AD-A068 386 SRI INTERNATIONAL MENLO PARK CA NAVAL WARFARE RESEAR--ETC F/G 17/1
A MULTIUNIT RELATIVE LOCALIZATION AND TRACKING ALGORITHM FOR OT--ETC(U)
MAR 79 J R OLMSTEAD, L C GOHEEN N00014-78-C-0398
UNCLASSIFIED NWRC-TR-22 NL

| OF |
AD A068386

END
DATE FILMED
6-79
DDC

Technical Report NWRC-TR-22 *March 1979*

A MULTIUNIT RELATIVE LOCALIZATION AND TRACKING ALGORITHM FOR OTH TARGETING

By: JEFFREY R. OLMSTEAD LOLA C. GOHEEN

Prepared for:

NAVAL ANALYSIS PROGRAMS (Code 431)
OFFICE OF NAVAL RESEARCH
DEPARTMENT OF THE NAVY
ARLINGTON, VIRGINIA 22217

CONTRACT N00014-78-C-0398

Task NR 274-297

333 Ravenswood Avenue
Menlo Park, California 94025 U.S.A.
(415) 326-6200
Cable: SRI INTL MNP
TWX: 910-373-1246

SRI International

NAVAL WARFARE RESEARCH CENTER

Technical Report NWRC-TR-22 *March 1979*

A MULTIUNIT RELATIVE LOCALIZATION AND TRACKING ALGORITHM FOR OTH TARGETING

By: JEFFREY R. OLMSTEAD LOLA C. GOHEEN

Prepared for:

NAVAL ANALYSIS PROGRAMS (Code 431)
OFFICE OF NAVAL RESEARCH
DEPARTMENT OF THE NAVY
ARLINGTON, VIRGINIA 22217

CONTRACT N00014-78-C-0398
Task NR 274-297

SRI Project 7467

Approved by:

AL BIEN, *Director*
Naval Warfare Research Center

DAVID D. ELLIOTT, *Executive Director*
Systems Research and Analysis Division

SRI International

333 Ravenswood Avenue • Menlo Park, California 94025 • U.S.A.
(415) 326-6200 • Cable: SRI INTL MNP • TWX: 910-373-1246

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: 14 NWRC-TR-22
2. GOVT ACCESSION NO.:
3. RECIPIENT'S CATALOG NUMBER:

4. TITLE (and Subtitle): A MULTIUNIT RELATIVE LOCALIZATION AND TRACKING ALGORITHM FOR OTH TARGETING
5. TYPE OF REPORT & PERIOD COVERED: Final Report, May 1978 - Apr 1979

6. PERFORMING ORG. REPORT NUMBER: SRI Project 7467

7. AUTHOR(s): Jeffrey R. Olmstead, Lola C. Goheen

8. CONTRACT OR GRANT NUMBER(s): Contract N00014-78-C-0398

9. PERFORMING ORGANIZATION NAME AND ADDRESS:
SRI International
333 Ravenswood Avenue
Menlo Park, California 94025

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:
65152N R0145
NR 274-297

11. CONTROLLING OFFICE NAME AND ADDRESS:
Office of Naval Research
Naval Analysis Programs (Code 431)
Arlington, VA 22217

12. REPORT DATE: Mar 1979
13. NO. OF PAGES: 80

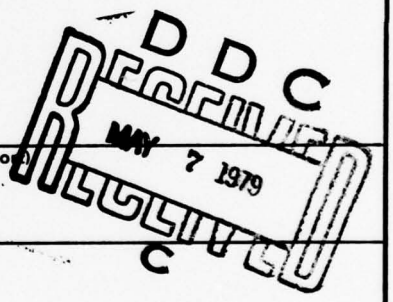
15. SECURITY CLASS. (of this report): UNCLASSIFIED

14. MONITORING AGENCY NAME & ADDRESS (if diff. from Controlling Office): 12 72p

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this report)

Approved for Public Release; Distribution Unlimited.



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

18. SUPPLEMENTARY NOTES

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

Multiunit tracking	Maneuvering target
Relative localization	Kalman filter
OTH targeting	Adaptive filter
Bearing-only	Nonlinear least-squares method
	Gauss-Newton method

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

A Kalman filter tracking algorithm, called MURLOC, is described and favorably evaluated against a batch-processed least-squares algorithm. MURLOC is applicable to locating and tracking multiple units in OTH targeting scenarios. The algorithm uses a multiunit state vector, transforms measurements before filtering, determines a pseudo range to use with bearing-only measurements, and adapts the model noise covariance to the measurement residuals so that a maneuvering target can be tracked. MURLOC can filter x-and-y, range-and-bearing, bearing-only, and course-and-speed measurements.

DD FORM 1 JAN 73 1473
EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

410 521

CONTENTS

APPENDICES

ILLUSTRATIONS

TABLES

I INTRODUCTION

This study describes and evaluates a tracking algorithm that can follow a maneuvering target by filtering time-sequenced measurements from multiple observers. The algorithm can be used when several units are passing targeting data to a single unit on which the tracking solution is computed. The data may consist of (1) x-and-y (latitude-and-longitude), (2) range-and-bearing, (3) bearing-only, or (4) course-and-speed measurements, along with estimates of the measurement errors.

The tracking algorithm is applicable to the OTH (over-the-horizon) targeting situation shown in Figure 1. Here two ships cross-fix the target ship by using towed-array bearing-only measurements, and locate each other by range-and-bearing measurements using an intermediary helicopter as a radar target. The helicopter also relays the targeting information from the second ship to the first ship. The tracking algorithm was specifically designed for this kind of problem, in which the locations of several units (both friendly and target units) are unknown but can be deduced by time-processing position and velocity measurements, especially bearing-only measurements from an ESM receiver or a passive sonar. The tracking algorithm is called MURLOC, an acronym for Multiunit Relative Localization.

MURLOC has several unique characteristics that make it a versatile tracker in OTH targeting scenarios:

- All units are correlated by employing a single state vector that contains the positions and velocities of all the units, instead of multiple state vectors, one for each unit. The reason for using a super-state vector is to capture the statistical correlations between units. Thus a range-and-bearing measurement on an own-force unit automatically adjusts the estimates of relative position on all units including the target.

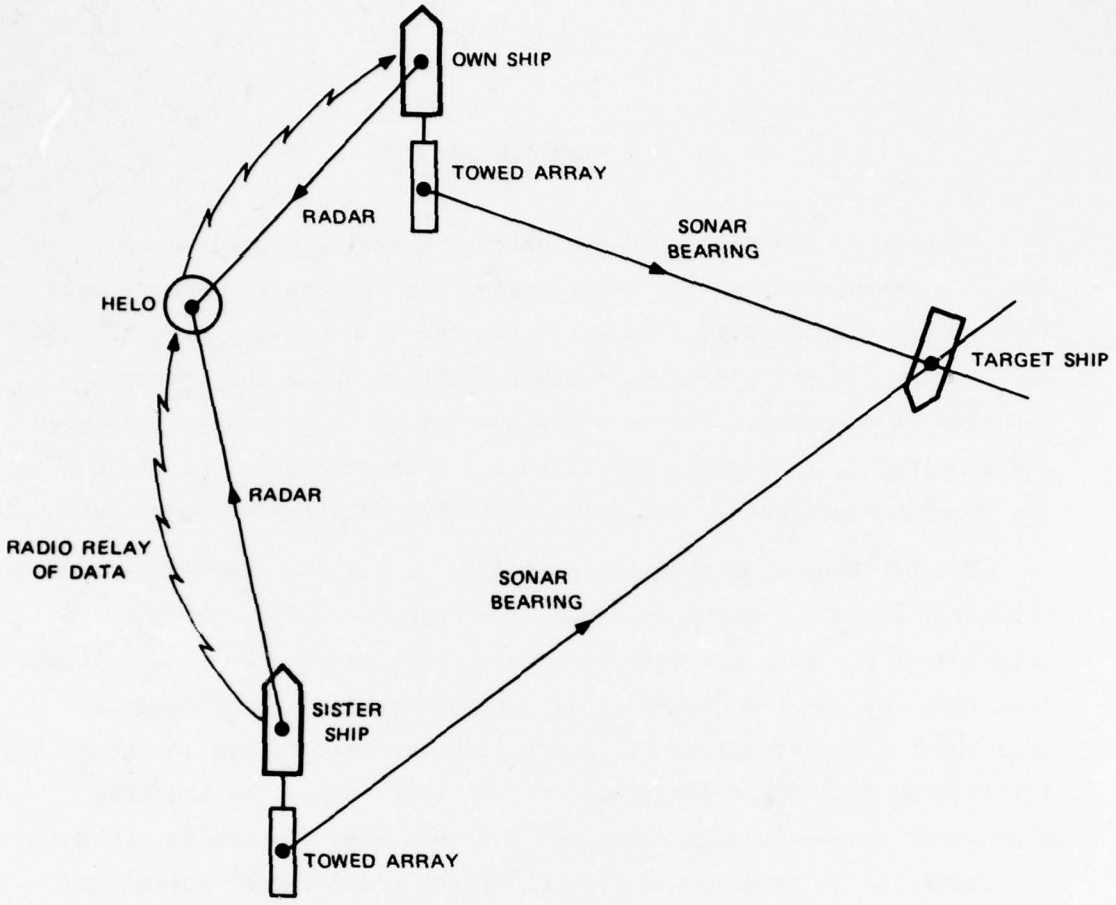


FIGURE 1 OTH APPLICATION OF MURLOC

- Measurements are transformed to Cartesian coordinates before they are filtered; thus MURLOC uses a standard Kalman filter without having to compute partial derivatives at predicted positions, as done in an "extended" Kalman filter. This "transformed-measurement" approach avoids problems that are inherent in extended Kalman filters.

- Bearing-only measurements are treated as range-and-bearing measurements by estimating a pseudo range using a special "ellipse tangent" algorithm. The range-and-bearing values are then transformed to Cartesian coordinates and filtered. The "transformed-measurement" approach to filtering and the "ellipse tangent" algorithm for bearing-only measurements seem to work very well in the multiunit-type scenario.

- A model noise covariance matrix is adapted to the measurement residuals so that unknown maneuvers can be tracked. MURLOC's maneuvering-target adaptive algorithm is simple and seems to be effective.

The impetus for creating MURLOC was the ship-launched Harpoon targeting problem of (1) processing data from radar, sonar, and ESM sensors that are located on the Harpoon ship and other ships or aircraft, (2) tracking several targets and own-force units at the same time, and (3) producing decision aids to help determine when to launch a missile. The main computations needed are the fire-control solution, the estimate of errors associated with the solution, the probability that the Harpoon missile seeker will illuminate the target, and the probability that the target is within counterattack range.

These shipboard computational requirements can be marginally satisfied with a multiplicity of plotting techniques, tables, graphs, slide rules, and hand-held calculators. Many different techniques must be developed for the many different situations that can arise. These techniques will be complicated if they are designed for situations in which the data are distributed in time and come from sources other than own-ship sensors. The calculation of targeting errors and other decision aids will be quite difficult without automated methods.

An alternative to the multiplicity of techniques is to use a tabletop programmable calculator (such as the Wang 2200 or HP 9830) to compute the required decision-aid and fire-control parameters. The idea is that all the various targeting methods can be integrated into one system by developing programmable-calculator software that can incorporate each new measurement as it arrives in the command center. The data may be from own-ship radar, sonar, or ESM; another ship's sensors; or the LAMPS helicopter's sensors. In addition to originating from different sources, the measurements may be staggered in time, may come in late, or may be out of time order. The targeting software should be capable of calculating tracks on several targets at the same time; and since the position of friendly units is important to targeting solutions, the software should also calculate tracks on friendly units.

MURLOC was created as a prototype algorithm for the tracking portion of such a targeting system. So far, only the multiunit, bearing-only, and maneuvering-target tracking capabilities of MURLOC have been investigated. A major unanswered question is whether or not the

processing of MURLOC's super-state vector requires so much computation time on a tabletop programmable calculator that the tracking solution cannot keep up with the data. The major factors in this question are the number of units in the tracking problem and the data rate. Another question is how to filter data that arrive out of time sequence. Because MURLOC is a recursive algorithm, solutions would probably have to be saved periodically so that the tracker could be initialized at the closest solution to the old (but newly arrived) measurement, and then recycled through the stored measurements following that old measurement.

Whether or not MURLOC is used in a shipboard targeting system, it can stand alone as a method for analyzing multiunit tracking scenarios. MURLOC can be used to process a time sequence of simulated random measurements and predict the target track. Many replications of the random-measurement sequences can be processed and the average estimated track compared to the true track. This was, in fact, the method used to investigate MURLOC's tracking capability.

A time sequence of average position and velocity errors was computed using MURLOC on 50 replications of a maneuvering-target scenario. MURLOC was compared to another tracking algorithm that used a batch-processed least-squares method. The comparison analysis showed that MURLOC tracked the maneuver quite well, especially in estimating the target's relative position. MURLOC does, however, underestimate its own errors.

The least-squares algorithm is also a product of the study. Research on how to perform weighted nonlinear least-squares computations and how to select the best set of measurements resulted in a tracking methodology that, with further research, might be useful in targeting software. In any case, the use of the least-squares algorithm provided a good benchmark for judging MURLOC's tracking capability.

The MURLOC computer program is coded in FORTRAN for SRI's CDC 6400 computer. The source deck is approximately 500 cards, and with array dimensions that can accommodate 10 units, the program takes about

15,100 words of memory. The running time depends on a number of factors; for example, one replication of a scenario with 3 units, 10 time steps, and 5 measurements each time step, required about 16 seconds of computer time. The MURLOC computer program source code is listed in Appendix A.

II MURLOC TRACKING ALGORITHM

This chapter describes the major features of the MURLOC algorithm and gives an example of how multiunit measurements can be processed with MURLOC.

A. MURLOC Description

MURLOC can accept time-sequenced position or velocity data on multiple targets from multiple observers. MURLOC processes the input data at each step in time and estimates the position and velocity vectors, the range and bearing between units, and error ellipses. MURLOC uses a Kalman filter with one multiunit state vector that causes full correlation between units. A unique adaptive covariance scheme reduces the problem of the linear motion assumption. Table 1 defines the various symbols that are used in this chapter, and Table 2 summarizes the major equations that are the essence of the MURLOC algorithm.

1. State Vector

MURLOC uses one multiunit state vector. For example, if there are five units in the problem, the state vector will have 20 components; the first four components are the position and velocity ($x\ y\ \dot{x}\ \dot{y}$) of the first unit, the next four components are the position and velocity of the second unit, and so on. The covariance matrix of this example is a 20-by-20 matrix; thus, the state values can become correlated, not only by position and velocity correlation on a single unit, but also by correlation between units.

MURLOC is designed so that units report data on other units; no distinction is made between friendly and enemy units other than knowing which is which by unit number. Thus, data from a radar fix on a companion ship are processed in the same way as data from an ESM fix on an enemy ship. This design was chosen because of the real-world

Table 1

FILTER VARIABLES

Variable	Definition	Dimension*
$\bar{x}_o$	Estimate of the state vector, given the measurement at t_o (the filtered state)	4n-by-1
P_o	Covariance matrix of $\bar{x}_o$	4n-by-4n
y	Measurement vector at time t_1	2 -by-1
R	Covariance matrix of y	2 -by-2
$\bar{x}$	Estimate of the state vector at t_1 before the measurement at t_1 is filtered (the predicted state)	4n-by-1
P	Covariance matrix of $\bar{x}$	4n-by-4n
F	State transition matrix from time t_o to t_1	4n-by-4n
Q	Model noise covariance matrix	4n-by-4n
M	Measurement matrix (state-to-measurement transformation)	2 -by-4n
r	Predicted measurement residual error (also called "innovation")	2 -by-1
S	Covariance matrix of r	2 -by-2
K	Filter gain matrix	4n-by-2
$\bar{x}_1$	Estimate of the state vector, given the measurement at t_1 (the filtered state)	4n-by-1
P_1	Covariance matrix of x_1	4n-by-4n

* n = Number of units being tracked.

Table 2

MURLOC ADAPTIVE ALGORITHM

Filtered State at Time t_o

$$\bar{x}_o$$

$$P_o$$

Measurement at Time t_1

$$y$$

$$R$$

Predicted State at Time t_1

$$\bar{x} = F \bar{x}_o$$

$$r = y - M \bar{x}$$

$$S = R + M F P_o F^T M^T$$

$$\beta = 1 - \exp(-\tfrac{1}{2} r^T S^{-1} r)$$

$$\tau = t_1 - t_o$$

$$q = (r_1 \quad r_2 \quad r_1/\tau \quad r_2/\tau)^T$$

$$Q = \beta q q^T$$

$$P = F P_o F^T + Q$$

Filtered State at Time t_1

$$K = P M^T (R + M P M^T)^{-1}$$

$$L = I - K M$$

$$\bar{x}_1 = \bar{x} + K r$$

$$P_1 = L P L^T + K R K^T$$

problem of tracking friendly units, in addition to the more obvious problem of tracking enemy units. Because the position-velocity state of friendly units is part of the problem, the friendly units' state vectors must also be estimated. The only way this can be done without restrictions and ad hoc assumptions is to incorporate all state vectors into one large state vector. Although this approach requires considerable computation to process one measurement, it is beneficial because it uses data properly. For example, the range-and-bearing data from Unit 3 on Unit 5 will, in general, change the state estimates of <u>all</u> the units because of past interaction among them.

The state vector at time t_o is denoted x_o. The estimate of x_o is denoted $\bar{x}_o$, and the covariance matrix of the error in $\bar{x}_o$ is denoted P_o. The state at time $t = t_1$ evolves from the state at time t_o by:

$$x = F x_o + w$$

where F is the transition matrix from t_o to t_1, and w is a random vector that is normally distributed with zero mean and covariance Q. As discussed later, the Q-matrix is defined as a function of the predicted measurement residuals and is the mechanism for adapting the filter for maneuvering targets. The transition matrix, F, is built up from submatrices, f:

$$f = \begin{bmatrix} f & 0 & - \\ 0 & f & - \\ - & - & - \end{bmatrix}$$

where the submatrix f is given by:

$$f = \begin{bmatrix} 1 & 0 & \tau & 0 \\ 0 & 1 & 0 & \tau \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and τ is the time step: $\tau = t_1 - t_o$.

2. Measurements

Four kinds of measurements can be used in MURLOC: (1) x-and-y position, (2) range-and-bearing, (3) bearing-only, and (4) course-and-speed. Standard deviations of errors in the measurements are also input parameters. Measurements are processed each time step, time steps may be variable, and multiple measurements at any one time may occur. Measurement errors are assumed to be uncorrelated from one measurement to another. Also, range errors, bearing errors, course errors, and speed errors are assumed to be uncorrelated with each other. However, the x-and-y position errors are assumed to be correlated and are given in terms of error ellipse parameters--the standard deviations along the two principal axes, and the angle from North to the major axis. Latitude-and-longitude measurements can be processed by first transforming to a Cartesian coordinate system a few hundred miles in size. Latitude-and-longitude measurements are the assumed origin of the x-and-y position measurements.

Before being filtered, the measurements are transformed to Cartesian coordinates and the coordinate errors are approximated by linear functions of the measurement errors. Usually tracking algorithms use an "extended" Kalman filter that linearizes the measurement equations around the predicted state by calculating first-order partial derivatives. This procedure is acceptable if the true state values are inside the linear region; however, if the predicted state is in error by a large amount, the tracker will behave poorly. Instead, MURLOC linearizes around the data point before filtering, and thus avoids the problem of calculating partial derivatives at the wrong place.

Range-and-bearing, bearing-only, and course-and-speed measurements are transformed; x-and-y measurements do not need to be transformed. For example, range-and-bearing (r,θ) are transformed into a measurement vector, y, by:

$$y = \begin{bmatrix} r \sin \theta \\ r \cos \theta \end{bmatrix}.$$

The measurement covariance, R, is calculated by an approximation that assumes that the standard deviations of the errors in range and bearing (σ_r,σ_θ) are small:

$$R = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} r^2\sigma_\theta^2 & 0 \\ 0 & \sigma_r^2 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}.$$

Bearing-only measurements require an estimate of range before the transformation can be performed. The range is estimated from the predicted positions and covariance of the observer and target units. The estimated range is calculated by expanding (or contracting) the predicted relative error ellipse until the ellipse just touches the measured bearing line; the tangent point then defines the estimated range, as shown in Figure 2. The "predicted relative error ellipse" is the error ellipse that is relative to the observer; it is computed from the predicted covariance elements of the observer and target (the equations are in Subroutine REPORT, which is in Appendix A). The standard deviation of range error is assumed to equal the estimated range. Thus, as shown in Figure 2, the bearing-only measurement is transformed to a long, thin ellipse that lies along the measured bearing

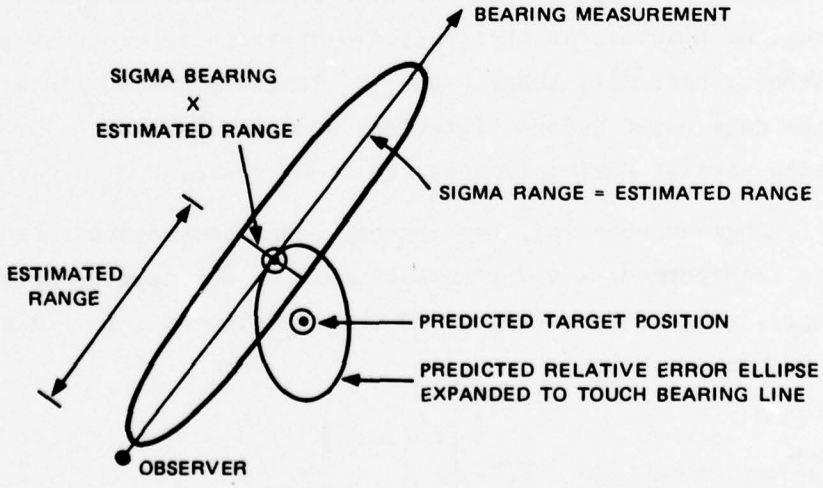


FIGURE 2 LINEARIZATION OF BEARING-ONLY MEASUREMENT

and is centered on the estimated range. The measurement vector, y, and the covariance matrix, R, are then computed as though the bearing-only measurement were a range-and-bearing measurement.

Even though a psuedo-range is used as data, and the assumption of small range error is violated, the above method of tracking with bearing-only data seemed to work quite well. Another method for estimating range was tried but it introduced large range errors after a target maneuver. The other method estimated the range by the intersection of a line that was perpendicular to the bearing line, and contained the predicted target position. The other method was computationally simple, but the "ellipse tangent" algorithm shown in Figure 2 proved to be far superior to the "perpendicular intersection" algorithm.

Course-and-speed (γ,s) measurements are transformed in the same way as range-and-bearing measurements, except that the measurement vector consists of velocity components:

$$y = \begin{bmatrix} s \sin \gamma \\ s \cos \gamma \end{bmatrix} .$$

The covariance, R, is also similar to the range-and-bearing case:

$$R = \begin{bmatrix} \cos \gamma & \sin \gamma \\ -\sin \gamma & \cos \gamma \end{bmatrix} \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix} \begin{bmatrix} \cos \gamma & -\sin \gamma \\ \sin \gamma & \cos \gamma \end{bmatrix}$$

where the input parameters σ_1 and σ_2 are the principal-axis components of the velocity error elipse: σ_1 is perpendicular to the velocity vector, and σ_2 is parallel to the velocity vector. This method of input is used instead of course and speed sigmas so that zero-velocity error distributions can be input easily. Course and speed measurements are thought of as a polar-coordinate representation of the velocity vector; the error along the vector is assumed to be uncorrelated with the error

across the vector. When the course and speed standard deviations (σ_γ,σ_s) are small, the approximation:

$$\sigma_1 = s\,\sigma_\gamma$$

$$\sigma_2 = \sigma_s$$

can be used to estimate the velocity error ellipse.

By linearizing the input data, the measurement vector, y, becomes a random vector that is a linear function of the true state values, X:

$$y = M\,X + v$$

where M is the measurement matrix, and v is a random vector that is normally distributed with zero mean and covariance, R.

The value of the measurement matrix, M, depends on the kind of measurement and the units involved. M is represented as a matrix of submatrices:

$$M = (M_1 \;\; M_2 \;\; .. \;\; M_n)$$

where k = 1,2,..,n is the index of a unit being tracked. For x-and-y position measurements, M is determined by:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$M_k = \begin{cases} A & \text{for k = index of the reporting unit} \\ 0 & \text{otherwise.} \end{cases}$$

For range-and-bearing or bearing-only measurements, M is determined by:

$$M_k = \begin{cases} A & \text{for k = index of the target unit} \\ -A & \text{for k = index of the observing unit} \\ 0 & \text{otherwise.} \end{cases}$$

For course-and-speed measurements, M is determined by:

$$B = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M_k = \begin{cases} B & \text{for } k = \text{index of the reporting unit} \\ 0 & \text{otherwise.} \end{cases}$$

3. Predicted State

MURLOC assumes that the units move with constant course and speed. The estimated state at time $t = t_1$ is predicted from the estimated state at t_o:

$$\bar{x} = F \bar{x}_o$$

The predicted covariance is not, however, the usual $F P_o F^T$ matrix with some constant model noise, Q, thrown in. Instead, the predicted covariance is required to be consistent with the measurement by making Q a function of the measurement residuals. The residual vector, r, is supposed to be normally distributed with zero mean and covariance matrix, S:

$$r = y - M \bar{x}$$

$$S = R + M F P_o F^T M^T.$$

However, when a unit changes course or speed, the residual vector can be many standard deviations from zero and the statistics of r are not what they are supposed to be because the assumptions of the model are violated. A procedure that is philosophically similar to Jazwinski's[1*] (but not mathematically the same) is used to let the model noise Q adapt to the residuals.

The adaptive algorithm is very simple. First a beta factor is calculated:

$$\beta = 1 - \exp\left(-\tfrac{1}{2} r^T S^{-1} r\right).$$

---

* References are listed at the end of this report.

An error vector, q, is then defined in terms of the residual vector, but dimensioned the same as $\bar{x}$:

$$q = (0\ 0\ \ldots\ q_k\ \ldots\ 0\ 0)^T$$

where k is the index of the target unit. The vector q_k depends on whether the measurement is a position measurement (x-and-y, range-and-bearing, or bearing-only) or velocity measurement:

$$\text{Position:}\quad q_k = (r_1 \quad r_2 \quad r_1/\tau \quad r_2/\tau)$$

$$\text{Velocity:}\quad q_k = (0 \quad 0 \quad r_1 \quad r_2).$$

Finally, the model noise, Q, is constructed by taking the outer product of q with itself and reducing the resulting matrix by the beta factor:

$$Q = \beta\ q\ q^T.$$

The predicted covariance of $\bar{x}$ is then given by:

$$P = F\ P_o\ F^T + Q\ .$$

The purpose of the adaptive model-noise matrix, Q, is to open up the predicted covariance so that the most recent measurement has a significant influence on the filtered state. When the residual vector is large, the Q matrix will also be large, P will then be large, and the relatively smaller measurement covariance, R, will cause the filtered state to be drawn to the measurement. This idea is shown in Figure 3.

Residuals can be large when a unit maneuvers; unfortunately, residuals can also be large for large but natural fluctuations in the measurement. Thus, there is a tradeoff between (1) making the algorithm sensitive to target maneuvers and letting it be influenced by bad data, and (2) decreasing the maneuver sensitivity so that bad data can be smoothed out. For the beta factor as defined above, the algorithm is rather sensitive to the residuals, as can be seen in Figure 4. For example, a two-sigma residual produces $\beta = 0.86$ and will cause a moderate-to-large influence on the predicted covariance. To reduce the influence

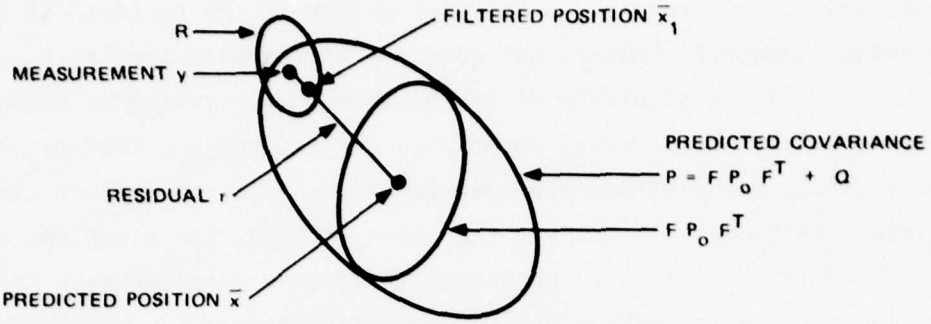


FIGURE 3 EXPANSION OF PREDICTED COVARIANCE IN THE DIRECTION OF THE RESIDUAL VECTOR

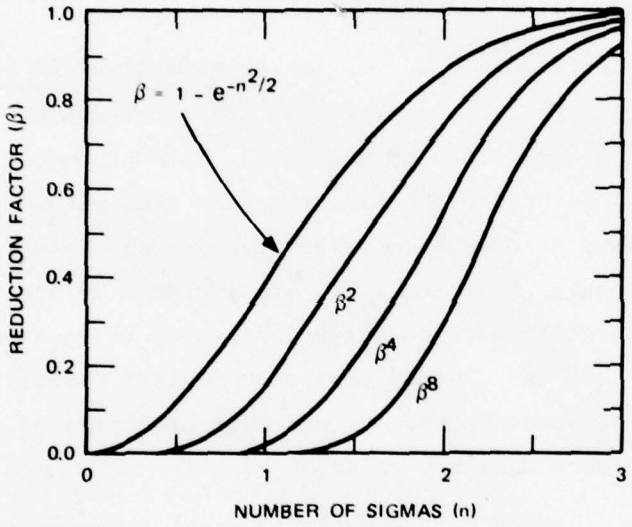


FIGURE 4 MODEL NOISE COVARIANCE REDUCTION FACTOR

of residuals, the beta factor can be raised to some power; for example, β^8 will cause the algorithm to react only to very large residuals.

The construction of the Q-matrix was intuitive rather than based on a probabilistic derivation. Our desire was to create an algorithm that was continuous--that is, a Q-matrix that does not suddenly jump in value as the residual becomes large.

An example of a Q-matrix that does jump is the following: let Q equal zero if the residual is inside, say,a two-sigma elliptical

window; and let Q equal a given noise matrix if the residual is outside the window. Another similar, but more sophisticated method is Bayesian in nature: Let the algorithm decide between a maneuvering or nonmaneuvering hypothesis. The maneuver covariance matrix, the a-priori probabilities, and a tactical value-of-decision matrix can be input to reflect the scenario conditions. In both of the above methods, the algorithm must choose between two (or more) predicted covariance matrices each time a measurement is taken. These decision-type algorithms can underreact to a slowly maneuvering target that stays inside the window, and overreact to bad data that fall outside the window. In an attempt to smooth out the under- and overreaction properties of the decision-type algorithm, the continuous-Q algorithm was created.

The choice of $q\,q^T$ as the basis matrix for Q was prompted by the desire to enlarge the predicted state covariance only in the direction that is indicated by the data. This procedure keeps the Q-matrix as small as possible but at the same time makes the predicted state covariance, P, consistent with the residual vector, r. The velocity components, r_1/τ and r_2/τ, are included in the q vector because the residual position error is assumed to be caused by a maneuvering target that has changed course and speed; therefore the velocity submatrix of the predicted covariance must be increased to let the state velocity move to a new value.

When multiple measurements occur at the same time, the order in which they are filtered affects the solution. The reason is that the covariance matrix, P, is a function of the data, whereas in a nonadaptive Kalman filter the covariance is independent of the data, and the order in which the simultaneous measurements are filtered makes no difference. With the adaptive algorithm, the solution will be better for one ordering of simultaneous measurements versus some other ordering. The best order cannot be determined as the calculations are made; this nonoptimum property of the MURLOC adaptive algorithm is part of the price that is paid for the capability of tracking a maneuvering target while using a constant-velocity model.

Two ways of computing simultaneous measurements were tried. The first method computed the Q-matrix for the first measurement only, and then let Q equal zero for the rest of the measurements. The first method did not perform as well as the second method, in which Q was computed for each of the measurements. For the second method, the value of tau in the velocity error computation was held constant and was set equal to the time duration between sets of measurements. The second method caused more fluctuations in the solution, especially in the velocity components, but it seemed to react to a maneuvering target more quickly than the first method.

4. Filtered State

Once the predicted state vector and covariance, $\bar{x}$ and P, were calculated, standard Kalman filter equations were used to compute the filtered state vector and covariance matrix, $\bar{x}_1$ and P_1:

$$K = P M^T (R + M P M^T)^{-1}$$

$$L = I - K M \quad (I \approx \text{Identity matrix})$$

$$\bar{x}_1 = \bar{x} + K r$$

$$P_1 = L P L^T + K R K^T .$$

The above covariance equation was used instead of the more common $P_1 = (I - K M) P$ equation, because the above method (1) produced a symmetric, positive definite matrix, and (2) was less sensitive to numerical errors in the filter gain matrix, K.

B. MURLOC Example

As a way of describing the input and output format of the MURLOC computer program, a four-unit scenario was created as shown in Figure 5. There are two observer ships, one helicopter, and one target ship. The first ship (Unit 1) measures a sonar bearing to the target ship (Unit 4) and launches the helicopter (Unit 3) to relay information and serve as a radar target. The first ship measures target bearings, radar range-and-bearing to the helicopter, and own course-and-speed. The second

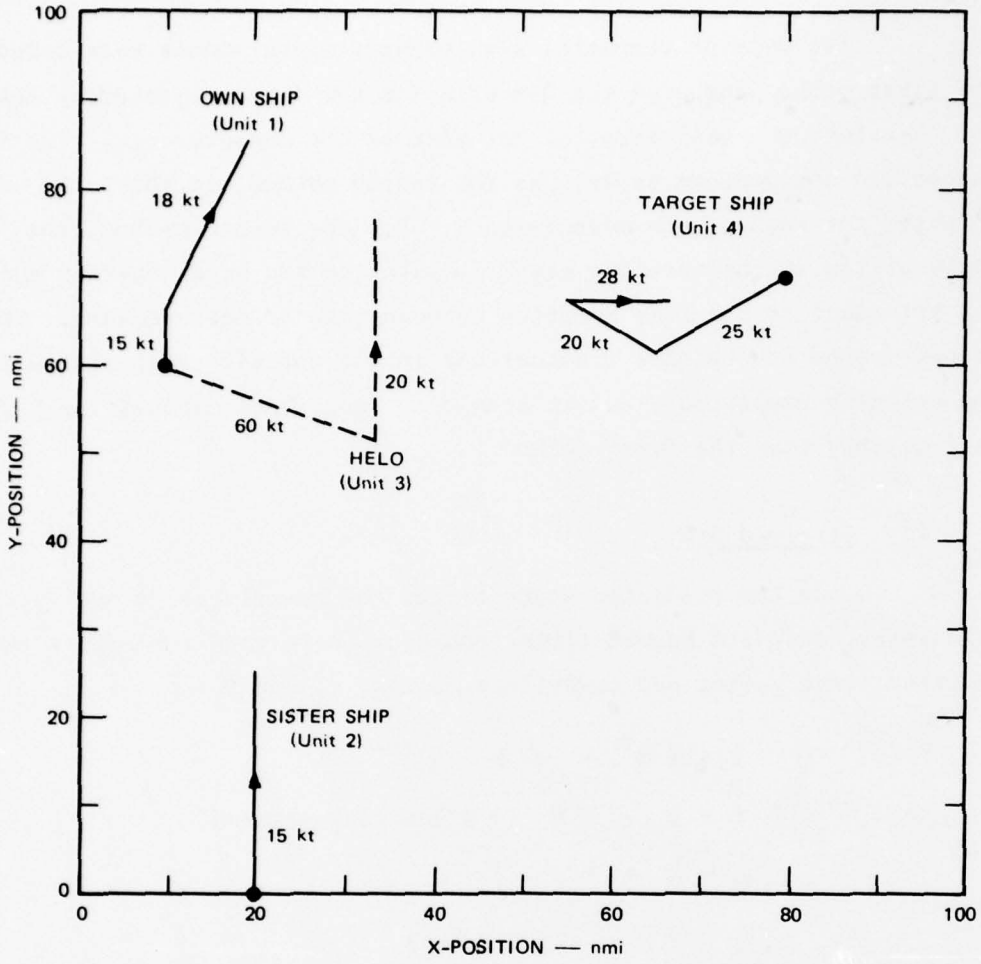


FIGURE 5 FOUR-UNIT SCENARIO FOR THE MURLOC EXAMPLE

ship (Unit 2) also measures target bearings, radar range and bearing to the helicopter, and own course and speed, and then relays this information to the first ship. The sonar bearing measurements from both ships are assumed to be smoothed for 10 minutes and have a standard deviation of 1.5 degrees. Table 3 shows the assumed sigmas of the various measurements.

Measurements from Unit 1 are assumed to be processed 5 minutes late and measurements from Unit 2 are processed 10 minutes late. Table 4 shows the time sequence of measurements. "Data time" is the time when the measurement is taken; "filter time" is the time when the measurement is processed. For example, at time t = 10, Unit 2 measures (1) bearing

Table 3

STANDARD DEVIATION OF MEASUREMENT ERRORS
IN THE MURLOC EXAMPLE

Measurement	Sigma
Bearing (1,4)*	1.5 deg
Bearing (2,4)	1.5 deg
Range (1,3)	0.5 nmi
Bearing (1,3)	2 deg
Range (2,3)	0.5 nmi
Bearing (2,3)	2 deg
Course (1)	1.9 deg
Speed (1)	1 kt
Course (2)	3.8 deg
Speed (2)	2 kt

* Bearing (1,4) is the bearing from Unit 1 to Unit 4, etc.

to Unit 4, (2) range and bearing to Unit 3, and (3) own course and speed. These data are relayed to Unit 1 via the helicopter, and 10 minutes later (t = 20) they are filtered onboard Unit 1. Table 4 shows that the example data are mostly bearing measurements to the target with occasional radar measurements on the helicopter and course-and-speed measurements on the ships.

The numbers listed in Table 4 are the true values of the parameters calculated at the data time; thus, the tracking example uses measurements with no errors. Tracking results produced by zero-error measurements cannot be used to evaluate the tracking algorithm, but instread can be used to roughly estimate the magnitude of tracking errors that would be expected if the measurements were randomized around the true values. Zero-error measurements were used for the MURLOC example because the

Table 4

LIST OF MEASUREMENTS FOR THE MURLOC EXAMPLE

Time (min)		Sonar Bearing (deg)		Radar Range Bearing (nmi, deg)		Course, Speed (deg, kt)	
Data	Filter	Units 1,4	Units 2,4	Units 1,3	Units 2,3	Unit 1	Unit 2
0	5	081.9				000, 15	
5	10	083.6		5.6, 122.2			
10	20		040.8		54.1, 359.4		000, 15
15	20	087.2					
20	30		040.9				
25	30	091.4		27.8, 122.2		025, 18	
30	40		041.2		47.6, 016.5		
35	40	096.4		Effect of these measurements shown in Appendix B			
40	50		041.4				
45	50	100.1					
50	60		040.0				
55	60	102.2					
60	70		038.5				
65	70	104.6					
70	80		036.9				
75	80	107.5					
80	90		038.5				
85	90	109.5					
90	100		043.3				
95	100	111.2					
100	110		048.0	17.3, 126.1	53.2, 014.7		

emphasis of this section is on the input and output format of MURLOC rather than its performance. Tracking performance is the subject of the next chapter.

1. Input

There are three versions of the MURLOC computer program: (1) a time-share prototype version that requires the data to be entered interactively, (2) a generalized version that uses card input, and (3) a specialized version that uses randomized data contained on a disc file. The card-input version of MURLOC is listed in Appendix A and is the subject of this section. The file-output version of MURLOC was used in the Monte Carlo performance analysis of the next chapter.

Table 5 shows the data card format that is used to input measurements to the program. For example, the card for the bearing measurement from Unit 2 to Unit 4 at time t = 10 reads:

TP	TD	KEY	KO	K	X1	X2
20.0	10.0	3	2	4	40.8	1.5

Values for X3 through X5 do not need to be punched for a bearing measurement card.

To initialize the program, Card 1 contained TD = 0, KEY = 6, and KO = 4. This card told the program to initialize the state at t = 0 and to set the number of units to 4. Card 2 was a position measurement for Unit 1 so that the coordinate system could be defined. Card 3 was a course-and-speed measurement on Unit 1. Card 4 initialized the position of the target by using a range-and-bearing format; bearing was measured (81.9°) and range was estimated. The bearing-only format could not be used for the first bearing measurement because the position of the target was not yet defined; therefore a large, but reasonable, value of range was used--in this case, 50 nmi, as compared to the true value of 70 nmi. Card 5 set the course and speed of the target to zero and the velocity sigmas to 30 knots. Cards 6 through 34 contained the data shown in Table 4 starting with the second line. The final card contained KEY = 7 to end the run.

Table 5

DATA CARD FORMAT

Variable	Meaning	Units	Columns	Format*
TP	Time at which the measurement is filtered	min	1-10	F
TD	Time of measurement	min	11-20	F
KEY	Indicator: 1 = Position measurement 2 = Range-and-bearing measurement 3 = Bearing-only measurement 4 = Course-and-speed measurement 5 = Output 6 = New run 7 = End	-	21	I
KO	Index of observing unit	-	25	I
K	Index of target	-	27	I
X1	Key: 1 = x-position† 2 = Range 3 = Bearing (true) 4 = Course	 nmi nmi deg deg	31-40	F
X2	1 = y-position 2 = Bearing (true) 3 = Bearing sigma 4 = Speed	nmi deg deg kt	41-50	F
X3	1 = Minor axis sigma 2 = Range sigma 4 = Sigma perpendicular to velocity vector	nmi nmi	51-60	F
X4	1 = Major axis sigma 2 = Bearing sigma 4 = Sigma parallel to velocity vector	nmi deg kt	61-70	F
X5	1 = Angle to major axis from North	deg	71-80	F

* F = Floating point, I = Integer.

† Value of KEY parameter determines the meaning of X1 through X5.

2. Output

A partial listing of the output from the example scenario is contained in Appendix B. The program prints out the information contained on a data card and then prints the results of filtering those data. The filtered results are: (1) the time of the data and the time of the calculation, (2) the location of one unit relative to another, and (3) the position and velocity of each unit. The relative location parameters are:

OBSR UNIT RNG BRG SIG1 SIG2 ANG CEP.

Under the OBSR and UNIT headings are listed all combinations of observer and target unit indices. RNG and BRG are the "predicted range and bearing"; this parameter pair is a polar coordinate representation of the vector from the predicted position of the observer to the target (RNG and BRG are not average values of range and bearing). SIG1 and SIG2 are the predicted minor axis and major axis standard deviations, and ANG is the angle from North to the major axis of the error ellipse. The error ellipse is in a coordinate system that is relative to the observer unit. CEP is the predicted radius of a circle, centered on the point defined by the predicted range and bearing, such that there is a 50 percent probability that the point defined by true values of range and bearing lies inside the circle. CEP is a single-parameter measure of the three-parameter error ellipse, and is useful in quickly assessing the track quality. CEP is a function of SIG1 and SIG2; the equations are given in Function FCEP, which is in Appendix A.

The output parameters for the predicted position of the units are:

UNIT X Y SIG1 SIG2 ANG CEP

and the parameters for the predicted velocities are:

CRS SPD SIG1 SIG2 ANG CEP.

The predicted x-and-y position of each unit is listed along with the parameters describing the position error ellipse (SIG1, SIG2, ANG, and CEP). Error ellipses are in absolute (geographic) coordinates, not in relative coordinates as are the range-and-bearing position error ellipses. CRS and SPD are the "predicted course and speed" of each unit; this parameter pair is a polar coordinate representation of the predicted velocity vector (CRS and SPD are not average values of course and speed). The next four parameters (SIG1, SIG2, ANG, and CEP) are parameters of the velocity error ellipse. A velocity error ellipse is used because this is a more natural way to show the velocity covariance terms than numerically calculating the predicted course and speed error standard deviations and correlation between course and speed errors.

Figure 6 shows the results of the four-unit scenario example. The range from the first ship to the target ship is plotted against time. The target maneuvers at t = 40 and again at t = 75 minutes. The first maneuver is not discernible on the range plot, but the second maneuver is more radical and is easily seen. The true range is plotted and the tracking results using zero-error measurements are shown as a dotted region. The region shows the approximate magnitude of position errors that would be expected if randomized measurements were used. The dotted region was plotted by using the predicted range plus and minus the relative-position CEP value of Unit 4 relative to Unit 1. The stairstep effect is caused by the ever-expanding predicted-position CEP being periodically contracted by measurements that are filtered at 10-minute intervals. The range estimate overshoots the second maneuver but comes back and brackets the true range. The overshoot is mostly due to the 10-minute delay in information from the second ship, rather than slow reaction of the tracker. Figure 6 shows that position errors of roughly 5 nmi can be expected in a four-unit targeting scenario involving over-the-horizon ranges and measurement errors as shown in Table 2.

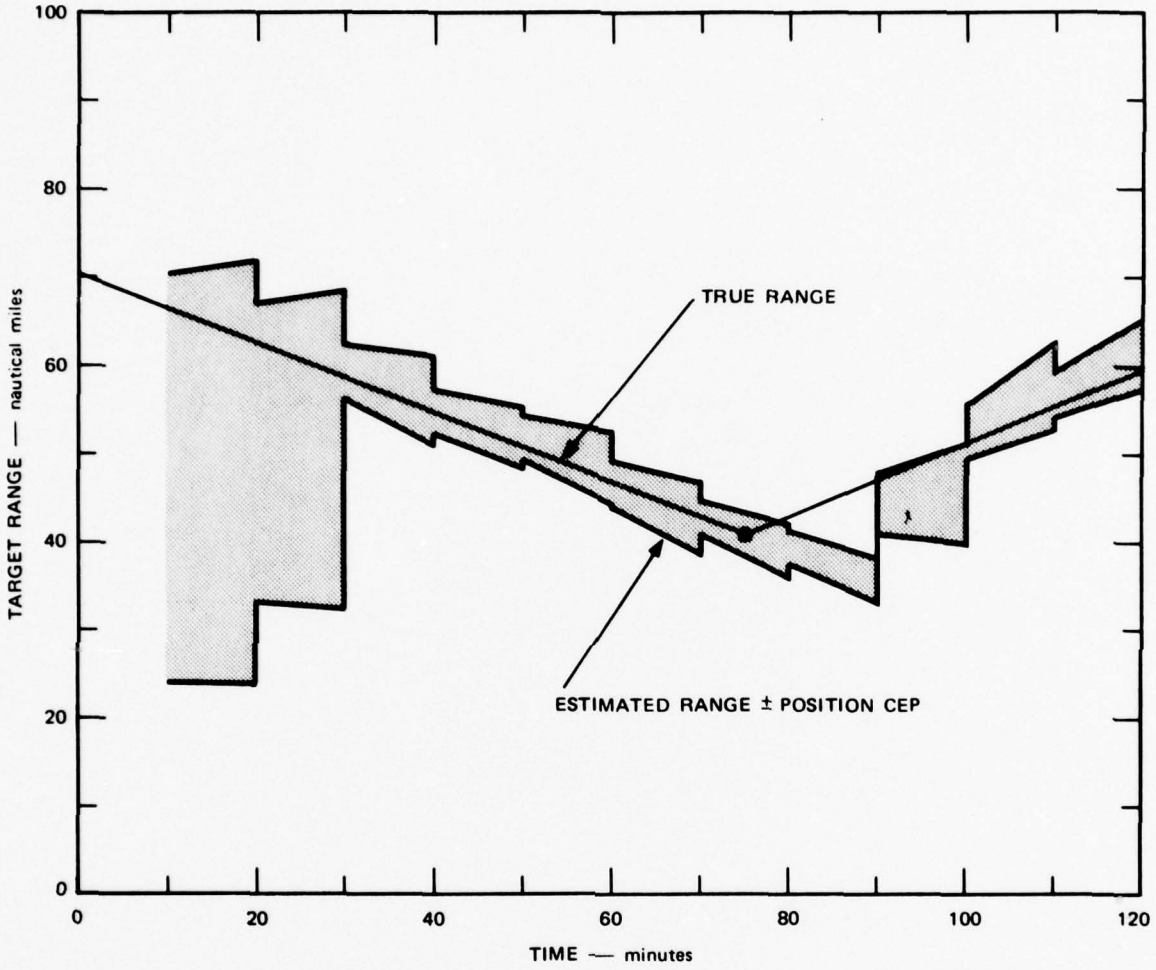


FIGURE 6 MURLOC TRACKING RESULTS USING ZERO-ERROR MEASUREMENTS IN THE FOUR-UNIT SCENARIO

III MURLOC TRACKING CAPABILITY

MURLOC's tracking capability was investigated by a Monte Carlo analysis of tracking errors. A three-unit scenario was defined and 50 replications of randomized measurements were calculated and stored on a disc file. Then MURLOC was run 50 times using the random measurements; the resulting position and velocity errors were averaged and output as measures of tracking capability. In addition to running the MURLOC algorithm, two versions of another tracking algorithm were run to compare against MURLOC. Thus, by using the same file of 50 replications in all three algorithms, a valid comparison could be made. The comparison algorithms employed a batch-processed least-squares methodology that is described in part in this chapter and in detail in the next chapter.

A. Three-Unit Scenario

Figure 7 shows the three-unit scenario used for the tracking capability analysis. Two ships (Units 1 and 2) cross-fix the target ship (Unit 3) with bearing-only measurements, and find their baseline with range-and-bearing measurements. Measurement sets were 10 minutes apart; there were 10 sets of measurements (10 time points) each replication, and 8 measurements were made within each measurement set. Table 6 lists the 8 measurements that were generated at each time point, along with the standard deviations used in the randomizing the measurements. A normal distribution of measurement error was used and the mean value of the measurement of a parameter was set equal to the true value of the parameter at the time point. There was no delay time between measurement and calculation; in the vocabulary of the previous chapter, "data time" equaled "filter time." The two observing ships maintained a constant course and speed throughout the scenario. The target ship made a 90-degree course change at the fifth time step (t = 40 min), but maintained its speed of 24 knots.

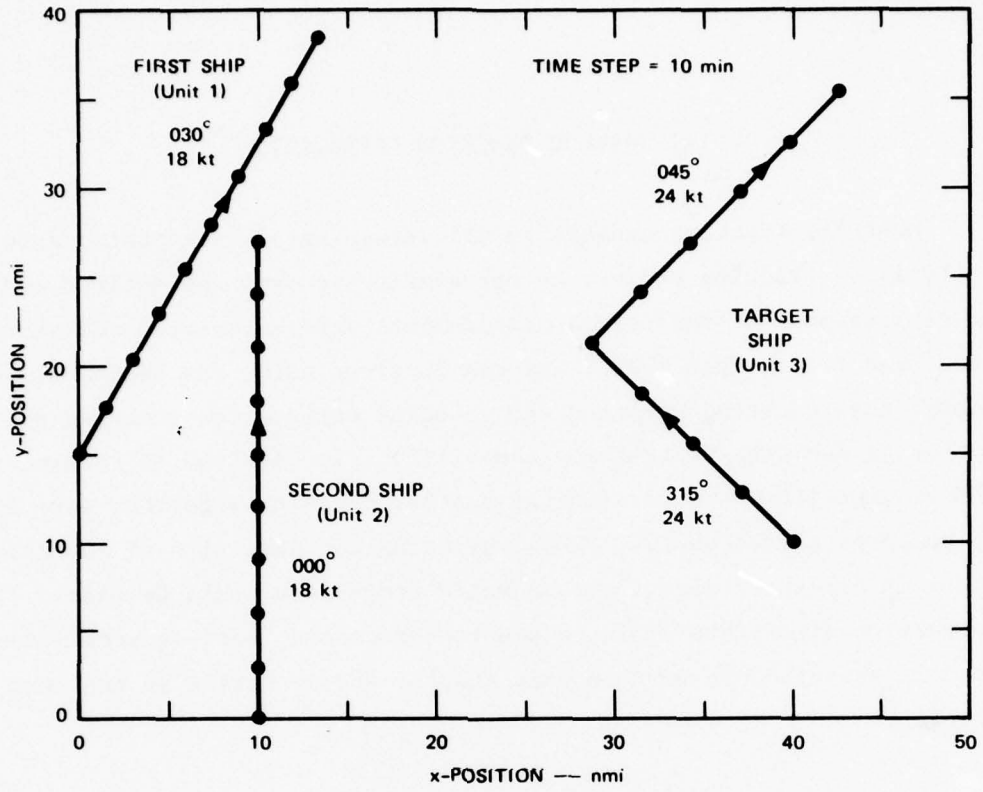


FIGURE 7 THREE-UNIT SCENARIO FOR THE MURLOC TRACKING CAPABILITY ANALYSIS

At each time step, MURLOC processed the 8 measurements as 5 groups of simultaneous measurements. The order in which the measurement groups were filtered is shown in Table 7. Switching the order of Bearing (2,3) and Bearing (1,3) degraded the tracking result slightly. Other than investigating the above bearing switch, no attempt was made to find the best order for the simultaneous measurements.

B. Least-Squares Comparison

A least-squares methodology was used to compare against MURLOC by formulating two algorithms: An algorithm that decided the best set of measurements using an optimization procedure, and (2) an algorithm that was told when the target maneuvered and could therefore process the best set of measurements. These two algorithms are called "self-optimizing" and "maneuver-known," and are discussed below.

Table 6

STANDARD DEVIATION OF MEASUREMENTS
IN THE THREE-UNIT SCENARIO

Measurement	Sigma
1. Bearing (1,3)*	1.5 deg
2. Bearing (2,3)	1.5 deg
3. Range (1,2)	0.5 nmi
4. Bearing (1,2)	2 deg
5. Course (1)	1 deg
6. Speed (1)	1 knot
7. Course (2)	1 deg
8. Speed (2)	1 knot

* Bearing (1,3) is the bearing *from Unit 1 to Unit 3, etc.*

Table 7

ORDER OF FILTERING THE MEASUREMENTS IN MURLOC

Order	Measurement Group
1	Bearing (2,3)
2	Range-and-bearing (1,2)
3	Bearing (1,3)
4	Course-and-speed (1)
5	Course-and-speed (2)

1. Self-Optimizing Algorithm

The self-optimizing least-squares algorithm processed a variable number of measurements at each time step. For example, at the n-th time step, the algorithm first processed 16 measurements: 8 measurements from the current time, t = n, and 8 from the immediately preceding time, t = n - 1. These 16 measurements were used to estimate 10 state parameters: 2 velocity components for Unit 1, 4 position and velocity components for Unit 2, and 4 components for Unit 3. With a state estimate based on 16 measurements, the algorithm then calculated a test statistic and saved it. The process was repeated using 24 measurements from time steps t = n, n - 1, and n - 2, and again a test statistic was calculated and saved. The calculations were repeated as many times as it took to use up all of the measurements; thus at the n-th time step, n - 1 state estimates and test statistics were calculated. Finally, the measurement set that produced the minimum test statistic was picked as the "best" set to use at time t = n.

The above optimization procedure was used because of the maneuvering-target problem. If tracks were always straight lines, then the calculation of state estimates would utilize as many measurements as possible. But the chance of a maneuvering target required the algorithm to decide what measurements to use, since measurements taken before a maneuver cause large errors in estimating the state after the maneuver. Ideally, the optimization procedure would process all measurements from the time of the maneuver onward. However, in practice, the test that is used to determine the measurement set necessarily operates on random values and cannot always determine the truly best set of measurements.

2. Maneuver-Known Algorithm

The maneuver-known, least-squares algorithm was allowed to process the truly best set of measurements each time step. For example, at Time Step 9, the measurements at 9, 8, 7, 6, and 5 were used, but measurements before Step 5 were not used because they occurred before the target maneuver. The reason for defining the maneuver-known algorithm

was to show the best possible use of the measurements. Since it was a batch-processed least-squares methodology that is allowed to know the time of target maneuver, the algorithm produced a lower bound on the tracking error. It must be remembered, though, that MURLOC and the self-optimizing least-squares algorithm were real trackers, whereas the maneuver-known algorithm was not real, in the sense that it could not calculate state estimates based solely on measurements.

C. Results

The position and velocity errors as a function of time were averaged over 50 replications and used to determine tracking capability.

1. Position Errors

"Position error" is defined as the distance from the true position of the target to the estimated position in a coordinate system that is relative to Unit 1. Position error is thus relative position error, and it is the magnitude of the vector difference of the estimated position vector minus the true position vector, using Unit 1 as the origin for the two vectors.

Figure 8 shows the average position error as a function of time for MURLOC and the two least-squares algorithms. MURLOC performed very well when compared to the lower bound.

The fact that MURLOC and the self-optimizing algorithm performed about the same suggests that the optimizing procedure could be improved for the least-squares tracker. The test statistic that was used is related to the sum-of-squares of the residuals. Another test statistic is suggested in the next chapter, but we did not have the resources to investigate it or other alternative self-optimizing procedures.

Figure 9 shows the percent of replications for which the position error is less than the estimated CEP of Unit 3 relative to Unit 1. For each replication and at each time step, CEP is estimated from the covariance matrix. This single-parameter estimate of position error is

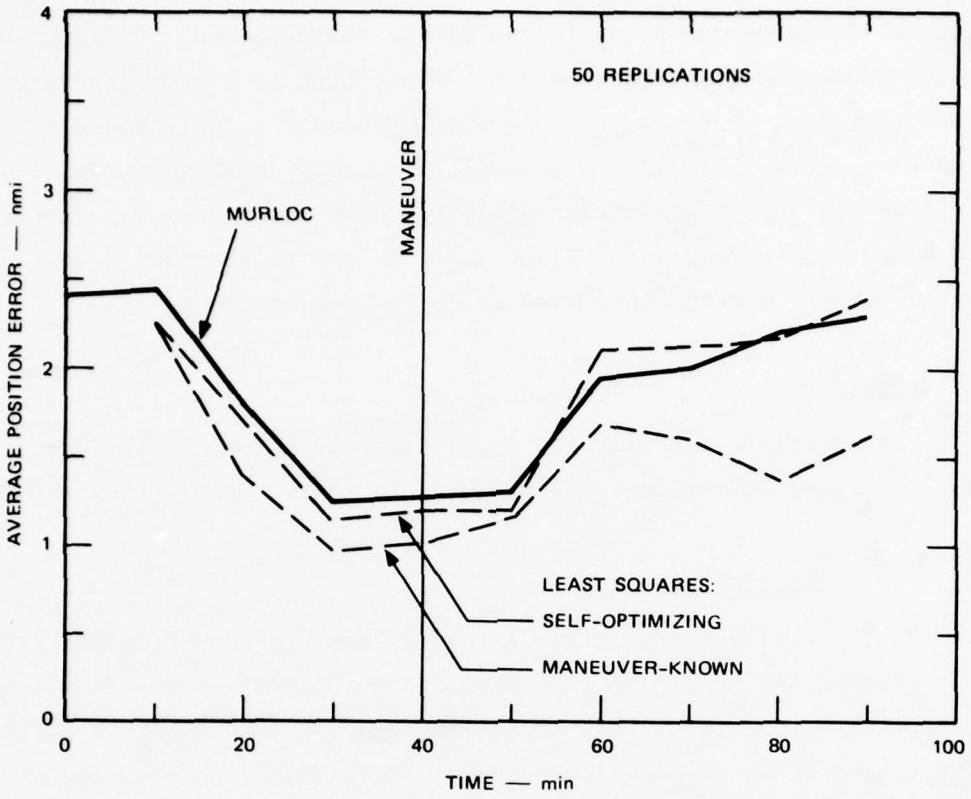


FIGURE 8 AVERAGE POSITION ERROR COMPARISON OF MURLOC TO LEAST-SQUARES ALGORITHMS

useful in making targeting decisions; however, CEP is a random variable and may not represent the actual position errors. Figure 9 indicates that, in both MURLOC and the self-optimizing least-squares algorithm, the CEP values were smaller than they should have been. If CEP were estimated better, approximately 50 percent of the replications would result in position errors less than the CEP values.

The reason for the overly optimistic estimate of the error is not known, but since the maneuver-known algorithm approached the theoretical curve better than the other two algorithms, optimum use of the data may play an important part in correctly estimating error variances.

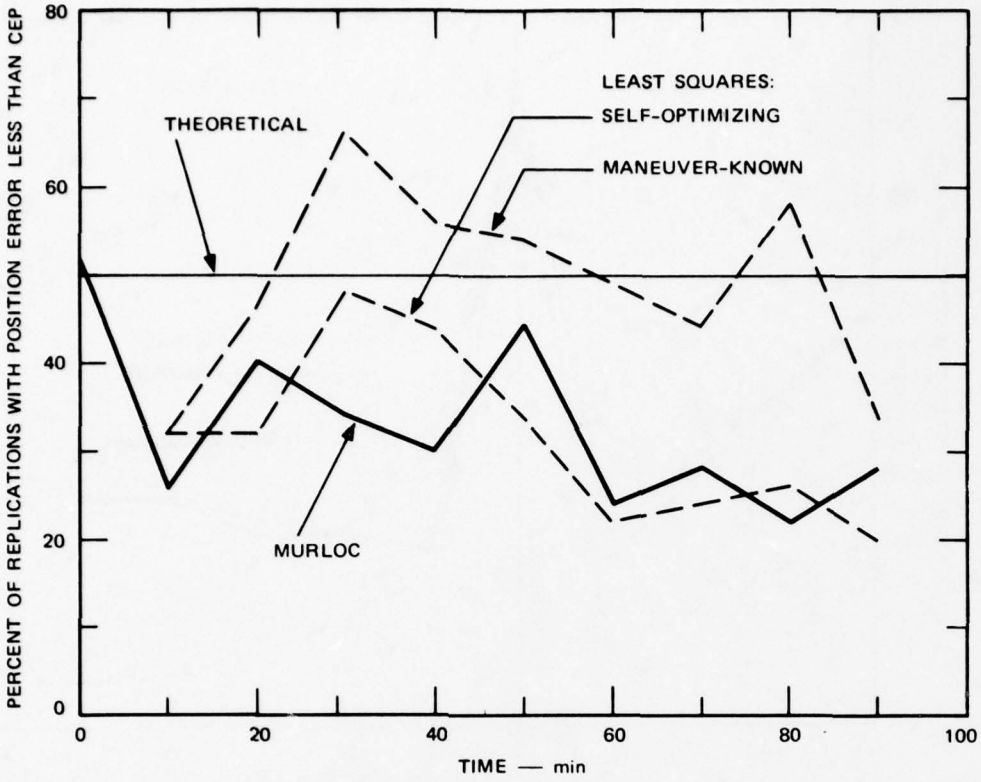


FIGURE 9 POSITION CEP PREDICTIVE CAPABILITY

2. Velocity Errors

"Velocity error" is the distance, in velocity space, from the true velocity point to the estimated velocity point of Unit 3. In other words, velocity error is the magnitude of the vector difference of the estimated velocity vector and the true velocity vector of the target.

Figure 10 shows the average velocity errors as a function of time for the 3 tracking algorithms. Even at best, 10 knots error is not too impressive, for MURLOC or the self-optimizing algorithm, but the long-range geometry combined with the 1.5-degree bearing sigmas make the estimation of velocity most difficult. The maneuver-known algorithm results are significantly lower, and show the improvement that might be possible with an accurate maneuver detector.

Even though the maneuver is known to occur at t = 40 min, a large velocity error is recorded for the maneuver-known algorithm

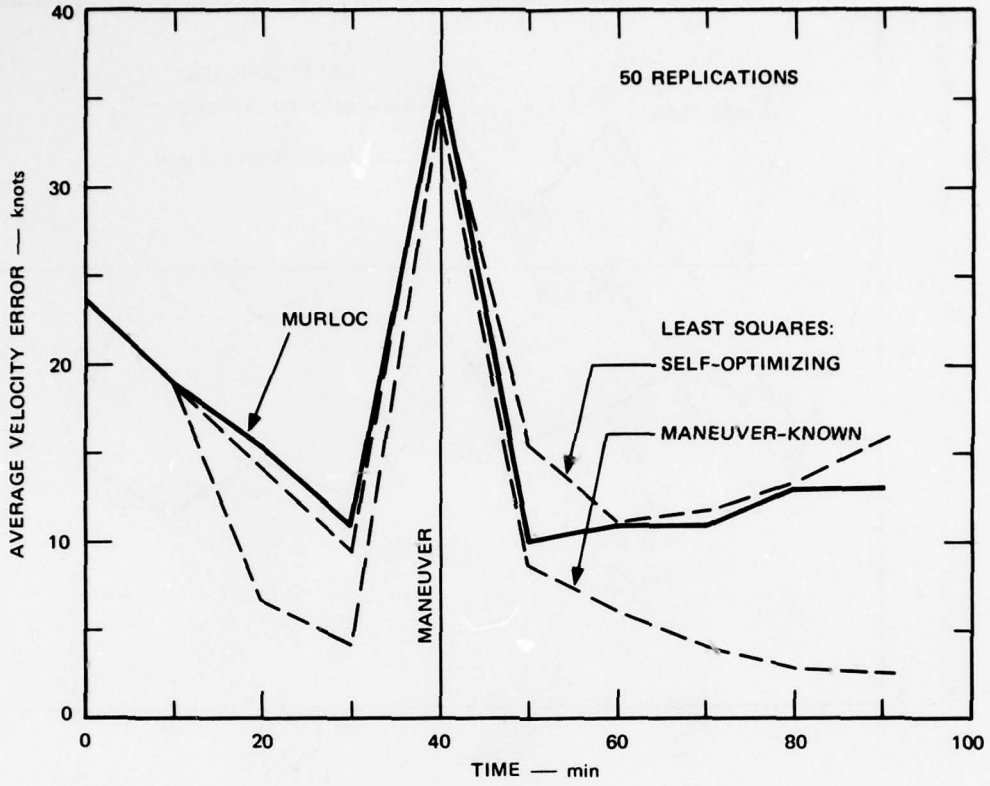


FIGURE 10 AVERAGE VELOCITY ERROR COMPARISON OF MURLOC TO LEAST-SQUARES ALGORITHMS

because the velocity estimate was based on data from Time Steps 1 through 5. The target course estimate was close to 315 degrees, but the program defined the target's course at Step 5 as 045 degrees, whereas actually the course is both 315 and 045 degrees. Thus, a large velocity error shown at the time of maneuver is a result of a programming definition, rather than a result of estimation difficulties. These comments also apply to the two other algorithms.

Figure 11 shows the effect on the MURLOC results when the bearing errors for both Bearing (1,3) and Bearing (2,3) are drawn from a normal distribution with a standard deviation of 1 degree instead of 1.5 degrees. The improvement indicates that the velocity errors are significantly affected by the underlying error structure of the scenario.

Figure 12 shows the percent of replications that had velocity errors less than the estimated velocity CEP. The results show that

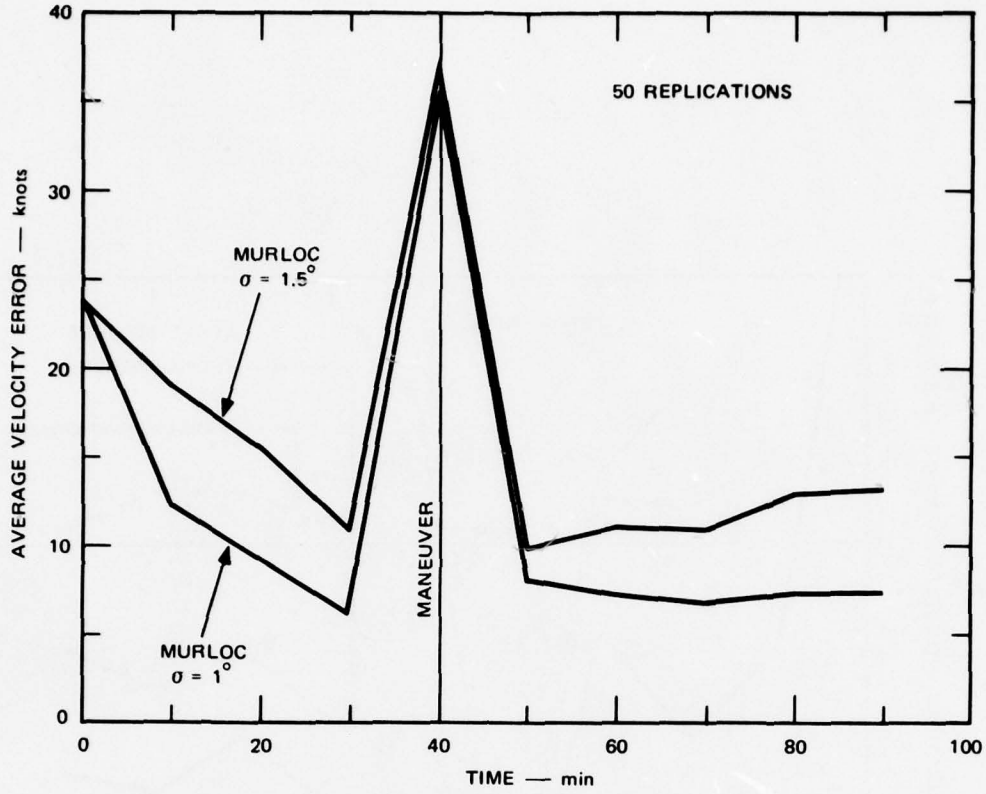


FIGURE 11 EFFECT OF SIGMA BEARING ON MURLOC VELOCITY ERROR

MURLOC and the self-optimizing algorithm were as overly optimistic in velocity estimates as they were in position estimates. The maneuver-known algorithm again approximates the theoretical.

3. Summary

The Monte Carlo results for the defined scenario show that MURLOC can track a maneuvering target, that the target position errors in relative coordinates will be close to the lower bound, and that the target velocity errors will be reasonably small, as compared to a self-optimizing least-squares method. MURLOC estimates its own errors smaller than they really are, and therefore decisions based on error estimates would have to take MURLOC's overly optimistic behavior into account. An examination of other scenarios and other least-squares optimization procedures would undoubtedly add to the knowledge of MURLOC's tracking capability, but the above basic analysis does show that MURLOC is a reasonably good tracking algorithm.

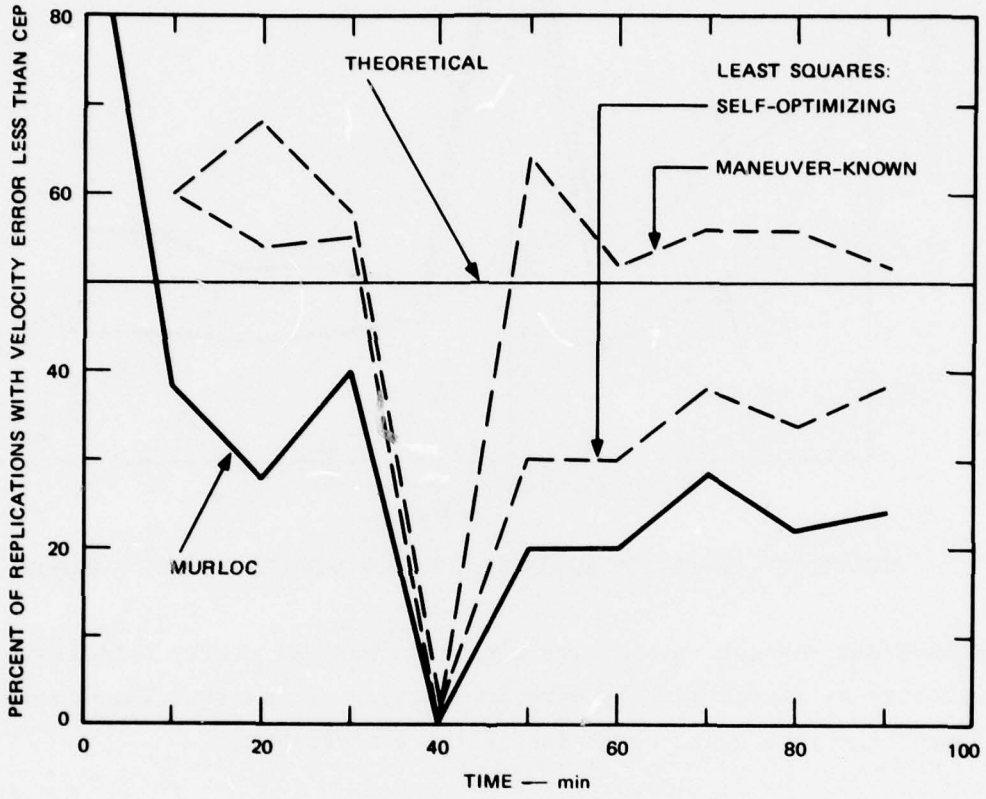


FIGURE 12 VELOCITY CEP PREDICTIVE CAPABILITY

IV THE COMPARISON ALGORITHM

The tracking capability of MURLOC was compared to the capability of another tracking algorithm that employed a batch-processed self-optimizing, least-squares methodology. Upon receiving measurements at the n-th time step, the self-optimizing algorithm computed an estimate of the state vector using measurements from time n and n - 1; then it computed another state estimate using measurements from time n, n - 1, and n - 2; and so on down to the first time step. The n - 1 state estimates were checked to see which one produced the smallest value of a defined test statistic; the chosen one was then used as the best state estimate at time step n. State estimates were calculated by processing measurements in a nonlinear weighted least-squares algorithm. The algorithm employed an iterative method to find the best least-squares solution. The sections that follow describe the comparison algorithm in detail.

A. Least-Squares Problem

At each time step, n, a state estimate, z_{pn} (where p = 1 to n - 1), is computed such that a sum of weighted squares is minimized. The sum of squares, G_{pn}, is given by:

$$G_{pn} = \sum_k \sum_j \left[m_k^j - F_k^j(z_n) \right]^2 \Big/ M_k^j$$

for time-step summation indices k = p to n, and measurement indices j = 1 to 8. The actual state vector at time n is denoted z_n, the measurements at time k are denoted m_k^j, and the measurement models at time k are denoted $F_k^j(z_n)$; these three objects are defined below. The measurement variances at time k are denoted M_k^j; they are input parameters.

There are n - 1 state estimates, z_{pn}, at time step n, and one of them is better than the others. The best one is found by computing a test

statistic, S_{pn}, and choosing the state estimate that produces the smallest value S_{pn}. The test statistic is defined in Section C of this chapter.

1. State Vector

At each time, n, where n ranges from 2 to N, an estimate of position and velocity for each of 3 units is required. This estimate is based on a contiguous time-sequence of measurements. Let

x_{ni} = The x-coordinate of Unit i at time n

y_{ni} = The y-coordinate of Unit i at time n

$\dot{x}_{ni}$ = The velocity in the x-direction at time n

$\dot{y}_{ni}$ = The velocity in the y-direction at time n

where i = 1,2,3 is the unit index. Position is expressed in nautical miles and velocity is expressed in knots. The state vector has 10 components and is defined as:

$$z_n = (\dot{x}_{n1}\ \dot{y}_{n1}\ x_{n2}\ y_{n2}\ \dot{x}_{n2}\ \dot{y}_{n2}\ x_{n3}\ y_{n3}\ \dot{x}_{n3}\ \dot{y}_{n3})^T \quad .$$

It is convenient to let z_{nj} denote the j-th component of z_n.

The state vector does not contain the x and y positions of Unit 1 because only the relative positions need to be estimated. In effect, the x,y-coordinate system is redefined at each time step relative to the position of Unit 1.

2. Measurement Vector

The measurement vector at time n is defined as:

$$m_n = \left(m_n^1 \ \ldots \ m_n^8\right)^T$$

where the m_n^j (j = 1 to 8) are defined in Table 8. We assume that m_n is a random vector distributed as a multivariate normal deviate and that the components of m_n are uncorrelated. We call M_n the covariance matrix of m_n and use M_n^j to denote the j-th diagonal term of M_n.

Table 8

DEFINITION OF MEASUREMENTS FOR THE COMPARISON ALGORITHM

Symbol	Meaning
m_k^1	Bearing (1,2)*
m_k^2	Bearing (1,3)
m_k^3	Course (1)
m_k^4	Range (1,2)
m_k^5	Speed (1)
m_k^6	Bearing (2,3)
m_k^7	Course (2)
m_k^8	Speed (2)

*Bearing (1,2) is the bearing from Unit 1 to Unit 2 at time step k.

3. Measurement Model

A model of the measurements m_k at time k can be constructed from the state vector and the position of Unit 1 at time n . First define the relative x-position by:

$$x_{ij}^{nk} = (x_{nj} - x_{ni}) + (k - n)\,\tau\,(\dot{x}_{nj} - \dot{x}_{ni})$$

where τ is the time-step duration, and i and j are unit indices. Define a similar equation for y_{ij}^{nk}. Then components of the measurement model can be written; for example, the components corresponding to m_k^1 and m_k^8 are:

$$F_k^1(z_n) = \tan^{-1}\left(x_{12}^{nk} / y_{12}^{nk}\right)$$

$$F_k^8(z_n) = \left((\dot{x}_{n2})^2 + (\dot{y}_{n2})^2\right)^{\frac{1}{2}} \quad .$$

We say that

$$F_k(z_n) = \left(F_k^1(z_n) \ \ldots \ F_k^8(z_n)\right)^T$$

is the measurement model at time k.

B. Nonlinear Least-Squares Algorithm

References that surveyed literature directly addressing the solution of nonlinear-least squares problems were examined; see Bard,[2] Broyden,[3] Draper and Smith,[4] Fletcher,[5] and Powell.[6] The examination revealed that all algorithms that solve nonlinear least-squares problems are iterative. They calculate a sequence of points V_1, V_2, ... , that should converge to a V*, which solves the nonlinear least-squares problem. These algorithms frequently substitute linear approximations evaluated at V_k for the nonlinear functions, and these linear approximations are then used to calculate V_{k+1}. A conclusion of this review of relevant literature was that the Gauss-Newton method, and variants of it, were the best algorithms with which to minimize the sum of squares, G_{pn}. The particular variant of the Gauss-Newton method that we used in the optimized, iterative, least-squares tracking algorithm makes use of a technique to ensure that G_{pn} strictly decreases for each point of the sequence generated.

In the following sections, G_{pn} is denoted G(z); in other words, the p and n subscripts are suppressed and the functional dependence on the state vector $z \equiv z_n$ is emphasized.

1. Gauss-Newton Method

Algorithm (a) in Bard[2] was used as the variant of the Gauss-Newton method for handling the nonlinear problem. For the first iteration, the algorithm was provided an initial estimate of the state. To reduce the chance of a nonconverging solution, we used the true state values to start the Gauss-Newton algorithm.

At time step n and iteration i, $F_k^j(z)$ is approximated by the linear function:

$$f_k^j(z) = F_k^j[z(i)] + \sum_s J_{ks}^j(i)\,[z - z(i)]$$

where the state summation index is s = 1 to 10, the measurement index is j = 1 to 8, the time step index is k = p to n, and:

$$J_{ks}^j(i) = \left.\frac{\partial F_k^j(z_n)}{\partial z_{ns}}\right|_{z_n = z_n(i)} .$$

The values $f_k^j(z)$ are used to approximate $G(z)$ by the sum of squares:

$$g(z) = \sum_k \sum_j \left[m_k^j - f_k^j(z)\right]^2 \Big/ M_k^j \quad .$$

The vector $d(i)$, which minimizes $g(z)$, is then determined.

The estimate of state at iteration i is denoted $z(i)$. A new estimate $z(i+1)$ is found using the formula:

$$z(i+1) = z(i) + r\,d(i)$$

where $d(i)$ is the solution of least-squares problem at time n and iteration i, and r is a scalar chosen so that:

$$G[z(i+1)] < G[z(i)] \quad .$$

The iterations at time n continue until any one of the following four termination criteria is satisfied:

- The number of interpolations (defined below) reaches its maximum allowed value (10 were allowed) and

$$G[z(i) + r\, d(i)] \geq G[z(i)] \quad .$$

- The iteration counter, i, reaches its maximum allowed value (5 iterations were allowed).
- All absolute components $|r\, d(i)|$ are less than 0.0001 times the corresponding absolute components $|z(i)| + 0.001$; see Bard,[2] page 168.
- $G[z(i)] - 0.001 < G[z(i+1)] < G[z(i)]$.

The scalar r at iteration i is calculated by the following steps:

(1) Use the value of the interpolation index q from iteration $i - 1$ (if $i = 1$ set $q = 0$).

(2) Divide q by 2 and retain the integral part.

(3) Set $r = 2^{-q}$.

(4) If $G[z(i) + r\, d(i)] < G[z(i)]$, use $z(i+1) = z(i) + r\, d(i)$ as the new estimate of state. Otherwise perform Steps 5 through 8.

(5) Construct as a function of s the polynomial of degree 2 that equals $G[z(i) + s\, d(i)]$ for the two points $s = 0$ and $s = r$, and for which the slope equals $h^T V h$ at $s = 0$. The vector h and the matrix V are defined in the next section.

(6) Compute the value s at which the first derivative of the polynomial is zero and call this value s_1. Let

$$s_2 = \max [0.25r, \min (0.75\, r, s_1)] \quad .$$

An execution of Step 6 is called an "interpolation."

(7) Increase q by one.

(8) Replace r with s_2 and return to Step 4.

2. Least-Squares Solution

Several algorithms were available to compute $d(i)$, (see Lawson and Hanson).[7] Since the covariance matrix of $d(i)$ was desired, an algorithm was selected that provides the covariance matrix as a

byproduct of the computation of $d(i)$. Execution time and programming ease were also considered. Consequently, the normal equations of the linear weighted least-squares problem were derived, the Cholesky (or square-root) method was applied to the matrix determined by the normal equations, the upper triangular Cholesky factor was inverted, and the $d(i)$ that solved the linear weighted least-squares problem and its covariance matrix was then computed.

Below we show the form of the normal equations for the linear least-squares problem. Let p and n be integers such that $p = 1$ to $n - 1$. Let the 8-by-10 matrix, $J_k(i)$, contain the (j,s)-th component, $J^j_{ks}(i)$ for $k = p$ to n. Define the matrixes:

$$J = \left[J_n^T(i) \ \ldots \ J_p^T(i) \right]^T$$

$$M = \begin{bmatrix} M_n & & 0 \\ & \ddots & \\ 0 & & M_p \end{bmatrix} .$$

Define the vector:

$$D = \begin{bmatrix} m_n - F_n[z(i)] \\ \vdots \\ m_p - F_p[z(i)] \end{bmatrix} .$$

The normal equations at time n and iteration i are then given by:

$$J^T M^{-1} J \, d(i) = J^T M^{-1} D .$$

Now $J^T M^{-1} J$ is symmetric and positive definite; thus Cholesky's method (see Lawson and Hanson[7] or Forsythe and Moler[8]) is used to find an upper triangular matrix U such that:

$$U^T U = J^T M^{-1} J$$

and U is invertible by using formulas from Lawson and Hanson.[7] The covariance matrix of the solution of the linear weighted least-squares problem, V, is computed from the formula:

$$V = U^{-1} (U^{-1})^T \quad .$$

The vector h is:

$$h = J^T M^{-1} D$$

and the solution of the linear weighted least-squares problem, d(i), is calculated by:

$$d(i) = V h \quad .$$

C. Test Statistic

Two test statistics are presented below; the first was used in the self-optimizing least-squares algorithm presented in Chapter III. The second statistic is suggested as a possible candidate for the optimization process, but no experimentation was performed to see if it would work better than the first statistic.

1. Residual Mean Square

The residual mean square, S, was the test statistic minimized by the self-optimizing algorithm. For each time step n, the statistic S_{pn} (where p = 1 to n - 1) was computed and the estimate of state, z_{pn}, which produced the smallest S_{pn} was chosen as the best estimate of state at time n. The statistic was calculated by:

$$S_{pn} = G(z_{pn}) / [8(n - p + 1) - 10]$$

where the sum of squares, G_{pn}, was evaluated at each of the state estimates, z_{pn}.

2. Multiple Correlation Coefficient

The square of the multiple correlation coefficient, R^2, is a statistic that is frequently computed for multiple regression analysis (see Draper and Smith[4]). R^2 is a measure of how well the regression explains the data: R^2 is between 0 and 1, and the larger it is, the better the regression. R^2 is offered here as a candidate test statistic to determine the best of the $n - 1$ state estimates, z_{pn}. Suppose z_{pn} was obtained on Iteration i; then the test statistic is given by:

$$R^2_{pn} = (h^T V h - a) / (D^T M^{-1} D - a)$$

where

$$a = \left[\sum_k \sum_j D^j_k / \sqrt{M^j_k}\right]^2 / 8(n - p + 1)$$

for time-step indices $k = p$ to n and measurement indices $j = 1$ to 8. The state estimate that is associated with the <u>largest</u> value of R^2_{pn} (where $p = 1$ to $n - 1$) would then be chosen as the best estimate at time n.

Appendix A

MURLOC COMPUTER PROGRAM

```
      PROGRAM MURLOC(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
C
C   ** KEY
C   **  1  GEOGRAPHICAL POSITION DATA
C   **  2  RANGE AND BEARING DATA
C   **  3  BEARING-ONLY DATA
C   **  4  COURSE AND SPEED DATA
C   **  5  OUTPUT
C   **  6  NEW RUN
C   **  7  END
C   ** K      UNIT NUMBER
C   ** KO     OBSERVER NUMBER
C   ** TM     UPDATE TIME (MIN)
C   ** RNG    RANGE (NMI)
C   ** BRG    TRUE BEARING (DEG)
C   ** RSIG   RANGE STANDARD DEVIATION (NMI)
C   ** BSIG   BEARING STANDARD DEVIATION (NMI)
C   ** XE     EAST COORDINATE (NMI)
C   ** YE     NORTH COORDINATE (NMI)
C   ** R1     X-ROTATED PRINCIPLE STANDARD DEVIATION (NMI)
C   ** R2     Y-ROTATED PRINCIPLE STANDARD DEVIATION (NMI)
C   ** ANG    ANGLE OF ROTATION FROM NORTH (DEG)
C   ** CRS    COURSE (DEG)
C   ** SPD    SPEED (KT)
C   ** TO          LAST TIME OF UPDATE  (MIN)
C   ** E(I)        STATE VECTOR (X1,Y1,VX1,VY1, X2,Y2,VX2,VY2, ETC.)
C   ** V(I,J)      COVARIANCE MATRIX
C   ** H(I,J)      STATE-TO-DATA TRANSFORMATION MATRIX
C   ** D(N)        DIFFERENCE OF DATA AND PREDICTED DATA
C   ** W(I,J)      DATA COVARIANCE MATRIX
C
C
      COMMON/A/ KEY,TM,TO,KO,K,KM,IMAX,UB,TDO
      COMMON/B/ D(2),W(2,2),H(2,36)
      COMMON/C/ E(36),EE(36),V(36,36),VV(36,36),A(36,36)
      COMMON/E/ RNG,BRG,RSIG,BSIG,XE,YE
      DIMENSION X(5)
C
      PI=3.1415926536
      TPI=2.*PI
      UB=PI/180.
    5 WRITE(6,450)
      READ(5,620) TP,TD,KEY,KO,K,X
      TM=TP
      LL=1
      IF(KEY-6)110,20,150
C
C **  INITIALIZE
   20 CONTINUE
      TPS=TP $TDS=TD
      TO=TD $TDO=0.
      KM=KO
      IMAX=4*KM
      IM36=36*IMAX
      DO 22  I=1,IM36
      A(I)=0.
   22 V(I)=0.
      DO 24  I=1,IMAX
      E(I)=0.
      A(I,I)=1.
   24 V(I,I)=25.E4
      GO TO 5
```

```
C
C   ** GEOGRAPHICAL DATA
   30 CONTINUE
      IF(TM.GT.TO) CALL UPDATE(1)
      XE=X(1)  $YE=X(2)
      R1=X(3)  $R2=X(4)  $ANG=X(5)
      I2=K*4-2 $I1=I2-1
      D(1)=XE-E(I1)
      D(2)=YE-E(I2)
      IM2=2*IMAX
      DO 33 I=1,IM2
   33 H(I)=0.
      H(1,I1)=1. $H(2,I2)=1.
      B=ANG*UB
      CALL ROTATE(R1,R2,B,W(1,1),W(1,2),W(2,2))
      W(2,1)=W(1,2)
      CALL KALMAN(2,F)
      LL=2  $TM=TP   $GO TO 110
C
C   ** RANGE AND BEARING DATA
   40 CONTINUE
      RNG=X(1)  $RSIG=X(3)
      BRG=X(2)  $BSIG=X(4)
      CALL REPORT
      CALL KALMAN(2,F)
      LL=2  $TM=TP   $GO TO 110
C
C   ** BEARING ONLY DATA
   44 CONTINUE
      BRG=X(1)
      BSIG=X(2)
      CALL REPORT
      CALL KALMAN(2,F)
      LL=2  $TM=TP   $GO TO 110
C
C   ** VELOCITY DATA
   50 CONTINUE
      CRS=X(1)  $SPD=X(2)
      S1=X(3)  $S2=X(4)
      IF(TM.GT.TO) CALL UPDATE(1)
      B=CRS*UB
      I4=K*4 $I3=I4-1
      D(1)=SPD*SIN(B)-E(I3)
      D(2)=SPD*COS(B)-E(I4)
      IM2=2*IMAX
      DO 52 I=1,IM2
   52 H(I)=0.
      H(1,I3)=1. $H(2,I4)=1.
      CALL ROTATE(S1,S2,B,W(1,1),W(1,2),W(2,2))
      W(2,1)=W(1,2)
      CALL KALMAN(2,F)
      LL=2  $TM=TP   $GO TO 110
C
C **  OUTPUT
  110 CONTINUE
      KOZ=KO  $KZ=K
      IF(TP.EQ.TPS.AND.LL.EQ.1) GO TO 130
      WRITE(6,650)
      WRITE(6,530) TO
      WRITE(6,535) TM
      WRITE(6,700)
      CALL UPDATE(0)
      IF(KM.EQ.1) GO TO 120
```

```
C
C  ** OUTPUT UNIT LOCATION RELATIVE TO OBSERVER
      WRITE(6,540)
      KMM=KM-1
      DO 115 KO=1,KMM
      J2=KO*4-2 $J1=J2-1
      KK=KO+1
      DO 115 K=KK,KM
      I2=K*4-2 $I1=I2-1
      EX=EE(I1)-EE(J1)
      EY=EE(I2)-EE(J2)
      RNG=0. $BRG=0.
      IF(EX.EQ.0..AND.EY.EQ.0.)GO TO 112
      RNG=SQRT(EX*EX+EY*EY)
      BRG=ATAN2(EX,EY)
      IF(BRG.LT.0.)BRG=TPI+BRG
      BRG=BRG/UB
  112 V11=VV(I1,I1)-2.*VV(I1,J1)+VV(J1,J1)
      V22=VV(I2,I2)-2.*VV(I2,J2)+VV(J2,J2)
      V12=VV(I1,I2)-VV(I1,J2)-VV(J1,I2)+VV(J1,J2)
      CALL ELLIPSE(V11,V12,V22,R1,R2,AA)
      ANG=AA/UB
      CEP=FCEP(R1,R2)
  115 WRITE(6,550) KO,K,RNG,BRG,R1,R2,ANG,CEP
      WRITE(6,700)
C
C  ** OUTPUT UNIT STATE
  120 CONTINUE
      WRITE(6,545)
      DO 125 K=1,KM
      I4=K*4 $I3=I4-1 $I2=I3-1 $I1=I2-1
      V11=VV(I1,I1)
      V12=VV(I1,I2)
      V22=VV(I2,I2)
      CALL ELLIPSE(V11,V12,V22,R1,R2,AA)
      ANG=AA/UB
      CEP=FCEP(R1,R2)
      IF(EE(I3).EQ.0..AND.EE(I4).EQ.0.)GO TO 121
      C=ATAN2(EE(I3),EE(I4))
      IF(C.LT.0.)C=TPI+C
      C=C/UB
      GO TO 122
  121 C=0.
  122 S=SQRT(EE(I3)**2+EE(I4)**2)
      V33=VV(I3,I3)
      V34=VV(I3,I4)
      V44=VV(I4,I4)
      CALL ELLIPSE(V33,V34,V44,A1,A2,AA)
      AAA=AA/UB
      VEP=FCEP(A1,A2)
  125 WRITE(6,555) K,EE(I1),EE(I2),R1,R2,ANG,CEP,
     +              C,S,A1,A2,AAA,VEP
C
  130 CONTINUE
      IF(LL.EQ.2) GO TO 5
      KO=KOZ  $K=KZ
      TPS=TP  $TM=TD
      IF(TD.GT.TDS) TDO=(TD-TO)/60.
      TDS=TD
      WRITE(6,650)
      WRITE(6,590)
      WRITE(6,600)TP,TD,KEY,KO,K,X
      GO TO (30,40,44,50,5) KEY
```

```
C
  450 FORMAT(1H1)
  650 FORMAT(1H /1H )
  700 FORMAT(1H )
  590 FORMAT(*  DATA    TP *,
     +       *    TD  KEY KO K    X1     X2     X3     X4     X5 *)
  600 FORMAT(6X,2F6.1,I3,I4,I2, 5F6.1)
  620 FORMAT(2F10.0,I1,I4,I2,3X,5F10.0)
  530 FORMAT(*  TIME OF LAST DATA  =*,F6.1)
  535 FORMAT(*  TIME OF PREDICTION =*,F6.1)
  540 FORMAT(*  OBSR  UNIT   RNG   BRG   SIG1   SIG2   ANG    CEP*)
  550 FORMAT(I5,I6,F7.1,6F6.1)
  545 FORMAT(*         UNIT     X      Y    SIG1   SIG2   ANG    CEP*,
     +       9X,              *CRS    SPD   SIG1   SIG2   ANG    CEP*)
  555 FORMAT(I11,1X,6F6.1,6X,6F6.1)
C
  150 CONTINUE
      END
```

```
      SUBROUTINE UPDATE(L)
      COMMON/A/ KEY,TM,TO,KO,K,KM,IMAX,UB,TDO
      COMMON/C/ E(36),EE(36),V(36,36),VV(36,36),A(36,36)
      T=(TM-TO)/60.
      IMM3=IMAX-3
      DO 10 I=1,IMM3,4
      A(I,I+2)=T
   10 A(I+1,I+3)=T
      DO 20 I=1,IMAX
      EE(I)=0.
      DO 20 M=1,IMAX
   20 EE(I)=EE(I)+A(I,M)*E(M)
      DO 30 I=1,IMAX
      DO 30 J=1,IMAX
      VV(I,J)=0.
      DO 30 M=1,IMAX
      DO 30 N=1,IMAX
      VV(I,J)=VV(I,J)+A(I,M)*V(M,N)*A(J,N)
   30 VV(J,I)=VV(I,J)
      IF(L.EQ.0)RETURN
      DO 70 I=1,IMAX
      E(I)=EE(I)
      DO 70 J=1,IMAX
   70 V(I,J)=VV(I,J)
      TO=TM
      RETURN
      END
```

```
      SUBROUTINE REPORT
      COMMON/A/ KEY,TM,TO,KO,K,KM,IMAX,UB,TDO
      COMMON/B/ D(2),W(2,2),H(2,36)
      COMMON/C/ E(36),EE(36),V(36,36),VV(36,36),A(36,36)
      COMMON/E/ RNG,BRG,RSIG,BSIG,XE,YE
      IF(TM.GT.TO) CALL UPDATE(1)
      I2=K*4-2 $I1=I2-1
      J2=KO*4-2 $J1=J2-1
      B=BRG*UB
      IF(KEY.EQ.2) GO TO 15
      XD=E(I1)-E(J1)
      YD=E(I2)-E(J2)
      V11= V(I1,I1)-2.* V(I1,J1)+ V(J1,J1)
      V22= V(I2,I2)-2.* V(I2,J2)+ V(J2,J2)
      V12= V(I1,I2)- V(I1,J2)- V(J1,I2) +V(J1,J2)
      CALL ELLIPSE(V11,V12,V22,R1,R2,AA)
      CAA=COS(AA)
      SAA=SIN(AA)
      XC=XD*CAA-YD*SAA
      YC=YD*CAA+XD*SAA
      BA=B-AA
      SBA=SIN(BA)
      CBA=COS(BA)
      IF(ABS(SBA).LT..0001) GO TO 10
      IF(ABS(CBA).LT..0001) GO TO 11
      TBA=SBA/CBA
      R=(R1/R2)  $RR=R*R
      Y=(RR*YC+TBA*XC)/(TBA*TBA+RR)
      X=TBA*Y
      RNG=SQRT(X*X+Y*Y)  $GO TO 12
   10 RNG=ABS(YC)  $GO TO 12
   11 RNG=ABS(XC)
   12 CONTINUE
      RSIG=RNG
   15 CONTINUE
      R1=RNG*BSIG*UB
      R2=RSIG
      XE=E(J1)+RNG*SIN(B)
      YE=E(J2)+RNG*COS(B)
      D(1)=XE-E(I1)
      D(2)=YE-E(I2)
      IM2=2*IMAX
      DO 25 I=1,IM2
   25 H(I)=0.
      H(1,I1)=1. $H(2,I2)=1.
      H(1,J1)=-1.  $H(2,J2)=-1.
      CALL ROTATE(R1,R2,B,W(1,1),W(1,2),W(2,2))
      W(2,1)=W(1,2)
      RETURN
      END
```

```
      SUBROUTINE KALMAN(KA,F)
C
C  ** FILTER EQUATIONS, JAZWINSKI P.270
C  ** G = V*H * (H*V*H + W)**-1
C  ** E = E + G*D
C  ** V = (I-G*H)*V*(I-G*H) + G*W*G
C
      COMMON/A/ KEY,TM,TO,KO,K,KM,IMAX,UB,TDO
      COMMON/B/ D(2),W(2,2),H(2,36)
      COMMON/C/ E(36),EE(36),V(36,36),VV(36,36),A(36,36)
      DIMENSION VH(36,2),G(36,2),P(36,36),VS(36,36),U(2,2)
C
      I4=K*4 $I3=I4-1 $I2=I3-1 $I1=I2-1
      L=0
C
    1 DO 10 I=1,IMAX
      DO 10 J=1,2
      SUM=0.
      DO 5 N=1,IMAX
    5 SUM=SUM+V(I,N)*H(J,N)
   10 VH(I,J)=SUM
C
      DO 20 I=1,2
      DO 20 J=1,2
      SUM=W(I,J)
      DO 15 N=1,IMAX
   15 SUM=SUM+H(I,N)*VH(N,J)
   20 U(I,J)=SUM
C
      DET=U(1,1)*U(2,2)-U(1,2)*U(2,1)
      U11=U(1,1)
      U(1,1)=U(2,2)/DET
      U(1,2)=-U(1,2)/DET
      U(2,1)=U(1,2)
      U(2,2)=U11/DET
      IF(L.EQ.1.OR.TDO.EQ.0.) GO TO 24
C
      R=0.
      DO 21 M=1,2
      DO 21 N=1,2
   21 R=R+D(M)*U(M,N)*D(N)
      F=EXP(-R/2.)  $FF=1.-F
      IF(KA.EQ.1) RETURN
C
      GO TO (22,22,22,23) KEY
   22 V(I1,I1)=V(I1,I1) + D(1)*D(1)*FF
      V(I2,I2)=V(I2,I2) + D(2)*D(2)*FF
      V(I1,I2)=V(I1,I2) + D(1)*D(2)*FF
      V(I2,I1)=V(I1,I2)
      D3=D(1)/TDO
      D4=D(2)/TDO
      V(I3,I3)=V(I3,I3) + D3*D3*FF
      V(I4,I4)=V(I4,I4) + D4*D4*FF
      V(I3,I4)=V(I3,I4) + D3*D4*FF
      V(I4,I3)=V(I3,I4)
      V(I1,I3)=V(I1,I3) + D(1)*D3*FF
      V(I1,I4)=V(I1,I4) + D(1)*D4*FF
      V(I2,I3)=V(I2,I3) + D(2)*D3*FF
      V(I2,I4)=V(I2,I4) + D(2)*D4*FF
      V(I3,I1)=V(I1,I3)
      V(I4,I1)=V(I1,I4)
      V(I3,I2)=V(I2,I3)
      V(I4,I2)=V(I2,I4)
      L=1 $GO TO 1
```

```
   23 V(13,13)=V(13,13) + D(1)*D(1)*FF
      V(14,14)=V(14,14) + D(2)*D(2)*FF
      V(13,14)=V(13,14) + D(1)*D(2)*FF
      V(14,13)=V(13,14)
      L=1 $GO TO 1
C
   24 DO 30 I=1,IMAX
      DO 30 J=1,2
      SUM=0.
      DO 25 N=1,2
   25 SUM=SUM+VH(I,N)*U(N,J)
   30 G(I,J)=SUM
C
      DO 40 I=1,IMAX
      DO 40 J=1,IMAX
      SUM=0.
      IF(I.EQ.J)SUM=1.
      DO 35 N=1,2
   35 SUM=SUM-G(I,N)*H(N,J)
   40 P(I,J)=SUM
C
      DO 55 I=1,IMAX
      DO 55 J=I,IMAX
      SUM=0.
      DO 45 N=1,IMAX
      DO 45 M=1,IMAX
   45 SUM=SUM+P(I,N)*V(N,M)*P(J,M)
      DO 50 N=1,2
      DO 50 M=1,2
   50 SUM=SUM+G(I,N)*W(N,M)*G(J,M)
   55 VS(I,J)=VS(J,I)=SUM
C
      DO 65 I=1,IMAX
      DO 60 N=1,2
   60 E(I)=E(I)+G(I,N)*D(N)
      DO 65 J=I,IMAX
   65 V(I,J)=V(J,I)=VS(I,J)
      RETURN
      END
```

```
      SUBROUTINE ELLIPSE(V11,V12,V22,R1,R2,A)
      IF(V12.EQ.0.) GO TO 10
      A=.5*ATAN2(2.*V12,V22-V11)
      SINA=SIN(A)
      COSA=COS(A)
      SSIN=SINA*SINA
      SCOS=SINA*COSA
      CCOS=COSA*COSA
      RR1=CCOS*V11-2.*SCOS*V12+SSIN*V22
      RR2=SSIN*V11+2.*SCOS*V12+CCOS*V22
      R1=SQRT(RR1)
      R2=SQRT(RR2)
      RETURN
   10 R1=SQRT(V11)
      R2=SQRT(V22)
      A=0.
      IF(R2.GE.R1) RETURN
      R=R1   $R1=R2   $R2=R   $A=3.14159265359/2.
      RETURN
      END
```

```
      SUBROUTINE ROTATE(R1,R2,A,V11,V12,V22)
      SINA=SIN(A)
      COSA=COS(A)
      SSIN=SINA*SINA
      SCOS=SINA*COSA
      CCOS=COSA*COSA
      RR1=R1*R1
      RR2=R2*R2
      V11=CCOS*RR1+SSIN*RR2
      V12=SCOS*(RR2-RR1)
      V22=SSIN*RR1+CCOS*RR2
      RETURN
      END
```

```
      FUNCTION FCEP(R1,R2)
      CEP=.562*R2+.615*R1
      IF(R1.LT..3*R2) CEP=R2*(.675+.79444*(R1/R2)**2)
      FCEP=CEP
      RETURN
      END
```

TIME OF LAST DATA = 25.0
TIME OF PREDICTION = 40.0

OBSR	UNIT	RNG	BRG	SIG1	SIG2	ANG	CEP
1	2	60.9	172.5	1.2	2.0	89.5	1.9
1	3	43.0	123.9	1.0	1.7	32.8	1.6
1	4	56.1	98.5	2.0	6.2	87.2	4.7
2	3	45.8	37.3	1.8	2.3	76.2	2.4
2	4	70.5	42.4	2.2	6.0	84.0	4.7
3	4	25.2	51.5	2.5	6.4	85.6	5.1

UNIT	X	Y	SIG1	SIG2	ANG	CEP	CRS	SPD	SIG1	SIG2	ANG	CEP
1	1.9	70.3	2.0	3.0	44.2	2.9	24.9	18.0	.5	.8	34.8	.7
2	9.9	9.9	2.5	3.5	58.6	3.5	.1	14.9	1.0	2.0	.1	1.7
3	37.6	46.3	2.3	3.5	41.0	3.3	110.0	60.0	2.2	3.0	31.2	3.1
4	57.4	62.0	3.2	6.8	83.1	5.8	235.5	20.4	4.7	14.3	86.3	10.9

DATA	TP	TD	KEY	KO	K	X1	X2	X3	X4	X5
	40.0	30.0	2	2	3	47.6	16.5	.5	2.0	-0.0

TIME OF LAST DATA = 30.0
TIME OF PREDICTION = 40.0

OBSR	UNIT	RNG	BRG	SIG1	SIG2	ANG	CEP
1	2	61.1	172.2	1.1	1.6	-75.4	1.6
1	3	27.3	123.0	1.3	5.7	-61.0	4.1
1	4	56.4	98.5	2.0	6.2	87.0	4.7
2	3	48.0	17.7	.7	4.9	-61.3	3.4
2	4	70.6	42.4	2.2	6.0	84.3	4.7
3	4	33.6	78.8	3.3	7.3	-79.9	6.1

UNIT	X	Y	SIG1	SIG2	ANG	CEP	CRS	SPD	SIG1	SIG2	ANG	CEP
1	1.9	70.3	2.0	3.0	44.2	2.9	24.9	18.0	.5	.8	35.1	.7
2	10.2	9.8	2.5	3.3	53.5	3.4	.2	14.7	1.0	1.9	-1.2	1.7
3	24.8	55.5	3.2	6.0	-64.4	5.4	17.2	16.4	2.3	22.8	-57.8	15.6
4	57.7	62.0	3.2	6.7	82.7	5.7	235.1	20.4	4.7	14.3	86.4	10.9

```
DATA   TP    TD  KEY KØ K   X1    X2    X3    X4    X5
      40.0  30.0  3   2 4  41.2   1.0  -0.0  -0.0  -0.0

TIME ØF LAST DATA  =  30.0
TIME ØF PREDICTIØN =  40.0

ØBSR  UNIT   RNG   BRG   SIG1  SIG2   ANG   CEP
  1     2   61.1 172.2   1.1   1.6 -75.4   1.6
  1     3   27.3 123.0   1.3   5.7 -61.0   4.1
  1     4   55.4  98.6   1.8   3.7  74.8   3.2
  2     3   48.0  17.7    .7   4.9 -61.3   3.4
  2     4   69.9  41.7   1.7   3.5  61.8   3.0
  3     4   32.6  78.4   3.1   5.4 -69.3   4.9

      UNIT    X     Y    SIG1  SIG2   ANG   CEP        CRS   SPD  SIG1  SIG2   ANG   CEP
        1    1.9  70.3   2.0   3.0  44.2   2.9       24.9  18.0   .5    .8   35.1   .7
        2   10.2   9.8   2.5   3.3  53.5   3.4        .2   14.7  1.0   1.9   -1.2  1.7
        3   24.8  55.5   3.2   6.0 -64.4   5.4       17.3  16.4  2.3   22.8 -57.8  15.6
        1   56.7  62.0   2.8   4.7  65.0   4.4      238.8  22.4  4.2   8.8   73.6  7.5

DATA   TP    TD  KEY KØ K   X1    X2    X3    X4    X5
      40.0  35.0  3   1 4  96.4   1.0  -0.0  -0.0  -0.0

TIME ØF LAST DATA  =  35.0
TIME ØF PREDICTIØN =  40.0

ØBSR  UNIT   RNG   BRG   SIG1  SIG2   ANG   CEP
  1     2   61.1 172.2   1.1   1.6 -75.4   1.6
  1     3   27.3 123.0   1.3   5.7 -61.0   4.1
  1     4   55.1  99.0   1.0   3.2 -88.9   2.4
  2     3   48.0  17.7    .7   4.9 -61.3   3.4
  2     4   69.5  41.7   1.4   2.8  78.6   2.4
  3     4   32.2  78.9   2.2   5.3 -64.6   4.3

      UNIT    X     Y    SIG1  SIG2   ANG   CEP        CRS   SPD  SIG1  SIG2   ANG   CEP
        1    1.9  70.3   2.0   3.0  44.2   2.9       24.9  18.0   .5    .8   35.1   .7
        2   10.1   9.8   2.5   3.3  53.5   3.4        .2   14.7  1.0   1.9   -1.2  1.7
        3   24.8  55.5   3.2   6.0 -64.4   5.4       17.0  16.4  2.3   22.8 -57.8  15.6
        4   56.3  61.7   2.7   4.2  75.5   4.0      238.1  23.3  3.0   7.8   86.6  6.2
```

TIME OF LAST DATA = 35.0
TIME OF PREDICTION = 50.0

OBSR	UNIT	RNG	BRG	SIG1	SIG2	ANG	CEP
1	2	61.2	173.4	1.4	1.7	-63.6	1.8
1	3	26.9	123.8	1.7	9.5	-59.6	6.6
1	4	51.7	105.1	1.5	4.3	-89.5	3.3
2	3	48.4	18.6	1.0	8.5	-60.1	5.9
2	4	63.9	42.2	1.9	4.0	81.0	3.4
3	4	27.5	86.8	3.0	9.1	-62.2	7.0

UNIT	X	Y	SIG1	SIG2	ANG	CEP	CRS	SPD	SIG1	SIG2	ANG	CEP
1	3.2	73.1	2.0	3.0	43.8	3.0	24.9	18.0	.5	.8	35.1	.7
2	10.2	12.2	2.6	3.3	50.4	3.5	.2	14.7	1.0	1.9	-1.2	1.7
3	25.6	58.1	3.4	9.7	-60.7	7.5	17.0	16.4	2.3	22.8	-57.8	15.6
4	53.0	59.6	2.9	5.0	80.9	4.6	238.1	23.3	3.0	7.8	86.6	6.2

DATA	TP	TD	KEY	KØ	K	X1	X2	X3	X4	X5
	50.0	40.0	3	2	4	41.4	1.0	-0.0	-0.0	-0.0

TIME OF LAST DATA = 40.0
TIME OF PREDICTION = 50.0

OBSR	UNIT	RNG	BRG	SIG1	SIG2	ANG	CEP
1	2	61.2	173.4	1.4	1.7	-63.6	1.8
1	3	26.9	123.8	1.7	9.5	-59.6	6.6
1	4	51.2	105.1	1.4	3.0	84.0	2.6
2	3	48.4	18.6	1.0	8.5	-60.1	5.9
2	4	63.7	41.8	1.4	2.9	57.9	2.4
3	4	27.1	86.6	2.8	8.6	-59.4	6.6

UNIT	X	Y	SIG1	SIG2	ANG	CEP	CRS	SPD	SIG1	SIG2	ANG	CEP
1	3.2	73.1	2.0	3.0	43.8	3.0	24.9	18.0	.5	.8	35.1	.7
2	10.2	12.2	2.6	3.3	50.4	3.5	.2	14.7	1.0	1.9	-1.2	1.7
3	25.6	58.1	3.4	9.7	-60.7	7.5	17.0	16.4	2.3	22.8	-57.8	15.6
4	52.6	59.7	2.7	4.1	66.7	4.0	239.3	24.0	2.8	5.5	74.9	4.8

DATA	TP	TD	KEY	KØ	K	X1	X2	X3	X4	X5
	50.0	45.0	3	1	4	100.1	1.0	-0.0	-0.0	-0.0

TIME OF LAST DATA = 45.0
TIME OF PREDICTION = 50.0

OBSR	UNIT	RNG	BRG	SIG1	SIG2	ANG	CEP
1	2	61.2	173.4	1.4	1.7	-63.7	1.8
1	3	27.0	123.8	1.7	9.5	-59.6	6.6
1	4	51.8	102.8	1.4	2.8	-78.8	2.5
2	3	48.4	18.6	1.0	8.5	-60.1	5.9
2	4	65.8	41.4	1.8	2.3	63.4	2.4
3	4	28.3	82.7	2.4	8.7	-58.3	6.4

UNIT	X	Y	SIG1	SIG2	ANG	CEP	CRS	SPD	SIG1	SIG2	ANG	CEP
1	3.2	73.1	2.0	3.0	43.8	3.0	24.9	18.0	.5	.8	35.1	.7
2	10.2	12.2	2.6	3.3	50.4	3.4	.3	14.6	1.0	1.9	-1.2	1.7
3	25.6	58.0	3.4	9.7	-60.7	7.5	17.4	16.3	2.3	22.8	-57.8	15.6
4	53.7	61.6	2.9	3.8	73.5	3.9	257.3	18.3	4.9	10.2	6.0	8.7

REFERENCES

1. A. H. Jazwinski, Stochastic Processes and Filtering Theory, p. 311 (Academic Press, New York, NY, 1970).

2. Y. Bard, "Comparison of Gradient Methods for the Solution of Nonlinear Parameter Estimation Problems," SIAM J. Numer. Anal., 7, pp. 157-186 (1970).

3. C. G. Broyden, "Quasi-Newton Methods," in W. Murray (ed), Numerical Methods for Unconstrained Optimization, pp. 87-106 (Academic Press, New York, NY, 1972).

4. N. R. Draper and H. Smith, Applied Regression Analysis, pp. 263-304 (John Wiley & Sons, Inc., New York, NY, 1966).

5. R. Fletcher, "A Survey of Algorithms for Unconstrained Optimization," in W. Murray (ed), Numerical Methods for Unconstrained Optimization, pp. 123-129 (Academic Press, New York, NY 1972).

6. M.J.D. Powell, "Problems Related to Unconstrained Optimization," in W. Murray (ed), Numerical Methods for Unconstrained Optimization, pp. 29-55 (Academic Press, New York, NY, 1972).

7. C. L. Lawson and R. J. Hanson, Solving Least Squares Problems (Prentice-Hall, Inc., Englewood Cliffs, NJ, 1974).

8. G. Forsythe and C. B. Moler, Computer Solution of Linear Algebraic Systems (Prentice-Hall, Inc., Englewood Cliffs, NJ, 1967).

DISTRIBUTION LIST

Name	Number of Copies
Advanced Research Projects Agency Department of Defense Washington, DC 20301	
Technical Library	1
Defense Documentation Center Cameron Station Alexandria, VA 22314	12
Commander - Submarine Development Squadron 12 Box 70, Naval Submarine Base New London, CT 06342	1
Under Secretary, Defense Research and Engineering and Systems Washington, DC 20301	
Dr. McKinney	1
Chief of Naval Operations Department of the Navy Washington, DC 220350	
OP-951	1
OP-953	1
OP-961	1
Chief of Naval Material Department of the Navy Washington, DC 20360	
Mr. Flum	1
Office of Naval Research Department of the Navy Arlington, VA 22217	
Code 431	2
Code 222	1
Code 436	1
Naval Sea Systems Command Department of the Navy Washington, DC 20360	
Code 06H2-3	1
Code 06H1	1

Name	Number of Copies
Commander, Second Fleet Norfolk, VA 23511 (Code N4)	2
Naval Intelligence Support Center 4301 Suitland Road Washington, DC 20390	1
Center for Naval Analyses 1401 Wilson Boulevard Arlington, VA 22209 Library	1
Lockheed Missiles and Space Co. Building 102 - ORG 61-82 1111 Lockheed Way Sunnyvale, CA 94086 Mr. Frye	1
Naval Electronics Systems Command PME 108-2 Department of the Navy Washington, DC 20360	1
Naval Surface Weapons Center Code U-202 Silver Spring, MD 20910	1
Naval Ocean Systems Center San Diego, CA 92152	
Code 6212	1
Code 16	1
David Taylor Naval Research and Development Center Bethesda, MD 20034 Code 1806	1
Commander - Third Fleet FPO San Francisco, CA 96610 N7 Division	2
Commander, Cruiser Destroyer Group 2 Fleet Post Office New York, NY 09501	1
Naval Underwater Systems Center New London Laboratory New London, CT 06329	2

Name	Number of Copies
Naval Research Laboratory Washington, DC 20375	
Code 2620	2
Code 8109	1
Naval War College Newport, RI 02840	
Technical Library	1
Summit Research Corporation 1 West Deer Park Road Gaithersburg, MD 20760	1
Analysis and Technology, Inc. Technology Park Box 220 North Stonington, CT 06359	1