MEASURE, CRITERIA AND PROCEDURE FOR TRACK AND SEARCH ALLOCATION

Robert J. Polge, et al

Alabama University

Prepared for:

Army Missile Command

February 1973

UAH Research Report No. 138 February 1973

MEASURE, CRITERIA AND PROCEDURE FOR TRACK AND SEARCH ALLOCATION

by

R. J. Polge
E. R. McKee
B. K. Bhagavan
R. D. Hays

Final Technical Report

This research work was supported by
the U. S. Army Missile Command under
Contract DAAH01-72-C-0585

The University of Alabama in Huntsville
Huntsville, Alabama

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA – R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*
The University of Alabama in Huntsville
P. O. Box 1247
HuntsvilleAL 35807

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP

3. REPORT TITLE

MEASURE, CRITERIA AND PROCEDURE FOR TRACK AND SEARCH ALLOCATION

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
Final Technical Report, February, 1972 - February 1973

5. AUTHOR(S) *(First name, middle initial, last name)*
Robert J. Polge, Edward R. McKee, Jr., B. K. Bhagavan, Roy D. Hays -

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
February 1973	~~127~~ 133	11

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
DAAH01-72-C-0585	UAH Research Report No. 138
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
d.	

10. DISTRIBUTION STATEMENT

Approved for public release - distribution unlimited

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	U. S. Army Missile Command Redstone Arsenal AL 35809

13. ABSTRACT

A general procedure for radar resource allocation is presented. It requires the computation of the covariance matrix of the position estimates. Therefore, four estimation algorithms commonly used for track are evaluated and compared. A tracking measure applicable to the non-tactical EAR system is defined. Finally, a schema for track and search allocation is presented. An example illustrates how the tracking measure is computed and used in the track and search algorithm.

(Experimental array radar)

DD FORM 1 NOV 65 1473

UNCLASSIFIED

Security Classification

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

This report is a documentation of work performed under contract DAAH01-72-C-0585. The main objectives of this effort were: (1) to develop a measure of tracking quality, (2) to establish criteria for allocating resources to track and search, (3) to evaluate the quality of the tracking measure as a function of computation time, and (4) to develop a track and search allocation procedure for the EAR (Experimental Array Radar), a computer controlled phased array radar air defense system being developed by the U.S. Army Missile Command.

The work was performed in three phases. First, the general problem of resource allocation was analyzed, and an allocation procedure was developed. Then, various estimation algorithms were implemented and evaluated. Finally, a track and search allocation scheme was proposed for the EAR system.

In Chapter 2, a methodology for dynamic resource allocation is developed which is based on a measure of total defense posture. It effectively combines the pertinent threat and system parameters into a single numerical index upon which the allocation decisions are based. This procedure requires the knowledge of the rms error of the smoothed estimate and of the predicted estimates(with and without allocation) for each target.

The estimation algorithm is an important part of the allocation procedure. Therefore, several estimation algorithms are investigated. In Chapter 3, a system model is developed to represent a maneuvering target. Based on this model, several estimation algorithms are discussed in Chapter 4. They include the g-h filter (or

α - β filter), two types of fixed memory polynomial filters and the Kalman filter.

These four filters are evaluated in Chapter 5 through computer simulations using two typical trajectories. The evaluation takes into account the mean-squared-error performance, computation time and storage requirement.

In Chapter 6, a measure of tracking quality is defined and a procedure for track and search allocation for the EAR system is proposed. An example shows how the tracking measure is computed from the error covariance matrices of the Kalman filter.

Chapter 7 contains the summary and conclusions. The procedure for generating trajectories, with and without maneuvering, is described in Appendix A. Recursive relations for the computation of the error covariance matrices of the g-h filter are derived in Appendix B. Appendix C contains FORTRAN listings of all subprograms along with brief descriptions.

CHAPTER 2

PHASED ARRAY RADAR RESOURCE ALLOCATION FOR AIR DEFENSE

2.1 Introduction

A modern aircraft or ballistic missile defense system employs a multiple function phased array radar, interceptors and a high-speed digital computer to perform: (1) search of a specified volume, (2) detection of objects, (3) track initiation, (4) tracking, and (5) interceptor planning and firing. The sequence of events for engaging a single target begins when both the search and confirmation returns indicate the presence of an object. After confirmation, pulses must be allocated to the object for track initiation, early track, and precision tracking prior to intercept planning. When several objects are present, the problem is to decide which particular object(s) should be tracked or if search should be performed. The effectiveness of a given defense system depends on how wisely this decision is made within the resource allocation schema.

The operational aspects of a representative defense system are shown in Figure 2.1. The system computer hosts various tactical software functions which control the actions of the radar as well as the interception of threatening objects. The resource allocation function assigns radar resources (energy transmitted) for a fixed interval of time to either the search function or the track function. The duration and type of assignments are chosen to maximize the incremental gain in system performance as determined from observations of the threat and from a knowledge of the system's resources and deployment.

The radar scheduling function generates radar commands based on the requests from the search or track function. The radar commands contain the information

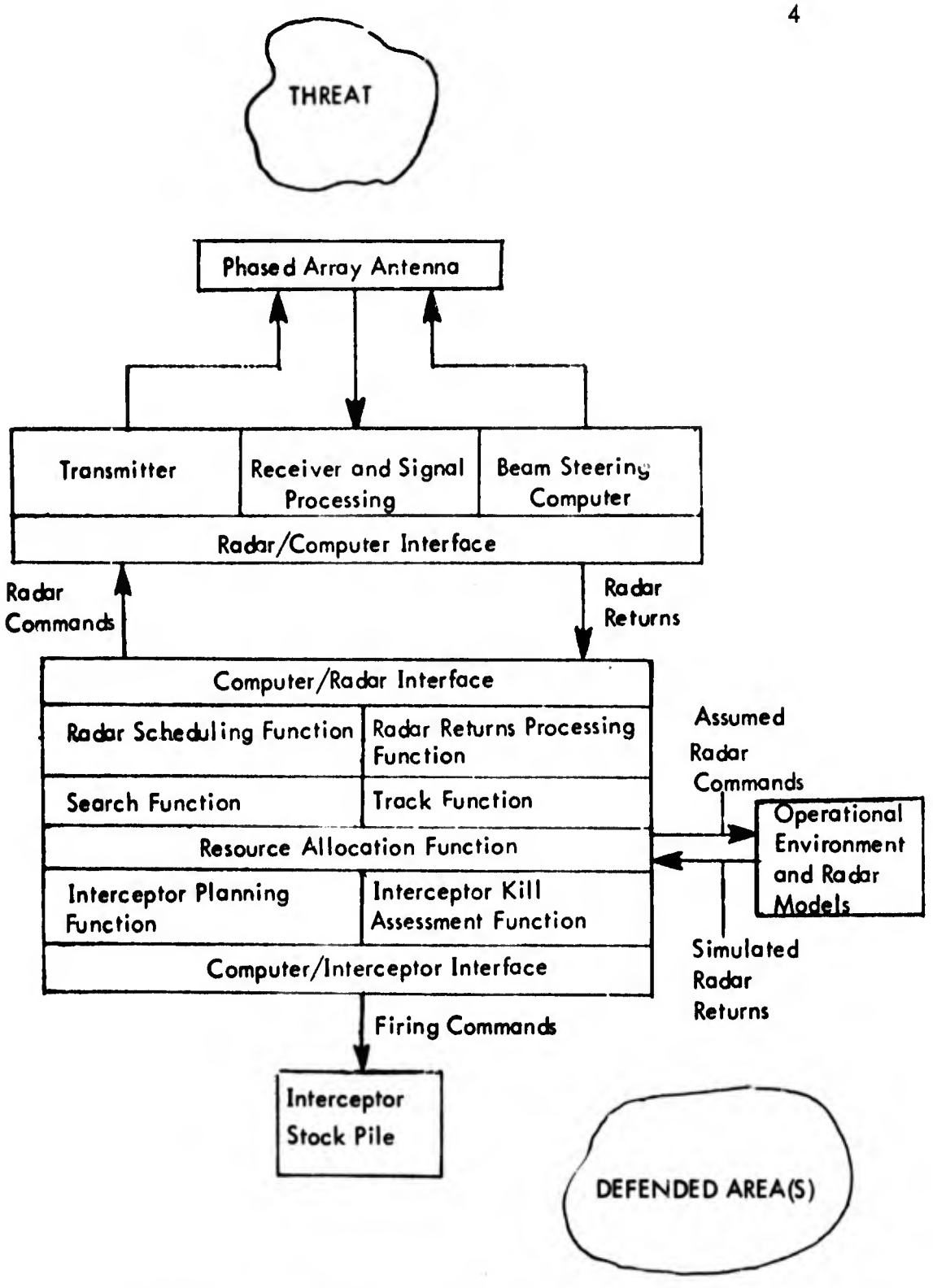


Figure 2.1 Representative Defense System Employing Resource Allocation

required to direct the following: (1) the antenna beam position for transmit and receive, (2) the transmit time, (3) the transmit frequency and waveform type, and (4) the signal processing configuration, including channel selection and processing window timing. The radar returns processing function routes the radar returns to the search or track function.

Upon obtaining enough information to provide an acceptable rms predicted miss distance, the interceptor planning function orders an interceptor firing. The kill assessment function evaluates the kill probability for each intercept attempted, and initiates additional firing at the same target when required.

There are two basic approaches to the resource allocation problem: static resource allocation and dynamic resource allocation. The static approach uses a fixed set of rules and can be optimized only for a given attack scenario; system performance may drop considerably against a different attack scenario. The dynamic approach is based on the measured and predicted state of the system and provides near optimum system performance over a wide variation of attack and defense scenarios; its effectiveness depends on a meaningful measure of system performance gain versus resource allocation. Optimum system performance means providing the most favorable balance between damage to the enemy and damage to the defended areas, given limited resources.

This chapter describes a methodology for <u>dynamic</u> resource allocation which is based on a measure of total defense posture called System Worth. The computation of System Worth effectively combines the pertinent threat and system parameters into a single numerical index upon which the allocation decisions are based. For tracking allocations, the best selection of object(s) and allocation time interval is the one which

maximizes the per unit time increment of System Worth, denoted as Fictitious Incremental Gain. This tracking allocation is compared to a search allocation of the same duration to determine if search or track should be performed.

The resource allocation schema described in this chapter was designed, implemented, and demonstrated by exercising it against a simulated raid containing ten attacking aircraft. The results are presented in Section 2.3. The structure of the simulation model is general and includes all the essential components of a modern defense system, including a multiple function phased array radar, computer software, and interceptors. This preliminary investigation, described more fully in references [1,2], illustrated that the procedures and algorithms produced a logical resource allocation sequence.

2.2 Resource Allocation and Methodology

2.2.1 Object Worth

Dynamic resource allocation requires that each observable object be assigned an index which indicates the benefits to be derived from allocating resources to the object, as well as the penalty paid when no allocation is made. This index, denoted as Object Worth, defines the relative value of each object to the defense. The computation of Object Worth is based on the following premises:

(1) The intrinsic value of an object is related to the amount and quality of information available on the object. This factor is called Object Knowledge, OK.

(2) An object which poses an immediate threat to a defended area is given additional weighting (higher priority) to insure that it can be intercepted before

reaching its objective. This factor is called Potential Damage to the Defense, PDD.

(3) An object receives additional weighting when it becomes vulnerable to the defense, i.e., when: (a) its position and velocity are known with sufficient accuracy to attempt an intercept, (b) the object is within intercept range, and (c) an interceptor is available. This is called Potential of a Successful Intercept, PSI.

Object Worth is defined as the sum of these factors, and will be written in general form as

$$OW(t,i,k) = OK(t,i,k) + PDD(t,i) + PSI(t,i,k) \qquad (2.1)$$

where the first argument indicates the time at which the calculation is valid, i denotes the object number being considered, and k defines the number of pulses assumed for the allocation interval. Note that PDD has only two arguments because it is not a function of the pulse allocation. An acronym symbology is used because the letters immediately identify the parameter being discussed, and because the acronym name can be used without confusion in the computer simulation program.

The resource allocation schema depends on the change in the Object Worth of each object during the allocation interval, with or without pulse allocation on that object. Therefore, computation of Object Worth is required under three assumptions: (1) the actual worth at time t_s corresponding to the start of an allocation interval, $OW(t_s, i, 0)$, assuming no new information on the object, (2) the predicted worth at the end of the allocation interval (time $t_s + k\,\Delta t$), assuming that k-pulses are allocated, $OW(t_s + k\,\Delta t, i, k)$, and (3) the predicted worth at the end of the allocation interval, assuming that no pulses are allocated, $OW(t_s + k\,\Delta t, i, 0)$. The time increment Δt denotes the basic pulse repetition period of the radar.

The corresponding Object Knowledge, Potential Damage to the Defense, and Potential of a Successful Intercept under these same three assumptions are discussed next.

2.2.1.1 Object Knowledge

The amount and quality of information available on each object are defined adequately by two parameters: the estimated tracking error and the predicted miss distance, assuming an intercept is attempted immediately. Since the tracking error in range is usually much smaller than the angular error, Object Knowledge, $OK(t,i,k)$, varies inversely as the estimated or predicted rms angle error, $e(t,i,k)$. Clearly, the angle error must be less than half the radar antenna beam width to allow continuation of track. A smoothing and prediction algorithm is used to supply values of the angular error for each object, under the assumptions noted above, i.e., (1) the error at time t_s, using radar measurements taken through time t_s, $e(t_s,i,0)$, (2) the predicted error at time $t_s + k\Delta t$, assuming k-pulses are allocated to the object, $e(t_s + k\Delta t,i,k)$, and (3) the predicted error at time $t_s + k\Delta t$, assuming no allocation is made to the object, $e(t_s + k\Delta t,i,0)$.

The predicted rms miss distance computations are based on: (1) the projected error in the position of the object at the time of predicted intercept, $t_I(t)$, using its estimated dynamics in conjunction with the prediction algorithm, and (2) the known characteristics of the interceptor. The object position errors and interceptor errors are combined in rms fashion to obtain the predicted rms miss distance, d. The following values of d are determined for each object under the conditions indicated by the arguments: $d(t_I(t_s),i,0)$, $d(t_I(t_s + k\Delta t),i,k)$, and $d(t_I(t_s + k\Delta t),i,0)$.

Note that the predicted miss distance should be smaller than the interceptor kill radius before an actual firing of an interceptor is attempted.

The general expression for Object Knowledge, as used in (2.1), is of the form

$$OK(t,i,k) = \frac{a_1}{e(t,i,k)} + \frac{a_2}{d(t,i,k)} \qquad (2.2)$$

where a_1 and a_2 are proportionality constants. It should be pointed out that the assumed k measurements for the computation of $OK(t_s + k\,\Delta t,i,k)$ are not available and must be simulated by appropriate radar and environmental models, as depicted in Figure 2.1.

2.2.1.2 Potential Damage to the Defense and Potential of a Successful Intercept

The Potential Damage to the Defense, PDD(t,i,k), with respect to the i th object is a function of the value of the defended area threatened, distance to the defended area, object velocity, and relative heading. The particular function will depend upon the defensive system. For the simulation discussed later in this paper, the following simplified expression is used,

$$PDD(t,i) = \begin{cases} P_{max} - \frac{P_{max}}{R_t} R(t,i) & \text{, for } R(t,i) < R_t \\ 0 & \text{, otherwise} \end{cases} \qquad (2.3)$$

where R is the range to object i at time t, P_{max} is the value of PDD at zero range, and R_t is a risk threshold (range at which an object becomes a potential risk to the defense).

The Potential of a Successful Intercept is a measure of an object's vulnerability to the defense. PSI by definition varies inversely with d, the predicted rms miss

distance. If d is greater than the interceptor kill radius, the probability of a successful intercept is small and PSI is assumed zero; otherwise, PSI is a nonlinear function of d which depends on the particular defensive weapon. In the simulation discussed later, PSI is assumed to have the form

$$PSI(t,i,k) = 1 - \exp\left\{-1.5\left[\frac{R_k}{d(t,i,k)}\right]^2\right\} \tag{2.4}$$

where R_k is the interceptor kill radius.

2.2.2 System Worth and Fictitious Incremental Gain

The System Worth is defined as the summation of all Object Worths. Thus, it represents a useful measure of total defense posture whereby the information on each observable object and on the defense is combined in a rational and systematic manner. Two System Worths are considered: the Actual System Worth, ASW, which represents the system worth at the start of the allocation interval, and the Potential System Worth, PSW(i,k), which is the predicted system worth assuming a k-unit allocation on the i th object. It follows that

$$ASW = \sum_{i=1}^{N} OW(t_s,i,0) \tag{2.5}$$

and

$$PSW(i,k) = OW(t_s + k\,\Delta t,i,k) + \sum_{\substack{j=1 \\ (j \neq i)}}^{N} OW(t_s + k\,\Delta t,j,0) \tag{2.6}$$

where N is the number of objects in track.

A measure of system performance gain when k-units are allocated to the i th object is denoted as Fictitious Incremental Gain, FIG(i,k), and is given by

$$FIG(i,k) = \frac{PSW(i,k) - ASW}{k} \quad . \tag{2.7}$$

2.2.3 Resource Allocation Schema

The resource allocation schema is the step-by-step procedure for implementing the resource allocation concepts and deciding which of the numerous alternatives is most favorable to the defense. The approach utilized here is to first determine the best tracking allocation, and, second, to compare this tracking allocation with a search allocation of the same duration to determine if search or track should be performed. This procedure, discussed in the next two sections, is summarized in Figure 2.2.

2.2.3.1 Selection of Tracking Allocation

The selection of the best tracking allocation is based on determining the maximum FIG(i,k) with respect to object number (i) and number of pulses to be allocated (k), where i ranges in value from one to the number of objects in track, and k assumes a finite set of values $\{k_1, k_2, \ldots, k_\ell\}$. The object number i_m, and allocation value, k_m, which maximizes FIG defines the best object selection and allocation time interval for track. The determination of an optimum lower and upper limit for the discrete values of k (k_1 and k_ℓ), as well as the number of choices allowed in the iterations, must be ascertained from the available radar and data processing resources.

The computer resources required for these computations may be reduced when targets approach a defended area in groups. Targets within a group may have nearly

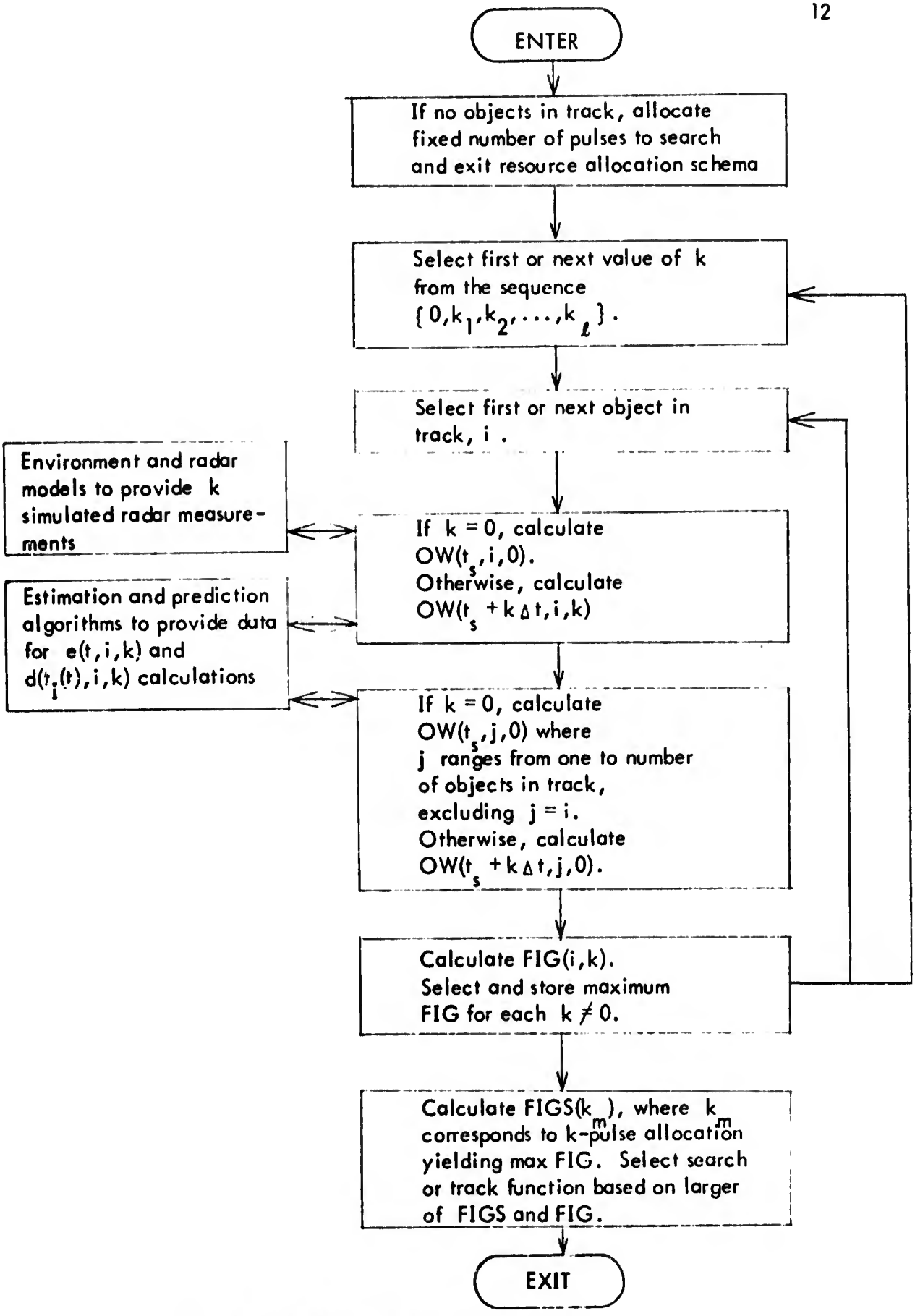


Figure 2.2 Resource Allocation Schema

equal Object Knowledge; consequently, the Fictitious Incremental Gain computed for one of the targets will be nearly equal to the others. Therefore, by forming sets of targets, based on their Object Knowledge, the Fictitious Incremental Gain is computed for only one member of each set and the remaining members are assigned the same value.

2.2.3.2 Decision Between Search and Track

Once the optimum allocation for track has been determined by the maximum FIG(i,k), one must decide if search should be preferred. For this purpose, a Fictitious Incremental Gain for Search { $FIGS(k_m)$ } is defined, assuming an allocation time interval corresponding to k_m for the best tracking allocation. FIGS takes into account that the average time for search must be a function of the number of targets being tracked, and should become zero when high priority tracking saturation occurs. While the average search rate is a deterministic function of system load, the instantaneous rate can vary so that high priority tracking { large FIG (i,k) } is not interrupted. These ideas are implemented by making $FIGS(k_m)$ dependent on the past average value of FIG(i,k), the time elapsed since the last search, and the number of objects being tracked.

Finally, if $FIGS(k_m)$ is greater than the maximum FIG(i,k), the next k_m pulses are allocated to search. Otherwise, k_m pulses are allocated to the i_mth object for continued track.

2.3 Defense System Modeling and Simulation

The demonstration of the radar resource allocation schema is most effectively accomplished by modeling and simulation of a defense system engagement. Computer

models of the components of an air defense system, the environment and the threat, and the corresponding computer simulation program were developed. Finally, a ground-to-air battle was simulated on the Univac 1108, using Fortran IV as the source language.

2.3.1 System Model

A model for the representative defense system employing the resource allocation schema of Figure 2.2 is depicted in Figure 2.3. The models for the search, track, interceptor planning, and interceptor kill functions provide the operational capabilities identified in Section 2.1. Models for the interface functions are not required since the system model resides in total within one computer. Those models peculiar to the simulation, and specific algorithms of importance to the resource allocation, are discussed briefly in the following paragraphs.

The threat model receives time and object identification from the radar model and uses deterministic equations of motion to generate the exact state vector for the object being interrogated by the radar. The deterministic equations of motion are based on position/time samples provided as input data. The constants by which these time dependent equations are defined are stored in the threat data file.

The radar model interrogates the threat model to synthesize radar returns. For a search or confirmation beam, the radar model simulates processing and detection. For a track beam, it simulates the measurements for range and angular position. The position data, perturbed by random noise, are fed into the track function model for smoothing and prediction.

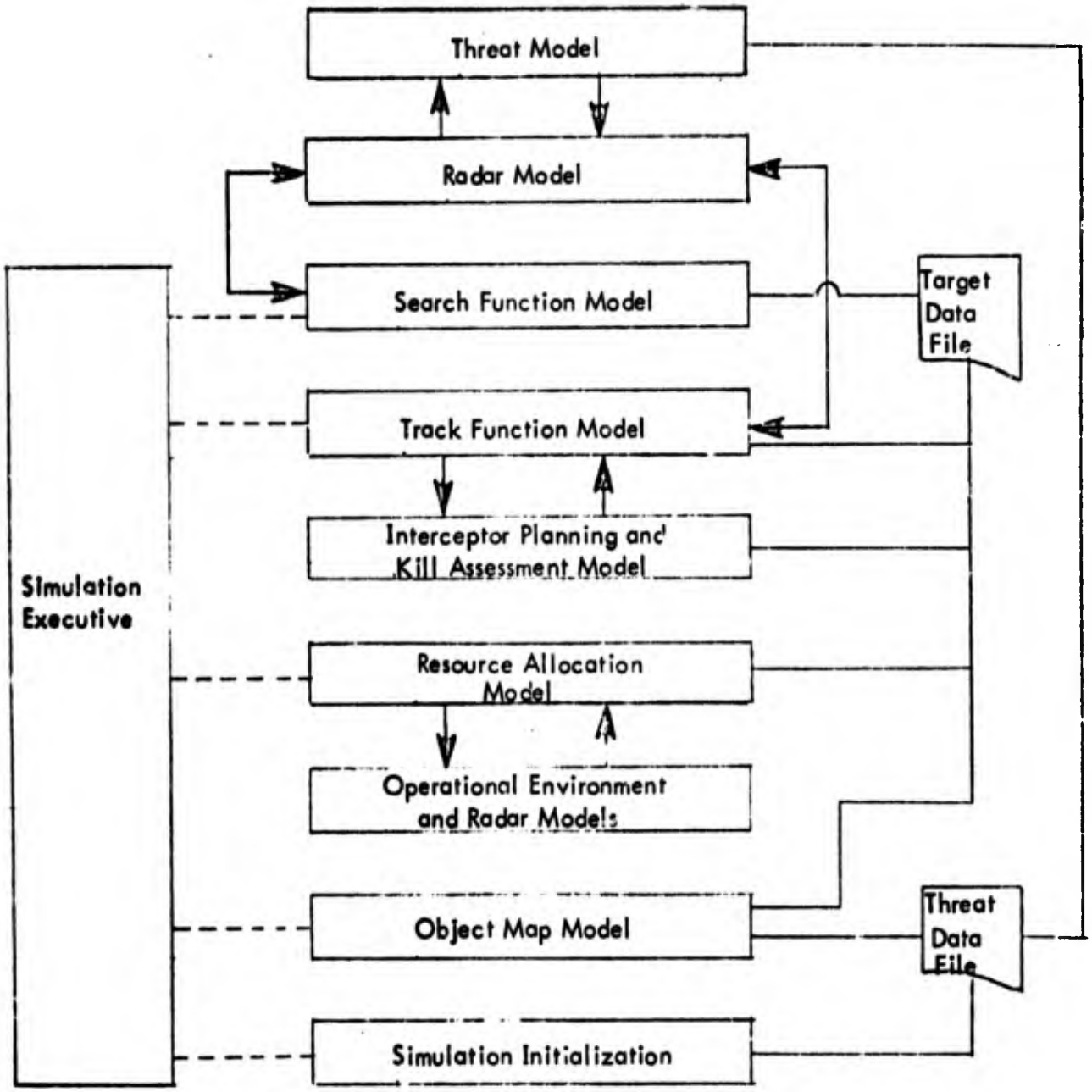


Figure 2.3 Model of Representative Defense System Employing Resource Allocation

The object map model forms and updates two maps: the true object map and the simulated object map. The true object map shows the true azimuth and elevation angles generated by the threat model to form a map representing the field of view scanned by the radar. The simulated object map is the object map seen by the radar. It pictures the information generated by the radar model and track function.

The resource allocation model is the computer implementation of the fundamental schema described in Section 2.2. It incorporates a complete concept for multiple target allocation within the same allocation interval, and a procedure for reducing computer resources through object grouping. The set of values which k can assume in the computations of FIG(i,k) was taken to be { 5, 10, 15, 20 }.

The search function model identifies the position of the antenna beam for random scanning. It schedules a confirmation beam when the radar model detects an object. Known object rejection is provided to prevent redundant tracks by comparing the new detection with objects in the simulated object map.

The track function model implements a tracking algorithm that is based on a generalized fixed memory polynomial filter. The tracking algorithm performs smoothing and trajectory prediction, and provides an estimate of the object state vector variance used in calculating $e(t,i,k)$ and $d(t_I(t),i,k)$.

The interceptor planning and kill assessment model computes the predicted intercept and determines the success of all firing attempts based on the state vector variance of the object and interceptor variances.

2.3.2 Simulated Program Overview

The simulation executive controls the simulation initialization, initializes and controls the sequencing among the system models, and provides the necessary interfacing and outputs. The inputs read by the simulation initialization program consist of parameters for the radar, resource allocation, interceptor and threat models.

The outputs describe the object(s) and the system at each event step, and provide statistical information on resource allocation and effectiveness. All the pertinent information on the objects comes from the object map model, which also identifies object(s) being tracked at that event step. The system description at each event step includes mode of operation and allocation, System Worth, Fictitious Incremental Gain, and damage to offense and defense. Analysis of the sequence of events indicates if the decisions are in agreement with accepted procedures.

The statistical information on resource allocation shows the Search-to-Track ratio versus time and number of targets, the relative track allocation for each object, and the average allocation duration. Effectiveness included comparison of damage to the enemy to resources spent.

2.3.3 Results for a Typical Ground-to-Air Battle

A threat consisting of ten aircrafts arranged in three groups, converging on the radar, was exercised against the defense system models. The inputs consist of parameters to describe the targets, the radar, and the interceptors. The outputs provide information on the sequence of events and the resource allocation performance.

Figure 2.4 shows the cumulative pulse allocations for search and track during a 71 second period, as well as the tracking allocations for one of the targets up

FIGURE 2.4 TRACK AND SEARCH ALLOCATION

to interception. Analysis of the simulation results verifies that the Search-to-Track ratio decreases when the number of detected targets increase, as desired, and that the sequencing from Search-to-Track, or from object to object, is satisfactory. The track rate increases immediately before firing. Details for the inputs and outputs and listings of the computer program can be found in Reference [2].

2.4 Summary and Conclusions

The objectives of this preliminary work were: (1) to develop criteria, procedures, and algorithms for optimum resource allocation, (2) to develop and implement system models, and (3) to develop a modular simulation program to demonstrate resource allocation and execute an example. The objectives were attained.

Two important contributions are the development of a quantitative measure of defense posture (System Worth) based on all available information, and optimization criteria for selecting the best resource allocation strategy. These basic concepts are applicable to any type of defense system.

The resource allocation schema is based on maximization of System Worth which is obtained by summing the Object Worths, over all objects. The Object Worth is the sum of Object Knowledge, Potential Damage to the Defense, and Potential of a Successful Intercept. The Object Knowledge is inversely related to the actual and predicted error of the position estimate. Each resource allocation requires the computation of many position estimates and of their errors. Various estimation algorithms will be developed and evaluated in the following chapters.

CHAPTER 3

SYSTEM MODEL

3.1 Introduction

A suitable mathematical model of the system (target and sensor) must be developed for implementation and evaluation of estimation algorithms. The model must be a good representation of the actual system and must be simple enough for easy implementation and reasonable computer requirements.

A target model is developed to represent a typical trajectory. Maneuvering is modelled as random accelerations whose statistics correspond to typical target capabilities. The model is discretized for use on the digital computer.

Then, an observation model is defined to compute the measurement samples from the actual true target position. The measurement vector is the sum of the true position vector and an error vector which combines all the sensor and environmental errors and is assumed to have Gaussian statistics.

3.2 Target Model

3.2.1 Analog Model

The three coordinates which completely describe the position of a target with respect to the radar are assumed to be the range R and two directional angles, α and β (Figure 3.1). The targets under consideration normally move on a constant velocity trajectory. All maneuvers will be viewed as perturbations upon this constant velocity trajectory. These will be modelled as random accelerations whose statistics adequately describe possible vehicle maneuvers. Further, since the range and angle measurement errors are independent, the filtering operation in the three

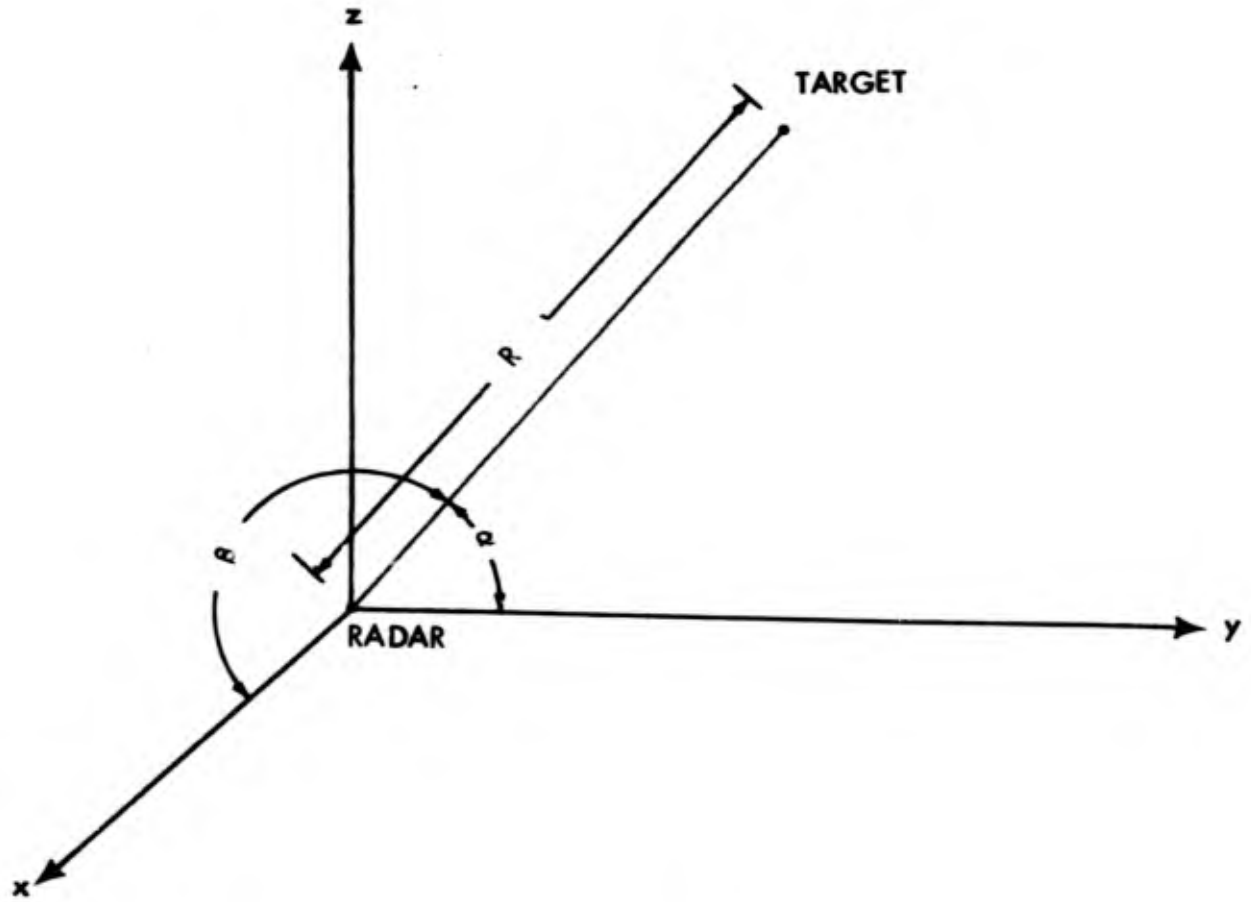


Figure 3.1 The Coordinate System

coordinates can be decoupled. In each coordinate, the target is described by the linear constant coefficient system

$$\frac{d}{dt} \underline{X}'(t) = F' \underline{X}'(t) + G' a(t) \quad , \tag{3.1}$$

where $\underline{X}'(t) = \begin{bmatrix} \text{target position } (R, \alpha, \text{ or } \beta) \text{ at time } t \\ \text{target position rate at time } t \end{bmatrix}$, (3.2)

$a(t)$ = target acceleration (in R, α, or β) at time t, (3.3)

$$F' = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \quad , \tag{3.4}$$

and

$$G' = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad . \tag{3.5}$$

The acceleration $a(t)$ is a random process representing the target maneuver and is called the target maneuver variable. In each coordinate, the target maneuver can be specified by its magnitude (variance) and its duration (time-constant). Because the time constant is nonzero, the target maneuver is correlated in time. A typical correlation function associated with this target maneuver is

$$R_a(\tau) = E[a(t)\, a(t+\tau)] = \sigma_m^2 \, e^{\gamma|\tau|} \quad , \gamma \geq 0 \tag{3.6}$$

where $E[\cdot]$ denotes mathematical expectation, σ_m^2 is the variance of the target

maneuver and $\frac{1}{\gamma}$ is the maneuver time constant. An evasive maneuver is represented by a high value of γ while a small value of γ is used to simulate atmospheric turbulence.

Smoothing and prediction filters, especially the Kalman filter, require that the system be driven with white noise instead of a time correlated random process. Therefore, the random process $a(t)$ in equation (3.1) must be expressed as the output of a linear system excited by white noise. This can easily be achieved by following the procedure described in Singer [3 ,4] and Wiener [5]. The resulting equation is

$$\frac{da(t)}{dt} = -\gamma a(t) + u(t) \quad , \tag{3.7}$$

where $u(t)$ is a white noise process with correlation function

$$R_u(\tau) = 2\gamma\sigma_m^2\,\delta(\tau) \quad . \tag{3.8}$$

The process $a(t)$ given by equation (3.7) can be adjoined to the vector $\underline{X}'(t)$ to give an augmented vector $\underline{X}(t)$ where

$$\underline{X}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} = \begin{bmatrix} \text{target position (R, } \alpha\text{, or } \beta\text{) at time } t \\ \text{target velocity at time } t \\ \text{target acceleration at time } t \end{bmatrix} . \tag{3.9}$$

Combining equations (3.7) and (3.1), the equations for $\underline{X}(t)$ can be written as

$$\frac{d}{dt}\underline{X}(t) = F\,\underline{X}(t) + G\,u(t) \quad , \tag{3.10}$$

where

$$F = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -\nu \end{bmatrix} , \tag{3.11}$$

and

$$G = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} . \tag{3.12}$$

3.2.2 Discrete Model

For digital analysis, it is necessary that the equations of motion be represented by difference equations rather than by differential equations. Let t_k represent the kth sampling instant and let $\Delta t_k = t_{k+1} - t_k$ be the kth sampling interval for $k = 0,1,2,\ldots$. The solution at time t_{k+1} to equation (3.10) can be written as

$$\underline{X}(k+1) = \Phi(\Delta t_k)\, \underline{X}(k) + \int_{t_k}^{t_{k+1}} \Phi(t_{k+1} - \tau)\, G u(\tau)\, d\tau \tag{3.13}$$

where $\underline{X}(k)$ denotes $\underline{X}(t_k)$ and $\Phi(t)$ is the state transition matrix of the linear time invariant system of equation (3.10). This transition matrix is given by $\Phi(t) = e^{Ft}$, or

$$\Phi(t) = \begin{bmatrix} 1 & t & \frac{1}{\nu^2}(-1 + \nu t + e^{-\nu t}) \\ 0 & 1 & \frac{1}{\nu}(1 - e^{-\nu t}) \\ 0 & 0 & e^{-\nu t} \end{bmatrix} . \tag{3.14}$$

For small values of $_{\nu}t$, the transition matrix may be approximated as

$$\Phi(t) = \begin{bmatrix} 1 & t & \frac{t^2}{2} \\ 0 & 1 & t \\ 0 & 0 & 1 \end{bmatrix} \quad . \tag{3.15}$$

Equation (3.13) can be written as

$$\underline{X}(k+1) = \Phi(\Delta t_k)\, \underline{X}(k) + \underline{W}(k) \tag{3.16}$$

where

$$\underline{W}(k) = \int_{t_k}^{t_{k+1}} \Phi(t_{k+1} - \tau)\, G u(\tau)\, d\tau \tag{3.17}$$

Since the time intervals are mutually disjoint, and since $u(t)$ is a white noise process, it is easily seen that $\underline{W}(k)$ is an independent vector sequence with zero mean. That is

$$E[\underline{W}(k)\, \underline{W}^T(j)] = Q(k)\, \delta_{j,k} \tag{3.18}$$

where $Q(k)$ is the covariance matrix of the vector sequence $\underline{W}(k)$. Using equations (3.8) and (3.17), it can be shown that the covariance matrix is given by

$$Q(k) = 2\gamma\sigma_m^2 \begin{bmatrix} q_{11}(k) & q_{12}(k) & q_{13}(k) \\ q_{12}(k) & q_{22}(k) & q_{23}(k) \\ q_{13}(k) & q_{23}(k) & q_{33}(k) \end{bmatrix} \tag{3.19}$$

where

$$q_{11}(k) = \frac{1}{2\gamma^5}\left[1 - e^{-2\gamma\Delta t_k} + 2\gamma\Delta t_k + \frac{2\gamma^3(\Delta t_k)^3}{3} - 2\gamma^2(\Delta t_k)^2 - 4\gamma\Delta t_k e^{-\gamma\Delta t_k}\right] \tag{3.20}$$

$$q_{12}(k) = \frac{1}{2\gamma^4}\left[1 + e^{-2\gamma\Delta t_k} - 2e^{-\gamma\Delta t_k} + 2\gamma\Delta t_k e^{-\gamma\Delta t_k} - 2\gamma\Delta t_k + \gamma^2(\Delta t_k)^2\right] \tag{3.21}$$

$$q_{13}(k) = \frac{1}{2\gamma^3}\left[1 - e^{-2\gamma\Delta t_k} - 2\gamma\Delta t_k e^{-\gamma\Delta t_k}\right] \tag{3.22}$$

$$q_{22}(k) = \frac{1}{2\gamma^3}\left[4e^{-\gamma\Delta t_k} - 3 - e^{-2\gamma\Delta t_k} + 2\gamma\Delta t_k\right] \tag{3.23}$$

$$q_{23}(k) = \frac{1}{2\gamma^2}\left[e^{-2\gamma\Delta t_k} + 1 - 2e^{-\gamma\Delta t_k}\right] \tag{3.24}$$

$$q_{33}(k) = \frac{1}{2\gamma}\left[1 - e^{-2\gamma\Delta t_k}\right] \tag{3.25}$$

3.3 The Observation Model

It is assumed that the position of the target is being measured by the sensor. In each of the three coordinates, the observation sequence is given by

$$Z(k) = M\,\underline{X}(k) + V(k) \tag{3.26}$$

where $M = [1 \quad 0 \quad 0]$

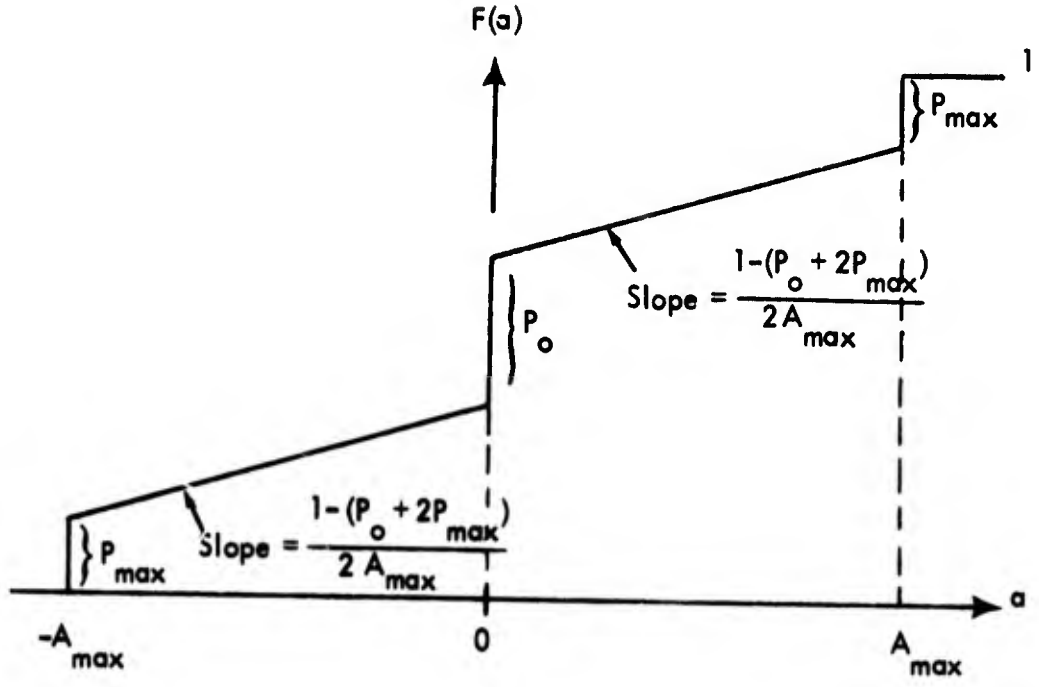
and $V(k)$ is an additive independent Gaussian noise sequence with zero mean and covariance given by

$$E[V(k)\,V(j)] = \sigma_v^2\,\delta_{k,j} \tag{3.27}$$

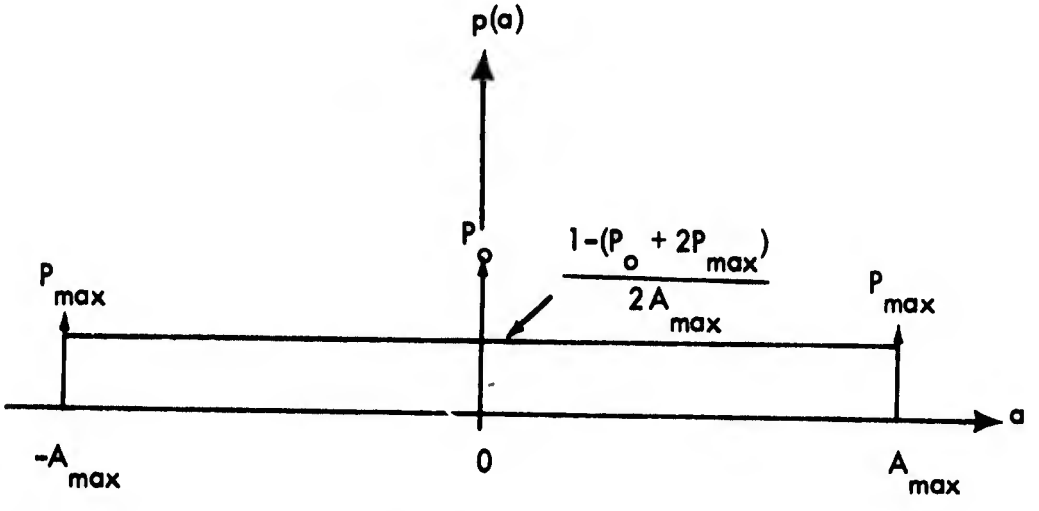
The value of σ_v^2 depends on the coordinate being measured and the sensitivity of the sensors.

3.4 Selection of Parameters

In order to choose a value for σ_m^2, it is necessary to choose a probability distribution which adequately represents the target acceleration. One such distribution function and the corresponding probability density function are shown in Figures 3.2(a) and 3.2(b). According to this model, the target undergoes no maneuvering at all with probability P_o, it can accelerate at a maximum value



(a) Probability Distribution Function of Vehicle Acceleration



(b) Probability Density Function of Vehicle Acceleration

Figure 3.2 Probability Distribution and Density Functions of Vehicle Acceleration

of $\pm A_{max}$ with probabilities of P_{max} each and can accelerate between limits $-A_{max}$ to A_{max} with a uniform distribution. It is easy to see that such a random variable has zero mean and variance given by

$$\sigma_m^2 = \frac{A^2_{max}}{3} \quad [1 - P_o + 4 P_{max}]. \tag{3.28}$$

Typical values of A_{max} are $60\ m/sec^2$ in range and $6\ mrad/sec^2$ in the two angles for an average elevation of about 10000 m.

For the type of trajectories considered, it is assumed that one sigma measurement accuracies in range and angle are 10 m and 8 mrad, respectively. This gives $\sigma_v^2 = 100$ for range measurement and $\sigma_v^2 = 64 \times 10^{-6}$ for angle measurement.

3.5 Summary and Conclusions

A model was derived for the system which describes adequately the vehicle motion including maneuvering due to random perturbations or pilot control. The target model is a linear time-invariant system driven by white noise. Equivalent discrete state equations were derived to facilitate implementation on a computer. Discrete observations were assumed to be corrupted by an independent sequence whose variance represents the measurement accuracy of the sensor. It was assumed that each of the three coordinates follows similar system models. The system model will be used in the next chapter.

CHAPTER 4

ESTIMATION ALGORITHMS

4.1 Introduction

In this chapter a brief description of three commonly used estimation algorithms is presented. Two of these filters, the g-h filter (sometimes called $\alpha - \beta$ filter) and the Kalman filter, are recursive in nature, while the third, the fixed-memory polynomial filter, requires that the computation be performed on all samples in memory at each step. These three filters were selected for comparison since they are of the type often considered for implementation in tactical systems. The g-h filter is of particular interest here because it is presently being considered for implementation in the EAR system.

The description of each filter includes the implementation algorithm, the initialization procedure where applicable, and formulas for the variance of the error in the estimate. Computer programs are written for each filter from the algorithms and initialization procedures and are listed, along with a brief description, in Appendix C. These programs will be used in Chapter 5 to evaluate the performance of the three estimation algorithms for two typical trajectories. The error variance of the estimator serves as the basis of the track and search algorithms to be described in Chapter 6.

4.2 The g-h Filter

4.2.1 g-h Filter Implementation

The g-h filter is designed to minimize the mean-squared error in filtered position and velocity under the assumption of straight line (constant velocity)

target motion. The smoothing and prediction algorithms for this filter are given by

$$\underline{\hat{X}}(k+1\,|\,k+1) = \underline{\hat{X}}(k+1\,|\,k) + \begin{bmatrix} g \\ \underline{h} \\ \Delta t_k \end{bmatrix} [\, Z(k+1) - M\,\underline{\hat{X}}(k+1\,|\,k)\,] \qquad (4.1)$$

$$\underline{\hat{X}}(k+1\,|\,k) = \begin{pmatrix} 1 & \Delta t_k \\ 0 & 1 \end{pmatrix} \underline{\hat{X}}(k\,|\,k) \qquad (4.2)$$

where

$$\underline{\hat{X}}(k\,|\,k) = \begin{bmatrix} \hat{x}_1(k\,|\,k) = \text{estimate of the position at } k \text{ th sampling instant} \\ \hat{x}_2(k\,|\,k) = \text{estimate of the position rate} \end{bmatrix}$$

and $M = [\, 1 \quad 0\,]$.

A recursive relation for the error covariance matrix can be shown(See Appendix B) to be

$$P(k+1\,|\,k+1) = B_k\, P(k+1\,|\,k)\, B_k^T + \sigma_v^2\, K_k K_k^T$$

$$P(k+1\,|\,k) = \Phi_k\, P(k\,|\,k)\, \Phi_k^T + \sigma_m^2\, G_k G_k^T + \Phi_k C(k)\, G_k^T + G_k C^T(k)\, \Phi_k^T \qquad (4.3)$$

where all quantities are defined in Appendix B.

It is interesting to see what effects a maneuvering target will have on this covariance matrix. To simplify matters, let the system noise given by the term $\sigma_v^2\, K_k K_k^T$ be zero. Then, from the second element of (4.1), the error in predicted position is

$$\epsilon = x_1(k+1) - \hat{x}_1(k+1\,|k) = \frac{(\Delta t_k)^2}{h}\left(\frac{\hat{x}_2(k+1\,|k+1) - \hat{x}_2(k+1\,|k)}{\Delta t_k}\right) \quad (4.4)$$

Obviously, the maximum error in predicted position occurs in the interval of maximum vehicle acceleration and is given by

$$\epsilon_{max} = \frac{A_{max}(\Delta t_k)^2}{h} \quad (4.5)$$

Equation (4.4) represents the element of the first row and first column of the matrix P under the assumption that no sensor noise is present.

4.2.2 Selection of Parameters

The servo-loop generated by (4.1) and (4.2) has the following characteristics [6]:

$$\begin{aligned} &(2g + h) < 4 && \text{for stability,} \\ &(g+h)^2 < 4h && \text{for underdamping,} \\ &(g+h)^2 = 4h && \text{for critical damping,} \\ &(g+h)^2 > 4h && \text{for overdamping.} \end{aligned} \quad (4.6)$$

The position smoothing coefficient specifies the required system bandwidth. The smaller the required bandwidth, the smaller g can be made. The position rate smoothing coefficient h specifies the amount of damping in the system. Since bandwidth and damping are inverse quantities, a compromise must be used in selecting g and h. If the required system bandwidth is known, Benedict and Bordner [7] show that the optimum choice of h is

$$h = g^2 / (2 - g) \tag{4.7}$$

which yields a damping ration of $\sqrt{2}/2 = .707$.

The second element of (4.2) is a result of the g-h filter requirement of constant velocity trajectory. In order to detect vehicle maneuvers, circuitry external to the g-h filter is required. Since there is no provision for acceleration smoothing in the g-h filter, it is not possible to use the transition relation discussed in Chapter 3 and (4.2) will be used directly.

4.2.3 Filter Initialization

In order to minimize transient error, it is necessary to have an initialization procedure for the g-h filter. The first meaningful smoothed values obtained will be $\hat{x}_1(3 \mid 3)$ and $\hat{x}_2(3 \mid 3)$. The computation of these requires predicted values from the previous sampling instances. The computation of these predicted values requires the initial conditions

$$\hat{x}_1(2 \mid 2) = Z(2) \tag{4.8}$$

and

$$\hat{x}_2(2 \mid 2) = \frac{1}{\Delta t_1} \left(Z(2) - Z(1)\right) \quad . \tag{4.9}$$

The recursive relations representing this filter can now be set in motion by computing $\hat{x}_1(3 \mid 2)$ and $\hat{x}_2(3 \mid 2)$ from these initial conditions.

4.3 Fixed Memory Polynomial Filter

4.3.1 General Comments

This filter operates on the last (L + 1) observations where L is a preselected

integer. It computes the coefficients of a polynomial which fits these observations in the least square sense. This polynomial is an estimate of the process being observed. The derivatives of the estimating polynomial are used as estimates of the derivates of the true process. The polynomial is also used for prediction.

Two types of polynomial filters will be considered. The orthogonal polynomial filter is used for equal sampling intervals, while a more general polynomial filter is derived which will be used for unequal sampling intervals. The general polynomial filter requires a matrix inversion and is, thus, less efficient than the orthogonal polynomial filter.

4.3.2 Orthogonal Polynomial Filter

The fitting polynomial can be expressed as a sum of basis polynomials. These must be orthogonal in the estimation interval in order to avoid a matrix inversion during the computation of the coefficients. The discrete Legendre polynomial will be used as the orthogonal basis.

The normalized discrete Legendre polynomial of degree j, $\varphi_j(r)$, is defined in terms of the discrete Legendre polynomial $p(r;j,L)$, by

$$\varphi_j(r) = \frac{1}{c(j,L)}\, p(r;j,L) \ . \tag{4.10}$$

where

$$p(r;j,L) = \sum_{v=0}^{j} (-1)^v \binom{j}{v}\binom{j+v}{v} \frac{r^{(v)}}{L^{(v)}} \ , \tag{4.11}$$

with $L^{(v)} = (L)\,(L-1)\,\ldots(L-v+1)$ and $\binom{j}{v} = \dfrac{j!}{(j-v)!\,v!}$

and

$$[c(j,L)]^2 = \frac{(L+j+1)^{(j+1)}}{(2j+1)\,L^{(j)}} \quad . \tag{4.12}$$

The parameter L specifies the range of orthogonality. The estimation polynomial is given [8] by

$$[\hat{p}(r)]_k = \sum_{j=0}^{m} \left[\sum_{n=0}^{L} y_{k-L+n}\, \varphi_j(n) \right] \varphi_j(r) \tag{4.13}$$

where $[\hat{p}(r)]_k$ is the estimation polynomial based on the observations

$$\underline{Z}(k;L) = [Z(k-L)\;\; Z(k-L+1)\;\ldots\; Z(k)]^T \tag{4.14}$$

and m is the desired degree of the polynomial.

In general,

$$[D^i \hat{p}(r)]_k = \frac{1}{\tau^i} \sum_{j=0}^{m} \left[\sum_{n=0}^{L} y_{k-L+n}\, \varphi_j(n) \right] \frac{d^i}{dr^i} \varphi_j(r) \,, \tag{4.15}$$

where D^i denotes the i th derivative and τ is the interval between observations.

The unscaled estimate vector

$$\hat{X}(k+\ell \mid k) = \begin{pmatrix} \hat{p}(L+\ell) \\ D\hat{p}(L+\ell) \\ \vdots \\ D^m\hat{p}(L+\ell) \end{pmatrix}_k = \begin{pmatrix} \hat{p}(r) \\ D\hat{p}(r) \\ \vdots \\ D^m\hat{p}(r) \end{pmatrix}_k \tag{4.16}$$

can be written as

$$\underline{\hat{X}}(k + \zeta \mid k) = W(\zeta, \tau)\ \underline{\hat{X}}(k + \zeta \mid k) = W(\zeta, \tau)\ \underline{Z}(k; L) \tag{4.17}$$

where

$$W(\zeta, \tau) = \Phi(\zeta \tau)\, SGCB\,, \tag{4.18}$$

$$[\Phi(\zeta \tau)]_{ij} = \frac{(\zeta \tau)^{j-i}}{(j-i)\,!} \quad \text{for } 0 \le i,j \le m \tag{4.19}$$

S is the associate Sterling matrix of the first kind,

$$[G]_{ij} = (-1)^j \binom{j}{i} \binom{j+i}{i} \frac{1}{L^{(i)}} \text{ for } 0 \le i,j \le m\,, \tag{4.20}$$

$$[C]_{ij} = \frac{1}{[c(j,L)]^2}\,\delta_{ij} = \frac{(2j+1)\,L^{(j)}}{(L+j+1)\,(j+1)}\,\delta_{ij} \text{ for } 0 \le i,j \le m\,, \tag{4.21}$$

$$[B]_{ij} = [p(r;i,L)]\big|_{r=L-j} = \sum_{v=0}^{i} (-1)^v \binom{i}{v}\binom{i+v}{v} \frac{(L-j)^{(v)}}{L^{(v)}}$$

$$\text{for } 0 \le i \le m \text{ and } 0 \le j \le L\,. \tag{4.22}$$

The associate Sterling matrix of the first kind is given by the recurrence relation

$$[S]_{i,j} = [S]_{i-1,j-1} + (j-1)\,[S]_{i,j-1} \quad \text{for } 0 \le i,j \le m \tag{4.23}$$

with

$$[S]_{o,o} = 1 \text{ and } [S]_{o,j} = [S]_{i,o} = 0 \text{ for } i,j \leq 1 . \qquad (4.24)$$

It should be noticed that (4.19) will give the same result as (3.15) for m = 2.

A computer program was developed to compute the matrix for a given number of observations, prediction interval, and degree. This program is listed in Appendix C.

The L + 1 observations are received at time $t = t_o + \ell\tau$, where t_o is the start of the observation window and $\ell = 0, \ldots, L$. The true process can be estimated in the interval $t_o \leq t \leq t_o + L\tau$ and predicted for $t > t_o + L\tau$. The estimate of the true process is $\hat{p}(r)$ where $r = L$ gives the updated smoothed estimate, while $r = L + \zeta$ gives the ζ-step prediction. The highest possible degree of the polynomial filter is L.

4.3.3 Filter Implementation

The filter implementation is demonstrated on an example. Assume that the number of observations available is five (L = 4), the sampling interval is one ($\tau = 1$), and that a second degree polynomial (m = 2) will adequately describe the process. Then, from equation (4.15), it is seen that the following quantities are needed:

$$\varphi_o(r) = \frac{1}{c(o,L)} p(r;o,L) = \frac{1}{\sqrt{L+1}} = \frac{1}{\sqrt{5}}, \qquad (4.25)$$

$$\varphi_1(r) = \frac{1}{c(1,L)} p(r;1,L) \sqrt{\frac{3L}{(L+2)(L+1)}} \left[1 - \frac{2r}{L}\right] = \sqrt{\frac{2}{5}}(1 - r/2), \qquad (4.26)$$

$$\phi_2(r) = \frac{1}{c(2,L)} p(r;2,L) = \sqrt{\frac{5(L)(L-1)}{(L+3)(L+2)(L+1)}} \left[1 - \frac{6r}{L} + \frac{6(r)(r-1)}{(L)(L-1)}\right]$$

$$= \sqrt{\frac{2}{7}} \left[\frac{r^2 - 4r + 2}{2}\right]. \qquad (4.27)$$

Using these values in (4.15) yields

$$\begin{bmatrix} \hat{p}(r) \\ D\hat{p}(r) \\ D^2\hat{p}(r) \end{bmatrix}_k = \frac{1}{70} \begin{bmatrix} 10r^2 - 26r + 6 & -5r^2 + 27r - 10 & -10r^2 + 40r - 6 & -5r^2 + 13r + 19 & 10r^2 - 54r + 62 \\ 20r - 26 & -10r + 27 & -20r + 40 & -10r + 13 & 26r - 54 \\ 20 & -10 & -20 & -10 & 20 \end{bmatrix} \begin{bmatrix} Z(k) \\ Z(k-1) \\ Z(k-2) \\ Z(k-3) \\ Z(k-4) \end{bmatrix} \qquad (4.28)$$

A ζ-step prediction is obtained by evaluating this expression for $r = L + \zeta = 4 + \zeta$. For instance, if a 2.5-step prediction is desired, (4.28) becomes

$$\hat{p} = 3.71\,Z(k) - .65\,Z(k-1) - 2.41\,Z(k-2) - 1.55\,Z(k-3) + 1.91\,Z(k-4), \qquad (4.29)$$

$$D\hat{p} = 1.49\,Z(k) - .54\,Z(k-1) - 1.29\,Z(k-2) - .74\,Z(k-3) + 1.09\,Z(k-4), \qquad (4.30)$$

$$\hat{D^2_p} = .28\,Z(k) - .14\,Z(k-1) - .28\,Z(k-2) - .14\,Z(k-3) + .28\,Z(k-4). \quad (4.31)$$

If more than one process is being observed, a number of identical and completely independent filters will be required. For instance, suppose range, azimuth, and elevation are being monitored; then three filters will be required.

4.3.4 Effect of the Polynomial Degree on Computer Requirements

In this section only the second order polynomial filter is considered. However, a higher-ordered polynomial may be needed for a better fit of the observations. Increasing the order of the polynomial, while reducing the systematic errors, increases the random errors of the estimate and also the computer requirements.

The computer requirements as a function of polynomial degree were considered for polynomials of degree 1 through 8. For each degree only the position estimate based on nine observations and a 3-step prediction interval were considered. The experimental results are demonstrated in Figures 4.1 and 4.2. In both cases the results are normalized with respect to the 8th degree polynomial. We see in Figure 4.1 that the computer running time is approximately an exponential function of degree. However, the absolute computer running time for the 8th degree polynomial position estimate was only 9.456 milliseconds, and this time does not account for any interruptions which may have resulted due to the time-sharing nature of the computer system used. Also, Figure 4.1 resulted from a variable - ζ filter. If the required prediction interval is known beforehand, the filter coefficients all become constants and can be precomputed in which case the computer running time for all degree filters will be essentially equal. From Figure 4.2, it is obvious that the increase in computer storage requirements is negligible.

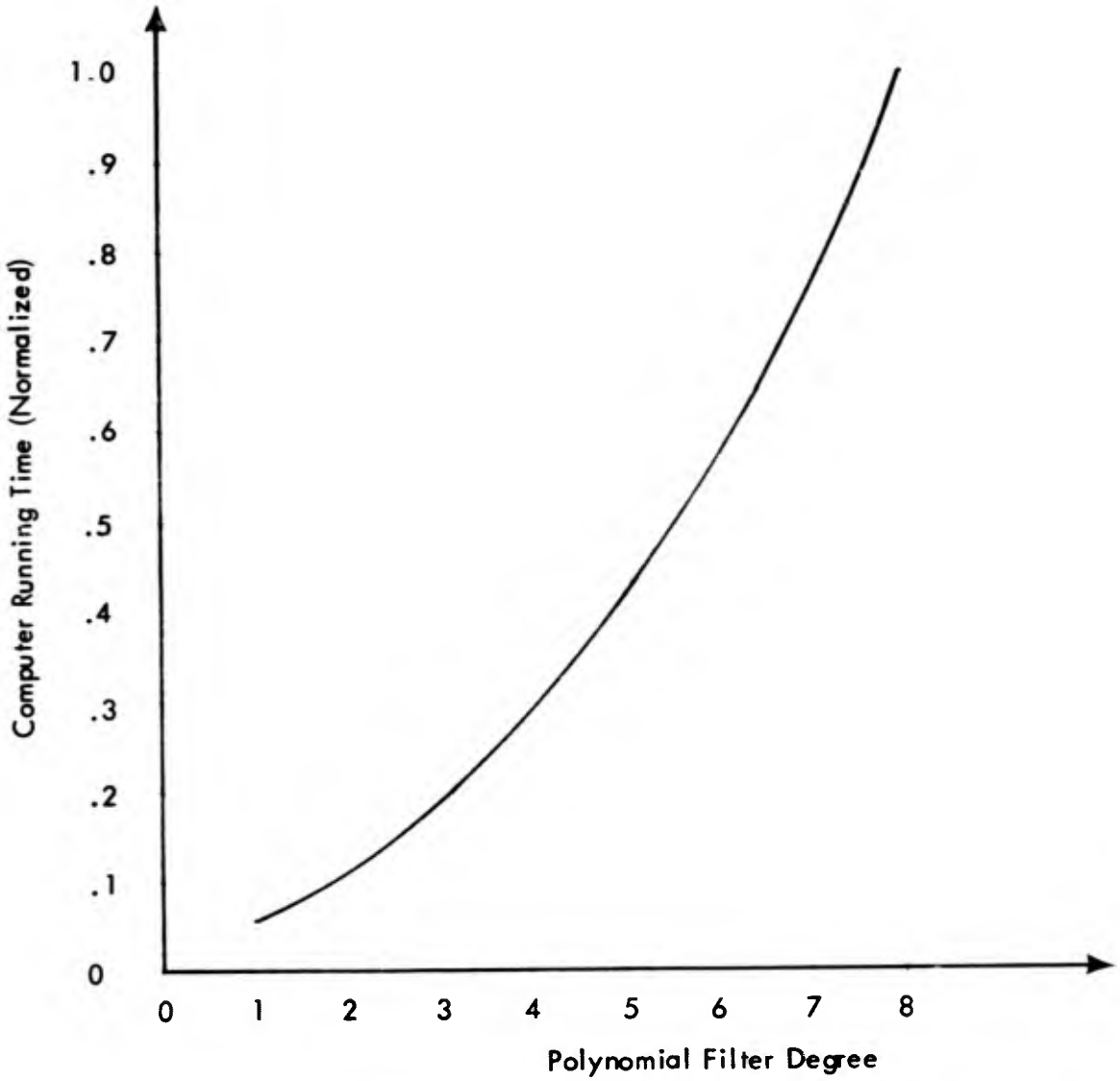


Figure 4.1 Computer Running Time vs. Fixed Memory Polynomial Filter Degree

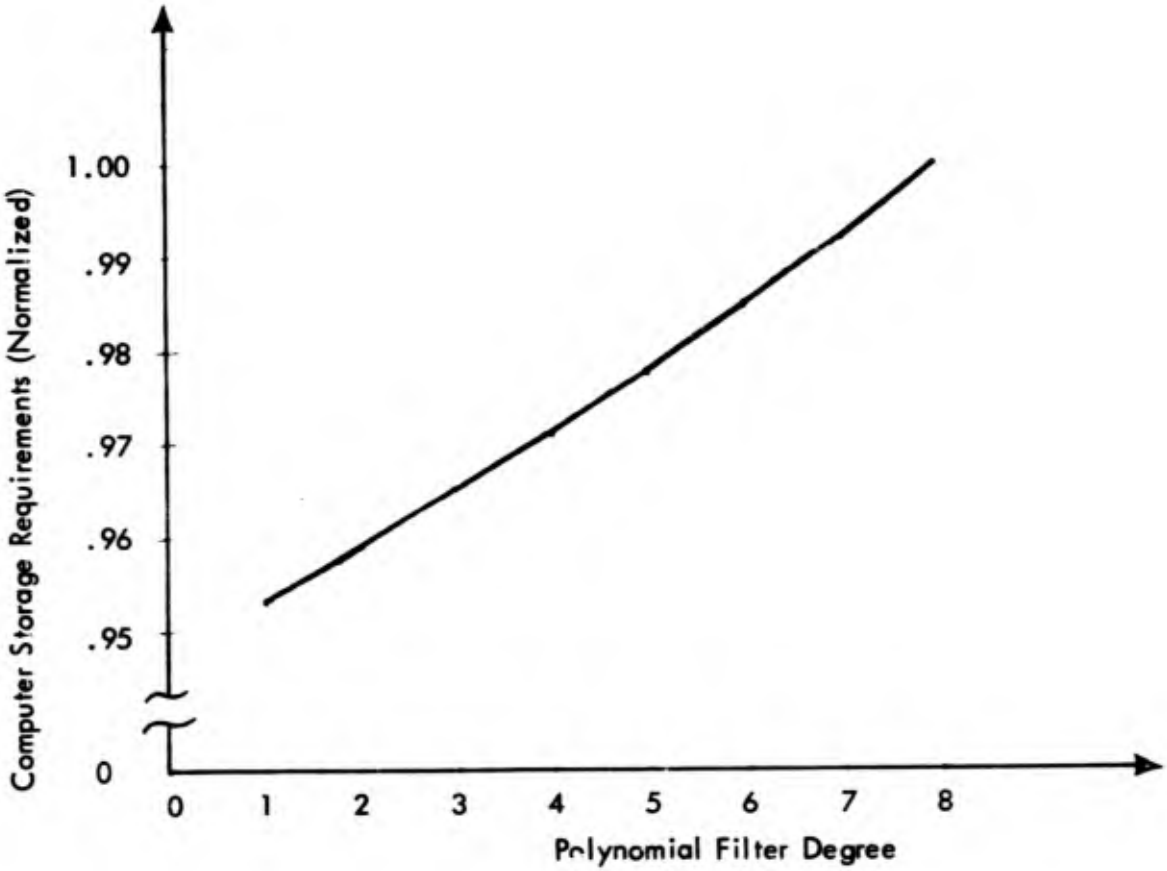


Figure 4.2 Computer Storage Requirements vs. Fixed Memory Polynomial Filter Degree

4.3.5 General Fixed Memory Polynomial Filter

In order to simplify the following discussion as well as to extract something useful, only the second-order filter will be implemented. We shall assume that a sequence of scalar observations... Z(k-1), Z(k) ... is being obtained from some linear process, and that we wish to fit a second-order polynomial to a fixed length record of them. The observation instants are assumed to be unequally spaced and we store both these and the observations themselves in push-down tables of adequate length.

The least-squares polynomial approximation to the sequence of observations,

$$\underline{Z}(k;L) = (Z(k)\ \ Z(k-1)\ \ldots\ Z(k-L))^T\ , \tag{4.32}$$

is

$$\hat{\underline{X}}(k\,|\,k) = (T_k^T T_k)^{-1}\ T_k^T \underline{Z}(k;L) \tag{4.33}$$

where

$$T_k = \begin{bmatrix} M_k \\ \hline M_{k-1}\,\Phi\,(t_{k-1},\ t_k) \\ \hline \vdots \\ M_{k-L}\,\Phi\,(t_{k-L},\ t_k) \end{bmatrix}\ , \tag{4.34}$$

$M_k = (1\ 0\ 0)$, and Φ is given by (3.15). Equation (4.34) can be written as

$$T_k = \begin{bmatrix} 1 & 0 & 0 \\ 1 & -(t_k - t_{k-1}) & \frac{(t_k - t_{k-1})^2}{2} \\ 1 & -(t_k - t_{k-2}) & \frac{(t_k - t_{k-2})^2}{2} \\ \vdots & \vdots & \vdots \\ 1 & -(t_k - t_{k-L}) & \frac{(t_k - t_{k-L})^2}{2} \end{bmatrix} . \qquad (4.35)$$

Therefore,

$$T_k^T T_k = \begin{bmatrix} \sum_{n=0}^{L} 1 & -\sum_{n=0}^{L} (t_k - t_{k-n}) & \sum_{n=0}^{L} \frac{(t_k - t_{k-n})^2}{2} \\ -\sum_{n=0}^{L} (t_k - t_{k-n}) & \sum_{n=0}^{L} (t_k - t_{k-n})^2 & -\sum_{n=0}^{L} \frac{(t_k - t_{k-n})^3}{2} \\ \sum_{n=0}^{L} \frac{(t_n - t_{n-k})^2}{2} & -\sum_{n=0}^{L} \frac{(t_k - t_{k-n})^3}{2} & \sum_{n=0}^{L} \frac{(t_k - t_{k-n})^4}{4} \end{bmatrix} \qquad (4.36)$$

and

$$T_k^T \underline{Z}(k;L) = \begin{bmatrix} \sum_{n=0}^{L} Z_{(k-n)} \\ -\sum_{n=0}^{L} (t_k - t_{k-n}) Z_{(k-n)} \\ \sum_{n=0}^{L} \frac{(t_k - t_{k-n})^2}{2} Z_{(k-n)} \end{bmatrix} . \qquad (4.37)$$

The error covariance matrix for the fixed-memory filter is given by [8]

$$P(k|k) = (T_k^T T_k)^{-1} T_k^T R_{(k)} T_k (T_k^T T_k)^{-1} .$$

For $R_{(k)} = \sigma_v^2 I$, this equation reduces to

$$P(k|k) = \sigma_v^2 (T_k^T T_k)^{-1} .$$

This random error covariance matrix can be predicted to any instant by

$$P(k+\zeta|k) = \Phi(\zeta) P(k|k) \Phi^T(\zeta) .$$

The computer implementation of this filter consists now of only programming equation (4.33) utilizing the relations (4.36) and (4.37). This program is listed in Appendix C.

4.4 The Kalman Filter

4.4.1 General Comments

The Kalman filter [9] is simply the recursive computation algorithm for Wiener [5] filtering. The Wiener filter, based on the Wiener-Hopf integral equation, is the best linear minimum-mean-squared error filter using the available observations. The estimate at a particular point in time is a linear combination of the current observation and all the previous observations. This results in an expanding memory situation unless all the information required of the previous observations is contained in the estimates at a fixed number of previous instants. This is indeed the case and the Kalman filter utilizes the fact that the latest estimate and the current observation completely determine the current estimate. The optimal

(in the minimum mean squared error sense) prediction is entirely dependent on the current estimate. It is known that the minimum mean squared estimate is also the conditional mean estimate, and that the conditional mean is a linear combination of the observations if the conditional density of the process to be estimated conditioned on the observations is Gaussian. Therefore, in the case of Gaussian processes, the Kalman filter does yield the minimum mean squared error estimate.

4.4.2 Estimation Algorithms

Since the Kalman filter algorithms have been widely published, only the final equations will be repeated here. A detailed derivation of the equations can be found in reference [10].

The system under consideration (as developed in Chapter 3, equation 3.16) is given by the following message model

$$\underline{X}(k+1) = \Phi(\Delta t_k)\ \underline{X}(k) + \underline{W}(k)\ , \tag{4.38}$$

where $\Phi(t)$ is given by equation (3.14) and $\underline{W}(k)$ is an independent vector noise sequence with zero mean and covariance $Q(k)$ given by equations (3.19) through (3.25). The observation sequence is given by

$$Z(k) = M\ \underline{X}(k) + V(k)\ , \tag{4.39}$$

where $M = [\,1\ 0\ 0\,]$ and $V(k)$ is an additive independent noise sequence with zero mean and a variance of σ_v^2. The problem is to find an estimate of $\underline{X}(k)$ based on measurements $Z(1), Z(2), \ldots, Z(j)$. This estimate will be denoted by

$\hat{\underline{X}}(k \mid j)$ where $k = j$ for filtering and $k > j$ for prediction. The two cases of equal and unequal sampling intervals are discussed next.

4.4.2.1 Equal Sampling Intervals ($\Delta t_k = \Delta t$ for all k)

In this case we restrict ourselves to filtering and one-step prediction. Thus it is required to find $\hat{\underline{X}}(k \mid k)$ and $\hat{\underline{X}}(k+1 \mid k)$. Following the development in reference [10], these are generated recursively using the following equations:

$$\hat{\underline{X}}(k+1 \mid k) = \Phi(\Delta t)\, \hat{\underline{X}}(k \mid k) \tag{4.40}$$

$$P(k+1 \mid k) = \Phi(\Delta t)\, P(k \mid k)\, \Phi^T(\Delta t) + Q(k) \tag{4.41}$$

$$K(k+1) = P(k+1 \mid k)\, M^T \left[M P(k+1 \mid k) M^T + \sigma_v^2 \right]^{-1} \tag{4.42}$$

$$\hat{\underline{X}}(k+1 \mid k+1) = \hat{\underline{X}}(k+1 \mid k) + K(k+1) \left[Z(k+1) - M\, \hat{\underline{X}}(k+1 \mid k) \right] \tag{4.43}$$

$$P(k+1 \mid k+1) = \left[I - K(k+1)\, M \right] P(k+1 \mid k) \tag{4.44}$$

where I is the identity matrix of order three.

In the above equations $G(k)$ is the gain matrix, $P(k+1 \mid k)$ is the error covariance matrix of the one-step prediction and $P(k \mid k)$ is the error covariance matrix of the filtered estimate. In order to start the recursive scheme, the initial conditions $\hat{\underline{X}}(1 \mid 1)$ and $P(1 \mid 1)$ are needed. The selection of these initial conditions is discussed in Section 4.4.3. The above recursive relation can be represented schematically as shown in Figure 4.3. In this case, since Δt remains constant, the matrix $Q(k)$ is independent of k and will have to be computed only once. The computational algorithm follows exactly the same sequence as the above equations.

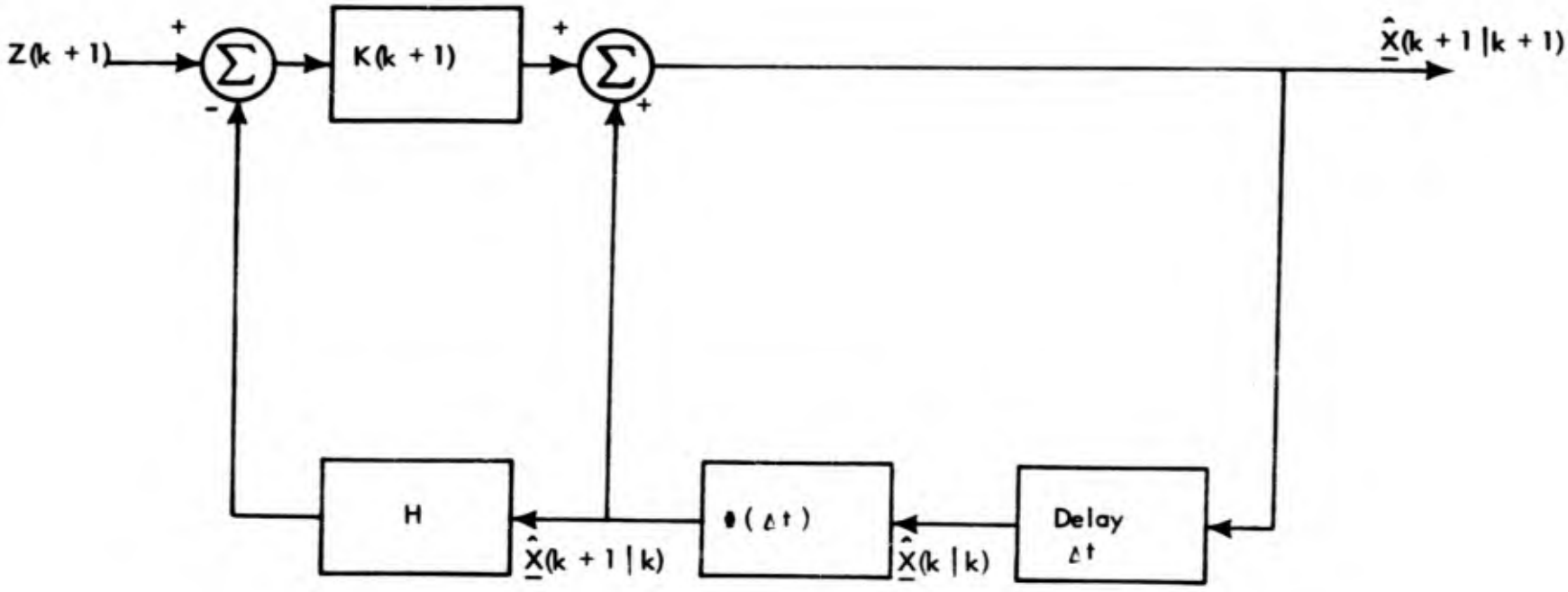


Figure 4.3 Block Diagram of Kalman Filter (Equal Sampling Intervals)

4.4.2.2 Unequal Sampling Intervals

From a practical standpoint, it is rarely possible for the sampling intervals to take a large number of different values. In most cases the sampling interval can take a finite number of different values each of which is an integral multiple of a minimum value.

Let the k th sampling interval be given by

$$\Delta t_k = T N_k \tag{4.45}$$

where N_k is a positive integer.

The estimates at the instants

$$t_k + T,\ t_k + 2T,\ \ldots,\ t_k + (N_k - 1) T$$

will have to be computed as predicted values, based on the filtered estimate at t_k, since no new observations are available. Since Δt_k is varying, the covariance matrix $Q(k)$ should be computed at every stage of filtering. The filtering equations are similar to those for equal sampling interval and are as follows:

$$\hat{\underline{X}}(t_k + i T \mid k) = \phi(T)\, \hat{\underline{X}}(t_k + (i-1) T \mid k) \tag{4.46}$$

$$\text{for } i = 1, 2, \ldots, N_k - 1$$

$$\hat{\underline{X}}(k+1 \mid k) = \phi(T)\, \hat{\underline{X}}(t_k + (N_k - 1) T \mid k) \tag{4.47}$$

$$P(k+1 \mid k) = \phi(\Delta t_k)\, P(k \mid k)\, \phi^T(\Delta t_k) + Q(k) \tag{4.48}$$

$$K(k+1) = P(k+1 \mid k)\, M^T \left[H\, P(k+1 \mid k)\, M^T + \sigma_v^2 \right]^{-1} \tag{4.49}$$

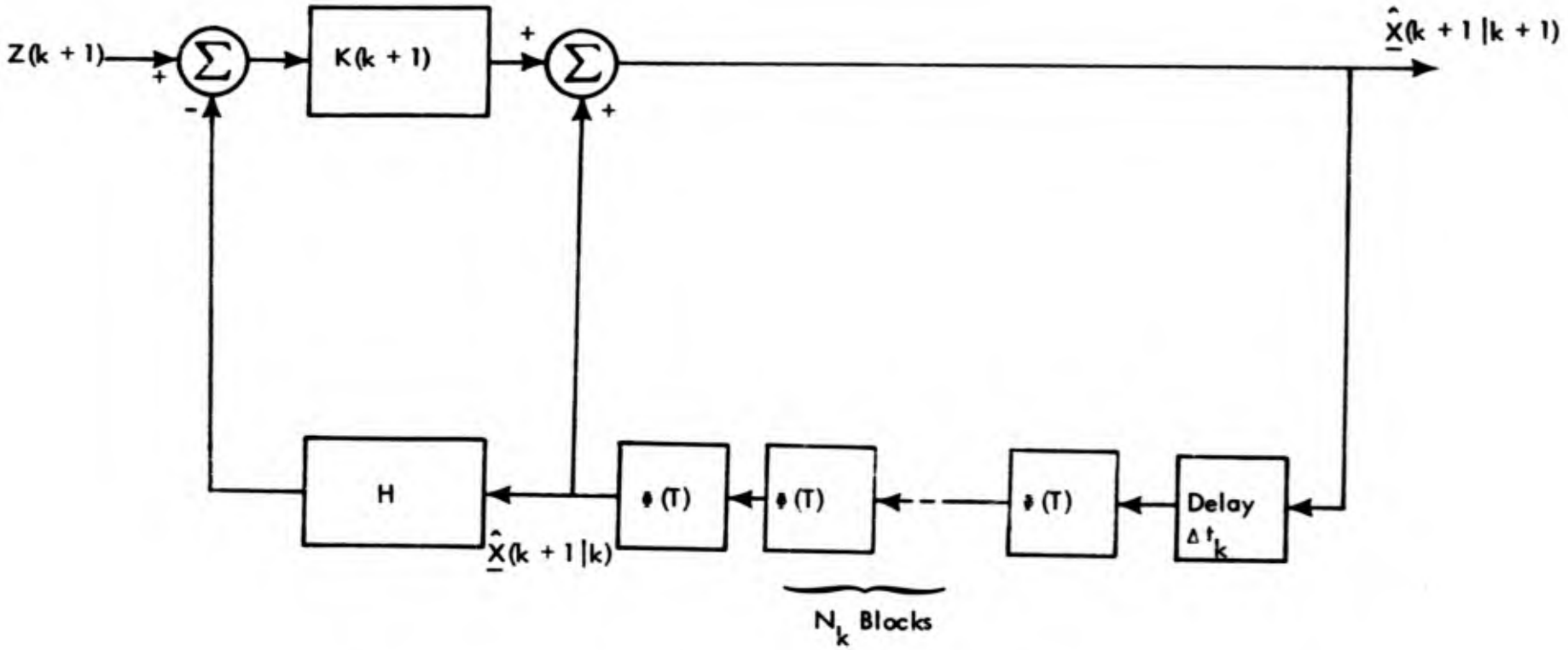


Figure 4.4 Schematic of Kalman Filter (Unequal Sampling Intervals)

$$\hat{\underline{X}}(k+1 \mid k+1) = \hat{\underline{X}}(k+1 \mid k) + K(k+1)\,[Z(k+1) - M\,\hat{\underline{X}}(k+1 \mid k)] \quad (4.50)$$

$$P(k+1 \mid k+1) = [I - K(k+1)\,M]\,P(k+1 \mid k) \quad (4.51)$$

where the various quantities are defined in the same manner as before. The computation algorithm follows the same sequence as the above equations. A block diagram schematic is shown in Figure 4.4.

4.4.3 Initial Conditions

As was noted earlier, initial conditions $\hat{\underline{X}}(1 \mid 1)$ and $P(1 \mid 1)$ are required to start the recursive algorithms. The estimator is initialized as

$$\hat{\underline{X}}(1 \mid 1) = \begin{bmatrix} Z(1) \\ \frac{1}{\Delta t_o}\left(Z(1) - Z(0)\right) \\ 0 \end{bmatrix} \quad (4.52)$$

If $\tilde{\underline{X}}(k \mid k)$ represents the filtering error, then

$$\tilde{\underline{X}}(k \mid k) = \underline{X}(k) - \hat{\underline{X}}(k \mid k) \quad (4.53)$$

and $P(k \mid k)$, the covariance of $\tilde{\underline{X}}(k \mid k)$, is given by

$$P(k \mid k) = E\,[\tilde{\underline{X}}(k \mid k)\;\tilde{\underline{X}}^T(k \mid k)]\,. \quad (4.54)$$

Using equations (4.52), (4.39) and (4.38), one can write

$$\tilde{x}_1(1 \mid 1) = -V(1) \quad (4.55)$$

$$\tilde{x}_2(1 \mid 1) = \frac{1}{\Delta t_o}\,[V(0) - V(1) - w_1(0) + T\,w_2(0)] + x_3(0)\left[\frac{-e^{-\gamma \Delta t_o}}{\gamma} + \frac{1}{\gamma^2 \Delta t_o} - \frac{e^{-\gamma \Delta t_o}}{\gamma^2 \Delta t_o}\right] \quad (4.56)$$

$$\tilde{x}_3(1 \mid 1) = e^{-\nu \Delta t_o} x_3(0) + w_3(0) \tag{4.57}$$

Using these equations and the fact that $E\left[x_3^2(0)\right] = \sigma_m^2$, it can be shown that

$$P(1 \mid 1) = \begin{bmatrix} P_{11} & P_{12} & P_{13} \\ P_{12} & P_{22} & P_{23} \\ P_{13} & P_{23} & P_{33} \end{bmatrix} \tag{4.58}$$

where

$$P_{11} = \sigma_v^2 \tag{4.59}$$

$$P_{12} = \frac{\sigma_v^2}{\Delta t_o} \tag{4.60}$$

$$P_{13} = 0 \tag{4.61}$$

$$P_{22} = \frac{2\sigma_v^2}{(\Delta t_o)^2} + \frac{\sigma_m^2}{\nu^4 (\Delta t_o)^2}\left[2 - \nu^2 (\Delta t_o)^2 + \frac{2\nu^3 (\Delta t_o)^3}{3} - 2e^{-\nu(\Delta t_o)} - 2\nu(\Delta t_o)\, e^{-\nu(\Delta t_o)}\right] \tag{4.62}$$

$$P_{23} = \frac{\sigma_m^2}{\nu^2 (\Delta t_o)}\left[e^{-\nu(\Delta t_o)} + \nu(\Delta t_o) - 1\right] \tag{4.63}$$

and

$$P_{33} = \sigma_m^2 \quad . \tag{4.64}$$

4.5 Summary and Conclusions

Algorithms were presented for implementation of the g-h filter, fixed-memory polynomial filter, and Kalman filter. The g-h filter and the Kalman filter were shown to be recursive filters of the predictor-corrector type, while the fixed-memory filter operates on a fixed length set of samples at each sampling instant. As a consequence, the g-h and Kalman filters must store and operate on only the latest observation, while the fixed-memory filter must store and operate on a set of observations.

Although both are recursive in nature, the g-h and Kalman filters differ greatly in operation. The g-h filter assumes that a constant-velocity trajectory is being observed, and thus uses a constant gain matrix. An expression was derived to show the effect of maneuvering on the position estimate of the g-h filter. On the other hand, the Kalman filter is adaptive to the actual vehicle trajectory in that its gain matrix is a function of the error covariance matrix. Obviously, this feature represents an increase in computation over the g-h filter since the gain matrix will have to be computed at each step.

The fixed-memory filter has an entirely different structure from the g-h and Kalman filters. It was first developed as a combination of orthogonal polynomials, from which smoothing and prediction for equal sampling intervals were computed without matrix inversion. However, since unequal sampling intervals are required for the track and search algorithms described in Chapter 6, a general fixed-memory filter was developed whose implementation necessitates a matrix inversion.

Formulas for the variance of the errors in the estimates were presented for each filter. These are readily available for the fixed-memory and Kalman filters. A recursive relation for computing the variance of the g-h filter is derived in Appendix B.

CHAPTER 5

EVALUATION OF ESTIMATION ALGORITHMS

5.1 Introduction

In this chapter, the four estimation algorithms discussed in the last chapter are evaluated through computer simulations. The simulation program consists of three parts. First, the radar measurements for a selected trajectory are simulated. Then, the target position estimates are computed from the simulated measurements using one of the estimation algorithms. Finally, the mean-squared error between the actual and estimated trajectory is computed and is used as a measure of performance of the estimator. The simulation program is applied to two typical trajectories. The results are evaluated taking into account performance, computation time, and storage requirement. The computer programs were written in FORTRAN and executed on the UNIVAC-1108 located at UAH.

5.2 Simulation Experiments

The simulation experiments for the four estimation algorithms are conducted on two trajectories. Both trajectories are assumed to have constant tangential velocity, one corresponds to a maneuvering target and the other to a non-maneuvering target. Maneuvering is modelled by a randomly varying radial acceleration. The procedure for generating the two trajectories is a part of Appendix A, and the programs are listed in Appendix C.

Sampled values of the target position in the three coordinates are computed at intervals of 0.1 sec and simulated samples of radar measurements are obtained by adding noise. All the simulated radar measurements are stored.

For each trajectory four different sampling schemes will be considered. The first three have constant sampling intervals of 0.1, 0.2 and 0.3 secs, respectively, and the fourth has a variable sampling interval. The selection of the variable sampling intervals are based on the trajectory curvature as discussed in Appendix A. Note that only the simulated measurements designated by the sampling scheme are used in the estimation algorithms.

Regardless of the sampling scheme selected, smoothed estimates and predicted estimates of the three position coordinates are computed at intervals of 0.1 sec. Therefore, three trajectories have been defined: the true trajectory, the smoothed trajectory, and the predicted trajectory. The measure of performance of the algorithm for smoothing and predicting is the mean-squared error (m.s.e.) between true and smoothed trajectory and the mean squared error between true and predicted trajectory, respectively. In each coordinate the m.s.e. of the smoothed and predicted estimates is computed as follows:

$$\sigma_s^2 = \frac{1}{N} \sum_{k=1}^{N} [x_1(k) - \hat{x}_1(k|k)]^2 \tag{5.1}$$

and

$$\sigma_p^2 = \frac{1}{N} \sum_{k=2}^{N} [x_1(k) - \hat{x}_1(k|k-1)]^2 \tag{5.2}$$

where $N = 1000$.

5.3 Simulation Results

Tables 5.1 to 5.7 show the results of the simulation study for the four filters assuming a maneuvering target. Tables 5.8 to 5.14 present the results for a non-maneuvering target. Each table shows the m.s.e. of the smoothed and predicted estimates and the computer time in C.P.U. for the four sampling schemes. This computer time represents the time used by the estimation algorithm to compute 1000 smoothed and predicted estimates in any one coordinate. In the case of the g-h filter, several combinations of the parameters g and h were considered. Since the performance of the two fixed memory polynomial filters is identical except in computer time usage, the results are compiled in the same table. However, the orthogonal polynomial filter cannot be used for variable sampling rates.

5.4 Comparison and Evaluation of Filters

Comparison and evaluation of the filters are based on performance and computer requirements. The performance is defined as the m.s.e. between the actual and estimated trajectories. The computer requirement will be defined as the computer time in C.P.U. because the storage requirements are essentially the same for all the filters considered.

A few remarks apply to all filters. The m.s.e. increases with decrease in sampling rate, as expected. Constant sampling yields a better performance than variable sampling at the same average rate. The difference in tracking performance between maneuvering and nonmaneuvering targets is not as great as expected. This is because the error due to the random noise dominates that due to maneuvering for the selected trajectory.

Sampling time seconds	Mean Squared Error						Computer time-CPU UNIVAC 1108
	Range in m^2		α in $rad^2 \times 10^{-6}$		β in $rad^2 \times 10^{-6}$		
	Smoothing	Prediction	Smoothing	Prediction	Smoothing	Prediction	
0.1	32.0	45.7	15.4	22.0	15.3	21.7	.110
0.2	53.6	67.6	25.9	32.7	26.4	33.3	.067
0.3	95.5	116.1	37.5	45.0	37.4	44.9	.050
Varying	96.7	115.5	37.4	44.1	37.4	44.0	.068

Table 5.1 Maneuvering target, g-h filter (g = 0.3, h = 0.1)

Sampling time seconds	Mean Squared Error						Computer time–CPU UNIVAC 1108
	Range in m^2		α in $rad^2 \times 10^{-6}$		β in $rad^2 \times 10^{-6}$		
	Smoothing	Prediction	Smoothing	Prediction	Smoothing	Prediction	
0.1	39.3	54.8	21.0	28.8	21.0	28.6	.110
0.2	54.0	70.6	25.9	33.4	26.1	33.6	.067
0.3	81.7	104.5	35.9	44.9	35.8	44.8	.050
Varying	77.7	98.0	36.0	44.1	35.9	43.9	.068

Table 5.2 Maneuvering target, g-h filter (g = 0.5, h = 0.1)

Sampling time seconds	Mean Squared Error						Computer time-CPU UNIVAC 1108
	Range in m^2		α in $rad^2 \times 10^{-6}$		β in $rad^2 \times 10^{-6}$		
	Smoothing	Prediction	Smoothing	Prediction	Smoothing	Prediction	
0.1	56.4	74.2	31.3	40.5	31.3	40.4	.110
0.2	67.8	87.2	33.5	42.1	33.5	42.1	.067
0.3	90.0	115.4	45.8	56.7	45.7	56.7	.050
Varying	83.8	106.3	43.9	53.7	43.8	53.5	.068

Table 5.3 Maneuvering target, g-h filter (g = 0.7, h = 0.1)

Sampling time seconds	Mean Squared Error						Computer time-CPU UNIVAC 1108
	Range in m^2		α in $rad^2 \times 10^{-6}$		β in $rad^2 \times 10^{-6}$		
	Smoothing	Prediction	Smoothing	Prediction	Smoothing	Prediction	
0.1	28.7	38.8	13.0	16.5	12.8	16.2	.111
0.2	51.6	64.2	18.4	22.2	18.4	22.2	.067
0.3	89.6	107.6	26.0	30.1	25.5	29.5	.050
Varying	76.7	92.6	26.8	30.5	26.5	30.2	.067

Table 5.4 Maneuvering target, g-h filter ($g = 0.3$, $h = g^2/(2-g)$)

Sampling time seconds	Mean Squared Error						Computer time-CPU UNIVAC 1108
	Range in m^2		α in $rad^2 \times 10^{-6}$		β in $rad^2 \times 10^{-6}$		
	Smoothing	Prediction	Smoothing	Prediction	Smoothing	Prediction	
0.1	42.0	66.1	22.8	36.1	22.8	36.0	.111
0.2	62.3	87.5	31.1	43.6	31.3	43.8	.067
0.3	102.4	136.1	46.7	62.8	46.6	62.7	.050
Varying	96.3	125.5	49.6	64.7	49.5	64.6	.067

Table 5.5 Maneuvering target, g-h filter ($g = 0.5$, $h = g^2/(2 - g)$)

Sampling time seconds	Mean Squared Error						Computer time-CPU UNIVAC 1108
	Range in m^2		α in $rad^2 \times 10^{-6}$		β in $rad^2 \times 10^{-6}$		
	Smoothing	Prediction	Smoothing	Prediction	Smoothing	Prediction	
0.1	65.1	157.9	37.7	93.3	37.7	93.3	0.303 * 6.71
0.2	85.0	132.1	41.8	64.6	41.8	64.7	3.29
0.3	120.7	175.0	53.6	72.8	53.8	73.1	2.19
Varying	91.3	137.5	62.3	93.8	61.4	92.5	3.02

* This value is for the orthogonal polynomial filter.

Table 5.6 Maneuvering target, fixed memory polynomial filter

Sampling time seconds	Mean Squared Error						Computer time-CPU UNIVAC 1108
	Range in m^2		α in $rad^2 \times 10^{-6}$		β in $rad^2 \times 10^{-6}$		
	Smoothing	Prediction	Smoothing	Prediction	Smoothing	Prediction	
0.1	25.9	33.9	8.9	10.6	8.2	9.6	2.7339
0.2	49.5	62.8	15.9	18.4	13.1	14.9	1.5398
0.3	70.9	90.4	24.5	28.5	21.8	25.3	1.1506
Variable	67.5	84.9	27.9	32.0	23.9	27.3	1.4321

Table 5.7 Maneuvering Target, Kalman Filter

Sampling time seconds	Mean Squared Error						Computer time–CPU UNIVAC 1108
	Range in m^2		α in $rad^2 \times 10^{-6}$		β in $rad^2 \times 10^{-6}$		
	Smoothing	Prediction	Smoothing	Prediction	Smoothing	Prediction	
0.1	30.5	43.1	15.3	21.8	15.4	21.8	.110
0.2	48.5	60.9	25.7	32.4	25.8	32.4	.067
0.3	84.9	103.8	36.6	43.9	36.6	43.9	.050
Varying	108.5	124.9	65.6	75.4	65.6	75.4	.068

Table 5.8 Nonmaneuvering target, g-h filter (g = 0.3, h = 0.1)

Sampling time seconds	Mean Squared Error						Computer time-CPU UNIVAC 1108
	Range in m^2		α in $rad^2 \times 10^{-6}$		β in $rad^2 \times 10^{-6}$		
	Smoothing	Prediction	Smoothing	Prediction	Smoothing	Prediction	
0.1	38.4	52.2	21.0	28.7	21.0	28.7	.110
0.2	50.2	64.3	25.8	33.1	25.8	33.1	.067
0.3	72.3	92.1	35.3	44.0	35.3	44.0	.050
Varying	89.4	107.8	53.7	64.7	53.7	64.7	.068

Table 5.9 Nonmaneuvering target, g-h filter (g = 0.5, h = 0.1)

Sampling time seconds	Mean Squared Error						Computer time–CPU UNIVAC 1108
	Range in m^2		α in $rad^2 \times 10^{-6}$		β in $rad^2 \times 10^{-6}$		
	Smoothing	Prediction	Smoothing	Prediction	Smoothing	Prediction	
0.1	55.9	71.7	31.3	40.4	31.3	40.4	.110
0.2	65.2	81.6	33.4	41.8	33.4	41.8	.067
0.3	82.9	104.1	45.4	56.1	45.4	56.1	.050
Varying	99.2	120.4	59.6	72.3	59.6	72.3	.068

Table 5.10 Nonmaneuvering target, g-h filter ($g = 0.7$, $h = 0.1$)

Sampling time seconds	Mean Squared Error						Computer time-CPU UNIVAC 1108
	Range in m^2		α in $rad^2 \times 10^{-6}$		β in $rad^2 \times 10^{-6}$		
	Smoothing	Prediction	Smoothing	Prediction	Smoothing	Prediction	
0.1	25.0	32.2	12.8	16.2	12.8	16.2	.111
0.2	38.7	46.9	17.7	21.1	17.7	21.1	.067
0.3	56.4	67.8	23.9	27.5	23.9	27.5	.050
Varying	71.2	81.6	37.2	41.9	37.9	41.9	.067

Table 5.11 Nonmaneuvering target, g-h filter ($g = 0.3$, $h = g^2/(2 - g)$)

Sampling time seconds	Mean Squared Error						Computer time-CPU UNIVAC 1108
	Range in m^2		α in $rad^2 \times 10^{-6}$		β in $rad^2 \times 10^{-6}$		
	Smoothing	Prediction	Smoothing	Prediction	Smoothing	Prediction	
0.1	41.5	64.8	22.8	36.4	22.8	36.1	.111
0.2	60.1	83.9	31.1	43.5	31.1	43.5	.067
0.3	98.2	131.1	46.4	62.3	46.4	62.3	.050
Varying	129.1	162.5	79.1	99.7	79.1	99.7	.067

Table 5.12 Nonmaneuvering target, g-h filter ($g = 0.5$, $h = g^2/(2-g)$)

Sampling time seconds	Mean Squared Error						Computer time–CPU UNIVAC 1108
	Range in m^2		α in $rad^2 \times 10^{-6}$		β in $rad^2 \times 10^{-6}$		
	Smoothing	Prediction	Smoothing	Prediction	Smoothing	Prediction	
0.1	65.3	157.7	37.7	93.3	37.7	93.3	0.303 * 6.63
0.2	84.3	130.5	41.9	64.7	41.8	64.7	3.29
0.3	108.5	148.5	53.0	71.4	53.0	71.4	2.18
Varying	105.6	129.9	64.1	80.0	64.1	80.0	1.47

* This value is for the orthogonal filter.

Table 5.13 Nonmaneuvering trajectory, fixed memory polynomial filter

Sampling time seconds	Mean Squared Error						Computer time-CPU UNIVAC 1108
	Range in m^2		α in $rad^2 \times 10^{-6}$		β in $rad^2 \times 10^{-6}$		
	Smoothing	Prediction	Smoothing	Prediction	Smoothing	Prediction	
0.1	25.3	32.6	7.3	8.3	7.3	8.3	2.8496
0.2	46.6	58.1	12.5	13.9	12.5	13.9	1.5189
0.3	67.1	85.2	18.8	21.2	18.8	21.2	1.1341
Varying	101.6	123.9	27.6	31.2	27.6	31.2	0.9955

Table 5.14 Nonmaneuvering Target, Kalman Filter

The performance of the g-h filter depends on the choice of the parameters g and h. Many combinations of these parameters were tested and the results confirm that the selection of the parameter h, based on equation (4.7), is near optimum. Only the results for the five best combinations are tabulated. The g-h filter is best for nonmaneuvering targets in both performance and computational speed.

The two fixed-memory filters did not perform as well as the other filters. For a given constant sampling rate, the orthogonal polynomial filter is much faster than the general polynomial filter in computing the same estimate. It is also faster than the Kalman filter. However, the orthogonal polynomial filter cannot be used with variable sampling rates.

The performance of the Kalman filter is the best for maneuvering targets and is near that of the best g-h filter for a nonmaneuvering target. The Kalman filter requires a relatively large computer time, but it provides the covariance matrices of the estimates at each step, which is necessary for the tracking allocation.

CHAPTER 6

PROCEDURE FOR EAR TRACK AND SEARCH ALLOCATION

6.1 Introduction

This chapter develops a track and search allocation schema for the EAR system. It is a simplification of the general resource allocation procedure of Chapter 2 and uses the tracking algorithms derived in Chapter 4. A tracking measure based on rms angular errors is developed and a block diagram for the allocation schema is presented. An example shows how the tracking algorithm is used to compute the rms angular errors of the actual and predicted estimates.

6.2 Tracking Measure for the EAR System

In Chapter 2, we considered the overall requirements for resource allocation in a modern air defense system. However, an appropriate surface-to-air interceptor has not been selected for the EAR system, and no specific deployments have been investigated. Therefore, the definitions for Object Worth and Object Knowledge provided in Chapter 2 must be altered to provide a suitable tracking measure for the current, non-tactical EAR system. This can be accomplished simply by removing the interceptor and deployment related parameters from the defining equations, viz., the predicted rms miss distance d, the Potential Damage to the Defense PDD, and the Potential of a Successful Intercept PSI. The restricted definitions are given by

$$OW(t,i,k) = OK(t,i,k) = \frac{a_1}{e(t,i,k)} \tag{6.1}$$

where $e(t,i,k)$ is the estimated angular tracking error computed by the tracking algorithm.

Except for the simplified computation of object worth, the procedure described in Chapter 2 is valid. Equations (2.5), (2.6), and (2.7) of Chapter 2 provide the bases for allocating radar resources to the track function. The Fictitious Incremental Gain, FIG(i,k), computed from the restricted Object Worths in (6.1), defines a useful tracking measure for the non-tactical EAR system. Clearly, the concepts can be readily extended to include the defensive weapon and deployment geometry of a future, tactical EAR system by using the original definitions in Section 2.

6.3 EAR Resource Allocation Schema

6.3.1 Description of General EAR Schema

The EAR Resource Allocation Schema is shown in Figure 6.1. It is similar to the general resource allocation schema shown in Figure 2.2. The main difference is that, in the EAR, only one cycle is allocated, i.e., k takes only the values 0 and 1.

The resource allocation schema consists of four steps: 1) computation and updating of a track file which contains, for each target being tracked, the smoothed state, the predicted state, the actual object worth $OW(t_s,i,0)$, and the two predicted object worths, with and without allocation, $OW(t_s + \Delta t,i,1)$ and $OW(t_s + \Delta t,i,0)$, 2) computation of the actual and potential system worth for each choice of track allocation, 3) selection of the best track allocation, and 4) decision between best track allocation and search. The allocation algorithms are given by (6.1), (2.5), (2.6), and (2.7).

Several tracking algorithms were developed in Chapter 4 and evaluated in Chapter 5. The tracking algorithm receives the radar measurements and computes the estimated and predicted positions as well as the errors in the estimates. The object worth is easily computed from the rms error.

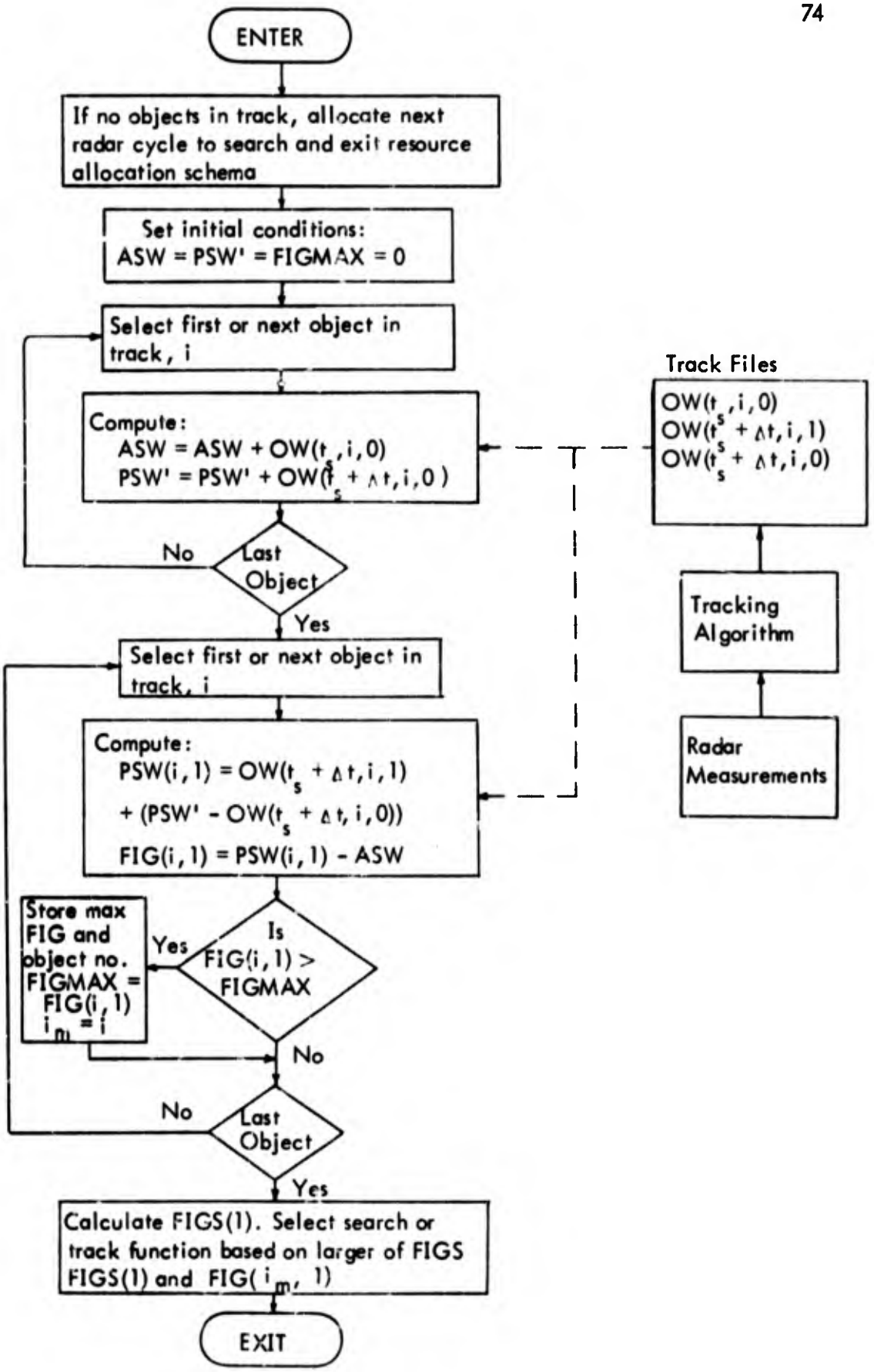


Figure 6.1 General EAR Resource Allocation Schema

Once the optimum allocation for track has been determined by the maximum FIG(i,1), one must decide if search should be preferred. For this purpose, a Fictitious Incremental Gain for search FIGS(1) can be defined. The average time for search must be a function of the number of targets being tracked, and should become zero when high priority tracking saturation occurs. It is recommended that the average value of FIGS decreases linearly,

$$FIGS(1) = A(1 - \frac{N}{N_s}) , \qquad (6.2)$$

where N is the actual number of targets, N_s is the saturation number, and A is an adaptive scaling factor which is corrected so that the average search rate is 100% for zero targets and 0% for N_s targets.

6.3.2 Computer Time Considerations

The resource allocation schema described in Section 6.3.1 requires a minimum amount of storage, but the computation time can be reduced if necessary. In particular, the target which has just received a track allocation does not need to be included in the System Worth computation for the next several radar cycles. That is, there is a minimum tracking update interval which is a function of the maneuvering capability of the target. For example, the baseline EAR system performs track updating on individual targets at a 0.1 sec. time interval, i.e. ten radar cycles.

This consideration of minimum track rate will automatically leave an average time for search which decreases with the number of targets. Thus, the FIGS computation and comparison to FIG can be eliminated.

6.4 Example

An example illustrates the computation of object worth starting from the radar measurements, using the Kalman filter algorithm discussed in Section 4.4.

The Kalman filter includes a recursive algorithm for the computation of the covariance matrix of the smoothed and predicted angle estimates. The covariance matrices are denoted by the letter P. Assuming that the covariance matrix $P(k|k)$ has been computed, the recursive relations (4.41) and (4.44) yield the covariance matrices for prediction and smoothing, $P(k+1|k)$ and $P(k+1|k+1)$.

The actual and predicted (with or without allocation) rms angular errors for the i th target are given by

$$e(t_s, i, 0) = \sqrt{P_{11}(k|k)}$$

$$e(t_s + \Delta t, i, 1) = \sqrt{P_{11}(k+1|k+1)}$$

$$e(t_s + \Delta t, i, 0) = \sqrt{P_{11}(k+1|k)}$$

where P_{11} is the first element of the matrix P, k corresponds to t_s, and k + 1 to $t_s + \Delta t$. The corresponding object worths are given by

$$OW(t_s, i, 0) = a_1 \Big/ e(t_s, i, 0)$$

$$OW(t_s + \Delta t, i, 1) = a_1 \Big/ e(t_s + \Delta t, i, 1)$$

$$OW(t_s + \Delta t, i, 0) = a_1 \Big/ e(t_s + \Delta t, i, 0) \quad .$$

CHAPTER 7

SUMMARY AND CONCLUSIONS

System analysts at MICOM presently have the need for a measure of quality for the evaluation and comparison of defense systems engaging a complex mix of targets. This effort is aimed at satisfying this need. The main goals were: 1) to develop a general procedure for radar resource allocation, 2) to implement and evaluate candidate tracking algorithms to be used with the allocation procedure, 3) to define a tracking measure, and 4) to develop a track and search resource allocation procedure.

A general procedure for radar resource allocation was developed in Chapter 2. A quantitative measure of defense posture (System Worth) based on all available information and an optimization criteria for selecting best resource allocation strategy were defined. These concepts are applicable to any type of defense system.

Four estimation algorithms commonly used for tracking were investigated. Algorithms were defined for the computation of the smoothed and predicted estimates. Formulas for the error covariance matrices are presented for each filter for use with the resource allocation scheme. These are readily available for the fixed memory and Kalman filters, but it was necessary to derive a relation for computing the variance of the g-h filter.

A system model and a procedure for generating maneuvering target trajectories were developed and used to evaluate the performance and computer requirements of the four filters. The four filters were compared to provide guidelines for

selection of a tracking filter to be used in the EAR system. Comparison was based on the performance of the filter and its computational requirements. The mean squared error in the smoothed and predicted estimates was used as a measure of performance.

The g-h filter is recommended for nonmaneuvering targets because it is most accurate and requires least computer time. The Kalman filter performs very well for maneuvering targets and it provides, for each step, a covariance matrix which can be used in the track and search allocation scheme. The g-h filter also performed well for the maneuvering target and is an order of magnitude faster than the Kalman filter. However, since the results are based on only one particular type of maneuver, a further comparison of the g-h and Kalman filters is required before a specific recommendation can be made. The fixed-memory polynomial filter is always inferior to the g-h filter and, thus, did not prove to be an acceptable choice.

A tracking measure was defined for the nontactical EAR system. It is based on the covariance of the smoothed and predicted estimate. This tracking measure can be readily extended to become a measure of defense posture for a future tactical EAR system.

Finally, a scheme for track and search allocation for the EAR system was presented. It is a simplification of the more general resource allocation scheme discussed earlier. An example illustrated how the tracking measure is computed and used in the track and search allocation procedure.

APPENDIX A

TRAJECTORY GENERATION

A.1 Introduction

In order to study the behavior of the different tracking algorithms, it is necessary to generate a trajectory which is representative of a vehicle in motion. Two different trajectories, one with a constant velocity and one which includes target maneuvers superimposed on a constant velocity path, are considered in order to amplify the advantages of the different type filters. Also included in this appendix is a brief description of the selection of sampling intervals.

A.2 Generation of Trajectory with Maneuver

The maneuver will have a distribution and correlation as discussed in Chapter 3. The random numbers, making up the random maneuvering, are generated with the required distribution and correlation by following the procedure described by Holliday [11]. The values used for A_{max} and γ are $60\ m/sec^2$ and 0.1, respectively. These random numbers are denoted by $a(k)$ for $k = 1,2,\ldots$.

For purposes of trajectory generation, it is assumed that the vehicle has a constant elevation of $Z = 10{,}000$ m and a constant tangential velocity of $V = 200$ m/sec. At the kth instant let x_{1k} and x_{2k} be the position of the vehicle with respect to an orthogonal coordinate system in the plane defined by the constant elevation. Also, let the direction of motion at the k th instant be defined by the unit normal vector N_k (Figure A.1). The acceleration at this instant is equal to $a(k)$. A positive acceleration is assumed to result in a counterclockwise rotation whereas a negative acceleration results in a clockwise rotation.

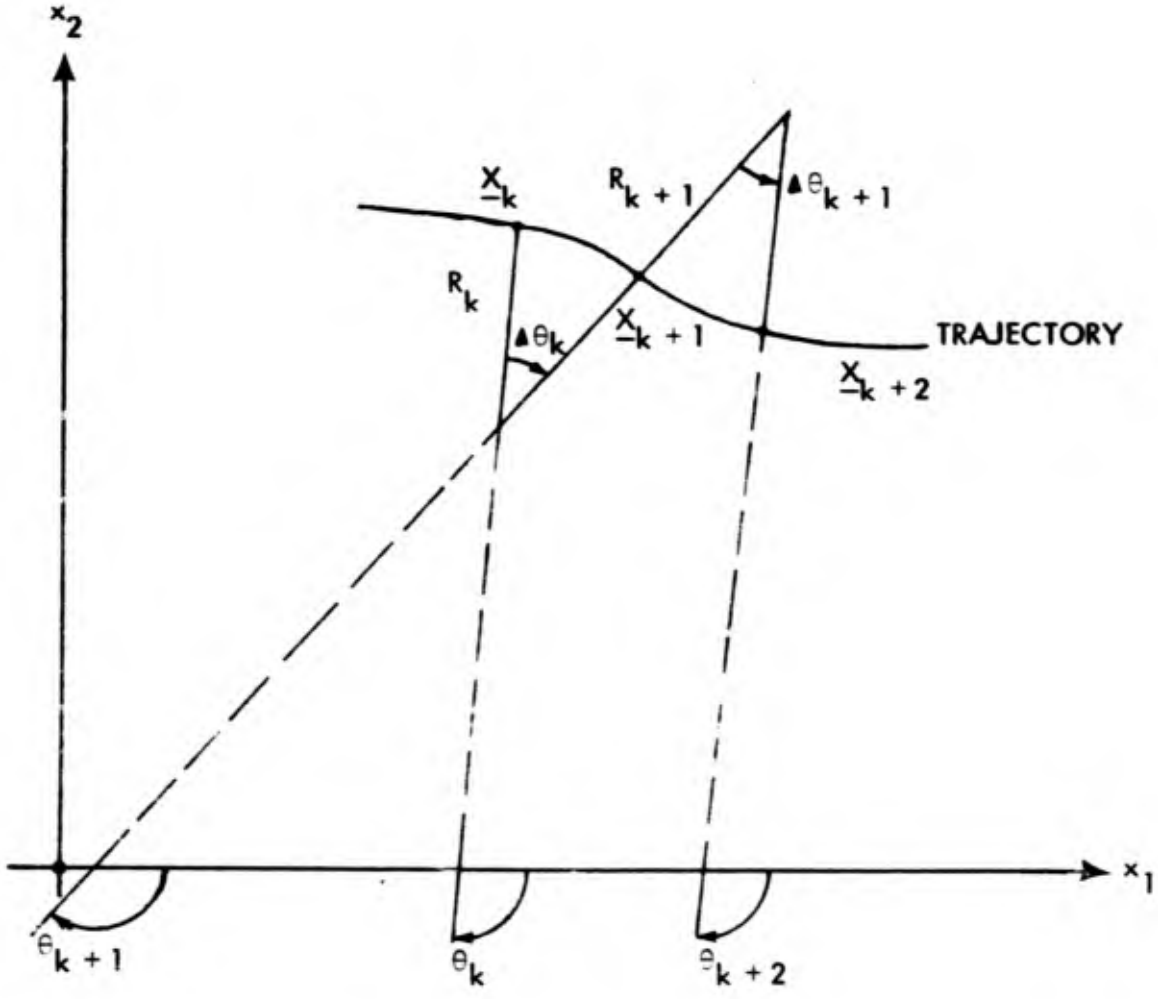


Figure A.1 Trajectory Generation

Let $\underline{X}_k$ denote the position vector at the k th instant. This vector is computed at every sampling instant, assuming a uniform sampling interval of T sec, using the following relations

$$\omega_k = \frac{a(k)}{V} \tag{A.1}$$

$$R_k = \frac{V}{W_k} \tag{A.2}$$

$$\Delta\theta_k = T\,\omega_k \tag{A.3}$$

$$\underline{N}_{k+1} = \underline{N}_k \text{ rotated by } \Delta\theta_k \tag{A.4}$$

$$\underline{X}_{k+1} = \underline{X}_k + R_k\,(\underline{N}_k - \underline{N}_{k+1}) \tag{A.5}$$

Unequal sampling intervals are obtained by selecting samples at random muitiples of T as input to the filters. Equation (A.4) is easily implemented by defining θ_k such that

$$\underline{N}_k = \begin{pmatrix} \cos\theta_k \\ \sin\theta_k \end{pmatrix} \tag{A.6}$$

and noting that $\theta_{k+1} = \theta_k + \Delta\theta_k$. (A.7)

The range at k th instant is computed from

$$R_a(k) = \sqrt{x_{1k}^2 + x_{2k}^2 + Z^2} \ . \tag{A.8}$$

The two angles α and β (Figure 3.1) are computed using (assuming the x,y coordinates of Figure 3.1 and x_1, x_2 coordinates of Figure A.1 are parallel)

$$\cos \alpha(k) = x_{1k} / R_a(k) \tag{A.9}$$

$$\cos \beta(k) = x_{2k} / R_a(k) \quad , \tag{A.10}$$

The observations are generated by adding observation noise to $R_a(k)$, $\alpha(k)$ and $\beta(k)$.

A.3 Generation of Nonmaneuvering Trajectory

Here it is assumed that the target has constant elevation, and has constant velocities V_x and V_y in the x_1, x_2 coordinates. If

$$\underline{X}_k = \begin{bmatrix} x_{1k} \\ x_{2k} \end{bmatrix} \tag{A.11}$$

denotes the position vector at the kth instant, then the position vector at the $k+1$th sampling instant, assuming a uniform sampling interval of T sec, is computed from

$$\underline{X}_{k+1} = \underline{X}_k + \begin{bmatrix} T V_x & x_{1k} \\ T V_y & x_{2k} \end{bmatrix} \tag{A.12}$$

For simulation purposes the values used for V_x and V_y are both equal to 150 m/sec. The rest of the computation is exactly similar to that for the maneuvering trajectory.

A.4 Selection of Sampling Intervals

It is not necessary to have the same sampling interval when the target is not maneuvering as when it is maneuvering. Thus, the sampling interval should be inversely proportional to the amount of maneuvering. For simulation purposes, the sampling interval is selected on the basis of the angle subtended at the radar by two consecutive positions of the target.

Let $\underline{X}'_k$ denote the position vector of the target at time $t = kT$ where

$$\underline{X}'_k = \begin{bmatrix} x_{1k} \\ x_{2k} \\ Z \end{bmatrix} . \qquad (A.13)$$

Then the angle $\Delta\theta$ subtended at the radar between $\underline{X}'_k$ and $\underline{X}'_{k+1}$ is given approximately by the magnitude of their cross-product. If $t = kT$ is the current measurement instant, then the next measurement instant is given by

$$t = (k + \ell)\, T$$

where ℓ is inversely proportional to $\Delta\alpha$. The following scheme is used in the simulation program

$$\ell = 5 - \left[\frac{\Delta\alpha - 0.001}{0.2}\right] \quad \text{for } \Delta\alpha \leq 0.0018$$

and $\ell = 1$ for $\Delta\alpha > 0.0018$ where [ ] stands for "integer part of".

APPENDIX B

ERROR COVARIANCE OF THE g-h FILTER

In this appendix, a set of equations is derived to compute recursively the error covariance in the g-h filter which was discussed in Chapter 4. For the sake of completeness, all relevant equations are repeated.

B.1 Derivation of Error Covariance Equations

Following the system model described in Chapter 3, the system can be approximated by

$$\underline{X}(k+1) = \Psi_k \underline{X}(k) + G_k a(k) \tag{B.1}$$

where

$$\underline{X}(k) = \begin{bmatrix} x_1(k) = \text{target position at } k\text{th sampling instant} \\ x_2(k) = \text{target position rate at } k\text{th instant} \end{bmatrix} \tag{B.2}$$

$$G_k = \begin{bmatrix} 0 \\ \Delta t_k \end{bmatrix} \tag{B.3}$$

$$\Psi_k = \begin{bmatrix} 1 & \Delta t_k \\ 0 & 1 \end{bmatrix} \tag{B.4}$$

and $a(k)$ is the maneuver variable which has zero mean and a correlation function given by

$$E[a(k)\ a(k-i)] = \sigma_m^2\ e^{-\nu(t_k - t_{k-i})} \quad . \tag{B.5}$$

The observation sequence is given by

$$Z(k) = M \underline{X}(k) + V(k) \tag{B.6}$$

where

$$M = [1 \quad 0] \tag{B.7}$$

and $V(k)$ is an independent Gaussian noise sequence with zero mean and a mean squared value of σ_v^2 .

From Chapter 4, the estimator equations are

$$\hat{\underline{X}}(k+1 \mid k) = \Psi_k \hat{\underline{X}}(k \mid k) \tag{B.8}$$

$$\hat{\underline{X}}(k+1 \mid k+1) = \hat{\underline{X}}(k+1 \mid k) + K_k [Z(k+1) - M \hat{\underline{X}}(k+1 \mid k)] \tag{B.9}$$

where

$$K_k = \begin{bmatrix} g \\ h/\Delta t_k \end{bmatrix} \tag{B.10}$$

The covariance matrices of the errors in the filter and the predictor respectively are defined as

$$P(k \mid k) = E[\tilde{\underline{X}}(k \mid k) \tilde{\underline{X}}^T(k \mid k)] \tag{B.11}$$

$$P(k+1 \mid k) = E[\tilde{\underline{X}}(k+1 \mid k) \tilde{\underline{X}}^T(k+1 \mid k)] \tag{B.12}$$

where

$$\tilde{\underline{X}}(k \mid k) = \underline{X}(k) - \hat{\underline{X}}(k \mid k) \tag{B.13}$$

and $$\tilde{\underline{X}}(k+1 \mid k) = \underline{X}(k+1) - \hat{\underline{X}}(k+1 \mid k) \, . \tag{B.14}$$

Combining the equations of the system and the estimator, one can write,

$$\tilde{\underline{X}}(k+1 \mid k) = \Psi_k \, \tilde{\underline{X}}(k \mid k) + G_k \, a(k) \tag{B.15}$$

$$\tilde{\underline{X}}(k+1 \mid k+1) = B_k \, \tilde{\underline{X}}(k+1 \mid k) - K_k \, V(k+1) \tag{B.16}$$

where

$$B_k = (I - K_k M) \, . \tag{B.17}$$

Using equations (B.15) and (B.16) and noting that $V(k+1)$ is independent of $\tilde{\underline{X}}(k+1 \mid k)$, the following expressions can be written,

$$P(k+1 \mid k) = \Psi_k P(k \mid k) \, \Psi_k^T + \sigma_m^2 \, G_k \, G_k^T + \Psi_k C(k) \, G_k^T + G_k C^T(k) \, \Psi_k^T \tag{B.18}$$

$$P(k+1 \mid k+1) = B_k P(k+1 \mid k) \, B_k^T + \sigma_v^2 \, K_k \, K_k^T \tag{B.19}$$

where

$$C(k) = E \left[a(k) \; \tilde{\underline{X}}(k \mid k) \right] \, . \tag{B.20}$$

Initial conditions $P(1 \mid 1)$, required to start equation (B.18), are derived next. From Chapter 4, the initial conditions of the estimator are

$$\hat{\underline{X}}(1 \mid 1) = \begin{bmatrix} Z(1) \\ \frac{1}{\Delta t_o} \left[Z(1) - Z(0) \right] \end{bmatrix} \, . \tag{B.21}$$

Therefore,

$$\tilde{\underline{X}}(1 \mid 1) = \begin{bmatrix} x_1(1) - x_1(1) - V(1) \\ x_2(0) + \Delta t_o\, a(0) - \frac{1}{\Delta t_o}[x_1(0) + \Delta t_o x_2(0) + V(1) - x_1(0) - V(0)\,] \end{bmatrix}$$

$$= \begin{bmatrix} V(1) \\ \Delta t_o\, a(0) - \frac{1}{\Delta t_o}[V(1) - V(0)\,] \end{bmatrix} \qquad (B.22)$$

It follows that

$$P(1 \mid 1) = \begin{bmatrix} \sigma_v^2 & \frac{\sigma_v^2}{\Delta t_o} \\ \frac{\sigma_v^2}{\Delta t_o} & \frac{2\sigma_v^2}{\Delta t_o} + \sigma_m^2\, \Delta t_o \end{bmatrix} \qquad (B.23)$$

which is the necessary initial condition.

B.2 Recursive Relations for C(k)

In order to implement equation (B.18), it is necessary to compute the vector C(k). This can be done as follows,

Equations (B.15) and (B.16) can be combined to yield

$$\tilde{\underline{X}}(k+1 \mid k+1) = B_k \Phi_k \tilde{\underline{X}}(k \mid k) + B_k G_k a(k) - K_k V(k+1)\ . \qquad (B.24)$$

A solution to the above equation can be written as

$$\tilde{\underline{X}}(k+1\,|\,k+1) = \left(\prod_{j=1}^{k} B_j \psi_j\right) \tilde{\underline{X}}(1\,|\,1)$$

$$+ \sum_{i=1}^{k-1} \left(\prod_{j=i+1}^{k} B_j \psi_j\right) \left(B_i G_i\, a(i) - K_i V(i+1)\right)$$

$$+ B_k G_k\, a(k) - K_k V(k+1) \qquad \text{(B.25)}$$

where

$$\prod_{j=i+1}^{k} B_j \psi_j = \left(B_k \psi_k\right)\left(B_{k-1} \psi_{k-1}\right) \cdots\cdots \left(B_{i+1} \psi_{i+1}\right) . \qquad \text{(B.26)}$$

From equations (B.20) and (B.26),

$$C(k+1) = \left(\prod_{j=1}^{k} B_j \psi_j\right) E\,[a(k+1)\, \tilde{\underline{X}}(1\,|\,1)\,]$$

$$+ \sum_{i=1}^{k-1} \left(\prod_{j=i+1}^{k} B_j \psi_j\right) B_i G_i\; E\,[a(k+1)\, a(i)\,]$$

$$+ B_k G_k\; E\,[a(k+1)\, a(k)\,] . \qquad \text{(B.27)}$$

By definition,

$$E\,[a(k+1)\, a(i)\,] = \sigma_m^2\, e^{-\gamma(t_{k+1} - t_i)} , \qquad \text{(B.28)}$$

and using equation (B.22), one obtains

$$E\left[a(k+1)\ \tilde{\underline{X}}(1\mid 1)\right]=\begin{bmatrix} 0 \\ \sigma_m^2\ \Delta t_o\ e^{-\gamma(t_{k+1}-t_o)} \end{bmatrix} . \tag{B.29}$$

Therefore, equation (B.20) reduces to,

$$C(k+1)=\left(\prod_{j=1}^{k} B_j \psi_j\right)\begin{bmatrix} 0 \\ \sigma_m^2\ \Delta t_o\ e^{-\gamma(t_{k+1}-t_o)} \end{bmatrix}$$

$$+\sum_{i=1}^{k-1}\left(\prod_{j=i+1}^{k} B_j \psi_j\right) B_i G_i\ \sigma_m^2\ e^{-\gamma(t_{k+1}-t_i)}$$

$$+ B_k G_k\ \sigma_m^2\ e^{-\gamma\Delta t_k}$$

$$= e^{-\gamma\Delta t_k} B_k \psi_k\left[\left(\prod_{j=1}^{k} B_j \psi_j\right)\begin{bmatrix} 0 \\ \sigma_m^2\ \Delta t_o\ e^{-\gamma(t_k-t_o)} \end{bmatrix}\right.$$

$$+\sum_{i=1}^{k-2}\left(\prod_{j=i+1}^{k-1} B_j \psi_j\right) B_i G_i\ \sigma_m^2\ e^{-\gamma(t_k-t_i)}$$

$$+ B_{k-1} G_{k-1} \sigma_m^2 e^{-\gamma \Delta t_{k-1}} \Big] + B_k G_k \sigma_m^2 e^{-\gamma \Delta t_k}$$

$$= e^{-\gamma \Delta t_k} [B_k \psi_k C(k) + B_k G_k \sigma_m^2] . \qquad \text{(B.30)}$$

Equation (B.30) can be used recursively to generate $C(k + 1)$ starting with the initial condition

$$C(1) = E [a(1) \tilde{\underline{X}}(1 \mid 1)]$$

$$= \begin{bmatrix} 0 \\ \sigma_m^2 \Delta t_o e^{-\gamma \Delta t_o} \end{bmatrix} . \qquad \text{(B.31)}$$

APPENDIX C

COMPUTER PROGRAMS

Computer programs for the generation of the trajectories and for the four filters are listed in this appendix along with brief descriptions. All the programs are coded in FORTRAN IV.

C.1 Subroutine TRAJ

C.1.1 Symbols Used

XVEL, YVEL	Velocities for nonmaneuvering trajectory
DELT	Estimation interval
ELEV	Elevation of the vehicle
ALPHA, ACCMAX, PMAX, PZERO	Parameters γ, A_{max}, P_{max} and P_o of the maneuver variable
NPOINT	Number of estimation points (DELT apart)
XN	Array containing the values for the maneuver variable
KPOINT	Size of array XN
KEQUNQ	1 for equal sampling intervals 2 for unequal sampling intervals
Z	Array of observations
X	Array of actual position values
KTIME	Array showing the number of estimation intervals between observations for variable sampling
RVAR	Observation noise covariance σ_v^2 for range measurement
RVARB	Observation noise covariance σ_v^2 for angle measurement
RNOISE	Range observation noise sequence

RNOISB	Angle observation noise sequence
XP, YP	X and Y coordinates of the target
VEL	Constant tangential velocity of the maneuvering target
THETA1	Angle subtended by the normal with the X-axis
KTRJ	1 for maneuvering trajectory 2 for nonmaneuvering trajectory
KOPT	1 for range measurement 2 for measurement of α 3 for measurement of β

C.1.2 Description of the Program

The calling sequence of this subroutine is

CALL TRAJ(KOPT,KTRJ) .

Every time the subroutine is called, one set of measurements (range or one of the angles) is generated. The observations are always obtained at intervals equal to the estimation interval. In order not to duplicate computation, it is required that the range measurements (KOPT = 1) be generated first, followed by the two angle measurements (KOPT = 2,3). Setting KTRJ = 1 produces the maneuvering trajectory, while KTRJ = 2 generates the non-maneuvering trajectory. If unequal sampling is desired (KEQUNQ = 2) the elements of the array KTIME are also computed.

C.1.3 Input and Output

The initial values of THETA1, XP and YP and the magnitude of VEL are the inputs to this subroutine. These are provided through DATA statements in the subroutine. Other variables such as XVEL, YVEL, DELT, ELEV, ALPHA, RVAR, RVARB, ACCMAX, NPOINT, KPOINT, PMAX, PZERO and KEQUNQ are assigned values

in the main program and transferred to TRAJ through common blocks.

The outputs of the subroutine are contained in the arrays Z, X and KTIME. Array Z contains the measurement, the actual values of the position are stored in array X and the array KTIME is useful only in the case of unequal sampling intervals. These outputs are transferred to the main program through a common block.

C.1.4 Subprograms Used

(1) XNOISE: The calling sequence is

CALL XNOISE(X) .

This subprogram computes KPOINT values of the maneuver variable and stores them in the array X. Other maneuver variables ALPHA, ACCMAX, PMAX, and PZERO are transferred through common blocks. This subprogram uses subroutines RANDU to generate a set of uniformly distributed samples. The required correlation is then injected into this sequence by using a difference relation. An integer array with KPOINT integers in natural order is then sorted with respect to this random array using subroutine SORTL, yielding a new integer array I. Subroutine DISTFN is used to generate an array X with the desired distribution. The array X is then sorted with respect to the array I using subroutine ISORTL to yield the required values for the maneuver variable. Reference [11] contains the detailed development of generation of a set of random numbers with desired distribution and correlation.

(2) RANDN: The calling sequence is

CALL RANDN (X,N,XM,STD) .

This subprogram computes N samples of a normal deviate with mean equal to XM and standard deviation equal to STD and stores them in the array X. Subroutine RANDN is available in most computer libraries and is not listed here.

(3) CROSPR: The calling sequence is

CALL CROSPR (XA,XB,XC,YA,YB,YC,DELANG)

This subprogram computes the cross product between two unit vectors which are parallel to the two vectors defined by XA, XB, XC and YA, YB, YC.

```
 1*            SUBROUTINE TRAJ(KOPT,KTRJ)
 2*            COMMON/BLKA/XVEL,YVEL,DELT,ELEV,KEGUNO,KDELT
 3*            COMMON/BLKB/ALPHA,KVAR,PVARB,ACCMAX,NPOINT,KPOINT,PMAX,PZERO
 4*            COMMON/BLKC/Z(1002),X(1002),KTIME(1000)
 5*            DIMENSION RNOISE(1002),XN(2500),RNOISE(1002)
 6*            DIMENSION XP(1002),YP(1002)
 7*            REAL NX1,NX2,NY1,NY2
 8*            DATA THETA1/1.570796/
 9*            DATA VEL/200.0/
10*            IF(KOPT.NE.1) GO TO 3
11*            XP(1)=3000.0
12*            YP(1)=3000.0
13*            NPONT1=NPOINT-1
14*            IF(KTRJ.EQ.2) GO TO 10
15*            NX1=COS(THETA1)
16*            NY1=-SIN(THETA1)
17*            CALL XNOISE(XN)
18*            ACCOLD=1.0
19*            EPS=1.0E-4
20*            TPI=2.0*3.141592
21*            THETA1=0.5*3.141592
22*            DO 1 K=1,NPONT1
23*            ACCL=XN(K)
24*            IF(ABS(ACCL).LT.EPS) ACCL=SIGN(EPS,ACCOLD)
25*            ACCOLD=ACCL
26*            OMEGA=ACCL/VEL
27*            OMEGAT=OMEGA*DELT
28*            RADIUS=-VEL/OMEGA
29*            THETA2=THETA1-OMEGAT
30*            IF(THETA2.GT.TPI) THETA2=THETA2-TPI
31*            NX2=COS(THETA2)
32*            NY2=-SIN(THETA2)
33*            XP(K+1)=XP(K)+RADIUS*(NX1-NX2)
34*            YP(K+1)=YP(K)+RADIUS*(NY1-NY2)
```

```
      THETA1=THETA2
      NX1=NX2
      NY1=NY2
    1 CONTINUE
      GO TO 15
   10 CONTINUE
      DO 11 K=1,NPONT1
      XP(K+1)=XP(K)+DELT*XVEL
   11 YP(K+1)=YP(K)+DELT*YVEL
   15 RNOISE(1)=1446.0
      RNOISB(1)=9999.99
      STDEVR=SQRT(RVAR)
      STDERB=SQRT(RVARB)
      CALL RANDN(RNOISB,NPOINT,0.0,STDERB)
      CALL RANDN(RNOISE,NPOINT,0.0,STDEVR)
    3 CONTINUE
      DO 2 K=1,NPOINT
      XRANG=SQRT(XP(K)**2+YP(K)**2)
      RANGE=SQRT(XRANG**2+ELEV**2)
      GO TO(5,6,7),KOPT
    5 X(K)=RANGE
      Z(K)=X(K)+RNOISE(K)
      GO TO 2
    6 CPHI11=XP(K)/RANGE
      X(K)=ACOS(CPHI11)
      Z(K)=X(K)+RNOISB(K)
      GO TO 2
    7 CPHI21=YP(K)/RANGE
      X(K)=ACOS(CPHI21)
      Z(K)=X(K)+RNOISB(K)
    2 CONTINUE
```

```
      IF(KOPT.NE.1) RETURN
      IF(RECUR.EQ.1) GO TO 30
      KTIME(1)=1
      K=2
      M=2
   28 XA=XP(K-1)
      XB=YP(K-1)
      XC=ELEV
      YA=XP(K)
      YB=YP(K)
      YC=ELEV
      CALL CROSPR(XA,XB,XC,YA,YB,YC,DELANG)
      IF(DELANG.GT.0.0012) GO TO 20
      KTIME(M)=5
      GO TO 25
   20 IF(DELANG.GT.0.0014) GO TO 21
      KTIME(M)=4
      GO TO 25
   21 IF(DELANG.GT.0.0016) GO TO 22
      KTIME(M)=3
      GO TO 25
   22 IF(DELANG.GT.0.0018) GO TO 23
      KTIME(M)=2
      GO TO 25
   23 KTIME(M)=1
   25 K=K+KTIME(M)
      M=M+1
      IF(K.GT.1000) GO TO 30
      GO TO 28
   30 RETURN
      END
```

```
 1*            SUBROUTINE SORTL(VAL,N,L)
 2*            DIMENSION VAL(1)
 3*            DIMENSION L(1)
 4*            M = N
 5*         10 CONTINUE
 6*            M = M/2
 7*            IF(M.EQ.0) RETURN
 8*            K = N-M
 9*            J = 1
10*         20 CONTINUE
11*            I = J
12*         30 CONTINUE
13*            II = I+M
14*            IF(VAL(I).LT.VAL(II)) GO TO 40
15*            F= VAL(I)
16*            LF=L(I)
17*            VAL(I) = VAL(II)
18*            L(I)=L(II)
19*            VAL(II) = F
20*            L(II)= LF
21*            I = I-M
22*            IF (I.GE.1) GO TO 30
23*         40 CONTINUE
24*            J = J+1
25*            IF(J.GT.K) GO TO 10
26*            GO TO 20
27*            END
```

```
 1*            SUBROUTINE ISORTL(VAL,N,L)
 2*            DIMENSION VAL(1)
 3*            DIMENSION L(1)
 4*            M = N
 5*         10 CONTINUE
 6*            M = M/2
 7*            IF(M.EQ.0) RETURN
 8*            K = N-M
 9*            J = 1
10*         20 CONTINUE
11*            I = J
12*         30 CONTINUE
13*            II = I+M
14*            IF (L(I).LT.L(II)) GO TO 40
15*            F= VAL(I)
16*            LF=L(I)
17*            VAL(I) = VAL(II)
18*            L(I)=L(II)
19*            VAL(II) = F
20*            L(II)= LF
21*            I = I-M
22*            IF (I.GE.1) GO TO 30
23*         40 CONTINUE
24*            J = J+1
25*            IF(J.GT.K) GO TO 10
26*            GO TO 20
27*            END
```

```
 1*            SUBROUTINE XNOISE(X)
 2*            COMMON/BLKA/XVEL,YVEL,DELT,ELEV,KEGUNO,KDELT
 3*            COMMON/BLKB/ALPHA,RVAR,RVARB,ACCMAX,NPOINT,KPOINT,PMAX,PZERO
 4*            DIMENSION XA(2500),I(2500),X(2500)
 5*            XA(1)=1476.0
 6*            CALL RANDU(XA,KPOINT)
 7*            X(1)=XA(1)
 8*            KPONT1=KPOINT-1
 9*            DO 1 K=1,KPONT1
10*        1   X(K+1)=X(K)*(1.0-ALPHA*DELT)+DELT*XA(K)
11*            DO 5 K=1,KPOINT
12*        5   I(K)=K
13*            CALL SORTL(X,KPOINT,I)
14*            CALL DISTFN(X)
15*            CALL ISORTL(X,KPOINT,I)
16*            RETURN
17*            END
```

```
 1*            SUBROUTINE CROSPR(BX,BY,BZ,CX,CY,CZ,SINANG)
 2*            DX=(BY*CZ)-(BZ*CY)
 3*            DY=(BZ*CX)-(BX*CZ)
 4*            DZ=(BX*CY)-(BY*CX)
 5*            DXYZ=SQRT(DX**2+DY**2+DZ**2)
 6*            BXYZ=SQRT(BX**2+BY**2+BZ**2)
 7*            CXYZ=SQRT(CX**2+CY**2+CZ**2)
 8*            SINANG=DXYZ/(BXYZ*CXYZ)
 9*            RETURN
10*            END
```

```
 1*              SUBROUTINE DISTEN(X)
 2*              DIMENSION X(1)
 3*              COMMON/BLKA/XVEL,YVEL,DELT,ELEV,KEGUNG,KDELT
 4*              COMMON/BLKB/ALPHA,RVAR,RVARB,ACCMAX,NPOINT,KPOINT,PMAX,PZERO
 5*              K1=PZERO*FLOAT(KPOINT)+0.5
 6*              K2=PMAX*FLOAT(KPOINT)+0.5
 7*              K3=(KPOINT-K1-2*K2)/2
 8*              XINC=ACCMAX/FLOAT(K3)
 9*              DO 1 I=1,K2
10*          1   X(I)=-ACCMAX
11*              K21=K2+1
12*              KA=K2+K3
13*              DO 2 I=K21,KA
14*          2   X(I)=-(ACCMAX-0.5*XINC)+XINC*FLOAT(I-K21)
15*              KA1=KA+1
16*              KB=KA+K3
17*              DO 3 I=KA1,KB
18*          3   X(I)=0.0
19*              KB1=KB+1
20*              KC=KB+K3
21*              DO 4 I=KB1,KC
22*          4   X(I)=0.5*XINC+XINC*FLOAT(I-KB1)
23*              KC1=KC+1
24*              DO 5 I=KC1,KPOINT
25*          5   X(I)=ACCMAX
26*              RETURN
27*              END
```

C.2 Subroutine GH

C.2.1 Definition of Arguments and Symbols

ZMEAS	is the incoming measured value of the target position
DELTT	is the estimation interval
K	is a counter which gives the value of the subscript of the arrays RANES and RANPD for the first computed estimate. RANES and RANPD store all smoothed and predicted estimates, respectively
G	is the value of the filter parameter, g
H	is the value of the filter parameter, h
XS	is the position smoothed estimate
DXS	is the velocity smoothed estimate
XP	is the position predicted estimate
DXP	is the velocity predicted estimate
KDELT	specifies the number of missed samples for the case when a measurement is not received at time t_ℓ + DELTT, where t_ℓ is the time of the previous measurement
KEXIT	updates the value of K as input to the filter at the next increment

```
      SUBROUTINE GH(ZMEAS,DELTT,K,G,H,XS,DXS,XP,DXP,KDELT)
      COMMON/BLK0/Z(1002),X(1002),KTIME(1005)
      COMMON/BLK0/KEXIT,RANES(1002),RANPD(1002),KREF
      IF(K .GT. 2) GO TO 1
      DXS=(ZMEAS-XS)/(DELTT*FLOAT(KDELT))
      XS=ZMEAS
      KEXIT=K+1
      KREF=1
      IF(K.EQ.2) KREF=KDELT+1
      RETURN
    1 CONTINUE
      IF(KDELT .EQ. 1) GO TO 3
      KDELTM=KDELT-1
      DO 2 KK=1,KDELTM
      XP=XS+DXS*DELTT
      XS=XP
      KKKM1=KK+K-1
      RANES(KKKM1)=XS
      RANPD(KKKM1)=XP
    2 CONTINUE
    3 XP=XS+DXS*DELTT
      DXP=DXS
      XS=XP+G*(ZMEAS-XP)
      DXS=DXP+H*(ZMEAS-XP)/DELTT
      KEXIT=KDELT+K
      KEXITM=KEXIT-1
      RANES(KEXITM)=XS
      RANPD(KEXITM)=XP
      RETURN
      END
```

C.3 Subroutine FMPCSU

FMPCSU is an acronym for fixed-memory-polynomial-coefficients-scaled-unscaled. As the name implies, subroutine FMPCSU computes the coefficients of the estimating polynomial for the fixed-memory orthogonal polynomial filter discussed in Section 4.3.2. In addition to the unscaled case discussed there, an output of polynomial coefficients may also be obtained for a scaled vector defined by

$$\underline{X}_k = \begin{bmatrix} x \\ \tau D x \\ \tau^2/2! D^2 x \\ \vdots \\ \frac{\tau^m}{m!} D^m x \end{bmatrix}_k .$$

Although the weighting matrix used for the scaled case is slightly different from the one defined in Section 4.3.2, it will reduce to the matrix given by (4.18) for the unscaled case.

C.3.1 Definition of Arguments

MP	= M + 1, where M is the desired degree of the estimating polynomial
LP	is the number of observations on which the polynomial is based
H	is the desired prediction interval
KSU	selects scaled (KSU = 1) or unscaled (KSU = 2) output
TAL	is the sampling interval between observations. If scaled output is selected, any real constant may be used for TAL

A	is the output array containing the computed polynomial coefficients
B and C	are arrays used internal to the subroutine

Note: The dimensions of A, B, and C are (MP x LP), (MP x LP), and (MP x MP), respectively, and must be so dimensioned in the main program.

C.3.2 Additional Subprograms

C.3.2.1 FUNCTION BINCOE (N,K)

This function subprogram computes the binomial coefficients $\binom{n}{k}$.

C.3.2.2 FUNCTION FACTK(L,K)

This function subprogram computes (L) (L-1) (L-2) ... (L - K + 1) where K must be a nonnegative integer.

```
      SUBROUTINE FMPCSU(MP,LP,H,KSU,TAL,A,B,C)
C
C
C     THE POLYNOMIAL COEFFICIENTS MAY BE OBTAINED IN TWO WAYS--
C
C
C     1.  USING ONLY ONE STORAGE ARRAY, BUT CALCULATING THE SAME
C     QUANTITY MANY TIMES.
C
C
C     2.  USING THREE STORAGE ARRAYS, BUT CALCULATING EACH QUANTITY
C     ONLY ONCE.
C
C
C     IT WAS DECIDED TO USE METHOD 2.
C
C
C
C
C     THIS IS A
C     SUBROUTINE TO CALCULATE FIXED-MEMORY POLYNOMIAL FILTER
C     COEFFICIENTS IN EITHER SCALED OR UNSCALED OUTPUT.  M IS THE DEGREE
C     OF THE POLYNOMIAL AND MP = M+1.  LP IS THE NUMBER OF OBSERVATIONS
C     ON WHICH THE POLYNOMIAL IS BASED.  TAL IS THE SAMPLING INTERVAL.
C     TAL = ANY REAL CONSTANT FOR SCALED OUTPUT.  KSU SELECTS SCALED(KSU=1) OR
C     UNSCALED(KSU=2) OUTPUT.  H IS THE DESIRED PREDICTION INTERVAL.  A
C     IS THE OUTPUT ARRAY OF COEFFICIENTS.  B AND C ARE ARRAYS USED
C     INTERNAL TO THE SUBROUTINE.
C
C
C     CAUTION**THE MAIN PROGRAM MUST CONTAIN 'DIMENSION A(MP,LP),B(MP,LP),
C     C(MP,MP)' WHERE MP AND LP ARE FIXED-POINT CONSTANTS.  THE USE OF
C     ANY OTHER VALUES REQUIRES THE DIMENSION STATEMENT IN THE SUBROUTINE
C     TO BE CHANGED TO CORRESPOND WITH THE MAIN PROGRAM.
```

```
C
C
      DIMENSION A(MP,LP),B(MP,LP),C(MP,MP)
      EPSLN = 1.E-30
      L=LP-1
C
C
C     THE QUANTITY TO BE OBTAINED IS W(H,T) = D(T)PHI(H)SGCR.
C
C
C     1.  CALCULATE C(I,I) (SINCE C IS A DIAGONAL MATRIX).
C
C
      DO 1 I=1,MP
    1 C(I,I) = FLOAT(2*I-1)*FACTK(L,I-1)/FACTK(L+I,I)
C
C
C     2.  CALCULATE B(I,J).
C
C
      DO 2 J=1,LP
      DO 2 I=1,MP
      DUM = 0.
      DO 3 NU = 1,I
    3 DUM = DUM+FLOAT((-1)**(NU-1))*BINCOE(I-1,NU-1)*
     1BINCOE(I+NU-2,NU-1)*FACTK(L-J+1,NU-1)/FACTK(L,NU-1)
    2 B(I,J) = DUM
C
C
C     3.  FORM THE PRODUCT C(I,I)*B(I,J).
C
C
      DO 4 I=1,MP
```

```
      DO 4 J=1,LP
    4 A(I,J) = C(I,I)*B(I,J)
C
C
C     4.  CALCULATE G(I,J) AND STORE IN C(I,J).
C
C
      DO 5 I=1,MP
      DO 5 J=1,MP
    5 C(I,J) = FLOAT((-1)**(J-1))*BINCOE(J-1,I-1)*BINCOE(J+I-2,I-1)/
     1FACTK(L,I-1)
C
C
C     5.  FORM THE PRODUCT G(I,J)*C(I,I)*B(I,J) = C(I,J)*A(I,J).
C
C
      DO 6 I=1,MP
      DO 6 J=1,LP
      DUM = 0.
      DO 7 K=1,MP
    7 DUM = DUM + C(I,K)*A(K,J)
    6 B(I,J) = DUM
C
C
C     6.  CALCULATE S(I,J) AND STORE IN C(I,J).
C
C
      C(1,1) = 1.
      DO 8 I=2,MP
    8 C(I,1) = 0.
      DO 9 J=2,MP
    9 C(1,J) = 0.
      DO 10 I=2,MP
      DO 10 J=2,MP
```

```
   10 C(I,J) = C(I-1,J-1) + (FLOAT(J-2))*C(I,J-1)
C
C
C     7.  FORM THE PRODUCT A(I,J) = C(I,J)*B(I,J).
C
C
      DO 11 I=1,MP
      DO 11 J=1,LP
      DUM = 0.
      DO 12 K=1,MP
   12 DUM = DUM + C(I,K)*B(K,J)
   11 A(I,J) = DUM
C
C
C     8.  CALCULATE PHI(I,J) AND STORE IN C(I,J).
C
C
      IF (ABS(H) .LT. EPSLN) GO TO 131
      DO 13 I=1,MP
      DO 13 J=1,MP
      IF (J .LT. I) GO TO 135
      C(I,J) = BINCOE(J-1,I-1)*(H**(J-I))
      GO TO 13
  135 C(I,J)=0.
   13 CONTINUE
      GO TO 134
  131 DO 132 I=1,MP
      DO 132 J=1,MP
      IF(I .EQ. J) GO TO 133
      C(I,J) = 0.
      GO TO 132
  133 C(I,J) = BINCOE(J-1,I-1)
  132 CONTINUE
  134 CONTINUE
```

```
C
C
C     9.  FORM THE PRODUCT B(I,J) = C(I,J)*A(I,J).
C
C
      DO 14 I=1,MP
      DO 14 J=1,LP
      DUM = 0.
      DO 15 K=1,MP
   15 DUM = DUM + C(I,K)*A(K,J)
   14 B(I,J) = DUM
      DO 141 I=1,MP
      DO 141 J=1,LP
  141 A(I,J) = B(I,J)
C
C
C     10. SELECT SCALED OR UNSCALED OUTPUT.  IF SCALED OUTPUT IS
C     SELECTED, RETURN AND PRINT OUT A(I,J) IN DESIRED FORMAT.
C
C
      GO TO (19,16), KSC
C
C
C     11.  FOR UNSCALED OUTPUT, CALCULATE D(I,J) AND STORE IN C(I,I)
C     SINCE D IS A DIAGONAL MATRIX.
C
C
   16 CONTINUE
      DO 17 I=1,MP
   17 C(I,I) = FACTK(I-1,I-1)/(TAL**(I-1))
C
C
C     12.  FORM THE PRODUCT A(I,J) = C(I,I)*B(I,J).
C
```

```
170*      C
171*            DO 18 I=1,MP
172*            DO 18 J=1,LP
173*         18 A(I,J) = C(I,I)*B(I,J)
174*      C
175*      C
176*      C     IF UNSCALED OUTPUT WAS SELECTED, PRINT OUT A(I,J) IN DESIRED
177*      C     FORMAT IN MAIN PROGRAM.
178*      C
179*      C
180*         19 RETURN
181*            END
```

```
  1*            FUNCTION BINCOE(N,K)
  2*      C
  3*      C
  4*      C
  5*      C     THIS FUNCTION SUBPROGRAM COMPUTES THE BINOMIAL COEFFICIENTS (N,K).
  6*      C
  7*      C
  8*      C
  9*            IF(K-0) 3,2,1
 10*         1  BINCOE=FACTK(N,K)/FACTK(K,K)
 11*            GO TO 4
 12*         2  BINCOE=1.
 13*            GO TO 4
 14*         3  BINCOE=0.
 15*         4  RETURN
 16*            END
```

```
 1*            FUNCTION FACTK(L,K)
 2*      C
 3*      C
 4*      C     THIS FUNCTION SUBPROGRAM COMPUTES (L)(L-1)(L-2). . .(L-K+1),
 5*      C     WHERE K MUST BE A NON-NEGATIVE INTEGER.
 6*      C     ***NOTE THAT FACTK(K,K) = FACTORIAL K***
 7*      C
 8*      C
 9*            IF(K-0) 5,3,1
10*          1 IF (K .EQ. 1) GO TO 4
11*            KM=K-1
12*            FACTK=FLOAT(L)
13*            DO 2 I=1,KM
14*          2 FACTK=FACTK*FLOAT(L-I)
15*            GO TO 7
16*          3 FACTK=1.
17*            GO TO 7
18*          4 FACTK=FLOAT(L)
19*            GO TO 7
20*          5 PRINT 6
21*          6 FORMAT(77H*****ERROR IN INPUT TO FUNCTION FACTK--K MUST BE A NON-
22*           1NEGATIVE INTEGER*****)
23*          7 RETURN
24*            END
```

C.4 General Fixed Memory Filter

The most important symbols used in the subroutine FIXMEM are defined below. This program is a straightforward implementation of the algorithm described in Chapter 4. The Univac subroutine INVTIT is used to invert the 3 x 3 matrix.

Symbols	Definitions
KTIM	Timing index for most recent observation
I	Index which defines the most recent observation interval
LP	Memory length
DELT	Estimation interval
U	Array of observations
X	Array of actual position values
INC	Array showing the number of intervals between observations
S	Array of smoothed positions
P	Array of predicted positions

```
 1*          SUBROUTINE FIXMEM (KTIM,I,LP,DELT)
 2*          COMMON/BLKC/U(1002),X(1002),INC(1005)
 3*          COMMON/BLKD/S(1002),P(1002)
 4*          DIMENSION W(3,9),XST(3),KZ(11),Z(11),Y(11)
 5*          DIMENSION TTT(3,3),TTY(3),TTTI(3,3)
 6*          L=LP-1
 7*          HDELT=DELT/2.
 8*          IML=I-L
 9*          KZ(LP)=0
10*          DO 1 K=L,1,-1
11*        1 KZ(K)=KZ(K+1)-INC(IML+K)
12*          DO 6 K=1,LP
13*        6 Z(K)=DELT*FLOAT(KZ(K))
14*          DO 7 K=LP,1,-1
15*          J=KTIM+KZ(K)
16*        7 Y(K)=U(J)
17*          SZ1=0.
18*          SZ2=0.
19*          SZ3=0.
20*          SZ4=0.
21*          DO 2 K=1,L
22*          Z2=Z(K)*Z(K)
23*          SZ1=SZ1+Z(K)
24*          SZ2=SZ2+Z2
25*          SZ3=SZ3+Z2*Z(K)
26*        2 SZ4=SZ4+Z2*Z2
27*          TTT(1,1)=FLOAT(LP)
28*          TTT(1,2)=SZ1
29*          TTT(1,3)=SZ2/2.
30*          TTT(2,1)=SZ1
31*          TTT(2,2)=SZ2
32*          TTT(2,3)=SZ3/2.
33*          TTT(3,1)=SZ2/2.
34*          TTT(3,2)=SZ3/2.
35*          TTT(3,3)=SZ4/4.
36*          CALL INVIT(TTT,3,3,TTTI,2,W)
```

```
      DO 3 K=2,3
    3 TTY(K)=0.
      TTY(1)=Y(LP)
      DO 4 K=1,L
      TTY(1)=TTY(1)+Y(LP-K)
      TTY(2)=TTY(2)+Y(LP-K)*Z(LP-K)
    4 TTY(3)=TTY(3)+Y(LP-K)*Z(LP-K)*Z(LP-K)/2.
      DO 5 K=1,3
      XST(K)=0.
      DO 5 J=1,3
    5 XST(K)=XST(K)+TTT1(K,J)*TTY(J)
      S(KTIM)=XST(1)
      NP=INC(I+1)
      NPM1=NP-1
      IF(NPM1.EQ.0) GO TO 9
      DO 8 K=1,NPM1
      XST(1)=XST(1)+DELT*(XST(2)+HDELT*XST(3))
      XST(2)=XST(2)+DELT*XST(3)
      P(KTIM+K)=XST(1)
    8 S(KTIM+K)=XST(1)
    9 P(KTIM+NP)=XST(1)+DELT*(XST(2)+HDELT*XST(3))
      I=I+1
      KTIM=KTIM+INC(I)
      RETURN
      END
```

C.5 The Kalman Filter Program

C.5.1 Symbols Used

Following is a list of symbols used in the program and the Kalman filter variables (as defined in Chapter 4) they stand for.

XHAT	Filtered vector , $\hat{\underline{X}}(k \mid k)$
XHATP	Predicted vector , $\hat{\underline{X}}(k \mid k-1)$
Q	System noise covariance , $Q(k)$
COVAR	Filtered error covariance , $P(k \mid k)$
COVARP	Prediction error covariance , $P(k \mid k-1)$
DELTA	Estimation interval , Δt
PHI	State transition matrix for DELTA = $\Phi(\Delta t)$
H	The vector M in the observation model
KTIME	Array of sampling intervals, k th sampling interval is DELT = KTIME(K) * DELTA = Δt_k
PHIA	Transition matrix for DELT = $\Phi(\Delta t_k)$
FILTER	Array of smoothed positions, FILTER(k) = $\hat{x}_1(k \mid k)$
PRDICT	Array of predicted positions, PRDICT(k) = $\hat{x}_1(k \mid k-1)$
XVEL, YVEL	Velocities for non-maneuvering trajectory (See Appendix A)
ELEV	Elevation of the vehicle (See Appendix A)
KEQUNQ	1 for equal sampling intervals 2 for unequal sampling intervals
KDELT	1, 2, or 3 -- Each sampling interval is equal to KDELT * DELTA

ALPHA, ACCMAX, PMAX, PZERO	Parameters of the maneuvering variable, γ, A_{max}, P_{max} and P_o
Z	Array of observations
X	Array of actual position values
RVAR	Observation noise covariance σ_v^2 for range measurement
RVARB	Observation noise covariance σ_v^2 for angle measurement
QVAR	σ_m^2 for the range model
QVARB	σ_m^2 for the angle model
NPOINT	Total number of estimation points (DELTA apart)
KPOINT	Total number of values generated for the maneuver variable a(k) (See Appendix A)

C.5.2 Description of the Program

This program is used to compute the filtered estimate and the one step predicted estimate based on the observations in the array Z. The estimation interval is constant at DELTA and the observation interval is an integral multiple of DELTA. In the case of sampling interval not equal to DELTA, the observation interval is given by DELTA * KDELT. In the case of unequal sampling intervals, the kth sampling interval is given by DELTA * KTIME(k) where the array KTIME is generated in the subroutine TRAJ. In both the case of unequal sampling interval and the case of equal sampling interval greater than DELT, estimates in between observation samples are predicted based on the latest available filtered estimate. For the case of equal sampling interval, the matrices PHIA and Q are computed only once. However,

they have to be computed at every sampling instant in the case of unequal sampling interval.

In its present form the program completes the filtering in all three coordinates and for both the trajectories discussed in Appendix A.

C.5.3 Input and Output

The input to the program is the set of values for the parameters XVEL, YVEL, DELTA, ELEV, ALPHA, RVAR, RVARB, ACCMAX, NPOINT, KPOINT, PMAX, PZERO, KEQUNQ and KDELT. In the present format these are defined through DATA statements.

The outputs are the mean squared values of the errors in the filtered estimate and the one step predicted estimate. If the program is used for only one coordinate of one of the trajectories, then the arrays FILTER and PRDICT provide the filtered and one step predicted estimates. The covariance of the error (filter or predictor) can be stored if necessary.

C.5.4 Subprograms Used

(1) TRAJ: This is used to generate the trajectory. The calling sequence is

CALL TRAJ(KOPT,KTRJ)

For a complete discussion see Appendix A and C.1.

(2) MXADD: The calling sequence is

CALL MXADD(A,B,C,M,N)

and is used to add two MxN matrices A and B and store the result in the matrix C.

(3) MXSUB: The calling sequence is

CALL MXSUB(A,B,C,M,N)

and is used to subtract two M x N matrices A and B and store the result in the matrix C.

(4) MXTRN: The calling sequence is

CALL MXTRN(A,B,M,N)

and is used to transpose the M x N matrix A and store the result in the N x M matrix B.

(5) MXSCA: The calling sequence is

CALL MXSCA(A,M,N,S)

and is used to multiply the elements of the M x N matrix A by the Scalar S and store the result in matrix A.

(6) MXMLT: The calling sequence is

CALL MXMLT(A,B,C,M,L,N)

and is used to multiply the M x L matrix A by the L x N matrix B on the right, and store the result in the M x N matrix C.

THE KALMAN FILTER PROGRAM

```
      DIMENSION PHIA(3,3)
      DIMENSION XHAT(3,1),XHATP(3,1),G(3,3),COVAR(3,3),COVARP(3,3)
      DIMENSION PHI(3,3),GAIN(3,1),H(1,3),HT(3,1),PHT(3,1),HPHT(1,1)
      DIMENSION PHIT(3,3),PPHIT(3,3),UNITY(3,3),GAINH(3,3)
      DIMENSION HXHAT(1,1),PHPPHI(3,3),UMKH(3,3)
      DIMENSION FILTER(1002),PRDICT(1002)
      EQUIVALENCE(GAIN(1),PHT(1))
      COMMON/BLKA/XVEL,YVEL,DELTA,ELEV,KEQUNO,KDELT
      COMMON/BLKB/ALPHA,RVAR,RVARB,ACCMAX,NPOINT,KPOINT,PMAX,PZERO
      COMMON/BLKC/Z(1002),X(1002),KTIME(1000)
      REAL MALPT
      DATA XVEL,YVEL/150.,150./
      DATA DELTA,ELEV/.1,10000.0/
      DATA ALPHA,RVAR,RVARB,ACCMAX/0.1,100.00,.64E-4,60.0/
      DATA NPOINT,KPOINT,PMAX,PZERO/1002,2500,.1,.5/
      DATA H(1,1),H(1,2),H(1,3)/1.0,0.0,0.0/
      DATA KEQUNO,KDELT/2,1/
      QVAR=(ACCMAX**2)*(1.0-PZERO+4.0*PMAX)/3.0
      QVARB=QVAR/ELEV**2
      RVARA=RVAR
      QVARA=QVAR
      ALPT=ALPHA*DELTA
      MALPT=-ALPT
      EX=EXP(MALPT)
```

```
C
C  COMPUTE TRANSITION MATRIX FOR DELTA
C
      PHI(1,1)=1.0
      PHI(1,2)=DELTA
      PHI(1,3)=(-1.0+ALPT+EX)/(ALPHA**2)
      PHI(2,1)=0.0
      PHI(2,2)=1.0
      PHI(2,3)=(1.0-EX)/ALPHA
      PHI(3,1)=0.0
      PHI(3,2)=0.0
      PHI(3,3)=EX
      DO 90 NTRAJ=1,2
      DO 50 NCASE=1,3
C
C  COMPUTE THE TRAJECTORY
C
      CALL TRAJ(NCASE,NTRAJ)
      IF(KEQUNO.NE.2) GO TO 75
      KTIM=KTIME(1)
      GO TO 80
   75 KTIM=KDELT
   80 DELT=DELTA*FLOAT(KTIM)
      ALPT=ALPHA*DELT
      MALPT=-ALPT
      EX=EXP(MALPT)
C
C  INITIALIZZE THE ESTIMATOR
C
      XHAT(1,1)=Z(1+KTIM)
      XHAT(2,1)=(Z(1+KTIM)-Z(1))/(DELTA*FLOAT(KTIM))
      XHAT(3,1)=0.0
      FILTER(2)=XHAT(1,1)
C
C  INITIALIZE THE COVARIANCE MATRIX
C
```

```
*            COVAR(1,1)=PVAR
*            COVAR(1,2)=RVAR/DELT
*            COVAR(2,1)=COVAR(1,2)
*            COVAR(2,2)=(2.0*RVAR/(DELT**2))+(QVAR/((ALPHA**4)*(DELT**2)))
*           1*(2.0-ALPT**2+(2.0/3.0)*(ALPT**3)-2.0*EX-2.0*ALPT*EX)
*            COVAR(1,3)=0.0
*            COVAR(3,1)=0.0
*            COVAR(2,3)=(QVAR/(ALPHA*ALPT))*(EX+ALPT-1.0)
*            COVAR(3,2)=COVAR(2,3)
*            COVAR(3,3)=QVAR
*            MM=1+KTIM
*            M=3
*            IF(KEGUNG.NE.2) GO TO 85
*     C
*     C  CALCULATE DELT, THE M TH SAMPLING INTERVAL
*     C
*       70   KTIM=KTIME(M-1)
*            DELT=DELTA*FLOAT(KTIM)
*            ALPT=ALPHA*DELT
*     .      MALPT=-ALPT
*            EX=EXP(MALPT)
*       85   CONTINUE
*     C
*     C      COMPUTE THE G MATRIX
*     C
*            G(1,1)=(1.0-(EX**2)+2.0*ALPT+(2.0/3.0)*(ALPT**3)-2.0*(ALPT**2)
*           1-4.0*ALPT*EX)/(2.0*(ALPHA**5))
*            G(1,2)=(1.0+(EX**2)-2.0*EX+2.0*ALPT*EX-2.0*ALPT
*           1+(ALPT**2))/(2.0*(ALPHA**4))
*            G(1,3)=(1.0-(EX**2)-2.0*ALPT*EX)/(2.0*(ALPHA**3))
*            G(2,2)=(4.0*EX-3.0-(EX**2)+2.0*ALPT)/(2.0*(ALPHA**3))
*            G(2,3)=(1.0+(EX**2)-2.0*EX)/(2.0*(ALPHA**2))
*            G(3,3)=(1.0-(EX**2))/(2.0*ALPHA)
```

```
 94*            Q(3,2)=Q(2,3)
 95*            Q(3,1)=Q(1,3)
 96*            Q(2,1)=Q(1,2)
 97*            S=QVAR*2.0*ALPHA
 98*            CALL MXSCA(Q,3,3,S)
 99*      C
100*      C  COMPUTE TRANSITION MATRIX FOR DELT
101*      C
102*            PHIA(1,1)=1.0
103*            PHIA(1,2)=DELT
104*            PHIA(1,3)=(-1.0+ALPT+EX)/(ALPHA**2)
105*            PHIA(2,1)=0.0
106*            PHIA(2,2)=1.0
107*            PHIA(2,3)=(1.0-EX)/ALPHA
108*            PHIA(3,1)=0.0
109*            PHIA(3,2)=0.0
110*            PHIA(3,3)=EX
111*       110  CONTINUE
112*            IF(KTIM.EQ.1) GO TO 60
113*      C
114*      C  PREDICTION IN BETWEEN OBSERVATION SAMPLES
115*      C
116*            KTIM1=KTIM-1
117*            DO 61 NTIM=1,KTIM1
118*            MM=MM+1
119*            CALL MXMLT(PHI,XHAT,XHATP,3,3,1)
120*            DO 62 NHAT=1,3
121*         62 XHAT(NHAT,1)=XHATP(NHAT,1)
122*            FILTER(MM)=XHAT(1,1)
123*            PRDICT(MM)=XHATP(1,1)
124*         61 CONTINUE
125*         60 MM=MM+1
126*      C
127*      C  ONE STEP PREDICTOR
128*      C
129*            CALL MXMLT(PHI,XHAT,XHATP,3,3,1)
```

```
      C
      C  COVARIANCE OF ONE STEP PREDICTOR
      C
           CALL MXTRN(PHIA,PHIT,3,3)
           CALL MXMLT(COVARP,PHIT,PPHIT,3,3,3)
           CALL MXMLT(PHIA,PPHIT,PHPPHI,3,3,3)
           CALL MXADD(PHPPHI,G,COVARP,3,3)
      C
      C  COMPUTE THE GAIN VECTOR
      C
           CALL MXTRN(H,HT,1,3)
           CALL MXMLT(COVARP,HT,PHT,3,3,1)
           CALL MXMLT(H,PHT,HPHT,1,3,1)
           S=1.0/(HPHT(1,1)+RVAR)
           CALL MXSCA(PHT,3,1,S)
           DO 1 I=1,3
           DO 1 J=1,3
           UNITY(I,J)=0.0
           IF(I.EQ.J) UNITY(I,J)=1.0
         1 CONTINUE
      C
      C  THE FILTERED ESTIMATE AND ITS COVARIANCE
      C
           CALL MXMLT(H,XHATP,HXHAT,1,3,1)
           S=Z(MM)-HXHAT(1,1)
           CALL MXMLT(GAIN,H,GAINH,3,1,3)
           CALL MXSCA(GAIN,3,1,S)
           CALL MXADD(XHATP,GAIN,XHAT,3,1)
           CALL MXSUB(UNITY,GAINH,UMKH,3,3)
           CALL MXMLT(UMKH,COVARP,COVAR,3,3,3)
           FILTER(MM)=XHAT(1,1)
           PREDICT(MM)=XHATP(1,1)
           M=M+1
           IF(MM.GE.NPOINT) GO TO 10
           IF(KEQUNC.EQ.1) GO TO 110
```

```
      GO TO 70
   10 CONTINUE
      SA=0.0
      SB=0.0
      DO 100 K=10,NPOINT
      SA=SA+(FILTER(K)-X(K))**2
  100 SB=SB+(PRDICT(K)-X(K))**2
      SA=SA/(FLOAT(NPOINT-10))
      SB=SB/(FLOAT(NPOINT-10))
      PRINT 30,NTRAJ,NCASE,SA,SB
      QVAR=QVARB
      RVAR=RVARB
   50 CONTINUE
      RVAR=RVARA
   90 QVAR=QVARA
   30 FORMAT(//'  TRAJ=',I2,'  CASE= '12,' FILTER MSE=',E12.5,'PREDICTOR
     1  MSE=',E12.5//)
      STOP
      END
```

```
      SUBROUTINE MXADD(A,B,C,M,N)
      DIMENSION A(M,N),B(M,N),C(M,N)
      DO 1 I=1,M
      DO 1 J=1,N
    1 C(I,J)=A(I,J)+B(I,J)
      RETURN
      END
```

```
      SUBROUTINE MXSUB(A,B,C,M,N)
      DIMENSION A(M,N),B(M,N),C(M,N)
      DO 1 I=1,M
      DO 1 J=1,N
    1 C(I,J)=A(I,J)-B(I,J)
      RETURN
      END
```

```
1*              SUBROUTINE MXSCA(A,M,N,S)
2*              DIMENSION A(M,N)
3*              DO 1 I=1,M
4*              DO 1 J=1,N
5*        1     A(I,J)=S*A(I,J)
6*              RETURN
7*              END
```

```
1*              SUBROUTINE MXTRN(A,B,M,N)
2*              DIMENSION A(M,N),B(N,M)
3*              DO 1 I=1,M
4*              DO 1 J=1,N
5*        1     B(J,I)=A(I,J)
6*              RETURN
7*              END
```

```
1*              SUBROUTINE MXMLT(A,B,C,M,L,N)
2*              DIMENSION A(M,L),B(L,N),C(M,N)
3*              DO 1 I=1,M
4*              DO 1 J=1,N
5*              S=0.0
6*              DO 2 K=1,L
7*        2     S=S+A(I,K)*B(K,J)
8*              C(I,J)=S
9*        1     CONTINUE
10*             RETURN
11*             END
```

LIST OF REFERENCES

[1] Polge, R. J., and J. E. Scalf, "Radar and Battery Resource Allocation," The University of Alabama in Huntsville, UARI Research Report No. 79, December 1970.

[2] Scalf, J. E., "Radar and Battery Resource Allocation," The University of Alabama in Huntsville, Master's Thesis, 1971.

[3] Singer, R. A., "Estimating Optimal Tracking Filter Performance for Manned Maneuvering Targets," IEEE Trans. Aerospace and Electronic Systems, Vol. AES-6, pp. 473 - 483, July 1970.

[4] Singer, R. A., and Behnke, K. W., "Real-time Tracking Filter Evaluation and Selection for Tactical Applications," IEEE Trans. Aerospace and Electronic Systems, Vol. AES-7, pp. 100 - 110, January 1971.

[5] Wiener, N., Extrapolation, Interpolation and Smoothing of Stationary Time Series, Wiley, 1949.

[6] Kahrilas, P. J., "Design of Electronic Scanning Radar Systems (ESRS)," Proc. IEEE, Vol. 56, pp. 1763 - 1771, November 1968.

[7] Benedict, T. R., and Bordner, G. W., "Synthesis of an Optimal Set of Radar Track-While-Scan Smoothing Equations," IRE Trans. Automatic Control, Vol. AC-7, pp. 27 - 32, July 1962.

[8] Morrison, N., Introduction to Sequential Smoothing and Prediction, McGraw-Hill, 1969

[9] Kalman, R. E., "A New Approach to Linear Filtering and Prediction Problems," Trans. ASME, J. Basic Engineering, Vol. 82, pp. 34 - 45, March 1960

[10] Meditch, J. S., Stochastic Optimal Linear Estimation and Control, McGraw-Hill, 1969.

[11] Holliday, E. M., "Transformation of a Set of Pseudo-Random Numbers into a Set Representing any Desired Probability and Correlation," M.S. Thesis, The University of Alabama in Huntsville, 1970.