8-1985

A Study of Adaptive Kalman Filtering for Transfer Alignment

Peter Hansen
University of Tennessee, Knoxville

To the Graduate Council:

I am submitting herewith a thesis written by Peter Hansen entitled "A Study of Adaptive Kalman Filtering for Transfer Alignment." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

J. C. Hung, Major Professor

We have read this thesis and recommend its acceptance:

Robert W. Rochelle, Robert Bodenheimer

Accepted for the Council:
Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a thesis written by Peter Hansen entitled "A Study of Adaptive Kalman Filtering for Transfer Alignment." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

James C. Hung

J. C. Hung, Major Professor

We have read this thesis and recommend its acceptance:

Robert W. Rochelle

[illegible]

Accepted for the Council:

[illegible]

Vice Provost
and Dean of The Graduate School

A STUDY OF ADAPTIVE KALMAN FILTERING FOR TRANSFER ALIGNMENT

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Peter Hansen

August 1985

DEDICATION

I dedicate this work to my wife, Carolyn, who has an utter disgust for computers. Despite this, she never failed to support me in the pursuit of this project, which forced me to spent a fair amount of time with my computer.

ACKNOWLEDGMENTS

I am indebted to the encouragement and guidance provided by my major professor, Dr. J. C. Hung. His professionalism and guidance were truly appreciated. I also wish to thank Dr. R. W. Rochelle and Dr. R. E. Bodenheimer who served on my committee. Their enthusiasm as instructors provided me with a great deal of incentive to complete this project. Any errors in this work are solely my responsibility.

ABSTRACT

Prior to launching an inertially navigated weapon from the wing of an aircraft, the Inertial Measurement Unit (IMU) of the weapon must be in agreement with the master IMU of the aircraft. In order to correct the IMU of the weapon, it is required that the angles of alignment error between the two units be known. A model for the alignment error can be developed. A Kalman filter can then be used to estimate the angles of alignment error. The modeling of alignment error is complicated by the flexible nature of the aircraft. Since the environment of the aircraft can change dramatically during the alignment process, the model becomes time varying. This further compounds the complexity of the overall model of alignment error.

A possible solution to the alignment problem for weapons attached to the wings of an aircraft with a flexible body is proposed. This solution centers around the use of an adaptive Kalman filter. The adaptive Kalman filter can concurrently identify the time varying dynamics of flexing and estimate the angles of alignment error. This capability might substantially simplify the alignment problem.

Three adaptive Kalman filtering algorithms were investigated. These algorithms differ only in the method by which they identify the parameters of the system. The relative performance of these algorithms was determined by a simulation. The simulation was based on a simplified dynamic system.

The simulation demonstrated that only one of the adaptive Kalman filters provided sufficient performance to be considered for use in the alignment problem. This adaptive Kalman filter identifies the parameters through a stochastic Newton algorithm. The use of this adaptive Kalman filter, along with an appropriately developed model, appear to provide a viable solution to the alignment of inertially guided missiles attached to the wings of an aircraft with a flexible body.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE PAGE

FIGURE PAGE

CHAPTER 1

INTRODUCTION

Transfer alignment is the process of determining the angles of alignment error between the coordinate frames of two Inertial Measurement Units (IMUs) located on the same body. This is a fundamental problem in the area of fire control. An example of an application deals with the alignment of inertially guided missiles attached to the wings of an aircraft with a flexible body. In order for the missile to reach a preselected target with a high degree of accuracy, the IMU in the missile must be in agreement with the IMU in the aircraft when the missile is launched. The IMU in the aircraft is referred to as the master IMU. It is very precise and generally located in the cockpit. An IMU in a missile is referred to as a slave IMU. Because of the expendable nature of the slave IMUs, they are lower in cost and precision than the master IMU. The process of correcting the slave IMU to that of the master IMU is known as transfer alignment. The application to be considered in this study is the airborne transfer alignment of inertially guided weapons.

Two types of angular alignment error need to be considered. First, the master and slave IMU can be assumed to have a static alignment error. Static alignment error is caused by the relatively large tolerance allowed in physically mounting the missile to the wing of the aircraft. It is unlikely that the slave and master IMU will ever have an identical orientation in space, even when the aircraft is stationary.

The other type of alignment error is dynamic. Dynamic alignment error is caused by the flexing of the aircraft. The flexing of the aircraft is due to the elasticity of the airframe, and is aggravated by the changing payload, flight maneuvers, and weather turbulence. The angles of static and dynamic alignment error must be identified. They can then be used to correct the slave IMU during the completion of the transfer alignment process.

A strategy to compute the alignment error is needed. Because of the static alignment error, it is not possible to merely reference the slave IMU to the body of the aircraft [13, p. 72]. Optical alignment is not practical because of the large distance between the master and slave IMUs [13, p. 72]. Another approach starts with developing a state space model for the static and dynamic alignment error. Assuming this model is properly formulated, a Kalman filter can be used to determine the minimum variance estimate of the alignment error between the IMUs. The transfer alignment is completed by correcting the slave IMU. This process of computing the transfer alignment is well known.

The Kalman filter approach to transfer alignment is not without difficulties. The use of a Kalman filter requires the model of the system dynamics and the noise statistics to be known. Unfortunately, accurate modeling of the flexure dynamics of an aircraft is difficult. Even if an accurate model of the flexure dynamics can be determined, some of the parameters of the model might not be time invariant. Time varying parameters in this model can be introduced by many factors. These factors include the changing payload of the aircraft and varying turbulence conditions. Another factor is the variability of prelaunch maneuvers, ranging from level flight to evasion. Time varying parameters are not considered in the derivation of the standard Kalman filter. The problem to be considered in this study is how to perform the transfer alignment when the flexure dynamics of the aircraft is time varying, and has a relatively high degree of uncertainty to begin with.

This study will explore a solution to the above problem. Recently the literature has discussed various adaptive or self-tuning Kalman filter algorithms. These algorithms are capable of recursively identifying the system dynamics and the noise statistics associated with the model. Generally, these algorithms assume the system dynamics is time invariant. An adaptive Kalman filter can be made to track slow time varying parameters with minor modifications. The goal of this study is to demonstrate that an adaptive Kalman filter, capable of identifying time varying parameters, presents an attractive solution to the transfer alignment problem associated with inertially guided missiles attached to the wings of an aircraft with a flexible body.

This study will begin by developing a state space model for the alignment error. Next, three variations of an adaptive Kalman filter capable of tracking time varying parameters will be presented. These variations differ only in the complexity of the parameter identification process. The first variation is the easiest to implement. Some modifications were made to this algorithm to improve its performance. The last algorithm uses a more sophisticated parameter identification algorithm than the first two. The performance of these algorithms will be compared by a simulation based on a simplified model. Potential problems with adaptive Kalman filters will also be considered.

CHAPTER 2

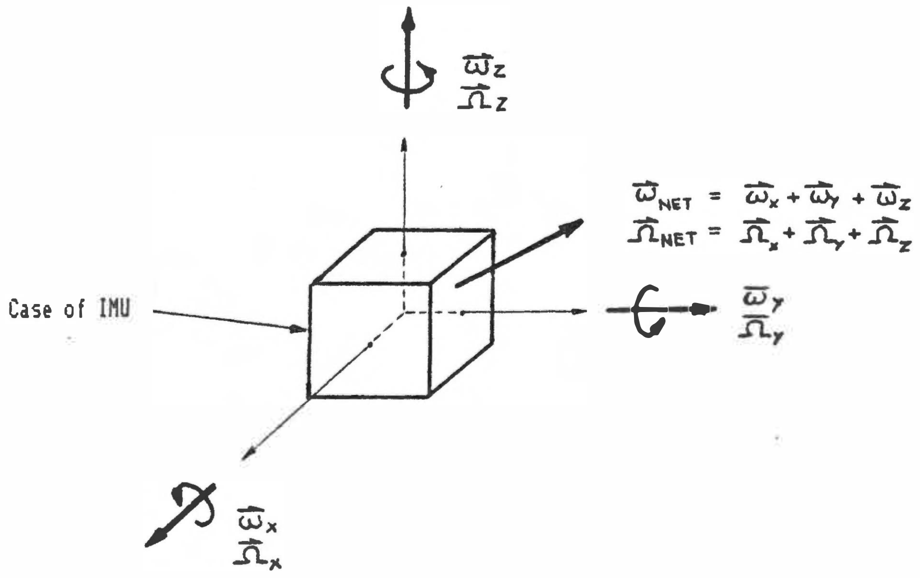
MODELING OF ALIGNMENT ERROR

This chapter deals with developing a state space model for the static and dynamic alignment error between the master and slave IMU. This model will be formulated in such a way that the alignment error can be estimated with either a standard or an adaptive Kalman filter. To apply a Kalman filter requires a state transition and a measurement equation to be identified. The noise covariances associated with these equations must be known. The static and dynamic alignment error will be discussed independently, and then combined to give the alignment error model for an aircraft with a flexible body. Problems with this model will be discussed to provide a rationale for the use of an adaptive Kalman filter.

Static Alignment Error

It would be unlikely for the master and slave IMU to have identical orientations in space, even if the aircraft is perfectly rigid. The result is static alignment error. The angles of static alignment error, ϕ_x, ϕ_y, and ϕ_z, are assumed not to be time varying.

The master and slave IMU are capable of resolving the angular velocity about each axis of a cartesian coordinate reference frame. These angular velocities are illustrated in Figure 1. The angular velocity about the x, y, and z coordinates, as sensed by the master IMU, are $\hat{\Omega}_x$, $\hat{\Omega}_y$, and $\hat{\Omega}_z$. The corresponding angular velocities sensed by the



$\vec{\omega}_z$
$\vec{\Omega}_z$
$\vec{\omega}_{NET} = \vec{\omega}_x + \vec{\omega}_y + \vec{\omega}_z$
$\vec{\Omega}_{NET} = \vec{\Omega}_x + \vec{\Omega}_y + \vec{\Omega}_z$
Case of IMU
$\vec{\omega}_y$
$\vec{\Omega}_y$
$\vec{\omega}_x$
$\vec{\Omega}_x$

Figure 1. Coordinate Reference Frame.

slave IMU are $\vec{\omega}_x$, $\vec{\omega}_y$, $\vec{\omega}_z$. Due to alignment error between the IMUs, the slave IMU in the missile will not sense the same angular velocities as the master IMU in the aircraft. This is illustrated in Figure 2. With a properly developed model, the alignment error with respect to the x, y and z axis is linearly related to the difference in the angular velocities sensed by the two IMUs.

The relationship between the alignment error around a coordinate axis, and the difference in angular velocity sensed by the master and slave IMU is illustrated in Figure 3. In this illustration, only the angle of alignment error around the z-axis, $\vec{\phi}_z$, is shown. This angle can be represented by a vector directed along the z-axis with a magnitude equal to the angle of alignment error around the z-axis. The difference in the angular velocity sensed by each IMU around its respective y-axis is $\overrightarrow{\Delta\Omega}_y$. If the actual alignment error around the z-axis is small, a simple expression relating this angle to the difference in sensed angular velocity can be found:

$$\overrightarrow{\Delta\Omega}_y = \vec{\Omega}_y - \vec{\omega}_y \sim \vec{\phi}_z \times \vec{\Omega}_x \,. \qquad (2\text{-}1)$$

Equation (2-1) can be generalized to consider the static angles of alignment error around each axis:

$$\overrightarrow{\Delta\Omega} = \vec{\Omega} - \vec{\omega} \sim \vec{\phi} \times \vec{\Omega} \,. \qquad (2\text{-}2)$$

Equation (2-2) can be represented by a determinant for the cross-product. The unit vectors in the x, y, and z directions are represented by $\hat{i}$, $\hat{j}$, and $\hat{k}$.

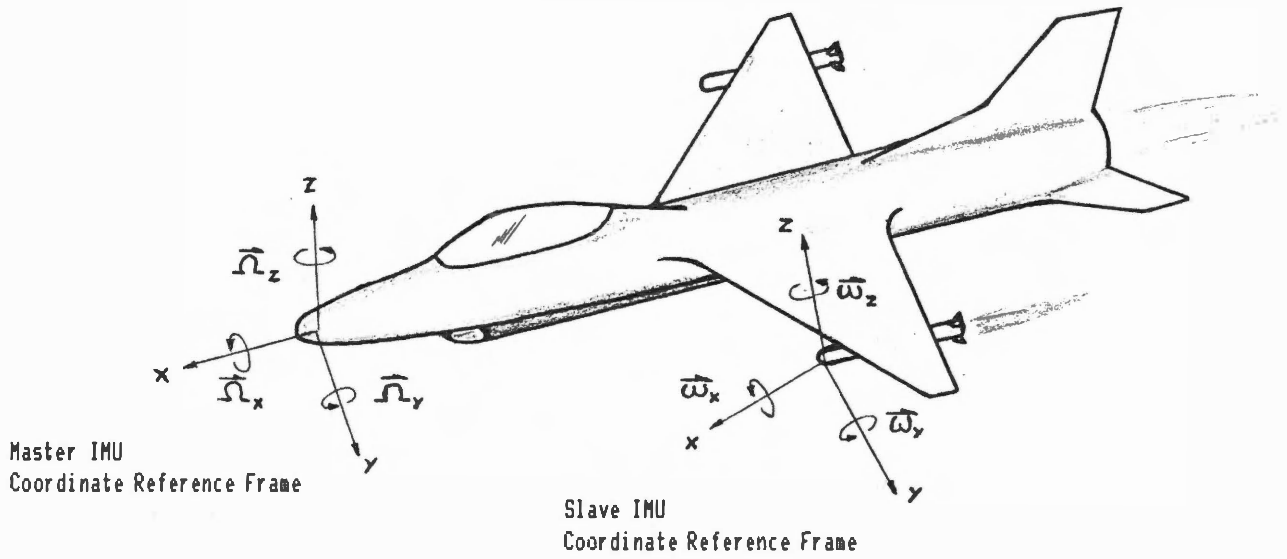


Figure 2. Angular Velocities as Sensed by the Master and Slave IMUs.

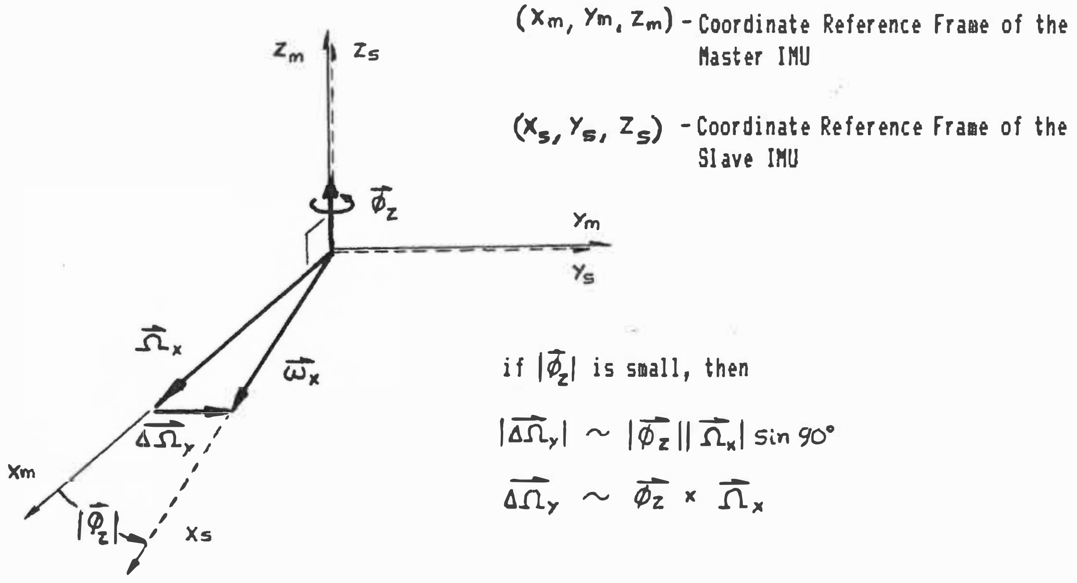

(X_m, Y_m, Z_m) - Coordinate Reference Frame of the Master IMU
(X_s, Y_s, Z_s) - Coordinate Reference Frame of the Slave IMU
Z_m
Z_s
$\vec{\phi}_z$
Y_m
Y_s
$\vec{\Omega}_x$
$\vec{\omega}_x$
$\overrightarrow{\Delta\Omega}_y$
X_m
$|\vec{\phi}_z|$
X_s
if $|\vec{\phi}_z|$ is small, then
$|\overrightarrow{\Delta\Omega}_y| \sim |\vec{\phi}_z||\vec{\Omega}_x| \sin 90°$
$\overrightarrow{\Delta\Omega}_y \sim \vec{\phi}_z \times \vec{\Omega}_x$

Figure 3. Cross-Product Relationship Between $\vec{\phi}$ and $\overrightarrow{\Delta\Omega}$.

$$\vec{\Delta\Omega} = \begin{vmatrix} \vec{\Delta\Omega}_x \\ \vec{\Delta\Omega}_y \\ \vec{\Delta\Omega}_z \end{vmatrix} = \begin{vmatrix} \vec{\Omega}_x - \vec{\omega}_x \\ \vec{\Omega}_y - \vec{\omega}_y \\ \vec{\Omega}_z - \vec{\omega}_z \end{vmatrix} = \begin{vmatrix} \hat{\imath} & \hat{\jmath} & \hat{k} \\ \phi_x & \phi_y & \phi_z \\ \Omega_x & \Omega_y & \Omega_z \end{vmatrix} \quad . \qquad (2\text{-}3)$$

Expanding equation (2-3) by components results in the following relationship between the sensed angular velocities and the alignment errors:

$$\begin{aligned} (\Omega_x - \omega_x)\hat{\imath} &= (\phi_y\Omega_z - \phi_z\Omega_y)\hat{\imath} \\ (\Omega_y - \omega_y)\hat{\jmath} &= (\phi_z\Omega_x - \phi_x\Omega_z)\hat{\jmath} \quad . \qquad (2\text{-}4) \\ (\Omega_z - \omega_z)\hat{k} &= (\phi_x\Omega_y - \phi_y\Omega_x)\hat{k} \end{aligned}$$

The vector notation is dropped and the symbol k is used to designate the discrete time interval. Equation (2-4) becomes:

$$\Delta\Omega = \begin{vmatrix} \Omega_x - \omega_x \\ \Omega_y - \omega_y \\ \Omega_z - \omega_z \end{vmatrix}_k = \begin{vmatrix} 0 & \Omega_z & -\Omega_y \\ -\Omega_z & 0 & \Omega_x \\ \Omega_y & -\Omega_x & 0 \end{vmatrix}_k \begin{vmatrix} \phi_x \\ \phi_y \\ \phi_z \end{vmatrix}_k \quad . \qquad (2\text{-}5)$$

The angles of alignment error are unknown. The angular velocities around each axis are measured quantities obtained from the master and slave IMU. In the process of acquiring these measurements some noise will be encountered. Measurement noise is modeled by adding a noise vector to equation (2-5):

$$\Delta\Omega = \begin{vmatrix} 0 & \Omega_z & -\Omega_y \\ -\Omega_z & 0 & \Omega_x \\ \Omega_y & -\Omega_x & 0 \end{vmatrix}_k \begin{vmatrix} \phi_x \\ \phi_y \\ \phi_z \end{vmatrix}_k + \begin{vmatrix} v_x \\ v_y \\ v_z \end{vmatrix}_k . \qquad (2\text{-}6)$$

Equation (2-6) is the measurement equation suitable for use with a Kalman filter in which only static alignment error is present.

The state transition equation for the static alignment error is an identity matrix. This is because the static alignment error does not change from one time interval to the next. However, this state transition matrix would cause a Kalman filter to become insensitive to new measurements after only a few iterations. To prevent this, a small amount of fictitious process noise will be added to the state transition equation. This state transition equation is:

$$\begin{vmatrix} \phi_x \\ \phi_y \\ \phi_z \end{vmatrix} = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} \phi_x \\ \phi_y \\ \phi_z \end{vmatrix} + \begin{vmatrix} w_x \\ w_y \\ w_z \end{vmatrix} . \qquad (2\text{-}7)$$

Two noise processes have been identified. Measurement noise has been included in equation (2-6), and process noise in equation (2-7). The statistics of this noise must be known. The process noise and measurement noise are both assumed to be gaussian with a zero mean. The covariance matrix of the process noise is assumed to be an identity matrix multiplied by a small constant, α. A larger constant will cause the filter to place more emphasis on recently acquired measurements.

The covariance of the process noise is, where Δt_k is the time interval between measurements and I is an identity matrix:

$$Q_w = E[w_k w_k^T] = \alpha \Delta t_k I \ . \tag{2-8}$$

The covariance of the measurement noise associated with equation (2-6) depends on the variance of the measurement error at the output of each IMU. The measurement error covariance for each axis will be the sum of the variances of the error associated with the master and slave IMU. Assuming the master and slave IMU have the same output error variance for each axis, σ_v^2, the measurement error covariance is:

$$R = E[v_k v_k^T] = 2\sigma_v^2 I \ . \tag{2-9}$$

Equations (2-6, 7, 8, 9) represent a model of the static alignment error suitable for use with a Kalman filter. This model will be combined with a model for the dynamic alignment error.

Dynamic Alignment Error

Dynamic alignment error implies that a bending or flexing of the airframe is taking place during the period in which the alignment process is carried out. There are many contributing factors to this flexing. A few include the weight distribution on the wings of the aircraft, weather turbulence, and the type of maneuvering taking place during the alignment process. In order to identify the dynamic alignment error, a suitable model must be generated for the flexure dynamics of the aircraft.

There are many ways to approach the development of a model for the flexure dynamics. One approach starts with a rigorous analysis of the structure of the specific aircraft for which the model is intended. This approach requires detailed information about that particular aircraft and is very complicated. A more general approach begins with recognizing that the flexure dynamics of the aircraft may be modeled as a random variable excited by a random forcing function. The latter approach will be taken in this study. The conclusions drawn in this study will apply even if a more sophisticated model of the flexure dynamics is used. This is because similar problems will be encountered when using either model of the flexure dynamics. These problems will be discussed later. Therefore, the latter model can be used without a loss of generality.

A random process used in the model of the flexure dynamics must be selected. A Markov process excited by white noise is an example of a suitable random process. Each axis of the body of the aircraft connecting the master and slave IMU can be approximately modeled as a damped mass-spring system. Such a system can be described by three second order differential equations, one equation for each axis of flexing. Since the Markov process can be associated with a differential equation excited by white noise, second order Markov processes are a logical choice for modeling the flexure dynamics. Better results might be obtained by using higher order Markov processes. Second order Markov processes should provide reasonable results.

The continuous time model of the flexure dynamics for the aircraft using a second order Markov process will now be developed. This model will then be linearized to give a discrete time equivalent of the continuous model. Lastly, the effects of static and dynamic alignment error will be combined to give the final model. The differential equation associated with a second order Markov process is:

$$\frac{d^2\theta}{dt^2} + 2\beta(t)\,\frac{d\theta}{dt} + \beta^2(t)\theta = q \,. \tag{2-10}$$

In this equation q is the random forcing function, and θ is the dynamic alignment error for one axis. If q is gaussian, then equation (2-10) is called a Gauss-Markov process. Each axis of flexing will be represented by an independent equation like that in (2-10). Equation (2-10) may be written as two first order differential equations. Letting θ' represent the time derivative of θ, equation (2-10) can be written as:

$$\begin{aligned} \dot{\theta} &= \theta' \\ \dot{\theta}' &= -\beta^2\theta - 2\beta\theta' + q \,. \end{aligned} \tag{2-11}$$

The dynamic flexing around each axis will be represented by an independent Markov process. This leads to the following set of state equations for the angles of dynamic alignment error:

$$\begin{vmatrix} \dot{\theta}_x \\ \dot{\theta}_y \\ \dot{\theta}_z \\ \dot{\theta}'_x \\ \dot{\theta}'_y \\ \dot{\theta}'_z \end{vmatrix} = \begin{vmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -\beta_x^2 & 0 & 0 & -2\beta_x & 0 & 0 \\ 0 & -\beta_y^2 & 0 & 0 & -2\beta_y & 0 \\ 0 & 0 & -\beta_z^2 & 0 & 0 & -2\beta_z \end{vmatrix} \begin{vmatrix} \theta_x \\ \theta_y \\ \theta_z \\ \theta'_x \\ \theta'_y \\ \theta'_z \end{vmatrix} + \begin{vmatrix} 0 \\ 0 \\ 0 \\ q_x \\ q_y \\ q_z \end{vmatrix} \quad . \qquad (2\text{-}12)$$

The statistics of the noise for a continuous system may be described by its mean and spectral characteristics. The noise exciting each Markov process in equation (2-12) has a zero mean. The spectral density matrix associated with the noise in equation (2-12) has only three none zero terms. This is because the noise processes in equation (2-12) are mutually independent. Assuming the statistics are time stationary, the three none zero elements of the spectral density matrix can be written as:

$$\begin{aligned} E[q_x(t_1)q_x(t_2)] &= Q_x(t_1)\delta(t_1 - t_2) = Q_x(\tau)\delta(\tau) \\ E[q_y(t_1)q_y(t_2)] &= Q_y(t_1)\delta(t_1 - t_2) = Q_y(\tau)\delta(\tau) \quad . \qquad (2\text{-}13) \\ E[q_z(t_1)q_z(t_2)] &= Q_z(t_1)\delta(t_1 - t_2) = Q_z(\tau)\delta(\tau) \end{aligned}$$

For convenience, let the three none zero elements of the spectral density matrix be selected as follows:

$$Q_x(\tau)\delta(\tau) = 4\beta_x^3\sigma_x^2\delta(\tau)$$

$$Q_y(\tau)\delta(\tau) = 4\beta_y^3\sigma_y^2\delta(\tau) \quad . \qquad (2\text{-}14)$$

$$Q_z(\tau)\delta(\tau) = 4\beta_z^3\sigma_z^2\delta(\tau)$$

With this selection, the power spectral density function, the autocorrelation function, and the correlation time of the flexing of the aircraft around each axis can be found from Table 1. Because the flexing around each axis has a zero mean, the variances are given by the autocorrelation functions when the time lag, τ, is zero. With the none zero elements of the spectral density matrix chosen as in equation (2-14), the variances of the bending are given by σ_x^2, σ_y^2, and σ_z^2. It is obvious that the variances in equation (2-14) are the actual variances of the flexing of the aircraft around each axis.

The correlation time is defined as the amount of time lag required for the autocorrelation function of the random variable to decrease by a value of 1/e. Figure 4 illustrates the autocorrelation functions of several Markov processes. For the second order Markov process the correlation time is given by:

$$t_{x,y,z} \sim \frac{2.146}{\beta_{x,y,z}} \quad . \qquad (2\text{-}15)$$

Identifying the process noise spectral density matrix requires knowledge of the variance and the correlation time of flexing around each axis of the aircraft. Similarly, the state transition matrix can

Table 1. Characteristics of Stationary Gauss-Markov Processes.

Order of Markov Process	Power Spectral Density, $\Phi_{xx}(\omega)$	Autocorrelation Function, $\varphi_{xx}(\tau)$	Correlation Time
1	$\dfrac{2\beta_1\sigma^2}{\omega^2 + \beta_1^2}$	$\sigma^2 e^{-\beta_1\lvert\tau\rvert}$	$\dfrac{1}{\beta_1}$
2	$\dfrac{4\beta_2^3\sigma^2}{(\omega^2 + \beta_2^2)^2}$	$\sigma^2 e^{-\beta_2\lvert\tau\rvert}\{1 + \beta_2\lvert\tau\rvert\}$	$\dfrac{2.146}{\beta}$
3	$\dfrac{16\beta_3^5\sigma^2}{3(\omega^2 + \beta_3^2)^3}$	$\sigma^2 e^{-\beta_3\lvert\tau\rvert}\{1 + \beta_3\lvert\tau\rvert + \frac{1}{3}\beta_3^2\lvert\tau\rvert^2\}$	$\dfrac{2.903}{\beta_3}$
n	$\dfrac{(2\beta_n)^{2n-1}[\Gamma(n)]^2}{(2n - 2)!(\omega^2 + \beta_n^2)^n}$	$\sigma^2 e^{-\beta_3\lvert\tau\rvert}\sum_{k=0}^{n-1}\dfrac{\Gamma(n)(2\beta_n\lvert\tau\rvert)^{n-k-1}}{(2n - 2)!k!\Gamma(n - k)}$	Solved arithmetically for each n
$n\to\infty$	$2\pi\sigma^2\delta(\omega)$	σ^2	∞

A. Gelb, Applied Optimal Estimation, The M.I.T. Press, Cambridge, Mass., 1974, p. 45.

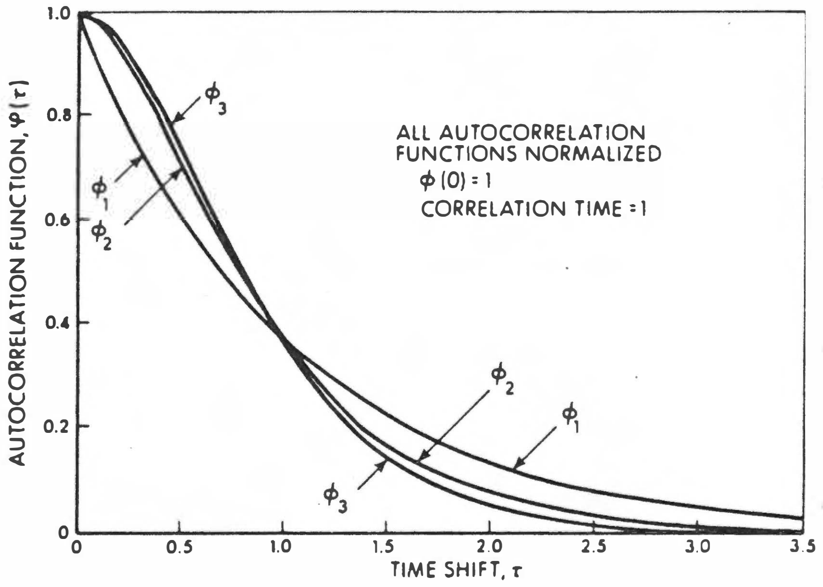


Figure 4. Autocorrelation Functions of Gauss-Markov Processes.

A. Gelb, Applied Optimal Estimation, The M.I.T. Press, Cambridge, Mass., 1974, p. 44.

be deduced from just the knowledge of the correlation time. It should be possible to identify these parameters based on the structural characteristics of the aircraft and the conditions in which the aircraft is being operated in. These considerations will be discussed later.

Next, the discrete time equivalent of the state transition equation (2-12), and the none zero terms in the process noise spectral density matrix (2-14) will be found. A first order discrete time approximation of the state transition matrix can be found by using [5, p. 77]:

$$\Phi_k = I + A\Delta t_k \ . \tag{2-16}$$

Using this approximation, the discrete time state transition equation may be written as:

$$\begin{vmatrix} \theta_x \\ \theta_y \\ \theta_z \\ \theta_x' \\ \theta_y' \\ \theta_z' \end{vmatrix}_{k+1} = \begin{vmatrix} 1 & 0 & 0 & \Delta t_k & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t_k & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t_k \\ -\beta_x^2\Delta t_k & 0 & 0 & 1-2\beta_x\Delta t_k & 0 & 0 \\ 0 & -\beta_y^2\Delta t_k & 0 & 0 & 1-2\beta_y\Delta t_k & 0 \\ 0 & 0 & -\beta_z^2\Delta t_k & 0 & 0 & 1-2\beta_z\Delta t_k \end{vmatrix} \begin{vmatrix} \theta_x \\ \theta_y \\ \theta_z \\ \theta_x' \\ \theta_y' \\ \theta_z' \end{vmatrix}_k + \begin{vmatrix} 0 \\ 0 \\ 0 \\ q_x \\ q_y \\ q_z \end{vmatrix}_k \ . \tag{2-17}$$

The discrete equivalent of the spectral density matrix is the covariance matrix [5, p. 75]. The none zero terms of the covariance matrix corresponding to equation (2-14) are:

The measurement equation (2-6) was derived from equation (2-2). These equations must be modified to include the effects of dynamic alignment error. Two considerations must be made. First, the net alignment error is the sum of the static and dynamic alignment error. Second, a correction for the relative motion of the slave IMU with respect to the master IMU must be made. While the aircraft is flexing, the slave IMU will be moving relative to the master IMU. At any one instant in time there may be no dynamic error even if the slave IMU senses a different angular velocity than the master IMU. This suggests that the differences in angular velocities sensed by the master and slave IMUs be corrected by the instantaneous dynamic angular velocity of the slave IMU. Combining these two effects, and letting $\vec{\theta}'$ represent the time derivative of $\vec{\theta}$, equation (2-2) becomes:

$$\Delta\vec{\Omega} = \vec{\Omega} - \vec{\omega} \sim (\vec{\phi} + \vec{\theta}) \times \vec{\Omega} - \vec{\theta}' \, . \qquad (2\text{-}20)$$

The measurement equation (2-6) modified to include the effects discussed above is:

$$\Delta\Omega_k = \begin{vmatrix} \Omega_x - \omega_x \\ \Omega_y - \omega_y \\ \Omega_z - \omega_z \end{vmatrix}_k = \begin{vmatrix} 0 & \Omega_z & -\Omega_y \\ -\Omega_z & 0 & \Omega_x \\ \Omega_y & -\Omega_x & 0 \end{vmatrix}_k \begin{vmatrix} \phi_x + \theta_x \\ \phi_y + \theta_y \\ \phi_z + \theta_z \end{vmatrix}_k - \begin{vmatrix} \theta_x' \\ \theta_y' \\ \theta_z' \end{vmatrix}_k + \begin{vmatrix} v_x \\ v_y \\ v_z \end{vmatrix}_k \, . \qquad (2\text{-}21)$$

This equation can be written in a more standard form as:

$$\begin{vmatrix} \Omega_x - \omega_x \\ \Omega_y - \omega_y \\ \Omega_z - \omega_z \end{vmatrix}_k = \begin{vmatrix} 0 & \Omega_z & -\Omega_y & 0 & \Omega_z & -\Omega_y & -1 & 0 & 0 \\ -\Omega_z & 0 & \Omega_x & -\Omega_z & 0 & \Omega_x & 0 & -1 & 0 \\ \Omega_y & -\Omega_x & 0 & \Omega_y & -\Omega_x & 0 & 0 & 0 & -1 \end{vmatrix}_k \begin{vmatrix} \phi_x \\ \phi_y \\ \phi_z \\ \theta_x \\ \theta_y \\ \theta_z \\ \phi_x' \\ \phi_y' \\ \phi_z' \end{vmatrix}_k + \begin{vmatrix} v_x \\ v_y \\ v_z \end{vmatrix}_k . \quad (2\text{-}22)$$

Equation (2-22) is the measurement equation for the model of static and dynamic alignment error. The measurement noise reflects the inaccuracy in obtaining angular velocity information from the master and slave IMU. The covariance of the measurement error was given in equation (2-9).

Combining the effects of the process noise associated with the static alignment error (2-8) and the dynamic alignment error (2-14) results in the following process noise covariance matrix:

$$Q = \begin{vmatrix} \alpha\Delta t_k & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \alpha\Delta t_k & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \alpha\Delta t_k & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4\beta_x^3\sigma_x^2\Delta t_k & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4\beta_y^3\sigma_y^2\Delta t_k & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4\beta_z^3\sigma_z^2\Delta t_k \end{vmatrix} . \quad (2\text{-}23)$$

At this point, the matrices of the model needed to design a Kalman filter to estimate the angles of static and dynamic alignment error have been determined. The state and measurement equations are defined in (2-19) and (2-22). The process and measurement noise covariance matrices are defined in equations (2-19) and (2-22). The process and measurement noise covariance matrices are defined in equations (2-23) and (2-9).

Uncertain Parameters in the Alignment Error Model and the Adaptive Kalman Filter

An accurate model of the static and dynamic alignment error must be known for the Kalman filter to provide good estimates of the actual alignment error. The model just developed relies on a priori knowledge of the variance of dynamic bending about each axis and the correlation time for each Markov process. These parameters depend on many factors. Some of these factors include the internal structural characteristics of the aircraft, such as the elasticity of the airframe and its weight distribution. External factors include weather turbulence and the type of maneuvering the aircraft is engaged in. The latter could range from level flight to evasive maneuvers. The large variability of conditions make the task of identifying these parameters, based on assumed conditions, very difficult. At best, the parameters can be identified for only a small subset of the possible conditions that the aircraft might encounter.

An algorithm capable of concurrently identifying the time varying parameters of the alignment error model and estimating the angles of alignment error would be very desirable. The adaptive Kalman filter may provide this solution to the transfer alignment problem.

The computational burden associated with the use of an adaptive Kalman filter needs to be considered. Because an adaptive Kalman filter must also identify the parameters of the model, more calculations are involved. To perform the transfer alignment in a certain period of time might require a more powerful computer. Any information about the value or the range of the parameters will enhance the ability to do an accurate alignment in a shorter period of time. The estimated parameters will converge to their true values faster when the range over which the parameters can vary is narrow. The best policy is to use as much information as is available and let the adaptive Kalman filter refine the parameter estimates to achieve the most accurate alignment possible. If the adaptive Kalman filter is used in this context, the transfer alignment process can be carried out with greater precision, despite the added computational burden.

The next chapter of this paper will examine the details of three adaptive Kalman filters. The system models used in these discussions will be general. The algorithms may be made specific to the transfer alignment problem by substituting the alignment error model discussed in this chapter.

CHAPTER 3

THE ADAPTIVE KALMAN FILTER

This chapter will present three related variations of an adaptive Kalman filter. These variations differ only in the complexity of how they identify the parameters of the system. The model of the system and the statistics of the noise used in the following discussion is general in nature. The specific model used to solve the transfer alignment problem was presented in the previous chapter. The following discussion can be applied to this model without a loss of generality.

A simulation of the three adaptive Kalman filters, yet to be presented, was done to assess their performance. The details of this simulation will be presented in the following chapter. However, references to the outcome of the simulation will be made in this chapter. The outcome of the simulation provided incentive to find algorithms with improved performance.

To begin, background information helpful in developing the adaptive Kalman filters will be presented. These concepts will include a brief discussion of convergence analysis applied to the extended Kalman filter. Parameter identification using a stochastic approximation will be discussed next. Following this, three versions of an adaptive Kalman filter will be developed. These algorithms will be modified to enable them to adapt to systems with slow time varying parameters. Finally, potential problems which might be encountered with an adaptive Kalman filter will be considered.

Convergence Analysis

The Kalman filter can provide the minimum variance estimate of the state of a system when the noise associated with the system is gaussian. When the gaussian assumption is removed, the Kalman filter gives the linear minimum variance estimate of the state. This estimate has the smallest unconditional error covariance among all linear estimates. In general, this estimate will be biased [6, p. 248]. A biased estimate has an expected value different than that of the quantity being estimated [5, p. 102]

When one or more of the system parameters is unknown they can be treated as additional states of the system. The original problem is no longer linear because of the occurrence of products between states and parameters. A solution to this problem is to linearize the equations about a nominal trajectory which is continually updated. Once this is done, the Kalman filter may be used to estimate the states and the parameters of the system. The resulting algorithm is known as the extended Kalman filter [6, p. 362]

The extended Kalman filter is one solution to the problem of adaptive filtering. A recent paper discusses the convergence properties of the extended Kalman filter [10]. This paper concludes that the convergence properties of the extended Kalman filter are not good because of a lack of coupling between the Kalman gain used in estimating the state of the system and the parameter identification process. Based on this, an improvement to the extended Kalman filter is proposed. The lesson to be learned is that an adaptive Kalman filter that has coupling between the parameter identification process and the Kalman gain will have better convergence properties than if this coupling was absent.

The adaptive Kalman filters to be considered in this paper will have coupling between the parameter identification process and the Kalman gain through the sensitivity of the Kalman gain. Thus, the convergence properties of these algorithms should be better than a standard extended Kalman filter. In fact, later, it will be pointed out that one of the adaptive Kalman filters to be presented is virtually identical to the extended Kalman filter which is modified to consider the sensitivity of the Kalman gain. This adaptive Kalman filter is developed from a different viewpoint than the extended Kalman filter. It will be seen that this adaptive Kalman filter provides excellent performance.

Stochastic Approximation

Stochastic approximation will be central to the parameter identification process in the adaptive Kalman filters considered in this study. Stochastic approximation may be defined as a technique for successive approximation of a quantity when the observations involve random errors due to the stochastic nature of the problem [15, p. 68]. Stochastic approximation is easy to implement because a priori knowledge of the noise statistics is not needed. It can be applied to any problem in which repeated observations are made [15, p. 68]

A typical problem which may be solved by stochastic approximation is as follows. Consider a function $Q(\theta,e_k)$. This is a function of an unknown parameter θ, and e_k, where e_k is a sequence of zero-mean random variables of the same unknown distribution. Observations of $Q(\theta,e_k)$ are available for any chosen θ. It is desired to solve the following equation for θ, where E denotes expectation:

$$E[Q(\theta,e_k)] = 0 \ . \tag{3-1}$$

When applied to the parameter identification process of an adaptive Kalman filter, $Q(\theta)$ will be identified as a scalar cost function based on the innovation sequence and involving one or more unknown parameters. The innovation sequence will be defined in the next section. The adaptive Kalman filter will be required to identify the unknown parameters. The error associated with the measurements used in obtaining the innovation sequence will be represented by e_k.

A stochastic approximation [15, p. 69] may be used to estimate the parameter, θ, that satisfies equation (3-1):

$$\hat{\theta}_k = \hat{\theta}_{k-1} - \gamma_k Q(\theta,e_k) \ . \tag{3-2}$$

The symbol γ_k denotes a gain sequence which must have the following three properties in order for the estimate of θ to converge to its true value [15, p. 69]:

$$\begin{gathered} \gamma_k \geq 0 \\ \sum_{k=1}^{\infty} \gamma_k = \infty \quad . \\ \sum_{k=1}^{\infty} \gamma_k^2 < \infty \end{gathered} \tag{3-3}$$

Any gain sequence that satisfies equation (3-3) may be used. The first condition given in equation (3-3) is required for consistency with equation (3-2). The second condition implies that any target parameter to be identified can be reached. The final condition assumes the gain converges to zero. This asymptotically eliminates the effects of noisy measurements [9, p. 59]. An example of a simple gain sequence which satisfies these conditions is:

$$\gamma_k = \frac{1}{k} \qquad k = 1, 2, 3, \ldots \,. \tag{3-4}$$

If the parameter to be identified is time varying, the gain sequence must be modified. The time varying parameter can be tracked by letting the gain sequence converge to a small positive constant instead of zero. This will require relaxing the third condition given in equation (3-3). The size of this constant will be determined by how fast the quantity is varying with time. An alternative technique is to reset the gain sequence to an intermediate value between some upper bound and zero, and let it converge again. The frequency at which this must be done will depend on how fast the parameter being identified is changing with time. In either case, there will be a trade-off between tracking ability and noise insensitivity [9, p. 59]. Reference to this discussion will be made later when the adaptive Kalman filter is modified to track slow time varying parameters.

Preliminary Equations

This section will present some preliminary equations useful in developing the adaptive Kalman filters. These will include the model of the system and a set of equations for a standard Kalman filter. Initial conditions for the Kalman filter will also be discussed.

A general system model is used for convenience in the following presentation of the adaptive Kalman filter. The discussion in the remainder of this study applies equally to the transfer alignment model presented in Chapter 1. Using a general model allows all features of the adaptive Kalman filter to be presented, and not just those used in the transfer alignment problem.

In the following discrete time model, the time interval is designated by k. In general, the matrices of the model are functions of an unknown parameter, θ. The dimensions of vectors and matrices will be given in parentheses where useful. The model is:

$$\begin{aligned} X_{k+1} &= A(\theta)X_k + B(\theta)U_k + w_k \\ Y_k &= C(\theta)X_k + v_k \qquad k = 0, 1, 2, \ldots \end{aligned} \tag{3-5}$$

where

$A(\theta)$ = state transition matrix (n x n),

$B(\theta)$ = input matrix (n x c),

$C(\theta)$ = output matrix (r x n),

X_k = current state vector (n x 1),

U_k = current input vector (c x 1),

Y_k = current output vector (r x 1),

w_k = process noise vector (n x 1), and

v_k = measurement noise vector (r x 1).

The noise sequences are assumed to be uncorrelated, zero mean gaussian random vectors. They have the following covariance matrices which may also contain unknown parameters:

$$\begin{aligned} E[w_k w_k^T] &= Q(\theta) \qquad (n \times n) \\ E[v_k v_k^T] &= R(\theta) \qquad (r \times r) \end{aligned} \qquad (3\text{-}6)$$

Next, the equations for a standard Kalman filter will be presented for the system given in equation (3-5) using the noise statistics in equation (3-6). In this set of equations the unknown parameters are replaced by there most current estimate. How these estimates are generated will be discussed in subsequent sections. The Kalman filter equations are:

(a) State estimate propagation:

$$\hat{X}_{k+1/k} = A(\theta)\hat{X}_{k/k} + B(\theta)U_k \ . \qquad (3\text{-}7)$$

(b) Error covariance propagation: (n x n)

$$P_{k+1/k} = A(\theta)P_{k/k}A(\theta)^T + Q(\theta) \ . \qquad (3\text{-}8)$$

(c) Innovation sequence: (r x 1)

$$\varepsilon_{k+1} = Y_{k+1} - C(\theta)\hat{X}_{k+1/k} \ . \qquad (3\text{-}9)$$

(d) Kalman gain: (n x r)

$$K_{k+1} = P_{k+1/k}C(\theta)^T[C(\theta)P_{k+1/k}C(\theta)^T + R(\theta)]^{-1} \quad . \tag{3-10}$$

(e) State estimate update:

$$\hat{X}_{k+1/k+1} = \hat{X}_{k+1/k} + K_{k+1}\varepsilon_{k+1} \quad . \tag{3-11}$$

(f) Error covariance update:

$$P_{k+1/k+1} = [I - K_{k+1}C(\theta)]P_{k+1/k}[I - K_{k+1}C(\theta)]^T + K_{k+1}R(\theta)K^T_{k+1}. \tag{3-12}$$

To initialize the Kalman filter, an initial state estimate and an error covariance associated with this estimate must be determined. These two quantities are given by:

$$\begin{aligned} E[X_0] &= \hat{X}_{0/0} \\ E[(X_0 - \hat{X}_{0/0})(X_0 - \hat{X}_{0/0})] &= P_{0/0} \end{aligned} \quad . \tag{3-13}$$

The adaptive Kalman filter used in conjunction with the alignment error model is only required to identify parameters in the state transition and process noise covariance matrices. The adaptive Kalman filter is capable of identifying parameters in the other system matrices as well. However, some researchers have found that attempting to identify parameters in both noise covariance matrices simultaneously is not well behaved [11, p. 122]. This is not an obstacle to the transfer alignment problem. The measurement noise covariance can be determined directly from the variance of the measurements from the IMUs. This variance is known and not expected to be time varying.

The Adaptive Kalman Filter Based on the Stochastic Gradient Algorithm

The first adaptive Kalman filter to be presented in this study estimates the system parameters using a stochastic gradient algorithm. This version of the adaptive Kalman filter is proposed in reference [3]. It is potentially attractive because it is relatively easy to implement. This derivation will begin with developing the parameter identification algorithm. The supporting equations will then be derived. The complete algorithm will be summarized on a flow chart.

The innovation sequence of the Kalman filter can be viewed as the error in predicting the output of the system. This variation of the adaptive Kalman filter will recursively adjust the parameter estimates to minimize the mean square prediction error. Thus, the mean square prediction error can be identified as a scalar cost function to be minimized:

$$J(\theta) = \frac{1}{2} E[\varepsilon_k^T \varepsilon_k] \quad . \tag{3-14}$$

Let the unknown parameters be arranged in a vector, denoted by $\underline{\theta}$. This vector will have a dimension of (p x 1) where p represents the number of parameters to be identified. A particular parameter in this vector will be represented by θ_j. The mean square prediction error is minimized by setting its gradient to zero:

$$\frac{\partial}{\partial \theta_j} J(\theta) = E[\frac{\partial}{\partial \theta_j} \varepsilon_k)^T \varepsilon_k] = 0 \ . \tag{3-15}$$

Equation (3-15) is similar to (3-1). Stochastic approximation can be applied to equation (3-15) to estimate the jth parameter. The result is:

$$(\hat{\theta}_k)_j = (\hat{\theta}_{k-1})_j - \gamma_k[(\frac{\partial}{\partial \theta_j} \varepsilon_k)^T \varepsilon_k] \ . \tag{3-16}$$

It is worthwhile to consider a geometric interpretation of equation (3-16). The cost function in equation (3-14) is a scalar field. The gradient of a scalar field is a vector that points in the direction of maximum increase. Adjusting the parameter estimates in the negative gradient direction will minimize the cost function. Therefore, this method to identify the parameters will be referred to as the stochastic gradient algorithm.

The supporting equations will be found next. They will be found by evaluating partial derivatives of equation (3-7) through (3-12) with respect to each of the unknown parameters. From this point on, references to partial derivatives are with respect to a specific parameter to be identified. To begin, the partial derivative of the innovation sequence is needed in equation (3-16). It is found by differentiating equation (3-9):

$$\frac{\partial}{\partial \theta_j} \varepsilon_{k+1} = -C(\theta)(\frac{\partial}{\partial \theta_j} \hat{X}_{k+1/k}) - (\frac{\partial}{\partial \theta_j} C(\theta))\hat{X}_{k+1/k} \ . \tag{3-17}$$

The partial derivative of $C(\theta)$ will be a known quantity. The partial derivative of $\hat{X}_{k+1/k}$ can be found by first differentiating equation (3-7):

$$\frac{\partial}{\partial\theta_j}\hat{X}_{k+1/k} = (\frac{\partial}{\partial\theta_j}A(\theta))\hat{X}_{k/k} + A(\theta)(\frac{\partial}{\partial\theta_j}\hat{X}_{k/k}) + (\frac{\partial}{\partial\theta_j}B(\theta))U_k \; . \tag{3-18}$$

Equation (3-18) involves partial derivatives of $\hat{X}$ at different time intervals. A more convenient expression for equation (3-18) can be found by first determining the partial derivative of $\hat{X}_{k/k}$. To do this, the time interval associated with equation (3-11) is first reduced by one unit:

$$\hat{X}_{k/k} = \hat{X}_{k/k-1} + K_k\varepsilon_k \; . \tag{3-19}$$

Changing the time interval of any of the equations does not affect their form. The partial derivative of equation (3-19) is taken next:

$$\frac{\partial}{\partial\theta_j}\hat{X}_{k/k} = (\frac{\partial}{\partial\theta_j}\hat{X}_{k/k-1}) + (\frac{\partial}{\partial\theta_j}K_k)\varepsilon_k + K_k(\frac{\partial}{\partial\theta_j}\varepsilon_k) \; . \tag{3-20}$$

Equation (3-20) involves the gradient of the innovation. This term may be eliminated by reducing the time interval of equation (3-17) by one and substituting the result into equation (3-20):

$$\frac{\partial}{\partial\theta_j}\hat{X}_{k/k} = [I - K_kC(\theta)](\frac{\partial}{\partial\theta_j}\hat{X}_{k/k-1}) + (\frac{\partial}{\partial\theta_j}K_k)\varepsilon_k - K_k(\frac{\partial}{\partial\theta_j}C(\theta))\hat{X}_{k/k-1} \; . \tag{3-21}$$

Finally, equations (3-19) and (3-21) are substituted into equation (3-18). After performing some algebra the result is:

$$\begin{aligned}\frac{\partial}{\partial\theta_j}\hat{X}_{k+1/k} &= A(\theta)[I - K_k C(\theta)]\left(\frac{\partial}{\partial\theta_j}\hat{X}_{k/k-1}\right)\\ &+ [A(\theta)(\frac{\partial}{\partial\theta_j}K_k) + (\frac{\partial}{\partial\theta_j}A(\theta))K_k]\varepsilon_k\\ &+ [(\frac{\partial}{\partial\theta_j}A(\theta)) - A(\theta)K_k(\frac{\partial}{\partial\theta_j}C(\theta))]\hat{X}_{k/k-1}\\ &+ (\frac{\partial}{\partial\theta_j}B(\theta))U_k \ . \qquad (3\text{-}22)\end{aligned}$$

The partial derivatives of the A(θ), B(θ), and C(θ) matrices are known from the nature of the problem. The partial derivative, or gradient, of the Kalman gain matrix is also needed. This is sometimes referred to as the sensitivity of the Kalman gain [10, p. 43]. This sensitivity appears to be a key factor in enhancing the convergence properties of an adaptive Kalman filter.

The gradient of the Kalman gain is found by differentiating equation (3-10), after reducing the time interval by one unit. The result is:

$$\begin{aligned}\frac{\partial}{\partial\theta_j}K_k &= (\frac{\partial}{\partial\theta_j}P_{k/k-1})C(\theta)^T[C(\theta)P_{k/k-1}C(\theta)^T + R(\theta)]^{-1}\\ &+ P_{k/k-1}(\frac{\partial}{\partial\theta_j}C(\theta))^T[C(\theta)P_{k/k-1}C(\theta)^T + R(\theta)]^{-1}\\ &- P_{k/k-1}C(\theta)^T[C(\theta)P_{k/k-1}C(\theta)^T + R(\theta)]^{-1}[C(\theta)(\frac{\partial}{\partial\theta_j}P_{k/k-1})C(\theta)^T\end{aligned}$$

$$+ (\frac{\partial}{\partial\theta_j} C(\theta))P_{k/k-1}C(\theta)^T + C(\theta)P_{k/k-1}(\frac{\partial}{\partial\theta_j} C(\theta))^T + (\frac{\partial}{\partial\theta_j} R(\theta))]\cdot$$

$$[C(\theta)(\frac{\partial}{\partial\theta_j} P_{k/k-1})C(\theta)^T + R(\theta)]^{-1} . \qquad (3\text{-}23)$$

To evaluate the gradient of the Kalman gain requires the gradient of $P_{k/k-1}$. This is found by reducing the time interval of equation (3-8) by one unit and differentiating the result:

$$\frac{\partial}{\partial\theta_j} P_{k/k-1} = (\frac{\partial}{\partial\theta_j} A(\theta))P_{k-1/k-1}A(\theta)^T + A(\theta)(\frac{\partial}{\partial\theta_j} P_{k-1/k-1})A(\theta)^T$$

$$+ A(\theta)P_{k-1/k-1} (\frac{\partial}{\partial\theta_j} A(\theta))^T + (\frac{\partial}{\partial\theta_j} Q(\theta)) . \qquad (3\text{-}24)$$

Finally, the gradient of $P_{k-1/k-1}$ is needed in equation (3-24). It is found by adjusting the time interval of equation (3-12) appropriately and then differentiating the result:

$$\frac{\partial}{\partial\theta_j} P_{k/k} = -[(\frac{\partial}{\partial\theta_j} K_k)C(\theta) + K_k(\frac{\partial}{\partial\theta_j} C(\theta))]P_{k/k-1}[I - K_kC(\theta)]^T$$

$$- [I - K_kC(\theta)]P_{k/k-1}[(\frac{\partial}{\partial\theta_j} K_k)C(\theta) + K_k(\frac{\partial}{\partial\theta_j} C(\theta))]^T$$

$$+ [I - K_kC(\theta)](\frac{\partial}{\partial\theta_j} P_{k/k-1})[I - K_kC(\theta)]^T$$

$$+ (\frac{\partial}{\partial\theta_j} K_k)R(\theta)K_k^T + K_k(\frac{\partial}{\partial\theta_j} R(\theta))K_k^T + K_kR(\theta)(\frac{\partial}{\partial\theta_j} K_k)^T .$$

$$(3\text{-}25)$$

Equations (3-17), (3-22), (3-24) and (3-25) support equation (3-16) in estimating the parameters. Special consideration needs to be made the first time the algorithm is executed to find the gradient of $\hat{X}_{1/0}$ and $P_{1/0}$. During the initialization of the algorithm, an approximation for the gradient of $\hat{X}_{1/0}$ is found from equation (3-18) by neglecting the derivative of $\hat{X}_{k/k}$. The result is:

$$\frac{\partial}{\partial\theta_j}\hat{X}_{1/0} = (\frac{\partial}{\partial\theta_j}A(\theta))\hat{X}_{0/0} + (\frac{\partial}{\partial\theta_j}B(\theta))U_0 \quad . \qquad (3\text{-}26)$$

Similarly, an initial expression for the gradient of $P_{1/0}$ is found by neglecting the derivative of $P_{k-1/k-1}$ in equation (3-24):

$$\frac{\partial}{\partial\theta_j}P_{1/0} = (\frac{\partial}{\partial\theta_j}A(\theta))P_{0/0}A(\theta)^T + A(\theta)P_{0/0}(\frac{\partial}{\partial\theta_j}A(\theta))^T + (\frac{\partial}{\partial\theta_j}Q(\theta)) \quad . \qquad (3\text{-}27)$$

The complete set of equations to realize this variation of an adaptive Kalman filter has now been derived. The parameter identification process is carried out by equation (3-16). A projection facility must be implemented to ensure the parameter estimates remain within a stable range. A simple projection facility is:

$$\underline{\hat{\theta}}_k \in [\underline{a},\underline{b}]$$

$$(\hat{\theta}_k)_j = \left\{ \begin{array}{lll} b_j & \text{if} & (\hat{\theta}_k)_j \geq b_j \\ (\hat{\theta}_k)_j & \text{if} & a_j < (\hat{\theta}_k)_j < b_j \\ a_j & \text{if} & (\hat{\theta}_k)_j \leq a_j \end{array} \right\} \quad . \qquad (3\text{-}28)$$

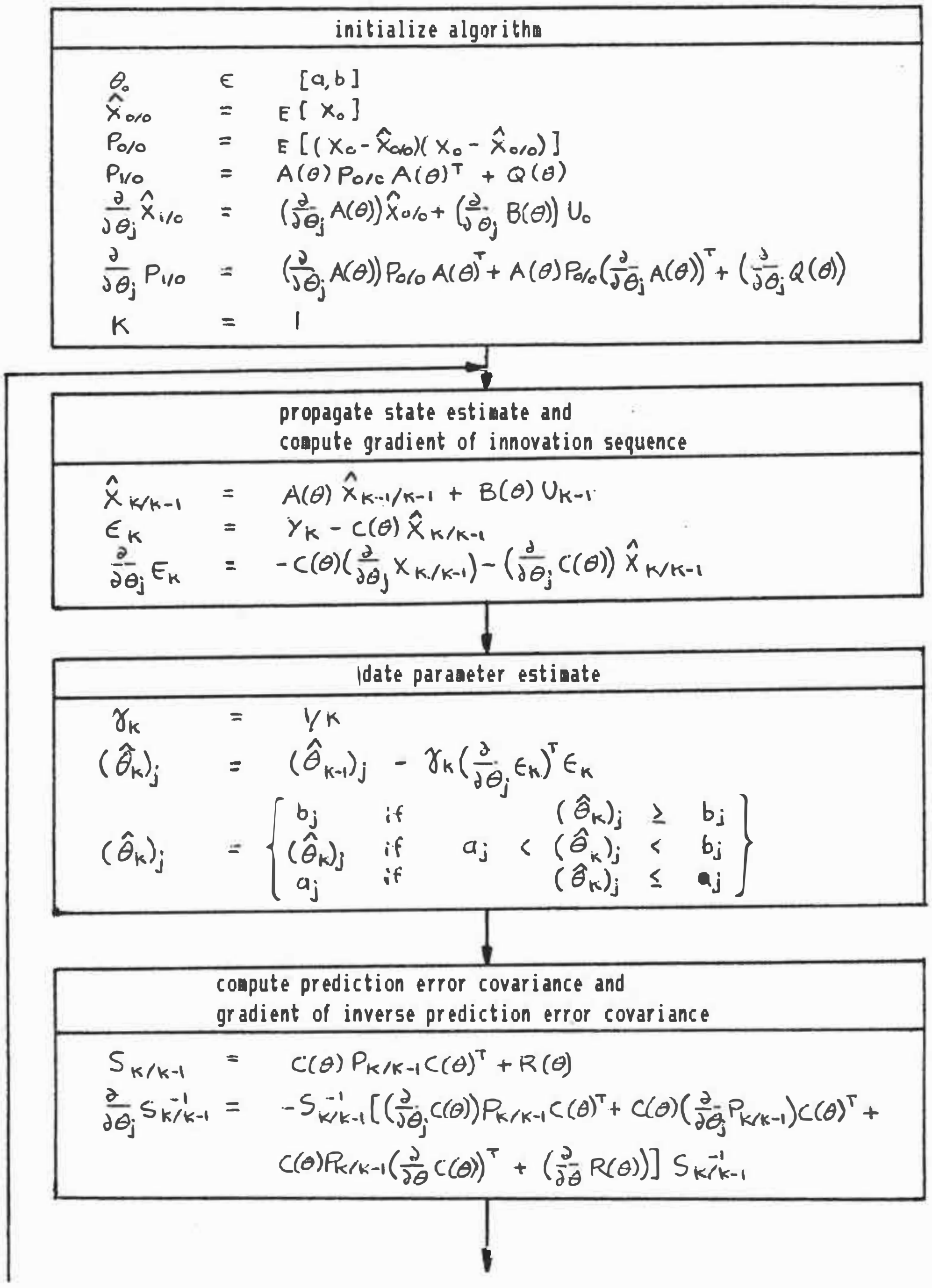
Once the parameters are estimated they are substituted into the appropriate matrices and the Kalman filter given in equations (3-7) through (3-12) is executed. A nice feature of this algorithm is that it is relatively easy to implement. It is summarized in the form of a flowchart shown in Figure 5.

This algorithm was simulated to determine how well it performs. The details of this simulation are given in the following chapter. However, the results indicate that this algorithm performs poorly under the test conditions used. The parameter estimates appear to converge at a very slow rate. An improvement to this algorithm which would not result in an excessive amount of additional computation was sought after. This improvement is the topic of the next section of this chapter.

The Adaptive Kalman Filter Based on the Modified Stochastic Gradient Algorithm

In this section a modification to the previous algorithm will be proposed. It does not result in a large amount of additional computation. The simulation results of this algorithm will also be briefly discussed.

The stochastic gradient algorithm will be modified by introducing a weighting factor into the cost function given in equation (3-14). This weighting factor, $S_{k/k-1}$, will be the prediction error covariance matrix. The modified algorithm wiil minimize the following weighted mean square prediction error:


initialize algorithm
propagate state estimate and
compute gradient of innovation sequence
|date parameter estimate
compute prediction error covariance and
gradient of inverse prediction error covariance

initialize algorithm

$$\theta_0 \in [a,b]$$
$$\hat{X}_{0/0} = E[X_0]$$
$$P_{0/0} = E[(X_0-\hat{X}_{0/0})(X_0-\hat{X}_{0/0})]$$
$$P_{1/0} = A(\theta)P_{0/0}A(\theta)^T + Q(\theta)$$
$$\frac{\partial}{\partial\theta_j}\hat{X}_{1/0} = \left(\frac{\partial}{\partial\theta_j}A(\theta)\right)\hat{X}_{0/0} + \left(\frac{\partial}{\partial\theta_j}B(\theta)\right)U_0$$
$$\frac{\partial}{\partial\theta_j}P_{1/0} = \left(\frac{\partial}{\partial\theta_j}A(\theta)\right)P_{0/0}A(\theta)^T + A(\theta)P_{0/0}\left(\frac{\partial}{\partial\theta_j}A(\theta)\right)^T + \left(\frac{\partial}{\partial\theta_j}Q(\theta)\right)$$
$$K = 1$$

propagate state estimate and
compute gradient of innovation sequence

$$\hat{X}_{K/K-1} = A(\theta)\hat{X}_{K-1/K-1} + B(\theta)U_{K-1}$$
$$\epsilon_K = Y_K - C(\theta)\hat{X}_{K/K-1}$$
$$\frac{\partial}{\partial\theta_j}\epsilon_K = -C(\theta)\left(\frac{\partial}{\partial\theta_j}X_{K/K-1}\right) - \left(\frac{\partial}{\partial\theta_j}C(\theta)\right)\hat{X}_{K/K-1}$$

|date parameter estimate

$$\gamma_K = \gamma_K$$
$$(\hat{\theta}_K)_j = (\hat{\theta}_{K-1})_j - \gamma_K\left(\frac{\partial}{\partial\theta_j}\epsilon_K\right)^T\epsilon_K$$
$$(\hat{\theta}_K)_j = \left\{\begin{array}{lllll} b_j & \text{if} & & (\hat{\theta}_K)_j \geq & b_j \\ (\hat{\theta}_K)_j & \text{if} & a_j < & (\hat{\theta}_K)_j < & b_j \\ a_j & \text{if} & & (\hat{\theta}_K)_j \leq & a_j \end{array}\right\}$$

compute prediction error covariance and
gradient of inverse prediction error covariance

$$S_{K/K-1} = C(\theta)P_{K/K-1}C(\theta)^T + R(\theta)$$
$$\frac{\partial}{\partial\theta_j}S^{-1}_{K/K-1} = -S^{-1}_{K/K-1}\left[\left(\frac{\partial}{\partial\theta_j}C(\theta)\right)P_{K/K-1}C(\theta)^T + C(\theta)\left(\frac{\partial}{\partial\theta_j}P_{K/K-1}\right)C(\theta)^T + C(\theta)P_{K/K-1}\left(\frac{\partial}{\partial\theta}C(\theta)\right)^T + \left(\frac{\partial}{\partial\theta}R(\theta)\right)\right]S^{-1}_{K/K-1}$$

Figure 5. Flowchart for the Adaptive Kalman Filter Based on the Stochastic Gradient Algorithm.

update state estimate and error covariance

$$K_k = P_{k/k-1}\,C(\theta)^T S^{-1}_{k/k-1}$$
$$D_k = (I - K_k C(\theta))$$
$$P_{k/k} = D_k P_{k/k-1} D_k^T + K_k R(\theta) K_k^T$$
$$\hat{x}_{k/k} = \hat{x}_{k/k-1} + K_k \epsilon_k$$

compute gradient of Kalman gain,
gradient of propagated error covariance
and gradient of propagated state estimate

$$\frac{\partial}{\partial\theta_j} K_k = \left(\frac{\partial}{\partial\theta_j} P_{k/k-1}\right) C(\theta)^T S^{-1}_{k/k-1} + P_{k/k-1}\left(\frac{\partial}{\partial\theta_j} C(\theta)\right)^T S^{-1}_{k/k-1} + P_{k/k-1} C(\theta)^T \left(\frac{\partial}{\partial\theta_j} S^{-1}_{k/k-1}\right)$$

$$\begin{aligned}\frac{\partial}{\partial\theta_j} P_{k/k} = {} & -\left[\left(\frac{\partial}{\partial\theta_j} K_k\right) C(\theta) + K_k\left(\frac{\partial}{\partial\theta_j} C(\theta)\right)\right] P_{k/k-1} D_k^T \\ & - D_k P_{k/k-1}\left[\left(\frac{\partial}{\partial\theta_j} K_k\right) C(\theta) + K_k\left(\frac{\partial}{\partial\theta_j} C(\theta)\right)\right]^T \\ & + D_k\left(\frac{\partial}{\partial\theta_j} P_{k/k-1}\right) D_k^T + K_k\left(\frac{\partial}{\partial\theta_j} R(\theta)\right) K_k^T \\ & + \left(\frac{\partial}{\partial\theta_j} K_k\right) R(\theta) K_k^T + K_k R(\theta)\left(\frac{\partial}{\partial\theta_j} K_k\right)^T\end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial\theta_j} P_{k+1/k} = {} & \left(\frac{\partial}{\partial\theta_j} A(\theta)\right) P_{k/k} A(\theta)^T + A(\theta)\left(\frac{\partial}{\partial\theta_j} P_{k/k}\right) A(\theta)^T \\ & + A(\theta) P_{k/k}\left(\frac{\partial}{\partial\theta_j} A(\theta)\right)^T + \left(\frac{\partial}{\partial\theta_j} Q(\theta)\right)\end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial\theta_j} \hat{x}_{k+1/k} = {} & A(\theta) D_k\left(\frac{\partial}{\partial\theta_j} \hat{x}_{k/k-1}\right) \\ & + \left[A(\theta)\left(\frac{\partial}{\partial\theta_j} K_k\right) + \left(\frac{\partial}{\partial\theta_j} A(\theta)\right) K_k\right] \epsilon_k \\ & + \left[\left(\frac{\partial}{\partial\theta_j} A(\theta)\right) - A(\theta) K_k\left(\frac{\partial}{\partial\theta_j} C(\theta)\right)\right] \hat{x}_{k/k-1} \\ & + \left(\frac{\partial}{\partial\theta_j} B(\theta)\right) u_k\end{aligned}$$

compute propagated error covariance

$$P_{k+1/k} = A(\theta) P_{k/k} A(\theta)^T + Q(\theta)$$

increment time interval

$$k = k + 1$$

Figure 5 (continued)

$$J(\theta) = \frac{1}{2}\,[\varepsilon_k^T S_{k/k-1}^{-1} \varepsilon_k] \;. \tag{3-29}$$

To formulate a parameter identification algorithm based on minimizing equation (3-29) requires an estimate of the prediction error covariance matrix. The inverse of this matrix is already required in computing the Kalman gain, given in equation (3-10):

$$S_{k/k-1}^{-1} = [C(\theta)P_{k/k-1}C(\theta)^T + R(\theta)]^{-1}. \tag{3-30}$$

The gradient of the cost function is found by differentiating equation (3-29), with respect to each of the unknown parameters to be identified. The result is:

$$\frac{\partial}{\partial\theta_j} J(\theta) = E[(\frac{\partial}{\partial\theta_j}\,\varepsilon_k)^T S_{k/k-1}^{-1}\varepsilon_k + \frac{1}{2}\,\varepsilon_k^T(\frac{\partial}{\partial\theta_j}\,S_{k/k-1}^{-1})\varepsilon_k] \;. \tag{3-31}$$

As before, the parameter estimates can be adjusted in the negative gradient direction by using a stochastic approximation. This results in the following parameter identification algorithm:

$$\begin{aligned}(\hat{\theta}_k)_j = (\hat{\theta}_{k-1})_j - \gamma_k[(\frac{\partial}{\partial\theta_j}\,\varepsilon_k)^T S_{k/k-1}^{-1}\varepsilon_k \\ + \frac{1}{2}\,\varepsilon_k^T(\frac{\partial}{\partial\theta_j}\,S_{k/k-1}^{-1})\varepsilon_k] \;.\end{aligned} \tag{3-32}$$

The gradient of the innovation sequence is found as described in the last section. The gradient of the estimated prediction error covariance is determined by differentiating equation (3-30) with respect to the jth parameter:

$$\left(\frac{\partial}{\partial\theta_j} S^{-1}_{k/k-1}\right) = -S^{-1}_{k/k-1}\left[\left(\frac{\partial}{\partial\theta_j} C(\theta)\right)P_{k/k-1}C(\theta)^T + C(\theta)\left(\frac{\partial}{\partial\theta_j} P_{k/k-1}\right)C(\theta)^T + C(\theta)P_{k/k-1}\left(\frac{\partial}{\partial\theta_j} C(\theta)\right) + \left(\frac{\partial}{\partial\theta_j} R(\theta)\right)\right]S^{-1}_{k/k-1} . \qquad (3\text{-}33)$$

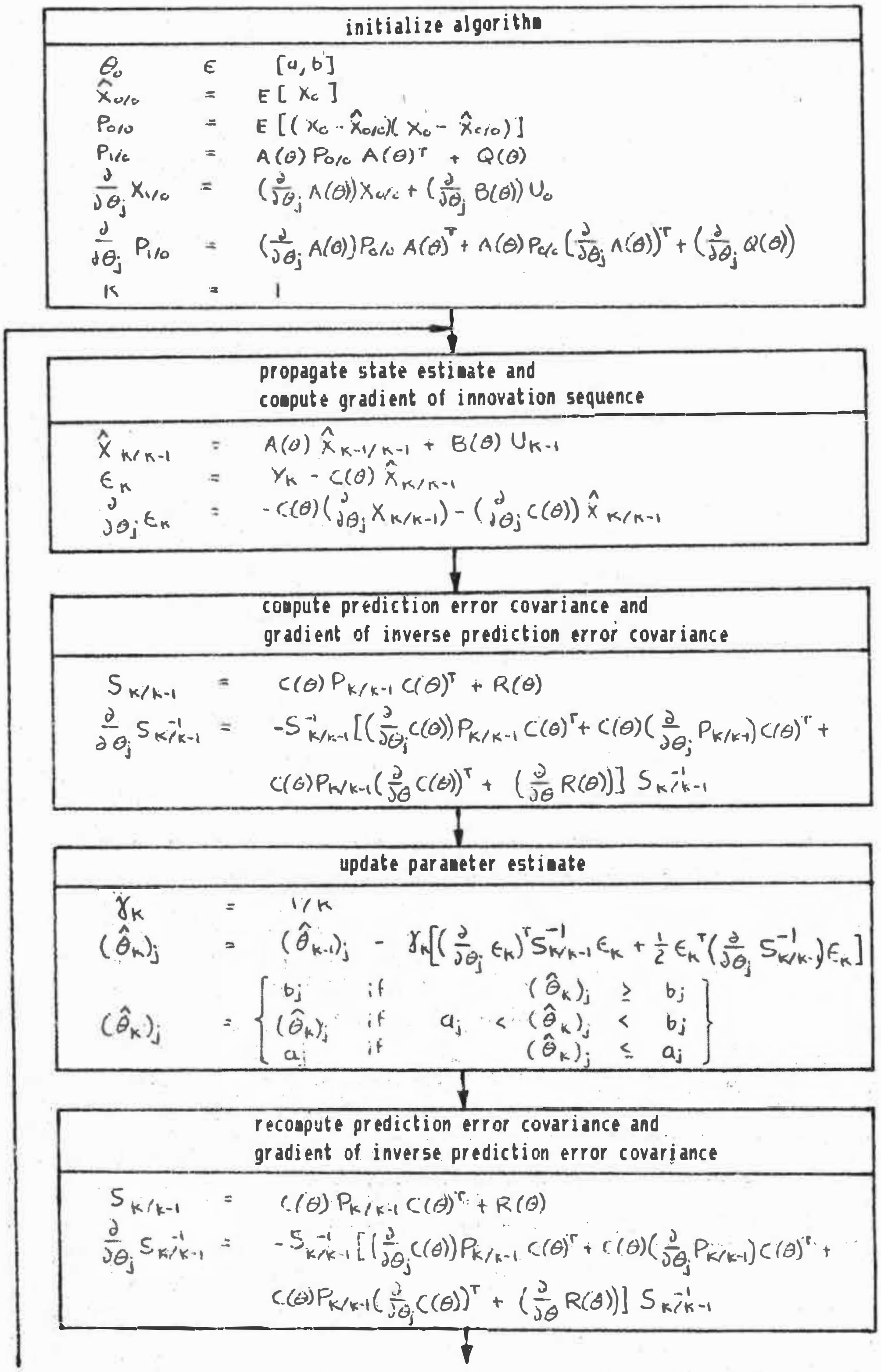
This modification does not present a large amount of added computational burden. Furthermore, equation (3-33) can be used in the equation to compute the gradient of the Kalman gain. The additional programming effort is minimal. The adaptive Kalman filter using the modified stochastic gradient parameter identification algorithm is summarized in the form of a flow chart shown in Figure 6.

The modified algorithm was simulated to determine how well it performed when compared to the original algorithm. The simulation indicated that there was an improvement. However, the results were not as good as had been hoped for. The simulation will be further discussed in the next chapter.

A more significant improvement in performance is needed for the adaptive Kalman filter to be a viable option for the transfer alignment problem. A more sophisticated parameter identification algorithm was found and implemented. This is the subject of the next section of this chapter.

The Adaptive Kalman Filter Based on the Stochastic Newton Algorithm

The adaptive Kalman filters discussed up to this point relied on a stochastic gradient algorithm to identify the unknown parameters.



initialize algorithm

$$\theta_0 \in [a, b]$$
$$\hat{x}_{0/0} = E[x_0]$$
$$P_{0/0} = E[(x_0 - \hat{x}_{0/0})(x_0 - \hat{x}_{0/0})']$$
$$P_{1/0} = A(\theta) P_{0/0} A(\theta)^T + Q(\theta)$$
$$\frac{\partial}{\partial\theta_j} x_{1/0} = (\frac{\partial}{\partial\theta_j} A(\theta)) x_{0/0} + (\frac{\partial}{\partial\theta_j} B(\theta)) U_0$$
$$\frac{\partial}{\partial\theta_j} P_{1/0} = (\frac{\partial}{\partial\theta_j} A(\theta)) P_{0/0} A(\theta)^T + A(\theta) P_{0/0} (\frac{\partial}{\partial\theta_j} A(\theta))^T + (\frac{\partial}{\partial\theta_j} Q(\theta))$$
$$K = 1$$

propagate state estimate and
compute gradient of innovation sequence

$$\hat{x}_{k/k-1} = A(\theta) \hat{x}_{k-1/k-1} + B(\theta) U_{k-1}$$
$$\epsilon_k = Y_k - C(\theta) \hat{x}_{k/k-1}$$
$$\frac{\partial}{\partial\theta_j} \epsilon_k = -C(\theta)(\frac{\partial}{\partial\theta_j} x_{k/k-1}) - (\frac{\partial}{\partial\theta_j} C(\theta)) \hat{x}_{k/k-1}$$

compute prediction error covariance and
gradient of inverse prediction error covariance

$$S_{k/k-1} = C(\theta) P_{k/k-1} C(\theta)^T + R(\theta)$$
$$\frac{\partial}{\partial\theta_j} S^{-1}_{k/k-1} = -S^{-1}_{k/k-1}[(\frac{\partial}{\partial\theta_j} C(\theta)) P_{k/k-1} C(\theta)^T + C(\theta)(\frac{\partial}{\partial\theta_j} P_{k/k-1}) C(\theta)^T + C(\theta) P_{k/k-1} (\frac{\partial}{\partial\theta} C(\theta))^T + (\frac{\partial}{\partial\theta} R(\theta))] S^{-1}_{k/k-1}$$

update parameter estimate

$$\gamma_k = 1/k$$
$$(\hat{\theta}_k)_j = (\hat{\theta}_{k-1})_j - \gamma_k[(\frac{\partial}{\partial\theta_j} \epsilon_k)^T S^{-1}_{k/k-1} \epsilon_k + \frac{1}{2} \epsilon_k^T (\frac{\partial}{\partial\theta_j} S^{-1}_{k/k-1}) \epsilon_k]$$
$$(\hat{\theta}_k)_j = \begin{cases} b_j & \text{if} \quad (\hat{\theta}_k)_j \geq b_j \\ (\hat{\theta}_k)_j & \text{if} \quad a_j < (\hat{\theta}_k)_j < b_j \\ a_j & \text{if} \quad (\hat{\theta}_k)_j \leq a_j \end{cases}$$

recompute prediction error covariance and
gradient of inverse prediction error covariance

$$S_{k/k-1} = C(\theta) P_{k/k-1} C(\theta)^T + R(\theta)$$
$$\frac{\partial}{\partial\theta_j} S^{-1}_{k/k-1} = -S^{-1}_{k/k-1}[(\frac{\partial}{\partial\theta_j} C(\theta)) P_{k/k-1} C(\theta)^T + C(\theta)(\frac{\partial}{\partial\theta_j} P_{k/k-1}) C(\theta)^T + C(\theta) P_{k/k-1} (\frac{\partial}{\partial\theta_j} C(\theta))^T + (\frac{\partial}{\partial\theta} R(\theta))] S^{-1}_{k/k-1}$$

Figure 6. Flowchart for the Adaptive Kalman Filter Based on the Modified Stochastic Gradient Algorithm.

update state estimate and error covariance

$$K_k = P_{k/k-1} C(\theta)^T S_{k/k-1}^{-1}$$
$$D_k = (I - K_k C(\theta))$$
$$P_{k/k} = D_k P_{k/k-1} D_k^T + K_k R(\theta) K_k^T$$
$$\hat{x}_{k/k} = \hat{x}_{k/k-1} + K_k \epsilon_k$$

compute gradient of Kalman gain, gradient of propagated error covariance and gradient of propagated state estimate

$$\frac{\partial}{\partial\theta_j} K_k = \left(\frac{\partial}{\partial\theta_j} P_{k/k-1}\right) C(\theta)^T S_{k/k-1}^{-1} + P_{k/k-1}\left(\frac{\partial}{\partial\theta_j} C(\theta)\right)^T S_{k/k-1}^{-1} + P_{k/k-1} C(\theta)^T \left(\frac{\partial}{\partial\theta_j} S_{k/k-1}^{-1}\right)$$

$$\begin{aligned}\frac{\partial}{\partial\theta_j} P_{k/k} = & -\left[\left(\frac{\partial}{\partial\theta_j} K_k\right) C(\theta) + K_k\left(\frac{\partial}{\partial\theta_j} C(\theta)\right)\right] P_{k/k-1} D_k^T \\ & - D_k P_{k/k-1}\left[\left(\frac{\partial}{\partial\theta_j} K_k\right) C(\theta) + K_k\left(\frac{\partial}{\partial\theta_j} C(\theta)\right)\right]^T \\ & + D_k\left(\frac{\partial}{\partial\theta_j} P_{k/k-1}\right) D_k^T + K_k\left(\frac{\partial}{\partial\theta_j} R(\theta)\right) K_k^T \\ & + \left(\frac{\partial}{\partial\theta_j} K_k\right) R(\theta) K_k^T + K_k R(\theta)\left(\frac{\partial}{\partial\theta_j} K_k\right)^T\end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial\theta_j} P_{k+1/k} = & \left(\frac{\partial}{\partial\theta_j} A(\theta)\right) P_{k/k} A(\theta)^T + A(\theta)\left(\frac{\partial}{\partial\theta_j} P_{k/k}\right) A(\theta)^T \\ & + A(\theta) P_{k/k}\left(\frac{\partial}{\partial\theta_j} A(\theta)\right)^T + \left(\frac{\partial}{\partial\theta_j} Q(\theta)\right)\end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial\theta_j} \hat{x}_{k+1/k} = & A(\theta) D_k\left(\frac{\partial}{\partial\theta_j} \hat{x}_{k/k-1}\right) \\ & + \left[A(\theta)\left(\frac{\partial}{\partial\theta_j} K_k\right) + \left(\frac{\partial}{\partial\theta_j} A(\theta)\right) K_k\right] \epsilon_k \\ & + \left[\left(\frac{\partial}{\partial\theta_j} A(\theta)\right) - A(\theta) K_k\left(\frac{\partial}{\partial\theta_j} C(\theta)\right)\right] \hat{x}_{k/k-1} \\ & + \left(\frac{\partial}{\partial\theta_j} B(\theta)\right) u_k\end{aligned}$$

compute propagated error covariance

$$P_{k+1/k} = A(\theta) P_{k/k} A(\theta)^T + Q(\theta)$$

increment time interval

$$k = k + 1$$

Figure 6 (continued)

Unfortunately, the performance of these algorithms was less than satisfactory for the application considered in this study. An improved algorithm will be presented in this section. The performance of this adaptive Kalman filter will then be compared to the first two algorithms.

The derivation of this improved algorithm starts with a second order deterministic parameter identification scheme. The stochastic counterpart of this algorithm is then found. Finally, an approximation will be introduced that allows the algorithm to be executed in a reasonable amount of time. Such an algorithm is discussed in reference [9].

This parameter identification algorithm will again attempt to minimize the weighted mean square prediction error given in equation (3-29). It is repeated here for convenience:

$$J(\theta) = \frac{1}{2} E[\varepsilon_k^T S_{k/k-1}^{-1} \varepsilon_k] \ . \tag{3-34}$$

As before, the mean square prediction error will be minimized when the gradient of the cost function is set to zero. This time a Taylor series expansion of the cost function about a nominal set of parameter estimates will be considered. A deterministic parameter identification scheme will now be derived [15, pp. 348-50].

The Taylor series expansion of the cost function in equation (3-34) about a nominal set of parameters, θ_0, is:

$$J(\theta) = J(\theta_0) + (\frac{\partial}{\partial\theta} J(\theta))|_{\theta_0}(\underline{\theta} - \underline{\theta}_0)$$

$$+ \frac{1}{2} (\underline{\theta} - \underline{\theta}_0)(\frac{\partial^2}{\partial\theta^2} J(\theta))|_{\theta_0}(\underline{\theta} - \underline{\theta}_0) + \dots \quad (3\text{-}35)$$

Equation (3-35) is truncated to two terms and differentiated:

$$\frac{\partial}{\partial\theta} J(\theta) = (\frac{\partial}{\partial\theta} J(\theta_0)) + (\frac{\partial^2}{\partial\theta^2} J(\theta_0))(\underline{\theta} - \underline{\theta}_0) \ . \quad (3\text{-}36)$$

Equation (3-36) is set to zero and then solved for $(\underline{\theta} - \underline{\theta}_0)$. The result is an expression for how much the parameters deviate from their nominal values:

$$(\underline{\theta} - \underline{\theta}_0) = -[\frac{\partial^2}{\partial\theta^2} J(\theta)]^{-1}[\frac{\partial}{\partial\theta} J(\theta)] \ . \quad (3\text{-}37)$$

A new parameter vector estimate can be found by adding this deviation to the old estimate:

$$\underline{\hat{\theta}}_k = \underline{\hat{\theta}}_{k-1} + (\underline{\theta} - \underline{\theta}_0) \ . \quad (3\text{-}38)$$

The resulting deterministic identification scheme is known as the Newton-Raphson method:

$$\underline{\hat{\theta}}_k = \underline{\hat{\theta}}_{k-1} - [\frac{\partial^2}{\partial\theta^2} J(\theta)]^{-1}[\frac{\partial}{\partial\theta} J(\theta)] \ . \quad (3\text{-}39)$$

The second derivative matrix in equation (3-39) is referred to as the Hessian matrix. The Hessian matrix modifies the search direction from that given by the negative gradient of the cost function. The result is improved convergence of the parameter estimates. When the function is close to being minimized the scheme in equation (3-39) is known to be very efficient. However, far from the minimum, it may be inefficient or even diverge. The divergence problem may be eliminated by ensuring the Hessian matrix is positive definite. This guarantees that the search direction is always "downhill" [9, p. 46].

A natural variant of the stochastic gradient algorithm applied to equation (3-39) results in a stochastic Newton approach to identifying the parameters [9, p. 47]. The stochastic counterpart of equation (3-39) is analagous to equation (3-16). Again, γ_k is a converging gain sequence having the properties discussed in equation (3-3). The stochastic Newton approach to identifying the parameters is:

$$\underline{\hat{\theta}}_k = \underline{\hat{\theta}}_{k-1} + \gamma_k [\frac{\partial^2}{\partial \theta^2} J(\theta)]^{-1} [\frac{\partial}{\partial \theta} J(\theta)] \ . \qquad (3\text{-}40)$$

This parameter identification scheme should give vastly improved results when compared to the first two algorithms. Unfortunately, computing the Hessian matrix would be computationally prohibitive.

Assume an approximation for the Hessian matrix can be found. Let this approximation be represented by R_k. Also, assume the weighting matrix, $S^{-1}_{k/k-1}$, in equation (3-34) is not time varying. The gradient of the cost function, $J(\theta)$, is found by differentiating equation (3-34).

Using this expression for the gradient, and the approximation of the Hessian matrix, equation (3-40) becomes:

$$\underline{\hat{\theta}}_k = \underline{\hat{\theta}}_{k-1} + \gamma_k R_k^{-1}[(\frac{\partial}{\partial\theta}\,\varepsilon_k)^T S_{k/k-1}^{-1}\varepsilon_k]\ . \tag{3-41}$$

Notice that an entire parameter vector is being estimated, unlike in the previous two algorithms. It is more convenient to estimate the parameters as a vector, rather than individually in this algorithm. The gradient of the innovation sequence in equation (3-41) is an (r x p) matrix. The gradient of the innovation sequence with respect to the jth parameter is still determined by equations (3-17), (3-22), (3-23), (3-24) and (3-25). These equations can only be evaluated to find the gradient of the innovation with respect to a specific parameter. This gives an (r x 1) vector. Doing so for each of the p unknown parameters, results in p (r x 1) vectors. These vectors are brought together to form an (r x p) matrix. This is the gradient of the innovation with respect to all parameters, used in equation (3-41). The form of this matrix is illustrated below for clarity:

$$\frac{\partial}{\partial\theta}\,\varepsilon_k = \left|\begin{array}{cccc} \frac{\partial\varepsilon_1}{\partial\theta_1} & \frac{\partial\varepsilon_1}{\partial\theta_2} & \cdots & \frac{\partial\varepsilon_1}{\partial\theta_p} \\ \frac{\partial\varepsilon_2}{\partial\theta_1} & \frac{\partial\varepsilon_2}{\partial\theta_2} & \cdots & \frac{\partial\varepsilon_2}{\partial\theta_p} \\ \vdots & & & \vdots \\ \frac{\partial\varepsilon_r}{\partial\theta_1} & \frac{\partial\varepsilon_r}{\partial\theta_2} & \cdots & \frac{\partial\varepsilon_r}{\partial\theta_p} \end{array}\right| (r \times p)\ . \tag{3-42}$$

The dimension of the Hessian matrix is (p x p). The prediction error covariance matrix has a dimension of (r x r). The result of equation (3-41) is a (p x 1) vector of parameter estimates.

A suitable approximation of the Hessian matrix will be determined next. It can be developed by substituting a Taylor series expansion of the innovation sequence into the cost function. The second derivative of the cost function will then be evaluated. Finally, the Hessian matrix will be estimated by using a stochastic approximation.

The Taylor series expansion of the innovation sequence about some nominal value of the parameters is:

$$\varepsilon_k(\theta) = \varepsilon_k(\theta_0) + \left(\frac{\partial}{\partial\theta}\,\varepsilon_k(\theta)\right)\Big|_{\theta_0}(\underline{\theta} - \underline{\theta}_0) + \ldots \; . \tag{3-43}$$

Using the first two terms of this Taylor series, the cost function given in equation (3-34) can be written as:

$$J(\theta) = \frac{1}{2}\,E\,[[\varepsilon_k(\theta_0) + (\frac{\partial}{\partial\theta}\,\varepsilon_k(\theta))|_{\theta_0}(\underline{\theta} - \underline{\theta}_0)]S^{-1}_{k/k-1}[\varepsilon_k(r_0)$$

$$+ (\frac{\partial}{\partial\theta}\,\varepsilon_k(\theta))|_{\theta_0}(\underline{\theta} - \underline{\theta}_0)]]. \tag{3-44}$$

An approximation to the Hessian matrix can be found by evaluating the second derivative of equation (3-44):

$$\frac{\partial^2}{\partial\theta^2}\,J(\theta) \sim R_k = E[(\frac{\partial}{\partial\theta}\,\varepsilon_k)^T S^{-1}_{k/k-1}\,(\frac{\partial}{\partial\theta}\,\varepsilon_k)] \; . \tag{3-45}$$

Equation (3-45) is difficult to evaluate because it involves an expectation. However, it can be written in such a manner that an estimate of the Hessian matrix can be determined by another stochastic approximation:

$$E[R_k - (\frac{\partial}{\partial\theta}\,\varepsilon_k)^T S_{k/k-1}^{-1}(\frac{\partial}{\partial\theta}\,\varepsilon_k)] = 0 \;. \tag{3-46}$$

Applying the principle of stochastic approximation to equation (3-46) results in the following equation to estimate the Hessian matrix:

$$R_k = R_{k-1} + \gamma_k[(\frac{\partial}{\partial\theta}\,\varepsilon_k)^T S_{k/k-1}^{-1}(\frac{\partial}{\partial\theta}\,\varepsilon_k) - R_{k-1}] \;. \tag{3-47}$$

The gain sequence, γ_k, does not have to be the same as in equation (3-40), but must have the properties given in equation (3-3). If it is chosen as suggested by equation (3-4), then equation (3-47) recursively estimates the mean value of the term within the expectation of equation (3-45).

Equation (3-41) requires inversion of the Hessian matrix. Numerical problems may be encountered when this matrix is singular, or nearly so. A singular condition of the Hessian matrix is caused by parameter estimates which do not lead to a unique, well defined minimum of the cost function. This corresponds to a "valley" along the null space of the Hessian matrix [9, p. 202]. There are two conditions which may lead to this problem. First, the model might be overparameterized. The second condition is caused by input signals having insufficient spectral characteristics to identify the parameters [6, pp. 72-74].

Several techniques are available to correct these problems. Overparameterization may be solved by reducing the order of the model. Insufficient excitation may be prevented by adding a small external persistently exciting input signal [6, pp. 216-17]. An alternative is to temporarily halt the parameter identification process until the input signal is sufficiently exciting. To further guarantee that numerical problems do not develop, the Hessian matrix may be regularized. One technique is the Levenberg-Marquardt method [9, p. 365]. This consists of simply adding an identity matrix multiplied by a small constant, δ, to equation (3-47):

$$R_k = R_{k-1} + \gamma_k[(\frac{\partial}{\partial\theta} \varepsilon_k)^T S_{k/k-1}^{-1}(\frac{\partial}{\partial\theta} \varepsilon_k) - R_{k-1} + \delta I] \ . \qquad (3\text{-}48)$$

A more sophisticated technique involves U-D factorization. A modification, similar to above, is made to an emerging null space of R_k to prevent any eigenvalue from approaching zero [9, p. 365]. The added computational burden might be prohibitive.

The adaptive Kalman filter discussed in this section uses equations (3-41) and (3-48) to identify the parameters, along with the necessary supporting equations. It will be called the stochastic Newton algorithm. It can be shown that this algorithm is essentially identical to an extended Kalman filter modified to couple the gradient of the Kalman gain with the parameter identification process [9, p. 130]. This coupling is absent in an unmodified extended Kalman filter. This absence has a harmful effect on the convergence properties of the extended

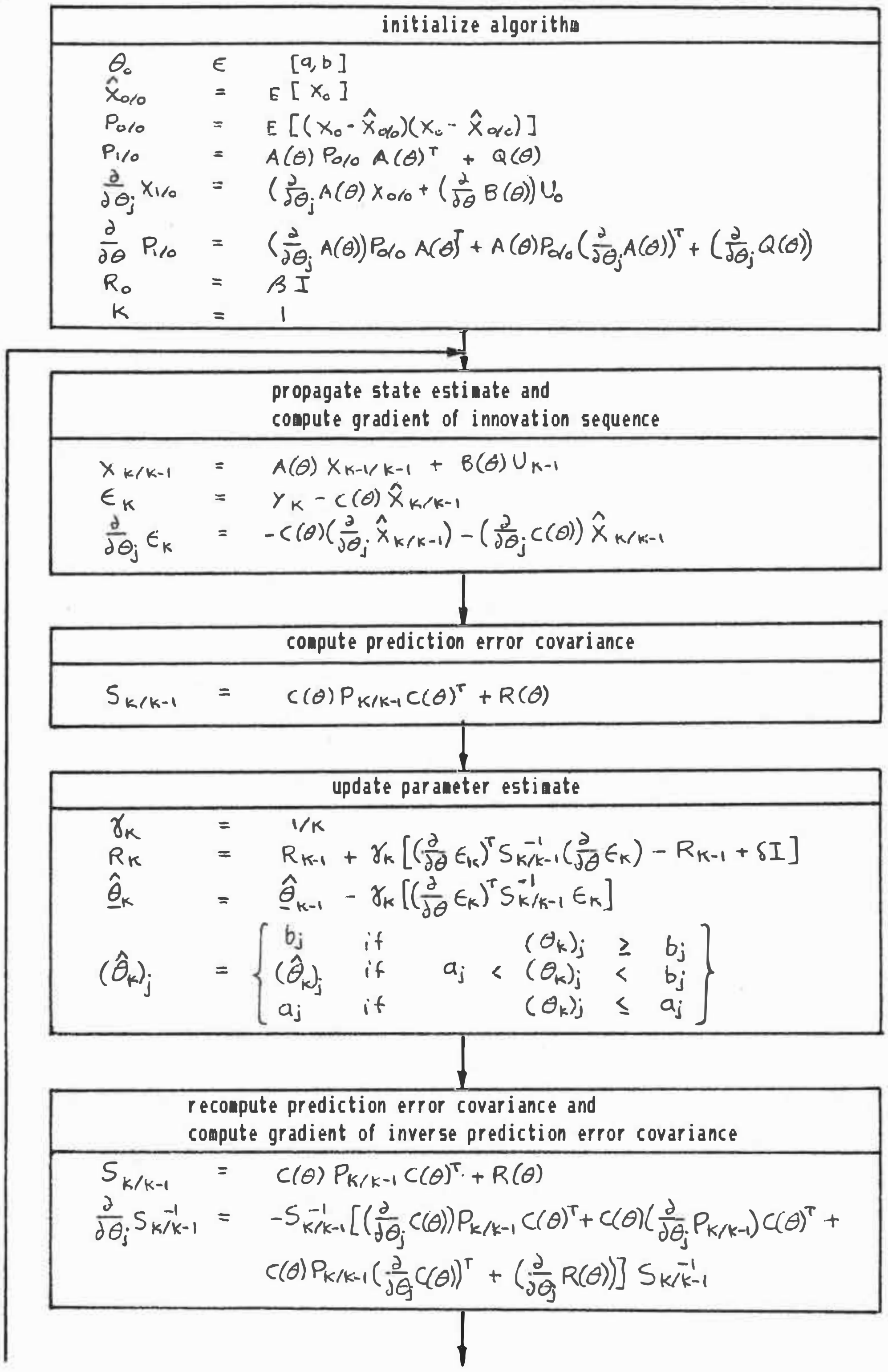
Kalman filter. The stochastic Newton algorithm is illustrated in the form of a flowchart shown in Figure 7.

The stochastic Newton algorithm was simulated to evaluate its performance. The simulation indicated that the performance of this algorithm was significantly better than the first two. The disadvantage of this algorithm is that it requires inversion of the Hessian matrix with an order equal to the number of parameters to be identified. This might impose an unacceptable computational burden for some applications. The model used in the transfer alignment problem will have at most six parameters to be identified. The inversion of a sixth order Hessian matrix is not unreasonable.

Modifications to Account for Time Varying Parameters

A model for alignment error was presented in Chapter 2. It was reasoned that the dynamics of the alignment error is time varying. This is caused by the changing conditions within the aircraft and its environment during the alignment procedure. It was proposed to estimate the angles of alignment error using an adaptive Kalman filter. The adaptive Kalman filter would determine the best model for the flexure dynamics, at a given time, from the available data.

Three versions of an adaptive Kalman filter have been presented in this chapter. These algorithms were developed assuming the parameters of the model are not time varying. The adaptive Kalman filters may be modified to identify time varying parameters. The state estimates of



initialize algorithm

$$
\begin{aligned}
\theta_0 &\in [a,b] \\
\hat{X}_{0/0} &= E[X_0] \\
P_{0/0} &= E[(X_0-\hat{X}_{0/0})(X_0-\hat{X}_{0/0})] \\
P_{1/0} &= A(\theta)\,P_{0/0}\,A(\theta)^T + Q(\theta) \\
\frac{\partial}{\partial\theta_j}X_{1/0} &= (\frac{\partial}{\partial\theta_j}A(\theta)\,X_{0/0} + (\frac{\partial}{\partial\theta}B(\theta))U_0 \\
\frac{\partial}{\partial\theta}P_{1/0} &= (\frac{\partial}{\partial\theta_j}A(\theta))P_{0/0}\,A(\theta)^T + A(\theta)P_{0/0}(\frac{\partial}{\partial\theta_j}A(\theta))^T + (\frac{\partial}{\partial\theta_j}Q(\theta)) \\
R_0 &= \beta I \\
K &= 1
\end{aligned}
$$

propagate state estimate and
compute gradient of innovation sequence

$$
\begin{aligned}
X_{K/K-1} &= A(\theta)\,X_{K-1/K-1} + B(\theta)\,U_{K-1} \\
\epsilon_K &= Y_K - C(\theta)\,\hat{X}_{K/K-1} \\
\frac{\partial}{\partial\theta_j}\epsilon_K &= -C(\theta)(\frac{\partial}{\partial\theta_j}\hat{X}_{K/K-1}) - (\frac{\partial}{\partial\theta_j}C(\theta))\,\hat{X}_{K/K-1}
\end{aligned}
$$

compute prediction error covariance

$$
S_{K/K-1} = C(\theta)\,P_{K/K-1}\,C(\theta)^T + R(\theta)
$$

update parameter estimate

$$
\begin{aligned}
\gamma_K &= 1/K \\
R_K &= R_{K-1} + \gamma_K\left[(\frac{\partial}{\partial\theta}\epsilon_K)^T S^{-1}_{K/K-1}(\frac{\partial}{\partial\theta}\epsilon_K) - R_{K-1} + \delta I\right] \\
\underline{\hat{\theta}}_K &= \underline{\hat{\theta}}_{K-1} - \gamma_K\left[(\frac{\partial}{\partial\theta}\epsilon_K)^T S^{-1}_{K/K-1}\,\epsilon_K\right] \\
(\hat{\theta}_K)_j &= \left\{\begin{array}{llrcl} b_j & \text{if} & & (\theta_K)_j & \geq b_j \\ (\hat{\theta}_K)_j & \text{if} & a_j < & (\theta_K)_j & < b_j \\ a_j & \text{if} & & (\theta_K)_j & \leq a_j \end{array}\right\}
\end{aligned}
$$

recompute prediction error covariance and
compute gradient of inverse prediction error covariance

$$
\begin{aligned}
S_{K/K-1} &= C(\theta)\,P_{K/K-1}\,C(\theta)^T + R(\theta) \\
\frac{\partial}{\partial\theta_j}S^{-1}_{K/K-1} &= -S^{-1}_{K/K-1}\Big[(\frac{\partial}{\partial\theta_j}C(\theta))P_{K/K-1}\,C(\theta)^T + C(\theta)(\frac{\partial}{\partial\theta_j}P_{K/K-1})C(\theta)^T + \\
&\quad C(\theta)P_{K/K-1}(\frac{\partial}{\partial\theta_j}C(\theta))^T + (\frac{\partial}{\partial\theta_j}R(\theta))\Big]\,S^{-1}_{K/K-1}
\end{aligned}
$$

Figure 7. Flowchart for the Adaptive Kalman Filter Based on the Stochastic Newton Algorithm.

update state estimate and error covariance

$$K_k = P_{k/k-1}C(\theta)^T S_{k/k-1}^{-1}$$

$$D_k = (I - K_k C(\theta))$$

$$P_{k/k} = D_k P_{k/k-1} D_k^T + K_k R(\theta) K_k^T$$

$$\hat{x}_{k/k} = \hat{x}_{k/k-1} + K_k \epsilon_k$$

compute gradient of Kalman gain,
gradient of propagated error covariance
and gradient of propagated state estimate

$$\begin{aligned}
\frac{\partial}{\partial\theta_j}K_k &= \left(\frac{\partial}{\partial\theta_j}P_{k/k-1}\right)C(\theta)^T S_{k/k-1}^{-1} + P_{k/k-1}\left(\frac{\partial}{\partial\theta_j}C(\theta)\right)^T S_{k/k-1}^{-1} \\
&\quad + P_{k/k-1}C(\theta)^T\left(\frac{\partial}{\partial\theta_j}S_{k/k-1}^{-1}\right) \\
\frac{\partial}{\partial\theta_j}P_{k/k} &= -\left[\left(\frac{\partial}{\partial\theta_j}K_k\right)C(\theta) + K_k\left(\frac{\partial}{\partial\theta_j}C(\theta)\right)\right]P_{k/k-1}D_k^T \\
&\quad - D_k P_{k/k-1}\left[\left(\frac{\partial}{\partial\theta_j}K_k\right)C(\theta) + K_k\left(\frac{\partial}{\partial\theta_j}C(\theta)\right)\right]^T \\
&\quad + D_k\left(\frac{\partial}{\partial\theta_j}P_{k/k-1}\right)D_k^T + K_k\left(\frac{\partial}{\partial\theta_j}R(\theta)\right)K_k^T \\
&\quad + \left(\frac{\partial}{\partial\theta_j}K_k\right)R(\theta)K_k^T + K_k R(\theta)\left(\frac{\partial}{\partial\theta_j}K_k\right)^T \\
\frac{\partial}{\partial\theta_j}P_{k+1/k} &= \left(\frac{\partial}{\partial\theta_j}A(\theta)\right)P_{k/k}A(\theta)^T + A(\theta)\left(\frac{\partial}{\partial\theta_j}P_{k/k}\right)A(\theta)^T \\
&\quad + A(\theta)P_{k/k}\left(\frac{\partial}{\partial\theta_j}A(\theta)\right)^T + \left(\frac{\partial}{\partial\theta_j}Q(\theta)\right) \\
\frac{\partial}{\partial\theta_j}\hat{x}_{k+1/k} &= A(\theta)D_k\left(\frac{\partial}{\partial\theta_j}\hat{x}_{k/k-1}\right) \\
&\quad + \left[A(\theta)\left(\frac{\partial}{\partial\theta_j}K_k\right) + \left(\frac{\partial}{\partial\theta_j}A(\theta)\right)K_k\right]\epsilon_k \\
&\quad + \left[\left(\frac{\partial}{\partial\theta_j}A(\theta)\right) - A(\theta)K_k\left(\frac{\partial}{\partial\theta_j}C(\theta)\right)\right]\hat{x}_{k/k-1} \\
&\quad + \left(\frac{\partial}{\partial\theta_j}B(\theta)\right)u_k
\end{aligned}$$

compute propagated error covariance

$$P_{k+1/k} = A(\theta)P_{k/k}A(\theta)^T + Q(\theta)$$

increment time interval

$$k = k + 1$$

Figure 7 (continued)

the adaptive Kalman filter correspond to the angles of alignment error between the master and slave IMU. These angles are also time varying. Therefore, a similar modification is needed for the state estimation process.

To enable the adaptive Kalman filter to identify time varying parameters requires a modification of the gain sequence, γ_k, used in estimating the parameters. When the gain sequence is allowed to decay to zero all measurements are weighted equally. Letting the gain sequence decay to a small positive constant places more weight on recently acquired data. A relationship between this constant and the effective memory length of the parameter identification process may be derived [9, p. 274]. Define T_0 as the memory time constant of the filter. Data older than T_0 units of time has an effective weighting that is approximately less than 1/e, or about 36 percent, of newly acquired data. An approximate expression for T_0 is:

$$T_0 = [\lim_{k\to\infty} \gamma_k]^{-1} \; . \tag{3-49}$$

Equation (3-49) illustrates that as the gain sequence decays to a larger constant, previous data tends to be neglected after a shorter period of time. This shifts more weight to recently acquired data. An alternative is to reset the gain sequence at regular time intervals. The length of this interval would depend on how fast the parameters are changing. The value to which the gain sequence is reset would be between some small constant and what was used when the algorithm was initialized.

this case, the input signal is not uniformly persistently exciting [6, p. 72]. This may not be a problem if the aircraft is maneuvering during the alignment process.

A nice feature of the adaptive Kalman filter using the stochastic Newton algorithm to identify the parameters is that it automatically provides a tool to analyze the input data. This tool is the Hessian matrix. If the determinant of the Hessian matrix falls below a certain value, this is an indication that the input signal has insufficient spectral characteristics to identify the parameters of the system. At this point the parameter identification process can be temporarily halted. The two variations of the adaptive Kalman filter based on the stochastic gradient algorithm do not provide this attractive feature.

Summary

The use of an adaptive Kalman filter to estimate the angles of alignment error appears to be an attractive solution to an otherwise complicated problem. This chapter has presented the development of three variations of an adaptive Kalman filter. The performance of these algorithms have been briefly discussed. The adaptive Kalman filter that identifies the parameters using a stochastic Newton algorithm provided superior results when compared to the other two algorithms. This adaptive Kalman filter is recommended for the transfer alignment problem. Not only does it provide better results, it can also be used to indicate the spectral quality of the measurement data.

CHAPTER 4

COMPUTER SIMULATION

This chapter discusses the computer simulations of the adaptive Kalman filters presented in Chapter 3. As previously stated, the results of these tests indicate that the adaptive Kalman filter based on the stochastic Newton algorithm provides superior results when compared to the first two algorithms. Two independent simulations were performed. The first simulation produced initial data regarding the relative performance of the algorithms. The second simulation was conducted to investigate the effect of modifying the assumed statistics of the noise used to excite the model. This chapter will conclude with a brief discussion of the computer language in which the simulation was written.

Simplified Test Model

A general discrete time state space model around which the adaptive Kalman filters were developed was presented in equation (3-5). In Chapter 2, a model for alignment error was presented. This model was developed in such a manner that the angles of alignment error could be estimated by a Kalman filter. An actual simulation of this application was not done because of the complexity in setting up the test conditions. Instead, a comparatively simple, third order, single output model was used to generate measurement data. The relative performance of the adaptive Kalman filters was assessed by determining which algorithm produced the most accurate state estimates from the noisy measurements produced by this model. Despite these simplified test conditions, the simulation still provided valid information about the performance of the adaptive Kalman filters.

The steps in arriving at the third order, state space model used to generate the test data will now be presented. The form of this model is a simplified version of the one given in equation (3-5):

$$\begin{aligned} X_{k+1} &= A(\theta)X_k + w_k \\ Y_k &= C(\theta)X_k + v_k \end{aligned} \qquad (4\text{-}1)$$

The matrices used in equation (4-1) are defined in equation (3-5). They are functions of a set of parameters, θ, and will be assigned values shortly.

A deterministic input was not used in this model. This was done to obtain some added similarity to the alignment error model, discussed in Chapter 2. In the alignment error model, the dynamic error was represented by a Gauss-Markov process for each axis of flexing. Each of these equations was independently excited by a zero mean, gaussian noise sequence, generated by the flexing of the aircraft. In a like manner, each of the state equations in the model used to generate the measurement data was excited by an independent, zero mean, gaussian noise sequence. The three noise sequences had a covariance of one unit each.

The state transition matrix was selected by first developing a suitable characteristic equation directly in the z-plane [4, p. 109].

Two imaginary poles, and one real pole were placed within the unit circle of the z-plane. This guaranteed a stable model would result. The imaginary poles were placed at $z = \pm j\ 0.8$, the real pole was placed at $z = +0.5$. The resulting characteristic equation is:

$$Z^3 + .5Z^2 + .64Z + .32 = 0\ . \qquad (4\text{-}2)$$

A state transition matrix corresponding to this characteristic equation is:

$$A = \begin{vmatrix} 0 & 0 & -0.32 \\ 1 & 0 & -0.64 \\ 0 & 1 & -0.5 \end{vmatrix}\ . \qquad (4\text{-}3)$$

The initial state vector for the model was arbitrarily defined to be:

$$X_0 = \begin{vmatrix} 1.0 & 1.0 & 1.0 \end{vmatrix}\ . \qquad (4\text{-}4)$$

An output matrix was selected to produce a scalar output:

$$C = \begin{vmatrix} 1.0 & 1.0 & 1.0 \end{vmatrix}\ . \qquad (4\text{-}5)$$

The measurements obtained from the model were derived by contaminating the true output of the model with a zero mean, gaussian noise sequence. The covariance of the measurement noise sequence was selected as one unit.

Simulation Procedure

This section will describe the details of the test procedure used to evaluate the performance of the adaptive Kalman filters. Their relative performance will be assessed by determining which algorithm can estimate the state of the model with the least cumulative mean square error. In the following discussion, θ_j, will represent the jth parameter to be identified by the adaptive Kalman filter.

Each adaptive Kalman filter will be required to identify the dynamics of the test model. The characteristic equation of the test model used to generate the noisy measurements was given in equation (4-2). Assume that the exact location of the poles is unknown. The imaginary poles are located at $z = \pm j\theta_1$, the real pole is located at $z = +\theta_2$. The characteristic equation corresponding to these unknown parameters is:

$$Z^3 + \theta_2 Z^2 + \theta_1^2 Z + \theta_1^2 \theta_2 = 0 . \qquad (4\text{-}6)$$

A state transition matrix corresponding to equation (4-6) is:

$$A(\theta) = \begin{vmatrix} 0 & 1 & -\theta_1^2\theta_2 \\ 1 & 0 & -\theta_1^2 \\ 0 & 0 & -\theta_2 \end{vmatrix} . \qquad (4\text{-}7)$$

The adaptive Kalman filters were also required to identify the covariance of the process noise used to excite each of the state equations. The process noise covariance matrix, in terms of unknown parameters is:

$$Q(\theta) = \begin{vmatrix} \theta_3 & 0 & 0 \\ 0 & \theta_4 & 0 \\ 0 & 0 & \theta_5 \end{vmatrix} . \tag{4-8}$$

This is a diagonal matrix because the noise sequences are assumed to be independent.

The output matrix was selected as a known quantity identical to equation (4-5). This is a realistic assumption since the output matrix in the transfer alignment problem is known. It is derived from the sensed angular velocities from the master and slave IMUs.

The measurement noise covariance matrix was selected as:

$$R = \begin{vmatrix} 1.0 \end{vmatrix} . \tag{4-9}$$

This describes the actual measurement noise used to corrupt the output of the model. Knowledge of the covariance of the measurement noise is also a realistic assumption since it is known in the transfer alignment problem. The measurement noise covariance is based on the accuracy of angular velocity information available from the master and slave IMUs.

The adaptive Kalman filters were required to identify two parameters from the state transition matrix, and three parameters from the process noise covariance matrix. For convenience, these parameters will be arranged in a vector. The true value of these parameters was:

$$\begin{vmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \end{vmatrix} = \begin{vmatrix} 0.8 \\ 0.5 \\ 1.0 \\ 1.0 \\ 1.0 \end{vmatrix} . \qquad (4\text{-}10)$$

To perform a fair comparison, the adaptive Kalman filters were initialized identically. The initial conditions were selected as follows.

1. Initial parameter vector estimate:

$$\begin{vmatrix} \hat{\theta}_1 \\ \hat{\theta}_2 \\ \hat{\theta}_3 \\ \hat{\theta}_4 \\ \hat{\theta}_5 \end{vmatrix} = \begin{vmatrix} 0.2 \\ 0.2 \\ 1.0 \\ 1.0 \\ 1.0 \end{vmatrix} . \qquad (4\text{-}11)$$

2. Initial state vector estimate:

$$\hat{X}_0 = \begin{vmatrix} 1.0 & 1.0 & 1.0 \end{vmatrix} . \qquad (4\text{-}12)$$

3. Initial error covariance matrix:

$$P_{0/0} = \begin{vmatrix} 10.0 & 0 & 0 \\ 0 & 10.0 & 0 \\ 0 & 0 & 10.0 \end{vmatrix} . \qquad (4\text{-}13)$$

4. Initial Hessian matrix:

$$R_0 = \begin{vmatrix} 1.0 & 0 & 0 & 0 & 0 \\ 0 & 1.0 & 0 & 0 & 0 \\ 0 & 0 & 1.0 & 0 & 0 \\ 0 & 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & 0 & 1.0 \end{vmatrix} . \qquad (4\text{-}14)$$

Note: The Hessian matrix is required only by the adaptive Kalman filter using the stochastic Newton algorithm to identify the parameters.

With this information, the test procedure to evaluate the relative performance of the adaptive Kalman filters may be presented. The model used to generate the noisy measurements and the adaptive Kalman filters were initialized as just described. The model was iterated to produce 1,000 noisy measurements. At every time interval the adaptive Kalman filters produced a new estimate of the unknown parameters and the state of the model from the noisy measurement. The cumulative mean square state estimation error was also computed at every time interval. This error was used to assess the relative performance of the adaptive Kalman filters.

The relative performance of the adaptive Kalman filters did not completely satisfy questions concerning their potential for use in the transfer alignment problem. An indication of the absolute performance of the adaptive Kalman filters was determined by making a comparison to a standard Kalman filter where the dynamics of the model and noise statistics were known exactly. This standard Kalman filter did not have the additional burden of identifying these quantities. Therefore, it should have set a lower bound on the cumulative mean square error in estimating the state. If the performance of an adaptive Kalman filter could not at least approach this lower bound, then it might be concluded that a better solution to the transfer alignment problem is needed.

It was also instructive to monitor the performance of a standard Kalman filter that used the initial estimates of the model dynamics and the noise statistics. This Kalman filter was incapable of adjusting these parameters to improve its performance. Therefore, it should have set an upper bound on the cumulative mean square error in estimating the state of the test model.

To summarize, both standard Kalman filters used a state transition and a process noise covariance matrix as given in equations (4-7) and (4-8). The standard Kalman filter with the exact model used the true parameters in these matrices given in equation (4-10). The standard Kalman filter with the inexact model used the initial estimates of these parameters given in equation (4-11). All remaining conditions were identical to those for the adaptive Kalman filters.

Preliminary Simulation Results

The performance of the Kalman filters is graphically illustrated on the first set of plots located in Appendix A. The plots will be divided into the following three categories for convenience.

1. Cumulative mean square error. The first three plots illustrate the cumulative mean square error in estimating the states of the model. There is one plot for each state of the model. The performance of all five Kalman filters is recorded on each plot.
2. Estimated process noise covariance matrix. The following three plots illustrate the estimates of the three parameters from the process noise covariance matrix. There is one plot for each of the adaptive Kalman filters.
3. Estimated state transition matrix. The last three plots illustrate the estimates of the two parameters from the state transition matrix. Again, there is one plot for each of the three adaptive Kalman filters.

The plots of cumulative mean square error give the best indication of the overall performance of the Kalman filters. The most significant finding is that the adaptive Kalman filter using the stochastic Newton approach to identify the parameters provided a high degree of accuracy in estimating the states of the system. The performance of this algorithm was almost as good as the standard Kalman filter using the exact model of the system dynamics and noise statistics. For this reason, this algorithm was recommended for the transfer alignment problem.

The adaptive Kalman filters relying on either variation of the stochastic gradient algorithm to identify the parameters did not perform well. However, the modified stochastic gradient algorithm did show an improvement over the original version. In some cases the Kalman filter using the inexact model of the system dynamics performed better than these two versions of the adaptive Kalman filter. These algorithms are not considered to be reliable enough to be used in the transfer alignment problem. Some insight about the performance of the adaptive Kalman filters can be gained by examining the parameter estimates generated by these algorithms. The parameter estimates may not necessarily converge to their true values. This can happen when there is a local minimum associated with the cost function around which the algorithm was developed [9, pp. 195-96]. The parameter estimation process can then stagnate at estimates satisfying that local minimum.

It is difficult to make conclusions based on the plots of the estimated parameters from the process noise covariance matrix. A problem with the adaptive Kalman filter using the stochastic gradient algorithm is that it severely underestimates the process noise covariance associated with the first state of the system. This appears to be improved by the adaptive Kalman filter using the modified stochastic gradient algorithm. The estimates of these parameters generally deviate from their true values. This is difficult to explain, except that the convergence properties of these estimates might be governed by a local minimum of the appropriate cost function.

The plots of the parameters associated with the state transition matrix are more revealing. The parameter estimates generated by the adaptive Kalman filter using the stochastic gradient algorithm showed a great deal of erratic behavior. This algorithm has very poor convergence properties. The parameter estimates of the modified version of this algorithm are less erratic, but more biased. The parameter estimates produced by the adaptive Kalman filter using the stochastic Newton algorithm, by contrast, are very good. These estimates converge roughly to their true values and display almost no erratic behavior.

The results of this simulation clearly indicate that the adaptive Kalman filter based on the stochastic Newton algorithm performs very well. This algorithm is recommended for use in the transfer alignment problem. The other algorithms perform poorly by comparison.

Simulation Procedure Based on Using a Modified Process Noise Covariance Matrix

The motivation for the second simulation concerns divergence problems in Kalman filters [4, p. 279]. In practice, models of dynamic systems are never known exactly. Modeling error can be introduced by many sources. Computer round-off error is one example. Modeling error causes the performance of a Kalman filter to be degraded from what would theoretically be expected. This is called "apparent divergence." A Kalman filter that displays an apparent divergence problem is said to be suboptimal. Another type of divergence problem is referred to as "true divergence." A Kalman filter affected by a true divergence problem will produce state estimates that have a mean square error that grows indefinitely. This Kalman filter is said to be unstable.

There are several solutions to reduce the severity of divergence problems in Kalman filters. An attractive technique is to add fictitious process noise to the Kalman filter [4, p. 279]. This forces the

Kalman gain to converge in such a manner that the filter places more emphasis on recently acquired measurements, than it would otherwise.

The second set of simulations was done to investigate the effects of adding artificial process noise to the standard and adaptive Kalman filters. The initial conditions and assumptions used in this simulation were identical to those used in the first simulation, except as discussed below.

In the case of the standard Kalman filters, the process noise covariance matrix was selected as:

$$Q = \begin{vmatrix} 10.0 & 0 & 0 \\ 0 & 10.0 & 0 \\ 0 & 0 & 10.0 \end{vmatrix} . \tag{4-15}$$

The difference between the Kalman filters with the exact and inexact model was, as in the last simulation, only in the state transition matrix.

Each adaptive Kalman filter was again required to estimate the covariance of the process noise used to excite each state equation. The form of the resulting process noise covariance matrix was given in equation (4-8). The true value of these parameters was unity. If the adaptive Kalman filters estimated these parameters correctly, then all five Kalman filters should have used a numerically equivalent process noise covariance matrix. To accomplish this, each of the estimated process noise covariance matrix parameters were increased by a value of 9.0. Assuming the adaptive Kalman filters correctly estimated the parameters from the process noise covariance matrix, the actual process noise covariance matrix used would be identical to equation (4-15). This allowed a fair comparison of the performance of all the Kalman filters using a modified process noise covariance matrix to be made.

The remaining conditions and the test procedure were identical to those discussed for the first simulation. The results of this simulation will be discussed in the following section.

Simulation Results Based on Using a Modified Process Noise Covariance Matrix

The performance of the Kalman filters in this simulation is graphically illustrated on a second set of plots in Appendix B. The plots are categorized in an identical manner as those from the first simulation.

The overall performance of the adaptive Kalman filters is best illustrated by the plots of cumulative mean square error. Again, the adaptive Kalman filter using the stochastic Newton parameter identification algorithm provided superior results. The performance of this version of the adaptive Kalman filter, and the standard Kalman filter with the exact model, were essentially unaffected by using the modified process noise covariance matrix.

The performance of the adaptive kalman filters based on either variation of the stochastic gradient algorithm was generally improved by using the modified process noise covariance matrix. However, they still were inadequate when compared to the adaptive Kalman filter based on the stochastic Newton algorithm. The adaptive Kalman filter using the modified stochastic gradient algorithm showed a more dramatic improvement than the original version.

The use of the modified process noise covariance matrix had the most significant effect on the accuracy of the parameters describing the covariance of the process noise used to excite the model. This was true for all versions of the adaptive Kalman filter. The coefficients in these matrices remained closer to their true initial estimates than in the previous simulation.

The use of the modified process noise covariance matrix also improved the quality of the estimated parameters in the state transition matrix. Again, this was noted for all three adaptive Kalman filters. The parameter estimates of the state transition matrix produced by the stochastic gradient algorithm were as erratic as in the first simulation. The modified stochastic gradient algorithm did not display this erratic behavior. The stochastic Newton algorithm produced very accurate estimates of these parameters.

Several observations were made about the effect of adding artificial process noise to the adaptive Kalman filter algorithms. The accuracy of the parameter estimates improved noticeably. As a result, the quality of the state estimates for the adaptive Kalman filters based on either the original or modified stochastic gradient algorithm were typically improved. However, there was essentially no improvement in the state estimates of the adaptive Kalman filter based on the stochastic Newton algorithm.

It is interesting to consider under what conditions the accuracy of the parameter estimates improve, without an improvement in the accuracy of the state estimates. Increasing the process noise covariance matrix generally improves the quality of the parameter estimates. It also causes the state estimates to be influenced more by recent measurements. This is desirable if unmodeled system dynamics are expected. This would also be desirable if the adaptive Kalman filter is slow or erratic in identifying the parameters. If the unmodeled dynamics are not significant, or the adaptive Kalman filter is efficient in identifying the parameters, there will be a trade-off in adding artificial process noise. On one hand, the parameter estimates tend to be improved. On the other, the filter becomes more sensitive to the adverse effects of measurement noise. These combined effects influence the quality of the state estimates. This explains why the state estimates of the adaptive Kalman filters based on either variation of the stochastic gradient algorithm benefited from adding artificial process noise, while the adaptive Kalman filter based on the stochastic Newton algorithm did not.

Simulation Software

The software for the simulations used in this study was written in Forth. Forth is an unusual computer language, when compared with more traditional languages such as Fortran. Forth has generally been favored by hardware vendors and process control experts [16, p. 303]. Forth is widely used in many industrial applications [1, p. 3]. One example is in the field of robotics. Using Forth for a computer simulation tends to break with well established traditions. Generally, such simulations are written in more customary languages such as Fortran. The intent of this section is to supply a justification for using Forth for the simulation.

Forth is a threaded interpretive language [16, p. 303]. A Forth system consists of a nucleus of basic definitions. As an example, some of these definitions cause arithmetic operations to take place between two numbers on a stack. The user can define a new set of definitions in terms of the basic definitions from the nucleus. The programmer now has an enlarged set of definitions available for use. This process can be repeated as often as necessary to develop a high level language customized for a particular application. This characteristic of allowing the programmer to add new features to the existing language is called extensibility [1, p. 28]. Forth is sometimes called a meta-application language because it can be used to create other problem oriented languages [1, p. 4].

Forth has not gained wide acceptance outside the few applications previously discussed. Part of the reason is that Forth has been implemented as a fixed point system. This is obviously satisfactory for the limited applications for which it was intended. Fixed point systems generally provide a speed advantage over floating point systems. Recently, several vendors have developed floating point packages for Forth. These floating point options are either written in software or written so as to accommodate a floating point math coprocessor. The hardware floating point system offers a substantial speed advantage over the software floating point version.

The decision to use Forth to develop the software for the simulation in this study was based on two factors. First was the availability of hardware floating point support for Forth. Second, the extensible nature of Forth allowed a customized language, convenient for writing a simulation program, to be developed. The simulation program is provided in Appendix C.

A high level set of definitions was created through which the simulation program was conveniently written. These definitions include a complete set of basic matrix operations. Screens 76 through 96 in the program contain names of definitions which can operate on sets of data in a matrix or vector form. They were written so that they could be easily used. For example, assume two matrices have already been defined. It is desired to multiply these two matrices and place the result in a third matrix. This is accomplished by listing, in order, the name of the first matrix, the second matrix, the destination matrix, and then the definition for matrix multiplication. The programmer does not need to worry about the dimensions of the matrices as long as they conform. The destination matrix will be properly dimensioned based on the two source matrices. The other matrix operations are just as easy to use. Being able to create a convenient high level set of definitions simplified the overall programming effort.

The remainder of the program is devoted to implementing the Kalman filter algorithms. The equations illustrated on the flowcharts in

Chapter 3 were written using the high level set of matrix definitions. With Forth, the complexity of developing the programs for these algorithms was considerably simpler than what it might have been with other high level languages. Writing definitions to implement long equations became no more complicated than to simply write the equations down. It was for this reason that Forth was selected as the language in which to write the simulation program. Forth should be given more attention for applications requiring floating point arithmetic, such as simulations. The overall convenience of Forth was very impressive.

CHAPTER 5

CONCLUSION

This study has discussed the alignment of inertially guided weapons attached to the wings of an aircraft with a flexible body. The slave IMU of the weapon must be aligned with the master IMU of the aircraft, prior to launching the weapon. One approach to estimating the alignment error between the slave and master IMU starts with developing an accurate model for the flexure dynamics of the aircraft. The angles of alignment error can then be estimated using a standard Kalman. Unfortunately, developing such a model is a complicated task. These complications arise from the large number of time varying factors which can affect the flexure dynamics of the aircraft. For example, the changing payload conditions within the aircraft, and the varying turbulence conditions in the aircrafts environment, are just two factors that would have to be considered.

This study has proposed an alternative solution to the transfer alignment problem. This solution starts with using a more general model for the flexure dynamics of the aircraft. An adaptive Kalman filter can then be used to concurrently identify the dynamics of this model and the angles of alignment error. This appears to be a practical solution to the transfer alignment problem. Three adaptive Kalman filters were presented as candidates for use in this problem. They were simulated to determine their relative performance. The simulation was based on a relatively simple dynamic system. The adaptive Kalman filters were required to identify the dynamics of the model and the statistics of the noise used to excite the model. The first adaptive Kalman filter used a stochastic gradient algorithm to identify the unknown parameters. The performance of this adaptive Kalman filter was very poor. It was modified in an attempt to improve its performance. The modified algorithm did show an improvement, but its performance was still inadequate. The third adaptive Kalman filter used a stochastic Newton parameter identification algorithm. This version of the adaptive Kalman filter performed very well in the simulation. In fact, it performed almost as well as a standard Kalman filter with an exact model of the system dynamics and the noise statistics.

The results of this study indicate that the adaptive Kalman filter based on the stochastic Newton parameter identification algorithm should receive serious consideration for the transfer alignment problem. The use of this algorithm, together with a model such as the one presented in this study, provide a relatively simple and effective approach to computing the transfer alignment for weapons attached to the wings of an aircraft with a flexible body.

LIST OF REFERENCES

LIST OF REFERENCES

1. Brodie, L., Starting Forth, Prentice-Hall, Englewood Cliffs, N.J., 1981.

2. Brown, R. G., Introduction to Random Signal Analysis and Kalman Filtering, John Wiley, New York, N.Y., 1983.

3. El-Fattah, Y. M., "Recursive Self-Tuning Algorithm for Adaptive Kalman Filtering," Proc. IEE 130, 341-344, 1983.

4. Franklin, G. F. and Powell, J. D., Digital Control of Dynamic Systems, Addison-Wesley, Reading, Mass., 1980.

5. Gelb, A., Applied Optimal Estimation, The M.I.T. Press, Cambridge, Mass., 1974.

6. Goodwin, G. C. and Sin, K. S., Adaptive Filtering Prediction and Control, Prentice-Hall, Englewood Cliffs, N.J., 1984.

7. Hung, J. C., "Transfer Alignment for Missile Initialization," The Design and Analysis of Inertial System Operating Modes, Poly-Analytics, Inc., Knoxville, Tn., 1984.

8. Jazwinski, A., Stochastic Processes and Filtering Theory, Academic Press, New York, N.Y., 1970.

9. Ljung, L. and Soderstrom T., Theory and Practice of Recursive Identification, The M.I.T. Press, Cambridge, Mass., 1983.

10. Ljung, L., "Asymptotic Behavior of the Exended Kalman Filter as a Parameter Estimator for Linear Systems," IEEE Trans. on Aut. Contr. AC-24(1), 36-50, 1979.

11. Maybeck, P. S., Stochastic Models, Estimation and Control, Vol 2, Academic Press, New York, N.Y., 1979.

12. Peebles, P. Z., Probability, Random Variables, and Random Signal Priniciples, McGraw-Hill, New York, N.Y., 1980.

13. Schneider, A. M., "Kalman Filter Formulations for Transfer Alignment of Strapdown Inertial Units," Navigation 30(1), 72-88, 1983.

14. Schweppe, F. C., Uncertain Dynamic Systems, Prentice Hall, Englewood Cliffs, N.J., 1973.

15. Sinha, N. K. and Kuszta, B., Modeling and Identification of Dynamic Systems, Van Nostrand Reinhold, New York, N.Y., 1983.

16. Tello, E., "PolyForth and PC/Forth," Byte, 9(12), 303-14, 1984.

APPENDICES

APPENDIX A

PLOTS OF PRELIMINARY SIMULATION RESULTS

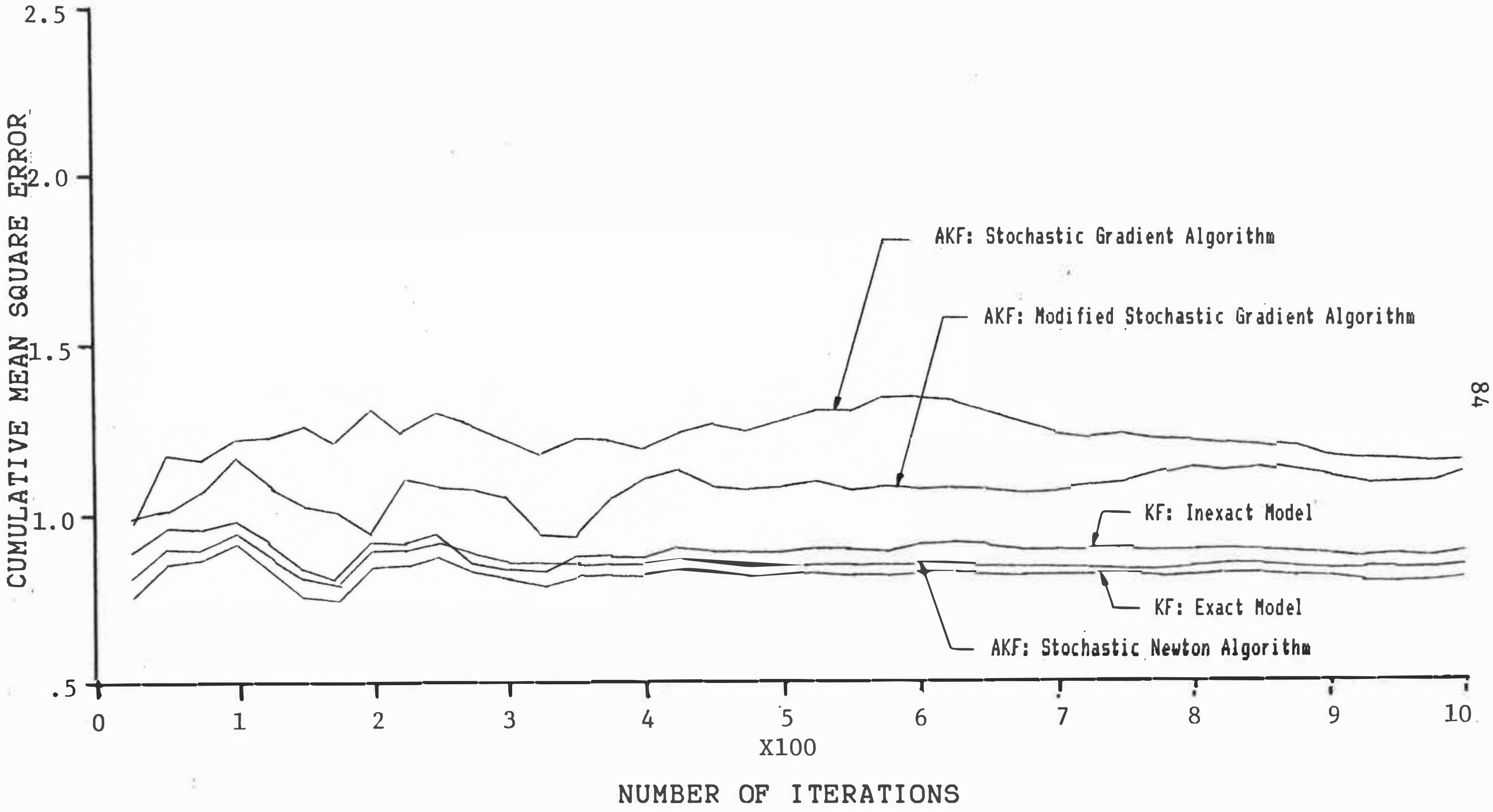


Figure 8. Cumulative Mean Square Error in Estimating State X1; Preliminary Simulation Results.

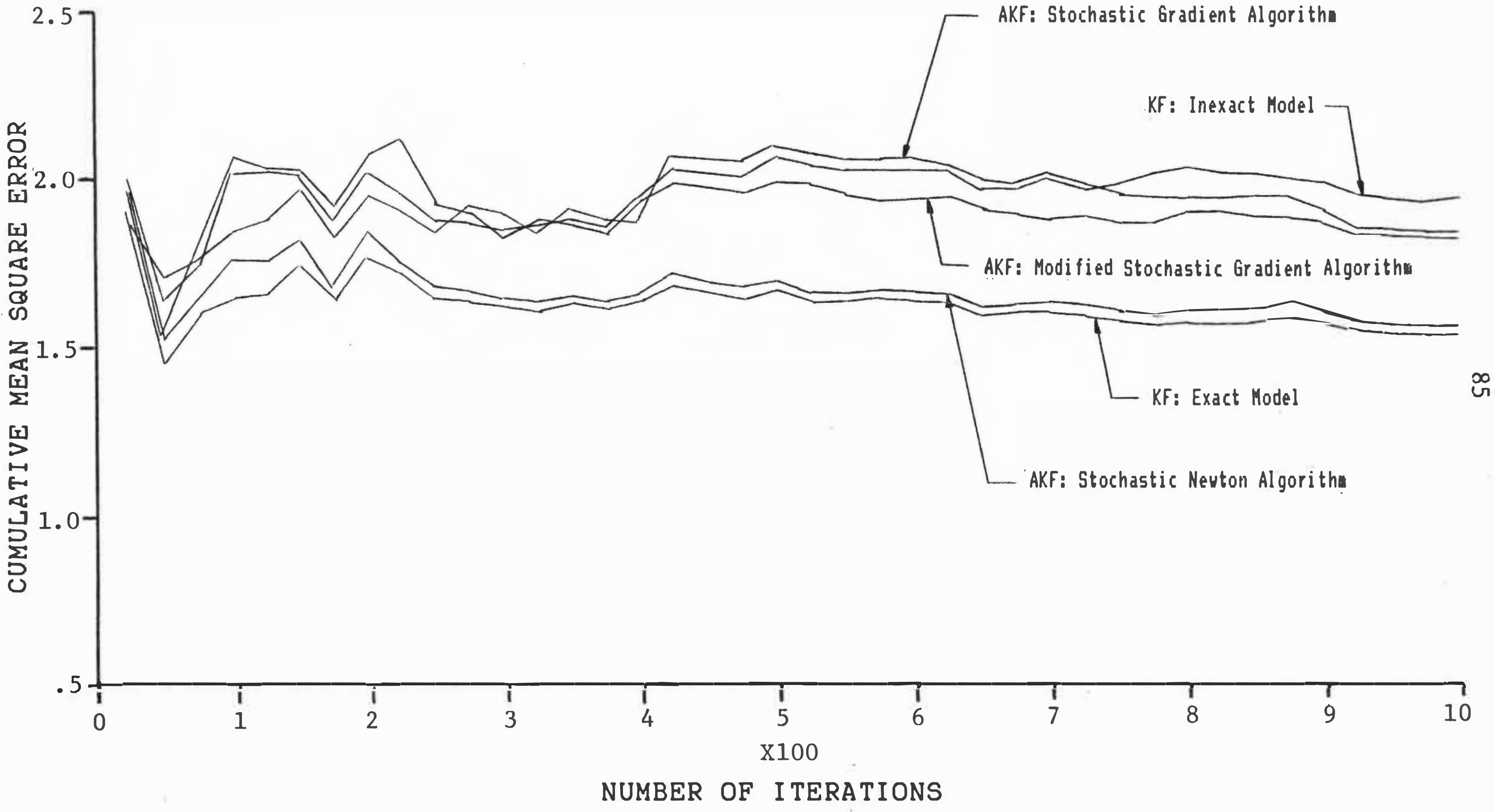


Figure 9. Cumulative Mean Square Error in Estimating State X2; Preliminary Simulation Results.

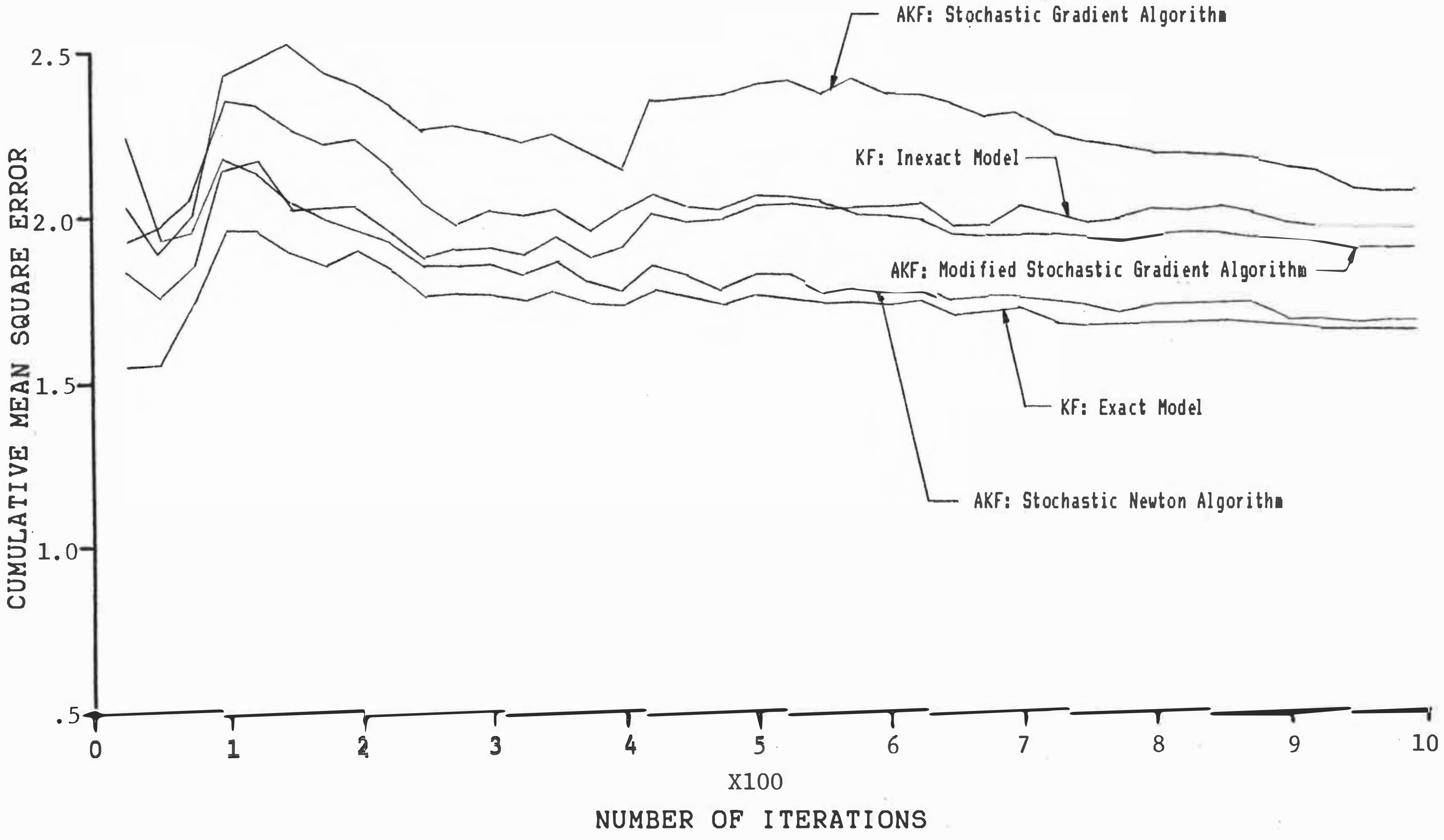


Figure 10. Cumulative Mean Square Error in Estimating State X3; Preliminary Simulation Results.

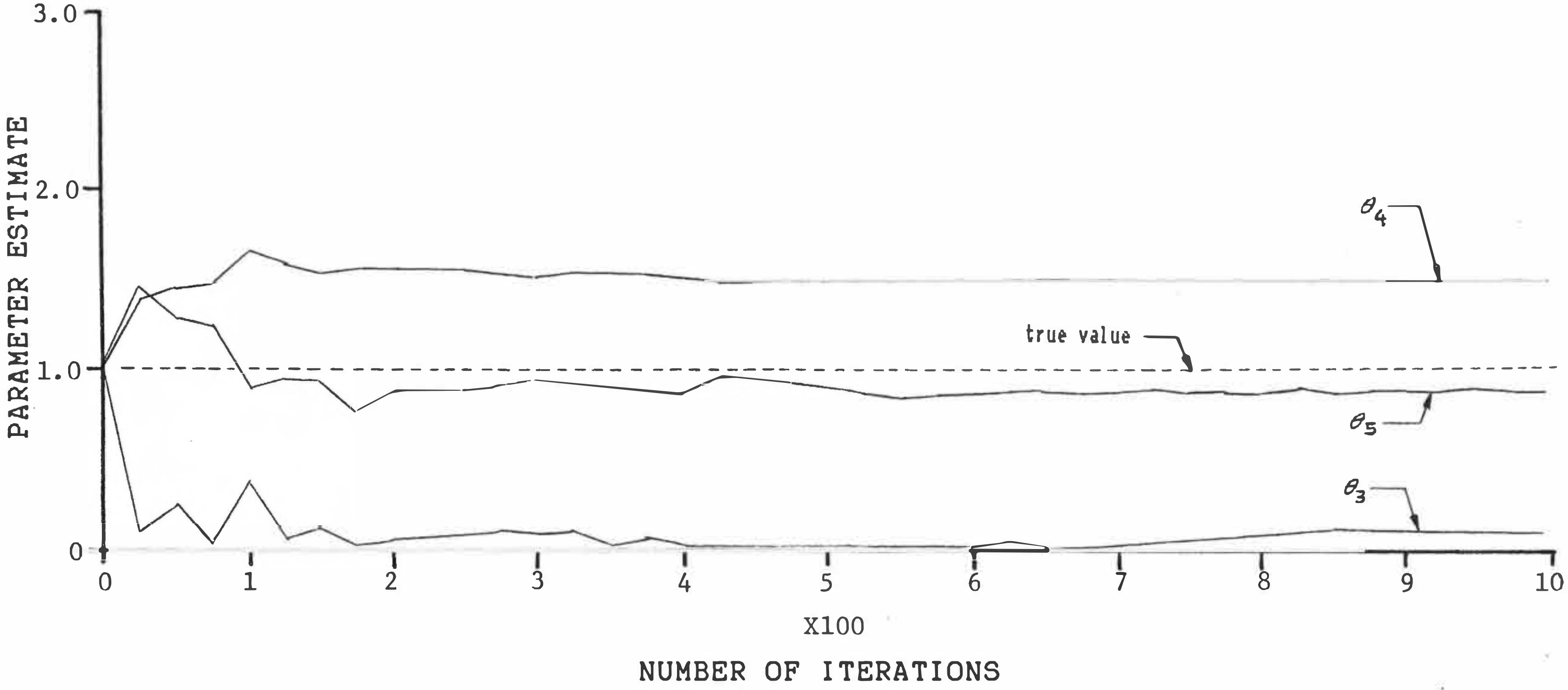


Figure 11. Estimated Parameters from the Process Noise Covariance Matrix, by the Adaptive Kalman Filter Based on the Stochastic Gradient Algorithm; Preliminary Simulation Results.

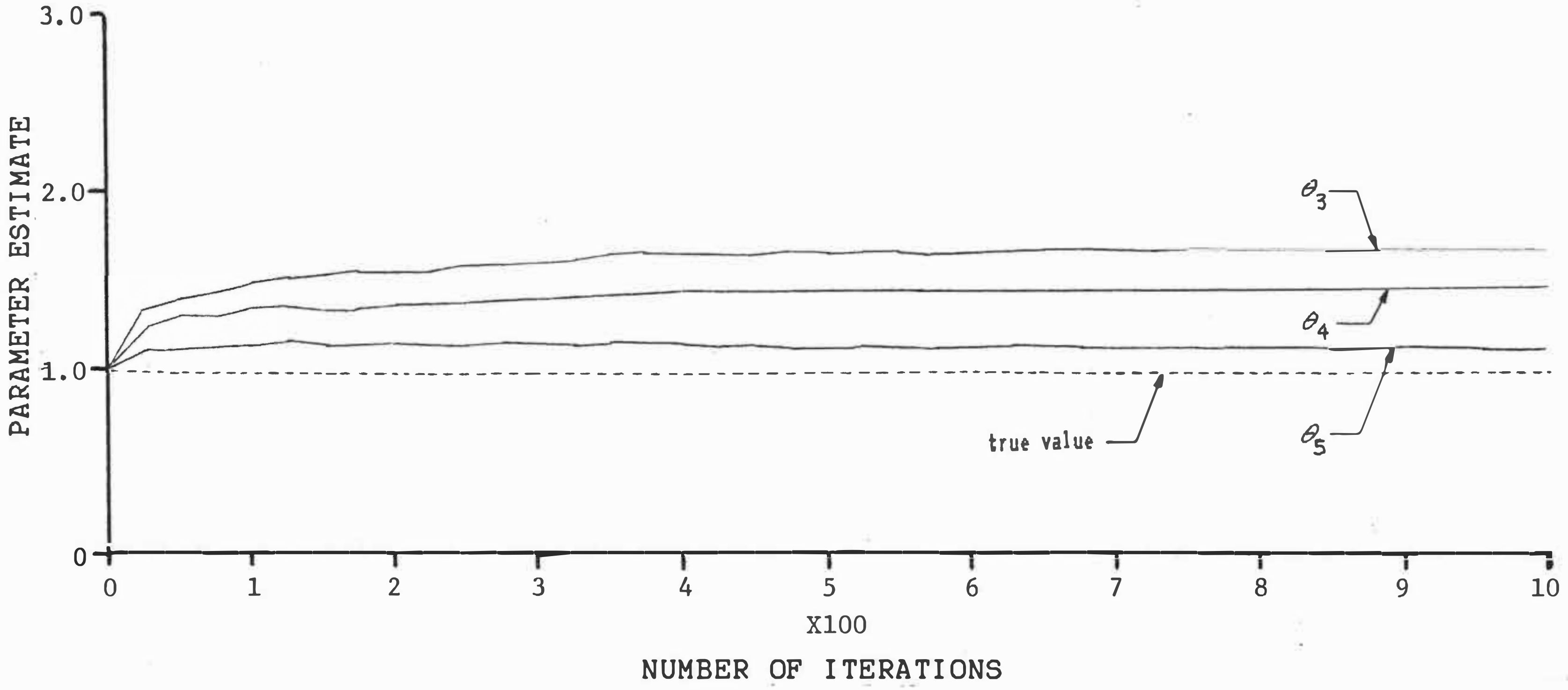


Figure 12. Estimated Parameters from the Process Noise Covariance Matrix, by the Adaptive Kalman Filter Based on the Modified Stochastic Gradient Algorithm; Preliminary Simulation Results.

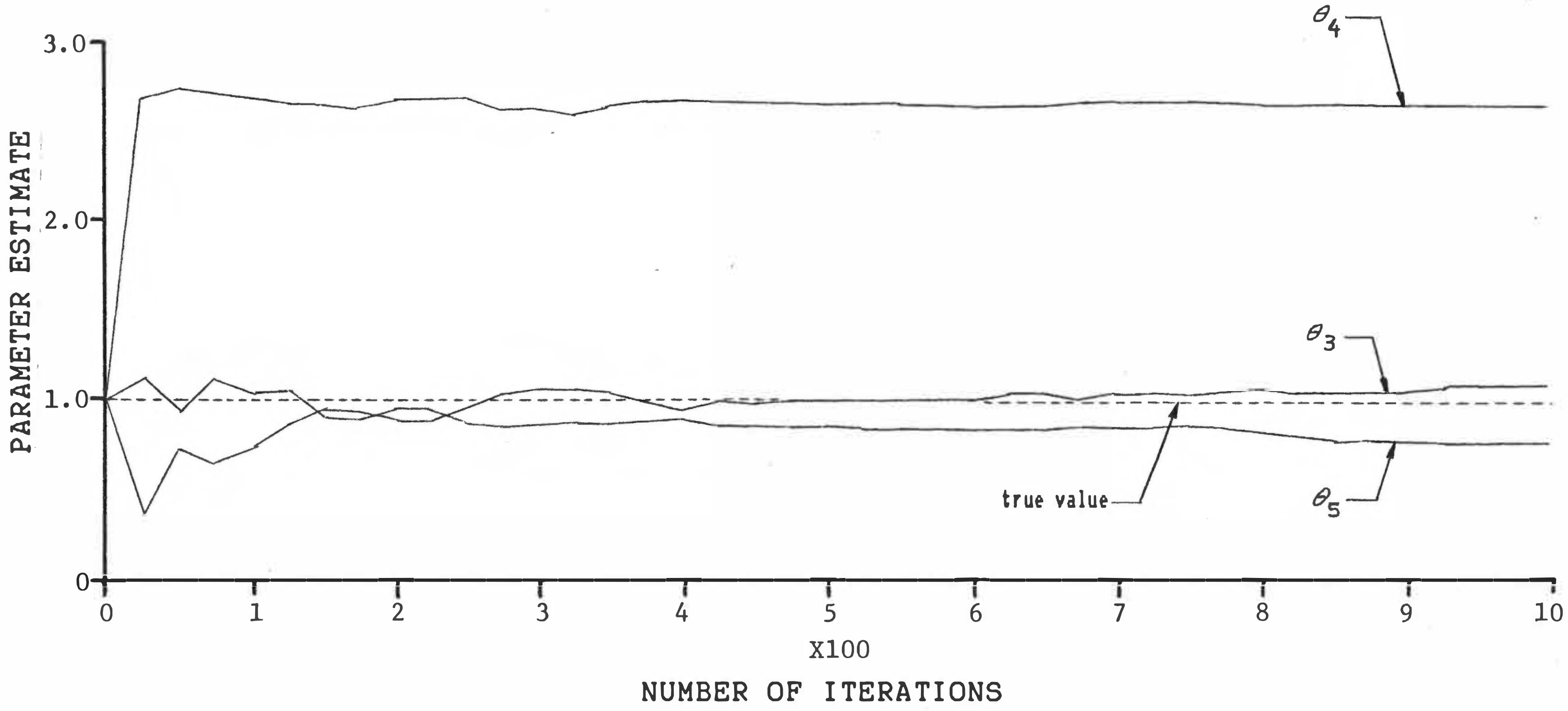


Figure 13. Estimated Parameters from the Process Noise Covariance Matrix, by the Adaptive Kalman Filter Based on the Stochastic Newton Algorithm; Preliminary Simulation Results.

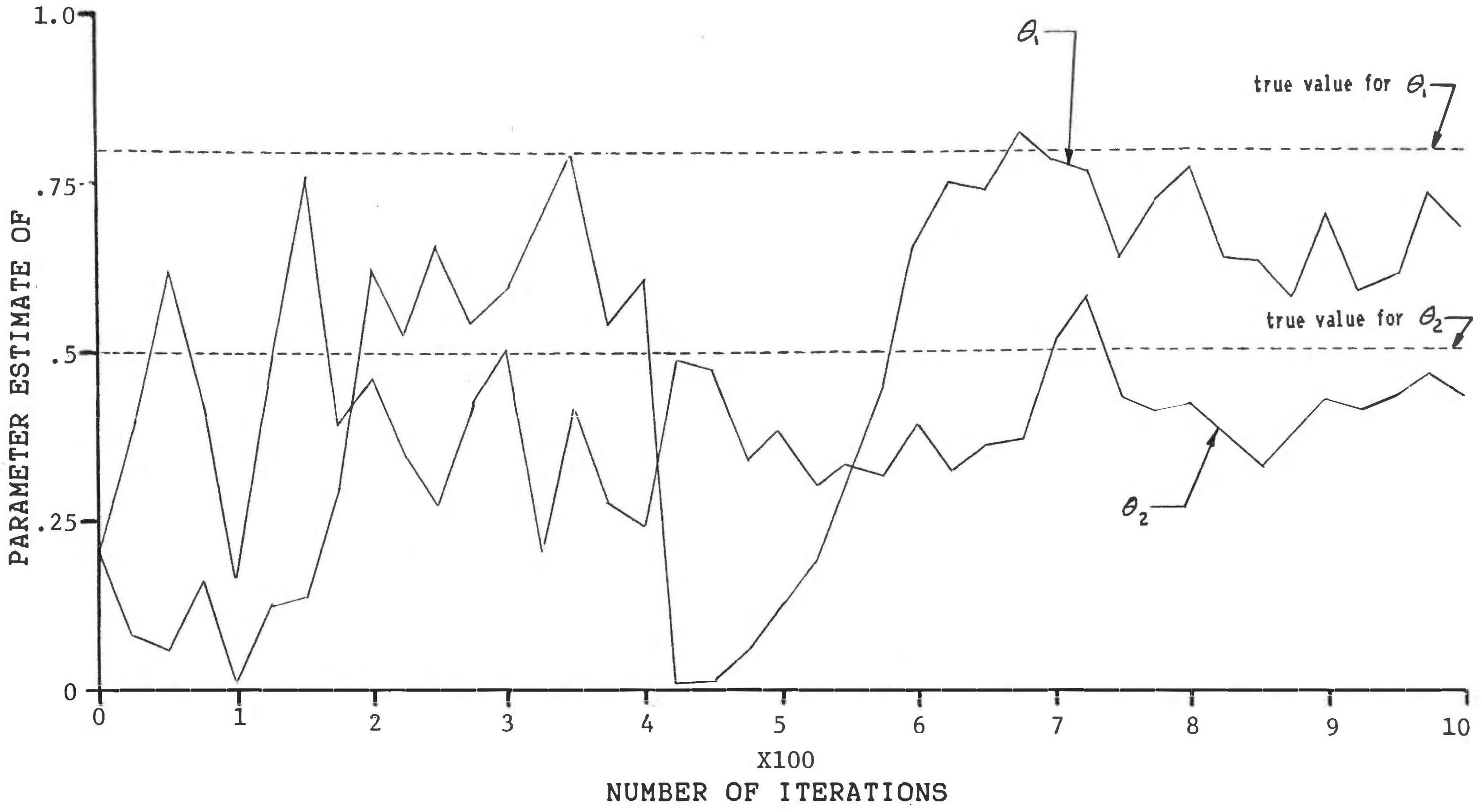


Figure 14. Estimated Parameters from the State Transition Matrix, by the Adaptive Kalman Filter Based on the Stochastic Gradient Algorithm; Preliminary Simulation Results.

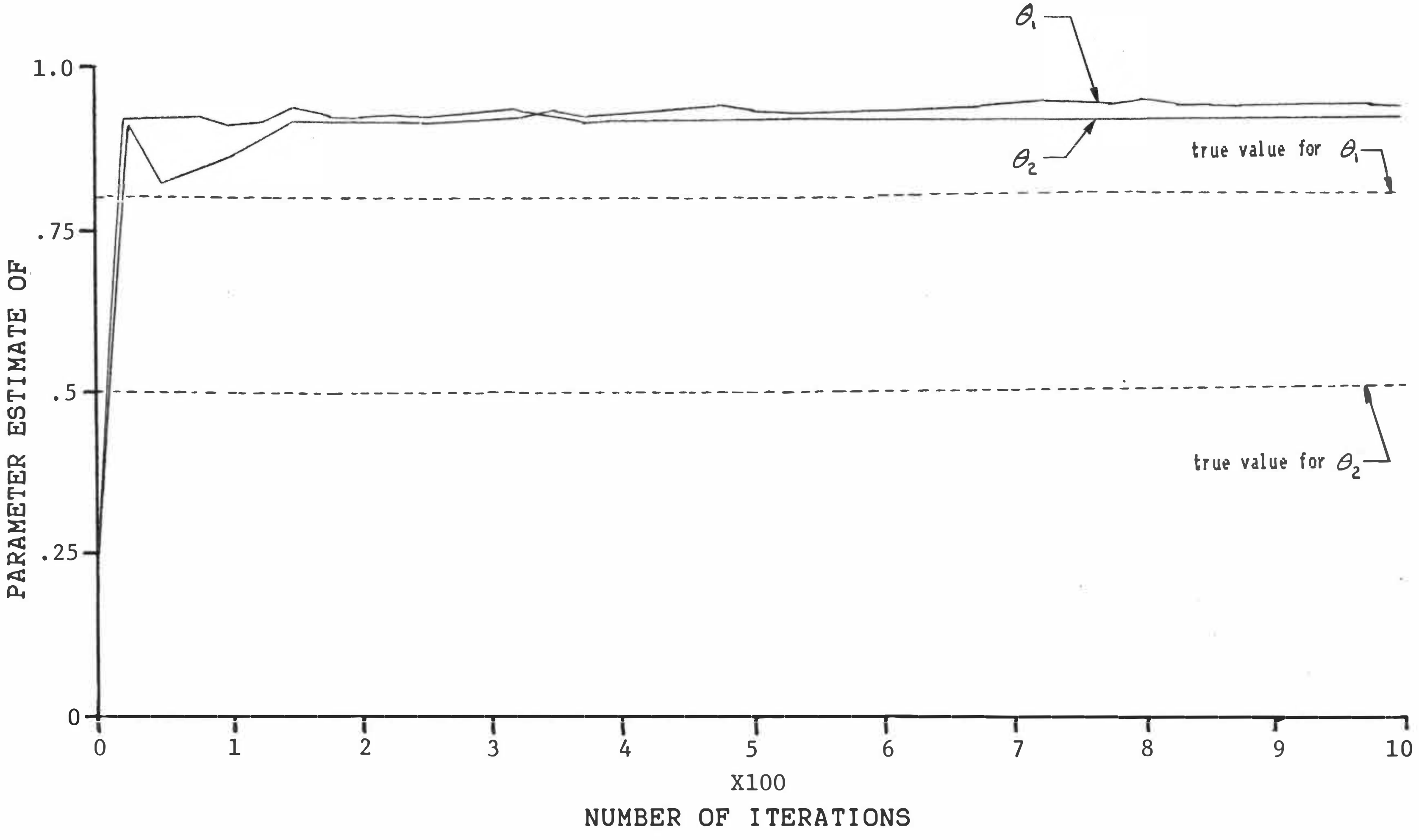


Figure 15. Estimated Parameters from the State Transition Matrix, by the Adaptive Kalman Filter Based on the Modified Stochastic Gradient Algorithm; Preliminary Simulation Results.

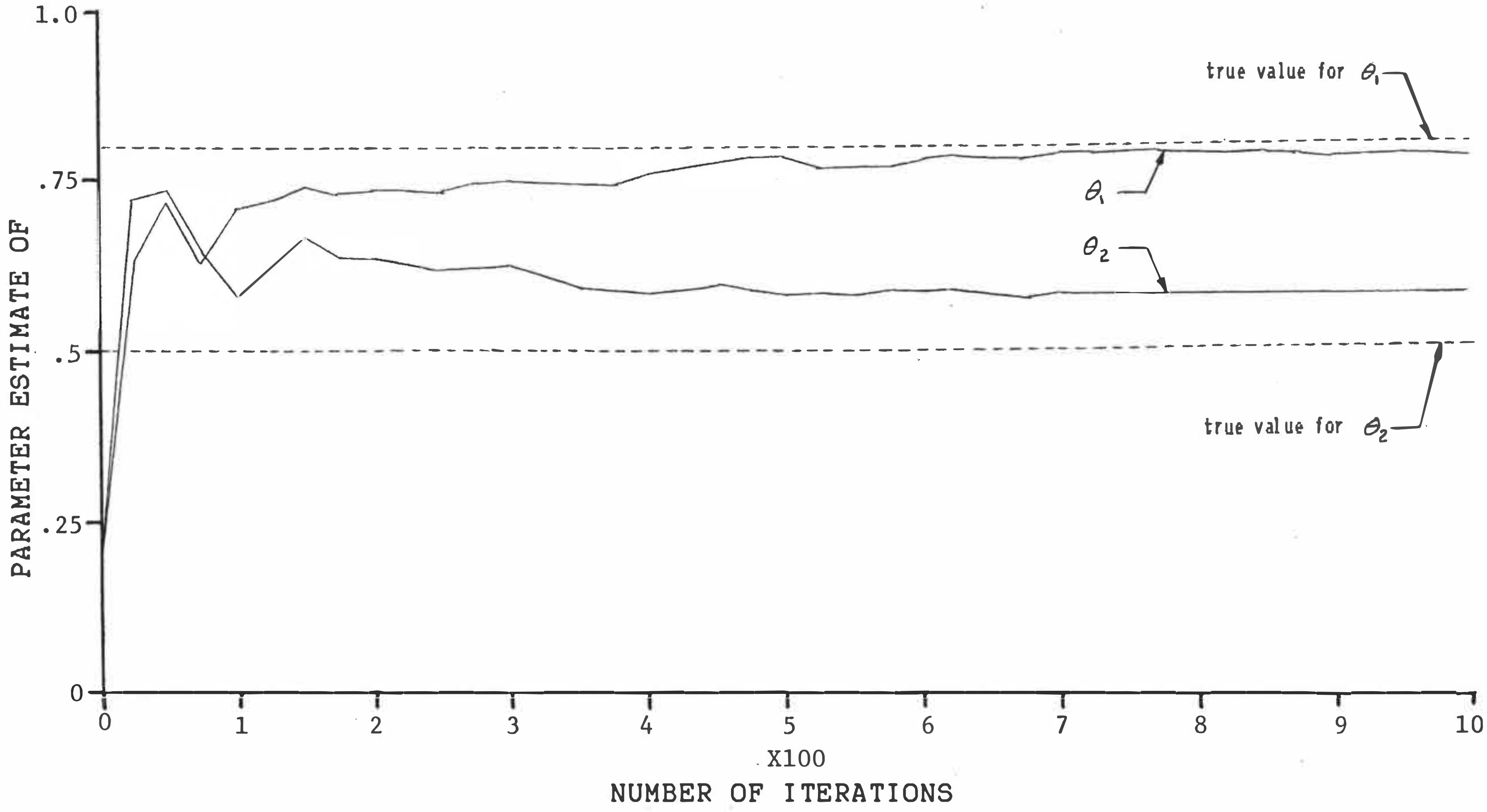


Figure 16. Estimated Parameters from the State Transition Matrix, by the Adaptive Kalman Filter Based on the Stochastic Newton Algorithm; Preliminary Simulation Results.

APPENDIX B

PLOTS OF SIMULATION RESULTS BASED ON USING A MODIFIED PROCESS NOISE COVARIANCE MATRIX

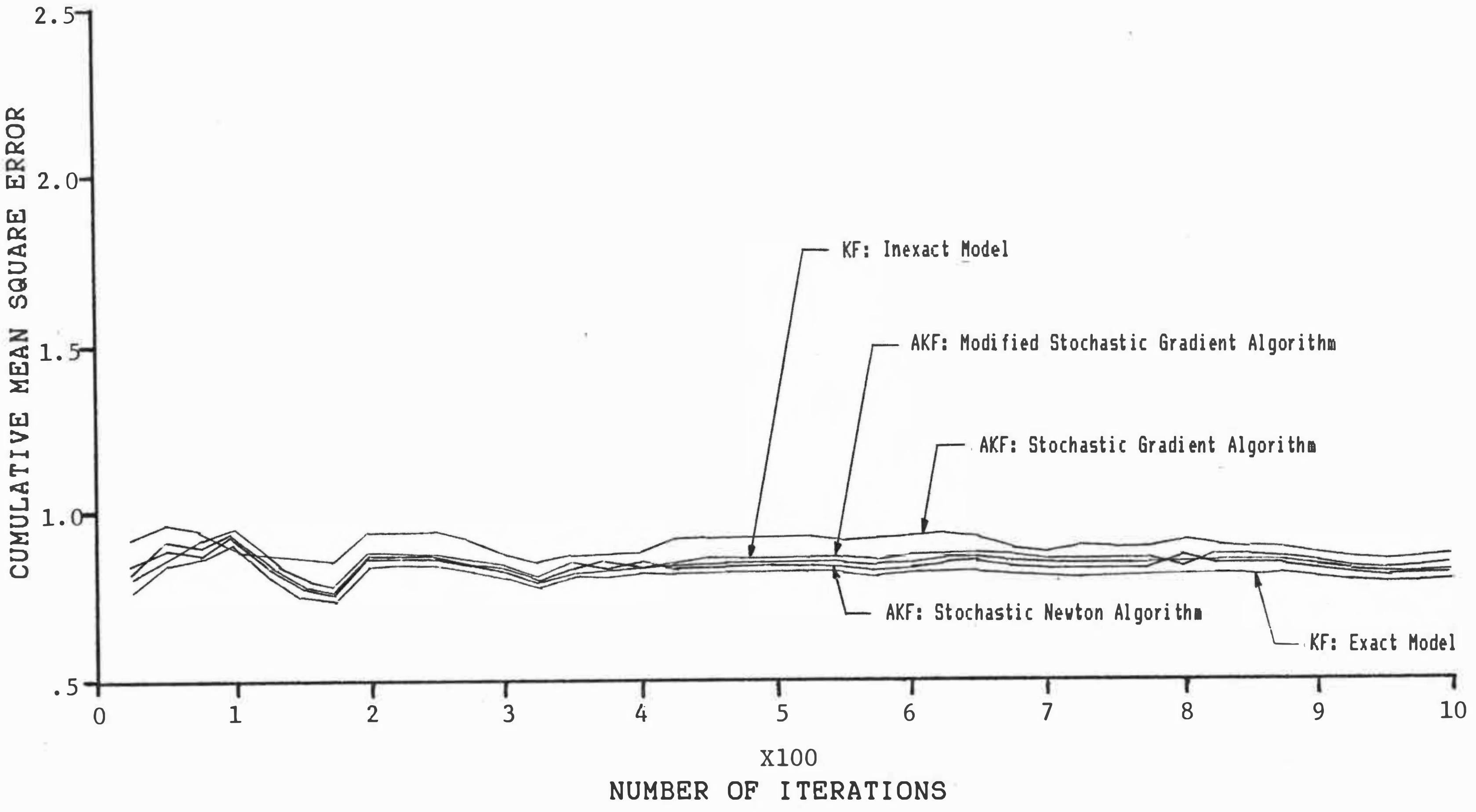


Figure 17. Cumulative Mean Square Error in Estimating State X1; Simulation Results Based on Using a Modified Process Noise Covariance Matrix.

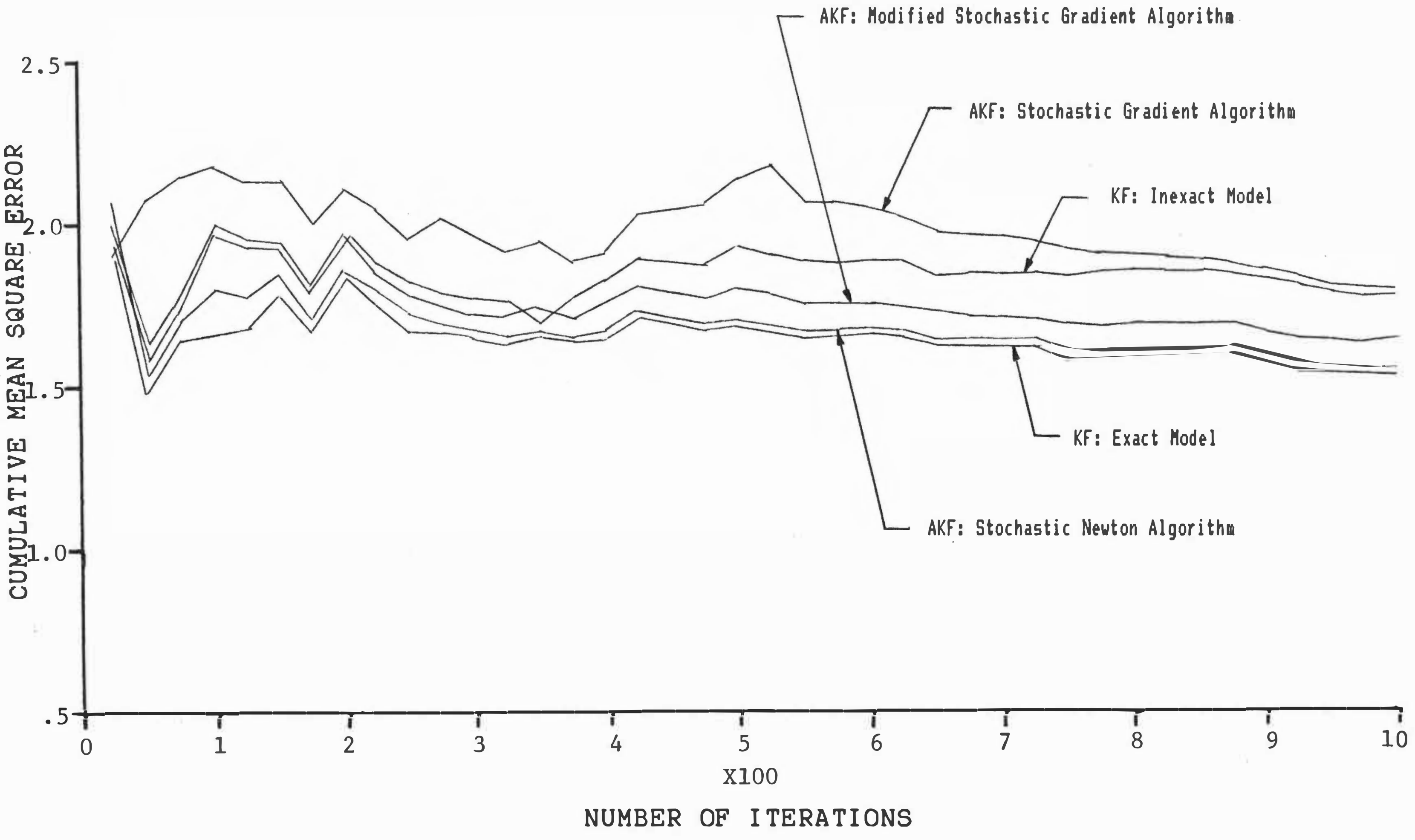


Figure 18. Cumulative Mean Square Error in Estimating State X2; Simulation Results Based on Using a Modified Process Noise Covariance Matrix.

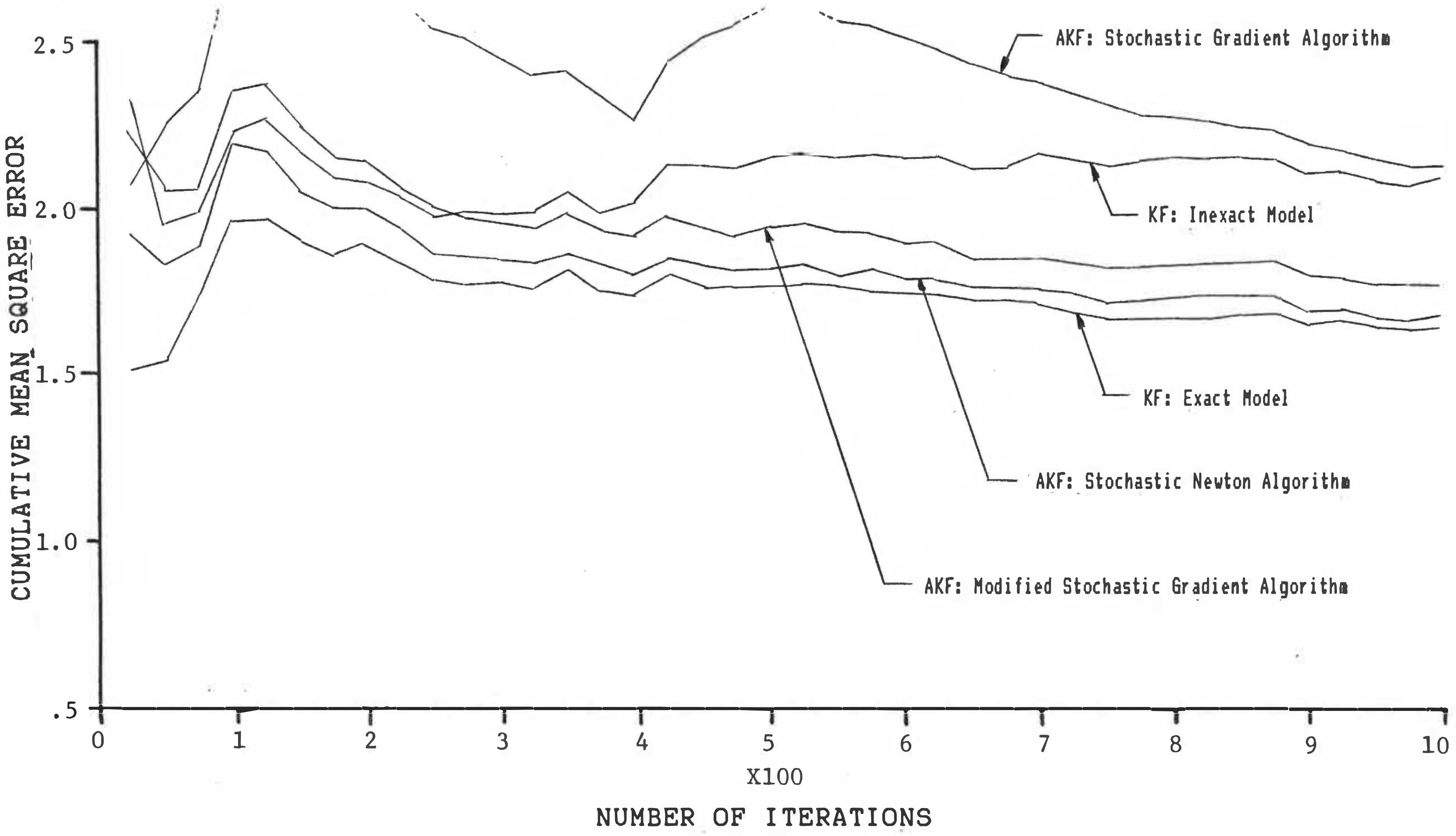


Figure 19. Cumulative Mean Square Error in Estimating State X3; Simulation Results Based on Using a Modified Process Noise Covariance Matrix.

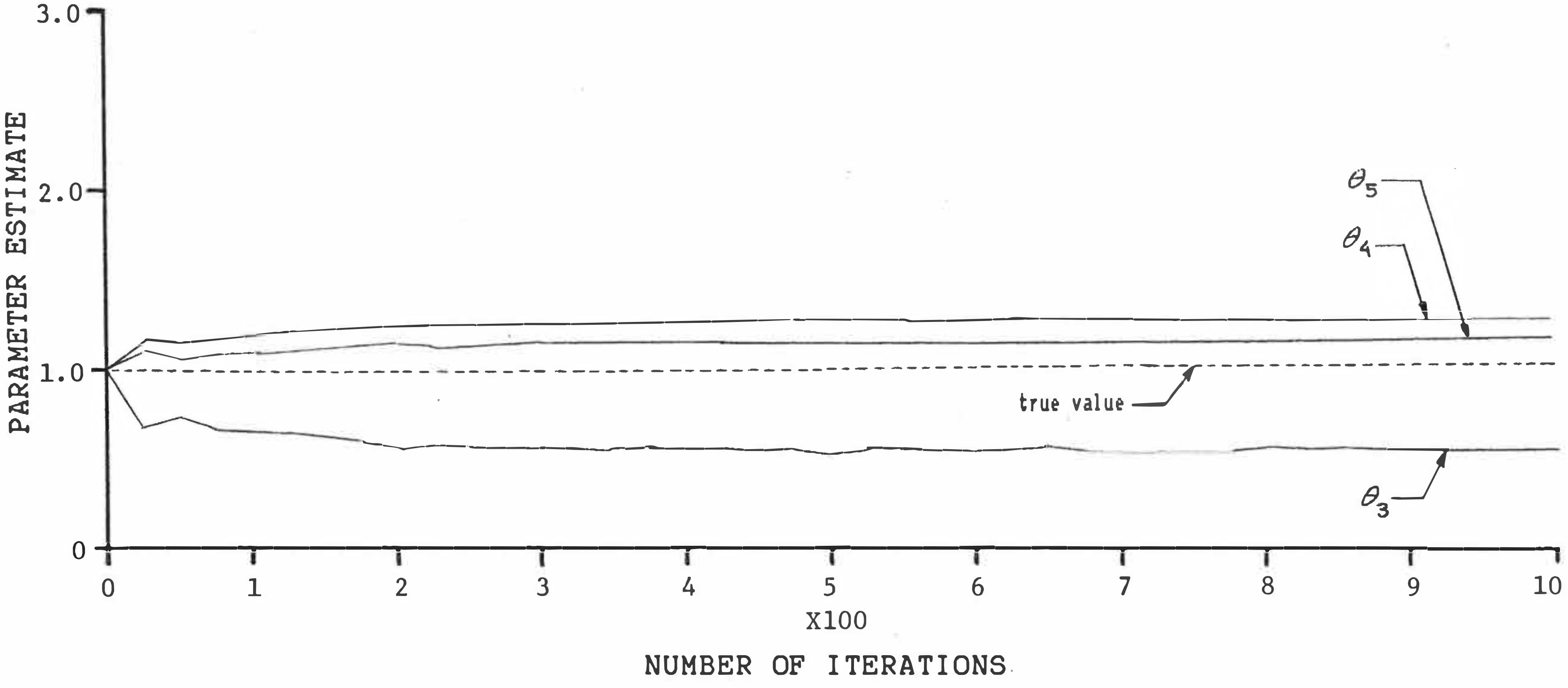


Figure 20. Estimated Parameters from the Process Noise Covariance Matrix, by the Adaptive Kalman Filter Based on the Stochastic Gradient Algorithm; Simulation Results Based on Using a Modified Process Noise Covariance Matrix.

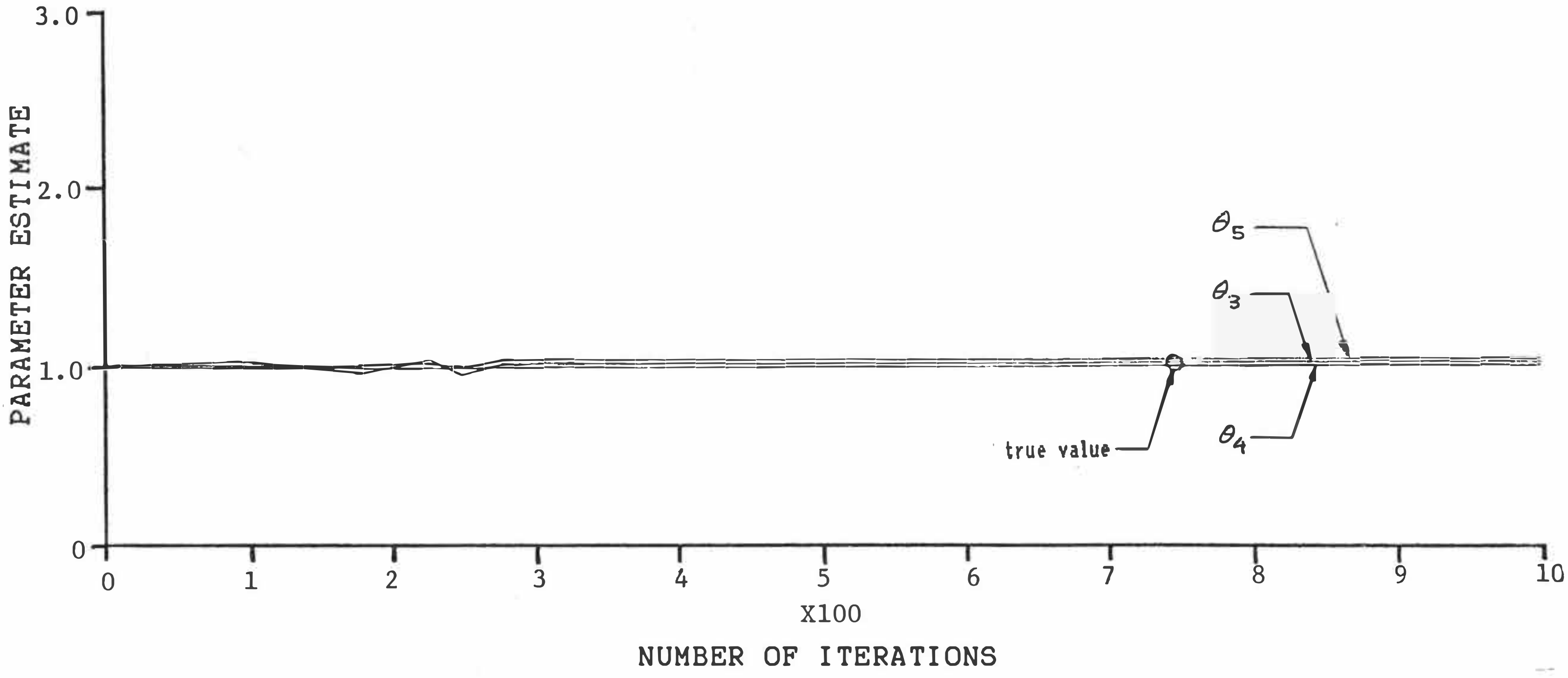


Figure 21. Estimated Parameters from the Process Noise Covariance Matrix, by the Adaptive Kalman Filter Based on the Modified Stochastic Gradient Algorithm; Simulation Results Based on Using a Modified Process Noise Covariance Matrix.

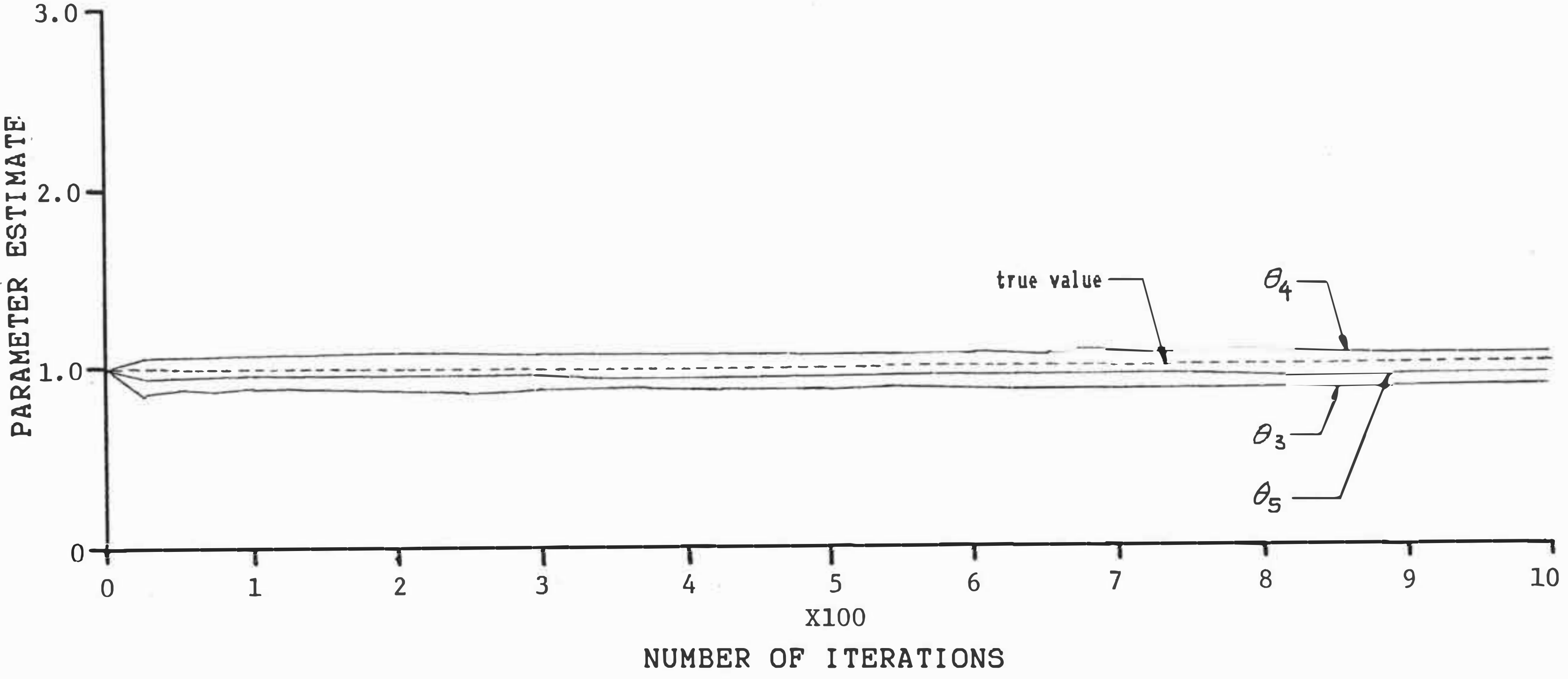


Figure 22. Estimated Parameters from the Process Noise Covariance Matrix, by the Adaptive Kalman Filter Based on the Stochastic Newton Algorithm; Simulation Results Based on Using a Modified Process Noise Covariance Matrix.

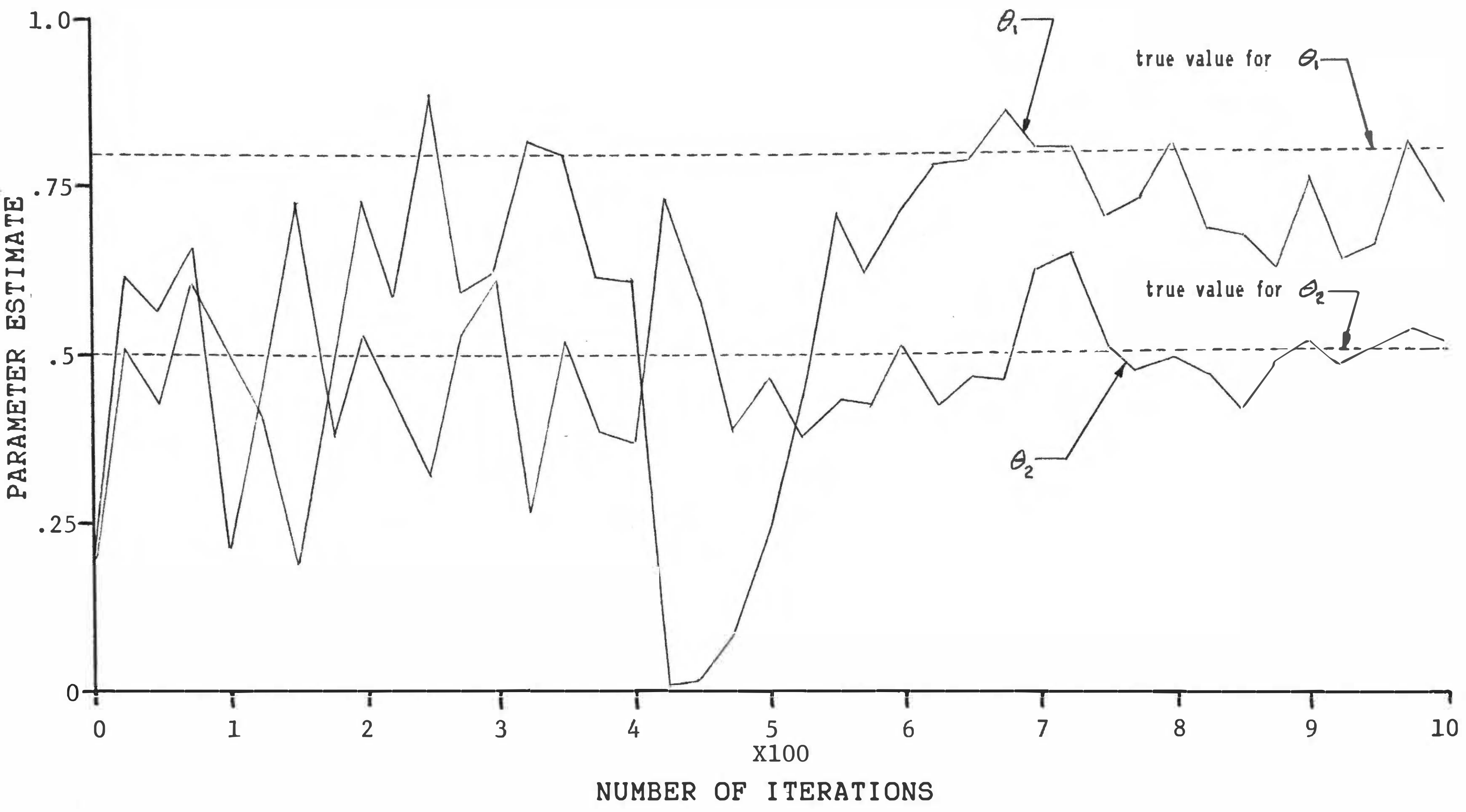


Figure 23. Estimated Parameters from the State Transition Matrix, by the Adaptive Kalman Filter Based on the Stochastic Gradient Algorithm; Simulation Results Based on Using a Modified Process Noise Covariance Matrix.

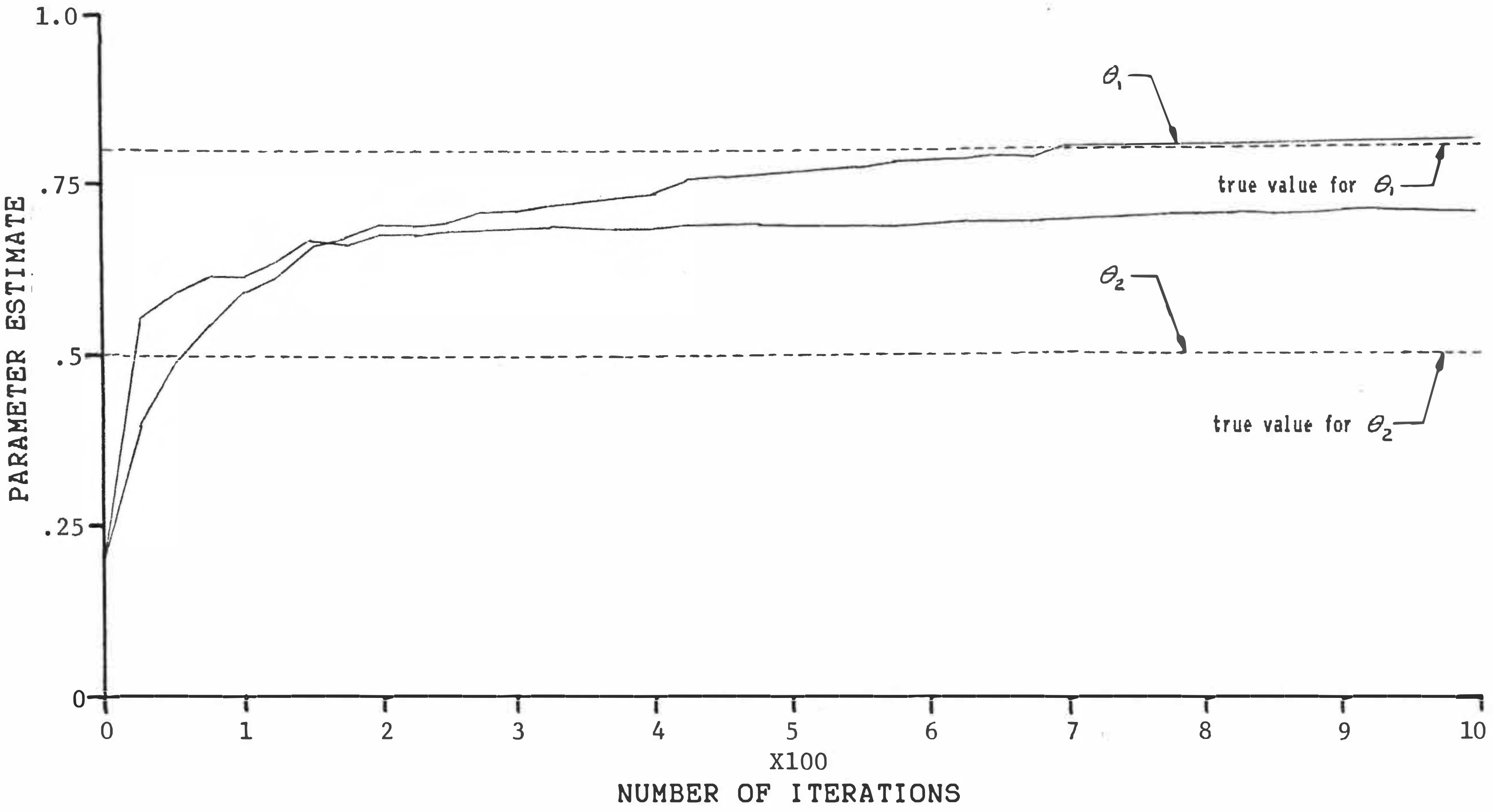


Figure 24. Estimated Parameters from the State Transition Matrix, by the Adaptive Kalman Filter Based on the Modified Stochastic Gradient Algorithm; Simulation Results Based on Using a Modified Process Noise Covariance Matrix.

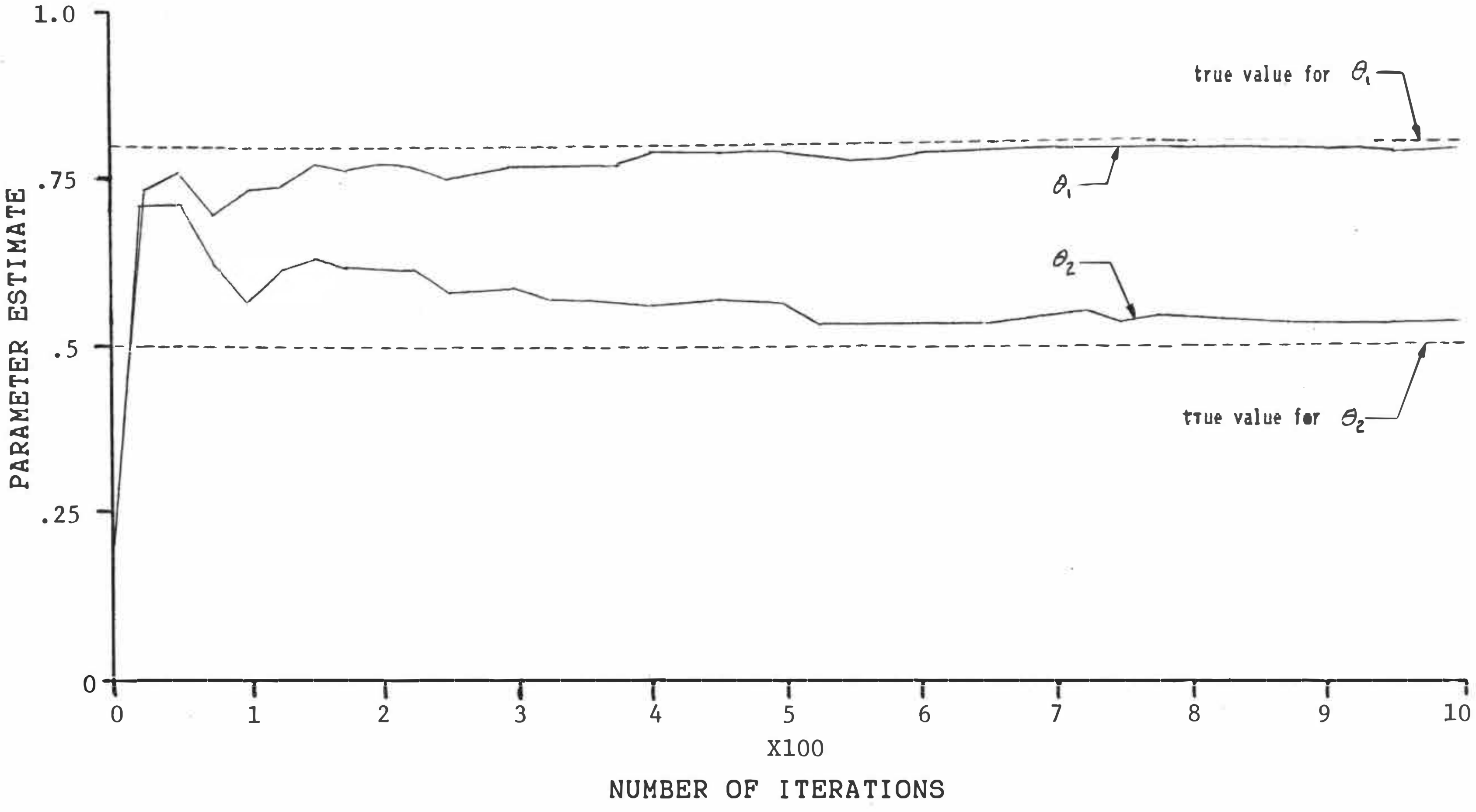


Figure 25. Estimated Parameters from the State Transition Matrix, by the Adaptive Kalman Filter Based on the Stochastic Newton Algorithm; Simulation Results Based on Using a Modified Process Noise Covariance Matrix.

APPENDIX C

FORTH SIMULATION PROGRAM

Screen # 76

```
( ****************** MATRIX OPERATIONS ****************** ) -->

( A: 1st operand, B: 2nd operand, R: destination              )

( inversion of a matrix:                 A,         INVERT    )
( multiplication of matrices:            A, B, R,   MPY       )
( addition/subtraction of matrices:      A, B, R,   ADD/SUB   )
( transpose a matrix                     A, R,      TRSP      )
( multiply a matrix by a const:   const, A,         CMPY      )
( move a matrix:                         A, R,      MOVE      )
( create an ID matrix:                   A,         ID        )
( clear a matrix:                        A,         NULL      )
( dimension a matrix:        rows, cols, A,         DIM       )
( display dimension of a matrix:         A,         DIM?      )
( return matrix trace on the stack:      A,         TRACE     )
( transfer a vector to a matrix:      A, R, col     VMAT      )
```

Screen # 77

```
( ****************** MATRIX OPERATIONS ****************** ) -->

( square elements of a matrix:             A,        SQM       )
( display contents of a matrix:            A,        READ      )
( correlate vectors:                       A, B, R,  CORRELATE )
( designate matrix to be loaded:           A,        LD        )
( load a number into matrix:      number, row, col   L         )
( return address of matrix location: row, col, A     DSEEK     )
( above for indirectly named matrix: row, col, A     SEEK      )

( miscellaneous operations .................................. )

( fill NOISE1 and NOISE2 with gaussian noise:        RANDOM    )
( display statistics of noise:        NOISE1/NOISE2  STAT      )
( set noise pointers to top of noise vectors:        INPOINT   )
( fill WK and VK with new noise from noise vectors: NWNSE      )
```

Screen # 78

```
( set print offset by assigning the first column to SKIP .... )
VARIABLE     SKIP    0 SKIP !

( DIM & DIM? are used for dimensioning of arrays ............ )
: DIM      DUP ROT SWAP 4+ ! ! ;
: DIM?     DUP CR ." rows: " @ . 10 SPACES ." colms: " 4+ @ . ;
                                                             -->
```

Screen # 79

```
( variables used in matrix operations ....................... )

( ********************************************************* )
FVARIABLE SCRATCH    200 ALLOT
( NOTE:  always allot SCRATCH for the largest matrix which    )
(        will be inverted                                     )
( ********************************************************* )


VARIABLE ORDER          VARIABLE STRT          FVARIABLE MULT
VARIABLE TEMP           VARIABLE ARRAY         VARIABLE Q
VARIABLE MATX           VARIABLE MATY          VARIABLE MATR
                                                             -->
```

Screen # 80

```
( matrix inversion using partial pivoting .................. )
: SRCH       1- SWAP 1- ORDER @ * + 8 * 8 + ;
( shadow creates an identity matrix in which the inverse forms)
: SHADOW     SCRATCH 8 + ORDER @ DUP * 8 * ERASE ORDER @ 1+ 1
              DO 1.0 E 0 I DUP SRCH SCRATCH + F! LOOP ;

( pivot searches for the largest pivot element ............. )
: PIVOT     ORDER @ I =
             IF I TEMP !
              ELSE 0.0 E 0 ORDER @ 1+ Q @
               DO I Q @ SRCH ARRAY @ + F@ FABS FOVER FOVER F<=
                IF FSWAP FDROP I TEMP !
                 ELSE FDROP
                THEN
               LOOP
             THEN FDROP ;                                    -->
```

Screen # 82

```
( normalize the row such that pivot becomes unity ........... )
: NORMAL     Q @ DUP SRCH ARRAY @ + F@ ORDER @ 1+ Q @
              DO FDUP Q @ I SRCH ARRAY @ + DUP TEMP !
                 F@ FSWAP F/ TEMP @ F!
              LOOP ORDER @ 1+ 1
              DO FDUP Q @ I SRCH SCRATCH + DUP TEMP !
                 F@ FSWAP F/ TEMP @ F!
              LOOP FDROP ;

( transfer inverted matrix from scratch location to matrix R  )
: TRANS      ORDER @ 1+ 1
              DO ORDER @ 1+ 1
               DO  I J SRCH SCRATCH + F@
                   I J SRCH ARRAY @ + F!
               LOOP
              LOOP ;                                         -->
```

Screen # 81

```
( exchange the row containing the pivot with current row .... )
: EXCHG     TEMP @ Q @ =
              IF
               ELSE ORDER @ 1+ 1
                DO TEMP @ I SRCH ARRAY @ + F@
                   Q    @ I SRCH ARRAY @ + F@
                   TEMP @ I SRCH ARRAY @ + F!
                   Q    @ I SRCH ARRAY @ + F!
                   TEMP @ I SRCH SCRATCH + F@
                   Q    @ I SRCH SCRATCH + F@
                   TEMP @ I SRCH SCRATCH + F!
                   Q    @ I SRCH SCRATCH + F!
                LOOP
              THEN ;
                                                             -->
```

Screen # 83

```
: GAUSS     ORDER @ 1+ 1   ( ........ invert matrix ......... )
            DO I Q ! PIVOT EXCHG NORMAL ORDER @ 1+ 1
             DO J I =
              IF ELSE
               I J SRCH ARRAY @ + F@ MULT F!
               ORDER @ 1+ STRT @ 1+
                DO     J I SRCH ARRAY @ + F@
                     Q @ I SRCH ARRAY @ + F@ MULT F@ F*
                 F-    J I SRCH ARRAY @ + F!
                LOOP ORDER @ 1+ 1
                DO     J I SRCH SCRATCH + F@
                     Q @ I SRCH SCRATCH + F@ MULT F@ F*
                 F-    J I SRCH SCRATCH + F!
                LOOP
              THEN
            LOOP STRT @ 1+ STRT ! LOOP TRANS ;               -->
```

Screen # 84

```
: INVERT     DUP ARRAY ! DUP DUP @ SWAP 4+ @ = NOT
              ABORT" must be a square matrix "
             @ ORDER ! SHADOW 0 STRT ! GAUSS ;
( ------------- end of matrix inversion ------------------- )

( return address of element from an indirectly named matrix . )
: DSEEK     DUP 4+ @ 3 ROLL 1- * ROT 1- + 8 * 8 + + ;
( return address of element from a directly named matrix .... )
: SEEK      @ DSEEK ;
( return the trace of a matrix ............................ )
: TRACE     DUP MATR ! DUP @ SWAP 4+ @ =
             IF 0.0 E 0 MATR @ @ 1+ 1
               DO I I MATR SEEK  F@ F+
               LOOP
              ELSE ABORT" must be a square matrix "
             THEN ;                                    -->
```

Screen # 86

```
( matrix addition and subtraction ........................... )
: A+/-    MATR ! OVER OVER MATY ! MATX ! OVER OVER @ SWAP @ =
          ROT ROT 4+ @ SWAP 4+ @ = AND
          IF MATX @ DUP @ SWAP 4+ @ MATR @ DUP 4+ ROT SWAP
            ! ! MATR @ @ 1+ 1
            DO MATR @ 4+ @ 1+ 1
             DO J I MATX SEEK F@
                J I MATY SEEK F@ Q @ IF F+ ELSE F-
                 THEN J I MATR SEEK F!
             LOOP
            LOOP
           ELSE ABORT" matrix dimensions do not conform "
          THEN ;
: ADD     1 Q ! A+/- ;
: SUB     0 Q ! A+/- ;
                                                             -->
```

Screen # 85

```
( matrix multiplication ..................................... )
: MPY     MATR ! OVER OVER MATY ! MATX ! @ SWAP 4+ @ =
          IF MATX @ @ MATR @ ! MATY @ 4+ @ MATR @ 4+ !
            MATR @ @ 1+ 1
            DO I Q !     MATR @ 4+ @ 1+ 1
             DO 0.0 E 0 MATY @ @ 1+ 1
              DO Q @ I MATX SEEK F@
                  I    J MATY SEEK F@ F* F+
              LOOP  J I MATR SEEK F!
             LOOP
            LOOP
           ELSE ABORT" matrix dimensions do not conform "
          THEN ;

                                                             -->
```

Screen # 87

```
( transpose a matrix ........................................ )
: TRSP    OVER OVER MATR ! MATX ! SWAP DUP 4+ @ SWAP @ ROT DUP
          ROT SWAP 4+ ! ! MATX @ @ 1+ 1
            DO MATX @ 4+ @ 1+ 1
             DO J I MATX SEEK F@ I J MATR SEEK F!
             LOOP
            LOOP ;
( clearing a matrix ......................................... )
: NULL    DUP DUP @ SWAP 4+ @ * 8 * SWAP 8 + SWAP ERASE ;
( setting a matrix to the identity matrix .................. )
: ID      DUP DUP @ SWAP 4+ @ =
           IF DUP TEMP ! DUP NULL @ 1+ 1
             DO 1.0 E 0 I DUP TEMP SEEK F!
             LOOP
            ELSE ABORT" identity matrix must be square " DROP
           THEN ;                                            -->
```

Screen # 88

```
( multiply a matrix by a constant .......................... )
: CMPY     DUP MATR ! @ 1+ 1
           DO MATR @ 4+ @ 1+ 1
            DO FDUP J I MATR SEEK F@ F* J I MATR SEEK F!
            LOOP
           LOOP FDROP ;

( moving a matrix from one location to another ............. )
: MOVE     OVER OVER MATR ! MATX ! SWAP DUP @ SWAP 4+ @
           ROT DUP 4+ ROT SWAP ! ! MATR @ @ 1+ 1
           DO MATR @ 4+ @ 1+ 1
            DO J I MATX SEEK F@ J I MATR SEEK F!
            LOOP
           LOOP ;

                                                          -->
```

Screen # 89

```
( display the contents of a matrix ......................... )
: SP       SKIP @ SPACES ;
: READ     DUP MATR ! @ 1+ 1
           DO CR SP MATR @ 4+ @ 1+ 1
            DO  I 4 MOD 0 =
             IF CR SP
             THEN  J . ." -" I . ." ="
             J I MATR SEEK F@ 8 F.R  ." "
            LOOP
           LOOP ;

( prepare a matrix for loading ............................. )
: L        TEMP SEEK F! ;
: LD       TEMP ! ;
                                                          -->
```

Screen # 90

```
( correlate two vectors .................................... )
FVARIABLE  XAVG   FVARIABLE YAVG  VARIABLE YPNT
: MEAN     Q ! 0.0 E 0 Q @ @ 1+ 1  DO I 1 Q SEEK F@ F+
                                   LOOP Q @ @ S>F F/ ;
: CORRELATE     MATR ! OVER OVER MATY ! MATX !
          MEAN YAVG F! MEAN XAVG F! MATY @ @ MATR @ @ <=
          IF MATY @ @ DUP MATR @ ! 1 MATR @ 4+ ! 1+ 1
           DO I YPNT ! 0.0 E 0 MATX @ @ 1+ 1
            DO I 1 MATX SEEK F@ XAVG F@ F-   YPNT @ MATY @ @ >
             IF 1 YPNT !
              THEN YPNT @ 1 MATY SEEK F@ YAVG F@ F- F* F+
              YPNT @ 1+ YPNT !
            LOOP MATX @ @ S>F F/ I 1 MATR SEEK F!
           LOOP
          ELSE ." insufficient space alloted in result vector"
          THEN ;                                          -->
```

Screen # 91

```
( square each element of a matrix .......................... )
: SQM      DUP MATR ! @ 1+ 1
           DO MATR @ 4+ @ 1+ 1
            DO J I MATR SEEK F@ FDUP F* J I MATR SEEK F!
            LOOP
           LOOP ;

( load vectors into a matrix by column ..................... )
: VMAT     Q ! MATR ! MATX !
           MATX @ @ DUP MATR @ @ = Q @ MATR @ 4+ @ <= * 0=
           IF ABORT" improper destination dimension"
            ELSE  1+ 1  DO  I 1   MATX SEEK F@
                            I Q @ MATR SEEK F!
                        LOOP
           THEN ;
                                                          -->
```

Screen # 92

```
( random noise generator .................................. )
( ********************************************************* )
(          name    allotment     rows  col. name            )
( --------------------------------------------------------- )
FVARIABLE    NOISE1  24000 ALLOT    3000   1  NOISE1    DIM
FVARIABLE    NOISE2  24000 ALLOT    3000   1  NOISE2    DIM
FVARIABLE         W     24 ALLOT       3   1       W    DIM
FVARIABLE         V      8 ALLOT       1   1       V    DIM
( ********************************************************* )

VARIABLE SEED         FVARIABLE SUM1       FVARIABLE VA
VARIABLE NSE1         FVARIABLE SUM2       FVARIABLE VB
VARIABLE NSE2          VARIABLE RANGE      FVARIABLE FQ
VARIABLE WPOINT        VARIABLE VPOINT
                                                        -->
```

Screen # 93

```
( find the statistics of noise in the noise vectors ......... )
: STAT     NSE1 ! 0.0 E 0 NSE1 @ @ 1+ 1
             DO I 1 NSE1 SEEK F@ F+
             LOOP NSE1 @ @ S>F F/ FDUP FQ F!
           CR SP ." sample mean:  "
           8 F.R 0.0 E 0 NSE1 @ @ 1+ 1
             DO I 1 NSE1 SEEK F@ FQ F@ F- FDUP F* F+
             LOOP NSE1 @ @ S>F F/ FSQRT
           CR SP ." std. dev.:    "
           8 F.R ;

( random noise generator .................................. )
1350756162  SEED !  1000000000 RANGE ! ( range of random no's )
: RD        SEED @ 259 * 3 + 32767 AND DUP SEED ! ;
: RND       RANGE @ RD 32767 */ S>F RANGE @ S>F F/ ;
                                                          -->
```

Screen # 94

```
( random noise generator .................................. )
: RDM     BEGIN RND 2.0 E 0 F* 1.0 E 0 F- FDUP VA F! FDUP F*
                RND 2.0 E 0 F* 1.0 E 0 F- FDUP VB F! FDUP F*
                F+ FDUP FQ F! 1.0 E 0 F<=
          UNTIL -2.0 E 0 FQ F@ F/ FQ F@ FLN F* FSQRT ;

: RANDOM     1350756162 SEED ! 0.0 E 0 FDUP SUM1 F! SUM2 F!
    NOISE1 DUP NSE1 ! NOISE2 NSE2 ! @ 1+ 1
     DO RDM FDUP
      VA F@ F* FDUP I 1 NSE1 SEEK F! SUM1 F@ F+ SUM1 F!
      VB F@ F* FDUP I 1 NSE2 SEEK F! SUM2 F@ F+ SUM2 F!
     LOOP  ( SUM1 F@ NOISE1 @ S>F F/ SUM1 F!                 )
           ( SUM2 F@ NOISE1 @ S>F F/ SUM2 F!  NOISE1 @ 1+ 1 )
   ( DO I 1 NSE1 SEEK F@ SUM1 F@ F- I 1 NSE1 SEEK F!        )
   (    I 1 NSE2 SEEK F@ SUM2 F@ F- I 1 NSE2 SEEK F!        )
   ( LOOP ) ;                                              -->
```

Screen # 95

```
( load vectors W and V with new-noise elements ............. )

: INPOINT    1 WPOINT ! 1 VPOINT ! ;

: NOISE     W @ 1+ 1
            DO     WPOINT @ NOISE1 @ >
             IF 1 WPOINT ! CR ." using old process noise" CR
             THEN NOISE1 WPOINT @ 8 * + F@
             W I 8 * + F!  WPOINT @ 1+ WPOINT !
            LOOP   V @ 1+ 1
            DO     VPOINT @ NOISE2 @ >
             IF 1 VPOINT ! CR ." using old measurement noise" CR
             THEN NOISE2 VPOINT @ 8 * + F@
             V I 8 * + F!  VPOINT @ 1+ VPOINT !
            LOOP ;
                                                          -->
```

Screen # 96

```
-->
```

Screen # 97

```
( ************************************************************** ) -->
(                                                                   )
(                       NOMINAL SYSTEM MODEL                        )
(                                                                   )
(                  X{k+1}  =  A X{k}    +    W{k}                   )
(                  Y{k+1}  =  C X{k+1}  +    V{k}                   )
(                                                                   )
(                     I  0   0  -{.5}{.8}**2   I                    )
(                A =  I  1   0      -{.8}**2   I                    )
(                     I  0   1      -{.5}      I                    )
(                                                                   )
(                                                                   )
(                        C =  I  1   1   1  I                       )
(                                                                   )
(                                                                   )
( ****************************************************************** )
```

Screen # 98

```
( matrices used in the system model .......................... )
( ************************************************************ )
(               name  allotment   rows  columns   name         )
( ------------------------------------------------------------ )
FVARIABLE        A      72 ALLOT     3        3        A   DIM
FVARIABLE        C      24 ALLOT     1        3        C   DIM
FVARIABLE        X      24 ALLOT     3        1        X   DIM
FVARIABLE       NZ       8 ALLOT     1        1       NZ   DIM  -->
( ************************************************************ )
```

Screen # 99

```
( temporary storage matrices ................................. )
( ************************************************************ )
(               name  allotment   rows  columns    name        )
( ------------------------------------------------------------ )
FVARIABLE       T1    200 ALLOT     5        5       T1    DIM
FVARIABLE       T2    200 ALLOT     5        5       T2    DIM
FVARIABLE       T3    200 ALLOT     5        5       T3    DIM
FVARIABLE       T4    200 ALLOT     5        5       T4    DIM
FVARIABLE       T5    200 ALLOT     5        5       T5    DIM
FVARIABLE       T6    200 ALLOT     5        5       T6    DIM
FVARIABLE       T7    200 ALLOT     5        5       T7    DIM
FVARIABLE       T8    200 ALLOT     5        5       T8    DIM
FVARIABLE       T9    200 ALLOT     5        5       T9    DIM
( ************************************************************ )
 VARIABLE KOUNT
                                                             -->
```

Screen # 100

```
( initialize the state transition matrix ................... )
A LD
    0.0 E 0  1 1 L   0.0 E 0  1 2 L   -0.32   E  0  1 3 L
    1.0 E 0  2 1 L   0.0 E 0  2 2 L   -0.64   E  0  2 3 L
    0.0 E 0  3 1 L   1.0 E 0  3 2 L   -0.5    E  0  3 3 L

( initialize the input matrix .............................. )
( when used )

( initialize the observation matrix ........................ )
C LD
    1.0 E 0  1 1 L   1.0 E 0  1 2 L   1.0 E 0  1 3 L
                                                          -->
```

Screen # 101

```
( define the initial state vector .......................... )
: IX  X LD           1.0 E 0  1 1 L
                     1.0 E 0  2 1 L
                     1.0 E 0  3 1 L ;
                                                          -->
```

Screen # 102

```
( system model ............................................. )
: MODEL    NOISE              A X T1 MPY
           T1  W X ADD        C X T1 MPY       T1 V NZ ADD ;

( initialize the system model .............................. )
: IMODEL    IX  INPOINT  NZ NULL ;
                                                          -->
( MODEL leaves the state of the model driven by process noise )
( in X, and the output contaminated by measurement noise in   )
( in NZ                                                       )
```

Screen # 103

```
( ***********************************************************) -->
(                                                             )
(   ADAPTIVE KALMAN FILTER ALGORITHM FOR THE SYSTEM :         )
(                                                             )
(              X{k+1}  =  AF X{k}      +    W{k}              )
(              Y{k+1}  =  CF X{k+1}    +    V{k}              )
(                                                             )
(         I  0   0  -{&2}{&1}**2  I             I  1  I       )
(   AF =  I  1   0      -{&1}**2  I    XF{0} =  I  1  I       )
(         I  0   1      -{&2}     I             I  1  I       )
(                                                             )
(               I  &3   0   0  I                              )
(  COV[W{k}]  = I   0  &4   0  I   COV[V{k}]  = I  1  I       )
(               I   0   0  &5  I                              )
(                                                             )
( *********************************************************** )
```

Screen # 104
```
( ********************************************************* ) -->
(                                                             )
(          &n - indicates uncertain parameters                )
(                                                             )
(        the initial estimate and true value of the           )
(        parameter vector is :                                )
(                                                             )
(                        est.                 true            )
(                                                             )
(     I  &1  I          I  0.2  I          I  0.8  I          )
(     I  &2  I          I  0.2  I          I  0.5  I          )
(     I  &3  I          I  1.0  I          I  1.0  I          )
(     I  &4  I          I  1.0  I          I  1.0  I          )
(     I  &5  I          I  1.0  I          I  1.0  I          )
(                                                             )
( *********************************************************** )
```

Screen # 105
```
(             name    allotment    rows  col.    name        )
( ---------------------------------------------------------- )
FVARIABLE        K   24 ALLOT     3     1        K   DIM
FVARIABLE        S    8 ALLOT     1     1        S   DIM
FVARIABLE        D   72 ALLOT     3     3        D   DIM
FVARIABLE      RES    8 ALLOT     1     1      RES   DIM
FVARIABLE      @1A   72 ALLOT     3     3      @1A   DIM
FVARIABLE      @2A   72 ALLOT     3     3      @2A   DIM
FVARIABLE      @3A   72 ALLOT     3     3      @3A   DIM
FVARIABLE      @4A   72 ALLOT     3     3      @4A   DIM
FVARIABLE      @5A   72 ALLOT     3     3      @5A   DIM
FVARIABLE    @1RES    8 ALLOT     1     1    @1RES   DIM
FVARIABLE    @2RES    8 ALLOT     1     1    @2RES   DIM
FVARIABLE    @3RES    8 ALLOT     1     1    @3RES   DIM
FVARIABLE    @4RES    8 ALLOT     1     1    @4RES   DIM
FVARIABLE    @5RES    8 ALLOT     1     1    @5RES   DIM  -->
```

Screen # 106
```
(            name  allotment   rows  columns   name          )
( ---------------------------------------------------------- )
FVARIABLE     GRAD   40 ALLOT     1      5      GRAD   DIM
FVARIABLE    @5ROW   40 ALLOT     1      5     @5ROW   DIM
FVARIABLE     @1QN   72 ALLOT     3      3      @1QN   DIM
FVARIABLE     @2QN   72 ALLOT     3      3      @2QN   DIM
FVARIABLE     @3QN   72 ALLOT     3      3      @3QN   DIM
FVARIABLE     @4QN   72 ALLOT     3      3      @4QN   DIM
FVARIABLE     @5QN   72 ALLOT     3      3      @5QN   DIM
FVARIABLE     @1RN    8 ALLOT     1      1      @1RN   DIM
FVARIABLE     @2RN    8 ALLOT     1      1      @2RN   DIM
FVARIABLE     @3RN    8 ALLOT     1      1      @3RN   DIM
FVARIABLE     @4RN    8 ALLOT     1      1      @4RN   DIM
FVARIABLE     @5RN    8 ALLOT     1      1      @5RN   DIM
FVARIABLE   PARAMH   40 ALLOT     5      1    PARAMH   DIM
FVARIABLE   PARAML   40 ALLOT     5      1    PARAML   DIM  -->
```

Screen # 107
```
(             name  allotment   rows  columns    name        )
( ---------------------------------------------------------- )
FVARIABLE      @1K   24 ALLOT     3      1       @1K   DIM
FVARIABLE      @2K   24 ALLOT     3      1       @2K   DIM
FVARIABLE      @3K   24 ALLOT     3      1       @3K   DIM
FVARIABLE      @4K   24 ALLOT     3      1       @4K   DIM
FVARIABLE      @5K   24 ALLOT     3      1       @5K   DIM
FVARIABLE      @1S    8 ALLOT     1      1       @1S   DIM
FVARIABLE      @2S    8 ALLOT     1      1       @2S   DIM
FVARIABLE      @3S    8 ALLOT     1      1       @3S   DIM
FVARIABLE      @4S    8 ALLOT     1      1       @4S   DIM
FVARIABLE      @5S    8 ALLOT     1      1       @5S   DIM
FVARIABLE      IDM   72 ALLOT     3      3       IDM   DIM
                                                            -->
( these matrices are not required to define the state of the )
( filter; they are used as temporary storage only            )
```

Screen # 108

```
( variables containing addresses of corresponding matrices    )
( that define the state of the filter ....................... )

 VARIABLE  AF                        VARIABLE  CF    VARIABLE  XF
 VARIABLE  RN    VARIABLE  QN        VARIABLE  P     VARIABLE  R
 VARIABLE  SOLD  VARIABLE  GOLD      VARIABLE  PARAM

 VARIABLE  @1X   VARIABLE  @2X       VARIABLE  @3X   VARIABLE  @4X
 VARIABLE  @5X
 VARIABLE  @1P   VARIABLE  @2P       VARIABLE  @3P   VARIABLE  @4P
 VARIABLE  @5P

( miscellaneous variables .................................. )
 VARIABLE  ALPHA   1 ALPHA !   FVARIABLE  BETA  1.0 E 0 BETA F!
 VARIABLE  S1    VARIABLE  S2     VARIABLE  S3              -->
```

Screen # 109

```
( the following definitions were defined to allow several      )
( algorithms to use the same set of equations .............. )

: AF* AF @ ;                   : CF* CF @ ;    : XF* XF @ ;
: P* P @ ;       : RN* RN @ ;  : QN* QN @ ;    : PARAM* PARAM @ ;
: R* R @ ;

: SOLD* SOLD @ ;   : GOLD*  GOLD @ ;

: @1X* @1X @ ;  : @2X* @2X @ ;  : @3X* @3X @ ;  : @4X* @4X @ ;
: @5X* @5X @ ;

: @1P* @1P @ ;  : @2P* @2P @ ;  : @3P* @3P @ ;  : @4P* @4P @ ;
: @5P* @5P @ ;
                                                             -->
```

Screen # 110

```
( initialize vector of estimated parameters ................. )
: IPARAM PARAM* LD
              0.20   E 0  1 1 L   ( estimated  a             )
              0.20   E 0  2 1 L   ( estimated  b             )
              1.00   E 0  3 1 L   ( estimated COV[W1]        )
              1.00   E 0  4 1 L   ( estimated COV[W2]        )
              1.00   E 0  5 1 L ; ( estimated COV[W3]        )
                                                           -->
```

Screen # 111

```
( initialize vector of lower bound of estimated parameters .. )
PARAML LD      0.01    E 0   1 1 L  ( low estimate of a         )
               0.01    E 0   2 1 L  ( low estimate of b         )
               0.01    E 0   3 1 L  ( low estimate of COV[W1]   )
               0.01    E 0   4 1 L  ( low estimate of COV[W2]   )
               0.01    E 0   5 1 L  ( low estimate of COV[W3]   )

( initialize vector of upper bound of estimated parameters .. )
PARAMH LD      0.99    E 0   1 1 L  ( high estimate of a        )
               0.99    E 0   2 1 L  ( high estimate of b        )
              10.00    E 0   3 1 L  ( high estimate of COV[W1]  )
              10.00    E 0   4 1 L  ( high estimate of COV[W2]  )
              10.00    E 0   5 1 L  ( high estimate of COV[W3]  )
                                                             -->
```

Screen # 112

```
( update the AF matrix with parameters in PARAM ............. )
: UPAF  2 1 PARAM* DSEEK F@ 1 1 PARAM* DSEEK F@ FDUP F* F* FCHS
        1 3 AF* DSEEK F!
        1 1 PARAM* DSEEK F@ FDUP F* FCHS 2 3  AF* DSEEK F!
        2 1 PARAM* DSEEK F@ FCHS         3 3  AF* DSEEK F! ;

( initialize the AF matrix with parameters in PARAM ......... )
: IAF   AF* NULL  AF* LD
        1.0 E 0  2 1 L  1.0 E 0  3 2 L  UPAF ;

( initialize the XF vector ................................. )
: IXF   XF* LD
        1.0 E 0  1 1 L
        1.0 E 0  2 1 L
        1.0 E 0  3 1 L ;
                                                           -->
```

Screen # 113

```
( update the BF matrix with parameters in PARAM ............. )
( not used )

( initialize the BF matrix with parameters in PARAM ......... )
( not used )

( update the CF matrix with parameters in PARAM ............. )
( not used )

( initialize the CF matrix with parameters in PARAM ......... )
: ICF   C CF* MOVE ;
                                                           -->
( the simulation may be expanded to estimate parameters in    )
( any of these matrices                                       )
```

Screen # 114

```
( update the QN matrix with parameters in PARAM ............. )
: UPQN   3 1 PARAM* DSEEK F@    1 1 QN* DSEEK F!
         4 1 PARAM* DSEEK F@    2 2 QN* DSEEK F!
         5 1 PARAM* DSEEK F@    3 3 QN* DSEEK F! ;

( initialize the QN matrix with parameters in PARAM ......... )
: IQN   QN* NULL UPQN ;
                                                           -->
```

Screen # 115

```
( initialize the error covariance matrix .................... )
: IP    P* NULL P* LD
        1.0 E 1  1 1 L   1.0 E 1  2 2 L   1.0 E 1  3 3 L ;

( update the RN matrix with parameters in PARAM ............. )
: UPRN ;

( initialize the RN matrix with parameters in PARAM ......... )
: IRN   RN* ID ;

( initialize the Hessian matrix ............................. )
: IR    R* ID ;


                                                           -->
```

Screen # 116

```
( compute the partials of A wrt unknown parameters .......... )
: P@1A  @1A NULL  1 1 PARAM* DSEEK F@ -2.0 E 0 F* FDUP
                  2 1 PARAM* DSEEK F@ F*
                  1 3   @1A  DSEEK F!
                  2 3   @1A  DSEEK F! ;
: P@2A  @2A NULL  1 1 PARAM* DSEEK F@ FDUP F* FCHS
                  1 3   @2A  DSEEK F!
         -1.0 E 0 3 3   @2A  DSEEK F! ;

: P@3A  @3A NULL ;
: P@4A  @4A NULL ;
: P@5A  @5A NULL ;

: P@#A P@1A P@2A P@3A P@4A P@5A ;                           -->
```

Screen # 117

```
( initialize the partials of RN wrt unknown parameters ...... )
: IP@#RN  @1RN NULL  @2RN NULL  @3RN NULL  @4RN NULL
          @5RN NULL ;

( initialize the partials of QN wrt unknown parameters ...... )
: IP@#QN  @1QN NULL  @2QN NULL
          @3QN NULL   1.0 E 0   1 1 @3QN DSEEK  F!
          @4QN NULL   1.0 E 0   2 2 @4QN DSEEK  F!
          @5QN NULL   1.0 E 0   3 3 @5QN DSEEK  F! ;
                                                            -->
( Partial derivatives of elements in the noise covariance     )
( matrices do not change.  Updating these partrial            )
( derivative matrices is not required.                        )
```

Screen # 118

```
( compute initial partials of X1/0 wrt to unknown parameters  )
: I@#X1/0     @1A  XF*  @1X*  MPY
              @2A  XF*  @2X*  MPY
              @3A  XF*  @3X*  MPY
              @4A  XF*  @4X*  MPY
              @5A  XF*  @5X*  MPY ;
( compute initial partials of P1/0 wrt to unknown parameters  )
: EI@#P1/0    S3 ! S2 ! S1 !
              S1 @ T1 T3 MPY        S1 @ T4 TRSP   T2 T4 T5 MPY
                T3 T5 T4 ADD    T4 S2 @ S3 @ ADD ;
: I@#P1/0     AF* T3 TRSP       P* T3 T1 MPY     AF* P* T2 MPY
              @1A  @1QN  @1P*  EI@#P1/0
              @2A  @2QN  @2P*  EI@#P1/0
              @3A  @3QN  @3P*  EI@#P1/0
              @4A  @4QN  @4P*  EI@#P1/0
              @5A  @5QN  @5P*  EI@#P1/0 ;                   -->
```

Screen # 119

```
( compute the inverse of the error covariance, ............. )
: CS        CF* T1 TRSP    P* T1 T2 MPY    CF* T2 T1 MPY
          T1 RN*  S ADD    S  INVERT ;

( compute the partials of S wrt to unknown parameters ....... )
: EP@#S  S3 !  S2 !  S1 !
          S1 @ T1 T2 MPY    CF* T2 T3 MPY      S2 @ T3 T2 ADD
            S T2 T3 MPY    T3 S S3 @ MPY   -1.0 E 0 S3 @ CMPY ;
: P@#S    CF* T1 TRSP
           @1P* @1RN  @1S  EP@#S
           @2P* @2RN  @2S  EP@#S
           @3P* @3RN  @3S  EP@#S
           @4P* @4RN  @4S  EP@#S
           @5P* @5RN  @5S  EP@#S ;
                                                            -->
```

Screen # 120

```
( propagate the error covariance ... ......................... )
: PROPP     AF* T1 TRSP    P* T1 T2 MPY
         AF* T2 T1 MPY    T1 QN* P* ADD ;

( propagate the state estimate ............................... )
: PROPX   AF* XF* T1 MPY  T1 XF* MOVE ;

( find the residual from a noisy measurement ................ )
: CRES    CF* XF* T1 MPY  NZ T1 RES SUB ;

( compute the partials of the resid wrt to unknown parameters )
: P@#RES  CF* @1X* @1RES MPY   -1.0 E 0 @1RES CMPY
          CF* @2X* @2RES MPY   -1.0 E 0 @2RES CMPY
          CF* @3X* @3RES MPY   -1.0 E 0 @3RES CMPY
          CF* @4X* @4RES MPY   -1.0 E 0 @4RES CMPY
          CF* @5X* @5RES MPY   -1.0 E 0 @5RES CMPY  ;      -->
```

Screen # 121

```
( ************ parameter estimation algorithms ************ )

( mult. correction by none accelerated gain sequence ........ )
: GAMA-N  KOUNT @ S>F ;

( clamp parameter to compact subset and store in PARAM vector )
: CLAMP    DUP S1 ! 1 PARAMH  DSEEK F@ FMIN
               S1 @ 1 PARAML  DSEEK F@ FMAX
               S1 @ 1 PARAM*  DSEEK F! ;

( form gradient matrix from derivatives of residual vectors   )
: PGRAD       @1RES   GRAD   1 VMAT
              @2RES   GRAD   2 VMAT
              @3RES   GRAD   3 VMAT
              @4RES   GRAD   4 VMAT
              @5RES   GRAD   5 VMAT ;                       -->
```

Screen # 122

```
( ******************** first variation ******************** )

( update parameters, unweighted stochastic gradient approach  )
: UPPARAM1   CRES    P@#RES    PGRAD
             GRAD T1 TRSP    T1 RES T2 MPY
             GAMA-N 1/F T2 CMPY    PARAM* T2 T1 SUB
             6 1 DO   I 1 T1 DSEEK F@  I CLAMP  LOOP ;

( ********************************************************** )
                                                           -->
```

Screen # 123

```
( ******************* second variation ******************* )

( compute approximate Hessian matrix ....................... )
: PHESS   GRAD T1 TRSP  T1 S T2 MPY  T2 GRAD T1 MPY
          T1 R* T2 SUB  T1 ID  0.01 E 0 T1 CMPY  T1 T2 T3 ADD
          GAMA-N 1/F T3 CMPY   R* T3 R* ADD ;

( update parameters, recursive prediction error ............ )
: UPPARAM2   CRES  P@#RES  PGRAD  CS  PHESS
             GRAD T1 TRSP     T1 S T2 MPY    T2 RES T1 MPY
               R* T2 MOVE       T2 INVERT
             T2 T1 T3 MPY   GAMA-N 1/F T3 CMPY
             PARAM* T3 T2 SUB
             6 1 DO   I 1 T2 DSEEK F@  I CLAMP  LOOP ;      -->

( ********************************************************** )
```

Screen # 124

```
( reserved for expansion ............................... ) -->
```

Screen # 125

```
( compute the kalman gain .................................... )
: GAIN     CF* T1 TRSP      P* T1 T2 MPY    T2 S K MPY ;

( compute D matrix ........................................... )
: CD       K CF* T1 MPY   IDM T1 D SUB ;

( compute the partials of K wrt to unknown parameters ....... )
: EP@#K    S3 !  S2 !  S1 !
           S1 @ T1 T3 MPY    T2 S2 @ T4 MPY    T3 T4 S3 @ ADD ;

: P@#K     CF* T3 TRSP      T3 S T1 MPY   P* T3 T2 MPY
           @1P*  @1S  @1K  EP@#K
           @2P*  @2S  @2K  EP@#K
           @3P*  @3S  @3K  EP@#K
           @4P*  @4S  @4K  EP@#K
           @5P*  @5S  @5K  EP@#K ;                             -->
```

Screen # 126

```
( compute the partials of P1/1 wrt to unknown parameters .... )
: EP@#P1/1  S3 !  S2 !  S1 !
            S1 @ T4 T6 MPY    D S2 @ T7 MPY       T7 T6 T8 SUB
              T8 T1 T6 MPY  S1 @ CF* T7 MPY          T7 T8 TRSP
              T3 T8 T7 MPY      T6 T7 T8 SUB   S1 @ RN* T6 MPY
             K S3 @ T7 MPY      T6 T7 T9 ADD       T9 T2 T6 MPY
              T6 T8 T7 ADD       S1 @ T6 TRSP      T5 T6 T8 MPY
            T7 T8 S2 @ ADD ;

: P@#P1/1         D T1 TRSP      K T2 TRSP      D P* T3 MPY
            CF* P* T4 MPY  K RN* T5 MPY
            @1K  @1P*  @1RN  EP@#P1/1
            @2K  @2P*  @2RN  EP@#P1/1
            @3K  @3P*  @3RN  EP@#P1/1
            @4K  @4P*  @4RN  EP@#P1/1
            @5K  @5P*  @5RN  EP@#P1/1 ;                     -->
```

Screen # 127

```
( compute the new state estimate ............................ )
: UPXF    K RES T1 MPY    XF* T1 XF* ADD ;

( compute the new error covariance .......................... )
: UPP          D T1 TRSP      D P* T2 MPY     T2 T1 T3 MPY
               K T1 TRSP     K RN* T2 MPY     T2 T1 T4 MPY
           T3 T4 P* ADD ;

                                                               -->
```

Screen # 128

```
( compute the partials of X1/0 wrt to unknown parameters .... )
: EP@#X1/0  S3 !  S2 !  S1 !
            AF* S1 @ T2 MPY    S2 @ K T3 MPY      T2 T3 T4 ADD
             T4  RES T2 MPY   T1 S3 @ T3 MPY      T2 T3 T4 ADD
            S2 @ XF* T2 MPY   T2 T4 S3 @ ADD ;

: P@#X1/0   AF* D T1 MPY
            @1K  @1A  @1X*   EP@#X1/0
            @2K  @2A  @2X*   EP@#X1/0
            @3K  @3A  @3X*   EP@#X1/0
            @4K  @4A  @4X*   EP@#X1/0
            @5K  @5A  @5X*   EP@#X1/0 ;                        -->
```

Screen # 129

```
( compute the partials of P1/0 wrt to unknown parameters .... )
: EP@#P1/0  S3 !  S2 !  S1 !
          S1 @ T2 T4 MPY    AF* S2 @ T5 MPY       T5 T1 T6 MPY
            T4 T6 T5 ADD         S1 @ T4 TRSP      T3 T4 T6 MPY
            T5 T6 T4 ADD   S3 @ T4 S2 @ ADD ;
: P@#P1/0   AF* T1 TRSP   P* T1 T2 MPY   AF* P* T3 MPY
            @1A  @1P*  @1QN  EP@#P1/0
            @2A  @2P*  @2QN  EP@#P1/0
            @3A  @3P*  @3QN  EP@#P1/0
            @4A  @4P*  @4QN  EP@#P1/0
            @5A  @5P*  @5QN  EP@#P1/0 ;

( initialize either adaptive kalman filter ................. )
: IAFILTER  IPARAM   IAF   ICF   IR   IRN   IQN   IP   IXF
            P@#A  I@#X1/0  IP@#QN  IP@#RN  I@#P1/0  IDM ID ;
                                                              -->
```

Screen # 130

```
( unmodified gradient approach adaptive filter .............. )
: AFILTER1   PROPP      PROPX     UPPARAM1
             UPAF       UPRN      UPQN
             CS         P@#S      GAIN
             CD         P@#K      P@#P1/1  UPP    P@#A
             P@#P1/0    P@#X1/0   UPXF ;

( modified gradient approach adaptive filter ................ )
: AFILTER2   PROPP      PROPX     UPPARAM2
             UPAF       UPRN      UPQN
             CS         P@#S      GAIN
             CD         P@#K      P@#P1/1  UPP    P@#A
             P@#P1/0    P@#X1/0   UPXF ;

( conventional kalman filtering ............................ )
: CFILTER    PROPP  PROPX  CRES  CS  GAIN  CD  UPP  UPXF ;  -->
```

Screen # 131

```
( The equations on the previous screens will next be modified )
( so more than one set of matrices can be passed to and from  )
( either algorithm.  This simplifies debugging since the main )
( algorithm was verified to give correct results.  Three sets )
( of matrices and parameters will be defined:                 )
( Set 1 : PARAM  AF, BF, CF, XF, P, RN, QN, @#X, @#P          )
(  -these will be passed between either modification and the  )
(    main program.  They determine the state of the filter.   )
( Set 2 : RES, S, K, D, @#RES, @#S, @#K, @#A                  )
(  -these are not passed, they are computed each iteration    )
(    of the algorithm.                                        )
( Set 3 : @#RN, @#QN, NZ, U, IDM, PARAML, PARAMH, ALPHA       )
(  -these are used by either algorithm.  They are not expected)
(    to change during the entire simaulation.                 )
                                                              -->
```

Screen # 132

```
( arrays defining the original adaptive filter ............. )
( ********************************************************** )
(              name  allotment   rows  columns   name         )
( ---------------------------------------------------------- )
FVARIABLE        AF1    72 ALLOT     3       3       AF1   DIM
FVARIABLE        CF1    24 ALLOT     1       3       CF1   DIM
FVARIABLE        XF1    24 ALLOT     3       1       XF1   DIM
FVARIABLE        QN1    72 ALLOT     3       3       QN1   DIM
FVARIABLE        RN1     8 ALLOT     1       1       RN1   DIM
FVARIABLE         P1    72 ALLOT     3       3        P1   DIM
FVARIABLE         R1   200 ALLOT     5       5        R1   DIM
FVARIABLE      GOLD1    40 ALLOT     5       1     GOLD1   DIM
FVARIABLE      SOLD1    40 ALLOT     5       1     SOLD1   DIM
FVARIABLE     PARAM1    40 ALLOT     5       1    PARAM1   DIM
FVARIABLE       MSE1    24 ALLOT     3       1      MSE1   DIM
                                                              -->
```

Screen # 133

```
( arrays defining the original adaptive filter ............. )
( ********************************************************** )
(              name  allotment   rows  columns    name        )
( ---------------------------------------------------------- )
FVARIABLE        @1X1   24 ALLOT     3       1       @1X1  DIM
FVARIABLE        @2X1   24 ALLOT     3       1       @2X1  DIM
FVARIABLE        @3X1   24 ALLOT     3       1       @3X1  DIM
FVARIABLE        @4X1   24 ALLOT     3       1       @4X1  DIM
FVARIABLE        @5X1   24 ALLOT     3       1       @5X1  DIM
FVARIABLE        @1P1   72 ALLOT     3       3       @1P1  DIM
FVARIABLE        @2P1   72 ALLOT     3       3       @2P1  DIM
FVARIABLE        @3P1   72 ALLOT     3       3       @3P1  DIM
FVARIABLE        @4P1   72 ALLOT     3       3       @4P1  DIM
FVARIABLE        @5P1   72 ALLOT     3       3       @5P1  DIM
                                                              -->
```

Screen # 134

```
( arrays defining the modified adaptive filter ............. )
( ********************************************************** )
(              name  allotment   rows  columns   name         )
( ---------------------------------------------------------- )
FVARIABLE        AF2    72 ALLOT     3       3       AF2   DIM
FVARIABLE        CF2    24 ALLOT     1       3       CF2   DIM
FVARIABLE        XF2    24 ALLOT     3       1       XF2   DIM
FVARIABLE        QN2    72 ALLOT     3       3       QN2   DIM
FVARIABLE        RN2     8 ALLOT     1       1       RN2   DIM
FVARIABLE         P2    72 ALLOT     3       3        P2   DIM
FVARIABLE         R2   200 ALLOT     5       5        R2   DIM
FVARIABLE      GOLD2    40 ALLOT     5       1     GOLD2   DIM
FVARIABLE      SOLD2    40 ALLOT     5       1     SOLD2   DIM
FVARIABLE     PARAM2    40 ALLOT     5       1    PARAM2   DIM
FVARIABLE       MSE2    24 ALLOT     3       1      MSE2   DIM
                                                              -->
```

Screen # 135

```
( arrays defining the modified adaptive filter ............. )
( ********************************************************** )
(              name  allotment   rows  columns    name        )
( ---------------------------------------------------------- )
FVARIABLE        @1X2   24 ALLOT     3       1       @1X2  DIM
FVARIABLE        @2X2   24 ALLOT     3       1       @2X2  DIM
FVARIABLE        @3X2   24 ALLOT     3       1       @3X2  DIM
FVARIABLE        @4X2   24 ALLOT     3       1       @4X2  DIM
FVARIABLE        @5X2   24 ALLOT     3       1       @5X2  DIM
FVARIABLE        @1P2   72 ALLOT     3       3       @1P2  DIM
FVARIABLE        @2P2   72 ALLOT     3       3       @2P2  DIM
FVARIABLE        @3P2   72 ALLOT     3       3       @3P2  DIM
FVARIABLE        @4P2   72 ALLOT     3       3       @4P2  DIM
FVARIABLE        @5P2   72 ALLOT     3       3       @5P2  DIM
                                                              -->
```

Screen # 136

```
( arrays defining the conventional filter, inexact model .... )
( ************************************************************ )
(              name  allotment   rows  columns   name          )
( ------------------------------------------------------------ )
FVARIABLE        AF3   72 ALLOT     3       3        AF3  DIM
FVARIABLE        CF3   24 ALLOT     1       3        CF3  DIM
FVARIABLE        XF3   24 ALLOT     3       1        XF3  DIM
FVARIABLE        QN3   72 ALLOT     3       3        QN3  DIM
FVARIABLE        RN3    8 ALLOT     1       1        RN3  DIM
FVARIABLE         P3   72 ALLOT     3       3         P3  DIM
FVARIABLE     PARAM3   40 ALLOT     5       1     PARAM3  DIM
FVARIABLE       MSE3   24 ALLOT     3       1       MSE3  DIM
                                                             -->
```

Screen # 137

```
( arrays defining the conventional filter, exact model ...... )
( ************************************************************ )
(              name  allotment   rows  columns   name          )
( ------------------------------------------------------------ )
FVARIABLE        XF4   24 ALLOT     3       1        XF4  DIM
FVARIABLE        QN4   72 ALLOT     3       3        QN4  DIM
FVARIABLE        RN4    8 ALLOT     1       1        RN4  DIM
FVARIABLE         P4   72 ALLOT     3       3         P4  DIM
FVARIABLE       MSE4   24 ALLOT     3       1       MSE4  DIM


 VARIABLE        MSE
                                                             -->
```

Screen # 138

```
( prepare to iterate the original adaptive filter ........... )
: PREP1   AF1 AF !                 CF1 CF !    XF1 XF.!
          QN1 QN !    RN1 RN !     P1  P !     R1  R !
          @1X1 @1X !  @2X1 @2X !   @3X1 @3X !  @4X1 @4X !
          @5X1 @5X !  GOLD1 GOLD ! SOLD1 SOLD ! PARAM1 PARAM !
          @1P1 @1P !  @2P1 @2P !   @3P1 @3P !  @4P1 @4P !
          @5P1 @5P !  MSE1 MSE ! ;

( prepare to iterate the modified adaptive filter ........... )
: PREP2   AF2 AF !                 CF2 CF !    XF2 XF !
          QN2 QN !    RN2 RN !     P2  P !     R2  R !
          @1X2 @1X !  @2X2 @2X !   @3X2 @3X !  @4X2 @4X !
          @5X2 @5X !  GOLD2 GOLD ! SOLD2 SOLD ! PARAM2 PARAM !
          @1P2 @1P !  @2P2 @2P !   @3P2 @3P !  @4P2 @4P !
          @5P2 @5P    MSE2 MSE ! ;
                                                             -->
```

Screen # 139

```
( prepare to iterate the conventional filter, inexact model.. )
: PREP3   AF3 AF !                 CF3 CF !    XF3 XF !
          QN3 QN !    RN3 RN !     P3  P !     PARAM3 PARAM !
          MSE3 MSE ! ;

( prepare to iterate original adaptive filter .............. )
: PREP4   A   AF !                 C   CF !    XF4 XF !
          QN4 QN !    RN4 RN !     P4  P !     MSE4 MSE ! ;
                                                             -->
```

Screen # 140

```
( initialize the original adaptive filter .................. )
: IAFILTER1  PREP1 IAFILTER  ;

( initialize the modified adaptive filter .................. )
: IAFILTER2  PREP2 IAFILTER  ;

( initialize the conventional filter, inexact model ......... )
: ICFILTER3   PREP3  IPARAM  IAF          ICF
                     IRN  IQN  IP  IXF ;

( initialize the conventional filter, exact model ........... )
( note: in filter 4 the true error covariances must be entered)
: ICFILTER4   PREP4  RN* ID  QN* ID  IP IXF ;
                                                            -->
```

Screen # 141

```
( compute recursively the mean square state estimation error  )
: MSE* MSE @ ;
: CMSE    X XF* T1 SUB  T1 SQM  T1 MSE* T1 SUB
          KOUNT @ S>F 1/F T1 CMPY  T1 MSE* MSE* ADD ;

( compute the estimation error squared ...................... )
: CSE     X XF* MSE* SUB  MSE* SQM ;
                                                            -->
```

Screen # 142

```
( noise and parameter bounds ................................ )
: MSSG  CR
      SP ." process noise statistics"       NOISE1 STAT  CR CR
      SP ." measurement noise statistics"   NOISE2 STAT  CR CR
      SP ." parameter lower bound"          PARAML READ  CR CR
      SP ." parameter upper bound"          PARAMH READ  CR CR
      SP ." initial parameter estimate"     PARAM1 READ  CR ;
( list the status of the cumulative mean square error ....... )
: STATS    ( KOUNT @ 25 MOD 0= )
 ( IF   )                                           CR CR
    SP ." MEAN SQUARE ERROR AT TIME "         KOUNT @ . CR
    SP ." original adaptive filter"           MSE1 READ CR CR
    SP ." modified adaptive filter"           MSE2 READ CR CR
    SP ." conventional filter, inexact model" MSE3 READ CR CR
    SP ." conventional filter, exact model"   MSE4 READ CR CR
 ( THEN ) ;                                                 -->
```

Screen # 143

```
( status messages for model and filter algorithms ........... )
: STATM    ( KOUNT @ 25 MOD 0= )
        ( IF   )                                      CR CR
           SP ." MODEL RESPONSE AT TIME "  KOUNT @ .  CR
           SP ." system input "                W READ CR CR
           SP ." true system state"            X READ CR CR
           SP ." noisy system output"         NZ READ
        ( THEN ) ;

: STATCF   ( KOUNT @ 25 MOD 0= )
        ( IF   )                                      CR CR
           SP ." CONVENTIONAL FILTER No. "
              ." AT TIME "                 KOUNT @ .  CR
           SP ." kalman gain"                  K READ CR CR
           SP ." state estimate"             XF* READ
        ( THEN ) ;                                          -->
```

Screen # 144

```
( status messages for model and filter algorithms ........... )

: STATAF   ( KOUNT @ 25 MOD 0= )
     ( IF    )                                    CR CR
        SP ." ADAPTIVE FILTER No. "               .
           ." AT TIME "               KOUNT @ .    CR
        SP ." estimated parameter vector"  PARAM* READ CR CR
     (  SP ." kalman gain"                    K READ CR CR  )
     (  SP ." state estimate"               XF* READ        )
     ( THEN ) ;


( initialize the simulation ................................ )
: ISIM   IMODEL  IAFILTER1  IAFILTER2  ICFILTER3  ICFILTER4 ;
                                                          -->
```

Screen # 145

```
( simulate the filtering algorithms......................... )
: IT#1         1 KOUNT !    MODEL                   (   STATM  )
        PREP1 AFILTER1  X XF* MSE* SUB  MSE* SQM     1 STATAF
        PREP2 AFILTER2  X XF* MSE* SUB  MSE* SQM     2 STATAF
        PREP3 CFILTER   X XF* MSE* SUB  MSE* SQM   ( 3 STATCF )
        PREP4 CFILTER   X XF* MSE* SUB  MSE* SQM   ( 4 STATCF )
        STATS ;
: ITN   KOUNT @ 1+ KOUNT !     MODEL                (   STATM  )
                   PREP1 AFILTER1 CMSE               1 STATAF
                   PREP2 AFILTER2 CMSE               2 STATAF
                   PREP3 CFILTER  CMSE             ( 3 STATCF )
                   PREP4 CFILTER  CMSE             ( 4 STATCF )
                                                     STATS    ;
: RUN    ( PRINTER ) CR CR  ISIM   MSSG    IT#1
          1001 2 DO  ITN  LOOP 10 1000 BEEP ( CONSOLE ) ;
```

Screen # 146

```
( alternate definitions .................................... )

( mult. correction by accel. gain seq. and find new parameter )
: GAMA-A  S1 !  KOUNT @ 1 =
          IF  FDUP F0>  S1 @ 1 SOLD* DSEEK !
              ALPHA @  S1 @ 1 GOLD* DSEEK !
          THEN
        FDUP F0> DUP S1 @ 1 SOLD* DSEEK DUP @ ROT ROT ! =
          IF        1 S1 @ GOLD* DSEEK @
           ELSE     1 S1 @ GOLD* DSEEK DUP @ 1+ DUP ROT !
          THEN      S>F 1/F F*
                    S1 @ 1 PARAM* DSEEK F@ FSWAP F- ;
```

Screen # 147

```
( form gradient matrix from derivatives of residual vectors   )
: PGRAD       @1RES   GRAD   1 VMAT
              @2RES   GRAD   2 VMAT
              @3RES   GRAD   3 VMAT
              @4RES   GRAD   4 VMAT
              @5RES   GRAD   5 VMAT ;


( form a row vector containing partials of covariance matrix  )
: P@SROW    RES T1 TRSP    0.5 E 0 T1 CMPY
            T1 @1S T2 MPY   T2 RES T3 MPY   T3 @SROW  1 VMAT
            T1 @2S T2 MPY   T2 RES T3 MPY   T3 @SROW  2 VMAT
            T1 @3S T2 MPY   T2 RES T3 MPY   T3 @SROW  3 VMAT
            T1 @4S T2 MPY   T2 RES T3 MPY   T3 @SROW  4 VMAT
            T1 @5S T2 MPY   T2 RES T3 MPY   T3 @SROW  5 VMAT ;
```

Screen # 148

```
( update parameters, unweighted stochastic gradient approach  )
: UPPARAMX   CRES  P@#RES   PGRAD   GRAD T1 TRSP
             T1 RES T2 MPY   GAMA-N 1/F T2 CMPY
             PARAM* T2 T1 SUB
             6 1 DO   I 1 T1 DSEEK F@  I CLAMP  LOOP ;

( update parameters, weighted stochastic approx. approach ... )
: UPPARAMX   CRES  P@#RES  CS  P@#S  PGRAD  P@SROW
             GRAD T1 TRSP    T1 S T2 MPY    T2 RES T1 MPY
             @SROW T2 TRSP   T1 T2 T3 ADD
             GAMA-N 1/F T3 CMPY  PARAM* T3 T2 SUB
             6 1 DO   I 1 T2 DSEEK F@  I CLAMP  LOOP ;
```

Screen # 149

```
( compute approximate Hessian matrix ....................... )
: PHESS   GRAD T1 TRSP  T1 S T2 MPY  T2 GRAD T1 MPY
          T1 R* T2 SUB  T1 ID   0.01 E 0 T1 CMPY  T1 T2 T3 ADD
          GAMA-N 1/F T3 CMPY   R* T3 R* ADD ;



: GR  6 1 DO  I 1 GOLD* DSEEK @ CR . LOOP ;
: SR  6 1 DO  I 1 SOLD* DSEEK @ CR . LOOP ;

: ANS  PRINTER  STATS  PREP1 1 STATAF
                       PREP2 2 STATAF  CONSOLE ;
```

Screen # 150

```
( update parameters, unweighted recursive prediction error     )
: UPPARAMX   CRES  P@#RES  PGRAD  CS  PHESS
             GRAD T1 TRSP     T1 S T2 MPY    T2 RES T1 MPY
               R* T2 MOVE          T2 INVERT
             T2 T1 T3 MPY   GAMA-N 1/F T3 CMPY
             PARAM* T3 T2 SUB
             6 1 DO   I 1 T2 DSEEK F@  I CLAMP  LOOP ;
( update parameters, weighted recursive prediction error       )
: UPPARAMX   CRES  P@#RES  PGRAD  CS  P@#S  PHESS  P@SROW
             GRAD T1 TRSP     T1 S T2 MPY    T2 RES T1 MPY
             @SROW T2 TRSP   T1 T2 T3 ADD
               R* T2 MOVE          T2 INVERT
             T2 T3 T1 MPY   GAMA-N 1/F T1 CMPY
             PARAM* T1 T2 SUB
             6 1 DO   I 1 T2 DSEEK F@  I CLAMP  LOOP ;
```

Screen # 151

VITA

Peter Hansen was born in Chicago, Illinois, on March 30, 1956. He attended the University of Illinois at Urbana, and received a Bachelor of Science degree in Electrical Engineering in 1978.

Following the completion of his undergraduate education, he served in the United States Navy for almost six years. His experiences included teaching at the Naval Nuclear Power School, and serving aboard a Fleet Ballistic Missile Submarine.

In the winter of 1983 he began his graduate studies at The University of Tennessee in Knoxville. He was awarded the Master's degree in August 1985.