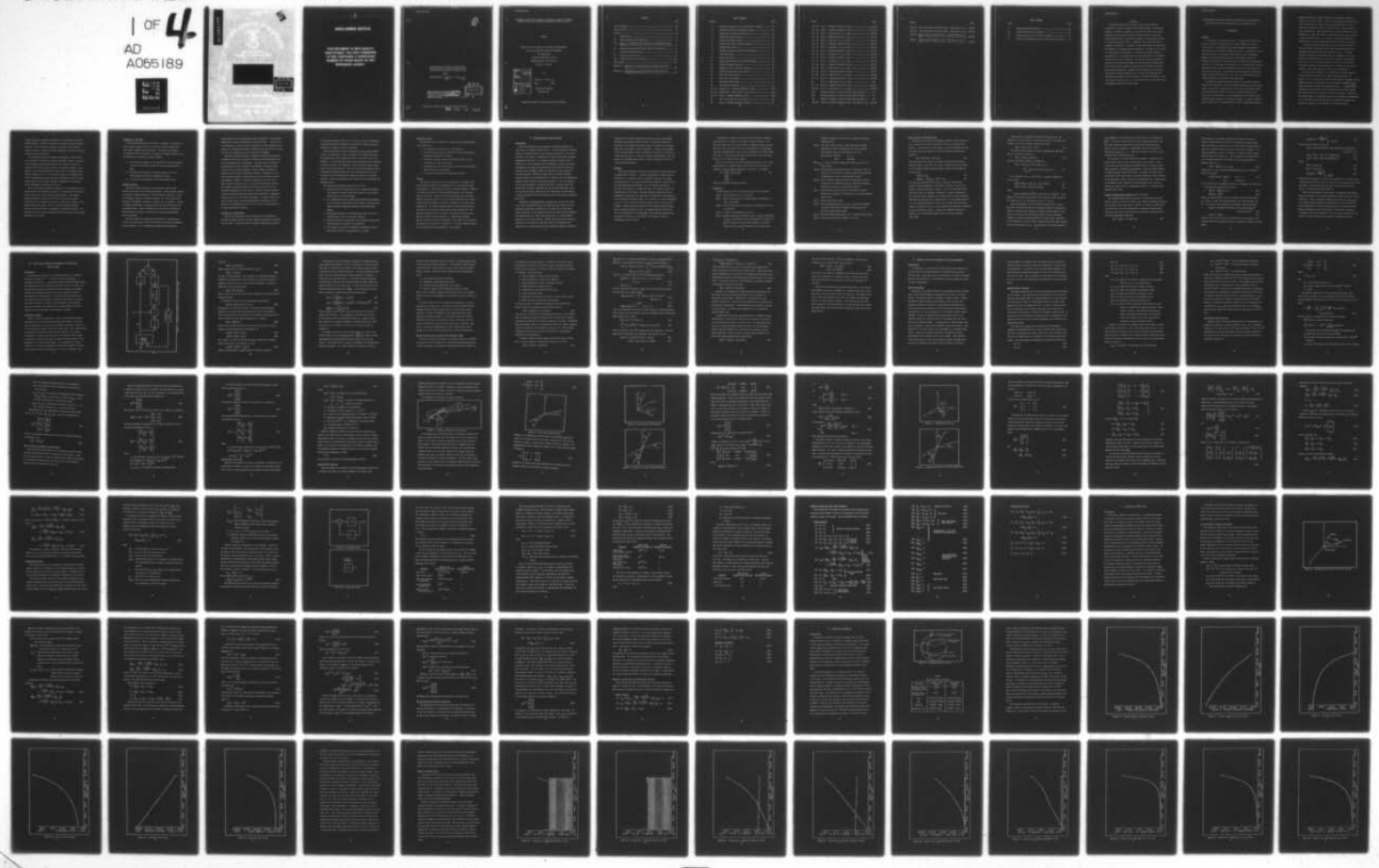
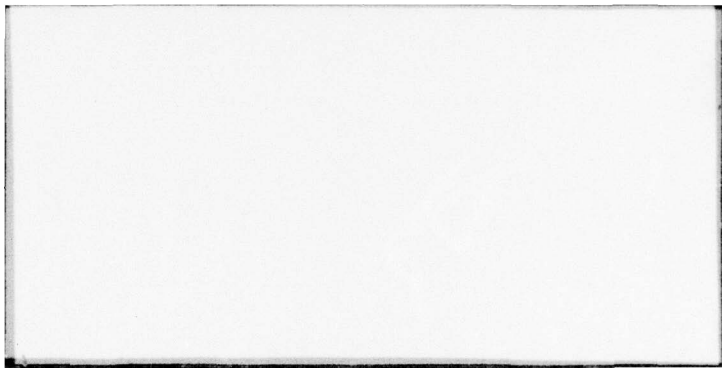
AD-A055 189 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 17/7
A GENERALIZED MONTE CARLO ANALYSIS PROGRAM FOR KALMAN FILTER DE--ETC(U)
DEC 77 K L JACKSON

UNCLASSIFIED AFIT/GGC/EE/77-6 NL

1 OF 4
AD
A055189





DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DDC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

AFIT/CGC/EE/77-6

0

A GENERALIZED MONTE CARLO ANALYSIS PROGRAM FOR KALMAN FILTER DESIGN WITH APPLICATION TO AN AIRCRAFT-TO-SATELLITE TRACKING FILTER

THESIS

AFIT/CGC/EE/77-6 Kenneth L. Jackson, Jr.
Capt USAF

AFIT/GGC/EE/77-6

A GENERALIZED MONTE CARLO ANALYSIS PROGRAM FOR KALMAN FILTER DESIGN WITH APPLICATION TO AN AIRCRAFT-TO-SATELLITE TRACKING FILTER

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

by

Kenneth L. Jackson, Jr.

Capt USAF

Graduate Engineering

December 1977

Contents

List of Figures

List of Tables

Abstract

A generalized Monte Carlo Analysis Program (MCAP) has been developed for linear or extended Kalman filter design. The computer program is similar in structure to the Air Force Avionics Laboratory developed General Covariance Analysis Program (GCAP) so users can easily transition from a covariance analysis to a Monte Carlo analysis of a (extended) Kalman filter. A detailed users' guide for MCAP is included in Appendix A. In addition, this study treats the high accuracy tracking of a satellite from an aircraft. The purpose of the study is to evaluate the feasibility of a reduced order system model for implementation in an extended Kalman filter formulation whose estimates would be used to aid the tracker. The six state model accomplishes tracker state estimation by exploiting the information already available in the tracking geometry, dominant modes of satellite dynamics, and the range measurement. Tracker state estimation is accomplished in the line-of-sight coordinate frame. A Monte Carlo analysis was performed, evaluating the filter against a 42-state truth model. With some proposed modifications, it was concluded that the six state filter is feasible and warrants further study.

A GENERALIZED MONTE CARLO ANALYSIS PROGRAM FOR KALMAN FILTER DESIGN WITH APPLICATION TO AN AIRCRAFT-TO-SATELLITE TRACKING FILTER

I. Introduction

Overview

A large class of estimation problems is concerned with finding an optimal estimate of some quantity (an unknown parameter, a random variable, or a random signal) given noise-corrupted measurements of a function of this quantity (corrupted by noise). In particular, there are many problems with aerospace applications, such as navigation, guidance, and weapon delivery systems, which require that noise-corrupted measurements be used to estimate certain physical parameters precisely. For instance, the techniques used to combine these external measurements with Inertial Navigation System (INS) outputs fall into two general categories: conventional continuous-feedback damping and Kalman filter damping (Ref 1:1). The trend in recent years has been toward extensive use of Kalman filter techniques.

In complex aerospace systems, the number of parameters needed to describe the system (the true system model) accurately may be extremely large. The real time implementation of this true system model within the "optimal" Kalman filter is often not practical because of the large number of parameters, memory requirements, and resultant computational burden imposed on the airborne computer. In order to obtain a Kalman filter which is computationally feasible, intentional modeling

approximations are introduced by deleting nondominant states and terms from the system model. Thus a "suboptimal" Kalman filter based on this reduced order model will give poorer performance, from a statistical standpoint, than the optimal filter. Computer simulation techniques are used to develop the required compensation (filter tuning) and to study the effects of uncertainties (an error analysis) in the true system model. Two types of simulation techniques are commonly used: the covariance analysis and the Monte Carlo analysis.

A covariance analysis generates the second order statistics of the error between suboptimal Kalman filter state estimates and the corresponding parameters from the true system model. A General Covariance Analysis Program (GCAP) has been developed by the United States Air Force Avionics Laboratory and is coming into widespread use (Ref 2). The covariance analysis is somewhat limited because stringent assumptions (see Appendix A) are necessary for the analysis results to be a valid depiction of the error characteristics.

The Monte Carlo analysis, on the other hand, actually conducts a sample-by-sample simulation using random number generators and shaping filters to generate the random error sources. The noisy system measurements are then processed by the suboptimal (extended) Kalman filter algorithm to generate the filter estimates. If many of these simulations are conducted, then the statistics of the error between the filter estimate and the truth model can be computed. Because of the larger number of simulations necessary to determine the performance of one set of Kalman filter parameters, a Monte Carlo analysis can require the expense of large amounts of computer resources.

Consequently when a covariance analysis can be performed, the Monte Carlo analysis is viewed as a step to be performed after a covariance analysis. The Monte Carlo analysis is thus used to fine tune the Kalman filter and validate the filter performance after the covariance analysis has been performed.

The extended Kalman filter cannot be evaluated or tuned totally on the basis of a covariance analysis (like GCAP) because the gain and error covariance matrices depend on the time history of the state estimates (which are not available in a covariance analysis). Nevertheless, the covariance analysis is used to tune a linearized Kalman filter operating over a nominal state trajectory as an approximation to the extended Kalman filter with small deviations from this nominal. Subsequently, the Monte Carlo analysis is conducted to investigate filter performance thoroughly (Ref 3:9-38).

Since a Monte Carlo analysis of promising filter designs should be performed after the covariance analysis, it is desirable to have available a general Monte Carlo Analysis Program (MCAP). This program should be similar in structure to GCAP so that users can easily transition from a covariance analysis to a Monte Carlo analysis of the Kalman filter under study. Also, if possible, problem dependent computer subroutines should be applicable to both GCAP and MCAP. The development of this general Monte Carlo Analysis Program is the primary objective of this study.

Objectives of the Study

The primary objective of this study is to develop a general Monte Carlo analysis program which can be used to evaluate Kalman filter and extended Kalman filter performance. In order to demonstrate the validity of the simulation, the following secondary objectives will be performed to investigate a specific problem.

1. To develop a simulation of the aircraft and satellite dynamics using the truth model developed by Mitchell and modified by Mann.
2. To perform a Monte Carlo performance analysis of the six state reduced order filter investigated by Mann.
3. To analyze the performance of the reduced order filter.

Problem Statement

The high accuracy tracking of one accelerating vehicle from another vehicle has many military applications. These include: aircraft to aircraft tracking, aircraft to missile tracking, aircraft to satellite tracking, missile to aircraft tracking, etc. This project will consider the problem of tracking a satellite from an aircraft. Tracking is usually accomplished by measuring range and angle information with a steerable radar. These measurements are then used to estimate the state (position, velocity, acceleration, or equivalent parameters) from the aircraft.

Since the radar observations of the satellite are uncertain due to noisy measurement data, the state estimates are not exact (deterministic) quantities. If the measurement corruptions are described

statistically and the system dynamics are described by a linear mathematical model, then a Kalman filter will provide the statistics of the state estimates. However, since the problem to be investigated is non-linear, the extended Kalman filter formulation has been chosen to determine the state estimates and associated error covariance.

Previous Air Force Institute of Technology (AFIT) thesis projects by Robert Mitchell (Ref 4) and Robert Mann (Ref 5) established a baseline of information for this problem. In particular, the thesis by Mann extended the work of Mitchell and utilized an extended Kalman filter to perform state estimation. Mann conducted a covariance analysis for two reduced order Kalman filters. As indicated by the Mann covariance analysis, the performance of the two filters was not satisfactory. However, a covariance analysis is viewed as only a first step in the evaluation of the proposed reduced order filters. The covariance equations provide filter error statistics but the covariance analysis does not represent a complete system simulation. A direct statistical simulation (Monte Carlo analysis) is required to continue analysis of the filter performance. In order to limit the scope of this study, a Monte Carlo analysis of the Mann six state filter will be performed since it is more likely to be implemented if it meets performance objectives.

Assumptions and Limitations

Since the system dynamics are non-linear for the aircraft to satellite tracking problem, the basic Kalman filter cannot be applied in this study. Several non-linear estimation methods are available

for handling problems of this type. However, as long as performance is satisfactory the extended Kalman filter will be used because the equations are simple and easy to calculate.

The six state filter will be analyzed with the tracking profile used in the Mann thesis. With this profile, the tracker (aircraft), at the beginning of the simulation, lies in the orbit plane of the satellite. As the simulation progresses, the aircraft moves orthogonal to the orbit plane. The assumption is made that this profile represents a worst case condition. This is a reasonable assumption because the tracking geometry restricts the flow of information to some of the states in the extended Kalman filter and creates observability problems. The six state reduced order filter will be tuned using this tracking profile.

The remaining assumptions parallel those of Mann:

1. Essentially perfect measurements of the tracker acceleration with respect to inertial space are available as the derived output from three accelerometers.
2. The tracking system will provide noise-corrupted measurements of the inertial angular rates of the tracker, range and small angle deviations between the boresight and line-of-sight frames.
3. The tracking system can be instantaneously corrected by the extended Kalman filter state error estimates.
4. Essentially perfect measurements of the tracker elevation and azimuth angles are available from resolvers.
5. The tracker y axis will be inertially stabilized so that it always lies parallel to the geocentric x-y plane.

Research Strategy

The research for this project is divided into the following five major categories:

1. To investigate the background of the problem.
2. To determine the computer simulation method for the general Monte Carlo performance analysis.
3. To develop a general Monte Carlo Analysis Program (MCAP) to perform the system simulation.
4. To perform the Monte Carlo analysis of the six state reduced order Kalman filter using MCAP.
5. To analyze the results of the computer simulation.

Summary

This study is concerned with the development of a general Monte Carlo analysis program and its application to a problem previously investigated in a limited manner through covariance analysis. Chapter II presents the Kalman filter and extended Kalman filter equations used in this study. Chapter III briefly describes the Monte Carlo analysis simulation method and how the error statistics are computed. A more detailed description of MCAP is contained in the users guide (Appendix A). The fourth chapter presents a detailed description of the satellite and aircraft tracker system state equations for the problem investigated with the general Monte Carlo analysis program. Chapter V presents the development of a reduced order extended Kalman filter model used in the analysis. The results of the analysis are presented and discussed in Chapter VI. The final chapter presents the conclusions and recommendations of this project.

II. Extended Kalman Filter Equations

Introduction

This chapter presents the propagation and update equations for both linear and extended Kalman filters. A Kalman filter is, concisely stated, an optimal recursive data processing algorithm for the determination of the states or parameters of a system using noise corrupted measurements (Ref 6:16). If a physical system of interest can be modeled by a set of ordinary linear differential equations and linear measurements with system and measurement noises which are white and Gaussian, then the Kalman filter will provide the best estimate of the system states. However, in many cases of practical interest, physical systems must be represented by a nonlinear set of differential equations and/or nonlinear measurement equations. For such problems, it is often convenient to linearize the system equations about some assumed set of nominal conditions and use developed algorithms (such as the extended Kalman filter which uses reevaluation of the nominal at each measurement time) for estimation about these nominal conditions (Ref 7:57).

Considering the approximations necessary and the fact that there is no "best" suboptimal filter, the extended Kalman filter gain and covariance propagation equations have the same form as the Kalman filter equations, but are linearized about the current state estimates. The linearization is a first term approximation to a Taylor series expansion to the state estimate. Higher order and more exact approximations can be achieved by using more terms of the Taylor series expansion for the nonlinearities and by deriving approximate recursive

relations for the higher moments of the state vector $\underline{x}$ (Ref 8:184), yielding higher order nonlinear filters. As might be expected, if the system nonlinearities are significant, then neglecting the higher order terms will result in biased estimates. However, when compared to the extended Kalman filter, the higher order filters are both more complex and more costly in terms of computer implementation. For this reason, the extended Kalman filter is often considered first in nonlinear estimation problems.

Notation

This study has adopted the notation presented by Wrigley, Hollister, and Denhard (Ref 9:20-23). A vector (represented by a letter with an underbar, $\underline{x}$) is considered to be a geometric entity in real, three dimensional space. The vector represents some physical quantity which has both magnitude and direction. When the physical quantity is measured with respect to a coordinate frame, the vector is said to be coordinatized in that frame. The three numbers associated with the mathematical vector are the components of the physical vector relative to the specified coordinate frame. As an example, if $\underline{x}$ is coordinatized in the "i" frame, the vector would be denoted by $\underline{x}^i$, a three-tuple of numbers. Another vector of interest is the physical angular-velocity vector, generally denoted by a $\underline{w}$. The angular velocity vector will have two subscripts, as an example $\underline{w}^i_{nb}$. The subscripts indicate that the angular velocity, $\underline{w}$, is of frame b with respect to frame n, coordinatized in the i frame.

A 3x3 direction cosine matrix is used in this study to transform the components of a vector in one frame to those in another. The direction cosine matrix is represented by a capital $\underline{C}$ and an underbar. Associated with the letter $\underline{C}$ will be a subscript for the frame *from* which the transformation is made, and a superscript for the new coordinate frame. As an example, $\underline{C}_b^n$ would transform the vector $\underline{x}$ from the b frame to the n frame, as $\underline{x}^n = \underline{C}_b^n \underline{x}^b$.

Where it is necessary to address individual components of a vector coordinatized in a specific frame, the vector will be specified and subscripts used to indicate individual components. For example:

$$\underline{w}^i = \begin{bmatrix} w_x^i \\ w_y^i \\ w_z^i \end{bmatrix} = \begin{bmatrix} w_x^i & w_y^i & w_z^i \end{bmatrix}^T \qquad (1)$$

where T denotes the transpose operator.

Definitions

Listed below are some of the definitions used in this chapter:

$\underline{x}(t_i)$ = system state at time t_i (n-vector)

$\hat{\underline{x}}(t_i^-)$ = filter estimate prior to incorporating a measurement at time t_i (n-vector)

$\hat{\underline{x}}(t_i^+)$ = filter estimate after incorporating a measurement at time t_i (n-vector)

$\underline{z}(t_i)$ = m vector of measurements at time t_i

$\underline{w}(t_i)$ = system dynamics white Gaussian noise s-vector, independent of $\underline{x}(t_o)$, where $E[\underline{w}(t)] = \underline{0}$, and $E[\underline{w}(t)\underline{w}(\tau)^T] = \underline{Q}(t)\delta(t-\tau)$. [$\underline{w}(t)$ is assumed to be zero mean, Gaussian, and white (uncorrelated in time)] with $\underline{Q}(t)$ an sxs positive semi-

definite symmetric matrix that is, in general, piecewise continuous in t.

$\underline{v}(t_i)$ = zero mean, white Gaussian, measurement noise sequence independent of $\underline{w}(t)$ and $\underline{x}(t_o)$ for all time (m-vector). The statistics of $\underline{v}(t_i)$ are $E[\underline{v}(t_i)] = \underline{0}$, and

$$E[\underline{v}(t_i)\underline{v}(t_j)] = \begin{cases} \underline{R}(t_i) & t_i = t_j \\ \underline{0} & \text{otherwise} \end{cases}$$

$\underline{\Phi}(t_i, t_{i-1})$ = state transition matrix from time t_{i-1} to time t_i (nxn matrix)

$\underline{P}(t_i^-)$ = filter computer covariance matrix of state $\underline{x}(t_i)$, also of the error in the estimate of $\underline{x}(t_i)$, prior to incorporating a measurement at time t_i (nxn symmetric matrix)

$\underline{P}(t_i^+)$ = filter computer covariance matrix of state $\underline{x}(t_i)$, also of the error in the estimate of $\underline{x}(t_i)$, after incorporating a measurement at time t_i (nxn symmetric matrix)

$\underline{F}(t)$ = system dynamics matrix (nxn) or the matrix of partial derivatives of $\underline{f}[\underline{x}(t), t]$ with respect to $\underline{x}$ for the extended Kalman filter

$\underline{G}(t)$ = system input matrix (nxs)

$\underline{H}(t_i)$ = system observation matrix at time t_i (mxn) or the matrix of partial derivatives of $\underline{h}[\underline{x}(t), t]$ with respect to $\underline{x}$ for the extended Kalman filter

$\underline{R}(t_i)$ = positive definite measurement noise covariance matrix (mxm)

$\underline{K}(t_i)$ = Kalman gain matrix (nxm) defined at time t_i

Linear Kalman Filter Formulation

The linear Kalman filter formulation presented in this section is for a continuous time system model with discrete time measurement updates. Assume that the system modeling has been completed and that the state vector $\underline{x}(t)$ satisfies the vector stochastic differential equation:

$$\dot{\underline{x}}(t) = \underline{F}(t)\,\underline{x}(t) + \underline{G}(t)\,\underline{w}(t) \tag{2}$$

The state equation is propagated forward in time from the initial condition $\underline{x}(t_o)$. Since the exact initial condition may not be known, it is modeled as being a Gaussian random variable with mean $\hat{\underline{x}}_o$ and covariance $\underline{P}_o$:

$$E[\underline{x}(t_o)] = \hat{\underline{x}}_o \tag{3}$$

$$E[\{\underline{x}(t_o) - \hat{\underline{x}}_o\}\{\underline{x}(t_o) - \hat{\underline{x}}_o\}^T] = \underline{P}_o \tag{4}$$

The initial covariance matrix $\underline{P}_o$ may be positive semidefinite, admitting exact knowledge of the initial conditions of some of the states or linear combinations thereof. It can be shown (Ref 10:157-163) that, under the assumption that $\underline{x}(t_o)$ is either deterministic or a Gaussian random variable, the solution $\underline{x}(t)$ to linear stochastic differential equations such as Equation (2) is a Gauss-Markov process, i.e. the conditional density of $\underline{x}$ at time t_i based upon all realizations of $\underline{x}$ through time t_{i-1} is both Gaussian and completely determined by the process value at t_{i-1}. Because the conditional density is Gaussian, it is completely specified by its mean and covariance (Ref 7:92).

Measurements are available at discrete time points and are assumed to be of the form of a linear combination of the states and corrupted by a white Gaussian sequence (Ref 3:2):

$$\underline{z}(t_i) = \underline{H}(t_i)\ \underline{x}(t_i) + \underline{v}(t_i) \tag{5}$$

The state estimate propagates between measurements (from time t_{i-1}^+ to time t_i^-) according to:

$$\hat{\underline{x}}(t_i) = \underline{\Phi}(t_i, t_{i-1})\ \hat{\underline{x}}(t_{i-1}^+) \tag{6}$$

and the covariance propagates according to:

$$\underline{P}(t_i) = \underline{\Phi}(t_i, t_{i-1})\ \underline{P}(t_{i-1}^+)\ \underline{\Phi}(t_i, t_{i-1})^T + \int_{t_{i-1}}^{t_i} \underline{\Phi}(t_i, \tau)\ \underline{G}(\tau)\ \underline{Q}(\tau)^T\ \underline{\Phi}(t_i, \tau)^T\ d\tau \tag{7}$$

At measurement time t_i, the estimate is updated according to (Ref 10:233):

$$\hat{\underline{x}}(t_i^+) = \hat{\underline{x}}(t_i^-) + \underline{K}(t_i)\ [\underline{Y}_i - \underline{H}(t_i)\ \hat{\underline{x}}(t_i^-)] \tag{8}$$

$$\underline{P}(t_i^+) = \underline{P}(t_i^-) - \underline{K}(t_i)\ \underline{H}(t_i)\ \underline{P}(t_i^-) \tag{9}$$

where

$$\underline{K}(t_i) = \underline{P}(t_i^-)\ \underline{H}(t_i)^T\ [\underline{H}(t_i)\ \underline{P}(t_i^-)\ \underline{H}(t_i)^T + \underline{R}(t_i)]^{-1} \tag{10}$$

where $[\ ]^{-1}$ indicates the inverse of the bracketted matrix and Y_i is the realized value of the measurement $\underline{z}(t_i)$ at time t_i.

Under the assumption that the adequate system model is linear, and that the dynamic driving and measurement noises are Gaussian and white, the Kalman filter provides the optimal estimate $\hat{\underline{x}}(t_i^+)$ of the state of the system (Ref 10:66,214), relative to many optimality criteria with these assumptions $\hat{\underline{x}}(t_i^+)$ is the mean, mode, and median of the conditional density of $\underline{x}(t_i)$, conditioned on the entire measurement history through time t_i. The covariance of the error committed

by using $\hat{\underline{x}}(t_i^+)$ as the estimate of the state at time t_i is denoted by $\underline{P}(t_i^+)$. It should be noted that for a linear estimation problem, the covariance propagation [Equations (7,9)], while depending on the sequence of $\underline{H}(t_i)$ and $\underline{R}(t_i)$, is independent of the time history of measurements $Y_i(Y_1, Y_2, \ldots)$. This will no longer be the case in the extended Kalman filter formulation.

The assumption that the system can be modeled as being driven by white Gaussian noise is often well founded on two accounts. First, it has been found from practical experience that the Gaussian distribution provides a reasonable approximation to observed random behavior in certain physical systems (Ref 7:92). Secondly, the central limit theorem (Ref 7:96) states that if the random phenomenon that we observe at the macroscopic level, is due to the superposition of a large number of independent random variables on the microscopic level, then the macroscopic phenomenon can be adequately modeled as a Gaussian random variable (Ref 10:40).

Extended Kalman Filter Formulation (Ref 3:179-189)

The extended Kalman filter formulation is commonly used in estimation problems in which the adequate state and/or measurement equations are nonlinear rather than linear. Consider, as before, a system that is continuous in time with measurements at discrete sampling times. Assume that the system state satisfies the following nonlinear vector stochastic differential equation:

$$\dot{\underline{x}}(t) = \underline{f}[\underline{x}(t), t] + \underline{G}(t)\, \underline{w}(t) \qquad (11)$$

where $\underline{f}[\underline{x},t]$ is a nonlinear function of the states and time [in general $\underline{f}$ could also be a function of deterministic control inputs $\underline{u}(t)$], and where the vector $\underline{w}(t)$ of zero mean white Gaussian driving noises and covariance $\underline{Q}(t)$ enters in a linear additive manner. The initial condition of $\underline{x}(t_o)$ is modeled as a Gaussian random variable with mean $\hat{\underline{x}}_o$ and covariance $\underline{P}_o$. Noise corrupted vector measurements of a nonlinear function of the states and time are available at discrete times t_i as:

$$\underline{z}(t_i) = \underline{h}[\underline{x}(t_i),\ t_i] + \underline{v}(t_i) \tag{12}$$

where $\underline{v}(t_i)$ is a zero mean white Gaussian sequence with covariance kernel (Ref 3:180).

$$E[\underline{v}(t_i)\ \underline{v}(t_j)] = \begin{cases} \underline{R}(t_i) & \text{for } i = j \\ \underline{0} & \text{otherwise} \end{cases} \tag{13}$$

It is assumed that the processes $\underline{w}$ and $\underline{v}$ are independent of each other.

The filter propagation equations are:

$$\dot{\hat{\underline{x}}}(t/t_i) = \underline{f}[\hat{\underline{x}}(t/t_i),\ t] \tag{14}$$

$$\hat{\underline{x}}(t_i/t_i) = \hat{\underline{x}}(t_i^+) \tag{15}$$

where the notation $\hat{\underline{x}}(t/t_i)$ means the optimal estimate of the state, $\underline{x}$, at time, t, given the updated estimates up to and including time t_i. In addition, (the covariance is propagated approximately by):

$$\underline{P}(t/t_i) = \underline{F}[t;\hat{\underline{x}}(t/t_i)]\ \underline{P}(t/t_i) + \underline{P}(t/t_i)\ \underline{F}[t;\hat{\underline{x}}(t/t_i)]^T + \underline{G}(t)\ \underline{Q}(t)\ \underline{G}(t)^T \tag{16}$$

$$\underline{P}(t_i/t_i) = \underline{P}(t_i^+) \tag{17}$$

where $\underline{F}$ is the matrix of partial derivatives of $\underline{f}$ with respect to $\underline{x}$, evaluated along the trajectory [which is propagated by means of Equation (14)]:

$$\underline{F}[t;\hat{\underline{x}}(t/t_i)] = \left.\frac{\partial \underline{f}[\underline{x},t]}{\partial \underline{x}}\right|_{\underline{x} = \hat{\underline{x}}(t/t_i)} \tag{18}$$

The measurement update is given by:

$$\underline{K}(t_i) = \underline{P}(t_i^-)\,\underline{H}[t_i;\hat{\underline{x}}(t_i^-)]^T \{\underline{H}[t_i;\hat{\underline{x}}(t_i^-)]\,\underline{P}(t_i^-)\,\underline{H}[t_i;\hat{\underline{x}}(t_i^-)] + \underline{R}(t_i)\}^{-1} \tag{19}$$

$$\hat{\underline{x}}(t_i) = \hat{\underline{x}}(t_i) + \underline{K}(t_i)\{\underline{Y}_i - \underline{h}[\hat{\underline{x}}(t_i^-),t_i]\} \tag{20}$$

$$\underline{P}(t_i^+) = \underline{P}(t_i^-) - \underline{K}(t_i)\,\underline{H}[t;\hat{\underline{x}}(t_i^-)]\underline{P}(t_i^-) \tag{21}$$

where

$$\hat{\underline{x}}(t_i^-) = \hat{\underline{x}}(t_i/t_{i-1}) \tag{22}$$

$$\underline{P}(t_i^-) = \underline{P}(t_i/t_{i-1}) \tag{23}$$

$$\underline{H}[t;\hat{\underline{x}}(t_i^-)] = \left.\frac{\partial \underline{h}[\underline{x},t]}{\partial \underline{x}}\right|_{\underline{x} = \hat{\underline{x}}(t_i^-)} \tag{24}$$

The notation t_i^- implies the value of the quantity at the instant prior to the update at time t_i, and t_i^+ is the value of the quantity just after the update. The notation used thus far, except as noted for the $\underline{F}$ matrix, is developed and used in (Ref 3).

Unlike the conventional Kalman filter, the extended Kalman filter gain and estimation error covariance matrices depend on the time history of $\hat{\underline{x}}(t/t_i)$. Since a covariance analysis could only use an $\underline{x}_{nominal}(t)$ as an approximation to $\hat{\underline{x}}(t/t_i)$ as actually used by the filter in its online operation (the linearized Kalman filter), the extended Kalman filter performance should be verified by a Monte Carlo simulation. In general, this form of analysis is both more time consuming and more costly in terms of computer usage than is a covariance analysis.

III. Monte Carlo Simulation and Analysis of a Sub-Optimal Kalman Filter

Introduction

As mentioned in Chapter I, a Monte Carlo analysis is a computer simulation technique that can be used to develop the required (extended) Kalman filter coding, portray filter performance capabilities, and study the effects of parameter variations in the true system model. The Monte Carlo simulation uses random number generators and shaping filters which generate random errors to noise corrupt both system state dynamics and measurements. The noisy system measurements are then processed by the suboptimal (extended) Kalman filter to generate the filter estimates. If many of these simulations are conducted, then one can compute the statistics of the error between the filter estimate and the truth model for quantities of interest.

Performance Analysis

Figure 1 depicts schematically a means of conducting the performance analysis of a given (extended) Kalman filter design. The truth model by definition is the best, most complete mathematical model that can be developed to describe the system under study. Such a truth model is the product of extensive study and data analysis of the system. As noted in the figure, the truth model is an n_t-state model, linear or nonlinear, driven by noise $\underline{w}_t(t)$ (assumed white and Gaussian), that generates the true state values $\underline{x}_t(t)$. It is assumed that the true values for the critical quantities of interest are related to the true state values by the vector function $\underline{c}_t$ (a p vector - generally less

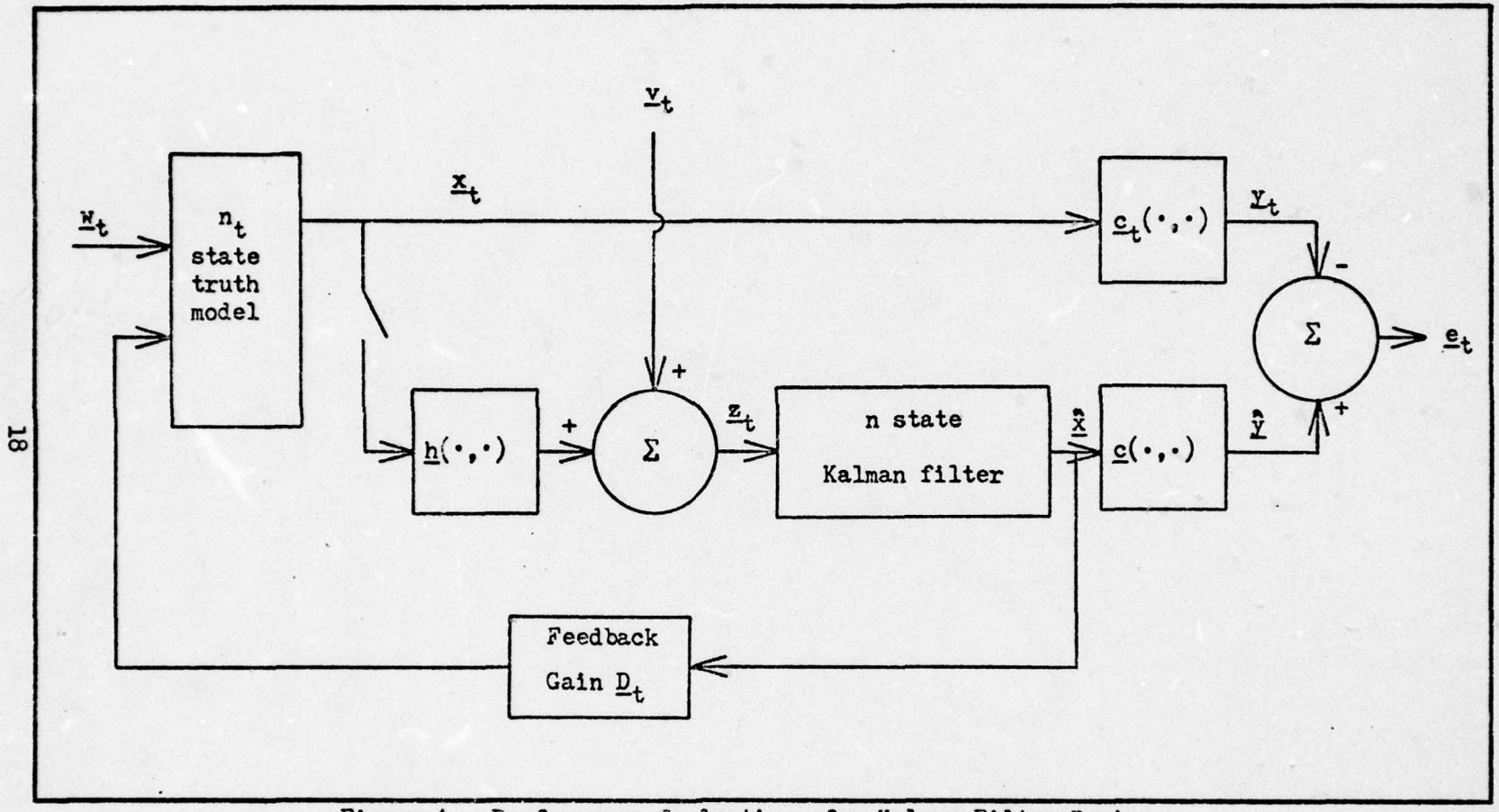


Figure 1. Performance Evaluation of a Kalman Filter Design

than n_t):

$$\underline{y}_t(t) = \underline{c}_t[\underline{x}_t(t), t] \tag{25}$$

Often, $\underline{c}_t[x_t(t), t]$ is a linear function of $\underline{x}_t$, as

$$\underline{y}_t(t) = \underline{c}(t)\underline{x}(t) \tag{26}$$

In fact, in many instances, the quantities of interest are simply a subset of the truth model states; letting the components of $\underline{x}(t)$ be ordered so that the first p states are the quantities of interest, this yields the particular form:

$$\underline{y}_t(t) = [\underline{I} \mid \underline{0}]\, \underline{x}_t(t) \tag{27}$$

Thus we have access to the "real world" values for comparison with the filter estimates.

At discrete times (t_i) the m-dimensional measurements:

$$\underline{z}_t(t_i) = \underline{h}[\underline{x}_t(t_i), t_i] + \underline{v}_t(t_i) \tag{28}$$

are presented to the Kalman filter. The Kalman filter algorithm processes the measurements and produces the state estimates $\hat{\underline{x}}(t)$. These state estimates are related to the estimates of critical quantities of interest by the vector function $\underline{c}$ (a p vector):

$$\hat{\underline{y}}(t) = \underline{c}[\hat{\underline{x}}(t), t] \tag{29}$$

Thus the true error committed by the Kalman filter ($\underline{e}_t$) at time t_i, before and after measurement incorporation, is:

$$\underline{e}_t(t_i^-) = \hat{\underline{y}}(t_i^-) - \underline{y}_t(t_i) \tag{30}$$

$$\underline{e}_t(t_i^+) = \hat{\underline{y}}(t_i^+) - \underline{y}_t(t_i) \tag{31}$$

If a feedback control system with impulsive correction is assumed, then a third error is of interest as well:

$$\underline{e}_t(t_i^{+c}) = \hat{\underline{y}}(t_i^{+c}) - \underline{y}_t(t_i^{\,c}) \tag{32}$$

where the superscript c denotes after the control is applied.

The objective of the performance analysis is to characterize the error process statistically. This is accomplished in the Monte Carlo simulation by generating many samples of the error process and then computing the sample statistics directly. If enough samples are generated, then the sample statistics should approximate the process statistics very well (Ref 3:6-71). The sample statistics computed at each point in time are the calculated mean error ($\bar{\underline{e}}$), calculated standard deviation of the error ($\underline{s}_e$), and the ensemble average of the filter covariance diagonal terms ($\overline{\underline{PF}}_{kk}$). The calculations are made over the ensemble of runs (N) for each time point (t_j):

$$\bar{\underline{e}}(t_j) = \frac{1}{N} \sum_{i=1}^{N} [\hat{\underline{x}}_i(t_j) - \underline{x}_{ti}(t_j)] \tag{33}$$

$$\underline{s}_e(t) = \left[\{ \sum_{i=1}^{N} [\hat{\underline{x}}_i(t_j) - \underline{x}_{ti}(t_j)]^2 - N\, \bar{\underline{e}}^2(t_j) \} / N-1 \right]^{1/2} \tag{34}$$

$$\overline{\underline{PF}}_{kk}(t_j) = \frac{1}{N} \sum_{i=1}^{N} \underline{PF}_{kki}(t_j) \tag{35}$$

These equations can be found in any good statistics text (see for example Ref 11:26,40). It is important to note the use of 1/(N-1) in the expression for the standard deviation, $\underline{s}_e$, instead of 1/N; this results in an unbiased estimator of variance before the $[\ ]^{1/2}$ is taken and is customarily used for sampled data (for small or moderate N).

One useful output of the Monte Carlo analysis is a plot of the filter estimate of the standard deviation, $\sqrt{\overline{\underline{PF}}_{kk}(t_j)}$, along with the corresponding computed standard deviation, $\underline{s}_e(t_j)$ for all t_j or interest. The Kalman filter is tuned by minimizing the computed error standard deviations. As a rule of thumb, good tuning is achieved

when the filter estimated standard deviations are approximately equal to the computed standard deviations. If the problem under investigation is sensitive to small changes in the filter tuning parameters [$\underline{P}(t_o)$, and time histories of $\underline{Q}(t)$ and $\underline{R}(t)$] then the investigator must:

1. Conduct the Monte Carlo simulation
2. Examine the standard deviation plots
3. Determine tuning parameter(s) to be changed
4. Repeat 1 through 3 until adequate tuning is achieved

Thus it can be readily visualized that precise tuning of the Kalman filter by Monte Carlo techniques is very costly in terms of effort and time.

A second useful output of the Monte Carlo analysis is a plot of the mean error versus time. This plot is usually obtained with one standard deviation bounds plotted above and below the mean error. This plot is useful to determine if the suboptimal Kalman filter provides the desired accuracy. If the filter starts from initial conditions other than the truth model initial conditions, then the filter performance to non-zero initial error can be observed. Bias errors, a problem of significance to many extended Kalman filter designs, also become readily apparent from the plots of this form.

Digital Simulation of the Truth Model and Filter Model

In order to simulate the system on a digital computer, the problem must be divided into segments which, when processed sequentially over a given time period, will provide a realistic portrayal of filter use

in actual on-line implementation. A convenient time period is the sample period between times at which measurements become available. Then the entire procedure is iterated over some desired time interval of interest. The segments are:

1. Time propagation of system state equations
2. Time propagation of filter state equations
3. Time propagation of filter covariance equations
4. Update of filter covariance matrix
5. Generation of measurements for the filter
6. Update of filter states
7. Application of impulsive feedback to "update" system states
8. Reset the filter states after the feedback

The first segment is simulated by integrating the n_t-dimensional system (truth model) stochastic differential equation:

$$\dot{\underline{x}}_t(t) = \underline{f}_t[\underline{x}_t(t), t] + \underline{G}_t(t)\underline{w}_t(t) \tag{36}$$

from time t_i to time of the next measurement time t_{i+1}. The integration is typically performed by either Euler or Runga-Kutta numerical integration methods with a specified integration step size. Integrating the deterministic part of the differential equation is readily apparent, however, the additive white Gaussian noise $\underline{w}_t$ is not. The following derivation shows how the solution to the stochastic differential equation can be approximated.

To provide insight into the approximate solution form, consider first a linear stochastic differential equation of the form:

$$\dot{\underline{x}}(t) = \underline{F}(t)\ \underline{x}(t) + \underline{G}(t)\ \underline{w}(t) \tag{37}$$

where $\underline{w}(t)$ is a zero mean white Gaussian noise with $E[\underline{w}(t)\underline{w}(t+t)^T] = \underline{Q}(t)\delta(t)$. An equivalent discrete time system equation would be:

$$\underline{x}(t_{i+1}) = \underline{\Phi}(t_{i+1},t_i)\,\underline{x}(t_i) + \int_{t_i}^{t_{i+1}} \underline{\Phi}(t_{i+1},t_i)\,\underline{G}(\tau)\underline{w}(\tau)\,d\tau \tag{38}$$

$$= \underline{\Phi}(t_{i+1},t_i)\,\underline{x}(t_i) + \underline{w}_d(t_i) \tag{39}$$

where $\underline{\Phi}(\cdot,\cdot)$ satisfies both of the following relationships:

$$\frac{d}{dt}\underline{\Phi}(t,t_i) = \underline{F}(t)\,\underline{\Phi}(t,t_i) \text{ for } t \text{ in the interval } (t_i,t_{i+1}) \tag{40}$$

$$\underline{\Phi}(t_i,t_i) = \underline{I} \tag{41}$$

For this discrete time system, $\underline{w}_d(t_i)$ is a zero mean white Gaussian discrete-time noise with strength:

$$E[\underline{w}_d(t_i)\underline{w}_d(t_i)^T] = \int_{t_i}^{t_{i+1}} \underline{\Phi}(t_{i+1},\tau)\,\underline{G}(\tau)\,\underline{Q}(\tau)\,\underline{G}(\tau)^T\,\underline{\Phi}(t_{i+1},\tau)\,d\tau \tag{42}$$

$$= \underline{Q}_d(t_i) \tag{43}$$

$$E[\underline{w}_d(t_i)\underline{w}_d(t_j)^T] = \underline{0} \quad \text{for } t_i \neq t_j \tag{44}$$

Now if the integration step size is small relative to system characteristic response time and constant (Δt sec in duration) then the following approximations can be made:

$$\underline{\Phi}(t_{i\ 1},t_i) \cong \underline{I} + \underline{F}(t_i)\,\Delta t \tag{45}$$

$$\int_{t_i}^{t_{i+1}} \underline{\Phi}\,\underline{G}\,\underline{Q}\,\underline{G}^T\,\underline{\Phi}^T\,d\tau \cong [\underline{G}(t_i)\,\underline{Q}(t_i)\,\underline{G}(t_i)^T]\,\Delta t \tag{46}$$

where all terms of order Δt^2 or higher have been neglected. Thus one approximate simulation of the continuous time system would be:

$$\underline{x}(t_{i+1}) = [\underline{I} + \underline{F}(t_i)\,\Delta t]\,\underline{x}(t_i) + \underline{w}_d(t_i) \tag{47}$$

$$= \underline{x}(t_i) + \underline{F}(t_i)\,\underline{x}(t_i)\,\Delta t + \underline{w}_d(t_i) \tag{48}$$

where $\underline{w}_d(t_i)$ is described by:

$$E[\underline{w}_d(t_i)\underline{w}_d(t_i)^T] = [\underline{G}(t_i)\ \underline{Q}(t_i)\ \underline{G}(t_i)^T]\ \Delta t \tag{49}$$

Thus the differential equation is integrated using Equation (46) and noise generators utilized for the $\underline{w}_d(t_i)$ term. Higher order integration techniques can be applied to the nonhomogeneous portion of the differential equation, with the same $\underline{w}_d(t_i)$ as above used as the forcing terms. It should be noted that this technique is directly extendable to nonlinear equations of the form:

$$\dot{\underline{x}}(t) = \underline{f}[\underline{x}(t),t] + \underline{G}(t)\ \underline{w}(t) \tag{50}$$

Segment two is performed by calculating the filter equation (6). The filter covariance propagation Equation (7) is calculated to accomplish segment three. Segment four is performed by solving Equations (10 and 9). Segment five uses Equation (5) to generate the m-dimensional measurements. Noise generators are utilized to provide the $\underline{v}_t(t_i)$ term. The filter update Equation (8) is calculated to perform segment six.

A Kalman filter will often be implemented in an indirect feedback configuration, in which filter error state estimates are fed back to the actual system to correct the actual error. If the system correction can be performed rapidly enough compared to the filter update period, then the feedback can be simulated as an impulsive change. After application of the correction, assuming linear feedback, the truth model state process becomes (Ref 3:6-76):

$$\underline{x}_t(t_i^c) = \underline{x}_t(t_i) - \underline{D}_t(t_i)\ \hat{\underline{x}}(t_i) \tag{51}$$

The filter should be "told" that this feedback to the system has occurred, so its state estimate is modified as:

$$\hat{\underline{x}}(t_i^{+c}) = \hat{\underline{x}}(t_i^{+}) - \underline{D}(t_i)\,\hat{\underline{x}}(t_i^{+}) \tag{52}$$

$$= [\underline{I} - \underline{D}(t_i)]\,\hat{\underline{x}}(t_i^{+}) \tag{53}$$

Note that if the filter is estimating error states then the $\underline{D}(t_i)$ matrix will typically be an identity matrix. Segments seven and eight are performed by calculating the results of Equations (51) and (52).

Each of these segments are performed sequentially at the measurement update time and the simulation time, system state, filter state, and filter covariance values saved on a data tape. This process continues until the simulation final time is reached, thus completing one run of a Monte Carlo simulation. After many runs of the Monte Carlo simulation have been performed, then the data tape can be read, the sample statistics calculated using Equations (33-35) and computer plots created.

IV. Satellite and Aircraft Tracker System State Equations

Introduction

In this chapter, the satellite (target) and aircraft tracker system state and measurement equations are developed for the system (truth) model. Since this study is a follow-on to the work of Mitchell (Ref 4) and Mann (Ref 5), the presentation in this chapter will parallel their development.

System Description

Physically, the problem consists of an orbiting satellite (target) and a moving aircraft which is equipped with some type of tracking device. The tracking device is typically a radar or laser. Because the aircraft and satellite are both moving in this problem, the tracking system must be capable of following the satellite motion. The tracking system is equipped with three rate gyros which provide measurements of the three components of the tracker inertial angular velocity. Because of tracking system dynamics and errors in the tracking system, the satellite will not lie exactly along the boresight of the tracking device. In this problem, a tracker control system is available to correct the estimated tracker error angles. These estimates of the error angles will be provided by an extended Kalman filter. Three accelerometers, mounted on the aircraft, provide measurements of the specific force of the tracker. These specific force measurements are used by the aircraft Inertial Navigation System (INS) to determine the tracker inertial position and velocity.

In this study, it is assumed that uncorrupted (treated as perfect) tracker inertial position and velocity information is available from the aircraft INS. In addition to the tracker angular velocity, the tracking system provides measurements of range to the target and the true line of sight. The measurements are all assumed to be imperfect. The nature of the noise corrupting the measurements will be discussed later.

Satellite State Equations

The system model used for the satellite dynamics will be presented in this section. The satellite state equations will not be formally derived; the approach taken is straightforward and the derivations of the models may be found by the interested reader in any good astrodynamics text (see for example Ref 12). As noted in Mann (Ref 5:8), due to limitations in the computer simulation, the solar pressure perturbative acceleration is deleted from Mitchell's system model. To compensate for this additional unmodeled effect, the strengths of the driving noise sources on the satellite equations are increased by an appropriate amount.

The target state equations are expressed in the geocentric equatorial nonrotating coordinate system with the x-axis lying along the line of the mean vernal equinox (Figure 2). This coordinate system will be considered to be an inertial ("i") frame for this application. The state equations describing the satellite's motion are:

$$\dot{x}_1 = x_4 \tag{54}$$

$$\dot{x}_2 = x_5 \tag{55}$$

$$\dot{x}_3 = x_6 \tag{56}$$

$$\dot{x}_4 = a_{gx} + a_{mx} + a_{sx} + a_{dx} + w_1 \tag{57}$$

$$\dot{x}_5 = a_{gy} + a_{my} + a_{sy} + a_{dy} + w_2 \tag{58}$$

$$\dot{x}_6 = a_{gz} + a_{mz} + a_{sz} + a_{dz} + w_3 \tag{59}$$

where

x_1, x_2, x_3 represent the target inertial position components along the x, y, z axes respectively

x_4, x_5, x_6 represent the target inertial velocity

$\underline{a}_g$ is the earth's gravitational acceleration vector

$\underline{a}_m$ is the lunar gravitational perturbation vector

$\underline{a}_s$ is the solar gravitational perturbation vector

$\underline{a}_d$ is the atmospheric drag acceleration vector

w_1, w_2, w_3 are zero-mean independent white, Gaussian noises included to account for unmodeled effects. These effects include: solar pressure perturbations, higher order gravitational terms, and uncertainties in the models used in this study, such as deviations in the atmospheric density.

In order to determine the strengths of the white noises, consider the following calculations. For a relatively small satellite - in a 200 Km circular near polar orbit - with a solar pressure coefficient equal to that of a vehicle with a projected surface towards the sun of 10 m^2 and a ballistic coefficient of 0.15, the terms of $\dot{x}_4$ have deterministic values of:

$a_{gx} = -7.55$ m/sec^2 - acceleration due to full gravity

$a_{mx} = +5.0 \times 10^{-7}$ m/sec^2 - lunar perturbative acceleration

$a_{sx} = +2.0 \times 10^{-9}$ m/sec^2 - solar gravitational perturbative acceleration

$a_{dx} = -9.0 \times 10^{-9}$ m/sec^2 - drag acceleration

These values have been determined using the models proposed later in this section with the satellite at 30° north latitude and the sun and moon positioned for worst case effects. The unmodeled solar pressure perturbation on the satellite under these assumptions would be -2.0×10^{-9} m/sec^2. Because of the aforementioned criteria of modeling all perturbative accelerations of magnitude greater than 10^{-9} m/sec^2, a reasonable value for the distribution standard deviation of w_1 due to modeling uncertainty and higher order effects is 1×10^{-9} m/sec^2. Taking into account the unmodeled effects due to the solar pressure perturbations, w_1, w_2, and w_3 are modeled as zero mean independent white Gaussian noises with distribution one sigma values of 3×10^{-9} m/sec^2.

Gravitational Field Modeling

Modeling of the earth's gravitational field is accomplished in a geocentric, equatorial, rotating coordinate frame. The relationship between this "r" frame $(x^r, y^r, z^r)^T$ and the inertial "i" $(x^i, y^i, z^i)^T$ coordinate frame used in the previous section is shown in Figure 2. The transformation matrix from the rotating (r) to the inertial (i) frame $\underline{C}_r^i$ is defined as:

$$\underline{C}_r^i = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{60}$$

where

$$\theta = \theta_g + w_{ie}\, t \tag{61}$$

and

θ_g = local sidereal time at t=0

w_{ie} = earth's angular rotation rate (7.292×10^{-5} rad/sec)

t = time

The potential model that was chosen includes tesseral, zonal, and sectorial harmonics up to and including (6,6) (Ref 12:173-180). The gravitational potential $U(x^r, y^r, z^r)$ in the "r" frame is (Ref 12: 175):

$$U = \frac{k_e^2\, m}{r}\left[1 + \sum_{k=2}^{6}\left(\sum_{m=0}^{k} P_k^{(m)} \frac{\sin\phi}{r^k}\left\{C_{k,m}\cos(m\lambda_E) + S_{k,m}\sin(m\lambda_E)\right\}\right)\right] \tag{62}$$

where the terms in Equation (62) are defined as follows:

$P_k^{(m)}(\sin\phi)$ are Legendre functions:

$$P_k^{(m)}(\sin\phi) \triangleq (1 - \sin\phi)^{m/2} \frac{d^m}{d(\sin\phi)^m}\{P_k(\sin\phi)\}$$

and $P_k(\sin\phi)$ is the Legendre polynomial with argument $\sin\phi$.

m is the mass of the earth (5.983×10^{24} Kg)

k_e is the gravitational constant for the earth ($k_e^2 = 3.986 \times 10^{14}$ m^3/sec^2)

r is the radial distance from the earth's center to the satellite

ϕ is the geocentric declination angle of the satellite

λ_E is the longitude of the satellite with respect to the prime meridian.

$S_{k,m}$ and $C_{k,m}$ are the harmonic coefficients for the gravitational potential such that $C_{k,0} = -J_k^{(0)}$ and $S_{k,0} = 0$ and the $J_k^{(0)}$ coefficients are the zonal harmonics.

$C_{k,m}$ and $S_{k,m}$ are termed tesseral harmonics if $m \neq k$, $m \neq 0$ and sectorial harmonics if $m = k$ (Ref 4:115-117).

The components of the gravitational acceleration vector along the x^r, y^r, z^r axes - a_{gx}^r, a_{gy}^r, a_{gz}^r - can now be determined by

$$(\underline{a}_g)^r \triangleq \begin{bmatrix} a_{gx}^r \\ a_{gy}^r \\ a_{gz}^r \end{bmatrix} = \begin{bmatrix} \frac{\partial U}{\partial x^r} \\ \frac{\partial U}{\partial y^r} \\ \frac{\partial U}{\partial z^r} \end{bmatrix} \tag{63}$$

and the gravitational acceleration vector in the inertial frame can be determined from

$$(\underline{a}_g)^i = \underline{C}_r^i \, (\underline{a}_g)^r \tag{64}$$

Lunar and Solar Perturbative Accelerations

The perturbative accelerations on the satellite due to the lunar and solar gravitational fields is presented in this section. Since the time elapsed during a complete tracking pass is small when compared to the inertial dynamics of the moon and sun, they are considered stationary for the purposes of this study.

The lunar perturbative acceleration vector in the inertial frame is denoted by $(\underline{a}_m)^i$, with the components along the inertial x, y, and z axes denoted by a^i_{mx}, a^i_{my}, and a^i_{mz} respectively. The position vector of the moon in the inertial frame is denoted as

$$(\underline{r}_m)^i \triangleq \begin{bmatrix} r^i_{mx} \\ r^i_{my} \\ r^i_{mz} \end{bmatrix} \tag{65}$$

The position vector of the moon relative to the vehicle is denoted by

$$(\underline{r}_{ms})^i = (\underline{r}_m)^i - (\underline{r}_s)^i = \begin{bmatrix} r^i_{mx} - x_1 \\ r^i_{my} - x_2 \\ r^i_{mz} - x_3 \end{bmatrix} \tag{66}$$

and the perturbative acceleration on the vehicle (satellite) due to the moon's gravitational field is

$$(\underline{a}_m)^i = \mu_m \begin{bmatrix} \dfrac{r^i_{mx} - x_1}{r^3_{ms}} - \dfrac{r^i_{mx}}{r^3_m} \\ \dfrac{r^i_{my} - x_2}{r^3_{ms}} - \dfrac{r^i_{my}}{r^3_m} \\ \dfrac{r^i_{mz} - x_3}{r^3_{ms}} - \dfrac{r^i_{mz}}{r^3_m} \end{bmatrix} \tag{67}$$

where

μ_m = gravitational parameter of the moon (4.903×10^{12} m^3/sec^2)

$r_{ms} \triangleq [(r^i_{mx}-x_1)^2 + (r^i_{my}-x_2)^2 + (r^i_{mz}-x_3)^2]^{1/2}$

$r_m \triangleq (r^{i\,2}_{mx} + r^{i\,2}_{my} + r^{i\,2}_{mz})^{1/2}$

x_1, x_2, x_3 = the satellite position in inertial frame.

In a similar manner, the perturbative acceleration due to the sun's gravitational field is

$$(\underline{a}_s)^i \triangleq \begin{bmatrix} a^i_{sx} \\ a^i_{sy} \\ a^i_{sz} \end{bmatrix} \tag{68}$$

The position vector of the sun in inertial frame is defined as

$$(\underline{r}_s)^i \triangleq \begin{bmatrix} r^i_{sx} \\ r^i_{sy} \\ r^i_{sz} \end{bmatrix} \tag{69}$$

and the perturbative acceleration on the vehicle due to the sun's gravitational field is

$$(\underline{a}_s)^i = \mu_s \begin{bmatrix} \dfrac{r^i_{sx} - x_1}{r^3_{ss}} - \dfrac{r^i_{sx}}{r^3_s} \\ \dfrac{r^i_{sy} - x_2}{r^3_{ss}} - \dfrac{r^i_{sy}}{r^3_s} \\ \dfrac{r^i_{sz} - x_3}{r^3_{ss}} - \dfrac{r^i_{sz}}{r^3_s} \end{bmatrix} \tag{70}$$

where

μ_s = gravitational parameter of the sun (1.327×10^{20} m^3/sec^2)

$$r_{ss} \triangleq \left[(r^i_{sx}-x_1)^2 + (r^i_{sy}-x_2)^2 + (r^i_{sz}-x_3)^2\right]^{1/2}$$

$$r_s \triangleq (r^{i\,2}_{sx} + r^{i\,2}_{sy} + r^{i\,2}_{sz})^{1/2}$$

Acceleration Due to Drag

Satellite acceleration due to drag is modeled as a function of the height above the earth's surface, the velocity of the satellite relative to the rotating atmosphere and the vehicle ballistic coefficient:

$$(\underline{a}_d)^i = \frac{1}{2} \lambda B \left|\underline{v}_a\right| \ (\underline{v}_a)^i \tag{71}$$

where

$(\underline{a}_d)^i$ = drag acceleration vector in inertial frame

$$(\underline{v}_a)^i = \begin{bmatrix} x_4 + w_{ie}x_2 \\ x_5 - w_{ie}x_1 \\ x_6 \end{bmatrix} = \text{velocity of satellite relative to rotating atmosphere}$$

B = ballistic coefficient of the satellite

λ = atmospheric density, modeled as $\lambda = \lambda_o e^{-\beta h}$

λ_o = mean sea level atmospheric density (1.376229×10^{-6} g/m^3)

β = altitude atmospheric density decay rate (1.395×10^{-1}/m)

$h = (x_1^2 + x_2^2 + x_3^2)^{1/2} - R_o$ = height above mean earth radius

R_o = mean earth radius (6.37817×10^6 m)

w_{ie} = angular rotation rate of the earth

While the ballistic coefficient of the vehicle is generally not known, it is known that for a nonthrusting vehicle it will not change significantly during the time of a tracking pass (an attitude maneuver could affect it by changing the surface area along the velocity vector). Hence, it is assumed in this study that the ballistic coefficient can be adequately modeled as a random constant or bias (a random variable that has 100% correlation in time)

$$\dot{B} = 0 \tag{72}$$

with an initial condition as a Gaussian random variable.

Tracker State Equations

While the target state equations are straightforward and represent a commonly used model for satellite dynamics, the tracker state

dynamics and measurement equations are very dependent upon the modeling assumptions made in this study. Therefore, a full development of the tracker dynamic state equations and then the tracking system measurement equations will be given in this section.

The geometry of the tracker is shown in Figure 3.

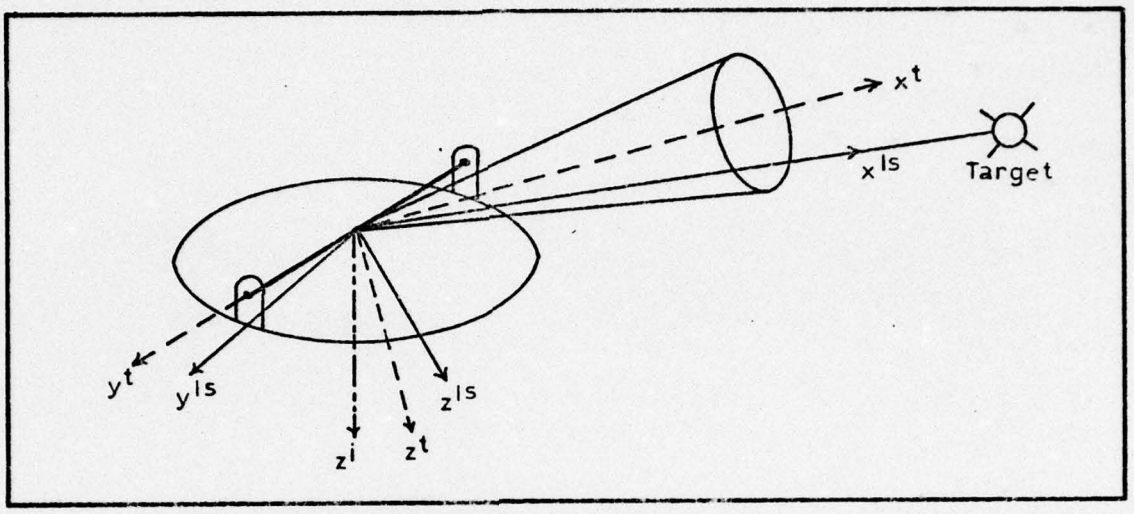


Figure 3. Tracker and Line-of-Sight Geometry

It is assumed that the tracker base is inertially stabilized such that the tracker elevation axis, y^t, always lies in the inertial X-Y plane. The x^t axis is along the boresight of the tracker and the z^t axis completes a right-hand orthogonal system. Assuming that the tracker base is inertially stabilized as above, the two Euler angle rotations needed to go from the inertial to the tracker frame are dependent only upon the relative position vector from the tracker to the target, expressed in inertial coordinates. The first Euler angle rotation is by an angle θ about the inertial z axis as shown in Figure 4. The superscript "a" indicates an intermediate frame and the transformation matrix $\underline{C}_i^a$ is

$$\underline{C}_i^a = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{73}$$

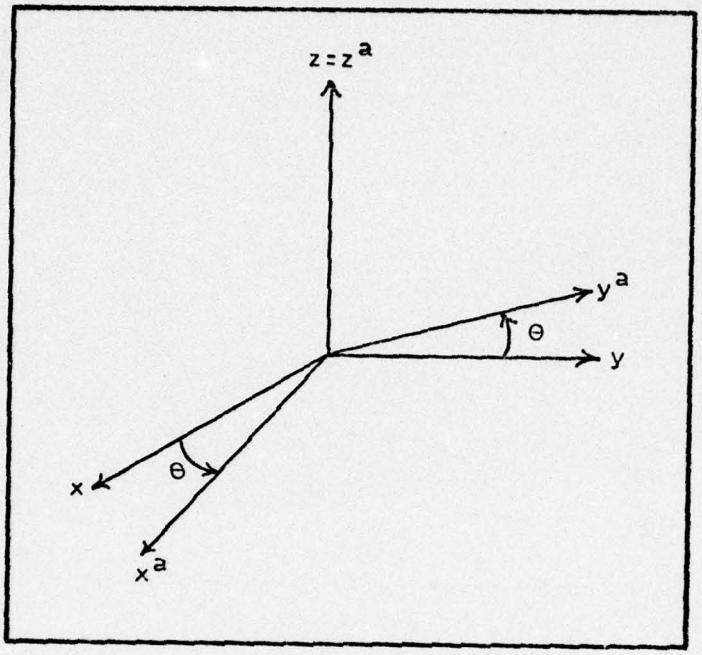


Figure 4. First Euler Angle Rotation

Because of the constraint on the y^t axis, the next Euler angle rotation is by an angle ϕ about the y^a axis - leaving the y^t axis in the inertial xy plane. A view of this rotation is shown in Figure 5. The transformation matrix between the intermediate "a" frame and the tracker "t" frame is

$$\underline{C}_a^t = \begin{bmatrix} \cos\phi & 0 & -\sin\phi \\ 0 & 1 & 0 \\ \sin\phi & 0 & \cos\phi \end{bmatrix} \tag{74}$$

Therefore, the Euler angle transformation from the inertial to the tracker t frame is give by $\underline{C}_i^t$ as is shown in Figure 6.

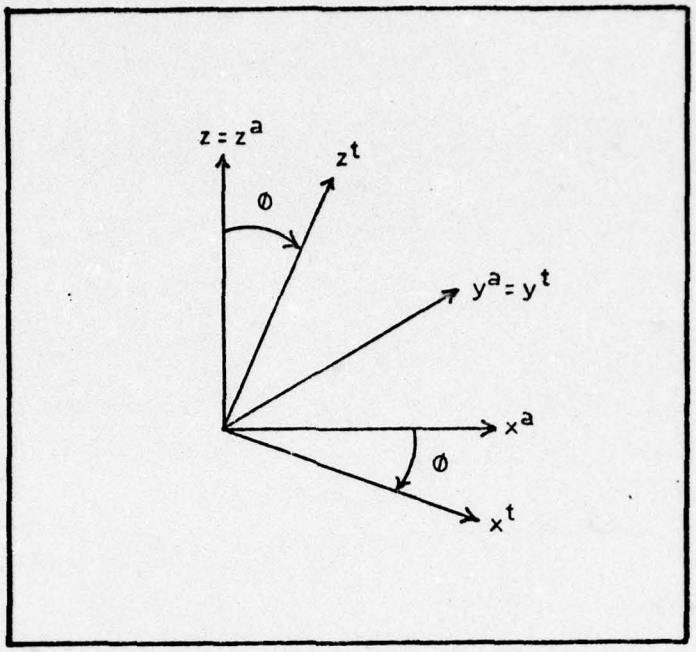


Figure 5. Second Euler Angle Rotation

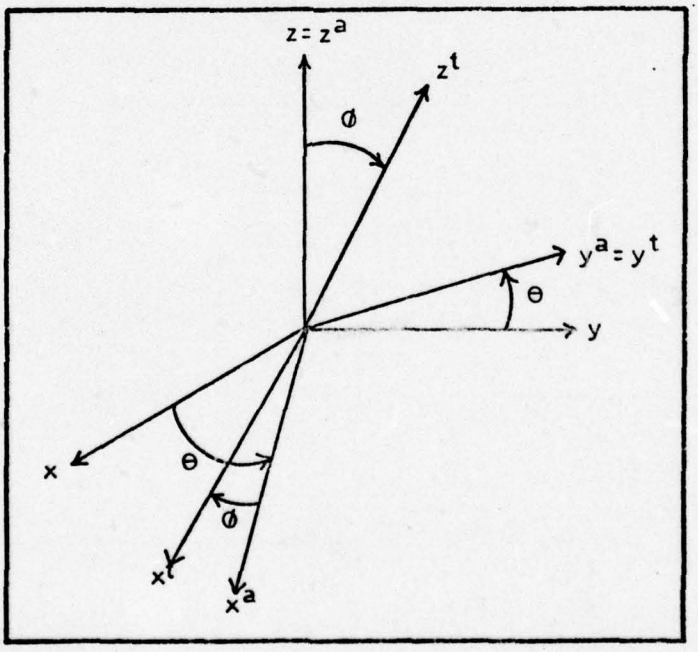


Figure 6. Inertial and Tracker Frame Orientations

$$\underline{C}_i^t = \underline{C}_a^t \, \underline{C}_i^a = \begin{bmatrix} \cos\theta\cos\phi & \cos\phi\sin\theta & -\sin\phi \\ -\sin\theta & \cos\theta & 0 \\ \cos\theta\sin\phi & \sin\phi\sin\theta & \cos\phi \end{bmatrix} \tag{75}$$

The line-of-sight (LS) coordinate frame is widely used in pointing and tracking problems. The LS and t frames have the same origin; however, the LS frame has one axis pointing exactly at the target while the t frame is misaligned from this line-of-sight. This study assumes that the LS x-axis points directly at the target as was shown in Figure 2. For perfect tracking, the LS and t frames are aligned, i.e. $\underline{C}_t^{LS} = \underline{I}$. Let $(\underline{r}_{ts})^i$ denote the position vector of the satellite with respect to the tracker expressed in the inertial frame:

$$(\underline{r}_{ts})^i \triangleq \begin{bmatrix} r_{tsx}^i \\ r_{tsy}^i \\ r_{tsz}^i \end{bmatrix} \tag{76}$$

Transforming this vector to the LS frame yields:

$$(\underline{r}_{ts})^{LS} = \underline{C}_i^{LS} (\underline{r}_{ts})^i \tag{77}$$

However, by definition of the LS frame $(\underline{r}_{ts})^{LS} \triangleq \begin{bmatrix} R \\ 0 \\ 0 \end{bmatrix}$ where R is the range between the target and the tracker. Remembering that $\underline{C}_i^{LS} \approx \underline{C}_i^t$ (for perfect tracking $\underline{C}_i^{LS} = \underline{C}_i^t$)

$$\begin{bmatrix} R \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \cos\theta\cos\phi & \sin\theta\cos\phi & -\sin\phi \\ -\sin\theta & \cos\theta & 0 \\ \cos\theta\sin\phi & \sin\theta\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} r_{tsx}^i \\ r_{tsy}^i \\ r_{tsz}^i \end{bmatrix} \tag{78}$$

then

$$-r_{tsx}^i \sin\theta + r_{tsy}^i \cos\theta = 0 \tag{79}$$

so

$$\tan\theta = \frac{r^i_{tsy}}{r^i_{tsx}} \tag{80}$$

or

$$\theta = \begin{cases} \tan^{-1}\dfrac{r^i_{tsy}}{r^i_{tsx}} & r^i_{tsx} > 0 \\ \tan^{-1}\dfrac{r^i_{tsy}}{r^i_{tsx}} + \pi & r^i_{tsx} < 0 \end{cases} \tag{81}$$

Also,

$$r^i_{tsx}\cos\theta\sin\phi + r^i_{tsy}\sin\theta\sin\phi + r^i_{tsz}\cos\phi = 0 \tag{82}$$

which implies, after some algebraic manipulation, that

$$\tan\phi = \frac{-r^i_{tsz}}{(r^{i2}_{tsx} + r^{i2}_{tsy})^{1/2}} \tag{83}$$

and therefore,

$$\phi = \begin{cases} \cos^{-1}\dfrac{(r^{i2}_{tsx} + r^{i2}_{tsy})^{1/2}}{(r^{i2}_{tsx} + r^{i2}_{tsy} + r^{i2}_{tsz})^{1/2}} & r^i_{tsz} > 0 \\ -\phi & r^i_{tsz} < 0 \end{cases} \tag{84}$$

This relationship can be seen in Figure 7.

In practice, perfect tracking in which the tracker x axis aligns perfectly with the LS x-axis will not be possible. The misalignment between the tracker and LS frames can be defined in terms of two Euler angle rotations. In a manner entirely similar to the previous derivation, the Euler angle rotations are $\delta\nu$ about the z^t axis and $\delta\epsilon$ about an intermediate ($y^a = y^{LS}$) axis as shown in Figure 8 and,

$$\underline{C}^{LS}_t = \begin{bmatrix} \cos\delta\nu\cos\delta\epsilon & \sin\delta\nu\cos\delta\epsilon & -\sin\delta\epsilon \\ -\sin\delta\nu & \cos\delta\nu & 0 \\ \cos\delta\nu\sin\epsilon & \sin\delta\nu\sin\delta\epsilon & \cos\delta\epsilon \end{bmatrix} \tag{85}$$

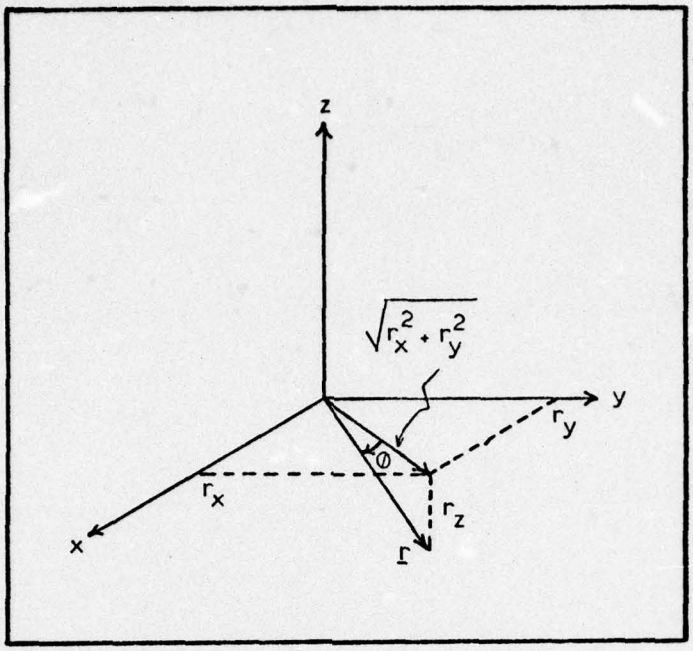


Figure 7. Relationship of ϕ to $\underline{r}$

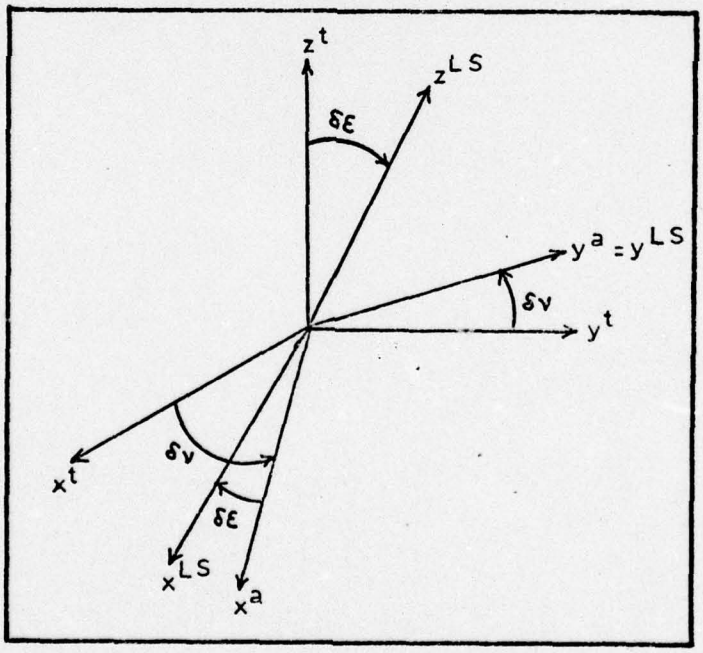


Figure 8. Tracker and Line-of-Sight Frame Orientations

It is reasonable to assume that $\delta\epsilon$ and $\delta\nu$ are sufficiently small, even at the beginning of the track, so the small angle approximations can be used:

$$\sin\delta\epsilon \approx \delta\epsilon \qquad \sin\delta\nu \approx \delta\nu$$

$$\cos\delta\epsilon \approx 1 \qquad \cos\delta\nu \approx 1$$

In which case, Equation (85) becomes

$$\underline{C}_t^{LS} = \begin{bmatrix} 1 & \delta\nu & -\delta\epsilon \\ -\delta\nu & 1 & 0 \\ \delta\epsilon & 0 & 1 \end{bmatrix} \tag{86}$$

A solution for the time rate of change of $\delta\epsilon$ and $\delta\nu$ is now sought. Since $\delta\epsilon$ and $\delta\nu$ are both small angles, then the time rate of change of these angles ($\dot{\delta\epsilon}$ and $\dot{\delta\nu}$) are nothing more than the y and z axis components, respectively, of the angular velocity of the LS frame with respect to the tracker frame, coordinatized in the LS coordinates. Hence,

$$\overset{LS}{W}_{tLS} = \begin{bmatrix} w_{tLSx}^{LS} \\ \dot{\delta\epsilon} \\ \dot{\delta\nu} \end{bmatrix} \tag{87}$$

But

$$\underline{W}_{tLS}^{LS} = \underline{W}_{iLS}^{LS} - \underline{W}_{it}^{LS} \tag{88}$$

$$= \underline{W}_{iLS}^{LS} - \underline{C}_t^{LS}\ \underline{W}_{it}^{t} \tag{89}$$

$$= \begin{bmatrix} w^{LS}_{iLSx} \\ w^{LS}_{iLSy} \\ w^{LS}_{iLSz} \end{bmatrix} - \begin{bmatrix} 1 & \delta\nu & -\delta\epsilon \\ -\delta\nu & 1 & 0 \\ \delta & 0 & 1 \end{bmatrix} \begin{bmatrix} w^{t}_{itx} \\ w^{t}_{ity} \\ w^{t}_{itz} \end{bmatrix} \tag{90}$$

$$= \begin{bmatrix} w^{LS}_{iLSx} - w^{t}_{itx} - \delta\nu\, w^{t}_{ity} + \delta\epsilon\, w^{t}_{itz} \\ w^{LS}_{iLSy} - w^{t}_{ity} + \delta\nu\, w^{t}_{itx} \\ w^{LS}_{iLSz} - w^{t}_{itz} - \delta\epsilon\, w^{t}_{itx} \end{bmatrix} \tag{91}$$

So since $w^{LS}_{tLSx} = 0$, it is evident that

$$\dot{\delta\epsilon} = w^{LS}_{iLSy} - w^{t}_{ity} + \delta\nu\, w^{t}_{itx} \tag{92}$$

$$\dot{\delta\nu} = w^{LS}_{iLSz} - w^{t}_{itz} - \delta\epsilon\, w^{t}_{itx} \tag{93}$$

$$w^{LS}_{iLSx} = w^{t}_{itx} + \delta\nu\, w^{t}_{ity} - \delta\epsilon\, w^{t}_{itz} \tag{94}$$

Equations (92) and (93) describe the time propagation of the error misalignment angles $\delta\epsilon$ and $\delta\nu$. Equation (94) will prove useful in the following development of the time evolution of the line-of-sight angular velocity vector $\underline{w}^{LS}_{iLS}$.

In order to determine expressions for the time rate of change of the line-of-sight angular velocity vector, consider the position vector of the satellite with respect to the aircraft, $\underline{r}_{ts}$. Differentiating $\underline{r}_{ts}$ twice with respect to time and applying the Coriolis Theorem each time yields

$$\left.\frac{d^2\underline{r}_{ts}}{dt^2}\right|_i = \left.\frac{d^2\underline{r}_{ts}}{dt^2}\right|_{LS} + 2\,\underline{w}_{iLS} \times \left.\frac{d\underline{r}_{ts}}{dt}\right|_{LS} + \left.\frac{d\underline{w}_{iLS}}{dt}\right|_{LS} \times \underline{r}_{ts} + \underline{w}_{iLS} \times (\underline{w}_{iLS} \times \underline{r}_{ts}) \tag{95}$$

where the vertical bar indicates the frame in which the differentiation takes place. Coordinatizing Equation (95) in the LS frame and defining the acceleration of the satellite relative to the tracker along the line-of-sight x, y, and z axes as

$$\left(\left.\frac{d^2\underline{r}_{ts}}{dt^2}\right|_i\right)^{LS} \triangleq \begin{bmatrix} a_{tsx}^{LS} \\ a_{tsy}^{LS} \\ a_{tsz}^{LS} \end{bmatrix} \triangleq (\underline{a}_{ts})^{LS} = (\underline{a}_t)^{LS} - (\underline{a}_s)^{LS} \tag{96}$$

and

$$(\underline{r}_{ts})^{LS} \triangleq \begin{bmatrix} R \\ 0 \\ 0 \end{bmatrix} \tag{97}$$

$$\left(\left.\frac{d\underline{r}_{st}}{dt}\right|_{LS}\right)^{LS} \triangleq \begin{bmatrix} V_r \\ 0 \\ 0 \end{bmatrix} \tag{98}$$

where $V_r \triangleq \dot{R}$ = range rate, the following is then obtained

$$\begin{bmatrix} a_{tsx}^{LS} \\ a_{tsy}^{LS} \\ a_{tsz}^{LS} \end{bmatrix} = \begin{bmatrix} \dot{V}_r \\ 0 \\ 0 \end{bmatrix} + 2\begin{bmatrix} 0 \\ V_r w_{iLSz}^{LS} \\ -V_r w_{iLSy}^{LS} \end{bmatrix} + \begin{bmatrix} 0 \\ R\dot{w}_{iLSz}^{LS} \\ -R\dot{w}_{iLSy}^{LS} \end{bmatrix} + \begin{bmatrix} -R(w_{iLSy}^{LS2} + w_{iLSz}^{LS2}) \\ R w_{iLSx}^{LS} w_{iLSy}^{LS} \\ R w_{iLSx}^{LS} w_{iLSz}^{LS} \end{bmatrix} \tag{99}$$

Examination of Equation (99) yields the following scalar state equations

$$\dot{w}^{LS}_{iLSy} = -\frac{a^{LS}_{tsz}}{R} - \frac{2\,V_r w^{LS}_{iLSy}}{R} + w^{LS}_{iLSx}\, w^{LS}_{iLSz} \tag{100}$$

$$\dot{w}^{LS}_{iLSz} = \frac{a^{LS}_{tsy}}{R} - \frac{2\,V_r w^{LS}_{iLSz}}{R} - w^{LS}_{iLSx}\, w^{LS}_{iLSy} \tag{101}$$

$$\dot{R} = V_r$$

$$\dot{V}_r = a^{LS}_{tsx} + R(w^{LS^2}_{iLSy} + w^{LS^2}_{iLSz}) \tag{102}$$

Defining $(\underline{a}_{ts})^t$ as the inertial acceleration of the satellite relative to the tracker, coordinatized in the tracker frame, Equation (86) can be used to obtain

$$(\underline{a}_{ts})^{LS} = \underline{C}^{LS}_t (\underline{a}_{ts})^t = \underline{C}^{LS}_t \begin{bmatrix} a^t_{tsx} \\ a^t_{tsy} \\ a^t_{tsz} \end{bmatrix} \tag{103}$$

thus, it follows that

$$a^{LS}_{tsx} = a^t_{tsx} + \delta\nu\, a^t_{tsy} - \delta\epsilon\, a^t_{tsz} \tag{104}$$

$$a^{LS}_{tsy} = a^t_{tsy} - \delta\nu\, a^t_{tsx} \tag{105}$$

$$a^{LS}_{tsz} = a^t_{tsz} + \delta\epsilon\, a^t_{tsx} \tag{106}$$

Equations (100) through (102) now become

$$\dot{w}^{LS}_{iLSy} = -\frac{a^t_{tsz} - \delta\epsilon\, a^t_{tsx}}{R} - \frac{2\,V_r w^{LS}_{iLSy}}{R} + w^{LS}_{iLSx}\, w^{LS}_{iLSy} \tag{107}$$

$$\dot{w}^{LS}_{iLSz} = \frac{a^t_{tsy} - \delta\nu\, a^t_{tsx}}{R} - \frac{2\, V_r w^{LS}_{iLSz}}{R} - w^{LS}_{iLSx}\, w^{LS}_{iLSy} \qquad (108)$$

$$\dot{V}_r = a^t_{tsx} + \delta\nu\, a^t_{tsy} - \delta\epsilon\, a^t_{tsz} + R(w^{LS^2}_{iLSy} + w^{LS^2}_{iLSz}) \qquad (109)$$

Using Equation (94) to eliminate w^{LS}_{iLSx} from Equations (107) and (108) yields

$$\dot{w}^{LS}_{iLSy} = -\frac{a^t_{tsz}}{R} - \frac{2\, V_r w^{LS}_{iLSy}}{R} + w^{LS}_{iLSz}\, w^t_{itx} + \{-\frac{\delta\epsilon\, a^t_{tsx}}{R} + w^{LS}_{iLSz}\,[\delta\nu\, w^t_{ity} - \delta\epsilon\, w^t_{itz}]\} \qquad (110)$$

$$\dot{w}^{LS}_{iLSz} = \frac{a^t_{tsy}}{R} - \frac{2\, V_r w^{LS}_{iLSz}}{R} - w^{LS}_{iLSy}\, w^t_{itx} + \{-\frac{\delta\nu\, a^t_{tsx}}{R} - w^{LS}_{iLSy}\,[\delta\nu\, w^t_{ity} - \delta\epsilon\, w^t_{itz}]\} \qquad (111)$$

The bracketed $\{\cdot\}$ terms in Equations (110) and (111) represent effects due to the misalignment between the tracker and LS frames. For the case of perfect tracking ($\delta\epsilon = \delta\nu = 0$) these terms equal zero.

Measurement Equations

The tracking system has the capability of measuring the inertial angular velocity of the tracker, the two angular deviations $\delta\epsilon$ and $\delta\nu$, and the range between the tracker and the target. The models used in the measurement equations are developed in References 13, 14, and 15.

The inertial angular velocity of the tracker is measured in the tracker coordinate frame by three rate gyros, one mounted along each tracker axis. While the system (truth) model propagates the true line-of-sight angular velocities w^{LS}_{iLSy} and w^{LS}_{iLSz} [Equations (110) and (111)],

measurements are available only in the tracker frame of w^t_{itx}, w^t_{ity}, and w^t_{itz}. Therefore, measurements of w^t_{ity} and w^t_{itz} are considered by the filter to be pseudo-measurements of w^{LS}_{iLSy} and w^{LS}_{iLSz}.

The dominant effects which contribute to errors in the rate gyro measurements are scale factor errors, drift errors, g-sensitive mass unbalance errors, misalignment errors, and white noise, v_1. A suggested gyro rate measurement model (Ref 13:300) is given as (tracker x-axis only)

$$w^t_{mx} = w^t_{itx} + B_{gsfx}\, w^t_{itx} + \sum_{i=1}^{3} B_{gmx_i}\, a_i + C_{gx} + \left[\Delta\underline{C}_{gma}\, \underline{w}^t_{it}\right]_x + v_1 \qquad (112)$$

where

w^t_{mx} = measured angular velocity along x_t axis

w^t_{itx} = true angular velocity along x_t axis

B_{gsfx} = constant bias gyro scale factor

B_{gmx_i} = coefficients (along x, y, z directions in tracker frame) of the g-sensitive mass unbalance to which the gyro is subject

a_i = accelerations (a^t_{tsx}, a^t_{tsy}, a^t_{tsz}) of the tracker origin with respect to inertial space

C_{gx} = gyro drift rate along x_t axis

$\Delta\underline{C}_{gma}$ = the error angle transformation matrix resulting from the misalignments of the three gyros

$$\Delta \underline{C}_{gma} \triangleq \begin{bmatrix} 0 & -B_{gma_{12}} & B_{gma_{13}} \\ B_{gma_{21}} & 0 & -B_{gma_{23}} \\ -B_{gma_{31}} & B_{gma_{32}} & 0 \end{bmatrix}$$

$B_{gma_{ij}}$ = gyro misalignment error angles between the desired gyro coordinates and the actual gyro coordinates (mounting errors)

$[\cdot]_i$, i = x, y, z = i^{th} component of the vector $[\cdot]$

v_1 = additive zero mean white Gaussian noise to account for unmodeled effects such as aniso-elastic drift, quantization error, etc.

Suppose a gyro has been studied in the laboratory. Results of the gyro testing indicate that the gyro has a drift component along the x-axis with a steady state standard deviation of σ radians per second and a process correlation time of τ seconds. These statistics characterize an exponentially time-correlated (first order Markov) process, modeled as the output of a first order lag feedback network driven by a zero-mean white Gaussian noise of strength $Q = 2\sigma^2/\tau$, (Ref 3:4-88), as shown in Figure 9. The state equation for C_{gx} is

$$\dot{C}_{g_x} = -\beta_4 C_{g_x} + w_7$$

The autocorrelation function of C_{g_x} is

$$E[C_{g_x}(t)\, C_{g_x}(t+\tau)] = \sigma^2\, e^{-\beta|\tau|} \qquad (113)$$

The remaining coefficients in the rate gyro measurement equation are modeled as random biases, (Ref 10:4-82), as shown in Figure 10.

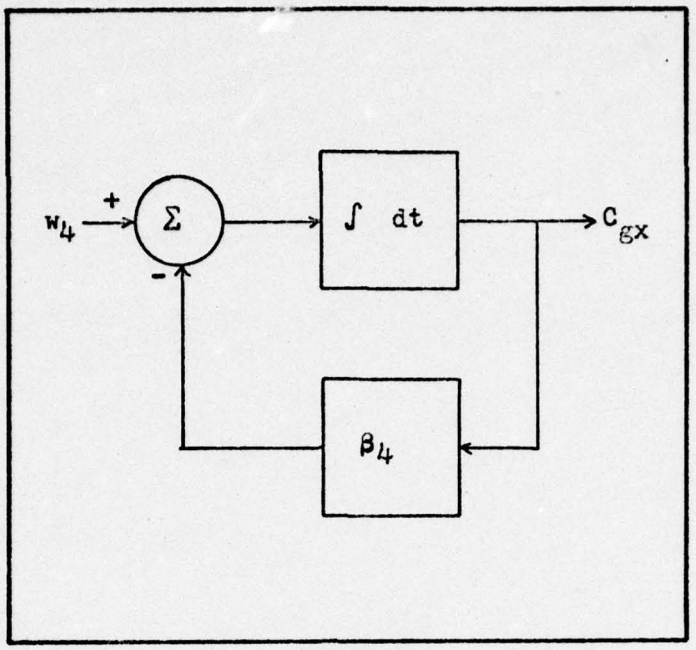


Figure 9. Gyro Drift Model

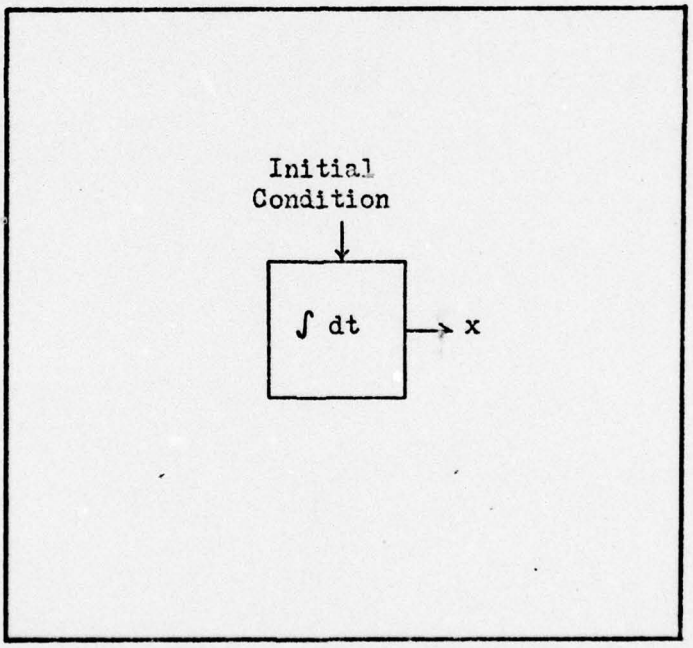


Figure 10. Random Bias Model

With this model, the filter is "told" that the value of the variable does not change in time, although you do not know its value apriori. This represents a reasonable model for the remaining coefficients because, while they may not be constant on a long term basis, they will remain essentially constant during a ten minute tracking pass. The general form of the state equation for these coefficients is $\dot{x} = 0$. The equation which describes the way the covariance propagates in time is

$$\dot{P} = 0 \tag{114}$$

This indicates that the variance of the coefficient does not change in time and that the initial condition on P represents the variance of the coefficient about its mean.

The measurements of the tracker angular velocity about the tracker y and z axes are modeled in a manner identical to w_{mx}. The values used for the standard deviations in the process correlation time in the gyro rate measurement model are representative of a typical aircraft rate gyro (Ref 13:302):

Quantity	Steady State Standard Deviation (σ)	Process Correlation Time (τ)
Gyro drift	1×10^{-6} rad/sec	3600 sec
Gyro scale factors	5×10^{-4}	∞
Gyro mass unbalance coefficients	3×10^{-6} rad-sec/m	∞
Gyro misalignment coefficients	1×10^{-4}	∞
Additive white Gaussian noise v_1	1×10^{-9} rad/sec	0

The error misalignment angles $\delta\epsilon$ and $\delta\nu$ are measured in the tracking coordinate frames. Effects which can degrade these measurements are: deterministic scale factors, scale factor errors, angle track biases, and angle track scintillation noises. No attempt has been made to model noises that are specific to a typical radar or laser ranger. Rather, the measurement model proposed below is representative of a large class of measurement devices (Ref 14:14)

$$\delta\epsilon_m = \delta\epsilon + S_\epsilon + C_{SF_\epsilon}\delta\epsilon + B_{AT_\epsilon} + v_4 \tag{115}$$

$$\delta\nu_m = \delta\nu + S_\nu + C_{SF_\nu}\delta\nu + B_{AT_\nu} + v_5 \tag{116}$$

where

δ_ϵ, δ_ν = true misalignment angles

S_ϵ, S_ν = angle track scintillation noises

C_{SF_ϵ}, C_{SF_ν} = scale factor errors

B_{AT_ϵ}, B_{AT_ν} = angle track biases

v_4, v_5 = zero mean white Gaussian noises to account for unmodeled effects

Both the angle track scintillation noises (S_ϵ and S_ν) and the scale factor errors (C_{SF_ϵ}, C_{SF_ν}) are modeled as exponentially time correlated random variables. Scintillation noise is dependent upon various factors such as atmospheric propagation, and amplifier characteristics which change as a function of time during a typical tracking pass. Scale factor errors are a function of tracker variables that undergo a change with respect to time (Ref 14:15). These variables are reasonably well modeled as exponentially time-correlated and the following equations are obtained:

$$\dot{S}_\epsilon = -\beta_4 S_\epsilon + w_4 \tag{117}$$

$$\dot{S}_\nu = -\beta_5 S_\nu + w_5 \tag{118}$$

$$\dot{C}_{SF_\epsilon} = -\beta_{10} C_{SF_\epsilon} + w_{10} \tag{119}$$

$$\dot{C}_{SF_\nu} = -\beta_{11} C_{SF_\nu} + w_{11} \tag{120}$$

where w_4, w_5, w_{10}, and w_{11} are zero mean white Gaussian noises with $Q_i = 2\sigma_i^2/\tau_i$. The β_i represent the inverse of the process correlation time, $\beta_i = 1/\tau_i$. The angle track bias coefficients B_{AT_ϵ} and B_{AT_ν} are modeled as random biases - initial value unknown but describable as a Gaussian random variable with mean zero and a known variance. The values used for the process standard deviations and correlation times are given below (Ref 4:149).

Quantity	Steady State Standard Deviation (σ)	Process Correlation Time (τ)
Angle track scintillations (S_ϵ,S_ν)	$1\text{x}10^{-6}$ rad	10 sec
Angle measurement scale factor errors (C_{SF_ϵ},C_{SF_ν})	10^{-4}	300 sec
Angle track bias (B_{AT_ϵ},B_{AT_ν})	$2\text{x}10^{-6}$ rad	∞
Additive white noise	$1\text{x}10^{-6}$ rad	0

The model of the measurement of range is very similar to those of the angular deviations. Uncertainties in the measurement of range are due primarily to scintillation noise and bias errors.

$$R_m = R + S_R + B_R + v_6 \tag{121}$$

where

S_R = range scintillation noise

R = true range

B_R = range bias

v_6 = zero mean white Gaussian noise to account for unmodeled effects

The range scintillation noise is due to atmospheric effects and errors in the digitization of the returned signal. The atmospheric effects in particular are a direct function of the elevation angle of the tracker - less scintillation error when the satellite is directly "overhead" and greater errors at the horizon. The scintillation error will show a degree of time correlation during a tracking pass and an exponentially time correlated random variable is used to model this state:

$$\dot{S}_R = -\beta_3 S_R + w_6 \tag{122}$$

where w_6 is a zero mean white Gaussian noise with $Q = 2\sigma^2/\tau$ and $\beta_3 = 1/\tau_3$. The range bias is modeled as a random bias. (Initial value unknown, but describable as a zero mean Gaussian random variable and known variance.) Values used for the standard deviations and correlation times are shown below (Ref 4:150).

Quantity	Steady State Standard Deviation (σ)	Process Correlation Time (τ)
Range Scintillation	20m	10 sec
Range bias	5m	∞
Additive white noise	5m	0

Summary of State and Measurement Equations

After augmenting the satellite and tracker state equations with the noise states needed to define the measurements, the truth model contains a total of 42 states. They are repeated here for clarity.

State Equations

$$\dot{x}_1 = x_4 \tag{123}$$

$$\dot{x}_2 = x_5 \tag{124}$$

$$\dot{x}_3 = x_6 \tag{125}$$

(1), (2), (3): Satellite inertial position

$$\dot{x}_4 = a_{gx} + a_{mx} + a_{sx} + a_{dx} + w_1 \tag{126}$$

$$\dot{x}_5 = a_{gy} + a_{my} + a_{sy} + a_{dy} + w_2 \tag{127}$$

$$\dot{x}_6 = a_{gz} + a_{mz} + a_{sz} + a_{dz} + w_3 \tag{128}$$

(4), (5), (6): Satellite inertial velocity

$$\dot{w}_{LSy} \triangleq \dot{w}^{LS}_{iLSy} = -\frac{a^t_{tsz}}{R} - \frac{2\,V_r w^{LS}_{iLSy}}{R} + w^{LS}_{iLSz}\, w^t_{itx} + \{-\frac{\delta\epsilon\, a^t_{tsx}}{R} + w^{LS}_{iLSz}\,[\delta\nu\, w^t_{ity} - \delta\epsilon\, w^t_{itz}]\} \tag{129}$$

$$\dot{w}_{LSz} \triangleq \dot{w}^{LS}_{iLSz} = \frac{a^t_{tsy}}{R} - \frac{2\,V_r w^{LS}_{iLSz}}{R} - w^{LS}_{iLSy}\, w^t_{itx} + \{-\frac{\delta\nu\, a^t_{tsx}}{R} - w^{LS}_{iLSy}\,[\delta\nu\, w^t_{ity} - \delta\epsilon\, w^t_{itz}]\} \tag{130}$$

(7), (8): Line-of-sight angular velocity

$$\dot{\delta\epsilon} = w^{LS}_{iLSy} - w^t_{ity} + \delta\nu\, w^t_{itx} \tag{131}$$

$$\dot{\delta\nu} = w^{LS}_{iLSz} - w^t_{itz} - \delta\epsilon\, w^t_{itx} \tag{132}$$

(9), (10): Error misalignment angle

$$\dot{R} = V_r \tag{133}$$

(11): Range

$$\dot{V}_r = a^t_{tsx} + \delta\nu\, a^t_{tsy} - \delta\epsilon\, a^t_{tsz} + R(w^{LS^2}_{iLSy} + w^{LS^2}_{iLSz}) \tag{134}$$

(12): Range rate

$$\dot{x}_{13} = 0 \tag{135}$$

(13): Satellite ballistic coefficient

$$\dot{s}_\epsilon = -\beta_1 s_\epsilon + w_4 \tag{136}$$

$$\dot{s}_\nu = -\beta_2 s_\nu + w_5 \tag{137}$$

(14), (15): Angle track scintillation

(16) $\dot{S}_R = -\beta_3 S_R + w_6$ — Range scintillation (138)

(17) $\dot{C}_{gx} = -\beta_4 C_{gx} + w_7$ (139)

(18) $\dot{C}_{gy} = -\beta_5 C_{gy} + w_8$ — Gyro drift (140)

(19) $\dot{C}_{gz} = -\beta_6 C_{gz} + w_9$ (141)

(20) $\dot{C}_{SF_\epsilon} = -\beta_7 C_{SF_\epsilon} + w_{10}$ (142)

(21) $\dot{C}_{SF_\nu} = -\beta_8 C_{SF_\nu} + w_{11}$ — Angle measurement scale factors (143)

(22) $\dot{B}_{gmx_1} = 0$ (144)

. . .

(30) $\dot{B}_{gmz_3} = 0$ — Coefficients of gyro mass unbalance (nine equations) (145)

(31) $\dot{B}_{gma_{12}} = 0$ (146)

(32) $\dot{B}_{gma_{13}} = 0$ (147)

(33) $\dot{B}_{gma_{21}} = 0$ (148)

(34) $\dot{B}_{gma_{23}} = 0$ — Gyro misalignment coefficients (six equations) (149)

(35) $\dot{B}_{gma_{31}} = 0$ (150)

(36) $\dot{B}_{gma_{32}} = 0$ (151)

(37) $\dot{B}_R = 0$ — Range bias (152)

(38) $\dot{B}_{AT_\epsilon} = 0$ (153)

(39) $\dot{B}_{AT_\nu} = 0$ — Angle track bias (154)

(40) $\dot{B}_{gsfx} = 0$ (155)

(41) $\dot{B}_{gsfy} = 0$ — Gyro scale factors (156)

(42) $\dot{B}_{gsfz} = 0$ (157)

Measurement Equations

$$(0)\quad w^t_{mx} = w^t_{itx} + B_{gsfx}\, w^t_{itx} + \sum_{i=1}^{3} B_{gmx_i}\, a_i + C_{gx} + \left[\Delta \underline{C}_{gma}\, \underline{w}^t_{it}\right]_x + v_1 \tag{158}$$

$$(1)\quad w^t_{my} = w^t_{ity} + B_{gsfy}\, w^t_{ity} + \sum_{i=1}^{3} B_{gmy_i}\, a_i + C_{gy} + \left[\Delta \underline{C}_{gma}\, \underline{w}^t_{it}\right]_y + v_2 \tag{159}$$

$$(2)\quad w^t_{mz} = w^t_{itz} + B_{gsfz}\, w^t_{itz} + \sum_{i=1}^{3} B_{gmz_i}\, a_i + C_{gz} + \left[\Delta \underline{C}_{gma}\, \underline{w}^t_{it}\right]_z + v_3 \tag{160}$$

$$(3)\quad \delta\epsilon_m = \delta\epsilon + S_\epsilon + C_{SF_\epsilon}\delta\epsilon + B_{AT_\epsilon} + v_4 \tag{161}$$

$$(4)\quad \delta\nu_m = \delta\nu + S_\nu + C_{SF_\nu}\delta\nu + B_{AT_\nu} + v_4 \tag{162}$$

$$(5)\quad R_m = R + S_R + B_R + v_6 \tag{163}$$

V. A Sub-Optimal Filter Model

Introduction

This chapter presents the development of a reduced order system model which is used as the suboptimal filter model for the generalized Monte Carlo Analysis Program. The reduced order system model should be computationally simpler than the full system (truth) model. How this simplification takes place is dictated by the prior experience, skill, and decisions of the designer. In many cases, simplification is accomplished by deleting states from the truth model that represent non-dominant effects in the problem under study. Typically this is accomplished with a corresponding increase in noise strengths driving the model to compensate for the deleted terms or states. As an example, if time correlated gyro drift rates are replaced by an additive white noise in a navigation filter, the filter performance may be significantly degraded. However, the deletion of these states may not be noticed for a tracking problem of ten minutes duration. In addition to deleting states, the designer might also choose to delete terms in the state equation that are about an order of magnitude less than the size of the other terms. So in addition to deleting non-dominant states, non-dominant terms are also deleted. However, careful judgement must be exercised when non-dominant terms are deleted because these small cross coupling terms may be extremely important in meeting system performance criteria.

The reduced order filter (a six-state filter) was suggested by the Air Force Avionics Laboratory to Mann (Ref 5). The main idea behind the filter is to use what is known about the dominant forces acting on the satellite being tracked and its geometry to aid in the simplification of the truth model.

Six State Filter Equation Development

The underlying concept for the development of the filter model proposed in this section is to delete the satellite inertial position and velocity states [x(1) through x(6)]. Other information already available in the remaining six states and INS data will then be used to determine the acceleration of the satellite relative to the tracker (a^t_{ts}) based upon the knowledge that the dominant acceleration of the satellite can be described by a two body point mass gravity model. This approach to the satellite tracking problem was suggested by Air Force Avionics Laboratory personnel.

Figure 11 represents a typical tracker line-of-sight to satellite geometry. Where

(x^i, y^i, z^i) is the geocentric equatorial inertial frame

(x^{LS}, y^{LS}, z^{LS}) is the tracker line-of-sight frame, with x^{LS} pointing at the satellite

$\underline{r}_{ot}$ is the vector from the center of the earth to the tracker

$\underline{r}_{os}$ is the vector from the center of the earth to the satellite

$\underline{r}_{ts}$ is the vector from the origin of the tracker system to the satellite along the line-of-sight x-axis

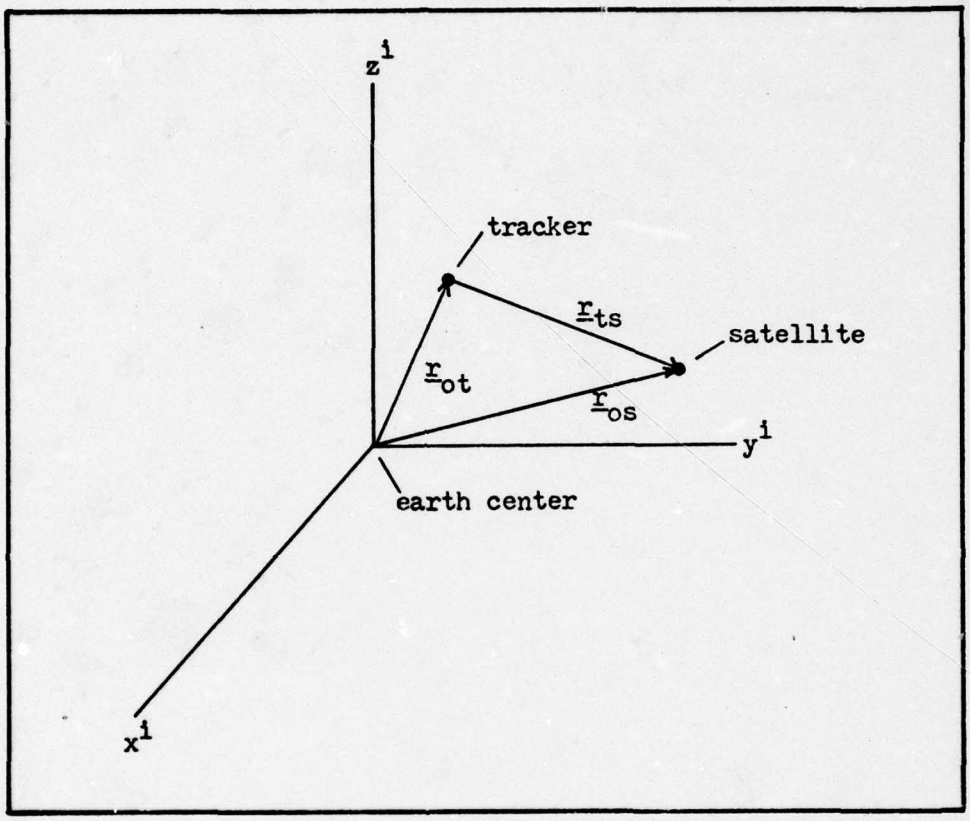


Figure 11. Tracker Line-of-Sight to Satellite Geometry

When the satellite inertial position and velocity states are deleted from the system, the following information remains available to formulate a filter model:

θ, ϕ precise resolver measurements of the tracker azimuth and elevation angles

$\underline{C}_i^{LS}(\theta,\phi)$ the transformation matrix from the inertial to the line-of-sight frames as a function of θ and ϕ

$(\underline{a}_{it})^t$ high precision measurements of the tracker acceleration with respect to inertial space in tracker coordinates, determined from the outputs of three accelerometers - one mounted along each of the tracker axes.

R_m, w_{my}^t, w_{mz}^t, $\delta\epsilon_m$, $\delta\nu_m$ noise corrupted measurements of range, tracker angular rates, and angular deviations as discussed in Chapter IV.

Consider now the tracker line-of-sight angular velocity state equations.

$$\dot{w}_{iLSy}^{LS} = -\frac{a_{tsz}^t}{R} - \frac{2\,V_r w_{iLSy}^{LS}}{R} + w_{iLSz}^{LS}\, w_{itx}^t + \left\{-\frac{\delta\epsilon\, a_{tsx}^t}{R} + w_{iLSz}^{LS}\left[\delta\nu\, w_{ity}^t - \delta\epsilon\, w_{itz}^t\right]\right\} \tag{164}$$

$$\dot{w}_{iLSz}^{LS} = \frac{a_{tsy}^t}{R} - \frac{2\,V_r w_{iLSz}^{LS}}{R} - w_{iLSz}^{LS}\, w_{itx}^t + \left\{-\frac{\delta\nu\, a_{tsx}^t}{R} - w_{iLSy}^{LS}\left[\delta\nu\, w_{ity}^t - \delta\epsilon\, w_{itz}^t\right]\right\} \tag{165}$$

The bracketted terms {·} result from the fact that the tracker and line-of-sight frames are not coincident and differ by the two small Euler angles $\delta\epsilon$ and $\delta\nu$. For high accuracy tracking, the small angular deviations $\delta\epsilon$ and $\delta\nu$ will have magnitudes on the order of 10^{-5} radians or less (Ref 10:67). For the particular tracking profile used in this study, the bracketted terms are on the order of 10^{-11} radians/sec^2 while the smallest values of $\dot{w}^{LS}_{iLSy}$ and $\dot{w}^{LS}_{iLSz}$ are $\approx 10^{-6}$ radians/sec^2. Thus, in the reduced order filter model the bracketted terms are neglected and replaced by a zero mean white Gaussian driving noise to account for the increased uncertainty in the dynamics model:

$$\dot{w}^{LS}_{iLSy} = -\frac{a^t_{tsz}}{R} - \frac{2\,V_r w^{LS}_{iLSy}}{R} + w^{LS}_{iLSz}\, w^t_{itx} + w_1 \tag{166}$$

$$\dot{w}^{LS}_{iLSz} = \frac{a^t_{tsy}}{R} - \frac{2\,V_r w^{LS}_{iLSz}}{R} - w^{LS}_{iLSy}\, w^t_{itx} + w_2 \tag{167}$$

before the filter was tuned to give the best performance, w_1 and w_2 were assigned one-sigma values of 10^{-11} radians/sec^2, based upon the values of the bracketted terms that were dropped.

The remaining state equations are:

$$\dot{\delta\epsilon} = w^{LS}_{iLSy} - w^t_{ity} + \delta\nu\, w^t_{itx} \tag{168}$$

$$\dot{\delta\nu} = w^{LS}_{iLSz} - w^t_{itz} - \delta\epsilon\, w^t_{itx} \tag{169}$$

$$\dot{R} = V_r \tag{170}$$

$$\dot{V}_r = a^t_{tsx} + \delta\nu\, a^t_{tsy} - \delta\epsilon\, a^t_{tsz} + R(w^{LS^2}_{iLSy} + w^{LS^2}_{iLSz}) \tag{171}$$

Equation (171) may be simplified using the same criteria as for Equations (164, 165). For high accuracy tracking, the terms containing $\delta\epsilon$ and $\delta\nu$ will be approximately five orders of magnitude smaller than

$\dot{V}_r$. Therefore, in the proposed filter model, these two terms are dropped and replaced by a zero mean white Gaussian driving noise (w_3) of strength equal to (1×10^{-13} m/sec^2);

$$\dot{V}_r = a^t_{tsx} + R(w^{LS^2}_{iLSy} + w^{LS}_{iLSz}) + w_3 \tag{172}$$

Consider the following development of the acceleration of the satellite with respect to the tracker $(\underline{a}_{ts})^{LS}$ expressed in the line-of-sight frame.

$$(\underline{r}_s)^i = (\underline{r}_t)^i + (\underline{r}_{ts})^i \tag{173}$$

where the superscript "i" again indicates coordinatization in the inertial frame. From the definition of the line-of-sight (LS) frame (the line-of-sight x axis - x^{LS} - points exactly at the target), we know that the position vector of the satellite relative to the tracker $(\underline{r}_{ts})^{LS}$ lies along x^{LS}.

$$(\underline{r}_{ts})^{LS} \triangleq \begin{bmatrix} R \\ 0 \\ 0 \end{bmatrix} \tag{174}$$

where R is the range between the tracker origin and the satellite. It follows that

$$(\underline{r}_{ts})^{LS} = \underline{C}^{LS}_i (\underline{r}_{ts})^i \tag{175}$$

From Equation (173) it is seen that the acceleration of the satellite with respect to the tracker expressed in inertial coordinates $(\underline{r}_{ts})^i$ is

$$(\ddot{\underline{r}}_{ts})^i = (\ddot{\underline{r}}_s)^i - (\ddot{\underline{r}}_t)^i \tag{176}$$

Elementary astrodynamics tells us that we can model the inertial acceleration of the satellite as

$$(\underline{\ddot{r}}_s)^i = \frac{-\mu_e (\underline{r}_s)^2}{\left|(\underline{r}_s)^i\right|^3} \tag{177}$$

where μ_e is the earth's gravitational constant and Equation (176) becomes

$$(\underline{\ddot{r}}_{ts})^i = \frac{-\mu_e (\underline{r}_s)^i}{\left|(\underline{r}_s)^i\right|^3} - (\underline{\ddot{r}}_t)^i \tag{178}$$

where from Equations (173) and (175)

$$(\underline{r}_s)^i = (\underline{r}_t)^i + \underline{C}^i_{LS}\,(\underline{r}_{ts})^{LS} \tag{179}$$

It is evident at this point that if the tracker inertial position $(\underline{r}_t)^i$ were made available from an INS on the aircraft, then the acceleration of the satellite relative to the tracker expressed in LS coordinates [which is $(\underline{a}_{ts})^{LS}$] can be found by

$$(\underline{a}_{ts})^{LS} \triangleq (\underline{\ddot{r}}_{ts})^{LS} = \underline{C}^{LS}_i\,(\underline{\ddot{r}}_{ts})^i \tag{180}$$

$$= \underline{C}^{LS}_i \left[\frac{-\mu_e (\underline{r}_s)^i}{\left|(\underline{r}_s)^i\right|^3} - (\underline{\ddot{r}}_t)^i\right] \tag{181}$$

$$= \underline{C}^{LS}_i \left\{ -\mu_e \left[\frac{(r_t)^i + \underline{C}^i_{LS}(\underline{r}_{ts})^{LS}}{(\underline{r}_s)^{i\,3}}\right] - (\underline{\ddot{r}}_t)^i \right\} \tag{182}$$

$$= \frac{-\mu_e\, \underline{C}^{LS}_i (\underline{r}_t)^i + (\underline{r}_{ts})^{LS}}{(\underline{r}_s)^{i\,3}} - (\underline{\ddot{r}}_t)^{LS} \tag{183}$$

Under the assumption that $\delta\epsilon$ and $\delta\nu$ are small, the tracker and line-of-sight frames are nearly aligned. It follows that the acceleration vector of the satellite relative to the tracker coordinatized in the tracker frame - $(\underline{a}_{ts})^t$ - is well approximated by $(\underline{a}_{ts})^{LS}$. Also, the acceleration of the tracker with respect to inertial space expressed in the LS frame - $(\underline{r}_t)^{LS}$ - is well approximated by the inertial

acceleration of the tracker in the tracker frame $(\underline{r}_t)^t$ which is derived from the outputs of the accelerometers. Equation (183) can now be approximated as

$$(\underline{a}_{ts})^t = \frac{-\mu_e \left[\underline{C}_i^{LS}(\underline{r}_t)^i + (\underline{r}_{ts})^{LS}\right]}{(r_s)^{i\,3}} - (\ddot{\underline{r}}_t)^t \qquad (184)$$

Equation (184) is readily implementable as all quantities in it are available:

$\underline{C}_i^{LS}$ is a known function of the resolver angles ϕ and θ

$(\underline{r}_t)^i$ is provided by the INS

$(\underline{r}_{ts})^{LS} = \begin{bmatrix} R \\ 0 \\ 0 \end{bmatrix}$ where R is the range

$(\ddot{\underline{r}}_t)^t$ is derived from the output of the accelerometers

$$(\underline{r}_s)^t = (\underline{r}_t)^i + \underline{C}_{LS}^i(\underline{r}_{ts})^{LS} \qquad (185)$$

Therefore, the form of the state equations for $\dot{w}_{iLSy}^{LS}$, $\dot{w}_{iLSz}^{LS}$, and $\dot{V}_r$ remain as given in Equations (166), (167), and (172) with the components of $(\underline{a}_{ts})^t$

$$(\underline{a}_{ts})^t \triangleq \begin{bmatrix} a_{tsx}^t \\ a_{tsy}^t \\ a_{tsz}^t \end{bmatrix} \qquad (186)$$

determined by the corresponding components of Equation (184).

Filter Measurement Equation Development

The measurement equations for the truth model are summarized at the end of Chapter IV. Each measurement is considered to be the sum of the state (which is a realization of one stochastic process), and additive noises (which are realizations or samples of other stochastic

processes). For instance, consider the measurement of the inertial angular velocity of the tracker along the tracker y axis

$$w^t_{my} = w^t_{ity} + B_{gsfy}\, w^t_{ity} + \sum_{i=1}^{3} B_{gmy_i}\, a_i + C_{gy} + \left[\Delta \underline{C}_{gma}\, \underline{w}^t_{it}\right]_y + v_1 \qquad (187)$$

where $\underline{w}^t_{it} \triangleq \left[w^t_{itx}\ w^t_{ity}\ w^t_{itz}\right]^T$ represents the true angular velocity of the tracker coordinatized in the tracker frame (it should be recalled that w_{my} and w_{mz} are considered to be pseudomeasurements of the line-of sight angular velocities w^{LS}_{iLSy} and w^{LS}_{iLSz} which are not directly measurable). The second through fifth terms in Equation (187) are stochastic models of the dominant noise processes that corrupt a rate gyro measurement. The last term, v_1, is a zero mean white Gaussian noise sequence added to account for errors in the modeling assumptions and unmodeled higher order effects. B_{gsfy}, B_{gmy_1}, B_{gmy_2}, B_{gmy_3} and the elements of the matrix $\Delta \underline{C}_{gma}$ are all modeled as random biases. For lack of better information they are modeled as zero mean with a variance determined from empirical data. Each of these stochastic processes is then multiplied by a known quantity (in the truth model) - the resulting product in each case being a random process. The inertial acceleration of the tracker origin in the tracker coordinates

$$(\underline{a}_t)^t \triangleq \begin{bmatrix} a^t_{tx} \\ a^t_{ty} \\ a^t_{tz} \end{bmatrix} \qquad (188)$$

is assumed to be a deterministic system parameter in this study. The component of the gyro drift along the tracker y axis, C_{gy}, is modeled as an exponentially time-correlated random process. In the filter

measurement model, it is assumed that the total effect of all of the corruptive effects in each of the truth model measurement equations can be replaced by a single zero-mean white Gaussian noise sequence. The filter measurement model, for each state, consists of the "true value" plus an additive white noise to account for modeling uncertainties. For the state we chose as an example,

$$w^t_{my} = w^t_{ity} + v_1 \tag{189}$$

This approach to modeling the measurements leads to the simplest filter implementation. If the performance should prove to be poor using this model, the variance of v_1 could be increased to indicate additional uncertainty in the assumed measurement model. If performance remains poor, this would be an indication that some of the effects appearing in the truth model measurement equations must be added - at the expense of a higher dimensioned filter - to the filter measurement equations.

Summary of Filter State and Measurement Equations

The state and measurement equations for the reduced order filter model are summarized below. The development of the linearized dynamics and measurement matrices $\underline{F}$ and $\underline{H}$ for the filter is given in Appendix B.

State Equations

$$(1)\quad \dot{w}_{LSy} \triangleq \dot{w}^{LS}_{iLSy} = -\frac{a^t_{tsz}}{R} - \frac{2\,V_r w^{LS}_{iLSy}}{R} + w^{LS}_{iLSz}\, w^t_{itx} + w_1 \tag{190}$$

$$(2)\quad \dot{w}_{LSz} \triangleq \dot{w}^{LS}_{iLSz} = \frac{a^t_{tsy}}{R} - \frac{2\,V_r w^{LS}_{iLSz}}{R} - w^{LS}_{iLSy}\, w^t_{itx} + w_2 \tag{191}$$

$$(3)\quad \dot{\delta\epsilon} = w^{LS}_{iLSy} - w^t_{ity} + \delta v\, w^t_{itx} \tag{192}$$

(4) $\delta\dot{\nu} = w^{LS}_{iLSz} - w^{t}_{itz} - \delta\epsilon\, w^{t}_{itx}$ (193)

(5) $\dot{R} = V_r$ (194)

(6) $\dot{V}_r = a^{t}_{tsx} + R(w^{LS^2}_{iLSy} + w^{LS}_{iLSz}) + w_3$ (195)

Measurement Equations

(1) $w^{t}_{my} = w^{t}_{ity} + v_1$ (196)

(2) $w^{t}_{mz} = w^{t}_{itz} + v_2$ (197)

(3) $\delta\nu_m = \delta\nu + v_3$ (198)

(4) $\delta\epsilon_m = \delta\epsilon + v_4$ (199)

(5) $R_m = R + v_5$ (200)

VI. Results and Discussion

Introduction

A generalized Monte Carlo Analysis Program (MCAP) has been developed and used in the analysis of an extended Kalman filter (for a users' guide and program description, see Appendix A). The purpose of this chapter is to present the results of this feasibility study using MCAP in evaluating the six-state extended Kalman filter and subsequently to discuss and interpret these results. Prior to presenting the results, the tracking profile used in the Monte Carlo analysis and the philosophy used to tune the filter will be presented, since they have direct bearing on the performance achieved in this study.

At the initialization of the tracking profile, the aircraft (tracker) lies on the Greenwich meridian, at a geocentric latitude of 30° north. For the duration of the 200 second tracking pass, the tracker moves at a constant velocity of 135 meters/sec (300 miles/hr) eastward while maintaining a constant altitude of 9000 meters and the same geocentric latitude. The satellite is in a 2.0×10^{5} meter circular, near polar orbit. The satellite is a relatively small vehicle, with a ballistic coefficient of 0.015 and a solar pressure coefficient equal to that of a vehicle with a projected surface area towards the sun of 10 meter^{2}. Initially, the satellite lies essentially on the prime meridian and is approaching a descending node (descending towards the equator), as shown in Figure 12. Pertinent tracking information for this flight profile is summarized in Table I. The angles θ and ϕ,

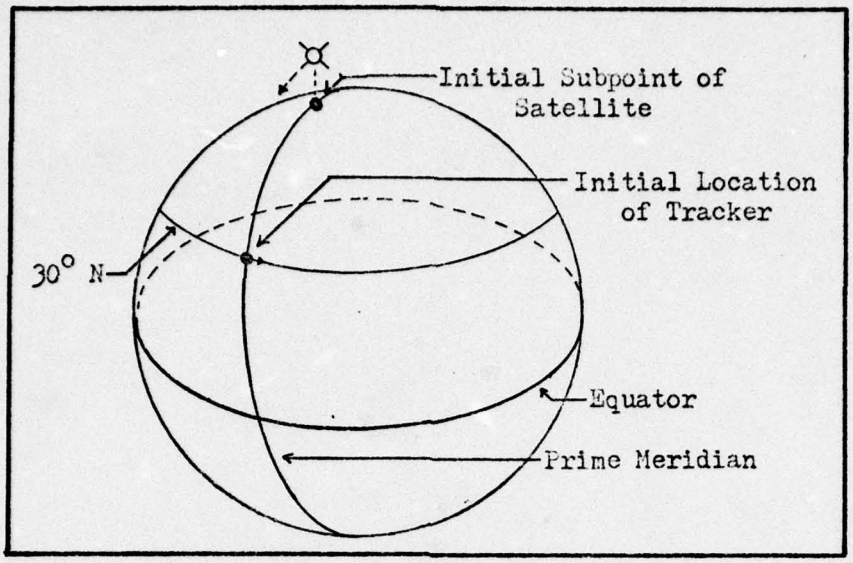


Figure 12. Satellite/Tracker Geometry

Table I

Typical Tracking Parameters

Parameter	Initial Value (t=0)	Final Value (t=200)
Elevation Angle of Tracker ($-\phi$)	56.6°	74.5°
Azimuth Angle of Tracker (θ)	180.0°	221.0°
w_{LSy}	-9.58×10^{-4} Rad/Sec	-4.98×10^{-3} Rad/Sec
w_{LSz}	2.57×10^{-4} Rad/Sec	2.47×10^{-3} Rad/Sec
Range (R)	2.10×10^{6} Meters	6.13×10^{5} Meters
Range Rate (V_r)	-7.50×10^{3} μ/Sec	-6.98×10^{3} μ/Sec

which define the coordinate transformation from the inertial to the line-of-sight frame (see Chapter IV), may also be used to describe the tracker azimuth and elevation angles. The term $-\phi$ is the elevation angle and θ the azimuth angle (0^{o} azimuth being defined as the condition in which the x_{LS} axis is parallel to the inertial x-axis). Figures 13 and 14 present the time history of ϕ and θ.

The truth model time history of states w_{LSy} and w_{LSz} [states x(1) and x(2)] are given in Figures 15 and 16. Note that the maximum tracker angular velocity is reached at the end of the tracking pass. Since the range is decreasing, this condition is expected from the geometry of this tracking profile. As the vehicle moves, the tracker traverses a trajectory confined to a plane perpendicular to the satellite orbital plane. Therefore, it is expected that the inertial angular velocity about the elevation axis (w_{LSy}) would continue to increase until the satellite approaches its zenith with respect to the tracker. In addition, the inertial angular velocity about the azimuth axis (w_{LSz}) is expected to continually increase until the satellite lies in the plane of the tracker path (an azimuth angle of 270^{o}). Thus, while the tracking profile used in this study did not represent worst case conditions, it does present a highly nonlinear angular rate history with which to evaluate the estimation accuracy of the Kalman filter.

The truth model time history of states R[x(5) = range] and V_r[x(6) = range rate] are shown in Figures 17 and 18. Note that the range plot is nearly linear (over the time period of interest) and the

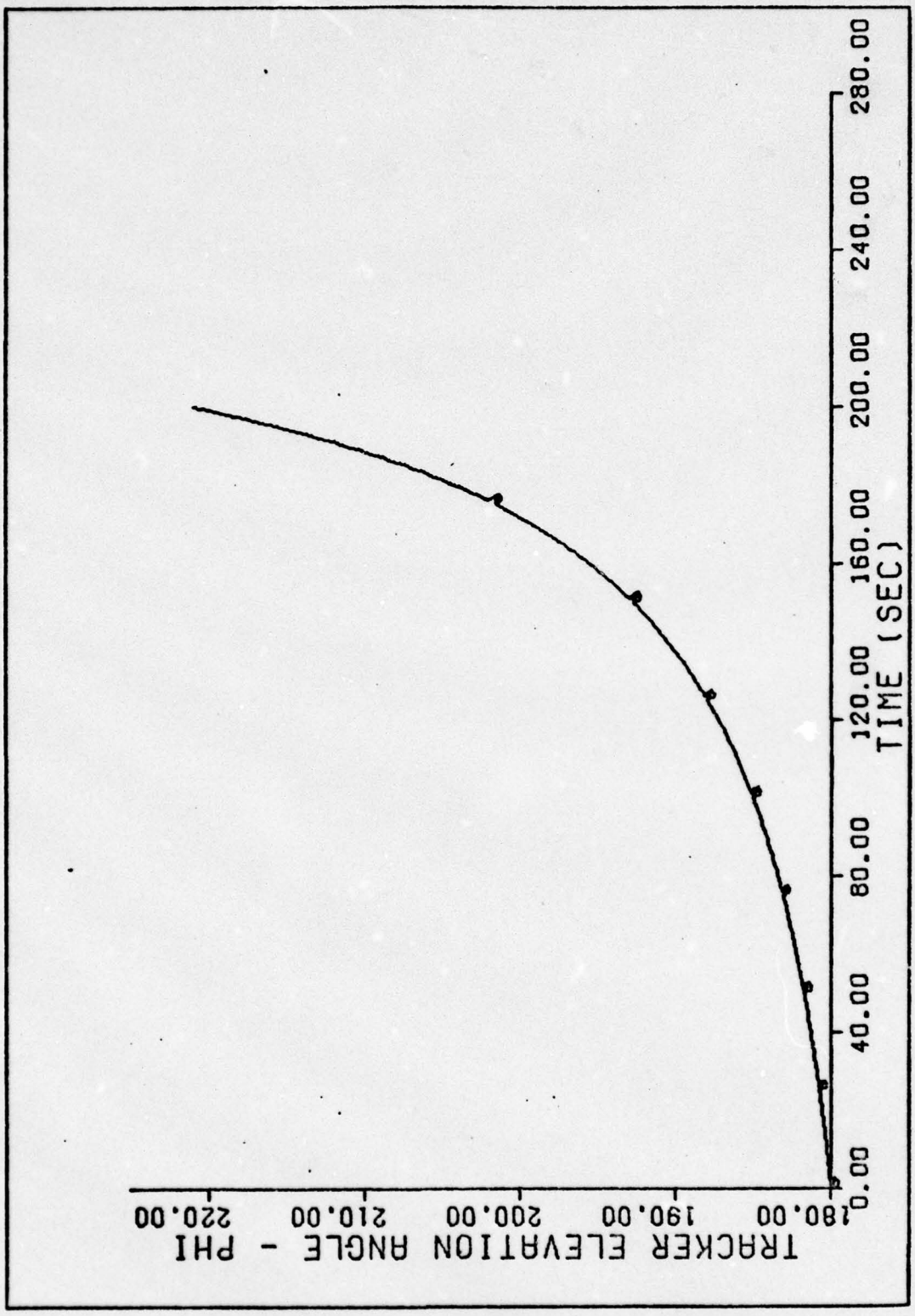


Figure 13. Tracker Elevation Angle Time History

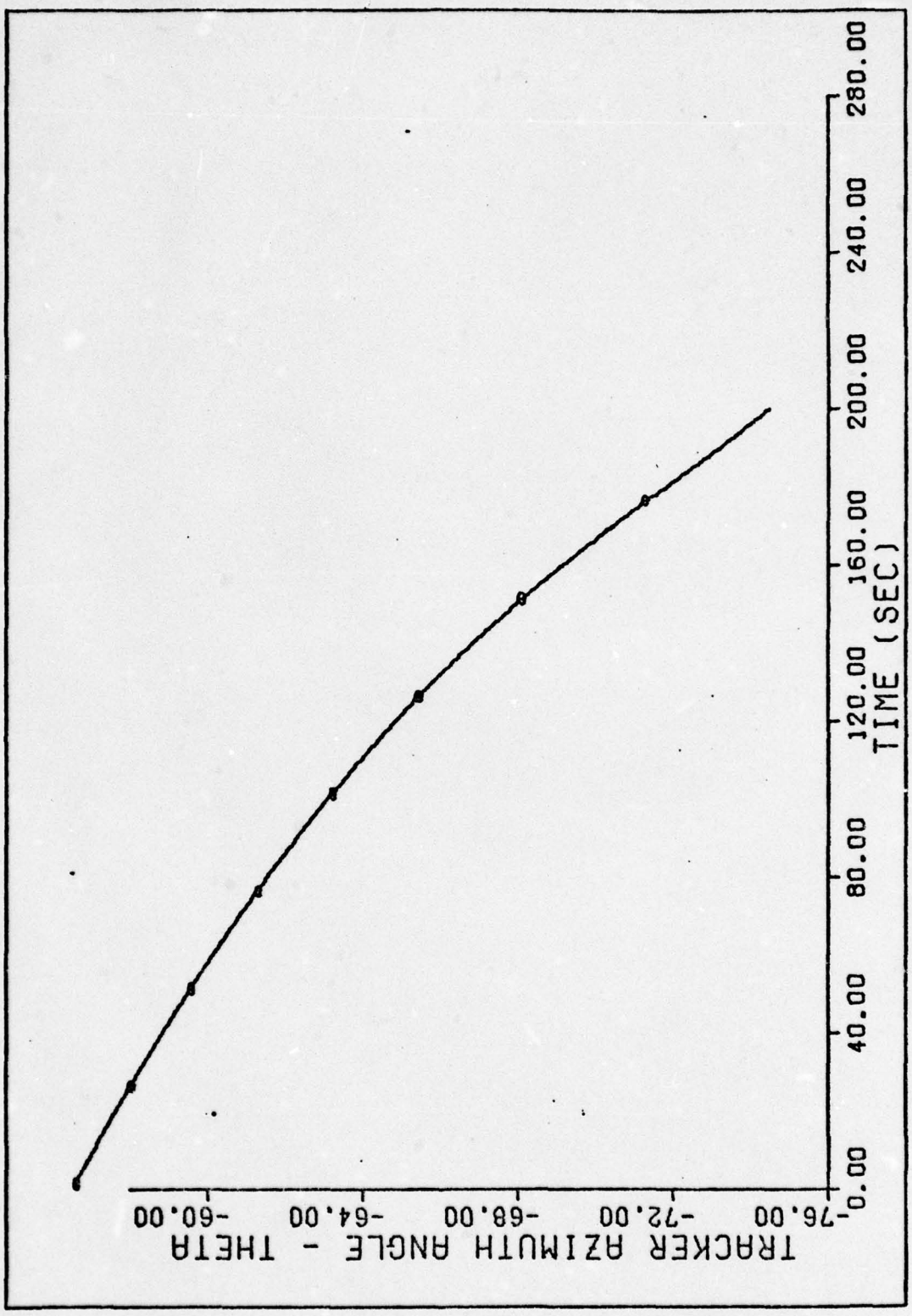


Figure 14. Tracker Azimuth Angle Time History

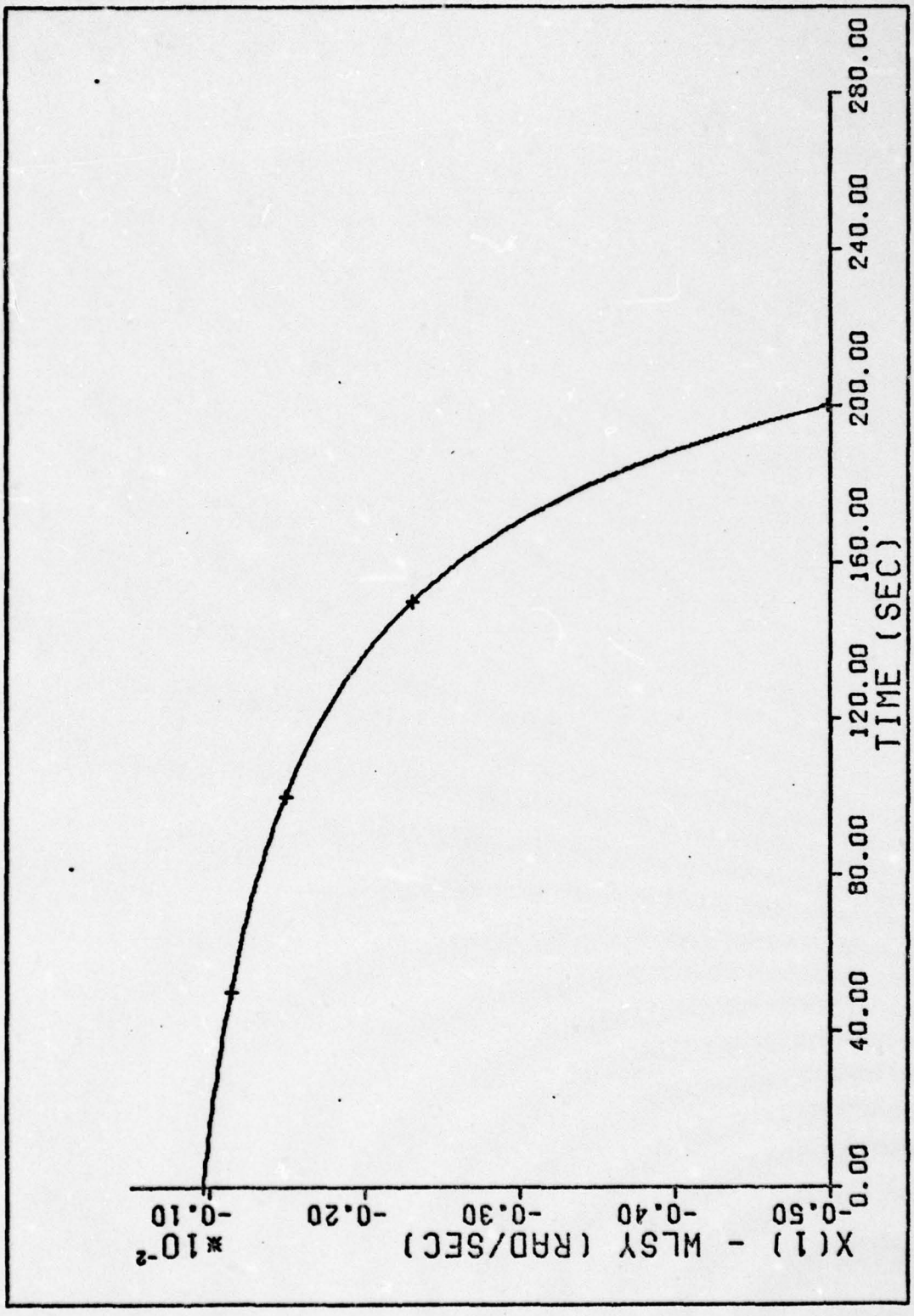


Figure 15. x(1) State Time History

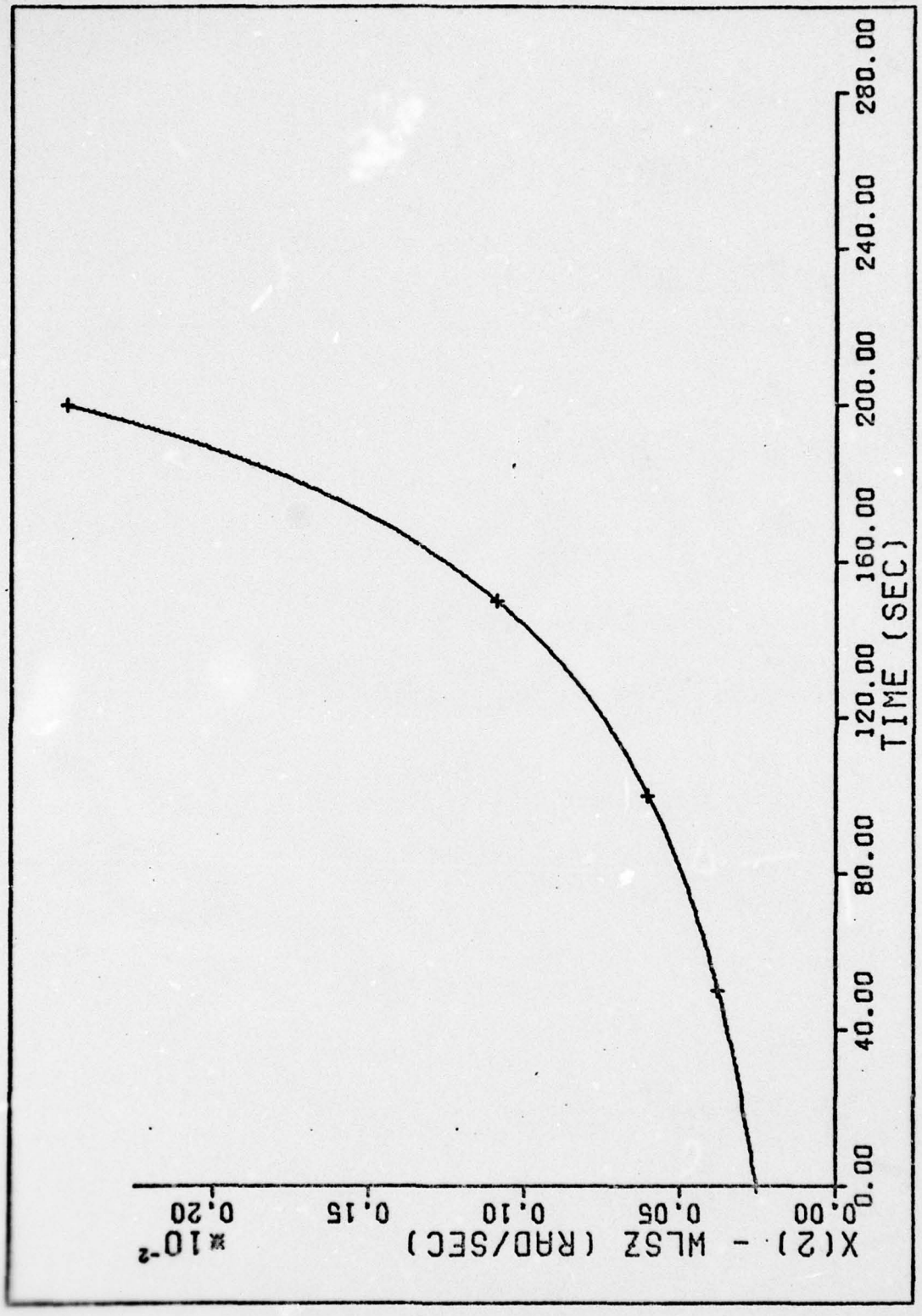


Figure 16. x(2) State Time History

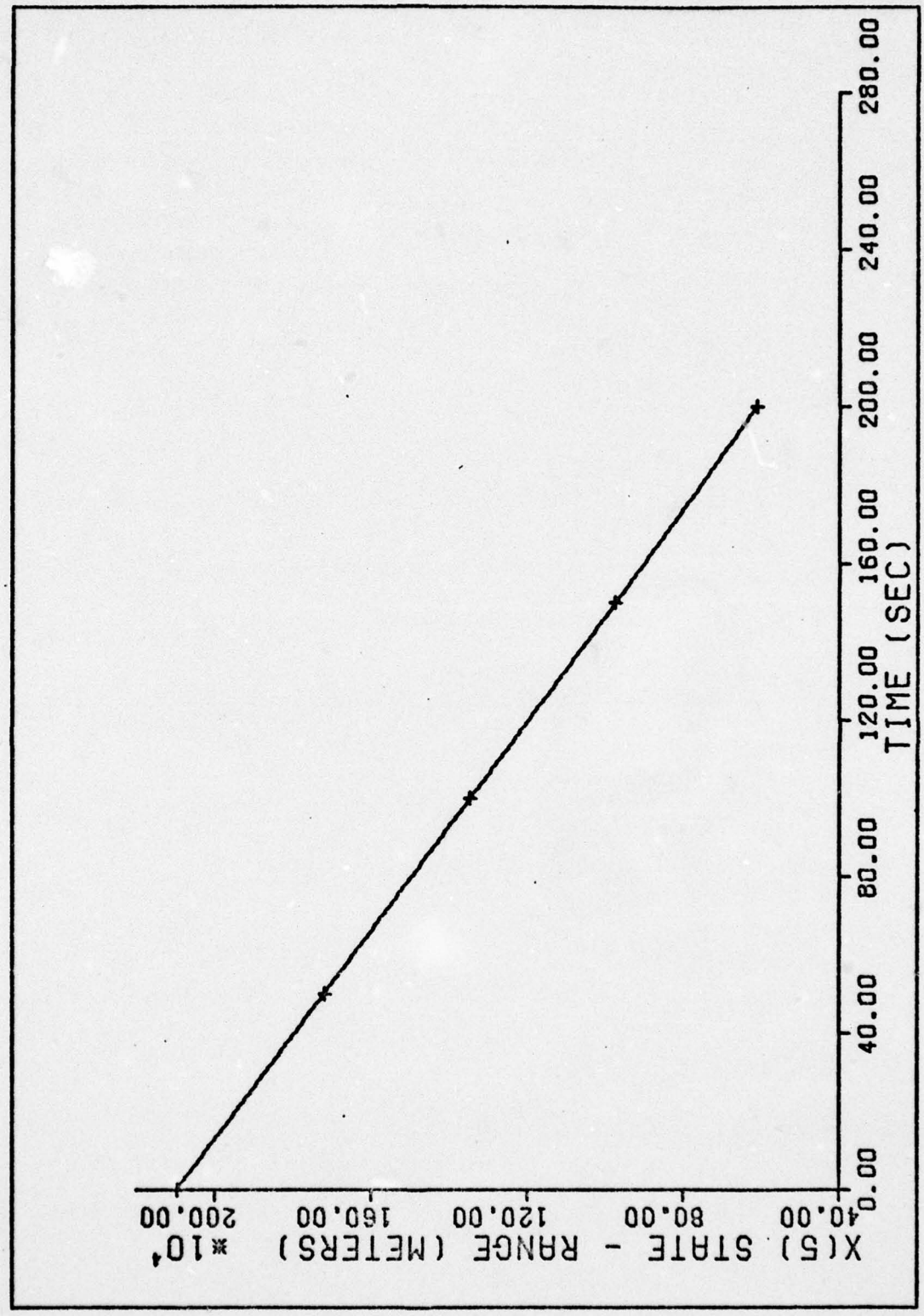


Figure 17. x(5) State Time History

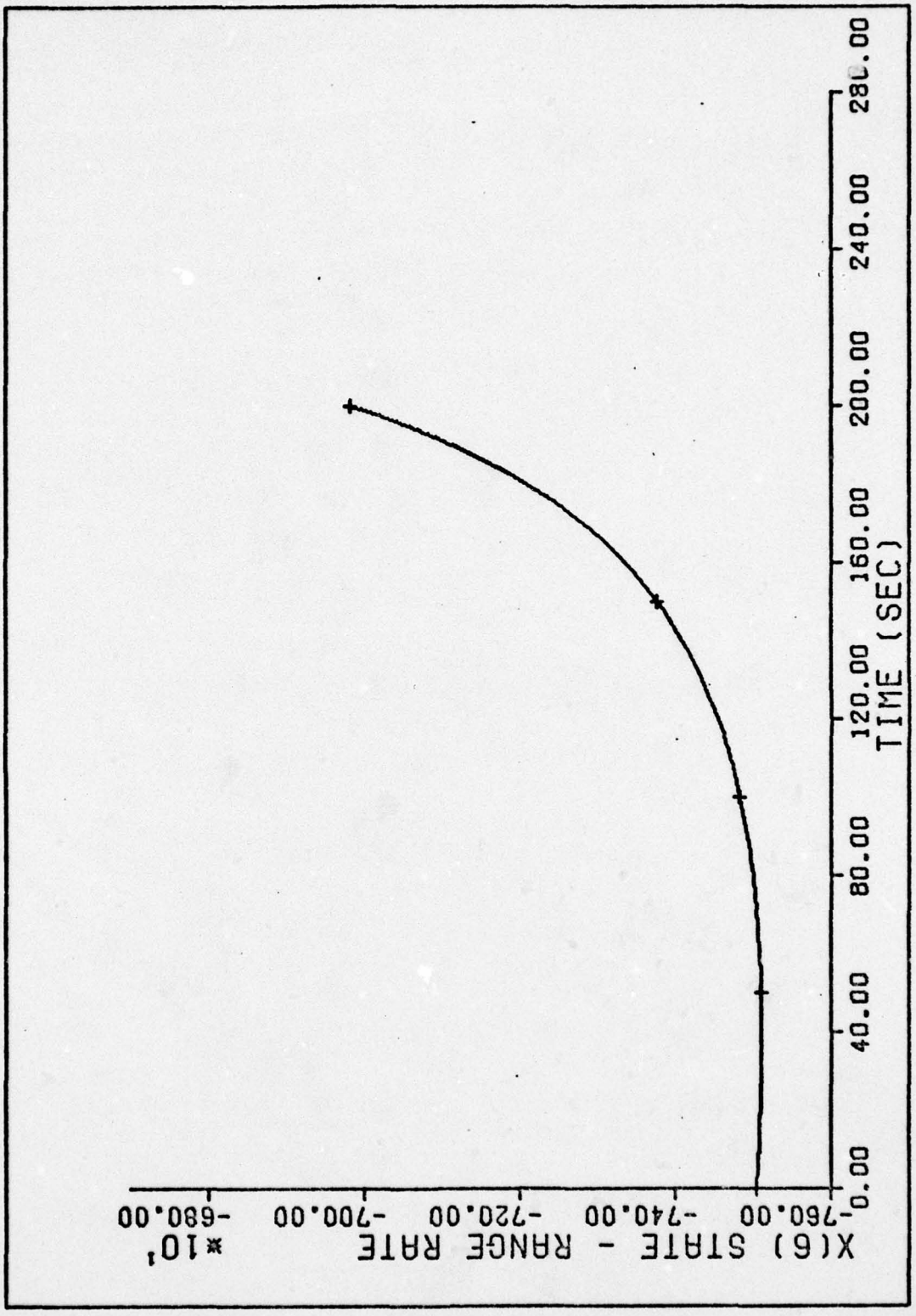


Figure 18. x(6) State Time History

satellite and tracker are closest at the end of the tracking pass. On the other hand, the plot of range rate is decreasing with time and is non-linear over the time interval.

Briefly stated, a Kalman filter is considered to be well "tuned" if the filter error variance follows the "true" system error variance as closely as possible without underestimating it. Typically this will result in the lowest possible true system error variance. This is accomplished by varying the initial filter covariance matrix $[\underline{P}_o]$, and the strengths of the state and measurement $[\underline{Q}$ and $\underline{R}]$ noises in the filter until the desired "tuning" is achieved. If the filter underestimates the error variance significantly, then the filter may diverge because the filter is weighting the filter internal model too heavily and the measurements too lightly. This is what Jazwinski has termed "learning the wrong state too well" (Ref 16:301-302). On the other hand, if the filter error variance weights the measurements too heavily, then the filter will track the noisy data and not exploit the internal model sufficiently. Divergence may also occur, in an extended Kalman filter, if constant noise strengths ($\underline{Q}$ and $\underline{R}$) are used (Ref 3:77). Such divergence characteristics can be remedied to some extent by admitting time varying (such as piecewise constant) noise strengths in the model upon which the filter is based. However, the scope of this study was confined to achieving adequate tracking performance over a reasonable time interval with the use of a single set of noise strengths. Decreased dynamic noise strengths would allow a

"tighter" tuning during the initial period while increased strengths would remove (or at least postpone) the onset of divergence. In eventual implementation, time-varying strengths, possibly set adaptively since their "best" evaluation would be trajectory dependent, could enhance filter performance (Ref 3:155).

Results of Filter Tuning

The initial Monte Carlo analysis and tuning was performed using the (unrealistic) assumption that at time t_o the filter state values were equal to the system state values (zero initial error) with variance equal to that used by Mann (Ref 5). Even though this assumption is unrealistic it is commonly used for initial tuning of the (extended) Kalman filter. The initial one sigma values for $\underline{Q}$ and $\underline{R}$ were obtained using the apriori estimates shown in Chapter V. Table II presents these initial filter tuning parameters.

Figures 19 through 30 present the results of the Monte Carlo analysis obtained from ten simulation runs. In Figures 19 through 24 each plot presents the square root of the average of the filter covariance (denoted by x on the plot) and the calculated sample standard deviation of the true error (denoted by + on the plot). Figures 25 through 30 present the calculated mean error (denoted by + on the plot) plus and minus one standard deviation. The three lines are not distinct on some plots, because the large valued mean error prevents adequate resolution. It should be noted that "automatic" scaling is used to prepare each plot. The scale factors are selected in order for the plot to fill the plot area, hence small valued changes can be exaggerated.

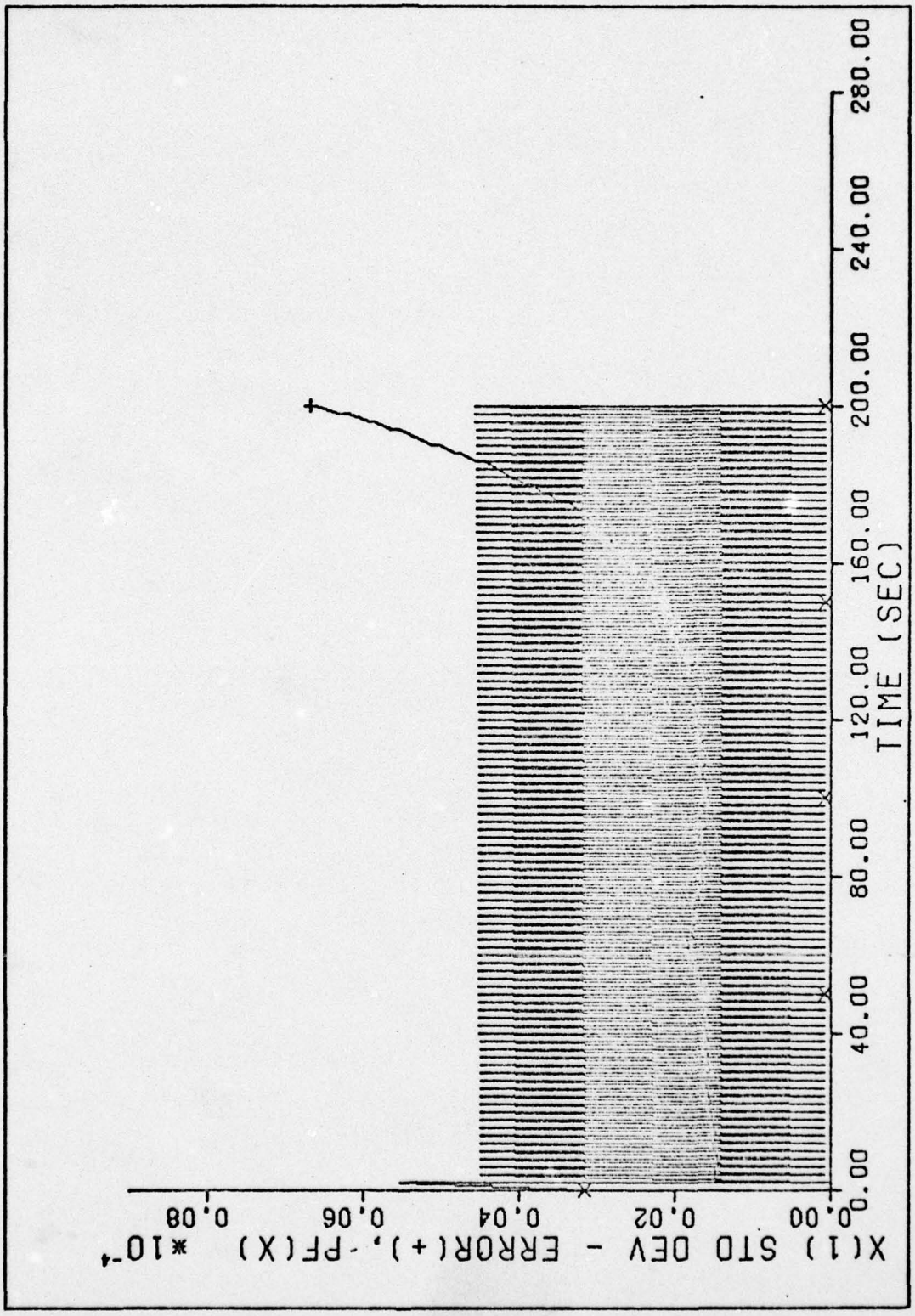


Figure 19. Initial Case - Standard Deviation vs Time

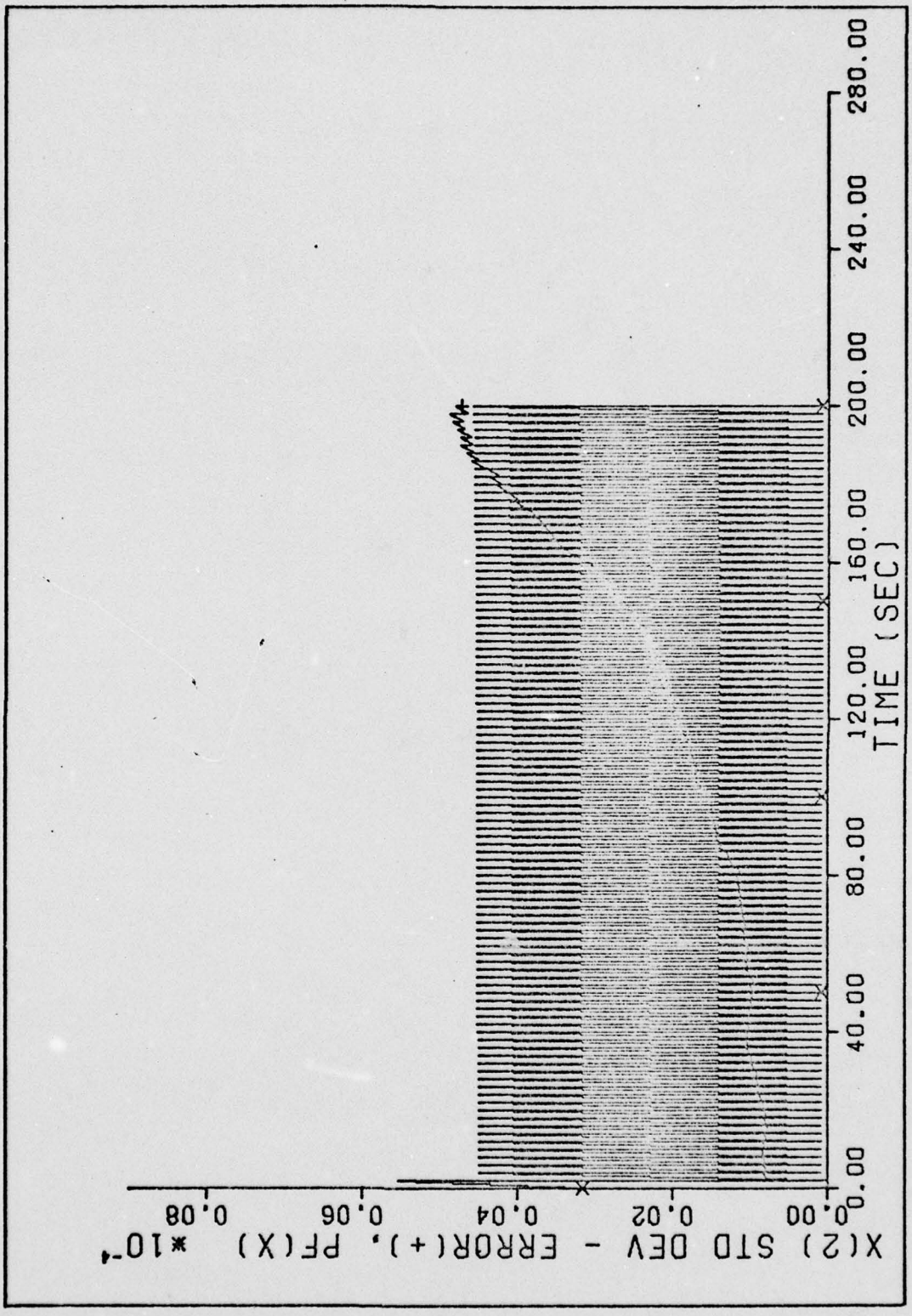


Figure 20. Initial Case - Standard Deviation vs Time

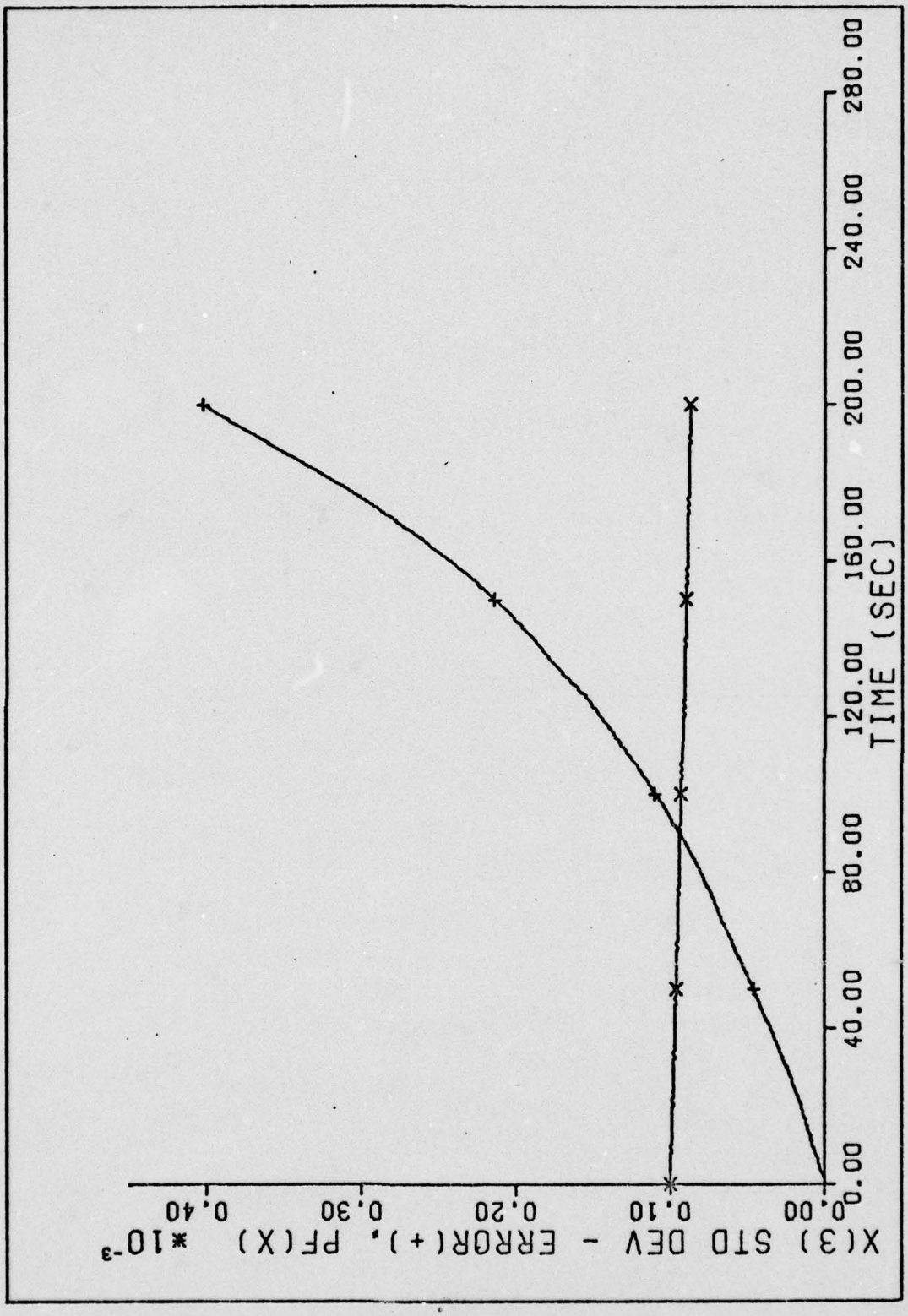


Figure 21. Initial Case - Standard Deviation vs Time

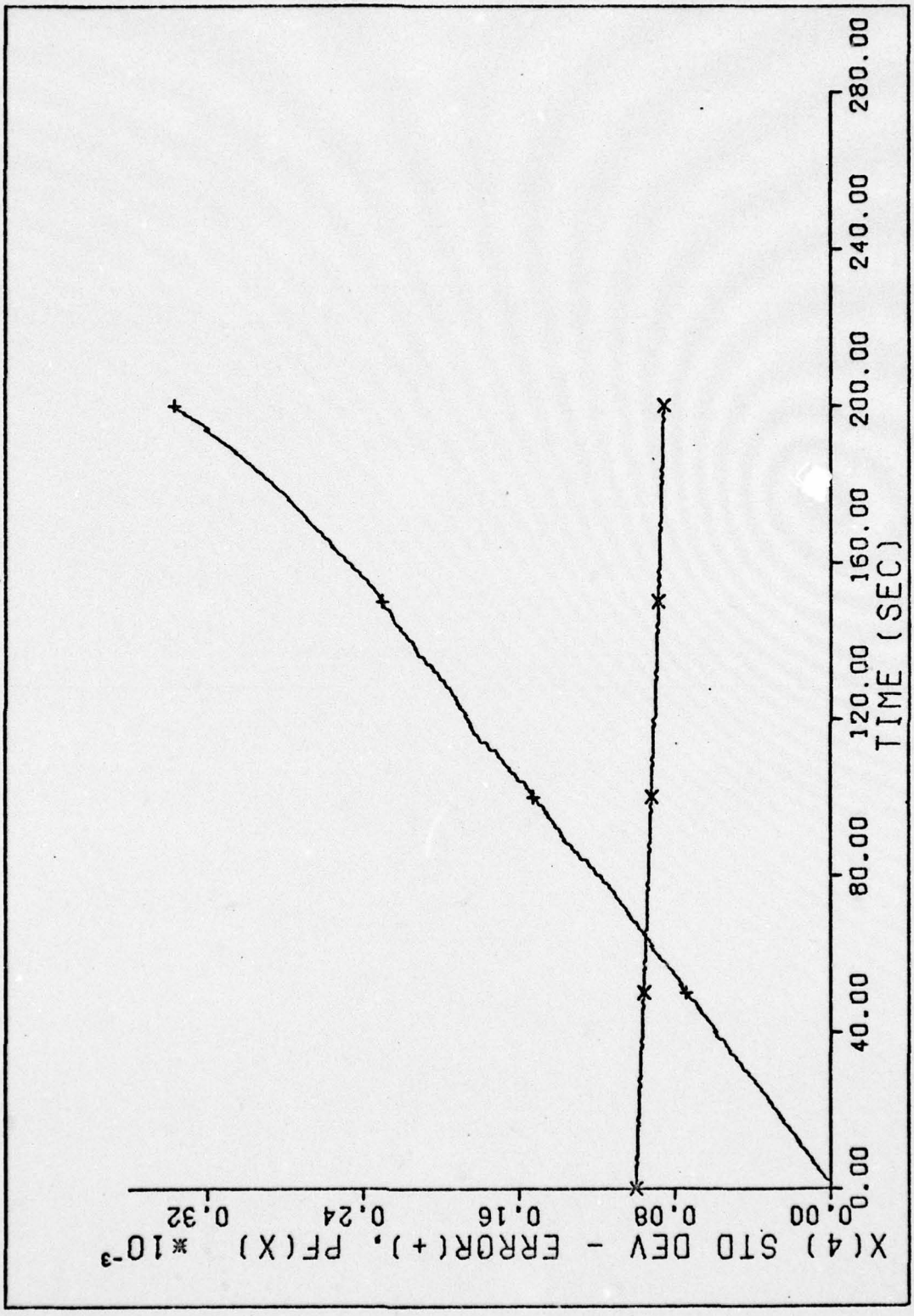


Figure 22. Initial Case - Standard Deviation vs Time

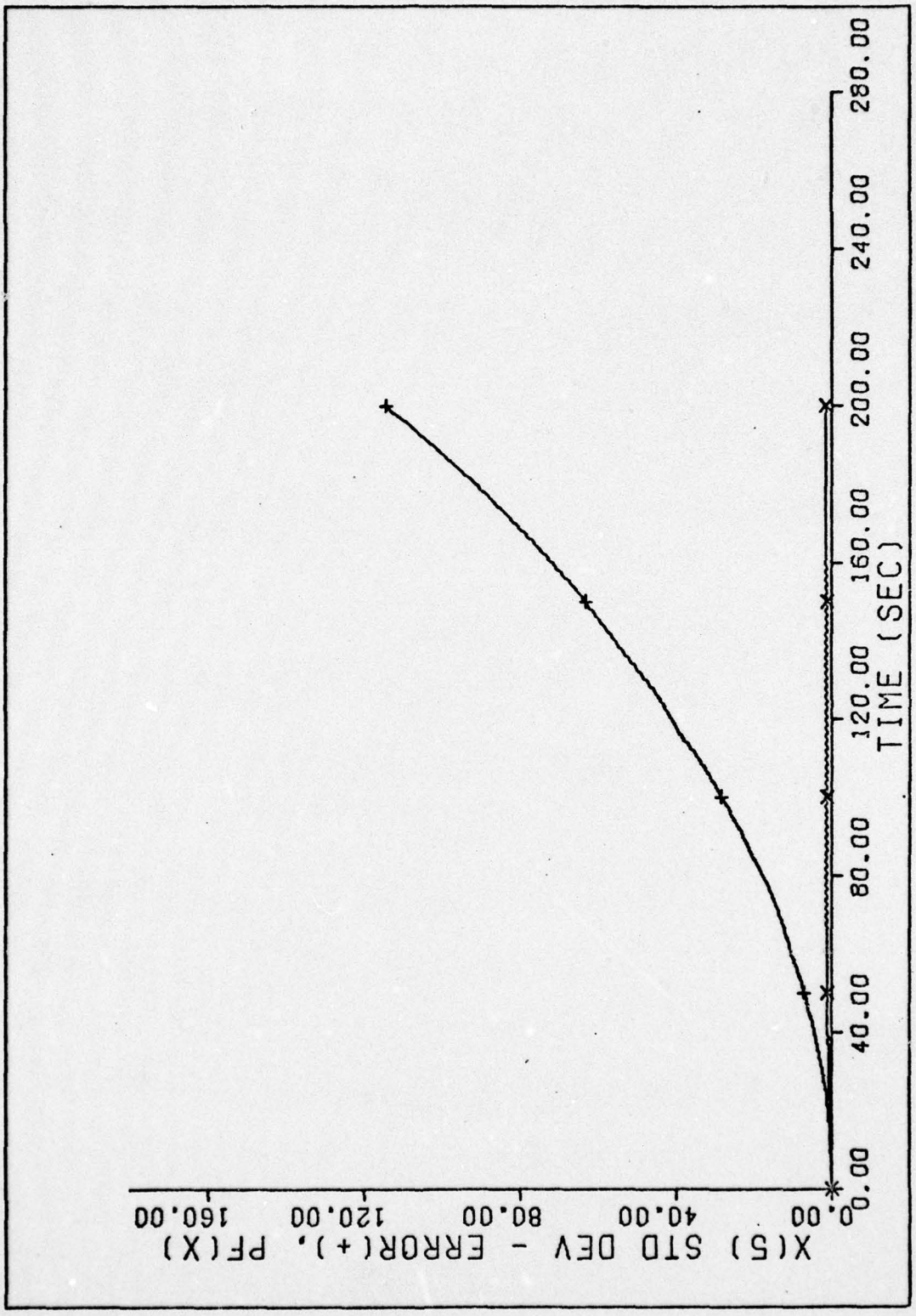


Figure 23. Initial Case - Standard Deviation vs Time

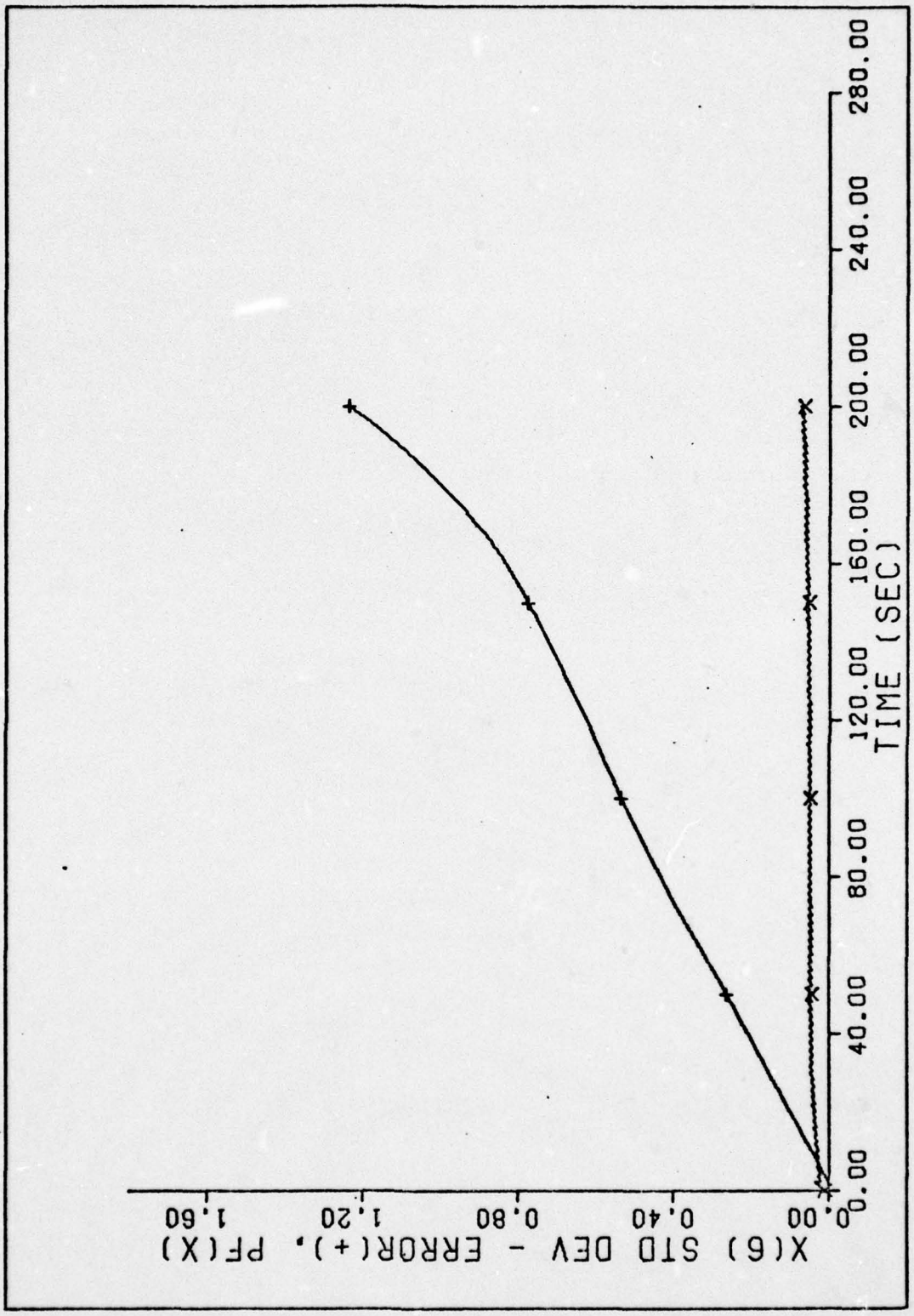


Figure 24. Initial Case - Standard Deviation vs Time

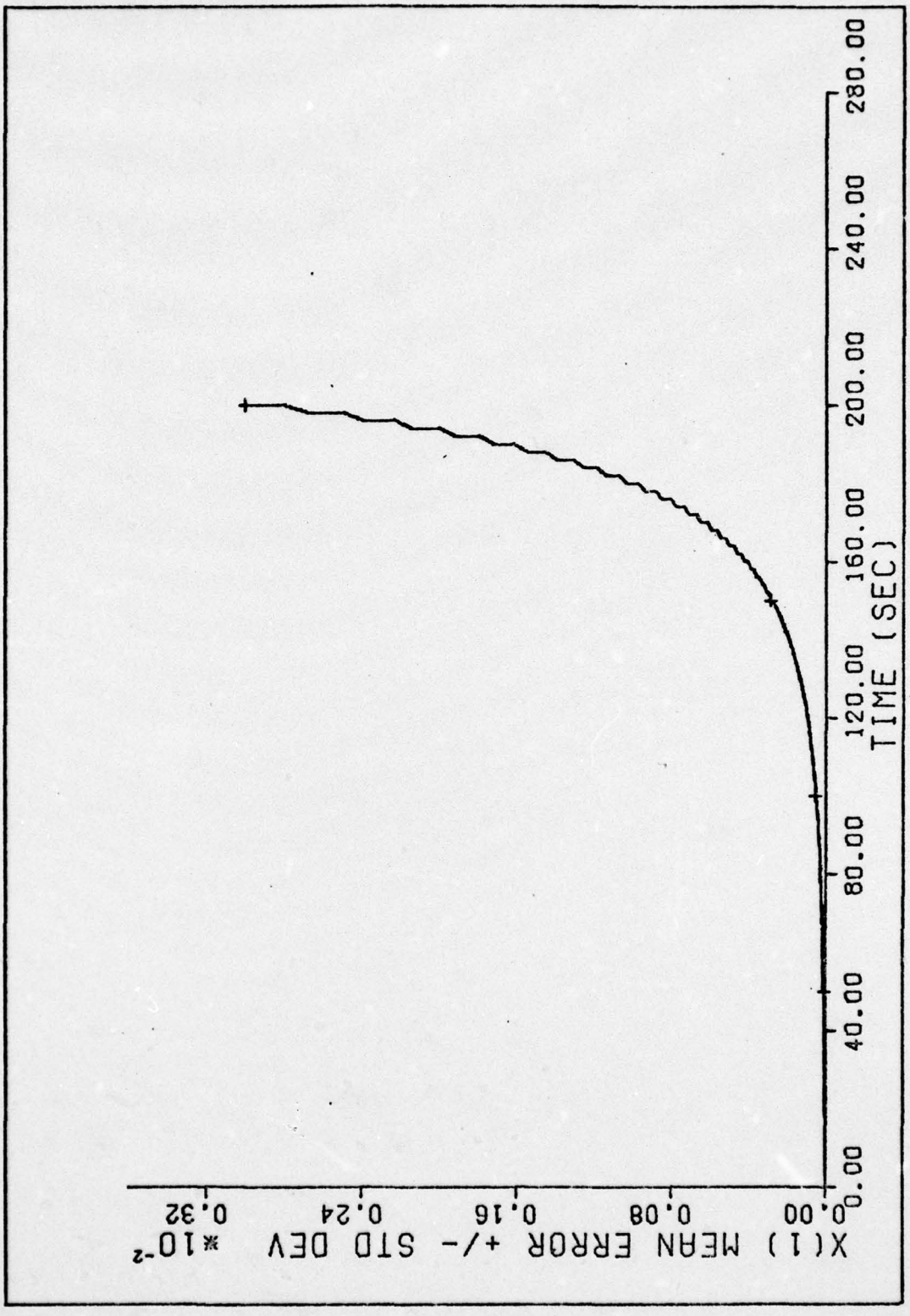


Figure 25. Initial Case - Mean Error +/- s_e vs Time

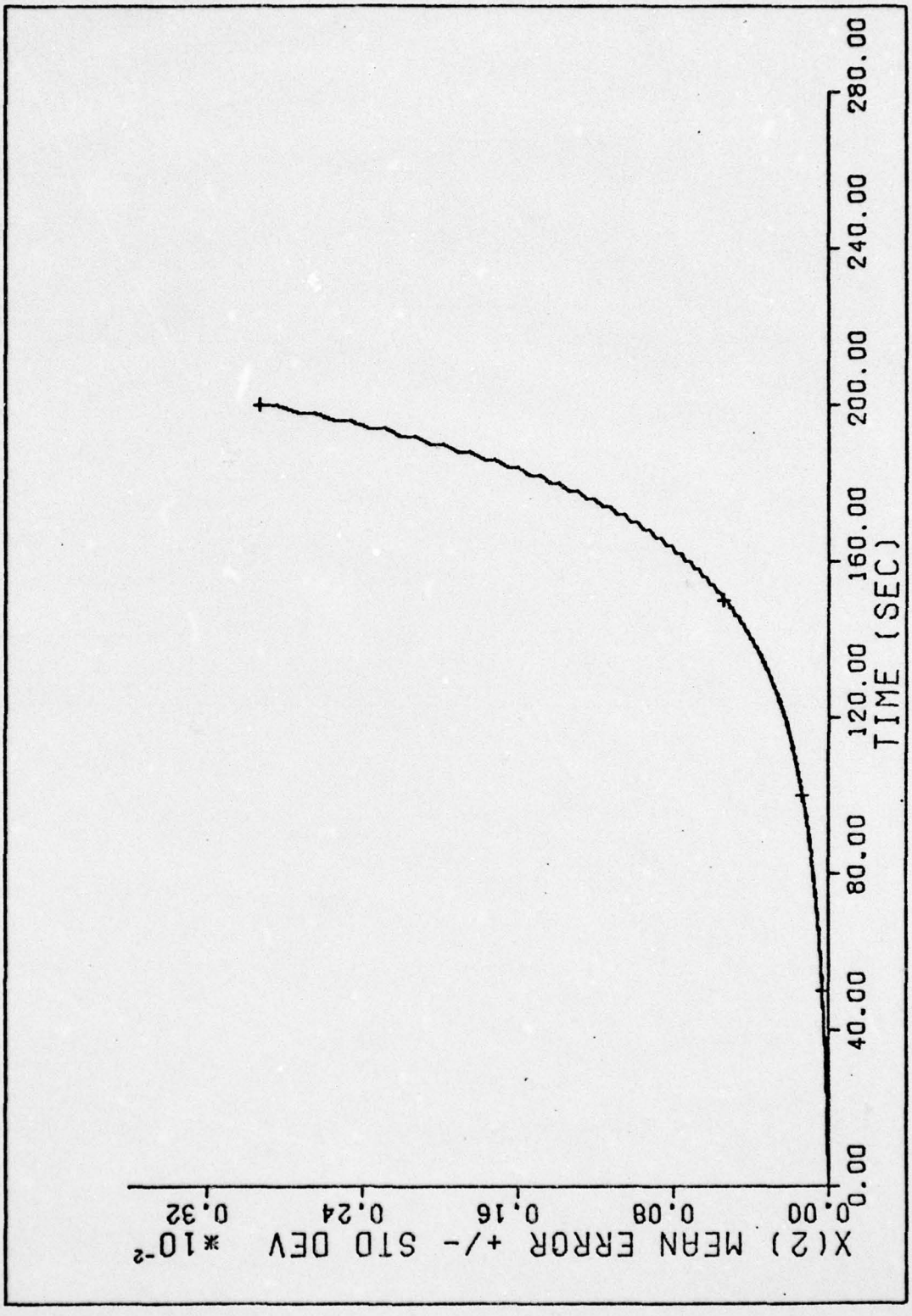


Figure 26. Initial Case - Mean Error +/- s_e vs Time

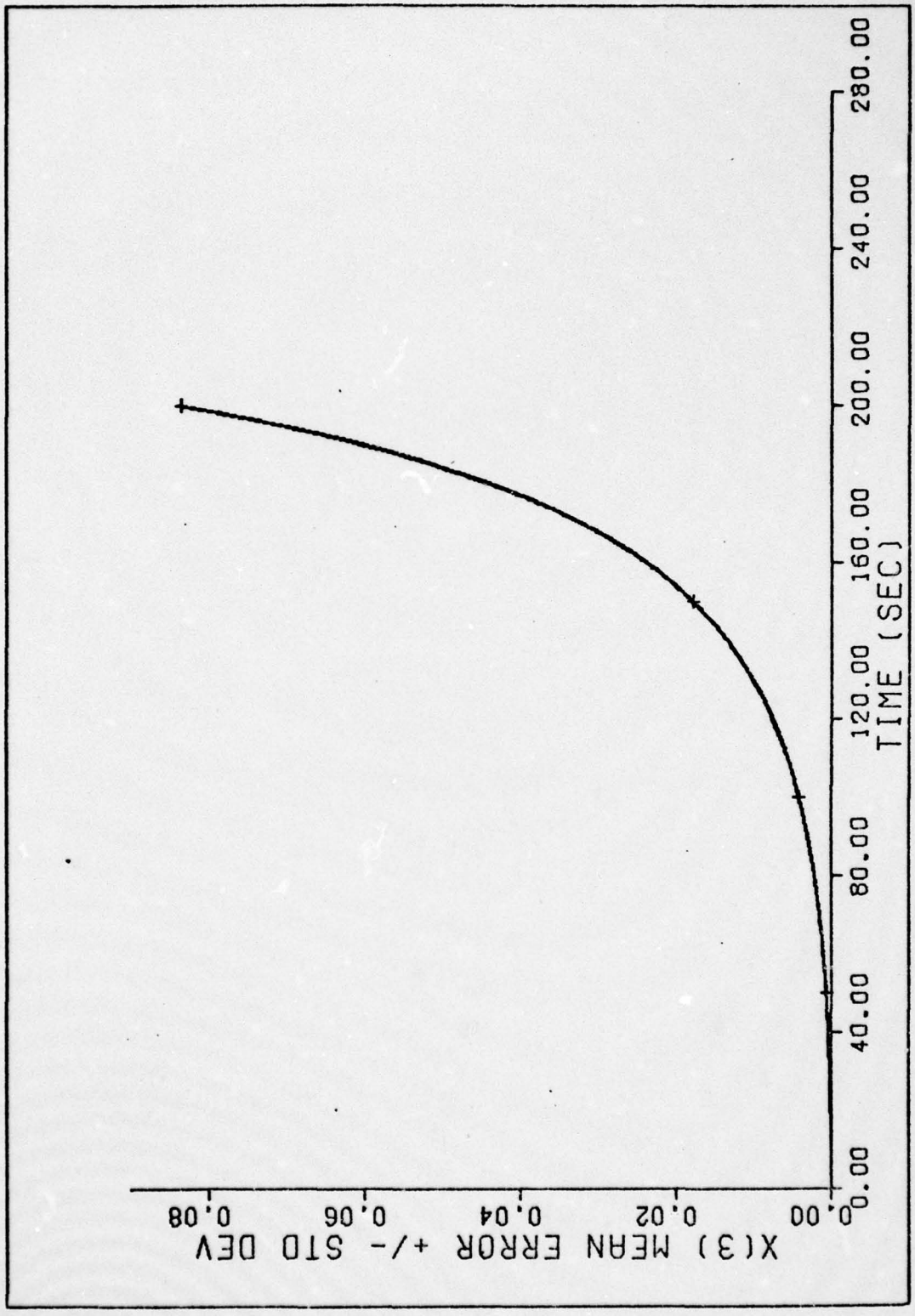
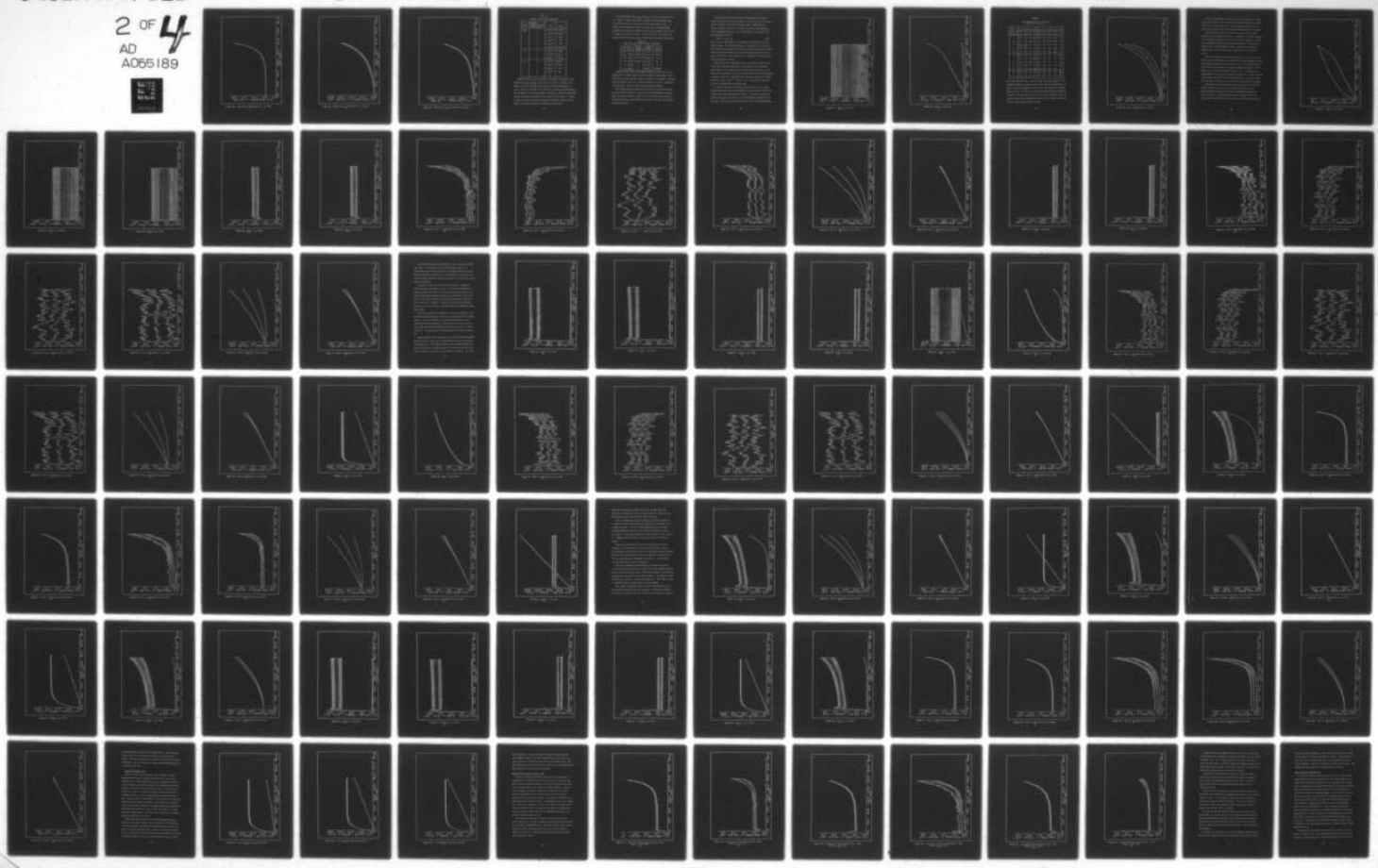


Figure 27. Initial Case - Mean Error +/- s_e vs Time

AD-A055 189 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 17/7
A GENERALIZED MONTE CARLO ANALYSIS PROGRAM FOR KALMAN FILTER DE--ETC(U)
DEC 77 K L JACKSON

UNCLASSIFIED AFIT/GGC/EE/77-6 NL

2 OF 4
AD
A055189



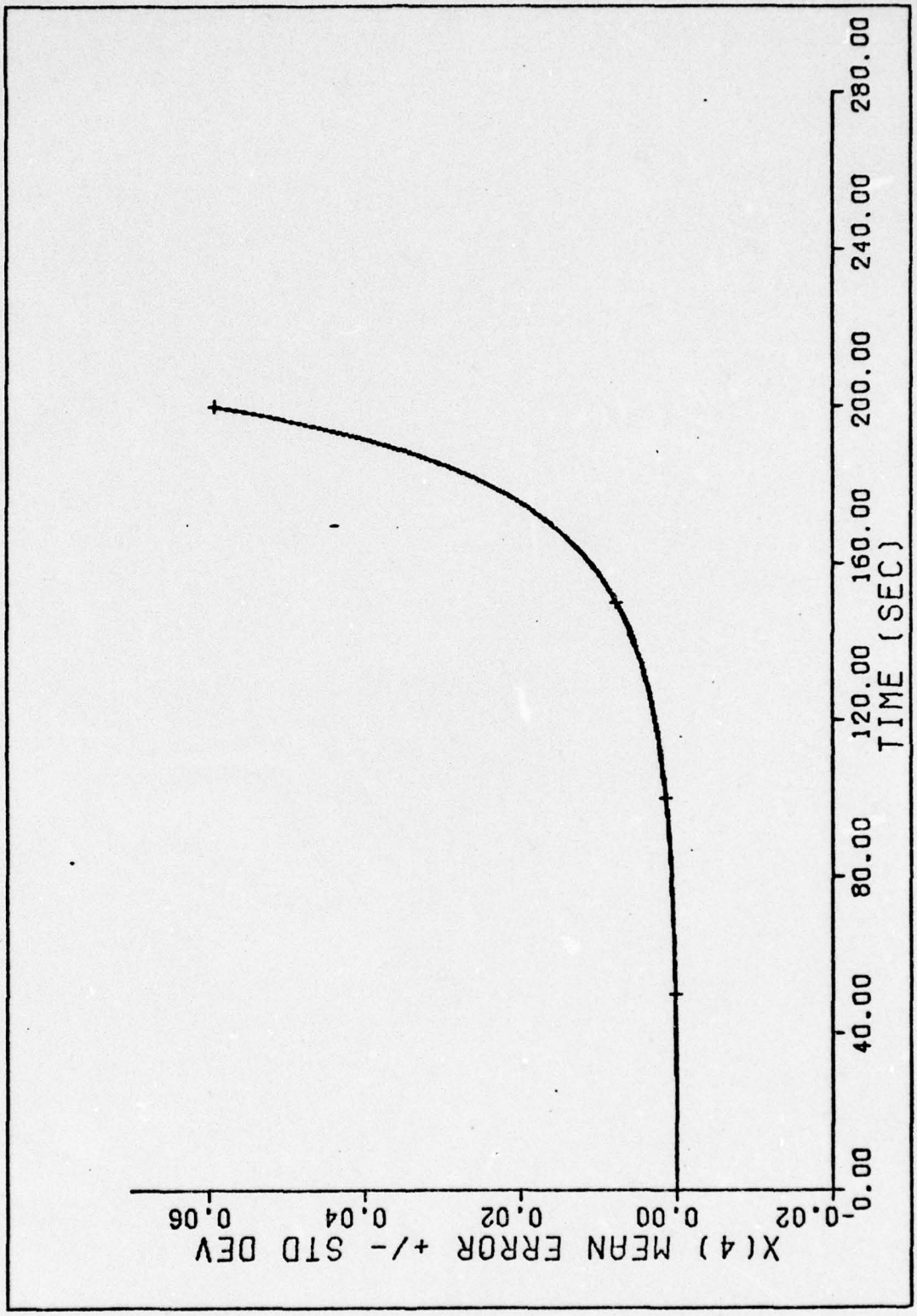


Figure 28. Initial Case - Mean Error +/- s_e vs Time

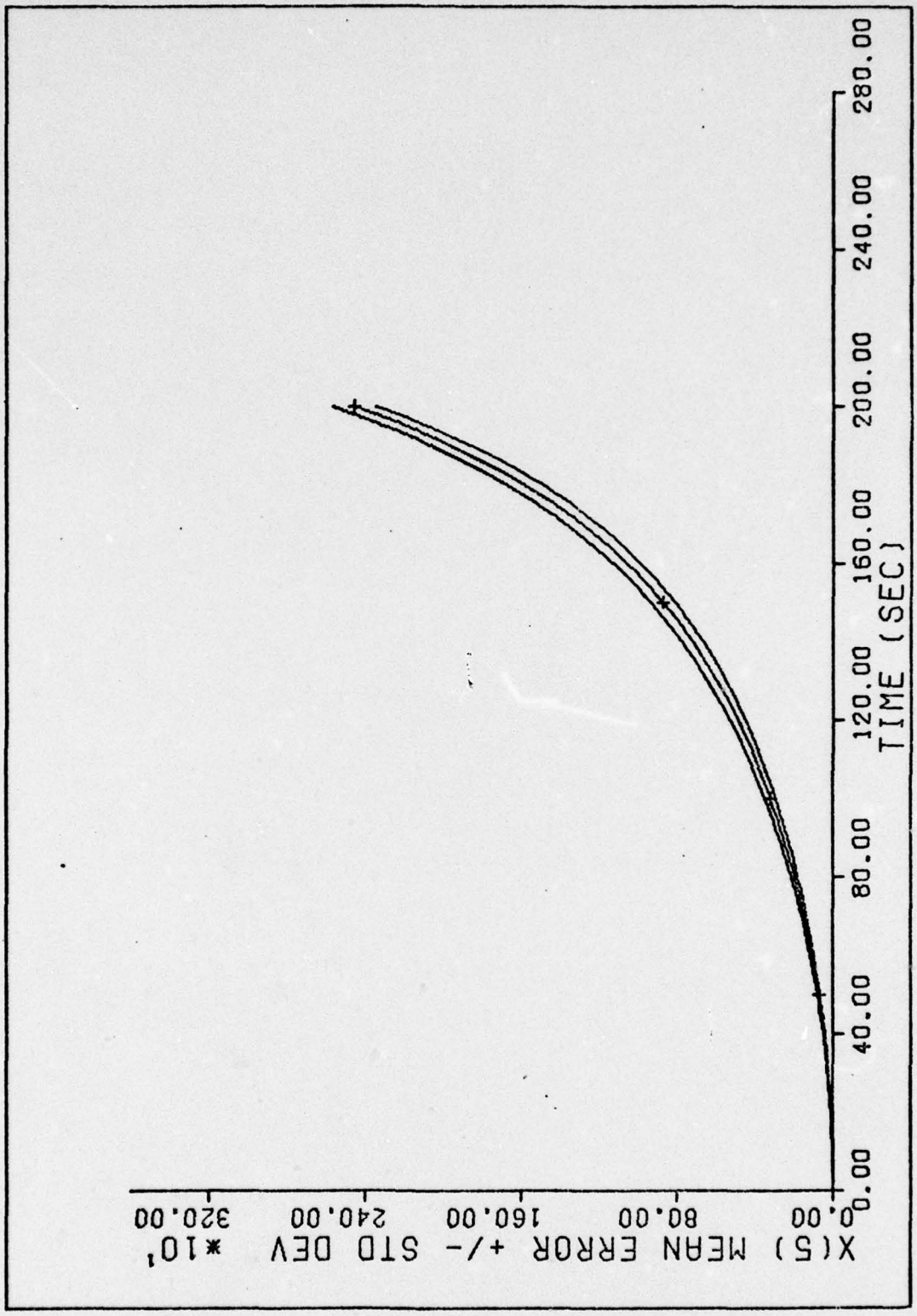


Figure 29. Initial Case - Mean Error +/- s_e vs Time

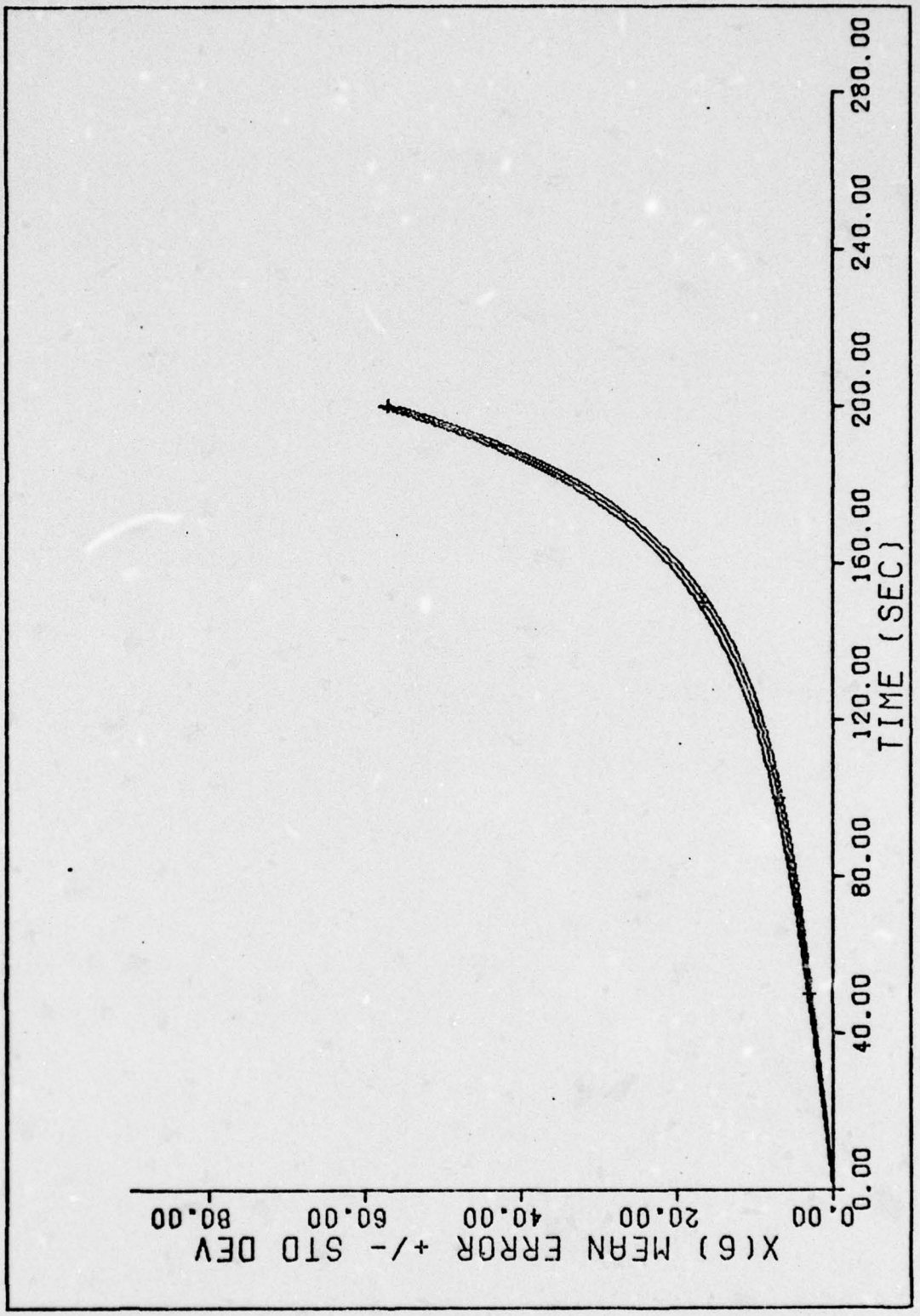


Figure 30. Initial Case - Mean Error +/- s_e vs Time

Table II

Initial Filter Tuning Parameters

Parameter		Value	Units
Matrix	Diagonal Element Index Number		
$\underline{PF}_o$	1	1.0×10^{-11}	rad/sec^2
	2	1.0×10^{-11}	rad/sec^2
	3	1.0×10^{-8}	rad^2
	4	1.0×10^{-8}	rad^2
	5	1.0×10^{-8}	Km^2
	6	1.0×10^{-10}	$(Km/sec)^2$
$\underline{Q}$	1	1.0×10^{-11}	rad/sec
	2	1.0×10^{-11}	rad/sec
	3	3.0×10^{-7}	Km/sec
$\underline{R}$	1	5.0×10^{-6}	rad/sec
	2	5.0×10^{-6}	rad/sec
	3	2.5×10^{-6}	rad
	4	2.5×10^{-6}	rad
	5	20	m

Figures 19 and 20 plot the standard deviation of the error in the estimates of w_{LSy} and w_{LSz} [states x(1) and x(2)]. As shown in the plots, the filter estimated error before the measurement update is greater than the true error until the time is greater than 180 seconds. At 180 seconds, the true error begins to diverge. After the measurement update, the filter estimated error is below the true error. It should be noted that after the initial transient (about five seconds on the plot), the filter estimates are essentially steady state.

The mean error of state w_{LSy} and w_{LSz} is plotted in Figures 25 and 26. Since the filter was tuned with filter initial conditions equal to the truth model, the plot begins at zero mean initial error. It should be noted that the magnitude of both plots are monotonically increasing over the time interval and that the measurement update tends to degrade the filter estimates. Table III presents the minimum and maximum values of the mean error starting from zero initial conditions.

Table III
State Mean Error (Zero Initial Conditions)

Parameter	Minimum Value	Maximum Value	Units
$w_{LSy} = x(1)$	$-7.3\text{x}10^{-7}$	$3.0\text{x}10^{-3}$	rad/sec
$w_{LSz} = x(2)$	$-3.4\text{x}10^{-8}$	$2.9\text{x}10^{-3}$	rad/sec
$\delta\epsilon = x(3)$	$-1.1\text{x}10^{-7}$	$8.4\text{x}10^{-2}$	rad
$\delta\nu = x(4)$	$-3.7\text{x}10^{-6}$	$5.6\text{x}10^{-2}$	rad
$R = x(5)$	0	$2.6\text{x}10^{+3}$	Km
$V_r = x(6)$	0	$5.8\text{x}10^{+1}$	Km/sec

Figures 21 and 22 plot the standard deviation of the error in the estimates of the error states $\delta\nu$ and $\delta\epsilon$ [states x(3) and x(4)]. Note that the true error standard deviation diverges in both cases. The mean error for states $\delta\nu$ and $\delta\epsilon$ is plotted in Figures 27 and 28. Again note that both plots start from zero initial error and increase.

The standard deviation plots for states R and V_r [states x(5) and x(6)] are shown in Figures 23 and 24 with the corresponding mean error plots in Figures 29 and 30. Again note that the true error standard deviation diverges in both cases and the mean starts from zero initial error and increases.

Several Monte Carlo simulations were performed while varying values of $\underline{Q}$ and $\underline{R}$ for each simulation. No value was found that would prevent the x(5) state (range) from diverging. Recall from the filter equations (Chapter V), the only x(5) state tuning parameter is the measurement noise R_5. The state equation for the x(5) state was modified as follows:

$$\dot{x}(5) = \dot{R} = V_r + w_4 \tag{201}$$

where w_4 is a "pseudonoise" with strength to be determined from the filter tuning. If a fictitious noise, or "pseudonoise", is added to the filter state equations, then the filter better accounts for (cannot neglect) the incompleteness of the filter representation of the true system (Ref 3:9-4). The addition of this "pseudonoise" yields another x(5) state tuning parameter.

Twelve Monte Carlo simulations (10 runs each) were then performed to tune the extended Kalman filter. The initial filter covariance terms ($\underline{PF}_o$) were not changed from the values shown in Table II. The filter tuning parameters ($\underline{Q}$ and $\underline{R}$) used for each of these 12 simulations are shown in Table IV. The entries in the table are shown in an abbreviated form where $1.0x10^{-11}$ is written 1-11, also entries underlined are values changed from the previous case.

In order to increase the standard deviation error in the filter estimate for state x(5), the "pseudonoise" (Q_4) was set to $1.0x10^{+6}$ and a Monte Carlo simulation performed (case 1). Except for states x(5) and x(6) the plots were the same as in the initial simulation and are not shown. Figures 31 and 32 show the plots for the standard deviation

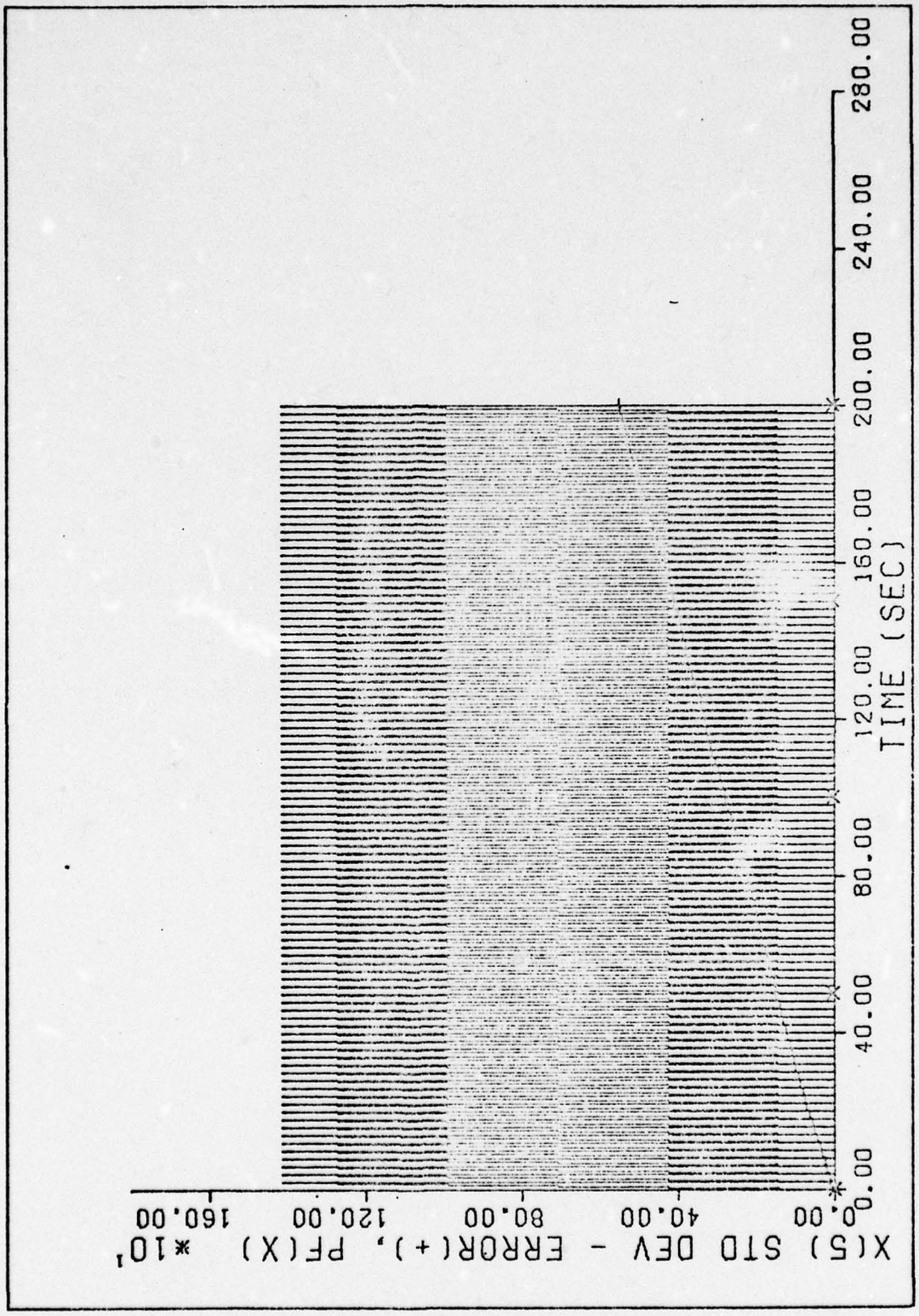


Figure 31. Case 1 - s_e vs Time

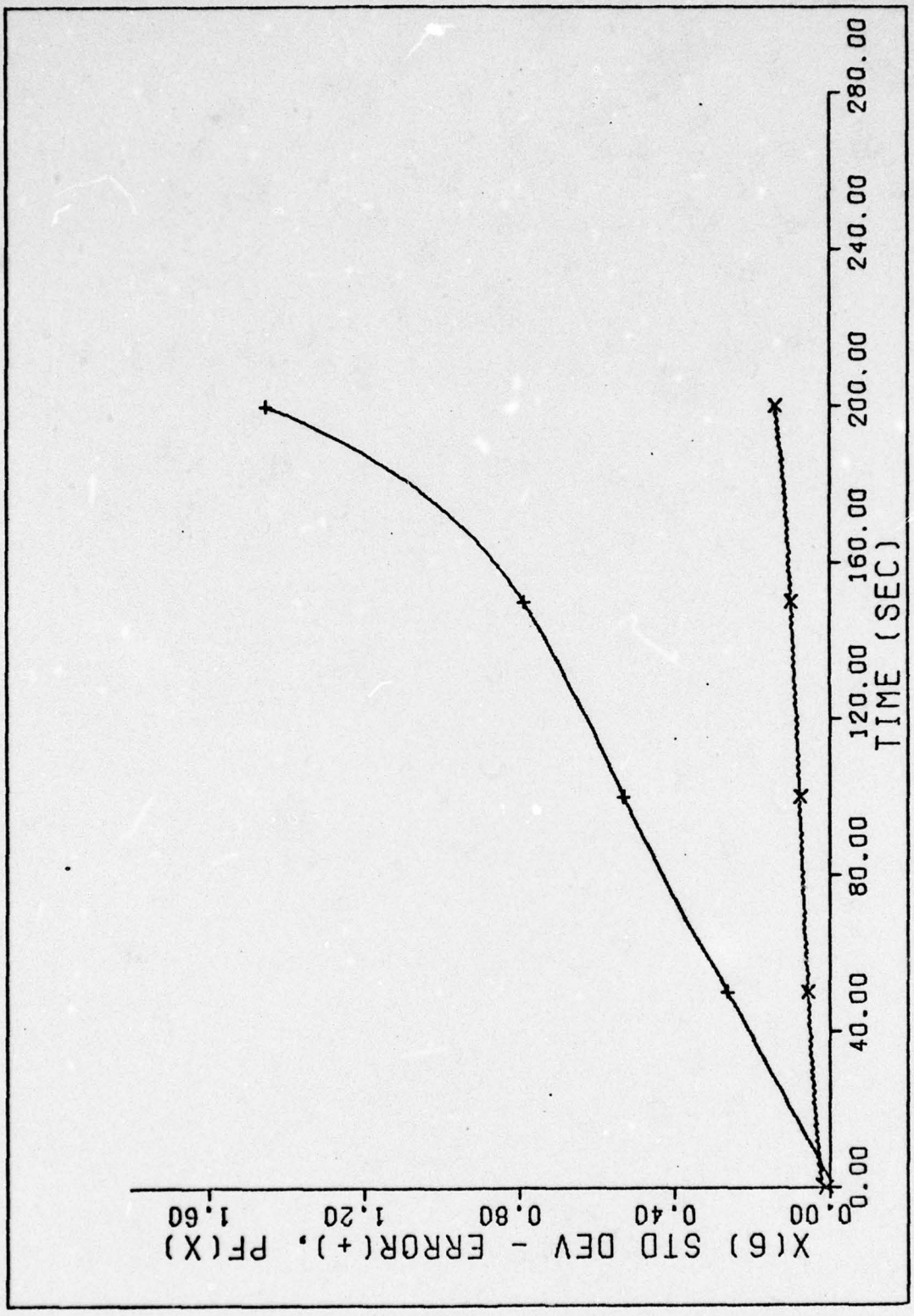


Figure 32. Case 1 - s_e vs Time

Table IV

Filter Tuning Parameters for Each
Monte Carlo Simulation

Case #	Q				R				
	1	2	3	4	1	2	3	4	5
Initial	1-11	1-11	3-7	---	5-6	5-6	2.5-6	2.5-6	20
1	1-11	1-11	3-7	1+6	5-6	5-6	2.5-6	2.5-6	20
2	1-11	1-11	1-2	1+6	5-6	5-6	2.5-6	2.5-6	20
3	5-11	5-11	1-2	1+6	5-6	5-6	2.5-6	2.5-6	20
4	5-11	5-11	1-2	1+6	5-6	5-6	1-8	1-8	20
5	5-11	5-11	1-2	1+6	5-6	5-6	1-9	1-9	20
6	5-11	5-11	1-2	1+6	5-6	5-6	1-9	1-9	1+5
7	5-11	5-11	1-2	1+3	5-6	5-6	1-9	1-9	1+5
8	5-11	5-11	1-4	1+3	5-6	5-6	1-9	1-9	1+3
9	5-11	5-11	1-4	1+3	5-6	5-6	1-9	1-9	1+4
10	5-11	5-11	1-4	1+2	5-6	5-6	1-9	1-9	1+4
11	5-11	5-11	1-4	1+2	5-6	5-6	1-9	1-9	5+4
12	5-11	5-11	1-4	1+2	5-6	5-6	1-9	1-9	3+4

for states x(5) and x(6). For state x(5) the filter estimate of the standard deviation increased significantly before the measurement update. Also, the standard deviation of the true error has increased. For state x(6), the true and the filter standard deviation plots both increased. The mean error plots for state x(5) is shown in Figure 33. The mean error is approximately the same as in the initial simulation, however, the one sigma bounds are wider.

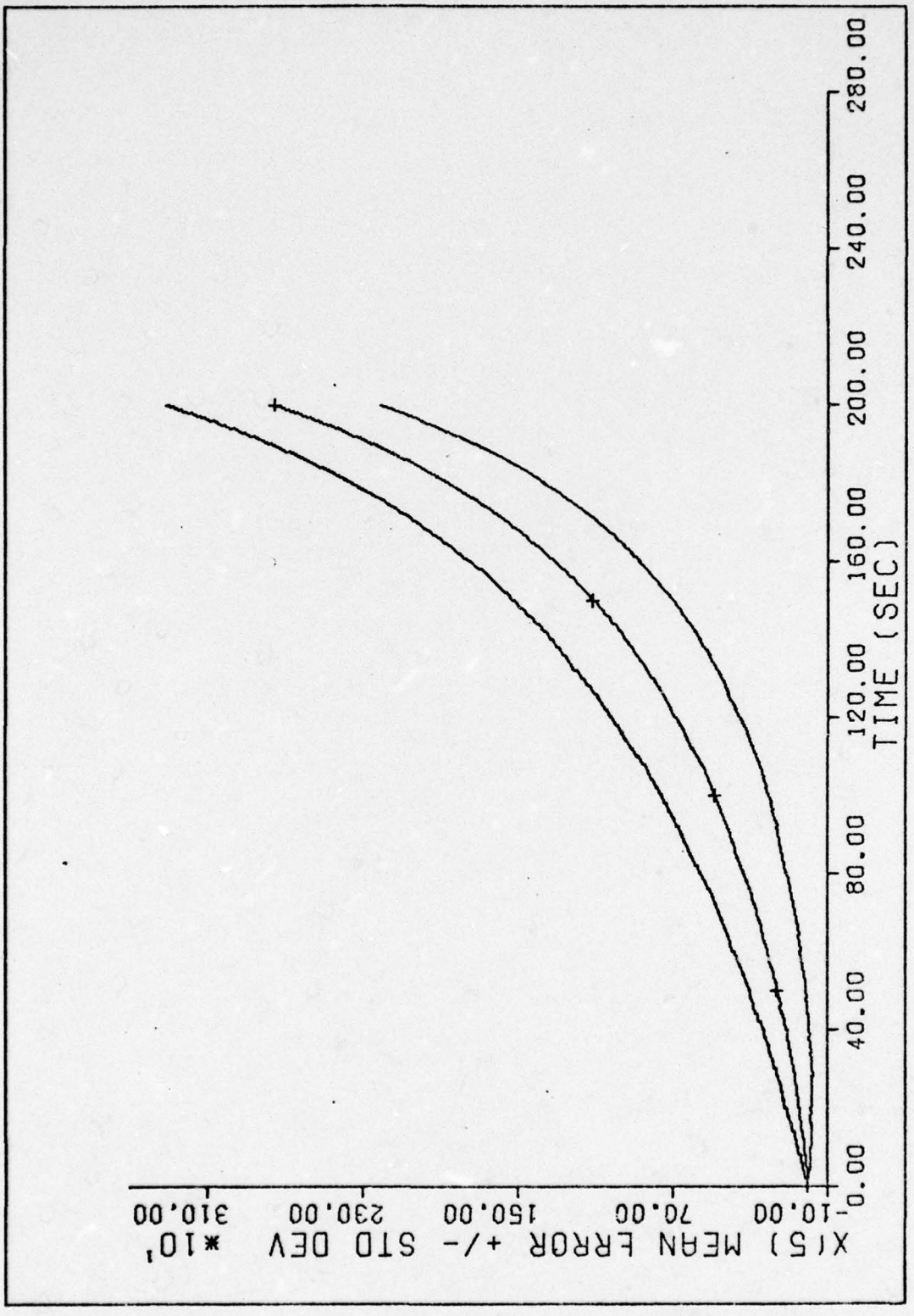


Figure 33. Case 1 - Mean Error s_e vs Time

Case 2 was performed by increasing Q_3 so that the error in state x(6) would increase. The only plot that changed noticeably was state x(6) and is shown in Figure 34. The true error standard deviation remained unchanged, however, the filter standard deviation increased.

Next, in order to increase the error in states x(1) and x(2), Q_1 and Q_2 were increased for case 3. As expected, the filter estimate of the standard deviation for states x(1) and x(2) increased, so that it is greater than the true error standard deviation, which did not change to any visually apparent degree. These plots are shown in Figures 35 and 36. The other plots remained the same and are not shown.

Case 4 was performed by decreasing R_3 and R_4 so that the error in state x(3) and x(4) would vary more before and after measurement update. The standard deviation for states x(3) and x(4) changed significantly, with the true error standard deviation decreasing by a factor of 1/3. These plots are shown in Figures 37 and 38. All of the mean error plots changed and are shown in Figures 39 through 44. The most significant change is noted on states x(1) through x(4). For these states the mean error decreased, some by as much as four orders of magnitude.

R_3 and R_4 were decreased again for case 5 in order to reduce the filter estimates of the standard deviation for states 3 and 4. Again, as shown in Figures 45 and 46, the standard deviation for states x(3) and x(4) changed significantly. Also all of the mean error plots changed noticeably and are shown in Figures 47 through 52.

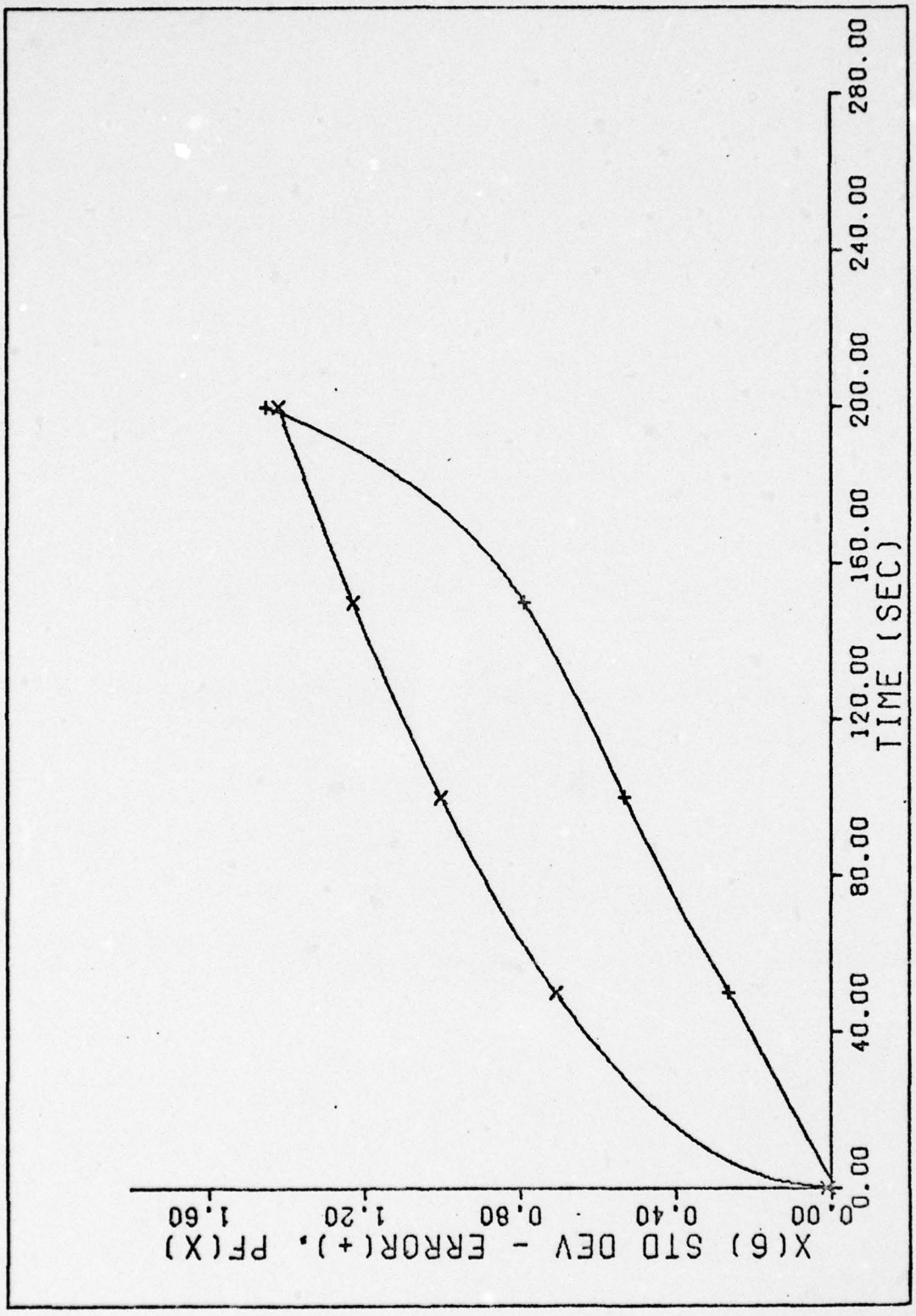


Figure 34. Case 2 - s_e vs Time

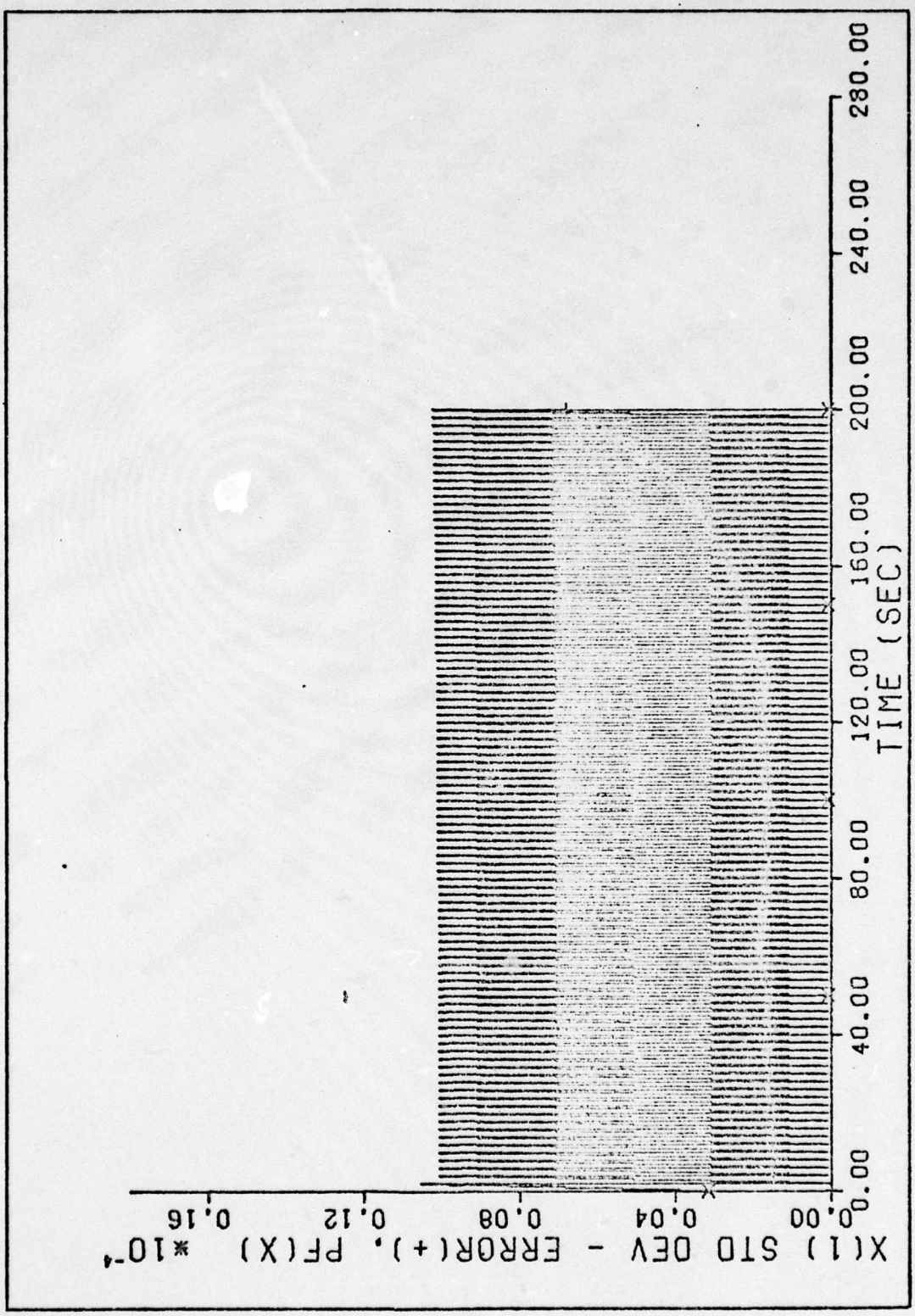


Figure 35. Case 3 - s_e vs Time

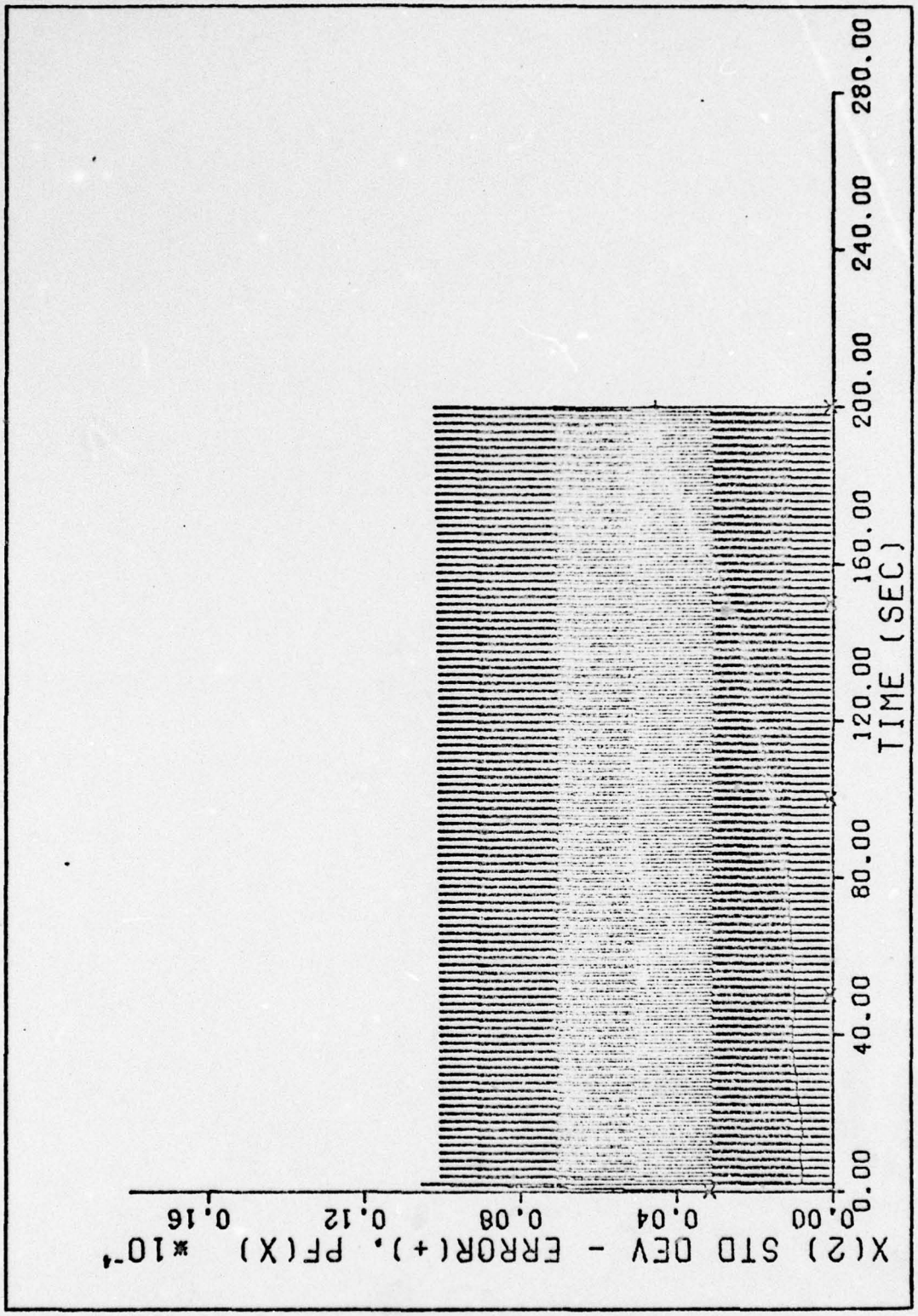


Figure 36. Case 3 - s_e vs Time

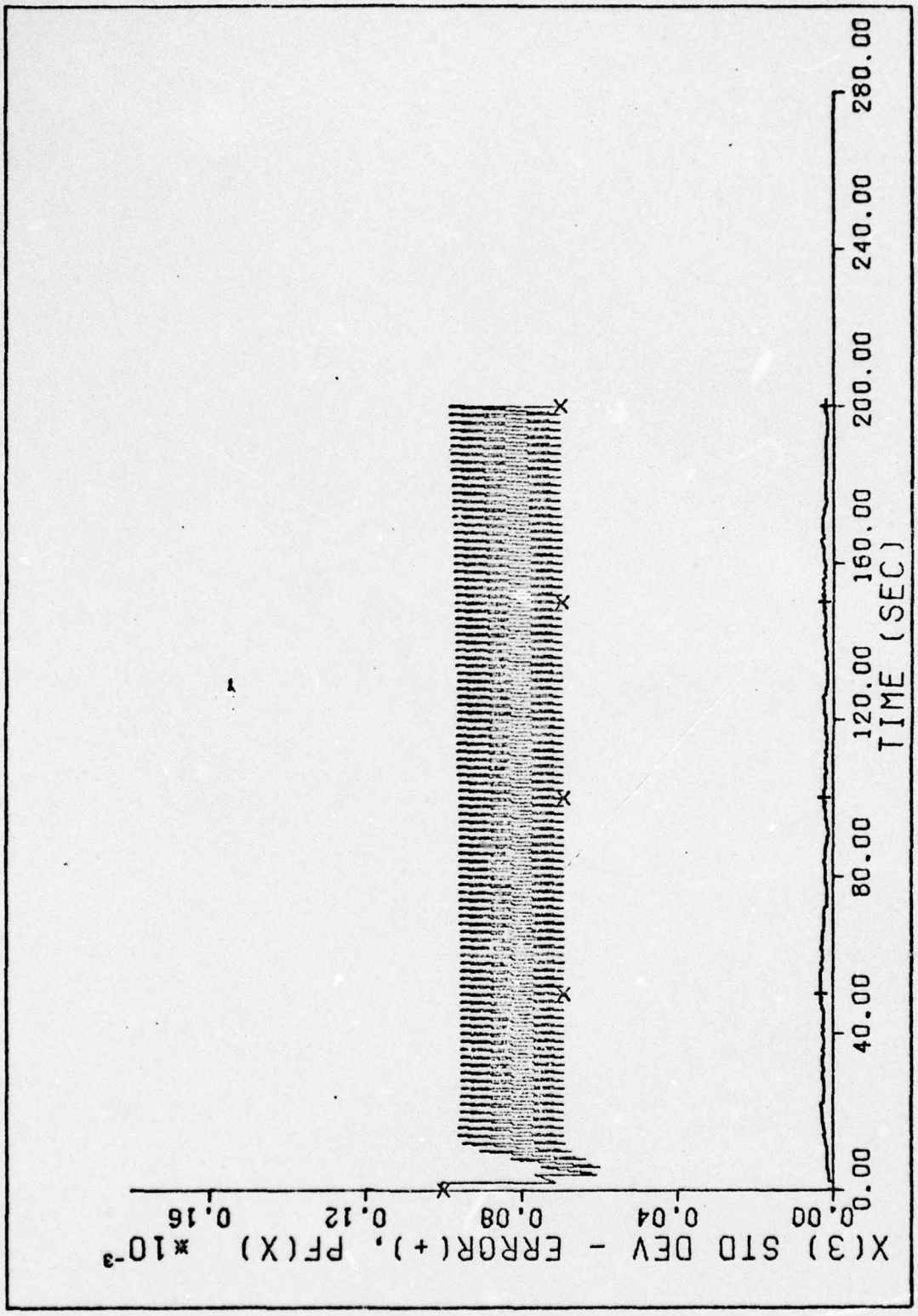


Figure 37. Case 4 - s_e vs Time

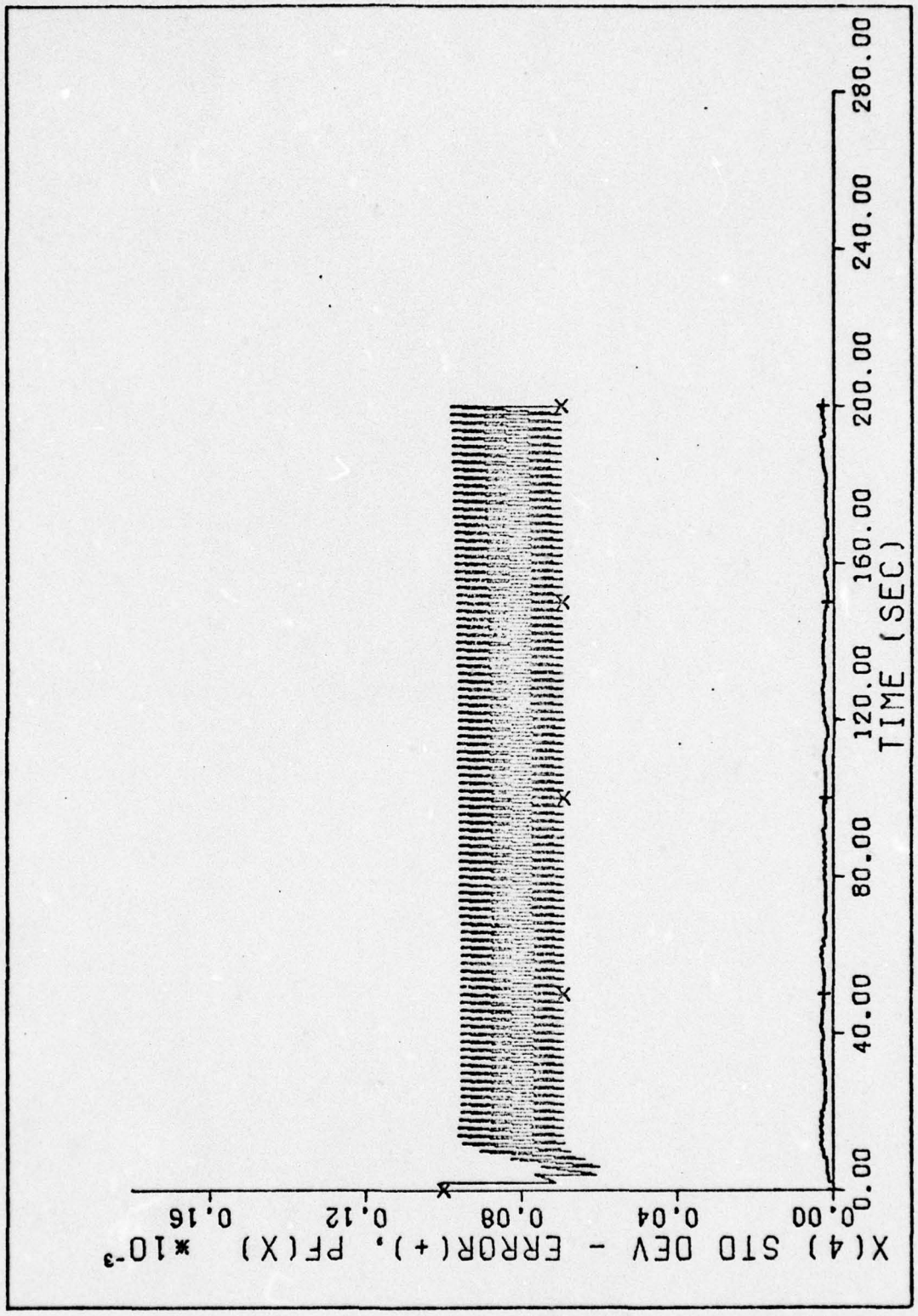


Figure 38. Case 4 - s_e vs Time

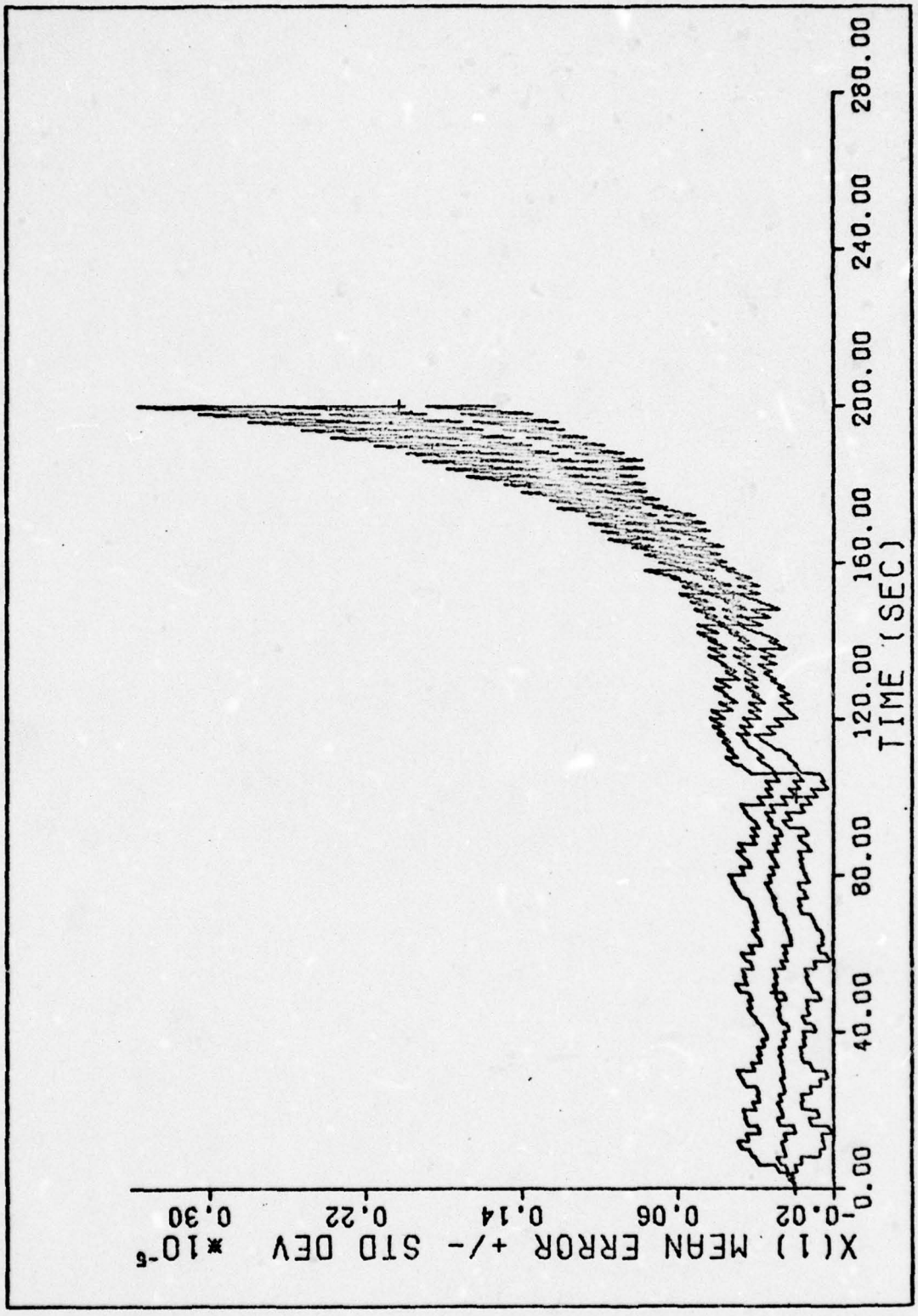


Figure 39. Case 4 - Mean Error +/- s_e vs Time

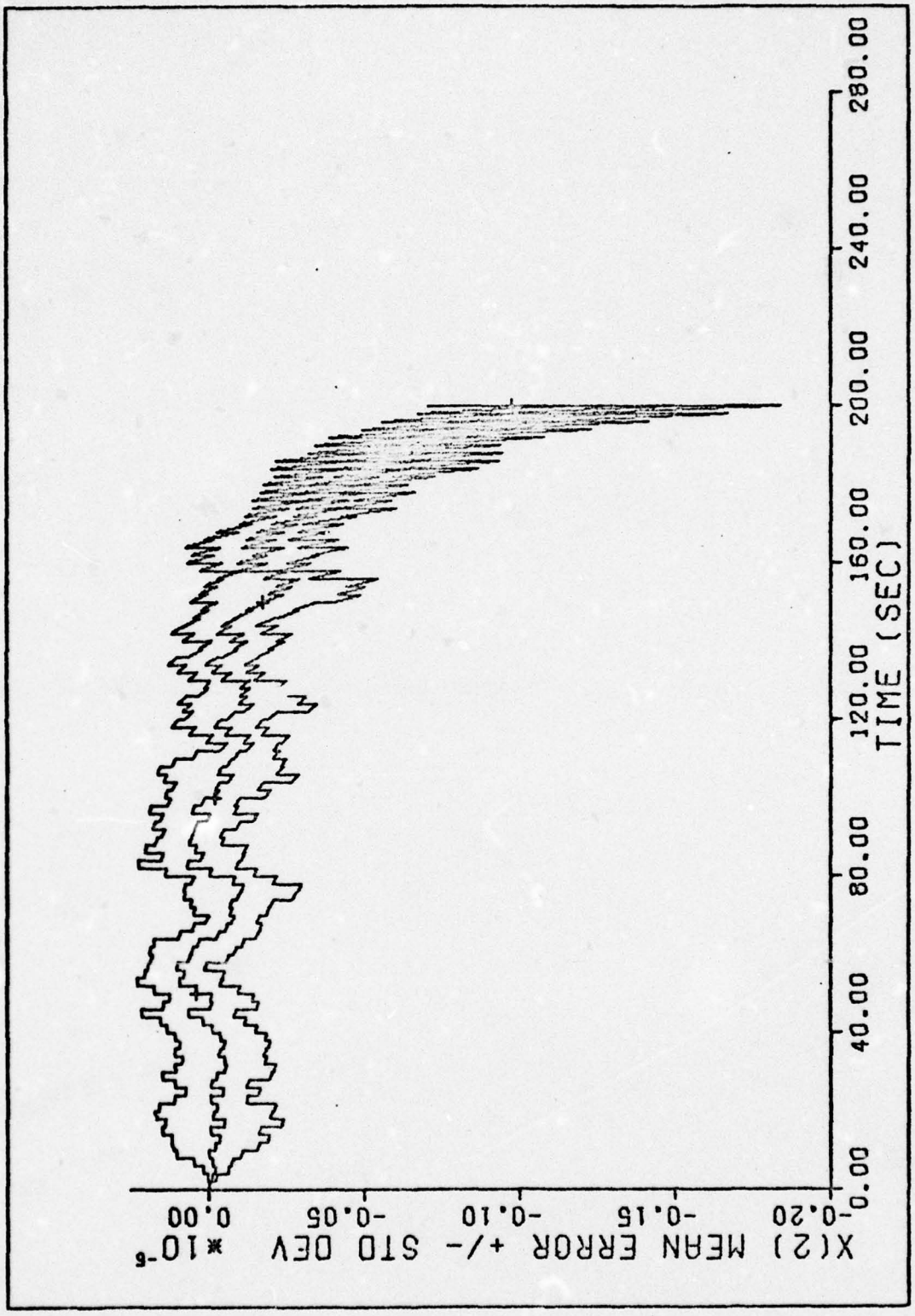


Figure 40. Case 4 - Mean Error +/- s_e vs Time

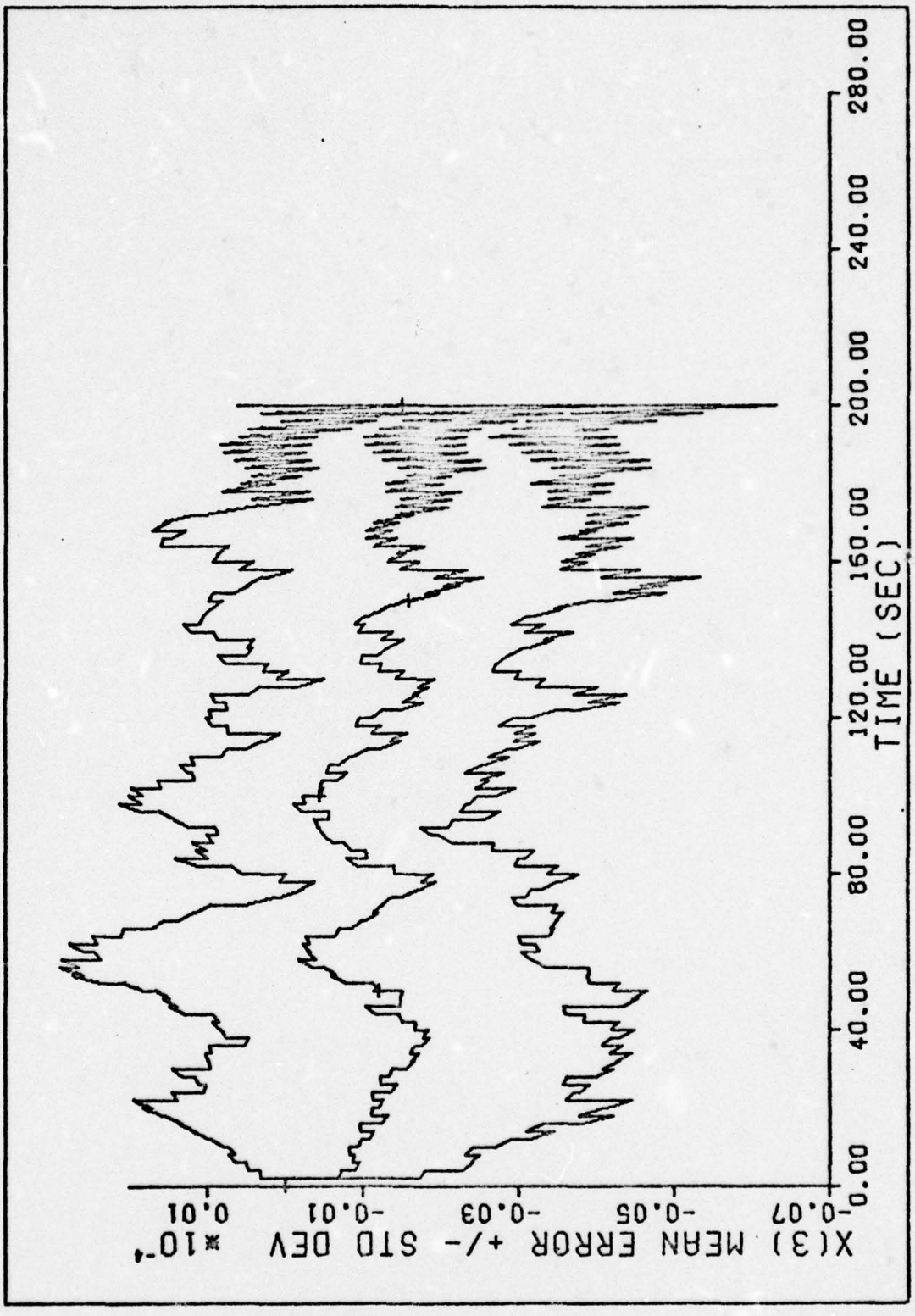


Figure 41. Case 4 - Mean Error +/- s_e vs Time

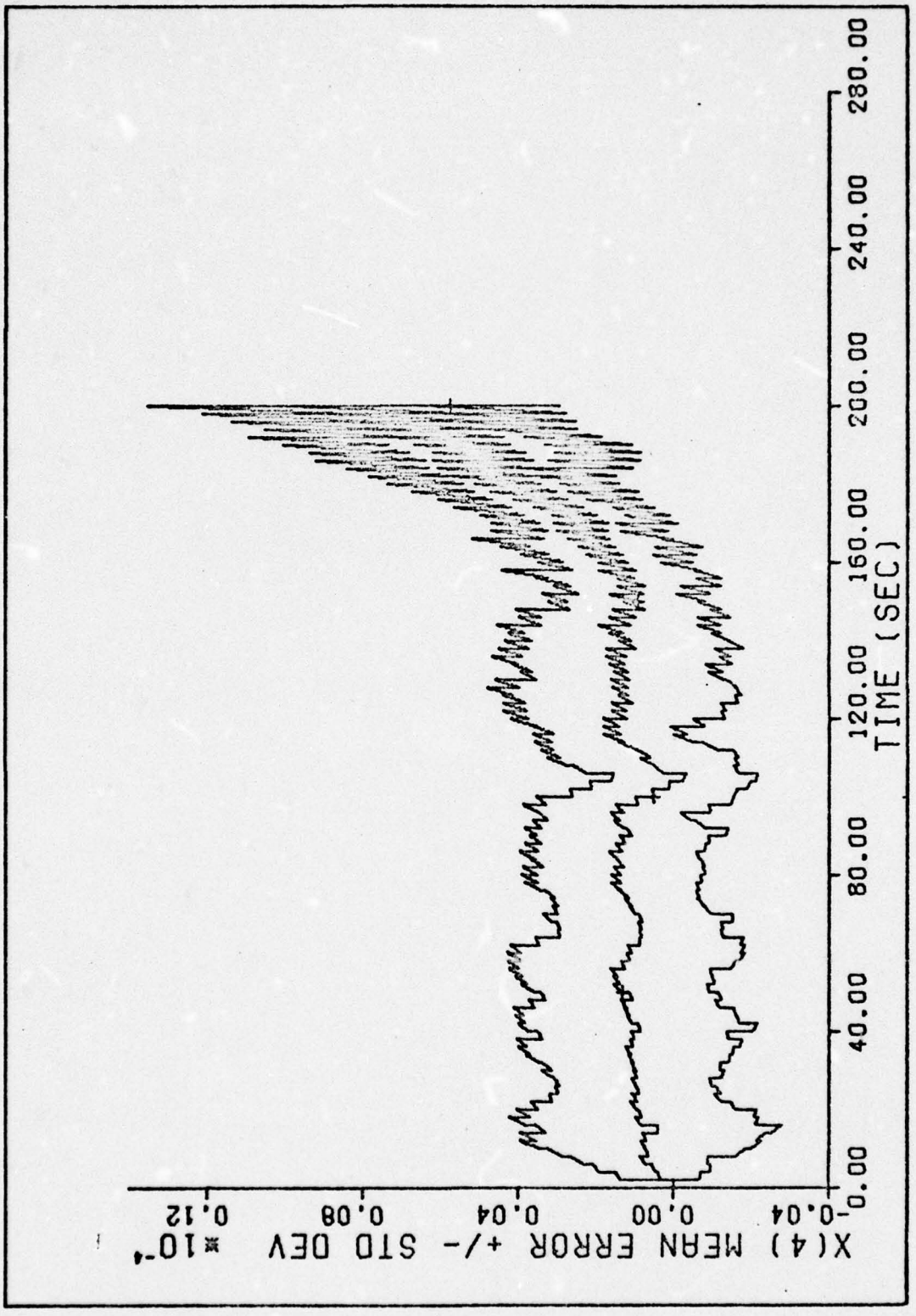


Figure 42. Case 4 - Mean Error +/- s_e vs Time

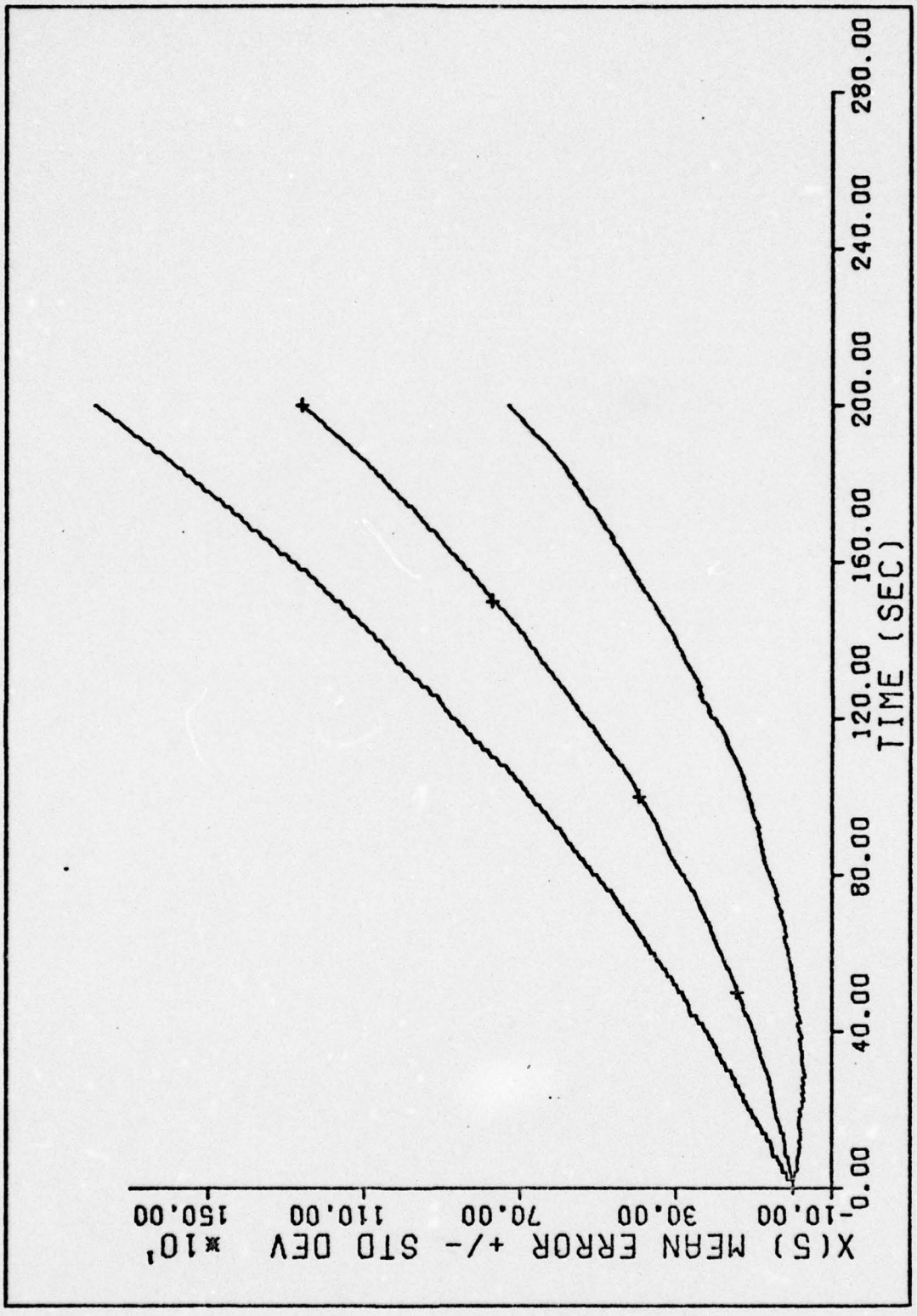


Figure 43. Case 4 - Mean Error +/- s_e vs Time

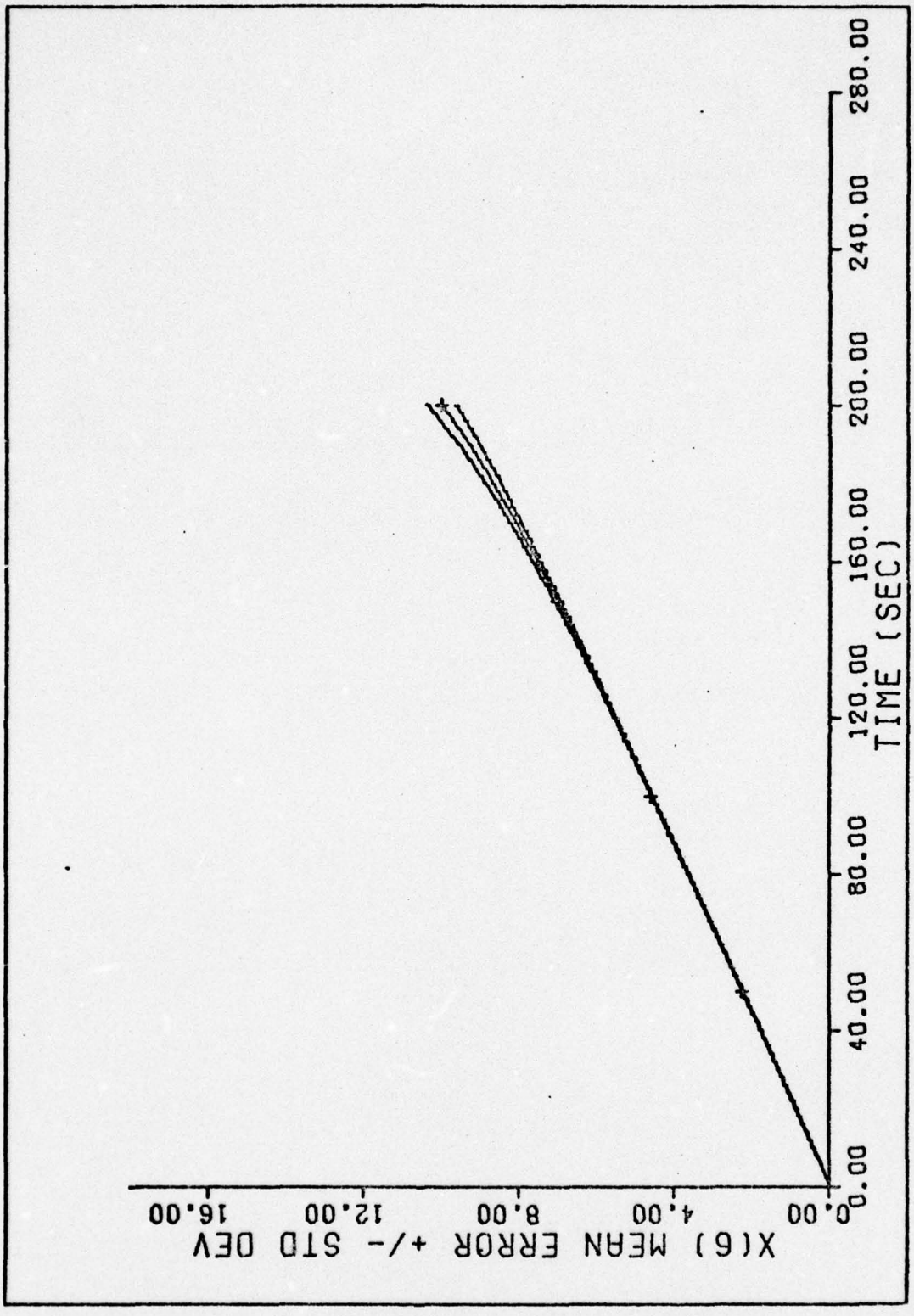


Figure 44. Case 4 - Mean Error +/- s_e vs Time

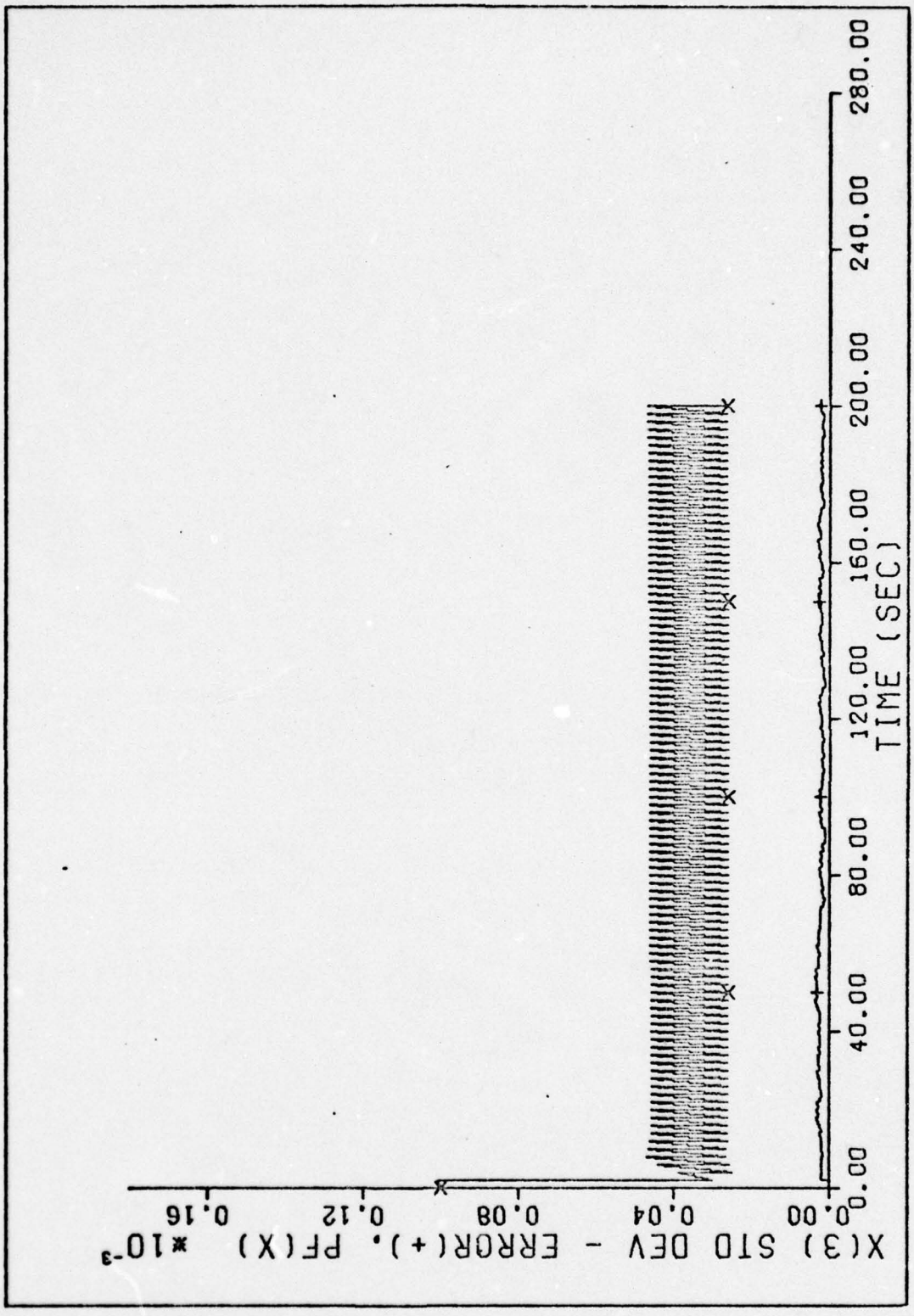


Figure 45. Case 5 - s_e vs Time

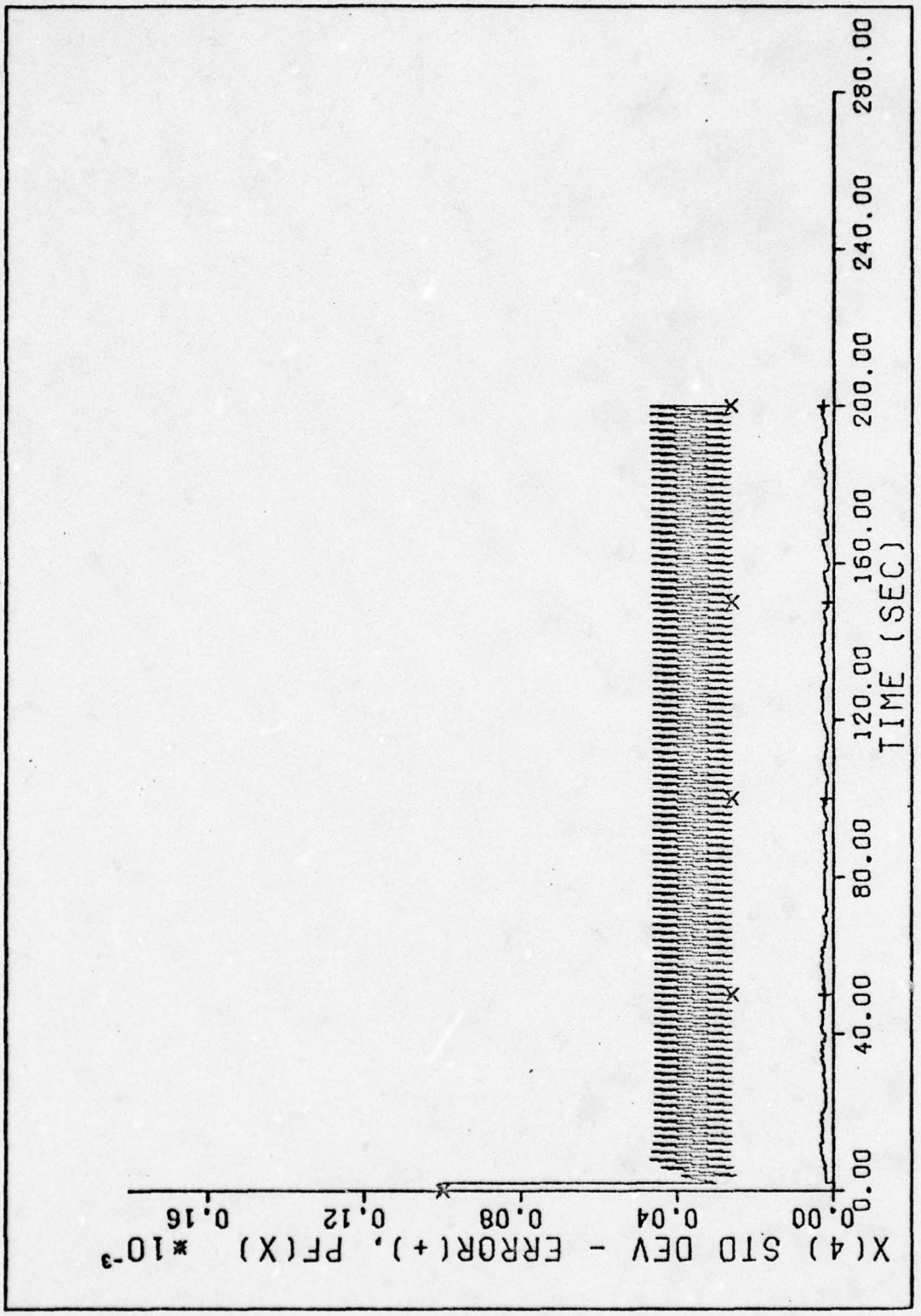


Figure 46. Case 5 - s_e vs Time

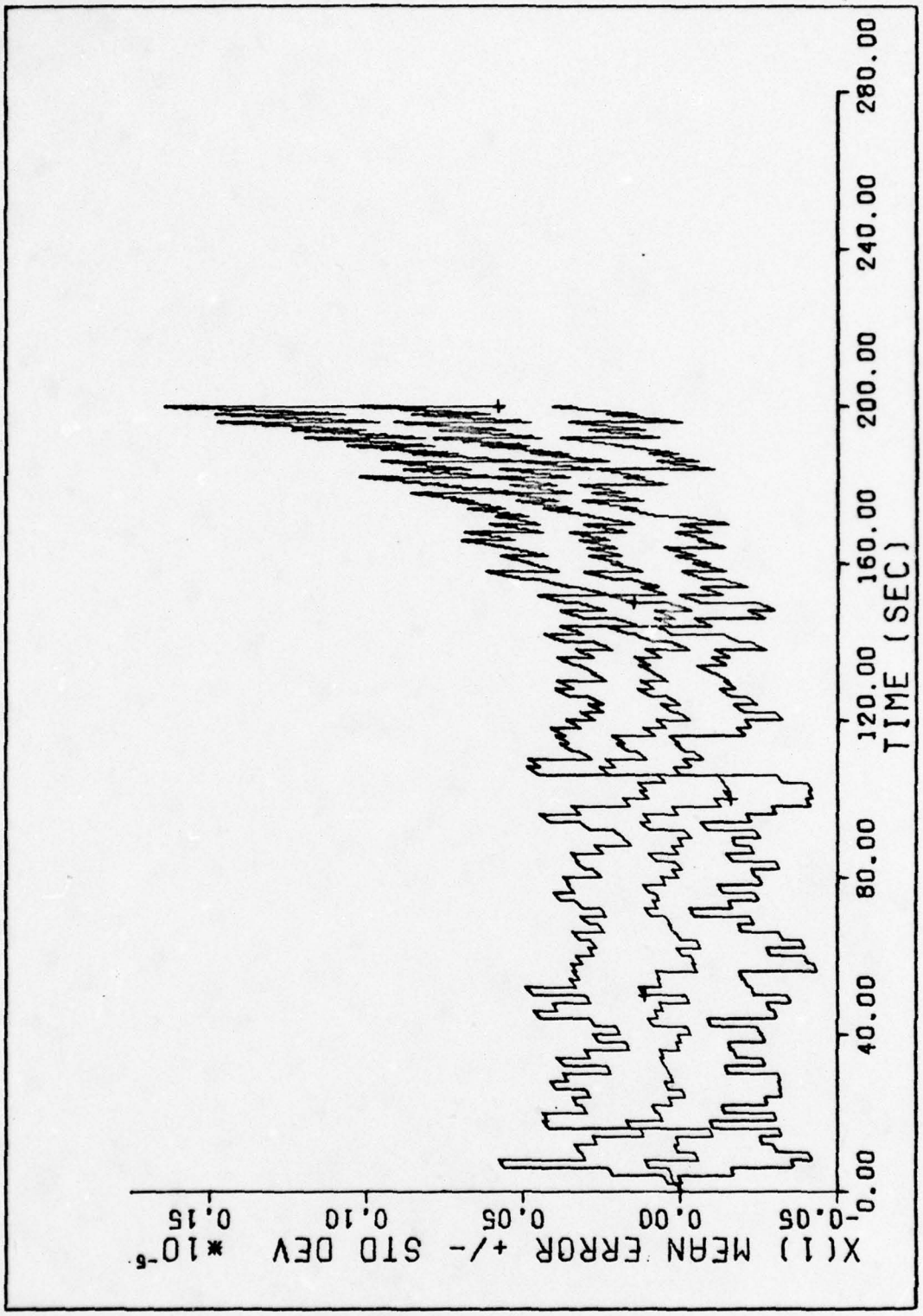


Figure 47. Case 5 - Mean Error +/- s_e vs Time

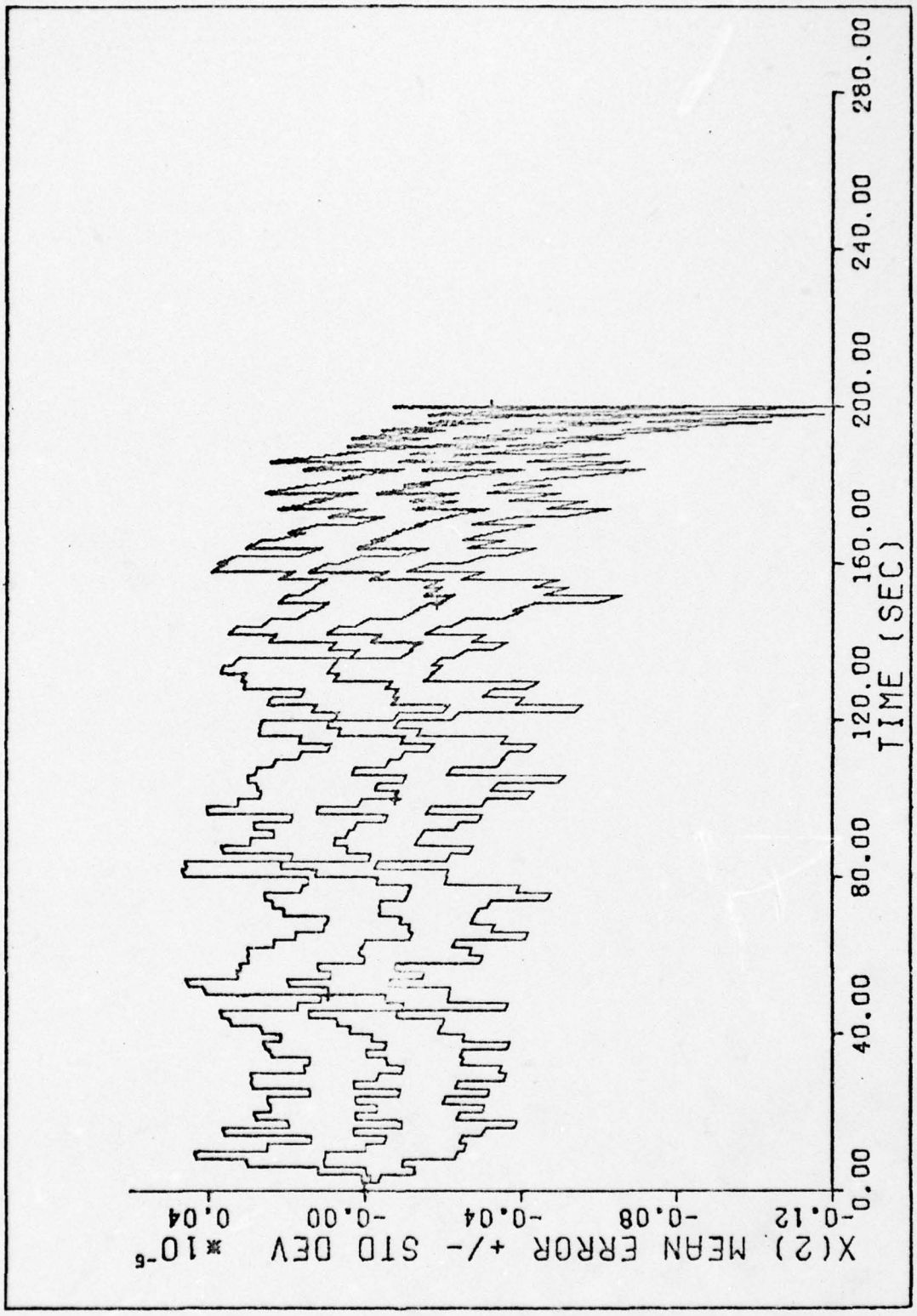


Figure 48. Case 5 - Mean Error +/- s_e vs Time

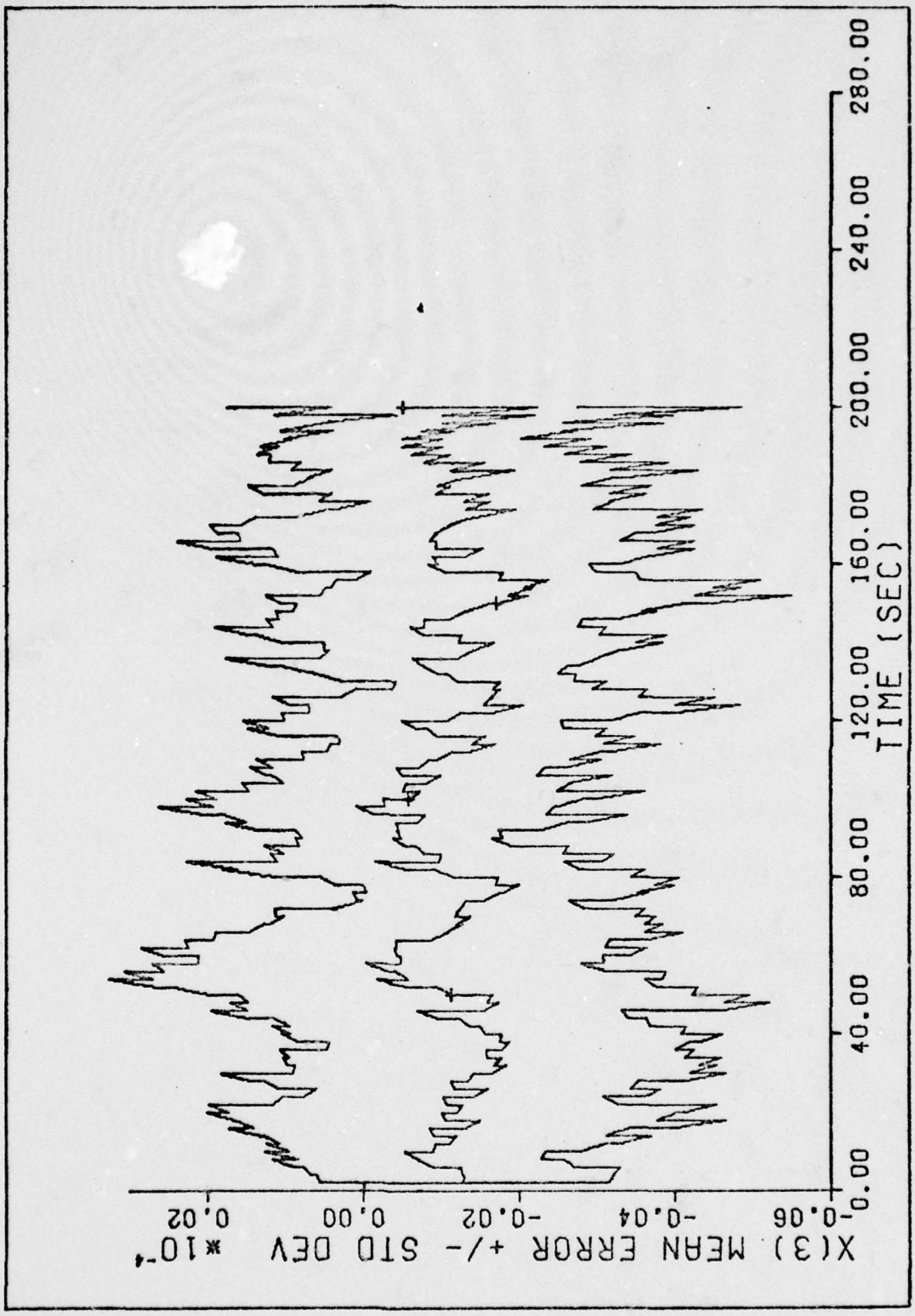


Figure 49. Case 5 - Mean Error +/- s_e vs Time

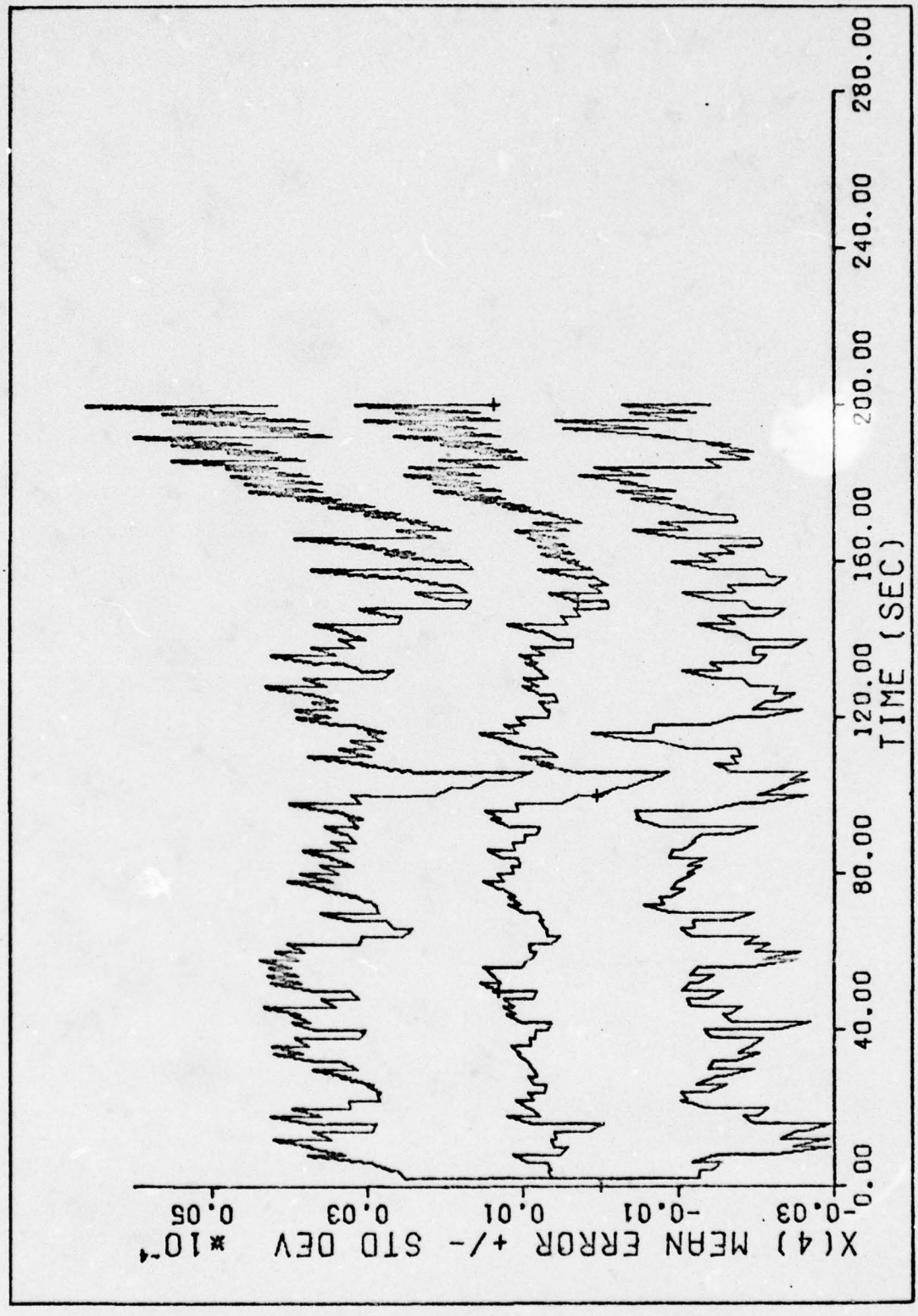


Figure 50. Case 5 - Mean Error +/- s_e vs Time

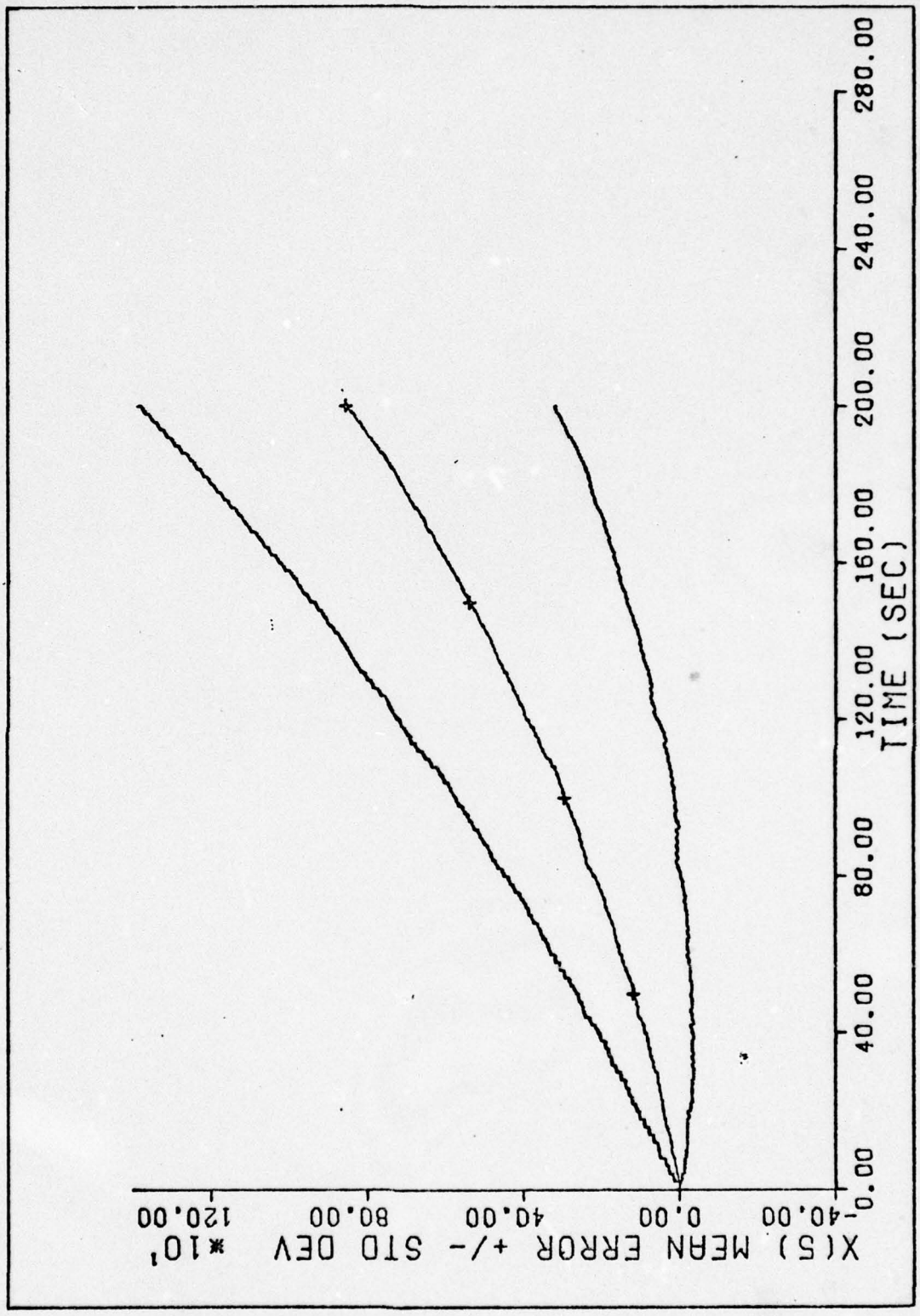


Figure 51. Case 5 - Mean Error +/- s_e vs Time

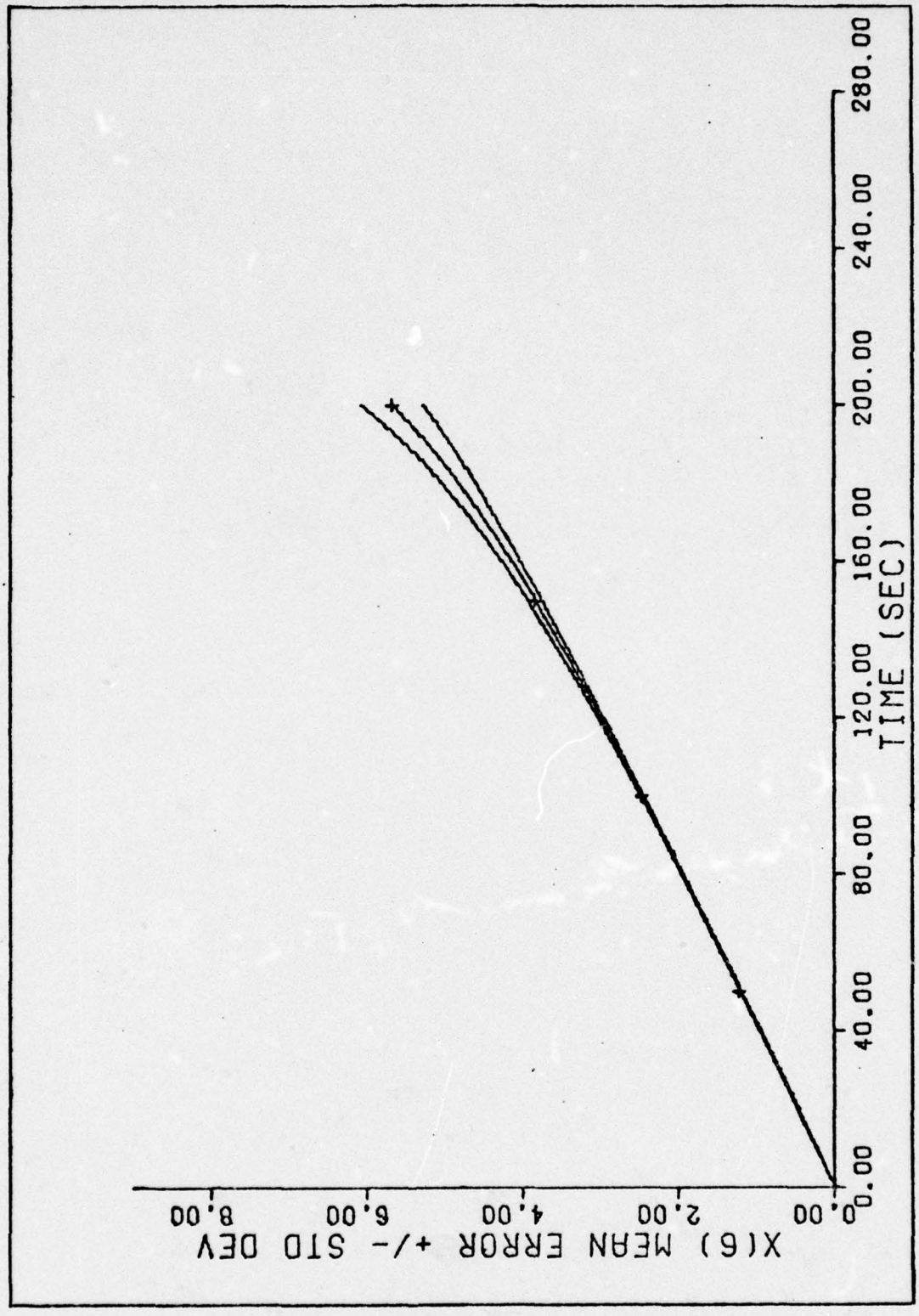


Figure 52. Case 5 - Mean Error +/- s_e vs Time

Case 6 was performed by increasing R_5 so that the error in state x(5) would vary less before and after measurement update. All standard deviation and mean plots changed significantly and are presented in Figures 53 through 64. Most noticeable is the decrease in the true error standard deviation on all plots, some by as much as two orders of magnitude.

In order to reduce the filter standard deviation estimate for state x(5) Q_4 was decreased for case 7. The standard deviation for states x(5) and x(6) changed and are presented in Figures 65 and 66. The most noticeable change in Figure 65 is the decrease in the variations of the filter estimates of the standard deviation before and after the measurement update. In Figure 66 the standard deviation for the true error decreased. Figures 67 through 72 present the mean error plots.

Case 8 was performed by decreasing the value of Q_3 and R_5 so that the filter standard deviation estimate for state x(5) would be reduced further. The filter estimate of the standard deviation for both states x(5) and x(6) decreased. However for both states, the true error standard deviation increased significantly as noted in Figures 73 and 74. The mean error plots are presented in Figures 75 through 80.

Case 8 yielded too much reduction in the filter standard deviation estimate for state x(5) so R_5 was increased by an order of magnitude to perform case 9. The filter estimate of the standard deviation for state x(5) increased slightly, however, the true error standard deviation decreased by a factor of 1/2, as shown in Figure 81. For state

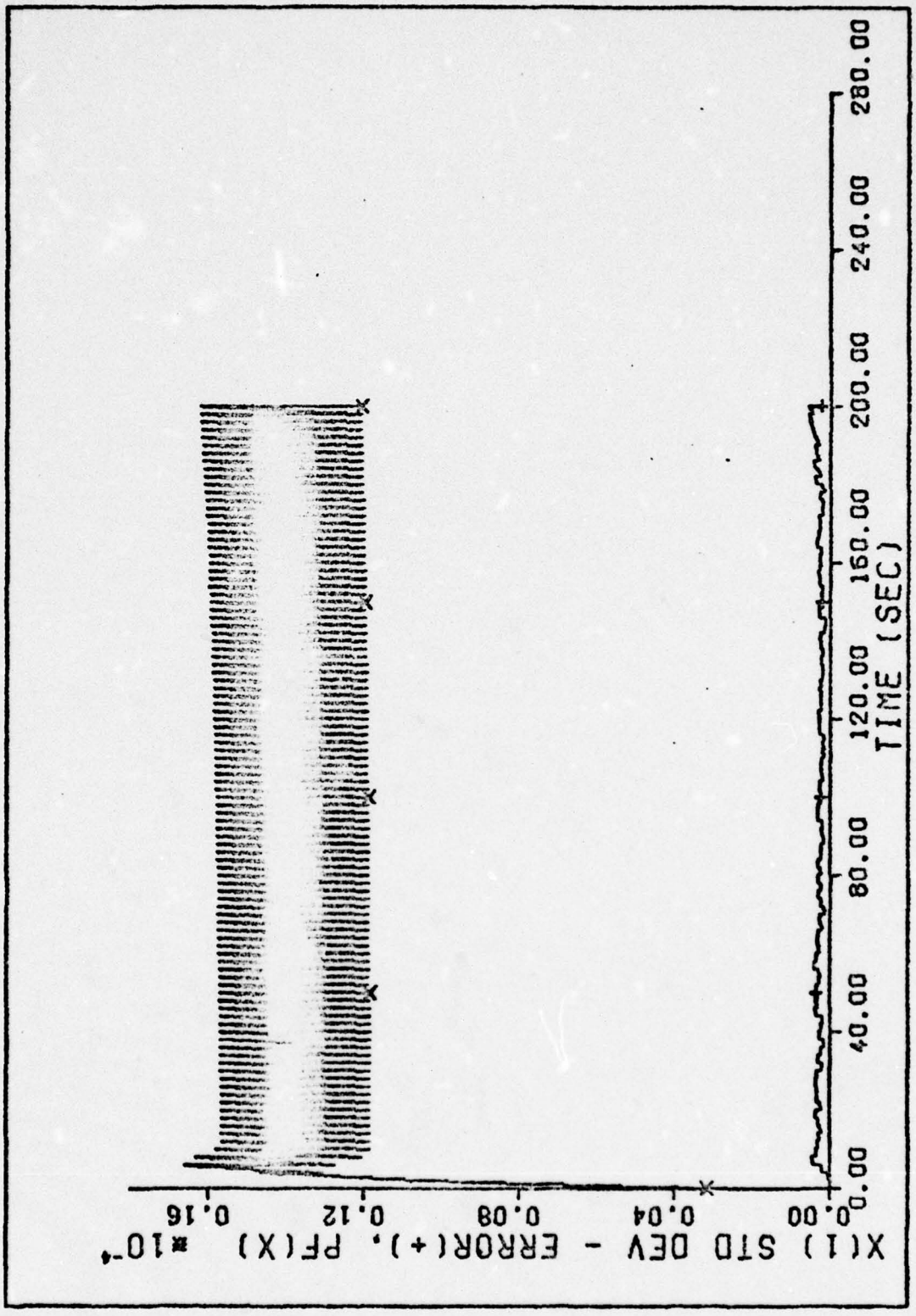


Figure 53. Case 6 - s_e vs Time

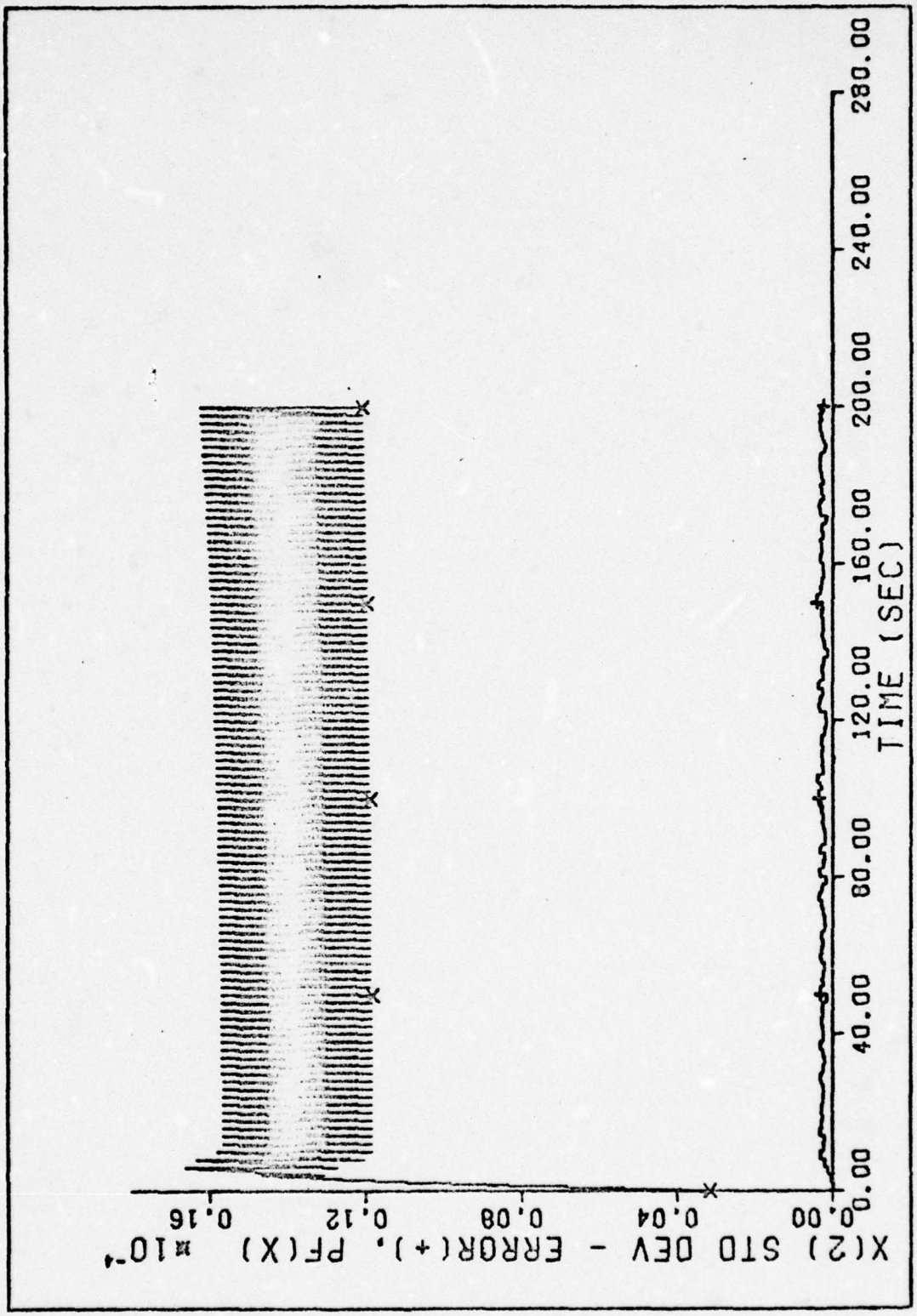


Figure 54. Case 6 - s_e vs Time

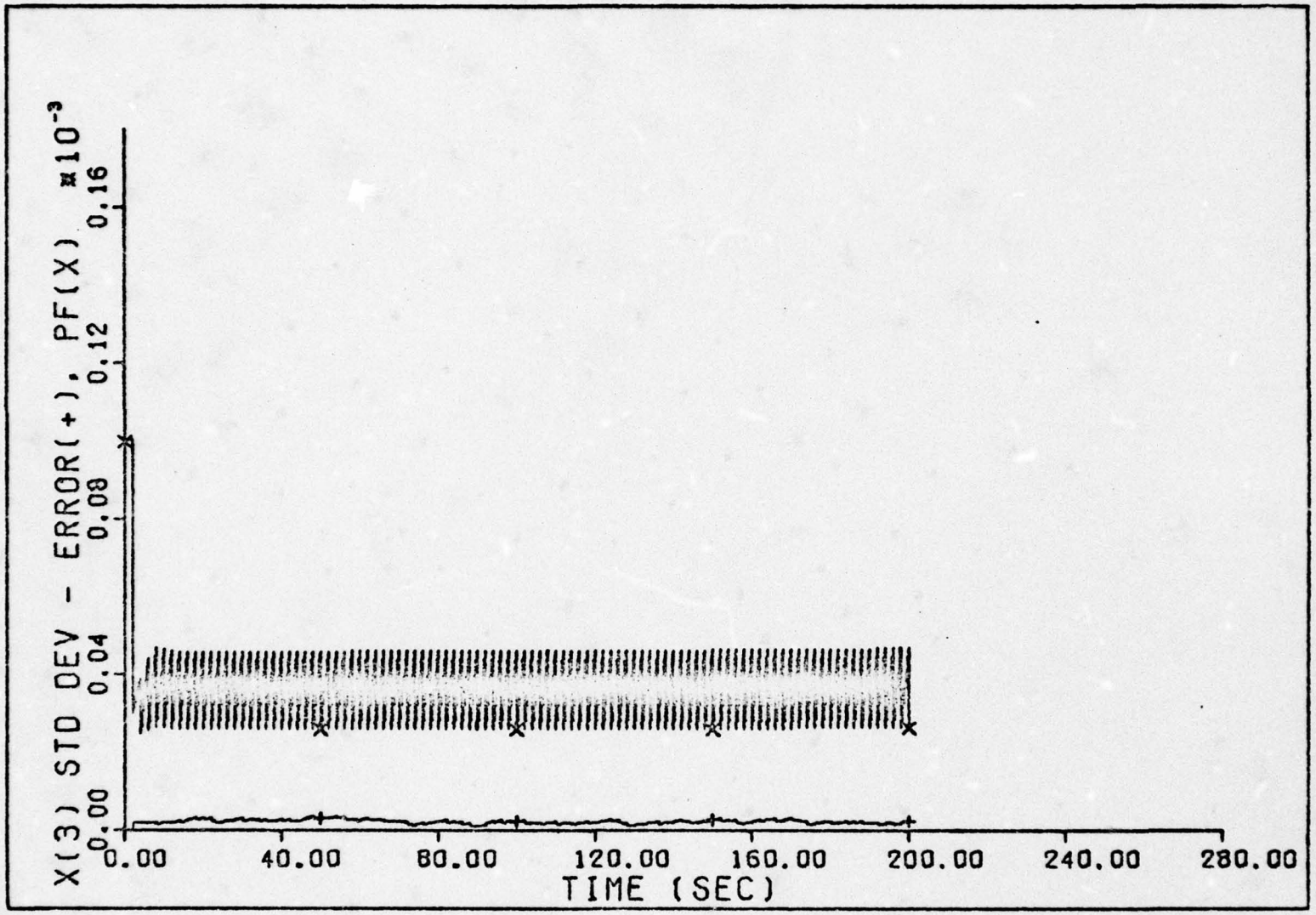


Figure 55. Case 6 - s_e vs Time

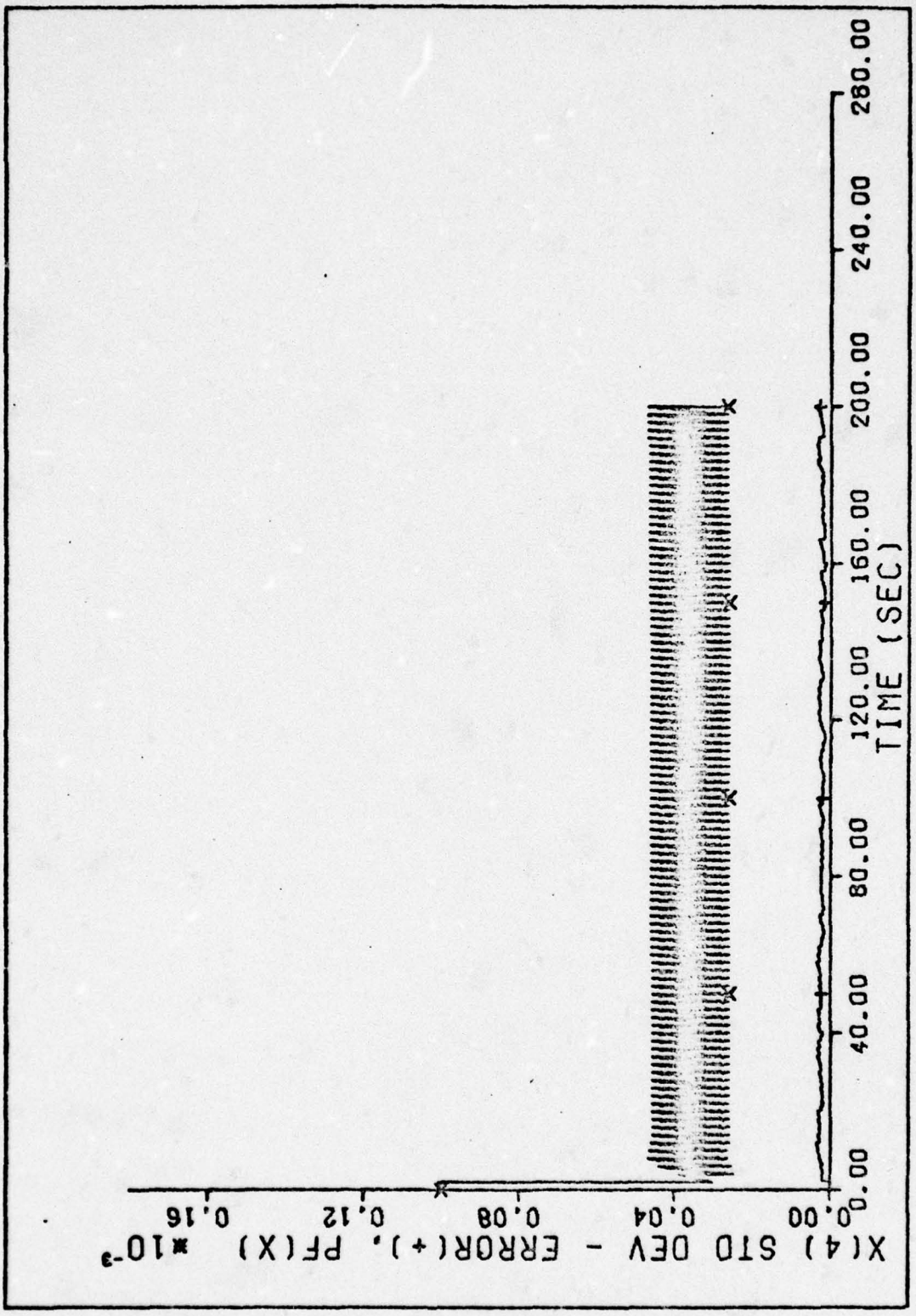


Figure 56. Case 6 - s_e vs Time

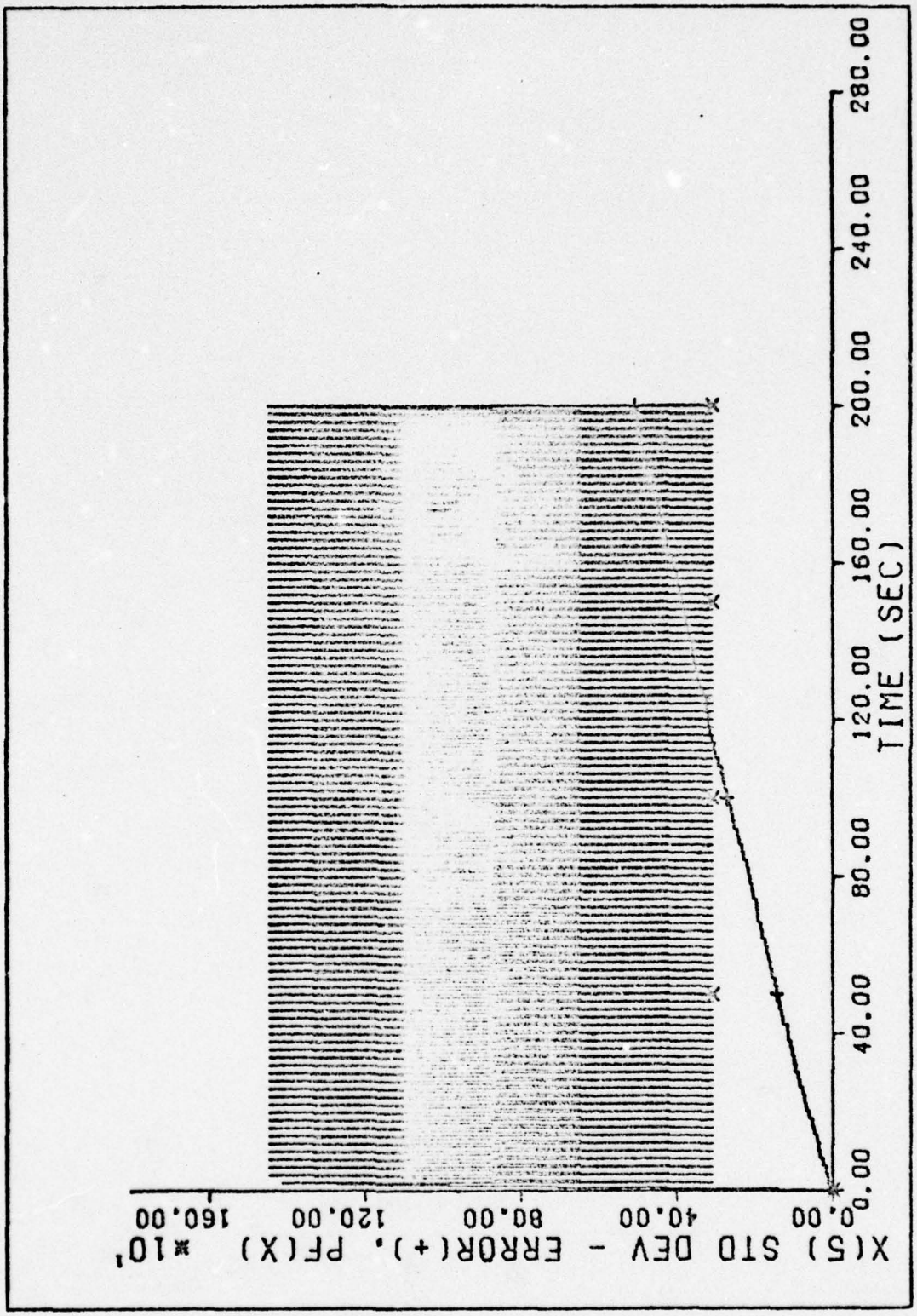


Figure 57. Case 6 - s_e vs Time

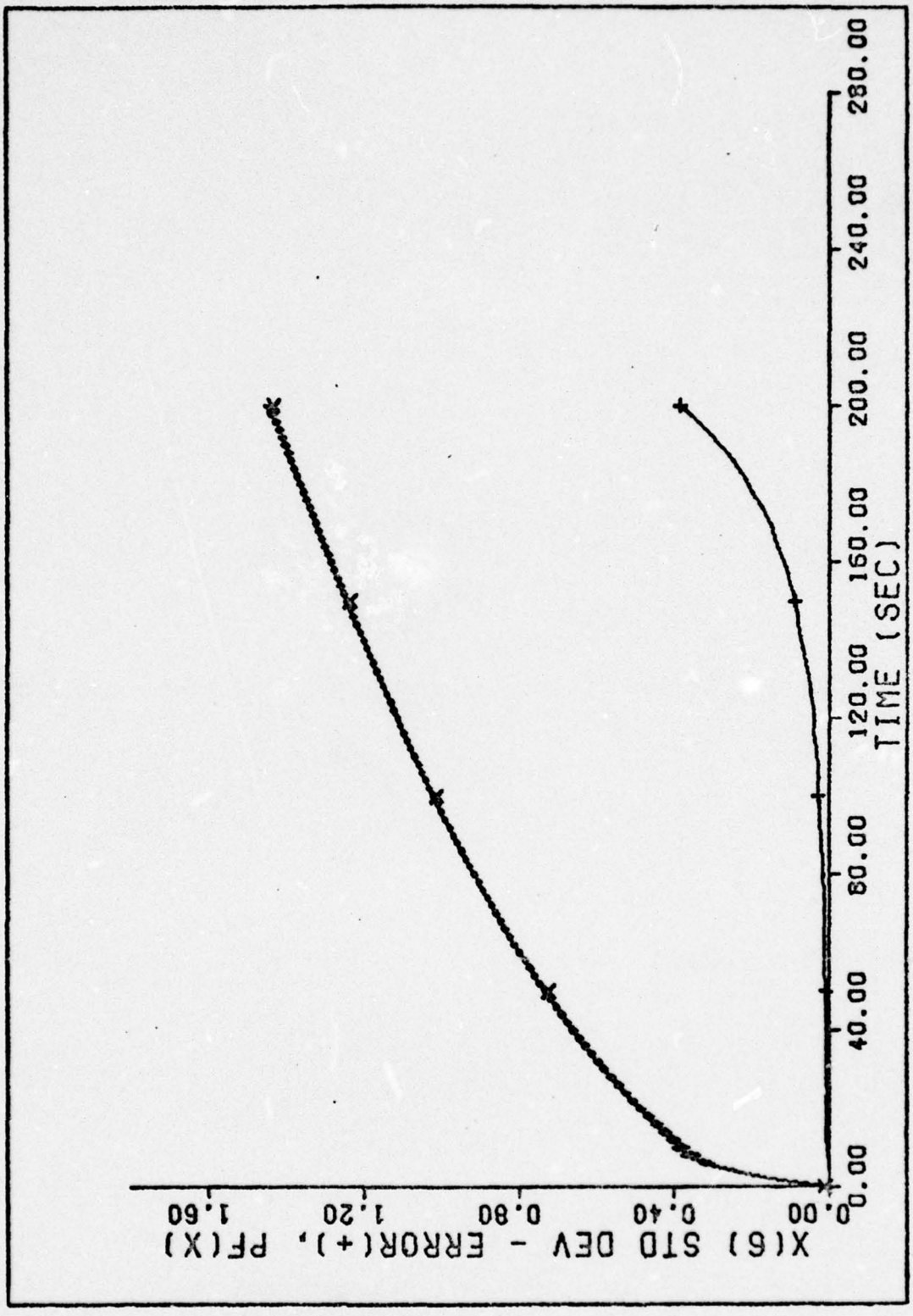


Figure 58. Case 6 - s_e vs Time

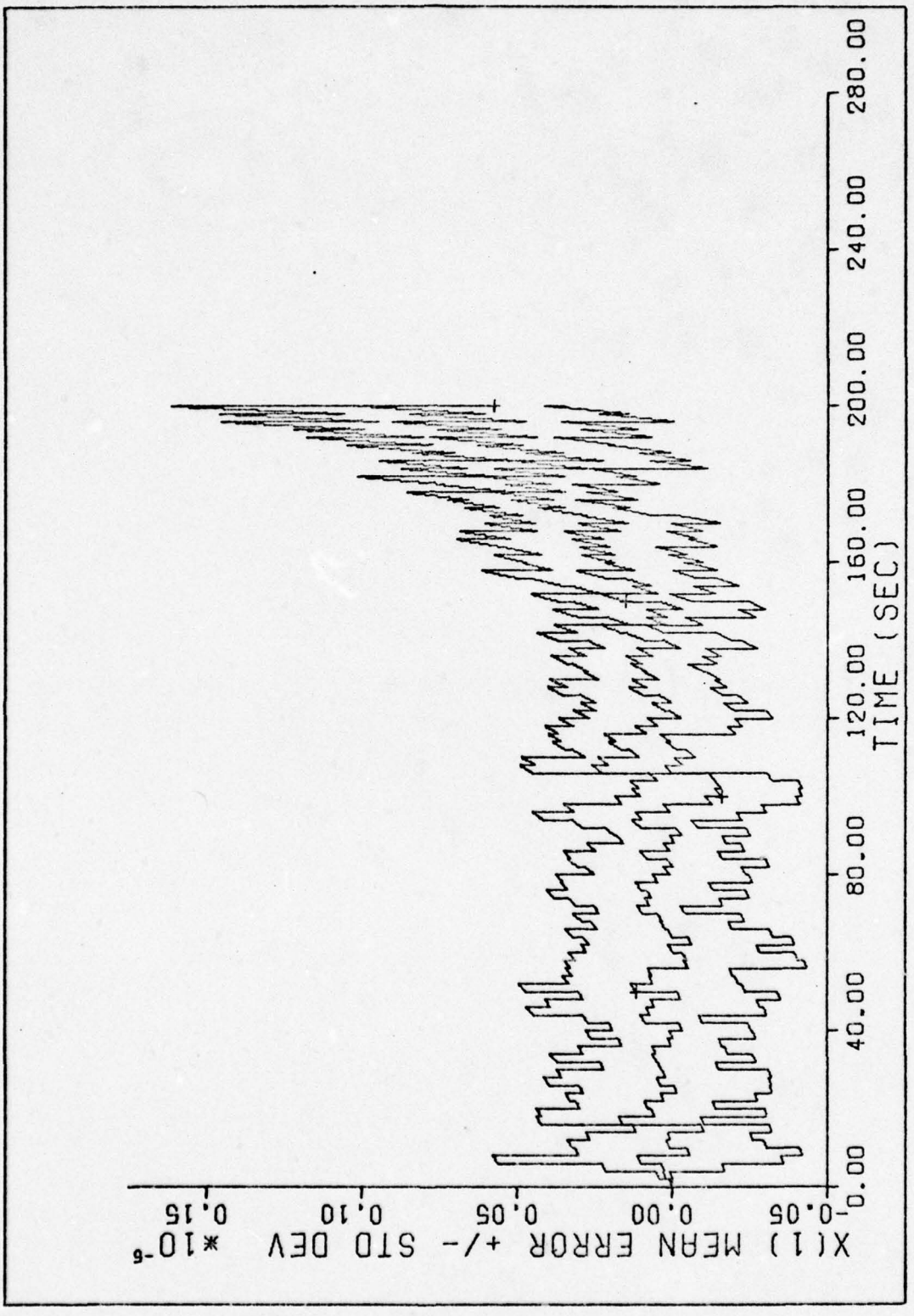


Figure 59. Case 6 - Mean Error +/- s_e vs Time

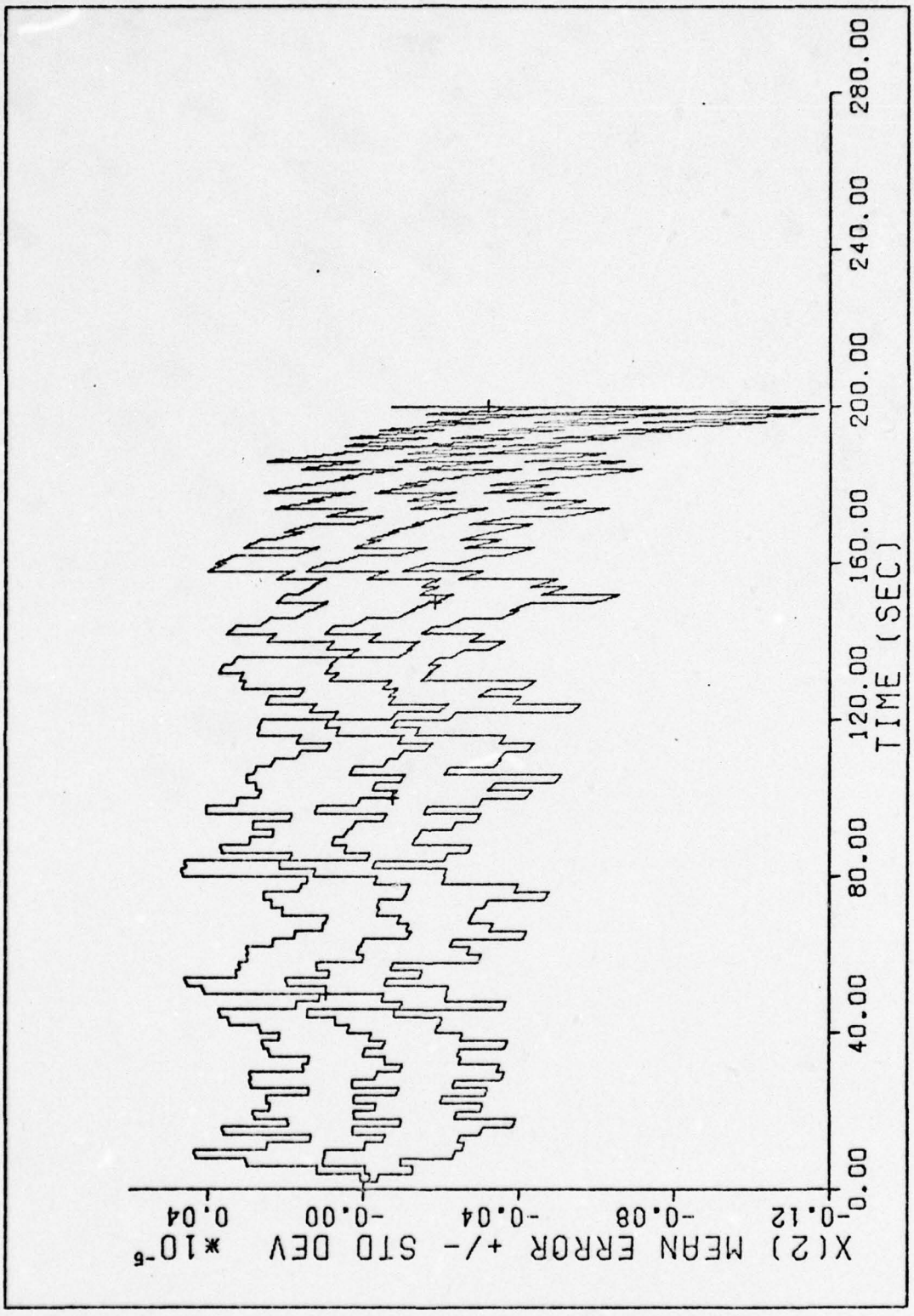


Figure 60. Case 6 - Mean Error +/- s_e vs Time

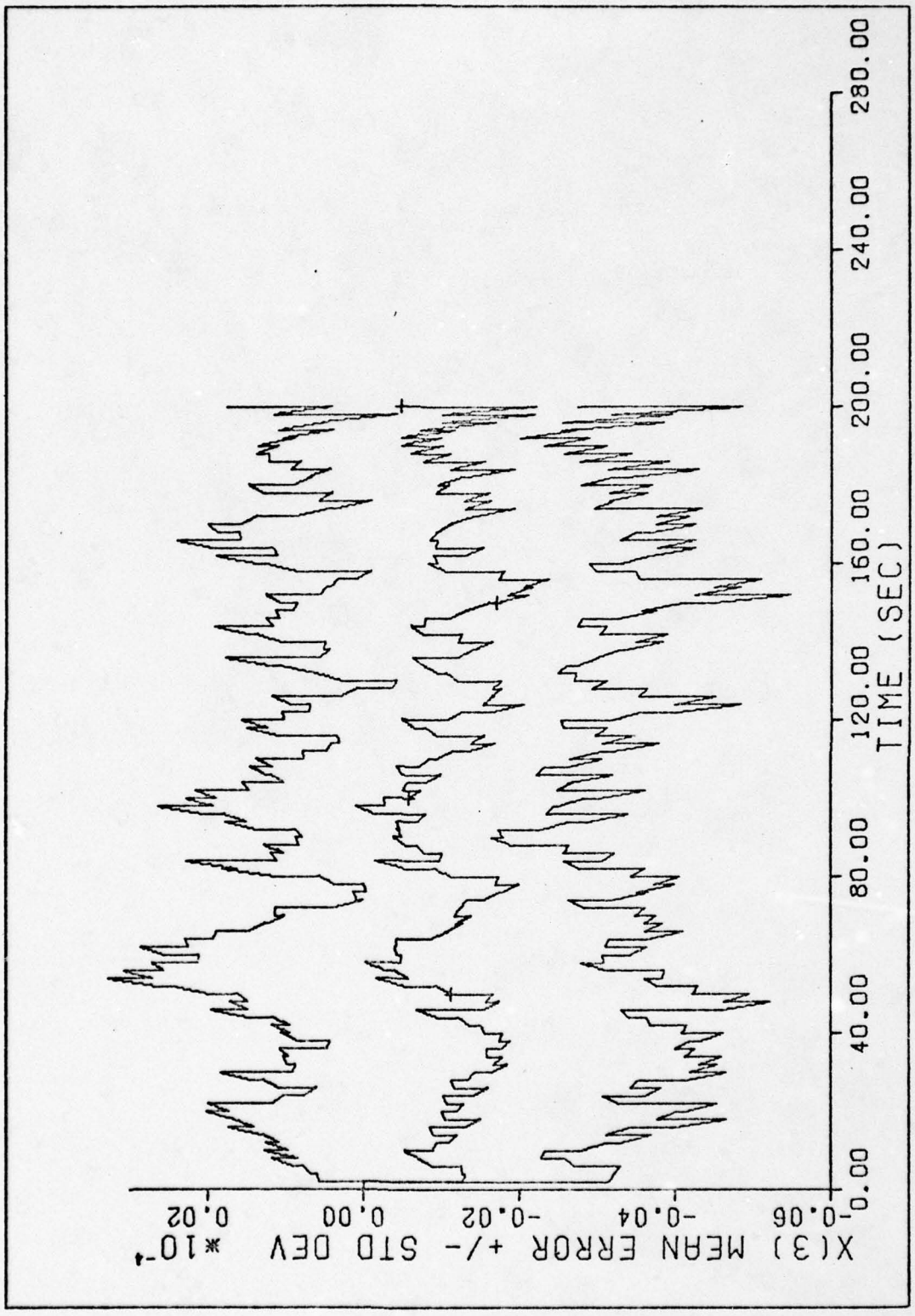


Figure 61. Case 6 - Mean Error +/- s_e vs Time

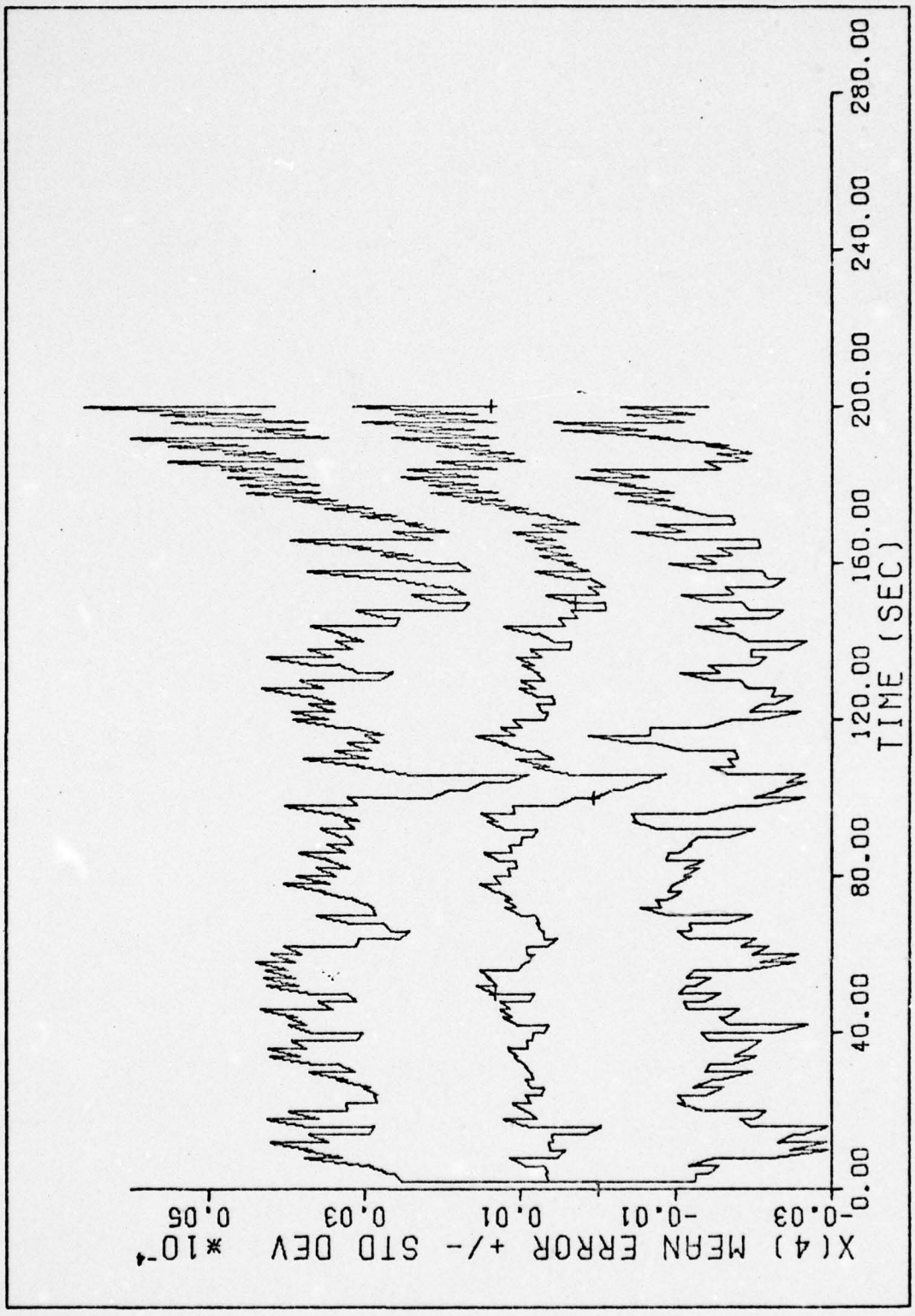


Figure 62. Case 6 - Mean Error +/- s_e vs Time

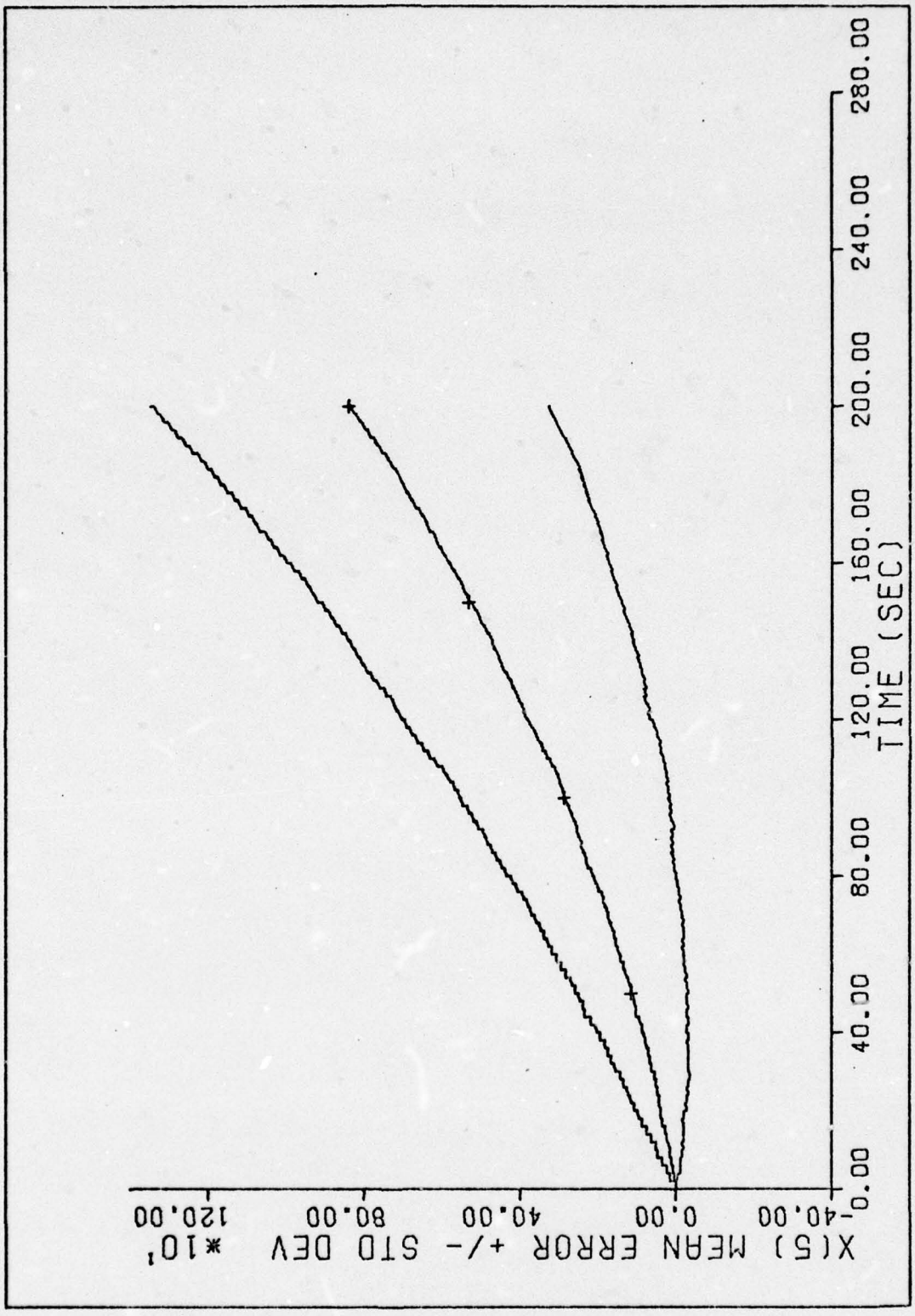


Figure 63. Case 6 - Mean Error +/- s_e vs Time

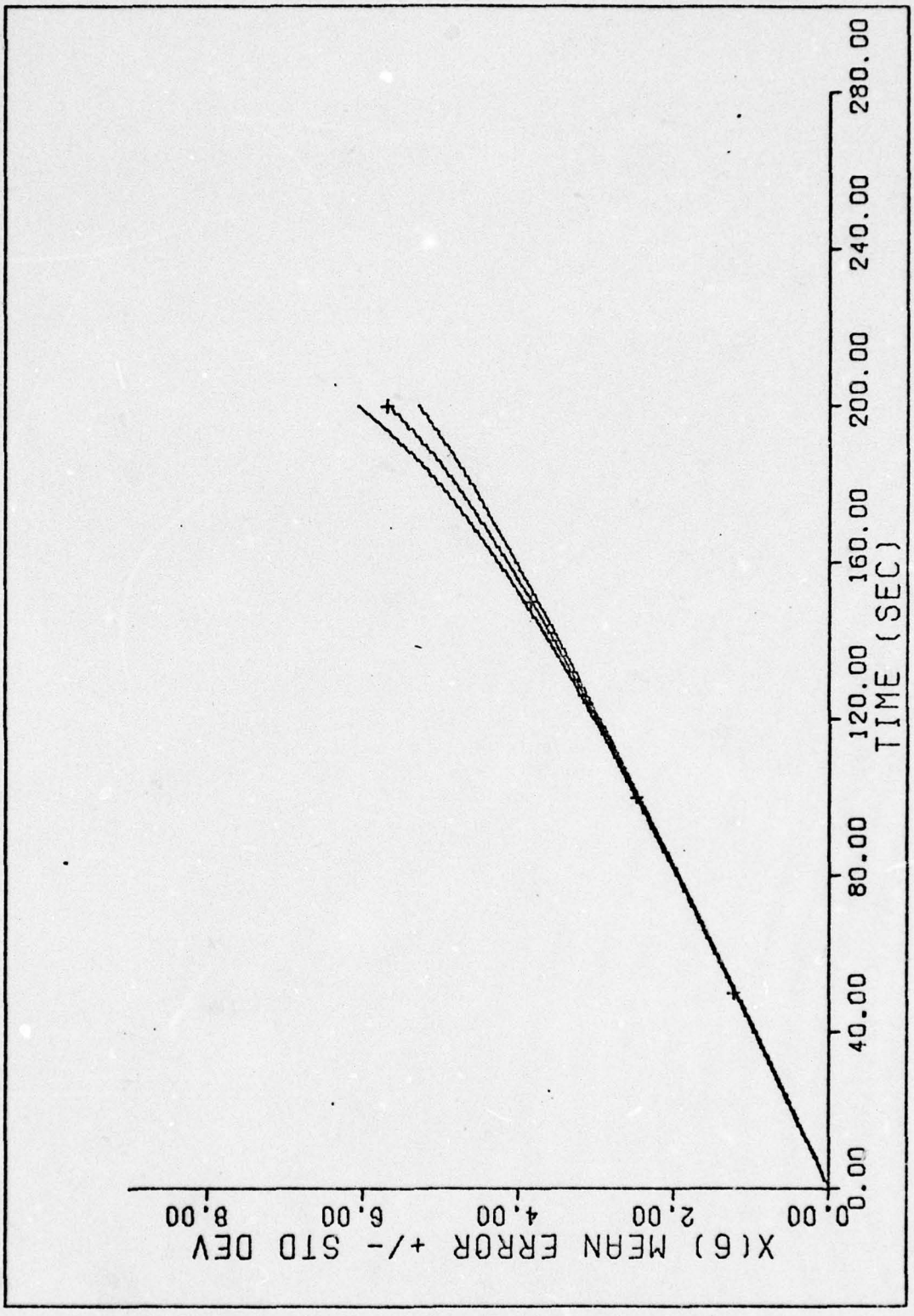


Figure 64. Case 6 - Mean Error +/- s_e vs Time

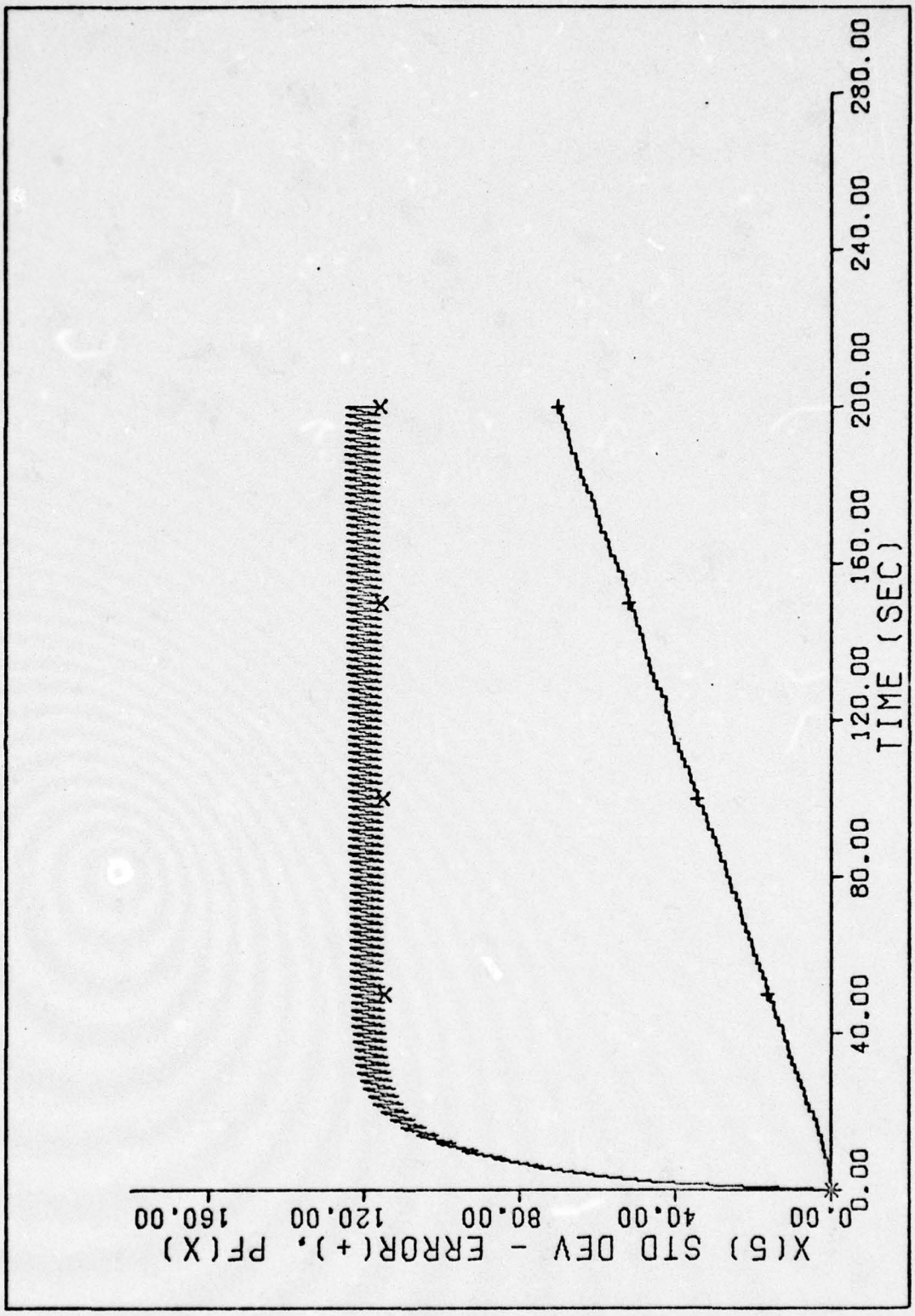


Figure 65. Case 7 - s_e vs Time

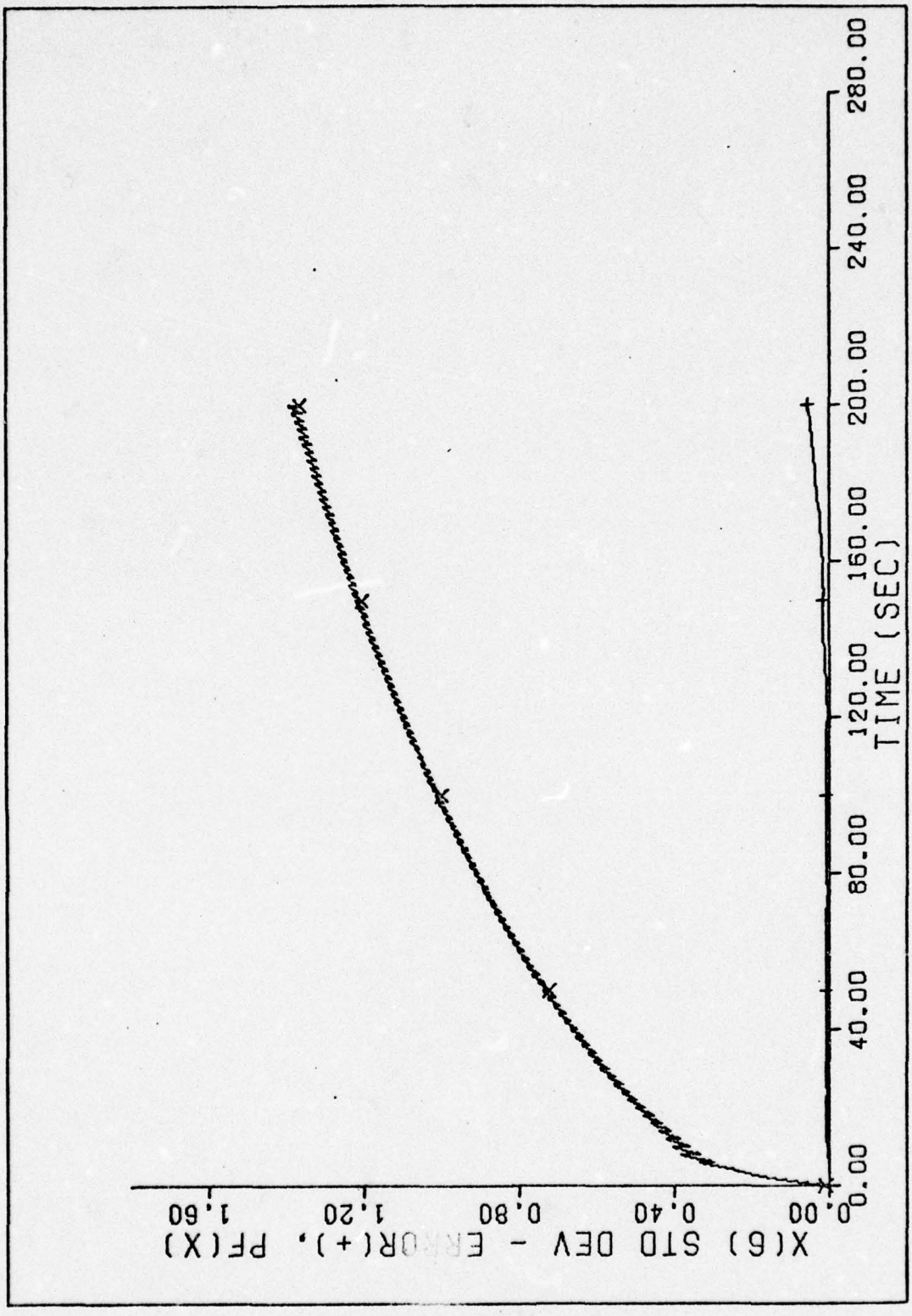


Figure 66. Case 7 - s_e vs Time

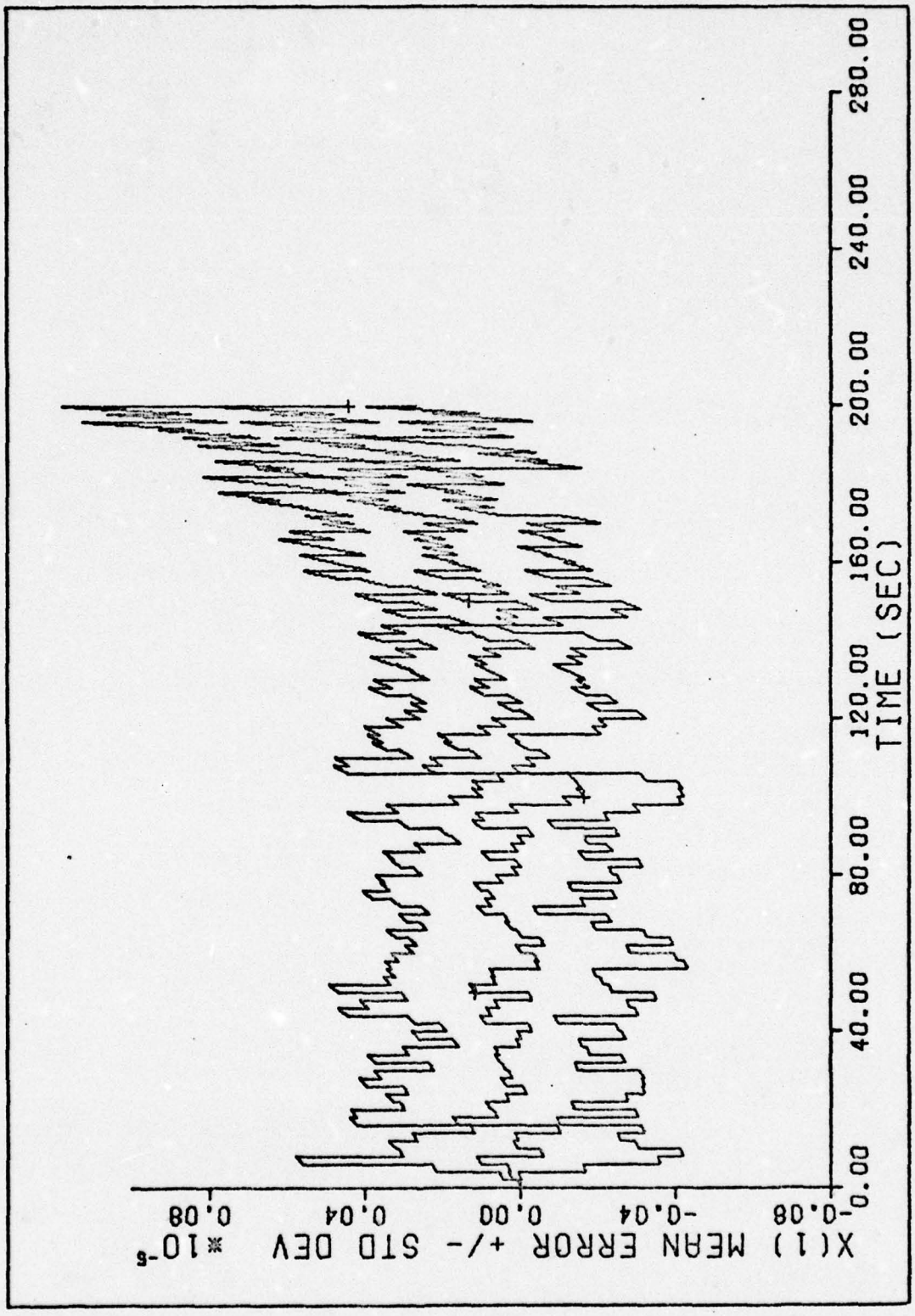


Figure 67. Case 7 - Mean Error +/- s_e vs Time

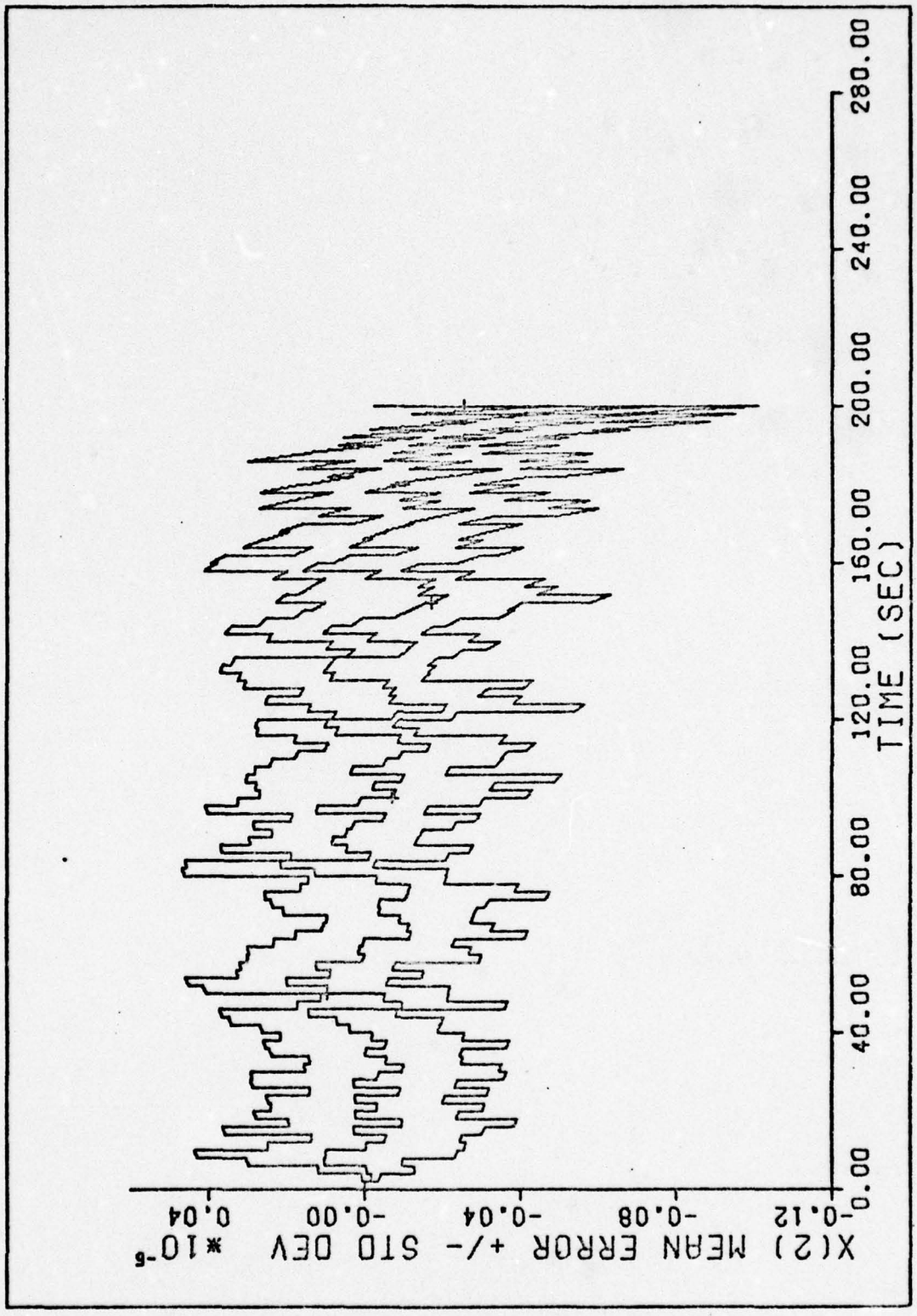


Figure 68. Case 7 - Mean Error +/- s_e vs Time

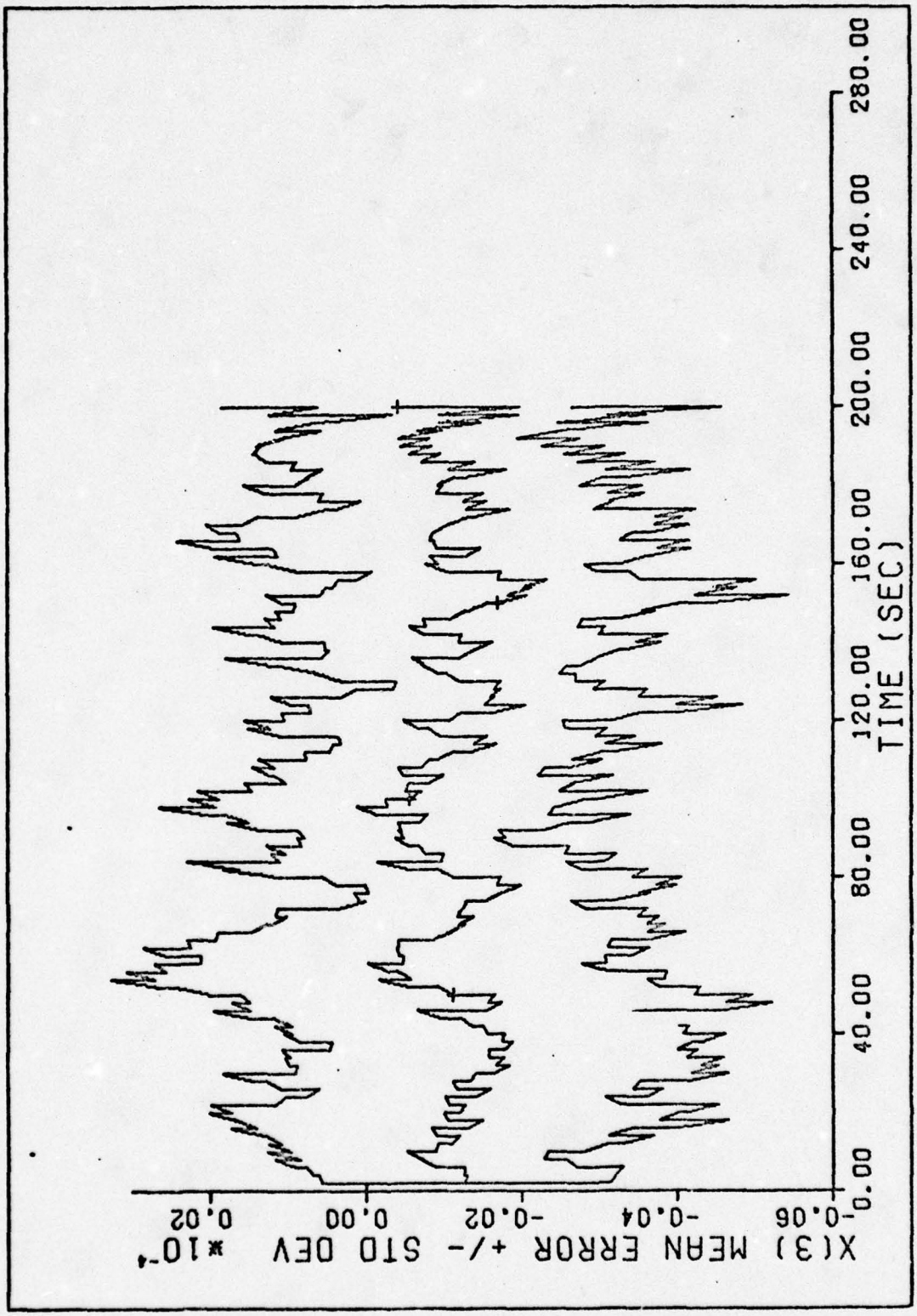


Figure 69. Case 7 - Mean Error +/- s_e vs Time

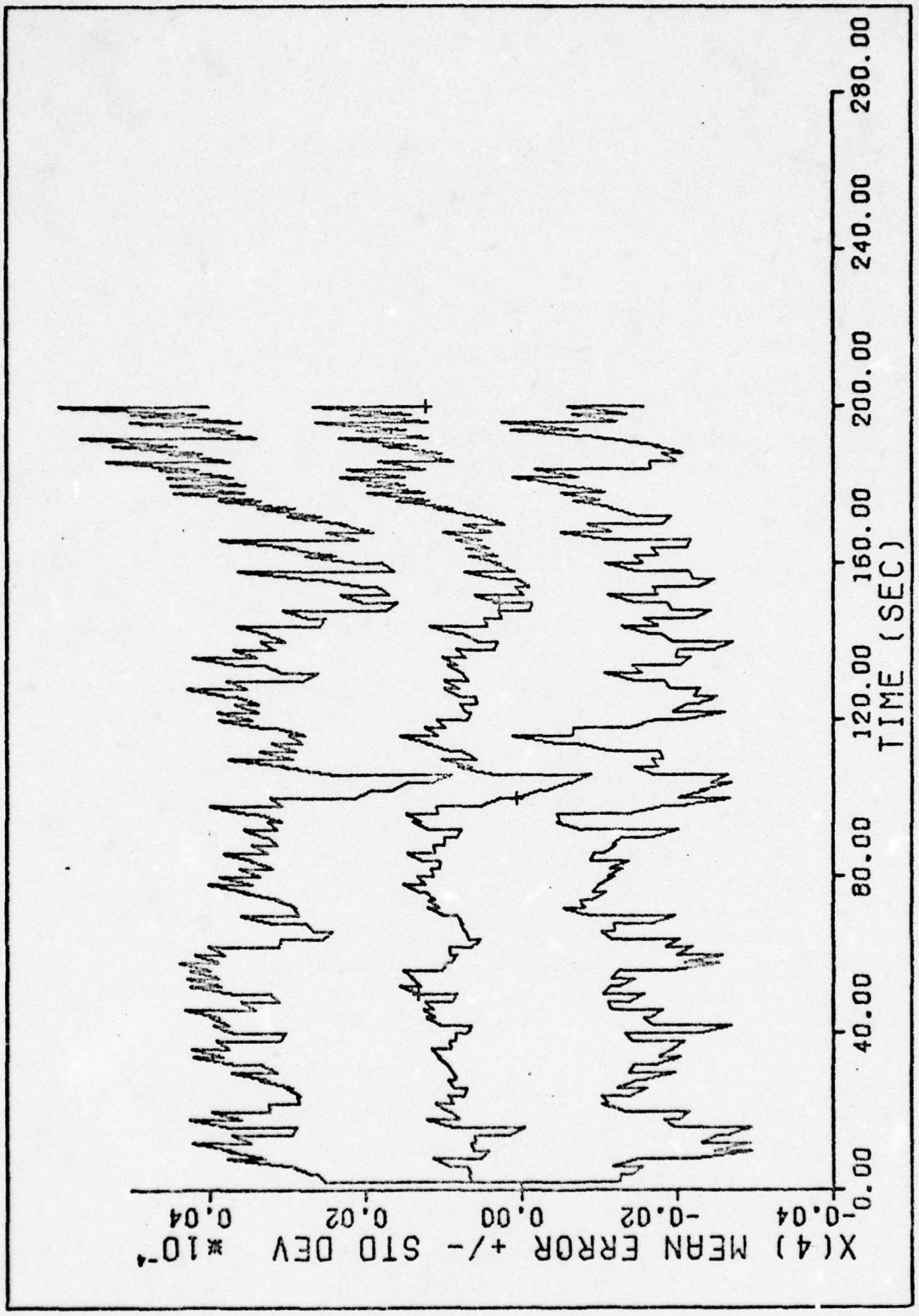


Figure 70. Case 7 - Mean Error +/- s_e vs Time

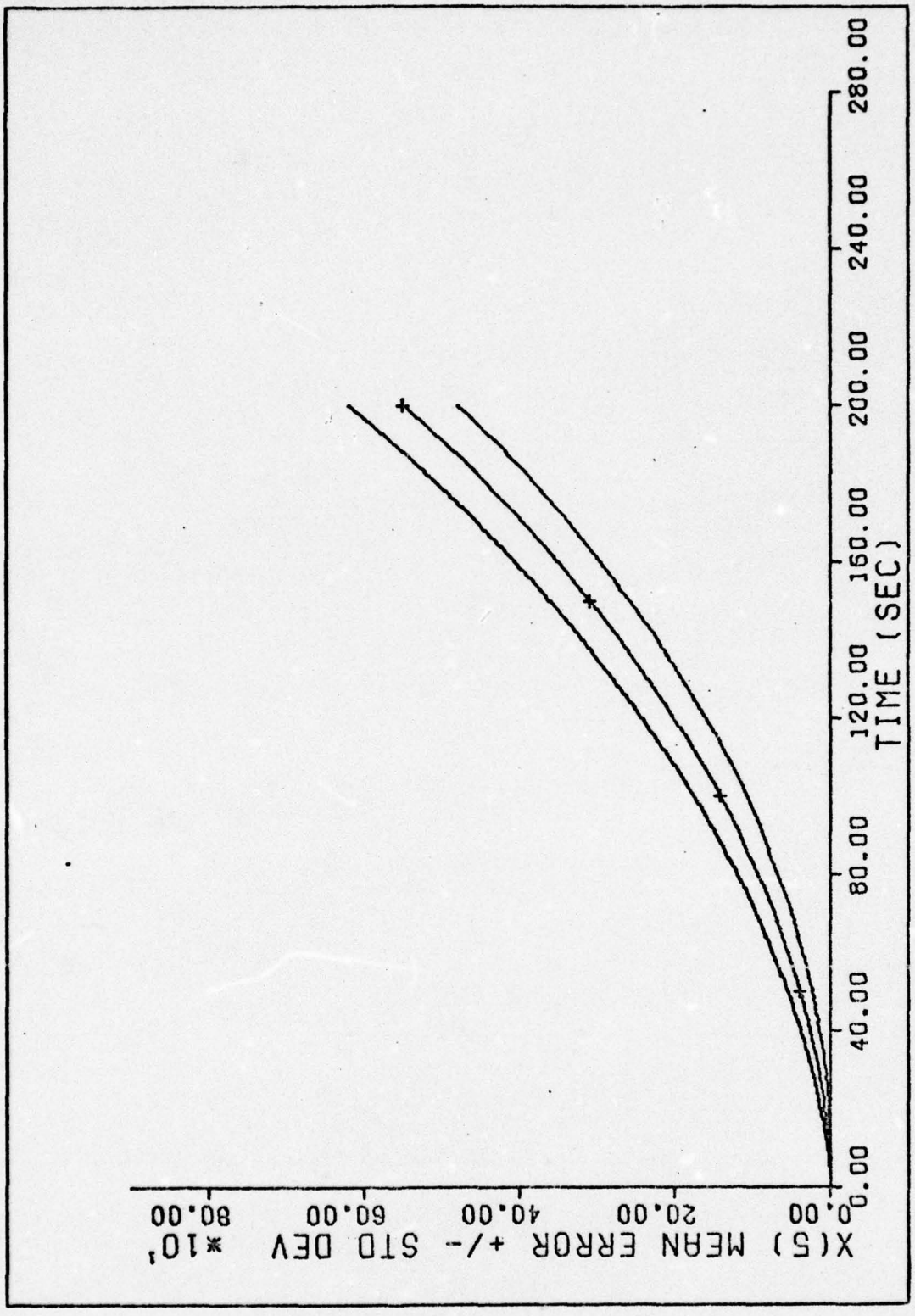


Figure 71. Case 7 - Mean Error +/- s_e vs Time

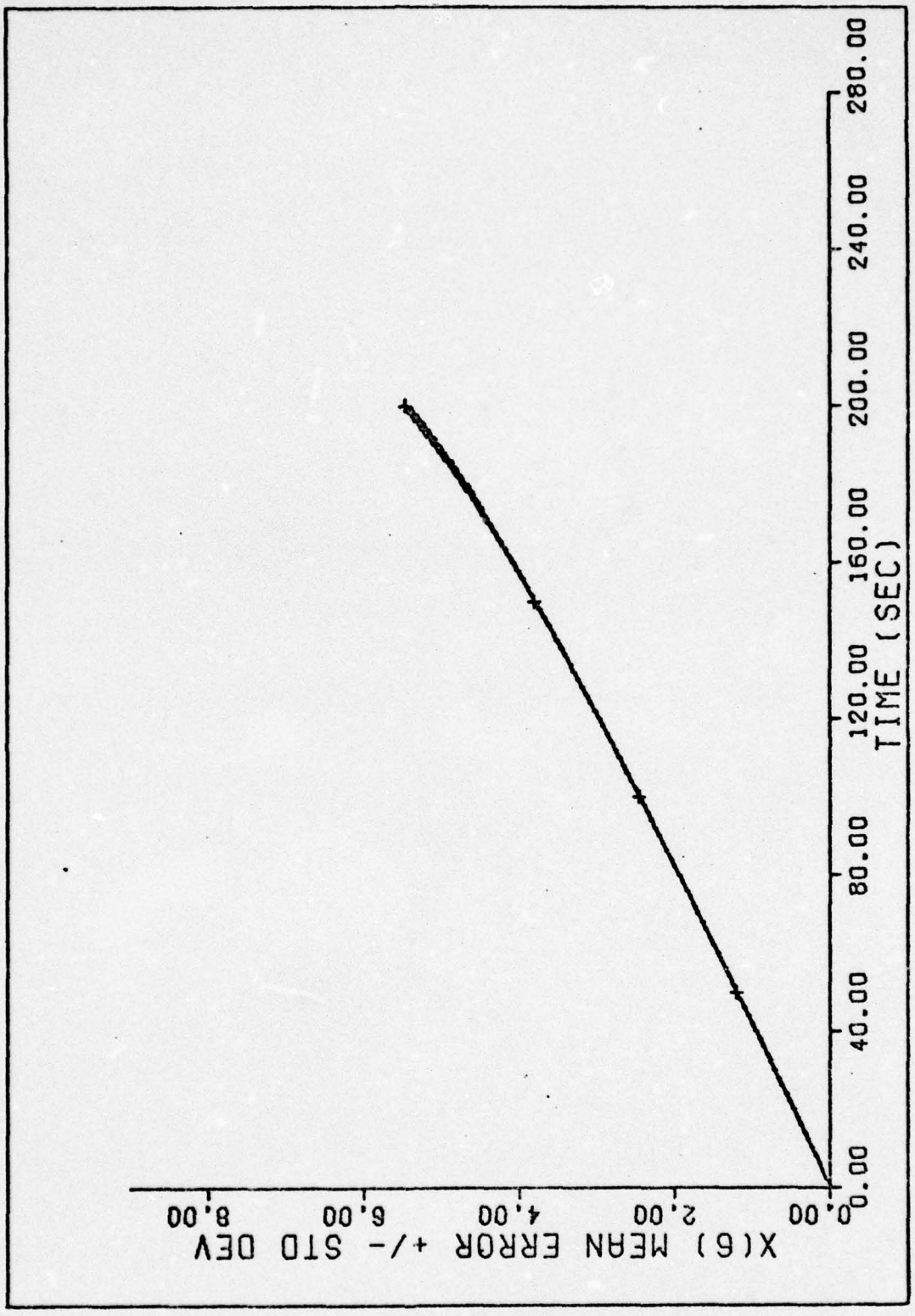


Figure 72. Case 7 - Mean Error +/- s_e vs Time

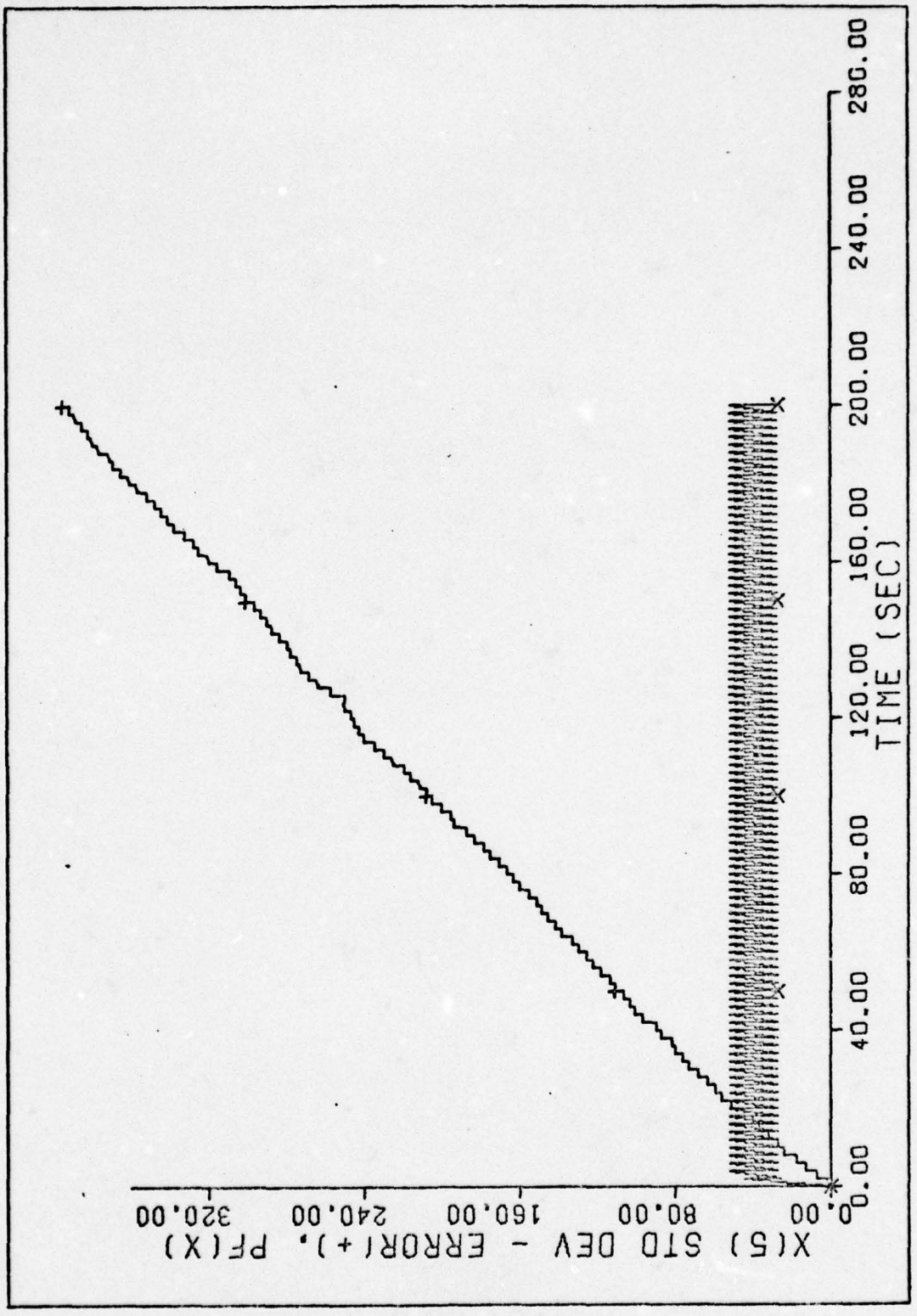


Figure 73. Case 8 - s_e vs Time

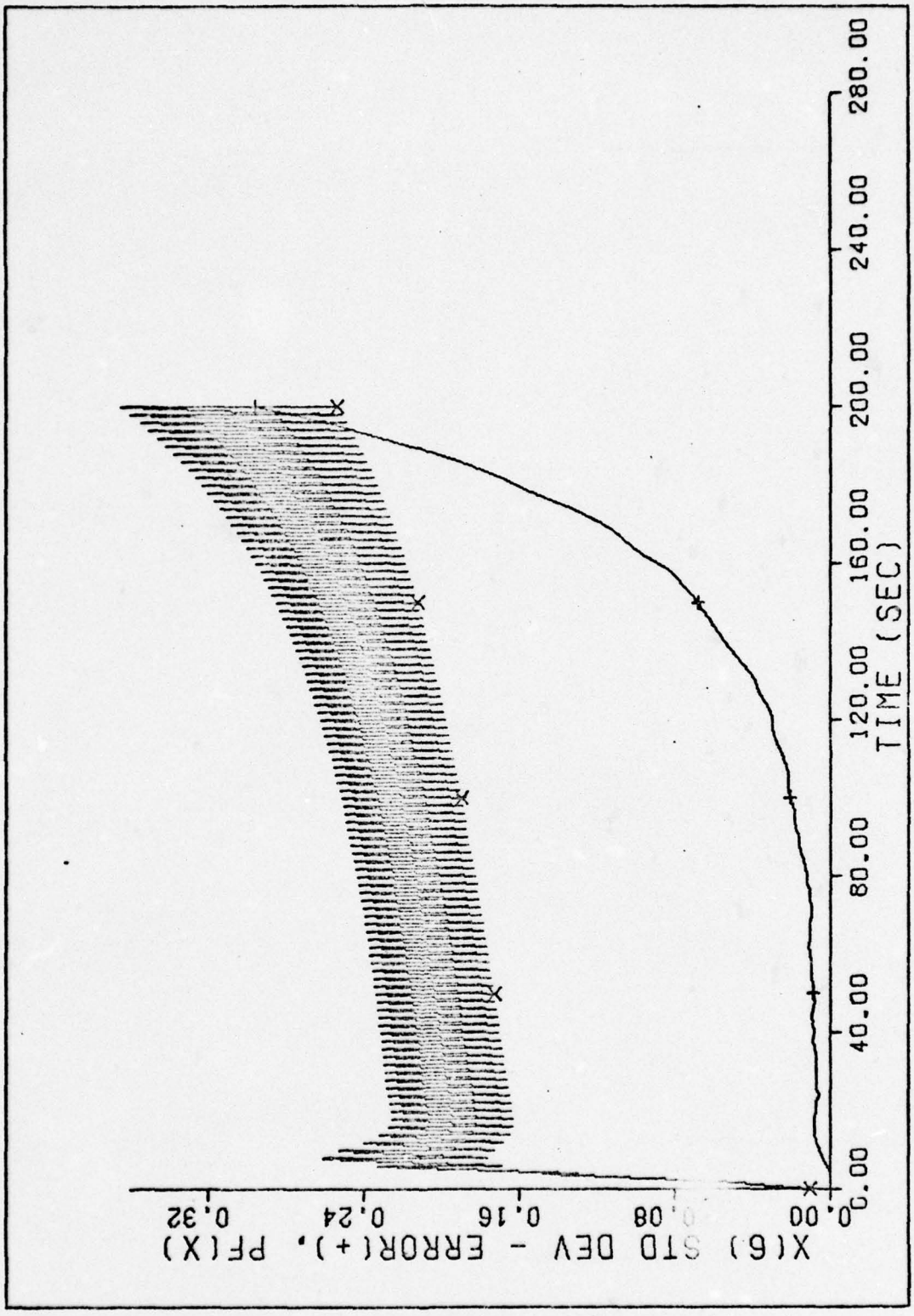


Figure 74. Case 8 - s_e vs Time

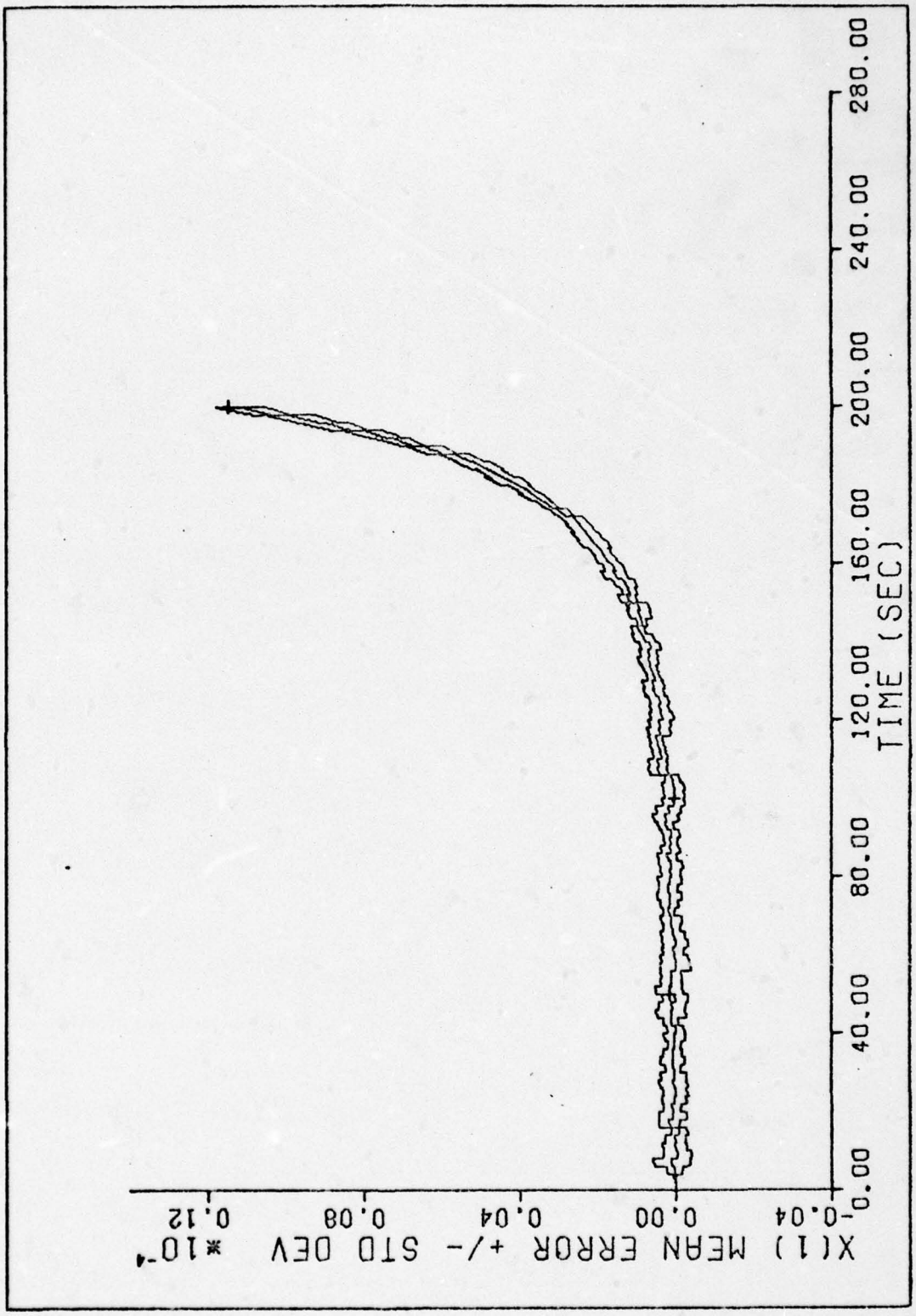


Figure 75. Case 8 - Mean Error +/- s_e vs Time

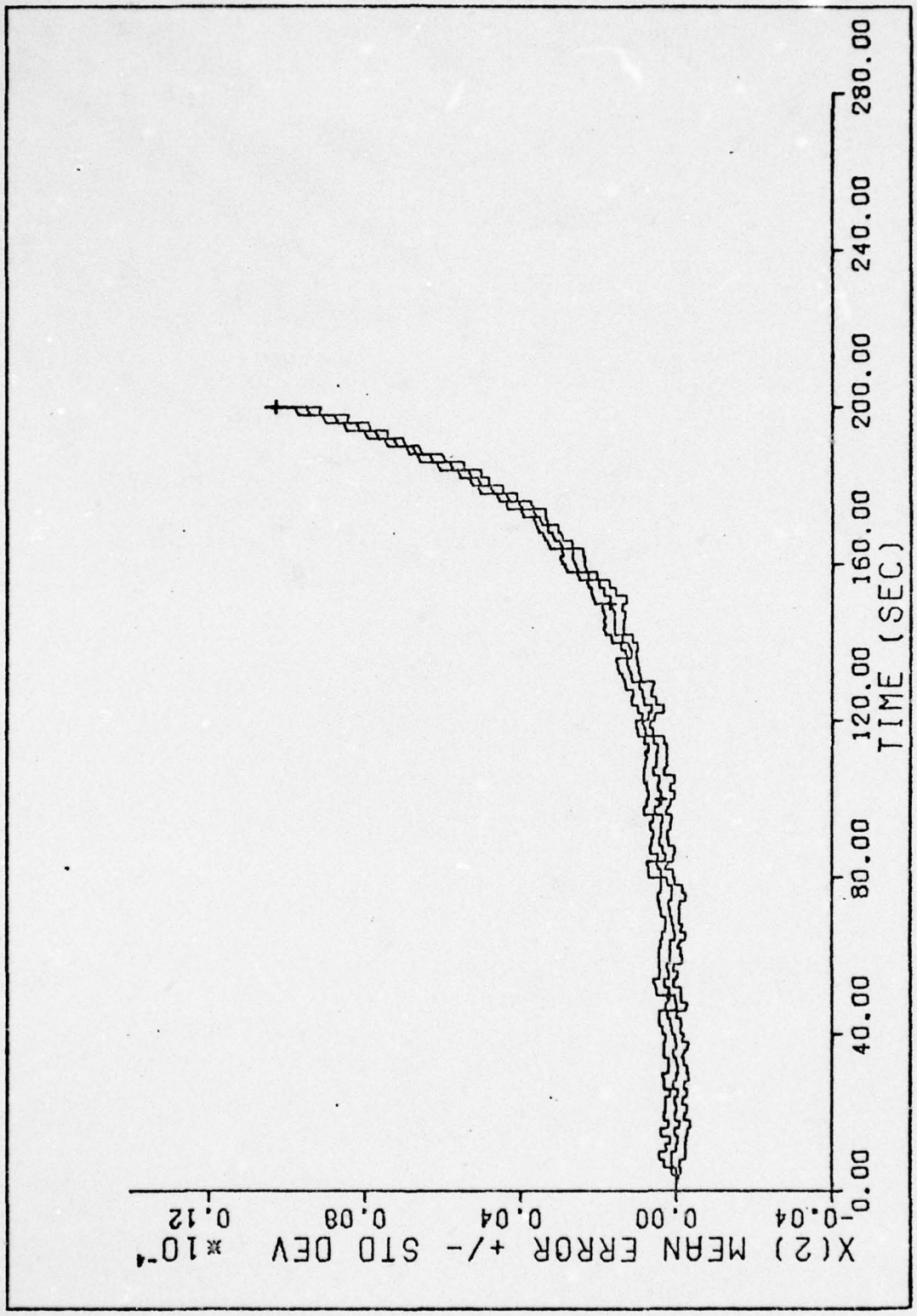


Figure 76. Case 8 - Mean Error +/- s_e vs Time

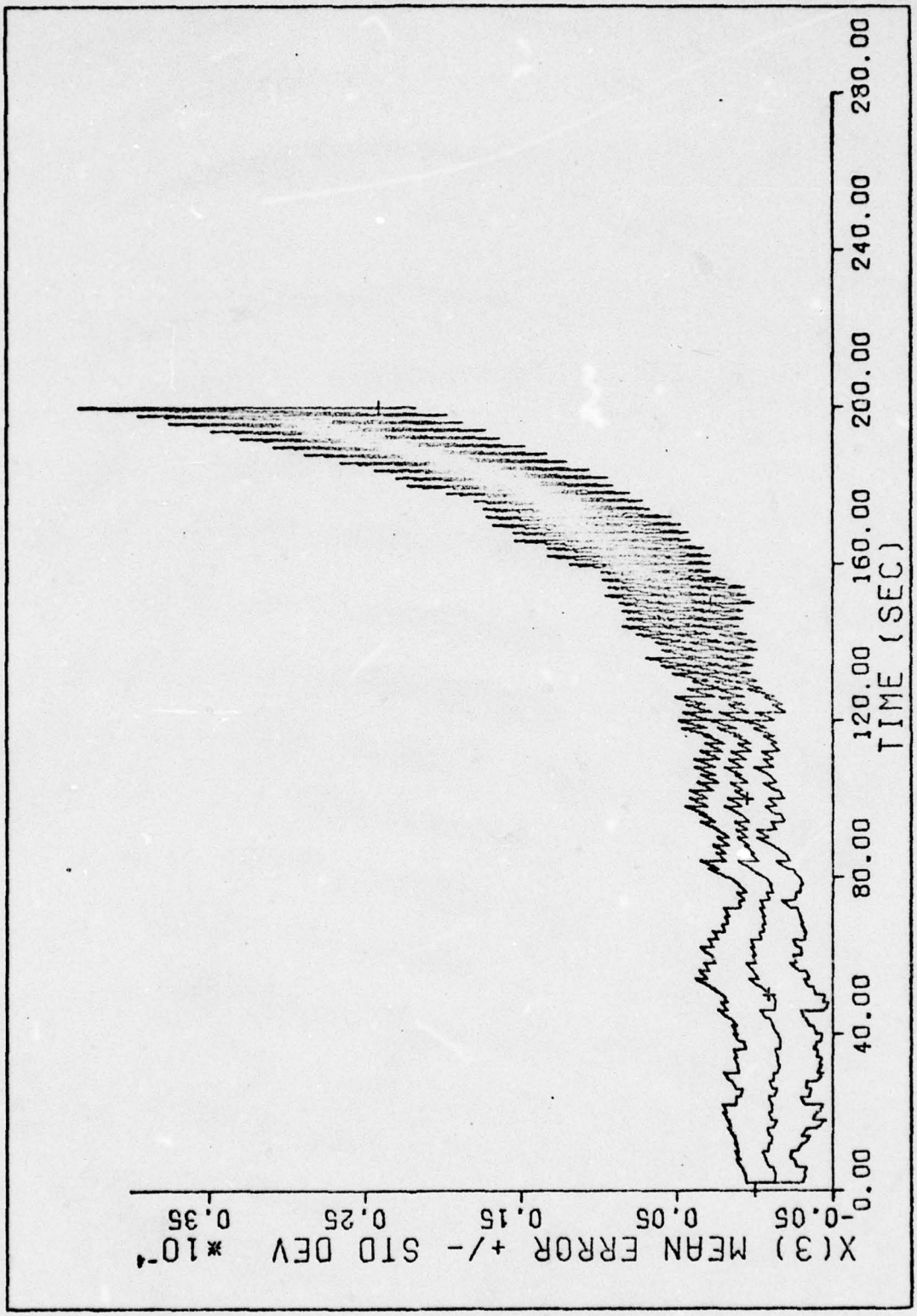


Figure 77. Case 8 - Mean Error +/- s_e vs Time

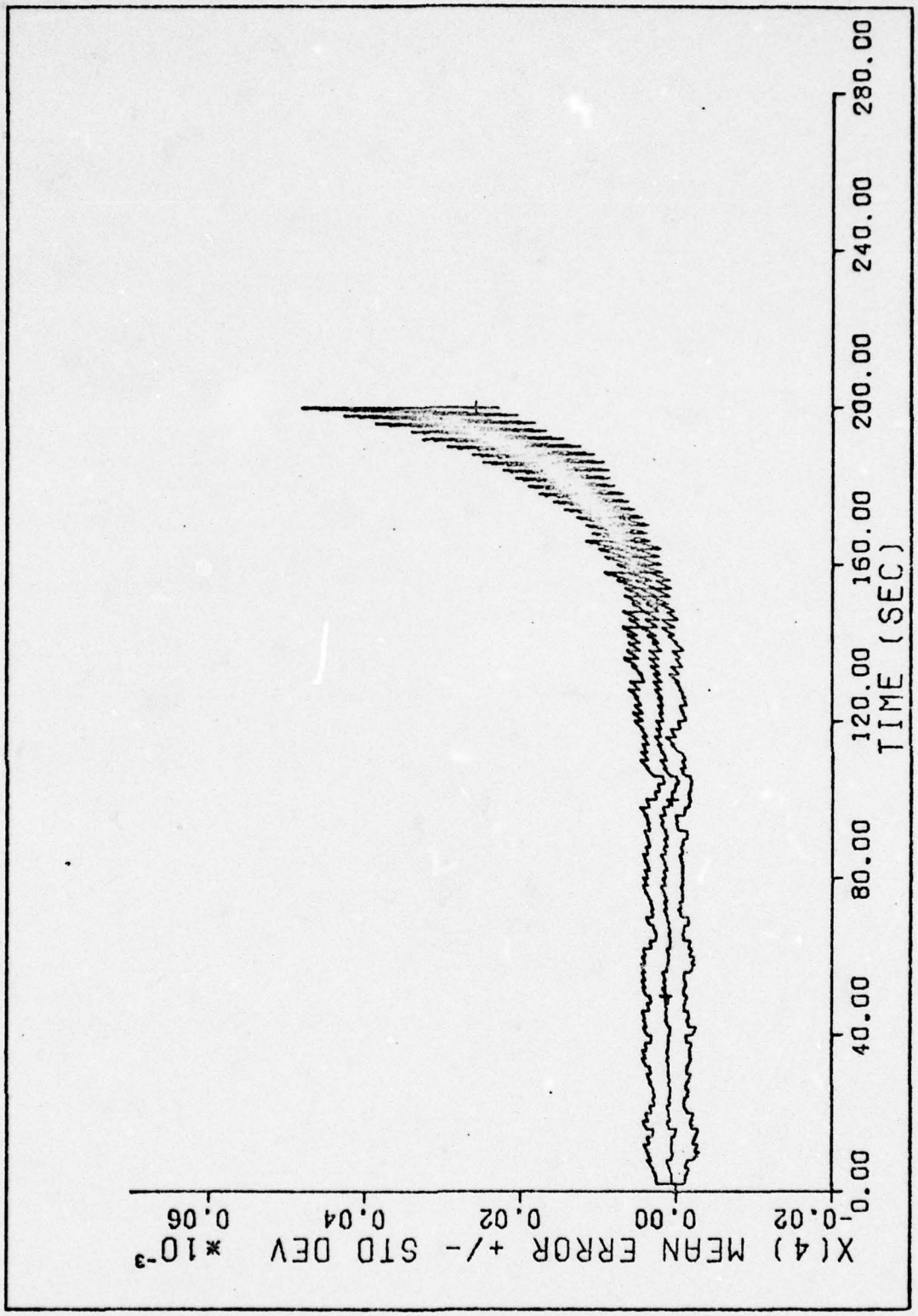


Figure 78. Case 8 - Mean Error +/- s_e vs Time

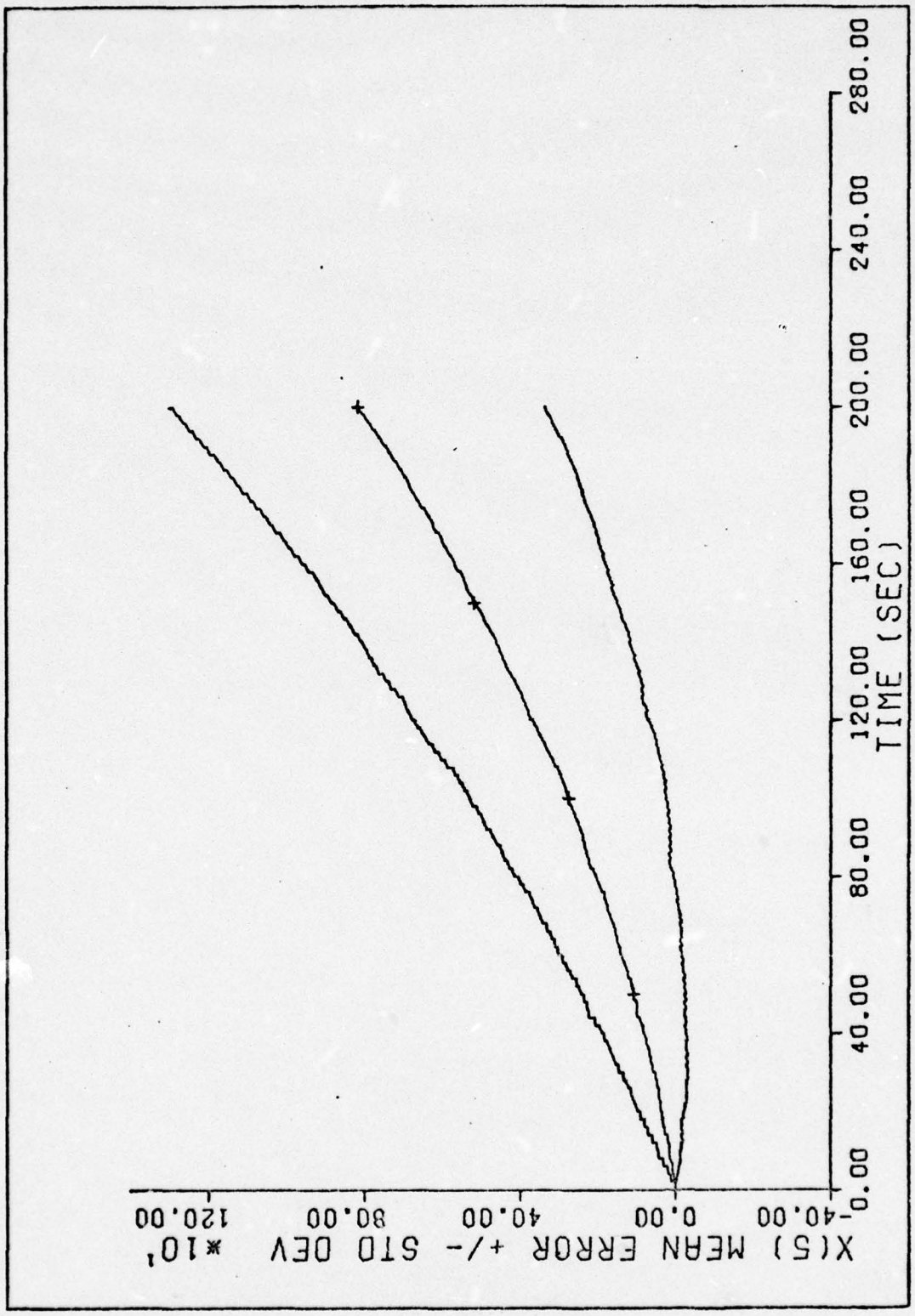


Figure 79. Case 8 - Mean Error +/- s_e vs Time

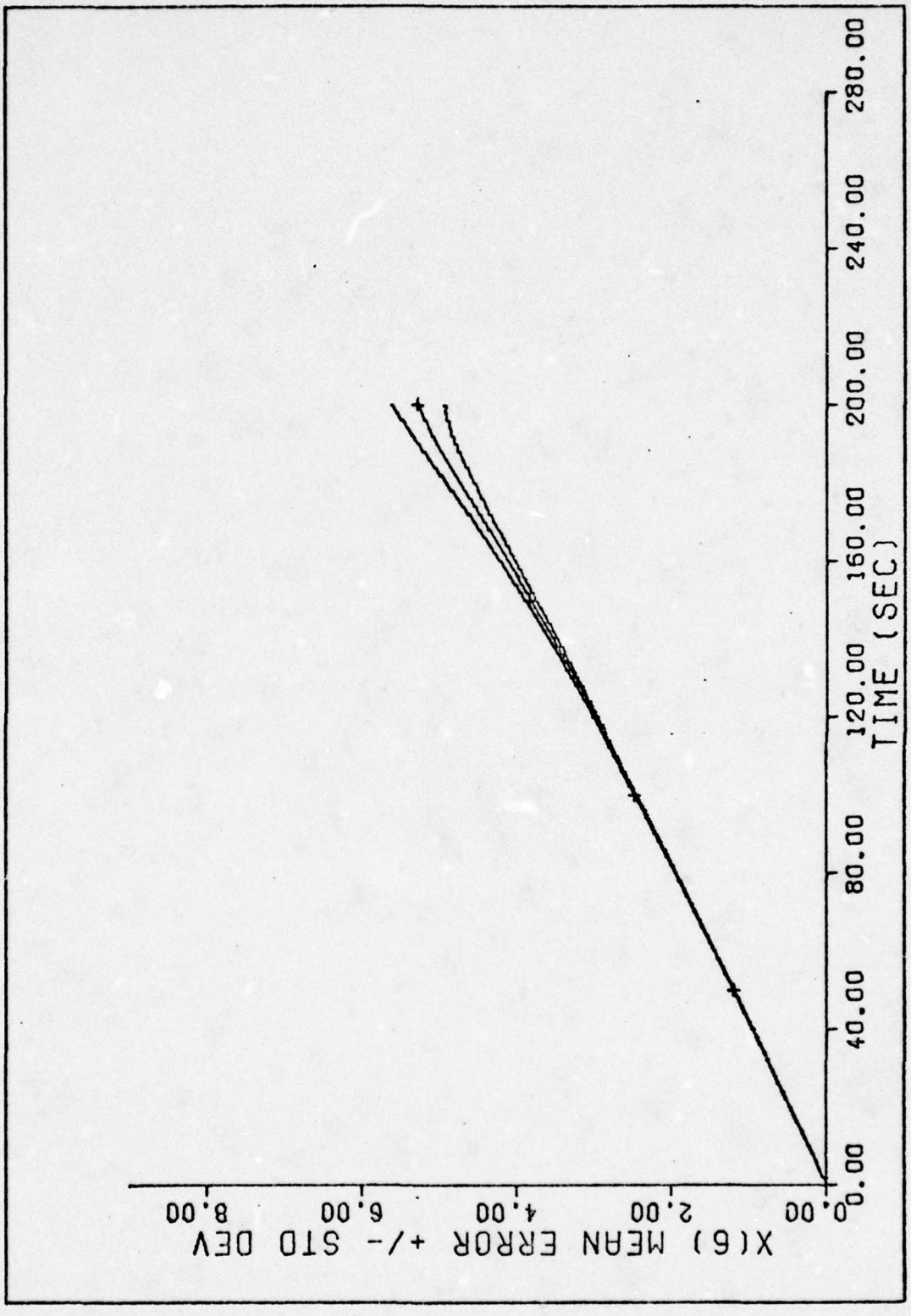


Figure 80. Case 8 - Mean Error +/- s_e vs Time

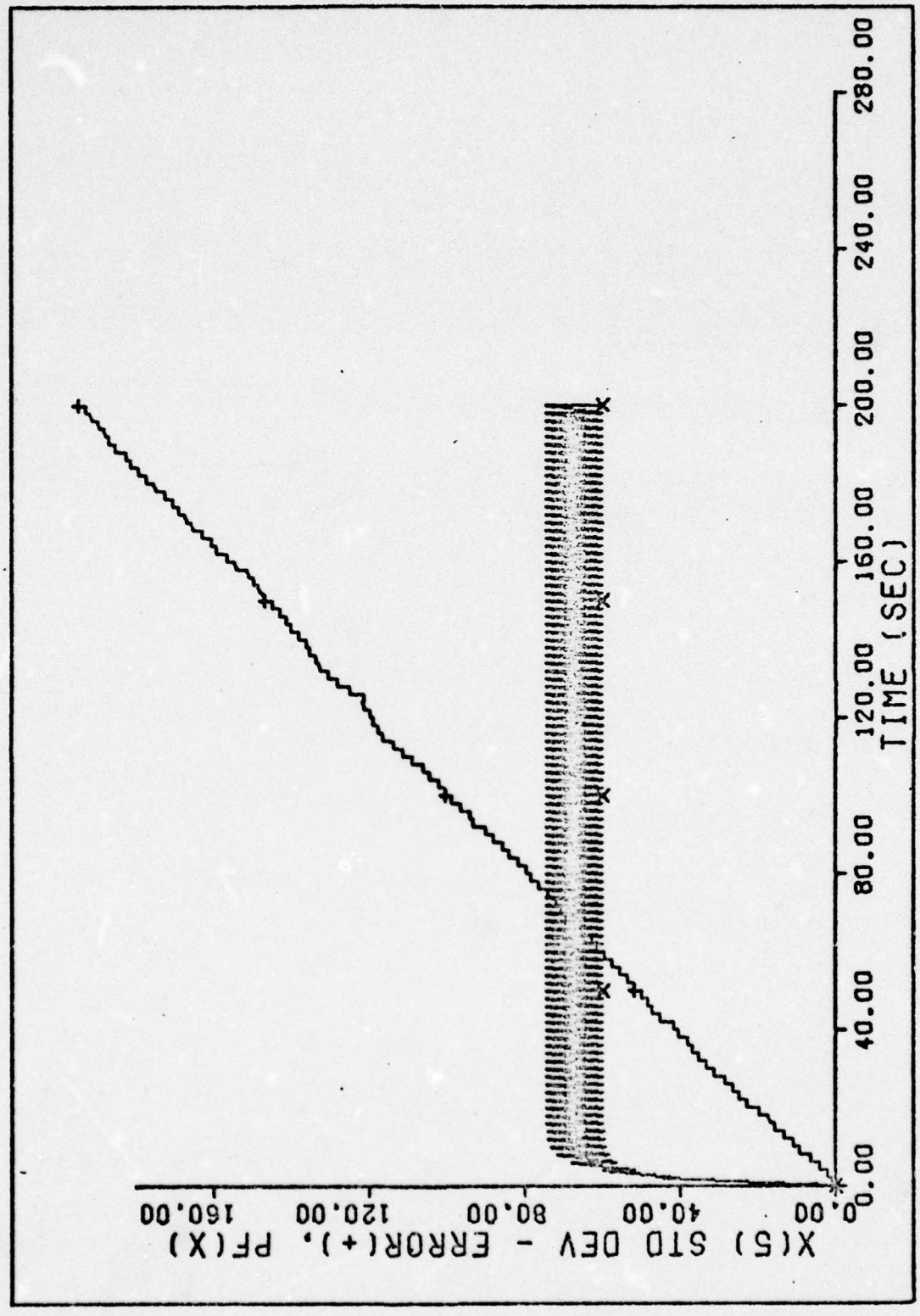


Figure 81. Case 9 - s_e vs Time

x(6), the filter estimate remained the same, but the true error decreased by a factor of 1/2 as shown in Figure 82. Figures 83 and 84 present the mean error plots for these two states.

Case 10 was performed by decreasing Q_4 by an order of magnitude so that the filter standard deviation estimate for state x(5) would be reduced further. Both the filter estimate and the true error standard deviation decreased for state x(5) and is shown in Figure 85. Again, the true error decreased for state x(6) as shown in Figure 86. Figures 87 and 88 present the mean error plots for these two states.

Since the true standard deviation was greater than the filter estimate, R_5 was increased by a factor of five to perform case 11. For state x(5), the filter estimate of the standard deviation increased and the true error decreased as shown in Figure 89. Figure 90 shows that the true error also decreased for state x(6). The mean error plot for state x(5) is shown in Figure 91.

Case 12 was performed by decreasing R_5 by approximately 1/2 so that the filter estimate would be closer to the true standard deviation at the end of the simulation time. The filter estimate of the standard deviation decreased and the true error increased. The complete results for this case are shown in Figures 92 through 103. Since this is only a feasibility study, further tuning was not attempted.

These results demonstrate that all filter state estimates of th true standard deviation error are "tunable". As the filter standard deviation is changed for some of the states, cross-coupling can affect

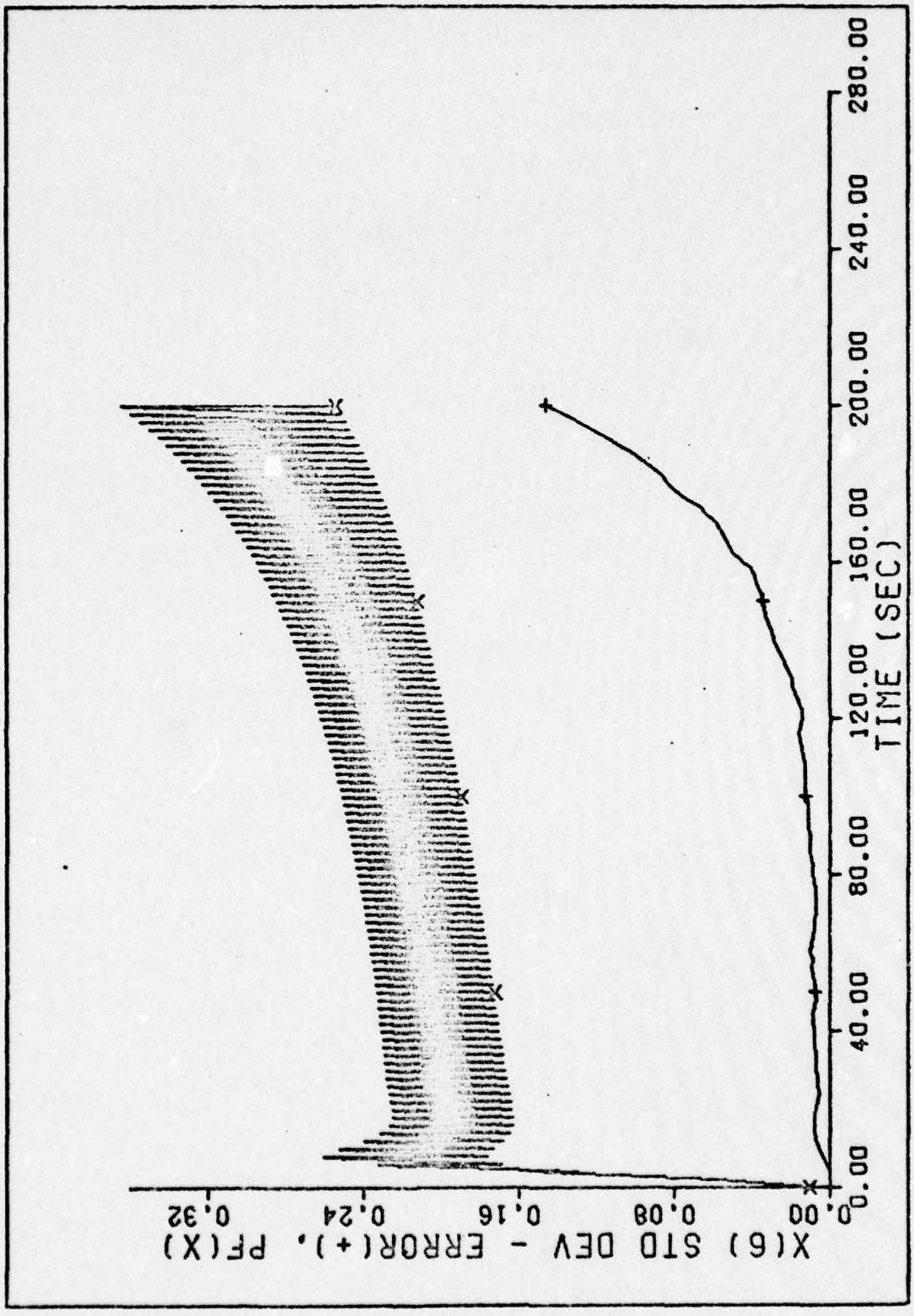


Figure 82. Case 9 - s_e vs Time

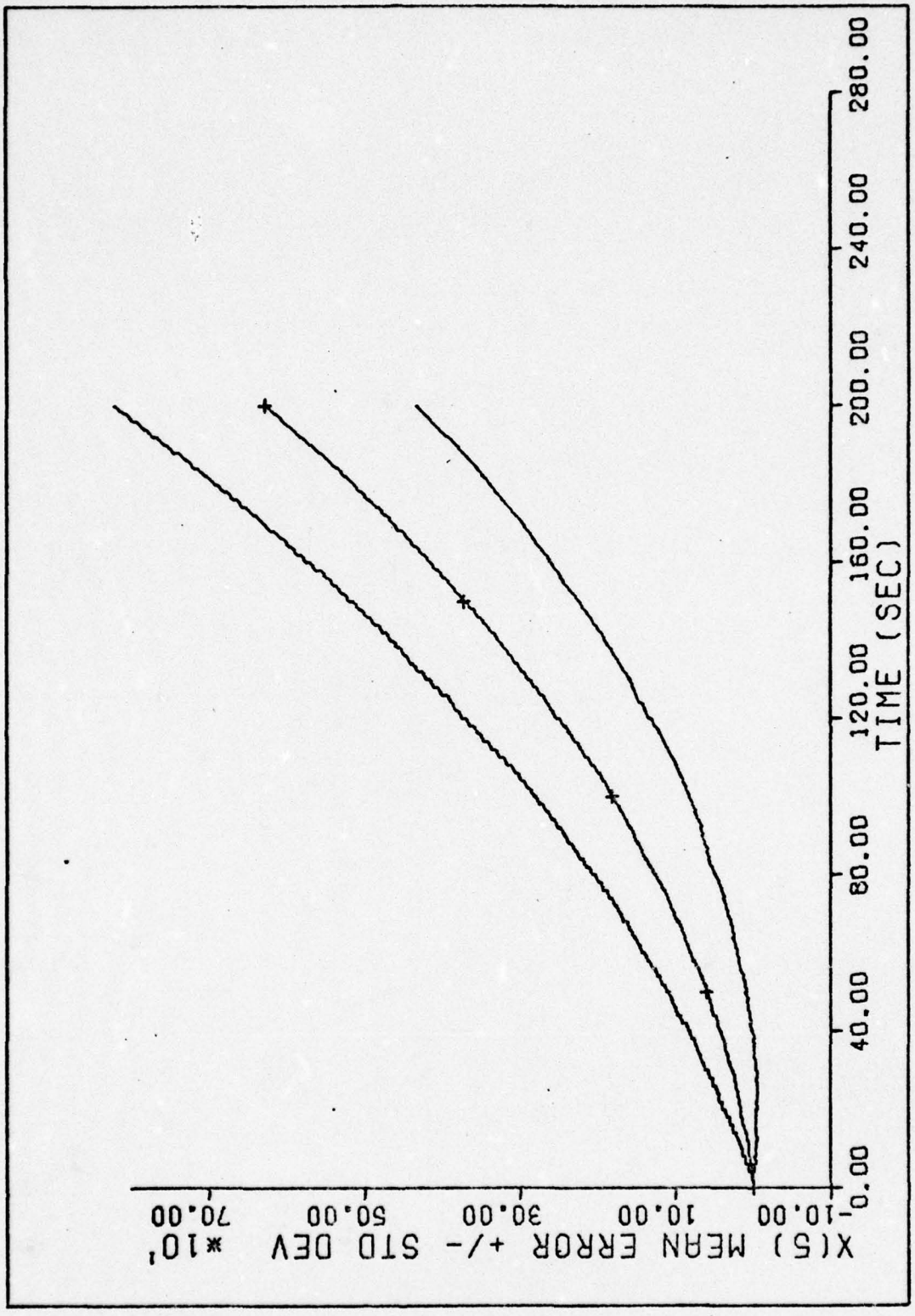


Figure 83. Case 9 - Mean Error +/- s_e vs Time

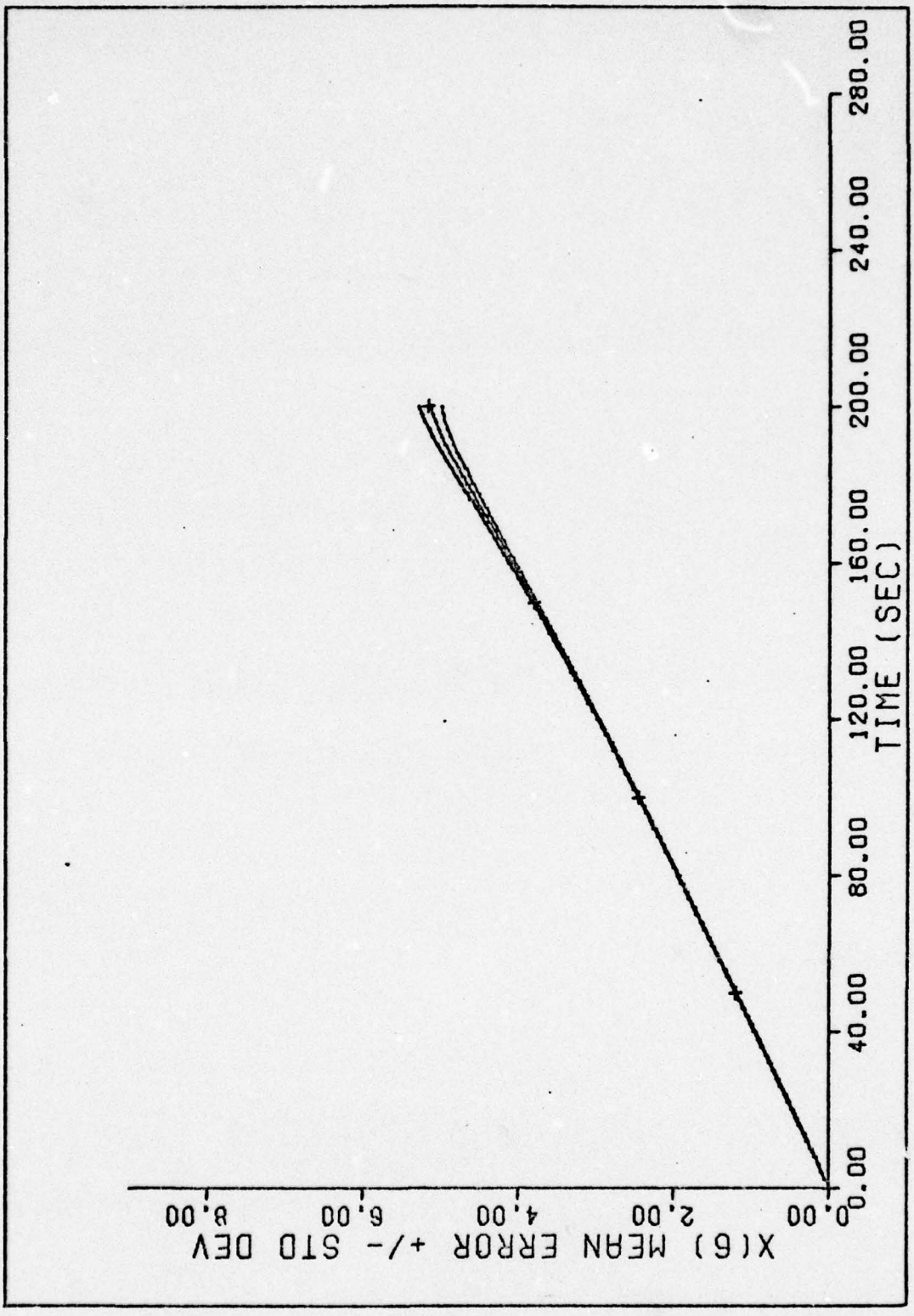


Figure 84. Case 9 - Mean Error +/- s_e vs Time

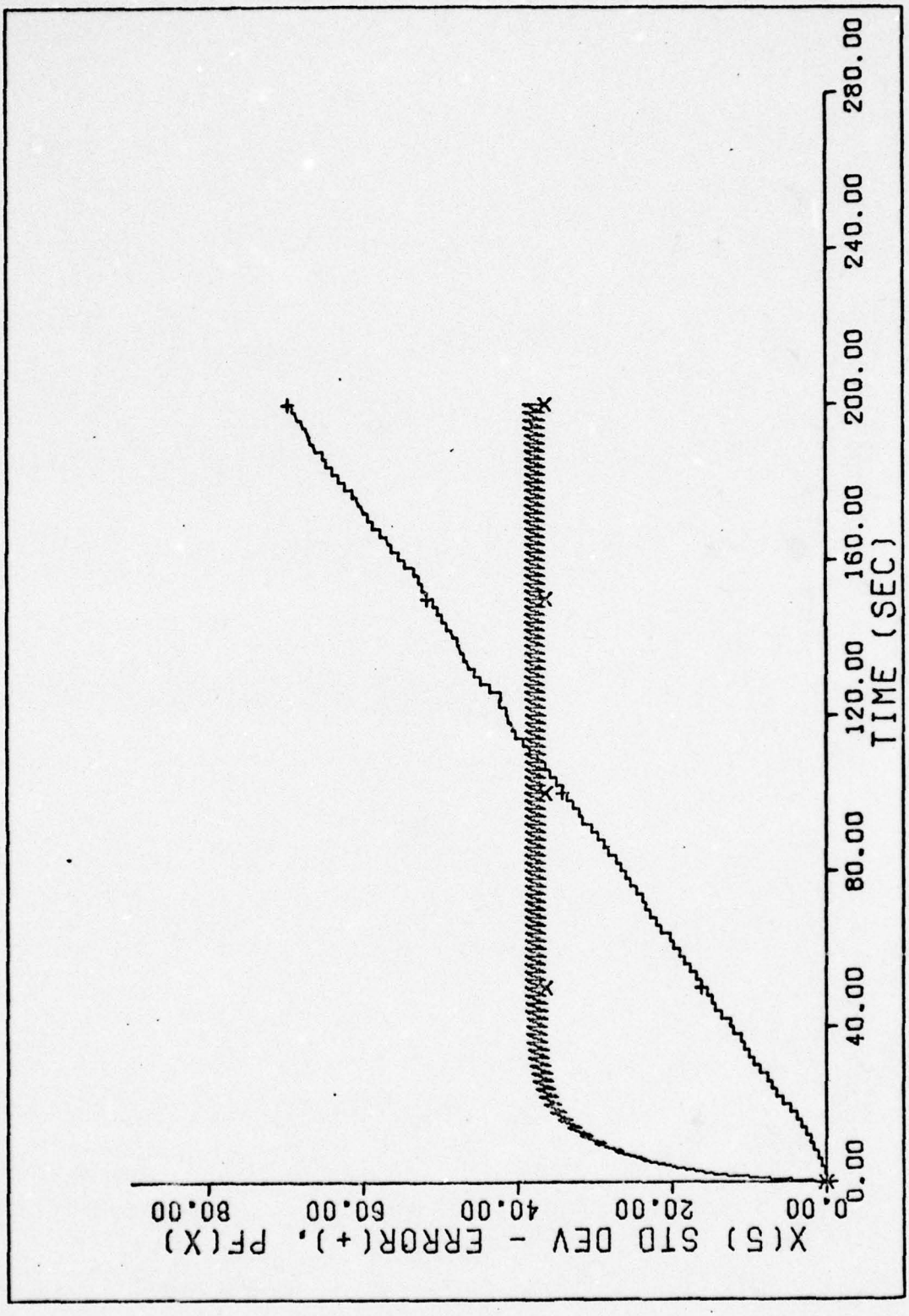


Figure 85. Case 10 - s_e vs Time

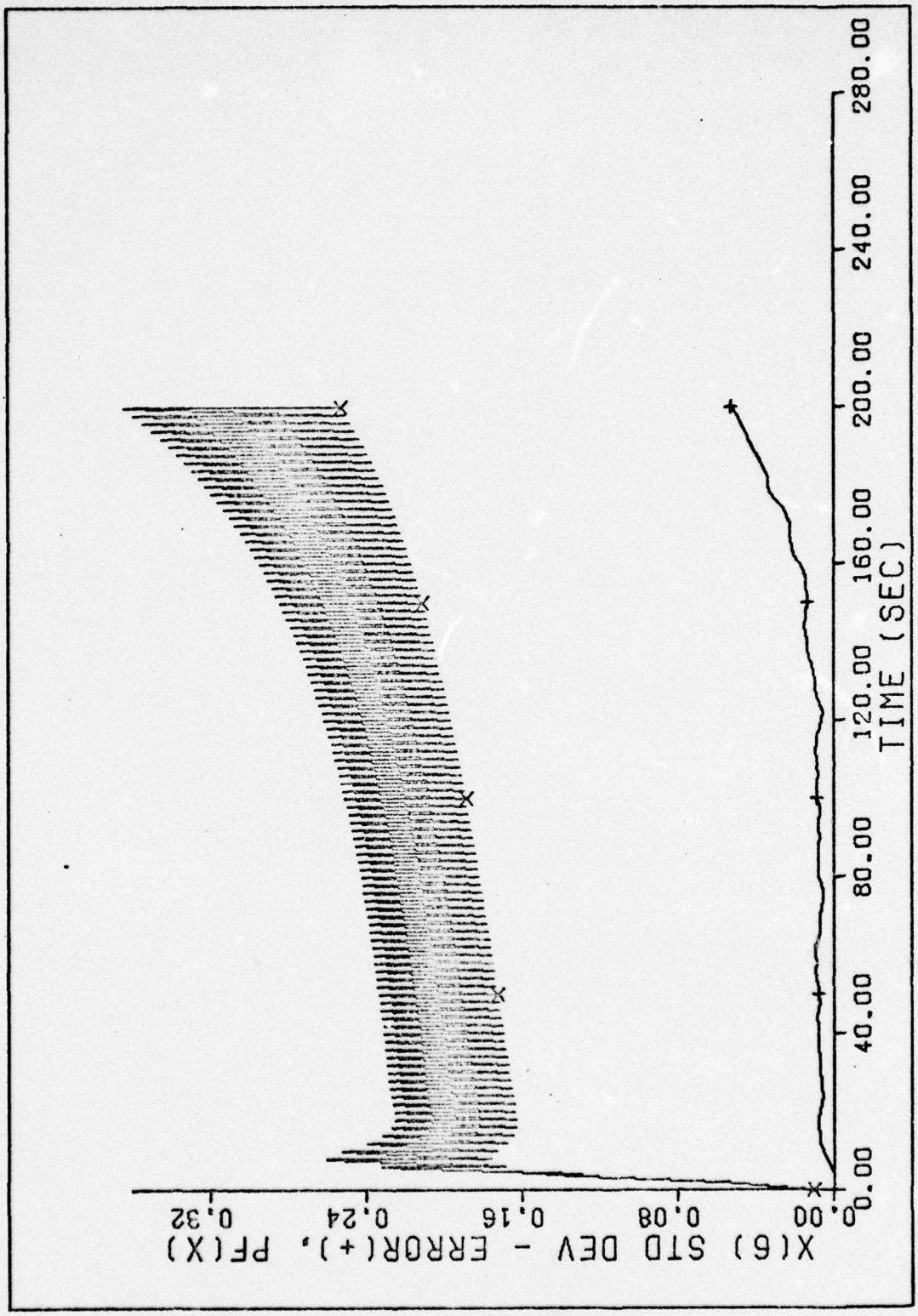


Figure 86. Case 10 - s_e vs Time

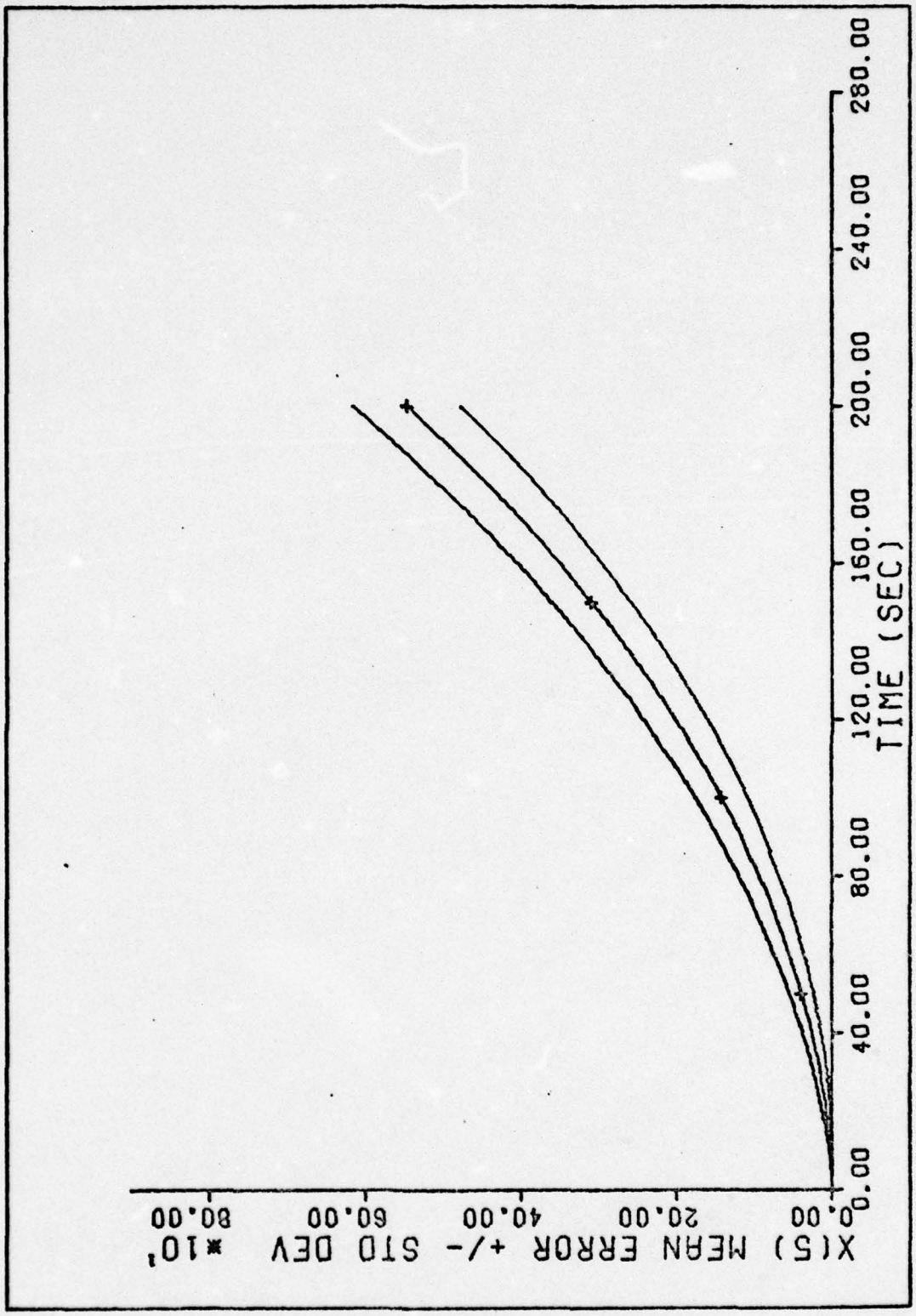


Figure 87. Case 10 - Mean Error +/- s_e vs Time

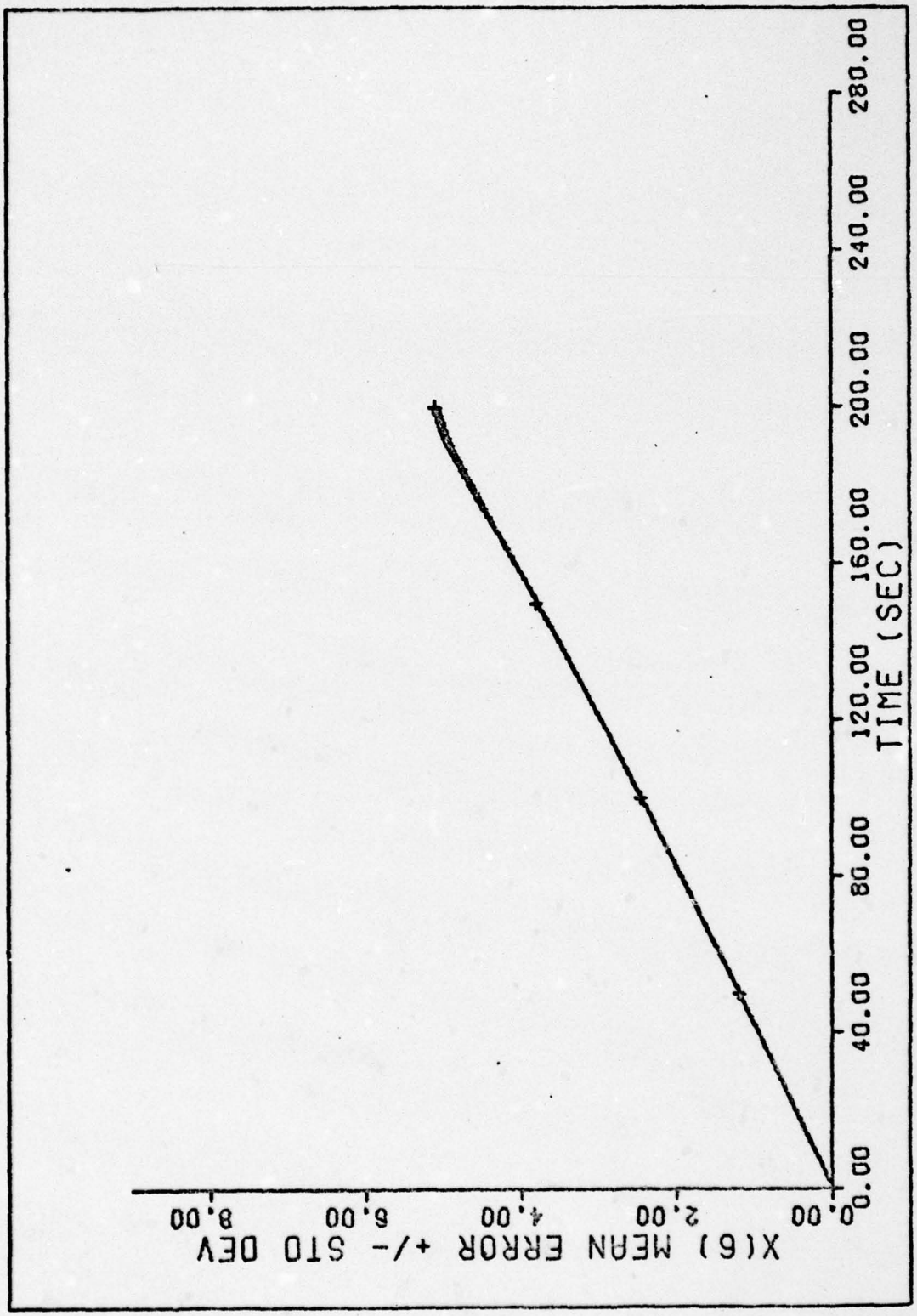


Figure 88. Case 10 - Mean Error +/- s_e vs Time

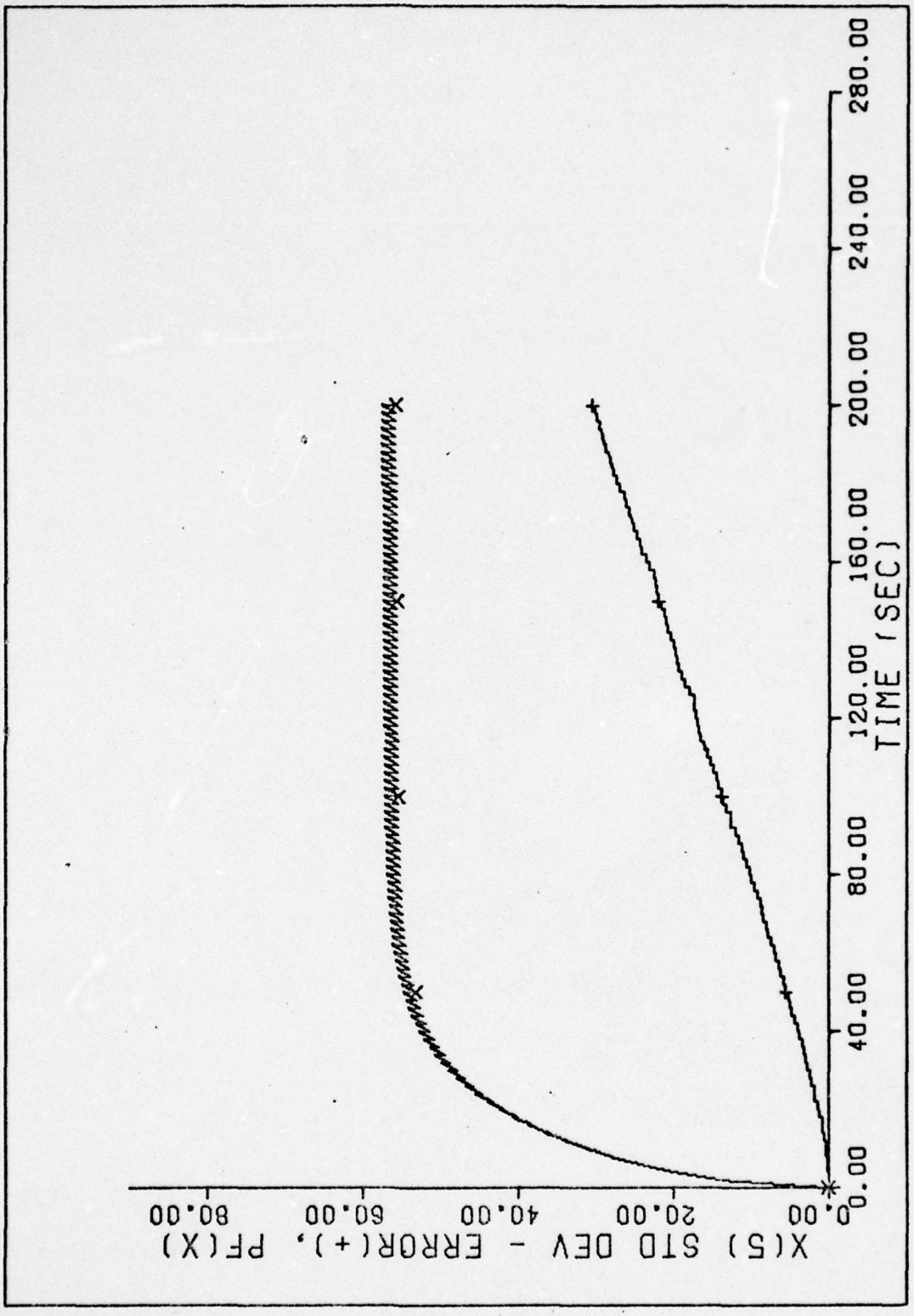


Figure 89. Case 11 - s_e vs Time

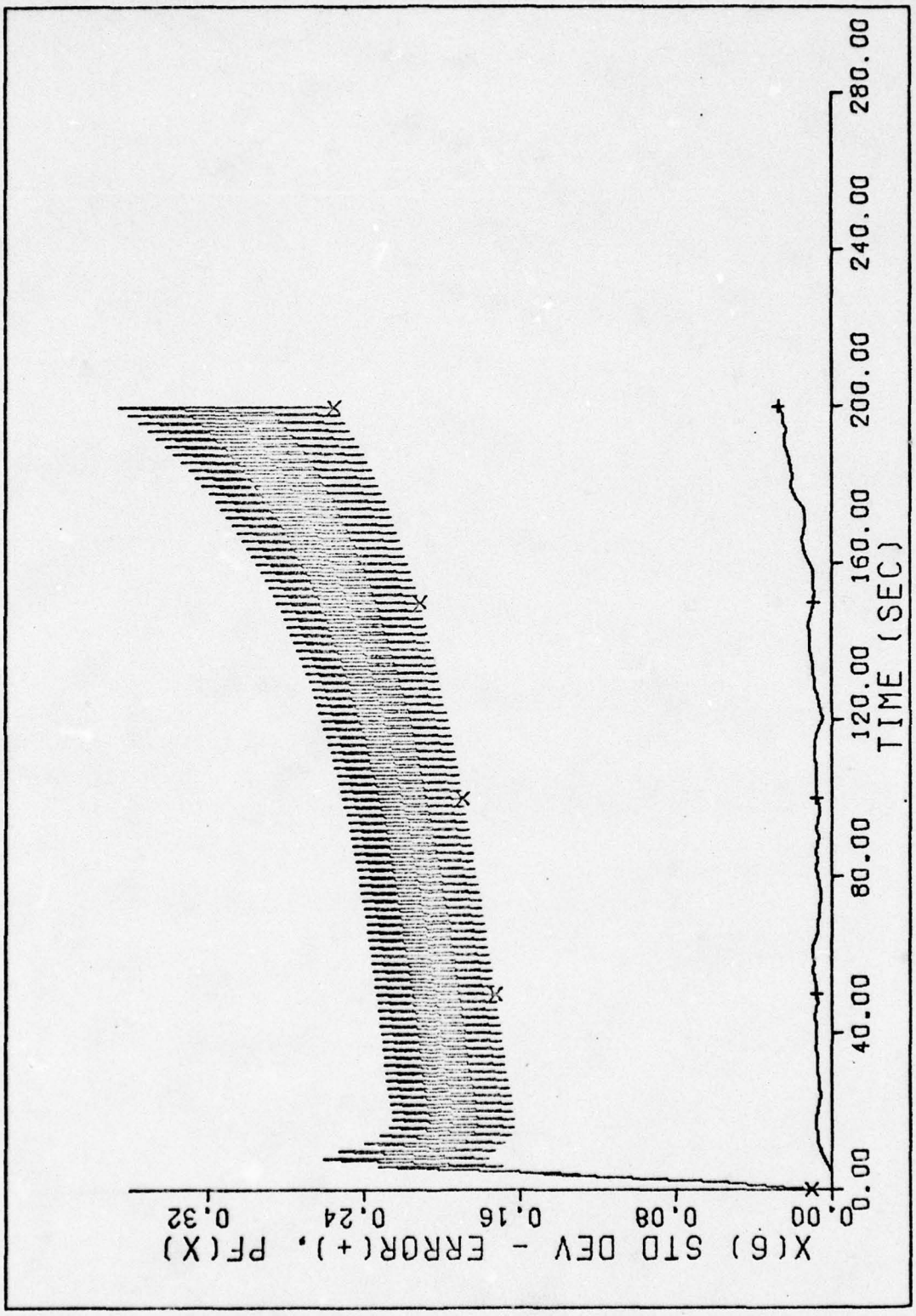


Figure 90. Case 11 - s_e vs Time

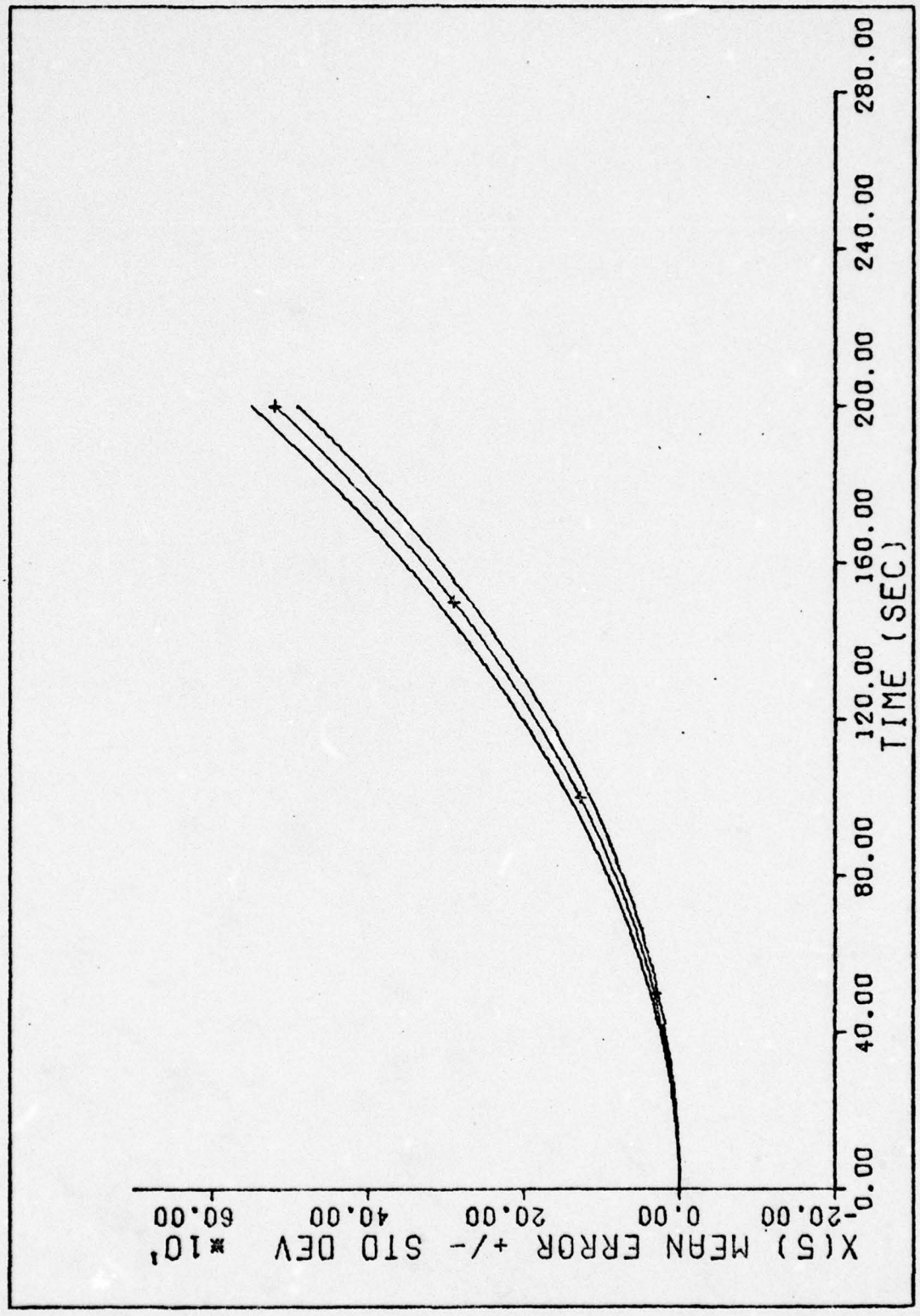


Figure 91. Case 11 - Mean Error +/- s_e vs Time

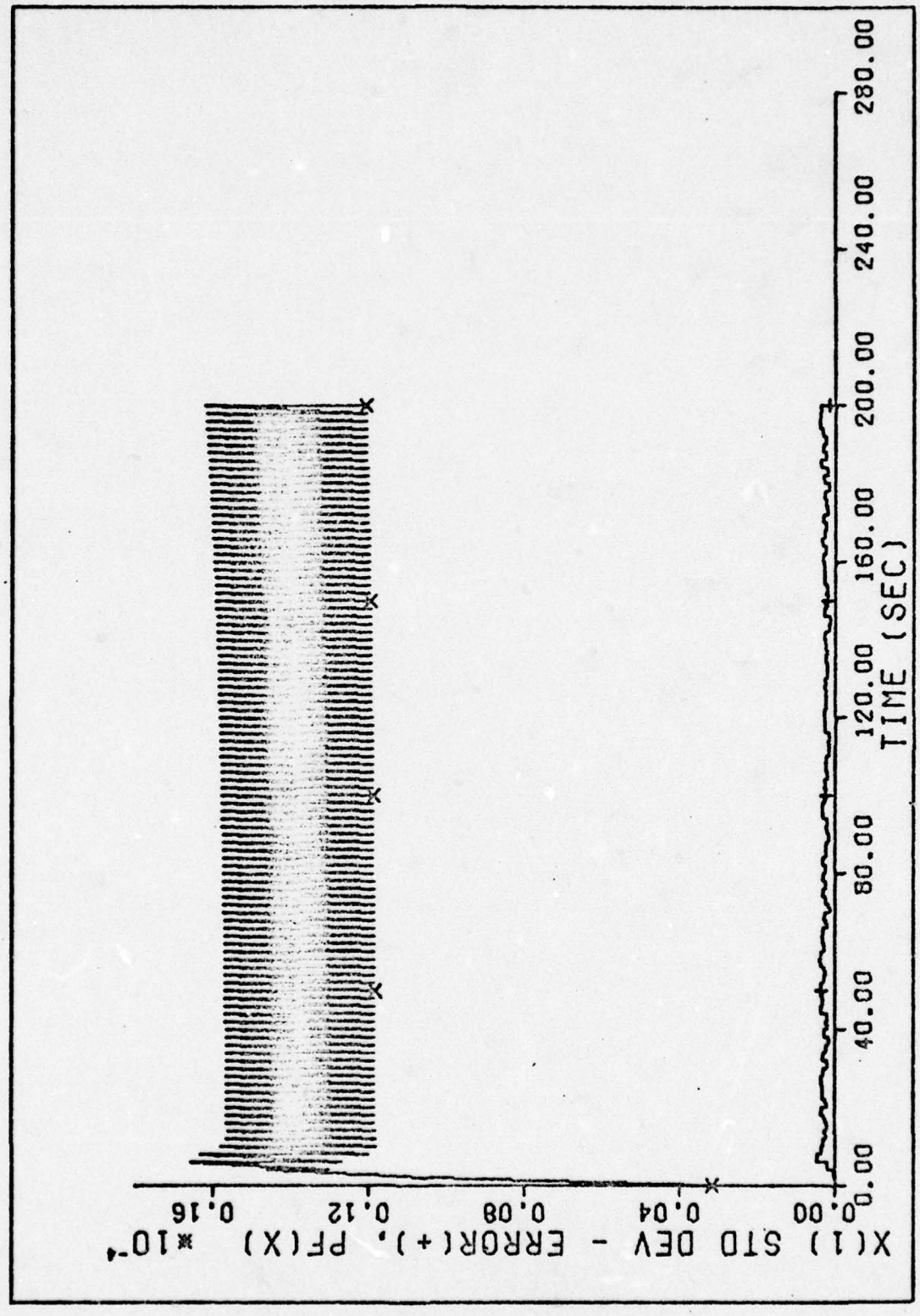


Figure 92. Case 12 - s_e vs Time

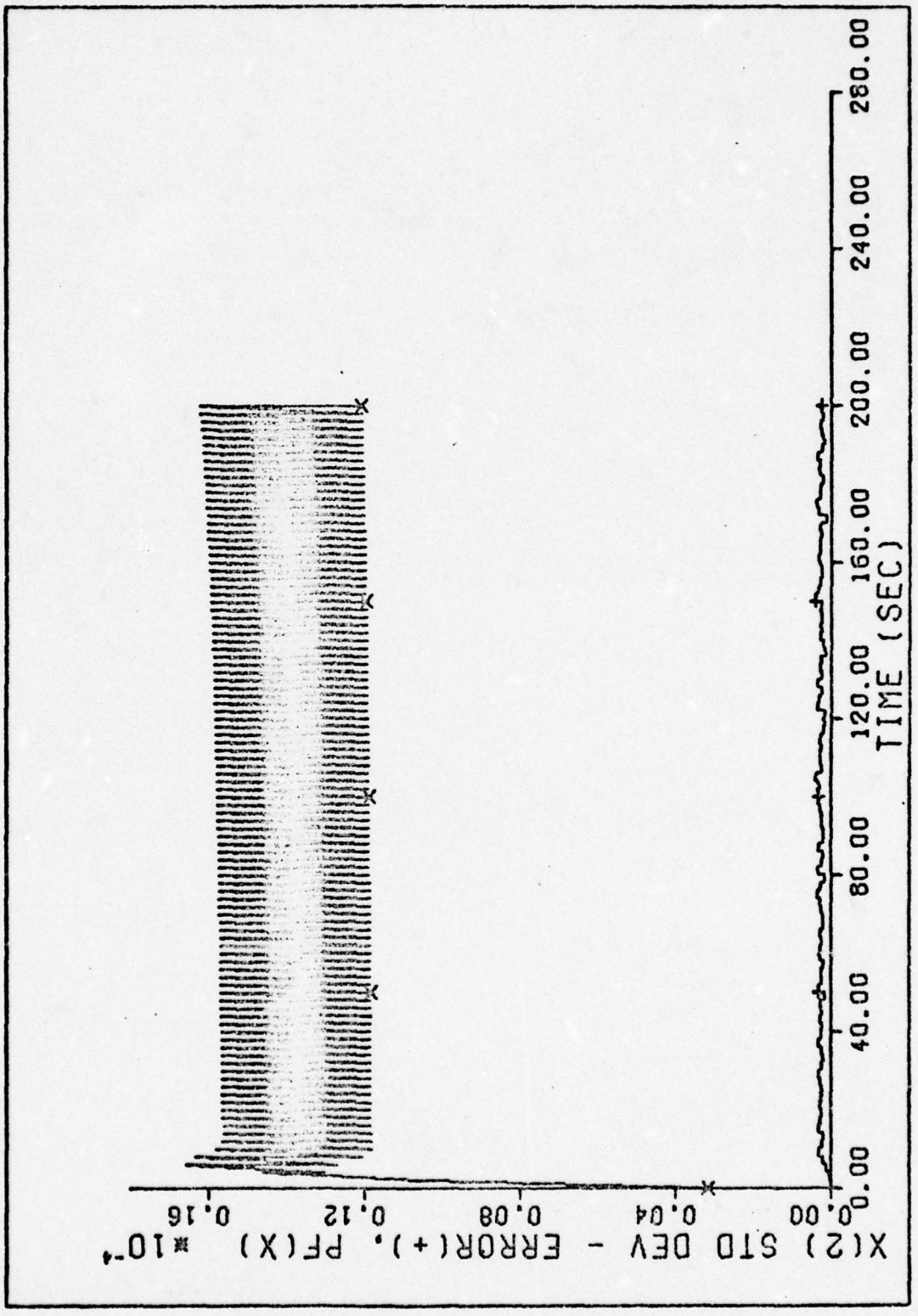


Figure 93. Case 12 - s_e vs Time

Figure 94. Case 12 - s_e vs Time

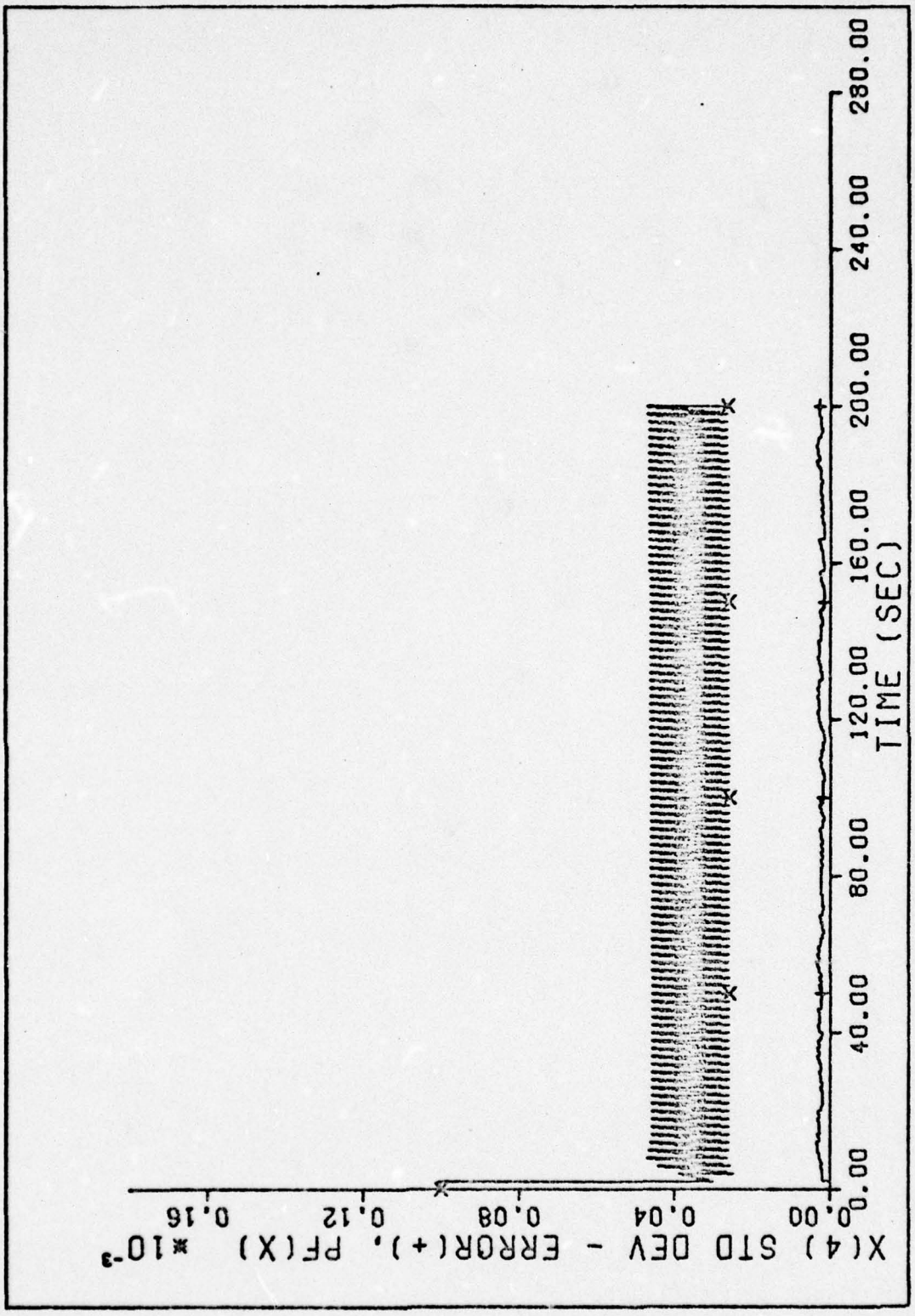


Figure 95. Case 12 - s_e vs Time

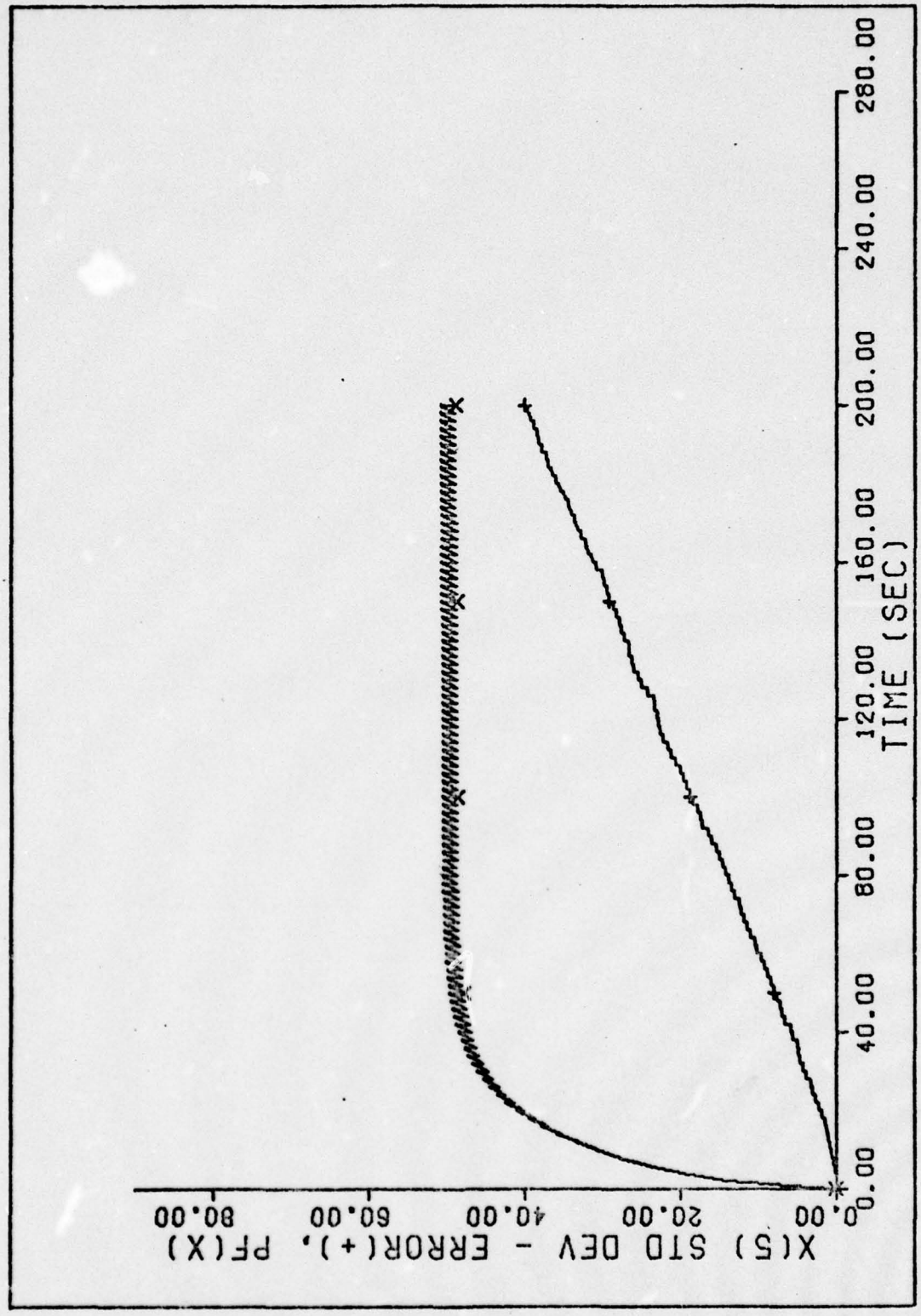


Figure 96. Case 12 - s_e vs Time

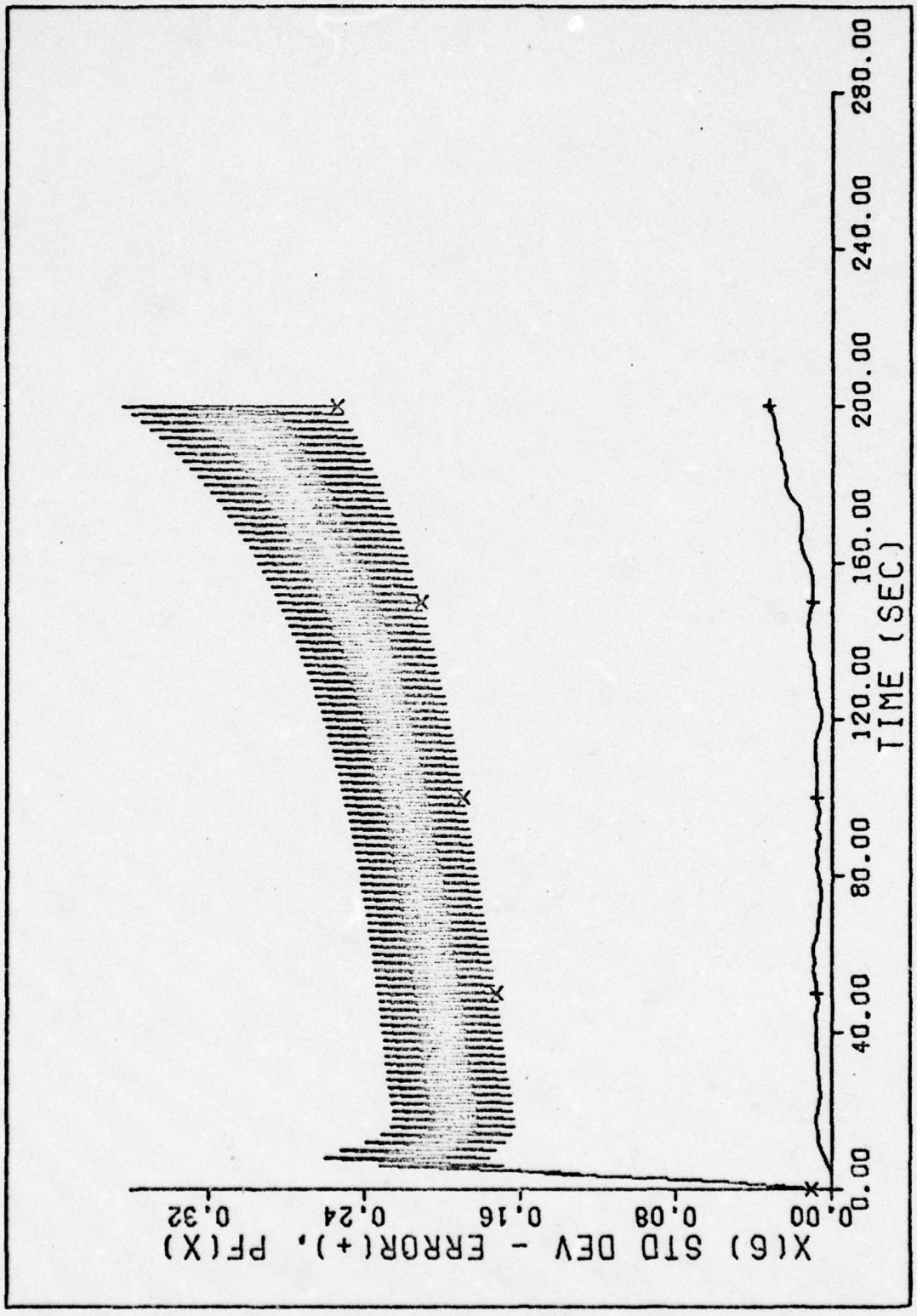


Figure 97. Case 12 - s_e vs Time

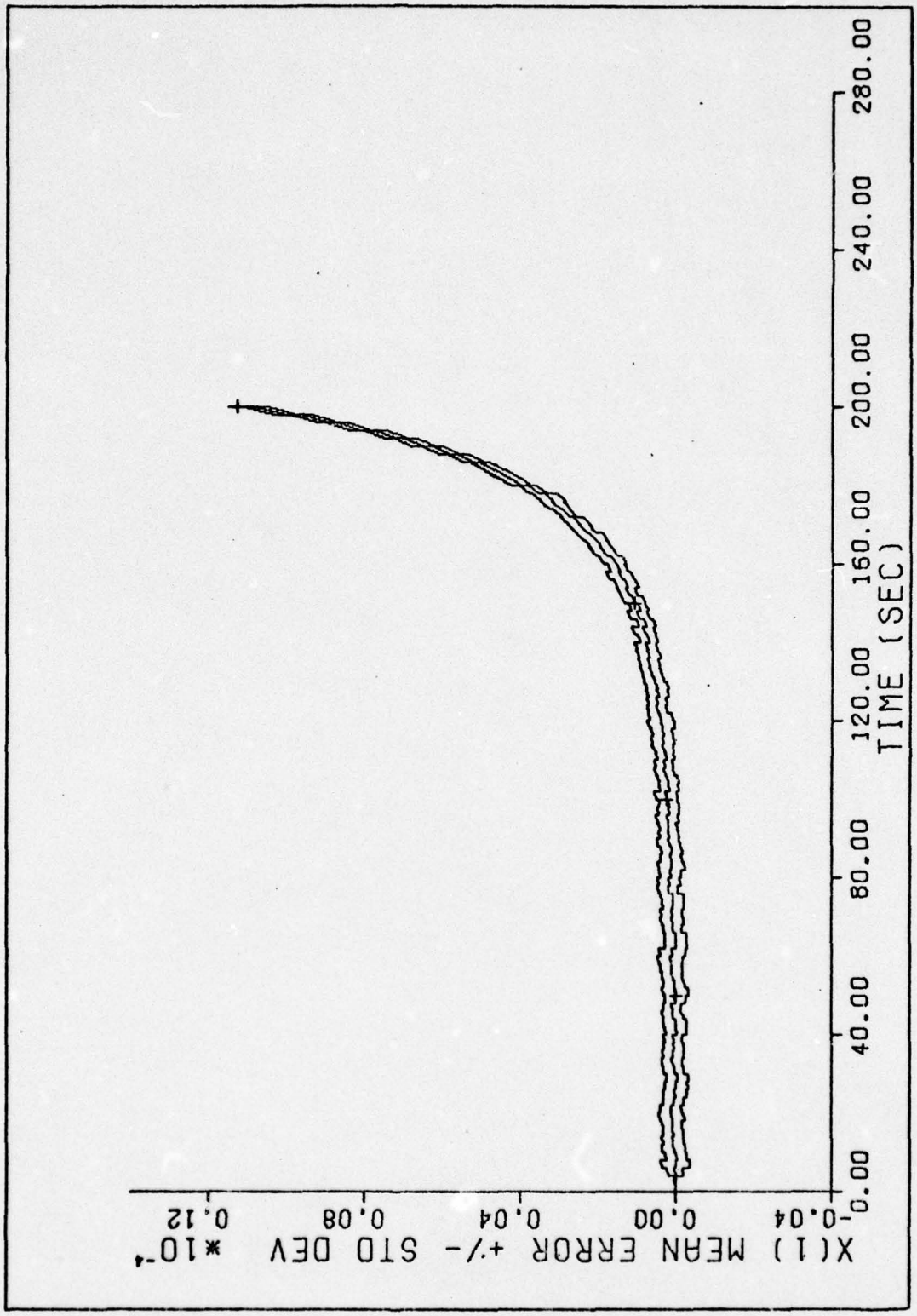


Figure 98. Case 12 - Mean Error +/- s_e vs Time

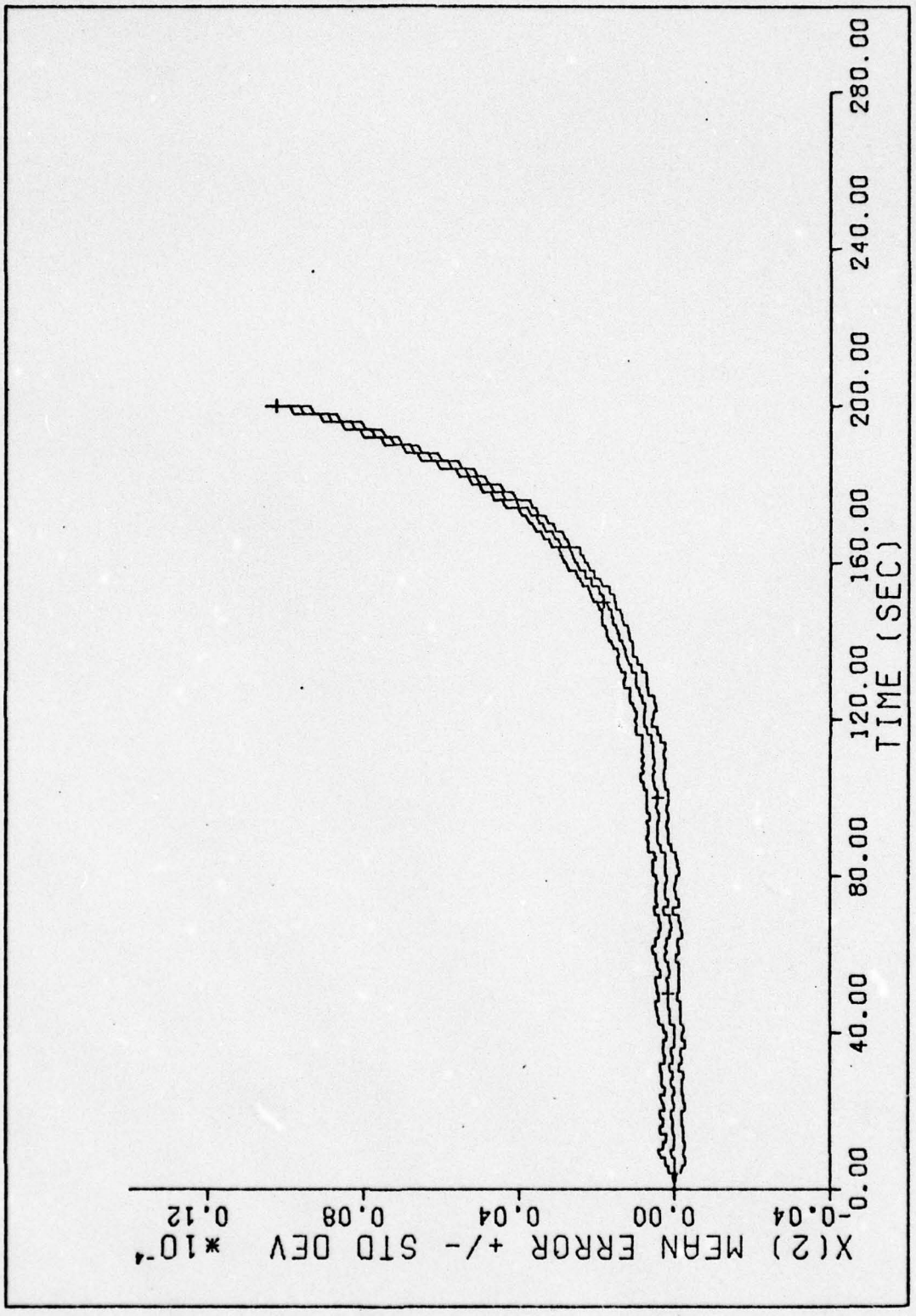


Figure 99. Case 12 - Mean Error +/- s_e vs Time

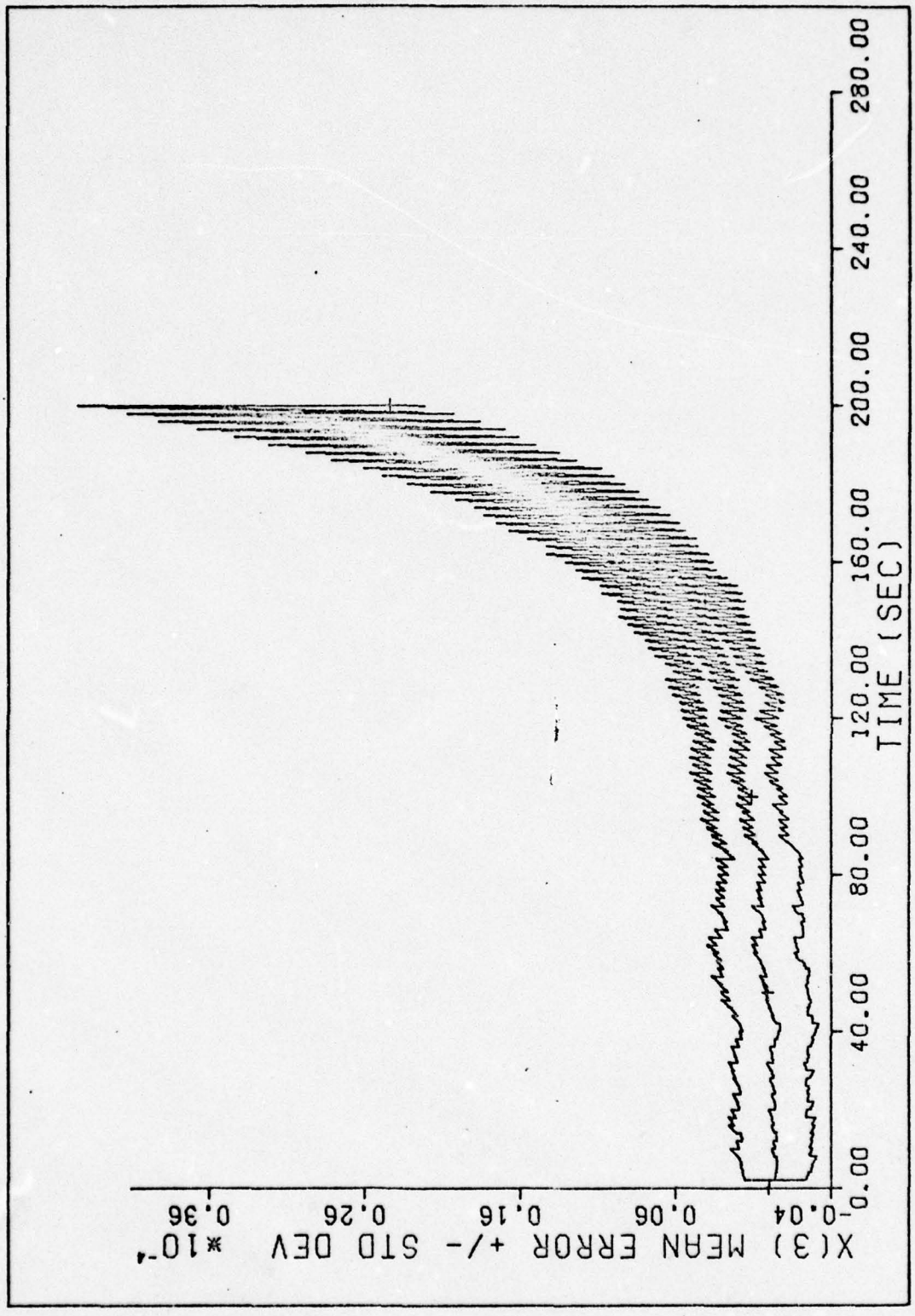


Figure 100. Case 12 - Mean Error +/- s_e vs Time

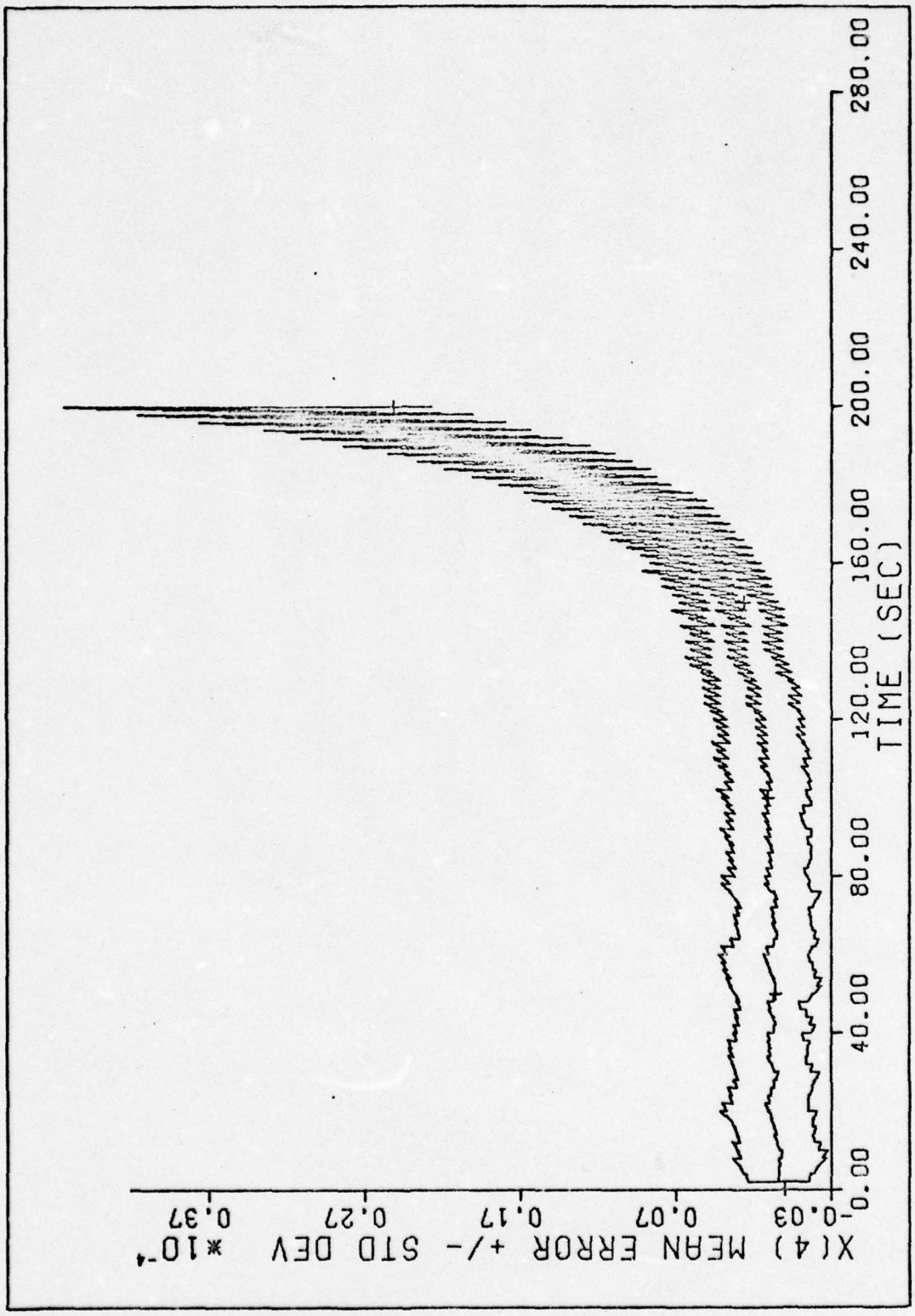


Figure 101. Case 12 - Mean Error +/- s_e vs Time

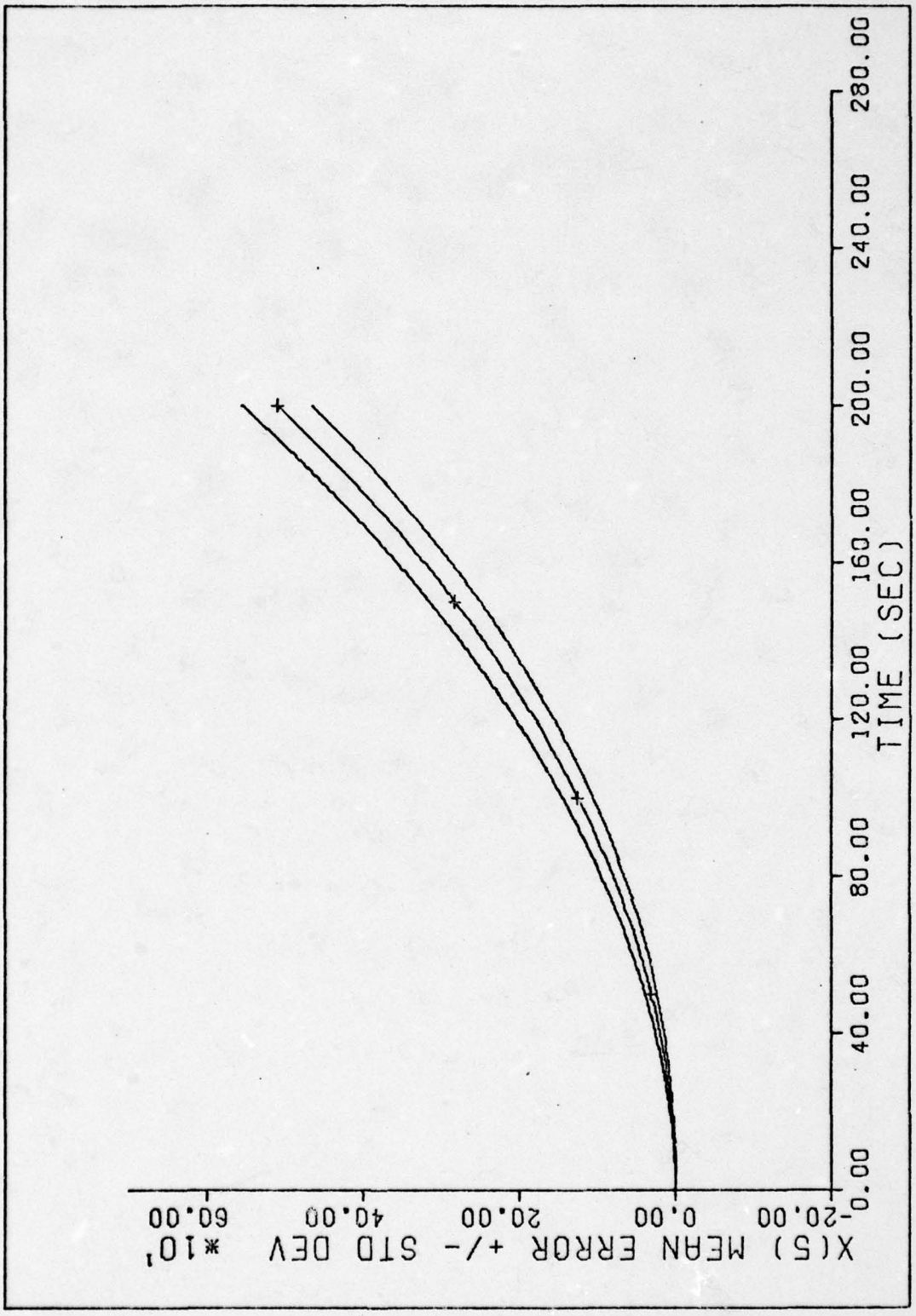


Figure 102. Case 12 - Mean Error +/- s_e vs Time

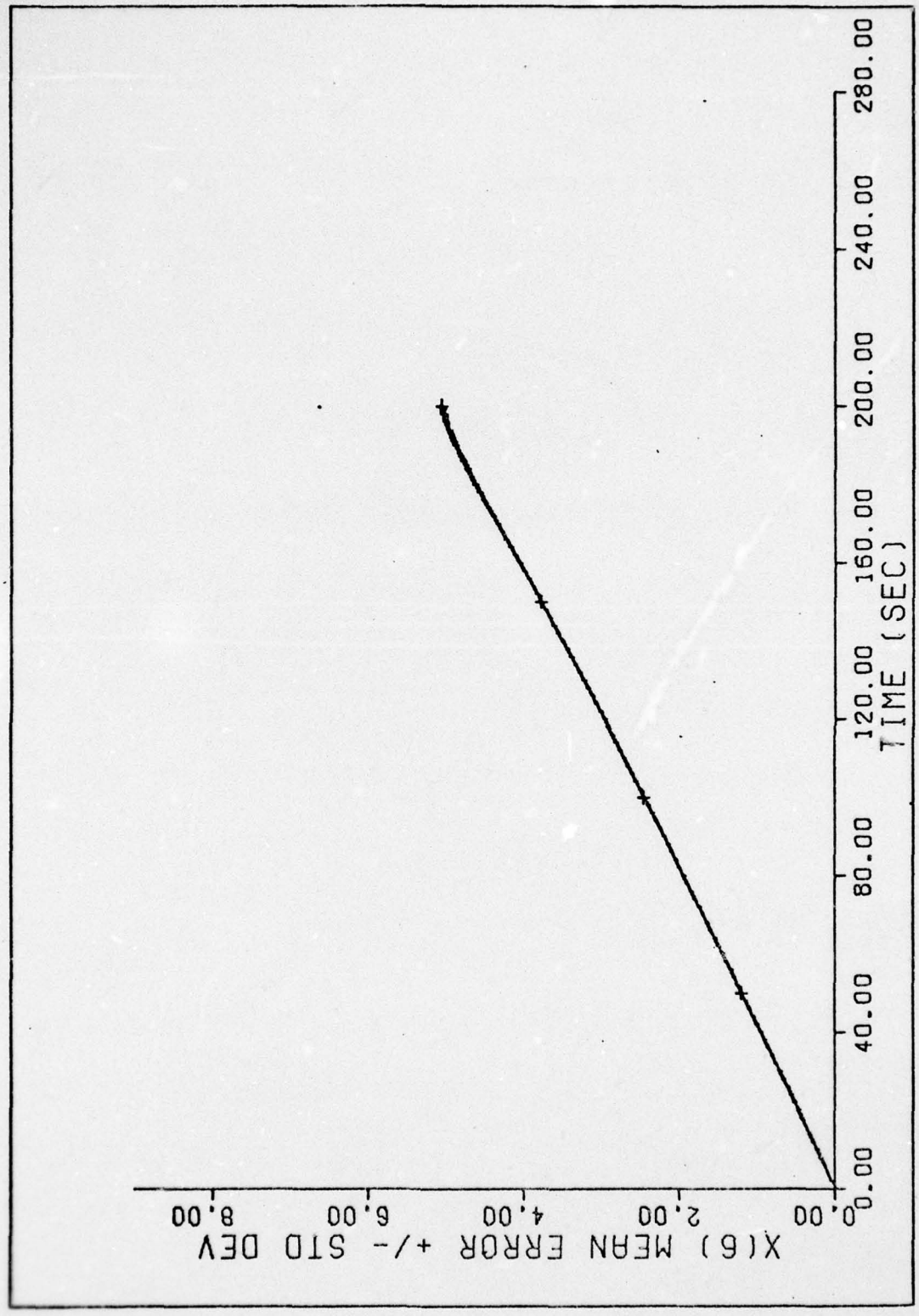


Figure 103. Case 12 - Mean Error +/- s_e vs Time

the estimates for the other states significantly. Most noticeable in these plots is that the filter estimate of the mean error is biased. This bias appears to grow without bound for the time period of interest. The next chapter will present some recommendations for compensating this bias.

Extended Simulation Runs

Additional Monte Carlo simulations were performed, using the parameters for case 12, to obtain the results for 5, 20, and 40 simulation runs. These additional runs were performed in order to show the change in calculated statistics with additional data points (samples). For each of these simulations (case A - 5 runs, case B - 20 runs, and case C - 40 runs), the average filter estimate of the standard deviation for each corresponding state was essentially the same. However, there is some variation in the plots of the true error standard deviation between the cases. Most noticeable is the true error for state x(5). Figures 104 through 106 show the standard deviation plots for state x(5): case A, case B, and case C. The mean error plots change only slightly from case to case with the number of simulation runs and are not shown.

These additional simulation runs were performed because Bucy, Hecht, and Senne (Ref 17:79-85) make the statement that significant conclusions (having a statistical confidence) cannot be made on the basis of only ten simulation runs. However, these results indicate that if a sufficient number of simulation runs are performed to obtain reasonable confidence in the resultant statistics, that the results

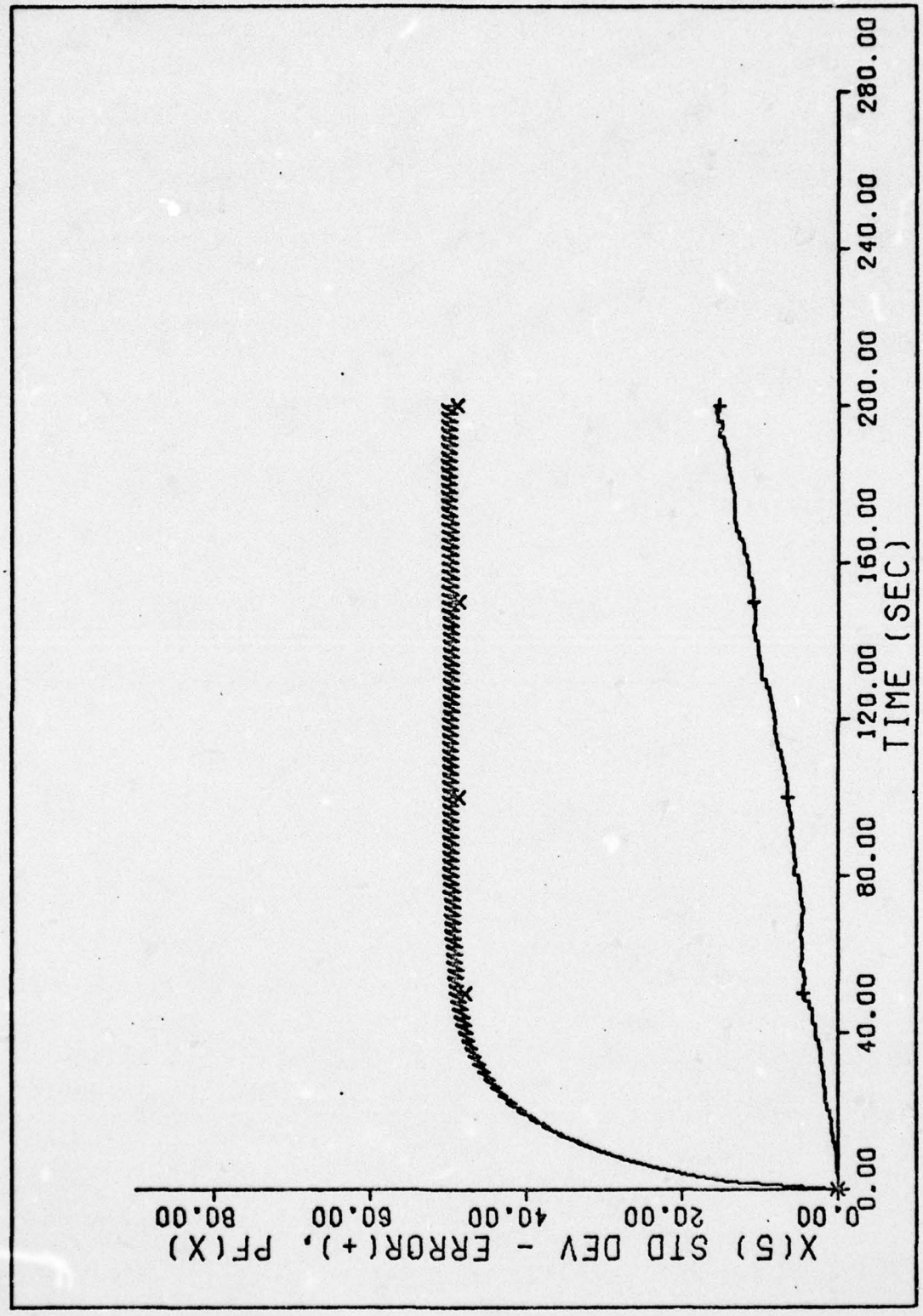


Figure 104. s_e vs Time for State x(5) (5 runs)

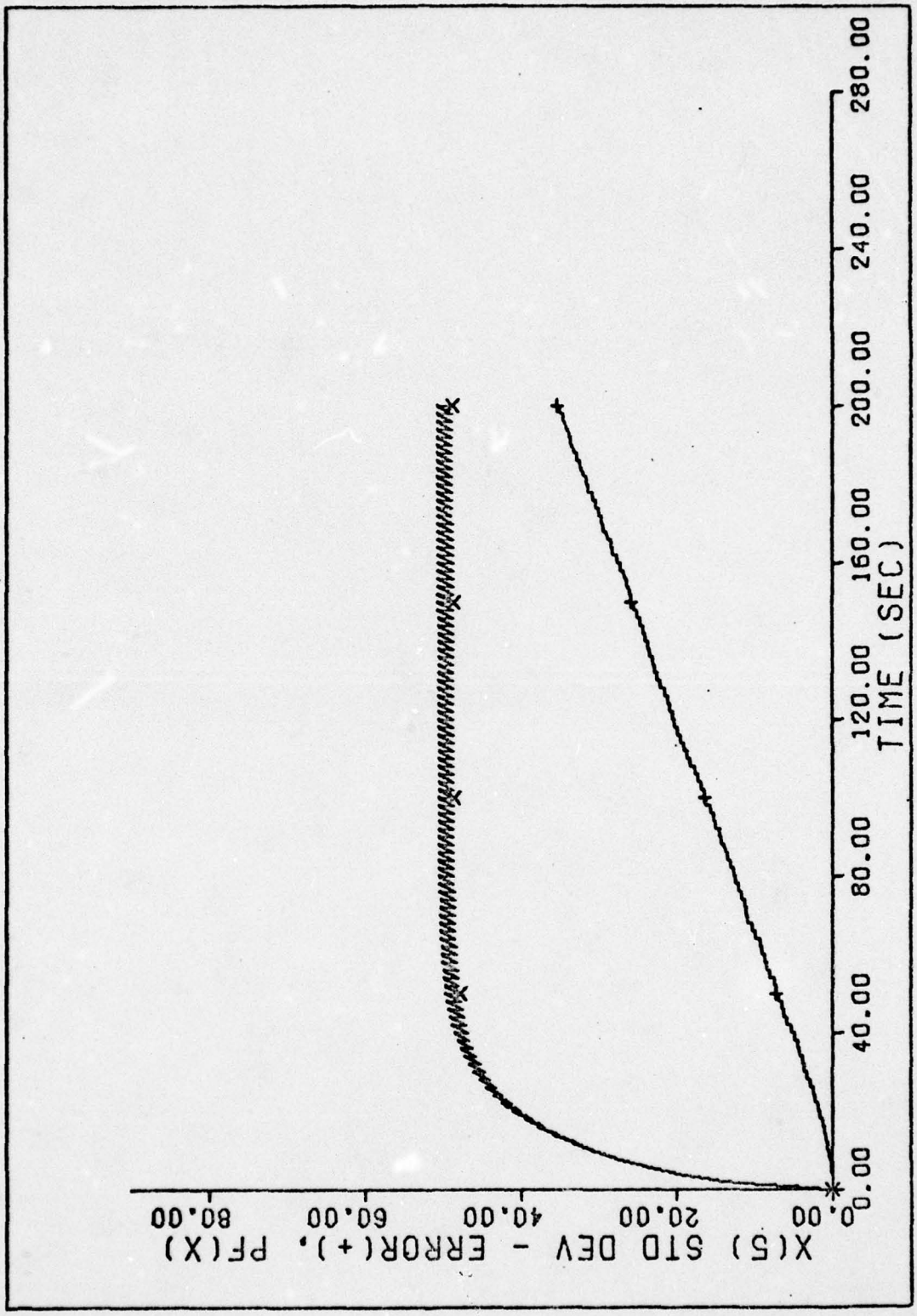


Figure 105. s_e vs Time for State x(5) (20 Runs)

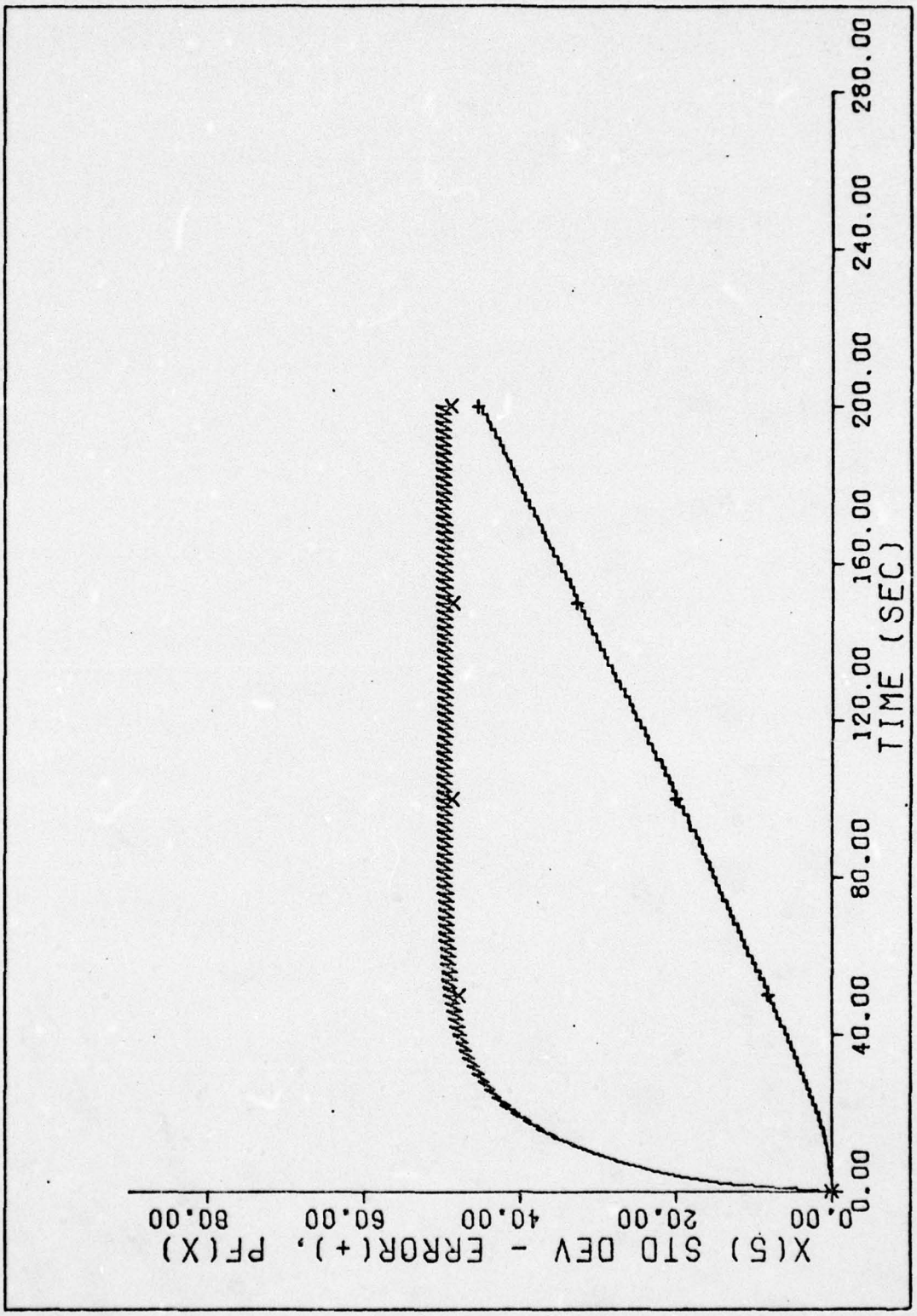


Figure 106. s_e vs Time for State x(5) (40 Runs)

will be similar to those presented in Figures 92 through 103 (for ten simulation runs). This result indicates that the filter can be tuned using only a relatively small number of simulation runs. Once the desired tuning is achieved, the number of runs can be increased to gain confidence in the resultant statistics.

Recovery from Initial Condition Error

While it is generally accepted practice to initially tune the (extended) Kalman filter with zero initial condition error and with $\underline{P}_o$ as the best apriori knowledge of the initial condition error, this is not the conclusion of the filter performance analysis. Since, in the real world, the starting conditions of the filter will almost always be different from the "true" system, the filter should be analyzed with non-zero initial error. The question then arises as to what initial error should be used. One accepted method is to displace the filter initial condition from the truth model by one sigma of the apriori standard deviation and observe the change in the mean error plots for a Kalman filter and, for the extended Kalman filter, the standard deviation plots as well.

One Monte Carlo simulation (10 runs) was performed with one filter state having an initial condition error displaced by one sigma of the apriori standard deviation. Repeating for each state variable initially offset, the standard deviation plots were the same as Figures 92 through 97. However, the mean plots did change slightly and are discussed below.

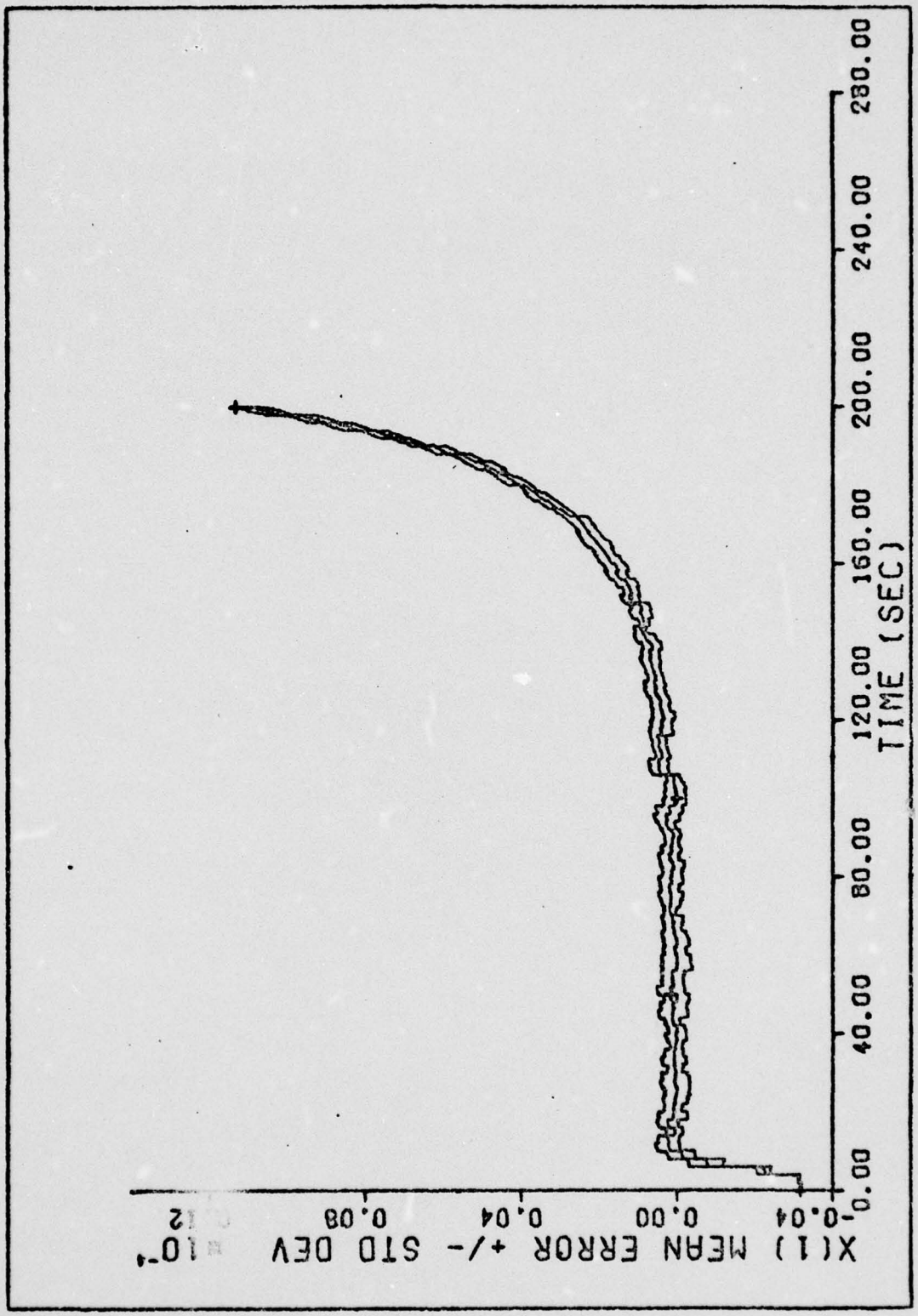


Figure 107. State x(1) Initial Condition Error - Mean Error +/- s_e vs Time

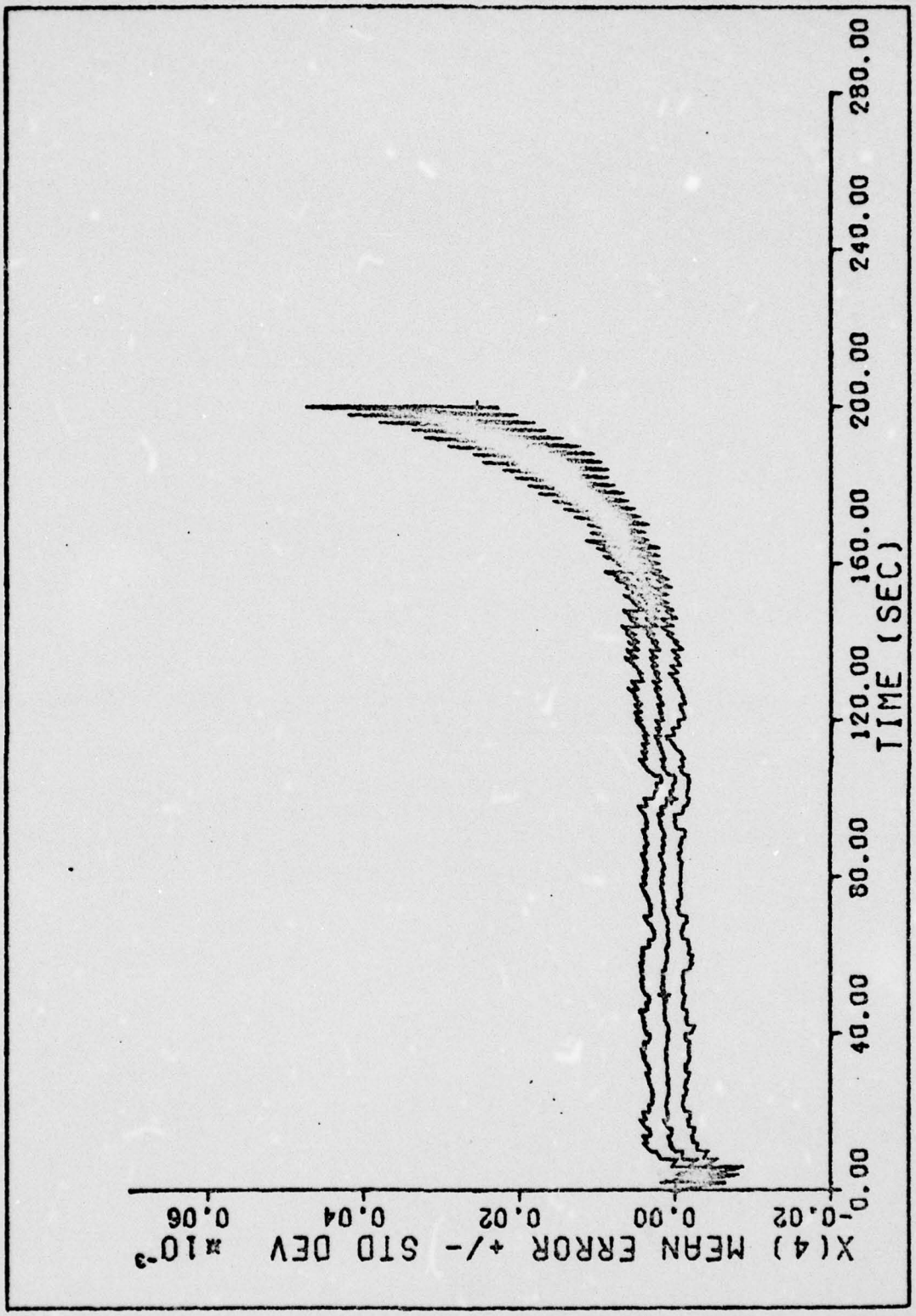


Figure 108. State x(1) Initial Condition Error - Mean Error +/- s_e vs Time

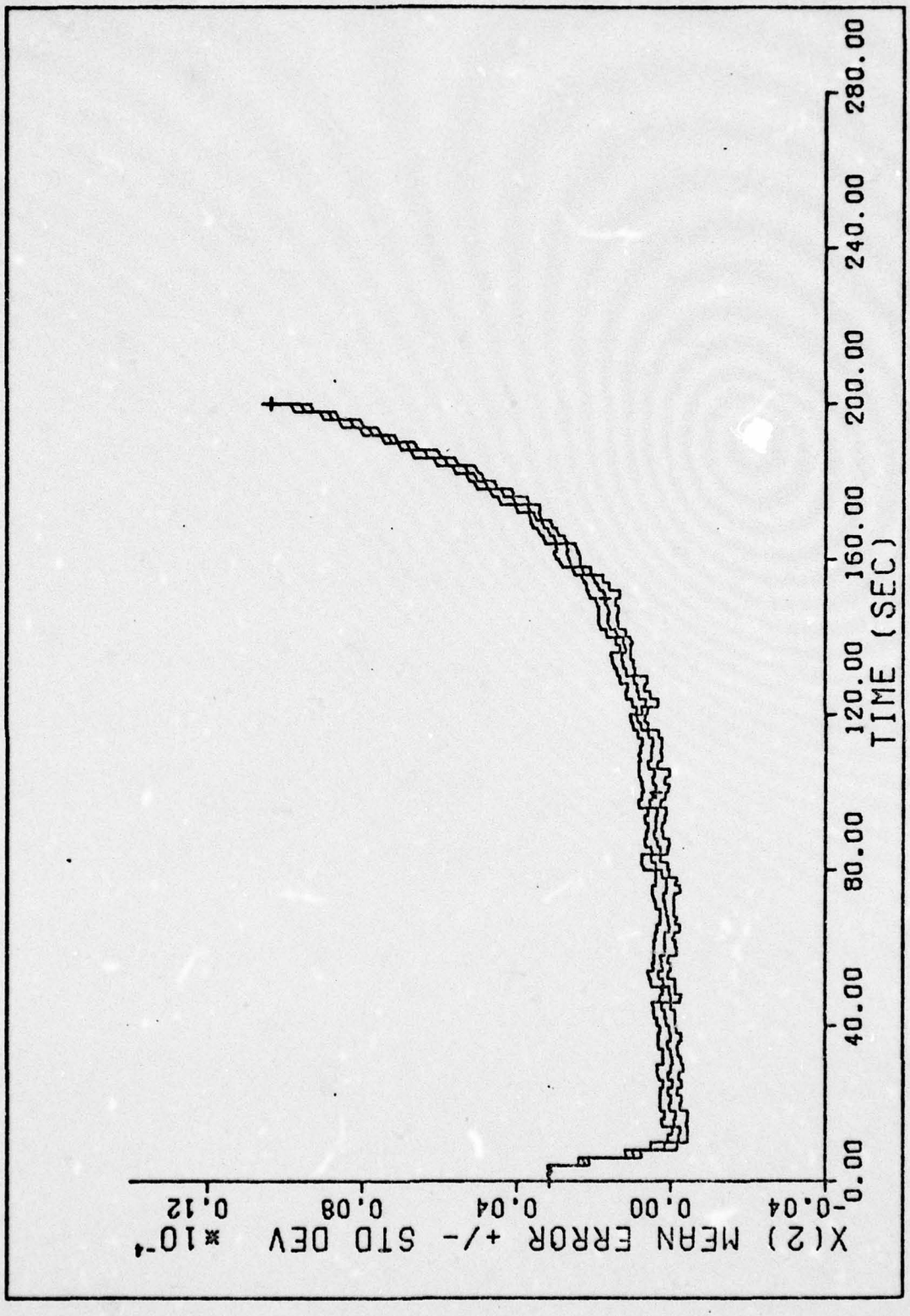


Figure 109. State x(2) Initial Condition Error - Mean Error +/- s_e vs Time

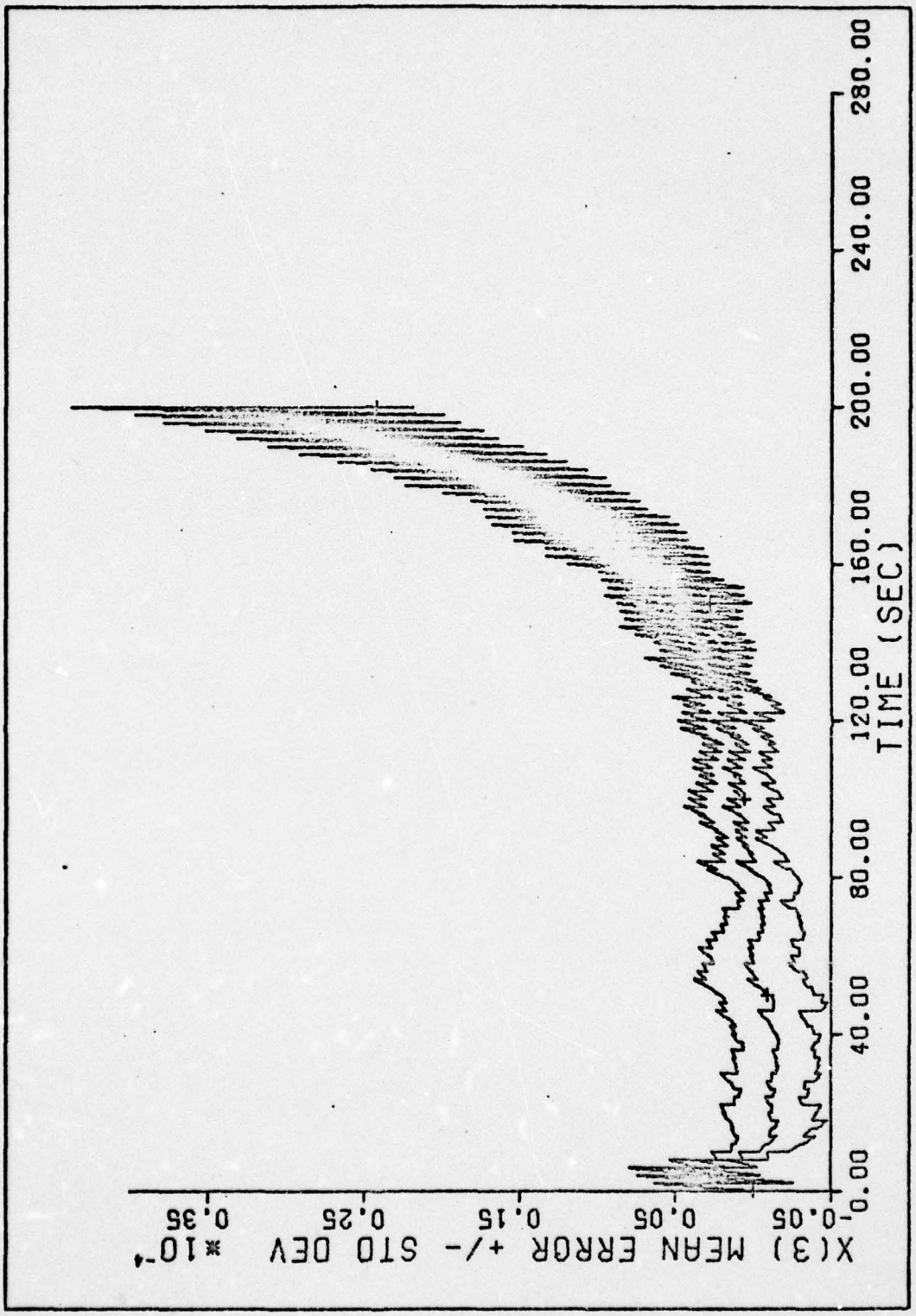


Figure 110. State x(2) Initial Condition Error - Mean Error +/- s_e vs Time

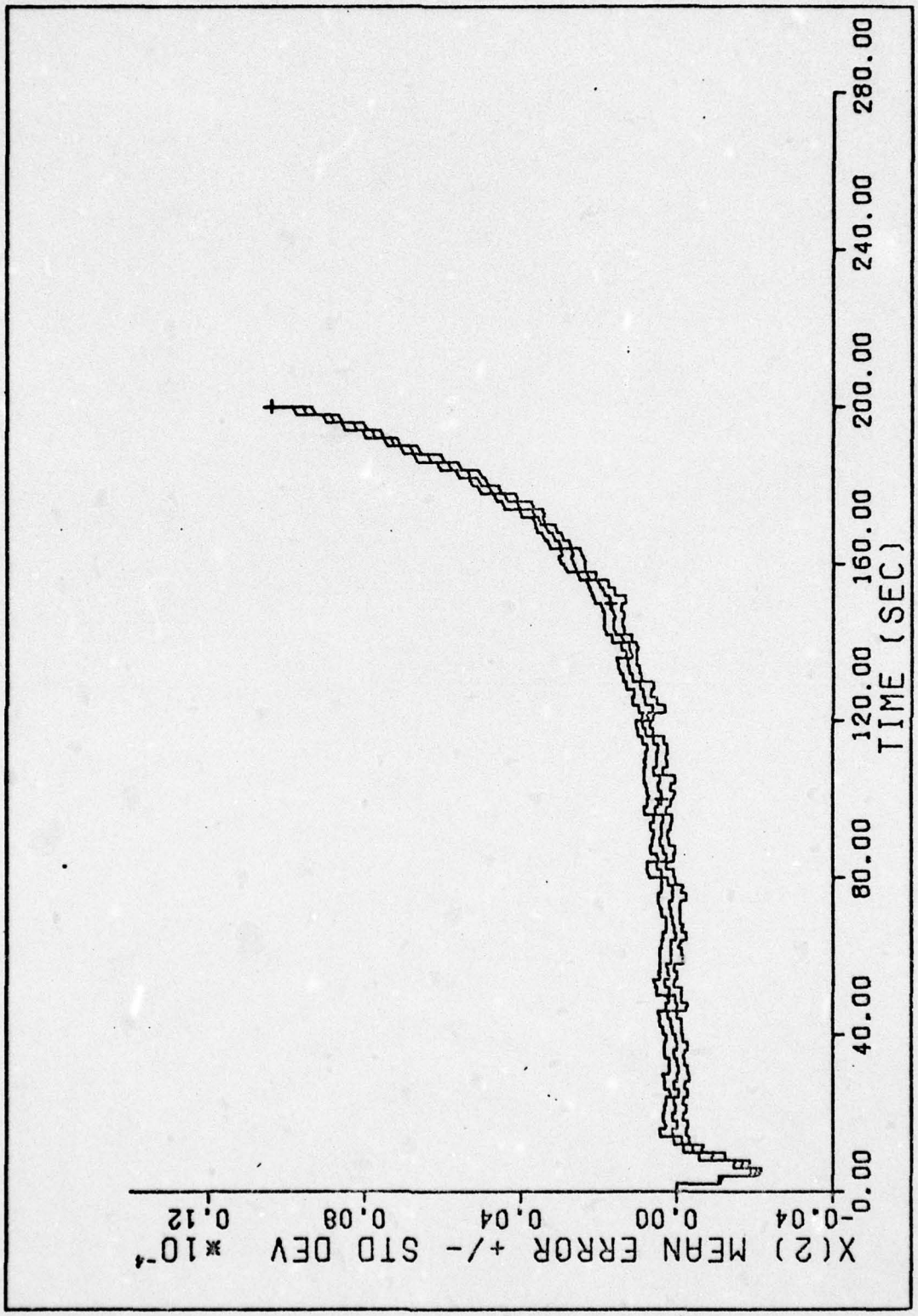


Figure 111. State x(3) Initial Condition Error - Mean Error +/- s_e vs Time

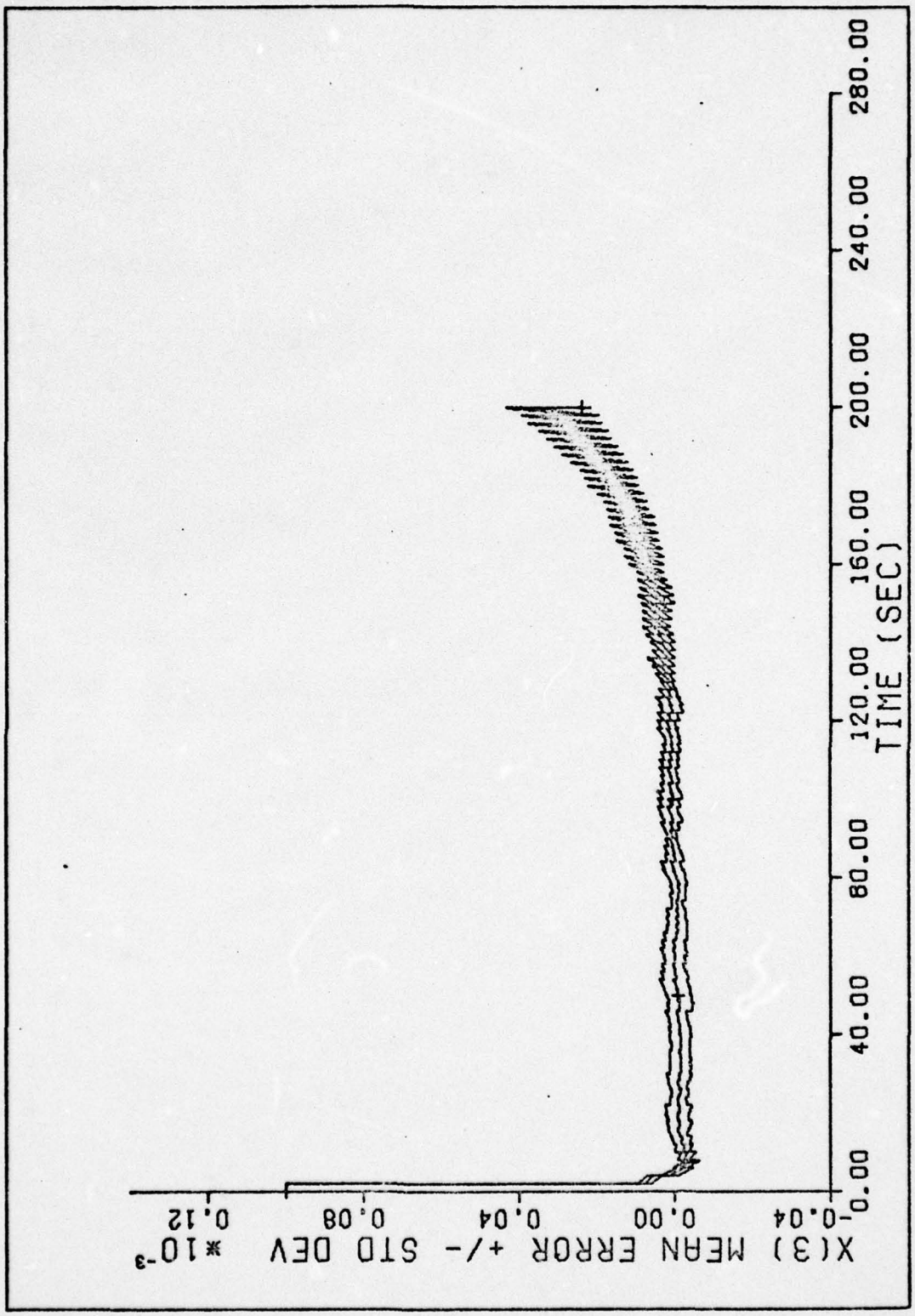


Figure 112. State x(3) Initial Condition Error - Mean Error +/- s_e vs Time

Figures 107 and 108 present the mean error plots for states x(1) and x(4) when the simulation was performed with x(1) having an initial condition error. Note in Figure 107 at time t_o state x(1) starts with an initial error of -3.2×10^{-6} rad/sec and recovers to the plot of Figure 98 within 10 seconds. This initial transient is also noted in state x(4), Figure 108, when compared to Figure 101.

The Monte Carlo simulation was subsequently performed with an x(2) initial condition error and the mean error plots for states x(2) and x(3) are presented in Figures 109 and 110. In this case, state x(2) and x(4) have an initial transient when compared to Figures 99 and 100.

Figures 111 and 112 present the mean error plots for states x(3) and x(2) when the simulation was performed with x(3) having an initial condition error. In this case, x(3) has an initial transient and recovers to Figure 100 within 10 seconds. This initial transient is also noted in state x(2); Figure 111 when compared with Figure 99, however, the transient lasts 15 seconds.

For the case with x(4) initial condition error, similar results were observed, with state x(4) having the transient and the transient also being observed in state x(1). When the Monte Carlo simulation was performed with states x(5) and x(6) having an initial condition error, no change was noted in the mean error plots. These results are not presented.

In summary, if states x(1) or x(2) have an initial condition error, then this will result in an initial transient in states x(3) and x(4)

or x(2) and x(3) respectively. This transient recovered to the case of zero initial condition error within 10 seconds. This result is also noted if the initial condition is reversed; x(3) or x(4) with initial condition error and the transient in states x(2) or x(1). The transient for this second set of cases lasted 15 seconds.

Random Initial Condition Error

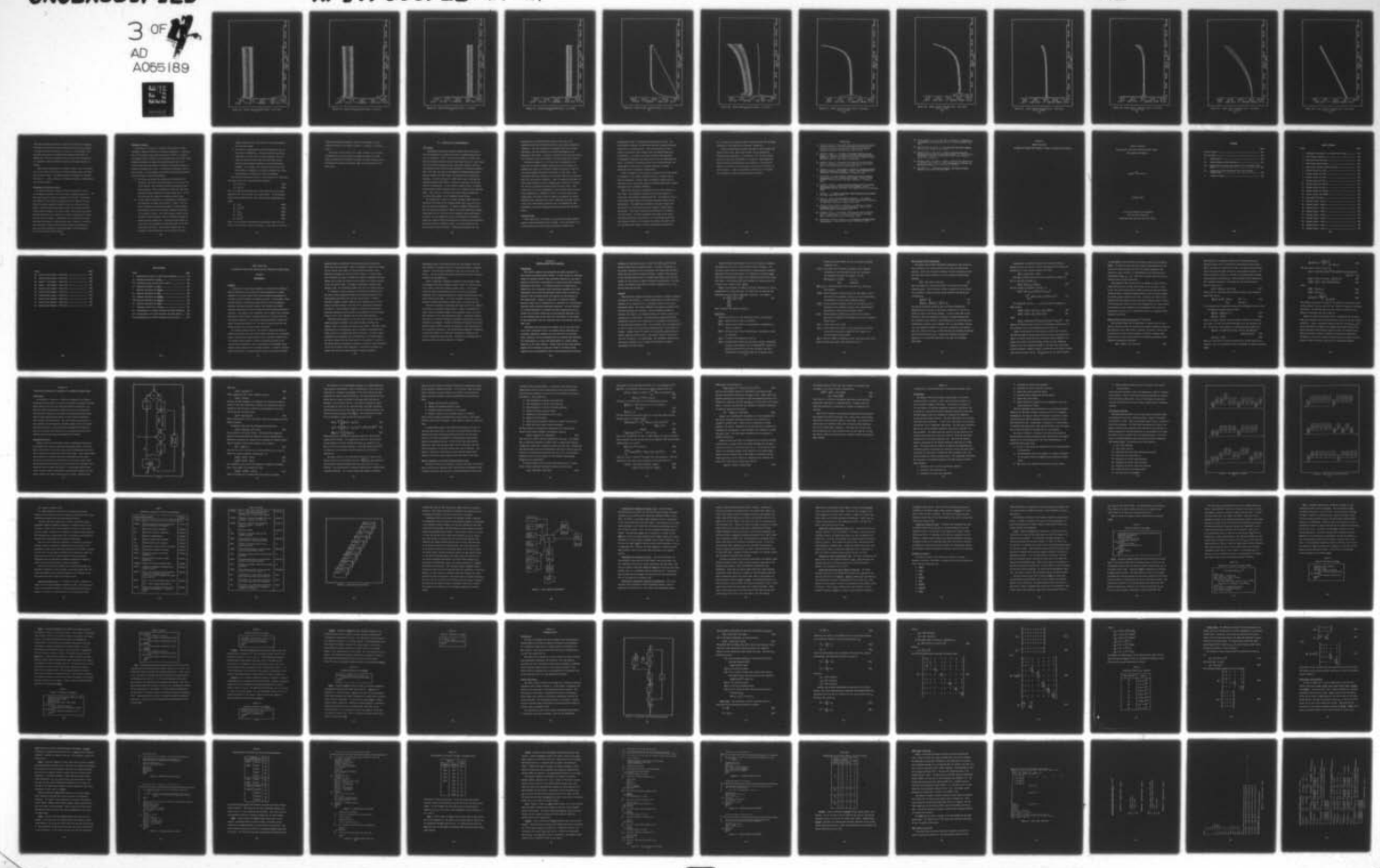
In order to simulate a condition where the filter initial conditions differ from the true system values, the filter values assumed initial values which were changed from the true value by an additive white, Gaussian noise having strength equal to the initial covariance. This was accomplished by calling subroutine NOISE (see Appendix A) in the first section of subroutine XFDOT. The value for the variable RMS was the square root of the initial covariance diagonal element. The result of NOISE, XNOISE, was added to the initial filter state value yielding a random initial condition for the initial filter states with every simulation run. While varying the truth model states would be more realistic, the initial condition value would be more difficult to obtain because of the interrelationship in the system states. (A variation in satellite position affects w_{LSy}, w_{LSz}, R, and V_r). Varying all the filter initial condition values roughly corresponds to the truth model starting from a different initial condition for each run.

The results of this Monte Carlo simulation (10 runs) is shown in Figures 113 through 124. The average filter estimate of the standard deviation is the same as case 12 (Figures 92 through 97), however, the

AD-A055 189 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 17/7
A GENERALIZED MONTE CARLO ANALYSIS PROGRAM FOR KALMAN FILTER DE--ETC(U)
DEC 77 K L JACKSON

UNCLASSIFIED AFIT/GGC/EE/77-6 NL

3 OF 4
AD
A055189



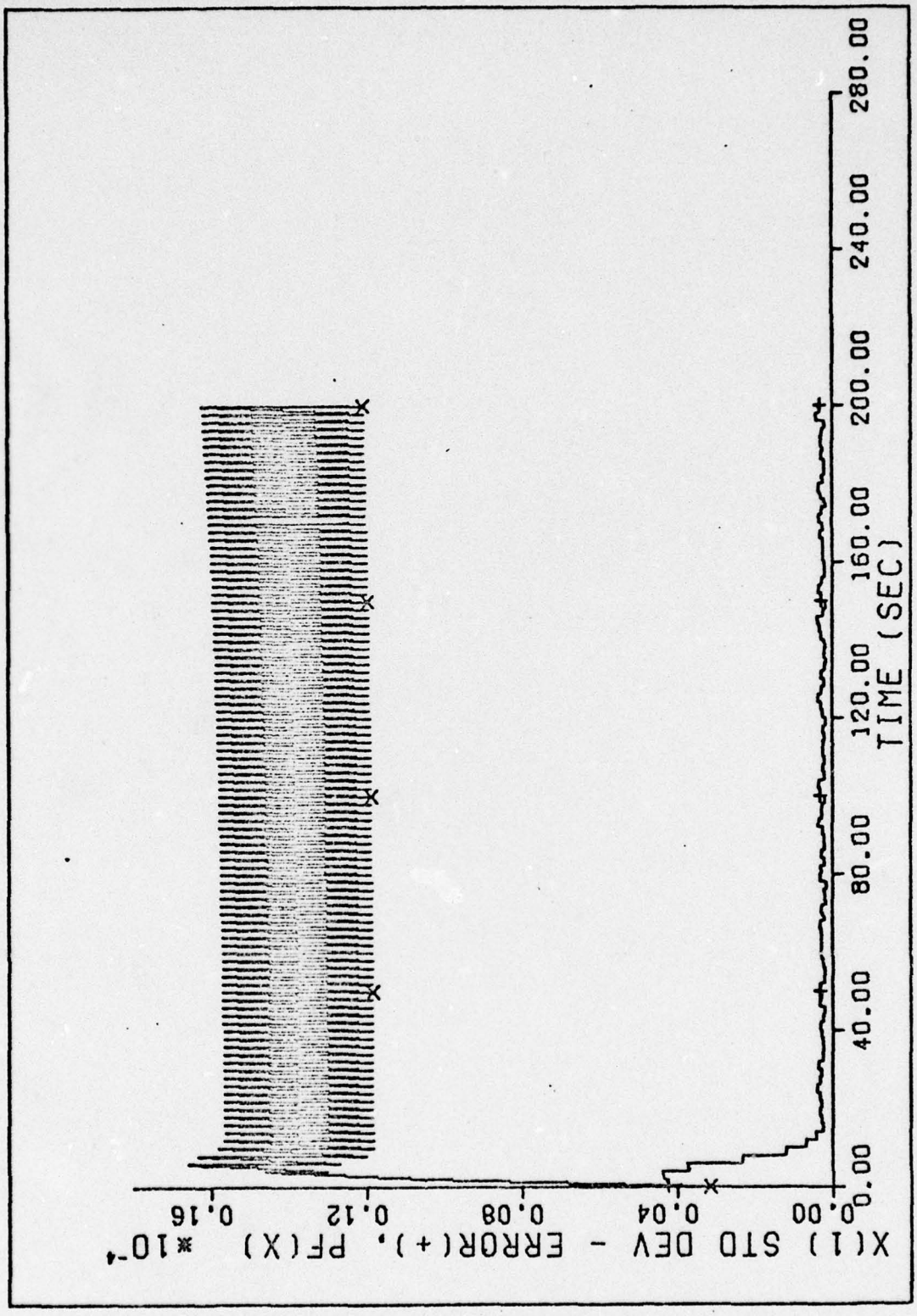


Figure 113. Random Initial Condition Error - s_e vs Time

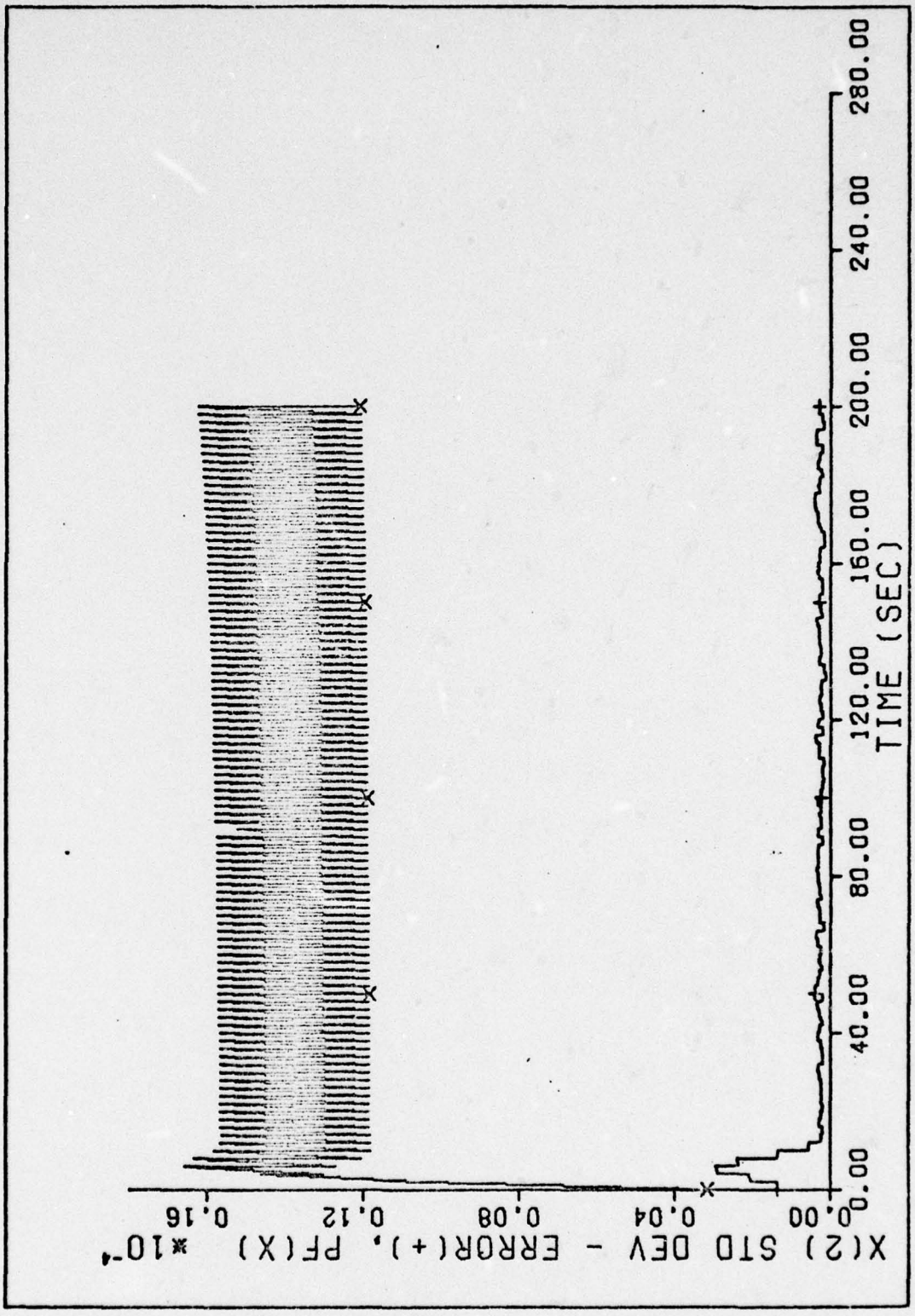


Figure 114. Random Initial Condition Error - s_e vs Time

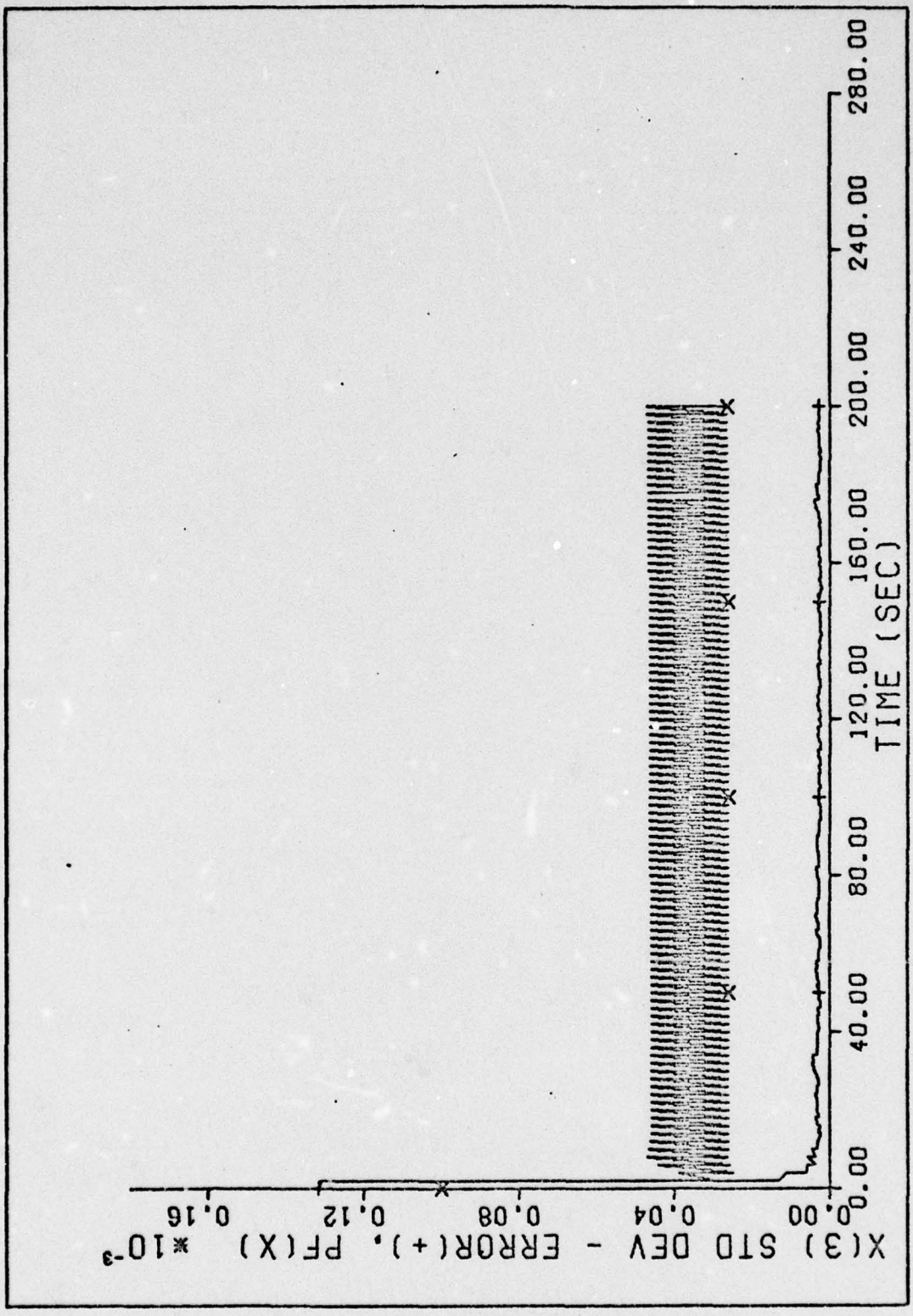


Figure 115. Random Initial Condition Error - s_e vs Time

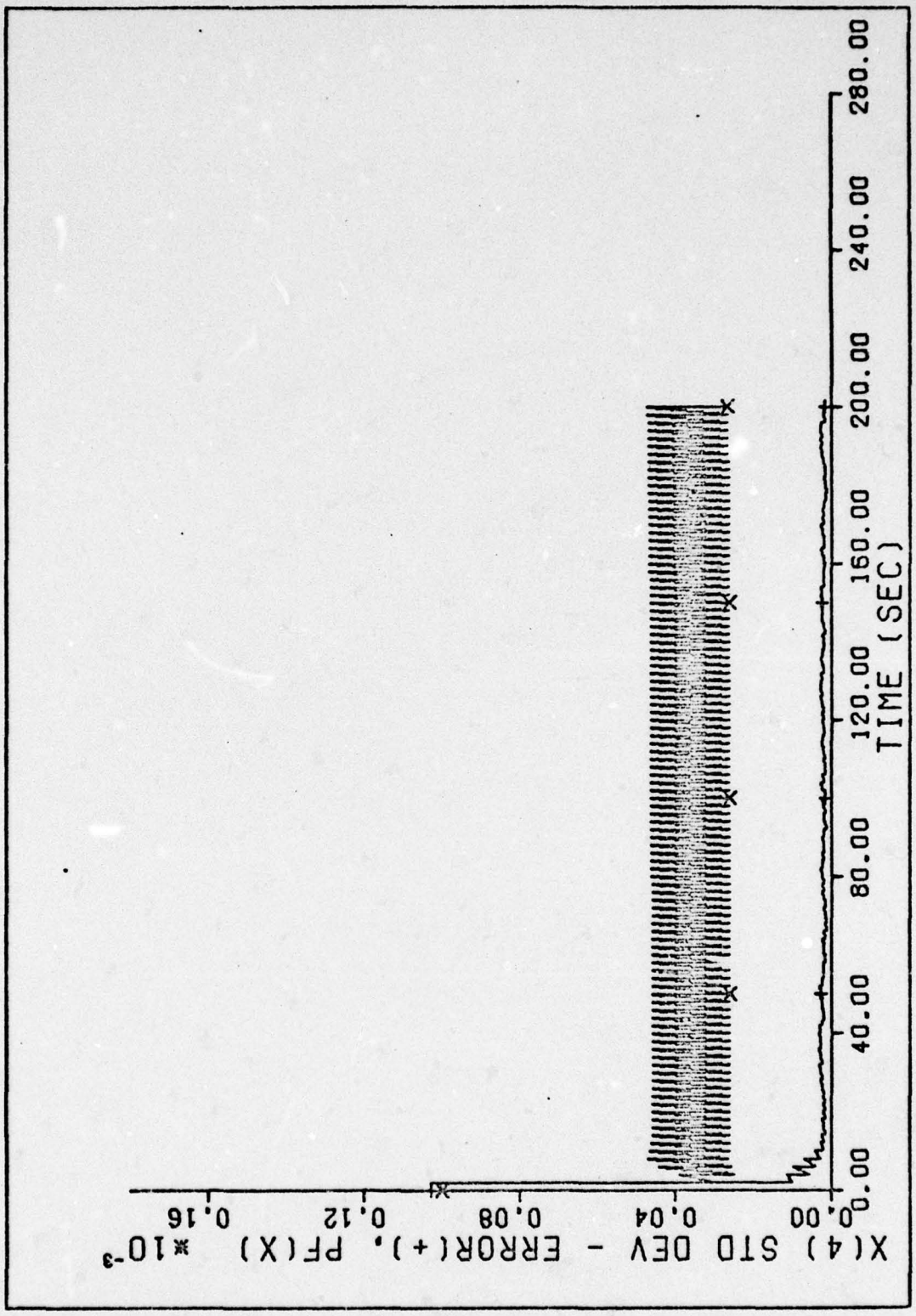


Figure 116. Random Initial Condition Error - s_e vs Time

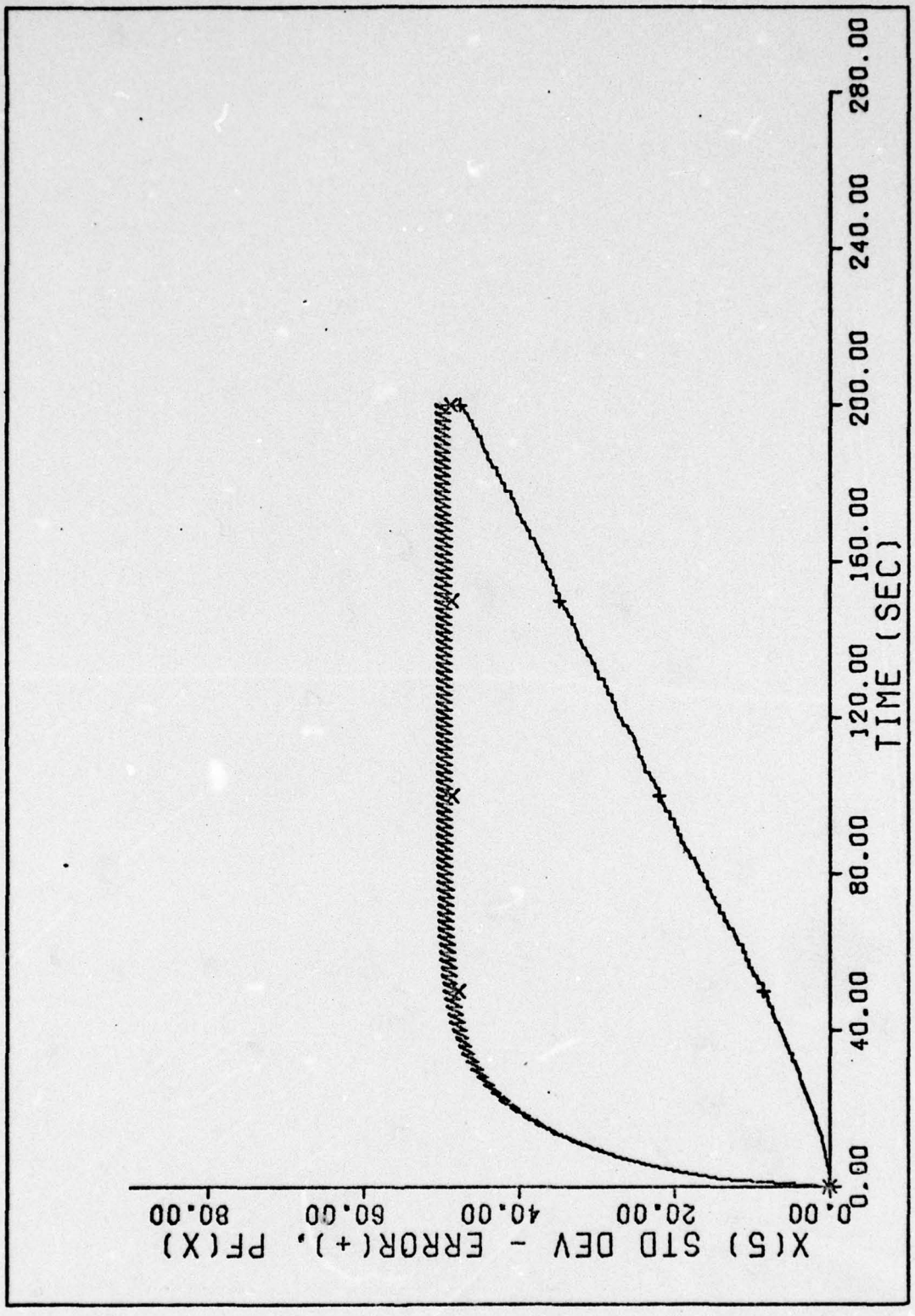


Figure 117. Random Initial Condition Error - s_e vs Time

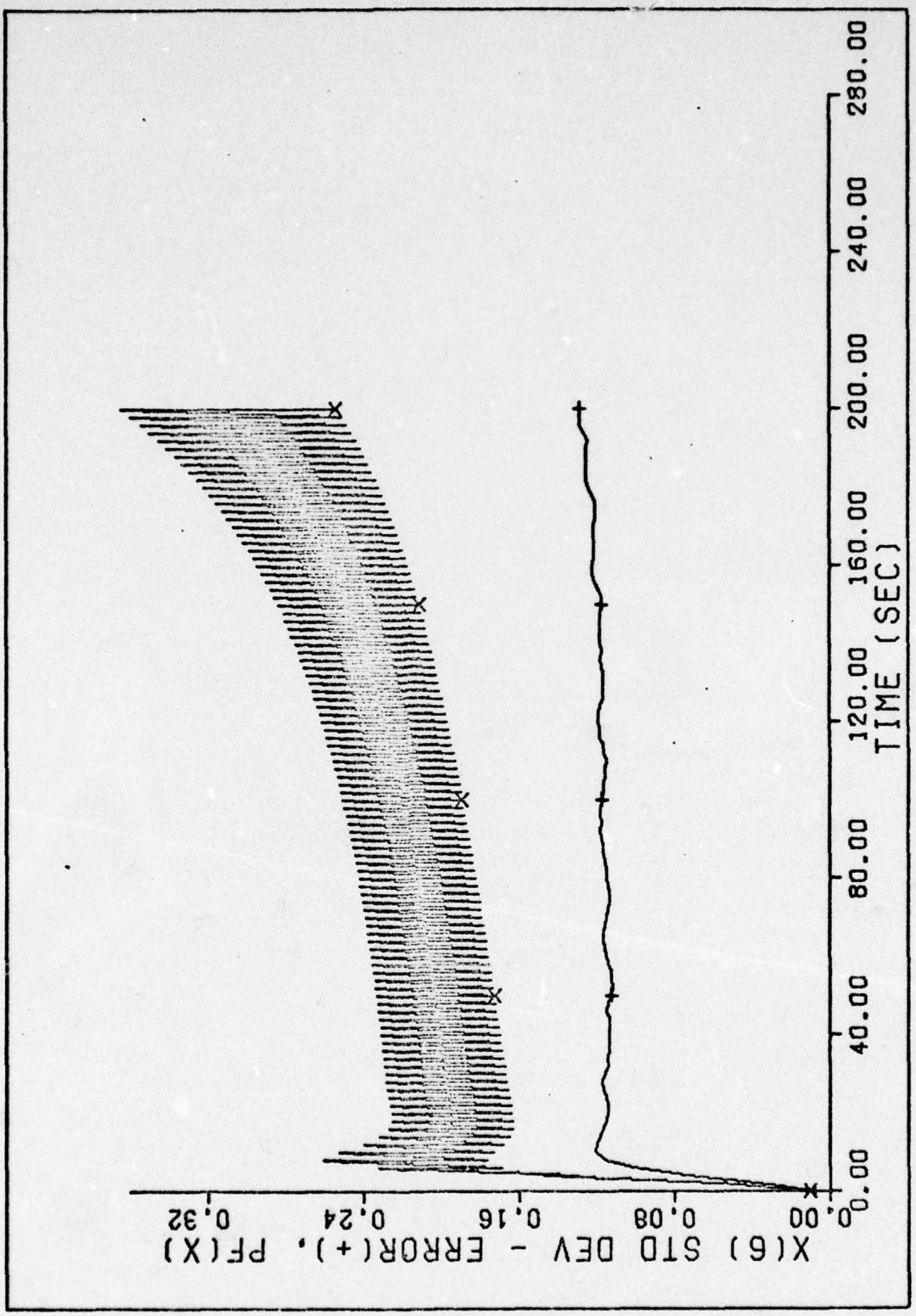


Figure 118. Random Initial Condition Error - s_e vs Time

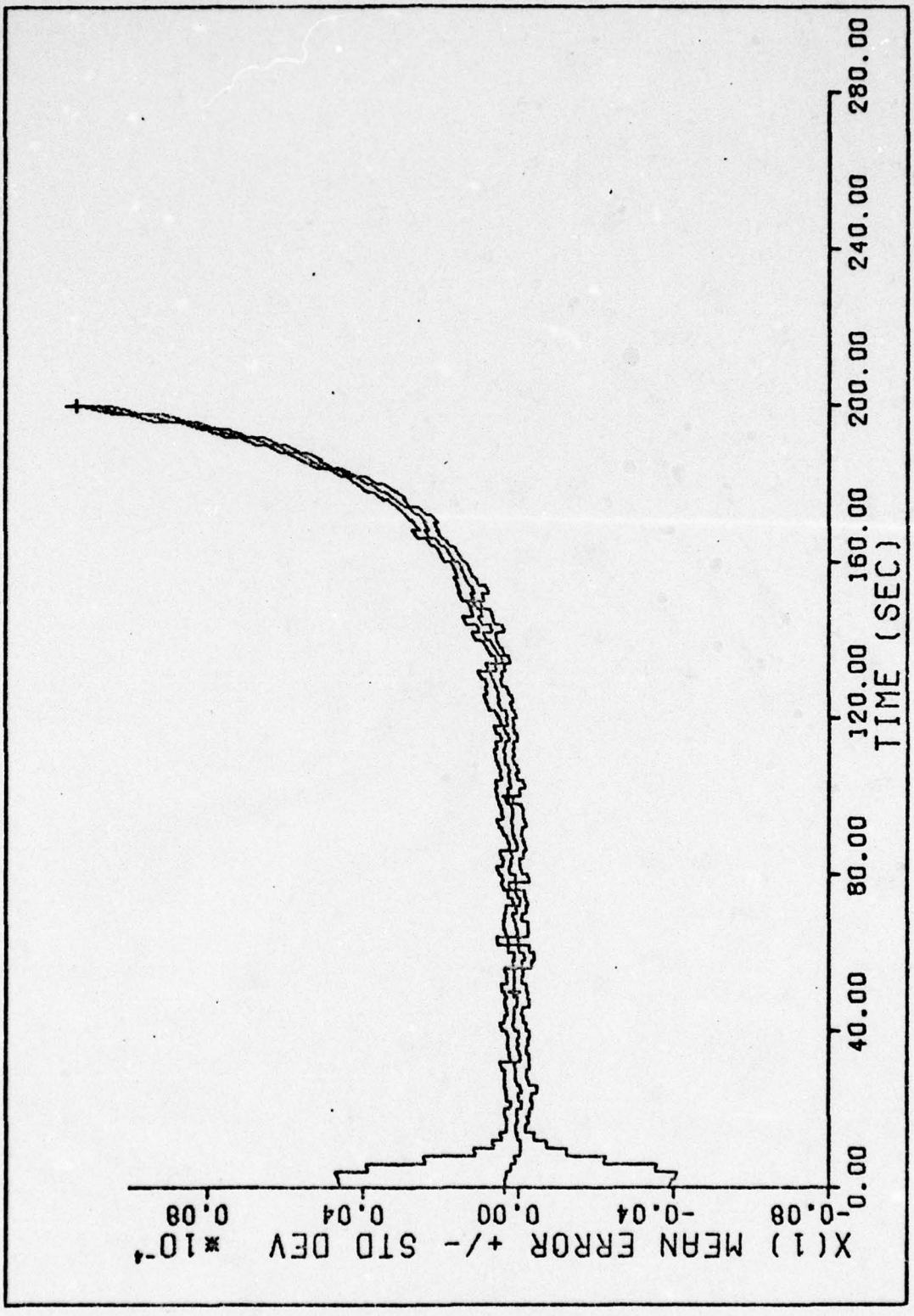


Figure 119. Random Initial Condition Error - Mean Error +/- s_e vs Time

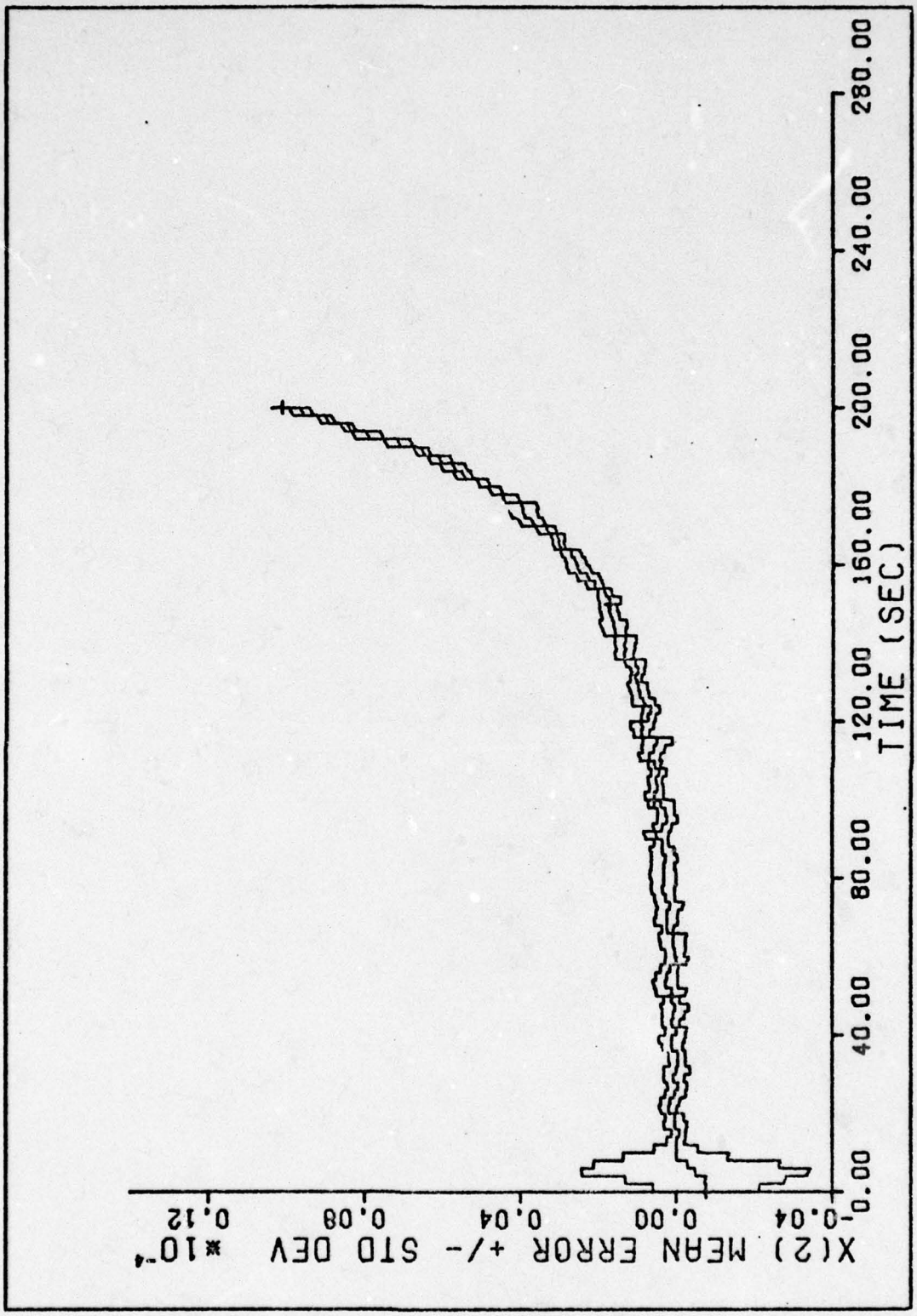


Figure 120. Random Initial Condition Error - Mean Error +/- s_e vs Time

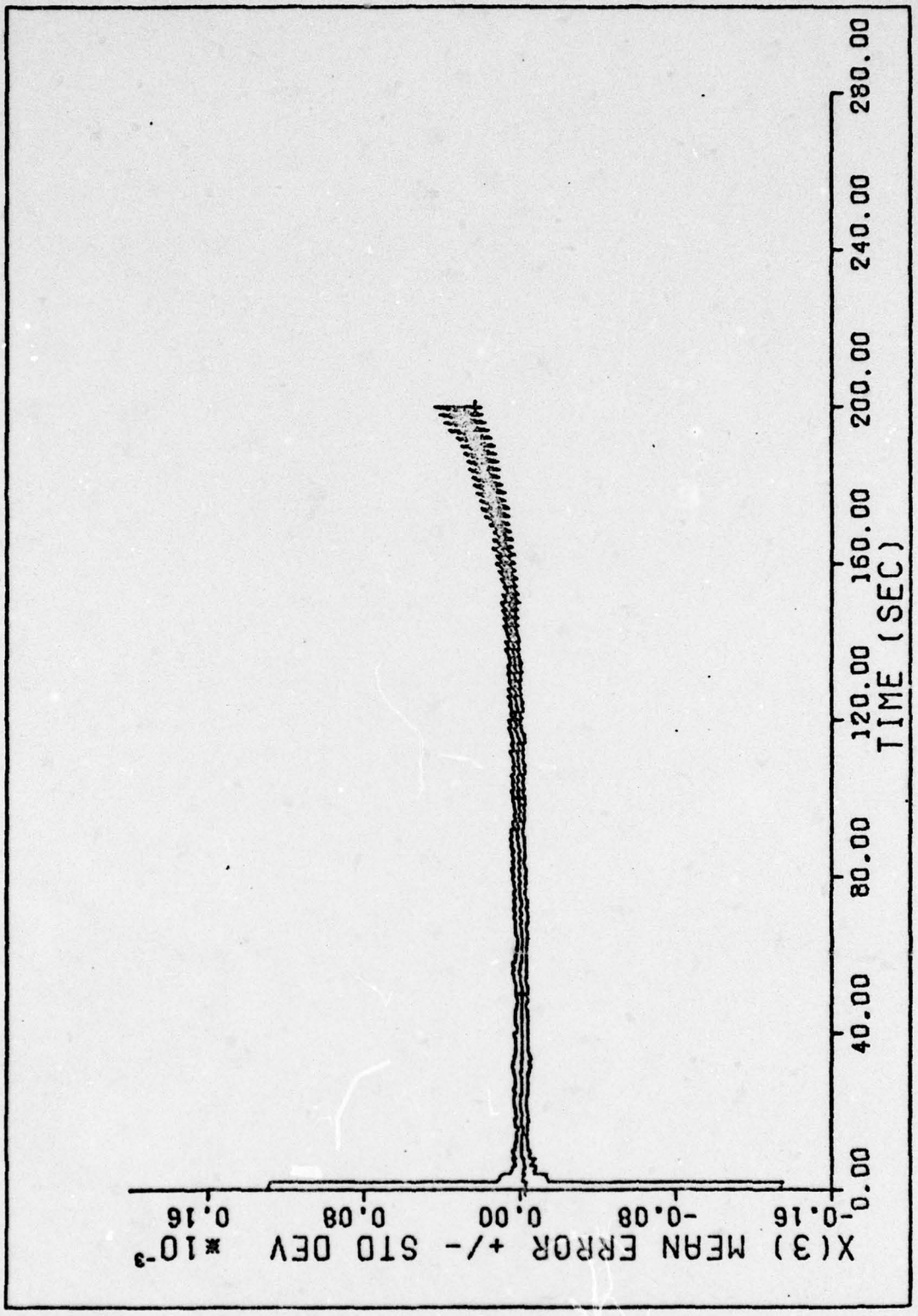


Figure 121. Random Initial Condition Error - Mean Error +/- s_e vs Time

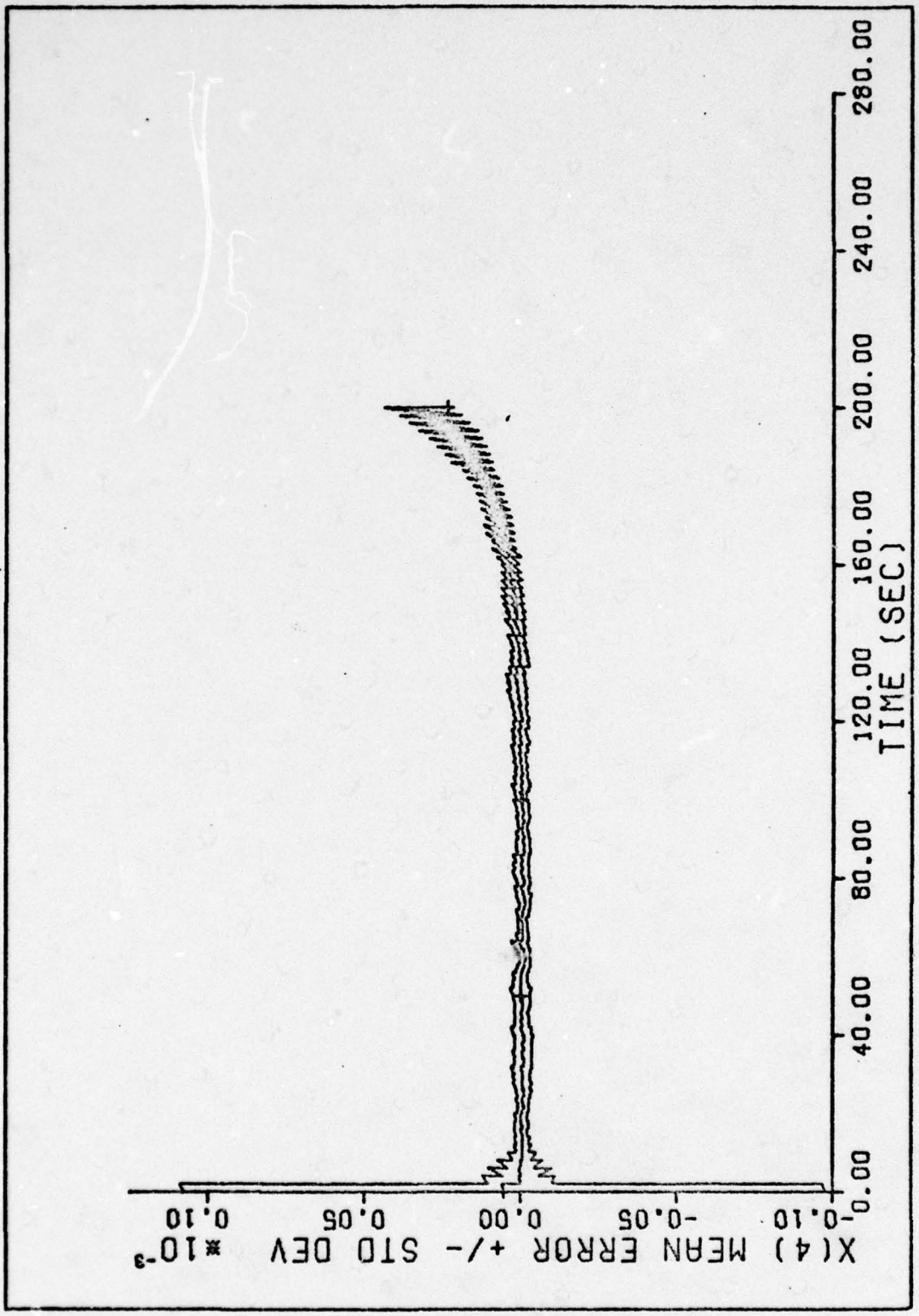


Figure 122. Random Initial Condition Error - Mean Error +/- s_e vs Time

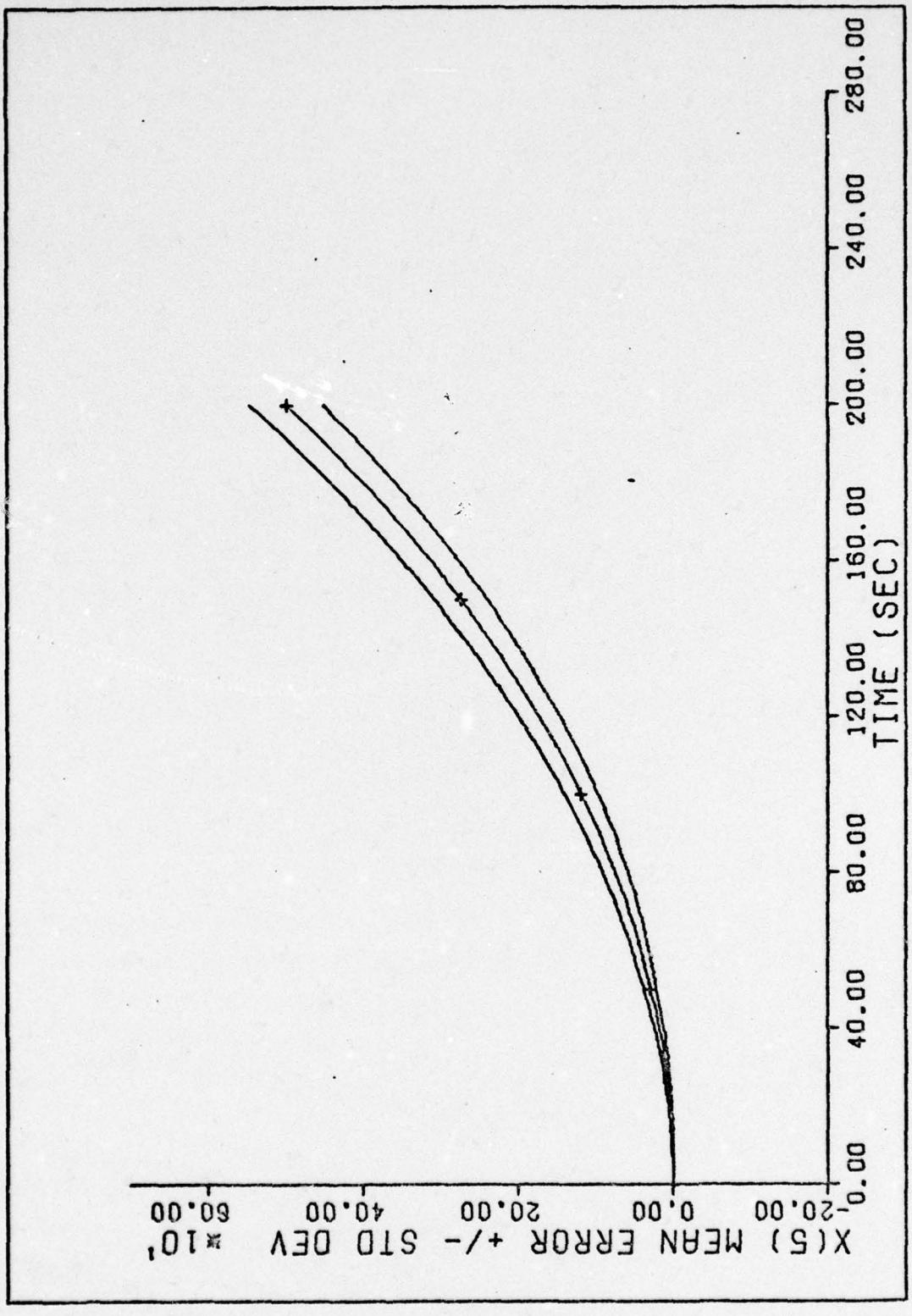


Figure 123. Random Initial Condition Error - Mean Error +/- s_e vs Time

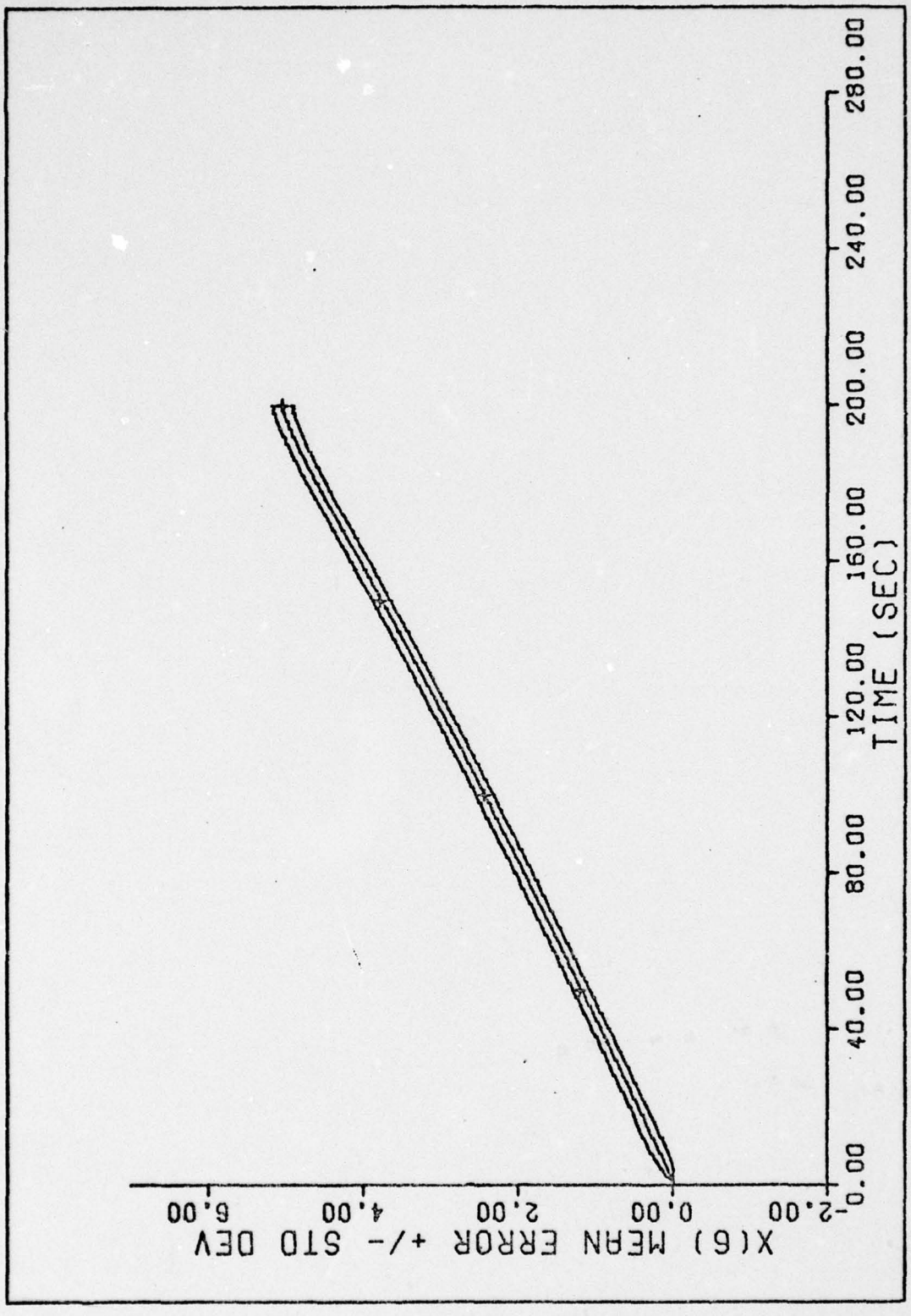


Figure 124. Random Initial Condition Error - Mean Error +/- s_e vs Time

true error standard deviation has changed for all states as expected. Note that in all cases the standard deviation for the filter estimate is greater than the true error so the filter is not divergent. The mean error plots have also changed when compared with Figures 98 through 103. The most noticeable result in states x(1) through x(4) is an initial standard deviation transient lasting approximately 15 seconds.

These results compare with the previous section, where now instead of only one filter state having an initial condition error, all states have an initial error. The transient noted is different (a standard deviation, not a mean), however, the transient duration is approximately the same (15 seconds).

Comparison with Previous Results

As noted in the users' guide for MCAP (see Appendix A) one of the stringent assumptions made for the GCAP covariance analysis is that total impulsive control is applied to all states. The problem investigated in this study (and presumably in the study of Mann and Mitchell) applied impulsive feedback to only two of the six filter states. Since the covariance analysis used in the previous studies (GCAP) assumed feedback of all error states to the system and several Fortran coding errors were detected, no direct comparison can be made with the previous studies. Consequently, the results of this study invalidate the results of their previous work. However, it should be noted that most of the corrected computer subroutines developed by Mann were easily adaptable to the requirements of the generalized Monte Carlo Analysis Program (MCAP).

Additional Comments

The tuning of a (linear or extended) Kalman filter is a time consuming, sometimes tedious and frustrating experience. In particular, if the initial tuning parameters are not "close" to a (proper but unknown) range of values, then the computations made in the finite wordlength computer may exceed the machine limits ("blow up"). If this occurs then the analyst must modify the simulation (through insight and experience) in order to determine an acceptable starting set of values. In investigating this problem three program modifications were used to start the simulation:

1. The filter and system state equations were decoupled (no feedback), the first measurement update time set to the end of the problem, and the filter covariance propagation equations bypassed. These modifications allowed the true error standard deviation transients to reach steady state. Filter $\underline{P}_o$ values were then modified considering these values.
2. In this problem formulation, no measurement is available to the filter for the range rate state $[V_r = x(6)]$. When the filter performed the measurement update, the Kalman filter gain matrix $\underline{K}$ would cause the filter estimate of range rate to be grossly in error. This error quickly rippled through the filter state equations, and the simulation exceeded the machine numeric capabilities. The program modification was to set the filter estimates of range rate equal to the true value after the update. After several simulations, the appropriate tuning parameters were determined which gave

better estimates of the range rate and so the modification was no longer needed.

3. The idea of a transformation of state variables is used in this program modification. Since this problem did not use error states for the filter or system equations, calculations are performed using large and small state values, causing severe numerical problems on a finite wordlength computer. If the large state values are scaled down, then this problem is eliminated. To better understand this idea, the development is presented in the next paragraph.

The idea of rescaling the problem is best understood by considering the linear filter state and measurement equations:

$$\dot{\underline{x}} = \underline{F}\,\underline{x} + \underline{G}\,\underline{w} \tag{202}$$

$$\underline{z} = \underline{H}\,\underline{x} + \underline{v} \tag{203}$$

Now, it is desired to rescale some of the states (the large valued states) to have values closer to the other states. If the rescaled state vector is denoted by $\underline{x}'$, then the following relationships are found:

$$\underline{x}' = \underline{T}\,\underline{x} \tag{204}$$

$$\underline{F}' = \underline{T}\,\underline{F}\,\underline{T}^{-1} \tag{205}$$

$$\underline{G}' = \underline{T}\,\underline{G} \tag{206}$$

$$\underline{H}' = \underline{H}\,\underline{T}^{-1} \tag{207}$$

$$\underline{P}' = \underline{T}\,\underline{P}\,\underline{T}^{T} \tag{208}$$

where T is the diagonal transformation (rescaling) matrix that contains the scale factors along the diagonal. In the case of non-linear

state and measurement equations, then the relationships are best found by writing out the matrix equations to determine the relationships.

The problem investigated in this study used this idea of matrix transformation to rescale the filter range and range rate states [x(5) and x(6)] and the satellite position, satellite velocity, range and range rate states [x(1) through x(6), x(11), and x(12)] in the truth model.

VII. Conclusions and Recommendations

Conclusions

A generalized Monte Carlo Analysis Program (MCAP) which evaluates the performance of either the linear or extended Kalman filter has been developed. MCAP is similar in structure to a widely used General Covariance Analysis Program (GCAP) [Ref 2], and so users should have little difficulty performing a Monte Carlo analysis after the (linearized) Kalman filter has been analyzed by a covariance analysis. Also, with MCAP, the user can now analyze an extended Kalman filter properly because the state values are available in the Monte Carlo analysis. As noted in Chapter VI, MCAP can be used to evaluate the (extended) Kalman filter under a variety of conditions, for example, results of performance: to zero initial condition error, to single/multiple state initial condition error, or to random initial condition error. Thus, MCAP is a versatile and useful computer program for the design and analysis of the (extended) Kalman filter.

The feasibility of using a six state extended Kalman filter for estimating the states of the tracking system (w_{LSy}, w_{LSz}, $\delta\epsilon$, $\delta\nu$, R, and V_r) has been demonstrated. As noted in Figures 19 through 103, when the tuning parameters are varied, the filter performance changes accordingly, and so the filter can be tuned for optimum performance. In general one can postpone the onset of divergence with constant $\underline{Q}$ parameters, however better performance can be achieved with nonstationary noise or using an adaptive filter. It is also noted that the filter produces biased estimates. These biased estimates are very

characteristic of extended Kalman filters, and is due to the neglected higher order effects inherent in using linear perturbation techniques. The more pronounced the nonlinearities are in a given application, the more serious one can expect performance to be degraded by this effect. In order to compensate for these bias errors, bias correction terms can be added to the filter model (Ref 3:9-44).

As noted in Chapter III, a filter often is considered to be tuned when the filter estimated standard deviation deviations are approximately equal to the computed standard deviations or when the true error standard deviations are smallest in some sense. This "covariance matching" technique may not always yield the best performance for a given set of conditions. This is observed for the case of zero initial condition error by comparing case 7 (Figures 67 through 72) and case 12 (Figures 98 through 103) mean error plots. Even though case 12 is "more appropriately" tuned (compare states 5 and 6) better performance is indicated in case 7 by the mean error plots. Consequently, the filter should be tuned by considering the problem conditions (how accurately the initial conditions are known, typical versus worst case tracking profiles, etc.) and obtaining the best performance in both the standard deviation plots and the mean error plots.

Recommendations

Even though MCAP, as written, is a general Monte Carlo analysis program, further improvements can be added. These improvements will be discovered after several different estimation problems are

analyzed using MCAP. Two additions currently come to mind. The first addition concerns the white Gaussian noise generator presently implemented in subroutine NOISE. This noise generator executes rapidly and provides random numbers which are approximately white and Gaussian. If a better but slower executing noise generator is desired; the implementation shown in Ref 17 can be substituted for NOISE. In addition to providing better white Gaussian numbers, the implementation shown in Ref 17 features computer to computer (different types of computers) repeatibility.

Since, in general, the user may want to specify the filter method of integration, the second improvement would be to modify MCAP to add this capability. Several filter integration methods could be programmed and the user would select the desired method with a problem dependent control parameter ($USRCTL).

Further work can also be recommended for the filter analyzed in this study. The performance variation of the filter should be studied for many different tracking profiles. The filter can then be tuned (both standard deviation and mean) considering these several trajectories. This additional information will give insight into the nature of the bias correction terms needed to improve filter performance. These bias correction terms should then be added to the filter model. If these correction terms still do not improve performance adequately, then either: 1) a time varying $\underline{Q}$ and $\underline{R}$ as some unknown function of the angular rates and/or range data could be determined after studying several representative trajectories,

or 2) an adaptive $\underline{Q}$ technique could be used based upon the real time evaluation of the innovations (residuals) sequence(s).

Several simplifying assumptions were made in order to limit the scope of this study. Additional analysis can be performed by relaxing these assumptions. In particular: The resolver measurements of θ and ϕ were considered to be perfect. These parameters should be modeled stochastically and included in a future filter analysis to determine the performance sensitivity of the filter to these variables. Also, the performance sensitivity of the filter to accelerometer noise corruptions should be studied.

Bibliography

1. Heller, Warren G., "Covariance Propagation Equations for Optimal and Suboptimal Kalman-Filter-Integrated Multisensor Inertial Systems", Technical Report TR-312-2, The Analytic Sciences Corporation, Reading Massachusetts, April 7, 1975.

2. Hamilton, Edward L., The General Covariance Analysis Program (GCAP), An Efficient Implementation of the Covariance Analysis Equations. Unpublished computer program guide, Wright-Patterson Air Force Base, Air Force Avionics Laboratory.

3. Maybeck, Peter S. "Stochastic Estimation and Control Systems: Part II", unpublished class notes, Wright-Patterson Air Force Base: Air Force Institute of Technology, 1975.

4. Mitchell, R. A. K., High Accuracy Aircraft to Satellite Tracking Using an Extended Kalman Filter. Thesis, Wright-Patterson Air Force Base: Air Force Institute of Technology, 1975.

5. Mann, Robert E., High Accuracy Aircraft to Satellite Tracking: An Evaluation of Two Proposed Filter Models. Thesis, Wright-Patterson Air Force Base: Air Force Institute of Technology, 1976.

6. Pollard, Joseph J., Orbital Parameter Determination by Weighted Least Square Error and Kalman Filtering Methods. Thesis, Wright-Patterson Air Force Base: Air Force Institute of Technology, 1973.

7. Meditch, J. S., Stochastic Optimal Linear Estimation and Control. New York, McGraw-Hill, 1969.

8. Gelb, A., editor, Applied Optimal Estimation. The Analytic Sciences Corporation, The M. I. T. Press, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1974.

9. Wrigley, Walter; Walter M. Hollister, and William G. Denhard, Gyroscopic Theory Design and Instrumentation. The M. I. T. Press, Cambridge, Massachusetts, 1969.

10. Maybeck, Peter S. "Stochastic Estimation and Control Systems: Part I", unpublished class notes, Wright-Patterson Air Force Base: Air Force Institute of Technology, 1975.

11. Mendenhall, William, Introduction to Probability and Statistics, Third Edition. Duxbury Press, Belmont, California, 1971.

12. Baker, Robert M. L., Jr., and Maud W. Makemson, An Introduction to Astrodynamics, 2nd Edition, New York, London, Academic Press, 1967.

13. Kayton, Myron and Walter R. Fried, Avionics Navigation Systems, New York, John Wiley and Sons, 1969.

14. Asher, Robert B. and David H. Watjen, Kalman Filtering for Precision Pointing and Tracking Applications. Air Force Avionics Laboratory, Wright-Patterson Air Force Base, AFAL technical report to be published.

15. Pearson, John B. and Edwin B. Stear, "Kalman Filter Applications in Airborne Radar Tracking", IEEE Transactions on Aerospace and Electronic Systems, Vol AES-10, No. 8, May 1974.

16. Jazwinski, A. H., Stochastic Processes and Filtering Theory. Academic Press, Inc., New York, 1970.

Appendix A

Users' Guide for:

A Generalized Monte Carlo Analysis Program for Kalman Filter Design

USERS' GUIDE FOR:

A GENERALIZED MONTE CARLO ANALYSIS PROGRAM (MCAP)

FOR KALMAN FILTER DESIGN

by
Kenneth L. Jackson, Jr.

December 1977

Air Force Institute of Technology
Air Force Air University
Wright-Patterson Air Force Base Ohio 45433

Contents

List of Figures

List of Tables

USERS' GUIDE FOR:

A GENERALIZED MONTE CARLO ANALYSIS PROGRAM FOR KALMAN FILTER DESIGN

Section I

Introduction

Overview

A large class of estimation problems is concerned with finding an optimal estimate of some quantity (an unknown parameter, a random variable, or a random signal) given noise-corrupted measurements of a function of this quantity (corrupted by noise). In particular, there are many problems with aerospace applications, such as navigation, guidance, and weapon delivery systems, which require that noise-corrupted measurements be used to estimate certain physical parameters precisely. For instance, the techniques used to combine these external measurements with Inertial Navigation System (INS) outputs fall into two general categories: conventional continuous-feedback damping and Kalman filter damping (Ref 1:1). The trend in recent years has been toward extensive use of Kalman filter techniques.

In complex aerospace systems, the number of parameters needed to describe the system (the true system model) accurately may be extremely large. The real time implementation of this true system model within the "optimal" Kalman filter is often not practical because of the large number of parameters, memory requirements, and resultant computational burden imposed on the airborne computer. In order to obtain a Kalman filter which is computationally feasible, intentional modeling

approximations are introduced by deleting nondominant states and terms from the system model. Thus a "suboptimal" Kalman filter based on this reduced order model will give poorer performance, from a statistical standpoint, than the optimal filter. Computer simulation techniques are used to develop the required compensation (filter tuning) and to study the effects of uncertainties (an error analysis) in the true system model. Two types of simulation techniques are commonly used: the covariance analysis and the Monte Carlo analysis.

A covariance analysis generates the second order statistics of the error between suboptimal Kalman filter state estimates and the corresponding parameters from the true system model. A General Covariance Analysis Program (GCAP) has been developed by the United States Air Force Avionics Laboratory and is coming into widespread use (Ref 2). The covariance analysis is somewhat limited because stringent assumptions (see Appendix A) are necessary for the analysis results to be a valid depiction of the error characteristics.

The Monte Carlo analysis, on the other hand, actually conducts a sample-by-sample simulation using random number generators and shaping filters to generate the random error sources. The noisy system measurements are then processed by the suboptimal (extended) Kalman filter algorithm to generate the filter estimates. If many of these simulations are conducted, then the statistics of the error between the filter estimate and the truth model can be computed. Because of the larger number of simulations necessary to determine the performance of one set of Kalman filter parameters, a Monte Carlo analysis can require the expense of large amounts of computer resources.

Consequently when a covariance analysis can be performed, the Monte Carlo analysis is viewed as a step to be performed after a covariance analysis. The Monte Carlo analysis is thus used to fine tune the Kalman filter and validate the filter performance after the covariance analysis has been performed.

Since a Monte Carlo analysis of promising filter designs should be performed after the covariance analysis, there is a definite need for a general Monte Carlo Analysis Program (MCAP). This program should be similar in structure to GCAP so that users can easily transition from a covariance analysis to a Monte Carlo analysis of the Kalman filter under study. It is also desirable that the problem dependent computer subroutines be applicable to both GCAP and MCAP.

This report presents a description and users guide for the generalized Monte Carlo Analysis Program (MCAP). Section II presents the Kalman filter and extended Kalman filter equations used in the Monte Carlo analysis of the suboptimal Kalman filter. Section III briefly describes the Monte Carlo analysis simulation method and how the error statistics are computed. Section IV describes the general Monte Carlo Analysis Program (MCAP) structure and provides a brief description of the constituent subroutines. The documentation presented is sufficient to allow the reader to use MCAP for the analysis, evaluation, and design of suboptimal Kalman filters. This report is concluded with a sample problem using MCAP, presented in Section V.

Section II
Extended Kalman Filter Equations

Introduction

This section presents the propagation and update equations for both linear and extended Kalman filters. A Kalman filter is, concisely stated, an optimal recursive data processing algorithm for the determination of the states or parameters of a system using noise corrupted measurements (Ref 6:16). If a physical system of interest can be modeled by a set of ordinary linear differential equations and linear measurements with system and measurement noises which are white and Gaussian, then the Kalman filter will provide the best estimate of the system states. However, in many cases of practical interest, physical systems must be represented by a nonlinear set of differential equations and/or nonlinear measurement equations. For such problems, it is often convenient to linearize the system equations about some assumed set of nominal conditions and use developed algorithms (such as the extended Kalman filter which uses reevaluation of the nominal at each measurement time) for estimation about these nominal conditions (Ref 7:57).

Considering the approximations necessary and the fact that there is no "best" suboptimal filter, the extended Kalman filter gain and covariance propagation equations have the same form as the Kalman filter equations, but are linearized about the current state estimates. The linearization is a first term approximation to a Taylor series expansion to the state estimate. Higher order and more exact approximations can be achieved by using more terms of the Taylor series expansion for the nonlinearities and by deriving approximate recursive

relations for the higher moments of the state vector $\underline{x}$ (Ref 8:184), yielding higher order nonlinear filters. As might be expected, if the system nonlinearities are significant, then neglecting the higher order terms will result in biased estimates. However, when compared to the extended Kalman filter, the higher order filters are both more complex and more costly in terms of computer implementation. For this reason, the extended Kalman filter is often considered first in nonlinear estimation problems.

Notation

This study has adopted the notation presented by Wrigley, Hollister, and Denhard (Ref 9:20-23). A vector (represented by a letter with an underbar, $\underline{x}$) is considered to be a geometric entity in real, three dimensional space. The vector represents some physical quantity which has both magnitude and direction. When the physical quantity is measured with respect to a coordinate frame, the vector is said to be coordinatized in that frame. The three numbers associated with the mathematical vector are the components of the physical vector relative to the specified coordinate frame. As an example, if $\underline{x}$ is coordinatized in the "i" frame, the vector would be denoted by $\underline{x}^i$, a three-tuple of numbers. Another vector of interest is the physical angular-velocity vector, generally denoted by a $\underline{w}$. The angular velocity vector will have two subscripts, as an example $\underline{w}^i_{nb}$. The subscripts indicate that the angular velocity, $\underline{w}$, is of frame b with respect to frame n, coordinatized in the i frame.

A 3x3 direction cosine matrix is used in this study to transform the components of a vector in one frame to those in another. The direction cosine matrix is represented by a capital $\underline{C}$ and an underbar. Associated with the letter $\underline{C}$ will be a subscript for the frame *from* which the transformation is made, and a superscript for the new coordinate frame. As an example, $\underline{C}_b^n$ would transform the vector $\underline{x}$ from the b frame to the n frame, as $\underline{x}^n = \underline{C}_b^n \underline{x}^b$.

Where it is necessary to address individual components of a vector coordinatized in a specific frame, the vector will be specified and subscripts used to indicate individual components. For example:

$$\underline{w}^i = \begin{bmatrix} w_x^i \\ w_y^i \\ w_z^i \end{bmatrix} = \begin{bmatrix} w_x^i & w_y^i & w_z^i \end{bmatrix}^T \qquad (1)$$

where T denotes the transpose operator.

Definitions

Listed below are some of the definitions used in this chapter:

$\underline{x}(t_i)$ = system state at time t_i (n-vector)

$\hat{\underline{x}}(t_i^-)$ = filter estimate prior to incorporating a measurement at time t_i (n-vector)

$\hat{\underline{x}}(t_i^+)$ = filter estimate after incorporating a measurement at time t_i (n-vector)

$\underline{z}(t_i)$ = m vector of measurements at time t_i

$\underline{w}(t_i)$ = system dynamics white Gaussian noise s-vector, independent of $\underline{x}(t_o)$, where $E[\underline{w}(t)] = \underline{0}$, and $E[\underline{w}(t)\underline{w}(\tau)^T] = \underline{Q}(t)\delta(t-\tau)$. [$\underline{w}(t)$ is assumed to be zero mean, Gaussian, and white (uncorrelated in time)] with $\underline{Q}(t)$ an sxs positive semi-

definite symmetric matrix that is, in general, piecewise continuous in t.

$\underline{v}(t_i)$ = zero mean, white Gaussian, measurement noise sequence independent of $\underline{w}(t)$ and $\underline{x}(t_o)$ for all time (m-vector). The statistics of $\underline{v}(t_i)$ are $E[\underline{v}(t_i)] = \underline{0}$, and

$$E[\underline{v}(t_i)\underline{v}(t_j)] = \begin{cases} \underline{R}(t_i) & t_i = t_j \\ \underline{0} & \text{otherwise} \end{cases}$$

$\underline{\Phi}(t_i, t_{i-1})$ = state transition matrix from time t_{i-1} to time t_i (nxn matrix)

$\underline{P}(t_i^-)$ = filter computer covariance matrix of state $\underline{x}(t_i)$, also of the error in the estimate of $\underline{x}(t_i)$, prior to incorporating a measurement at time t_i (nxn symmetric matrix)

$\underline{P}(t_i^+)$ = filter computer covariance matrix of state $\underline{x}(t_i)$, also of the error in the estimate of $\underline{x}(t_i)$, after incorporating a measurement at time t_i (nxn symmetric matrix)

$\underline{F}(t)$ = system dynamics matrix (nxn) or the matrix of partial derivatives of $\underline{f}[\underline{x}(t), t]$ with respect to $\underline{x}$ for the extended Kalman filter

$\underline{G}(t)$ = system input matrix (nxs)

$\underline{H}(t_i)$ = system observation matrix at time t_i (mxn) or the matrix of partial derivatives of $\underline{h}[\underline{x}(t), t]$ with respect to $\underline{x}$ for the extended Kalman filter

$\underline{R}(t_i)$ = positive definite measurement noise covariance matrix (mxm)

$\underline{K}(t_i)$ = Kalman gain matrix (nxm) defined at time t_i

Linear Kalman Filter Formulation

The linear Kalman filter formulation presented in this section is for a continuous time system model with discrete time measurement updates. Assume that the system modeling has been completed and that the state vector $\underline{x}(t)$ satisfies the vector stochastic differential equation:

$$\dot{\underline{x}}(t) = \underline{F}(t)\,\underline{x}(t) + \underline{G}(t)\,\underline{w}(t) \tag{2}$$

The state equation is propagated forward in time from the initial condition $\underline{x}(t_o)$. Since the exact initial condition may not be known, it is modeled as being a Gaussian random variable with mean $\hat{\underline{x}}_o$ and covariance $\underline{P}_o$:

$$E[\underline{x}(t_o)] = \hat{\underline{x}}_o \tag{3}$$

$$E[\{\underline{x}(t_o) - \hat{\underline{x}}_o\}\{\underline{x}(t_o) - \hat{\underline{x}}_o\}^T] = \underline{P}_o \tag{4}$$

The initial covariance matrix $\underline{P}_o$ may be positive semidefinite, admitting exact knowledge of the initial conditions of some of the states or linear combinations thereof. It can be shown (Ref 10:157-163) that, under the assumption that $\underline{x}(t_o)$ is either deterministic or a Gaussian random variable, the solution $\underline{x}(t)$ to linear stochastic differential equations such as Equation (2) is a Gauss-Markov process, i.e. the conditional density of $\underline{x}$ at time t_i based upon all realizations of $\underline{x}$ through time t_{i-1} is both Gaussian and completely determined by the process value at t_{i-1}. Because the conditional density is Gaussian, it is completely specified by its mean and covariance (Ref 7:92).

Measurements are available at discrete time points and are assumed to be of the form of a linear combination of the states and corrupted by a white Gaussian sequence (Ref 3:2):

$$\underline{z}(t_i) = \underline{H}(t_i)\ \underline{x}(t_i) + \underline{v}(t_i) \tag{5}$$

The state estimate propagates between measurements (from time t_{i-1}^+ to time t_i^-) according to:

$$\hat{\underline{x}}(t_i) = \underline{\Phi}(t_i, t_{i-1})\ \hat{\underline{x}}(t_{i-1}^+) \tag{6}$$

and the covariance propagates according to:

$$\underline{P}(t_i) = \underline{\Phi}(t_i, t_{i-1})\ \underline{P}(t_{i-1}^+)\ \underline{\Phi}(t_i, t_{i-1})^T + \int_{t_{i-1}}^{t_i} \underline{\Phi}(t_i, \tau)\ \underline{G}(\tau)\ \underline{Q}(\tau)^T\ \underline{\Phi}(t_i, \tau)^T\ d\tau \tag{7}$$

At measurement time t_i, the estimate is updated according to (Ref 10:233):

$$\hat{\underline{x}}(t_i^+) = \hat{\underline{x}}(t_i^-) + \underline{K}(t_i)\ [\underline{Y}_i - \underline{H}(t_i)\ \hat{\underline{x}}(t_i^-)] \tag{8}$$

$$\underline{P}(t_i^+) = \underline{P}(t_i^-) - \underline{K}(t_i)\ \underline{H}(t_i)\ \underline{P}(t_i^-) \tag{9}$$

where

$$\underline{K}(t_i) = \underline{P}(t_i^-)\ \underline{H}(t_i)^T\ [\underline{H}(t_i)\ \underline{P}(t_i^-)\ \underline{H}(t_i)^T + \underline{R}(t_i)]^{-1} \tag{10}$$

where $[\ ]^{-1}$ indicates the inverse of the bracketted matrix and Y_i is the realized value of the measurement $\underline{z}(t_i)$ at time t_i.

Under the assumption that the adequate system model is linear, and that the dynamic driving and measurement noises are Gaussian and white, the Kalman filter provides the optimal estimate $\hat{\underline{x}}(t_i^+)$ of the state of the system (Ref 10:66,214), relative to many optimality criteria with these assumptions $\hat{\underline{x}}(t_i^+)$ is the mean, mode, and median of the conditional density of $\underline{x}(t_i)$, conditioned on the entire measurement history through time t_i. The covariance of the error committed

by using $\hat{\underline{x}}(t_i^+)$ as the estimate of the state at time t_i is denoted by $\underline{P}(t_i^+)$. It should be noted that for a linear estimation problem, the covariance propagation [Equations (7,9)], while depending on the sequence of $\underline{H}(t_i)$ and $\underline{R}(t_i)$, is independent of the time history of measurements $Y_i(Y_1, Y_2, \ldots)$. This will no longer be the case in the extended Kalman filter formulation.

The assumption that the system can be modeled as being driven by white Gaussian noise is often well founded on two accounts. First, it has been found from practical experience that the Gaussian distribution provides a reasonable approximation to observed random behavior in certain physical systems (Ref 7:92). Secondly, the central limit theorem (Ref 7:96) states that if the random phenomenon that we observe at the macroscopic level, is due to the superposition of a large number of independent random variables on the microscopic level, then the macroscopic phenomenon can be adequately modeled as a Gaussian random variable (Ref 10:40).

Extended Kalman Filter Formulation (Ref 3:179-189)

The extended Kalman filter formulation is commonly used in estimation problems in which the adequate state and/or measurement equations are nonlinear rather than linear. Consider, as before, a system that is continuous in time with measurements at discrete sampling times. Assume that the system state satisfies the following nonlinear vector stochastic differential equation:

$$\dot{\underline{x}}(t) = \underline{f}[\underline{x}(t), t] + \underline{G}(t)\,\underline{w}(t) \qquad (11)$$

where $\underline{f}[\underline{x},t]$ is a nonlinear function of the states and time [in general $\underline{f}$ could also be a function of deterministic control inputs $\underline{u}(t)$], and where the vector $\underline{w}(t)$ of zero mean white Gaussian driving noises and covariance $\underline{Q}(t)$ enters in a linear additive manner. The initial condition of $\underline{x}(t_o)$ is modeled as a Gaussian random variable with mean $\hat{\underline{x}}_o$ and covariance $\underline{P}_o$. Noise corrupted vector measurements of a nonlinear function of the states and time are available at discrete times t_i as:

$$\underline{z}(t_i) = \underline{h}[\underline{x}(t_i),\ t_i] + \underline{v}(t_i) \tag{12}$$

where $\underline{v}(t_i)$ is a zero mean white Gaussian sequence with covariance kernel (Ref 3:180).

$$E[\underline{v}(t_i)\ \underline{v}(t_j)] = \begin{cases} \underline{R}(t_i) & \text{for } i = j \\ \underline{0} & \text{otherwise} \end{cases} \tag{13}$$

It is assumed that the processes $\underline{w}$ and $\underline{v}$ are independent of each other.

The filter propagation equations are:

$$\dot{\hat{\underline{x}}}(t/t_i) = \underline{f}[\hat{\underline{x}}(t/t_i),\ t] \tag{14}$$

$$\hat{\underline{x}}(t_i/t_i) = \hat{\underline{x}}(t_i^+) \tag{15}$$

where the notation $\hat{\underline{x}}(t/t_i)$ means the optimal estimate of the state, $\underline{x}$, at time, t, given the updated estimates up to and including time t_i. In addition, (the covariance is propagated approximately by):

$$\dot{\underline{P}}(t/t_i) = \underline{F}[t;\hat{\underline{x}}(t/t_i)]\ \underline{P}(t/t_i) + \underline{P}(t/t_i)\ \underline{F}[t;\hat{\underline{x}}(t/t_i)]^T + \underline{G}(t)\ \underline{Q}(t)\ \underline{G}(t)^T \tag{16}$$

$$\underline{P}(t_i/t_i) = \underline{P}(t_i^+) \tag{17}$$

where $\underline{F}$ is the matrix of partial derivatives of $\underline{f}$ with respect to $\underline{x}$, evaluated along the trajectory [which is propagated by means of Equation (14)]:

$$\underline{F}[t;\hat{\underline{x}}(t/t_i)] = \left.\frac{\partial \underline{f}[\underline{x},t]}{\partial \underline{x}}\right|_{\underline{x} = \hat{\underline{x}}(t/t_i)} \tag{18}$$

The measurement update is given by:

$$\underline{K}(t_i) = \underline{P}(t_i^-)\,\underline{H}[t_i;\hat{\underline{x}}(t_i^-)]^T \{\underline{H}[t_i;\hat{\underline{x}}(t_i^-)]\,\underline{P}(t_i^-)\,\underline{H}[t_i;\hat{\underline{x}}(t_i^-)] + \underline{R}(t_i)\}^{-1} \tag{19}$$

$$\hat{\underline{x}}(t_i) = \hat{\underline{x}}(t_i) + \underline{K}(t_i)\{\underline{Y}_i - \underline{h}[\hat{\underline{x}}(t_i^-),t_i]\} \tag{20}$$

$$\underline{P}(t_i^+) = \underline{P}(t_i^-) - \underline{K}(t_i)\,\underline{H}[t;\hat{\underline{x}}(t_i^-)]\underline{P}(t_i^-) \tag{21}$$

where

$$\hat{\underline{x}}(t_i^-) = \hat{\underline{x}}(t_i/t_{i-1}) \tag{22}$$

$$\underline{P}(t_i^-) = \underline{P}(t_i/t_{i-1}) \tag{23}$$

$$\underline{H}[t;\hat{\underline{x}}(t_i^-)] = \left.\frac{\partial \underline{h}[\underline{x},t]}{\partial \underline{x}}\right|_{\underline{x} = \hat{\underline{x}}(t_i^-)} \tag{24}$$

The notation t_i^- implies the value of the quantity at the instant prior to the update at time t_i, and t_i^+ is the value of the quantity just after the update. The notation used thus far, except as noted for the $\underline{F}$ matrix, is developed and used in (Ref 3).

Unlike the conventional Kalman filter, the extended Kalman filter gain and estimation error covariance matrices depend on the time history of $\hat{\underline{x}}(t/t_i)$. Since a covariance analysis could only use an $\underline{x}_{nominal}(t)$ as an approximation to $\hat{\underline{x}}(t/t_i)$ as actually used by the filter in its online operation (the linearized Kalman filter), the extended Kalman filter performance should be verified by a Monte Carlo simulation. In general, this form of analysis is both more time consuming and more costly in terms of computer usage than is a covariance analysis.

Section III

Monte Carlo Simulation and Analysis of a Sub-Optimal Kalman Filter

Introduction

As mentioned in Section I, a Monte Carlo analysis is a computer simulation technique that can be used to develop the required (extended) Kalman filter tuning, portray filter performance capabilities, and study the effects of parameter variations in the true system model. The Monte Carlo simulation uses random number generators and shaping filters which generate random errors to noise corrupt both system state dynamics and measurements. The noisy system measurements are then processed by the suboptimal (extended) Kalman filter to generate the filter estimates. If many of these simulations are conducted, then one can compute the statistics of the error between the filter estimate and the truth model for quantities of interest.

Performance Analysis

Figure 1 depicts schematically a means of conducting the performance analysis of a given (extended) Kalman filter design. The truth model by definition is the best, most complete mathematical model that can be developed to describe the system under study. Such a truth model is the product of extensive study and data analysis of the system. As noted in the figure, the truth model is an n_t-state model, linear or nonlinear, driven by noise $\underline{w}_t(t)$ (assumed white and Gaussian), that generates the true state values $\underline{x}_t(t)$. It is assumed that the true values for the critical quantities of interest are related to the true state values by the vector function $\underline{c}_t$ (a p vector - generally less

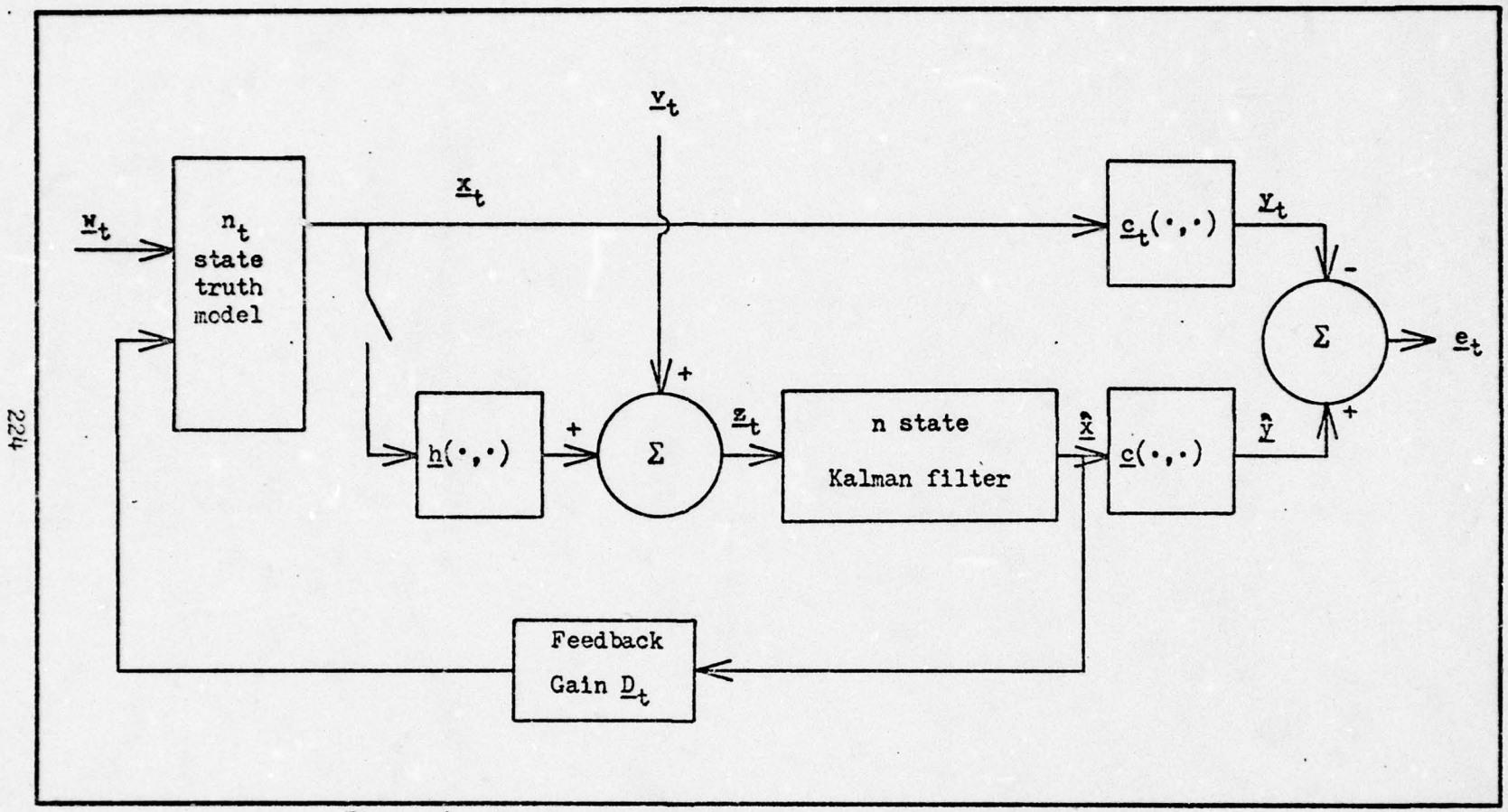


Figure 1. Performance Evaluation of a Kalman Filter Design

than n_t):

$$\underline{y}_t(t) = \underline{c}_t[\underline{x}_t(t), t] \tag{25}$$

Often, $\underline{c}_t[x_t(t), t]$ is a linear function of $\underline{x}_t$, as

$$\underline{y}_t(t) = \underline{c}(t)\underline{x}(t) \tag{26}$$

In fact, in many instances, the quantities of interest are simply a subset of the truth model states; letting the components of $\underline{x}(t)$ be ordered so that the first p states are the quantities of interest, this yields the particular form:

$$\underline{y}_t(t) = [\underline{I} \,\vdots\, \underline{0}]\, \underline{x}_t(t) \tag{27}$$

Thus we have access to the "real world" values for comparison with the filter estimates.

At discrete times (t_i) the m-dimensional measurements:

$$\underline{z}_t(t_i) = \underline{h}[\underline{x}_t(t_i), t_i] + \underline{v}_t(t_i) \tag{28}$$

are presented to the Kalman filter. The Kalman filter algorithm processes the measurements and produces the state estimates $\hat{\underline{x}}(t)$. These state estimates are related to the estimates of critical quantities of interest by the vector function $\underline{c}$ (a p vector):

$$\hat{\underline{y}}(t) = \underline{c}[\hat{\underline{x}}(t), t] \tag{29}$$

Thus the true error committed by the Kalman filter ($\underline{e}_t$) at time t_i, before and after measurement incorporation, is:

$$\underline{e}_t(t_i^-) = \hat{\underline{y}}(t_i^-) - \underline{y}_t(t_i) \tag{30}$$

$$\underline{e}_t(t_i^+) = \hat{\underline{y}}(t_i^+) - \underline{y}_t(t_i) \tag{31}$$

If a feedback control system with impulsive correction is assumed, then a third error is of interest as well:

$$\underline{e}_t(t_i^{+c}) = \hat{\underline{y}}(t_i^{+c}) - \underline{y}_t(t_i^{\,c}) \tag{32}$$

where the superscript c denotes after the control is applied.

The objective of the performance analysis is to characterize the error process statistically. This is accomplished in the Monte Carlo simulation by generating many samples of the error process and then computing the sample statistics directly. If enough samples are generated, then the sample statistics should approximate the process statistics very well (Ref 3:6-71). The sample statistics computed at each point in time are the calculated mean error ($\bar{e}$), calculated standard deviation of the error (s_e), and the ensemble average of the filter covariance diagonal terms ($\overline{PF}_{kk}$). The calculations are made over the ensemble of runs (N) for each time point (t_j):

$$\bar{e}(t_j) = \frac{1}{N} \sum_{i=1}^{N} [\hat{x}_i(t_j) - x_{ti}(t_j)] \tag{33}$$

$$s_e(t) = \left[\{ \sum_{i=1}^{N} [\hat{x}_i(t_j) - x_{ti}(t_j)]^2 - N \bar{e}^2(t_j) \} / N-1 \right]^{1/2} \tag{34}$$

$$\overline{PF}_{kk}(t_j) = \frac{1}{N} \sum_{i=1}^{N} PF_{kki}(t_j) \tag{35}$$

These equations can be found in any good statistics text (see for example Ref 11:26,40). It is important to note the use of 1/(N-1) in the expression for the standard deviation, s_e, instead of 1/N; this results in an unbiased estimator of variance before the $[\ ]^{1/2}$ is taken and is customarily used for sampled data (for small or moderate N).

One useful output of the Monte Carlo analysis is a plot of the filter estimate of the standard deviation, $\sqrt{\overline{PF}_{kk}(t_j)}$, along with the corresponding computed standard deviation, $s_e(t_j)$ for all t_j or interest. The Kalman filter is tuned by minimizing the computed error standard deviations. As a rule of thumb, good tuning is achieved

when the filter estimated standard deviations are approximately equal to the computed standard deviations. If the problem under investigation is sensitive to small changes in the filter tuning parameters [$\underline{P}(t_o)$, and time histories of $\underline{Q}(t)$ and $\underline{R}(t)$] then the investigator must:

1. Conduct the Monte Carlo simulation
2. Examine the standard deviation plots
3. Determine tuning parameter(s) to be changed
4. Repeat 1 through 3 until adequate tuning is achieved

Thus it can be readily visualized that precise tuning of the Kalman filter by Monte Carlo techniques is very costly in terms of effort and time.

A second useful output of the Monte Carlo analysis is a plot of the mean error versus time. This plot is usually obtained with one standard deviation bounds plotted above and below the mean error. This plot is useful to determine if the suboptimal Kalman filter provides the desired accuracy. If the filter starts from initial conditions other than the truth model initial conditions, then the filter performance to non-zero initial error can be observed. Bias errors, a problem of significance to many extended Kalman filter designs, also become readily apparent from the plots of this form.

Digital Simulation of the Truth Model and Filter Model

In order to simulate the system on a digital computer, the problem must be divided into segments which, when processed sequentially over a given time period, will provide a realistic portrayal of filter use

in actual on-line implementation. A convenient time period is the sample period between times at which measurements become available. Then the entire procedure is iterated over some desired time interval of interest. The segments are:

1. Time propagation of system state equations
2. Time propagation of filter state equations
3. Time propagation of filter covariance equations
4. Update of filter covariance matrix
5. Generation of measurements for the filter
6. Update of filter states
7. Application of impulsive feedback to "update" system states
8. Reset the filter states after the feedback

The first segment is simulated by integrating the n_t-dimensional system (truth model) stochastic differential equation:

$$\dot{\underline{x}}_t(t) = \underline{f}_t[\underline{x}_t(t),t] + \underline{G}_t(t)\underline{w}_t(t) \tag{36}$$

from time t_i to time of the next measurement time t_{i+1}. The integration is typically performed by either Euler or Runga-Kutta numerical integration methods with a specified integration step size. Integrating the deterministic part of the differential equation is readily apparent, however, the additive white Gaussian noise $\underline{w}_t$ is not. The following derivation shows how the solution to the stochastic differential equation can be approximated.

To provide insight into the approximate solution form, consider first a linear stochastic differential equation of the form:

$$\dot{\underline{x}}(t) = \underline{F}(t)\ \underline{x}(t) + \underline{G}(t)\ \underline{w}(t) \tag{37}$$

where $\underline{w}(t)$ is a zero mean white Gaussian noise with $E[\underline{w}(t)\underline{w}(t+t)^T] = \underline{Q}(t)\delta(t)$. An equivalent discrete time system equation would be:

$$\underline{x}(t_{i+1}) = \underline{\Phi}(t_{i+1},t_i)\,\underline{x}(t_i) + \int_{t_i}^{t_{i+1}} \underline{\Phi}(t_{i+1},t_i)\,\underline{G}(\tau)\underline{w}(\tau)\,d\tau \tag{38}$$

$$= \underline{\Phi}(t_{i+1},t_i)\,\underline{x}(t_i) + \underline{w}_d(t_i) \tag{39}$$

where $\underline{\Phi}(\cdot,\cdot)$ satisfies both of the following relationships:

$$\frac{d}{dt}\underline{\Phi}(t,t_i) = \underline{F}(t)\,\underline{\Phi}(t,t_i) \text{ for } t \text{ in the interval } (t_i,t_{i+1}) \tag{40}$$

$$\underline{\Phi}(t_i,t_i) = \underline{I} \tag{41}$$

For this discrete time system, $\underline{w}_d(t_i)$ is a zero mean white Gaussian discrete-time noise with strength:

$$E[\underline{w}_d(t_i)\underline{w}_d(t_i)^T] = \int_{t_i}^{t_{i+1}} \underline{\Phi}(t_{i+1},\tau)\,\underline{G}(\tau)\,\underline{Q}(\tau)\,\underline{G}(\tau)^T\,\underline{\Phi}(t_{i+1},\tau)\,d\tau \tag{42}$$

$$= \underline{Q}_d(t_i) \tag{43}$$

$$E[\underline{w}_d(t_i)\underline{w}_d(t_j)^T] = \underline{0} \quad \text{for } t_i \neq t_j \tag{44}$$

Now if the integration step size is small relative to system characteristic response time and constant (Δt sec in duration) then the following approximations can be made:

$$\underline{\Phi}(t_{i\ 1},t_i) \cong \underline{I} + \underline{F}(t_i)\,\Delta t \tag{45}$$

$$\int_{t_i}^{t_{i+1}} \underline{\Phi}\,\underline{G}\,\underline{Q}\,\underline{G}^T\,\underline{\Phi}^T\,d\tau \cong [\underline{G}(t_i)\,\underline{Q}(t_i)\,\underline{G}(t_i)^T]\,\Delta t \tag{46}$$

where all terms of order Δt^2 or higher have been neglected. Thus one approximate simulation of the continuous time system would be:

$$\underline{x}(t_{i+1}) = [\underline{I} + \underline{F}(t_i)\,\Delta t]\,\underline{x}(t_i) + \underline{w}_d(t_i) \tag{47}$$

$$= \underline{x}(t_i) + \underline{F}(t_i)\,\underline{x}(t_i)\,\Delta t + \underline{w}_d(t_i) \tag{48}$$

where $\underline{w}_d(t_i)$ is described by:

$$E[\underline{w}_d(t_i)\underline{w}_d(t_i)^T] = [\underline{G}(t_i)\ \underline{Q}(t_i)\ \underline{G}(t_i)^T]\ \Delta t \tag{49}$$

Thus the differential equation is integrated using Equation (46) and noise generators utilized for the $\underline{w}_d(t_i)$ term. Higher order integration techniques can be applied to the nonhomogeneous portion of the differential equation, with the same $\underline{w}_d(t_i)$ as above used as the forcing terms. It should be noted that this technique is directly extendable to nonlinear equations of the form:

$$\dot{\underline{x}}(t) = \underline{f}[\underline{x}(t),t] + \underline{G}(t)\ \underline{w}(t) \tag{50}$$

Segment two is performed by calculating the filter equation (6). The filter covariance propagation Equation (7) is calculated to accomplish segment three. Segment four is performed by solving Equations (10 and 9). Segment five uses Equation (5) to generate the m-dimensional measurements. Noise generators are utilized to provide the $\underline{v}_t(t_i)$ term. The filter update Equation (8) is calculated to perform segment six.

A Kalman filter will often be implemented in an indirect feedback configuration, in which filter error state estimates are fed back to the actual system to correct the actual error. If the system correction can be performed rapidly enough compared to the filter update period, then the feedback can be simulated as an impulsive change. After application of the correction, assuming linear feedback, the truth model state process becomes (Ref 3:6-76):

$$\underline{x}_t(t_i^{\,c}) = \underline{x}_t(t_i) - \underline{D}_t(t_i)\ \hat{\underline{x}}(t_i) \tag{51}$$

The filter should be "told" that this feedback to the system has occurred, so its state estimate is modified as:

$$\hat{\underline{x}}(t_i^{+c}) = \hat{\underline{x}}(t_i^{+}) - \underline{D}(t_i)\,\hat{\underline{x}}(t_i^{+}) \tag{52}$$

$$= [\underline{I} - \underline{D}(t_i)]\,\hat{\underline{x}}(t_i^{+}) \tag{53}$$

Note that if the filter is estimating error states then the $\underline{D}(t_i)$ matrix will typically be an identity matrix. Segments seven and eight are performed by calculating the results of Equations (51) and (52).

Each of these segments are performed sequentially at the measurement update time and the simulation time, system state, filter state, and filter covariance values saved on a data tape. This process continues until the simulation final time is reached, thus completing one run of a Monte Carlo simulation. After many runs of the Monte Carlo simulation have been performed, then the data tape can be read, the sample statistics calculated using Equations (33-35) and computer plots created.

SECTION IV

DESCRIPTION OF THE GENERALIZED MONTE CARLO ANALYSIS PROGRAM (MCAP)

Introduction

The general Monte Carlo Analysis Program (MCAP) is a modular program designed to perform a Monte Carlo performance analysis of a given (extended) Kalman filter design. The program is composed of the main program, 45 problem independent (constant) subroutines, and a minimum of 8 user defined problem dependent (variable) subroutines. The main program acts as an executive to sequence the simulation through the performance analysis. The MCAP subroutines can be divided into three major categories: 1) data input subroutines, 2) data output subroutines, and 3) "workhorse" subroutines. The data input subroutines read in the problem dependent parameters and control parameters for MCAP execution. The output from MCAP consists of data printed on the computer line printer and a data file (Tape 3) containing the sample realizations for the many simulation runs. The amount of printed output is determined by the control parameters specified in the data input. The output data file (Tape 3) is used by the statistics/plot (STATPLOT) program to calculate the statistics of the true error noted in Section III (Equations 33 through 35) and to prepare a plot file for the Calcomp (or other) plotting device. The "workhorse" subroutines are executed in the sequence managed by the main program to perform many functions:

1. Initialize the Monte Carlo performance analysis
2. Initialize the simulation run
3. Propagate the system state equations

4. Propagate the filter state equations
5. Propagate the filter covariance equations
6. Update the filter covariance matrix
7. Generate system measurements for the filter
8. Update the filter states
9. Apply feedback (if applicable to the problem) to reset the system and filter states

MCAP was developed using the General Covariance Analysis Program (GCAP) by the Air Force Avionics Laboratory (Ref 2) as a baseline. Consequently, the goals of having MCAP similar in structure to GCAP and having (some) problem dependent subroutines applicable to both GCAP and MCAP are met. Because MCAP is composed, to a large extent, of modified GCAP subroutines; users already familiar with GCAP should not have any difficulty in applying MCAP to their problem.

As noted in Section I, in general, the covariance analysis is limited because stringent assumptions are necessary for the analysis results to be a valid depiction of the error characteristics. In particular GCAP assumes that:

1. The filter and system are modeled by a set of linear differential equations.
2. The measurements used by the filter are a linear combination of the system states with additive noise-corrupted measurements.
3. The states being modeled in the filter are error states.

4. Total impulsive feedback control is applied to the system (error) states.

These stringent assumptions limit the application of GCAP to carefully posed problems. None of these assumptions are necessary for problems to be investigated using MCAP. Thus, at the expense of additional computer time, MCAP can be used to solve a larger class of estimation problems.

MCAP Program Structure

The performance analysis of the Kalman filter is conducted under the management of the main executive program. Figure 2 illustrates the structure of MCAP. There are generally three levels of subroutines within this structure. The first level subroutines are coded on the figure to note the main functions performed and if the subroutine is problem dependent (a variable subroutine to be specified by the user). The function code is located below the subroutine name. If the subroutine is problem dependent then the subroutine name is underlined (for example XSDOT). The coding used on this figure is:

I - Data input function

O - Data output function

W1 - Initialize the Monte Carlo performance analysis

W2 - Initialize the simulation run

W3 - Propagate the system state equations

W4 - Propagate the filter state equations

W5 - Propagate the filter covariance equations

W6 - Update the filter covariance matrix

W7 - Generate system measurements

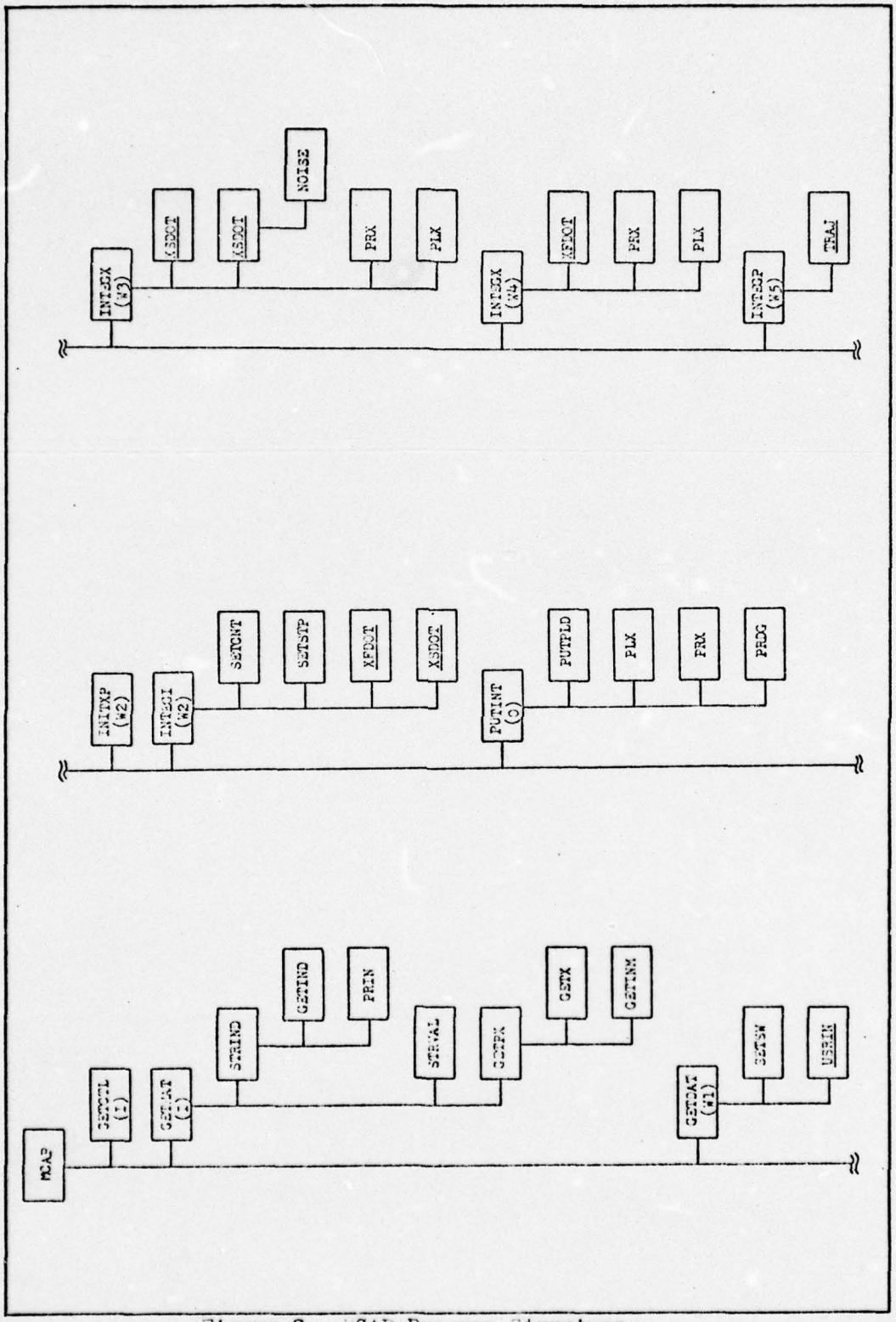


Figure 2. MCAP Program Structure

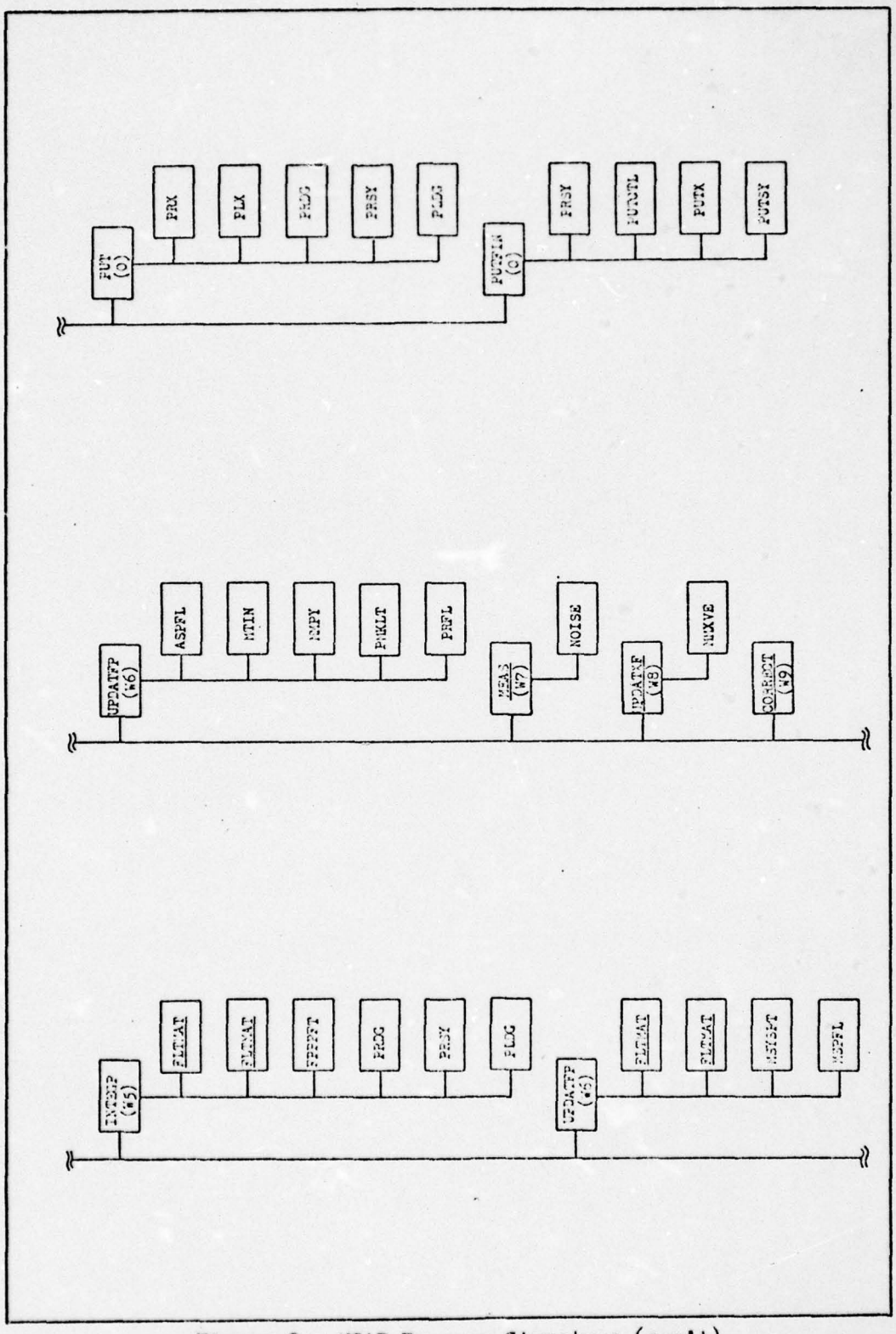


Figure 2. MCAP Program Structure (con't)

W8 - Update the filter states

W9 - Apply feedback to update the filter and system states

Several of the subroutine names are repeated to illustrate the program structure and sequence of subroutine execution better.

The first task MCAP performs is to read in the problem control parameters, problem dependent parameters, and assign memory storage locations. Function W1 is then performed in order to set the print and plot control "switches" and initialize the random number generator. The simulation run is initialized (W2) by calculating the integration constants and initializing the program time parameters.

One run of the Monte Carlo simulation is performed by sequencing through a cycle of functions W3 through W9 at a rate equal to the measurement interval until the final problem time is reached. Throughout this run, the data output function (O) is performed to print the data selected by the user with the problem control parameters. Subsequent runs are performed by repeating functions W2 through W9 until the user specified number of runs is completed.

The rest of this section expands on this brief description and provides a description of the variable subroutines required for MCAP. A good description of the constant subroutines is contained in the GCAP users guide (Ref 2) and is not repeated here.

Data Input Function (I). The variables listed and described in Table I are the input parameters required for MCAP. The normal input media for the problem variables is the card deck structure shown in Figure 3. However, the user may elect to read the values of TØ,

Table I

Definitions and Format of MCAP Input Parameters

Parameter	Description	Format
(title)	Problem title (limited to 80 characters)	8A10
$USRCTL	Problem dependent control parameters are contained in this namelist	Namelist
N1,	Suboptimal filter dimension	Namelist
N2,	Truth model (system dimension)	Namelist
M,	Number of measurements	Namelist
TØ,	Initial problem time (seconds)	Namelist
TF,	Final problem time (seconds)	Namelist
DTUP,	Measurement update interval (seconds)	Namelist
DTINT,	Integration interval (seconds)	Namelist
IPASS,	Number of runs in Monte Carlo simulation	Namelist
ISEED,	Initial seed for random number generation	Namelist
IRNPRCT,	Number of runs per printed output	Namelist
IPRCT,	Number of integration intervals per printed output	Namelist
IPRSYM,	if 0 - do not print lower triangular portion of symmetric matrix PF if 1 - print lower diagonal of symmetric matrix PF	Namelist
IPRK,	if 0 - do not print Kalman gain matrix K if 1 - print Kalman gain matrix K	Namelist
IPLCT,	Number of integrations intervals per plotted output (must be 1 to get plot)	Namelist

Table I (cont'd)

IPFILE,	if 0 - usual data input from cards if 1 - input restart values TØ, TIMUDØ, XF, XS, and PF from Tape 11	Namelist
TIMUDØ,	Starting value of the update time in seconds -- unused if IPFILE = 1	Namelist
TIMNTØ,	Starting value of the integration time in seconds -- unused if IPFILE = 1	Namelist
$	End of Namelist	Namelist
NZFF	Number of non-zero terms in the filter F matrix	I3
INDFF	The coordinate pairs of the non-zero terms in the filter F matrix	8(2I3,4X)
NZQF	Number of non-zero terms in the filter Q matrix	I3
INDQF	The coordinate pairs of the non-zero terms in the filter Q matrix	8(2I3,4X)
NZHF	Number of non-zero terms in the filter H matrix	I3
INDHF	The coordinate pairs of the non-zero terms in the filter H matrix	8(2I3,4X)
NZRF	Number of non-zero terms in the filter R matrix	I3
INDRF	The coordinate pairs of the non-zero terms in the filter R matrix	8(2I3,4X)
XFØ	if IPFILE = 0, each filter state initial condition value (N1 in number)	E20.0
XSØ	if IPFILE = 0, each system state initial condition value (N2 in number)	E20.0
PFØ	if IPFILE = 0, each filter covariance diagonal element initial condition value (N1 in number)	E10.0

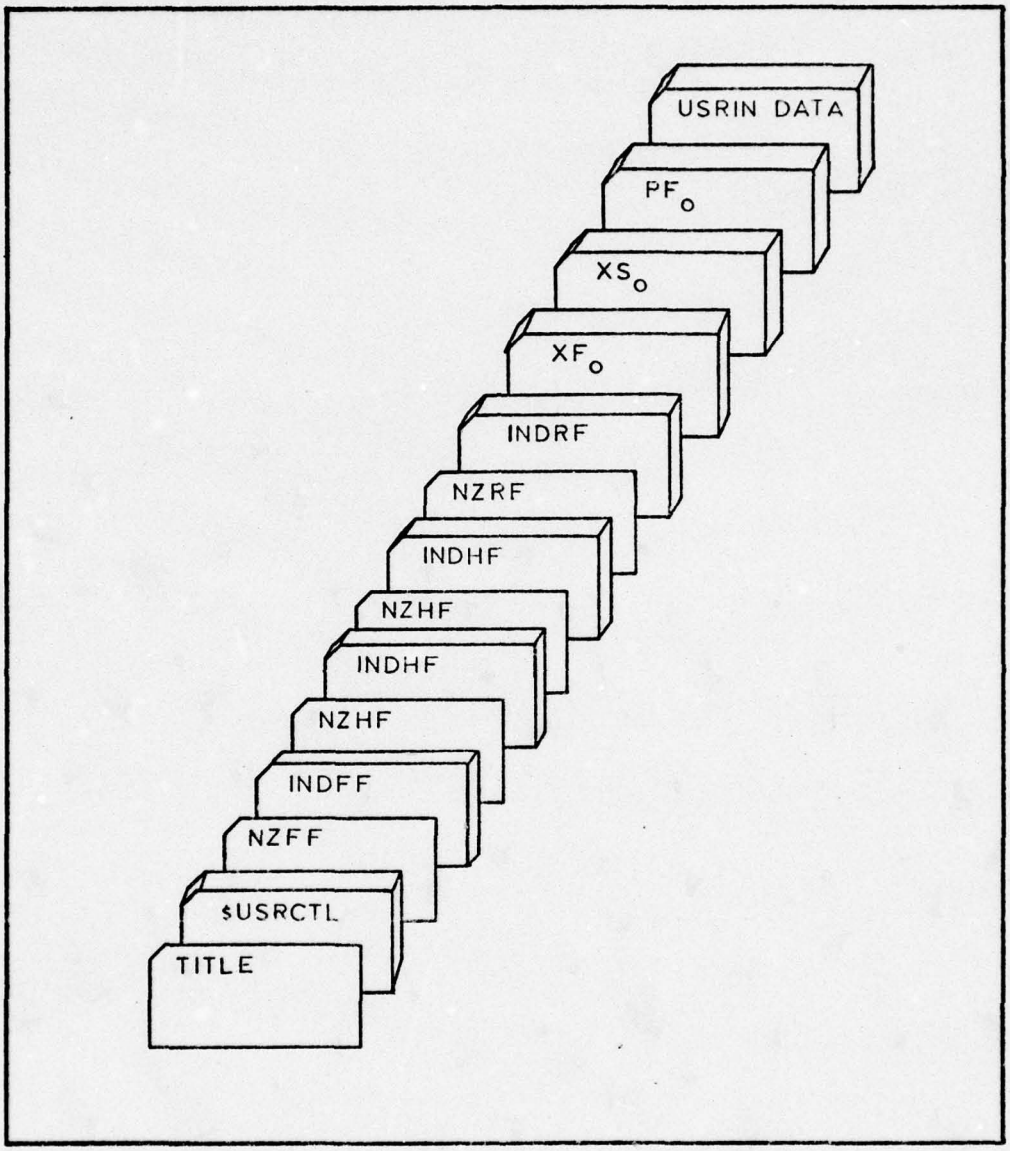


Figure 3. Input Card Deck Structure

TIMUDØ, XFØ, XSØ, and PFØ from the file named "Tape 11" as shown in Figure 4. This option facilitates "restarting" the simulation at the simulation time where a previous execution was terminated.

Concurrent with the data input, the variables N2 and M undergo a "reasonableness" test in order to insure program memory is not exceeded. If the values of these variables pass the test criteria (M less than 21 and N2 less than 401), then MCAP begins to assign storage locations within the unlabeled COMMON block A. These storage locations are used to store the indices of the filter state matrices ($\underline{F}$, $\underline{Q}$, $\underline{H}$, and $\underline{R}$). When this process is completed for the filter state matrices, MCAP generates the memory location pointers for the first word locations of the initial values of the filter states, the system states, and the covariance matrix; the time varying values of the filter states, the system states, and the covariance matrix; the time varying values of the filter state derivatives, the system state derivatives, and the filter covariance derivatives; the filter state matrices ($\underline{F}$, $\underline{Q}$, $\underline{H}$, and $\underline{R}$), the Kalman gain matrix; and memory locations for temporary intermediate calculations. This elaborate memory assignment scheme enables the reuse of storage locations while the simulation is executing. However, the user should be cautioned, that this "feature" may cause problems if values are needed in the user-supplied variable subroutines. The user then has the option of over-riding the MCAP storage assignment [by modifying the memory assignment subroutine (STRVAL)] or storing the values of the variables in the variable subroutines as labeled COMMON.

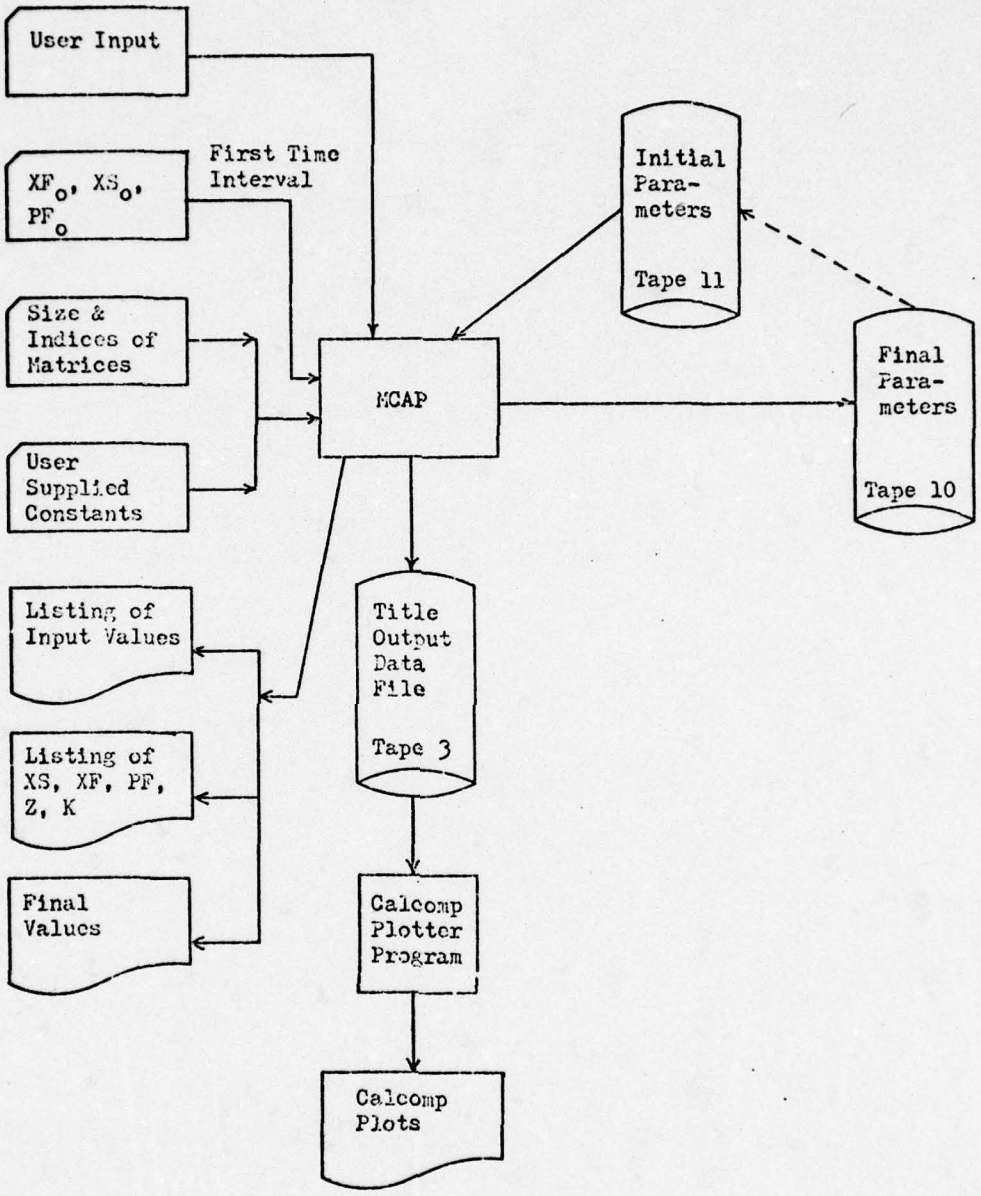


Figure 4. MCAP Program Relationships

Initialize the Performance Analysis (W1). Once the storage allocations are made, MCAP sets the print and plot control "switches" according to the problem control parameters (IRNPRCT, IPRCT, IPRSYM, IPRK, and IPLCT). The random number generator "seed" is initialized to the value specified by the user (ISEED). This seeding of the random number generator is for Monte Carlo simulation repeatability from one set of runs to another. Next, the user supplied subroutine USRIN is called. The subroutine USRIN may be employed in a variety of ways. USRIN may be used to input parameters which change between Monte Carlo simulations but which remain unchanged within a single simulation. Or the user may employ USRIN to create special initial conditions, such as a nondiagonal PF$\emptyset$. Whatever the usage, USRIN is a convenient method which permits a user to customize MCAP input data to the specific problem.

Initialize the Simulation Run (W2). MCAP then initializes the time dependent values for the filter states, the system states, and the covariance matrix to the values specified in the data input. The two user supplied subroutines XFDOT and XSDOT are called to allow these initial values to be modified with each simulation run. The integration constants and the program time parameters are then initialized. MCAP is now ready for a simulation run.

Propagate the Derivative Equations (W3 through W5). One run of the Monte Carlo simulation involves sequencing through a cycle of functions W3 through W9 at a rate equal to the measurement update

interval; until the final problem time is reached. Functions W3 through W5 require the extrapolation of the system state values, the filter state values, and the covariance matrix from the current simulation time, TIME, (which has an initial value of TØ) until the first update time, TIMUD. TIMUD is the sum of TIME and the measurement update interval, DTUP. Subroutine INTEGX propagates, in turn, the system state values and the filter state values between TIME and TIMUD. INTEGX accomplishes this propagation by successively integrating the system and filter propagation equations [Equations (6) or (14)], where each integration is DTINT long. The integration is performed using a fourth order Runge-Kutta numerical integration method. In the case of the system state equations, the zero mean white Gaussian noise sequence, $\underline{w}_d(t_i)$ [Equation (48)], is added to the system state values at this DTINT rate. Subroutine INTEGP propagates, in a similar manner, the covariance matrix from TIME to TIMUD.

In order to perform the fourth order Runge-Kutta calculations, both INTEGX and INTEGP require values for the derivative equations at the following time points: TIME, TIME+DTINT/2, TIME+DTINT/2, and TIME+DTINT. (Note that the fourth order Runge-Kutta method requires a derivative value twice at the midpoint of the integration interval). For the system and filter derivative states this is accomplished by calling either subroutine XSDOT or XFDOT. The calculations for $\dot{\underline{P}}$, on the other hand, requires values for the $\underline{F}$ and $\underline{Q}$ matrices, which in turn, derive their values from (possibly) time varying parameters. These time varying parameters are obtained when INTEGP calls the user subroutine TRAJ at each time point noted above. The time varying

parameters are transmitted between TRAJ and user subroutine FLTMAT via a user defined labeled COMMON. After the call to TRAJ, INTEGP calls FLTMAT two separate times in order to obtain the current (non-zero) values of the F and Q matrices respectively. INTEGP then calculates P and extrapolates the covariance matrix to the next time point, using Equation (7) or (16).

Update the Filter Covariance Matrix (W6). When the TIMUD time is reached, function W6 can be performed. In order to update the filter covariance matrix, the Kalman gain matrix, K, must be computed (see Section II). However, this requires the current values of the filter H and R matrices. These values are obtained when subroutine UPDATFP calls FLTMAT two separate times. Subroutine UPDATFP then calls other subroutines in order to calculate K [Equation (10) or (19)] and update the filter covariance matrix, P [Equation (9) or (21)].

Generate the System Measurements (W7). The system measurements are generated by the user supplied subroutine MEAS. MEAS calculates the measurements using the current system state values corrupted by additive noise [Equation (5) or (12)].

Update the Filter and System States (W8 and W9). The filter estimates (XF+) after the measurements obtained are generated by the user supplied subroutine UPDATXF. UPDATXF generates XF+ by performing the calculations indicated in Equation (20) [see Section II]. This routine is user supplied in order to accommodate the non-linear vector function h. After the XF+ estimates have been obtained, the user supplied subroutine CORRECT is called to apply impulsive feedback to

(selected) system states. (Note that if no feedback correction is required for the problem under study, subroutine CORRECT will simply have a RETURN and END statement). The feedback is applied by modifying the system state values using Equation (51) and the filter state values with Equation (52).

Additional Simulation Runs. Throughout the simulation run, data is printed (values of XF, XS, PF, K, and Z) according to the user specified values of the printer control parameters (IRNPRCT, IPRCT, IPRSYM, and IPRK). In addition, at the end of a simulation run (if IRNPRCT is greater than the current run number) MCAP prints the lower triangular values of the final PF matrix and "RUN NUMBER xx COMPLETE". If additional simulation runs are required, then functions W2 through W9 are repeated until the specified number of runs have been completed.

Variable Subroutines

As mentioned earlier, MCAP requires a minimum of 8 problem dependent (variable) subroutines to perform the Monte Carlo simulation. These variable subroutines are:

1. XFDOT
2. XSDOT
3. TRAJ
4. FLTHAT
5. MEAS
6. UPDATXF
7. CORRECT
8. USRIN

These subroutines are designed to meet most problem requirements and to be used easily by users having only a basic knowledge of Fortran programming.

The essential elements of each variable subroutine are described below. In order to illustrate the usage of these subroutines, the next section presents an example using a linear Kalman filter which should clarify this discussion.

XFDOT. Subroutine XFDOT major function is to provide MCAP with the derivative values of the filter states at each time point of the Runge-Kutta integration. The SUBROUTINE statement contains the following arguments (in the order shown): X, XDOT, N1, INIT, NOIS, DT, and TIME. X contains the current values (an N1-dimensioned vector) of the filter states. The calculated values of the filter state derivatives are returned to the calling program (INTEGX) in XDOT (an N1-dimensioned vector). N1 is the dimension of the filter. INIT and NOIS are declared logical variables to select one of the three sections of this subroutine code. INIT is true only if the subroutine has been called during function W2. NOIS is false for each time point of the Runge-Kutta integration. At the end of the integration interval. The current value of the simulation time is contained in TIME.

The first section (when INIT is true) of XFDOT is called once per simulation run (by INTEGI) and is used to initialize constants. unique to this subroutine. The second section (where INIT and NOIS are false) is used to calculate the values for the derivatives of the filter states using either $\dot{\underline{x}}_f = \underline{f}(\underline{x}_f, t)$ for the nonlinear filter or

$\dot{\underline{x}}_f = \underline{F}\ \underline{x}_f$ for the linear filter. The third section is normally not used, however, the option remains for the user to modify filter state values at the end of the integration interval.

Table II lists the required Fortran statements for subroutine XFDOT:

Table II

Required Statements for XFDOT

```
      SUBROUTINE XFDOT (X,XDOT,N,INIT,NOIS,DTINT,TIME)
      DIMENSION X(1),XDOT(1)
      LOGICAL INIT,NOIS
      IF (.NOT. INIT) GO TO 200
        [subroutine constants]
  200 CONTINUE
      IF (NOIS) GO TO 300
        [derivative calculations]
      RETURN
  300 CONTINUE
        [user option]
      RETURN
      END
```

XSDOT. Subroutine XSDOT provides MCAP with values for the derivative of the system states (of N2 dimension) and, at the end of the integration interval, adds the zero mean white Gaussian noise, $\underline{w}_d$, to the system state values. Subroutine XSDOT is identical in structure to XFDOT described above. The only change to Table II is to modify the subroutine name and replace the user option section with the calculations necessary to evaluate $\underline{w}_d$ and add it to the system states. The calculations performed in the second section again uses either the nonlinear equation $\dot{\underline{x}}_s = \underline{f}(\underline{x}_s, t)$ or the linear equation $\dot{\underline{x}}_s = \underline{F}_s\ \underline{x}_s$. The noise sequence $\underline{w}_d(t_i)$ covariance is found by Equation (49).

The vector $\underline{w}_d$ is calculated using the NCAP constant subroutine NOISE. Subroutine NOISE returns a Gaussian random number with mean and standard deviation specified on each subroutine call. The arguments for NOISE are: RMS, XMEAN, and XNOISE. Where RMS is the standard deviation of the desired random number, XMEAN is the desired mean of the random number, and XNOISE is the returned random number. If the system dynamics noise, $\underline{w}$, elements are independent of each other, then matrix $\underline{Q}$ of Equation (49) is diagonal. Also, often $\underline{G}$ is the identity matrix so the term in the brackets $[\underline{G}\underline{Q}\underline{G}^T]$ can be written $\underline{Q}'$ or simply reidentified $\underline{Q}$. The standard deviation for the random number is simply the square root of the $\underline{Q}_{kk}$ element times the square root of Δt (DTINT). Table III shows typical Fortran statements which calculate $\underline{w}_d$ and add it to the system states (for zero mean independent white Gaussian noise terms). If nondiagonal $\underline{G}\underline{Q}\underline{G}^T$ is desired, this can be handled by means of transformations of independent noises using square root matrices [see Ref (3:7-60)].

Table III

Calculation of $\underline{w}_d$ for the System States

for noise with $Q_{jj} = \sigma^2$

```
XMEAN = 0.
RMS = SIGMA * SQRT(DTINT)
CALL NOISE (RMS, XMEAN, XNOISE)
X(J) = X(J) + XNOISE
```

for time correlated noise with $Q_{kk} = \sigma^2/2\tau$

where x(k) is a shaping filter state:

```
XMEAN = 0.
RMS = SIGMA * SQRT(DTINT/(2.*TAU))
CALL NOISE (RMS, XMEAN, XNOISE)
X(K) = X(K) + XNOISE
```

TRAJ. Subroutine TRAJ provides trajectory constants or time varying parameters required by FLTMAT. The requirements of TRAJ for MCAP are the same as those for GCAP. The only argument for TRAJ is SETCON. SETCON is a declared logical variable to select either the time invariant (SETCON true) or the time varying (SETCON false) section of the subroutine. SETCON is true at the beginning of each integration interval and TRAJ calculates the time invariant parameters. For the rest of the integration interval TRAJ calculates the time varying parameters. Instead of computing the trajectory dependent parameters as the simulation is progressing, the user may desire to read the trajectory parameters from either a magnetic tape or disk unit. If this prestored parameters method is utilized, then the user must insure that TRAJ transfers data to FLTMAT at a rate which corresponds to twice the value of DTINT. Table IV lists the required Fortran statements for subroutine TRAJ.

Table IV

Required Statements for TRAJ

```
      SUBROUTINE TRAJ (SETCON)
      LOGICAL SETCON
      IF (.NOT. SETCON) GO TO 100
           [time invariant calculations]
100   CONTINUE
           [time varying calculations]
      RETURN
      END
```

FLTMAT. Subroutine FLTMAT provides MCAP with nonzero values of the filter matrices F, Q, H, and R in turn. The F matrix is calculated using Equation (18) for the nonlinear filter or simply F in the linear filter. Q is found by calculating the term in the brackets of Equation (46). The H matrix is calculated using Equation (24) for the nonlinear filter or simply H in the linear filter. R is the measurement noise covariance matrix. The requirements of FLTMAT for MCAP are identical to those of GCAP. The subroutine arguments for FLTMAT are: SETCON, MATNO, A, and IND. SETCON is again a declared logical variable used to select either time invariant or time varying calculations for the four matrix elements. The integer variable MATNO determines the block of coding accessed to calculate the nonzero values of the desired matrix. The subroutine returns the nonzero values in the variably dimensioned vector A. The vector IND contains the packed row-column pairs of indices which corresponds to the matrix elements returned in the A array. Table V lists the required Fortran statements for subroutine FLTMAT.

Table V

Required Statements for FLTMAT

```
    SUBROUTINE FLTMAT (SETCON, MATNO, A, IND)
    DIMENSION A(1), IND(1)
    LOGICAL SETCON
    GO TO (101, 201, 301, 401), MATNO
101 CONTINUE
    IF (.NOT. SETCON) GO TO 110
        [time invariant calculations of F]
110 CONTINUE
        [time varying calculations of F]
    RETURN
```

Table V (cont'd)

```
201 CONTINUE
    IF (.NOT. SETCON) GO TO 210
         [time invariant calculations of Q]
210 CONTINUE
         [time varying calculations of Q]
    RETURN
301 CONTINUE
         [time varying calculations of H]
    RETURN
401 CONTINUE
    IF (.NOT. SETCON) GO TO 410
         [time invariant calculations of R]
 410 CONTINUE
         [time varying calculations of R]
    RETURN
    END
```

MEAS. Subroutine MEAS calculates and provides to MCAP the noise-corrupted system measurements using either Equation (5) or (12). The subroutine statement contains the following arguments: X, DUM, Z, N2, M, T. The vector X contains the current values of the system states (an N2-dimensioned vector). DUM is an N2-dimensioned vector which can be used by the subroutine for temporary calculations. The calculated measurements are returned to MCAP in Z (an M-dimensioned vector). N2 is the dimension of the system. M is the number of measurements of the system. The current value of the simulation time is contained in T. Subroutine NOISE is called to corrupt the system measurements with noise. Table VI lists the required Fortran statements for subroutine MEAS.

Table VI

Required statements for MEAS

```
SUBROUTINE MEAS (X, DUM, Z, N2, M, T)
DIMENSION X(1), DUM(1), Z(1)
     [measurement calculations]
RETURN
END
```

UPDATXF. Subroutine UPDATXF calculates and provides to MCAP the updated filter estimate, XF+. The subroutine statement contains the following arguments: X, FK, Z, N1, M. The N1-dimensioned vector X contains the current filter state value (XF-). The array FK (of dimension N1xM) contains the Kalman gain matrix values. The noise system measurements are contained in the M-dimensioned vector Z. N1 is the dimension of the filter and M the number of measurements of the system. The updated filter estimate (XF+) is returned to MCAP in X.

UPDATXF is not a constant subroutine because, in general, nonlinear calculations may be necessary [see Equation (20)]. However, a constant subroutine MMXVE is available to perform calculations of a matrix times a vector, $\underline{C} = \underline{A}\ \underline{b}$. The arguments for MMXVE are: A, B, C, N, and M. Where A is an NxM matrix, B is an M-dimensional vector, and C is a vector of dimension M for result. Table VII lists the required Fortran statements for subroutine UPDATXF.

Table VII

Required Statements for UPDATXF

```
SUBROUTINE UPDATXF (X, FK, Z, N1, M)
DIMENSION X(1), FK(1), Z(1)
     [update calculations]
RETURN
END
```

CORRECT. Subroutine CORRECT applies impulsive feedback by updating (selected) system states, and after feedback, resetting the corresponding filter states to zero. The subroutine statement contains the following arguments: XS, XF, N2, and N1. The N2-dimensioned vector XS contains the current values of the system states. XF is an N1-dimensional vector that contains the current values of the filter states. N2 is the dimension of the system. N1 is the dimension of the filter. The impulsive feedback is performed by calculating Equations (51) and (53) and returning the result in the XS and XF arrays. Table VIII lists the required Fortran statements for subroutine CORRECT.

Table VIII

Required Statements for CORRECT

```
SUBROUTINE CORRECT (XS, XF, N2, N1)
     [impulsive feedback calculations]
RETURN
END
```

USRIN. Subroutine USRIN is used to read in the problem dependent user-supplied data for each Monte Carlo analysis. USRIN may be employed in a variety of ways. USRIN may be used to input parameters which change between Monte Carlo simulations but which remain unchanged within a single simulation. Or the user may employ USRIN to create special initial conditions. Whatever the usage, USRIN is a convenient method which permits a user to customize MCAP input data to the specific problem. It is important to note that USRIN is called only once (during function W1). Table IX lists the required Fortran statements for subroutine USRIN.

Table IX

Required Statements for USRIN

```
SUBROUTINE USRIN
     [user specified read]
RETURN
END
```

Section V

A Sample Problem

Introduction

In order to illustrate how the generalized Monte Carlo Analysis Program (MCAP) can be used to analyze and evaluate the performance of a suboptimal Kalman filter, a sample problem is illustrated in this section. The example used is the same problem investigated in the GCAP users guide (Ref 2).

The problem to be studied is a linear system from which multiple, noise-corrupted measurements are available. The truth model is presented and then a suboptimal Kalman filter is designed to estimate certain physical (error) states of this system precisely. These models are then programmed into the appropriate MCAP subroutines and 10 runs performed for a one hour performance analysis.

Problem Formulation

The sample problem concerns the analysis of a single-axis inertial navigation system shown in Figure 5. In the figure, R represents the radius of the earth and g is the acceleration due to gravity. The performance of the system is analyzed for a one-hour time period, during which time, position, and velocity measurements occur at 30 second intervals. The integration interval is 30 seconds. Printed output is required every 40 integration cycles and plotted output is required every integration cycle.

The mathematical model of the inertial navigation system errors is formulated using state variables. That is, the linear time-

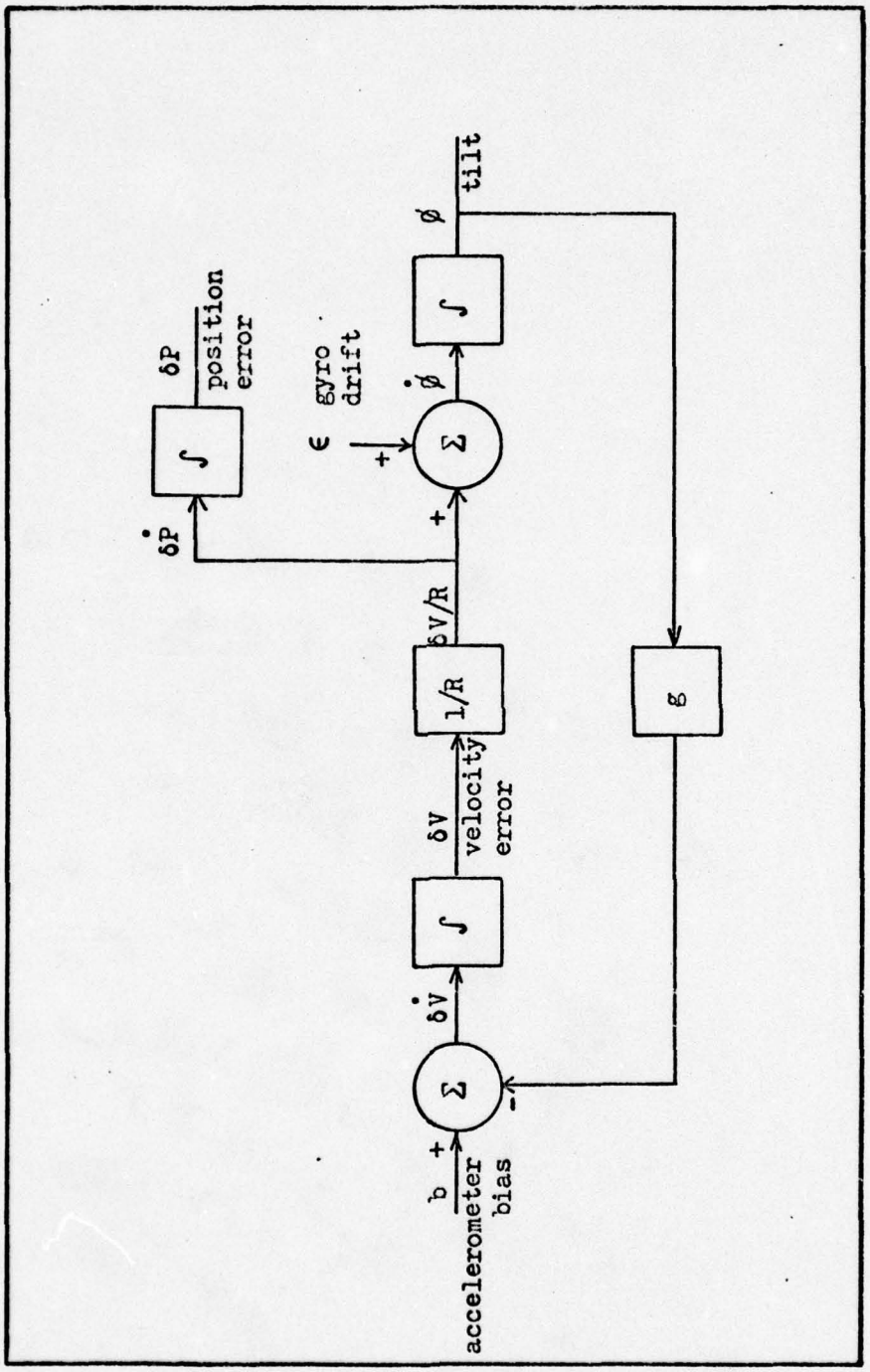


Figure 5. Single Axis Inertial Navigation System

varying system is described by the vector differential equation:

$$\dot{\underline{x}}(t) = \underline{F}(t)\ \underline{x}(t) + \underline{G}(t)\ \underline{w}(t) \tag{54}$$

and the discrete measurements are represented by:

$$\underline{z}(t_i) = \underline{H}(t_i)\ \underline{x}(t_i) + \underline{v}(t_i) \tag{55}$$

Considering the case where $\underline{x}(t)$, $\underline{w}(t)$, and $\underline{v}(t)$ are Gaussian random variables, their probability density functions are completely defined if their first and second moments are known. The following definitions apply:

$\underline{x}(t)$ is an n-vector denoting the system state with zero mean and covariance $\underline{P}(t)$:

$$E[\underline{x}(t)\ \underline{x}(t)^T] = \underline{P}(t)$$

$\underline{F}(t)$ is the nxn plant matrix

$\underline{w}(t)$ is an s vector of white noise inputs which corrupt the states, having zero mean and covariance $\underline{Q}(t)\delta(\tau)$:

$$E[\underline{w}(t)\ \underline{w}(t+\tau)^T] = \underline{Q}(t)\ \delta(\tau)$$

$\underline{G}(t)$ is the nxs noise matrix

$\underline{H}(t_i)$ is the mxn measurement matrix

$\underline{v}(t_i)$ is an m vector of white noise with zero mean and covariance $\underline{R}(t_i)$

$$E[v(t_i)\ v(t_j)^T] = R(t_i)\ \delta_{ij}$$

Truth Model. The single-axis inertial navigation system is described by the following differential equations:

$$\delta\dot{P} = \frac{\delta V}{R} \tag{56}$$

$$\delta\dot{V} = -g\phi + b \tag{57}$$

$$\dot{\phi} = \frac{\delta V}{R} + \epsilon \tag{58}$$

where the gyro drift, ϵ, is modeled as the sum of an exponentially time correlated variable, ϵ_1; and random constant, ϵ_2:

$$\dot{\epsilon}_1 = -\frac{\epsilon_1}{\tau_{\epsilon_1}} + w_1 \tag{59}$$

$$\dot{\epsilon}_2 = 0 \tag{60}$$

where the accelerometer bias is modeled as the sum of two distinct exponentially time correlated variables, b_1 and b_2:

$$\dot{b}_1 = -\frac{b_1}{\tau_{b1}} + w_2 \tag{61}$$

$$\dot{b}_2 = -\frac{b_2}{\tau_{b2}} + w_3 \tag{62}$$

and where

$\tau_{\epsilon_1} = 3600$ (seconds)

$\tau_{b_1} = 300$ (seconds)

$\tau_{b_2} = 3600$ (seconds)

Position and velocity measurements are provided to the filter. However, the system measurements are corrupted with exponentially time correlated noises ΔP and ΔV in addition to the white noises vector $\underline{v}$. ΔP and ΔV are modeled as:

$$\Delta\dot{P} = -\frac{\Delta P}{\tau_{\Delta P}} + w_4 \tag{63}$$

$$\Delta\dot{V} = -\frac{\Delta V}{\tau_{\Delta V}} + w_5 \tag{64}$$

where:

$$\tau_{\Delta P} = 1800 \text{ (seconds)}$$

$$\tau_{\Delta V} = 300 \text{ (seconds)}$$

If the system state vector $\underline{x}_s$ is identified as:

$$\underline{x}_s = \left[\delta P \; \delta V \; \emptyset \; \epsilon_1 \; b_1 \; \epsilon_2 \; b_2 \; \Delta P \; \Delta V\right]^T \tag{65}$$

and $\underline{z}_s$:

$$\underline{z}_s = \left[\delta P_m \; \delta V_m\right]^T$$

then the following matrices describe the system model:

$$\underline{F}_s = \begin{bmatrix} 0 & \frac{1}{R} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -g & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & \frac{1}{R} & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{\tau_{\epsilon_1}} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{\tau_{b_1}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{\tau_{b_2}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{\tau_{\Delta P}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{\tau_{\Delta V}} \end{bmatrix} \tag{66}$$

$$\underline{G}_S = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{67}$$

$$\underline{H}_S = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{68}$$

with:

$$\underline{Q}_S' = \underline{G}_S \underline{Q}_S \underline{G}_S^T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{2\sigma_{\epsilon_1}^2}{\tau_{\epsilon_1}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{2\sigma_{b_1}^2}{\tau_{b_1}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{2\sigma_{b_2}^2}{\tau_{b_2}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{2\sigma_{\Delta P}^2}{\tau_{\Delta P}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{2\sigma_{\Delta V}^2}{\tau_{\Delta V}} \end{bmatrix} \tag{69}$$

$$\underline{R}_S = \begin{bmatrix} \sigma_{\delta P}^2 & 0 \\ 0 & \sigma_{\delta V}^2 \end{bmatrix} \tag{70}$$

where:

$$\sigma_{\epsilon_1} = 4.85 \times 10^{-8} \text{ rad/sec}$$

$$\sigma_{b_1} = 6.44 \times 10^{-3} \text{ ft/sec}^2$$

$$\sigma_{b_2} = 3.22 \times 10^{-2} \text{ ft/sec}^2$$

$$\sigma_{\Delta P} = 3.0 \times 10^{+2} \text{ ft}$$

$$\sigma_{\Delta V} = 5.0 \times 10^{-1} \text{ ft/sec}$$

$$\sigma_{\delta P} = 1.0 \times 10^{+2} \text{ ft}$$

$$\sigma_{\delta V} = 5.0 \times 10^{-1} \text{ ft/sec}$$

To complete the specification of the optimal filter model, Table X lists the initial diagonal values of the apriori covariance matrix, with the off diagonal values zero initially.

Table X

Covariance Matrix $P(t_o)$ Elements

State Variable	$P_{ii}(t_o)$
δP	$3.6 \times 10^{+7}$
δV	$1.0 \times 10^{+2}$
ϕ	3.14×10^{-3}
ϵ_1	2.35×10^{-15}
b_1	4.15×10^{-5}
ϵ_2	5.88×10^{-14}
b_2	1.038×10^{-15}
ΔP	$9.0 \times 10^{+4}$
ΔV	2.5×10^{-1}

Filter Model. The underlying concept for the development of the filter model is to delete some of the less significant or non-dominant system states. Typically, when states are deleted from the truth model, the noise strengths driving the model are increased to compensate for the deleted states. The example used in the GCAP users' guide simply deleted states ϵ_2, b_2, ΔP, and ΔV from the truth model, without any increase in noise strengths.

The deletion of these states yields the filter state vector $\underline{x}_f$ as:

$$\underline{x}_f = \left[\delta P \ \delta V \ \emptyset \ \epsilon_1 \ b_1\right]^T \tag{71}$$

and measurement vector $\underline{z}$:

$$\underline{z}_f = \left[\delta P_m \ \delta V_m\right]^T \tag{72}$$

The following matrices describe the filter model:

$$\underline{F}_f = \begin{bmatrix} 0 & \frac{1}{R} & 0 & 0 & 0 \\ 0 & 0 & -g & 0 & 1 \\ 0 & \frac{1}{R} & 0 & 1 & 0 \\ 0 & 0 & 0 & -\frac{1}{\tau_{\epsilon_1}} & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{\tau_{b_1}} \end{bmatrix} \tag{73}$$

$$\underline{G}_f = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{74}$$

$$\underline{H}_f = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{75}$$

$$R_f = \begin{bmatrix} \sigma_{\delta_P}^2 & 0 \\ 0 & \sigma_{\delta_V}^2 \end{bmatrix} \tag{76}$$

with:

$$\underline{Q}_f' = \underline{G}_f \underline{Q}_f \underline{G}_f^T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{2\sigma_{\epsilon_1}^2}{\tau_{\epsilon_1}} & 0 \\ 0 & 0 & 0 & 0 & \frac{2\sigma_{b_1}^2}{\tau_{b_1}} \end{bmatrix} \tag{77}$$

With these σ's and τ's having the same values as for the truth model. The filter initial covariance $\underline{P}(t_o)$ values are the first five elements noted in Table X.

MCAP Program Implementation

In order to adapt MCAP to this example problem, the user must furnish subroutines USRIN, XFDOT, XSDOT, TRAJ, FLTMAT, MEAS, UPDATXF, and CORRECT. Subroutine USRIN reads, stores, and prints the constant parameters between computer runs. XFDOT calculates the derivatives of the filter states. XSDOT calculates the derivatives of the system states and adds the zero mean Gaussian noise, $\underline{w}_d$, to the system state values at the end of the integration interval. TRAJ provides the constants or time varying parameters required by FLTMAT. FLTMAT calculates the nonzero elements of the filter matrices $\underline{F}$, $\underline{Q}$, $\underline{H}$, and $\underline{R}$.

MEAS calculates the system noise-corrupted measurements. UPDATXF calculates the updated filter estimate XF+. CORRECT applies impulsive feedback by updating the system states and, after feedback, resets the filter states.

USRIN. Subroutine USRIN is usually coded first in order to specify the labeled COMMON variables used in the other user supplied subroutines. These variables are those parameters which may vary between computer runs but remain unchanged within a single computer run (Monte Carlo simulation). Variables typically in this category are the filter tuning parameters: σ_{ϵ_1}, σ_{b_1}, $\sigma_{\Delta P}$, and $\sigma_{\Delta V}$ for this problem. Since the user may also want to analyze the sensitivity of the filter to changes in the system noise strength or noise correlation time, these parameters are also read in by USRIN.

Figure 6 shows the USRIN Fortran coding used for this example. These parameters are input and printed by means of the NAMELIST statement. The values of the parameters are stored in the labeled common blocks: QFNOIS, RFNOIS, TCFIL, QSNOIS, RSNOIS, and TCSYS for use by the other user subroutines. Table XI shows the relationship between the Fortran variable names and the parameters of the filter and truth model.

XFDOT. Figure 7 shows the XFDOT Fortran coding used for this example. Note that since this problem models error states and full state feedback, the values for the filter X and XDOT are zero for all t. Two subroutine constants (RE and G) are used in the first section of the subroutine. In the second section, note that the derivative

```
      SUBROUTINE USRIN
C
C---- THIS ROUTINE READS AND PRINTS THE USER SUPPLIED CONSTANTS.
C
      COMMON/QFNOIS/QF(2)/RFNOIS/RF(2)/TCFIL/TAUF(2)
      COMMON/QSNOIS/QS(5),QSO/RSNOIS/RS(2)/TCSYS/TAUS(5)
C
      NAMELIST/IN/TAUF,QF,RF
      NAMELIST/INN/TAUS,QS,QSO,RS
C
      READ(5,IN)
      WRITE(6,IN)
      READ(5,INN)
      WRITE(6,INN)
      RETURN
      END
```

Figure 6. Fortran Coding for USRIN

```
      SUBROUTINE XFDOT (X,XDOT,N,INIT,NOIS,DT,TIME)
C
C ***************************************************************************
C
C     THIS ROUTINE CALCULATES THE FILTER DIFFERENTIAL EQUATIONS
C          XFDOT = F(XF,T)
C
C       ** NOTE -- SINCE THIS EXAMPLE PROBLEM MODELS ERROR STATES
C                THEN XF(T) = 0. FOR ALL TIME IT WOULD BE SUFFICIENT
C                  TO USE
C     RETURN
C     END
C
C     HOWEVER TO SHOW THE METHOD OF CODING THIS ROUTINE THE FOLLOWING
C     IS PRESENTED
      DIMENSION X(1),XDOT(1)
      COMMON/TCFIL/TAUF(2)
      LOGICAL INIT,NOIS
      IF (.NOT. INIT) GO TO 200
C     INITIALIZATION OF CONSTANTS
      RE = 2.09E+7
      G = 32.2
      RETURN
C     XDOT CALCULATIONS
  200 CONTINUE
      IF (NOIS) GO TO 300
      XDOT(1) = X(2)/RE
      XDOT(2) = -X(3)*G + X(5)
      XDOT(3) =  X(2)/RE + X(4)
      XDOT(4) = -X(4)/TAUF(1)
      XDOT(5) = -X(5)/TAUF(2)
      RETURN
  300 CONTINUE
      RETURN
      END
```

Figure 7. Fortran Coding for XFDOT

Table XI

Correspondence of Fortran Variables and Model Parameters

Fortran		Model
Filter	System	
QF(1)	QS (1)	σ_{ϵ_1}
QF(2)	QS(2)	σ_{b_1}
	QS(3)	σ_{b_2}
	QS(4)	$\sigma_{\Delta P}$
	QS(5)	$\sigma_{\Delta V}$
RF(1)	RS(1)	$\sigma_{\delta P}$
RF(2)	RS(2)	$\sigma_{\delta V}$
TAUF(1)	TAUS(1)	τ_{ϵ_1}
TAUF(2)	TAUS(2)	τ_{b_1}
	TAUS(3)	τ_{b_2}
	TAUS(4)	$\tau_{\Delta P}$
	TAUS(5)	$\tau_{\Delta V}$

calculations $[\underline{F}_f(t)\ \underline{x}_f(t)]$ are directly coded from the filter differential equations. The values for the time correlated variables are obtained from the user specified labeled COMMON. Table XII shows the correspondence between the Fortran variables and the model states.

XSDOT. Figure 8 shows the XSDOT Fortran coding used in this example. Subroutine NOISE is used to obtain a different initial value for the random constant ϵ_2, system state 6, for each run. The second section performs the derivative calculations $[\underline{F}_s(t)\ \underline{x}_s(t)]$ for the system. Note that the derivative calculations are directly coded

```
      SUBROUTINE XSDOT (X,XDOT,N,INIT,NOIS,DT,TIME)
C
C ***************************************************************************
C
C     THIS ROUTINE CALCULATES THE SYSTEM DIFFERENTIAL EQUATIONS
C          XSDOT = F(XS,T) + G*W
      DIMENSION X(1),XDOT(1)
      COMMON/TCSYS/TAUS(5)
      COMMON/QSNOIS/QS(5),QSO
      LOGICAL INIT,NOIS
      IF (.NOT. INIT) GO TO 200
C     INITIALIZATION OF CONSTANTS
      RE = 2.09E+7
      G = 32.2
      XMEAN = 0.
      CALL NOISE (QSO,XMEAN,XNOISE)
      X(6) = XNOISE
      RETURN
C     XDOT CALCULATIONS
  200 CONTINUE
      IF (NOIS) GO TO 300
      XDOT(1) = X(2)/RE
      XDOT(2) = -G*X(3) + X(5) + X(7)
      XDOT(3) = X(2)/RE + X(4) + X(6)
      XDOT(4) = -X(4)/TAUS(1)
      XDOT(5) = -X(5)/TAUS(2)
      XDOT(6) = 0.
      XDOT(7) = -X(7)/TAUS(3)
      XDOT(8) = -X(8)/TAUS(4)
      XDOT(9) = -X(9)/TAUS(5)
      RETURN
  300 CONTINUE
C     XDOT + NOISE CALCULATIONS
      J=3
      DO 310 I=1,5
      RMS = QS(I) * SQRT(2./TAUS(I))
      CALL NOISE (RMS,XMEAN,XNOISE)
      IF (I .EQ. 3) J=J+1
      J = J + 1
      X(J) = X(J) + XNOISE * SQRT(DT)
  310 CONTINUE
      RETURN
      END
```

Figure 8. Fortran Coding for XSDOT

```
      SUBROUTINE TRAJ(SETCON)
C
C---- THIS ROUTINE GENERATES THE FLIGHT PROFILE.
C
      LOGICAL SETCON
C
      COMMON/TRJCOM/RE,G
C
      IF(.NOT.SETCON) GO TO 100
C
C---- THE FOLLOWING CALCULATIONS ARE TIME INVARIANT.
C
      RE=2.09E+7
      G=32.2
  100 CONTINUE
C
C---- THE FOLLOWING CALCULATIONS VARY WITH TIME.
C
      RETURN
      END
```

Figure 9. Fortran Coding for Traj

Table XII

Correspondence of Fortran Variables and Model States

Fortran		Model State
Filter	System	
X(1)	X(1)	δP
X(2)	X(2)	δV
X(3)	X(3)	ϕ
X(4)	X(4)	ϵ_1
X(5)	X(5)	b_1
	X(6)	ϵ_2
	X(7)	b_2
	X(8)	ΔP
	X(9)	ΔV

from Equations (56) through (62). At the end of the integration interval, section three calculates $\underline{w}_d$ and adds it to the system state values. In this example the only noise terms are those added to states ϵ_1, b_1, b_2, ΔP, and ΔV, which correspond to state numbers 4, 5, 7, 8, and 9.

TRAJ. Figure 9 shows the TRAJ Fortran coding used in this example. Due to the simplicity of the problem, the example shown is trivial but it illustrates how TRAJ might be used. It should be noted that the TRAJ shown here is identical to subroutine TRAJ used in the GCAP users guide example.

FLTMAT. Figure 10 shows the FLTMAT Fortran coding used in this example. Subroutine FLTMAT computes the nonzero values of the state space matrices for the filter and stores these values in the variably dimensioned array A. A separate block of code is used for each matrix. Within each block of coding, the nonzero elements of that matrix are calculated using parameters and constants transmitted by labeled COMMON and equated to the appropriate elements of the A array.

The nonzero elements are determined by examining the filter dynamics matrix, Equation (73). Row 1 column 2 is the first nonzero element, row 2 column 3 is the second nonzero element, and so on. Table XIII shows the correspondence between the filter matrices nonzero elements and the A array. The nonzero values for matrices $\underline{F}_f$, $\underline{Q}_f$, $\underline{H}_f$, and $\underline{R}_f$ are coded using Equations (73), (77), (75), and (76). It should be noted that the FLTMAT shown here is identical to subroutine FLTMAT used in the GCAP users guide example.

MEAS. Figure 11 shows the MEAS Fortran coding used in this example. This subroutine calculates and provides to MCAP the noise-corrupted system measurements. The system measurements $[\underline{H}_s(t_i)\ \underline{x}_s(t_i)]$ are calculated and the results of subroutine NOISE added to obtain the realized value of the measurement.

UPDATXF. Figure 12 shows the UPDATXF Fortran coding used in this example. This subroutine calculates the filter estimate XF+ by solving the filter update equation, Equation (8). Subroutine MMXVE is used to multiply the filter Kalman gain matrix FK times the measurement residual $[\underline{Y}_i - \underline{H}_f(t_i)\hat{\underline{x}}_f(t_i^-)]$ stored in variable Z. The updated filter estimate XF+ is returned to MCAP in the array X.

```
      SUBROUTINE FLTMAT(SETCON,MATNO,A,IND)
C
C---- THIS ROUTINE CALCULATES THE NON-ZERO ELEMENTS OF THE
C---- FILTER MATRICES FF, QF, HF AND RF, STORING THEM IN THE ARRAY A.
C
C---- IF SETCON .EQ. FALSE, ONLY TIME VARIANT ELEMENTS ARE GENERATED.
C
      LOGICAL SETCON
C
      COMMON/QFNOIS/QF(2)/RFNOIS/RF(2)/TCFIL/TAUF(2)
      COMMON/TIMER/TIME,TIMUD,TIMINT
      COMMON/TRJCOM/RE,G
C
      DIMENSION A(1),IND(1)
C
C---- DETERMINE WHICH MATRIX IS TO BE GENERATED.
C
      GO TO(101,102,103,104),MATNO
  101 CONTINUE
      IF(.NOT.SETCON) GO TO 1001
C
C---- CALCULATE INVARIANT ELEMENTS OF FF.
C
      A(1)=1./RE
      A(2)=-G
      A(3)=1.
      A(4)=1./RE
      A(5)=1.
      A(6)=-1./TAUF(1)
      A(7)=-1./TAUF(2)
 1001 CONTINUE
C
C---- CALCULATE TIME VARIANT ELEMENTS OF FF.
C
      RETURN
  102 CONTINUE
      IF(.NOT.SETCON) GO TO 1002

C
C---- CALCULATE INVARIANT ELEMENTS OF QF.
C
      DO 1022 II=1,2
      A(II)=2.0*QF(II)*QF(II)/TAUF(II)
 1022 CONTINUE
 1002 CONTINUE
C
C---- CALCULATE TIME VARIANT ELEMENTS OF QF.
C
      RETURN
  103 CONTINUE
      IF(.NOT.SETCON) GO TO 1003
C
C---- CALCULATE INVARIANT ELEMENTS OF HF.
C
      A(1)=1.
      A(2)=1.
 1003 CONTINUE
C
C---- CALCULATE TIME VARIANT ELEMENTS OF HF.
C
      RETURN
  104 CONTINUE
      IF(.NOT.SETCON) GO TO 1004
C
C---- CALCULATE INVARIANT ELEMENTS OF RF.
C
      A(1)=RF(1)**2
      A(2)=RF(2)**2
 1004 CONTINUE
C
C---- CALCULATE TIME VARIANT ELEMENTS OF RF.
C
      RETURN
      END
```

Figure 10. Fortran Coding for FLTMAT

```
      SUBROUTINE MEAS (X,XDOT,Z,N,M,T)
C
C*****************************************************************************
C
C     THIS ROUTINE CALCULATES THE NOISE-CORRUPTED MEASUREMENTS USED
C     BY THE FILTER
C          Z = H(XS,T) + V
C
      COMMON/RSNOIS/RS(2)
      DIMENSION X(1),Z(1)
      XMEAN = 0.
      CALL NOISE (RS(1),XMEAN,XNOISE)
      Z(1) = X(1) + X(8) + XNOISE
      CALL NOISE (RS(2),XMEAN,XNOISE)
      Z(2) = X(2) + X(9) + XNOISE
      RETURN
      END
```

Figure 11. Fortran Coding for MEAS

```
      SUBROUTINE UPDATXF (X,FK,Z,N,M)
C
C  ***************************************************************************
C
C     THIS ROUTINE CALCULATES XF+ GIVEN THE FILTER GAIN MATRIX FK
C     AND THE SYSTEM MEASUREMENTS, Z
C
C          XF+ = XF- + K*(Z(SYSTEM) - H(XF-,T))
C
      COMMON WRK(1)
      DIMENSION X(1),FK(1),Z(1)
      DO 110 I=1,M
      Z(I) = Z(I) - X(I)
  110 CONTINUE
      CALL MMXVE (FK,Z,WRK,N,M)
      DO 120 I=1,M
      X(I) = X(I) + WRK(I)
  120 CONTINUE
      RETURN
      END
```

Figure 12. Fortran Coding for UPDATXF

```
      SUBROUTINE CORRECT (XS,XF,IXS,IXF)
C
C  ***************************************************************************
C
C     THIS ROUTINE APPLIES INSTANTANEOUS FEEDBACK CORRECTION TO
C     SYSTEM STATES AND RESETS THE FILTER ESTIMATES
C
      DIMENSION XS(1),XF(1)
      DO 100 I=1,IXF
      XS(I) = XS(I) - XF(I)
      XF(I) = 0.
  100 CONTINUE
      RETURN
      END
```

Figure 13. Fortran Coding for CORRECT

Table XIII

Correspondence of Nonzero Elements and the A Array

Nonzero Element			Array A
Matrix	Row	Column	
$\underline{F}_f$	1	2	1
	2	3	2
	2	5	3
	3	2	4
	3	4	5
	4	4	6
	5	5	7
$\underline{Q}_f$	4	4	1
	5	5	2
$\underline{H}_f$	1	1	1
	2	2	2
$\underline{R}_f$	1	1	1
	2	2	2

CORRECT. Figure 13 shows the CORRECT Fortran coding used in this example. All of the filter states estimated are used by the inertial navigation system to zero out the system error states. Consequently, CORRECT calculates the instantaneous feedback correction of the system states using Equation (51). After the correction has been applied, the filter states are reset to zero.

MCAP Sample Input Data

Table I listed and described the input parameters required for MCAP. Figure 14 shows the input card data deck used in this example. As mentioned in the problem formulation, the performance of the inertial navigation system is to be analyzed from the initial time (TØ = 0.) for a one hour time period (TF = 3600. seconds). The measurements occur at 30 second intervals (DTUP = 30.) and the integration time is 30 seconds (DTINT = 30.). Printed output is required every 40 integration cycles (IPRCT = 40) for two of the simulation runs (IRNPRCT = 2). The Kalman gain matrix $\underline{K}$ is also printed (IPRK = 1). Plotted output is required every integration cycle (IPLCT = 1). Ten runs are to be performed in the performance analysis (IPASS = 10). The random number generator is started from a "seed" of 77 (ISEED = 77).

The number of nonzero terms and indices of those terms is obtained from Table XIII and shown in the proper format in Figure 14. Since error states are being modeled and since this is an example, the initial conditions for the filter [XF(0)] and system [XS(0)] are set to zero. The filter covariance diagonal element initial condition [PF(0)] is obtained from Table X.

The USRIN data is shown in Figure 14 as the NAMELIST $IN and $INN group names. The values shown in the figure are obtained using Table XI and the problem formulation.

MCAP Sample Output Data

The printed data from the Monte Carlo simulation of 10 runs is shown in Figures 15 through 23. The printed data consists of the

```
MONTE CARLO ANALYSIS PROGRAM (MCAP) EXAMPLE CASE
 $USRCTL  TO = 0.0,  TF = 3600.0,  DTUP = 30.0,  DTINT = 30.,
  N1 = 5,  N2 = 9,  M = 2,
  IPASS = 10,  IRNPRCT = 2,  ISEED = 77,
  IPRCT = 40,  IPLCT = 1,  IPRK = 1 $
  7     ** NZFF **
  1  2       2  3       2  5       3  2       3  4       4  4       5  5
  2     ** NZQF **
  4  4       5  5
  2     ** NZHF **
  1  1       2  2
  2     ** NZRF **
  1  1       2  2
 0.                     ** XF(0) **
 0.
 0.
 0.
 0.
 0.                     ** XS(0) **
 0.
 0.
 0.
 0.
 0.
 0.
 0.
 3.600E+07              ** PF(0) **
 1.000E+02
 3.140E-04
 2.350E-15
 4.150E-05
 $IN TAUF(1) = 2*3600.,
 QF(1) = 4.85E-8,  6.44E-3,
 RF(1) = 100.0,  0.5,  $
 $INN TAUS(1) = 2*3600.,  300.0,  1800.0,  300.0,
 QS(1) = 4.85E-8,  6.44E-3,  3.22E-2,  3.0E+2,  5.0E-1,
 RS(1) = 100.0,  0.5,  $
```

Figure 14. Data Input Card Deck

```
GENERAL MONTE CARLO ANALYSIS PROGRAM  12/09/77    17.15.02.
MONTE CARLO ANALYSIS PROGRAM (MCAP) EXAMPLE CASE
```

Figure 15. Printed Output - Page 1

```
$IN
TAUF    = .35E+04, .36E+04,
OF      = .485E-07, .644E-02,
RF      = .1E+03, .5E+00,
$END
```

Figure 17. Printed Output - Page 3

```
$INN
TAUS    = .35E+04, .36E+04, .3E+03, .18E+04, .3E+03,
OS      = .485E-07, .644E-02, .322E-01, .3E+03, .5E+00,
OS0     = 0.0,
PS      = .1E+03, .5E+00,
$END
```

Figure 18. Printed Output - Page 4

```
$USRCTL
N1       = 5,
N2       = 9,
M        = 2,
T0       = 0.0,
TF       = .35E+04,
DTOP     = .3E+02,
DTINT    = .3E+02,
IPRCT    = 40,
IPLOT    = 1,
IPRSYM   = 0,
IPRK     = 1,
IPFILE   = 0,
TIMUDO   = 0.0,
TIMNTO   = 0.0,
ISEED    = 77,
IPASS    = 10,
IPNPRCT  = 2,
$END
WARNING--THE NUMBER OF NON-ZERO ELEMENTS OF ANY SPARSE MATRIX MAY NOT EXCEED 200

THE FOLLOWING   7 INDICES ARE FOR THE NON-ZERO ELEMENTS OF THE SPARSE MATRIX FF
  1     1   2         2     2   3         3     2   5         4     3   2         5     3   4         6     4   4         7     5   5

THE FOLLOWING   2 INDICES ARE FOR THE NON-ZERO ELEMENTS OF THE SPARSE MATRIX QF
  1     4   4         2     5   5

THE FOLLOWING   2 INDICES ARE FOR THE NON-ZERO ELEMENTS OF THE SPARSE MATRIX HF
  1     1   1         2     2   2

THE FOLLOWING   2 INDICES ARE FOR THE NON-ZERO ELEMENTS OF THE SPARSE MATRIX RF
  1     1   1         2     2   2

A MUST BE DIMENSIONED AT LEAST     564
THIS PROGRAM REQUIRES A FIELD LENGTH OF AT LEAST 000000000000034005
```

Figure 16. Printed Output - Page 2

```
GENERAL MONTE CARLO ANALYSIS PROGRAM  12/09/77    17.15.02.  RUN NUMBER -    1
MONTE CARLO ANALYSIS PROGRAM (MCAP) EXAMPLE CASE

VALUE OF XF(0)          TIME=     0.0000 TIMUD=     0.0000 TIMINT=     0.0000
   1  0.              2  0.              3  0.              4  0.              5  0.

VALUE OF XS(0)          TIME=     0.0000 TIMUD=     0.0000 TIMINT=     0.0000
   1  0.              2  0.              3  0.              4  0.              5  0.
   6  0.              7  0.              8  0.              9  0.

SQRT OF DIAGONAL ELEMENTS OF PF(0)        TIME=     0.0000 TIMUD=     0.0000 TIMINT=     0.0000
  1  1  5.000000E+03   2  2  1.000000E+01   3  3  1.772035E-02   4  4  4.847680E-08   5  5  6.442049E-03

VALUE OF XS-            TIME=  1170.0001 TIMUD=  1200.0001 TIMINT=  1200.0001
   1  1.398121E+02     2  1.595285E+00     3 -6.540515E-05     4 -1.277138E-08     5 -3.424934E-03
   6  0.               7  2.379569E-02     8 -1.518751E+02     9  7.685318E-01

VALUE OF XF-            TIME=  1170.0001 TIMUD=  1200.0001 TIMINT=  1200.0001
   1  0.              2  0.              3  0.              4  0.              5  0.

SQRT OF DIAGONAL ELEMENTS OF PF-          TIME= 1170.0001 TIMUD= 1200.0001 TIMINT= 1200.0001
  1  1  1.611276E+01   2  2  3.017159E-01   3  3  1.917105E-04   4  4  4.695444E-08   5  5  6.172167E-03

KF                                        TIME= 1170.0001 TIMUD= 1200.0001 TIMINT= 1200.0001
  1   1  2.439983E-02   2  1.192998E-06
  2   1  2.992494E-11   2  2.669320E-01
  3   1  5.692290E-17   2 -2.128930E-05
  4   1  1.876008E-20   2 -4.535581E-09
  5   1 -2.850859E-15   2  6.000405E-04

SQRT OF DIAGONAL ELEMENTS OF PF+          TIME= 1170.0001 TIMUD= 1200.0001 TIMINT= 1200.0001
  1  1  1.581133E+01   2  2  2.583273E-01   3  3  1.913070E-04   4  4  4.687968E-08   5  5  6.158998E-03

VALUE OF ZS             TIME=  1170.0001 TIMUD=  1200.0001 TIMINT=  1200.0001
   1 -1.570703E+02     2  1.374479E+00

VALUE OF XF+            TIME=  1170.0001 TIMUD=  1200.0001 TIMINT=  1200.0001
   1 -3.926723E+00     2  3.668925E-01     3  0.              4  0.              5  0.

VALUE OF XF +C          TIME=  1170.0001 TIMUD=  1200.0001 TIMINT=  1200.0001
   1  0.              2  0.              3  0.              4  0.              5  0.

VALUE OF XS+            TIME=  1170.0001 TIMUD=  1200.0001 TIMINT=  1200.0001
   1  1.437389E+02     2  1.228393E+00     3 -6.540515E-05     4 -1.277138E-08     5 -3.424934E-03
   6  0.               7  2.379569E-02     8 -1.518751E+02     9  7.685318E-01

VALUE OF XS-            TIME=  2370.0001 TIMUD=  2400.0001 TIMINT=  2400.0001
   1  2.035237E+02     2  1.932288E-01     3 -7.083044E-06     4 -2.431310E-08     5 -6.029883E-03
   6  0.               7  1.021271E-02     8 -2.568483E+02     9 -6.532894E-01

VALUE OF XF-            TIME=  2370.0001 TIMUD=  2400.0001 TIMINT=  2400.0001
   1  0.              2  0.              3  0.              4  0.              5  0.

SQRT OF DIAGONAL ELEMENTS OF PF-          TIME= 2370.0001 TIMUD= 2400.0001 TIMINT= 2400.0001
  1  1  1.125086E+01   2  2  3.017032E-01   3  3  1.833056E-04   4  4  4.548293E-08   5  5  5.910159E-03

KF                                        TIME= 2370.0001 TIMUD= 2400.0001 TIMINT= 2400.0001
  1   1  1.249995E-02   2  1.198128E-06
  2   1  2.955319E-11   2  2.669156E-01
  3   1  9.445273E-16   2 -2.085635E-05
  4   1 -6.474181E-20   2 -4.626555E-09
  5   1  9.851023E-15   2  7.037952E-04
```

Figure 19. Printed Output - Page 5

AD-A055 189 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 17/7
A GENERALIZED MONTE CARLO ANALYSIS PROGRAM FOR KALMAN FILTER DE--ETC(U)
DEC 77 K L JACKSON

UNCLASSIFIED AFIT/GGC/EE/77-6 NL

4 OF 4
AD
A055189

END
DATE
FILMED
7-78
DDC

```
SORT OF DIAGONAL ELEMENTS OF PF+          TIME= 2370.0001 TIMUD= 2400.0001 TIMINT= 2400.0001
  1  1  1.118032E+01    2  2  2.583194E-01    3  3  1.829005E-04    4  4  4.540261E-08    5  5  5.895851E-03

VALUE OF ZS              TIME=  2370.0001 TIMUD=  2400.0001 TIMINT=  2400.0001
   1  2.122113E+01     2 -3.527598E-01

VALUE OF XF+             TIME=  2370.0001 TIMUD=  2400.0001 TIMINT=  2400.0001
   1  2.552628E-01     2 -9.415708E-02    3  0.               4  0.               5  0.

VALUE OF XF +0           TIME=  2370.0001 TIMUD=  2400.0001 TIMINT=  2400.0001
   1  0.               2  0.               3  0.               4  0.               5  0.

VALUE OF XS+             TIME=  2370.0001 TIMUD=  2400.0001 TIMINT=  2400.0001
   1  2.032584E+02     2  2.873859E-01     3 -7.083044E-06     4 -2.431310E-08     5 -6.029863E-03
   6  0.               7  1.021271E-02     8 -2.568403E+02     9 -6.532894E-01

VALUE OF XS-             TIME=  3570.0002 TIMUD=  3600.0000 TIMINT=  3600.0002
   1  1.595003E+02     2  4.107415E+00     3  1.536307E-04     4 -1.630440E-08     5  7.864397E-03
   6  0.               7  7.119570E-03     8 -2.068486E+02     9  2.726407E-01

VALUE OF XF-             TIME=  3570.0002 TIMUD=  3600.0000 TIMINT=  3600.0002
   1  0.               2  0.               3  0.               4  0.               5  0.

SORT OF DIAGONAL ELEMENTS OF PF-          TIME= 3570.0002 TIMUD= 3600.0000 TIMINT= 3600.0002
  1  1  9.165974E+00    2  2  3.016970E-01    3  3  1.784563E-04    4  4  4.463838E-08    5  5  5.758500E-03

KF                                        TIME= 3570.0002 TIMUD= 3600.0000 TIMINT= 3600.0002
  1   1  8.333314E-03    2  1.199544E-06
  2   1  2.998859E-11    2  2.663075E-01
  3   1  1.446282E-15    2 -2.061586E-05
  4   1 -1.363582E-19    2 -4.677179E-09
  5   1  2.074401E-14    2  7.114508E-04

SORT OF DIAGONAL ELEMENTS OF PF+          TIME= 3570.0002 TIMUD= 3600.0000 TIMINT= 3600.0002
  1  1  9.128599E+00    2  2  2.583155E-01    3  3  1.780495E-04    4  4  4.455474E-08    5  5  5.743493E-03

VALUE OF ZS              TIME=  3570.0002 TIMUD=  3600.0000 TIMINT=  3600.0002
   1  5.094784E+00     2  4.166544E+00

VALUE OF XF+             TIME=  3570.0002 TIMUD=  3600.0000 TIMINT=  3600.0002
   1  4.246144E-02     2  1.112082E+00     3  0.               4  0.               5  0.

VALUE OF XF +0           TIME=  3570.0002 TIMUD=  3600.0000 TIMINT=  3600.0002
   1  0.               2  0.               3  0.               4  0.               5  0.

VALUE OF XS+             TIME=  3570.0002 TIMUD=  3600.0000 TIMINT=  3600.0002
   1  1.595578E+02     2  2.991333E+00     3  1.536307E-04     4 -1.630440E-08     5  7.864397E-03
   6  0.               7  7.119570E-03     8 -2.068486E+02     9  2.726407E-01

VALUE OF MAX PF          TIME=  3600.0002 TIMUD=  3600.0000 TIMINT=  3600.0002
   1  6.000000E+03     2  1.981853E+01     3  1.772005E-02     4  4.847847E-08     5  6.442049E-03

PFFINAL                                   TIME= 3600.0002 TIMUD= 3600.0000 TIMINT= 3600.0002
  1   1  8.333314E+01
  2   1  2.998859E-07    2  6.672687E-02
  3   1  1.446282E-11    2 -5.153964E-06    3  3.170172E-08
  4   1 -1.363582E-15    2 -1.169295E-09    3  2.151677E-12    4  1.985125E-15
  5   1  2.074401E-10    2  1.778627E-04    3  9.607743E-07    4  5.580183E-11    5  3.298771E-05

RUN NUMBER -     1 COMPLETE
```

Figure 20. Printed Output - Page 6

```
GENERAL MONTE CARLO ANALYSIS PROGRAM  12/09/77    17.15.02.  RUN NUMBER -    2
MONTE CARLO ANALYSIS PROGRAM (MCAP) EXAMPLE CASE

VALUE OF XF(0)          TIME=     0.0000 TIMUD=     0.0000 TIMINT=     0.0000
   1  0.              2  0.              3  0.              4  0.              5  0.

VALUE OF XS(0)          TIME=     0.0000 TIMUD=     0.0000 TIMINT=     0.0000
   1  0.              2  0.              3  0.              4  0.              5  0.
   6  0.              7  0.              8  0.              9  0.

SQRT OF DIAGONAL ELEMENTS OF PF(0)       TIME=    0.0000 TIMUD=    0.0000 TIMINT=    0.0000
  1  1  5.000000E+03   2  2  1.000000E+01   3  3  1.772005E-02   4  4  4.847680E-08   5  5  6.442049E-03

VALUE OF XS-            TIME=  1170.0001 TIMUD=  1200.0001 TIMINT=  1200.0001
   1  1.268747E+02     2 -1.474136E+00     3  6.659532E-05     4  1.486038E-08     5  7.556419E-04
   6  0.               7 -2.705682E-02     8 -1.518924E+02     9 -2.005660E-01

VALUE OF XF-            TIME=  1170.0001 TIMUD=  1200.0001 TIMINT=  1200.0001
   1  0.              2  0.              3  0.              4  0.              5  0.

SQRT OF DIAGONAL ELEMENTS OF PF-         TIME= 1170.0001 TIMUD= 1200.0001 TIMINT= 1200.0001
  1  1  1.601276E+01   2  2  3.017159E-01   3  3  1.917105E-04   4  4  4.695444E-08   5  5  6.172167E-03

KF                                       TIME= 1170.0001 TIMUD= 1200.0001 TIMINT= 1200.0001
  1   1  2.499983E-02   2  1.192998E-06
  2   1  2.982494E-11   2  2.669320E-01
  3   1  6.582298E-17   2 -2.128930E-05
  4   1  1.876008E-20   2 -4.536581E-09
  5   1 -2.850359E-15   2  6.900405E-04

SQRT OF DIAGONAL ELEMENTS OF PF+         TIME= 1170.0001 TIMUD= 1200.0001 TIMINT= 1200.0001
  1  1  1.581133E+01   2  2  2.583273E-01   3  3  1.913070E-04   4  4  4.687968E-08   5  5  6.158998E-03

VALUE OF ZS             TIME=  1170.0001 TIMUD=  1200.0001 TIMINT=  1200.0001
   1 -1.598525E+02     2 -1.741651E+00

VALUE OF XF+            TIME=  1170.0001 TIMUD=  1200.0001 TIMINT=  1200.0001
   1 -4.746285E+00     2 -4.649025E-01     3  0.              4  0.              5  0.

VALUE OF XF +0          TIME=  1170.0001 TIMUD=  1200.0001 TIMINT=  1200.0001
   1  0.              2  0.              3  0.              4  0.              5  0.

VALUE OF XS+            TIME=  1170.0001 TIMUD=  1200.0001 TIMINT=  1200.0001
   1  1.311210E+02     2 -1.000233E+00     3  6.659532E-05     4  1.486038E-08     5  7.556419E-04
   6  0.               7 -2.705682E-02     8 -1.515924E+02     9 -2.005660E-01

VALUE OF XS-            TIME=  2370.0001 TIMUD=  2400.0001 TIMINT=  2400.0001
   1  1.096735E+02     2  9.248441E-01     3  1.388530E-04     4  3.198902E-08     5  8.170805E-04
   6  0.               7  1.236270E-03     8 -2.857497E+02     9 -1.008738E-01

VALUE OF XF-            TIME=  2370.0001 TIMUD=  2400.0001 TIMINT=  2400.0001
   1  0.              2  0.              3  0.              4  0.              5  0.

SQRT OF DIAGONAL ELEMENTS OF PF-         TIME= 2370.0001 TIMUD= 2400.0001 TIMINT= 2400.0001
  1  1  1.125095E+01   2  2  3.017032E-01   3  3  1.833056E-04   4  4  4.548293E-08   5  5  5.910159E-03

KF                                       TIME= 2370.0001 TIMUD= 2400.0001 TIMINT= 2400.0001
  1   1  1.249996E-02   2  1.198128E-06
  2   1  2.995319E-11   2  2.669156E-01
  3   1  9.445273E-16   2 -2.085635E-05
  4   1 -6.474181E-20   2 -4.626555E-09
  5   1  9.851023E-15   2  7.037952E-04
```

Figure 21. Printed Output - Page 7

```
SQRT OF DIAGONAL ELEMENTS OF PF+          TIME= 2370.0001 TIMUD= 2400.0001 TIMINT= 2400.0001
  1  1  1.118032E+01   2  2  2.583194E-01   3  3  1.829005E-04   4  4  4.540261E-08   5  5  5.895851E-03

VALUE OF ZS              TIME=  2370.0001 TIMUD=  2400.0001 TIMINT=  2400.0001
   1 -9.335562E+01    2  1.557474E+00

VALUE OF XF+             TIME=  2370.0001 TIMUD=  2400.0001 TIMINT=  2400.0001
   1 -1.241939E+00    2  4.157139E-01    3  0.              4  0.              5  0.

VALUE OF XF +C           TIME=  2370.0001 TIMUD=  2400.0001 TIMINT=  2400.0001
   1  0.              2  0.              3  0.              4  0.              5  0.

VALUE OF XS+             TIME=  2370.0001 TIMUD=  2400.0001 TIMINT=  2400.0001
   1  1.109155E+02    2  5.091302E-01    3  1.388530E-04    4  3.198902E-08    5  8.170805E-04
   6  0.              7  1.236270E-03    8 -2.887497E+02    9 -1.008738E-01

VALUE OF XS-             TIME=  3570.0002 TIMUD=  3600.0000 TIMINT=  3600.0002
   1  1.362529E+02    2 -3.601644E+00    3  1.278268E-04    4  2.585933E-08    5 -1.839711E-03
   6  0.              7 -3.529936E-02    8 -1.270089E+02    9 -8.568652E-01

VALUE OF XF-             TIME=  3570.0002 TIMUD=  3600.0000 TIMINT=  3600.0002
   1  0.              2  0.              3  0.              4  0.              5  0.

SQRT OF DIAGONAL ELEMENTS OF PF-          TIME= 3570.0002 TIMUD= 3600.0000 TIMINT= 3600.0002
  1  1  9.166974E+00   2  2  3.016970E-01   3  3  1.784553E-04   4  4  4.463838E-08   5  5  5.758500E-03

KF                                        TIME= 3570.0002 TIMUD= 3600.0000 TIMINT= 3600.0002
  1   1  8.333314E-03   2  1.199544E-06
  2   1  2.998859E-11   2  2.669075E-01
  3   1  1.445282E-15   2 -2.061586E-05
  4   1 -1.363582E-19   2 -4.677179E-09
  5   1  2.074401E-14   2  7.114508E-04

SQRT OF DIAGONAL ELEMENTS OF PF+          TIME= 3570.0002 TIMUD= 3600.0000 TIMINT= 3600.0002
  1  1  9.128699E+00   2  2  2.583155E-01   3  3  1.780493E-04   4  4  4.455474E-08   5  5  5.743493E-03

VALUE OF ZS              TIME=  3570.0002 TIMUD=  3600.0000 TIMINT=  3600.0002
   1 -1.215627E+02    2 -4.528935E+00

VALUE OF XF+             TIME=  3570.0002 TIMUD=  3600.0000 TIMINT=  3600.0002
   1 -1.013026E+00    2 -1.208607E+00    3  0.              4  0.              5  0.

VALUE OF XF +C           TIME=  3570.0002 TIMUD=  3600.0000 TIMINT=  3600.0002
   1  0.              2  0.              3  0.              4  0.              5  0.

VALUE OF XS+             TIME=  3570.0002 TIMUD=  3600.0000 TIMINT=  3600.0002
   1  1.372659E+02    2 -2.392837E+00    3  1.278268E-04    4  2.585933E-08    5 -1.839711E-03
   6  0.              7 -3.529936E-02    8 -1.270089E+02    9 -8.568652E-01

VALUE OF MAX PF          TIME=  3600.0002 TIMUD=  3600.0000 TIMINT=  3600.0002
   1  6.000000E+03    2  1.981853E+01    3  1.772005E-02    4  4.647847E-06    5  6.442049E-03

PFFINAL                                   TIME= 3600.0002 TIMUD= 3600.0000 TIMINT= 3600.0002
  1   1  8.333314E+01
  2   1  2.998859E-07   2  6.672687E-02
  3   1  1.445282E-11   2 -5.153964E-06   3  3.170172E-08
  4   1 -1.363582E-15   2 -1.169295E-09   3  2.151677E-12   4  1.995125E-15
  5   1  2.074401E-10   2  1.778627E-04   3  9.607743E-07   4  5.580183E-11   5  3.298771E-05

RUN NUMBER -     2 COMPLETE

RUN NUMBER -     3 COMPLETE

RUN NUMBER -     4 COMPLETE

RUN NUMBER -     5 COMPLETE
```

Figure 22. Printed Output - Page 8

```
 RUN NUMBER -     5 COMPLETE
 RUN NUMBER -     7 COMPLETE
 RUN NUMBER -     8 COMPLETE
 RUN NUMBER -     9 COMPLETE
 RUN NUMBER -    10 COMPLETE
```

Figure 23. Printed Output - Page 9

following items:

1. The problem title, date, and time the simulation was conducted.
2. The echo printing of $USRCTL.
3. The echo printing of the number of nonzero elements for each state space matrix together with the corresponding indices of these elements.
4. The minimum required dimension of the unlabeled COMMON block.
5. The echo printing of USRIN data $IN and $INN.
6. The initial values of the filter and system states [XF(0) and XS(0)].
7. The square root of the initial diagonal values of the filter covariance matrix $[\{PF(0)\}^{1/2}]$.
8. The values of the filter and system states before and after the measurement and after the impulsive correction.
9. The values of the measurements.
10. The square root of the diagonal elements of the filter covariance matrix at times other than TØ and TF.
11. The full Kalman gain matrix, $\underline{K}$, at the update times.
12. The final lower triangle values of the filter covariance matrix.
13. The number of simulation runs completed.

The plotted data consists of the square root of the average filter estimate of the standard deviation $\sqrt{\overline{PF}_{kk}}$ along with the corresponding computed standard deviation, $\underline{s}_e(t_j)$, versus time; and a plot of the calculated mean error plus and minus one computed standard deviation versus time for each filter state. The plot data was originally

written onto TAPE3 by MCAP and then used by a separate Calcomp plotting program. Figures 24 through 28 are the standard deviation plots and Figures 29 through 33 are the mean error $\pm s_e$ plots for this example. It should be noted that this example has not been tuned.

FIG 24. LINEAR FILTER EXAMPLE (10 RUNS)

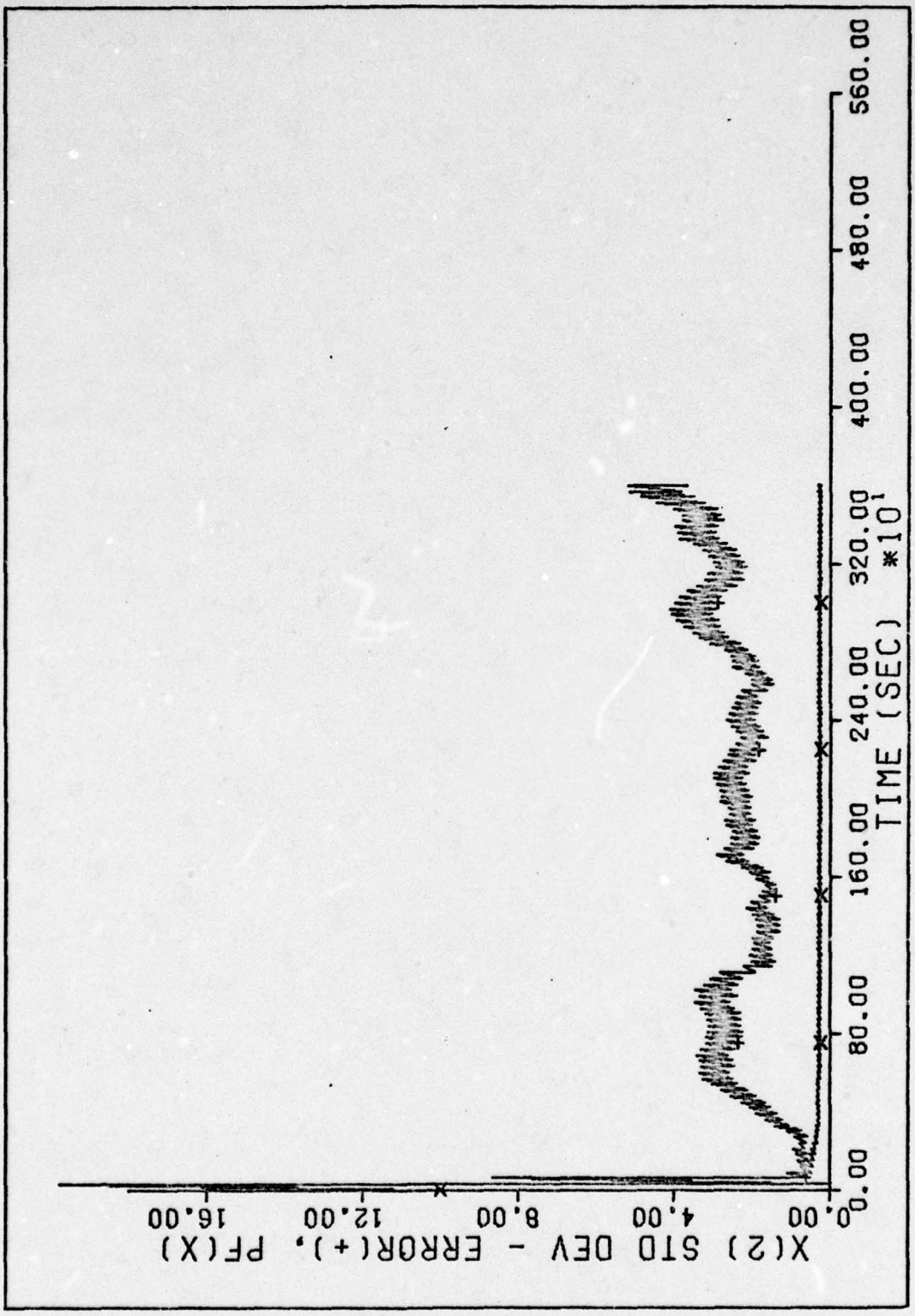


FIG 25. LINEAR FILTER EXAMPLE (10 RUNS)

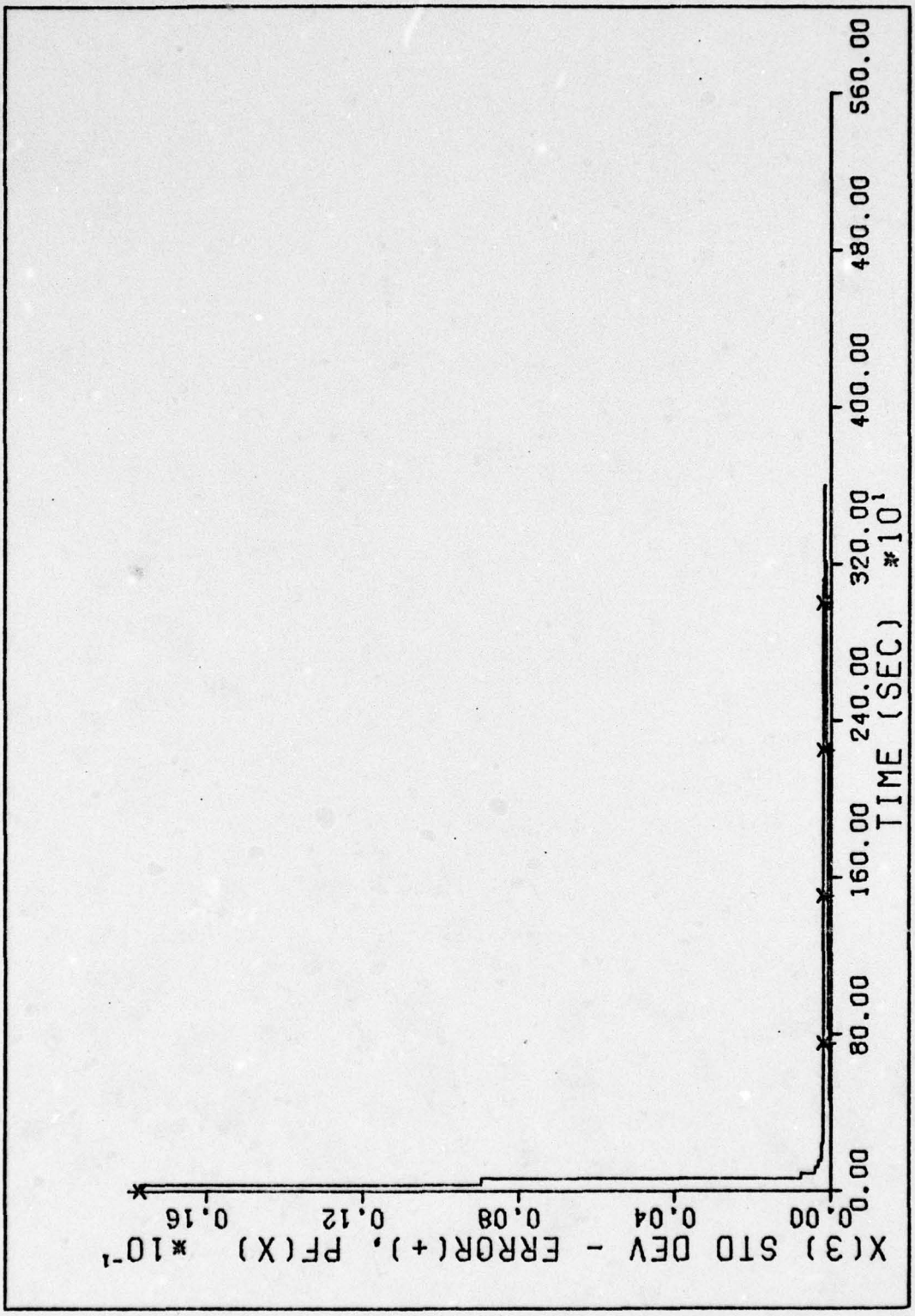


FIG 26. LINEAR FILTER EXAMPLE (10 RUNS)

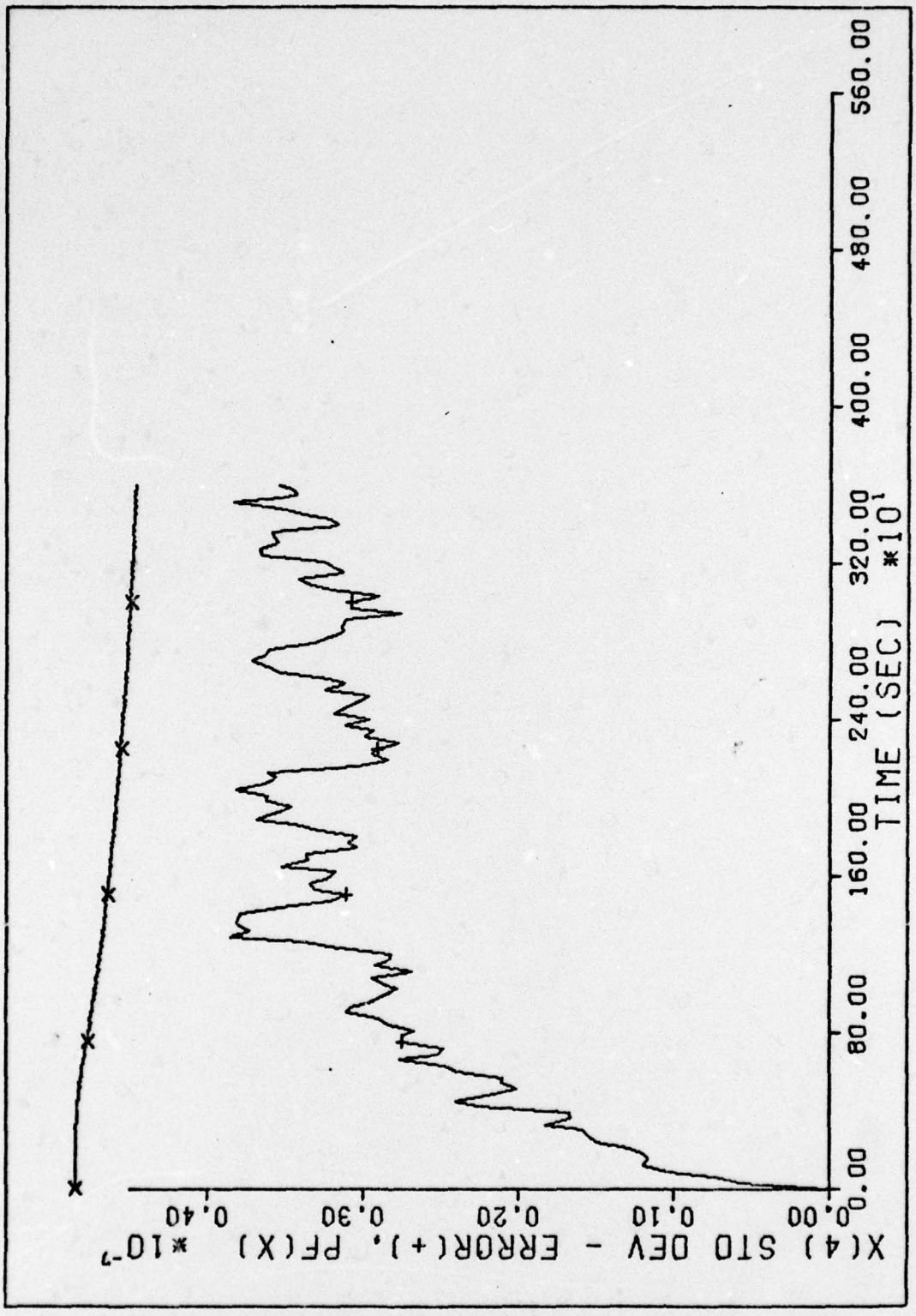


FIG 27. LINEAR FILTER EXAMPLE (10 RUNS)

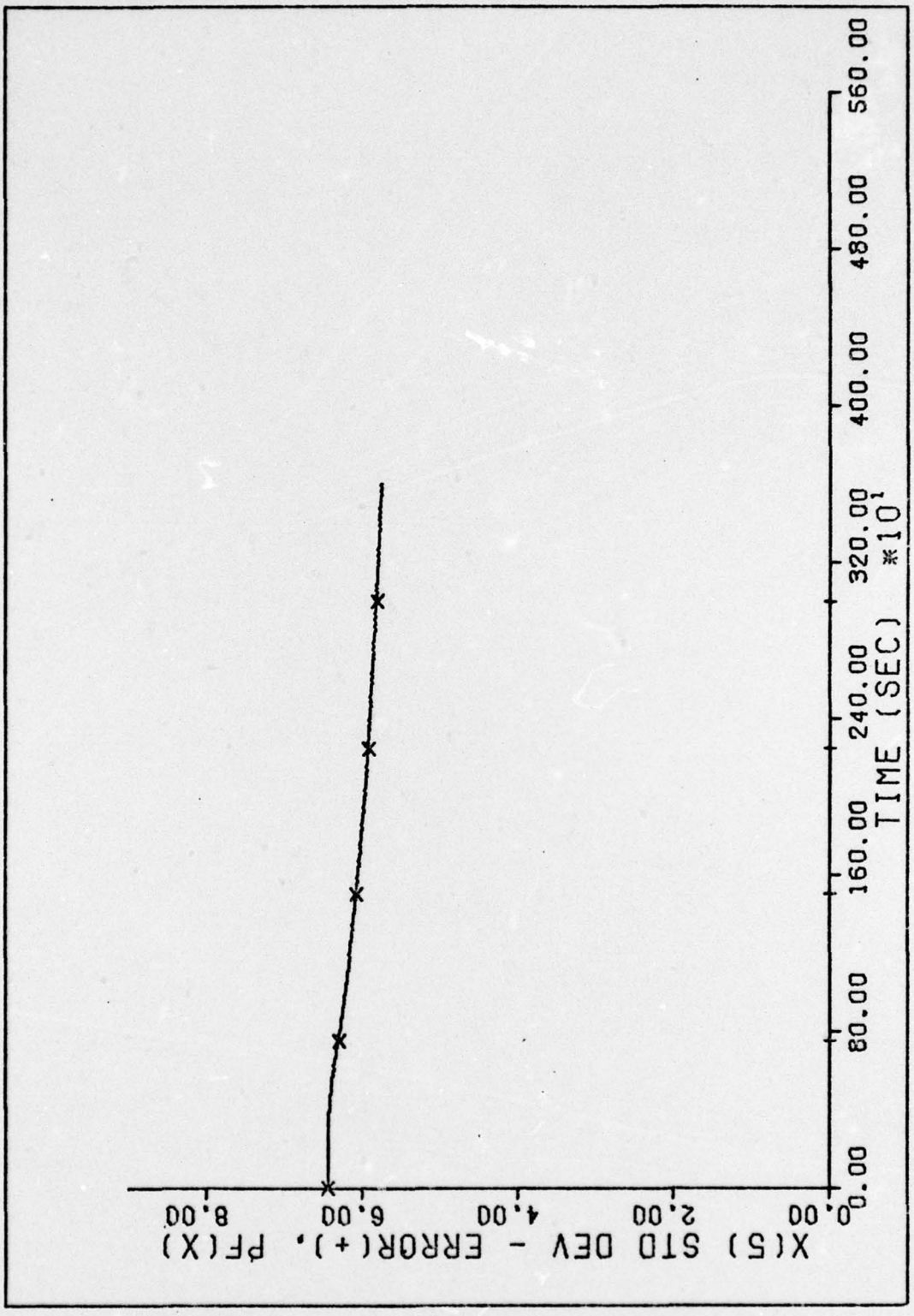


FIG 28. LINEAR FILTER EXAMPLE (10 RUNS)

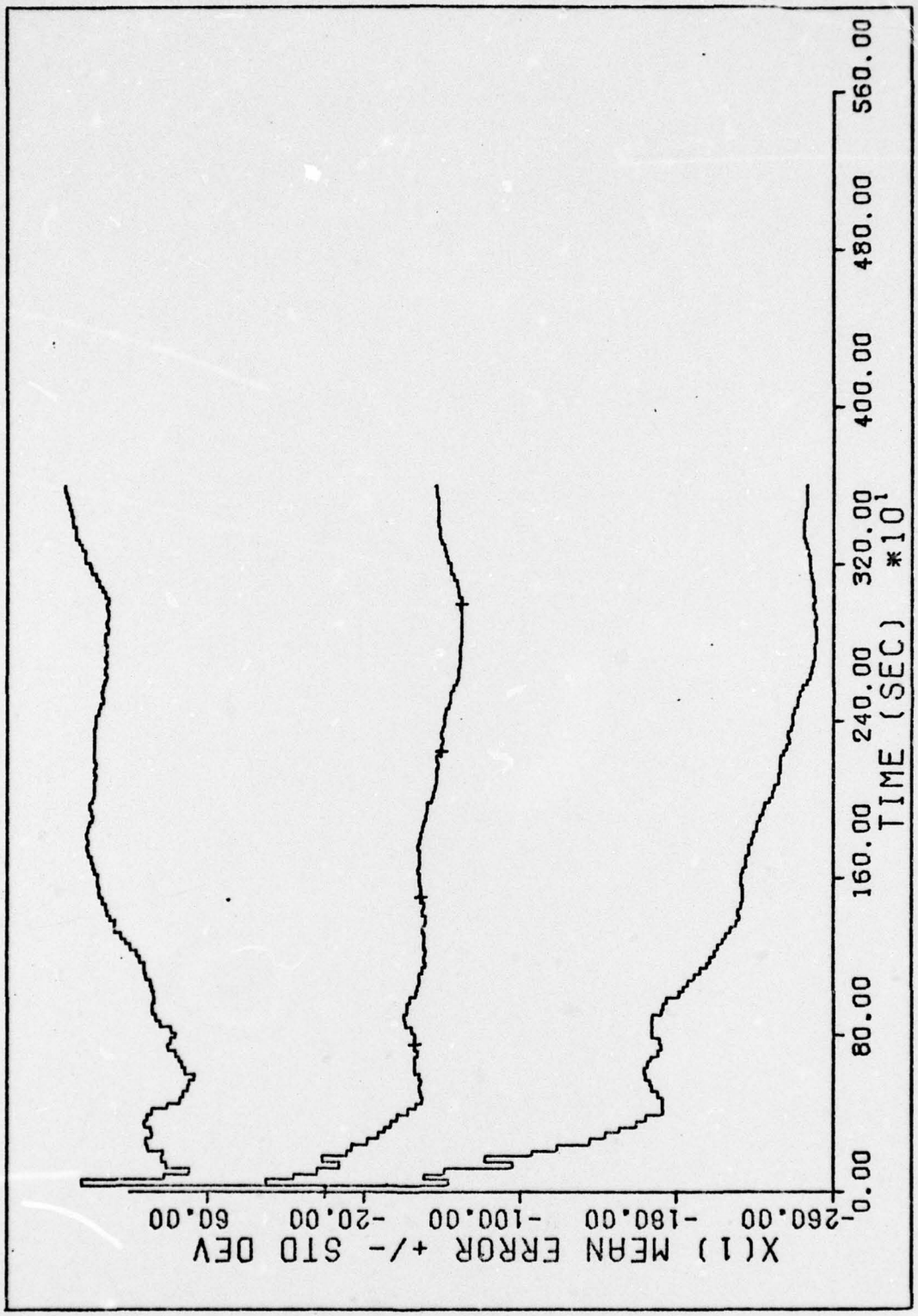


FIG 29. LINEAR FILTER EXAMPLE (10 RUNS)

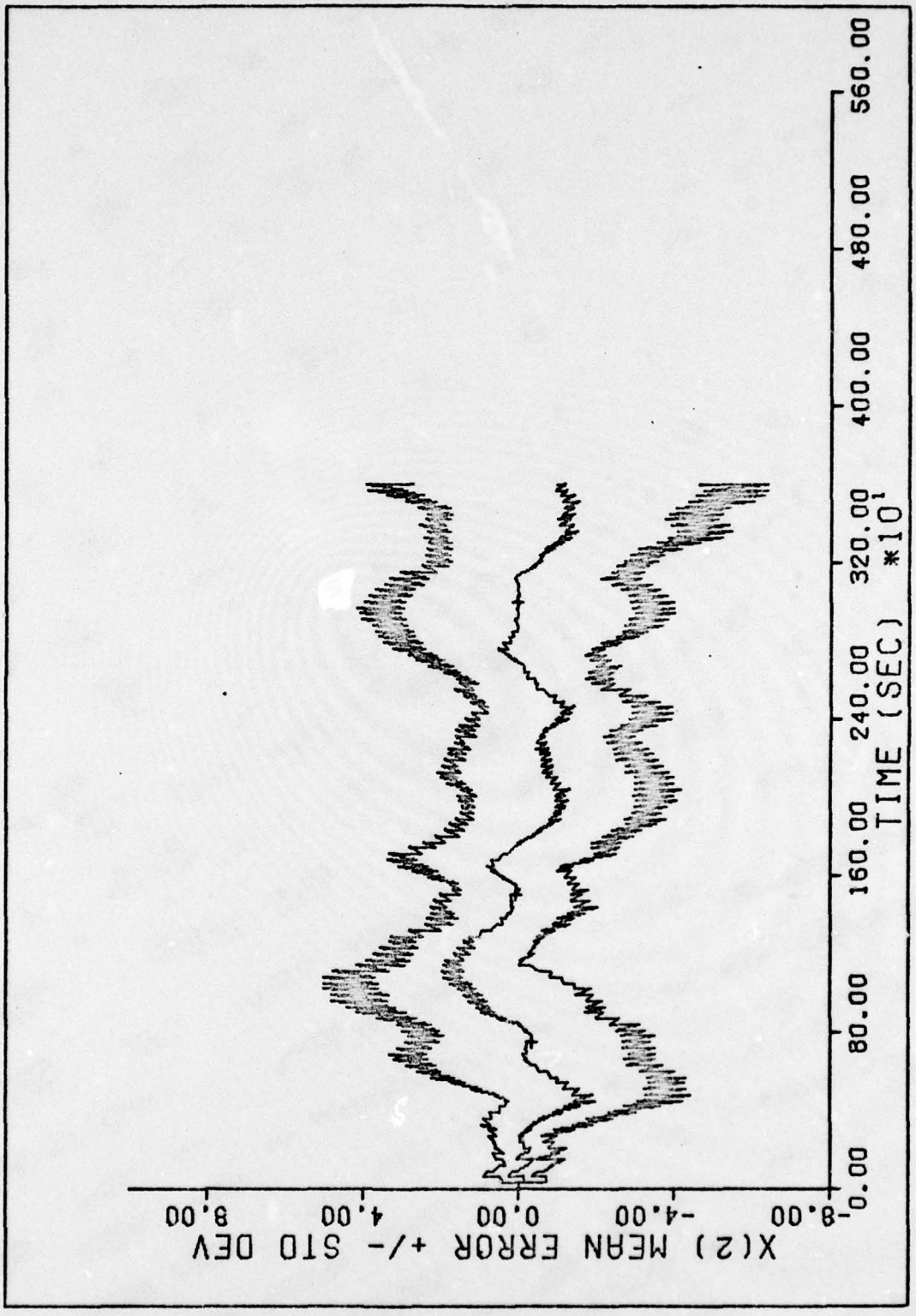


FIG 30. LINEAR FILTER EXAMPLE (10 RUNS)

FIG 31. LINEAR FILTER EXAMPLE (10 RUNS)

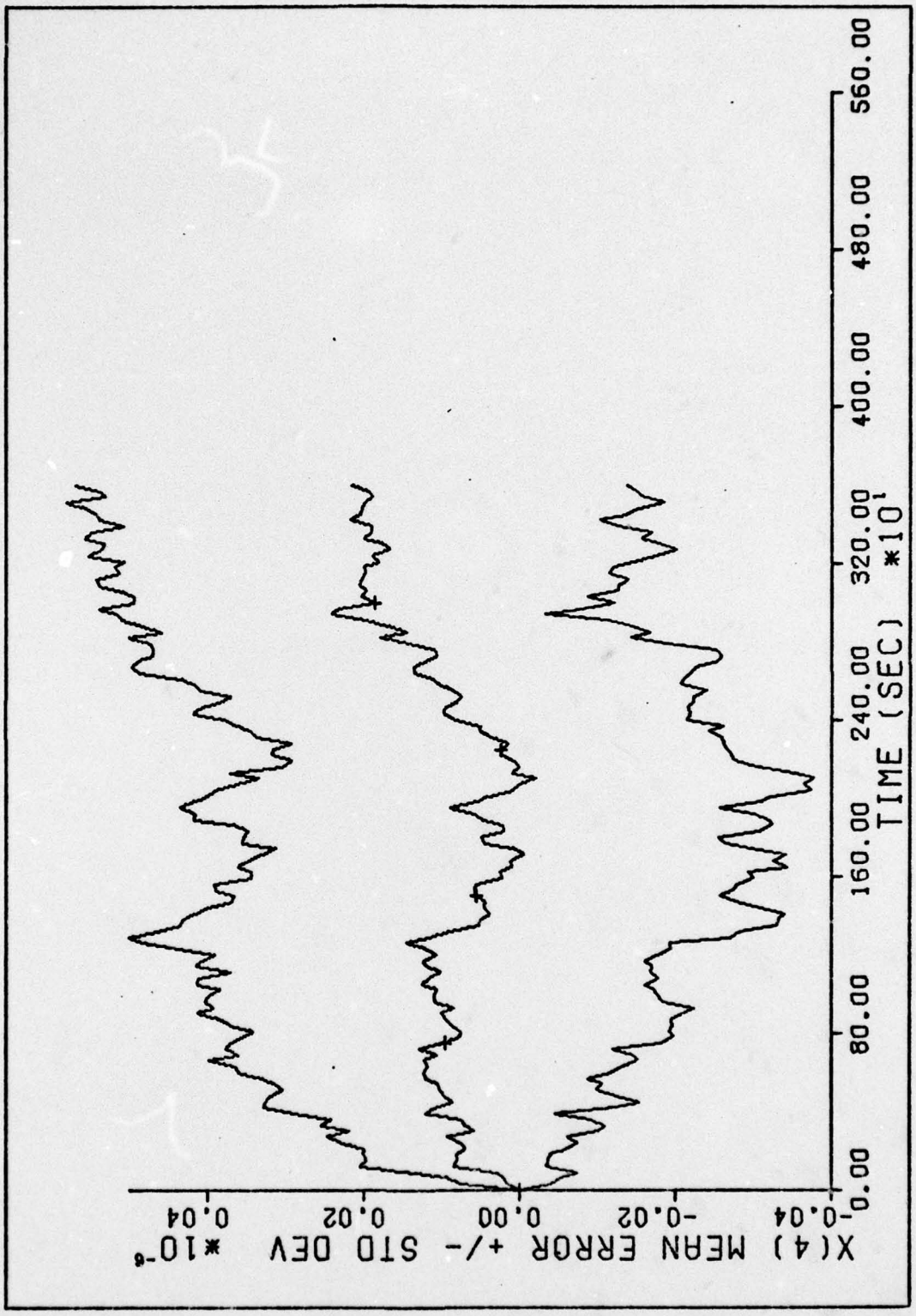


FIG 32. LINEAR FILTER EXAMPLE (10 RUNS)

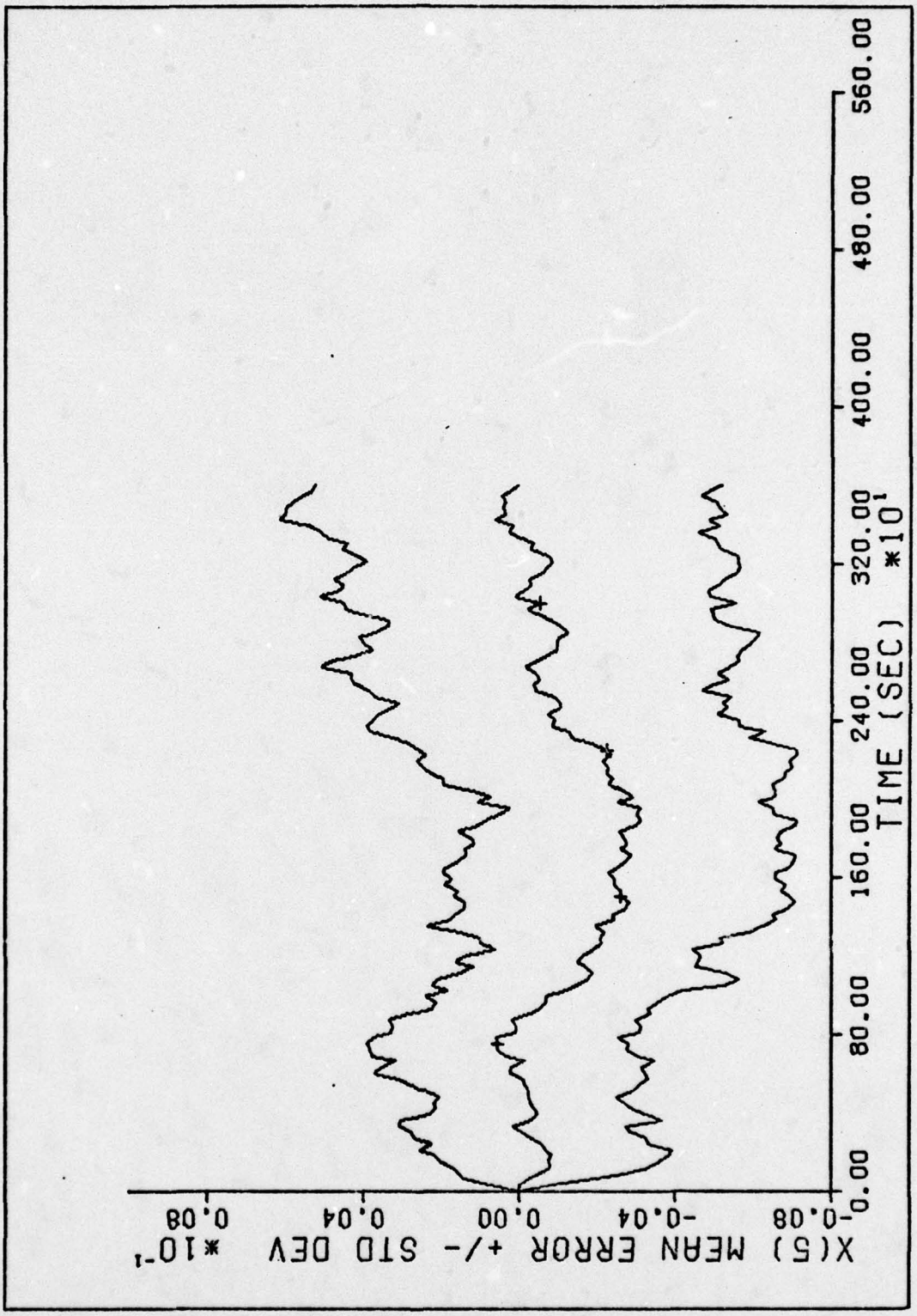


FIG 33. LINEAR FILTER EXAMPLE (10 RUNS)

Appendix B

Linearization of the Sub-Optimal Filter State and Measurement Equations

To develop the filter model, the first six states of the system model were deleted and the inertial acceleration of the satellite with respect to the tracker in the tracker frame, $(\underline{a}_{ts}^{t})$, was evaluated as

$$(\underline{a}_{ts})^{t} = \frac{-\mu_e[\underline{C}_i^{LS}(\underline{r}_t)^{i} + (\underline{r}_{ts})^{LS}]}{\left|(\underline{r}_s)^{i}\right|^3} - (\ddot{\underline{r}}_t)^{t} \tag{1}$$

Defining

$$(\underline{r}_t)^{i} \triangleq \begin{bmatrix} r_{tx}^{i} \\ r_{ty}^{i} \\ r_{tz}^{i} \end{bmatrix} \tag{2}$$

$$(\underline{r}_{ts})^{LS} \triangleq \begin{bmatrix} R \\ 0 \\ 0 \end{bmatrix} \tag{3}$$

$$(\underline{r}_t)^{t} \triangleq \begin{bmatrix} a_{tx}^{t} \\ a_{ty}^{t} \\ a_{tz}^{t} \end{bmatrix} \tag{4}$$

and with

$$(\underline{r}_s)^{i} = (\underline{r}_t)^{i} + \underline{C}_{LS}^{i}\,(\underline{r}_{ts})^{LS} \tag{5}$$

$$= (\underline{r}_t)^{i} + (\underline{C}_i^{LS})^{T}\,(\underline{r}_{ts})^{LS} \tag{6}$$

therefore,

$$(\underline{r}_s)^i = \begin{bmatrix} r^i_{tx} + C_{11} R \\ r^i_{ty} + C_{12} R \\ r^i_{tz} + C_{13} R \end{bmatrix} \tag{7}$$

where C_{ij} are elements of $\underline{C}_i^{LS}$ and

$$\left|(\underline{r}_s)^i\right|^3 = \{[r^i_{tx} + C_{11} R]^2 + [r^i_{ty} + C_{12} R]^2 + [r^i_{tz} + C_{13} R]^2\}^{3/2} \tag{8}$$

$$a^t_{tsx} = \frac{-\mu_e [r^i_{tx} + R]}{\left|(\underline{r}_s)^i\right|^3} - a^t_{tx} \tag{9}$$

$$a^t_{tsy} = \frac{-\mu_e r^i_{ty}}{\left|(\underline{r}_s)^i\right|^3} - a^t_{ty} \tag{10}$$

$$a^t_{tsz} = \frac{-\mu_e r^i_{tz}}{\left|(\underline{r}_s)^i\right|^3} - a^t_{tz} \tag{11}$$

It follows that

$$\frac{\partial}{\partial R} a^t_{tsx} = \frac{\partial}{\partial R}\left\{\left(\frac{-\mu_e}{\left|(\underline{r}_s)^i\right|^3}\right)[r^i_{tx} + R]\right\} \tag{12}$$

$$= \frac{-\mu_e}{\left|(\underline{r}_s)^i\right|^3} \frac{\partial}{\partial R}[r^i_{tx} + R] + [r^i_{tx} + R] \frac{\partial}{\partial R}\left(\frac{-\mu_e}{\left|(\underline{r}_s)^i\right|^3}\right) \tag{13}$$

$$= \frac{-\mu_e}{\left|(\underline{r}_s)^i\right|^3} - \mu_e[r^i_{tx} + R] \frac{\partial}{\partial R}\left(\frac{1}{\left|(\underline{r}_s)^i\right|^3}\right) \tag{14}$$

$$= \frac{-\mu_e}{\left|(\underline{r}_s)^i\right|^3} + \frac{1.5\mu_e[r^i_{tx} + R]}{\left|(\underline{r}_s)^i\right|^5} \frac{\partial}{\partial R}\left|(\underline{r}_s)^i\right| \tag{15}$$

so

$$\frac{\partial}{\partial R} a^i_{tsx} = \frac{-\mu_e}{\left|(\underline{r}_s)^i\right|^3} + \frac{1.5\mu_e [r^i_{tx} + R]\varphi}{\left|(\underline{r}_s)^i\right|^5} \tag{16}$$

$$\frac{\partial}{\partial R} a^i_{tsy} = \frac{1.5\mu_e \, r^i_{ty} \, \varphi}{\left|(\underline{r}_s)^i\right|^5} \tag{17}$$

$$\frac{\partial}{\partial R} a^i_{tsz} = \frac{1.5\mu_e \, r^i_{tz} \, \varphi}{\left|(\underline{r}_s)^i\right|^5} \tag{18}$$

where

$$\varphi = \frac{\partial}{\partial R} \left|(\underline{r}_s)^i\right| \tag{19}$$

$$= 2\left[C_{11}\left(r^i_{tx} + C_{11} R\right) + C_{12}\left(r^i_{ty} + C_{12} R\right) + C_{13}\left(r^i_{tz} + C_{13} R\right)\right] \tag{20}$$

Now, defining

$$f_1 \triangleq \dot{w}^{LS}_{iLSy} = \frac{-a^{LS}_{tsz}}{R} - \frac{2 V_r \, w^{LS}_{iLSy}}{R} + w^{LS}_{iLSx} \, w^{LS}_{iLSz} \tag{21}$$

$$f_2 \triangleq \dot{w}^{LS}_{iLSz} = \frac{a^{LS}_{tsy}}{R} - \frac{2 V_r \, w^{LS}_{iLSz}}{R} - w^{LS}_{iLSx} \, w^{LS}_{iLSy} \tag{22}$$

$$f_6 \triangleq \dot{V}_r = a^{LS}_{tsx} + R\left(w^{LS^2}_{iLSy} + w^{LS^2}_{iLSz}\right) \tag{23}$$

and therefore,

$$P_1 \triangleq \frac{\partial f_1}{\partial R} = \frac{1}{R^2}\left[R a^{LS}_{tsz} - \frac{\partial}{\partial R} a^{LS}_{tsz} + 2 V_r \, w^{LS}_{iLSy}\right] \tag{24}$$

$$P_2 \triangleq \frac{\partial f_2}{\partial R} = \frac{-1}{R^2}\left[R \, a^{LS}_{tsy} - \frac{\partial}{\partial R} a^{LS}_{tsy} - 2 V_r \, w^{LS}_{iLSz}\right] \tag{25}$$

$$P_3 \triangleq \frac{\partial f_6}{\partial R} = \frac{\partial}{\partial R} a^{LS}_{tsx} + w^{LS^2}_{iLSy} + w^{LS^2}_{iLSz} \tag{26}$$

then

$$\underline{F}\left[t \, ; \, \hat{\underline{x}}(t/t_i)\right] = \left.\frac{\partial \underline{f}\,[\underline{x}, t]}{\partial \underline{x}}\right|_{\underline{x}(t) \, = \, \hat{\underline{x}}(t/t_i)} \tag{27}$$

$$\underline{F} = \begin{bmatrix} \frac{-2\,V_r}{R} & w^t_{itx} & 0 & 0 & P_1 & \frac{-2\,w^{LS}_{iLSy}}{R} \\ -w^t_{itx} & \frac{-2\,V_r}{R} & 0 & 0 & P_2 & \frac{-2\,w^{LS}_{iLSz}}{R} \\ 0 & 1 & 0 & -w^t_{itx} & 0 & 0 \\ 1 & 0 & w^t_{itx} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 2\,Rw^{LS}_{iLSy} & 2\,Rw^{LS}_{iLSz} & 0 & 0 & P_3 & 0 \end{bmatrix} \tag{28}$$

and

$$\underline{H}[t;\ \underline{\hat{x}}(t\bar{i})] = \left.\frac{\partial \underline{h}[\underline{x},t]}{\partial \underline{x}}\right|_{\underline{x} = \underline{\hat{x}}(t\bar{i})} \tag{29}$$

$$\underline{H} = [\underline{I}\ (5x5)\ \vdots\ \underline{0}\ (5x1)] \tag{30}$$

Vita

Capt Kenneth L. Jackson, Jr. entered the United States Air Force in 1970. After attending technical school for the Minuteman weapon system at Chanute AFB, Illinois, he was stationed at Grand Forks AFB, North Dakota, as a Minuteman Combat Targeting Team Chief. In this capacity, he was responsible for aligning the missile, loading the missile computer memory, and bringing the Minuteman missile to launch readiness. Capt Jackson was assigned, in October 1971, to the Deputy Commander for Maintenance (DCM) engineering staff where he was responsible for the identification and resolution of Minuteman weapon system problems beyond the capability of the normal maintenance organization and procedures. In October 1974, he was assigned as a project officer to the Ground Electronics System Division of the Minuteman System Project Office (SPO), Space and Missiles System Organization (SAMSO) at Norton AFB, California. While at the Minuteman SPO, he was responsible for technical management of the integration of the Minuteman ground electronics command, communications, control, and security systems. Capt Jackson then entered the Air Force Institute of Technology in June 1976 to pursue a Master of Science degree in Electrical Engineering.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: AFIT/GGC/EE/77-6 (14)	2. GOVT ACCESSION NO. (9) Master's thesis
4. TITLE *(and Subtitle)*: A Generalized Monte Carlo Analysis Program for Kalman Filter Design with Application to an Aircraft-to-Satellite Tracking Filter. (6)	5. TYPE OF REPORT & PERIOD COVERED: MS Thesis
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): (10) Kenneth L. Jackson, Jr [illegible]	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Air Force Institute of Technology (AFIT/EN) Wright-Patterson AFB, Ohio 45433	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS (12) 309p.
11. CONTROLLING OFFICE NAME AND ADDRESS: Air Force Avionics Laboratory/RWI Wright-Patterson AFB, Ohio 45433 (11)	12. REPORT DATE: Dec[illegible]77 13. NUMBER OF PAGES: 307
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*	15. SECURITY CLASS. *(of this report)*: Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; IAW AFR 190-17

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

Approved for public release; IAW AFR 190-17

JERRAL F. GUESS, Captain, USAF
Director of Information

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

A generalized Monte Carlo Analysis Program (MCAP) has been developed for linear or extended Kalman filter design. The computer program is similiar to the Air Force Avionics Laboratory developed General Covariance Analysis Program (GCAP) so users can easily transition from a covariance analysis to a Monte Carlo analysis of a (extended) Kalman filter. A detailed users' guide for MCAP is included in Appendix A. In addition, this study treats the high accuracy tracking of a satellite from an aircraft. The purpose of

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

012 225

study is to evaluate the feasibility of a reduced order system model for implementation in an extended Kalman filter formulation whose estimates would be used to aid the tracker. The six state model accomplishes tracker state estimation by exploiting the information already available in the tracking geometry, dominant modes of satellite dynamics, and the range measurement. Tracker state estimation is accomplished in the line-of-sight coordinate frame. A Monte Carlo analysis was performed, evaluating the filter against a 42-state truth model. With some proposed modifications, it was concluded that the six state filter is feasible and warrants further study.