THESIS

AN APPLICATION OF KALMAN FILTERING
AND SMOOTHING TO TORPEDO TRACKING

by

Mustafa Işık

October 1982

Thesis Advisor: H. A. Titus

Approved for public release; distribution unlimited
**AN APPLICATION OF KALMAN FILTERING AND OPTIMAL SMOOTHING TO TORPEDO TRACKING**

Mustafa Isik

Naval Postgraduate School
Monterey, California 93940

Naval Postgraduate School
Monterey, California 93940

Naval Postgraduate School
Monterey, California 93940

Naval Postgraduate School
Monterey, California 93940

Approved for public release; distribution unlimited

**A sequential Extended Kalman Filter and Smoothing routine was developed to provide real time estimates of torpedo position and depth on the three dimensional underwater tracking range at the Naval Torpedo Station, Keyport, Washington. Inputs to the routine were acoustic pulse transit times from the target to receiving array elements which are non-linear functions of the position coordinates. These inputs were linearized and**
the filter gains and filtered estimates calculated on-line. By using a smoothing subroutine, all past filtered estimates were smoothed. Tests were conducted using simulated torpedo trajectories that traversed multiple hydrophone arrays. It was found that filter performance was dependent on system noise and the distance the torpedo was from the hydrophone array and the smoothed estimates of states were better than or equal to the filtered estimates.
An Application of Kalman Filtering and Smoothing to Torpedo Tracking

by

Mustafa Işık
Lieutenant Junior Grade, Turkish Navy
B.S.E.E., Naval Postgraduate School, 1982

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the
NAVAL POSTGRADUATE SCHOOL
October 1982

Author:

Approved by:

Thesis Advisor

Second Reader

Chairman, Department of Electrical Engineering

Dean of Science and Engineering
ABSTRACT

A sequential Extended Kalman Filter and Smoothing routine was developed to provide real time estimates of torpedo position and depth on the three dimensional underwater tracking range at the Naval Torpedo Station, Keyport, Washington. Inputs to the routine were acoustic pulse transit times from the target to receiving array elements which are non-linear functions of the position coordinates. These inputs were linearized and the filter gains and filtered estimates calculated on-line. By using a smoothing subroutine, all past filtered estimates were smoothed. Tests were conducted using simulated torpedo trajectories that traversed multiple hydrophone arrays. It was found that filter performance was dependent on system noise and the distance the torpedo was from the hydrophone array and the smoothed estimates of states were better than or equal to the filtered estimates.
# TABLE OF CONTENTS

I. INTRODUCTION ........................................... 8

II. DESCRIPTION OF RANGE TRACKING GEOMETRY ............... 10

III. THEORY ................................................... 12
    A. THE EXTENDED KALMAN FILTER ....................... 12
    B. OPTIMAL LINEAR SMOOTHING ......................... 15

IV. PROBLEM DEFINITION - TORPEDO TRACKING WITH THE 
    EXTENDED KALMAN FILTER AND OPTIMAL SMOOTHING ...... 17
    A. FILTER EQUATIONS .................................... 17
    B. THE SEQUENTIAL EXTENDED KALMAN FILTER .......... 20
    C. OPTIMAL SMOOTHING PROCESS ......................... 22

V. TESTING AND SIMULATION ................................. 24
    A. DESCRIPTION .......................................... 24
    B. THE GATING SCHEME .................................. 25
    C. MULTIPLE ARRAY TRACKING ............................. 26

VI. TEST RESULTS ............................................ 30
    A. SERIES ONE ........................................... 30
    B. SERIES TWO .......................................... 30
    C. SERIES THREE ......................................... 31
    D. SERIES FOUR .......................................... 33

VII. CONCLUSIONS ............................................ 34

FIGURES ...................................................... 35

APPENDIX A: PROGRAM DESCRIPTION AND FEATURES ............ 76
    A. PROGRAM SUBROUTINES ................................ 77
    B. UTILITY PROGRAMS .................................... 80
ACKNOWLEDGEMENT

The author is deeply indebted to Professor Hal Titus of the Naval Postgraduate School, Monterey, California for his counsel and professional guidance during this project.
I. INTRODUCTION

The NUWES at Keyport, Washington currently operates two three-dimensional (3-D) underwater tracking range utilizing a sonar transmitter installed in the torpedo to be tracked. The transmitter is synchronized with a master clock. Timed acoustic pulses are received by bottom mounted hydrophone arrays and then relayed via cable to a computer at the observation site. The computer calculates the positional coordinates of the torpedo and plots its trajectory through the water.

The measured data, which consists of the elapsed time from transmission of a pulse until its receipt at the hydrophone array, is corrupted with noise due to the combined effects of environmental factors and measurement instruments.

These noisy tracks are later analyzed, and measurements judged most inaccurate on the basis of total track statistics are removed in order to obtain a smooth representation of the track.

An opportunity exists for expanding the capability of the system by applying a real time Kalman Filter and post test Smoothing routine which can take as an input the transit times of the acoustic pulses, and produce the best
estimate of the position of the tracked object at a particular time. Previous research in this area [3] and [4], revealed that a Kalman filter utilizing a sequential estimation approach was desirable.

The intention is to develop and test a sequential Kalman filter and smoothing algorithm that can be interfaced with the current underwater range system.
II. DESCRIPTION OF RANGE TRACKING GEOMETRY

The hydrophone array, consisting of four independent elements, defines an orthogonal coordinate system in which transit time measurements are made. As shown in Figure 1, four hydrophones X, Y, Z, and C are on four adjacent vertices separated by a distance d, along the edge of the cube. The origin of the array coordinates is at the center of the cube with the orthogonal coordinates parallel to its edge. Positional information is computed from the transit times of a periodic synchronous acoustic signal travelling from the torpedo to the four hydrophones on the array. The range measures the tracked torpedo's position every 1.31 seconds to an accuracy that is typically within 3 to 30 feet. A more detailed description of the range tracking capability is described in [2].
Figure 1. Geometry of a Tracking Array
III. THEORY

A. THE EXTENDED KALMAN FILTER

Since the transit times were readily available and are nonlinear functions of position, these equations can be linearized and Kalman filter theory applied using the extended Kalman filter. This procedure produces a real-time system, filtering on the transit times $T_o$, $T_x$, $T_y$ and $T_z$, without the necessity of converting these times to positions.

For the three-dimensional location problem three position states $(x, y, z)$ and two velocity states $(v_x, v_y)$ specify target motion. The discrete linear and nonlinear observation equations are given by

$$\mathbf{x}(k + 1) = \Phi \cdot \mathbf{x}(k) + \Gamma \cdot \mathbf{w}(k) \quad (3.1)$$

and

$$\mathbf{z}(k) = \mathbf{h}(\mathbf{x}(k), k) + \mathbf{v}(k) \quad (3.2)$$

In these equations $\Phi$ and $\Gamma$ are constant matrices and $\mathbf{h}$ is a nonlinear function of the state variable $\mathbf{x}$. $\mathbf{w}(k)$ is plant excitation noise and $\mathbf{v}(k)$ is measurement noise. The plant noise and measurement noise are assumed uncorrelated (white) with zero mean. That is,
\[ E[w(k) \cdot w^T(j)] = Q'(k) \delta_{kj} \]

and

\[ E[v(k) \cdot v^T(j)] = R(k) \delta_{kj} \]

with

\[ \delta_{kj} = 1 \quad k = j \]
\[ = 0 \quad k \neq j \]

In order to apply the linear filter equation (3.2) is expanded in a Taylor series about the best estimate of the state at that time and only the first-order terms are kept. Equation (3.2) gives

\[ z(k) = H(k) \cdot \hat{x}(k) + v(k) \quad (3.3) \]

where

\[ H(k) = \frac{\partial h}{\partial x} \quad \hat{x}'(k) = \hat{x}(k/k-1) \quad (3.3a) \]

\( \hat{x}(k/k-1) \) is a predicted value of the state before the kth measurement.

A state error vector is defined by

\[ \bar{x}(k/k) = \hat{x}(k/k) - \hat{x}(k) \]

and a predicted state error vector is defined by

\[ \bar{x}(k/k-1) = \hat{x}(k/k-1) - \hat{x}(k) \]
The covariance of state error matrix is defined by

$$P(k/k) = E[\tilde{x}(k/k) \cdot \tilde{x}^T(k/k)],$$

and the predicted covariance of state error matrix is given by

$$P(k/k-1) = E[\tilde{x}(k/k-1) \cdot \tilde{x}^T(k/k-1)].$$

The state excitation matrix is given by

$$Q(k) = r(k) E[w(k) \cdot w^T(k)] \cdot r^T(k)$$

and the measurement noise covariance matrix is

$$R(k) = E[v(k) \cdot v^T(k)].$$

The Kalman filter equations are given by [1]:

$$P(k+1/k) = P(k/k) + Q(k)$$

(3.4a)

$$G(k) = P(k/k-1)H^T(k)[H(k) \cdot P(k/k-1)H^T(k) + R(k)]^{-1}$$

(3.4b)

$$P(k) = [I - G(k)H(k)] P(k/k-1)$$

(3.4c)

$$\hat{x}(k+1/k) = \phi \cdot \hat{x}(k/k)$$

(3.4d)

$$z(k/k-1) = h(\hat{x}(k/k-1), k)$$

(3.4e)

$$\hat{x}(k) = \hat{x}(k/k-1) + G(k)[z(k) - z(k/k-1)]$$

(3.4f)
The Q matrix serves not only to allow for maneuvering but also to account for any model inaccuracies, that is, any discrepancies between the true action of the torpedo and its characterization by Equation (3.1). The Q also serves to prevent the gain matrix $G(k)$ from approaching zero by always insuring uncertainty in the predicted covariance of error matrix $P(k+1/k)$.

B. OPTIMAL LINEAR SMOOTHING

Smoothing is a non-real-time data processing scheme that uses all measurements between 0 and T to estimate the state of a system at certain time $t$, where $0 \leq t \leq T$. The smoothed estimate of $\hat{x}(t)$ based on all the measurements between 0 and T is denoted by $\hat{x}(t/T)$.

Smotherer error covariance is denoted by $P(t/T)$ and $P(t/T) \leq P(t)$ means that the smoothed estimate of $\hat{x}(t)$ is always better than or equal to its filtered estimate. This is shown graphically in Figure 2.

Several forms of the smoothing equations may be derived. One is the Rauch-Tung-Striebel form, which was used in our particular case with the discrete-time expressions summarized as follows [1]:

$$\hat{x}(k/N) = \hat{x}(k/k) + A_k [\hat{x}(k+1/N) - \hat{x}(k+1/k)] \quad (3.5a)$$
Figure 2. Advantage of Performing Optimal Smoothing [1]

where

\[ \hat{x}_k = P(k/k)A(k)^T P(k+1/k)^{-1} \quad \text{for } k = N-1 \]

\[ P(k/N) = P(k/k) + A_k [P(k+1/N) - P(k+1/k)] A_k^T \quad (3.5b) \]

also for \( k = N-1 \).

In these equations \( \hat{x}(k/N) \) is smoothed State Estimate and \( P(k/N) \) is Error Covariance Matrix Propagation.
IV. PROBLEM DEFINITION - TORPEDO TRACKING WITH THE EXTENDED KALMAN FILTER AND OPTIMAL SMOOTHING

A. FILTER EQUATIONS

In the torpedo tracking problem, the non-linear observations are the four independent transit times from the tracked object to the hydrophones, \( T_c, T_x, T_y \) and \( T_z \). Thus the non-linear measurement matrix \( z(k) \) is defined as:

\[
\begin{bmatrix}
T_c(k) \\
T_x(k) \\
T_y(k) \\
T_z(k)
\end{bmatrix}
= \begin{bmatrix}
\frac{1}{\text{VEL}}[(x(k)+d/2)^2+(y(k)+d/2)^2+(z(k)+d/2)^2]^{1/2}+v(k) \\
\frac{1}{\text{VEL}}[(x(k)-d/2)^2+(y(k)+d/2)^2+(z(k)+d/2)^2]^{1/2}+v(k) \\
\frac{1}{\text{VEL}}[(x(k)+d/2)^2+(y(k)-d/2)^2+(z(k)+d/2)^2]^{1/2}+v(k) \\
\frac{1}{\text{VEL}}[(x(k)+d/2)^2+(y(k)+d/2)^2+(z(k)-d/2)^2]^{1/2}+v(k)
\end{bmatrix}
\]

The measurement noises, \( v(k) \)'s, are assumed to be zero-mean and independent with a covariance matrix

\[
R(k) = \begin{bmatrix}
\sigma^2_{T_c} & 0 & 0 & 0 \\
0 & \sigma^2_{T_x} & 0 & 0 \\
0 & 0 & \sigma^2_{T_y} & 0 \\
0 & 0 & 0 & \sigma^2_{T_z}
\end{bmatrix}
\]
Equation (3.3a) can be used to give the linearized observation matrix. When the derivatives are taken and evaluated at the predicted state values \( x(k/k-1) = x'(k) \) the result is

\[
H(k) = \frac{1}{\text{VEL}} \begin{bmatrix}
\frac{x'(k)+d/2}{\text{DEN1}} & 0 & \frac{y'(k)+d/2}{\text{DEN2}} & 0 & \frac{z'(k)+d/2}{\text{DEN3}} \\
\frac{x'(k)-d/2}{\text{DEN2}} & 0 & \frac{y'(k)+d/2}{\text{DEN3}} & 0 & \frac{z'(k)+d/2}{\text{DEN4}} \\
\frac{x'(k)+d/2}{\text{DEN3}} & 0 & \frac{y'(k)-d/2}{\text{DEN4}} & 0 & \frac{z'(k)-d/2}{\text{DEN4}} \\
\end{bmatrix}
\] (4.3)

where

\[
\text{DEN1} = \left[ (x'(k)+d/2)^2 + (y'(k)+d/2)^2 + (z'(k)+d/2)^2 \right]^{1/2}
\]
\[
\text{DEN2} = \left[ (x'(k)-d/2)^2 + (y'(k)+d/2)^2 + (z'(k)+d/2)^2 \right]^{1/2}
\]
\[
\text{DEN3} = \left[ (x'(k)+d/2)^2 + (y'(k)-d/2)^2 + (z'(k)+d/2)^2 \right]^{1/2}
\]
\[
\text{DEN4} = \left[ (x'(k)+d/2)^2 + (y'(k)+d/2)^2 + (z'(k)-d/2)^2 \right]^{1/2}
\]

The torpedo dynamics used for the tracking problem are assumed to be \( 1/s^2 \) with estimations on five states \( x \) position, \( x \) velocity, \( y \) position, \( y \) velocity and \( z \) position (height of torpedo above hydrophone array).

The means of the random excitation and random noise are assumed to be zero, i.e.,
\[ E[w(k)] = 0 \]
\[ E[v(k)] = 0 \]

Four measurements are taken every 1.31 seconds, which is one time slot, and with this sampling time the \( 1/s^2 \) plant has state transition, (PHI) and gamma, (Gamma) matrices equal to:

\[
\Phi = \begin{bmatrix}
1 & T & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & T & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix} \quad (4.4)
\]

and

\[
\Gamma = \begin{bmatrix}
T^2/2 & 0 & 0 \\
T & 0 & 0 \\
0 & T^2/2 & 0 \\
0 & T & 0 \\
0 & 0 & T
\end{bmatrix} \quad (4.5)
\]

The \( \Phi \) matrix, Q matrix, R matrix, and H matrix are then used in the Kalman filter equations (3.4).
B. THE SEQUENTIAL EXTENDED KALMAN FILTER

In the sequential approach, the basic Kalman filter equations (3.4) must be modified. Calculations are performed on each of the four independent transit times in the following order: $T_c, T_x, T_y$ and $T_z$ for each 1.31 second time slot. The estimate of the states $x(k/k)$, based on one transit time measurement are used as the prediction $x(k/k-1)$ for the calculations on the next measurement. Thus for the first time measurement $T_c$ only the first row of the linearizing $H$ matrix is calculated.

Next the first gain column corresponding to the first time measurement $T_c$ is calculated by

$$
G_{icol} = \frac{P(k/k-1) H_i^{T} \text{row} - P(k/k-1) H_i^{T} \text{row} + R_{ii}}{H_i \text{row} P(k/k-1) H_i^{T} \text{row} + R_{ii}}
$$

where $i = 1 \text{ to } 4$ corresponding to the four measured transit times. Thus, the first row of the $H$ matrix is used to calculate the first column of the gain matrix with both corresponding to the first measured time $T_c$.

Next, an estimate of the particular observation time is calculated using equation (3.4f) evaluated at the predicted state $x(k/k-1)$.

The difference between observed transit time and the estimated transit times forms the residual which is used in the estimate equation.
This equation gives an estimate of the states based on one of the four time measurements.

Next, covariance of error is calculated based on one measurement by

\[ P_i = [I - G_{icol} H_{irown}] P_{i-1} \] (4.8)

where

- \( I \) = identity matrix
- \( P_{i-1} \) = the covariance matrix calculated from the previous transit time measurement or if \( i = 1 \), the prediction \( P(k/k-1) \).

After the first iteration, \( x_1 \) becomes \( x(k/k-1) \) and \( P_1 \) becomes \( P(k/k-1) \) for the second iteration which calculates the estimate of the states based on the second measurement \( T_x \).

After four iterations (\( k = 4 \)), \( x_4 \) becomes the estimate for the time slot \( x(k/k) \) and \( P_4 \) becomes the covariance error \( P(k/k) \).

The predictions for the next time slot are calculated using equations (3.4a) and (3.4d). This process is repeated for each time slot.
C. OPTIMAL SMOOTHING PROCESS

During the running of the Extended Kalman filter and smoothing routine, after the forward filter pass for each time slot (except the first), the smoothing subroutine is called. By using the present and previous filtered estimate of \( x(t) \), a smoothed estimate of previous \( x(t) \) is calculated. This process is repeated for each past time slot.

Solution of the equations (3.5) proceeds as follows: As an example, and because it is slightly easier to see when actual times are used, suppose \( N = 30 \). On the forward filter pass, the values \( \hat{x}(k/k) \), \( \hat{x}(k/k-1) \), \( P(k/k) \), and \( P(k/k-1) \) would be computed and stored. On the final iteration of the forward pass, with \( k = N = 30 \),

\[
\hat{x}(30/30) = \hat{x}(30/29) + G(30) \left[ z(30) - H \hat{x}(30/29) \right]
\]

i.e., we have computed and stored \( \hat{x}(30/30) \).

Now, the smoothing process starts in the reverse direction. Decrement \( k \) to \( k = N - 1 = 29 \), then

\[
\hat{x}(29/30) = \hat{x}(29/29) + A(29) \left[ \hat{x}(30/30) - \hat{x}(30/29) \right]
\]

stored stored stored

and

\[
A(29) = P(29/29) \hat{T} P(30/29)^{-1}
\]

stored stored

22
Let \( k = N - 2 = 28 \), then

\[
\hat{x}(28/30) = \hat{x}(28/28) + A(28) [\hat{x}(29/30) - \hat{x}(29/28)]
\]

stored \hspace{1cm} computed \hspace{1cm} stored
last iteration

and \( A(28) = P(28/28) \cdot P(29/28)^{-1} \)

stored \hspace{1cm} stored

Also, for each of the two preceding iterations,

\[
P(29/30) = P(29/29) + A(29)[P(30/30) - P(30/29)] A^T(29)
\]

stored \hspace{1cm} computed \hspace{1cm} stored \hspace{1cm} stored \hspace{1cm} computed

\[
\]

stored \hspace{1cm} computed \hspace{1cm} stored \hspace{1cm} stored \hspace{1cm} computed

23
V. TESTING AND SIMULATION

A. DESCRIPTION

The sequential Extended Kalman Filter and Smoothing routine is tested using simulated torpedo tracks. A variety of track scenarios were produced to test the filter and smoothing performance during single and multiple arrays tracking.

Computer generated tracks were tested in the first series of straight running, constant depth and constant velocity torpedoes. A variety of track scenarios were used transiting through multiple quadrants including:

1. Crossing north of the array.
2. Crossing diagonally through the array.

The next series of tests demonstrates the ability of the filter to track through the areas of multiple arrays including:

1. Crossing above the arrays.
2. Crossing diagonally through the arrays.

All runs were made with a variety of initialization errors in position and velocity.

Zero mean Gaussian noise is added to corrupt the observed transit times for all runs.
B. THE GATING SCHEME

The operation of the filter may be adversely affected by large measurement noise. One error of a relatively large magnitude could invalidate the filtered output for many subsequent time slots. Before random measurement noise and random excitations could be added to the observed times for testing, a form of protection was designed to guard against catastrophic failure. This protection is provided by establishing limits of acceptability for each of the measurements.

Measurement errors can occur because of many factors including an error in the transit time of the acoustic pulse primarily due to the receipt of multipath signals that have bounced off the surface, bottom or different density layers.

A three-sigma gate was designed using the covariance of measurement noise (R) and the covariance of estimation error (P(k/k)).

For each calculation of a state estimate (x(k/k)), the largest positional covariance of error was used, either x, y or z, and converted to time in seconds using the average velocity of sound in water for Dabob Bay, 4860 ft/sec. The gate then was written for each time measurement i = 1 to 4:

\[
GATE = \sqrt{\frac{P(k/k)_{\text{largest}}}{(4860.)^2} + R_{11}}
\]

25
The gate expands or decreases depending on the confidence level of the position estimate and the transit time. If ZDIFF, which is the difference between the actual transit time received and the predicted transit time to a particular hydrophone, exceeds the gate, the measurement is considered unacceptable and the filter gain is set to zero causing the filter to ignore the data and take the prediction of the states as the estimate

\[ x(k/k) = x(k/k-1) \]

An invalid time measurement zeros only the gain column for that particular hydrophone causing only that hydrophone's data to be ignored.

C. MULTIPLE ARRAY TRACKING

Initial tests were performed on tracks in the area of one array. In order to more closely simulate a typical run on the range, a scheme was designed to track the torpedo through multiple arrays.

First, a coordinate system is defined as shown in Figure 3. The center of the coordinate system is geographically near the entrance to Dabob Bay in the simulation. Array number 6 is the closest array to be coordinate center. In the simulation array 1 is at 36,000 feet from coordinate center and array 6 is 6,000 feet. The C hydrophone is assumed to be the axis location of each array. Then each X
Coordinate System for Multiple Array Tracking

<table>
<thead>
<tr>
<th>C HYDRO</th>
<th>X HYDRO</th>
<th>V HYDRO</th>
<th>Z HYDRO</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Y</td>
<td>Z</td>
<td>X</td>
</tr>
<tr>
<td>36000</td>
<td>6000</td>
<td>0</td>
<td>36030</td>
</tr>
<tr>
<td>30000</td>
<td>6000</td>
<td>0</td>
<td>30030</td>
</tr>
<tr>
<td>24000</td>
<td>6000</td>
<td>0</td>
<td>24030</td>
</tr>
<tr>
<td>18000</td>
<td>6000</td>
<td>0</td>
<td>18030</td>
</tr>
<tr>
<td>12000</td>
<td>6000</td>
<td>0</td>
<td>12030</td>
</tr>
<tr>
<td>6000</td>
<td>6000</td>
<td>0</td>
<td>6030</td>
</tr>
</tbody>
</table>

HYDRO—Hydrophone Location Matrix

Figure 3
position for the X hydrophone in each array is $X_0 + 30$, each Y position for the Y hydrophone is $Y_0 + 30$, and each Z position for the Z hydrophone is $Z_0 + 30$. These 72 positions, an XYZ position for each of 4 hydrophones in 6 arrays, were placed into a 6 x 12 matrix HYDRO and referenced throughout the routine.

The geometry centered on each array is taken out of the problem and the target position is based on a central reference.

The non-linear time equation becomes

$$T = \frac{1}{\text{VEL}} \sqrt{(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2}$$

where $x_0$, $y_0$, or $z_0$ is the position of a particular hydrophone and array being used.

The decision parameter used to determine the switching from array to array is a straight handoff. If the predicted x position is greater than 3,000 feet from the array in use, then an index (18) is incremented and the next row of HYDRO is implemented. This placed into the routine the x, y and z positions of the hydrophones in the next array. The handoff can easily be utilized in real range operations, as the transit times from adjacent arrays are present at the computer for a particular time slot. For simulation, it is assumed that in all the arrays each axis pointed in the same direction. In actual range operations each array is tilted
about both the X and Y axis. Since the true transit times are derived in a tilted coordinate frame, the filter's estimate of transit time must also be calculated in a tilted coordinate frame. The tilt angle measurements along with the level rectangular coordinates of the array with respect to the central rectangular coordinate system can be input into the matrix HYDRO to rotate the coordinates of each hydrophone in the array.
VI. TEST RESULTS

A. SERIES ONE

Figure 5 shows the true trajectory of the torpedo in the horizontal X-Y plane during a straight run through single array. Torpedo velocity is 50 knots in the x-direction. Initial position errors are set to 25 feet for X and Y. Velocity errors are set to zero. Figures 6, 7 and 8 depict the position errors for both Kalman filter and Smoothing. Measurement noise is added to all runs. The steady state X and Y position errors ranged between -6 and +9 feet throughout the trajectory for Kalman filter and -2 and +4 feet for smoothing. The position errors are computed by subtracting the filter position estimate, x(k/k), for Kalman filter and x(k/N), for smoothing, from the computer generated true position for each time slot. Figures 9, 10, 11, 12, and 13 depict the mean square estimation errors of states. These estimation errors are obtained by taking the appropriate diagonal terms of the covariance matrix and smoothed covariance matrix.

B. SERIES TWO

Figure 14 shows the true trajectory of the torpedo in the horizontal X-Y plane, during a crossing run through single array. Torpedo velocity is 40 knots in X-direction.
and 25 knots in Y-direction. The torpedo depth is maintained at 300 feet. Figures 15, 16, and 17 depict the position errors. Since the initial position errors are set to zero, the position errors ranged between -3 and +4 feet. Figures 18, 19, 20, 21, and 22 depict the mean square estimation errors of states.

C. SERIES THREE

Figure 23 shows the true trajectory of the torpedo in the horizontal X-Y plane, during a straight run through multiple array. Because of storage problem of the computer, the runs through the multiple array were made for 190 time slots. Torpedo velocity is 50 knots in X-direction and the depth is 100 feet. Figures 24, 25, and 26 show the position errors ranged between -6 and 17 feet. Figures 27, 28, 29, 30, and 31 depict the mean square estimation errors of states. Figure 32 shows the error ellipsoids superimposed at every eighteenth observation. The error ellipsoids are expanded to twenty-five times their true value in order that they may be seen. The error ellipsoids provide a geometric interpretation of the behavior of the estimator. Before the hand-off point, at the ninetyth time slot, the major axis rotation of the error ellipsoid and magnitude of the axis were -16.98 degrees and 43.28 feet, respectively. After the hand-off point major axis rotation became 25.3 degrees, and its magnitude became 1.6 feet. When the magnitude of an
axis of the ellipsoid decreases, the conclusion is that the error in the estimate decreases, because the observation from the new array has an error covariance ellipse rotated over 40 degrees from the present covariance (-16.94 degree) of the estimate. The ellipsoids are extremely narrow. When combined the resultant covariance is reduced greatly.

Figure 4 depicts the result of reduction and rotation of the ellipsoids. Figure 33 shows the error ellipsoids before and after the hand-off point.

Figure 4. Result of Rotation and Reduction of the Error Ellipsoids
D. SERIES FOUR

Figure 34 shows the true trajectory of the torpedo in the horizontal X-Y plane, during a crossing run through multiple array. Torpedo velocity is 50 knots in X-direction and 40 knots in Y-direction. The torpedo depth is maintained at 300 feet. Initial position errors are set to 25 feet for X and Y and initial velocity errors are set to 5 knots. Figures 35, 26, and 37 show the position and depth errors. Since the initial position and velocity errors are set to 25 feet and 5 knots, the big position errors were taken at the beginning of the run. These values were ignored from the figures in order to see clearly to the rest of the run. Figures 38, 39, 40, 41, and 42 show the mean square estimation errors of states for both filtered and smoothed.
VII. CONCLUSIONS

The sequential Extended Kalman Filter and Smoothing satisfactorily provided real time estimates of torpedo position and depth. The average of steady state position and depth errors ranged between 3 and 1 feet for torpedo tracks within the specified radial tracking range after Kalman filter. These errors had a range of around 1 foot after smoothing.

The filter performance was dependent on system noise and the distance the torpedo was from the hydrophone array and the smoothed estimates of states were better than or equal to the filter estimates.

Implementation at the range computer facilities can be accomplished by real time Kalman Filtering and post run Smoothing of the raw time data. Future tests should include evaluating filter performance using trajectories generated from actual torpedo runs on the Dabob test range. These tests would verify the adequacy of the noise model in the filter and the ability of the software to edit erroneous transit time measurements.

The rotation and reduction of the error ellipsoids (i.e., the filter error covariance) was most instructive and gave much insight into the performance of the filter.
Figure 5. True Trajectory of the Torpedo During a Straight Run Through Single Array (Right to Left)

Note: Hydrophone array is located at coordinates (36000, 0)
Figure 6. Error in Torpedo X-Position During a Straight Run Through Single Array
Figure 7. Error in Torpedo Y-Position During a Straight Run Through Single Array

Δ - After Kalman Filter
○ - After Smoothing
Figure 8. Error in Torpedo Z-Position During a Straight Run Through Single Array
Figure 9. Mean Square Estimation Error (ft.$^2$) in X-Position During a Straight Run Through Single Array
Figure 10. Mean Square Estimation Error (ft.$^2$/sec.$^2$) in X-Velocity During a Straight Run Through Single Array
Figure 11. Mean Square Estimation Error (ft.²) in Y-Position During a Straight Run Through Single Array
Figure 12. Mean Square Estimation Error (ft.$^2$/sec.$^2$) in Y-Velocity During a Straight Run Through Single Array

$\Delta - P_{44}(k/k)$

$\circ - P_{44}(k/N)$
Figure 13. Mean Square Estimation Error ($\text{ft.}^2$) in Z-Position During a Straight Run Through Single Array
<table>
<thead>
<tr>
<th>TIME</th>
<th>TRUE-X</th>
<th>X-AFTER KALMAN</th>
<th>X-AFTER SMOOTHING</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>40000.0000</td>
<td>40004.3750</td>
<td>40002.1367</td>
</tr>
<tr>
<td>2</td>
<td>39934.5000</td>
<td>39936.9297</td>
<td>39936.3125</td>
</tr>
<tr>
<td>3</td>
<td>39869.0000</td>
<td>39868.2031</td>
<td>39870.5156</td>
</tr>
<tr>
<td>4</td>
<td>39803.5000</td>
<td>39802.1016</td>
<td>39804.7383</td>
</tr>
<tr>
<td>5</td>
<td>39738.0000</td>
<td>39738.0977</td>
<td>39738.9102</td>
</tr>
<tr>
<td>6</td>
<td>39672.5000</td>
<td>39672.0000</td>
<td>39673.1056</td>
</tr>
<tr>
<td>7</td>
<td>39607.0000</td>
<td>39605.4492</td>
<td>39607.3359</td>
</tr>
<tr>
<td>8</td>
<td>39541.5000</td>
<td>39538.3164</td>
<td>39541.5937</td>
</tr>
<tr>
<td>9</td>
<td>39476.0000</td>
<td>39473.9570</td>
<td>39475.8516</td>
</tr>
<tr>
<td>10</td>
<td>39410.5000</td>
<td>39408.0937</td>
<td>39410.1250</td>
</tr>
<tr>
<td>11</td>
<td>39345.0000</td>
<td>39342.6172</td>
<td>39344.5297</td>
</tr>
<tr>
<td>12</td>
<td>39279.5000</td>
<td>39277.1523</td>
<td>39278.7650</td>
</tr>
<tr>
<td>13</td>
<td>39214.0000</td>
<td>39211.5937</td>
<td>39213.1289</td>
</tr>
<tr>
<td>14</td>
<td>39148.5000</td>
<td>39146.7461</td>
<td>39147.4531</td>
</tr>
<tr>
<td>15</td>
<td>39083.0000</td>
<td>39082.1953</td>
<td>39081.8086</td>
</tr>
<tr>
<td>16</td>
<td>39017.5000</td>
<td>39015.8594</td>
<td>39016.2429</td>
</tr>
<tr>
<td>17</td>
<td>38952.0000</td>
<td>38949.3164</td>
<td>38950.6367</td>
</tr>
<tr>
<td>18</td>
<td>38886.5000</td>
<td>38884.5391</td>
<td>38884.9222</td>
</tr>
<tr>
<td>19</td>
<td>38821.0000</td>
<td>38819.3555</td>
<td>38819.4336</td>
</tr>
<tr>
<td>20</td>
<td>38755.5000</td>
<td>38753.9141</td>
<td>38753.9219</td>
</tr>
<tr>
<td>21</td>
<td>38690.0000</td>
<td>38688.0089</td>
<td>38686.3711</td>
</tr>
<tr>
<td>22</td>
<td>38624.5000</td>
<td>38622.3086</td>
<td>38622.6281</td>
</tr>
<tr>
<td>23</td>
<td>38559.0000</td>
<td>38555.3867</td>
<td>38557.3396</td>
</tr>
<tr>
<td>24</td>
<td>38493.5000</td>
<td>38490.3984</td>
<td>38491.8750</td>
</tr>
<tr>
<td>25</td>
<td>38428.0000</td>
<td>38425.1250</td>
<td>38426.4023</td>
</tr>
<tr>
<td>26</td>
<td>38362.5000</td>
<td>38360.2617</td>
<td>38360.9609</td>
</tr>
<tr>
<td>27</td>
<td>38297.0000</td>
<td>38296.1836</td>
<td>38295.9430</td>
</tr>
<tr>
<td>28</td>
<td>38231.5000</td>
<td>38229.8984</td>
<td>38220.1172</td>
</tr>
<tr>
<td>29</td>
<td>38166.0000</td>
<td>38163.2695</td>
<td>38164.1942</td>
</tr>
<tr>
<td>30</td>
<td>38100.5000</td>
<td>38098.9062</td>
<td>38099.2891</td>
</tr>
</tbody>
</table>
### TABLE 2

Y-Position (ft.) Results of Straight Run Through Single Array

<table>
<thead>
<tr>
<th>TIME</th>
<th>TRUE-Y</th>
<th>Y-AFTER KALMAN</th>
<th>Y-AFTER SMOOTHING</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8000.000</td>
<td>7991.1758</td>
<td>7995.6562</td>
</tr>
<tr>
<td>2</td>
<td>8000.000</td>
<td>7993.1875</td>
<td>7996.3359</td>
</tr>
<tr>
<td>3</td>
<td>8000.000</td>
<td>8001.5508</td>
<td>7997.0117</td>
</tr>
<tr>
<td>4</td>
<td>8000.000</td>
<td>8002.7852</td>
<td>7998.6523</td>
</tr>
<tr>
<td>5</td>
<td>8000.000</td>
<td>7999.8516</td>
<td>7998.2578</td>
</tr>
<tr>
<td>6</td>
<td>8000.000</td>
<td>8001.0273</td>
<td>7998.8086</td>
</tr>
<tr>
<td>7</td>
<td>8000.000</td>
<td>8002.8359</td>
<td>7999.3520</td>
</tr>
<tr>
<td>8</td>
<td>8000.000</td>
<td>8003.7891</td>
<td>7999.8047</td>
</tr>
<tr>
<td>9</td>
<td>8000.000</td>
<td>8003.6133</td>
<td>8000.2148</td>
</tr>
<tr>
<td>10</td>
<td>8000.000</td>
<td>8004.1641</td>
<td>8000.5742</td>
</tr>
<tr>
<td>11</td>
<td>8000.000</td>
<td>8004.0234</td>
<td>8000.8906</td>
</tr>
<tr>
<td>12</td>
<td>8000.000</td>
<td>8003.8906</td>
<td>8001.1680</td>
</tr>
<tr>
<td>13</td>
<td>8000.000</td>
<td>8003.9023</td>
<td>8001.4102</td>
</tr>
<tr>
<td>14</td>
<td>8000.000</td>
<td>8002.6914</td>
<td>8001.5620</td>
</tr>
<tr>
<td>15</td>
<td>8000.000</td>
<td>8001.7072</td>
<td>8001.7383</td>
</tr>
<tr>
<td>16</td>
<td>8000.000</td>
<td>8002.4609</td>
<td>8001.9102</td>
</tr>
<tr>
<td>17</td>
<td>8000.000</td>
<td>8003.9609</td>
<td>8002.0156</td>
</tr>
<tr>
<td>18</td>
<td>8000.000</td>
<td>8002.6680</td>
<td>8002.0547</td>
</tr>
<tr>
<td>19</td>
<td>8000.000</td>
<td>8002.2305</td>
<td>8002.1172</td>
</tr>
<tr>
<td>20</td>
<td>8000.000</td>
<td>8002.1562</td>
<td>8002.1680</td>
</tr>
<tr>
<td>21</td>
<td>8000.000</td>
<td>8002.6211</td>
<td>8002.1523</td>
</tr>
<tr>
<td>22</td>
<td>8000.000</td>
<td>8002.7734</td>
<td>8002.1016</td>
</tr>
<tr>
<td>23</td>
<td>8000.000</td>
<td>8004.5273</td>
<td>8002.0506</td>
</tr>
<tr>
<td>24</td>
<td>8000.000</td>
<td>8003.8164</td>
<td>8001.9766</td>
</tr>
<tr>
<td>25</td>
<td>8000.000</td>
<td>8002.3750</td>
<td>8001.9633</td>
</tr>
<tr>
<td>26</td>
<td>8000.000</td>
<td>8002.5586</td>
<td>8001.7530</td>
</tr>
<tr>
<td>27</td>
<td>8000.000</td>
<td>8000.9180</td>
<td>8001.6406</td>
</tr>
<tr>
<td>28</td>
<td>8000.000</td>
<td>8001.7578</td>
<td>8001.5117</td>
</tr>
<tr>
<td>29</td>
<td>8000.000</td>
<td>8002.9258</td>
<td>8001.3750</td>
</tr>
<tr>
<td>30</td>
<td>8000.000</td>
<td>8001.6562</td>
<td>8001.2344</td>
</tr>
</tbody>
</table>
### TABLE 3

Z-Position (ft.) Results of Straight Run Through Single Array

<table>
<thead>
<tr>
<th>TIME</th>
<th>TRUE-Z</th>
<th>Z-AFTER KALMAN</th>
<th>Z-AFTER SMOOTHING</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100.0000</td>
<td>100.1117</td>
<td>100.6952</td>
</tr>
<tr>
<td>2</td>
<td>100.0000</td>
<td>99.7705</td>
<td>100.6971</td>
</tr>
<tr>
<td>3</td>
<td>100.0000</td>
<td>99.1682</td>
<td>100.6939</td>
</tr>
<tr>
<td>4</td>
<td>100.0000</td>
<td>99.5596</td>
<td>100.7065</td>
</tr>
<tr>
<td>5</td>
<td>100.0000</td>
<td>98.1741</td>
<td>100.6970</td>
</tr>
<tr>
<td>6</td>
<td>100.0000</td>
<td>97.8086</td>
<td>100.7122</td>
</tr>
<tr>
<td>7</td>
<td>100.0000</td>
<td>97.7114</td>
<td>100.7122</td>
</tr>
<tr>
<td>8</td>
<td>100.0000</td>
<td>97.6788</td>
<td>100.7305</td>
</tr>
<tr>
<td>9</td>
<td>100.0000</td>
<td>98.0196</td>
<td>100.7323</td>
</tr>
<tr>
<td>10</td>
<td>100.0000</td>
<td>98.4320</td>
<td>100.7464</td>
</tr>
<tr>
<td>11</td>
<td>100.0000</td>
<td>98.6859</td>
<td>100.7537</td>
</tr>
<tr>
<td>12</td>
<td>100.0000</td>
<td>98.8770</td>
<td>100.7699</td>
</tr>
<tr>
<td>13</td>
<td>100.0000</td>
<td>100.9539</td>
<td>100.7732</td>
</tr>
<tr>
<td>14</td>
<td>100.0000</td>
<td>101.0273</td>
<td>100.7662</td>
</tr>
<tr>
<td>15</td>
<td>100.0000</td>
<td>101.9257</td>
<td>100.7623</td>
</tr>
<tr>
<td>16</td>
<td>100.0000</td>
<td>101.6473</td>
<td>100.7589</td>
</tr>
<tr>
<td>17</td>
<td>100.0000</td>
<td>101.3949</td>
<td>100.7525</td>
</tr>
<tr>
<td>18</td>
<td>100.0000</td>
<td>101.3649</td>
<td>100.7401</td>
</tr>
<tr>
<td>19</td>
<td>100.0000</td>
<td>101.5895</td>
<td>100.7365</td>
</tr>
<tr>
<td>20</td>
<td>100.0000</td>
<td>101.5225</td>
<td>100.7264</td>
</tr>
<tr>
<td>21</td>
<td>100.0000</td>
<td>101.1333</td>
<td>100.7166</td>
</tr>
<tr>
<td>22</td>
<td>100.0000</td>
<td>101.0129</td>
<td>100.7104</td>
</tr>
<tr>
<td>23</td>
<td>100.0000</td>
<td>101.1896</td>
<td>100.7113</td>
</tr>
<tr>
<td>24</td>
<td>100.0000</td>
<td>101.3688</td>
<td>100.7001</td>
</tr>
<tr>
<td>25</td>
<td>100.0000</td>
<td>101.1070</td>
<td>100.6941</td>
</tr>
<tr>
<td>26</td>
<td>100.0000</td>
<td>100.6891</td>
<td>100.6891</td>
</tr>
<tr>
<td>27</td>
<td>100.0000</td>
<td>100.7637</td>
<td>100.6861</td>
</tr>
<tr>
<td>28</td>
<td>100.0000</td>
<td>100.7554</td>
<td>100.6837</td>
</tr>
<tr>
<td>29</td>
<td>100.0000</td>
<td>100.5455</td>
<td>100.6821</td>
</tr>
<tr>
<td>30</td>
<td>100.0000</td>
<td>100.5455</td>
<td>100.6821</td>
</tr>
</tbody>
</table>
Figure 14. True Trajectory of the Torpedo During a Straight Run Through Single Array
Figure 15. Error in Torpedo X-Position During a Straight Run Through Single Array
Figure 16. Error in Torpedo Y-Position During a Straight Run Through Single Array

Δ - After Kalman Filter
○ - After Smoothing
Figure 17. Error in Torpedo Z-Position During a Straight Run Through Single Array
Figure 18. Mean Square Estimation Error (ft.\(^2\)) in X-Position During a Straight Run Through Single Array.
Figure 19. Mean Square Estimation Error (ft.²/sec.²) in X-Velocity During a Straight Run Through Single Array.
Figure 20. Mean Square Estimation Error (ft.\(^2\)) in Y-Position During a Straight Run Through Single Array
Figure 21. Mean Square Estimation Error (ft.\(^2\)/sec.\(^2\)) in Y-Velocity During a Straight Run Through Single Array
Figure 22. Mean Square Estimation Error \( (\text{ft.}^2) \) in Z-Position During a Straight Run Through Single Array
Figure 23. True Trajectory of the Torpedo During a Straight Run Through Multiple Array
Figure 25. Error in Torpedo Y-Position During a Straight Run Through Multiple Array

Δ - After Kalman Filter
o - After Smoothing
Figure 26. Error in Torpedo Z-Position During a Straight Run Through Multiple Array
Figure 27. Mean Square Estimation Error (ft.²) in X-Position During a Straight Run Through Multiple Array
Figure 28. Mean Square Estimation Error (ft.\(^2\)/sec.\(^2\)) in X-Velocity During a Straight Run Through Multiple Array
Figure 29. Mean Square Estimation Error (ft.\textsuperscript{2}) in Y-Position During a Straight Run Through Multiple Array
Figure 30. Mean Square Estimation Error (ft.$^2$/sec.$^2$) in Y-Velocity During a Straight Run Through Multiple Array
Figure 31. Mean Square Estimation Error (ft.$^2$) in Z-Position During a Straight Run Through Multiple Array

\[ \Delta = P_{5,5}(k/k) \]
\[ o = P_{5,5}(k/N) \]
Figure 32. Error Ellipsoids Presented on Every Eighteenth Observation During a Straight Run Through Multiple Array
Figure 33. Error Ellipsoids Before and After the Hand-off Point During a Straight Run Through Multiple Array
Figure 34. True Trajectory of the Torpedo During a Straight Run Through Multiple Arrays
Δ - After Kalman Filter
○ - After Smoothing

Figure 35. Errors in Torpedo X-Position During a Straight Run Through Multiple Arrays
Figure 36. Errors in Torpedo Y-Position During a Straight Run Through Multiple Arrays
Figure 37. Errors in Torpedo Z-Position During a Straight Run Through Multiple Arrays
Figure 39. Mean Square Estimation Error (ft.\(^2\)/sec.\(^2\)) in X-Velocity During a Straight Run Through Multiple Arrays
Figure 40. Mean Square Estimation Error (ft.$^2$) in Y-Position During a Straight Run Through Multiple Arrays
Figure 42. Mean Square Estimation Error \((\text{ft.}^2)\) in Z-Position During a Straight Run Through Multiple Arrays
APPENDIX A

PROGRAM DESCRIPTION AND FEATURES

The sequential Extended Kalman Filter and Smoothing routine used for torpedo tracking is modularized for ease of implementation. The program is general in nature and many of the parameters of the filter are variable including:

a. The number of states in the filter -- N
b. The number of random forcing functions -- M
c. The number of measurements -- JS
d. Number of time slots -- JTIME

The constant matrices PHI, R, COVW, and GAMMA are initialized in the beginning of the program using data statements. The filter is initialized with $P(1/0)$ and $x(1/0)$ (initial covariance of estimation error and states) using subroutine INIT. The first estimate is at time 1 and continues until $ITIME = JTIME + 1$. True measurement times (ZI) are computed using either subroutines TRAJEC or TRAJC3, depending on whether single array or multiple array tracking is implemented. Either subroutine will compute four measurement times ($T_0, T_x, T_y, T_z$) for each time slot. The measurement times are corrupted by zero-mean, white Gaussian noise using the IBM-3033 subroutine GGNML. For each of the four time measurements the corresponding row of the
linearizing $H$ matrix is calculated using either subroutine
CHROW or CHROW3, depending on whether single array or
multiple array tracking is used. The corresponding gain
matrix column GI is then found. These row and column values
are utilized in forming the covariance of estimation error.
$PI$, for that particular time measurement. Next the estimate
of the observation time $Z\text{HAT}$ from that particular hydrophone
is formed using the subroutine CZHAT or CZHAT3, depending on
whether single or multiple array tracking is implemented.
The residual $Z\text{DIFF}(I) = Z\text{IC}(I) - Z\text{HAT}$, is then calculated.
Finally the estimate of the states $XI$ based on one time
measurement is calculated and the process is repeated for
the next measurement. After four iterations, $XI$ becomes the
state estimate $XKK$ and $PI$ becomes the updated covariance of
estimation error $PKK$, and the predictions of the states and
covariances $XKKM1$ and $PKKM1$ are formed. Finally, for each
time slot (except the first) smoothed state estimates, $XKKS$, and
covariances, $PKKS$, are formed using the subroutine
SMOOTH. PLOTP is used to generate line printer plots and
PLOTG is used to generate VERSATEC plots.

A. PROGRAM SUBROUTINES

A brief description of the subroutines are described
below:

1. TRAJEC -- This subroutine develops the torpedo
trajectory which is used as truth data for the filter. The
subroutine outputs true transit times, $Z_I(I)$, and the x, y, z positions, $T_D(I)$, of the torpedo for each time slot. The subroutine is used during single array tracking.

2. TRAJC3 -- This subroutine performs the same function as TRAJEC but is used only during multiple array tracking.

3. INIT -- This subroutine generates the initial state vector $x(0/-1)$ and initial covariance matrix $P(0/-1)$.

4. CHROW -- This subroutine computes the appropriate row of the linearizing $H$ matrix. Each row corresponds to one of the four transit time measurements, $T_c$, $T_x$, $T_y$, $T_z$. This subroutine is used during single array tracking.

5. CHROW3 -- This subroutine performs the same functions as CHROW but is used only during multiple array tracking.

6. CZHAT -- This subroutine computes the estimated transit times for the filter. Four transit times, $Z_HAT$, are calculated corresponding to each of the four true transit times $Z_I(I)$. This subroutine is used during single array tracking.

7. CZHAT3 -- Subroutine performs same functions as CZHAT however it is used only during multiple array tracking.

8. QFIND -- This subroutine develops an adaptive $Q$ matrix which is a function of the torpedo velocity. Three input variables defined in a data statement at the beginning of the program can be adjusted:
aa. SIGACC -- Maximum expected horizontal acceleration of the torpedo.

bb. SIGDIV -- Maximum expected change in vertical velocity.

c. SIGCC -- Maximum expected turn rate of the torpedo in the horizontal plane.

The values listed in the program were used and kept constant during the simulation tests. If the user desires not to use the adaptive Q subroutine, software code is provided at the beginning of the program to calculate a constant Q matrix.

9. GGNML -- This is an IBM-3033 subroutine contained in the IMSL library. The routine generates zero mean white Gaussian noise with an RMS value normalized to 1. The main program scales the noise and adds it to the transit time measurements.

10. PLOTP -- This is an IBM-3033 subroutine used to generate the line printer plots. Information on this subroutine can be obtained from the IMSL library.

11. PLOTG -- This is an IBM-3033 subroutine used to generate the VERSATEC plots. Information on this subroutine can be obtained from the IMSL library.

12. SMOOTH -- This subroutine computes the smoothed state estimates and covariances.
B. UTILITY PROGRAMS

These subroutines were designed to be used for repetitive matrix and vector manipulations:

1. PROD -- multiplying two matrices
2. MMULT -- multiplying a matrix and a vector
3. VMULT -- multiplying two vectors
4. TRANS -- transposing a matrix
5. ADD -- adding two matrices
6. SUB -- subtracting two matrices
7. RECIP -- inversing a matrix
APPENDIX B

SEQUENTIAL EXTENDED KALMAN FILTER AND PROGRAM LISTING

COMMON/THE2/PKMK1(5), PPK(5,5), XKK(5)

DATA N/5, M/3, X15/6, NZDIFF/6, JTIME/45/
1 SIGACC/0.0, SIG1V/0.0, SIGCC/0.0/  
2 HYDRO/36000, 30000, 24000, 18000, 12000, 6000, 6*6000/  
3 6*0.0, 36000, 30030, 24000, 18000, 12000, 6000, 6*6000  
4 6*0.0, 36000, 30030, 24000, 18000, 12000, 6000, 6*6000  
5 6*0.0, 36000, 30030, 24000, 18000, 12000, 7*6000, 6*6000/  

DATA PH1/1.0, 4*0.0, 1.31, 1.0, 5*0.0, 1.0, 4*0.0, 1.31, 1.0, 5*0.0, 1.0,  
2 R/1.0E-8.4, 4*0.0, 1.0E-8.4, 4*0.0, 1.0E-8.4,  
3 COVW/1.0, 4*0.0, 1.0, 4*0.0, 1.0,  
4 GAMMA/8.6, 1.31, 5*0.0, 8.6, 1.31, 5*0.0, 1.31/  

DATA STATEMENT DEFINED IN SUBROUTINE TRAJECT OR TRAJECT3

50DATR/7500., 1300.0., -40.0., 3*0.0, 3*0.0, 4.71, 174., 1.81,  
6690., 800.0/
THE NEXT THREE STATEMENTS ARE REQUIRED FOR
SUBROUTINE WHICH GENERATES WHITE NOISE

DOUBLE PRECISION DSEED
MR=4
DSEED=76869.DO
SIGCC=.SIGCC*3.14159/180.

LOAD (0/-1),P(0/-1)

CALL INIT(XKK1,PKK1)

GET TRANSPOSES

CALL TRANS(GAMMA,N,M,GAMMAT)
CALL TRANS(PHI,N,N,PHIT)
WRITE(6,163)

163 FORMAT('Q',5R MATRIX')
DO 264 I=1,4
264 WRITE(6,136)(R(I,J),J=1,4)
136 FORMAT(4F14.11)
WRITE(6,265)
265 FORMAT('O',5R COV W MATRIX')
DO 266 I=1,3
266 WRITE(6,267)(COVW(I,J),J=1,3)
267 FORMAT(3F11.6)

CALCULATE THE Q MATRIX

CALL PROD(GAMMA,COW,N,M,M,QTEMP)
CALL PROD(TEMP,GAMMAT,N,M,N,Q)
WRITE(6,100)
100 FORMAT('Q',5R Q MATRIX')
DO 101 LJ=1,N
101 WRITE(6,102)(Q(LJ),LJ=1,N)
102 FORMAT(1X,10F15.4)

START THE TIME SLOT LOOP AND SET ARRAY HANDOFF POINT

ITIME=JTIME+1
18=1
XT=36000.
SW=XT-3000.
DO 99 KK=1,ITIME
99 WRITE(6,6) KK
6 FORMAT ///1X,100(*'), 'TIME=',14//
C GET HYDROPHONE ARRAY COORDINATES

601 DO 600 I3=1,4
  I4=3*I3
  I5=I4+2
  I6=I4-1
  XB(I3)=HYDRO(I8,I5)
  YB(I3)=HYDRO(I8,I6)
600 ZB(I3)=HYDRO(I8,I4)
WRITE(*,602)

602 FORMAT('HYDROPHONE COORDINATES')
WRITE(*,6)XB
WRITE(*,6)YB
WRITE(*,6)ZB
IF(XKKM1(I) .GT. SW) GO TO 610
I0=I0+1
WRITE(*,759)18,KK
759 FORMAT('0',15('**'),'ARRAY',1X,I2,'STARTS TRACKING AT TIME',1X,I3,
  1 I15('**')/)
  SW=SW-6000.
  XT=XT-6000.
  GO TO 601
WRITE(*,67)
67 FORMAT('0','XKKM1')
DN 51 LA=1:N
51 WRITE(*,50)XKKM1(LA)
50 FORMAT('1X,5F14.4')

C COMPUTE THE TRUE TIMES AND TRUE POSITIONS

610 CALL TRAJC3(DATR,ZI,TD,XB,YB,ZB)
610 CALL TRAJEC(KK,DATR,ZI,TD)
WRITE(*,700) (ZI(J),J=1,4)
700 FORMAT('TRUE TIMES : ',4F14.5)
WRITE(*,701) (TC(J),J=1,3)
701 FORMAT('TRUE POSITIONS : ',3F11.4)
T2=(.25)*(ZI(1)+ZI(2)+ZI(3)+ZI(4))
IF(KK.EQ.1)T1=T2
PHI(3,4)=PHI(1,2)
PHI(2,1)=PHI(1,2)
PHI(4,3)=PHI(1,2)
T1=T2
A14=PHI(1,2)
TRUX(KK)=TD(1)
TRUY(KK)=TD(2)
TRUZ(KK)=TD(3)
C
DO 52 LA=1,N
  WRITE(6,50) XKKM1(LA)
C
52  FIRST GET HROW-CALCULATE GAIN-ESTIMATE-COVARIANCE OF
    ERROR BASED ON ONE TIME MEASUREMENT-TC,TX,TV, T1
C
DO 97 I=1,JS
  NZDIFF=4
  CALL CHROW3(I,HROW,XB,YB,ZB)
  CALL CMULT(PKKM1,HROW,N,N,NUM)
  CALL VMULT(HROW,GNUM,N,GTEMP)
  GDENOM=GTEMP+R(I,1)
  DO 16 IX=1,N
     GI(IX)=GNUM(IX)/GDENOM
16      THIS IS THE FIRST GAIN COLUMN
      CALCULATE THE COVARIANCE OF ERROR PI
C
DO 77 IP=1,N
  DO 79 JP=1,N
     PDUM(IP,JP)=(1.-GI(IP))*HROW(JP)
  IF(IP.EQ.JP)PDUM(IP,JP)=1.+PDUM(IP,JP)
79 CONTINUE
77 CONTINUE
C
      CALL PRCD(PDUM,PKKM1,N,N,N,P)
C
      CALL CHAT(1,ZHAT)
      CALL CHAT3(1,ZHAT,XB,YB,ZB)
C
      CALL WHITE NOISE SCALE AND ADD TO TRUE MEASUREMENT TIME
C
      CALL GGM1(LSEED,NR,RN)
      ZIC(I)=ZIC(I)+RN(I)*.0001
      ZIC(I)=ZIC(I)
      ZDIFF(I)=ZIC(I)-ZHAT
C
WRITE(6,90005)*ZDIFF(I)
90005 FORMATT(7X,*ZDIFF,F15.8)
C
      COMPUTE THE GATE FOR ERRONEOUS TIME MEASURMENTS
C
PK1=DABS(PI(1,1))
PK1=ABS(PI(1,1))
PK3=DABS(PI(3,3))
C  PK3=ABS(P1(3,3))
C  PK5=DABS(P1(5,5))
C  PK5=ABS(P1(5,5))
C IF(PK1.GE.PK3).AND.(PK1.GE.PK5)P=PK1
C IF(PK3.GE.PK1).AND.(PK3.GE.PK5)P=PK3
C IF(PK5.GE.PK1).AND.(PK5.GE.PK3)P=PK5
C RGATE=DSQRT(DABS(R(1,1)))
C RGATE=SQR(DABS(R(1,1)))
C GATE=3.*SQR(PGATE+RGATE)
C WRITE(6,9000)GATE,1,ZDIFF(1)
C 9000 FORMAT(1X,GATE=",F12.7,4X,ZDIFF(",I2,’)=’,F12.7)
C---------------------------------------------
C EDIT INVALID TIME MEASUREMENTS
C---------------------------------------------
C IF(ZDIFF(I).GT.GATE) GO TO 500
C WRITE(6,501)I
C 501 FORMAT(’Q’,GATE HAS BEEN EXCEEDED TIME’,I4)
C DO 502 LG=1,N
C 502 G1(LG)=0.0
C---------------------------------------------
C TAG INVALID TIME MEASUREMENT
C---------------------------------------------
C ZDIFF(I)=995.
C---------------------------------------------
C CALCULATE THE ESTIMATE BASED ON ONE MEASUREMENT
C---------------------------------------------
C 500 DO 17 IZ=1,N
C 17 WRITE(6,43)XI(IZ)=XKKM1(IZ)+G1(IZ)*ZDIFF(I)
C 1,’GY’,16X,’GTY’,17X,’GZ’,’0’)
C WRITE(6,44)ZI(IZ),ZHAT,ZDIFF,G1(1),G1(2),G1(3),G1(4),G1(5)
C 44 FORMAT(’8F14.5)
C WRITE(6,148)
C 148 FORMAT(’3X,’ZC’,’8X,’ZI’,’8X,’ZI’)
C WRITE(6,149)ZC(1),ZI(1),ZC
C 149 FORMAT(’3F14.5)
C WRITE(6,48)
C 48 FORMAT(’3X,’H1’,’8X,’H2’,’8X,’H3’,’7X,’H4’,’8X,’H5’)
C WRITE(6,49)HROW(1),HROW(2),HROW(3),HROW(4),HROW(5)
C 49 FORMAT(’5F14.6)
C WRITE(6,35)
C 35 FORMAT(’0’,’XI AND XKKM1’,)
C DO 33 LC=1,N
C 33 WRITE(6,34)XI(LC),XKKM1(LC)
C 34 FORMAT(’2F11.4)
C WRITE(6,59)
59 FORMAT(*8E14.4*
   DO 33 LK=1,N
53 WRITE(6,52) (PI(LK,LB), LB=1,N)
52 FORMAT(1X,5F14.4)
   WRITE(6,68)
68 FORMAT(*8E14.4*
   DO 71 LE=1,N
71 WRITE(6,72) (PKKM1(LE,LF), LF=1,N)
72 FORMAT(1X,5F14.4)
C
IF(I.EQ.4)GO TO 56
   DO 19 IQ=1,N
   XKKM1(IQ)=XI(IQ)
   DO 23 IQ=1,N
   PKKM1(IQ,JQ)=PI(IQ,JQ)
23 CONTINUE
   CONTINUE

NOTE-CALLED ORIGINAL X(0/1),XKKM1. UPDATED AFTER 1 MEASUREMENT CALLED IT XI, THEN MADE XKKM1=XI AND WENT THRU ITERATION AGAIN. AFTER YOU HAVE UPDATED XKKM1 FOR EACH MEASUREMENT XKK=XI AND PKK=PI

56 DO 57 ID=1,N
   XKK(ID)=XI(ID)
   XP6(ID,KK)=XI(ID)
   XKKM1(ID)=XI(ID)
   DO 58 JD=1,N
   PKK(ID,JD)=PI(ID,JD)
   PS(KK,DD,JD)=PI(ID,JD)
58 CONTINUE

XP7(KK)=(TRUX(KK)-XP6(1,KK))**2
XP8(KK)=(TRUX(KK)-XP6(3,KK))**2
XP9(KK)=(TRUX(KK)-XP6(5,KK))**2
IF(KK.NE.1) GO TO 666
XP10(KK)=XP7(KK)
XP11(KK)=XP8(KK)
XP12(KK)=XP9(KK)
GO TO 667

666 CONTINUE
   XP10(KK)=XP10(KK-1)+XP7(KK)
   XP11(KK)=XP11(KK-1)+XP8(KK)
   XP12(KK)=XP12(KK-1)+XP9(KK)
667 CONTINUE
XP13(KK)=0.5*KK*XP10(KK)
XP14(KK)=0.5*KK*XP11(KK)
XP15(KK)=0.5*KK*XP12(KK)

C
P1(KK)=P1(I,KK)
P2(KK)=P2(I,KK)
P3(KK)=P3(I,KK)
P4(KK)=P4(I,KK)
P5(KK)=P5(I,KK)

C
WRITE(6,61)I
   FORMAT(1X,2F10.4)
   WRITE(6,62)XK(KK),YK(KK)
   WRITE(6,63)ZK(KK)

C
WRITE(6,64)DO 41LD=1,N
   WRITE(6,65)X(KK),Y(KK),Z(KK)
   WRITE(6,66)XG(KK),YG(KK),ZG(KK)

C
RECALCULATE TIME MEASUREMENTS, FORM ABSOLUTE VALUE OF RESIDUALS
DO 81 I=1,N
C
EDIT INVALID TIME MEASUREMENTS FOR ADAPTIVE MANEUVER ROUTINE
IF (ZDIFF(I),GE,999.) GO TO 82
C
CALL CZHAT(KK)
CALL CZHAT3(KK)
C
ZDIFF(I)=ABS(ZIC(I)-ZHAT)
ZDIFF(I)=ABS(ZIC(I)-ZHAT)
GO TO 81

82
ZDIFF(I)=0.0
NZDIFF = NZDIFF- 1
81
CONTINUE
C
IF ALL TIME MEASUREMENTS EXCEED GATE BYPASS ADAPTIVE
MANEUVER ROUTINE

IF (NZDIFF,EQ,0.) GO TO 80
ZDIFFAV=(ZDIFF(1)+ZDIFF(2)+ZDIFF(3)+ZDIFF(4))/NZDIFF
WRITE(6,9900)
9900 FORMAT(' ', 'ZDIFAV')
WRITE(6,83)ZDIFAV
83 FORMAT(5X,E14.5)

C-- IF FILTER HAS NOT ACHIEVED STEADY STATE
C-- DO NOT PERFORM ADAPTIVE MANEUVERING
C--
C-- IF(KK.LE.4) GO TO 80
C--
C-- IF ZDIFAV MEETS CRITERIA TRANSFER OUT OF
C-- ADAPTIVE MANEUVER ROUTINE
C--
C-- IF(ZDIFAV.LE.100000-04) GO TO 80
C--
C-- INCREASE THE GAIN
C--
C-- CALL QFIND(KK,SIGACC,SIGDIV,SIGCC,A14,Q)
C-- CALL ADD(PKK,Q,N,N,PKKML)
C--
C-- PERFORM ADAPTIVE MANEUVERING BY REITERATING SAME TIME SLOT
C--
C0007 FORMAT(1X,'ADAPT')
GO TO 711
80 NDIFF=6
XDIF1(KK)=XKK(1)-TRUX(KK)
XDIF3(KK)=XKK(3)-TRUY(KK)
XDIF5(KK)=XKK(5)-TRUZ(KK)

C-- CALCULATE THE PREDICTIONS FOR PKKML
C--
SI=0.
1081 CONTINUE
C-- CALL QFIND(KK,SIGACC,SIGDIV,SIGCC,A14,Q)
C--
DO 765 I=1,N
765 WRITE(6,766)(Q(I,J),J=1,N)
766 FORMAT(1X,6E10.5)
C--
C-- CALL PROD(PHI,PKK,N,N,N,N,PHIPK)
C-- CALL PROD(PHIPK,PHIT,N,N,N,N,PKTEMP)
C-- CALL ADD(PKTEMP,Q,N,N,PKKML)
C--
C-- CALCULATE NEW XKKML
C--
C-- CALL MMULT(PHI,XKKML,N,N,N)
C-- IF(SI.EQ.0.) GO TO 1096
C--
WRITE(6,6669)
6669 FORMAT("X1 AND SMOOTHED XKKM1")
DO 6670 LC=1,N
6670 WRITE(6,6671) X1(LC),XKKM1(LC)
6671 FORMAT(1,2F11.4)
1096 CONTINUE
DO 41 IG=1,N
41 XP{IG,KK}=XKKM1{IG}
DO 38 II=1,N
DO 39 JJ=1,N
SSI(KK,II,JJ)=PKKM1{II,JJ}
SSI(KK,II,JJ)=Q{II,JJ}
SSI(KK,II,JJ)=PKK{II,JJ}
39 CONTINUE
38 CONTINUE
C---CALCULATE SMOOTHED ESTIMATES
C---
IF(KK.EQ.1)GO TO 8888
IF(SI.NE.0.)GO TO 8888
CALL SMOOTH(S1,SS1,PHI,P5,XP6,KK,N)
C
WRITE(6,9910)
9910 FORMAT(1,2F11.4)
DO 6666 LE=1,N
DO 6667 LK=1,N
PKK{LE,LK}=P1{KK-1,LE,LK}
6667 CONTINUE
6666 CONTINUE
C
DO 1095 LK=1,N
1095 WRITE(6,1089){PKK{LK,LE},LE=1,N)
6668 FORMAT(1X,5F14.4)
WRITE(6,1089)
1089 FORMAT(1,2F11.4)
DO 1084 I=1,N
XKKM{I}=XP{I,KK-1}
WRITE(6,1087) XKK{I}
1084 CONTINUE
1087 FORMAT(1X,F11.4)
XDI{KK}=ABS{XKK{MIN}-TRUX{KK-1})
C
IF(SI.EQ.1.)GO TO 1081
C
8888 X9{KK}=KK
P{1}{KK}=PKK{1,1}
P{2}{KK}=PKK{2,2}
P{3}{KK}=PKK{3,3}
P4P(KK) = PKK(4,4)  
P5P(KK) = PKK(5,5)

CONTINUE

\[ \begin{align*}
\text{DO 9992} \ LP = 1, \ JTIME \\
P1P1(LP) &= P1(LP, 1, 1) \\
P2P2(LP) &= P1(LP, 2, 2) \\
P3P3(LP) &= P1(LP, 3, 3) \\
P4P4(LP) &= P1(LP, 4, 4) \\
P5P5(LP) &= P1(LP, 5, 5)
\end{align*} \]

CONTINUE

\[ \begin{align*}
\text{DO 9993} \ LP = 1, \ JTIME \\
P1P5(LP) &= P5(LP, 1, 1) \\
P2P2(LP) &= P5(LP, 2, 2) \\
P3P3(LP) &= P5(LP, 3, 3) \\
P4P4(LP) &= P5(LP, 4, 4) \\
P5P5(LP) &= P5(LP, 5, 5)
\end{align*} \]

CONTINUE

\[ \begin{align*}
\text{DO 9994} \ LP = 1, \ JTIME \\
XP1S(LP) &= XP(1, LP) \\
XP2S(LP) &= XP(2, LP) \\
XP3S(LP) &= XP(3, LP) \\
XP4S(LP) &= XP(4, LP) \\
XP5S(LP) &= XP(5, LP)
\end{align*} \]

CONTINUE

\[ \begin{align*}
\text{DO 9995} \ LP = 1, \ JTIME \\
XP1K(LP) &= XP6(1, LP) \\
XP2K(LP) &= XP6(2, LP) \\
XP3K(LP) &= XP6(3, LP) \\
XP4K(LP) &= XP6(4, LP) \\
XP5K(LP) &= XP6(5, LP)
\end{align*} \]

CONTINUE

\[ \begin{align*}
\text{DO 9996} \ LP = 1, \ JTIME \\
\end{align*} \]
C XKER(LP)=ABS(TRUX(LP)-XP1K(LP))
C YKER(LP)=ABS(TRUY(LP)-XP3K(LP))
C ZKER(LP)=ABS(TRUZ(LP)-XP5K(LP))
C XKER(LP)=TRUX(LP)-XP1K(LP)
C YKER(LP)=TRUY(LP)-XP3K(LP)
C ZKER(LP)=TRUZ(LP)-XP5K(LP)

996 CONTINUE

C FIND ERROR BETWEEN TRUE POSITIONS AND SMOOTHED ESTIMATIONS.
C
DO 997 LP=1,ITIME
C XSER(LP)=ABS(TRUX(LP)-XP1S(LP))
C YSER(LP)=ABS(TRUY(LP)-XP3S(LP))
C ZSER(LP)=ABS(TRUZ(LP)-XP5S(LP))
C XSER(LP)=TRUX(LP)-XP1S(LP)
C YSER(LP)=TRUY(LP)-XP3S(LP)
C ZSER(LP)=TRUZ(LP)-XP5S(LP)

997 CONTINUE

C GET COVARIANCE ERROR ELLIPSOIDS
C
LX=1
DO 998 LP=18,ITIME,18
ANGL(LK)=0.5*ATAN(2.*P6P6(LP)/(P1P1(LP)-P3P3(LP)))
ANGLD(LK)=ANGL(LK)*180./3.14159
SIG1(LK)=(P1P1(LP)+P3P3(LP))/2.
SIG2(LK)=P6P6(LP)/SIN(2.*ANGL(LK))
SIGX(LK)=SIG1(LK)+SIG2(LK)
SIGY(LK)=SIG1(LK)-SIG2(LK)
SX(LK)=(SIGX(LK)**.5)*25.
SY(LK)=(SIGY(LK)**.5)*25.
LK=LK+1

998 CONTINUE
WRITE(6,9960)
9960 FORMAT('** ANGLES IN RADIAN')
WRITE(6,9963)
9963 FORMAT('** ANGLES IN DEGREE')
WRITE(6,9961)
9961 FORMAT('** SIGMA X')
WRITE(6,9962)
9962 FORMAT('** SIGMA Y')
WRITE(6,9963)
C
PT=3.14159265/12.
DO 44445 J = 1, 7
CT = COS(ANGL(J))
ST = SIN(ANGL(J))
DO 44446 I = 1, 25
XII(J,I) = ST*COS(PTXIII(I)) - ST*SIX(J)*COS(PTXIII(I))*CT
44446 CONTINUE
K = K + 18
44445 CONTINUE
DO 44447 LP = 1, 25
XP19(LP) = X801(LP)
XP29(LP) = X802(LP)
XP39(LP) = X803(LP)
XP49(LP) = X804(LP)
XP59(LP) = X805(LP)
XP69(LP) = X806(LP)
XP79(LP) = X807(LP)
Y19(LP) = Y81(LP)
Y29(LP) = Y82(LP)
Y39(LP) = Y83(LP)
Y49(LP) = Y84(LP)
Y59(LP) = Y85(LP)
Y69(LP) = Y86(LP)
Y79(LP) = Y87(LP)
44447 CONTINUE
C WRITE(6, 44448)
FORMAT('X OF ELIPSE')
WRITE(6, 44449)
FORMAT('Y OF ELIPSE')
WRITE(6, 44450)
FORMAT(8F9.2)
C WRITE(6, 49916)
FORMAT(I)
C CALL PLOT(5001, TRUX, TRUY, 201, 4)
C CALL PLOT(5001, XP9, XD91, 3, 1)
C CALL PLOT(5001, XP9, XD9X, 30, 3)
C WRITE(6, 90007)
C CALL PLOT(5001, XP9, PX, 30, 1)
C CALL PLOT(5001, XP9, PXK, 31, 3)
C WRITE(6, 90008)
C0008 FORMAT('!')
C C0009 FORMAT('!')
C C0010 FORMAT('!')
C C0011 FORMAT('!')
C C0012 FORMAT('!')
C C0013 FORMAT('!')
C C0014 FORMAT('!')
C C0015 FORMAT('!')
C C0016 FORMAT('!')
C C0017 FORMAT('!')
C C0018 FORMAT('!')
C C0019 FORMAT('!')
C C0020 FORMAT('!')
C
CALL PLOT(X9, P2P, 30, 1)
CALL PLOT(X9, P2K, 31, 3)
WRITE(6, 90009)
CALL PLOT(X9, P3P, 30, 1)
CALL PLOT(X9, P3K, 31, 3)
WRITE(6, 90010)
CALL PLOT(X9, P4P, 30, 1)
CALL PLOT(X9, P4K, 31, 3)
WRITE(6, 90011)
CALL PLOT(X9, P5P, 30, 1)
CALL PLOT(X9, P5K, 31, 3)
WRITE(6, 90012)
CALL PLOT(X9, XSER, 30, 1)
CALL PLOT(X9, XKER, 31, 3)
WRITE(6, 90013)
CALL PLOT(X9, P1PP, 30, 1)
CALL PLOT(X9, P1P1, 31, 3)
WRITE(6, 90014)
CALL PLOT(X9, P2PP, 30, 1)
CALL PLOT(X9, P2P2, 31, 3)
WRITE(6, 90015)
CALL PLOT(X9, P3PP, 30, 1)
CALL PLOT(X9, P3P3, 31, 3)
WRITE(6, 90016)
CALL PLOT(X9, P4PP, 30, 1)
CALL PLOT(X9, P4P4, 31, 3)
WRITE(6, 90017)
CALL PLOT(X9, P5PP, 30, 1)
CALL PLOT(X9, P5P5, 31, 3)
WRITE(6, 90018)
WRITE(6, 90019)
WRITE(6, 90009)
ERROR OF KALMAN
WRITE(6, *)XKER
WRITE(6, 90020)
ERROR AFTER SMOOTHING
WRITE(6, *)XSER
```
WRITE(6,90024)
90024 FORMAT('TRUE X-POSITION')
WRITE(6,*)TRUX
WRITE(6,90025)
90025 FORMAT('TRUE Y-POSITION')
WRITE(6,*)TRUY
WRITE(6,90026)
90026 FORMAT('TRUE Z-POSITION')
WRITE(6,*)TRUZ
WRITE(6,90021)
90021 FORMAT('ESTIMATES OF KALMAN')
WRITE(6,*)XP1K
WRITE(6,*)XP2K
WRITE(6,*)XP3K
WRITE(6,*)XP4K
WRITE(6,*)XP5K
WRITE(6,90022)
90022 FORMAT('ESTIMATES AFTER SMOOTHING')
WRITE(6,*)XP1S
WRITE(6,*)XP2S
WRITE(6,*)XP3S
WRITE(6,*)XP4S
WRITE(6,*)XP5S
WRITE(6,90023)
90023 FORMAT('DIAGONAL TERMS OF COVARIANCE')
WRITE(6,*)P1P1
WRITE(6,*)P2P2
WRITE(6,*)P3P3
WRITE(6,*)P4P4
WRITE(6,*)P5P5
WRITE(6,90024)
99950 FORMAT('TIME',7X,'TRUE-X',7X,'X-AFTER KALMAN',5X,'X-AFTER
* SMOOTHING',7,5X,'-------------------------
*------------------')
DO 99951 IP=1,JTIME
WRITE(6,99952)IP,TIMEX(IP),XP1K(IP),XP1S(IP)
99952 FORMAT('TIME',7X,'TRUE-Y',7X,'Y-AFTER KALMAN',5X,'Y-AFTER
* SMOOTHING',7,5X,'-------------------------
*------------------')
DO 99954 IP=1,JTIME
```
SUBROUTINE TRAJECT(K, DATR, ZI, TD)

SUBROUTINE COMPUTES TRUE TRAJECTORY OF TORPEDO

DATR(1)=TRUE X POSITION;  DATR(2)=TRUE Y POSITION
DATR(3)=TRUE X VELOCITY;  DATR(4)=TRUE Y VELOCITY
DATR(5)=TRUE X ACCELERATION; DATR(6)=TRUE Y ACCELERATION
DATR(7)=TRUE ANGULAR VELOCITY; DATR(8)=TRUE ANGULAR ACCELERATION
DATR(9)=TRUE TIME

DOUBLE PRECISION DATR, ZI
REAL*4 DATR(8), ZI(4), TD(3)
T=0.0
VEL=4860.*ZI(1)**2
RANGE=DSQR(DATR(1)**2+DATR(2)**2+ZI(1)**2+ZI(2)**2)
RANGE=DSQR((DATR(1)**2+ZI(1)**2)**0.5
I=1
ZI(1)=1./VEL*((DATR(1)**2)**0.5
I=2
ZI(1)=1./VEL*((DATR(1)**2)**0.5
I=3
ZI(1)=1./VEL*((DATR(1)**2)**0.5
I=4
ZI(1)=1./VEL*((DATR(1)**2)**0.5
DO 9 I=1, 3
TD(I)=DATR(I)
CONTINUE
IF((K.LE.DATR(17)).AND.(K.GT. DATR(16))) GO TO 11
DATR(7)=0.0
DATR(8)=0.0
DATR(14)=1.31
GO TO 12
DATR(14)=0.05
DATR(12)=DATR(12)+DATR(13)*DATR(14)
DATR(7)=DATR(16)*DCOS(DATR(13))
DATR(7)=DATR(15)*COS(DATR(12))
DATR(16)=DATR(16)*DSIN(DATR(13))
DATR(8)=DATR(15)*SIN(DATR(12))
DO 10 I=1, 5
10   DATR(I)=DATR(I)+DATR(I+3)*DATR(14)+(( DATR(14))**2)/2)*DATR(I+6)
CONTINUE
T=T+DATR(14)
IF(ABS(T-1.31).LE.0.0001) RETURN
GO TO 13
END
SUBROUTINE TRAJC3(DATR, ZI, TD, XB, YB, ZB)

C THIS SUBROUTINE IS USED DURING MULTIPLE ARRAY TRACKING
C
REAL*4 DATR(17), ZI(4), TD(3), XB(4), YB(4), ZB(4)
T=0.0
VEL=4860.
RANGE=SQR(DATR(1)*DATR(1)+DATR(2)*DATR(2)+DATR(3)*DATR(3))
DO 5 I=1,4
   ZI(I)=I./VEL*(((DATR(I)-XB(I))**2+((DATR(2)-YB(I)))**2)**0.5
   TD(I)=DATR(I)
5 CONTINUE
DO 9 I=1,3
   TD(I)=DATR(I)
9 CONTINUE
IF(K.LE.DATR(17)).AND.(K.GT.DATR(16))) GO TO 11
DATR(7)=0.0
DATR(8)=0.0
DATR(14)=1.31
GO TO 12
DATR(14)=.005
13 DATR(12)=DATR(12)+DATR(13)*DATR(14)
C DATR(7)=DATR(15)*COS(DATR(13))
C DATR(7)=DATR(15)*COS(DATR(12))
C DATR(8)=DATR(16)*SIN(DATR(13))
C DATR(8)=DATR(15)*SIN(DATR(12))
12 DO 10 I=1,5
   DATR(I)=DATR(I)+DATR(I+3)*DATR(14)+(((DATR(14))**2)/2)*DATR(I+6)
10 CONTINUE
T=T+DATR(14)
IF(ABS(T-.31).LE.0.0001) RETURN
GO TO 13
END
SUBROUTINE INIT (XKKM1, PKKM1)

C THIS ROUTINE IS TO INITIALIZE THE ARRAYS XKKM1 AND PKKM1.
C THESE VARIABLES ARE PART OF A COMMON BLOCK.

C
D O U B L E  P R E C I S I O N  X K K M 1 ,  P K K M 1
R E A L * 4   X K K M 1 (5), P K K M 1 (5, 5)
D O 2 0  J = 1 , 5
   D O 1 0  I = 1 , 5
      P K K M 1 ( I , J ) = 0 . 0
1 0  C O N T I N U E
2 0  C O N T I N U E
   X K K M 1 (1) = 7 4 7 . 5
   X K K M 1 (2) = - 4 0
   X K K M 1 (3) = 1 2 7 . 5
   X K K M 1 (4) = 0.0
   X K K M 1 (5) = 0.
   D O 3 0  I = 1 , 5
       P K K M 1 ( I , 1 ) = 1 0 0 0 .
3 0  C O N T I N U E
   R E T U R N
E N D
SUBROUTINE CZHAT(I,ZHAT)

C THIS SUBROUTINE COMPUTES ESTIMATES OF TRANSIT TIME MEASUREMENTS
FOR SINGLE ARRAY TRACKING.

C DOUBLE PRECISION XKKM1,ZHAT
COMMON/THE1/XKKM1(5)
REAL*4 ZHAT
VEL=.860
IF(I.EQ.1)ZHAT=1./VEL*((XKKM1(1)+15.)*2)+(XKKM1(3)+15.)*2)
1+((XKKM1(5)+15.)*2)**2
IF(I.EQ.2)ZHAT=1./VEL*((XKKM1(1)-15.)*2)+(XKKM1(3)+15.)*2)
1+((XKKM1(5)+15.)*2)**2
IF(I.EQ.1)ZHAT=1./VEL*((XKKM1(1)+15.)*2)+(XKKM1(3)-15.)*2)
1+((XKKM1(5)+15.)*2)**2
IF(I.EQ.1)ZHAT=1./VEL*((XKKM1(1)+15.)*2)+(XKKM1(3)+15.)*2)
1+((XKKM1(5)-15.)*2)**2)**2
RETURN
END
SUBROUTINE C2HAT3 (I, ZHAT, XB, YB, ZB)

THIS SUBROUTINE COMPUTES ESTIMATES OF TRANSIT TIME MEASUREMENTS FOR MULTIPLE ARRAY TRACKING

DOUBLE PRECISION XKKM1, ZHAT
REAL*4 XB(4), YB(4), ZB(4)
REAL*4 PKK(5,5), PKK(5,5), XKK(5)
COMMON/THE1/XKKM1(5)
VEL=4860.
XO=XB(1)
YO=YB(1)
ZO=ZB(1)
ZHAT=(1./VEL)*((XKKM1(1)-XO)**2)+((XKKM1(3)-Y0)**2)+((XKKM1(5)
1-ZO)**2)**0.5
RETURN
END
SUBROUTINE CHKROW3 (I,HROW,XB,YB,ZB)

DOUBLE PRECISION XKMK1,HROW,DENOM
REAL*4 HROW(5),XB(4),YB(4),ZB(4)
COMMON/THE1/XKMK1(5)

VEL=4860.
X0=XB(I)
Y0=YB(I)
Z0=ZB(I)

DENOM= ((XKMK1(1)-X0)**2)+((XKMK1(3)-Y0)**2)+((XKMK1(5)-Z0)**2)

HROW(1)=(1./VEL)*((XKMK1(1)-X0)/DENOM)
HROW(3)=(1./VEL)*((XKMK1(3)-Y0)/DENOM)
HROW(5)=(1./VEL)*((XKMK1(5)-Z0)/DENOM)

DO 27 J=2,4,2
27 HROW(J)=0.
RETURN
END
SUBROUTINE CHROW(I,HROW)

THIs SUBROUTINE USED FOR SINGLE ARRAY TRACKING

DOUBLE PRECISION XKKM1,DENOM1, DENOM2, DENOM3, DENOM4, DENOM, HROW,

H1, H3, H5
COMMON/THE1/XKKM1(5)
REAL*4 HROW(5)
VEL=4860.

DENOM1=((XKKM1(1)+15.)*2.+(XKKM1(3)+15.)*2.+(XKKM1(5)+15.)*2.+(XKKM1(7)+15.)*2.)/(12.)**0.5
DENOM2=((XKKM1(1)-15.)*2.+(XKKM1(3)+15.)*2.+(XKKM1(5)+15.)*2.)/(12.)**0.5
DENOM3=((XKKM1(1)+15.)*2.+(XKKM1(3)-15.)*2.+(XKKM1(5)+15.)*2.)/(12.)**0.5
DENOM4=((XKKM1(1)+15.)*2.+(XKKM1(3)+15.)*2.+(XKKM1(5)-15.)*2.)/(12.)**0.5

A1=1.
A2=1.
A3=1.
DENOM=DENOM1
IF(I.EQ.2)DENOM=DENOM2
IF(I.EQ.3)DENOM=DENOM3
IF(I.EQ.4)DENOM=DENOM4
IF(I.EQ.2)A1=-1.
H1=(1./VEL)*((XKKM1(1)+A1*15.)/DENOM)
IF(I.EQ.3)A2=-1.
H3=(1./VEL)*((XKKM1(3)+A2*15.)/DENOM)
IF(I.EQ.4)A3=-1.
H5=(1./VEL)*((XKKM1(5)+A3*15.)/DENOM)
HROW(1)=H1
HROW(3)=H3
HROW(5)=H5
DO 27 J=2,4,2
HROW(J)=0.
27 RETURN
END
SUBROUTINE QFIND(K, SIGAAC, SIGDIV, SIGCC, A, Q)

C THIS SUBROUTINE COMPUTES THE ADAPTIVE Q MATRIX

C

DOUBLE PRECISION XHKKM1, PKKML1, PKK, XHKK, Q
REAL*4 Q(I, J)
COMMON/XK/MM1(5), THE2/PKKM1(5, 5), PKK(5, 5), XKK(5)
IF(K.EQ.1) GO TO 15
DO 10 I=1, 5
  DO 10 J=1, 5
  10 Q(I, J)=0.0

SIGAAC=SIGAAC**2
Q(5, 5)=(SIGDIV*A)**2
SIGCC=SIGCC**2
G1 = A**2/2.0
G2 = G1**2
G3 = A*G1
G4 = A**2
A1 = XKK (2)**2 + XKK (4)**2
A3 = XKK (2) / SQRT(A1)
B = XKK (2)
C = XKK (4) / SQRT(A1)
D = XKK (2)

C

D1 = D**2
E1 = (A)**2*SIGAAC + (B**2)*SIGCC
E2 = (C)**2*SIGAAC + (D**2)*SIGCC
E3 = ((B**2)/A**2)*SIGAC
E4 = ((C**2)/A**2)*SIGAC
Q(1, 1) = G2*E1
Q(1, 2) = G3*E1
Q(1, 3) = G2*E12
Q(1, 4) = G3*E12
Q(2, 2) = A2*E12
Q(2, 3) = G3*E12
Q(2, 4) = A2*E12
Q(3, 3) = G2*E2
Q(3, 4) = G3*E2
Q(4, 4) = A2*E2

DO 27 I=1, 4
  DO 27 J=1, 4
  27 Q(I, J)=Q(J, I)
RETURN
END
SUBROUTINE SMOOTH( SS, SS1, PHI, P5, XP6, KK, N )

! THIS SUBROUTINE COMPUTES SMOOTHED STATES AND COVARIANCES

COMMON / THE3 / XP(5, 210)/ THE4 / P1(5, 5, 5)/ THE5 / SI

DOUBLE PRECISION XP, SS, P1, SS1, SI, KK, P5, XP6
REAL*4 XNMN(5), PNMN(5, 5), SS(5, 210), SS1(210, 5, 5),
PHI(5, 5), PHIT(5, 5), SS3R(5, 5), AK(5, 5), AKT(5, 5),
PKKS(5, 5), TEMP1(5, 5), TEMP2(5, 5), TEMP3(5),
TEMP4(5, 5), TEMP5(5, 5), TEMP6(5, 5), CH(5, 5)

L = KK - 1
DO 20 K = 1, L
K1 = L - K + 1
DO 21 I = 1, N
XPI(I) = XP6(I, K1)
SS2(I) = SS(I, K1)
21 CONTINUE
DO 22 I = 1, N
DO 23 J = 1, N
P2(I, J) = P5(K1, I, J)
SS3(I, J) = SS1(K1, I, J)
22 CONTINUE
DO 24 I = 1, N
DO 25 J = 1, N
P3(I, J) = P5(K1, I, J)
25 CONTINUE
DO 26 I = 1, N
DO 27 J = 1, N
XNMN(I, J) = XP6(I, K1 + J)
27 CONTINUE

CALL TRANS(PHI, N, N, PHIT)
CALL RECIP(SS3, N, SS3R)
CALL PROD(SS3, SS3R, N, N, N, CH)

CALL A(K) = P(K/K) * TRANSPOSE(P(K) * INVERSE(P(K+1/K)))

CALL PROD(PHIT, SS3R, N, N, N, TEMP1)
CALL PROD(P2, TEMP1, N, N, AK)
DO 28 I = 1, N
DO 29 J = 1, N
XNMN(I, J) = XP6(I, K1 + J)
29 CONTINUE

CALL SUB(XNMN, SS2, N, 1, TEMP2)
CALL PROD(AK, TEMP2, N, 1, TEMP3)
CALL ADD(XP1, TEMP3, N, 1, XKKS)
C
DO 80 I=1,N
XP(I,K1)=XKKS(I)
C
DO 31 I=1,N
DO 32 J=1,N
32 PNNM1(I,J)=P1(K1+1,I,J)
31 CONTINUE
C
FIND SMOOTHEC COVARIANCES
C
CALL SUB(PNNM1,SS3,N,N,TEMP4)
CALL TRANS(AK,N,N,AKT)
CALL PROD(TEMP4,AKT,N,N,TEMP5)
CALL PROD(AK,TEMP5,N,N,TEMP6)
CALL ADD(P2,TEMP6,N,N,PKKS)
C
DO 38 I=1,N
DO 39 J=1,N
39 P1(K1,I,J)=PKKS(I,J)
38 CONTINUE
C
20 CONTINUE
IS=1.
RETURN
END
SUBROUTINE RECIP(A,N,C)
C
DOUBLE PRECISION A,D,C
DIMENSION A(5,5),D(5,10),C(5,5)
C
DO 10 K=1,N
DO 11 J=1,N
11 D(K,J)=A(K,J)
10 CONTINUE
DO 12 K=1,N
I=K+N
DO 14 J=6,10
IF(I.NE.J)GO TO 13
D(K,J)=1.
GO TO 14
13 D(K,J)=0.
14 CONTINUE
12 CONTINUE
DO 15 K=1,N
M=K+1
DO 16 J=M,10
D(K,J)=D(K,J)/D(K,K)
D(K,K)=1.
16 CONTINUE
DO 17 L=1,N
IF(L.EQ.K)GO TO 17
DO 19 I=1,10
IF(I.EQ.K)GO TO 19
D(L,I)=D(L,I)-D(L,K)*D(K,I)
19 CONTINUE
D(L,K)=0.
17 CONTINUE
15 CONTINUE
DO 20 K=1,N
DO 21 J=1,N
I=J+N
21 C(K,J)=D(K,I)
20 CONTINUE
RETURN
END
SUBROUTINE PROD(A,B,N,M,L,C)

C
C DOUBLE PRECISION A,B,C
REAL*4  A(N,M),B(M,L),C(N,L)
DO 1 I=1,N
DO 1 J=1:L
1 C(I,J)=0.
DO 2 I=1:N
DO 2 J=1:L
DO 2 K=1:M
2 C(I,J)=C(I,J)+A(I,K)*B(K,J)
RETURN
END
SUBROUTINE HMULT(A,B,N,M,C)

DOUBLE PRECISION A,B,C
REAL*4 A(N,M),B(M),C(N)
DO 3 I=1,N
  C(I)=0.
  DO 4 J=1,M
    C(I)=C(I)+A(I,J)*B(J)
  CONTINUE
  CONTINUE
RETURN
END
SUBROUTINE VMULT(A, B, N, C)

DOUBLE PRECISION A, B
REAL*4 A(N), B(N)
C = 0.
DO 6 I = 1, N
  C = C + A(I) * B(I)
6 RETURN
END
SUBROUTINE TRANS(A,N,M,B)

C DOUBLE PRECISION A,B
REAL*4 A(N,M),B(M,N)
DO 13 I=1,N
DO 13 J=1,M
13 B(J,I)=A(I,J)
RETURN
END
SUBROUTINE ADD (A, B, N, M, C)

DOUBLE PRECISION A, B, C
REAL*4 A(N,M), B(N,M), C(N,M)
DO 15 I=1,N
  DO 15 J=1,M
  C(I,J)=A(I,J)+B(I,J)
RETURN
END
SUBROUTINE SUB(A,B,N,M,C)

DOUBLE PRECISION A,B,C

DIMENSION A(N,N),B(N,N),C(N,N)

DO 10 I=1,N
    DO J=1,N
        C(I,J) = A(I,J) - B(I,J)
    10 CONTINUE

RETURN
END
LIST OF REFERENCES


<table>
<thead>
<tr>
<th>No.</th>
<th>Copies</th>
<th>Distribution List</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>2</td>
<td>Defense Technical Information Center&lt;br&gt;Cameron Station&lt;br&gt;Alexandria, Virginia 22314</td>
</tr>
<tr>
<td>2.</td>
<td>2</td>
<td>Library, Code 0142&lt;br&gt;Naval Postgraduate School&lt;br&gt;Monterey, California 93940</td>
</tr>
<tr>
<td>3.</td>
<td>1</td>
<td>Department Chairman, Code 62&lt;br&gt;Department of Electrical Engineering&lt;br&gt;Naval Postgraduate School&lt;br&gt;Monterey, California 93940</td>
</tr>
<tr>
<td>4.</td>
<td>2</td>
<td>Professor H. A. Titus, Code 62Ts&lt;br&gt;Department of Electrical Engineering&lt;br&gt;Naval Postgraduate School&lt;br&gt;Monterey, California 93940</td>
</tr>
<tr>
<td>5.</td>
<td>1</td>
<td>Professor Alex Gerba, Code 62Gz&lt;br&gt;Department of Electrical Engineering&lt;br&gt;Naval Postgraduate School&lt;br&gt;Monterey, California 93940</td>
</tr>
<tr>
<td>6.</td>
<td>5</td>
<td>Deniz Kuvvetleri Komutanligi&lt;br&gt;Elektim Dairesi&lt;br&gt;Ankara/Turkey</td>
</tr>
<tr>
<td>7.</td>
<td>1</td>
<td>Istanbul Teknik Universitesi&lt;br&gt;Elektrik Fakultesi&lt;br&gt;Gumussuyu, Istanbul/Turkey</td>
</tr>
<tr>
<td>8.</td>
<td>1</td>
<td>Bogazici Universitesi&lt;br&gt;P.K. 2 Bebek, Istanbul/Turkey</td>
</tr>
<tr>
<td>9.</td>
<td>1</td>
<td>Orta Dogu Teknik Universitesi&lt;br&gt;Ankara/Turkey</td>
</tr>
<tr>
<td>10.</td>
<td>2</td>
<td>Mustafa Isik&lt;br&gt;Dogancilar, Ramazanoglu sok&lt;br&gt;No: 10/5 Uskudar, Istanbul/Turkey</td>
</tr>
</tbody>
</table>