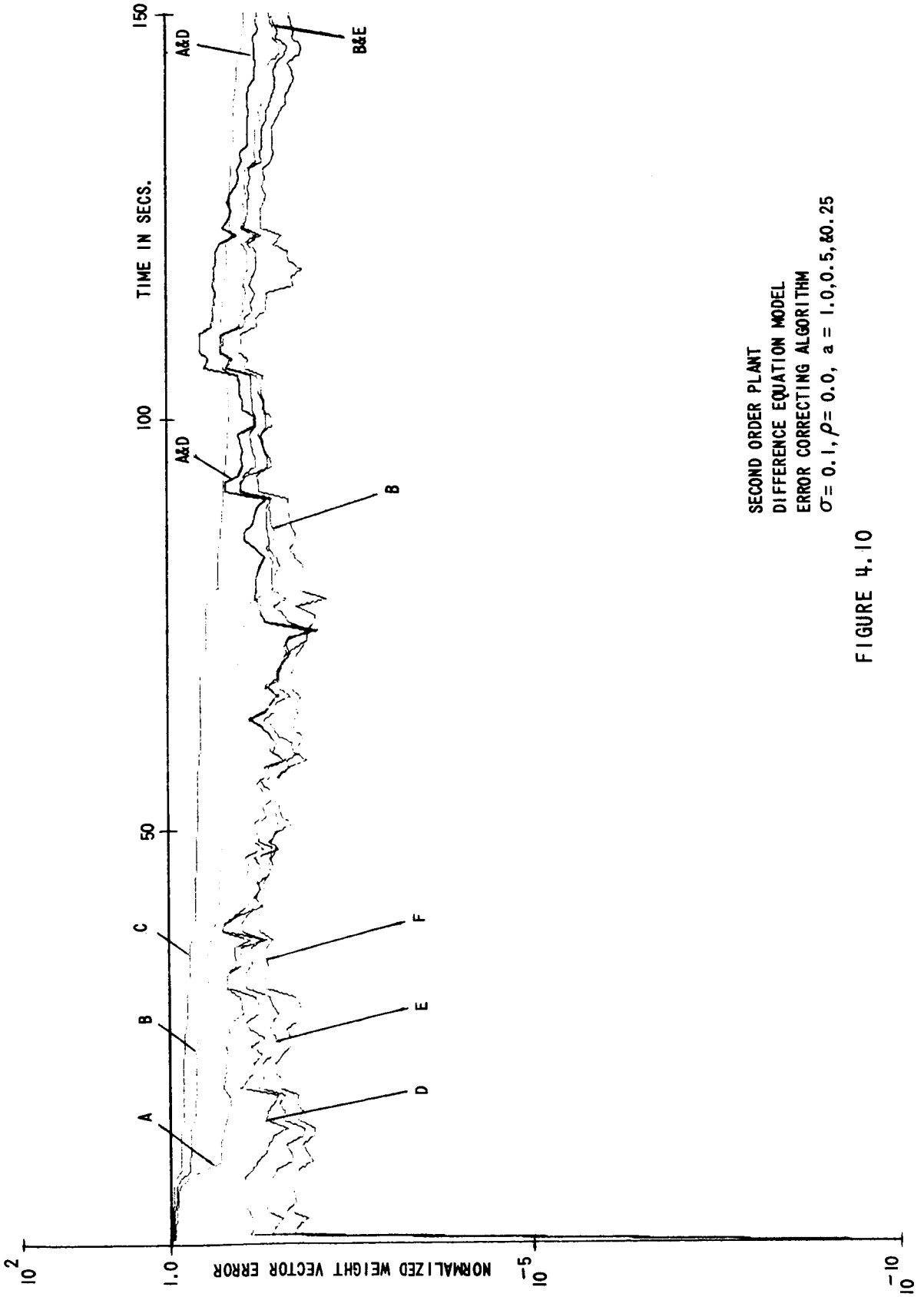

TIME IN SECS.
150
100
50
NORMALIZED WEIGHT VECTOR ERROR
10^{2}
1.0
10^{-5}
10^{-10}
A
B
C
D
E
F
SECOND ORDER PLANT
DELAY LINE MODEL
ERROR CORRECTING ALGORITHM
σ= 0.1, ρ= 0.0, a = 1.0,0.5,&0.25

FIGURE 4.9



SECOND ORDER PLANT
DIFFERENCE EQUATION MODEL
ERROR CORRECTING ALGORITHM
$\sigma = 0.1, \rho = 0.0, a = 1.0, 0.5, \& 0.25$

FIGURE 4.10

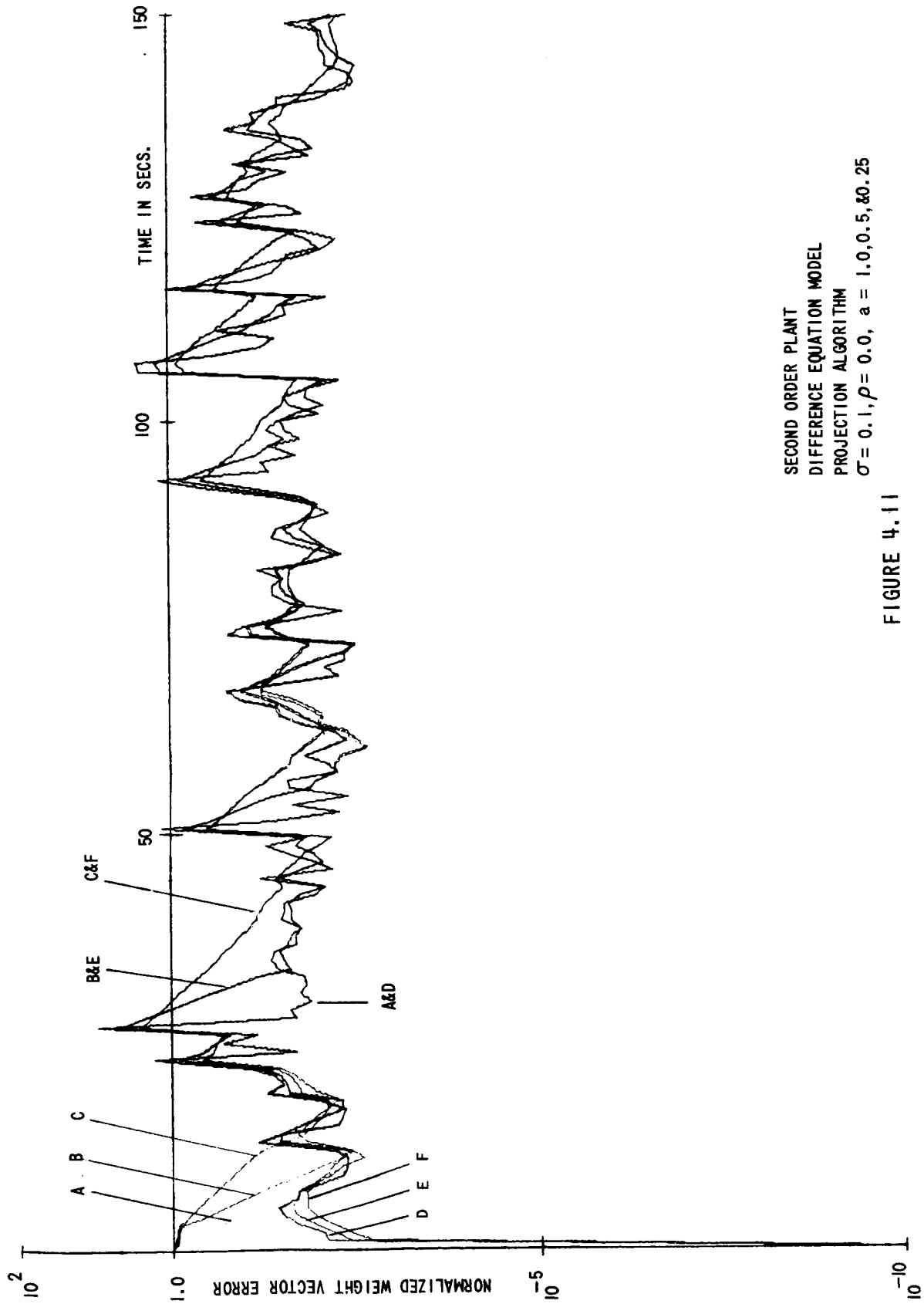

SECOND ORDER PLANT
DIFFERENCE EQUATION MODEL
PROJECTION ALGORITHM
σ= 0.1, ρ= 0.0, a = 1.0,0.5,&0.25
NORMALIZED WEIGHT VECTOR ERROR
TIME IN SECS.
150
100
50
1.0
10^2
10^-5
10^-10
A
B
C
D
E
F
C&F
B&E
A&D

FIGURE 4.11

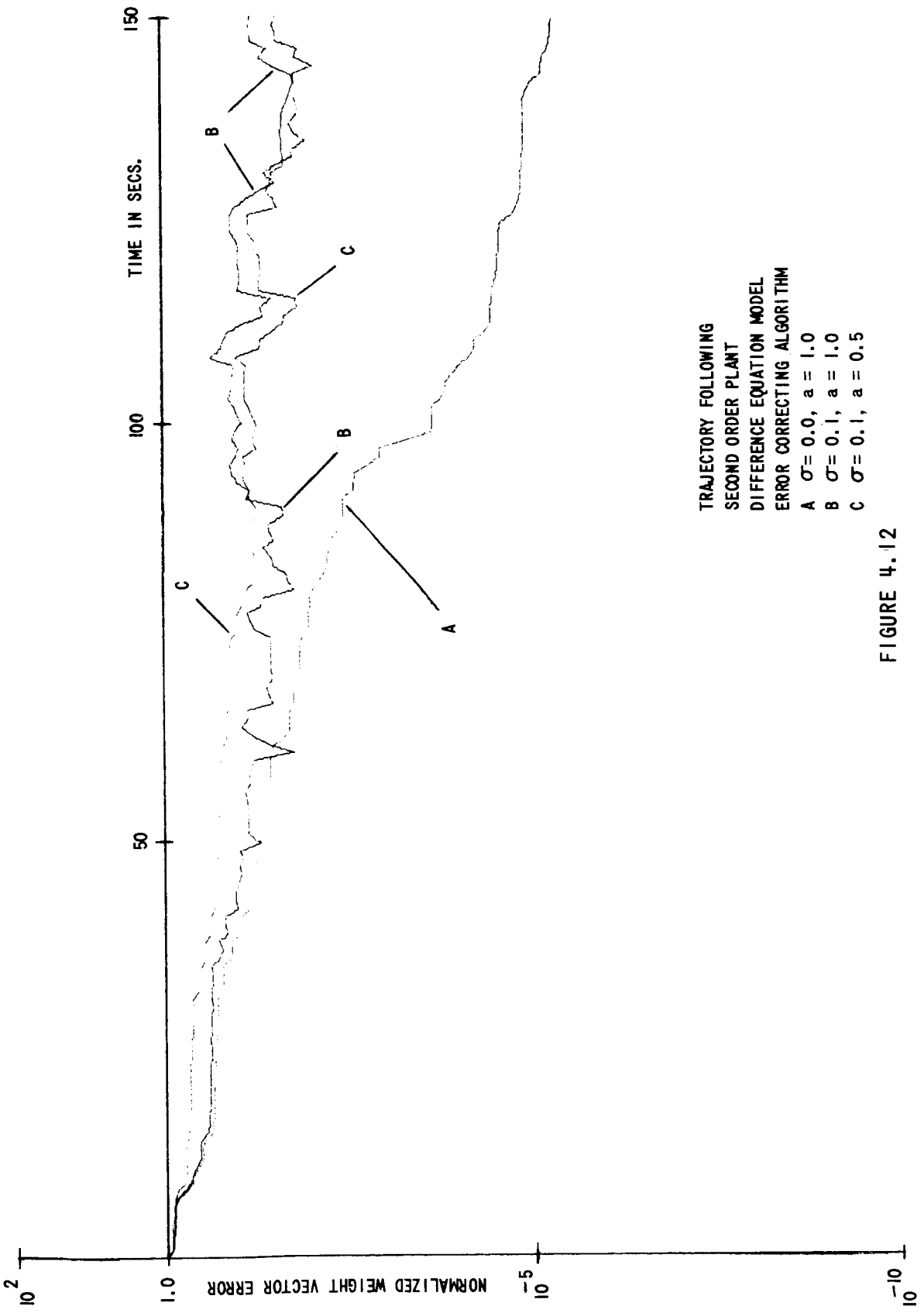

TRAJECTORY FOLLOWING
SECOND ORDER PLANT
DIFFERENCE EQUATION MODEL
ERROR CORRECTING ALGORITHM
A σ= 0.0, a = 1.0
B σ= 0.1, a = 1.0
C σ= 0.1, a = 0.5
TIME IN SECS.
NORMALIZED WEIGHT VECTOR ERROR
150
100
50
A
B
C

FIGURE 4.12

Another indication of the noise controlling ability of the convergence factor is given in Figure (4.13). The rms error between the outputs of the plant and of the learning model approaches the value of σ more closely for $a = 0.5$ than for $a = 1.0$. Thus, choosing the input based on the learning model does not affect the learning characteristics of the identification.

The cubic system was identified using the error correcting algorithm and a second order cubic difference equation model. This model has only eighteen weights. The results of the identification are shown in Figure (4.14). The convergence rate is about half that of the delay line model for the second order system using the error correcting algorithm. This is probably due to the limited range of the input. Again, the asymptote of the normalized weight vector error curve decreases with the convergence factor.

The inverse system was identified with the model tracking system using two combinations of models and training algorithms. The error correcting algorithm was used with an exact difference equation model which had thirty-six numerator weights and eight denominator weights. The projection algorithm was used with a second order difference equation model. This second order model is the model used for the linear system with a constant term added to represent the offset of the operating point.

Figures (4.15) and (4.16) reveal that the first case had much slower convergence than any other case tested. This slow convergence may be due to either the inclusion of inverse terms or the limited range of the input.

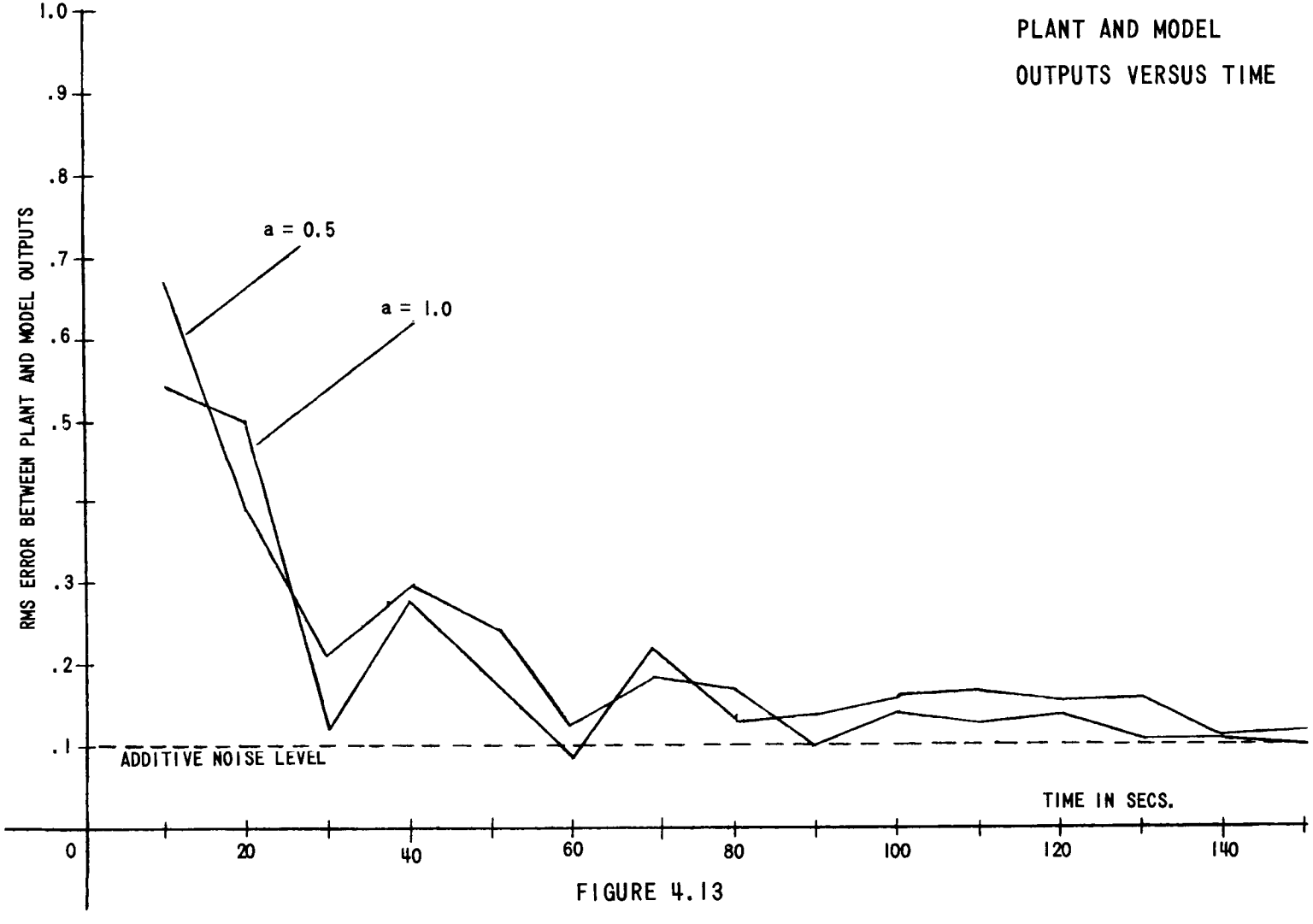

RMS ERROR BETWEEN
PLANT AND MODEL
OUTPUTS VERSUS TIME
RMS ERROR BETWEEN PLANT AND MODEL OUTPUTS
1.0
.9
.8
.7
.6
.5
.3
.2
.1
a = 0.5
a = 1.0
ADDITIVE NOISE LEVEL
TIME IN SECS.
0
20
40
60
80
100
120
140

FIGURE 4.13

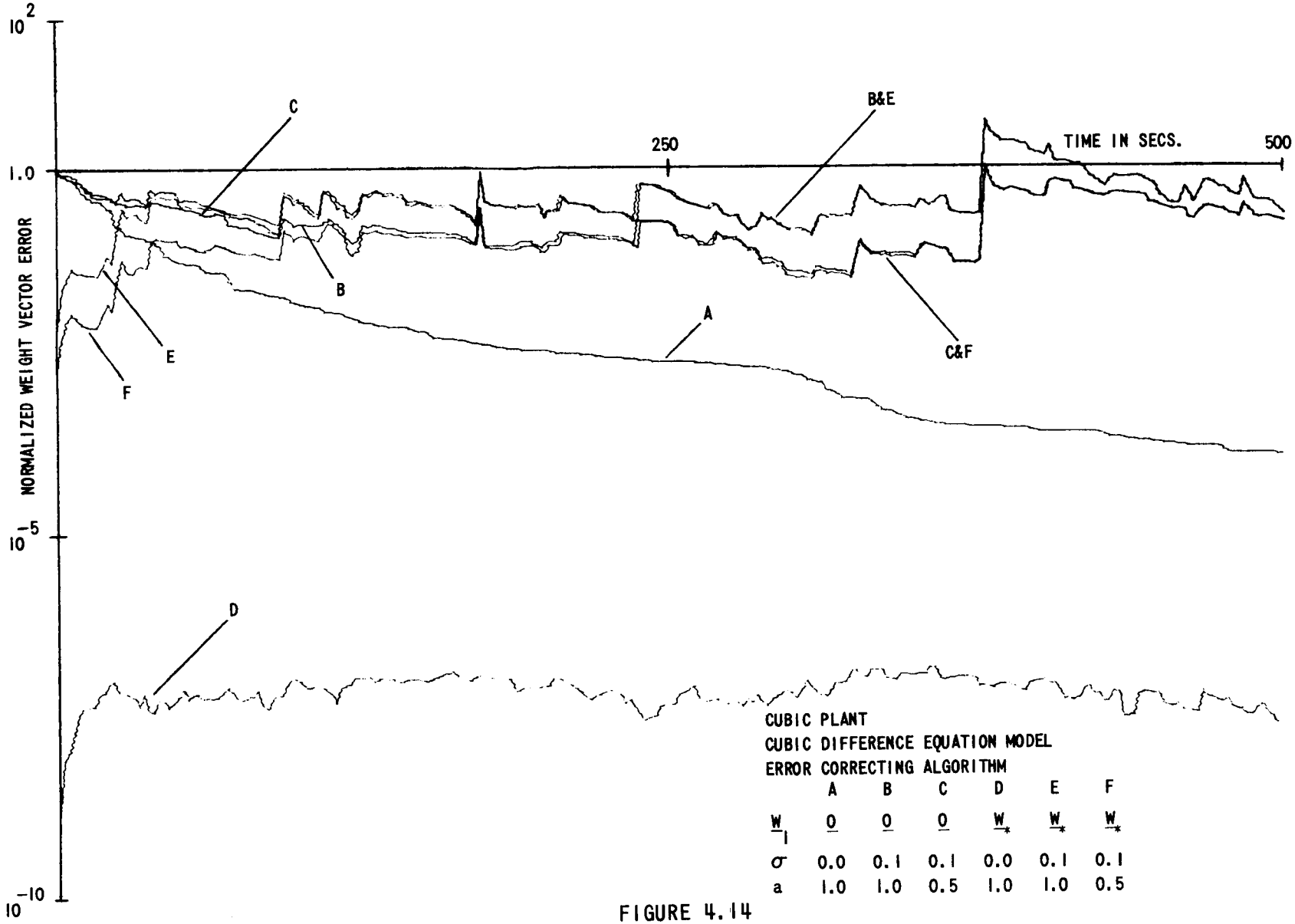

10^2
1.0
10^{-5}
10^{-10}
NORMALIZED WEIGHT VECTOR ERROR
TIME IN SECS.
250
500
C
B&E
B
A
E
F
C&F
D
CUBIC PLANT
CUBIC DIFFERENCE EQUATION MODEL
ERROR CORRECTING ALGORITHM
A B C D E F
$\underline{W}_1$ 0 0 0 W_* W_* W_*
σ 0.0 0.1 0.1 0.0 0.1 0.1
a 1.0 1.0 0.5 1.0 1.0 0.5

FIGURE 4.14

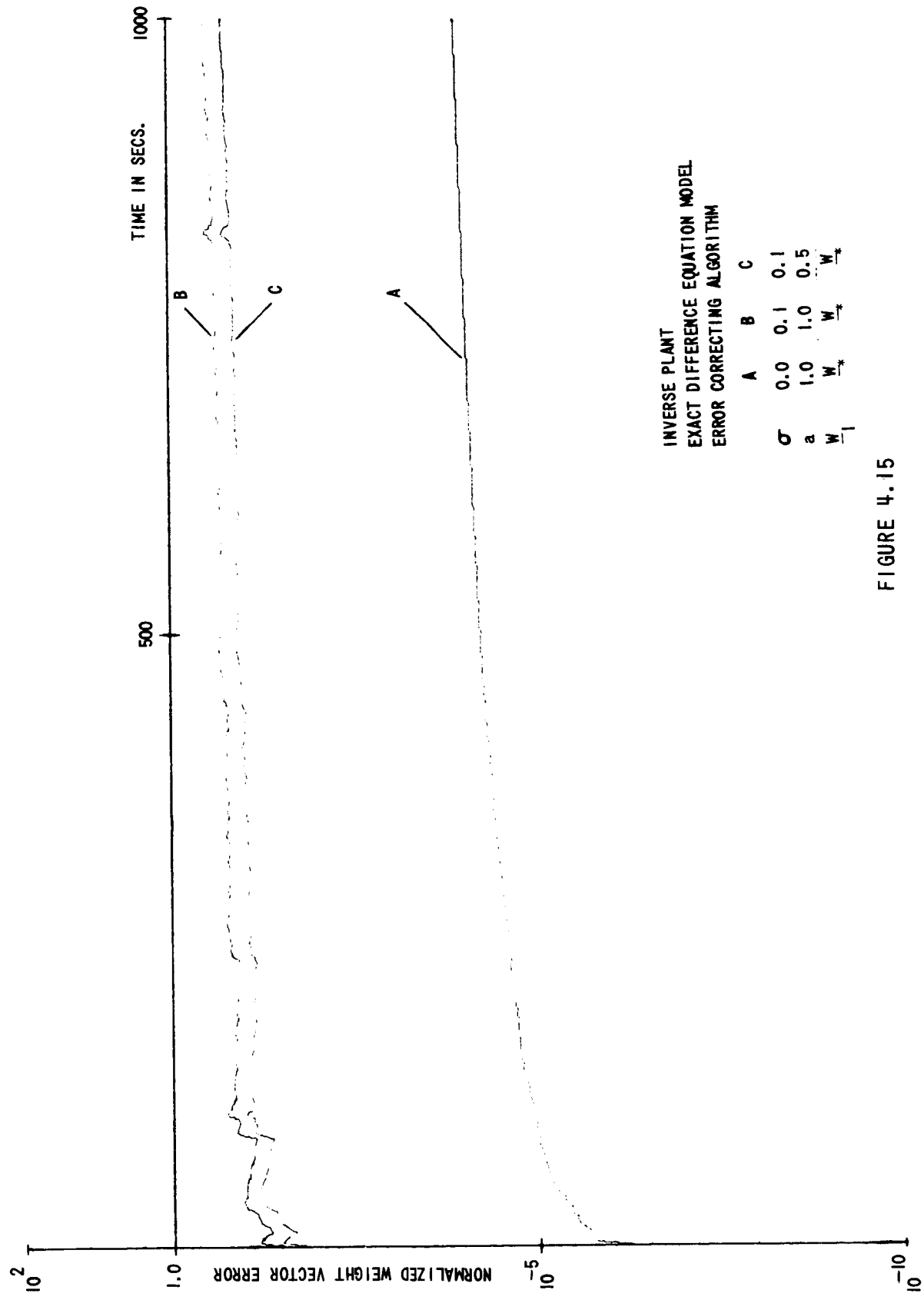

TIME IN SECS.
1000
500
1.0
10^{2}
10^{-5}
10^{-10}
NORMALIZED WEIGHT VECTOR ERROR
A
B
C
INVERSE PLANT
EXACT DIFFERENCE EQUATION MODEL
ERROR CORRECTING ALGORITHM
A B C
σ 0.0 0.1 0.1
a 1.0 1.0 0.5
$\underline{W}_1$ $\underline{W}^*$ $\underline{W}^*$ $\underline{W}^*$

FIGURE 4.15

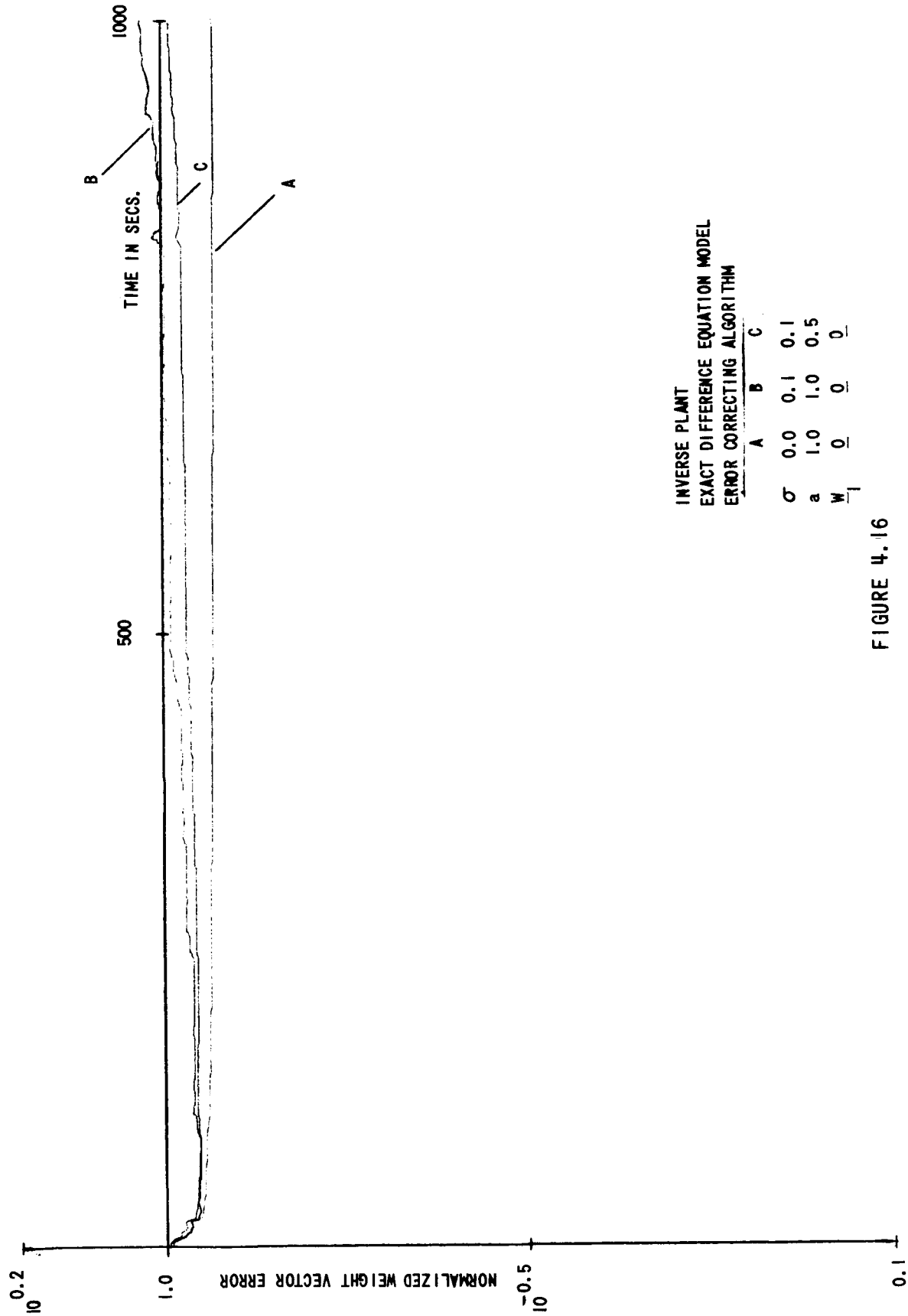

INVERSE PLANT
EXACT DIFFERENCE EQUATION MODEL
ERROR CORRECTING ALGORITHM
A B C
σ 0.0 0.1 0.1
a 1.0 1.0 0.5
W1 0 0 0
A
B
C
1000
500
TIME IN SECS.
NORMALIZED WEIGHT VECTOR ERROR
10^0.2
1.0
10^-0.5
0.1

FIGURE 4.16

For the second case, the plant was identified for various ranges of operation about an operating point, x_o. This was achieved by multiplying the usual input noise sequence by a constant and adding another constant. Based on the nonlinear function of the inverse system, four operating points were chosen. They are: $x_o = 0.0$ and $x_o = 1.5$ (the maximum and minimum points of the slope of the nonlinear function); and $x_o = 0.75$ and $x_o = 6.75$ (the unity slope points of the nonlinear function). The second derivative of the nonlinear function is nearly maximal at $x_o = .75$ and minimal at $x_o = 6.75$. The normalized weight vector error curves in Figures (4.17) through (4.20) show that perturbations of approximately .01 yield the best operating range for this case. If the perturbations about the operating point are too small, the projection algorithm will not be able to form a full set of linearly independent $\underline{F}_i$'s. However, if the operating range is too large, the modeling error will keep the learning model from accurately approximating the linearized plant. The results are poorest at $x_o = .75$ where the characteristics of the linearized plant change the most over the operating range. Thus, the best operating range depends on the operating point and the degree of nonlinearity at the operating point. The convergence rates are about half that for the linear second order system examples. Therefore, the model tracking identification has linearized the system.

4.3 Results for Non-Stationary Optimum Models

Two non-stationary situations were considered. In the first case, the model tracking system with the second order difference equation model and the projection algorithm was used. Based on the linear model, the plant

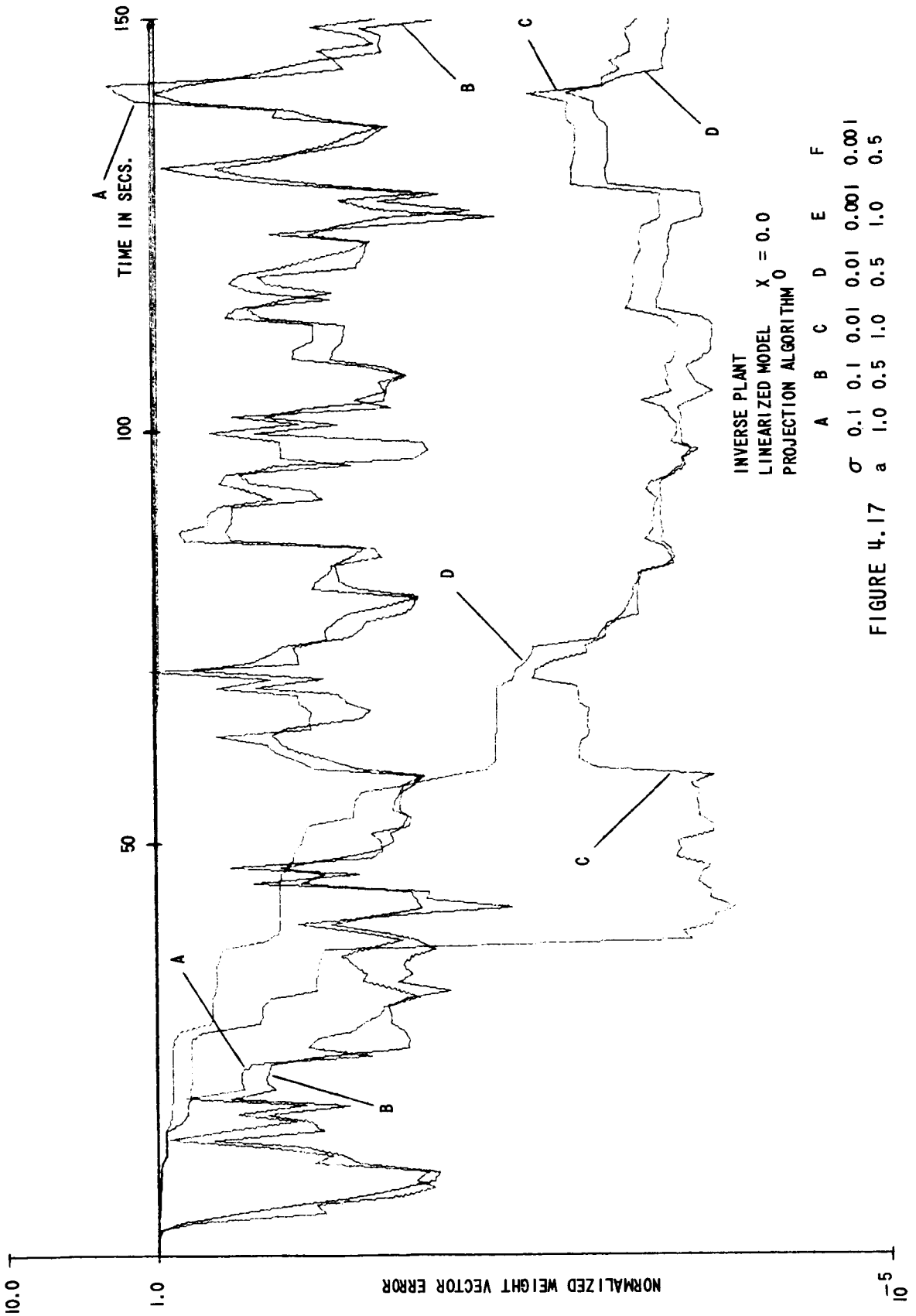

NORMALIZED WEIGHT VECTOR ERROR
10.0
1.0
10^{-5}
TIME IN SECS.
50
100
150
A
B
C
D
INVERSE PLANT
LINEARIZED MODEL $X_0 = 0.0$
PROJECTION ALGORITHM
A B C D E F
σ 0.1 0.1 0.01 0.01 0.001 0.001
a 1.0 0.5 1.0 0.5 1.0 0.5

FIGURE 4.17

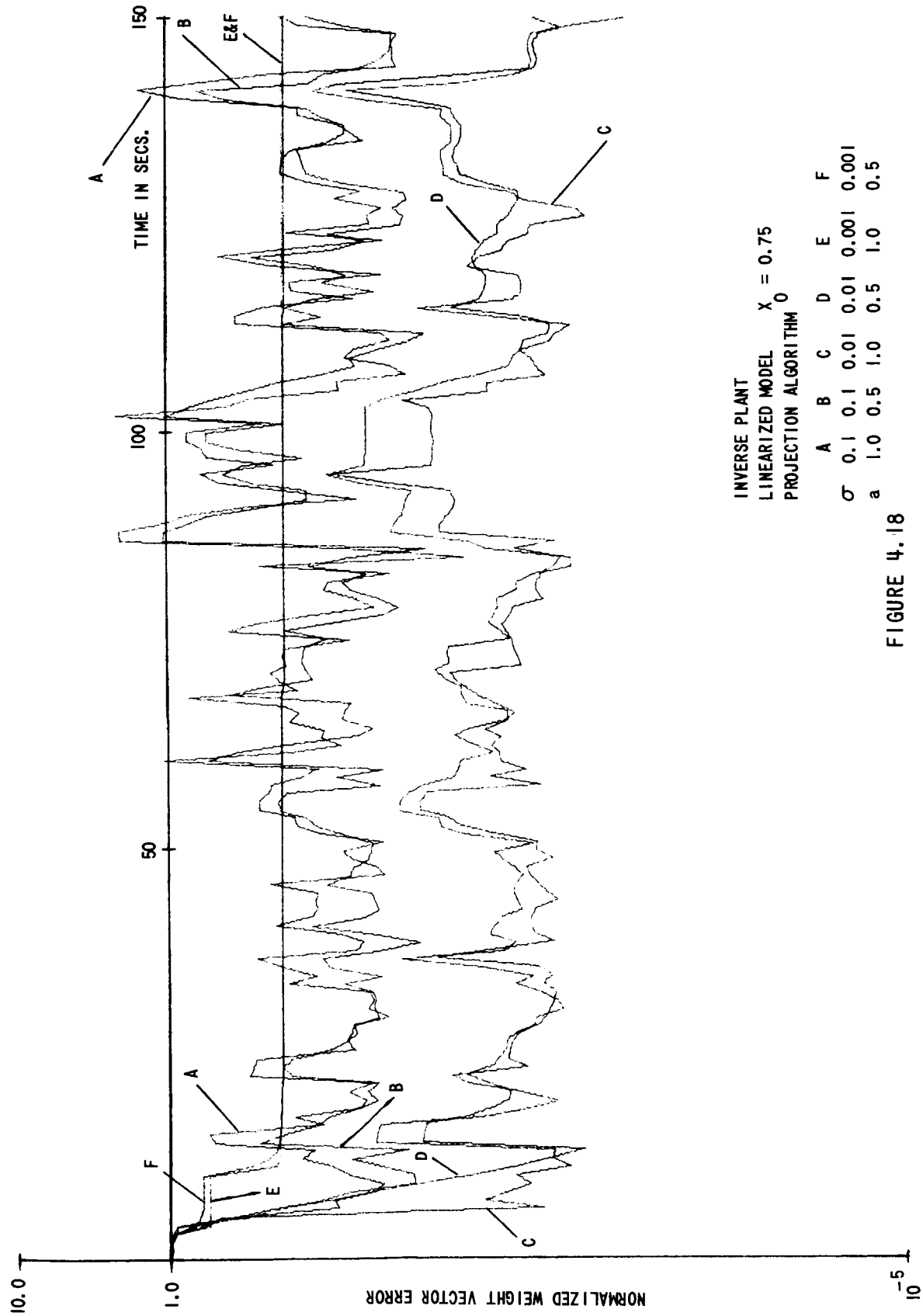

NORMALIZED WEIGHT VECTOR ERROR
10.0
1.0
10^{-5}
TIME IN SECS.
50
100
150
A
B
E&F
C
D
F
E
INVERSE PLANT
LINEARIZED MODEL $X_0 = 0.75$
PROJECTION ALGORITHM
A B C D E F
σ 0.1 0.1 0.01 0.01 0.001 0.001
a 1.0 0.5 1.0 0.5 1.0 0.5

FIGURE 4.18

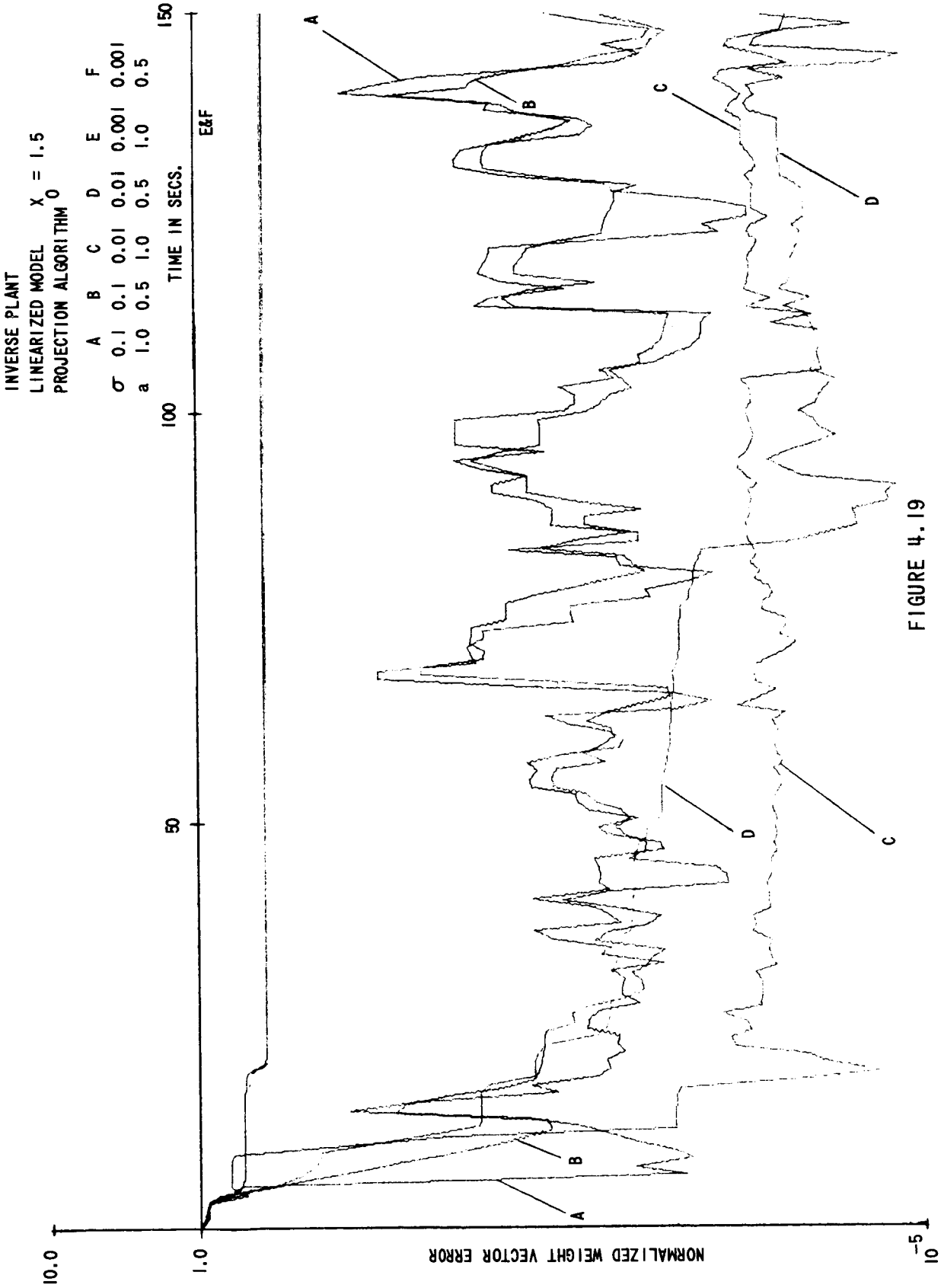

INVERSE PLANT
LINEARIZED MODEL X_0 = 1.5
PROJECTION ALGORITHM
A B C D E F
σ 0.1 0.1 0.01 0.01 0.001 0.001
a 1.0 0.5 1.0 0.5 1.0 0.5
TIME IN SECS.
E&F
NORMALIZED WEIGHT VECTOR ERROR
10.0
1.0
10^-5
50
100
150
A
B
C
D

FIGURE 4.19

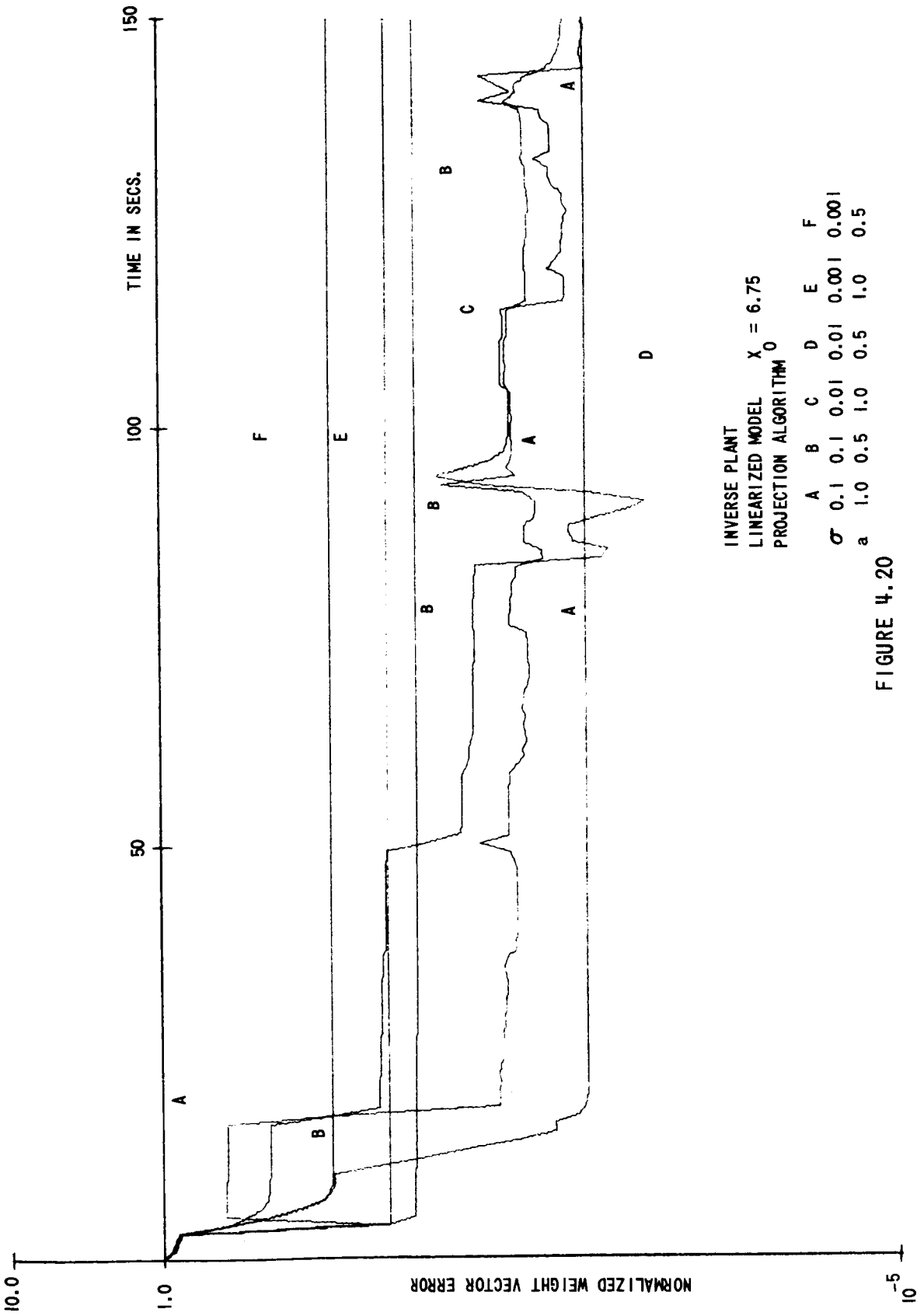

NORMALIZED WEIGHT VECTOR ERROR
10.0
1.0
10^{-5}
TIME IN SECS.
150
100
50
A
B
C
D
E
F
INVERSE PLANT
LINEARIZED MODEL $X_0 = 6.75$
PROJECTION ALGORITHM
A B C D E F
σ 0.1 0.1 0.01 0.01 0.001 0.001
a 1.0 0.5 1.0 0.5 1.0 0.5

FIGURE 4.20

was made to follow a few trajectories. For the second case, the bending mode of the booster was identified given a difference equation model for the rigid dynamics.

Three trajectories were used in the test: a series of steps, two truncated ramps, and a noisy sinusoid. The noisy sinusoid was a sinusoid of magnitude five added to the standard noise sequence with a σ of 0.1. The resulting normalized weight vector error curves are in Figure (4.21). Figures (4.22), (4.23), and (4.24) are plots of the actual and desired plant outputs. When the learning model is a poor representation of the system, the control will cause considerable deviation from the desired trajectory. This behavior will normally be oscillatory thereby providing sufficient information to identify the system. This is especially evident in the noisy sinusoid case. Thus, it should be possible to design a dynamic linearization identifier which can be used in the control of nonlinear plants over slowly varying trajectories.

The bending mode of the booster was identified using the model tracking system of Figure (4.25). Note that measurement errors are introduced by the use of a difference equation model for the rigid dynamics. The output of this model does not always agree with the actual rigid body states. Thus, when they are subtracted from the gyro measurements, an error is introduced. The learning model was initialized to the nominal bending model for 4.0 seconds after liftoff. Identification runs were made both for nominal bending and for bending with the natural frequency reduced to eighty per cent of nominal. The normalized weight vector error curves obtained using the error correcting algorithm are in Figures (4.26) and (4.27).

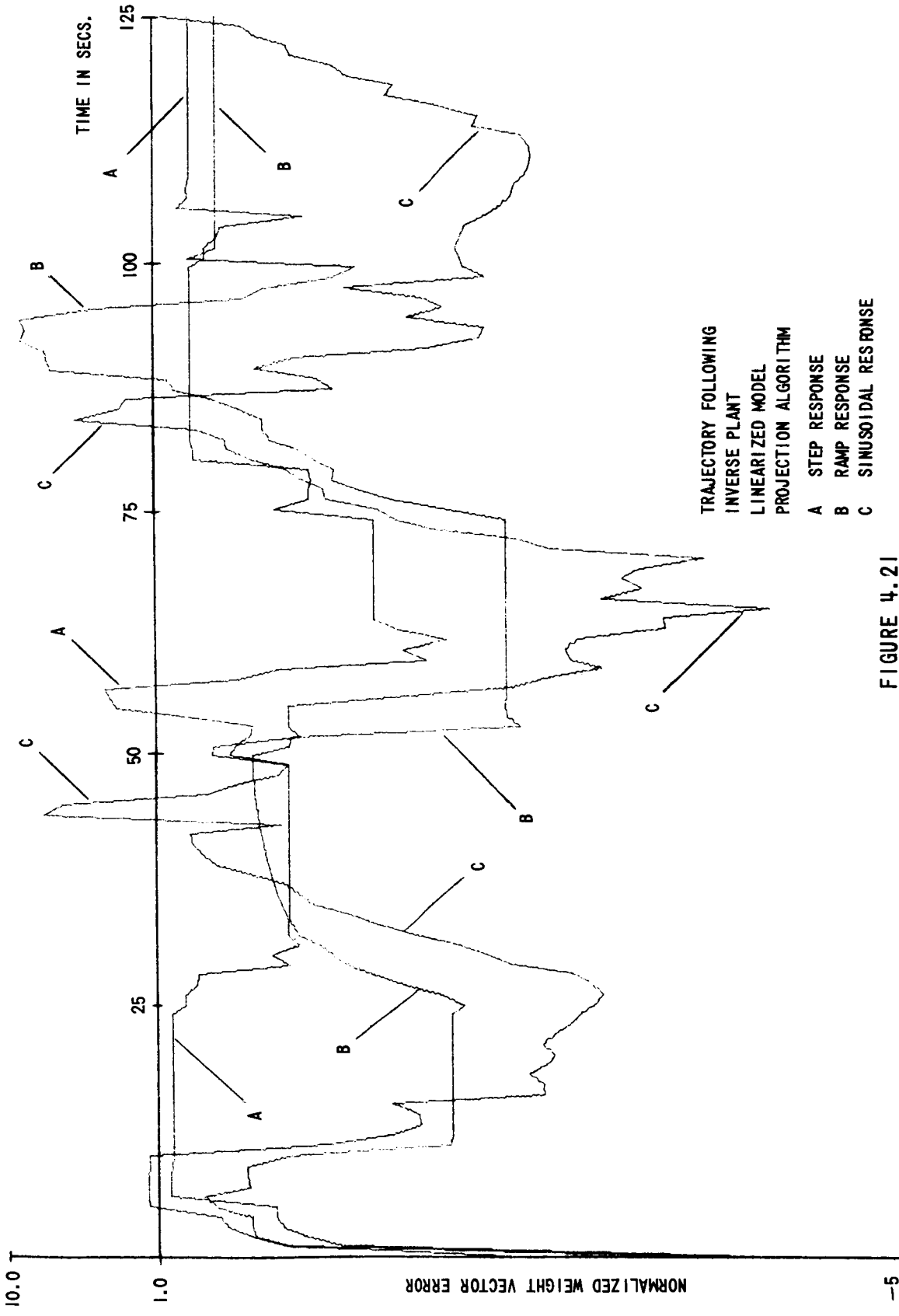

TIME IN SECS.
125
100
75
50
25
10.0
1.0
10^{-5}
NORMALIZED WEIGHT VECTOR ERROR
A
B
C
TRAJECTORY FOLLOWING
INVERSE PLANT
LINEARIZED MODEL
PROJECTION ALGORITHM
A STEP RESPONSE
B RAMP RESPONSE
C SINUSOIDAL RESPONSE

FIGURE 4.21

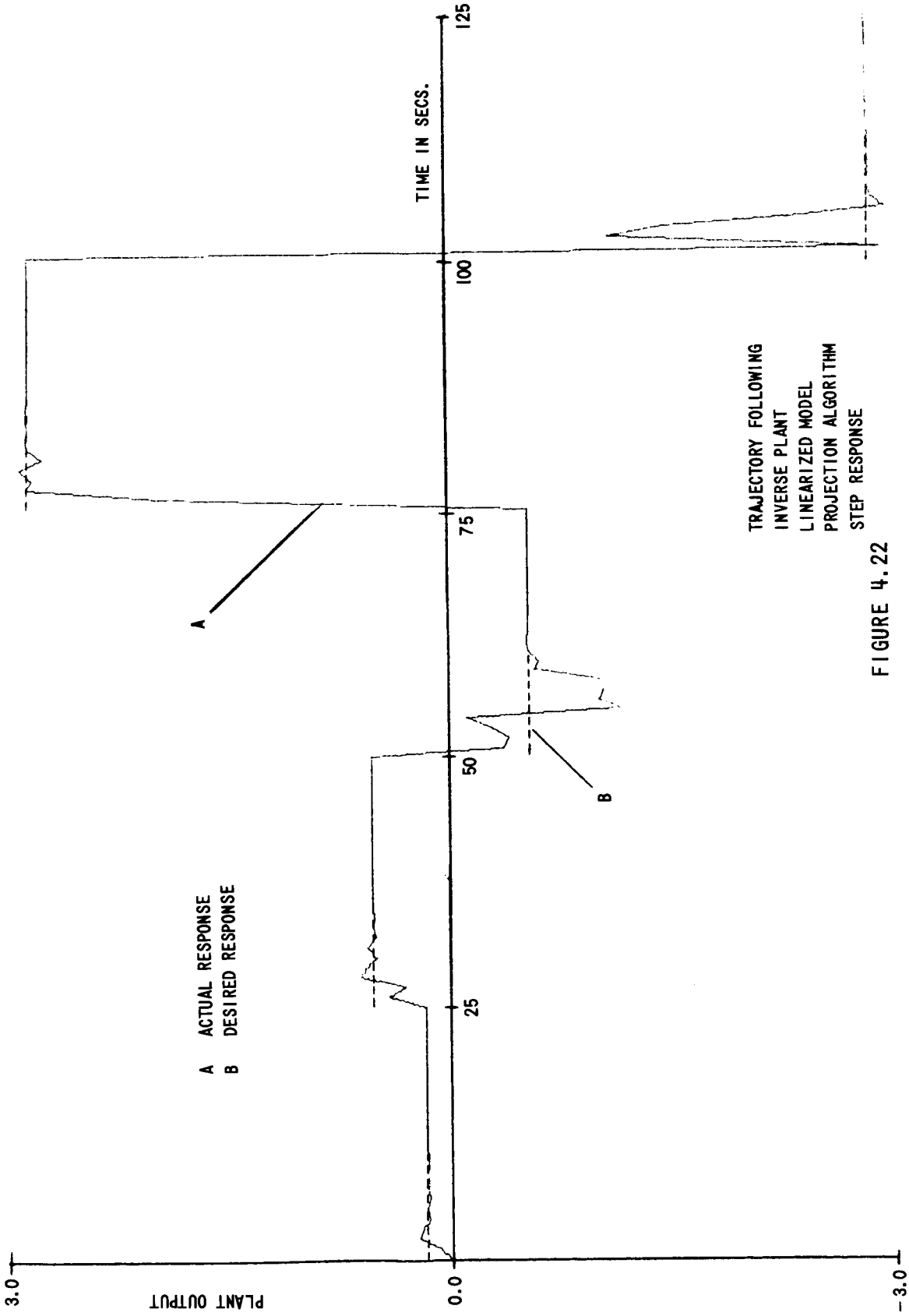

TRAJECTORY FOLLOWING
INVERSE PLANT
LINEARIZED MODEL
PROJECTION ALGORITHM
STEP RESPONSE
A ACTUAL RESPONSE
B DESIRED RESPONSE
PLANT OUTPUT
TIME IN SECS.
3.0
0.0
-3.0
25
50
75
100
125
A
B

FIGURE 4.22

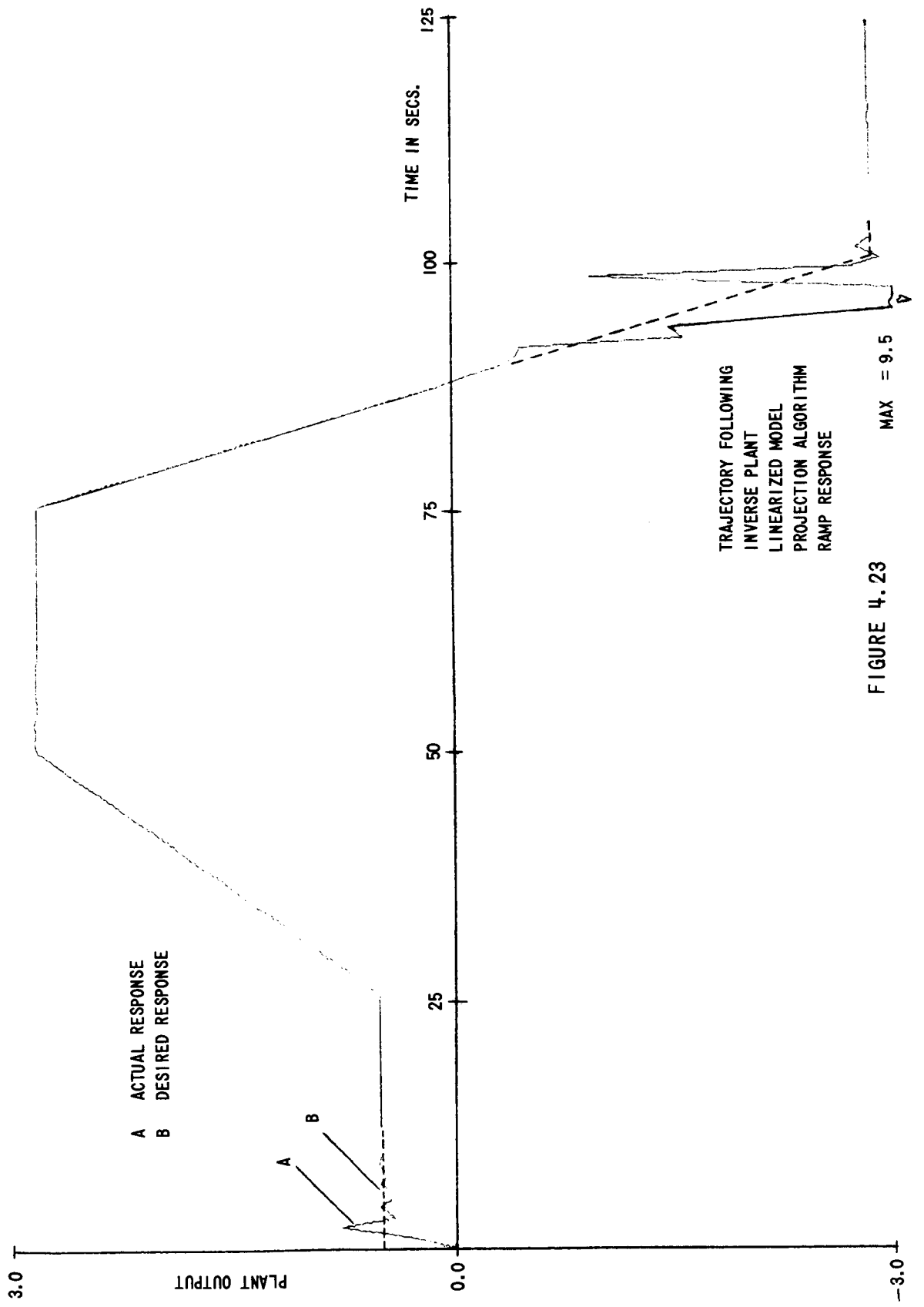

TRAJECTORY FOLLOWING
INVERSE PLANT
LINEARIZED MODEL
PROJECTION ALGORITHM
RAMP RESPONSE
MAX = 9.5
A ACTUAL RESPONSE
B DESIRED RESPONSE
A
B
PLANT OUTPUT
TIME IN SECS.
3.0
0.0
-3.0
25
50
75
100
125

FIGURE 4.23

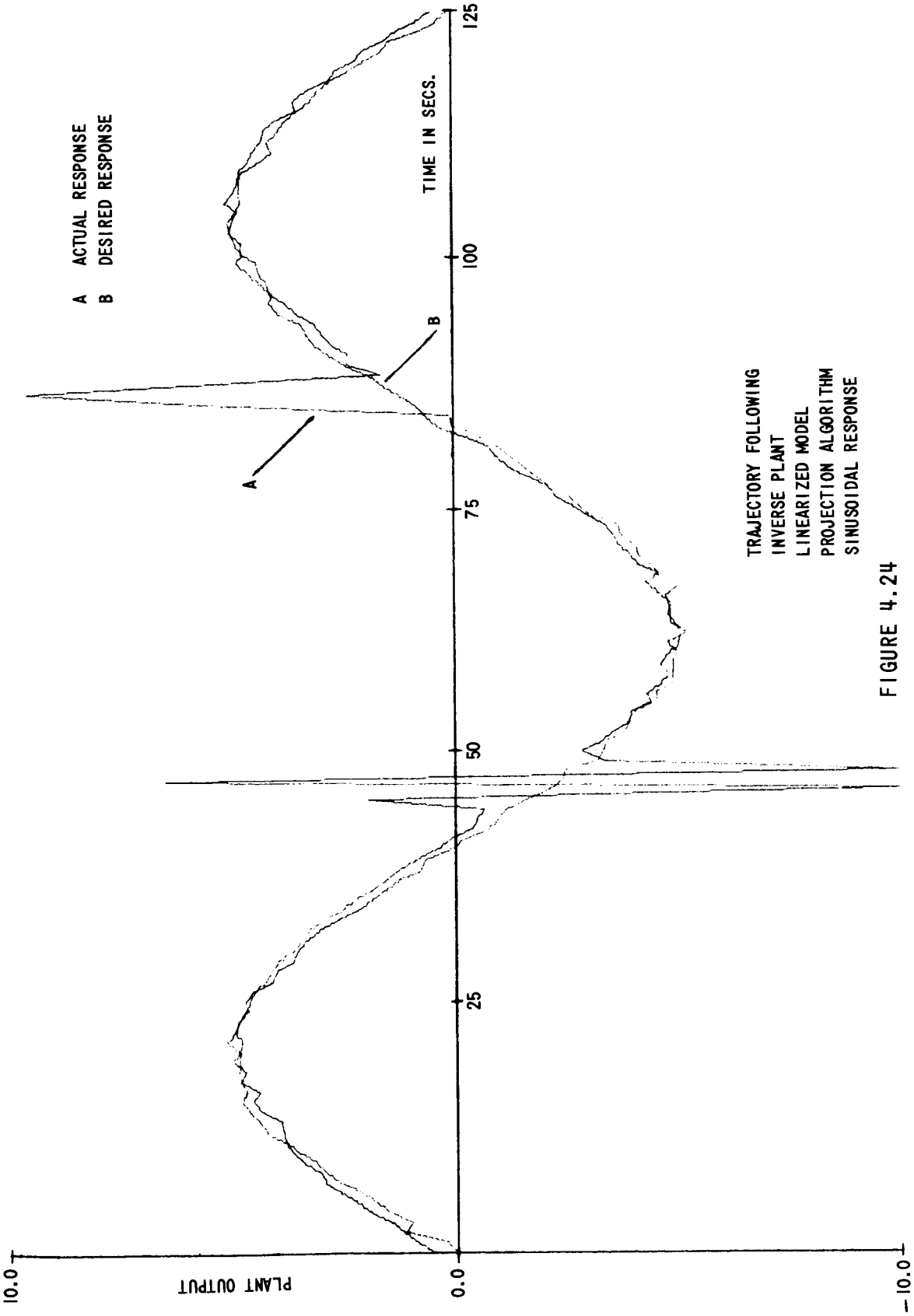
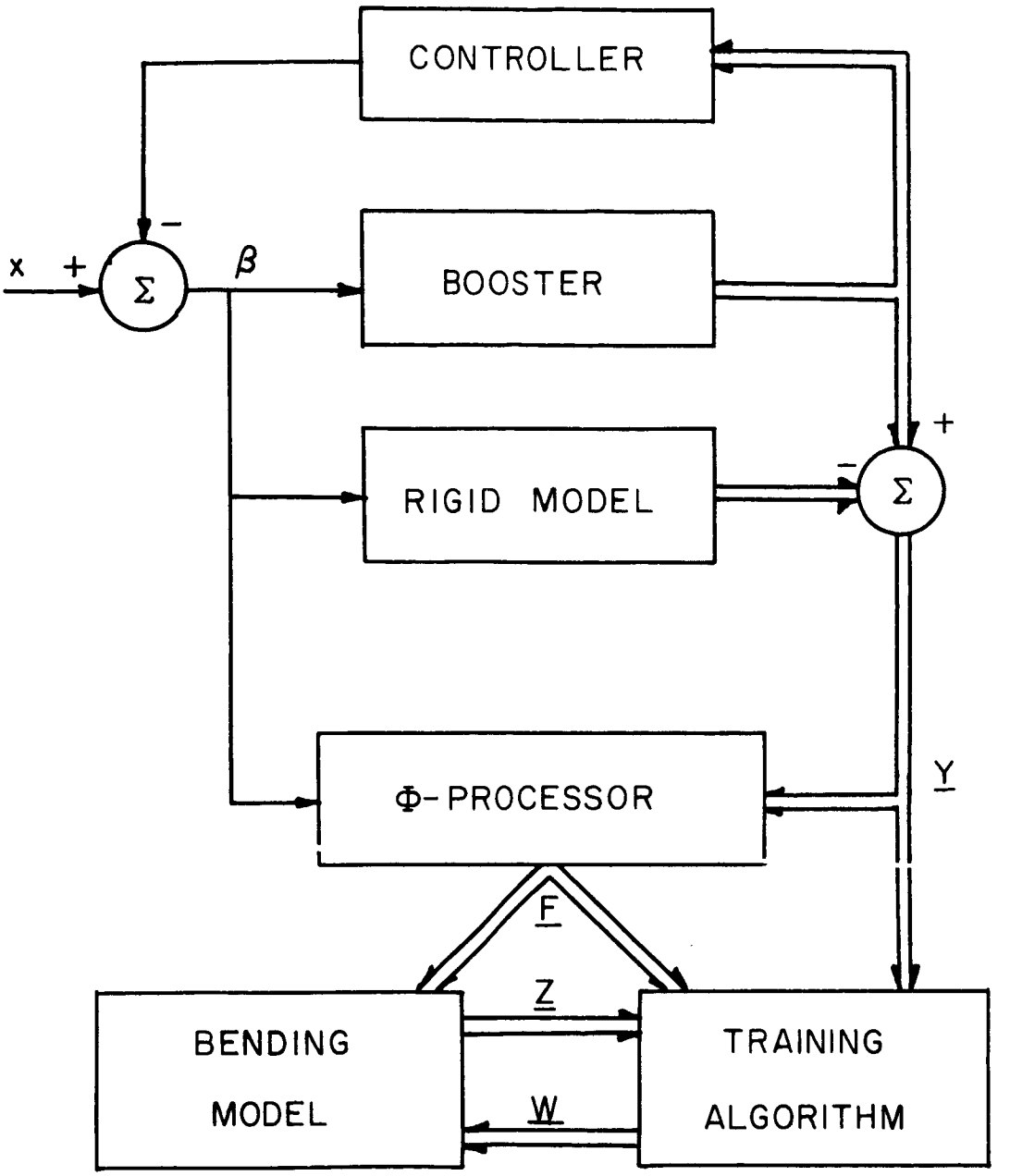

A ACTUAL RESPONSE
B DESIRED RESPONSE
TIME IN SECS.
PLANT OUTPUT
10.0
0.0
-10.0
25
50
75
100
125
A
B
TRAJECTORY FOLLOWING
INVERSE PLANT
LINEARIZED MODEL
PROJECTION ALGORITHM
SINUSOIDAL RESPONSE

FIGURE 4.24



BENDING IDENTIFIER

FIGURE 4.25

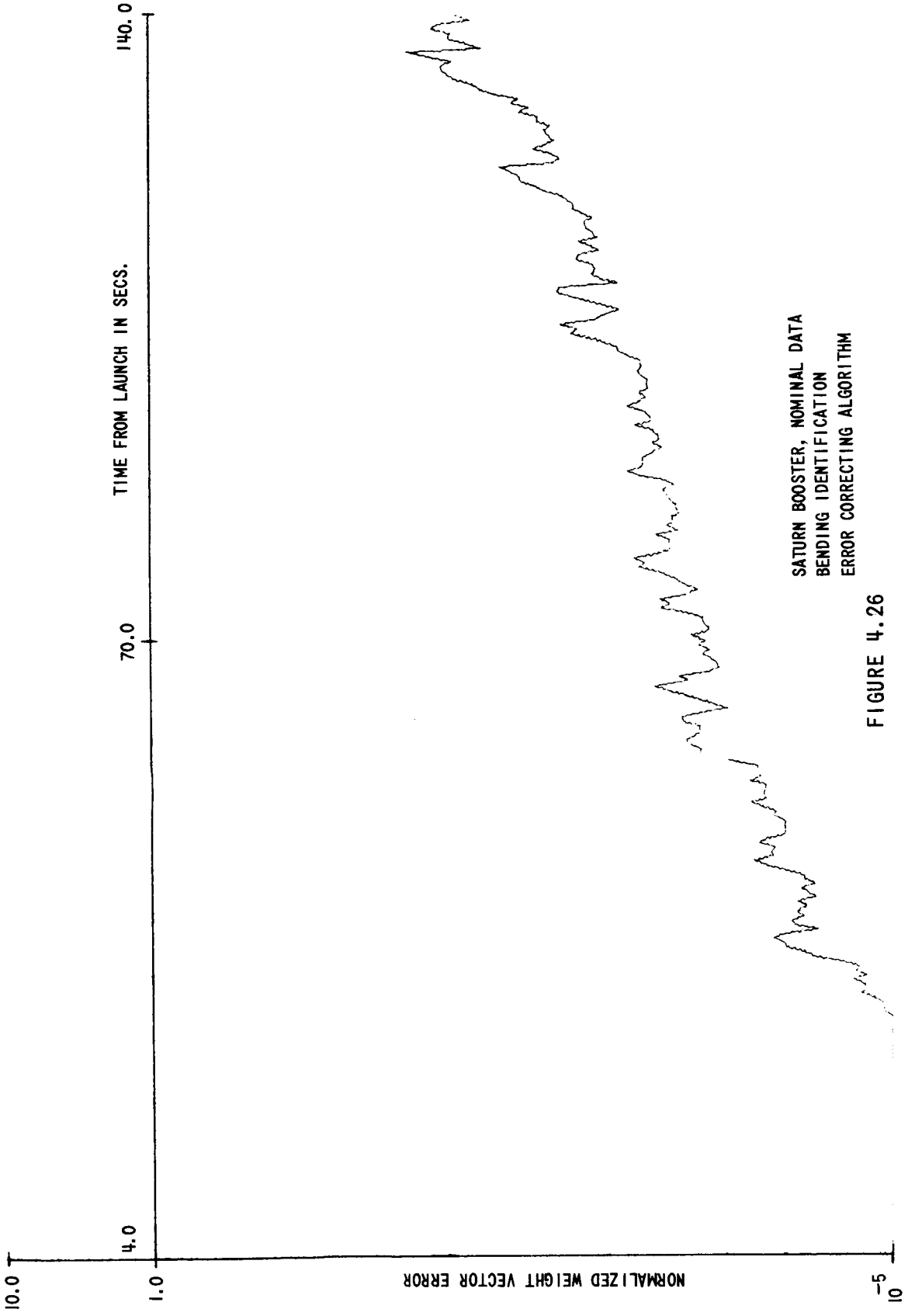

10.0
1.0
10^-5
NORMALIZED WEIGHT VECTOR ERROR
4.0
70.0
140.0
TIME FROM LAUNCH IN SECS.
SATURN BOOSTER, NOMINAL DATA
BENDING IDENTIFICATION
ERROR CORRECTING ALGORITHM

FIGURE 4.26

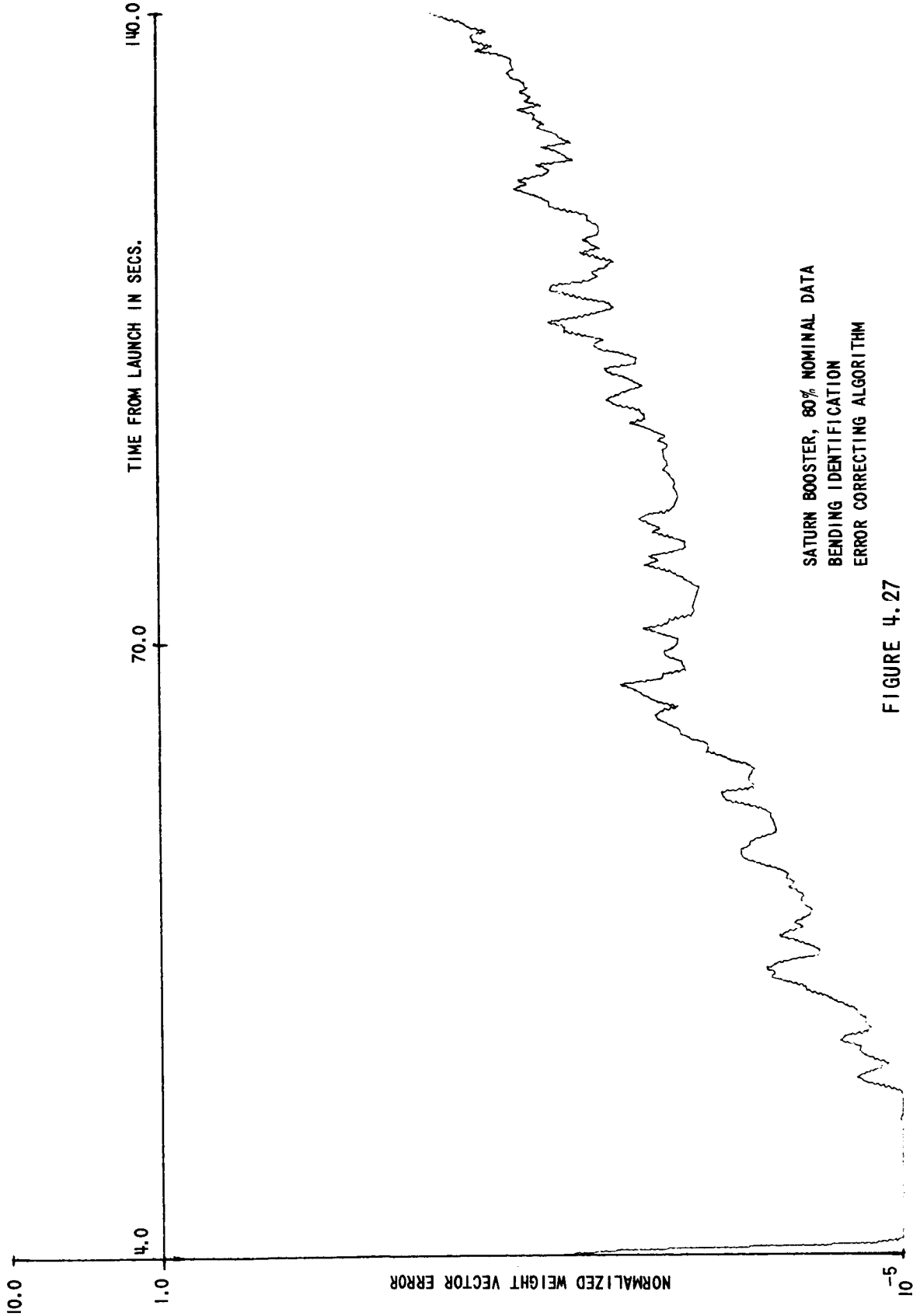

10.0
1.0
10^{-5}
NORMALIZED WEIGHT VECTOR ERROR
4.0
70.0
140.0
TIME FROM LAUNCH IN SECS.
SATURN BOOSTER, 80% NOMINAL DATA
BENDING IDENTIFICATION
ERROR CORRECTING ALGORITHM

FIGURE 4.27

The curves rise because the error correcting algorithm cannot keep up with the rate of change of the booster. The results for the projection algorithm are in Figures (4.28) and (4.29). The peaks are due to the sensitivity of the projection algorithm. The controlled booster had very smooth trajectories making identification very difficult. Due to the large amount of computer time needed for this type of run, no attempt was made to maximize the performance on this example. However, it can be seen from the methods used that a compromise of these methods should provide the desired identification of the bending.

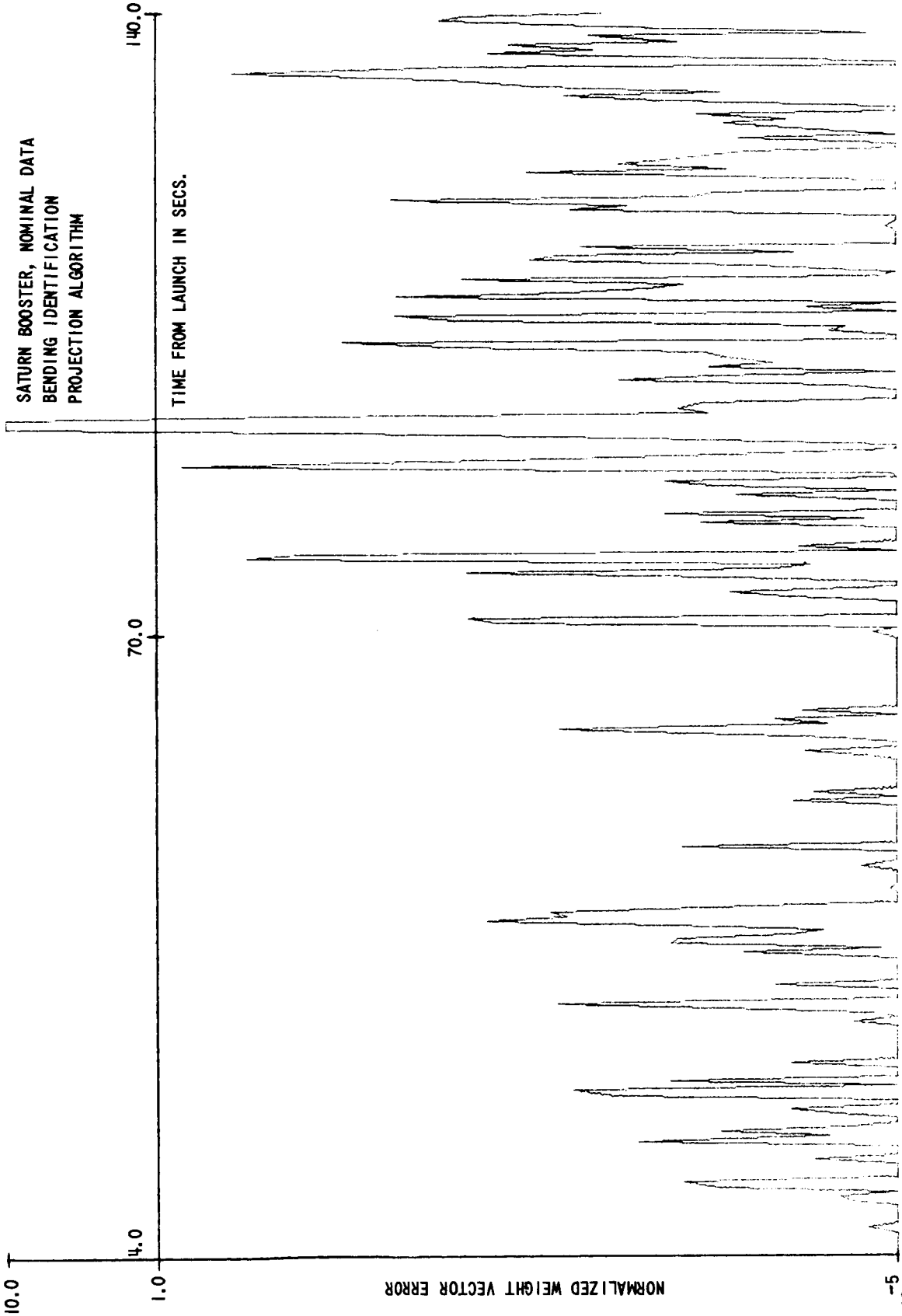

SATURN BOOSTER, NOMINAL DATA
BENDING IDENTIFICATION
PROJECTION ALGORITHM
TIME FROM LAUNCH IN SECS.
140.0
70.0
4.0
10.0
1.0
NORMALIZED WEIGHT VECTOR ERROR
10^{-5}

FIGURE 4.28

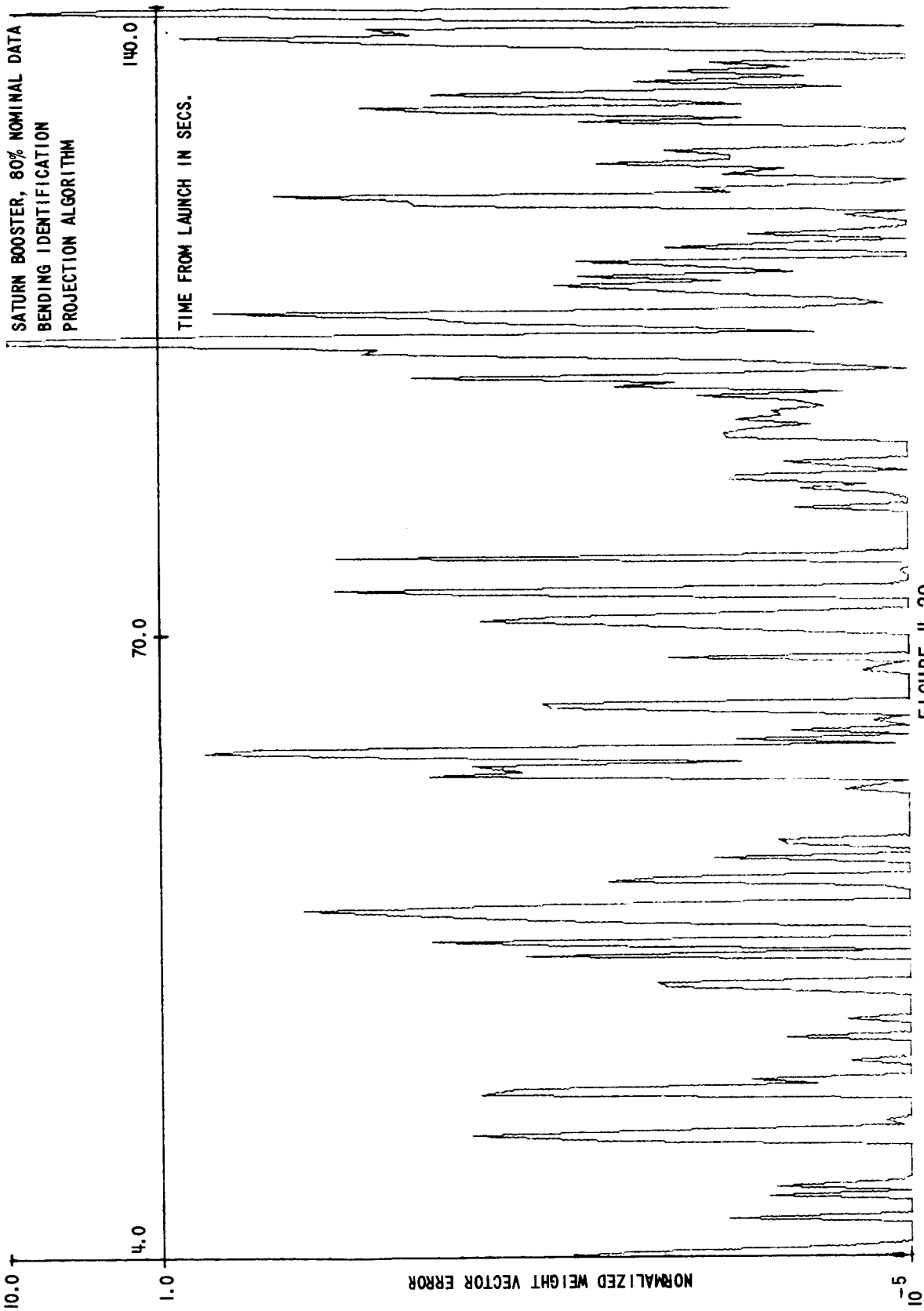

SATURN BOOSTER, 80% NOMINAL DATA
BENDING IDENTIFICATION
PROJECTION ALGORITHM
TIME FROM LAUNCH IN SECS.
4.0
70.0
140.0
10.0
1.0
10^{-5}
NORMALIZED WEIGHT VECTOR ERROR

FIGURE 4.29

CHAPTER V

CONCLUSION

5.1 Conclusions

By observing that the output of a system can be represented by a functional, a method of system identification has been developed using the pattern recognition technique of ϕ-learning machines. The functional can be approximated by a transformation from the operating history of the system to the output of the system. Viewing this transformation as a hypersurface in a N-dimensional space facilitates the use of the pattern recognition technique to identify the hypersurface. The approach yields a very general method of system identification.

The designer must choose a model for the physical system based on his a priori knowledge of the physical system and the situation in which the identification is to be used. Any model that can be expressed in the general form of Section (2.5) may be used with the training procedures of Chapter III. This general form includes models based on delay line and difference equation representations for the physical system. These models can be linear or nonlinear and may include inverse terms.

The ability of the learning technique to identify the physical system from its normal operating record was verified by the use of the model tracking system in Chapter IV. Table III is a summary of the experimental results discussed in Chapter IV.

Both analytical and experimental results imply that the learning technique has certain characteristics. First, the use of a simple model for a complex plant at an operating point is feasible. Second, by the use

of the convergence factor, speed of identification can be sacrificed for a greater accuracy of identification. Finally, the magnitude of the identification error $\|\underline{E}_\infty\|$ is proportional to the variance of the additive noise.

The trade-off between computational complexity and speed of identification comes down to choosing a method of approximating a solution to Equation (3.54). Two methods tried were the error correcting algorithm and the projection algorithm. The projection algorithm was developed as an extension of the geometric concept of the error correcting algorithm. Among the possibilities of improving the identification technique would be the use of techniques which computed the weight vector increment based on a large set of data points in order to lessen the effect of additive noise. Another possibility would be the scaling of the $f_i(\underline{X})$ to improve the searching for the weight vector.

TABLE III

SUMMARY OF EXPERIMENTAL RESULTS

Configuration of Model Tracking System	Results
(a) Linear, Second Order Underdamped Plant Delay Line Model Error correcting Algorithm White Noise Input	(1) Convergence was exponential (2) Convergence rate was independent of σ (3) Error asymptotes were proportional to σ^2 and to a^2 (4) ρ had very little affect on the error asymptotes (5) The rms error between plant and learning model approached σ as a decreases (6) The rms error between optimum model and learning model became less than σ as a decreases
(b) Linear, Second Order Underdamped Plant Difference Equation Model Error Correcting Algorithm White Noise Input	(1) Convergence was exponential (2) Convergence rate was independent of σ (3) Error asymptotes were proportional to σ^2 and to a^2 (4) ρ had very little affect on the error asymptotes (5) The rms error between plant and learning model approached σ as a decreases (6) The rms error between optimum model and learning model became less than σ as a decreases (7) Convergence rate was approximately twice of that for (a)

TABLE III (Cont'd)

Configuration of Model Tracking System	Results
(c) Linear, Second Order Underdamped Plant Difference Equation Model Projection Algorithm White Noise Input	(1) Convergence was exponential (2) Convergence rate was independent of σ (3) Error asymptotes were proportional to σ^2 and to a^2 (4) ρ had very little affect on the error asymptotes (5) Convergence rate is approximately thirteen times that for (b) (6) Projection algorithm is very sensitive to noise
(d) Linear, Second Order Underdamped Plant Difference Equation Model Error Correcting Algorithm Trajectory Following Input	(1) Convergence was exponential (2) Error asymptotes are proportional to a^2 (3) The rms error between plant and learning model approached σ as a decreases (4) Trajectory following control did not change characteristics of identification
(e) Cubic Plant Second Order Cubic Difference Equation Model Error Correcting Algorithm White Noise Input	(1) Error asymptotes were proportional to a^2 (2) Convergence rate is about half that of (a) (3) The range of input seemed insufficient to keep up a constant rate of convergence (4) Convergence was improved by decreasing a
(f) Inverse Plant Exact Difference Equation Model Error Correcting Algorithm White Noise Input	(1) Convergence is hampered by the limited range of the input (2) Error asymptotes were proportional to a^2 (3) Convergence was improved by decreasing a

TABLE III (Cont'd)

Configuration of Model Tracking System	Results
(g) Inverse Plant Linearized Difference Equation Model Projection Algorithm White Noise Input	(1) The Learning Model converged toward the linearized plant (2) Decreasing a improved the error asymptotes (3) Convergence rates were about half those of (c) (4) The projection algorithm did not converge quickly when the perturbations were too small
(h) Inverse Plant Linearized Difference Equation Model Projection Algorithm Trajectory Following Input	(1) The control and identification interacted in a benefical manner (2) The simplified control law gave reasonable responses when the normalized weight vector error was less than one (3) Identification became poor when the operating point was changed rapidly over extended periods of time
(i) Booster with First Bending Mode Model as Given in Figure (4.25) Error Correcting Algorithm White Noise Input	(1) Control of noise was good (2) Convergence rate was about half that was needed (3) Identification was obtained from well controlled trajectories (4) Normalized weight vector error curves were similar for both nominal and off nominal bending frequency
(j) Booster with First Bending Mode Model as Given in Figure (4.25) Projection Algorithm White Noise Input	(1) Identification was sensitive to noise (2) Convergence rate was much greater than needed (3) Identification was obtained from well controlled trajectories (4) Normalized weight vector error curves were similar for both nominal and off nominal bending frequency

LITERATURE CITED

1. Barrett, J. F., The Use of Functionals in the Analysis of Nonlinear Physical Systems, Statistical Advisory Unit, Report No. 1/57, Ministry of Supply, Great Britain, 1957.

2. Brilliant, M. B., Theory of the Analysis of Nonlinear Systems, Technical Report 345, Research Laboratory of Electronics, MIT, March 1958.

3. Bush, A. M., Some Techniques for the Synthesis of Nonlinear Systems, Technical Report 441, Research Laboratory of Electronics, MIT, March 1966.

4. Cameron, R. H., and Martin, W. T., The Orthogonal Development of Nonlinear Functionals in Series of Fourier-Hermite Functionals, Annals of Mathematics, 48, No. 2, pp. 385-389, April 1947.

5. Cuenod, M. and Sage, A. P., Comparison of Some Methods Used for Process Identification, Survey Paper, IFAC Symposium on Identification in Automatic Control Systems, Prague, June 1967.

6. Eykhoff, P., Process Parameter and State Estimation, Survey Paper, IFAC Symposium on Identification in Automatic Control Systems, Prague, June 1967.

7. Jenkins, K. W. and Roy, R. J., A Design Procedure for Discrete Adaptive Control Systems, Proceedings of the 7th Joint Automatic Control Conference, University of Washington, Seattle, Washington, August 17-19, 1966, pp. 624-633.

8. Kalman, R. E., Design of a Self-Optimizing Control System, Trans. ASME, February 1958, pp. 468-678

9. Kerr, R. B. and Surber, W. H., Precision of Impulse-Response Identification Based on Short, Normal Operating Records, IRE Transactions on Automatic Control, May 1961, pp. 173-182.

10. Kwatny, H. G., and Shen, D. W. C., Identification of Nonlinear Systems Using a Method of Stochastic Approximation, Proceedings of the 8th Joint Automatic Control Conference, University of Pennsylvania, Philadelphia, Pennsylvania, June 1967, pp. 814-826.

11. Levin, M. J., Optimum Estimation of Impulse Response in the Presence of Noise, IRE Transactions on Circuit Theory, March 1960, pp. 50-56.

12. Nagumo, J., and Noda, A., A Learning Method for System Identification, Proceedings of the 1966 National Electronics Conference, Vol. 22, pp. 584-589.

13. Nilsson, N. J., Learning Machines, McGraw-Hill Book Company, N. Y., 1965.

14. Ralston, A., and Wilf, H., Mathematical Methods for Digital Computers, "Multiple Regression Analysis," by M. A. Efroymson, John Wiley and Sons, Inc., 1960, pp. 191-203.

15. Roy, R. J. and DeRusso, P. M., A Digital Orthogonal Model for Nonlinear Processes, IRE Transactions on A. C., May 1962, pp. 93-101.

16. Roy, R. and Miller, R. W., Nonlinear Process Identification Using Decision Theory, IEEE Transactions on Automatic Control, October, 1964, Vol. AC-9, No. 4, pp. 538-540.

17. Roy, R. and Schley, C., Process Identification Using a Mode Learning Machine, Proceedings of the 7th Joint Automatic Control Conference, University of Washington, Seattle, Washington, August 17-19, 1966, pp. 639-648.

18. Shen, D. W. C., Artificial Intelligence in a Hierachy of Nonlinear Systems, IEEE Special Publication S-142 Artificial Intelligence, January 1963, pp. 18-30.

19. Smets, H. B., Analysis and Synthesis of Nonlinear Systems, IRE Transactions Professional Group on Circuit Theory, CT-7, No. 4, pp. 459-469, December 1960.

20. VanTrees, H. L., Synthesis of Optimum Nonlinear Control Systems, The M.I.T. Press, Cambridge, Mass., 1962.

21. Volterra, V., Theory of Functionals and of Integral and Integro-Differential Equations, Dover Publications, New York, 1959.

22. Wiener, N., Nonlinear Problems in Random Theory, The Technology Press of M.I.T., Cambridge, Mass., and John Wiley and Sons, New York, 1949.

23. Zaborsky, J., and Humphrey, W., Control Without Model or Plant Identification, Proceedings of the 5th Joint Automatic Control Conference, Stanford University, Stanford, California, 1964, pp. 361-366.

APPENDIX A

Computing $(\Psi^T \Psi)^{-1}$

The basis is formed by adding a column and possibly removing one or more columns from the previous basis. Therefore, it is possible to save a considerable amount of computational effort by using the value of the old $(\Psi^T \Psi)^{-1}$ in finding the value of the new $(\Psi^T \Psi)^{-1}$. This can be done by examining the inverse of a general partitioned matrix. Let M be a general partitioned matrix.

$$M = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \tag{A.1}$$

Then if M^{-1} is partitioned in the same manner, the result is

$$M^{-1} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \tag{A.2}$$

where

$$\begin{aligned} A &= (a - bd^{-1} c)^{-1} \\ B &= - Abd^{-1} \\ C &= - d^{-1} cA \\ D &= d^{-1} - d^{-1} cB \end{aligned} \tag{A.3}$$

Consider the case of appending a column $\underline{F}$ to Ψ_i.

$$\Psi_{i+1} = (\underline{F}, \Psi_i) \tag{A.4}$$

$$(\Psi_{i+1}{}^T \Psi_{i+1})^{-1} = \begin{bmatrix} \underline{F}^T\underline{F} & \underline{F}^T \Psi_i \\ \Psi_i{}^T\underline{F} & \Psi_i{}^T \Psi_i \end{bmatrix}^{-1} \qquad (A.5)$$

$$= \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

$$A = \frac{1}{\underline{F}^T\underline{F} - \underline{F}^T \Psi_i (\Psi_i{}^T \Psi_i)^{-1} \Psi_i{}^T\underline{F}}$$

$$B = - A\underline{F}^T \Psi_i (\Psi_i{}^T \Psi_i)^{-1}$$

$$C = - (\Psi_i{}^T \Psi_i)^{-1} \Psi_i{}^T \underline{F}A = B^T \qquad (A.6)$$

$$D = (\Psi_i{}^T \Psi_i)^{-1} - (\Psi_i{}^T \Psi_i)^{-1} \Psi_i{}^T \underline{F}B$$

$$= (\Psi_i{}^T \Psi_i)^{-1} + B^T BA^{-1}$$

This can be easily computed on a digital computer.

Consider the case of removing a column $\underline{F}$ from

$$\Psi_i = (\underline{F}, \Psi_{i+1}) \qquad (A.7)$$

$$(\Psi_i{}^T \Psi_i)^{-1} = \begin{bmatrix} \underline{F}^T\underline{F} & \underline{F}^T \Psi_{i+1} \\ \Psi_{i+1}{}^T\underline{F} & \Psi_{i+1}{}^T \Psi_{i+1} \end{bmatrix}^{-1} \qquad (A.8)$$

$$= \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

$$D = (\Psi_{i+1}^{T} \Psi_{i+1})^{-1} + B^{T} BA^{-1}$$

$$(\Psi_{i+1}^{T} \Psi_{i+1})^{-1} = D - B^{T} BA^{-1} \qquad (A.9)$$

Again, this is easily computed on a digital computer.

APPENDIX B

Five State Model of the Saturn Booster

The matrix differential equations for the five state model of the Saturn Booster are given below. The first three states are: pitch ϕ, pitch rate $\dot{\phi}$, and angle of attack α. These states describe the angular motion of the rigid body in radians for the pitch channel about a nominal trajectory. The last two states are normalized bending η, and normalized bending rate $\dot{\eta}$. These states describe the first bending mode as measured in meters at the engine gimble point.

$$\underline{\dot{X}} = \begin{bmatrix} \dot{\phi} \\ \ddot{\phi} \\ \dot{\alpha} \\ \dot{\eta} \\ \ddot{\eta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & A_{23} & 0 & 0 \\ A_{31} & 1 & A_{33} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & A_{54} & A_{55} \end{bmatrix} \begin{bmatrix} \phi \\ \dot{\phi} \\ \alpha \\ \eta \\ \dot{\eta} \end{bmatrix} + \begin{bmatrix} 0 \\ B_2 \\ B_3 \\ 0 \\ B_5 \end{bmatrix} \beta$$

$$\underline{Y} = \begin{bmatrix} 1 & 0 & 0 & .015 & 0 \\ 0 & 1 & 0 & 0 & .007 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \underline{X}$$

The variable elements of the matrix equations are plotted in Figures (B.1), (B.2), and (B.3).

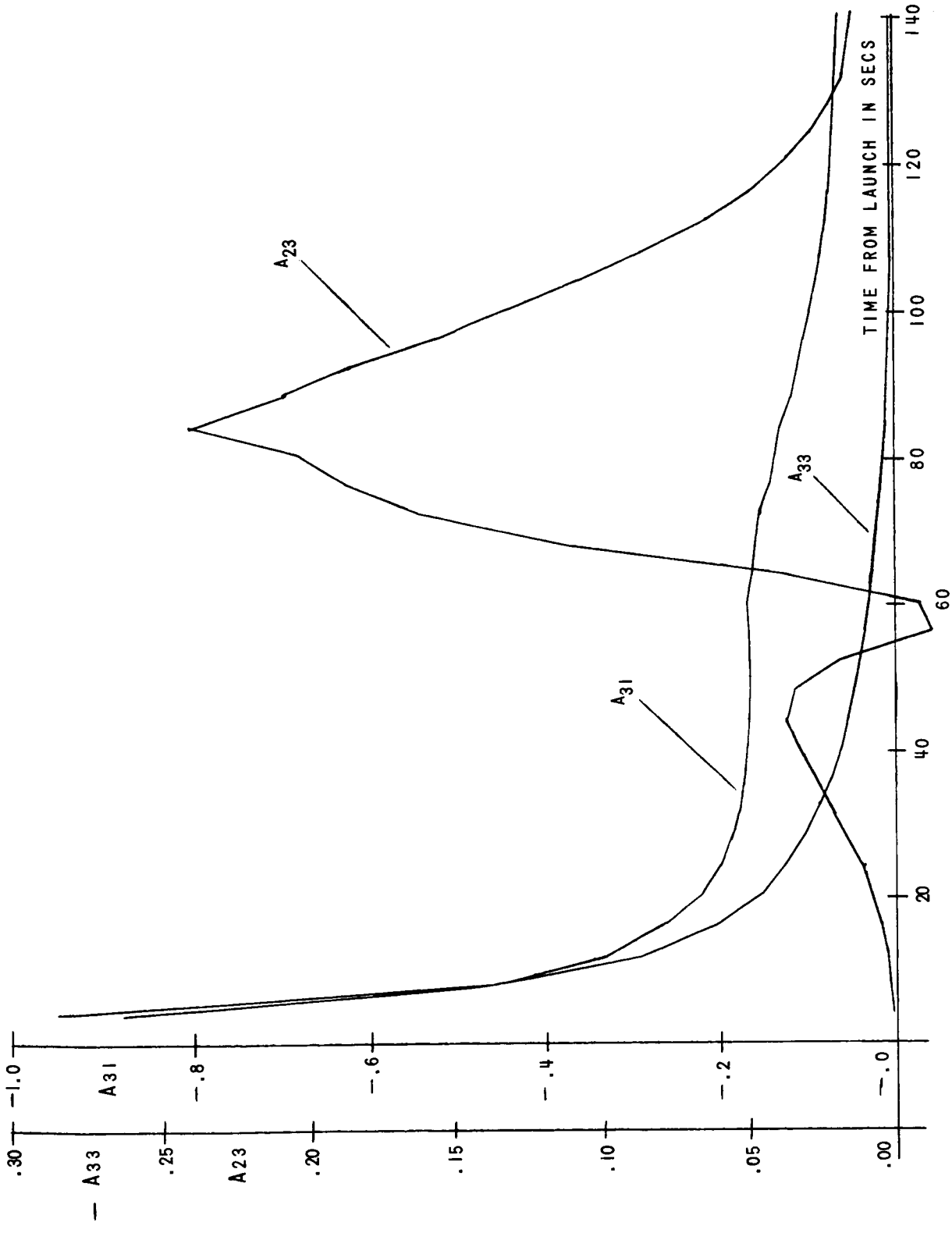


FIGURE B.1 A_{23}, A_{31}, A_{33} vs TIME

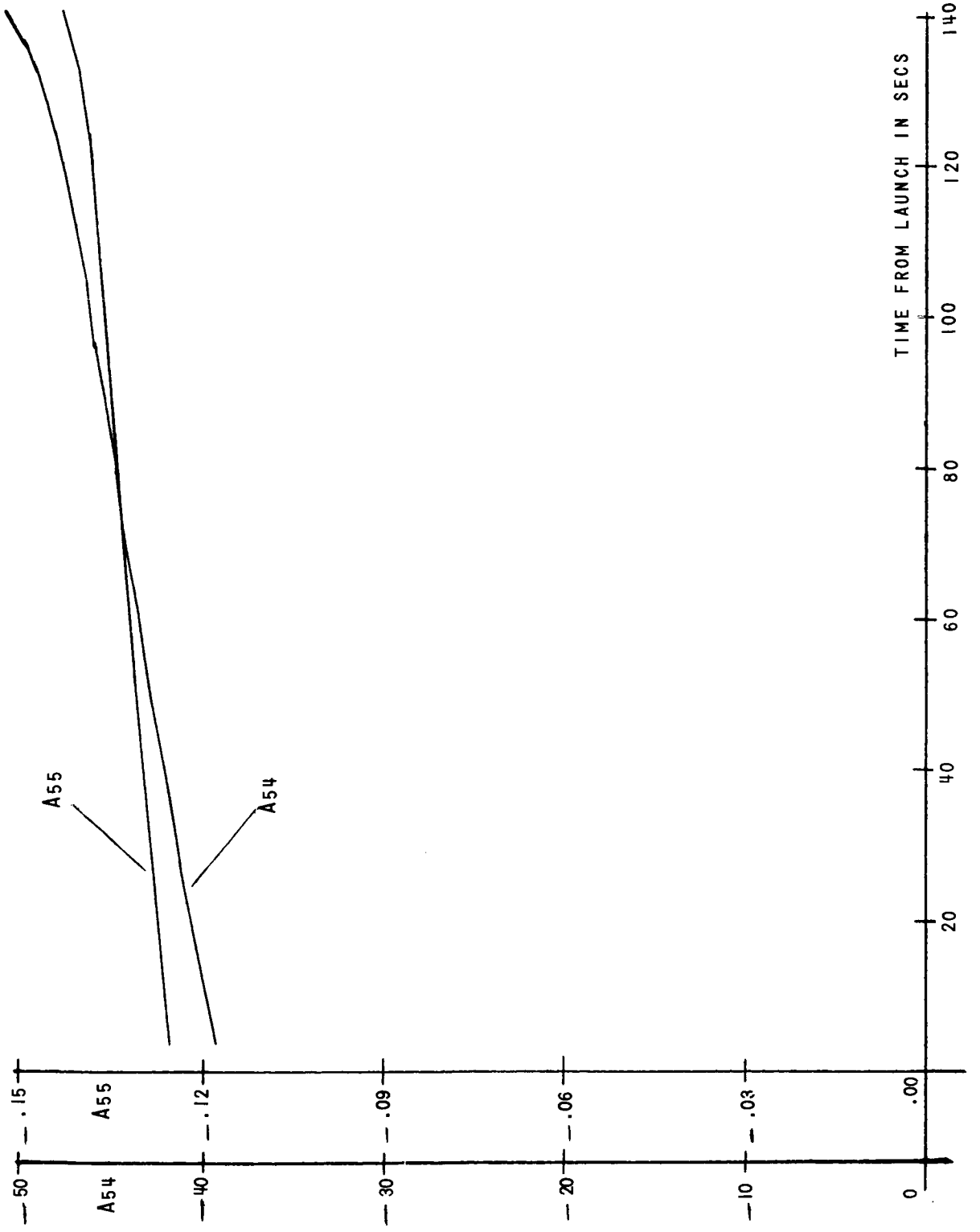


FIGURE B.2 A54, A55 VS TIME

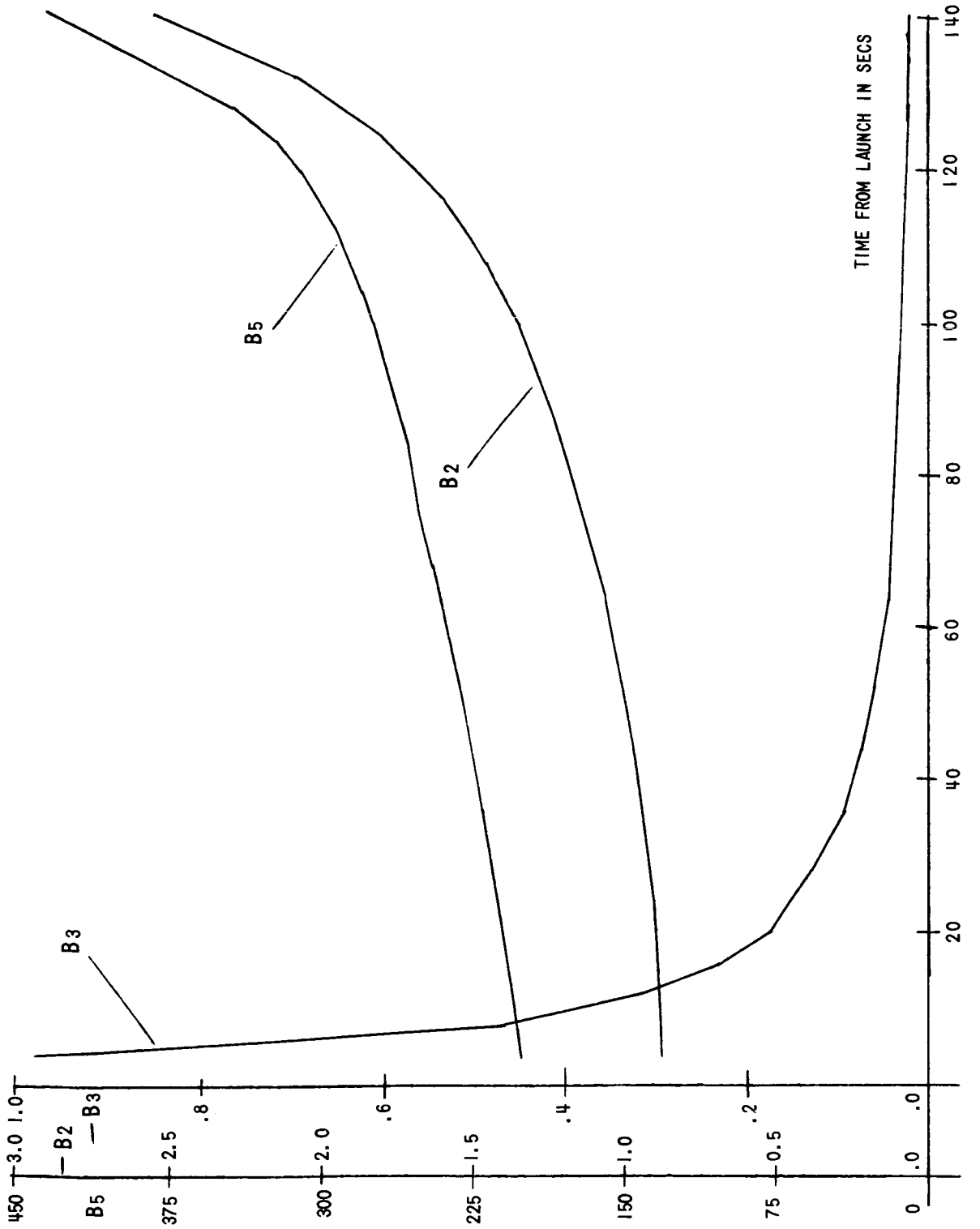
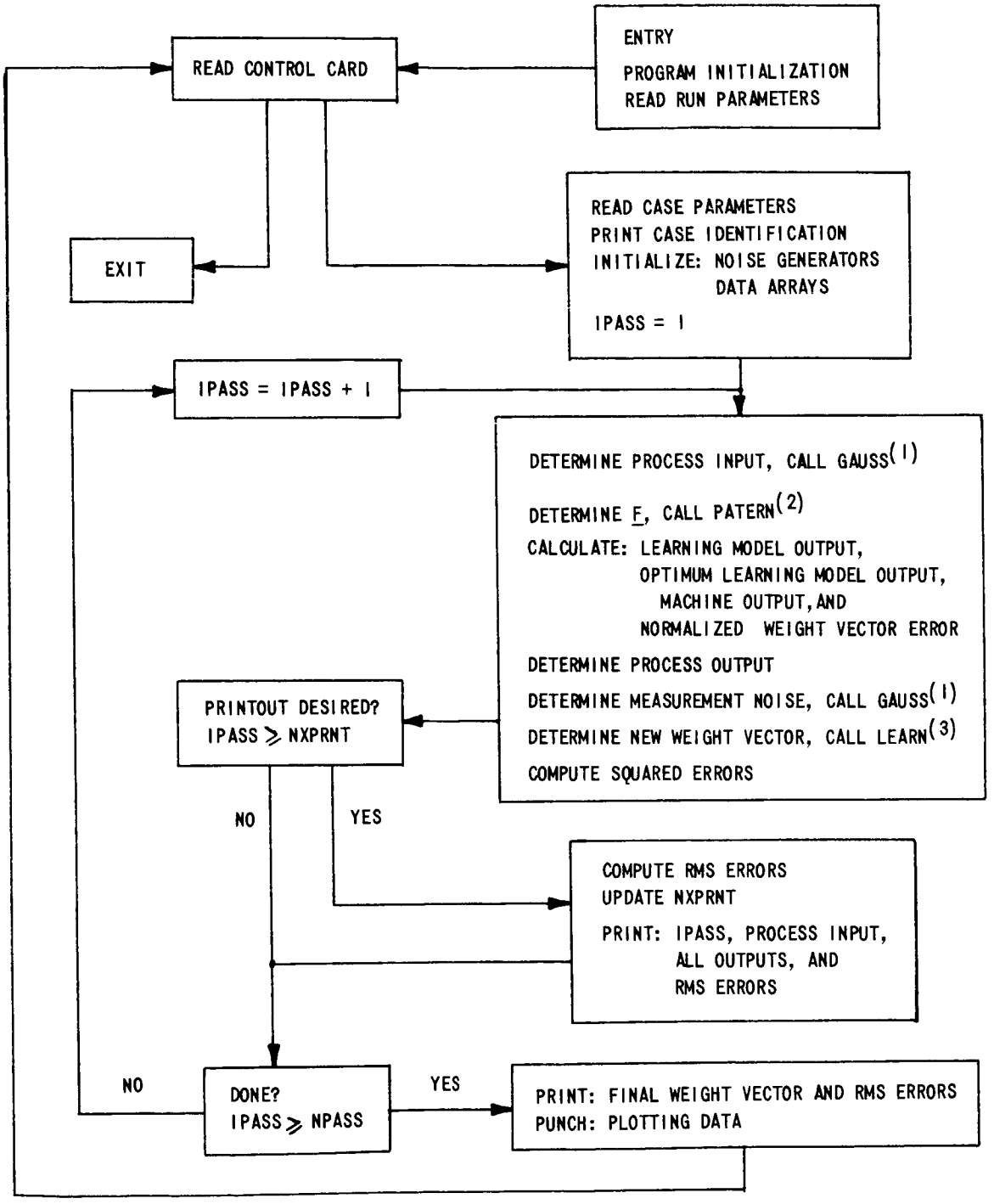


FIGURE B.3 B_2, B_3, B_5 vs TIME

APPENDIX C

Simulation Programs

The simulations of the model tracking system presented in Chapter IV were performed using several variations of a single program. The flow chart for all of these variations is in Figure (C.1). Three subroutines were used. The GAUSS subroutine provided independent sequences of digital gaussian pseudo noise with desired mean, variance, and auto-correlation. The PATTERN subroutine calculated the $\underline{G}$ and $\underline{H}$ vectors from the input and output history. This subroutine is the ϕ-processor of the learning model. The LEARN subroutine calculates the new weight vector. The flow charts of the LEARN subroutine for the error correcting algorithm and the projection algorithm are in Figures (C.2) and (C.3) respectively.



(1) SUBROUTINE GAUSS IS THE NOISE GENERATOR
(2) SUBROUTINE PATERN IS THE Φ-PROCESSOR
(3) SUBROUTINE LEARN IS THE TRAINING ALGORITHM

FIGURE C.1 FLOWCHART OF
MODEL TRACKING IDENTIFICATION PROGRAMS

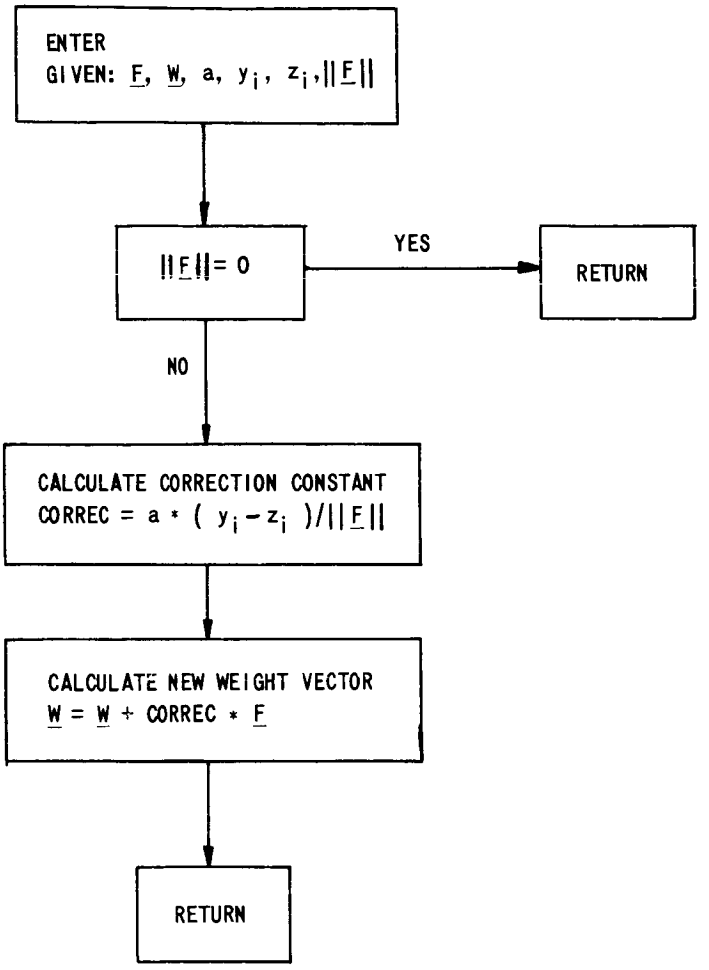
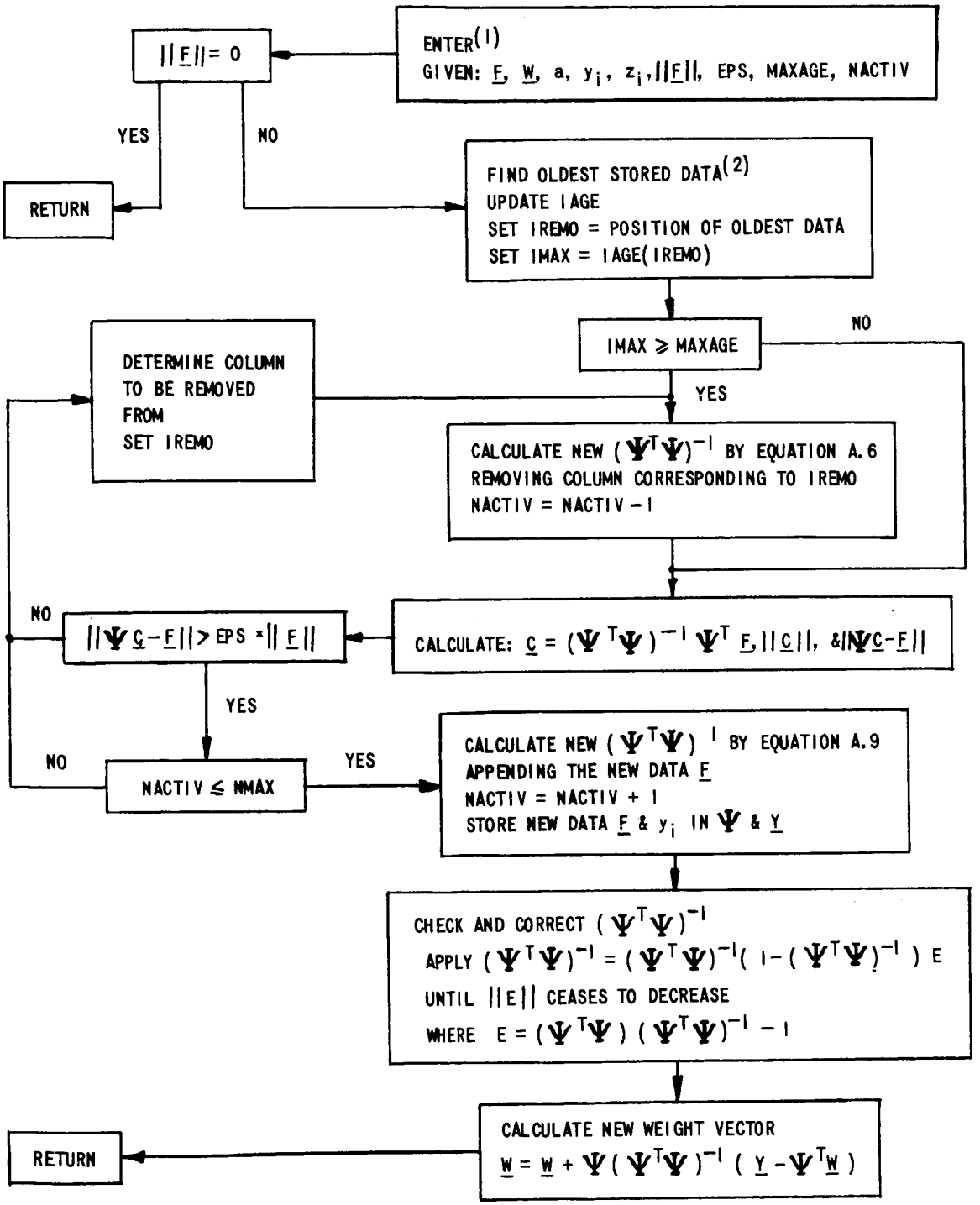


FIGURE C.2 FLOWCHART OF
LEARN SUBROUTINE FOR ERROR CORRECTING ALGORITHM



(1) ALL ARRAYS ARE INITIALIZED TO ZERO BY MAIN PROGRAM
(2) IAGE(J) IS THE NUMBER OF ITERATIONS SINCE THE CORRESPONDING DATA WAS STORED

FIGURE C.3 FLOWCHART OF
LEARN SUBROUTINE FOR PROJECTION ALGORITHM