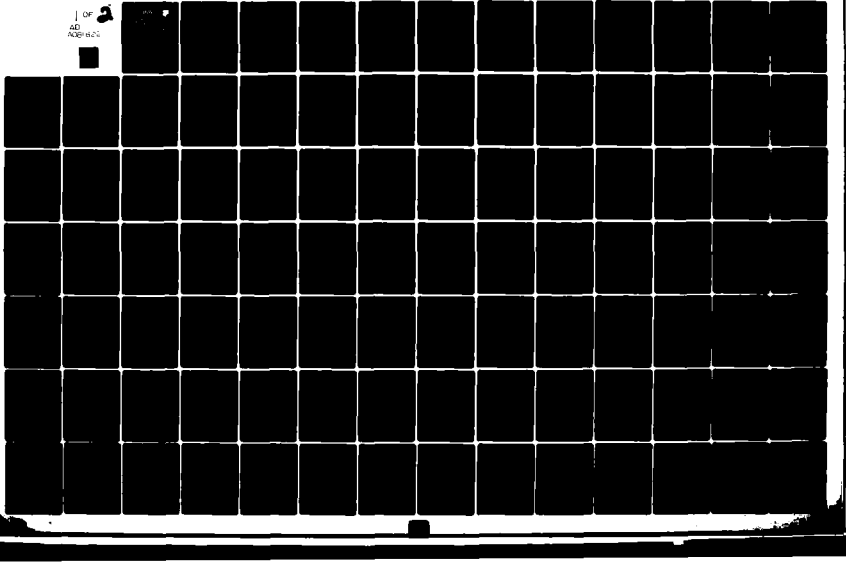
AD-A081 822 OHIO STATE UNIV COLUMBUS ELECTROSCIENCE LAB F/G 9/5
THE PERFORMANCE OF A SAMPLED DATA DELAY LOCK LOOP IMPLEMENTED W--ETC(U)
JAN 80 H S EILTS F30602-79-C-0068
UNCLASSIFIED ESL-711679-1 RADC -TR-79-333 NL



RADC-TR-79-333
Interim Report
January 1980

LEVEL II 12

THE PERFORMANCE OF A SAMPLED DATA DELAY LOCK LOOP IMPLEMENTED WITH A KALMAN LOOP FILTER

The Ohio State University ElectroScience Laboratory

H. S. Eilts

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-79-333 has been reviewed and is approved for publication.

APPROVED: Stuart H. Talbot

STUART H. TALBOT
Project Engineer

APPROVED: Fred I Diamond

FRED I. DIAMOND
Technical Director
Communications and Control Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (DCCR), Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return this copy. Retain or destroy.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: RADC-TR-79-333	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): THE PERFORMANCE OF A SAMPLED DATA DELAY LOCK LOOP IMPLEMENTED WITH A KALMAN LOOP FILTER	5. TYPE OF REPORT & PERIOD COVERED: Interim Report, Dec 78 - Mar 79
6. PERFORMING ORG. REPORT NUMBER: ESL-711679-1	
7. AUTHOR(s): H. S. Eilts	8. CONTRACT OR GRANT NUMBER(s): F30602-79-C-0068
9. PERFORMING ORGANIZATION NAME AND ADDRESS: The Ohio State University ElectroScience Laboratory, Department of Electrical Engineering, Columbus OH 43212	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 62702F, 45196308
11. CONTROLLING OFFICE NAME AND ADDRESS: Rome Air Development Center (DCCR), Griffiss AFB NY 13441	12. REPORT DATE: Jan [redacted] 80
	13. NUMBER OF PAGES: 98
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Same	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION DOWNGRADING SCHEDULE: N/A

16. DISTRIBUTION STATEMENT (of this Report)
Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)
Same

18. SUPPLEMENTARY NOTES
RADC Project Engineer: Stuart H. Talbot (DCCR)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Time Division Multiple Access	Kalman filter
TDMA	Tracking filter
Delay Lock Loop	Optimum filter
Sampled Data Delay Lock Loop	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The purpose of this study is to evaluate the steady-state and transient (lock up) performance of a tracking loop implemented with a Kalman filter. Steady-state performance criteria are errors due to measurement noise (jitter) and doppler errors due to motion of the tracking loop. Trade-offs exist between the two criteria such that increasing performance with respect to either one will cause a performance decrease with respect to the other. It is shown that by carefully selecting filter parameters reasonable performance can be obtained for both criteria simultaneously. It is also shown that lock-up performance for the loop is acceptable when these parameters are used.

DD FORM 1 JAN 73 1473

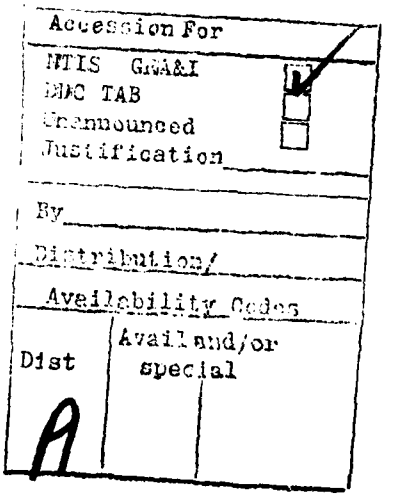
UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

PREFACE

This report, OSURF Report number 711679-1, was prepared by the ElectroScience Laboratory, Department of Electrical Engineering, The Ohio State University at Columbus, Ohio. Research was conducted under Contract F30602-79-C-0068. Mr. Stuart Talbot was the RADC Program Monitor for this research.

The material contained in this report is also used as a thesis submitted to the Department of Electrical Engineering, The Ohio State University as partial fulfillment for the degree Master of Science.



CONTENTS

CHAPTER I
INTRODUCTION

This study describes the performance of a Kalman filter implementation of a sample data delay lock loop. The (non-sampled) delay lock loop was originally studied by Spilker[17,18] and Gill[16] for use in the tracking of pseudo-noise (PN) coded signals. A sampled data version of this loop was developed by Huff et al.[19,20,21] for receiver timing in a time division multiple access (TDMA) satellite communications system.

To facilitate receiver timing in this system, a PN coded clock pulse is periodically transmitted from the satellite. In the receiver the sample data delay lock loop "tracks" the clock pulse by controlling the frequency and phase of a local square wave oscillator (local clock). In the interval between clock pulses, the transitions of the local clock are used to time the correlators in the receiver detection circuits. Errors in clock tracking (biases and/or jitter) result in imperfect timing of the detectors with a corresponding increase in bit error probability[7,22]. Thus, it is desirable to minimize the bias and error variance (jitter) of the local clock. The desire to increase PN code ratios to 40 MHz makes the timing requirement more critical. Furthermore, the incorporation of highly maneuverable terminals (i.e., aircraft) in the system compounds the tracking problem by adding time variable doppler effects.

In order to achieve the desired tracking accuracy the Kalman filter algorithm was chosen[6]. This algorithm is known to yield both optimum estimates (unbiased, minimum variance estimates) and a measure of its own performance, an error covariance matrix[1,2,3,4]. In order to be implemented, however, the Kalman filter does require a mathematical model. This is a set of equations which describes the process to be filtered. The error variances contained in the error covariance matrix are the error variances for the modeled (as opposed to the actual) situation. One should have some knowledge concerning model accuracy before basing a design upon the error variances of the model. The model chosen for the tracking filter application is a statistical representation of the time variable propagation delays generated by terminal motion, specifically aircraft flight. Checks on the accuracy of the model would involve comparisons of the error variances yielded by the algorithm with the error variances actually obtained by implementing the filter in a moving terminal. This is beyond the scope of this study.

The purpose of this study is to evaluate the performance of a Kalman tracking filter independently of the question of model accuracy. Given the mathematical model, the structure and gains of the filter are determined by the algorithm. The resulting filter may then be analyzed for its response to deterministic inputs and noise using conventional techniques. Thus, the model is used only to determine the filter structure and not to determine performance. Because the Kalman filter is in state variable form, the most convenient technique for analysis is computer simulation. Other techniques include state variable techniques and z-transform methods. Since the Kalman filter is linear, superposition may be used to determine the response to linear combinations of inputs.

Chapter II of this study presents an overview of the SDDLL, Kalman theory, the model chosen for this application, and some discussion of computer simulation. In Chapter III the steady-state response of the Kalman filter to input noise is analyzed using the state variable techniques. Monte Carlo simulation is used to verify the results. In addition, an analysis using z-transform techniques is presented. The steady-state response of the Kalman tracking filter to deterministic delay changes caused by a terminal maneuvering through 180^{o} turns is presented in Chapter IV. In Chapter V the combined response to noise and maneuvers is considered. Three examples are presented which illustrate the usefulness of the preceeding data. Chapter VI documents an analysis of the lockup characteristics of the tracking loop. Here the response to step, ramp, and parabolic delay changes is determined for Kalman tracking filters in both the transient and the steady state. The number of iterations required for lock-up (convergence) for special cases of interest is also determined. Chapter VII presents the conclusions of the study.

CHAPTER II
KALMAN FILTERING AND THE TDMA CLOCK LOOP

A. Introduction

In this chapter a short review of the TDMA clock loop is presented. This is followed by an overview of Kalman filter theory for the purpose of introducing the notation, equations, and concepts to be used in subsequent analyses.

B. The TDMA Clock Loop

The OSU/RADC TDMA system uses pseudo-noise (PN) coding for improved multipath and anti-jam performance[23]. A pulsed envelope PN coded "clock" pulse is periodically transmitted on the down-link to facilitate PN code tracking at the receiver. Figure 1 shows the envelope of the clock pulse. Here, Δ is the chip[1] duration, M is the number of chips per clock pulse, T_s is the duration of the clock pulse, and T_f is interval between leading edges of successive clock pulses.

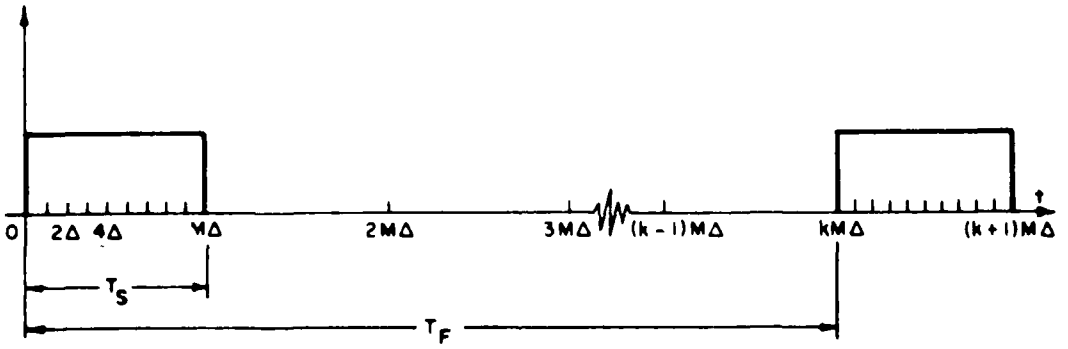


Figure 1. The envelope of the pseudo-random network clock signal.

[1] A chip is the elementary unit of a PN code and is analogous to a bit in a binary data stream.

Tracking of the PN code is accomplished at the receiver by a sampled-data delay-lock loop (SDDLL)[16,17,18,19,20,21], a block diagram of which is shown in Figure 2. The function of this tracking loop is to phase lock a locally generated replica (i.e., the "local clock") of the clock pulse with the down link signal. The local clock signal is generated by a pulse generator and feedback shift register. Delayed and advanced versions of the local clock are correlated with the down-link clock signal in the upper and lower mixer-bandpass filter-detector circuits, respectively. The difference of the correlator outputs is then sampled, yielding the sampled error voltage E_s. This is shown in Figure 3 as a function of the timing error (ϵ) between the received and local clock pulses. The error voltage is then filtered by the discrete loop filter. This filter can be implemented with a variety of algorithms. The present system uses a simple averaging filter. This study considers implementation with a Kalman algorithm. The output of the discrete filter is then fed to the clock correction circuitry to adjust the time base of the local clock.

C. Kalman Filter Theory

The Kalman estimator algorithms are optimum algorithms in that they yield minimum variance, unbiased estimates for gaussian processes. Unlike the Wiener filter, which produces optimum estimates in the steady-state, the Kalman algorithms produce optimum transient as well as optimum steady-state response. They also yield optimum estimates for non-stationary or periodically stationary processes for which a steady-state does not exist. Because of the feature of optimality the Kalman algorithms were chosen as a method for potentially upgrading the performance of the SDDLL.

These algorithms are cast in state variable form and require mathematical models of both the random process to be filtered and of the observations of the process. The random process should be modeled in the form

$$z(k+1) = \Phi(k)z(k) + \Gamma(k)u(k) \qquad (1)$$

where k is the iteration number and where

$z(k) \triangleq$ n dimensional state vector,

$u(k) \triangleq$ r dimensional zero mean white process noise vector,

$\Gamma(k) \triangleq$ nxr dimensional forcing matrix,

and

$\Phi(k) \triangleq$ nxn dimensional state transition matrix operating on the kth state

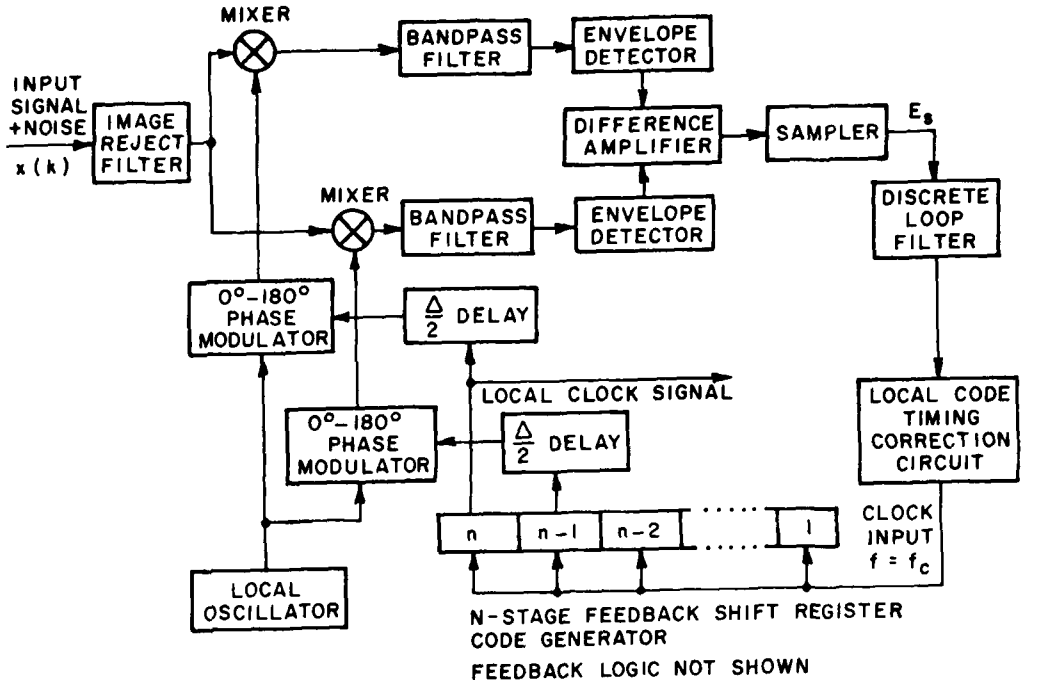


Figure 2. Block diagram of the sampled data delay lock loop (SDDLL)

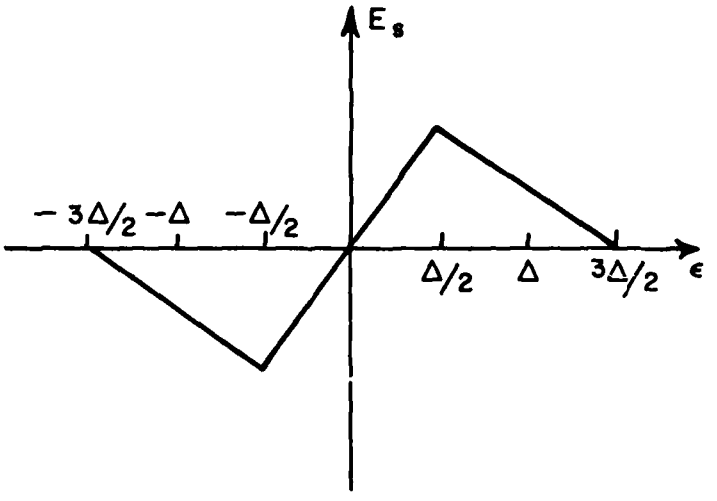


Figure 3. Sampled error voltage (Es) as a function of timing error(ϵ)

The observation model should be in the form

$$x(k) = H(k)\, z(k) + v(k) \qquad (2)$$

where $x(k) \triangleq$ s dimensional observation vector,
$v(k) \triangleq$ s dimensional zero mean White measurement, noise vector,

and

$H(k) \triangleq$ sxn dimensional observation matrix.

Together these two equations model the input to the algorithms. The first equation generates the state vector z. From Equation (2) the observation of the process is composed of a linear combination of the states (H(k)z(k)) plus additive white measurement noise (v(k)). The covariance matrices of the two noise vectors and v(k) are also required for implementation of the algorithm. These are defined by

$E\{u(j)u^T(k)\} \triangleq V_u(k)$, the rxr dimensional process noise covariance matrix,

and

$E\{v(j)u^T(k)\} \triangleq V_v(k)$, the sxs dimensional observation noise covariance matrix.

Here E is the expectation operation and T represents transposition. Thus, the models required for implementation of the Kalman algorithms are completely determined by specifying the matrices $\Phi(k)$, $\Gamma(k)$, $H(k)$, $V_u(k)$, and $V_v(k)$.

The two Kalman algorithms investigated for use in the SDDLL are the Kalman one step predictor algorithm and the Kalman filter algorithm. The one step predictor equation is given by[1,2,3,4]

$$\tilde{z}(k+1/k) = \Phi(k)\tilde{z}(k/k-1)+K_p(k)[x(k)-H(k)\tilde{z}(k/k-1)], \qquad (3)$$

where

$\tilde{z}(k+1/k) \triangleq$ the n dimensional prediction of the state vector at the (k+1)st iteration given k data samples

and

$K_p(k) \triangleq$ the nxs dimensional predictor gain matrix.

This algorithm is iterative; in order to generate the prediction of the (k+1)st state, the prediction of the kth state is required. Thus, to start the algorithm, an initial estimate must be obtained by means other than Equation (3). Also, notice that the term in brackets $(x(k)-H(k)\tilde{z}(k/k-1))$ is the observation minus the prediction observation. This term is sometimes called the innovation and represents the new information contained in the measurement. In the present application the observation will be modeled as noise corrupted propagation delay from a satellite. A measurement of the propagation delay

is impossible in the SDDLL, which can only measure the difference in the time bases of the downlink and local clock signals. However, by forcing the local clock to follow the predicted delay, a measurement of the term in brackets (i.e., delay minus predicted delay, the innovation) is obtained[1]. Equation (3) states that the new prediction ($\tilde{z}(k+1/k)$) is obtained by advancing the old prediction ($\tilde{z}(k/k1)$) one step ahead with the state transition matrix ($\Phi(k)$) and adding weighted (by the gain matrix, $K_p(k)$ amounts of new information (innovation).

The gain matrix needed for use in Equation (3) is given by

$$K_p(k) = \Phi(k)\tilde{P}(k)H^T(k)[H(k)\tilde{P}(k)H^T(k)+V_v(k)]^{-1} , \qquad (4)$$

Here, $\Phi(k)$, $H(k)$, and $V_v(k)$ were specified for the process and observation models. $\tilde{P}(k)$ is the prediction error covariance matrix, defined by

$$\tilde{P}(k) \triangleq E\{\tilde{z}_\epsilon(k)\tilde{z}_\epsilon^T(k)\}, \text{ the nxn dimensional prediction error covariance matrix,}$$

where

$$\tilde{z}_\epsilon(k) \triangleq z(k)-\tilde{z}(k/k-1), \text{ the one step prediction error at the kth point given k-1 data samples.}$$

The error matrix may be calculated iteratively via the equation

$$\tilde{P}(k+1) = \Phi(k)\tilde{P}(k)\Phi^T(k)+\Gamma(k)V_u(k)\Gamma^T(k) \qquad (5)$$
$$- \Phi(k)\tilde{P}(k)H^T(k)[H(k)\tilde{P}(k)H^T(k)+V_v(k)]^{-1} H(k)\tilde{P}(k)\Phi^T(k).$$

This equation is a function of $\Phi(k)$,$\Gamma(k)$, $H(k)$, $V_v(k)$ and $V_u(k)$, all of which are specified for the process and observation models. Since Equation (5) is iterative, an initial error covariance matrix is needed for the first iteration. This matrix may be obtained from an error analysis of the initial estimate used to begin the iterations of Equation (3). If the process and observation models are stationary (non time varying) the error covariance matrix (and hence also the predictor gain matrix) will approach asymptotic or steady-state values as the iterations proceed. In addition to its use for computing the gain matrix (Equation (4)), the error covariance matrix also provides a measure of the predictor performance in response to the modeled process. The diagonal elements of this matrix are the variances of

---

[1]To avoid confusion, the term observation will be used exclusively to represent x(k) and the term measurement will be used exclusively to represent the error signal obtained from the SDDLL (i.e., the innovation).

the prediction error vector $\tilde{z}_\epsilon$. If the process model accurately reflects the physical situation, then the error convariances obtained in an actual implementation will approach those given theoretically by Equation (5).

Equations (3), (4), and (5) are collectively known as the Kalman one step predictor algorithm. Equation (3) forms the actual prediction. A block diagram of the equation is shown in Figure 4. Equations (4) and (5) are used for the calculation of the gain matrix used in Equation (3). These equations can be used in either of two methods. Figure 5 shows a block diagram for implementing the entire prediction algorithm in real time. After updating the prediction, the error covariance and gain matrices are calculated prior to taking the next measurement. The measurement is then obtained, the prediction updated, and the process repeated. This method is possible as long as the time interval between measurements is long enough to allow the necessary computations. Alternately, since the error covariance and gain matrices are functions only of the process and measurement models and not of the actual measurements, they may both be computed and stored prior to enabling the predictor equation. With the gain matrix for each iteration available in memory, a considerable savings in computation time can be achieved for the operation of the predictor equation.

The optimal filtered estimate is related to the optimal one step prediction by[1,2]

$$\hat{z}(k-1) = \Phi^{-1}(k)\tilde{z}(k/k-1) \tag{6}$$

or

$$\tilde{z}(k/k-1) = \Phi(k-1)\hat{z}(k-1) \tag{7}$$

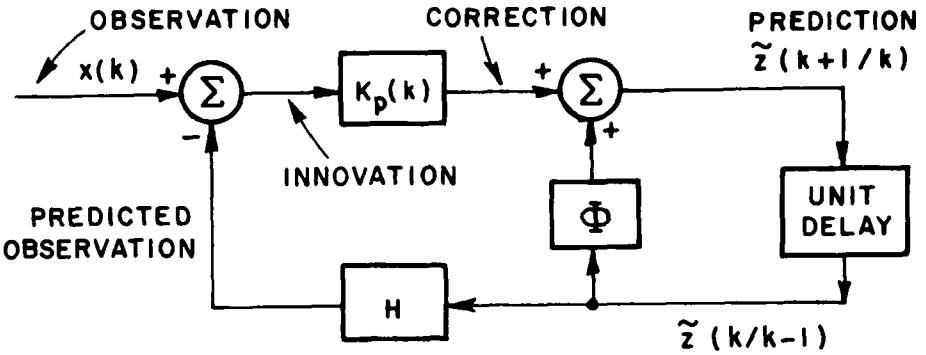
where $\Phi^{-1}(k)$ may be thought of as the reverse state transition matrix (i.e., the transition matrix going from the kth state to the (k-1)st state) and where

$\hat{z}(k-1) \triangleq$ the (n dimensional vector) filtered estimate of the state z(k-1) after processing the (k-1) data sample.

Equations (3), (6), and (7) may be used to derive Equation (8), the Kalman filter equation:

$$\hat{z}(k) = \Phi(k-1)\hat{z}(k-1)+K_f(k)[x(k)-H(k)\Phi(k-1)\hat{z}(k-1)]. \tag{8}$$

Here $K_f(k)$ is the filter gain matrix for the kth data point. Note that the term $\Phi(k-1)\hat{z}(k-1)$ is the one step prediction $\tilde{z}(k/k-1)$. Furthermore, the term inside the brackets is again the innovation. Equation (8) states that the filtered estimate is obtained from a sum of two components. The first component is the previous filtered estimate advanced one step forward by the state transition matrix. The



$$\tilde{z}(k+1/k) = \Phi \tilde{z}(k/k-1) + K_p(k)\left[x(k) - H\tilde{z}(k/k-1)\right]$$

Figure 4. Block diagram of the Kalman Predictor Equation

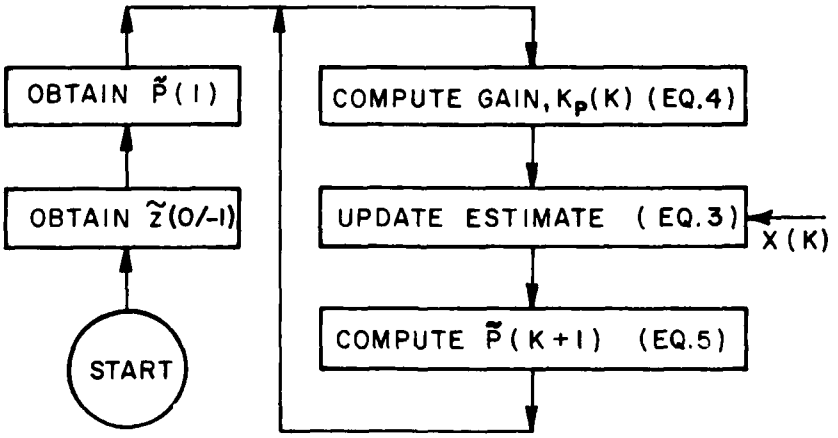


Figure 5. Flow chart for the Kalman Predictor Algorithm

second term is the innovation (new information) weighted by the filter gain matrix $K_f(k)$.

The filter gain matrix is given by

$$K_f(k) = \Phi^{-1}(k)K_p(k) = \tilde{P}(k)H^T(k)[H(k)\tilde{P}(k)H^T(k)+V_v(k)]^{-1}. \quad (9)$$

As in the case of the predictor gain matrix, the filter gain matrix is also a function of the prediction error covariance matrix. The prediction error covariance matrix is calculated iteratively in Equation (5), Equations (8), (9), and (5) collectively from the Kalman filter algorithm[1,2]. These equations are used in a manner analogous to the predictor algorithm with Equations (5) and (9) being used to compute the gain matrix necessary for the computation of the filtered estimate (Equation (8)).

The error covariance matrix for the filtered (as opposed to the predicted) estimate is defined by

$$\hat{P}(k) \triangleq E[\hat{z}_\epsilon(k)\hat{z}_\epsilon^T(k)]$$

where

$$\hat{z}_\epsilon(k) = z(k)-\hat{z}(k)$$

is the n dimensional filter error vector at the kth iteration. The filter error covariance matrix may be obtained from the predictor error covariance matrix via the equation

$$\hat{P}(k) = [I-K_f(k)H(k)]\tilde{P}(k) . \quad (10)$$

The matrix $\hat{P}(k)$ is a measure of filter performance just as the matrix $\tilde{P}(k)$ is a measure of predictor performance. Both $\tilde{P}(k)$ and $\hat{P}(k)$ are error covariance matrices for the modeled situation. The values given by these matrices may be considered accurate only if the model is accurate in its representation of the actual process. Thus, the accuracy of a given process model must be established before the error covariance matrices can be considered reliable.

Fortunately, it is possible to evaluate the performance of the Kalman algorithms without establishing the accuracy of the process model. Suppose the process and observation models are specified. This determines the structure and gain of the resulting predictor and filter equations (Equations (3) and (8), respectively). The responses of these equations to various inputs (such as steps, ramps, white noise, etc.) can then be determined using conventional techniques. Note that the responses obtained can be considered true responses even if the process model does not accurately represent the random process (i.e., even if there are modeling errors). This is so because the structure and gain of the Kalman filter and/or predictor are functions only of the model, not of the model accuracy.

For example, suppose a process model is chosen which accurately models the propagation delays generated by a slowly moving mobile terminal (such as an automobile). Using this process model, the step response of Equation (3) (the predictor equation) is determined. Now, suppose the same process model is used to represent the propagation delays generated by a rapidly moving mobile terminal (such as an aircraft). Obviously, some modeling errors will probably be present. However, the step response will be the same simply because the same process model is used.

This technique is used in the remainder of this study to obtain performance data for a wide variety of inputs. All of the data presented in the sequel can be considered valid regardless of the process and observation model accuracies. This is generally not true of the performance data contained in the previous study[6], most of which is valid only if the process and observation models are accurate.

D. The Process and Observation Models for the Clock Loop

The process model for the Kalman clock loop filter is chosen from Singer and Behnke[5] and models delay and the first two derivatives of delay with respect to time as a random process. Specifically,

$$T_o(k+1) = T_o(k)+\dot{T}_o(k)T_f+\ddot{T}_o(k)T_f^2/2, \tag{11a}$$

$$\dot{T}_o(k+1) = \dot{T}_o(k) + \ddot{T}_o T_f, \tag{11b}$$

and

$$\ddot{T}_o(k+1) \triangleq \rho\ddot{T}_o(k) + u(k) \tag{11c}$$

where

$T_o(k) \triangleq \frac{r(k)}{c}$ = downlink propagation delay at the kth sample instant,

$\dot{T}_o(k) \triangleq \frac{dT_o(k)}{dt}$ = downlink propagation delay rate at the kth sample sample instant (delay velocity),

$\ddot{T}_o(k) \triangleq \frac{d^2T_o(k)}{dt^2}$ = downlink propagation delay acceleration at the kth sample instant,

$c \triangleq$ speed of light,

$r(k) \triangleq$ downlink range at kth sample instant,

and

$$\rho \triangleq E\{\ddot{T}(t+\tau) \cdot \ddot{T}(t)\}\big|_{\tau=T_f} = \text{delay acceleration correlation coefficient,}$$

Equations (11a) and (11b) are Taylor's series representations for delay (three term Taylor series) and delay rate (two term Taylor series). Equation (11c) models acceleration as a correlated random process. The coefficient is chosen from an exponential correlation function so that

$$\rho(\tau) = e^{-\lambda\tau} = 1-\lambda\tau + \frac{(\lambda\tau)^2}{2!} - \frac{(\lambda\tau)^3}{3!} + \ldots \qquad (12)$$

where λ has units of sec^{-1} and is considered to be the inverse of the average maneuver duration[5,6]. For $\tau=T_f$ and $\lambda T_f << 1$, ρ is given approximately by

$$\rho \doteq 1 - \lambda T_f \ . \qquad (13)$$

Equations (11a), (11b), and (11c) are in the form of Equation (1) where

$$z(k) = \begin{pmatrix} T_o(k) \\ \dot{T}_o(k) \\ \ddot{T}_o(k) \end{pmatrix} = \text{state vector,} \qquad (14a)$$

$$\Phi(k) = \Phi = \begin{pmatrix} 1 & T_f & T_f^2/2 \\ 0 & 1 & T_f \\ 0 & 0 & \rho \end{pmatrix} = \text{constant state transition matrix,} \qquad (14b)$$

$$\Gamma(k) = \Gamma = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \text{constant forcing matrix,} \qquad (14c)$$

and

$u(k)$ = scalar white process noise with constant variance σ_u^2 (14d)

Note that the vector $u(k)$ in Equation (1) is of dimension one here (i.e., a scalar). Hence, the covariance matrix $V_u(k)$ in the Kalman algorithms (Equations (5) and 9)) reduces to the scalar σ_u^2. Squaring both sides of Equation (11c) and taking expected values yields

$$\sigma_{\ddot{T}}^2(1-\rho^2) = \sigma_u^2 \ . \tag{15}$$

Thus, σ_u^2 is easily found given ρ and $\sigma_{\ddot{T}}^2$. To find $\sigma_{\ddot{T}}^2$ the probability density function shown in Figure 6 is assumed[5,6]. The quantity A_T in this figure represents the maximum delay acceleration given by

$$A_T = A/c \tag{16}$$

where

A = maximum terminal acceleration (m/sec^2),

and

c = speed of light (m/sec).

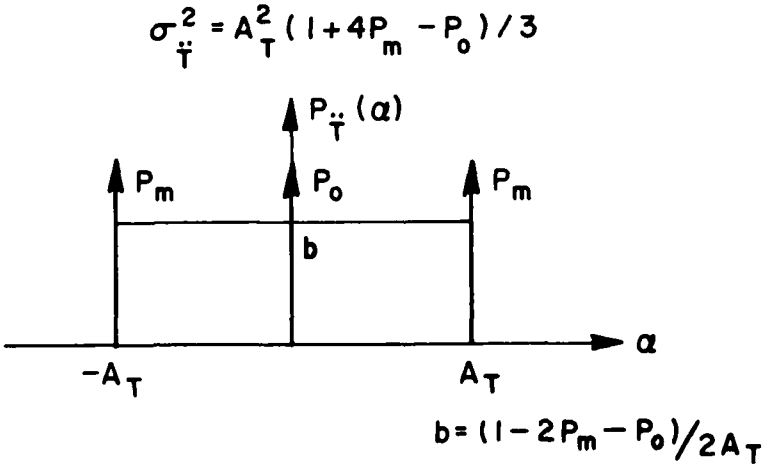


Figure 6. The Probability Density Function Function for Delay Acceleration

From this density function, σ_T^2 is easily found to be

$$\sigma_T^2 = \frac{A_T^2}{3}(1+4P_m-P_o) \tag{17}$$

where

P_m = probability of maximum acceleration

and

P_o = probability of zero acceleration.

The observation is modeled as

$$x(k) = T_o(k) + v(k). \tag{18}$$

This is in the form of Equation (2) where x(k) is now a scalar and where

$v(k)$ = scalar white measurement noise with stationary variance σ_v^2 (19a)

and

$H(k) = H = (1\ 0\ 0)$ = constant observation matrix. (19b)

Since v is a scalar, the covariance matrix $V_v(k)$ in Equations (4), (5), and (8) becomes σ_v^2. In Reference 7, Equation (39) it is shown that the variance of a single unprocessed measurement (sample) in the SDDLL is

$$\sigma_v^2 = \frac{1}{4}\left[\frac{1.215}{E/N_o} + C\ \frac{1.476}{(E/N_o)^2}\right] \tag{20}$$

where $E \triangleq$ received energy in the clock pulse (joules),

$N_o \triangleq$ one sided noise power spectral density (watts/Hz),

and

$$C = \begin{cases} 1 \text{ for linear detection} \\ 2 \text{ for square law detection.} \end{cases}$$

Thus, σ_v^2 can be obtained given a knowledge of E/N_o.

Equations (1), (2), (14a,b,c,d), and (19a,b) together with variances σ_u^2 and σ_v^2 completely define the model for the Kalman filter considered subsequently. This model is a function of four parameters: T_f, σ_v^2, σ_u^2, and λ (or ρ), Neither the equations nor the parameters vary with time; hence the model is stationary. A typical aircraft flight might consist predominantly of mild maneuvers with perhaps more severe manuevers occasionally. In other words, typical aircraft flight is non-stationary. This presents problems to the designer. If the selection of model parameters is based upon average maneuvers, the tracking error may become excessively large during severe maneuvers. Alternately, λ and σ_T^2 (and hence σ_u^2) may be selected to be representative of more severe maneuvers, perhaps with some performance degradation for less severe maneuvers[1]. Furthermore, even if a maneuver is specified, an appropriate selection of the statistical parameters λ and σ_T^2 to optimally represent the maneuver is not obvious. To confidently select usable model parameters, the designer needs to know either (1) the relationships of the model parameters to the performance of the aircraft or (2) the relationships of the model parameters to the filter behavior when subjected to a wide variety of inputs.

E. Computer Simulation

The problem of simulating a filter's response to an input signal may be broken into two parts, (1) simulating the input signal and (2) simulating the filter. In the Kalman filter SDDLL described in Reference 6 the time base of the local clock serves the function of both the optimal estimate and the optimal prediction of the delay function $T_Q(t)$. Immediately after the processing of a new sample, the time base of local clock serves as the filtered estimate. Between data samples, the time base is the optimum prediction. Immediately before the next data sample the local time base serves as the optimal one-step prediction. This is the "worst case" point for the filter, the time at which the largest tracking errors generally occur. To investigate the worst case behavior of the filter, the computer programs used in the remainder of this report simulate a Kalman one-step predictor. This is easily accomplished by direct simulation of Equations (3), (4), and (5).

Simulation of the predictor during transient studies requires an initial prediction vector $\tilde{z}(0/-1)$ and an initial covariance matrix $\tilde{P}(o)$.

---

[1] It should be noted that an adaptive scheme, perhaps with accelerometer inputs, could be used to adjust the model parameters to maintain an optimal filter for a wide variety of maneuvers. This, however, is beyond the scope of this report.

For steady-state studies, the covariance matrix may be iterated using Equation (5) until steady-state is achieved. The predictor gain vector may then be computed using Equation (4). Alternately, one may compute the steady state gain vector and covariance matrix using the equations given in Reference 6. The later was not done because (1) the computer programs described subsequently were intended to be general (i.e., used both for transient and steady-state simulations) and (2) solution by iteration serves as a useful check on the analytic solutions given in Reference 6.

CHAPTER III
STEADY-STATE RESPONSE TO NOISE

Since the Kalman predictor is linear, superposition may be used to find the response to a sum of a deterministic input plus noise. That is, one may find the response due to noise alone and then separately find the response due to the deterministic input (without noise). The total response is the sum of the individual response. This technique was used in this study, and this chapter documents the analysis of the response to noise.

A. State-Variable Approach

In state-variable terminology, the solution of the predictor response to measurement noise alone is accomplished by constraining the state vector to be zero for all iterations. The prediction error is then given by

$$\tilde{z}_{\varepsilon_n}(k) = -\tilde{z}_n(k/k-1) \tag{21}$$

where the subscript n indicates the contribution due to noise only. Thus, under the preceding constraint, the error covariance is the prediction covariance. That is

$$E\{\tilde{z}_{\varepsilon_n}(k) \cdot \tilde{z}^T_{\varepsilon_n}(k)\} = E\{\tilde{z}_n(k/k-1)\tilde{z}^T_n(k/k-1)\} \tag{22}$$

The prediction is given by the recursion relation (from Equation (3))

$$\tilde{z}(k+1/k) = (\Phi - k_p(k)H)\,\tilde{z}(k/k+1) + K_p(k)x(k) \tag{23}$$

where x(k) is the observation.

For the purpose of computing the gains, the problem is modeled normally by specifying the matrices given in Equation (14) and the variances σ_u^2 and σ_v^2. When computing the prediction, the observation x(k) is composed only of white noise (ξ) with variance σ_ξ^2. Here we are allowing for the possibility that the noise may not have the modeled variance (σ_v^2) but instead may have a different variance (σ_ξ^2). ξ and v both represent measurement noise, but ξ is necessary as a dummy variable to allow for noise variance other than those modeled. Equation (23) then becomes

$$\tilde{z}_n(k+1/k) = (\Phi - k_p(k)H)\tilde{z}_n(k/k+1) + K_p(k)\xi(k) \tag{24}$$

Under the preceding assumptions, the error covariance matrix (i.e., $E\{\tilde{z}_{\epsilon n}(k+1)\cdot\tilde{z}^{T}_{\epsilon n}(k+1)\}$) is given by

$$\tilde{P}_n(k+1)=(\Phi-K_p(k)H)\tilde{P}_n(k)(\Phi-K_p(k)H)^T + K_p\sigma_\xi^2K_p^T(k) \ .$$

When ξ is stationary, $\tilde{P}_n(k+1)$ equals $\tilde{P}_n(k)$ for sufficiently large k. Thus, in the steady state

$$\tilde{P}_n = (\Phi-K_pH)\tilde{P}_n(\Phi-K_pH)^T + \sigma_\xi^2K_pK_p^T \qquad (25)$$

where $\tilde{P}_n \triangleq \lim_{k\to\infty} \tilde{P}_n(k)$ and $K_p = \lim_{k\to\infty} K_p(k)$.

Let the elements of $\tilde{P}_n$ be given by

$$\tilde{P}_n = \begin{bmatrix} \tilde{P}_{1n} & \tilde{P}_{2n} & \tilde{P}_{3n} \\ \tilde{P}_{2n} & \tilde{P}_{4n} & \tilde{P}_{5n} \\ \tilde{P}_{3n} & \tilde{P}_{5n} & \tilde{P}_{6n} \end{bmatrix} \qquad (26)$$

By expanding Equation (25), a set of linear equations is obtained which may be written as

$$A\,\vec{P} = \vec{B} \qquad (27)$$

where A, $\vec{P}$, and $\vec{B}$ are given by

$$A \triangleq \begin{bmatrix} (1-k_1)^2-1 & 2T_f(1-k_1) & T_f^2(1-k_1) & T_f & T_f^3 & T_f^4/4 \\ -(1-k_1)k_2 & -(k_1+k_2T_f) & T_f(1-k_1)-(T_f^2k_2/2) & T_f & 3T_f^2/2 & T_f^2/2 \\ -(1-k_1)k_3 & -T_fk_3 & \rho(1-k_1)-(T_fk_3/2)-1 & 0 & \rho T_f & \rho T_f^2/2 \\ k_2^2 & -2k_2 & -2T_fk_2 & 0 & 2T_f & T_f^2 \\ k_2k_3 & -k_3 & -(\rho k_2+T_fk_3) & 0 & \rho-1 & \rho T_f \\ k_3^2 & 0 & -2k_3\rho & 0 & 0 & \rho^2-1 \end{bmatrix}$$

(28a)

where k_1, k_2, and k_3 are the 1st, 2nd, and 3rd components of the vector K_p

$$\vec{P} = \begin{bmatrix} \tilde{P}_{1n} \\ \tilde{P}_{2n} \\ \tilde{P}_{3n} \\ \tilde{P}_{4n} \\ \tilde{P}_{5n} \\ \tilde{P}_{6n} \end{bmatrix} \qquad (28b) \quad \text{and} \quad \vec{B} = \begin{bmatrix} -k_1^2\sigma_\xi^2 \\ -k_1k_2\sigma_\xi^2 \\ -k_1k_3\sigma_\xi^2 \\ -k_2^2\sigma_\xi^2 \\ -k_2k_3\sigma_\xi^2 \\ -k_3^2\sigma_\xi^2 \end{bmatrix} . \qquad (28c)$$

Once ρ, T_f, σ_u^2, and σ_v^2 (i.e., the model parameters) are specified, the steady-state gain vector and hence the matrix A are both uniquely determined. By specifying σ_ξ^2, the vector B is also determined, so that one may easily solve for $\vec{P}$ using a computer.

A computer program (named VPCALC) was written to solve for $\vec{P}$ for a wide variety of model parameters. A listing of VPCALC is given in Appendix I. The main computation in the program is accomplished in subroutine VPGAUS, which forms the matrix A and solves for P by Gaussian elimination followed by back substitution [8,9,10]. A solution by finding A^{-1} and using the equation

$$\vec{P} = A^{-1}\vec{B}$$

was not persued because Gaussian elimination followed by back substitution is considerably more economical in terms of computer time than matrix inversion.

Figure 7 (solid lines) shows values of $\sqrt{\tilde{P}_{1n}}/\sigma_\xi$, the normalized jitter, obtained from VPGAUS as a function of ρ and R*,

where $$R^* = \sigma_v\lambda^2/\sigma_T \; . \qquad (29)$$

For comparison, $\sqrt{\tilde{P}_1}/\sigma_v$ as given by Kalman theory for the process and observation models is plotted (dotted lines) versus the same parameters. The latter results were obtained by iterating Equation (5) to convergence[1]. The standard deviation of the timing error due to measurement noise alone is observed to be always less than the standard deviation of the timing error due to the modeled process. Also, as R* and ρ increase the two curves tend to converge. Large R* and ρ result from models where accelerations are highly correlated and

[1] Iterations were halted when all elements of $\tilde{P}(k)$ agreed to nine significant figures with the corresponding elements in $\tilde{P}(k-1)$.

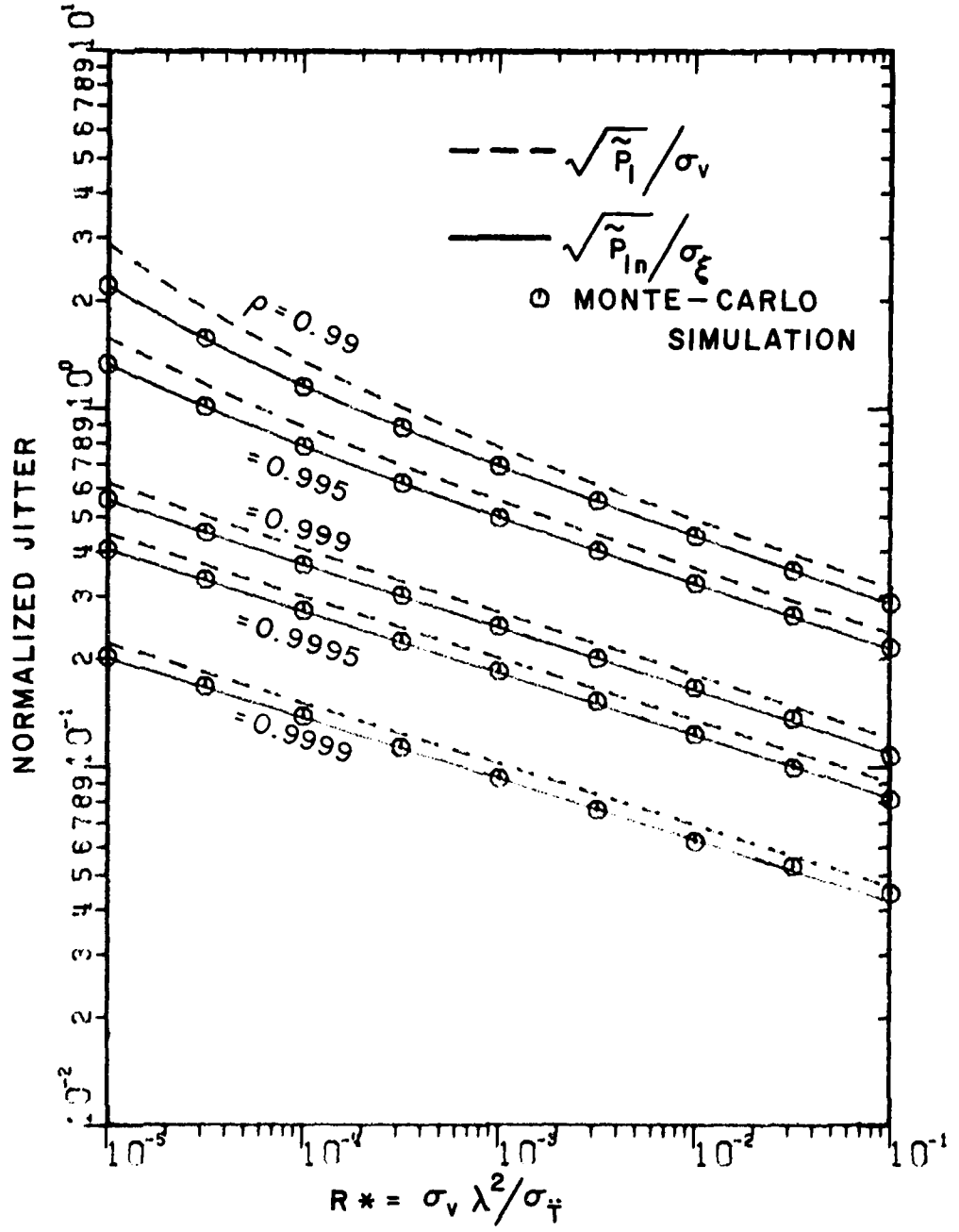


Figure 7. Jitter performance of the Kalman Predictor as a function of model parameters.

measurement noise (σ_v) dominates the maneuver noise ($\sigma_{\ddot{x}_T}$). Under these circumstances, measurement noise is the predominant source of tracking jitter. Thus, the error standard deviation due to noise alone approaches that due to the model.

It should be noted that the normalizations used in Figure 7 (i.e., R* and ρ) were checked repeatedly by running modified versions of VPCALC. Many different versions were run using different values for T_f, λ, σ_v^2, and $\sigma_{\ddot{x}_T}^2$. In each case, the resulting data agrees with that presented here for corresponding values of R* and ρ.

B. Noise Response by Computer Simulation

One method for checking the validity of the preceding derivation is Monte-Carlo simulation. Here, the algorithm given by Equation (3) is implemented directly on a computer with the observation x(k) supplied by a random number (noise) generator. The state vector z(k) is constrained to zero for all the iterations (i.e., all k). Since the error covariance and the predictor covariance are equal under this constraint, the tracking error variance ($\tilde{P}_{1n}$) may be approximated by the sample variance of the 1st component of the prediction. That is, for sufficiently large N

$$\tilde{P}_{1n} = \frac{\sum_{k=1}^{N} |\tilde{z}_{1n}(k)|^2}{N} \tag{30}$$

where N is the number of iterations. N must be chosen large enough to make the initial transient effects negligible.

The computer program VSIM, listed in Appendix II, performs the above procedure for a wide variety of model parameters. As the modeled acceleration becomes more correlated (i.e., as ρ increases) the steady-state time constant of the corresponding filter increases so N must be increased. Table 1 shows the number of iterations used in VSIM as a function of ρ. The data from the simulation is plotted with circles on Figure 7 and is virtually identical with that obtained in Section A.

C. Z-Transform Approach

While Equations (27) and (28) provide a convenient means for numerically computing the prediction error variance, they do not provide a simple analytical formula where the effects of each parameter are observable. The desire for a formula of the form

Table 1
The number of iterations per data point in the program VSIM

ρ	Iterations
.99	20000
.999	40000
.999	60000
.9995	80000
.999	100000

$$\tilde{P}_1 = f(k_1, k_2, k_3, T_f, \rho) \tag{31}$$

led to yet a third approach, Z-transform analysis. From Equation (3) and the model (Equations (17)), a signal flow graph of the Kalman predictor may be drawn. This flowgraph is shown in Figure 8. By using Mason's rule[15], the transfer function between the input and $\tilde{z}_1$ is found to be:

$$H(Z) = \frac{k_1 + A\,Z^{-1} + B\,Z^{-2}}{1 + C\,Z^{-1} + D\,Z^{-2} + E\,Z^{-3}} \tag{32}$$

where

$$A = -k_1(1+\rho) + k_2 T_f + k_3 T_f^2/2 \tag{33a}$$

$$B = k_1\rho - k_2 T_f \rho + k_3 T_f^2/2 \tag{33b}$$

$$C = k_1 - \rho - 2 \tag{33c}$$

$$D = A + 2\rho + 1 \tag{33d}$$

$$E = B - \rho \tag{33e}$$

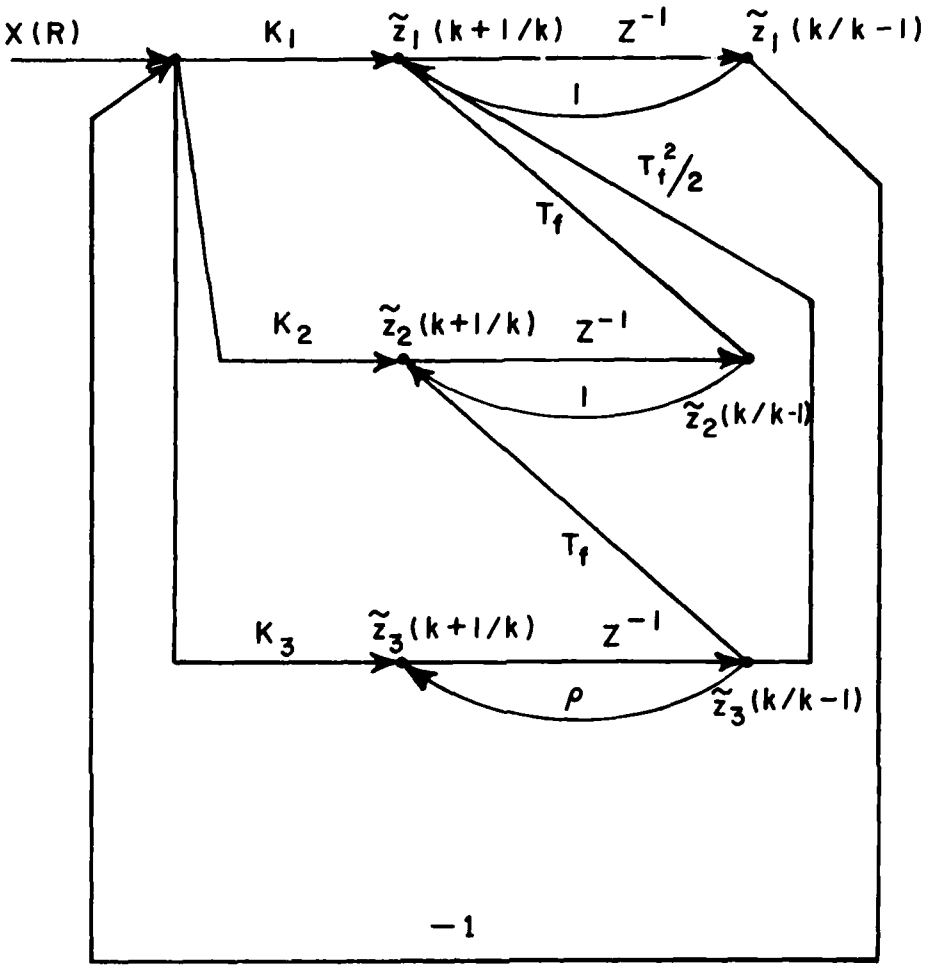


Figure 8. Signal flow graph of the Kalman Predictor.

From Z-transform theory[11], if x(k) has an autocorrelation sequence $\Phi_{xx}(k)$ with transform $\Phi_{xx}(z)$, the transformed autocorrelation sequence of the output (z_1) is given by

$$\Phi_{\tilde{z}_1\tilde{z}_1}(Z) = H(Z)H(Z^{-1})\Phi_{xx}(Z) \tag{34}$$

For the case at hand, we have an input of white noise, so that

$$\Phi_{xx}(Z) = \sigma_\xi^2 \tag{35}$$

and

$$\Phi_{\tilde{z}_1\tilde{z}_1}(Z) = H(Z)H(Z^{-1})\sigma_\xi^2 \tag{36}$$

Thus

$$\frac{\Phi_{\tilde{z}\tilde{z}}(Z)}{\sigma_\xi^2} = \frac{k_1^2+A^2+B^2+(Ak_1+AB)(Z+Z^{-1})+k_1B(Z^2+Z^{-2})}{1+C^2+D^2+E^2+(C+CD+DE)(Z+Z^{-1})+(D+CE)(Z^2+Z^{-2})+E(Z^3+Z^{-3})} \tag{37}$$

Note that $\Phi_{\tilde{z}\tilde{z}}(Z)$ is the Z-transform of a symmetric autocorrelation function, so it is a two-sided transform. We would like to employ the initial value theorem[11] to determine $\phi_{zz}(k)\big|_{k=0}$, which is the variance of the output. However, this theorem only applies to a one sided transform F(Z) for which f(k) is zero for negative k. In order to apply this theorem, we will split $\Phi_{\tilde{z}\tilde{z}}(Z)$ into two parts, one corresponding to non-negative k and the other to non-positive k. That is,

$$\frac{\Phi_{\tilde{z}\tilde{z}}}{\sigma_\xi^2}(Z) = G^+(Z) + G^-(Z) \tag{38}$$

where $G^+(Z)$ is the transform of $\frac{\phi_{\tilde{z}\tilde{z}}}{\sigma_\xi^2}(k)$ for $k \geq 0$ $\left(\phi^+_{\tilde{z}\tilde{z}}(k)\right)$

and $G^-(Z)$ is the transform of $\frac{\phi_{\tilde{z}\tilde{z}}}{\sigma_\xi^2}(k)$ for $k \leq 0$ $\left(\phi^-_{\tilde{z}\tilde{z}}(k)\right)$.

Note that at k=0, $G^+(Z)$ and $G^-(Z)$ each contain one half the total contribution of $\Phi_{\tilde{z}\tilde{z}}$.

From the properties of Z-transform[11], if f(k) has the Z-transform F(z), then f(-k) has the Z transform $F(Z^{-1})$. Since $\phi_{\tilde{z}\tilde{z}}(k)$ is an even function,

$$\Phi_{\tilde{z}\tilde{z}}(-k) = \Phi_{\tilde{z}\tilde{z}}(k) \ . \tag{39}$$

and

$$G^{-}(Z) = G^{+}(Z^{-1}) \tag{40}$$

so

$$\frac{\Phi_{\tilde{z}\tilde{z}}}{\sigma_{\xi}^{2}} = \underline{G^{+}(Z)} + \underline{G^{+}(Z^{-1})} \tag{41}$$

Let $G^{+}(Z)$ be represented by

$$G^{+}(Z) = \frac{\text{Num}(Z)}{\text{Den}(Z)} \tag{42}$$

where Num(Z) = the numerator of $G^{+}(Z)$

Den(Z) = the denominator of $G^{+}(Z)$

Now, $G^{+}(Z)$ is the Z-transform of a finite time function, so its poles will be those poles of $H(Z)\ H(Z^{-1})$ which lie inside the unit circle in the Z plane. These are the poles of H(Z). Thus, the denominator of $G^{+}(Z)$ is the same as the denominator of H(Z), or

$$\text{Den}(Z) = 1+CZ^{-1} + DZ^{-2} + EZ^{-3} \tag{43}$$

The numerator of $G^{+}(Z)$ will be a polynomial of the form

$$\text{Num}(Z) = V+WZ^{-1} + XZ^{-2} + YZ^{-3} \tag{44}$$

with V, W, X, Y to be determined. Thus,

$$G^{+}(Z) = \frac{V+WZ^{-1}+XZ^{-2}+YZ^{-3}}{1+CZ^{-1}+DZ^{-2}+EZ^{-3}} \tag{45}$$

and

$$\frac{\Phi_{\tilde{z}\tilde{z}}(Z)}{\sigma_{\xi}^{2}} = \frac{V+WZ^{-1}+XZ^{-2}+YZ^{-3}}{1+CZ^{-1}+DZ^{-2}+EZ^{-3}} + \frac{V+WZ+XZ^{2}+YZ^{3}}{1+CZ+DZ^{2}+EZ^{3}} \tag{46}$$

Combining terms yields

$$\frac{\Phi_{\underset{\sim}{z}\underset{\sim}{z}}(Z)}{\sigma_\xi^2} = \{2\cdot(V+CW+DX+EY)+(CV+DW+EX+W+CW+DY)(Z+Z^{-1}) \quad (47)$$

$$+ (DV+EW+X+CY)(Z^2+Z^{-2})+(EV+Y)(Z^3+Z^{-3})\}$$

$$\div \{(1+CZ^{-1}+DZ^{-2}+EZ^{-3})\ (1+CZ+DZ^2+EZ^3)\}$$

By equating coefficients of Z in the numerators of Equations (37) and (47), one obtains

$$EV+Y = 0 \quad (48a)$$

$$DV+EW+X+CY = k_1 B \quad (48b)$$

$$CV+DW+EX+W+CS+DY = Ak_1+AB \quad (48c)$$

$$2(V+CW+DX+EY) = k_1 + A^2 + B^2 \quad (48d)$$

Note that one could include coefficients for both lower and higher powers of Z in the expression for Num(Z) (Equation (44)). However, after cross multiplying and equating coefficients, one would find that the coefficients of these other powers of Z are zero. Application of the initial value theorem to the expresison for G+(Z) (37) yields:

$$\left.\frac{\phi^+_{\underset{\sim}{z}\underset{\sim}{z}}(k)}{\sigma_\xi^2}\right|_{k=0} = V$$

Thus, we will solve the set of Equations (48) for V. From (48a),

$$Y = -EV \quad (49)$$

and by substitution for Y the remaining equations can be written

$$(D-CE)V + EW + X = k_1 B \quad (50a)$$

$$(C-DE)V + (D+1)W+(C+E)X = Ak_1+AB \quad (50b)$$

$$2(1-E^2)V + 2CW+2DX = k_1^2 + A^2 + B^2 \quad (50c)$$

These may be solved via Cramer's Rule to yield

$$V = \frac{\begin{vmatrix} k_1B & E & 1 \\ Ak_1+AB & D+1 & C+E \\ k_1^2+A^2+B^2 & 2C & 2D \end{vmatrix}}{\begin{vmatrix} D-CE & E & 1 \\ C-DE & D+1 & C+E \\ 2(1-E^2) & 2C & 2D \end{vmatrix}} \tag{51}$$

where $|\cdot|$ indicates determinant. Thus,

$$\begin{aligned} V = \{&(k_1B)\,(D+1)(2D) + E(C+E)(k_1^2+A^2+B^2) + 2C(Ak_1+AB) \\ &-(D+1)(k_1^2+A^2+B^2)-2E(Ak_1+AB)D-2k_1B\,(C+E)C\} \div \\ &\{(D-CE)(D+1)(2D) + E(C+E)2(1-E^2) + 2(C-DE)C \\ &-(D+1)2(1-E^2)-2E(C-DE)D-2(D-CE)(C+E)C\} \end{aligned} \tag{52}$$

Finally, $\dfrac{\phi_{\tilde{z}\tilde{z}}(0)}{\sigma_\xi^2}$ is given by

$$\frac{\phi_{\tilde{z}\tilde{z}}(0)}{\sigma_\xi^2} = \frac{\phi^{+}_{\tilde{z}\tilde{z}}(0)}{\sigma_\xi^2} + \frac{\phi^{-}_{\tilde{z}\tilde{z}}(0)}{\sigma_\xi^2} = \frac{2\phi^{+}_{\tilde{z}\tilde{z}}(0)}{\sigma_\xi^2} = 2V \tag{53}$$

To check the validity of Equations (51) and (52) a computer program named ZSIM was written to evaluate them directly. A listing of ZSIM appears in Appendix III. For the cases of ρ=0.99 and ρ=0.995 the values for $\sqrt{\tilde{P}_{1n}}/\sigma_\xi$ obtained from running ZSIM agree with those presented in Figure 7 to five significant figures. For the larger values of ρ (0.999, 0.9995, and 0.9999) numerical instabilities (i.e., over and/or under flows) were observed for the larger values of R*. The cause of the instabilities was traced to the determinants in Equation (51). For large R* and large ρ, both of the determinants in Equation (51) approach zero. In evaluating the determinants, (via Equation (52)) subtraction of nearly equal terms to yield nearly zero determinants result in a loss of significant bits in the computation. If the machine precision (i.e., word length) is not sufficient, all of the significant bits may be lost. The initial runs of ZSIM were made in double precision on a Datacraft 6024 computer with 24 bits per word, resulting in approximately 13 decimal digits of precision. Many different versions of ZSIM were run, each using different combinations of model parameters (σ_v, $\sigma_{\dot{T}}$, T_f, λ) to generate R* and ρ. None of these were any more or less successful than the version given in Appendix III.

As a final check on the cause of the numerical instabilities of ZSIM, a modified version was run on an Amdahl 470 computer (32 bit words) using extended (quadruple) precision. This yields 128 bits per variable or about 34 decimal digits of precision. By computing the gains to 30 significant figures, it was possible to compute $\sqrt{\tilde{P}_{1n}}/\sigma_\xi$ to agree with the data in Figure 7 to at least 6 digits. Thus, Equation (52) appears to be correct but must be regarded as numerically unstable.

CHAPTER IV
NOISELESS RESPONSE OF THE KALMAN PREDICTOR TO AIRCRAFT MANEUVERS

Recall that we are using superposition to find the Kalman clock-loop predictor's response to an input composed of noise plus a deterministic signal. Chapter III documents the Kalman predictor's performance with white noise inputs. In this chapter, we solve the 2nd part of the problem - finding the response to deterministic inputs.

To accomplish this, it was decided to simulate (by computer) actual maneuvers rather than step changes in delay, velocity, or acceleration. Although step changes are much easier to simulate on the computer, with the possible exception of steps in acceleration they are not physically realistic of aircraft maneuvers. The maneuvers which were simulated are shown in Figure 9a and Figure 10a. Both are 180^0 turns. In Figure 9a, the initial and final flight paths are along the axis to the satellite. This is called the axial 180^0 turn. In Figure 10a, the initial and final flight paths are perpendicular to the axis to the satellite. This is called the orthogonal 180^0 turn. In both cases, the radius and angular velocity are given respectively by (see Reference 14, Section 3.4)

$$R = V^2/A \tag{54}$$

and

$$\dot{\theta} = A/V \tag{55}$$

where A = magnitude of the constant, radially inward acceleration

and V = magnitude of the initial velocity.

Of course, since we are simulating delay functions, A is given in units of chips/sec^2 and V is given in units of chips/sec. Letting t=0 at θ=0, θ is then given by

$$\theta = \dot{\theta}t \tag{56}$$

For the axial turn, the state vector z (i.e., $(T_o, \dot{T}_o, \ddot{T}_o)^T$) is given by (assuming that the satellite is at infinity)

$$z_1(t) = T_o(t) = \begin{cases} T_o(0) - Vt/c; & t \leq 0 \\ T_o(0) - \frac{R}{c} \sin \frac{A}{V}t; & 0 \leq t \leq \frac{\pi V}{A} \\ T_o(0) + Vt/c; & \frac{\pi V}{A} \leq t \end{cases} \tag{57a}$$

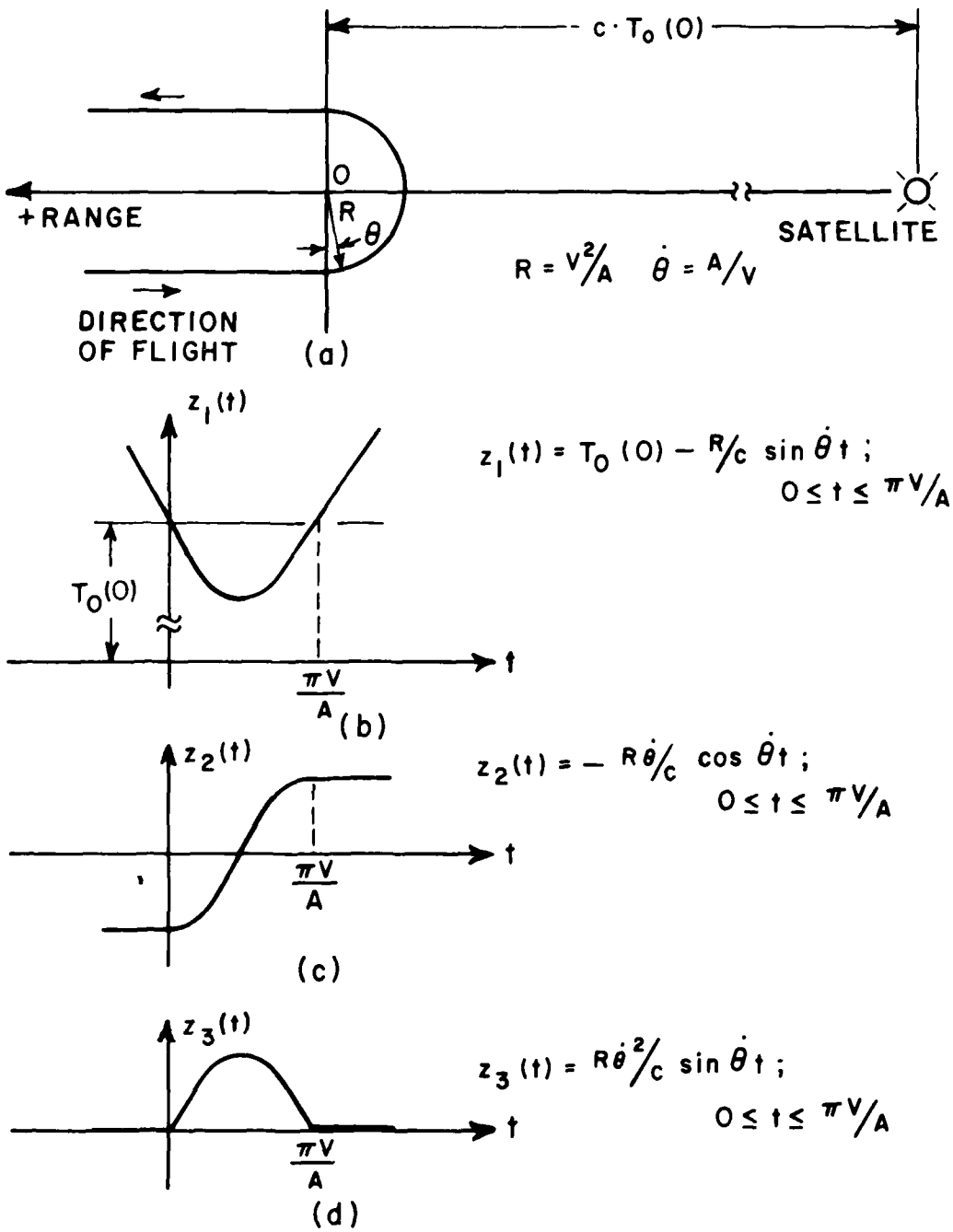


Figure 9. The axial 180° turn; (a)flight path, (b,c and d) components of the state vector.

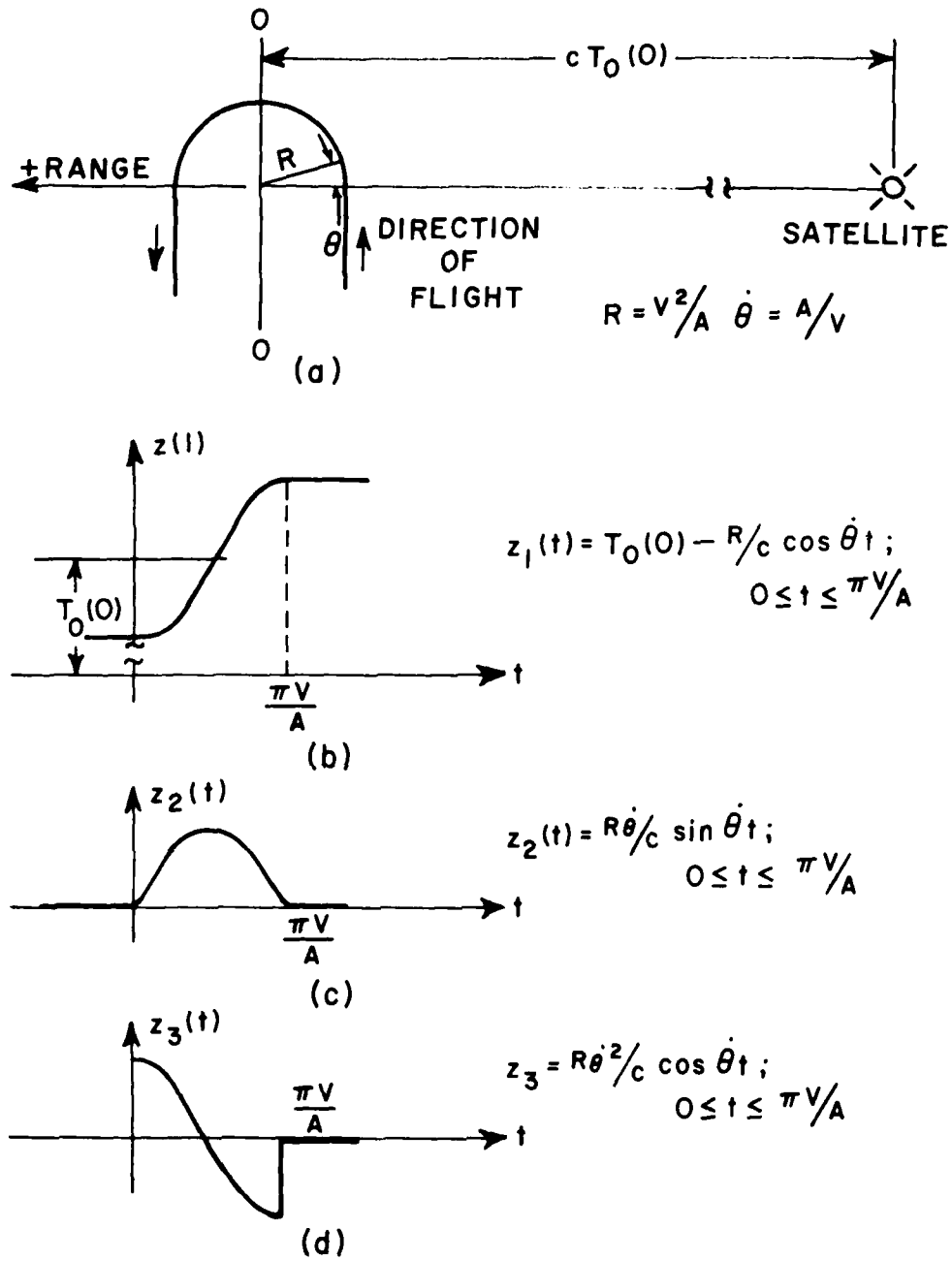


Figure 10. The orthogonal 180° turn; (a)flight path, (b,c and d) components of the state vector.

$$z_2(t) = \dot{T}_o(t) = \begin{cases} -V/c & ; \quad t \le 0 \\ -\frac{R\dot{\theta}}{c} \cos \frac{A}{V} t & ; \quad 0 \le t \le \frac{\pi V}{A} \\ +\frac{V}{c} & ; \quad \frac{\pi V}{A} \le t \end{cases} \tag{57b}$$

$$z_3(t) = \ddot{T}_o(t) = \begin{cases} 0 & ; \quad t \le 0 \\ \frac{R\dot{\theta}^2}{c} \sin \dot{\theta} t & ; \quad 0 \le t \le \pi V/A \\ 0 & ; \quad \frac{\pi V}{A} \le t \end{cases} \tag{57a}$$

where T_o is the delay from the satellite to the coordinate origin of the maneuver. These are sketched in Figures 9b, 9c, and 9d, respectively.

Similarly, the state vector z for the orthogonal turn is given by

$$z_1(t) = T_o(t) = \begin{cases} T_o(0) - \frac{R}{c} & ; \quad t \le 0 \\ T_o(0) - \frac{R}{c} \cos \frac{A}{v} t & ; \quad 0 \le t \le \frac{\pi V}{A} \\ T_o(0) + \frac{R}{c} & ; \quad \frac{\pi V}{A} \le t \end{cases} \tag{58a}$$

$$z_2(t) = \dot{T}_o(t) = \begin{cases} 0 & ; \quad t \le 0 \\ \frac{R\dot{\theta}}{c} \sin \frac{A}{v} t & ; \quad 0 \le t \le \frac{\pi V}{A} \\ 0 & ; \quad \frac{\pi V}{A} \le t \end{cases} \tag{58b}$$

$$z_3(t) = \ddot{T}_o(t) = \begin{cases} 0 & ; \quad t \leq 0 \\ \frac{R\dot{\theta}^2}{c} \cos \frac{A}{v} t; & 0 \leq t \leq \frac{V}{A} \\ 0 & ; \quad \frac{\pi V}{A} < \theta \end{cases} \qquad (58c)$$

These are sketched in Figures 10b, 10c, and 10d, respectively.

In order to study the behavior of the Kalman predictor when subject to the delay inputs given by Equations (57a) or (58a), a computer program named FLYBY was written. After setting the model parameters T_f, ρ and R^* (to uniquely determine the filter) and the aircraft's initial velocity and radial acceleration (to uniquely determine the turn), the program iterated the Kalman algorithm (Equation (3)) with a noiseless delay input using steady-state Kalman (i.e., Wiener) gains. The absolute value of the tracking error was observed at each iteration, and after completion of the maneuver, the maximum absolute error was written to an output array. The program was designed such that incremental changes in model parameters and aircraft parameters (initial velocity and radial acceleration) could be made in suitable do-loops. Thus, the response to a wide range of inputs for a wide variety of model parameters was obtained.

The structure of FLYBY will now be discussed in more detail. A listing of this program appears in Appendix IV. In order to generate the maneuvers, two subroutines were written. Subroutines TURN1 (axial turn) or TURN2 (orthogonal turn) generate the state vector z by implementing Equations (57a,b, and c) or Equations (58a,b, and c) respectively. Prior to beginning the maneuver, the filter is assumed to be tracking the delay function perfectly. Subroutines INIT1 and INIT2 initialize the predictor vector z (ZWIG) to the proper values corresponding to θ=0 for the axial and orthogonal turns, respectively. The absolute tracking error at each step (ERR) along with the maximum absolute tracking error thorugh the maneuver (ERMAX) is calculated in subroutine ERROR1. The prediction is performed in subroutine PRED using steady state gains. In addition to generating the state vector, the TURN subroutines generate a variable (ICT) which indicates the number of iterations elapsed since the end of the maneuver. An IF statement terminates the simulation when ICT is greater than 1000. After the simulation ceases, the maximum absolute tracking error for the maneuver (normalized to the radius of the turn) is written to the array XOUT. A flow chart of this sequence is given in Figure 11. One additional subroutine, GAINS, is used to load the gain vector (GP) from an array of gain vectors (XGAIN) corresponding to various model parameters.

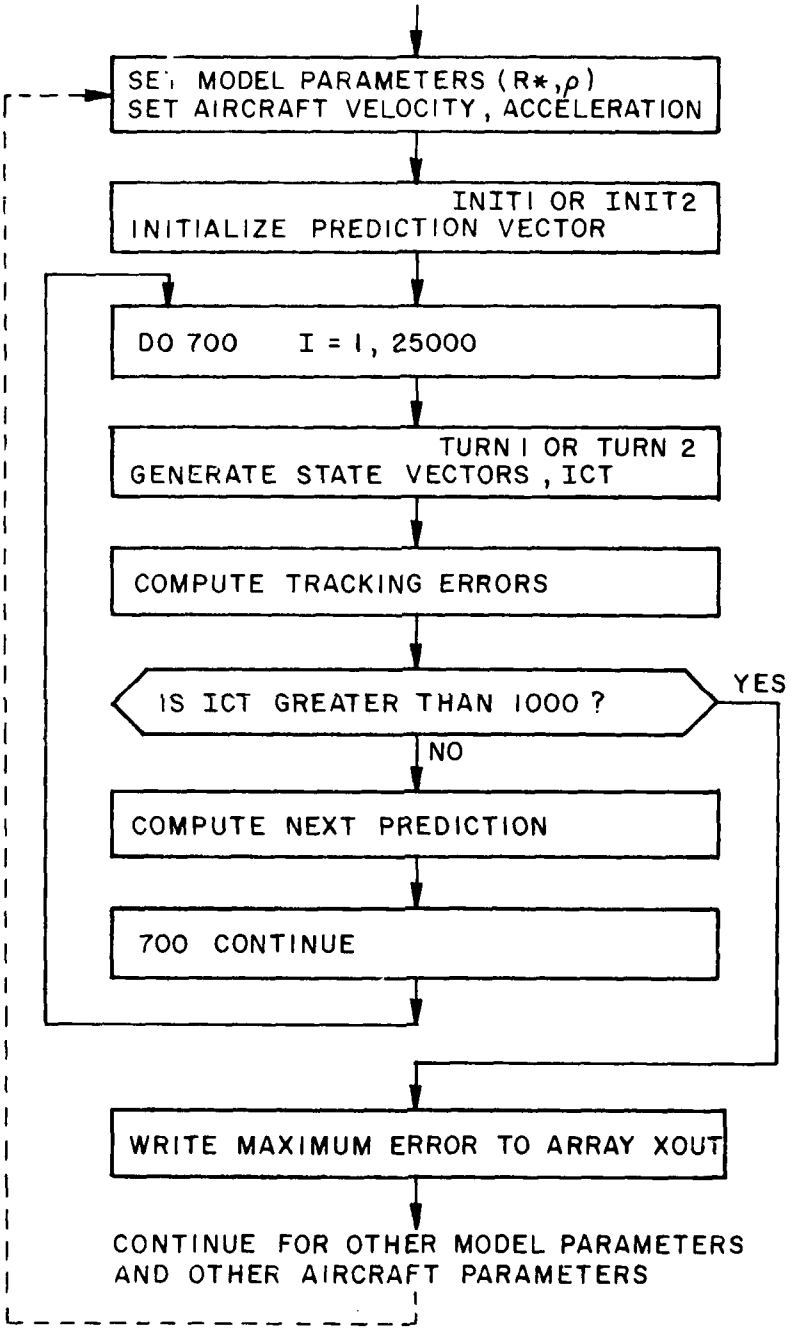


Figure 11. Flow chart for the simulation portion of the computer program FLYBY.

The data obtained from these programs is plotted in Figures 12a, b,c,d, and e for the axial turn and Figures 13a,b,c,d, and e for the orthogonal turn. Each of these figures shows maximum tracking error (normalized to the turn radius) as a function of $R^* (\sigma_v \lambda^2/\sigma_\tau)$ for a specific value of ρ. Plotted are families of curves for various angular velocities. Consider, for example, an aircraft flying at 500 mi/hr making a 5 g. (1 g. = 9.8 m/sec^2) 180^0 turn: What is the tracking error for a predictor with model parameters T_f = .01 sec, $R^* = 10^{-3}$, and ρ = .9997 At 40 mchips/sec, a 1 meter change in range generates a delay of .133Δ, so in terms of delay 500 mi./hr. becomes $V/c\Delta$ = 29.8 chips/sec and 5g becomes $A/c\Delta$ = 6.53 chps/sec^2. Thus, the radius of the turn is given by $R/c\Delta$ = 136 chips and the angular velocity is $\dot{\theta}$ = .00219 rad/sec. Turning to Figure 12c, corresponding to an axial turn for $\dot{\theta}T_f$ = .0025 and $R^* = 10^{-3}$, the normalized error is 4.5×10^{-4}. Multiplying by the radius of the turn (i.e., unnormalizing) yields a maximum tracking error of 0.06Δ. Similarly for the orthogonal turn, the maximum tracking error is found to be 0.31Δ.

In general, the tracking error for the orthogonal turn is higher that the error for the axial turn. This is due to the step discontinuity in acceleration for the orthogonal turn (Figure 10d). The axial turn has no step discontinuities. Also, the tracking error for both turns monotonically increases as R* increases. This behavior is opposite to the noise behavior of the filter presented in Figure 7 where both $\sqrt{\tilde{P}_{1_1}}/\sigma_v$ and $\sqrt{\tilde{P}_{1_0}}/\sigma_\xi$ monotonically decrease as R* increases. As R* becomes larger measurement noise tends to become the predominant factor. The filter algorithm compensates for this by narrowing the filter bandwidth. This results in improved noise performance but degraded tracking of maneuvers because the narrow bandwidth causes the filter response to lag behind the input.

The curves plotted in Figures 13c, d, and e exhibit behavior which may appear strange at first glance. The behavior was investigated and found to be caused by the response time of the predictor. The responses of the predictor to the orthogonal turn is composed of an initial lag followed by an overshoot and less significant ringing. For small R*, the response is relatively fast so that the maximum error occurs in initial lag. As R* becomes larger the filter response slows and the error in the overshoot becomes progressively more significant. The "bumps" in the curves of Figures 13c, d, and e results from the occurrance of the maximum errors in the first overshoot. As R* becomes even larger, the response of the predictor becomes so slow that the maneuver is completed before the initial lag of the response is completed. When this occurs, the errors in the lag response become progressively more significant. Eventually these errors again become larger than those in the overshoot. The regions of the curves above the "bumps" corresponds to the maximum errors occuring in the initial lag response.

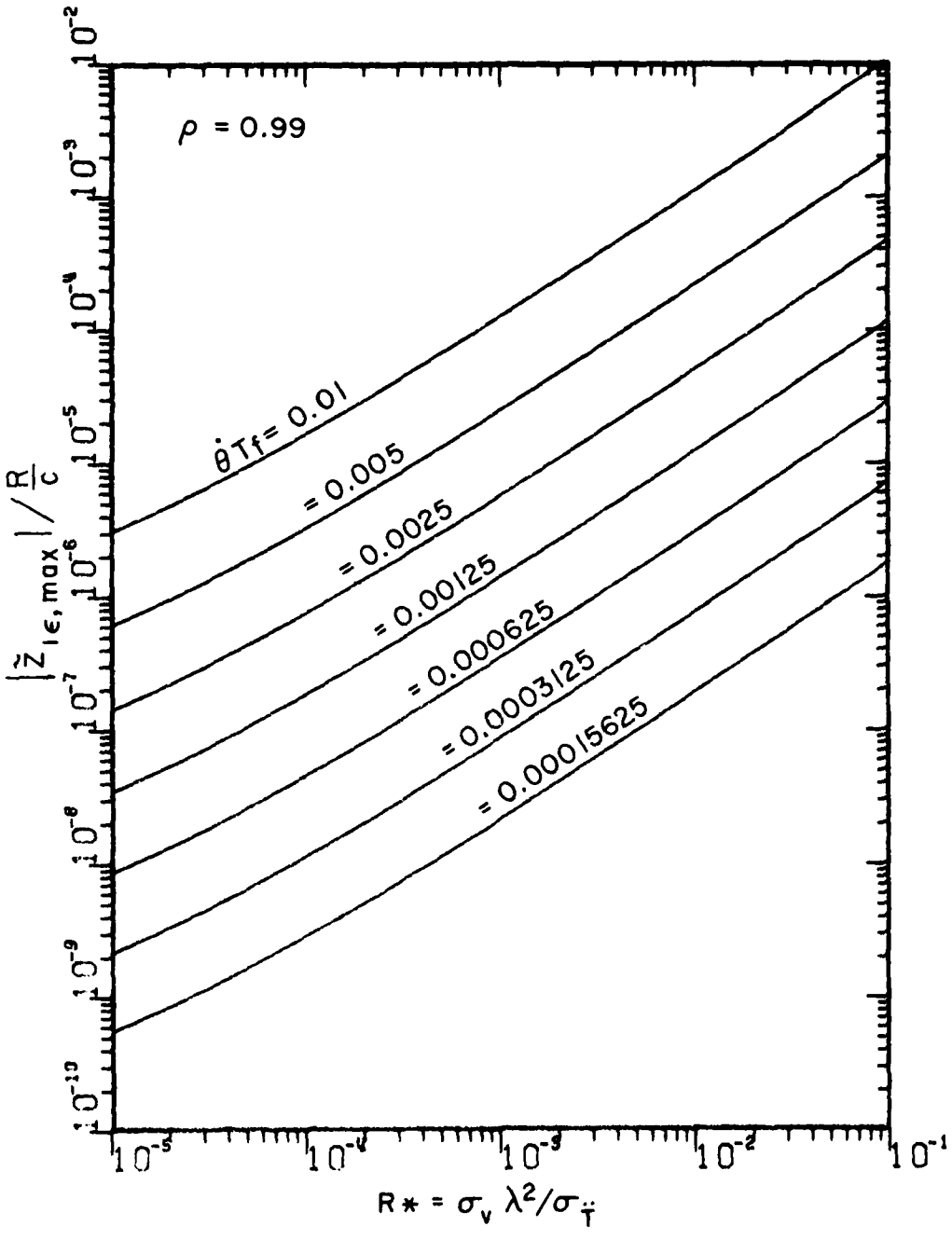


Figure 12a. The maximum delay tracking error for the axial turn as a function of model and maneuver parameters, ρ=0.99.

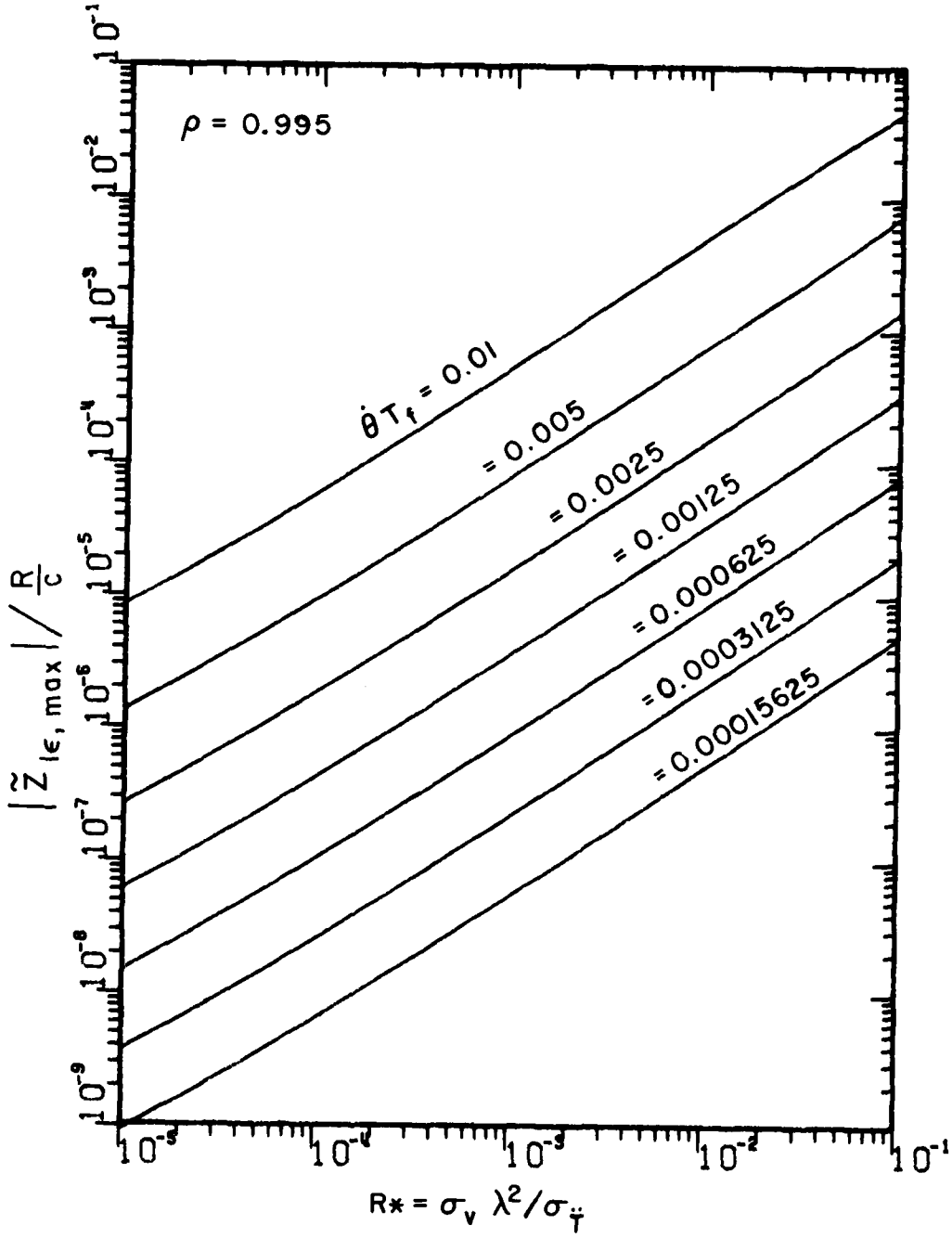


Figure 12b. The maximum delay tracking error for the axial turn as a function of model and maneuver parameters, ρ=0.995

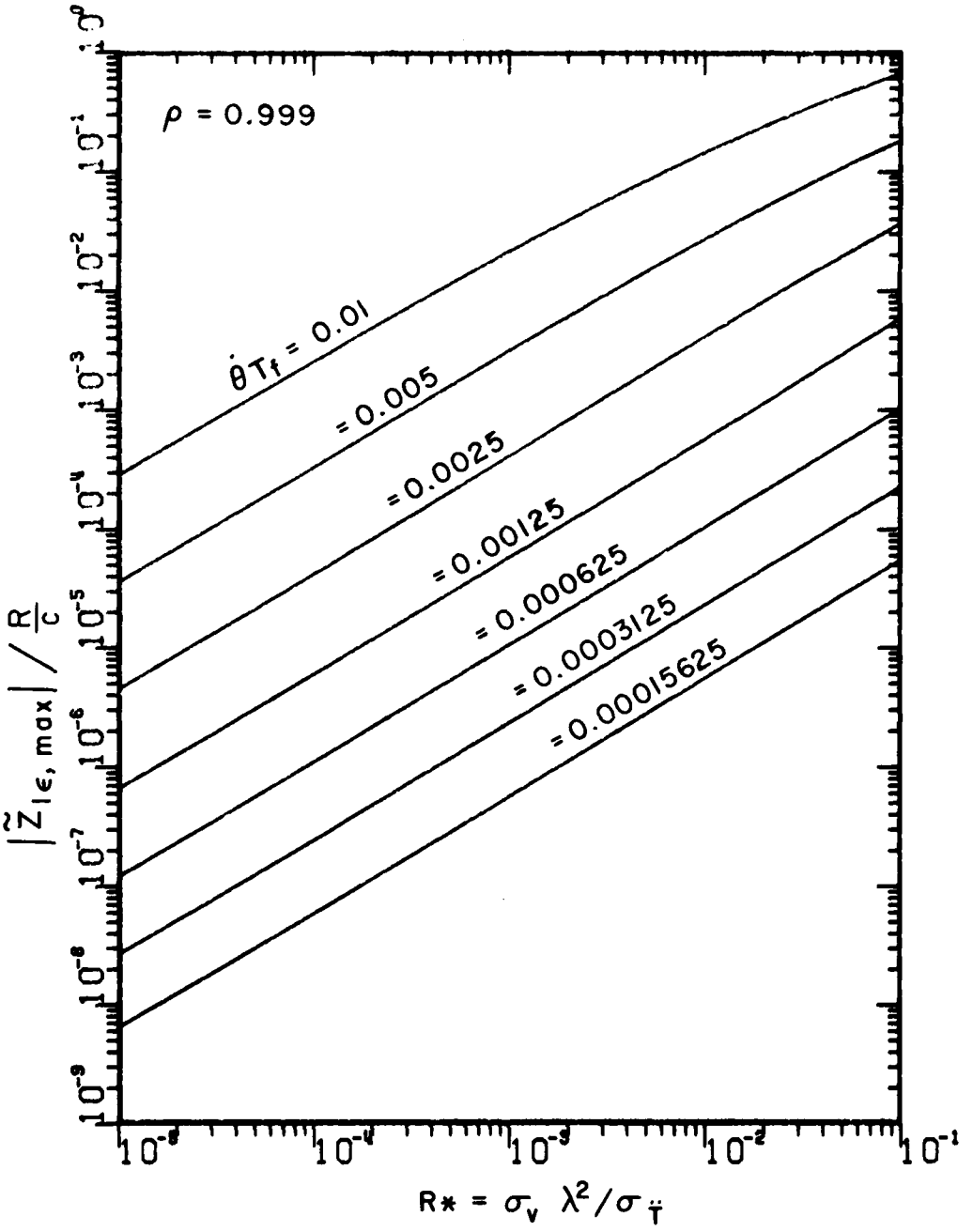


Figure 12c. The maximum delay tracking error for the axial turn as a function of model and maneuver parameters, ρ =0.999

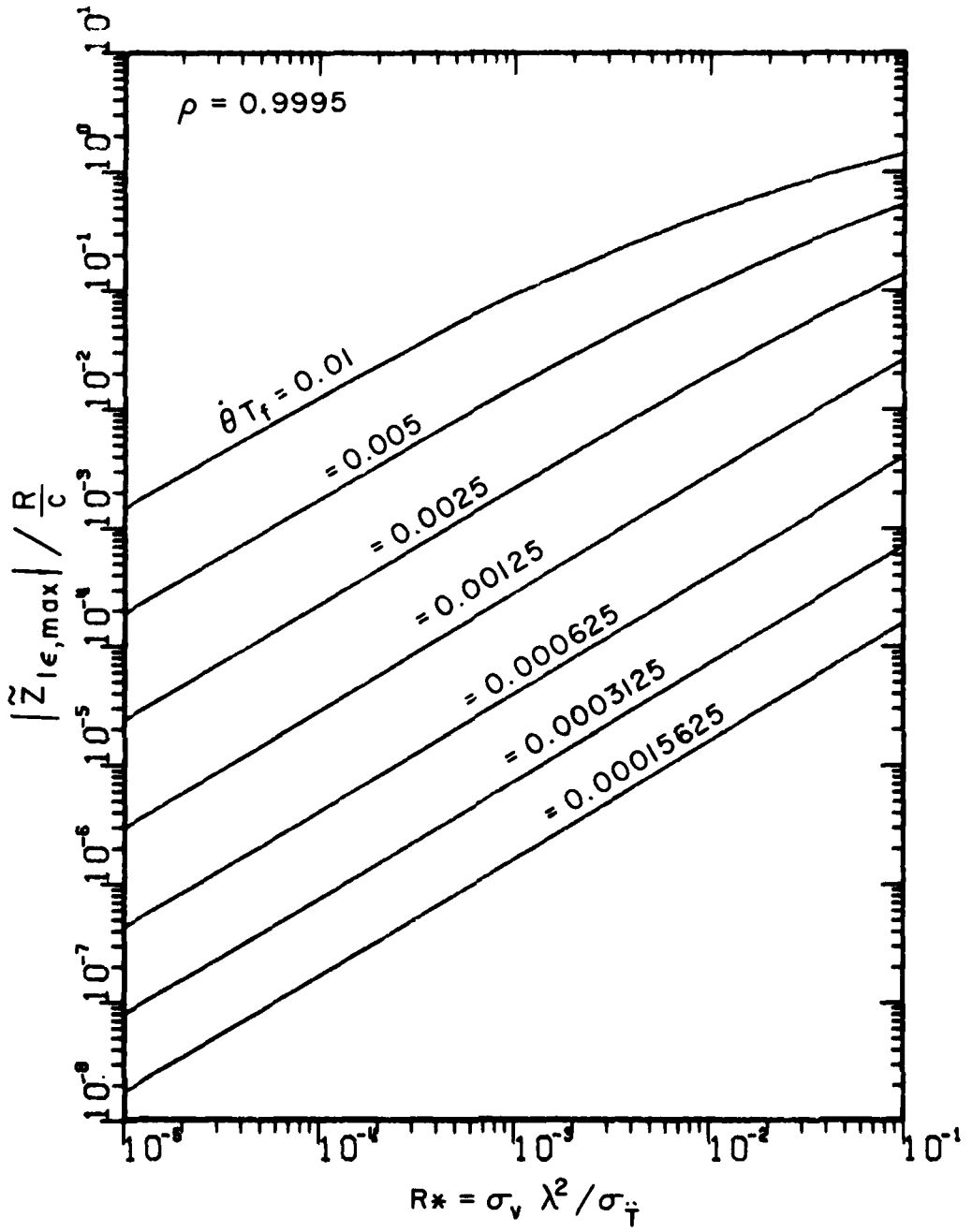


Figure 12d. The maximum delay tracking error for the axial turn as a function of model and maneuver parameters, ρ=0.9995

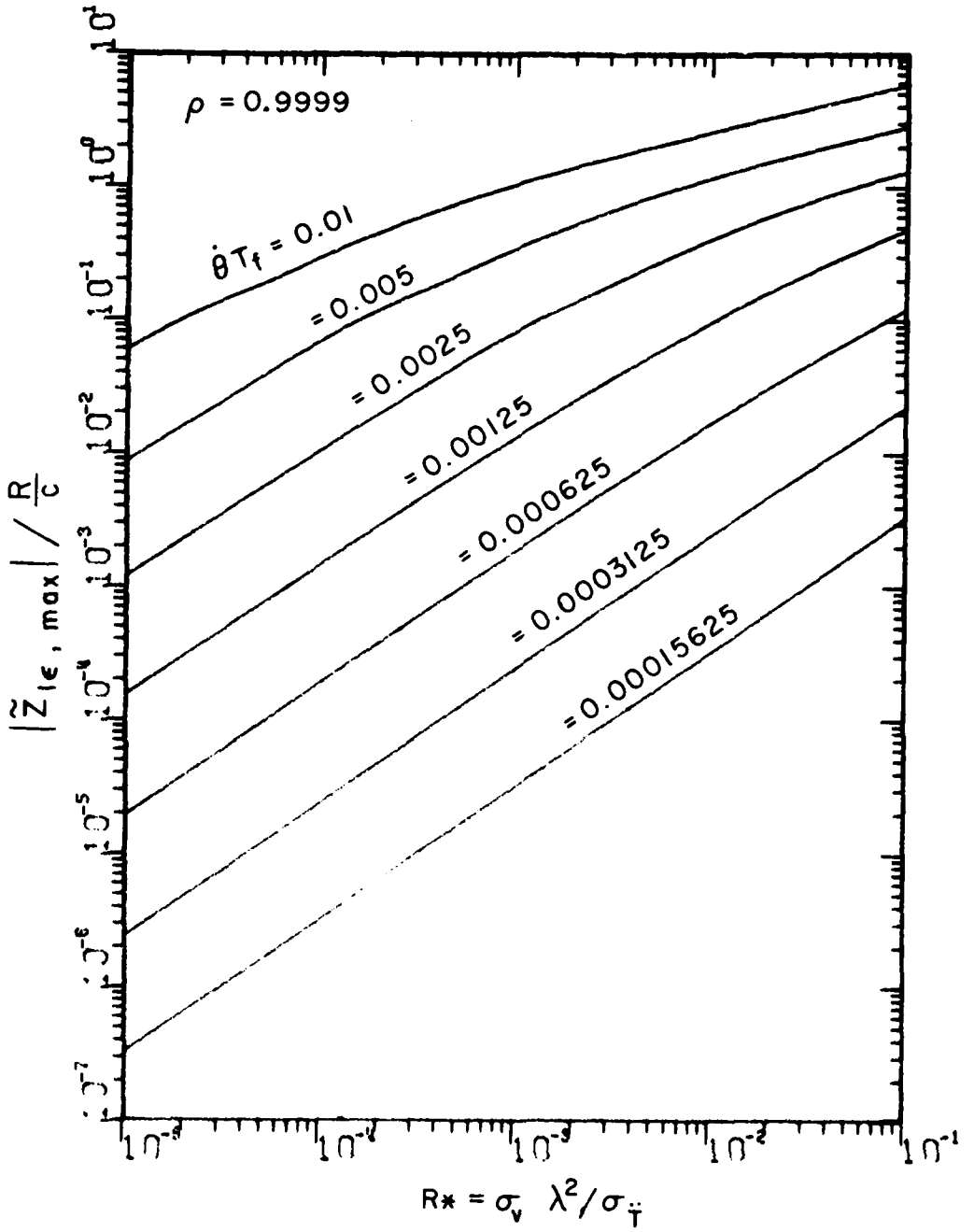


Figure 12e. The maximum delay tracking error for the axial turn as function of model and maneuver parameters, ρ=0.9999

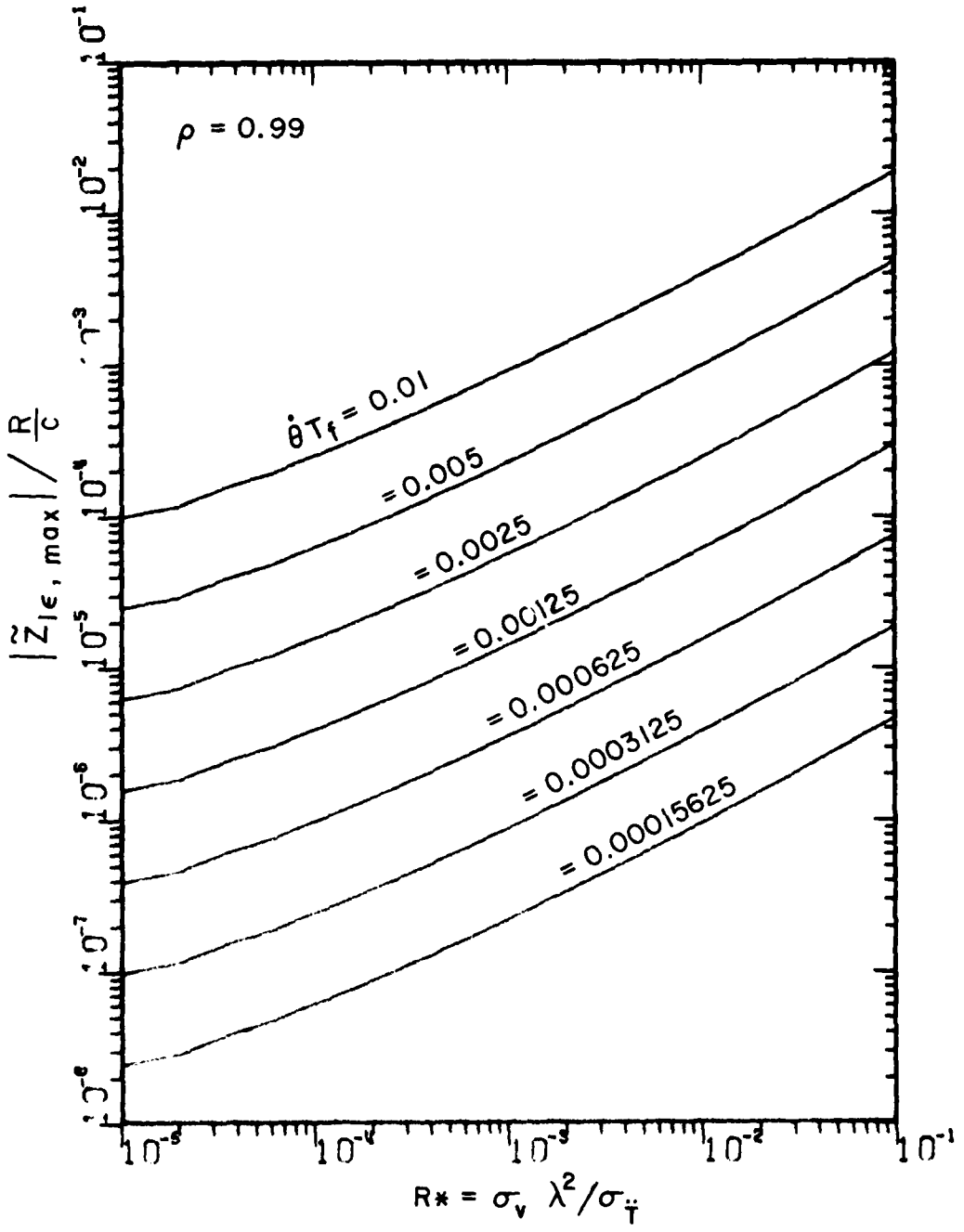


Figure 13a. The maximum delay tracking error for the orthogonal turn as a function of model and maneuver parameters, $\rho = 0.99$

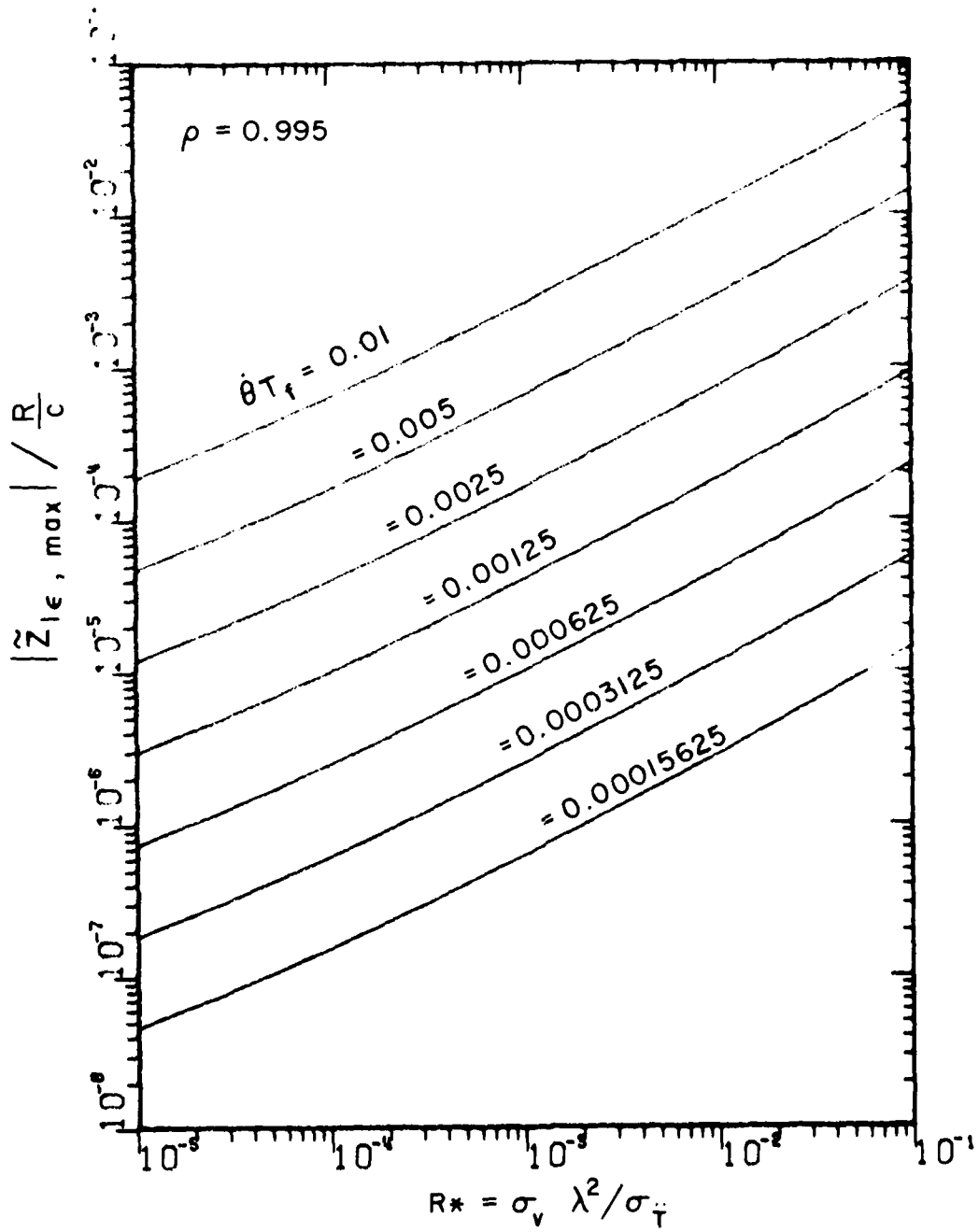


Figure 13b. The maximum delay tracking error for the orthogonal turn as a function of model and maneuver parameters, ρ=0.995

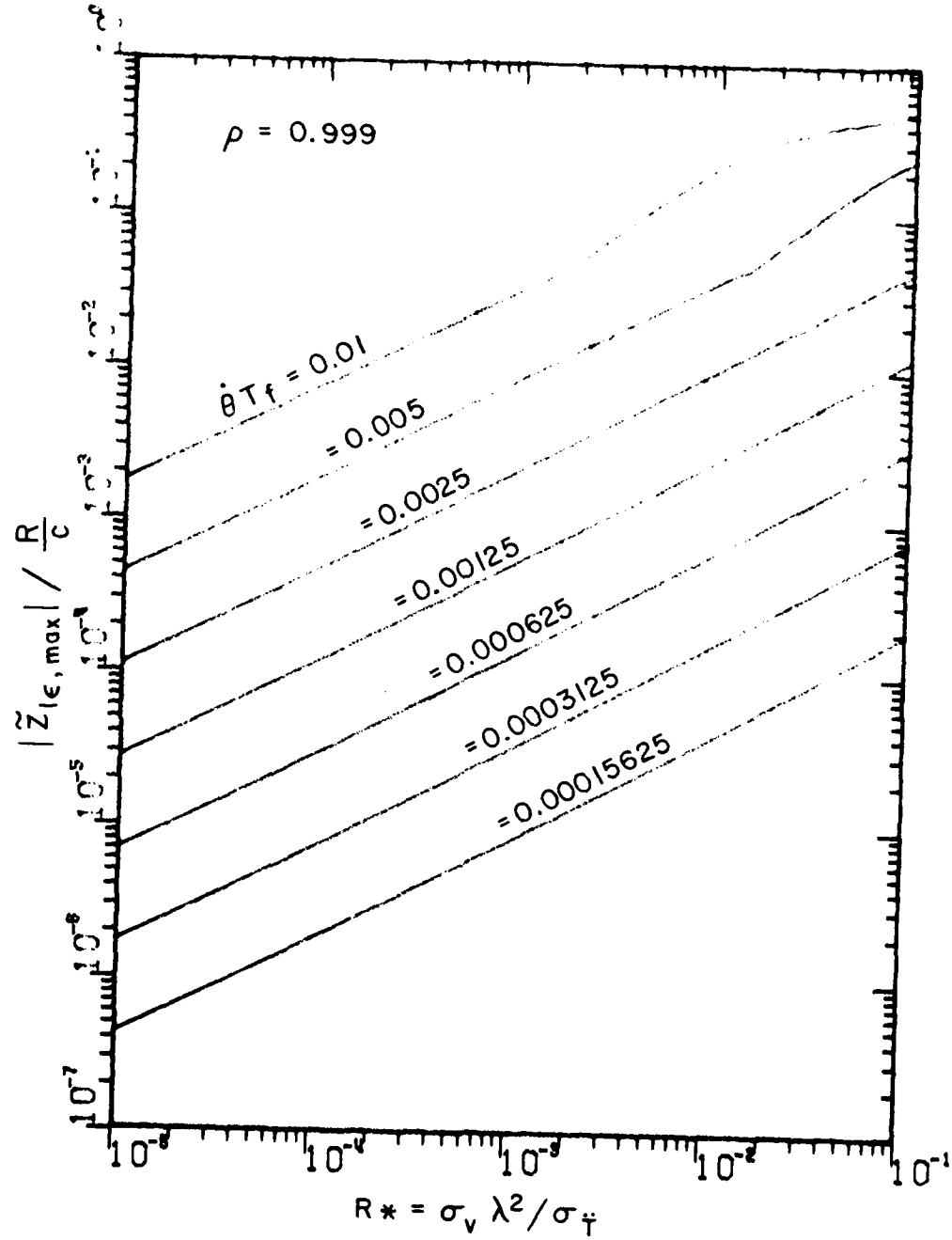


Figure 13c. The maximum delay tracking error for the orthogonal turn as a function of model and maneuver parameters, ρ=0.999

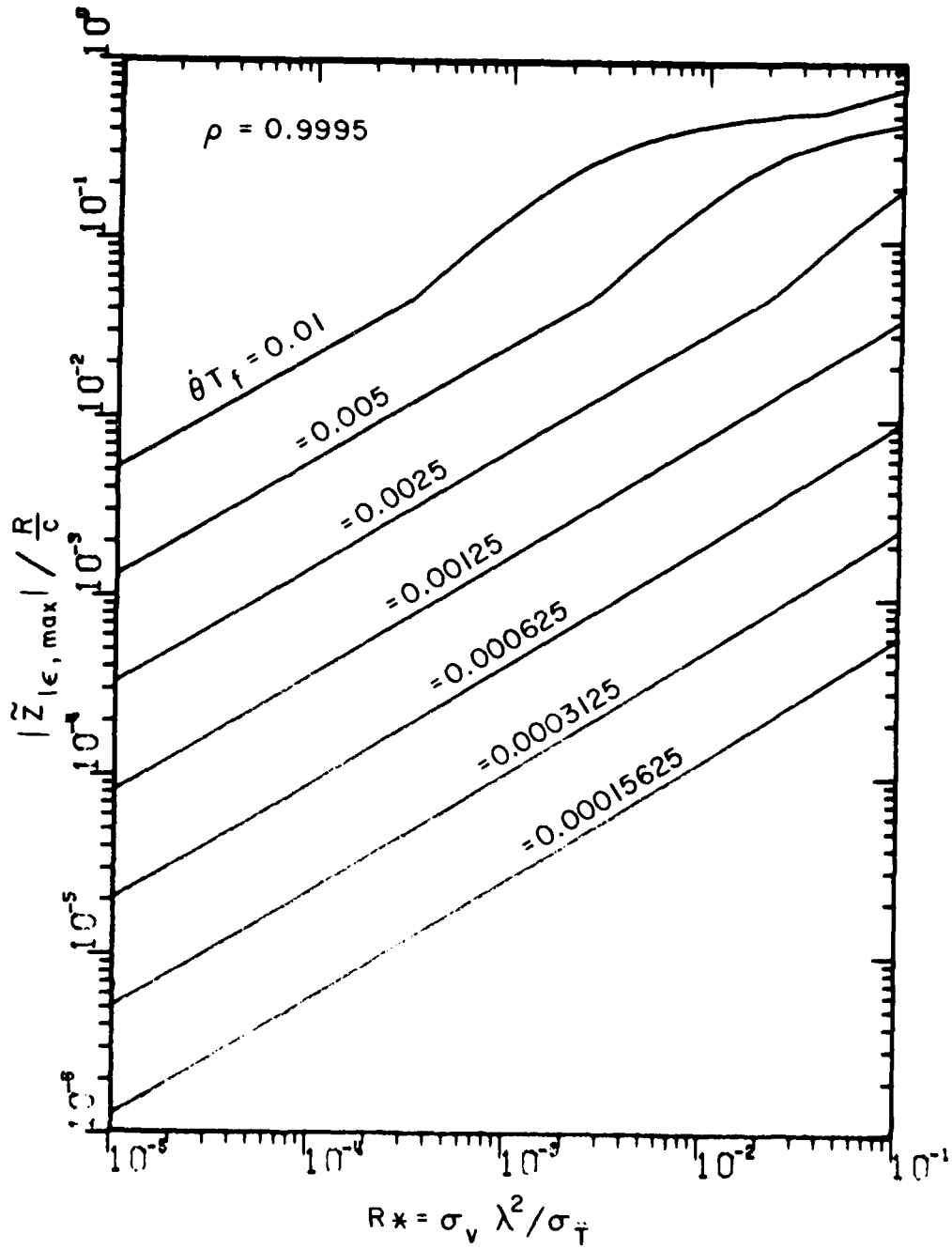


Figure 13d. The maximum delay tracking error for the orthogonal turn as a function of model and maneuver parameter, ρ=0.9995

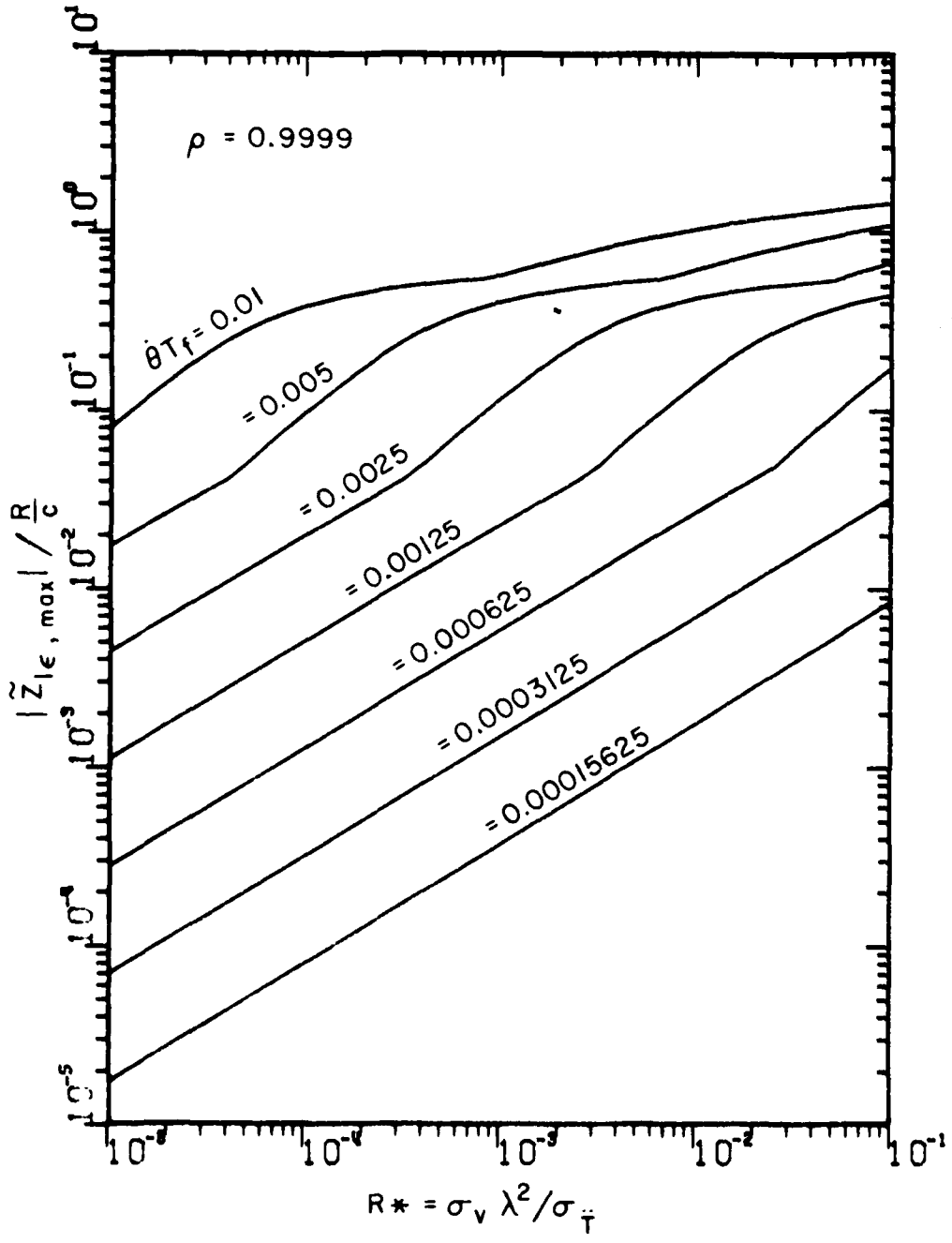


Figure 13e. The maximum delay tracking error for the orthogonal turn as a function of model and maneuver parameters, ρ=0.9999

CHAPTER V
COMBINED NOISE AND MANEUVER PERFORMANCE OF THE KALMAN TRACKING LOOP

A. Introduction

In the previous two chapters the performance of the Kalman clock loop in response to noise inputs (Chapter III) and delay inputs which simulate the inputs generated by a maneuvering terminal (Chapter IV) was found. At any given time the total delay tracking error $\tilde{z}_{1_\varepsilon}$ is the sum of the error due to noise $\tilde{z}_{1_{\varepsilon_n}}$ and of the error due to maneuvers $\tilde{z}_{1_{\varepsilon_m}}$. That is,

$$\tilde{z}_{1_\varepsilon} = \tilde{z}_{1_{\varepsilon_n}} + \tilde{z}_{1_{\varepsilon_m}} . \tag{59}$$

The noise component of this sum is a jitter term with standard deviation $\sqrt{P_{1_n}}$ whereas the maneuver term is deterministic. Thus, the error may be considered jittering with standard deviation $\sqrt{P_{1_n}}$ about a bias term $\tilde{z}_{1_{\varepsilon_m}}$. The worst case behavior occurs when the bias (maneuver) term is a maximum. The maximum delay tracking errors in response to two orientations of 180^0 turns were found in Chapter IV.

In this chapter, three examples of Kalman clock loop predictors are presented. In each case, the delay error standard deviation obtained from Kalman theory ($\sqrt{\tilde{P}_1}$) is compared with (1) the delay error standard deviation due to measurement noise only ($\sqrt{\tilde{P}_{1_n}}$) and (2) the maximum tracking error encountered in a 5 g. (1 g = 9.8 m/sec), 500 mi/hr. orthogonal 180^0 turn with no measurement noise. In each example the measurement noise standard deviation is assumed to be $.05\Delta$. This is approximately the same value as the measurement noise in the current experimental system.

B. Example 1

In this example model parameters identical to those considered in Reference 6, Example 1, pp. 111-116 are used. These are summarized in Table 2. A short description of how these parameters were obtained is given here. Values for the code rate, the average maneuver rate (λ), and the measurement noise (σ_v) were either given or assumed. The value of $\sigma_{\ddot{I}}$ was derived using Equation (17) under the assumption that P_0 (the probability of zero acceleration) is 0.5, P_m (the probability of maximum acceleration) is zero, and A (the maximum permitted

vehicle acceleration) is 5 g. (49 m/sec^2). The value of T_f was chosen (by an iteration scheme) to be the largest value which yields $\sqrt{\tilde{P}_1} < .05\Delta$. It has been shown[7] that timing jitters of greater than $.05\Delta$ may yield a significant degradation in bit error probability. The complete set of model parameters is given in Table 2.

From the set of model parameters the derived parameters ρ and R^* are found to be .9861 and $7.463\ 10^{-4}$, respectively. These values may be used with Figure 7 to find the tracking error standard deviation due to measurement noise only ($\sqrt{\tilde{P}_{1n}}$). Unfortunately a curve for $\lambda T_f = 0.014$ is not plotted. However, by extrapolating those values that are plotted, it is found that $\sqrt{\tilde{P}_{1n}}/\sigma_\xi = .88$, so that the standard deviation of the tracking error due to measurement noise only is $\sqrt{\tilde{P}_{1n}} = .044\Delta$.

The values of ρ and R^* may also be used with Figure 13 to find the tracking error generated by a terminal performing an orthogonal 180^o turn at 500 mi/hr. (232.52 m/sec) and 5 g. (49 m/sec^2) radial acceleration. This turn has a delay radius of

$$\frac{Rad}{c\Delta} = 135.95 \text{ chips} \tag{60}$$

and an angular velocity of

$$\dot{\theta} = 0.2192 \text{ radians/sec.} \tag{61}$$

Thus,

$$\dot{\theta} T_f = 0.0152 \text{ radian.} \tag{62}$$

Using Figures 13a, 13b, and 13c and extrapolating to obtain a value for each figure for $\dot{\theta} T_f = .0152$, then extrapolating these three values to obtain a value for $\lambda T_f = 0.014$ yields

$$\frac{\varepsilon_{max}}{Rad} \doteq 1.0 \cdot 10^{-3} \tag{63}$$

Unnormalizing, the maximum tracking error obtained is approximately

$$\frac{\varepsilon_{max}}{\Delta} \doteq .136 \text{ chip.} \tag{64}$$

This tracking error may be regarded as a bias in the prediction due to the maneuver.

The initial selection of T_f for this example was based upon obtaining a tolerable clock loop jitter of .05Δ. In selecting this value as tolerable, an implicit assumption was made that the clock loop yields estimates that are unbiased. This is true for a non-maneuvering terminal or a terminal making random maneuvers. However, with deterministic maneuvers such as the 180^0 turn considered here, biases in the estimate will occur. These may or may not be tolerable, depending upon the frequency of the maneuver, whether communications are expected during the maneuver, etc. In the present case, a clock loop operating with a jitter of .044Δ biased by .136Δ will result in a bit error rate of five to six times that of a similar clock loop operating unbiased with a jitter of 0.05Δ (assuming one operates with a bit error rate near 10^{-4}[7,22]).

To minimize the bias, it may be necessary to reduce T_f (i.e., to sample more often). Consider a system with the same model parameters as the previous system with the exception of T_f. Let T_f be 0.0228 (Example 1b). Since the maneuver rate (λ) is unchanged, ρ ($1-\lambda T_f$) becomes 0.995. Similarly, $\dot{\theta}T_f$ now becomes 0.005. Using Figure 7, the Kalman prediction error standard deviation ($\sqrt{\tilde{P}_1}$) and the prediction error standard deviation due to noise only ($\sqrt{\tilde{P}_{1n}}$) are found to be .03Δ and .027Δ, respectively. From Figure 13c (corresponding to ρ=0.995), the maximum tracking error for the 180^0 orthogonal turn is found to be 0.07Δ. The performance of the clock loop filter with this set of model parameters is summarized in Table 2 for both T_f = .0695 (Example 1a) and T_f = .0228 (Example 1b).

C. Example 2

In this example model parameters identical to those considered in Reference 6, Example 2, pp. 116-119 are used. These are summarized in Table 3. Here, σ_v and λ have the same values as Example 1. The code rate is doubled to 80×10^6 chips/sec, which doubles $\sigma_{\ddot{T}}$ to 5.36Δ /sec^2. T_f was reduced to .052 sec. Given $\sigma_{\ddot{T}}$, σ_v, λ and Δ (chip duration), T_f was chosen[6] such that the Kalman predictor standard deviation of the tracking error is

$$\sqrt{\tilde{P}_1} = .05\Delta. \tag{65}$$

From the model parameters, the derived parameters R* and ρ are given by

$$R^* = \frac{\sigma_v \lambda^2}{\sigma_{\ddot{T}}} = 3.731 \cdot 10^{-4} \tag{66}$$

Table 2
Model parameters and performance data for Examples 1a and 1b

Model Parameters	1a	1b
Code rate ($1/\Delta$)	40×10^6 chips/sec	same
Measurement noise standard deviation (σ_v/Δ)	.05 chip	same
Average maneuver rate (λ)	0.2 maneuvers/sec	same
Maneuver noise standard deviation ($\sigma_{\ddot{T}}/\Delta$)	2.68 chips/sec^2	same
Sample interval (T_f)	0.0695 sec	0.0228 sec
Derived Parameters		
Acceleration correlation coefficient (ρ)	0.9861	.9954
$R^* = \frac{\sigma_v \lambda^2}{\sigma_{\ddot{T}}}$	$7.463\ 10^{-4}$	same
Maneuver Parameters (for 5 g., 500 mi/hr. turn)		
Delay radius of turn (Rad/cΔ)	135.95 chips	same
Angular velocity ($\dot{\theta}$)	0.2192 rad/sec	same
Angle subtended in one sample interval ($\dot{\theta}T_f$)	0.0152 radian	.0050 rad
Performance		
Kalman prediction delay error deviation ($\sqrt{\tilde{P}_1}/\Delta$)	.05 chip	.03 chip
Prediction delay error standard deviation due to measurement noise only ($\sqrt{\tilde{P}_{1n}}/\Delta$)	.044 chip	0.027 chip
Prediction tracking error due to 5 g., 500 mi/hr. 180° orthogonal turn (ϵ_{max}/Δ)	0.136 chip	0.07 chip

and

$$\rho = 1 - \lambda T_f = 0.9896.$$

The 5 g. (49 m/sec^2), 500 mi/hr. (223.52 m/sec) turn has a radius (Rad) of 1019.6 meters. This generates a change in delay of

$$\frac{Rad}{c\Delta} = 271.90 \text{ chips.} \tag{67}$$

The angular velocity ($\dot{\theta}$) for the turn is 0.2192 radian/sec so the angle subtended in one sample interval is given by

$$\dot{\theta}T_f = .0114 \text{ rad.} \tag{68}$$

From Figure 7, the normalized standard deviation of the error due to measurement noise only ($\sqrt{\tilde{P}_{1n}}/\sigma_v$) is 0.88 (for ρ=0.99, $R^*=3.73 \cdot 10^{-3}$). Thus, the unnormalized jitter is

$$\sqrt{\tilde{P}_{1n}} = .044\Delta.$$

From Figure 13a (corresponding to ρ=0.99) the normalized maximum tracking error (ε_{max}/Rad) for $\dot{\theta}T_f$=.0114 is extrapolated to be $6.4 \cdot 10^{-4}$. The maximum tracking error is found to be

$$\varepsilon_{max} = 0.174\Delta.$$

A bias in the local clock of this magnitude will result in a degradation in the bit error probability (P_E) by at least a factor of ten for systems operating unbiased with P_E smaller than 10^{-4} [7,22]. As in Example 1, this may or may not be tolerable. Table 3 presents a short summary of this example.

D. Example 3

This example illustrates that the data of Chapters two and three may be used not only to evaluate the performance of a Kalman predictor, but also to select a set of model parameters to achieve some desired predictor performance criteria. This may be necessary because a designer may not have enough aircraft data to select the statistical parameters $\sigma_{\ddot{r}}$ and λ with confidence. This may also be desirable in that the designer may want to deliberately skew the model parameters (to reduce the errors in tracking of maneuvers while minimizing possible degradations in jitter performance, for example). The selection of model parameters based on the data of Chapters two and three is possible because the data in these chapters is independent of the accuracy of the parameters. That is, given a set of model parameters, the performance of the resulting Kalman predictor will be as described

Table 3
Model parameters and performance data for Example 2

Model Parameters	
Code rate $(1/\Delta)$	80×10^6 chips/sec
Measurement noise standard deviation (σ_v/Δ)	0.05 chip
Average maneuver rate (λ)	0.20 manuevers/sec
Manuever noise standard deviation $(\sigma_{\ddot{r}}/\Delta)$	5.36 chips/sec^2
Sample interval (T_f)	0.052 sec
Derived Parameters	
Acceleration correlation coefficient (ρ)	.99896
$R^* = \frac{\sigma_v \lambda^2}{\sigma_a}$	$3.73\ 10^{-4}$
Maneuver Parameters (for 5 g., 500 mi/hr. turn)	
Delay radius of turn (Rad/cΔ)	271.89 chips
Angular velocity $(\dot{\theta})$	0.2192 rad/sec
Angle subtended in one sample interval $(\dot{\theta}T_f)$	0.0114 rad
Performance	
Kalman prediction delay error standard deviation $(\sqrt{\tilde{P}_1}/\Delta)$	0.05 chip
Prediction delay error standard deviation due to measurement noise only $(\sqrt{\tilde{P}_{1n}}/\Delta)$	0.044 chip
Prediction tracking error due to 5 g., 500 mi/hr. 180^0 orthoronal turn (ϵ_{max}/Δ)	0.174 chip

by the figures of these chapters regardless of whether the model accurately reflects the physical situation. This is not true of the Kalman covariance matrix ($\tilde{P}$), which is accurate only if the model and model parameters accurately reflect the physical situation.

In Example 1a, given σ_v, $\sigma_{\dddot{T}}$, and λ, T_f was chosen to be .0695 sec. in order to yield a Kalman prediction error standard deviation ($\sqrt{\tilde{P}_1}$) of .05Δ. It was observed that the maximum tracking error through the 5 g., 500 mi/hr. 180° turn was 0.136Δ and that the jitter due to measurement noise only was .044Δ. In Example 1b T_f was reduced while holding the other model parameters constant in order to reduce the tracking error due to the maneuver. This resulted in a maximum tracking error of .07Δ and a jitter due to measurement noise of 0.027Δ. Consider now the following question: While holding the sample interval (T_f) and the measurement noise standard deviation constant at the same values as those considered in Example 1b (.0228 sec and .05Δ, respectively), is it possible to select $\sigma_{\dddot{T}}$ and λ such that the resulting filter tracks the 180° turn with an error less than .03 and has a jitter due to measurement noise of less than .04Δ?

To answer this question, a value for ρ (and hence λ) will be assumed. Figures 13 (a,b,c,d, or e) and 7 will be examined to find a value for R* such that both tracking and noise criteria can be met. If such a value is found, then the remaining parameter ($\sigma_{\dddot{T}}$) can be easily found, since σ_v, λ and R* have been specified. If no such value for R* is found, a different value for ρ will be assumed and the process repeated. This procedure will continue until a set of model parameters is found or until all of the plotted values for have been examined and shown to yield negative results.

Let ρ equal 0.99. From Equation (13) λ is 0.439 maneuvers/sec. From Figure 7, values for R* are desired such that $\tilde{P}_{1n}/\sigma_v \leq 0.8$ (i.e., $\sqrt{\tilde{P}_{1n}}$ .04 when σ_v=0.5). This criterion can be met if R* is greater than $5.56\ 10^{-4}$. From Figure 13a (corresponding to ρ=0.99), the maximum tracking error for the orthogonal turn (Rad/cΔ = 135.95 chips, T_f=0.005) is less than 0.03Δ (i.e., $\epsilon_{max}/\text{Rad} \leq 2.2 \cdot 10^{-4}$) if R* is less than $1.0\ \ 10^{-3}$. Thus, a useable range for R* is

$$5.56\ \ 10^{-4} \leq R^* \leq 1.0 \cdot 10^{-3} \ . \tag{69}$$

Choosing R* to be $7.5 \cdot 10^{-4}$ results in a value for $\sigma_{\dddot{T}}$ of $12.82\Delta/\text{sec}^2$. From Figure 7, the jitter due to measurement noise only is

$$\sqrt{\tilde{P}_{1n}} = .037\Delta \tag{70}$$

while the Kalman prediction delay tracking jitter is

$$\sqrt{\tilde{P}_1} = .044\Delta. \tag{71}$$

From Figure 13a the maximum tracking error due to the 180^o orthogonal turn is 0.027Δ. The model parameters and performance data for this example are summarized in Table 4.

Thus, while sampling at the same rate and with the same energy to noise density ratio in the clock pulse (i.e., the same σ_γ) as in example 1b, the predictor considered here will track the delay from a 180^o turn with much less error ($.027\Delta$) than the predictor of example 1b ($.07\Delta$). The jitter due to measurement noise is somewhat larger here ($.037\Delta$) than the previous example ($.027\Delta$), but still very reasonable.

Of course, in skewing the model parameters to optimize tracking performance, the relation between the model parameters and an actual aircraft may be lost. That is, if the model parameters were chosen based upon the physical characteristics of an aircraft (e.g., the maximum acceleration the aircraft is able to withstand), they may no longer be realistic with respect to those same characteristics. For example, in the case considered here the value of $12.82\Delta/sec^2$ for $\sigma_{\ddot{T}}$ corresponds to approximately 9.81 g., which is beyond the capabilities of most present day aircraft. This, however, does not indicate problems with either Kalman theory on the chosen model. The Kalman algorithms are minimum mean (i.e., average) square error algorithms. This study considers the performance with respect to two different performance criteria, the response due to measurement noise (jitter performance) and the tracking errors resulting from 180^o turns. The model parameters which are necessary to provide good performance with respect to these two criteria may be quite different than those required for the minimum square error criterion.

Obviously, good tracking performance for typical maneuvers is important. This example shows that (1) good tracking performance for typical maneuvers is obtainable simultaneously with good jitter performance and (2) that the curves in Chapters III and IV can be used to obtain the model parameters necessary for the implementation of a predictor with this performance.

Table 4
Model parameters and performance data for Example 3

Model Parameters	
Code rate $(1/\Delta)$	$40 \cdot 10^6$ chips/sec
Measurement noise standard deviation (σ_v/Δ)	.05 chips
Average maneuver rate (λ)	.4386 maneuvers/sec
Manuever noise standard deviation $(\sigma_{\ddot{T}}/\Delta)$	12.82 chips/sec^2
Sample interval (T_f)	0.0228 sec.
Derived Parameters	
Acceleration correlation coefficient (ρ)	0.99
$R^* = \frac{\sigma_v \lambda^2}{\sigma_{\ddot{T}}}$	$7.5 \cdot 10^{-4}$
Manuever Parameters for 5 g., 500 mi/hr. turn	
Delay radius $(Rad/c\Delta)$	135.95 chips
Angular velocity $(\dot{\theta})$	0.2192 rad/sec
Angle subtended in one sample period $(\dot{\theta}T_f)$	0.005 rad
Performance	
Kalman predictor delay error standard deviation $(\sqrt{\tilde{P}_1})$	$.044\Delta$
Predictor delay error standard deviation due to measurement noise only $(\sqrt{\tilde{P}_{1n}})$	$.037\Delta$
Predictor tracking error (maximum) due to 5 g., 500 mi/hr. 180^0 orthogonal turn (ε_{max})	$.027\Delta$

CHAPTER VI
TRANSIENT PERFORMANCE OF THE KALMAN FILTER

A. Introduction

In this chapter, some aspects of the Kalman and Wiener clock loop filters during acquisition or lock up are examined. Since the Kalman filter is computationally more costly to implement than the Wiener filter, knowledge of when it is advantageous to use one over the other is needed. The important criteria for comparison are the magnitude of the tracking errors (overshoots and/or lags) during lock up and the time required for the error to converge to reasonably small values. Two computer programs were written, named TRANS and STEADY, to examine the behavior of the Kalman filter (TRANS) and the Wiener filter (STEADY) under lock up conditions. These programs and the data they generate are examined in this chapter. Before discussing these programs, the initialization of the Kalman filter as described in Reference 6 is examined.

B. Initialization of the Kalman TDMA Clock Loop Filter

In Reference 6, it is proposed that the initial estimate ($\hat{z}(-1)$) be obtained from two measurements prior to enabling the filter. The observation models corresponding to the two measurements are

$$x(-2) = z_1(-2) + \xi(-2) \tag{72}$$

and

$$x(-1) = z_1(-1) + \xi(-1) \tag{73}$$

and the estimate $\hat{z}(-1)$ is

$$\begin{pmatrix} \hat{z}_1(-1) \\ \hat{z}_2(-1) \\ \hat{z}_3(-1) \end{pmatrix} = \begin{pmatrix} x(-1) \\ \dfrac{x(-1)-x(-2)}{T_f} \\ 0 \end{pmatrix} = \begin{pmatrix} z_1(-1) + \xi(-1) & (74) \\ \dfrac{z_1(-1)-z_1(-2)}{T_f} + \dfrac{\xi(-1)-\xi(-2)}{T_f} & (75) \\ 0 & (76) \end{pmatrix}$$

Note that no attempt is made to secure acceleration information. The initial prediction is obtained from the filtered estimate $\hat{z}(-1)$ via Equation (6b) and is

$$\tilde{z}(0/-1) = \Phi\hat{z}(-1) \ . \tag{77}$$

The effects of noise and the neglected acceleration component on the initial prediction $\tilde{z}(0/-1)$ will now be analyzed. Here, an acceleration with constant magnitude of $\ddot{T}_o$ (chips/sec) throughout the initialization period as assumed. Thus, with this constraint, the state vector z(k) is given by

$$z(k) = \begin{pmatrix} z_1(k) \\ z_2(k) \\ z_3(k) \end{pmatrix} = \begin{pmatrix} z_1(k) \\ z_2(k) \\ \ddot{T}_o \end{pmatrix} , \quad \ddot{T}_o \text{ constant} . \tag{78}$$

At k=-1, the 1st two components of the state vector z are given by

$$z_1(-1) = z_1(-2) + z_2(-2)T_f + \ddot{T}_o T_f^2/2 \tag{79a}$$

and

$$z_2(-1) = z_2(-2) + \ddot{T}_o T_f . \tag{79b}$$

From Equation (75) and Equation (79a and b),

$$\hat{z}_2(-1) = z_2(-1) - \frac{1}{2}\ddot{T}_o T_f + \frac{\xi(-1)-\xi(-2)}{T_f} \tag{80}$$

is obtained.

At k=0, the 1st two components of the state vector are

$$z_1(0) = z_1(-1) + z_2(-1)T_f + \ddot{T}_o T_f^2/2 \tag{81a}$$

$$z_2(0) = z_2(-1) + \ddot{T}_o T_f \tag{81b}$$

From Equation (77), the predicted values for $z_1(0)$ and $z_2(0)$ are

$$\tilde{z}_1(0/-1) = \hat{z}_1(-1) + \hat{z}_2(-1)T_f \tag{82a}$$

$$\tilde{z}_2(0/-1) = \hat{z}_2(-1) \tag{82b}$$

By substituting $z_1(-1)$ from Equation (74) and $z_2(-1)$ from Equation (80) into Equations (82a) and (82b) we obtain

$$\tilde{z}_1(0/-1) = z_1(-1) + z_2(-1)T_f - \frac{1}{2}\ddot{T}_o T_f^2 + 2\xi(-1)-\xi(-2)$$

which can be rewritten using $z_1(0)$ from Equation (81a) as

$$\tilde{z}_1(0/-1) = z_1(0) - \ddot{T}_o T_f^2 + 2\xi(-1) - \xi(-2) \tag{83}$$

Thus, the prediction $\tilde{z}_1(0/-1)$ can be considered biased by an acceleration term of $-\ddot{T}_0 T_f^2$ and corrupted by 2 noise samples, $\xi(-1)$ and $\xi(-2)$. Similarly for the velocity component of the state vector, by substituting $\dot{z}_2(-1)$ from Equation (80) into Equation (82b) we obtain

$$\tilde{z}_2(0/-1) = z_2(-1) - \frac{1}{2}\ddot{T}_0 T_f + \frac{\xi(-1)-\xi(-2)}{T_f} \tag{84}$$

From Equation (81b), $z_2(-1) = z_2(0) - \ddot{T}_0 T_f$, so Equation (84) becomes

$$\tilde{z}_2(0/-1) = z_2(0) - \frac{3}{2}\ddot{T}_0 T_f + \frac{\xi(-1)-\xi(-2)}{T_f} \tag{85}$$

Thus, the velocity prediction is biased by an acceleration term of $-3/2\ \ddot{T}_0 T_f$ and corrupted by 2 noise samples weighted by $1/T_f$. For PN code rates of $40 \cdot 10^6$ chips/sec and T_f=.01 sec (quite large, considering the chip rate), the bias errors introduced into the delay and velocity predictions are $1.3*10^{-4}$ chips and $2*10^{-3}$ chips/sec, respectively, per G (1 G $\doteq$ 9.8 m/sec^2) of acceleration. Both of these are indeed negligible.

From Equations (83) and (85), the errors in the delay and velocity components of the estimates due to noise are respectively

$$\tilde{z}_{1\epsilon_n}(0) = 2\xi(-1)-\xi(-2) \tag{86a}$$

and

$$\tilde{z}_{2\epsilon_n}(0) = \frac{\xi(-1)-\xi(-2)}{T_f} \tag{86b}$$

The error variances associated with these terms are (since the noise samples are uncorrelated)

$$E\{(\tilde{z}_{1\epsilon_n}(0))^2\} = 5\sigma_\xi^2 \tag{87a}$$

and

$$E\{(\tilde{z}_{2\epsilon}(0))^2\} = \frac{2\sigma_\xi^2}{T_f^2} \tag{87b}$$

with corresponding standard deviations of

$$\sigma_1 = \sqrt{E\{(\tilde{z}_{1\epsilon_n}(0))^2\}} = 2.24\ \sigma_\xi \tag{88a}$$

and

$$\sigma_2 = \sqrt{E\{(\tilde{z}_{2\varepsilon_n}(0))^2\}} = \frac{1.41}{T_f}\sigma_\xi \qquad (88b)$$

Equations (88a) and (88b) are important statistics because they give information about the magnitude of the errors. Assuming that the errors are Gaussian distributed, the prediction ($\tilde{z}(0/-1)$) will be with in one standard deviation of the correct value 68% of the time and within 1.65 standard deviations 90% of the time[13]. For example, if σ_ξ=0.05Δ (a reasonable figure for the present system), then the errors in the initial delay prediction $\tilde{z}_1(0)$ would be less than .185Δ 90% of the time. Similarly, for T_f = 0.01 sec., the errors in the initial velocity prediction $\tilde{z}_2(0)$ would be less than 7.05Δ/sec (or .07Δ/T_f) 90% of the time. The programs TRANS and STEADY simulate the behavior of Kalman and Wiener filters during lock-up with initial errors of these magnitudes.

C. The Steady State Kalman Filter Program

The program STEADY (Appendix V) is structured to make extensive use of subroutines. Because the input signal to the Kalman filters is the difference between the time bases of the local and received clock signals, initialization with an error in any of the components of prediction $\tilde{z}(0/-1)$ is equivalent to a step change of opposite sign in the corresponding component of state vector $z(0)$.

Three subroutines were written which generate the state vector simulating step changes in delay (T_0), velocity ($\dot{T}_0$), or acceleration ($\ddot{T}_0$). These are names STEP, RAMP, and PARAB, respectively. The names originate because a step change in velocity produces a ramp in delay and step change in acceleration produces a parabolic delay.

After the state vector is generated, it is compared with the prediction in subroutine ERROR1 or ERROR2. These subroutines keep track of the errors as the filter converges. It was discovered that for step changes in delay (T_0), the maximum error almost always occurs at the first iteration and is equal to the magnitude of the step. When this is not the case, the maximum error occurs in the first overshoot. Accordingly, ERROR2 was written which computes the magnitude of the first overshoot. This subroutine is used for step changes in delay. For steps in velocity and acceleration, ERROR1 is used. This subroutine computes the maximum error as the filter converges regardless of where it occurs.

After the error is observed, two subroutines are used to check convergence. The first, subroutine COUNT, counts the number of iterations for the first (the delay) component of the prediction to converge to within .05Δ of the actual delay. This is used to obtain data concerning the convergence time of the filter. The second subroutine, STOP2, is used to determine when all three components of $\tilde{z}(0)$ have converged to within .001 (chips, chips/sec, or chips/sec^2 depending upon the component) of the corresponding actual values. If all three components have converged, a variable ISTOP is set to one which ceases the iterations. If not, subroutine PRED is used to generate a new prediction by direct implementation of Equation (3). Subroutine PRED may be used with either time variable or steady-state gains. It should be noted that subroutine PRED implements a linear tracking characteristic. The nonlinear characteristic of the correlation circuitry could be simulated, but this would prohibit the normalization of output data. From this point, program flow is dependent upon whether the filter has converged (by the criteria of STOP2) or not. If it has converged, then either the number of iterations (from COUNT) to converge or the error (from ERROR1 or ERROR2) during the convergence process may be written in an output array. If the filter has not converged, then the next state is generated and the process is repeated.

D. The Transient Kalman Filter Program

The program TRANS (Appendix VI) uses the same subroutines and is organized in the same manner as STEADY. Three additional subroutines are also used for functions unique to the time variable (Kalman) filter. First, it is necessary to obtain an initial error covariance matrix, $\tilde{P}(0)$. In Ref. 6 it is shown that this matrix should be

$$\tilde{P}(0) = \begin{bmatrix} 5\left[1+\frac{8(1+\rho)}{5r^2}\right]\sigma_v^2 & \frac{3}{4}r\left[1+\frac{4(1+\rho)}{r^2}\right]\sigma_v\sigma_a T_f & \frac{2(1+\rho)}{r}\rho\sigma_v\sigma_a \\ \frac{3}{4}r\left[1+\frac{4(1+\rho)}{r^2}\right]\sigma_v\sigma_a T_f & \frac{1}{8}r^2\left[1+\frac{10(1+\frac{4}{5}\rho)}{r^2}\right]\sigma_a^2 T_f^2 & (1+\frac{1}{2}\rho)\rho\sigma_a^2 T_f \\ \frac{2(1+\rho)}{r}\rho\sigma_v\sigma_a & (1+\frac{1}{2}\rho)\rho\sigma_a^2 T_f & \rho\sigma_a^2 \end{bmatrix} \quad (89)$$

where $r = \dfrac{4\sigma_v}{\sigma_{\ddot{T}} T_f^2}$

Subroutine SETUP2 implements Equation (89) directly. Second, it is necessary to calculate the predictor gains. This is accomplished in subroutine GPCALC via direct implementation of Equation (4). In this equation, the gains are calculated as a function of the error covariance matrix, hence the need for SETUP2 prior to GPCALC. Third, after the prediction is obtained, it is necessary to update the error covariance matrix for the next iteration. This is performed in subroutine PW1 by implementing Equation (5).

E. Results

These computer programs were used to find (1) the errors in delay tracking $\tilde{z}_{1\varepsilon}$ during lock-up and (2) the number of iterations for the delay error to fall below .05 chips for a wide variety of model parameters and inputs. The magnitude of the delay errors are presented in Figures 14, 15, and 16 for inputs of steps, ramps, and parabolas, respectively. In each case, the results are normalized to the input to make the curves more general. That is, the data for the step input is normalized to T_0, the ramp data to $\dot{T}_0 T_f$, and the parabola data to $\ddot{T}_0 T_f^2/2$. In all three figures, data for the Kalman filter is presented in solid lines and data for the Weiner filter is presented in dotted lines.

While developing, and debugging of the programs, it was determined that the maximum errors in the ramp and parabola cases are undershoots (or lags) in the response, whereas the errors in the step input case are overshoots. In Figure 15 the data for the Kalman response lies on the R* axis, not below it.

The data for convergence is presented in Figures 17 and 18 as a function of model parameters for the Wiener and Kalman filters. Figure 17 shows the iterations to converge given a step offset in delay of 0.20Δ. This corresponds to the 90% confidence interval for $\sigma_\xi=0.05\Delta$, $T_f=.01$. Figure 18 shows the iterations to converge for a step change in delay rate of 7 chips/sec. Again, this corresponds to the 90% confidence interval for $\sigma_\xi=.05\Delta$ and $T_f=.01$ sec. These figures are specific for the inputs described (they are not normalized). However, the general characteristics may be generalized. First, the most significant improvement of the Kalman filter over the Wiener filter occurs for large R* and large ρ. This corresponds to a relatively slow responding filter in the steady state. Second, the number of iterations required for convergence of the Kalman predictor seems to be asymptotic to 13 in Figure 17 and 31 in Figure 18. This behavior is explainable. In Reference 6 it is shown that the Kalman gain vectors for different model parameters all follow the same monotonically decreasing trajectory until very near this final (steady-state) value. Consider two filters, the first of which has a gain vector which converges to (within 1% of) steady-state in 200 iterations and the second of which has a gain vector which converges (1%) in 400 iterations. Suppose that both filters are given

the same input, and the first filter acquires the signal within 100 iterations. Under these conditions the second filter will also acquire the signal within 100 iterations because the gain vectors for the two filters are almost identical until near the 200th iteration. Furthermore, any filter whose gains go to steady state in significantly more than 100 iterations will have similar convergence characteristics. Third, for the slower filters, the difference between the Kalman and Wiener filters is dramatic. Consider $R^* = 10^{-2}$, ρ=.999, and T_f=.01. Figure 17 shows that the Kalman filter requires 13 iterations to converge (.13 sec at T_f=.01 sec) from an initial delay offset of .2Δ whereas the Wiener filter requires 150 iterations (1.5 sec). Figure 14, when interpreted for $T_o = 0.2\Delta$, shows that the 1st overshoot is .1Δ for the Kalman case and 0.07Δ for the Wiener case. Thus, the filter remains in the linear portion of the correlation circuitry throughout lock-up (see Figure 3). Similarly, for a velocity offset of 7Δ/sec (Figure 18), the Kalman filter requires 31 iterations to converge while the Wiener filter required over 700 iterations to converge. From Figure 15 the maximum errors during convergence are .07 (Kalman) and 1.4Δ (Wiener). Since the correlation circuitry is linear only over a $\pm\frac{1}{2}\Delta$ range, the Wiener filter would probably not lock under these conditions.

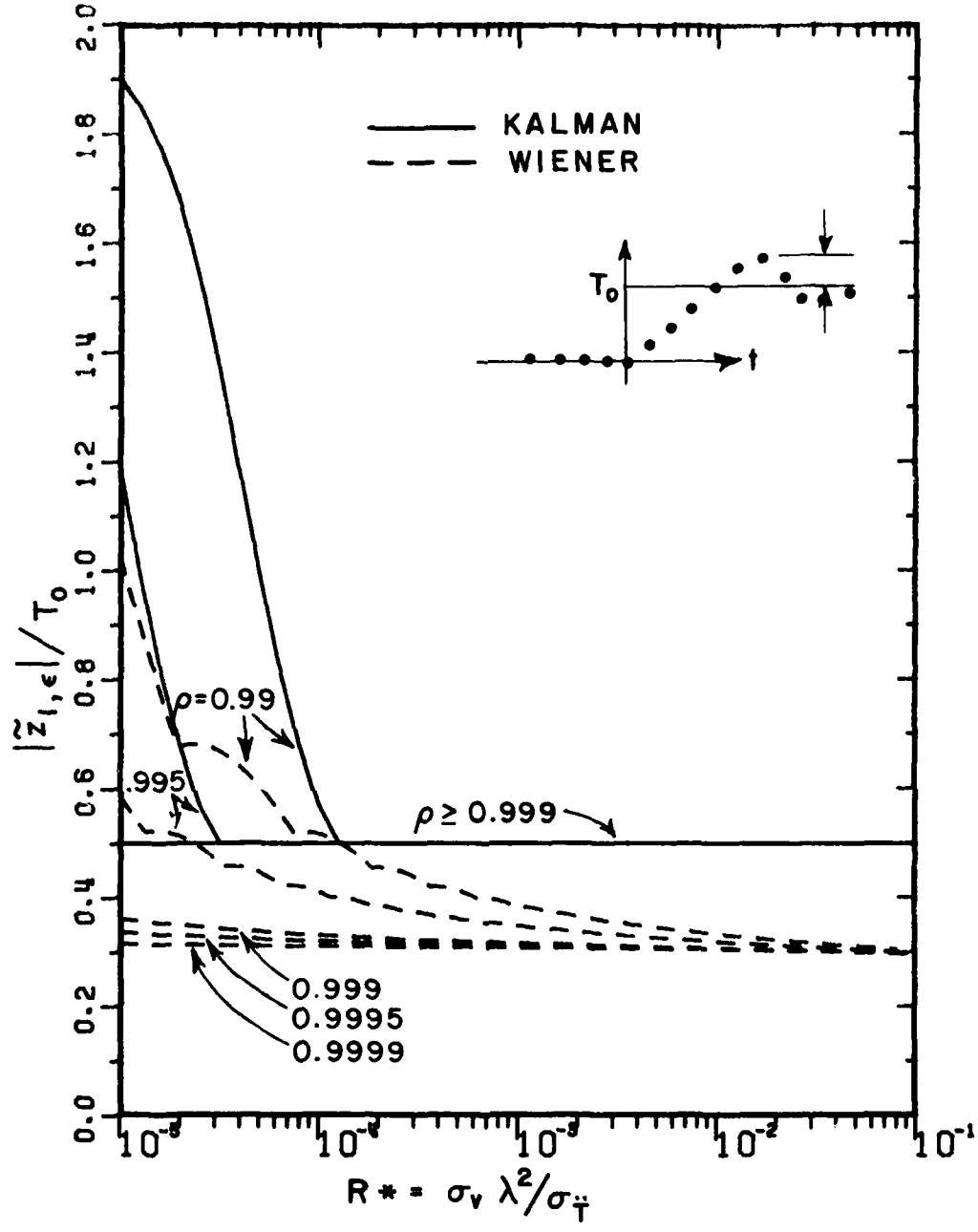


Figure 14. Magnitude of the 1st. overshoot of the predictor in response to step changes in delay of magnitude To.

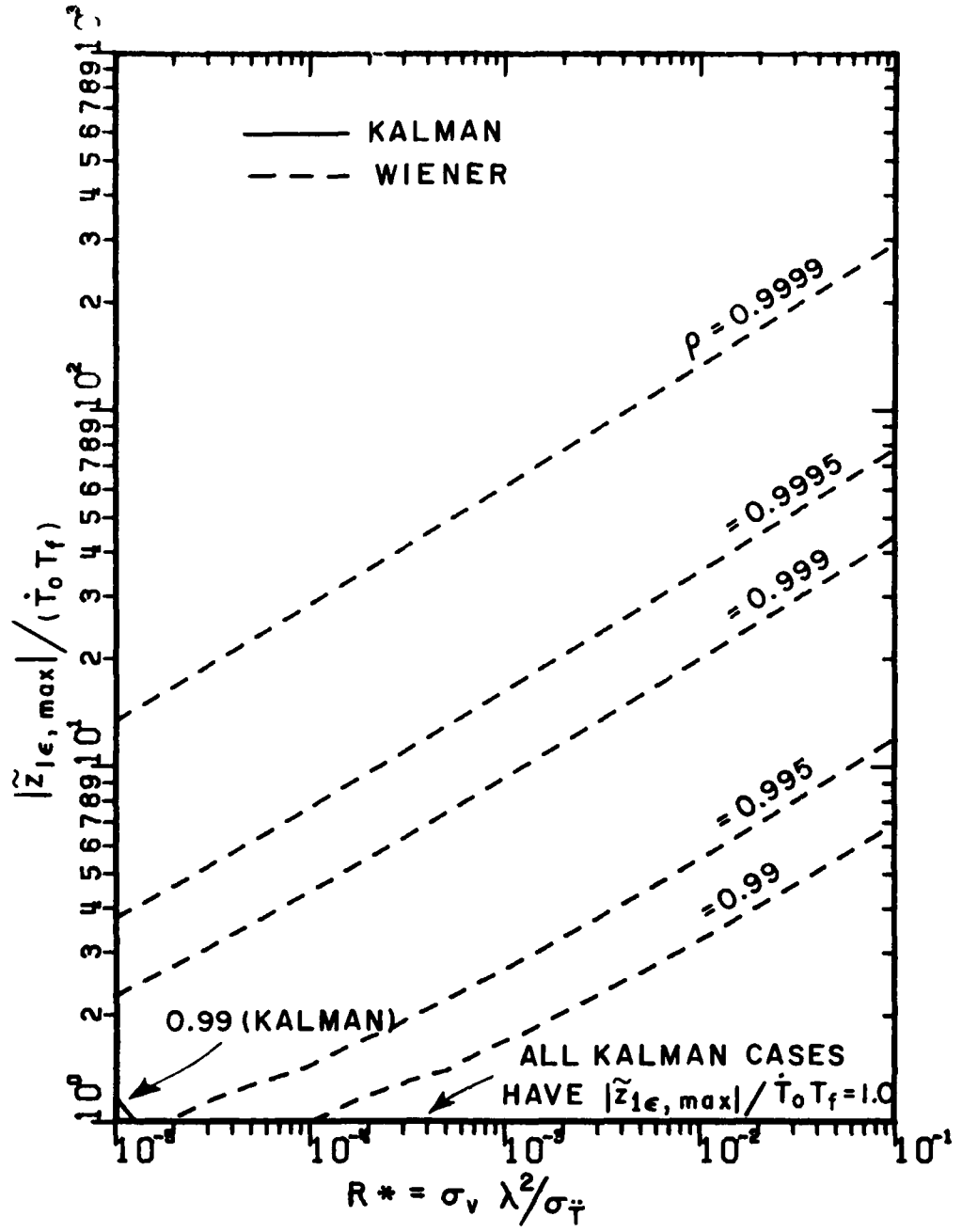


Figure 15. Magnitude of the maximum predictor tracking error in response to step changes in delay velocity of magnitude $\dot{T}_o$.

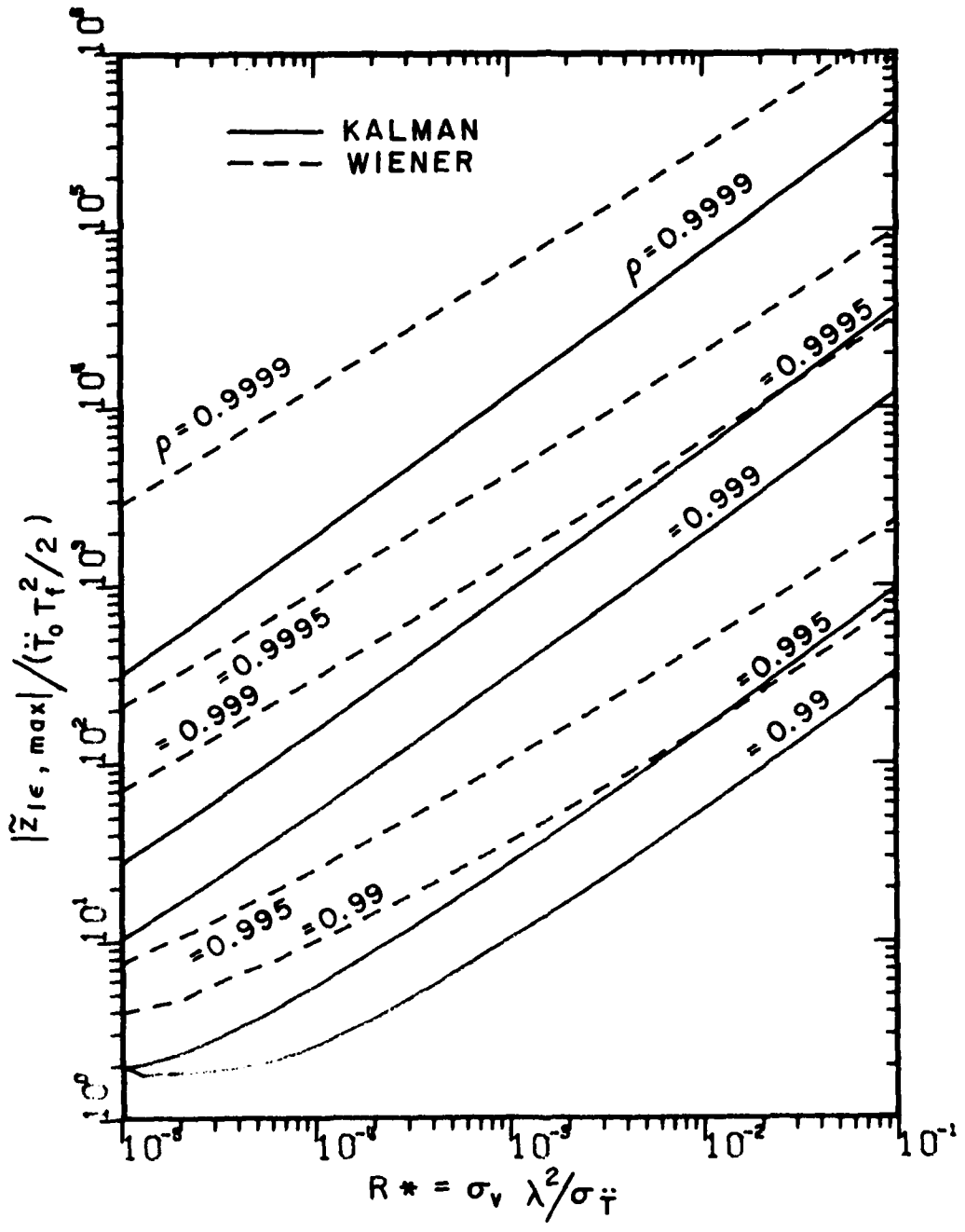


Figure 16. Magnitude of the maximum predictor tracking error in response to step changes in delay acceleration of magnitude $\ddot{T}_o$.

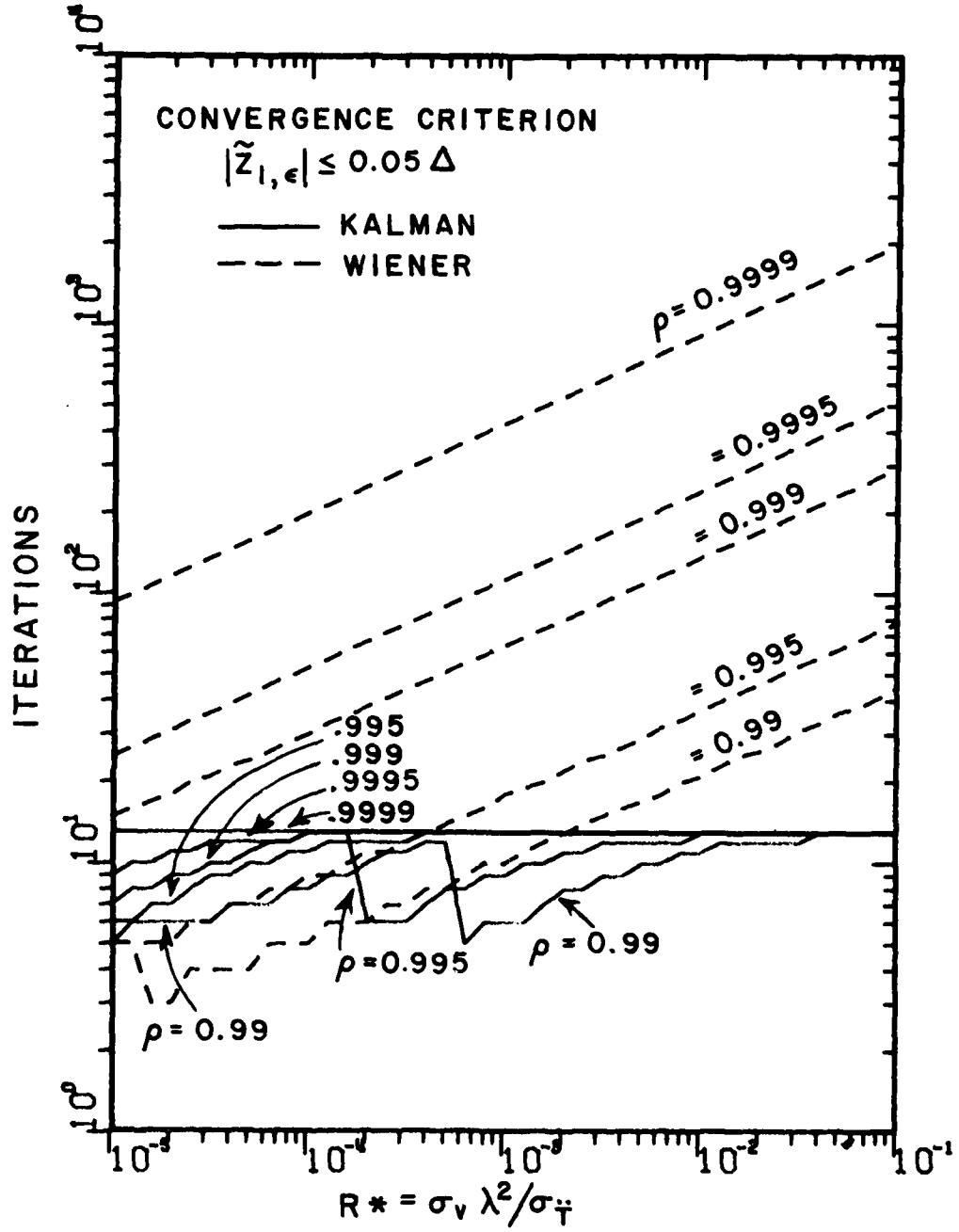


Figure 17. The number of iteration for the predictor tracking error to diminish to less than 0.05Δ given a step change in delay of 0.20Δ.

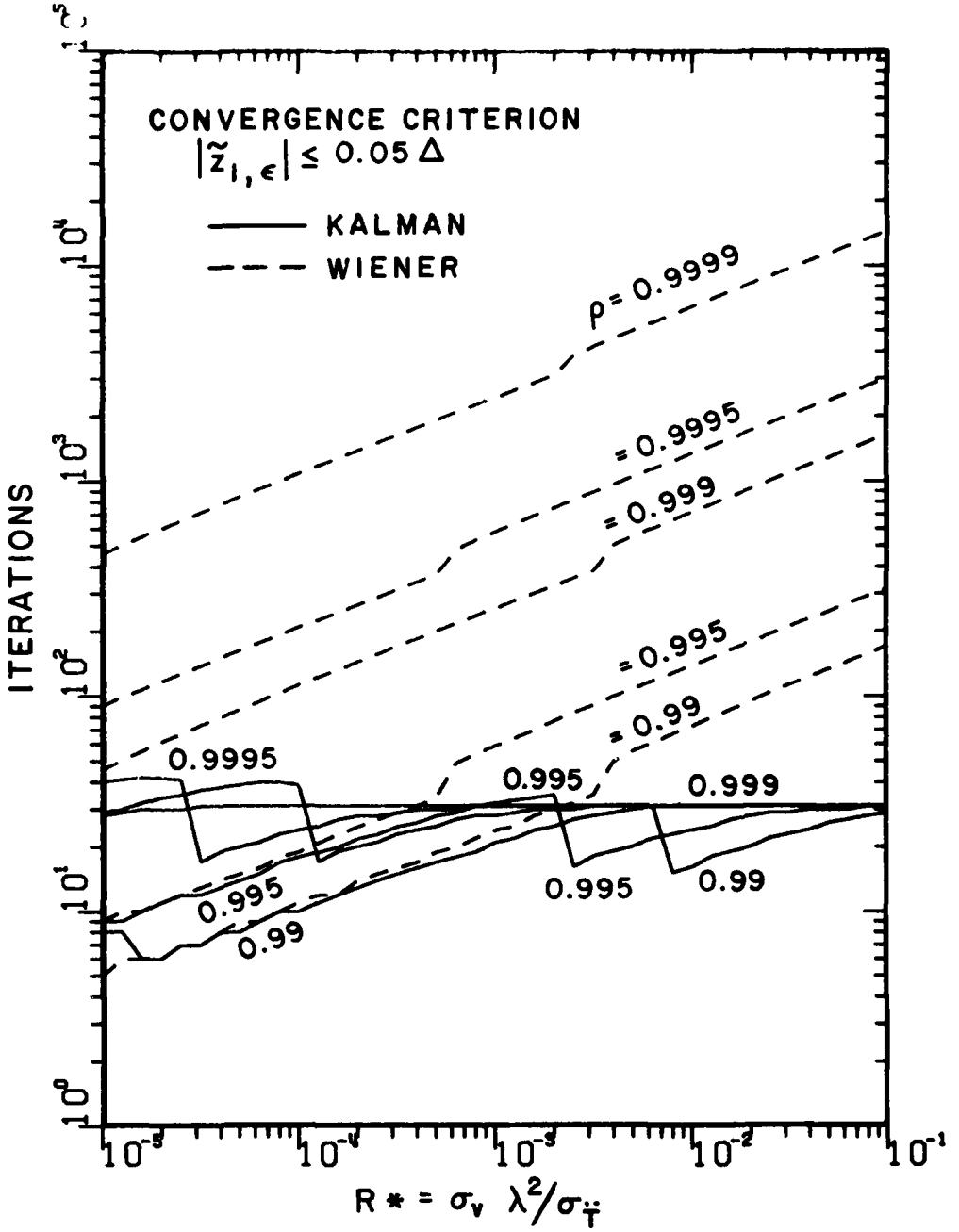


Figure 18. The number of iterations for the predictor tracking error to diminish to less than 0.05Δ given a step change in delay velocity of 7Δ/sec.

CHAPTER VII
CONCLUSIONS

The steady state and transient performance of a Kalman filter/predictor implementation of the receive clock loop in a TDMA satellite system has been analyzed. The steady-state response of the filter to measurement noise was determined by state-variable methods and verified by Monte-Carlo simulation and z-transform techniques. The steady-state responses of the filter to delay changes caused by a terminal performing 180^0 turns with two different orientations with respect to the satellite was determined by computer simulation.

Examples are given which show how the steady state responses may be used in the evaluation and design of the clock loop filter. The examples given show that the errors in response to terminals performing 180^0 turns are somewhat larger than those one would intuitively expect from the data given by the Kalman error covariance matrix. The Kalman algorithms, of course, minimize the average square error, not the error due to a single deterministic maneuver. Nevertheless, the response to typical maneuvers is important. The examples show how it is possible to deliberately skew the model parameters of the Kalman algorithms to improve the response to typical deterministic maneuvers with little degradation in noise performance (jitter).

The transient performance of the Kalman clock loop filter and the corresponding Wiener clock loop filter was analyzed by computer simulation. The tracking errors caused by errors in initialization for both Wiener and Kalman predictors were determined as a function of the model parameters of interest. The iterations to converge were also determined as a function of model parameters for two typical acquisitions. The results show that the Kalman filter has much better transient performance than the corresponding Wiener filter when the derived model parameters R^* and ρ are both relatively large.

Appendix I
The Computer Program VPCALC

```
C*************** PROGRAM VPCALC ***************
      OPTIONS DP
      DIMENSION PW(6),PWO(6),X(3),XHAT(3),GP(3),XNOI(500)
      DIMENSION XOUT(4,41,5),V(6)
      REAL LAMBDA(5)
      COMMON/PHI/RHO,T(4)
      COMMON/SET2/SIGV,VARV,SIGW,VARW,SIGA,VARA
      DATA LAMBDA/1...5,..1..05,.01/
      SIGV=.05
      VARV=SIGV**2
      CALL DEASSN
      T(1)=0.01
      T(2)=0.0001
      T(3)=0.000001
      T(4)=0.00000001
      FACTOR=10.**0.1
      DO 600 ILAM=1,5
      RHO=1-LAMBDA(ILAM)*T(1)
      R=10.**-5/FACTOR
      DO 600 IPT=1,41
      R=R*FACTOR
      SIGA=SIGV*LAMBDA(ILAM)**2/R
      VARA=SIGA**2
      VARW=VARA*(1.-RHO**2)
      SIGW=SQRT(VARW)
      IF(IPT.EQ.1)CALL SETUP(PW)
C*****ITERATE TO STEADY STATE
      IFLAG=0
      DO 100 I=1,10000
      CALL PWT(PW,PWO)
      CALL CHECKT(PW,PWO,IFLAG)
      IF(IFLAG.EQ.1)GO TO 110
  100 CONTINUE
C**************************
  110 DO 120 I=1,3
      XHAT(I)=0.
  120 CONTINUE
      CALL GPCALC(PW,GP)
      CALL VGAUS(GP,V)
      XOUT(1,IPT,ILAM)=R
      XOUT(2,IPT,ILAM)=SQRT(PW(1)/VARV)
      XOUT(3,IPT,ILAM)=SQRT(V(1)/VARV)
      XOUT(4,IPT,ILAM)=0.0
      CALL ASSIGN(6HDATAS6,5H4)ABS,2)
      WRITE(2)XOUT
      ENDFILE 2
  600 CONTINUE
      STOP 'DONE'
      CALL EXIT
      END
C**************************
C**************************
      SUBROUTINE CHECKT(PW,PWO,IFLAG)
      DIMENSION PW(6),PWO(6),P(6)
      DO 10 I=1,6
```

```
      IF(PW(I).EQ.0.)GO TO 11
      P(I)=(ABS(PW(I)-PWO(I)))/ABS(PW(I))
      GO TO 10
11    P(I)=0.
10    CONTINUE
      DO 15 I=1,6
15    IF(P(I).GE.10.**-11.)GO TO 50
      IFLAG=1
      RETURN
50    IFLAG=0
      RETURN
      END
C*************************************
      SUBROUTINE DECOMP(NN,A,UL)
      DIMENSION A(6,6),UL(6,6),SCALES(6),IPS(6)
      COMMON IPS
      N=NN
C
C     INITIALIZE IPS,UL,AND SCALES
      DO 5 I=1,N
      IPS(I)=I
      ROWNMR=0.0
      DO 2 J=1,N
      UL(I,J)=A(I,J)
      IF(ROWNMR-ABS(UL(I,J)))1,2,2
1     ROWNMR=ABS(UL(I,J))
2     CONTINUE
      IF(ROWNMR)3,4,3
3     SCALES(I)=1.0/ROWNMR
      GO TO 5
4     CALL SING(1)
      SCALES(I)=0.0
5     CONTINUE
C
C     GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING
      NM1=N-1
      DO 17 K=1,NM1
      BIG=0.0
      DO 11 I=K,N
      IP=IPS(I)
      SIZE=ABS(UL(IP,K))*SCALES(IP)
      IF(SIZE-BIG)11,11,10
10    BIG=SIZE
      IDXPIV=I
11    CONTINUE
      IF(BIG)13,12,13
12    CALL SING(2)
      GO TO 17
13    IF(IDXPIV-K)14,15,14
14    J=IPS(K)
      IPS(K)=IPS(IDXPIV)
      IPS(IDXPIV)=J
15    KP=IPS(K)
      PIVOT=UL(KP,K)
      KP1=K+1
```

```
      DO 15 I=KP1,N
      IP=IPS(I)
      EM=-UL(IP,K)/PIVOT
      UL(IP,K)=-EM
      DO 15 J=KP1,N
      UL(IP,J)=UL(IP,J)+EM*UL(KP,J)
C     INNER LOOP.  USE MACHINE LANGUAGE CODING IF COMPILER
C     DOES NOT PRODUCE EFFICIENT CODE
15    CONTINUE
17    CONTINUE
      KP=IPS(N)
      IF(UL(KP,N))19,18,19
18    CALL SING(2)
19    RETURN
      END
C*************************************
      SUBROUTINE SOLVE(NN,UL,B,X)
      DIMENSION UL(6,6),B(6),X(6),IPS(6)
      COMMON IPS
      N=NN
      NP1=N+1
C
      IP=IPS(1)
      X(1)=B(IP)
      DO 2 I=2,N
      IP=IPS(I)
      IM1=I-1
      SUM=0.0
      DO 1 J=1,IM1
1     SUM=SUM+UL(IP,J)*X(J)
2     X(I)=B(IP)-SUM
C
      IP=IPS(N)
      X(N)=X(N)/UL(IP,N)
      DO 4 IBACK=2,N
      I=NP1-IBACK
C     I GOES (N-1),....2,1
      IP=IPS(I)
      IP1=I+1
      SUM=0.0
      DO 3 J=IP1,N
3     SUM=SUM+UL(IP,J)*X(J)
4     X(I)=(X(I)-SUM)/UL(IP,I)
      RETURN
      END
C*************************************
      SUBROUTINE SING(IWHY)
      FORMAT(54H0MATRIX WITH ZERO ROW IN DECOMPOSE.
     )
12    FORMAT(54H0SINGULAR MATRIX IN DECOMPOSE.  ZERO DIVIDE IN SOLVE.
     )
13    FORMAT(54H0NO CONVERGENCE IN IMPROV. MATRIX IS NEARLY SINGULAR.
     )
      NOUT=8
C     NOUT=STANDARD OUTPUT UNIT
      GO TO (1,2,3),IWHY
1     WRITE(NOUT,11)
      GO TO 14
```

```
      WRITE(NOUT,12)
      GO TO 16
    3 WRITE(NOUT,13)
   16 RETURN
      END
C*****************************
      SUBROUTINE VPCALC(GP,V)
      DIMENSION A(6,6),UL(6,6),GP(3),V(6),B(6)
      COMMON/PPI/RHO,T(4)
      COMMON/SET2/SIGV,VARV,SIGW,VARW,SIGA,VARA
      A(1,1)=(1.-GP(1))**2-1.
      A(1,2)=2.*(1.-GP(1))*T(1)
      A(1,3)=(1.-GP(1))*T(2)
      A(1,4)=T(2)
      A(1,5)=T(3)
      A(1,6)=T(4)/4.
      A(2,1)=-(1.-GP(1))*GP(2)
      A(2,2)=-GP(1)-GP(2)*T(1)
      A(2,3)=(1.-GP(1))*T(1)-GP(2)*T(2)/2.
      A(2,4)=T(1)
      A(2,5)=3.*T(2)/2.
      A(2,6)=T(3)/2.
      A(3,1)=-(1.-GP(1))*GP(3)
      A(3,2)=-GP(3)*T(1)
      A(3,3)=(1.-GP(1))*RHO-GP(3)*T(2)/2.-1.
      A(3,4)=0.
      A(3,5)=RHO*T(1)
      A(3,6)=RHO*T(2)/2.
      A(4,1)=GP(2)**2
      A(4,2)=-2.*GP(2)
      A(4,3)=-2.*GP(2)*T(1)
      A(4,4)=0.
      A(4,5)=2.*T(1)
      A(4,6)=T(2)
      A(5,1)=GP(2)*GP(3)
      A(5,2)=-GP(3)
      A(5,3)=-RHO*GP(2)-T(1)*GP(3)
      A(5,4)=0.
      A(5,5)=RHO-1.
      A(5,6)=RHO*T(1)
      A(6,1)=GP(3)**2
      A(6,2)=0.
      A(6,3)=-2.*RHO*GP(3)
      A(6,4)=0.
      A(6,5)=0.
      A(6,6)=RHO**2-1.
      B(1)=-VARV*GP(1)**2
      B(2)=-VARV*GP(1)*GP(2)
      B(3)=-VARV*GP(1)*GP(3)
      B(4)=-VARV*GP(2)**2
      B(5)=-VARV*GP(2)*GP(3)
      B(6)=-VARV*GP(3)**2
      NN=6
      CALL DECOMP(NN,A,UL)
      CALL SOLVE(NN,UL,B,V)
```

```
      RETURN
      END
C*************************************
      SUBROUTINE VGAUS(PX,GP)
      DIMENSION PP(6),GP(3)
      COMMON/PHI/RHO,T(4)
      COMMON/SET2/SIGV,VARV,SIGW,VARW,SIGA,VARA
      P=1./(PP(1)+VARV)
      GP(1)=(PX(1)+PX(2)*T(1)+PW(3)*T(2)/2.)*P
      GP(2)=(PX(2)+PX(3)*T(1))*P
      GP(3)=PX(3)*RHO*P
      RETURN
      END
C*************************************
      SUBROUTINE PRI(PW,PWO)
C*****THIS SUBROUTINE CALCULATES THE ONE STEP PREDICTOR ERROR
C*****VARIANCES.  IT USES THE PRESENT ONE STEP PREDICTOR VARIANCES
C*****AND RETURNS WITH UPDATED ONE STEP VARIANCES(PW) AND THE
C*****OLD ONE STEP VARIANCES(PWO)
      DIMENSION PW(6),PWO(6)
      COMMON/PHI/RHO,T(4)
      COMMON/SET2/SIGV,VARV,SIGW,VARW,SIGA,VARA
      GAMMA=1.
      DO 10 I=1,6
      PWO(I)=PW(I)
10    CONTINUE
      P=(PWO(1)+VARV)
      A=PWO(1)+T(1)*PWO(2)+T(2)*PWO(3)/2.
      B=GAMMA*PWO(2)+T(1)*PWO(3)
      PW(1)=PWO(1)+2.*T(1)*PWO(2)+T(2)*(PWO(3)+PWO(4))+T(3)*PWO(5)
     & +T(4)*PWO(6)/4.-A**2/P
      PW(2)=GAMMA*PWO(2)+T(1)*PWO(3)+GAMMA*T(1)*PWO(4)+(1.+GAMMA/2.)
     & *T(2)*PWO(5)+T(3)*PWO(6)/2.-A*B/P
      PW(3)=RHO*(PWO(3)+T(1)*PWO(5)+T(2)*PWO(6)/2.-A*PWO(3)/P)
      PW(4)=GAMMA**2*PWO(4)+2.*GAMMA*T(1)*PWO(5)+T(2)*PWO(6)-B**2/P
      PW(5)=RHO*(GAMMA*PWO(5)+T(1)*PWO(6)-B*PWO(3)/P)
      PW(6)=(PWO(6)-PWO(3)**2/P)*RHO**2+VARV
      RETURN
      END
C*************************************
      SUBROUTINE SETUP(PX)
      DIMENSION PP(6)
      COMMON/PHI/RHO,T(4)
      COMMON/SET2/SIGV,VARV,SIGW,VARW,SIGA,VARA
      PX(1)=VARV
      PX(2)=VARV/T(1)
      PX(3)=0.
      PX(4)=VARA*T(2)/4.+2.*VARV/T(2)
      PX(5)=RHO*VARA*T(1)/2.
      PX(6)=VARV
      RETURN
      END
C*************************************
```

Appendix II
The Computer Program VSIM

```
C************** PROGRAM VSIM ***************
      OPTIONS DP
      INCLUDE 'ESSL,SYSY
      DIMENSION X(3),XHAT(3),GAINS(7,41,5),XOUT(4,41,5)
      REAL LAMBDA(5)
      COMMON/PHI/RHO,T(4)
      COMMON/SET2/SIGV,VARV,SIGW,VARW,SIGA,VARA
      DATA LAMBDA/1.,.5,.1,.05,.01/
      T(1)=.01
      T(2)=T(1)**2
      T(3)=T(2)*T(1)
      T(4)=T(2)**2
      CALL DEASSN
      CALL ASSIGN(5HGGGGG1,5H41033S,1)
      READ(1)GAINS
      CLOSE 1
      FACTOR=10.**(.1
      CALL XGEN01(X)
      S=1.
      AM=0.
      IX=72271
      DO 900 ILAM=1,5
      NUM=ILAM*20000
      RHO=1.0-LAMBDA(ILAM)*T(1)
      R=10.**-5/FACTOR
      DO 800 IPT=1,41
      R=R*FACTOR
C*****RUN THE SIMULATION
      CALL XGEN01(XHAT)
      SUM=0.0
      SMEAN=0.0
      CALL ASSIGN(4HSEED,5H41033S,1)
      READ(1)IX
      CLOSE 1
      DO 500 I=1,NUM
      CALL BKEEP(X,XHAT,SUM,SMEAN)
      CALL GAUSS(IX,S,AM,XNOI)
      XI=0.75*XNOI
      CALL PRED(GAINS(1,IPT,ILAM),XHAT,X,XI)
500   CONTINUE
      CALL ASSIGN(4HSEED,5H41033S,1)
      WRITE(1)IX
      CLOSE 1
      SUM=SUM/NUM
      SMEAN=SMEAN/NUM
      XOUT(1,IPT,ILAM)=R
      XOUT(2,IPT,ILAM)=GAINS(7,IPT,ILAM)
      XOUT(3,IPT,ILAM)=SQRT(SUM)/.05
      XOUT(4,IPT,ILAM)=SMEAN
      CALL ASSIGN(6HDATAA5,5H41033S,2)
      WRITE(2)XOUT
      CLOSE 2
      CONTINUE
      CONTINUE
      CALL EXIT
```

```
      END
C*****************************
      SUBROUTINE XCEAN(X)
      DIMENSION X(3)
      X(1)=1.
      X(2)=2.
      X(3)=3.
      RETURN
      END
C*****************************
      SUBROUTINE RANDU(IX,IY,YFL)
      IY=IX*65539
      IF(IY)5,6,6
    5 IY=IY+2147483647+1
    6 YFL=IY
      YFL=YFL*.4656613E-9
      RETURN
      END
C*****************************
      SUBROUTINE GAUSS(IX,S,AM,V)
      A=0.0
      DO 5 I=1,48
      CALL RANDU(IX,IY,Y)
      IX=IY
    5 A=A+Y
      V=(A/2.-12.)*S+AM
      RETURN
      END
C*****************************
      SUBROUTINE PRED(GP,XHAT,X,XI)
      DIMENSION GP(3),XHAT(3),X(3)
      COMMON/PRI/RHO,T(4)
      XINOV=X(1)-XHAT(1)+XI
      CALL SUPERR(2)
      XHAT(1)=XHAT(1)+T(1)*XHAT(2)+T(2)/2.*XHAT(3)+GP(1)*XINOV
      XHAT(2)=XHAT(2)+T(1)*XHAT(3)+GP(2)*XINOV
      XHAT(3)=(RHO*XHAT(3)+GP(3)*)INOV
      CALL SUPERR(-3)
      RETURN
      END
C*****************************
      SUBROUTINE ERROR(X,XHAT,SUM,SMEAN)
      DIMENSION X(3),XHAT(3)
      ERR=X(1)-XHAT(1)
      SUM=SUM+ERR**2
      SMEAN=SMEAN+ERR
      RETURN
      END
C*****************************
```

Appendix III
The Computer Program ZSIM

```
C************** PROGRAM ZSIM **************
      OPTIONS SP
      INCLUDE 'ESSL.SYS'
      DIMENSION G(7,41,5),X(4,41,5),GP(3)
      REAL LAM,A(5)
      COMMON /PI/RPO,T(4)
      DATA LAM/1.,.5,.1,.05,.01/
      T(1)=.5
      T(2)=T(1)**2
      T(3)=T(1)*T(2)
      T(4)=T(2)**2
      CALL ASSIGN(5HOUTPUT,5H41035,1)
      READ(1) G
      CLOSE 1
      FACT=10.**(-1
      DO 5 ILAM=1,5
      A(ILAM)=1.-LAM(ILAM)*T(1)
      R=1.**-5./FACTOR
      CALL SQRT(A(3)
      CALL SQRT(A(5)
      CALL SQRT(A(4)
      DO 40 IPT=1,41
      R=R**FACTOR
      GP(1)=G(1,IPT,ILAM)
      GP(2)=G(2,IPT,ILAM)
      GP(3)=G(3,IPT,ILAM)
      CALL ZBE3(GP,AMS)
      X(1,IPT,ILAM)=R
      X(2,IPT,ILAM)=G(7,IPT,ILAM)
      X(3,IPT,ILAM)=SQRT(AMS)
      X(4,IPT,ILAM)=0.
40    CONTINUE
50    CONTINUE
      CALL ASSIGN(6HDATAC1,5H41035,1)
      WRITE(1)X
      CLOSE 1
      CALL EXIT
      END
C*************************
      SUBROUTINE ZBE3(GP,AMS)
      DIMENSION GP(3)
      REAL NUM1,NUM2,NUM3,NUM
      COMMON /PI/RPO,T(4)
      A=-GP(1)*(RPO+1.)+GP(2)*T(1)+GP(3)*T(2)/2.
      B=GP(1)*RPO-GP(2)*T(1)*RPO+GP(3)*T(2)/2.
      C=GP(1)-2.-RPO
      D=A+1.+RPO
      E=1.-RPO
      NUM=(D-C*E)*(1.+D)*2.*D+E*(C+E)*2.*(1.-E**2)
     *+1.*(D-C*E)*2.*C-1.*(1.+D))*2.*(1.-E**2)
     *-E*(D-C*E)*2.*D-(D-C*E)*(C+E)*2.*C
      DEN=B*GP(1)*(1.+D)*2.*D+E*(C+E)*(GP(1)**2+A**2+B**2)
     *+1.*(A*B*GP(1)+A*B)*2.*C-1.*(1.+D)*(GP(1)**2+A**2+B**2)
     *-E*(A*GP(1)+A*B)*2.*D-E*GP(1)*(C+E)*2.*C
      AMS=2.*NUM/DEN
      RETURN
      END
```

Appendix IV
The Computer Program FLYBY

```
C*************** PROGRAM FLYBY ***************
      OPTIONS DP
      DIMENSION Z(3),ZBIG(3),GP(3),XOUT(7,41,5)
      DIMENSION XGAIN(3,41,5)
      REAL LAMBDA(5)
      COMMON/PHI/RHO,T(4)
      COMMON/SETS/ACC,VEL,THED,RAD,PI
      DATA LAMBDA/.5,.25,.05,0.025,0.005/
      CALL ASSIGN(5HMESSG,5H4103S,6)
      CALL DEASSN
      PI=3.14159265358979
      CALL ASSIGN(5HGGGW3,5H4103S,1)
      READ(1)XOUT
      CLOSE 1
      DO 10 I=1,3
      DO 10 J=1,41
      DO 10 K=1,5
10    XGAIN(I,J,K)=XOUT(I,J,K)
      DO 20 I=1,7
      DO 20 J=1,41
      DO 20 K=1,5
20    XOUT(I,J,K)=0.
      T(1)=.02
      T(2)=T(1)**2
      T(3)=T(1)**3
      T(4)=T(1)**4
      FACTOR=10.**6.1
      DO 40 ILAM=1,5
      RHO=1.-LAMBDA(ILAM)*T(1)
      VEL=30.
      ACC=2.*15.
      DO 900 ILINE=1,7
      IEXP=ILINE-1
      ACC=ACC/2.
      THED=ACC/PI
      RAD=VEL**2/ACC
      A=10.**-5/FACTOR
      DO 800 IPT=1,41
      CALL ASSIGN(4HPROG,5H4103S,1)
      READ(1,ILINE,IPT,ILAM
      CLOSE 1
      A=A*FACTOR
      CALL INIT2(ZBIG,ERMAX,ICT)
      CALL GAINS(XGAIN(1,IPT,ILAM),GP)
      DO 750 I=1,10000
      CALL ...822(Z,I,ICT)
      IF ICT.GE.1.0)GO TO 760
      CALL ...(Z(1),ZBIG(1),ERR,ERMAX)
      CALL ...(Z(1),ZBIG,GP)
...
      ...,ILINE,IPT,ILAM)=ERMAX/RAD
...
      CONTINUE
      CALL ASSIGN('FLYBY',5H4103S,1)
      WRITE(1)XOUT
```

PRACTICABLE

```
      CLOSE 1
      CALL EXIT
      END
C*****************************
C*****SUBROUTINE ERROR1 COMPUTES THE MAXIMUM ERROR (OVERSHOOT)
      SUBROUTINE ERROR1(Z,ZBIG,ERR,ERMAX)
      ERR=Z-ZBIG
      ABERR=ABS(ERR)
      IF(ERMAX.LT.ABERR) ERMAX=ABERR
      RETURN
      END
C*****************************
C*****SUBROUTINE PRED COMPUTES THE ONE STEP KALMAN PREDICTION
C*****USING EQUATION 3.  HERE THE VARIABLE X  IS THE
C*****INPUT AND INNOV IS THE INNOVATION.
      SUBROUTINE PRED(X,ZBIG,GP)
      DIMENSION ZBIG(3),GP(3)
      REAL INNOV
      COMMON/BBI/BBO,T(4)
      INNOV=X-ZBIG(1)
      ZBIG(1)=ZBIG(1)+ZBIG(2)*T(1)+ZBIG(3)*T(2)/2.+GP(1)*INNOV
      ZBIG(2)=ZBIG(2)+ZBIG(3)*T(1)+GP(2)*INNOV
      ZBIG(3)=BBO*ZBIG(3)+GP(3)*INNOV
      RETURN
      END
C*****************************
C*****SUBROUTINE GAINS TRANSFERS THE GAINS FROM THE ARRAY
C*****XGAIN TO THE VECTOR GP
      SUBROUTINE GAINS(XGAIN,GP)
      DIMENSION XGAIN(3),GP(3)
      GP(1)=XGAIN(1)
      GP(2)=XGAIN(2)
      GP(3)=XGAIN(3)
      RETURN
      END
C*****************************
      SUBROUTINE TURN1(Z,J,IXXX)
      DIMENSION Z(3)
      COMMON/BBI/BBO,T(4)
      COMMON/SET3/ACC,VEL,THED,RAD,PI
      IF(IXXX.GE.1)GO TO 20
      THETA=+THED*T(1)*(J-1)
      IF(THETA.GT.PI)GO TO 10
      Z(1)=-RAD*SIN(THETA)
      Z(2)=-RAD*THED*COS(THETA)
      Z(3)=RAD*THED**2*SIN(THETA)
      RETURN
C*****************************
C*****THIS SECTION COMPUTES THE X VECTOR DURING THE
C*****TRANSITION FROM MANEUVERING FLIGHT TO STRAIGHT FLIGHT.
   10 IXXX=1
      ALPHA=THETA-PI
      TIME=ALPHA/THED
      Z(3)=0.0
      Z(2)=+VEL
```

```
      Z(1)=Z(1)*PI
      RETURN
C*****************************
C*****THIS SUBROUTINE COMPUTES THE X VECTOR DURING THE
C*****FLYOUT PERIOD AFTER THE MANEUVER.
20    IXXX=IXXX+1
      Z(1)=Z(1)+Z(2)*T(1)
      RETURN
      END
C*****************************
      SUBROUTINE INIT1(ZZIG,ERMAX,ICT)
      DIMENSION ZZIG(3)
      COMMON/SET3/ACC,VEL,THED,RAD,PI
      ZZIG(1)=0.
      ZZIG(2)=-VEL
      ZZIG(3)=0.
      ERMAX=0.
      ICT=1
      RETURN
      END
C*****************************
      SUBROUTINE INIT2(ZZIG,ERMAX,ICT)
      DIMENSION ZZIG(3)
      COMMON/SET3/ACC,VEL,THED,RAD,PI
      ZZIG(1)=-RAD
      ZZIG(2)=0.
      ZZIG(3)=0.
      ERMAX=0.
      ICT=1
      RETURN
      END
C*****************************
      SUBROUTINE TURN2(Z,J,IXXX)
      DIMENSION Z(3)
      COMMON/SET1/REC,T(4)
      COMMON/SET3/ACC,VEL,THED,RAD,PI
      IF(IXXX.GE.1)GO TO 20
      THETA=+THED*T(1)*(J-1)
      IF(THETA.GT.PI)GO TO 10
      Z(1)=-RAD*COS(THETA)
      Z(2)=+RAD*THED*SIN(THETA)
      Z(3)=+RAD*THED**2*(COS(THETA))
      RETURN
C*****************************
C*****THIS SECTION COMPUTES THE X VECTOR DURING THE
C*****TRANSITION FROM MANEUVERING FLIGHT TO STRAIGHT FLIGHT.
10    IXXX=1
      ALPHA=THETA-PI
      TIME=ALPHA/THED
      Z(3)=0.0
      Z(2)=0.
      Z(1)=RAD
      RETURN
C*****************************
C*****THIS SECTION COMPUTES THE X VECTOR DURING THE
```

```
C*****FLYOUT PERIOD AFTER THE MANEUVER.
   20   IXXX=IXXX+1
        Z(1)=+R/D
        RETURN
        END
C**********************************
        SUBROUTINE TURN3(Z,J,ICT)
        DIMENSION Z(3)
        COMMON/PI/ZERO,T(4)
        COMMON/SET3/ACC,VEL,THED,RAD,PI
        IF(ICT.GT.1)GO TO 10
        THETA=THED*T(1)*(J-1)
        IF(THETA.GT.PI) GO TO 10
        Z(3)=+RAD*THED**2*(COS(THETA)-1.)
        Z(2)=Z(2)+Z(3)*T(1)
        Z(1)=Z(1)+Z(2)*T(2)+Z(3)*T(2)/2.
        IF(J.EQ.1)Z(1)=-RAD
        IF(J.EQ.1)Z(2)=0.
        RETURN
   10   ICT=ICT+1
        Z(1)=Z(1)
        Z(2)=Z(2)
        Z(3)=0.
        RETURN
        END
```

Appendix V
The Computer Program STEADY

```
C*************** PROGRAM STEADY ***************
      OPTIONS DP
      DIMENSION Z(3),ZWIG(3),GP(3) XOUT(4,41,5),XGAIN(7,41,5)
      REAL LAMBDA(5)
      COMMON/PHI/RHO,T(4)
      DATA LAMBDA/1.,.5,.1,.05,.01/
      CALL DEASSN
      CALL ASSIGN(5HGO001,5H41M3S,1)
      READ(1)XGAIN
      CLOSE 1
      T(1)=.01
      T(2)=T(1)**2
      T(3)=T(1)**3
      T(4)=T(1)**4
      FACTOR=10.**(.1
      DO 950 ICNTRL=1,3
      DO 900 ILAM=1,5
      RHO=1.-LAMBDA(ILAM)*T(1)
      R=1.**-5/FACTOR
      DO 800 IPT=1,41
      R=R*FACTOR
      CALL GAINS(XGAIN(1,IPT,ILAM),GP)
      CALL ZERO(Z,ZWIG,ERMAX)
      DO 700 I=1,50000
      GO TO (10,20,30),ICNTRL
   10 CALL STEP(Z)
      CALL ERROR2(Z(1),ZWIG(1),ERR,ERMAX)
      GO TO 50
   20 CALL RAMP(Z,T(1))
      GO TO 40
   30 CALL PARAB(Z,T(1))
   40 CALL ERROR1(Z(1),ZWIG(1),ERR,ERMAX)
   50 CALL STOP(ERR,ERMAX,ISTOP)
      CALL PRED(Z(1),ZWIG,GP)
      IF(ISTOP.EQ.1)GO TO 100
  700 CONTINUE
  100 XOUT(1,IPT,ILAM)=R
      GO TO (60,70,80),ICNTRL
   60 XOUT(2,IPT,ILAM)=ERMAX
      GO TO 90
   70 XOUT(3,IPT,ILAM)=ERMAX/T(1)
      GO TO 90
   80 XOUT(4,IPT,ILAM)=ERMAX/T(2)
   90 CALL ASSIGN(6HSRPOUT,5H41M3S,1)
      WRITE(1) XOUT
      CLOSE 1
  800 CONTINUE
  900 CONTINUE
  950 CONTINUE
      CALL EXIT
      END
C*************************************
C***SUBROUTINE STEP GENERATES A (UNIT) STEP.
      SUBROUTINE STEP(Z)
      DIMENSION Z(3)
```

```
      Z(1)=1.
      Z(2)=0.
      Z(3)=0.
      RETURN
      END
C*****************************
C*****SUBROUTINE RAMP GENERATES AN INPUT CORRESPONDING TO A
C*****(UNIT) STEP IN VELOCITY.  THIS IS A (UNIT)RAMP IN
C*****RANGE.
      SUBROUTINE RAMP(Z,T)
      DIMENSION Z(3)
      Z(1)=Z(1)+Z(2)*T
      Z(2)=1.0
      Z(3)=0.
      RETURN
      END
C*****************************
C*****SUBROUTINE PARAB GENERATES AN INPUT CORRESPONDING TO A
C*****(UNIT) STEP IN ACCELERATION.  THIS IS A (UNIT) PARABOLA
C*****IN RANGE.
      SUBROUTINE PARAB(Z,T)
      DIMENSION Z(3)
      Z(1)=Z(1)+Z(2)*T+Z(3)*T**2
      Z(2)=Z(2)+Z(3)*T
      Z(3)=1.0
      RETURN
      END
C*****************************
C*****SUBROUTINE ZERO INITIALIZES PARAMETERS PRIOR TO ITERATING
C*****THE KALMAN ALGORITHM BY SETTING THE STATE VECTOR(Z)
C*****AND THE PREDICTOR VECTOR(ZWIG) TO ZERO.  IT ALSO ZEROS
C*****OTHER MISC. PARAMETERS NECESSARY TO THE COMPUTATION.
      SUBROUTINE ZERO(Z,ZWIG,ERMAX)
      DIMENSION Z(3),ZWIG(3)
      DO 10 I=1,3
      Z(I)=0.
      ZWIG(I)=0.
10    CONTINUE
      ERMAX=0.
      RETURN
      END
C*****************************
C*****SUBROUTINE ERROR1 COMPUTES THE MAXIMUM ERROR (OVERSHOOT)
C*****FOR RAMPS AND PARABOLA INPUTS
      SUBROUTINE ERROR1(Z,ZWIG,ERR,ERMAX)
      ERR=Z-ZWIG
      ABERR=ALS(ERR)
      IF(ERMAX.LT.ABERR) ERMAX=ABERR
      RETURN
      END
C*****************************
C*****SUBROUTINE ERROR2 COMPUTES THE ABSOLUTE VALUE OF THE
C*****HEIGHT OF THE FIRST OVERSHOOT IN RESPONSE TO STEP
C*****CHANGES IN RANGE.
      SUBROUTINE ERROR2(Z,ZWIG,ERR,ERMAX)
```

```
      ERR=Z-ZHAT(1)
      IF(ERR.LT.0.)GO TO 1
      RETURN
1     ABSERR=ABS(ERR)
      IF(ABSERR.LT.ERMAX)ERMAX=ABSERR
      RETURN
      END
C*****************************
C*****SUBROUTINE KALPRD COMPUTES THE ONE STEP KALMAN PREDICTION
C*****EQUATIONS.  HERE THE VARIABLE X  IS THE
C*****KALMAN INNOV IS THE INNOVATION.
      SUBROUTINE KALPRD(X,ZHAT,GP)
      DIMENSION ZHAT(3),GP(3)
      COMMON INNOV
      DATA T/.1,1.,1./
      INNOV=X-ZHAT(1)
      ZHAT(1)=ZHAT(1)+ZHAT(2)*T(1)+ZHAT(3)*T(2)/2.+GP(1)*INNOV
      ZHAT(2)=ZHAT(2)+ZHAT(3)*T(1)+GP(2)*INNOV
      ZHAT(3)=ZHAT(3)+GP(3)*INNOV
      RETURN
      END
C*****************************
C*****SUBROUTINE GAINS TRANSFERS THE GAINS FROM THE ARRAY
C*****XGAIN TO THE VECTOR GP
      SUBROUTINE GAINS(XGAIN,GP)
      DIMENSION XGAIN(3),GP(3)
      GP(1)=XGAIN(1)
      GP(2)=XGAIN(2)
      GP(3)=XGAIN(3)
      RETURN
      END
C*****************************
C*****SUBROUTINE STOP CHECKS TO SEE IF THE MAXIMUM ERROR HAS
C*****BEEN ACHIEVED.  IF SO, THEN A FLAG, ISTOP, IS SET.
      SUBROUTINE STOP(ERR,ERMAX,ISTOP)
      ISTOP=0
      ABSERR=ABS(ERR)
      IF(ABSERR.LT.ERMAX)ISTOP=1
      RETURN
      END
C*****************************
```

Appendix VI
The Computer Program TRANS

```
C************** PROGRAM TRANS ***************
      OPTIONS DP
      INCLUDE 'MESSP,SYS9'
      DIMENSION Z(5),ZWIG(5),GP(5),PWIG(6),PWIGO(6)
      DIMENSION XOUT(4,41,5),YOUT(4,41,5)
      REAL LAMBDA(5)
      COMMON/PHI/RHO,T(4)
      COMMON/SET2/SIGV,VARV,SIGR,VARR,SIGA,VARA
      DATA LAMEDA/1.,.5,.1,.05,.01/
      CALL ASSIGN(6HTRN007,5H4103S,1)
      CALL ASSIGN(6HTOT007,5H41M2S,2)
      CALL DEASSN
      CALL SUPERR(3)
      T(1)=.01
      T(2)=T(1)**2
      T(3)=T(1)**3
      T(4)=T(1)**4
      SIGA=2.
      VARA=SIGA**2
      FACTOR=10.**.1
      DO 950 ICNTRL=1,2
      DO 900 ILAM=1,5
      RHO=1.-LAMBDA(ILAM)*T(1)
      R=R*FACTOR
      R=10.**-5/FACTOR
      DO 800 IPT=1,41
      R=R*FACTOR
      SIGV=R*SIGA/LAMBDA(ILAM)**2
      VARV=SIGV**2
      VARR=(1.-RHO**2)*VARA
      SIGR=SQRT(VARR)
      CALL ZERO(Z,ZWIG,ERMAX,K,ISTOP)
      CALL SETUP2(PWIG)
      DO 700 I=1,50000
      CALL GPCALC(PWIG,GP)
      GO TO (10,20,30),ICNTRL
10    CALL STEP(Z)
      CALL ERROR2(Z(1),ZWIG(1),ERR,ERMAX)
      GO TO 50
20    CALL RAMP(Z,T(1))
      CALL ERROR1(Z(1),ZWIG(1),ERR,ERMAX)
      GO TO 50
30    CALL PARAB(Z,T(1))
40    CALL ERROR1(Z(1),ZWIG(1),ERR,ERMAX)
50    CALL COUNT(I,K,ERR)
      CALL STOP2(Z,ZWIG,ISTOP)
      CALL PRED(Z(1),ZWIG,GP)
      CALL PWI(PWIG,PWIGO)
      IF(ISTOP.EQ.1)GO TO 100
700   CONTINUE
100   XOUT(1,IPT,ILAM)=R
      YOUT(1,IPT,ILAM)=R
      GO TO (60,70,80),ICNTRL
60    XOUT(2,IPT,ILAM)=ERMAX/.20
      YOUT(2,IPT,ILAM)=K
```

```
      GO TO 880
  770 XOUT(3,IPT,ILAM)=ERMAX/.28
      YOUT(3,IPT,ILAM)=K
      GO TO 880
  800 XOUT(4,IPT,ILAM)=ERMAX/(.5*T(2))
      YOUT(4,IPT,ILAM)=K
  880 CONTINUE
      WRITE(1)XOUT
      WRITE(2)YOUT
      CLOSE 1
      CLOSE 2
  910 CONTINUE
  950 CONTINUE
      CALL EXIT
      END
C*****************************
C*****SUBROUTINE STEP GENERATES A (UNIT) STEP.
      SUBROUTINE STEP(Z)
      DIMENSION Z(3)
      Z(1)=.2
      Z(2)=0.
      Z(3)=0.
      RETURN
      END
C*****************************
C*****SUBROUTINE RAMP GENERATES AN INPUT CORRESPONDING TO A
C*****(UNIT) STEP IN VELOCITY.  THIS IS A (UNIT)RAMP IN
C*****RANGE.
      SUBROUTINE RAMP(Z,T)
      DIMENSION Z(3)
      Z(1)=Z(1)+Z(2)*T
      Z(2)=T.
      Z(3)=0.
      RETURN
      END
C*****************************
C*****SUBROUTINE PARAB GENERATES AN INPUT CORRESPONDING TO A
C*****(UNIT) STEP IN ACCELERATION.  THIS IS A (UNIT) PARABOLA
C*****IN RANGE.
      SUBROUTINE PARAB(Z,T)
      DIMENSION Z(3)
      Z(1)=Z(1) Z(2)*T+Z(3)*T**2
      Z(2)=Z(2) Z(3)*T
      Z(3)=1.0
      RETURN
      END
C*****************************
C*****SUBROUTINE ZERO INITIALIZES PARAMETERS PRIOR TO ITERATING
C*****THE KALMAN ALGORITHM BY SETTING THE STATE VECTOR(Z)
C*****AND THE PREDICTOR VECTOR(ZPIG) TO ZERO.  IT ALSO ZEROS
C*****OTHER MISC. PARAMETERS NECESSARY TO THE COMPUTATION.
      SUBROUTINE ZERO(Z,ZPIG,ERMAX,K,ISTOP)
      DIMENSION Z(3),ZPIG(3)
      DO 10 I=1,3
      Z(I)=0.
```

```
      ZWIG(I)=0.
   10 CONTINUE
      ERMAX=0.
      K=0
      ISTOP=1
      RETURN
      END
C*****************************
C*****SUBROUTINE ERROR1 COMPUTES THE MAXIMUM ERROR (OVERSHOOT)
C*****FOR RAMPS AND PARABOLA INPUTS
      SUBROUTINE ERROR1(Z,ZWIG,ERR,ERMAX)
      ERR=Z-ZWIG
      ABERR=ALS(ERR)
      IF(ERMAX.LT.ABERR) ERMAX=ABERR
      RETURN
      END
C*****************************
C*****SUBROUTINE ERROR2 COMPUTES THE ABSOLUTE VALUE OF THE
C*****HEIGHT OF THE FIRST OVERSHOOT IN RESPONSE TO STEP
C*****CHANGES IN RANGE.
      SUBROUTINE ERROR2(Z,ZWIG,ERR,ERMAX)
      ERR=Z-ZWIG
      IF(ERR.LT.0.)GO TO 10
      RETURN
   10 ABERR=ALS(ERR)
      IF(ERMAX.LT.ABERR)ERMAX=ABERR
      RETURN
      END
C*****************************
C*****SUBROUTINE PRED COMPUTES THE ONE STEP KALMAN PREDICTION
C*****USING EQUATION 3.  HERE THE VARIABLE X  IS THE
C*****INPUT AND INNOV IS THE INNOVATION.
      SUBROUTINE PRED(X,ZWIG,GP)
      DIMENSION ZWIG(3),GP(3)
      REAL INNOV
      COMMON/PPI/RHO,T(4)
      INNOV=X-ZWIG(1)
      ZWIG(1)=ZWIG(1)+ZWIG(2)*T(1)+ZWIG(3)*T(2)/2.+GP(1)*INNOV
      ZWIG(2)=ZWIG(2)+ZWIG(3)*T(1)+GP(2)*INNOV
      ZWIG(3)=RHO*ZWIG(3)+GP(3)*INNOV
      RETURN
      END
C*****************************
C*****SUBROUTINE GAINS TRANSFERS THE GAINS FROM THE ARRAY
C*****XGAIN TO THE VECTOR GP
      SUBROUTINE GAINS(XGAIN,GP)
      DIMENSION XGAIN(3),GP(3)
      GP(1)=XGAIN(1)
      GP(2)=XGAIN(2)
      GP(3)=XGAIN(3)
      RETURN
      END
C*****************************
C*****SUBROUTINE STOP CHECKS TO SEE IF THE MAXIMUM ERROR HAS
C*****BEEN ACHIEVED.  IF SO, THEN A FLAG, ISTOP, IS SET.
```

```
      SUBROUTINE ST P(...,PMAX,ISTOP)
      ISTOP=0
      A...=ABS(P...)
      IF(A...GT..LT.PMAX)ISTOP=1
      RETURN
      END
C*****************************
C*****SUBROUTINE ST P2 IS USED FOR CHECKING TO SEE IF THE FILTER
C*****HAS CONVERGED TO THE CORRECT VALUE IN LOCK UP SIMULATIONS
      SUBROUTINE STOP2(Z,ZMIN,ISTOP)
      DIMENSION Z(3),ZMIN(3),EMA(3)
      DO 10 I=1,3
      EMA(I)=ABS(Z(I)-ZMIN(I))
      IF(EMA(I).GT..0001)GO TO 20
10    CONTINUE
      ISTOP=1
      RETURN
20    CONTINUE
      END
C*****************************
C*****SUBROUTINE COUNT IS USED TO COUNT THE ITERATIONS TO CONVERGE
      SUBROUTINE COUNT(I,K,EMA)
      IF(EMA(I).GT..05)K=I
      RETURN
      END
C*****************************
C*****************************
      SUBROUTINE P11(P1,P1O)
C*****THIS SUBROUTINE CALCULATES THE ONE STEP PREDICTOR ERROR
C*****VARIANCES.  IF J6=3 THE PRESENT ONE STEP PREDICTOR VARIANCES
C*****ARE RETURNS WITH UPDATED ONE STEP VARIANCES(P1O) AND IF
C*****OLD ONE STEP VARIANCES(P1O)
      DIMENSION P1(6),P1O(6)
      COMMON/PHI/RHO,T(4)
      COMMON/SET2/SIGV,VARV,SIGW,VARW,SIGA,VARA
      GAMMA=1.
      DO 10 I=1,6
      P1O(I)=P1(I)
10    CONTINUE
      P=(P1O(1)+VARV)
      A=P1O(1)+T(1)*P1O(2)+T(2)*P1O(3)/2.
      B=GAMMA*P1O(2)+T(1)*P1O(3)
      P1(1)=P1O(1)+2.*T(1)*P1O(2)+T(2)*(P1O(3)+P1O(4))+T(3)*P1O(5)
     .  +.(4)*P1O(6)/4.-A**2/P
      P1(2)=GAMMA*P1O(2)+T(1)*P1O(3)+GAMMA*T(1)*P1O(4)+(1.+GAMMA/2.)
     . *T(2)*P1O(5)+T(3)*P1O(6)/2.-A*B/P
      P1(3)=RHO*(P1O(3)+T(1)*P1O(5)+T(2)*P1O(6)/2.-A*P1O(3)/P)
      P1(4)=GAMMA**2*P1O(4)+2.*GAMMA*T(1)*P1O(5)+T(2)*P1O(6)-B**2/P
      P1(5)=RHO*(GAMMA*P1O(5)+T(1)*P1O(6)-B*P1O(3)/P)
      P1(6)=(P1O(6)-P1O(3)**2/P)*RHO**2+VARW
      RETURN
      END
C*****************************
      SUBROUTINE SETUP2(PN)
      DIMENSION PN(6)
      COMMON/PHI/RHO,T(4)
```

```
      COMMON/SET2/SIGV,VARV,SIGW,VARW,SIGA,VARA
      PW(1)=5.*VARV+(1.+RHO)*VARA*T(4)/2.
      PW(2)=3.*(VARV/T(1)+(1.+RHO)*VARA*T(3)/4.)
      PW(3)=(1.+RHO)*RHO*VARA*T(2)/2.
      PW(4)=2.*VARV/T(2)+(1.2+RHO)*VARA*T(2)
      PW(5)=(1.+RHO/2.)*VARA*T(1)
      PW(6)=RHO**2*VARA
      RETURN
      END
C*************************
C*************************************
      SUBROUTINE GPCALC(PW,GP)
      DIMENSION PW(6),GP(3)
      COMMON/PAI/RPO,T(4)
      COMMON/SET2/SIGV,VARV,SIGW,VARW,SIGA,VARA
      P=1./(PW(1)+VARV)
      GP(1)=(PW(1)+PW(2)*T(1)+PW(3)*T(2)/2.)*P
      GP(2)=(PW(2)+PW(3)*T(1))*P
      GP(3)=PW(3)*RHO*P
      RETURN
      END
```

REFERENCES

1. Sage, A. P., and Melsa, J. L., Estimation Theory with Application to Communications and Control, McGraw-Hill, New York, 1971.

2. Nahi, N. E., Estimation Theory and Applications, John Wiley and Sons, New York, 1969.

3. Kwakernaak, H., and Sivan, r., Linear Optimal Control Systems, Wiley-Interscie ıce, New York, 1972.

4. Kalman, R. E. "A New Approach to Linear Filtering and Prediction Problems," ASME J. Basic Eng., vol. 82D, pp.34-45, March, 1960.

5. Singer, R. A., and Behnke, K. W., "Real-Time Tracking Filter Evaluation and Selection for Tactical Applications. I.E.E.E. Transactions on Aerospace and Electronic Systems, vol. AES-7, no. 1, pp. 100-110, January 1971.

6. Swarner, W. E., Chuang, C., and Eilts, H. S., "TDMA Timing Loops for High Data Rate Systems," Report 710300-1, May 1978, The Ohio State University ElectroScience Laboratory, Department of Electrical Engineering; prepared under Contract F30602-75-C-0061 for Rome Air Development Center, Griffiss Air Force Base, New York.

7. Huff, R. J., "An Investigation of Time Division Multiple Access Space Communication Systems," Ph.D. Dissertation, The Ohio State University, 1969.

8. Forseyth, G., and Moler, C. B., Computer Solution of Linear Algebraic Systems, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1967.

9. Dahlquist, G., Bjorck, A., Numerical Methods, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1974.

10. Wilkinson, J. H., The Algebraic Eigenvalue Problem, Clarenden Press, Oxford, 1965.

11. Tou, J., Digital and Sampled-Data Control Systems, McGraw-Hill Book Company, New York, 1959.

12. Cazdow, J. A., Discrete Time Systems, Prentice Hall, Inc., 1973.

13. Abramowitz, M. and Stegun, I. A., Handbook of Mathematical Functions, Dover Publications, Inc., New York, New York, 1964.

14. Symon, K. R., Mechanics, Addison-Wesley Publishing Co., Reading, Massachusetts, 1971.

15. D'Azzo, J. J., and Houpis, C. H., Feedback Control Systems Analysis and Synthesis, Second Edition, McGraw-Hill Book Co., New York, 1964.

16. Gill, W. J., "A Comparison of Binary Delay-Lock Tracking Loop Implementations," IEEE Trans. Aerospace and Electronic Systems, Vol. AES-2, July 1966, pp. 415-424.

17. Spilker, J. J., Jr., and Magill, D. T., "The Delay-Lock Discriminator as Optimum Tracking Device," Proc. IRE, Vol. 59, september 1961, pp. 1403-1416.

18. Spilker, J. J., Jr., "Delay-Lock Tracking of Binary Signals," IRE Trans. Space Electronics and Telemetry, Vol. SET-9, March 1963, pp. 1-8.

19. Huff, R. J., Reinhard, K. L., and Upp, D. C., "The Synchronization of Time Division Multiple Access Systems - An Analytical and Experimental Study," Report 2358-9, 31 January 1969, The Ohio State University ElectroScience Laboratory, Department of Electrical Engineering; prepared under Contract F30602-67-C-0119 for Rome Air Development Center, Griffiss Air Force Base, New York (AD 689 223)

20. Wallace, K. A., "The Tracking Performance of Sampled-Data Delay Lock Loops with Pulsed Envelope Input Signals," Report 2358-7, 26 June 1968, The Ohio State University ElectroScience Laboratory, Department of Electrical Engineering; prepared under Contract F30602-67-C-0119 for Rome Air Development Center, Griffiss Air Force Base, New York (AD 821 657).

21. Reinhard, K. L., "Analysis of a Pseudo-Random Network Timing System for Time-Division Multiple Access Communications," Report 2358-2, 2 June 1967, The Ohio State University ElectroScience Laboratory, Department of Electrical Engineering; prepared under Contract F30602-67-C-0119 for Rome Air Development Center, Griffiss Air Force Base, New York (AD 820 3006)

22. Huff, R. J., "The Imperfectly Timed Demodulation of Differential Phase Shift Keyed Signals," Report 2738-1, June 1969, The Ohio State University ElectroScience Laboratory, Department of Electrical Engineering; prepared under Contract F30602-67-C-0112 for Rome Air Development Center, Griffiss Air Force Base, New York.

AD-A081 822 OHIO STATE UNIV COLUMBUS ELECTROSCIENCE LAB F/G 9/5
THE PERFORMANCE OF A SAMPLED DATA DELAY LOCK LOOP IMPLEMENTED W--ETC(U)
JAN 80 H S EILTS F30602-79-C-0068
UNCLASSIFIED ESL-711679-1 RADC -TR-79-333 NL

2 OF 2
AD A081 822

END
DATE FILMED
4-80
DTIC

23. Huff, R. J., "TDMA Space Communication Systems: Concepts and Practical Techniques," Bulletin 206, The Ohio State University Engineering Experiment Station, Columbus, Ohio.

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.