(2)

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A MICROCOMPUTER SIMULATION PROGRAM TO MODEL TRANSIENT AND STEADY-STATE DETECTION OF AN EVADING SUBMARINE BY A SEARCHING SUBMARINE IN A FALSE TRANSIENT ENVIRONMENT

by

J. Brad Kratovil

September 1988

Thesis Co-Advisors: R. N. Forrest
James N. Eagle

SECURITY CLASSIFICATION OF THIS PAGE

ADA201862

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: Naval Postgraduate School; 6b OFFICE SYMBOL (If applicable): 55	7a NAME OF MONITORING ORGANIZATION: Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code): Monterey, California 93943-5000	7b ADDRESS (City, State, and ZIP Code): Monterey, California 93943-5000
8a. NAME OF FUNDING / SPONSORING ORGANIZATION; 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO; PROJECT NO; TASK NO; WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification): A Microcomputer Simulation Program to Model Transient and Steady-State Detection of an Evading Submarine by a Searching Submarine in a Falso Transient Environment.

12 PERSONAL AUTHOR(S): Kratovil, J. Brad

13a. TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	FROM _____ TO _____	1988 September	128

16 SUPPLEMENTARY NOTATION: The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Submarine, Transient, Submarine Simulation, Submarine Transient, Submarine Detection.

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

This thesis describes the development and demonstrates the use of SUBTRAN, a submarine transient detection computer simulation model. SUBTRAN provides, at the microcomputer level, a framework that can be used to investigate how transient detection and false transient detection opportunities affect the expected time to steady-state (continuous) detection. Monte Carlo methods are employed to simulate a submarine versus submarine passive acoustic detection search scenario. The scenario terminates when steady-state detection occurs. Detection is modeled using a signal excess threshold crossing model. Random fluctuations in the acoustic signal excess are modeled using a Lambda-Sigma Jump process. Both submarines are assumed to have a fixed speed and are constrained within a defined search area. The "target" submarine is assumed to be unable to counter-detect the "searching" submarine. Transient signals and false transient signals are determined

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL: R. N. Forrest	22b TELEPHONE (Include Area Code): 408-646- 2653; 22c OFFICE SYMBOL: 55Fo

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

U.S. Government Printing Office 1988—606-24.

19. ABSTRACT (cont)

by independent Poisson processes. Summary statistics for the times to detection are provided as output of the simulation.

Accession For	
NTIS GRA&I	☑
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

Approved for public release; distribution is unlimited

A Microcomputer Simulation Program to Model Transient and Steady-State Detection of an Evading Submarine by a Searching Submarine in a False Transient Environment

by

J. Brad Kratovil
Lieutenant, United States Navy
B.S., University of Southern California, 1982

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September 1988

Author: J. Brad Kratovil
J. Brad Kratovil

Approved By:
R. N. Forrest, Thesis Co-Advisor

James N. Eagle, Thesis Co-Advisor

Peter Purdue, Chairman,
Department of Operations Research

Kneale T. Marshall
Dean of Information and Policy Sciences

ABSTRACT

This thesis describes the development and demonstrates the use of SUBTRAN, a submarine transient detection computer simulation model. SUBTRAN provides, at the microcomputer level, a framework that can be used to investigate how transient detection and false transient detection opportunities affect the expected time to steady-state (continuous) detection. Monte Carlo methods are employed to simulate a submarine versus submarine passive acoustic detection search scenario. The scenario terminates when steady-state detection occurs. Detection is modeled using a signal excess threshold crossing model. Random fluctuations in the acoustic signal excess are modeled using a Lambda-Sigma Jump process. Both submarines are assumed to have a fixed speed and are constrained within a defined search area. The "target" submarine is assumed to be unable to counter-detect the "searching" submarine. Transient signals and false transient signals are determined by independent Poisson processes. Summary statistics for the times to detection are provided as output of the simulation.

Keywords: SUBTRAN computer program, validation, passive detection. (KP)

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

Passive detection of a submarine can be divided into two mutually exclusive events: steady-state detection and transient detection. Steady-state detection involves the passive acoustic monitoring of a submarine's continuous operating sounds. With today's submarine quieting capabilities, steady-state detection typically requires short ranges. On the other hand, transient detection is the detection of a submarine's intermittent sounds that are produced during infrequent operations. Due to the high signal levels generated by many of these infrequent operations, transient detection can occur at very long ranges.

Of great concern to an evading submarine is the ability to freely move about without its actions being monitored by a searching submarine. This monitoring is generally based on an evading submarine's continuous operating sounds.

Due to the typical short ranges involved, the chance of a searching submarine being able to monitor an evading submarine's continuously generated signals is quite small when a large search area is involved. However, an evading submarine's transient signals offer long-range detection opportunities. The exploitation of these opportunities may improve the ability of the searching submarine to gain steady-state detection.

A drawback to the use of long range detection opportunities occurs in a false transient environment. In any search scenario, false detections can lead the searcher away from the actual target position.

The purpose of this thesis is to develop a means with which to address the issue of whether or not transient detection opportunities can be utilized by a searching submarine in a false transient environment to increase its chance of

gaining steady-state detection on a target submarine. This will be accomplished by use of a Monte Carlo simulation, SUBTRAN.

II. BACKGROUND

A. GENERAL DESCRIPTION OF SIMULATION MODEL

SUBTRAN is a discrete event simulation of two submarine platforms in a area search scenario. One platform is designated the "target" and one the "searcher". During each discrete event time step, the time to the next discrete event is computed, steady-state detection is assessed, target and searcher positions are updated and the simulation time is advanced. A simulation run is completed when steady-state detection occurs or the user-defined maximum search time is reached. Propagation loss curves and figure of merits are used to derive the deterministic element of signal excess. A Lambda-Sigma Jump process is used to describe stochastic variations in signal excess, and a threshold crossing model is used to evaluate total signal excess for the occurrence of detection. Target characteristics incorporated in the simulation model include: target's course, target's speed, and target's transient occurrence rate. Searcher characteristics incorporated in the simulation model include the searcher's course and speed. Target depth, relative to the searcher, is assumed constant. False transients are modeled in the same fashion as target transients by using a transient occurrence rate.

B. SEARCH SCENARIO

The search scenario modeled in SUBTRAN is a parallel sweep search of a square area whose boundaries are known by both the target and searcher. At the beginning of each simulation replication, the target is positioned such that its X component of position and Y component of position are independent

uniform random variables on the interval (0,L). L is the length dimension of the search area in nautical miles. The searcher's position, at the beginning of each replication, is in the lower left-hand corner of the search area. In order to maximize steady-state search effort, the searcher is positioned a distance equal to one-half a track width from both the bottom and left-hand search area boundaries. The search area is oriented such that the top area boundary is to the north for course reference. A typical search area, annotated with initial searcher and target positions, is displayed in Figure 2.1.

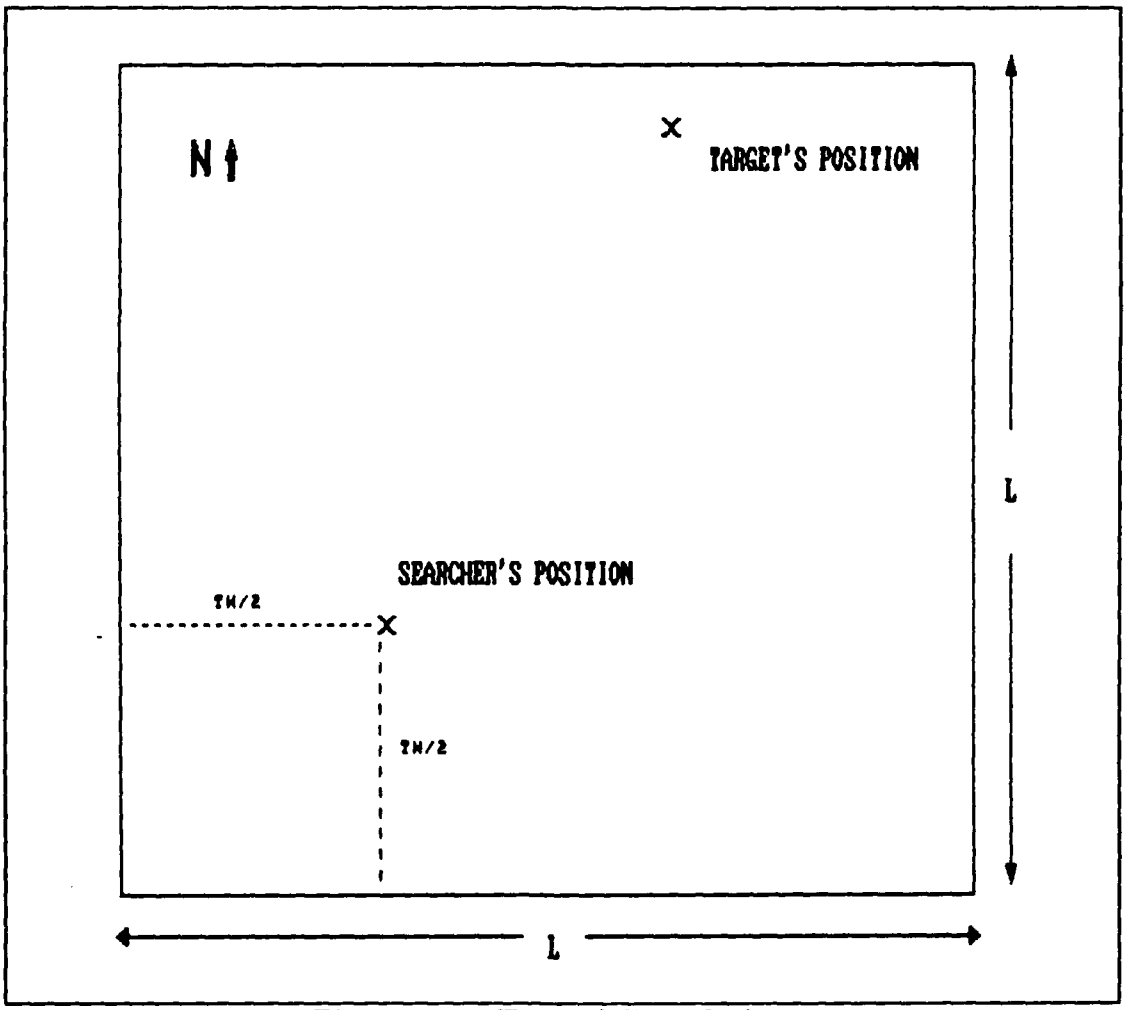

Figure 2.1. Typical Search Area.

C. DETECTION MODEL

The basic detection model used in SUBTRAN converts user-defined propagation loss curves, figure-of-merits (FOM), and Lambda-Sigma Jump process parameters into an assessment of detection occurrence by evaluating total signal excess using a threshold crossing detection model. Two types of detection are modeled: steady-state detection and transient detection. Due to the current trend in submarine quieting, convergence zone detection of the target's steady-state sound signature is assumed not to be possible. Convergence zone detection of the target's transient noise signature is assumed to be possible but only to the third convergence zone.

1. Propagation Loss

The propagation loss experienced by sound as it travels through the water is determined by many factors. Frequency of the sound, receiver depth, source depth, ocean bottom type, and sound velocity profile all have an effect on the intensity of the sound received from an underwater source. In the area search scenario modeled in SUBTRAN, most of these factors remain relatively static with the exception of the frequency of the sound. Since steady-state and transient sounds are produced differently, they are likely to have different frequency components. As a result, two propagation loss curves are modeled in SUBTRAN, a steady-state propagation loss curve and a transient propagation loss curve.

SUBTRAN allows the user to input up to 50 values of range and corresponding propagation loss for each of the two types of sound. From this input, the model constructs propagation loss curves using straight line approximation between successive points. One curve is constructed for steady-state signal loss and one is constructed for transient signal loss. Since

convergence zone detection of the target's steady-state sound signature is assumed not to be possible, only the direct path range on the steady-state propagation loss curve is used.

2. Figure-of-Merit

FOM, defined for passive sonar as the maximum propagation loss which still results in a 50 percent probability of signal detection [Ref. 1:p. 49], is modeled in SUBTRAN under the following assumptions and conditions:

- The target cannot counter-detect the searcher.
- The speeds of the target and searcher are held constant.
- There is only one type of steady-state sound source emitted by the target and it is omni-directional.
- There is only one type of transient sound source emitted by the target and it is omni-directional.
- The ambient noise level is constant throughout the simulation.

Under these assumptions and conditions, target FOM can be summarized by two constant values: steady-state FOM and transient FOM. SUBTRAN requires the user to determine the values of the steady-state and transient FOM for the target based on the search scenario being simulated.

3. Signal Excess

Signal excess (SE) is modeled in SUBTRAN as consisting of a deterministic part and a stochastic part. The deterministic part of SE is defined as the algebraic difference between FOM and propagation loss. The stochastic part of signal excess is used to describe the random fluctuations that occur in signal excess with respect to time. A Lambda-Sigma Jump process is used to model these random fluctuations.

4. Lambda-Sigma Jump Process

A Lambda-Sigma Jump process [Ref. 2:p. 8] is used to describe the random fluctuations in signal excess. This process models the time between fluctuations as well as the magnitude of the fluctuations. For the Lambda-Signal Jump process, the time between fluctuations is exponentially distributed with rate parameter lambda and the magnitude of the fluctuations is normally distributed with mean zero and standard deviation sigma.

The user inputs values for the parameters lambda and sigma. Appropriate values for these parameters are subjective and must be based on the experience of the user. A guide in the estimation of lambda and sigma, based on various sources, is [Ref. 3 and Ref. 4]:

- Lambda varies between 2 and 60 $hours^{-1}$.
- Sigma varies between 3 and 9 dB.

Other sources containing information helpful in determining appropriate values for lambda and sigma are listed by Tehan [Ref. 4].

5. Threshold Crossing Model

A threshold crossing model [Ref. 2:p. 7] is the final step in the overall effort to model detection. This model predicts that detection occurs when signal excess is equal to or exceeds some given threshold level. The most common threshold level, and the one employed in SUBTRAN, is zero. In other words, detection occurs whenever signal excess is greater than or equal to zero.

6. Detection Model Employment

The detection model outlined in the previous sections is implemented in SUBTRAN by extracting the current detection range from the propagation loss curve and comparing it to the range between the target and the searcher. If the

detection range is greater than or equal to the range between the target and the searcher, detection occurs.

The current detection range is determined by considering an adjusted value of the FOM. This adjusted value is obtained by adding the user-defined value of FOM and the current signal excess fluctuation value. The range at which this adjusted FOM value and the propagation loss curve are equal is the current detection range.

7. Steady-State Detection Assessment

SUBTRAN is a discrete event simulation. A discrete event is defined as any event whose occurrence could change the ability of the searcher to gain steady-state detection on the target. The discrete events modeled in SUBTRAN are: target course change, target boundary reflection, searcher course change, target transient signal occurrence, false transient signal occurrence, and signal excess fluctuation occurrence.

Since SUBTRAN is a discrete event simulation, steady-state detection must be assessed for the time period until the next discrete event. To do this, the current values of searcher and target course and position are used together with the user-defined speed values to define the positions of the searcher and target at any future time. These positions are then compared to find the time it would take the range between the searcher and target to decrease to a value less than or equal to the current steady-state detection range. If the time calculated is less than or equal to the time to the next discrete event, steady-state detection occurs. Appendix E contains the steady-state detection assessment calculations.

8. Transient Detection Assessment

Detection of target transients by the searcher is assessed in a different manner than steady-state detection. The target's transient sound signature is intermittent and when produced, is assumed to occur instantaneously. As a result of these characteristics, transient detection is assessed in SUBTRAN by comparing searcher and target positions at the time of a transient occurrence. If the range is within the transient detection range, transient detection occurs.

Transient detection range is extracted from the transient propagation loss curve using the transient FOM. Since transient detection can occur out to the third convergence zone, the transient detection range consists of four range brackets: a direct path range bracket and three convergence zone annulus range brackets.

D. TARGET CHARACTERISTICS

The target characteristics modeled in SUBTRAN include: speed, course, and transient emission rate. Target depth relative to the searcher is assumed to be constant. In order to model its worst case vulnerability to detection, the target does not have the ability to detect and thus avoid the searcher.

1. Target Speed

Target speed is held constant throughout the simulation at a value specified by the user.

2. Target Course

The search scenario modeled in SUBTRAN involves a randomly moving target which is constrained to a defined search area by boundary reflection. To incorporate differing target course and time to course change distributions,

SUBTRAN uses the inverse probability integral transformation [Ref. 5:pp. 20-21a] to model target movement.

The inverse probability integral transformation method involves selecting values of a desired random variable whose cumulative distribution function (CDF) is known by using random variables drawn from a uniform (0,1) distribution. This method is illustrated in Figure 2.2. A random variable drawn from a uniform (0,1) distribution is entered on the Y-axis of the CDF representing the random variable desired. The corresponding X-axis value is the associated random variable of the given distribution.

SUBTRAN allows the user to input up to 50 values for both the target's course CDF and the time to course change CDF. The corresponding CDF curves are constructed using straight-line approximation between successive points.

Due to the randomness of the target's movement, a procedure exists to constrain the target to the search area. Using its current course and speed, the position of the target is evaluated for the period until its next course change. If during this time the target encounters the search area boundary, its course is reflected back into the search area. The angle of reflection is equal to the angle of incidence between the search area boundary and the target's incoming course. An example of target boundary reflection is shown in Figure 2.3.

3. Target Transients

Target transient occurrence is modeled in SUBTRAN as a Poisson process with a user-defined rate parameter. Only one type of transient sound is assumed to be produced by the target. In addition, when the transient does occur, it is assumed to be instantaneous.

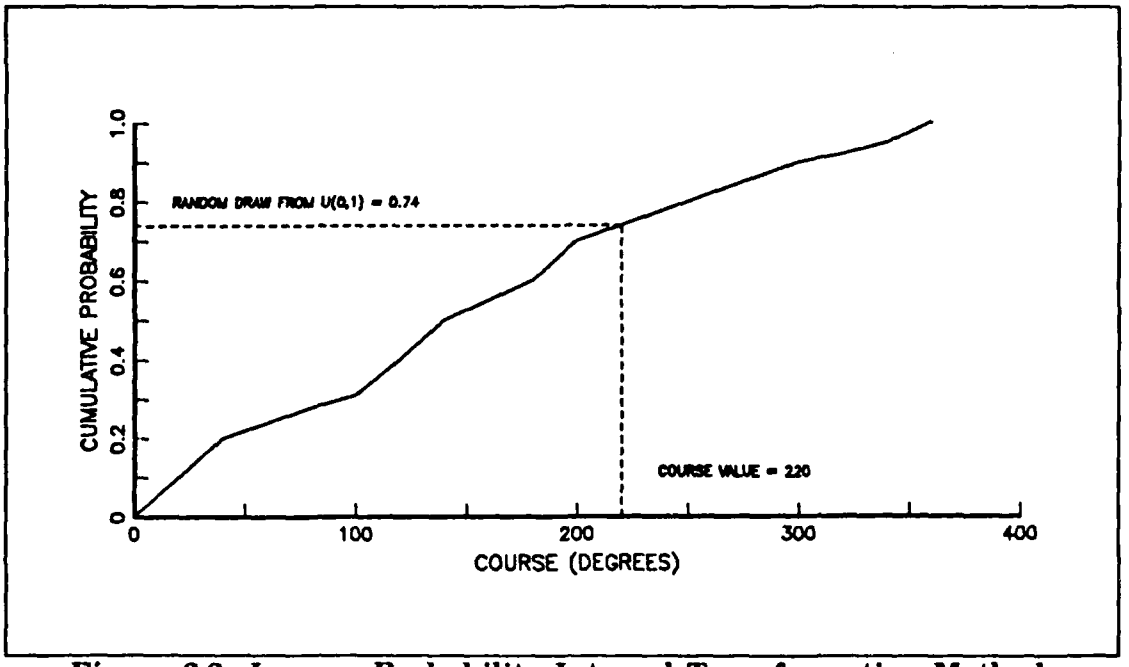

Figure 2.2 Inverse Probability Integral Transformation Method.

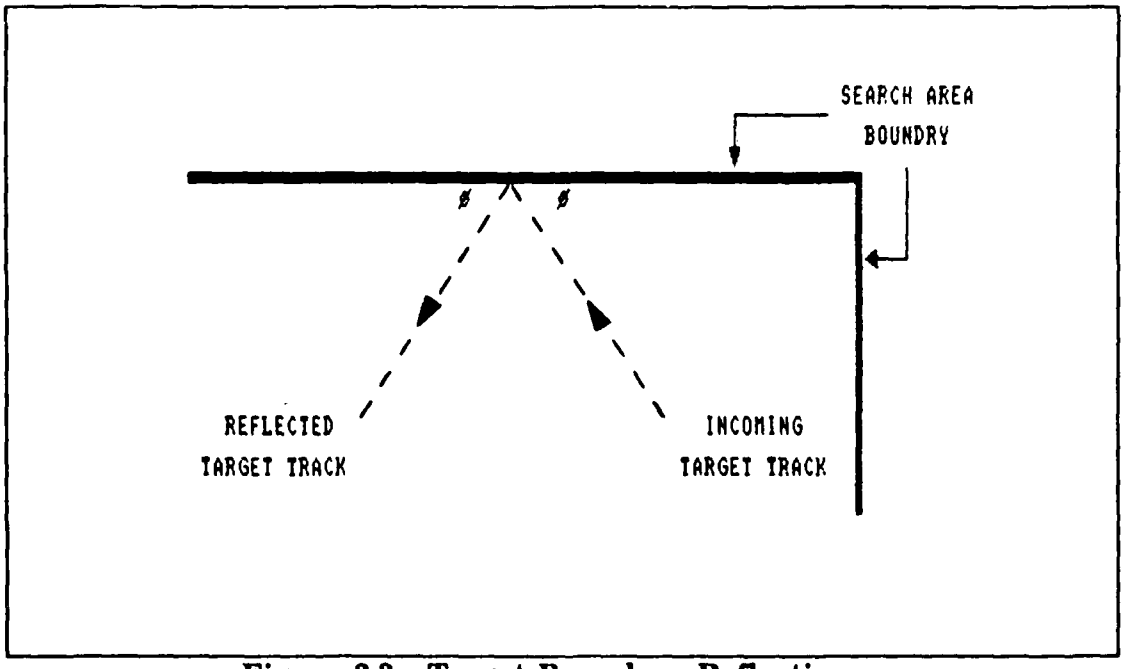

Figure 2.3. Target Boundary Reflection.

E. SEARCHER CHARACTERISTICS

Searcher characteristics modeled in SUBTRAN include: speed and course. The depth of the searcher relative to the target is assumed constant.

1. Searcher Speed

Searcher speed is held constant throughout the simulation at a value specified by the user.

2. Searcher Course

Searcher movement, unlike target movement is deterministic. The movement modeled in SUBTRAN consists of a general search movement and a transient prosecution movement.

The general search movement is an area search using parallel sweeps. The spacing between sweeps is defined as the track width, TW. TW can be user specified or defaulted. When defaulted, TW is set equal to twice the value of the mean steady-state detection range. To maximize the amount of search effort applied to the search area, the searcher is programmed to stay a distance of (0.5) x (TW) from the area boundary. This distance constraint forces an adjustment to the searcher's movement as it approaches the end of the area search. Specifically, after each vertical leg of the search, the position of the search is checked. If the horizontal distance between the searcher and the search area boundary that lies ahead of the searcher is less than (1.5) x (TW), the next horizontal leg of the search is adjusted. The adjustment that occurs is a shortening of the distance the searcher normally travels. The normal distance of the horizontal leg of the search is TW. The adjusted distance, TW_{adj}, is given by the following expression:

$$TW_{adj} = TW \times INT(L/TW)$$

where,

L = length dimension of the search area

INT(L/TW) = integer part of L/TW

Figure 2.4 illustrates the track width adjustment and displays the resulting return area search that occurs. Maximum search area coverage is achieved when L/TW is an integer value.

The prosecution phase of searcher movement involves suspending or terminating the current search, investigating a transient detection, and returning to the general area search if an additional transient detection does not occur.

Upon detection of a transient, the searcher either suspends the general area search or terminates a prosecution movement, depending on what search phase is being conducted, and commences a new prosecution movement. A prosecution movement involves steering the course on which the transient was detected until the search area boundary is approached. When the distance between the searcher and the search area boundary along the searcher's track reduces to (0.5) x (TW), the searcher reverses direction and continues down the reciprocal bearing until he returns to the point from which the prosecution movement was commenced. From this point, the searcher returns to the position where the general area search was suspended and resumes the general area search. If at any time during the prosecution movement phase another transient is detected, the searcher terminates the current movement phase and commences a new one based on the most recent transient detection.

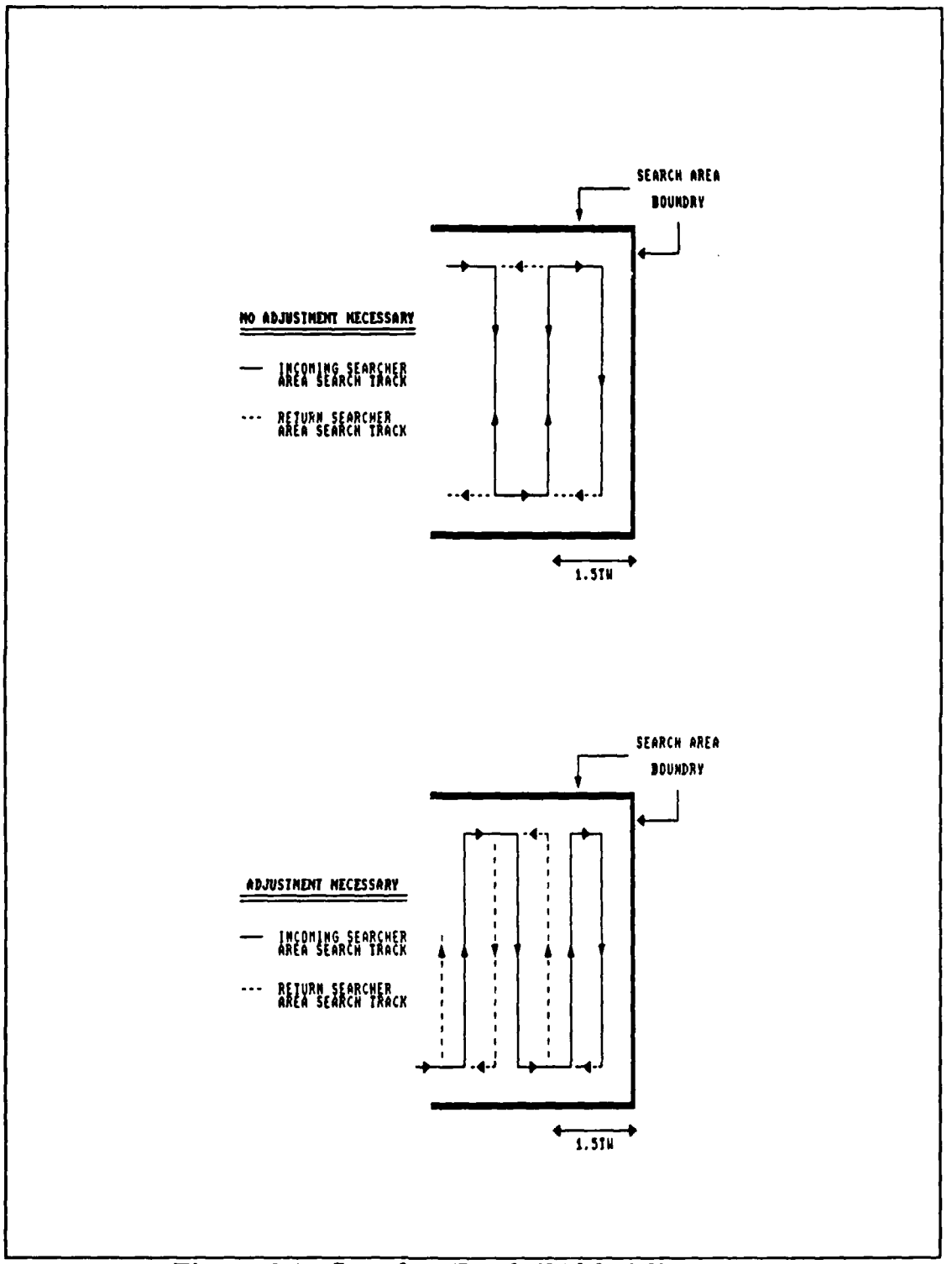

Figure 2.4. Searcher Track Width Adjustment.

F. FALSE TRANSIENT MODEL

False transients are included in SUBTRAN to represent transient sound sources that the searcher may mistakenly identify as target transients. The characteristics of false transients that are modeled in SUBTRAN include: the rate of occurrence, the position at time of occurrence, and the ability of the searcher to detect the occurrence.

False transient occurrence is modeled as a Poisson process. As a result, the rate of false transient occurrence is exponentially distributed with a user-defined rate parameter.

In an effort to best represent natural conditions, the position from which a false transient is emitted is assumed to be uniformly distributed over the search area.

The ability of the searcher to detect the occurrence of a false transient is assessed in the same manner as the detection of real transients. Specifically, the range between the searcher and the position that the false transient was emitted from is compared to the detection ranges extracted from the transient propagation loss curve. The FOM for false transients is assumed to be the same as the FOM for real transients.

G. SUMMARY OF IMPORTANT MODEL ASSUMPTIONS

The major assumptions made in the development of SUBTRAN include:

- The search area is square and its boundaries are known to both the target and searcher.
- No convergence zone steady-state detection.
- Third convergence zone transient detection.
- Target cannot counter-detect the searcher.
- Only one type of target transient sound.

- Target transient sounds are instantaneous.
- FOM is constant.
- Random fluctuations in signal excess are described by a Lambda-Sigma Jump process having the values of lambda and sigma specified by the user.
- Target and false transients are independent Poisson processes.
- False transients sounds have the same FOM as target transient sounds.
- Target position is constrained to the search area by reflection.

III. DESCRIPTION OF SUBTRAN

A. PROGRAM CODE AND STRUCTURE

SUBTRAN is coded in Fortran 77 [Ref. 6] and compiled using IBM Professional Fortran. It is designed to be implemented on a personal computer. The program is structured in modular form with major tasks such as boundary reflection, assessment of steady-state detection, and searcher movement being accomplished by separate subroutines. This type of structuring facilitates easy alteration of major program functions by a knowledgeable programmer. However, a compiled version of the program is all that is needed by most users. The compiled version was designed to allow a user, who accepts the assumptions made in this thesis, to change a select set of input parameters in an input data file and not have to perform any programming.

B. MAIN PROGRAM AND ASSOCIATED SUBROUTINES

The programming code for SUBTRAN is broken up into 19 parts: a main program and 18 supporting subroutines. The interrelationship of the various parts is shown in Figure 3.1.

1. Main Program

The main program, SUBTRAN.FOR, controls the call of the various subroutines. Specific functions performed in SUBTRAN.FOR include: initialization of counters, initial positioning of the searcher and target, calculation of the time to the next discrete event, control of the simulation time clock, storage of the times to steady-state detection, and control of the number of simulation replications.

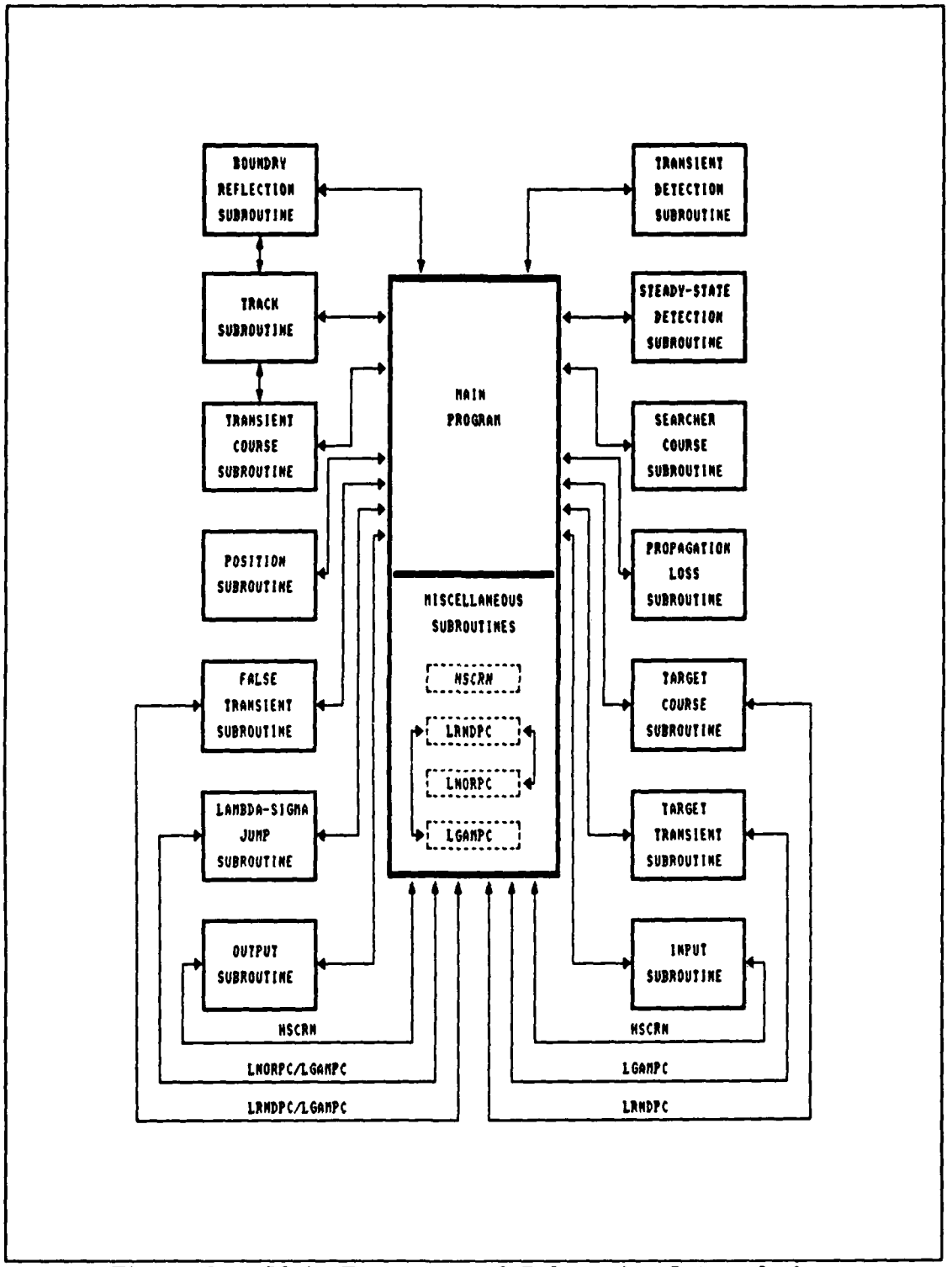

Figure 3.1. Main Program and Subroutine Interrelation.

2. Input Subroutine

The input subroutine, IN.FOR, is responsible for reading the input data file and passing the input parameters to the main program. Other functions performed in IN.FOR include: printing the input parameters to the user's terminal for verification and printing the input parameters to the output data file SUBTRAN.OUT.

3. Propagation Loss Subroutine

The propagation loss subroutine, PLOSS.FOR, is responsible for calculating the steady-state and transient detection ranges. Major parameters passed to PLOSS.FOR by the main program include: the propagation loss curve values and the adjusted FOM.

4. Target Course Subroutine

The target course subroutine, TGCR.FOR, is responsible for calculating the target's next course and time to course change. Major parameters passed to TGCR.FOR by the main program include: the target's course and time to course change CDF curve values and a random number seed.

5. Target Transient Subroutine

The target transient subroutine, TGTR.FOR, is responsible for calculating the time to the target's next transient. Major parameters passed to TGTR.FOR by the main program include: the target's transient occurrence rate parameter and a random number seed.

6. Searcher Course Subroutine

The searcher course subroutine, SRCR.FOR, is responsible for calculating the searcher's next course and time to course change during general area search movement. Major parameters passed to SRCR.FOR by the main program include: the searcher's position and speed, track width, length

dimension of the search area, and searcher's course flag. The searcher's course flag indicates which part of the general area search is to be performed next.

7. Transient Course Subroutine

The transient course subroutine, TRCR.FOR, is responsible for calculating the searcher's next course and time to course change during transient prosecution movement. Major parameters passed to TRCR.FOR by the main program include: the searcher's position and speed, track width, length dimension of the search area, and searcher's course flag and transient course flag. The searcher's transient course flag indicates two things: whether the searcher is performing a general area search movement or transient prosecution movement, and if a transient prosecution movement is being performed, which leg of the movement is to be performed next.

8. Steady-State Detection Subroutine

The steady-state detection subroutine, SSDET.FOR, is responsible for calculating the time to steady-state detection. Major parameters passed to SSDET.FOR by the main program include: the position of the searcher and target, the course and speed of the searcher and target, and the steady-state detection range.

9. Transient Detection Subroutine

The transient detection subroutine, TRDET.FOR, is responsible for assessing the occurrence of target transient and false transient detection. Major parameters passed to TRDET.FOR by the main program include: the position of the searcher and the transient emission, and the transient detection ranges.

10. Lambda-Sigma Jump Subroutine

The Lambda-Sigma Jump subroutine, LSJ.FOR, is responsible for calculating the magnitude of the next signal excess fluctuation as well as the

time to occurrence. Major parameters passed to LSJ.FOR by the main program include: the Lambda-Sigma Jump error process rate parameter and standard deviation, and a random number seed.

11. False Transient Subroutine

The false transient subroutine, FALTR.FOR, is responsible for calculating the position of and time to the next false transient. Major parameters passed to FALTR.FOR by the main program include: the false transient rate parameter, the length dimension of the search area, and a random number seed.

12. Position Subroutine

The position subroutine, POSIT.FOR, is responsible for updating the X and Y positions of the searcher and target. Major parameters passed to POSIT.FOR by the main program include: the previous X and Y positions of the searcher and target, the course and speed of the searcher and target, and the time passed since the last update.

13. Boundary Reflection Subroutine

The boundary reflection subroutine, REFLCT.FOR, is responsible for calculating the targets next boundary reflection course and the time to reflection. Major parameters passed to REFLCT.FOR by the main program include: the target's position, the target's course and speed, and the length dimension of the search area. REFLECT.FOR is used by the TRACK.FOR subroutine in addition to the main program.

14. Track Subroutine

The track subroutine, TRACK.FOR, is responsible for calculating the course and time necessary for the searcher to reach a given point in the search area. Major parameters passed to TRACK.FOR by the main program include: the positions of the searcher and the point to be reached, the speed of the searcher,

and the length dimension of the search area. TRACK.FOR is used by the TRCR.FOR subroutine in addition to the main program.

15. Output Subroutine

The output subroutine, OUT.FOR, is responsible for writing the output data to the user's terminal and to the two output data files.

16. Miscellaneous Subroutines

Four miscellaneous subroutines are called by the main program and other subroutines: HSCRN.FOR, LRNDPC.FOR, LNORPC.FOR, and LGAMPC.FOR.

The subroutine HSCRN.FOR is used by the main program, the input subroutine, and the output subroutine to stop the output display on the user's terminal.

The subroutines LRNDPC.FOR, LNORPC.FOR, and LGAMPC.FOR are used by the main program and various subroutines to generate random numbers. All three are an adapted version of the subroutines, with the same names, developed by Lewis, Orav, and Uribe [Ref. 7].

LRNDPC.FOR is used to generate uniform (0,1) random variables, LNORPC.FOR is used to generate normal (0,1) random variables, and LGAMPC.FOR is used to generate exponential (1) random variables.

C. PROGRAM OUTPUT

The output provided by SUBTRAN consists of: a verification of input parameters, various output statistics, and the specific times to detection. The output is written to two separate output files named SUBTRAN.OUT and TIME.OUT.

1. Verification of Input Parameters

All parameters entered via the input data file are printed in the output file SUBTRAN.OUT. This allows user verification of input as well as a hardcopy display.

2. Various Output Statistics

SUBTRAN performs elementary statistical analysis on the times to detection and provides the results in the output data file SUBTRAN.OUT. The statistics include: mean and standard deviation of the times to detection, 95% confidence bound for the mean, estimate of the mean time to detection assuming the times to detection are exponentially distributed, R^2 confidence factor for the estimated mean time to detection, and the cumulative distribution function of the times to detection in tabular form. Calculations are performed on the times to detection conditioned on no detection at time zero. In other words, runs with detection time zero were disregarded. The number of simulation runs with a zero detection time is included in the output summary.

The mean and standard deviation are computed using normal methods. For the replications in which steady-state detection does not occur, T_{max} is used as the detection time for the calculation of the mean and standard deviation. This results in a lower bound on the actual mean time to detection and standard deviation. The number of times to detection that are set to T_{max} is provided in the output summary.

The 95% confidence bound for the mean is calculated using:

$$95\% \text{ bound} = \sigma/N^{1/2}$$

where,

σ = standard deviation of the times to detection.

N = number of simulation replications.

The estimated mean time to detection, under the assumption that the times to detection are exponentially distributed, is calculated using straight line regression of the natural log of the empirical survival function on the ordered times to detection [Ref. 8:pp. 604-607]. The survival function of an exponential distribution when plotted on a natural log scale is a straight line. The magnitude of the slope of the straight line is equal to the rate parameter of the exponential distribution. The reciprocal of the rate parameter is the estimated mean time to detection. Replications in which steady-state detection did not occur are disregarded in the regression under the assumption that if the distribution is truly exponential, their inclusion would not affect the estimate of the slope.

The R^2 confidence factor for the estimated mean time to detection describes the proportion of the variation in the sample values of the natural log of the survival function that the regression estimate accounts for. R^2 was calculated using normal methods [Ref. 8:p. 644].

The cumulative distribution function of the times to detection is empirically derived from the ordered times to detection [Ref. 9:pp. 11-16] and is provided in tabular form. An example of the output statistics provided by SUBTRAN in contained in Appendix D.

3. Specific Times to Detection

The specific times to detection are provided in the output data file named TIME.OUT. The times are ordered from shortest to longest with zero times removed and T_{max} value included as necessary.

IV. VALIDATION OF SUBTRAN

Validation of a simulation model can be broken up into two major parts: validation that the computer code is functioning as expected and validation that the simulation model is an accurate assessment of the real world situation being modeled.

With regard to SUBTRAN, validation that the computer code is functioning as expected was conducted using two separate techniques. First, the simulation was conducted with a fixed target and the time to detection was compared to a uniform distribution. Second, the simulation was conducted with a fixed searcher and the time to detection was compared to model results for a random tour target developed by Abd El-Fadeel [Ref. 10]. The results of these two tests were used to validate the proper simulation coding of searcher movement, target positioning, target movement, target boundary reflection, and steady-state detection assessment.

Areas of program code not rigorously validated include searcher transient prosecution, transient occurrence, and false transient occurrence. In the case of transient occurrence and false transient occurrence, the program code is relatively straightforward and rigorous validation is not necessary. The program code for searcher transient prosecution is more complex and validation was conducted by reviewing searcher position data after simulation runs in which transients were allowed. The data reviewed indicated the searcher was performing transient prosecutions as modeled.

Validation, in a strict sense, that the simulation model is an accurate assessment of the real world situation being modeled is not possible. There

exists little to no data that could be used in a rigorous validation of SUBTRAN. However, the results produced by the example runs of SUBTRAN lend credibility to the modeling technique used. The results clearly indicated intuitive results.

A. UNIFORM DISTRIBUTION APPROXIMATION

By altering the search scenario modeled in SUBTRAN, a uniform distribution of the times to detection should be achieved. This contention is obvious from the fact that at the start of each simulation replication the target is uniformly positioned in the search area. With the target's speed set at zero, the time it takes the searcher to detect the target on one complete search of the area should be uniformly distributed on the open interval (0,T). Where T is the time it takes the searcher to make one complete search of the area.

This uniform distribution approximation was tested on SUBTRAN using four different sets of input. The input values are summarized in Table IV.1. Figures 4.1 through 4.4 graphically compare the estimated probability of non-detection as a function of search time with the theoretical probability of non-detection for the four cases. The theoretical probability of non-detection is indicated as a dotted line. The estimated mean time to detection (MTTD), its 95% confidence interval, and the theoretical MTTD are also included.

TABLE IV.1. UNIFORM APPROXIMATION INPUT PARAMETERS.

Case	R (nm)	L (nm)	S_t (nm/h)	S_e (nm/h)	T_{max} (h)
1	10	100	0	10	48
2	5	100	0	10	99
3	10	100	0	5	96
4	10	225	0	10	264.5

* 1000 replications performed in all cases.

The results of the uniform approximation test strongly indicate proper functioning of the simulation with regard to searcher movement and target positioning. One note worth mentioning deals with the slight graphical deviation in theoretical and estimated probability of non-detection shown in Figure 4.2, case 4. The graphical results indicate that detection in the simulated case occurs sooner than detection in the theoretical case at large values of time. The reason for this is explained by the geometry of the particular search. In the first three test cases the length dimension of the search area was evenly divisible by the searcher's track width. As a result, a complete search of the area required no overlap of search effort. However, in the fourth case tested, the division of the length dimension by the track width did not result in a integer and overlap occurred on the last leg. This overlap caused the deviation in the CDF. Target detections that should have theoretically taken place on the last search leg occurred one leg earlier.

B. RANDOM TOUR MODEL APPROXIMATION

Shifting to a moving target and a stationary searcher, the results of SUBTRAN can be compared directly with the random tour model [Ref. 10:pp. 59-62]. The random tour model was developed using Monte Carlo simulation techniques and applies to the detection of a randomly moving target constrained to a square search area by boundary reflection. The model also assumes that the time between target course changes is exponentially distributed with rate parameter τ and that the searcher is fixed in the center of the search area with a cookie-cutter detection radius. The output of the model estimates the probability of non-detection with respect to time for given values of searcher detection range (R), target speed (V), search area size (A), and the target course change rate parameter (τ). The equation for the probability of non-detection

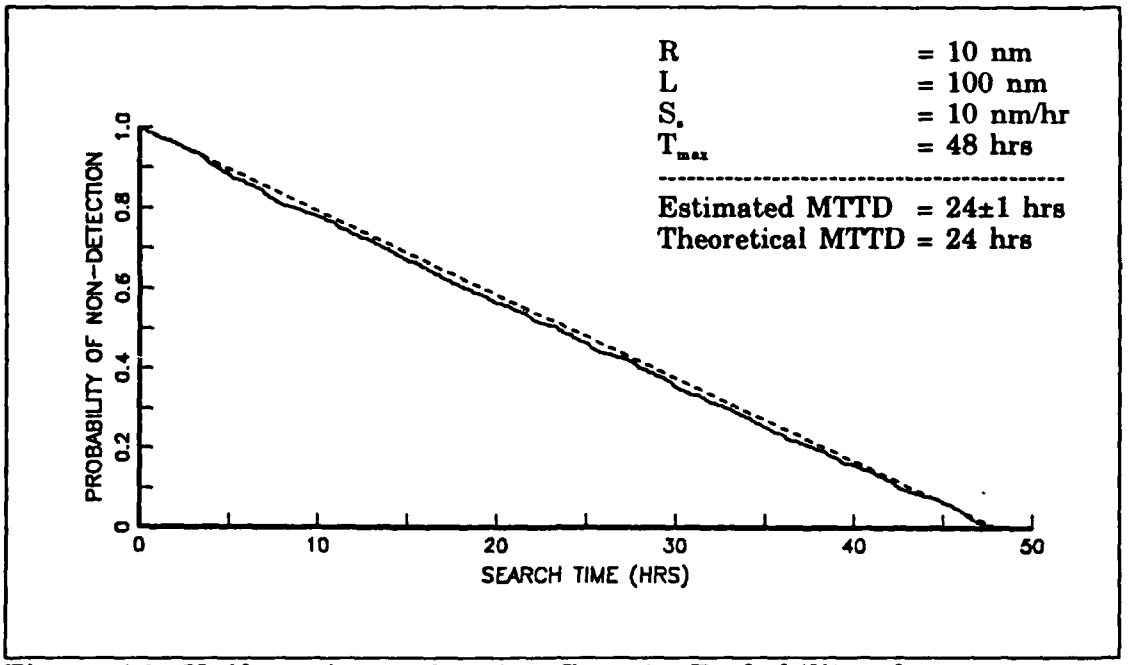

Figure 4.1. Uniform Approximation Case 1 - Probability of Non-Detection.

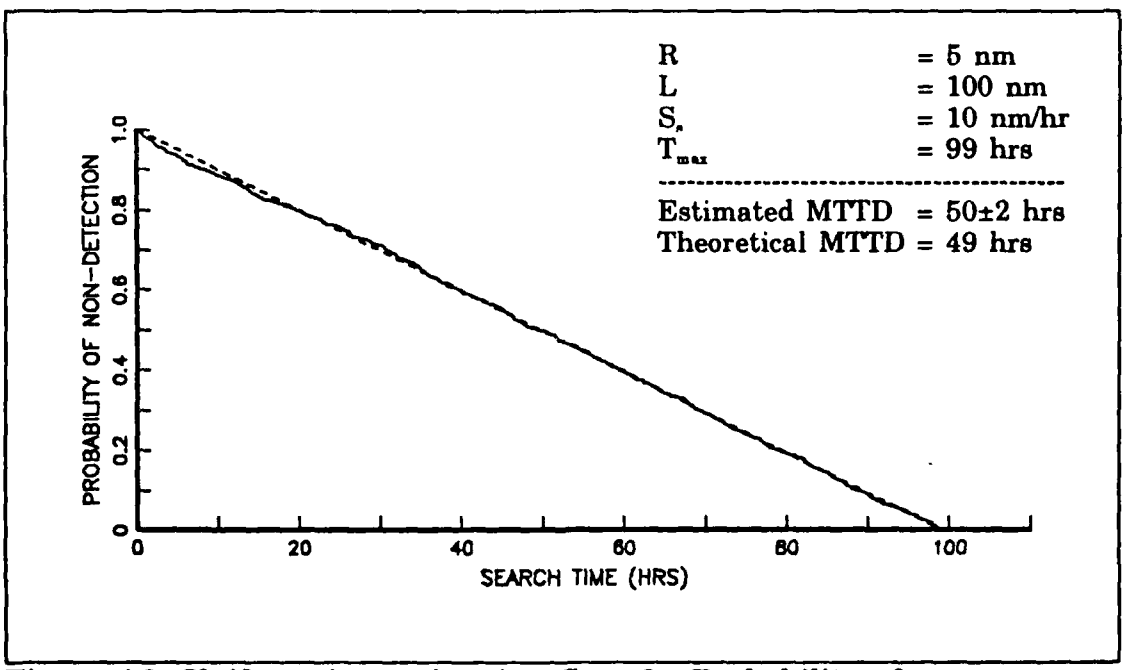

Figure 4.2. Uniform Approximation Case 2 - Probability of Non-Detection.

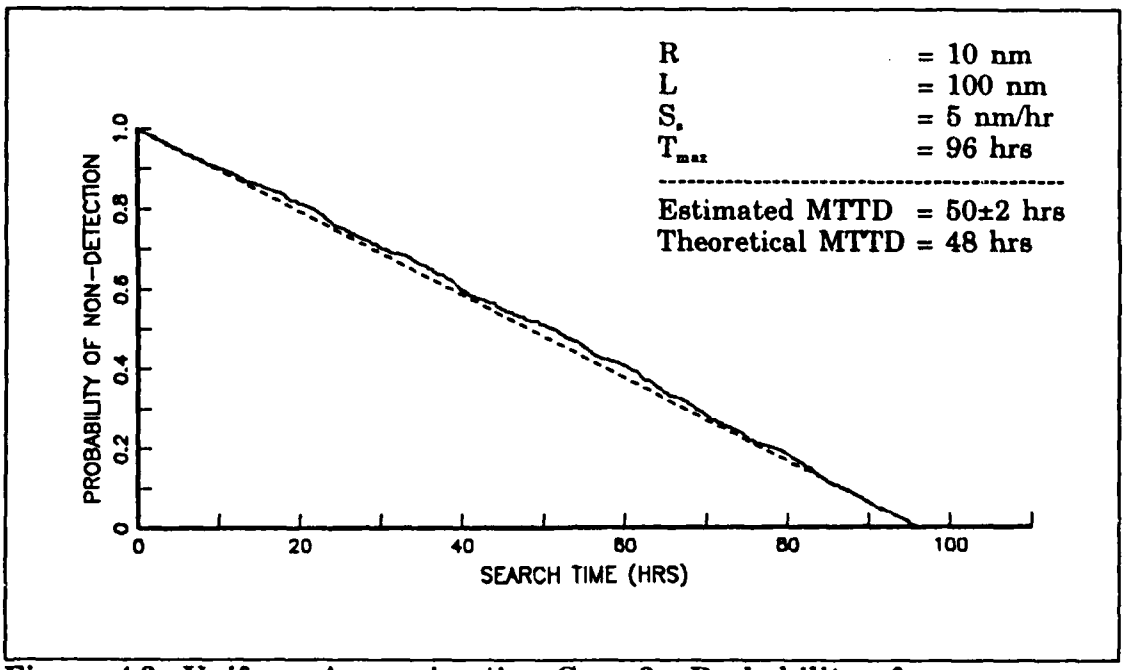

Figure 4.3. Uniform Approximation Case 3 - Probability of Non-Detection.

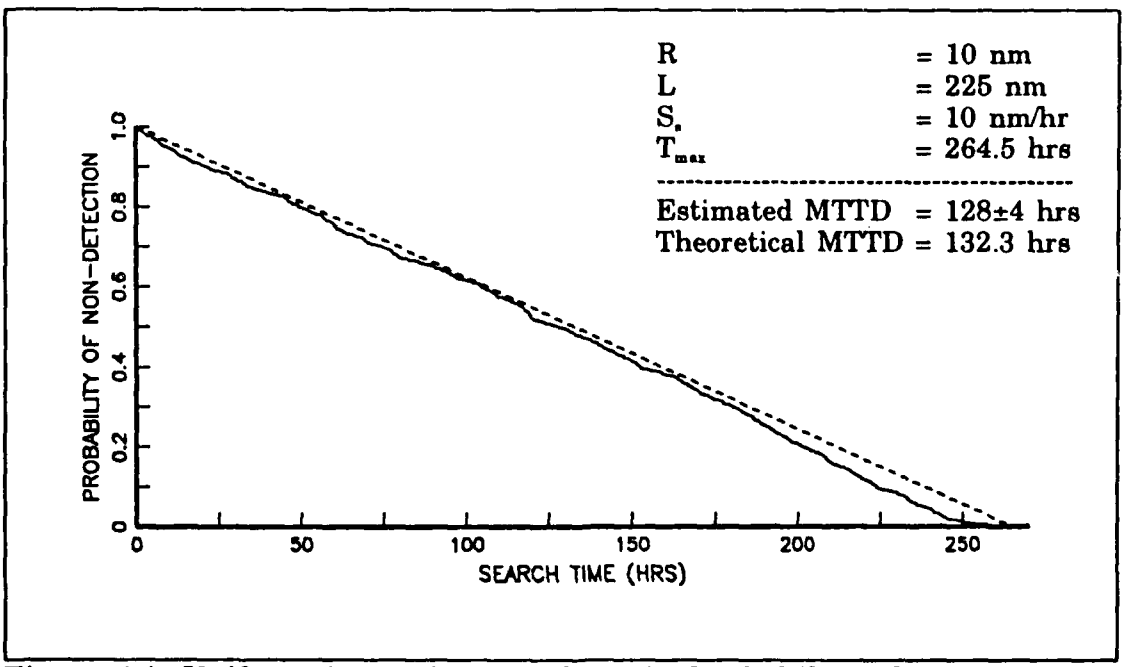

Figure 4.4. Uniform Approximation Case 4 - Probability of Non-Detection.

(PND), given that detection did not occur at time zero, proposed by the random tour model is:

$$PND(t) = \exp^{-\alpha t}$$

where,

$$\alpha = [(24.7RV^2) / \tau(A - \pi R^2)^{1.5}] \times [1 - \exp(-0.084\tau A^{0.5}/V)]$$

Four SUBTRAN simulation runs were compared to the random tour model. The input parameters are summarized in Table IV.2. Figures 4.5 through 4.8 graphically compare the estimated probability of non-detection as a function of search time with the random tour model estimate for probability of non-detection for the four cases. The random tour model estimate is indicated as a dotted line. The estimated MTTD, its 95% confidence interval, and the random tour model (RTM) estimate of MTTD are also included.

TABLE IV.2. RANDOM TOUR MODEL APPROXIMATION INPUT PARAMETERS.

Case	R (nm)	L (nm)	S_t (nm/h)	τ (h^{-1})	T_{max} (h)
1	10	100	10	1	1000
2	5	100	10	1	1000
3	10	100	5	1	1000
4	10	100	10	0.6	1000

* 1000 replications were performed in all cases.

This test shows the close comparison between the results produced by SUBTRAN and those predicted by the random tour model. In all cases tested, SUBTRAN produced a slightly shorter time to detection than the random tour

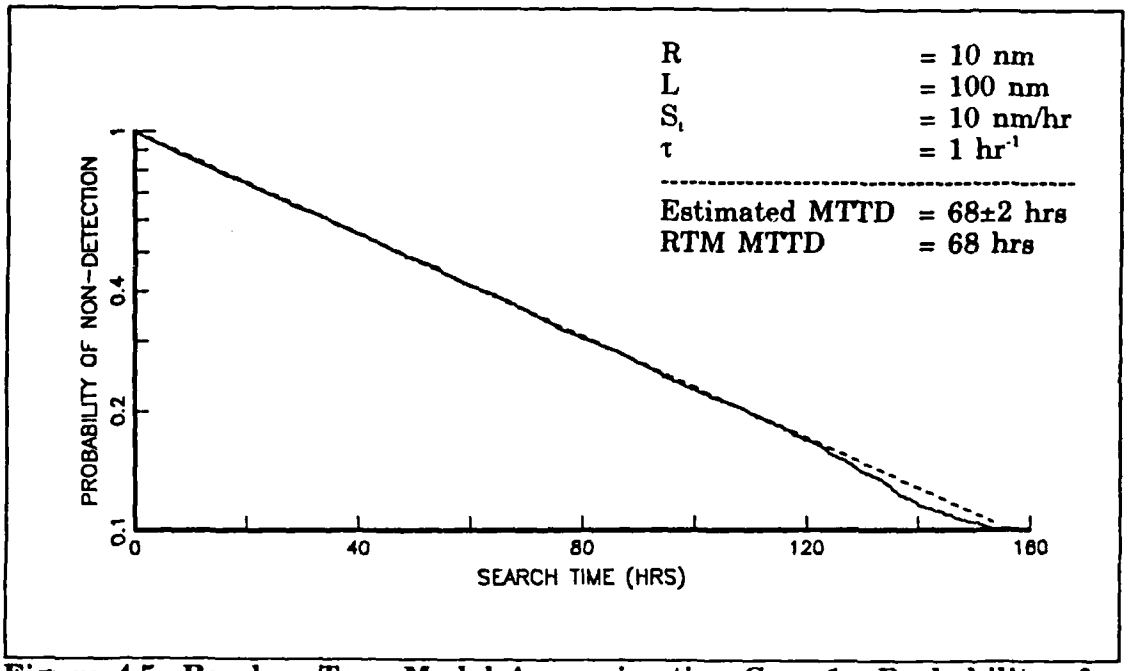

Figure 4.5. Random Tour Model Approximation Case 1 - Probability of Non-Detection.

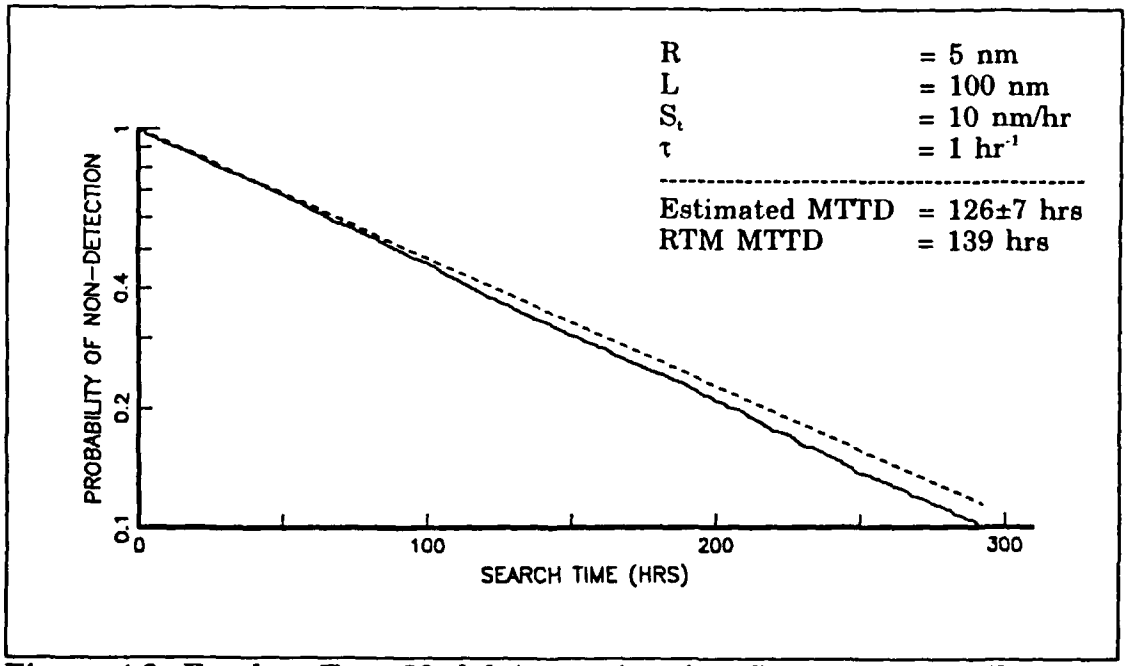

Figure 4.6. Random Tour Model Approximation Case 2 - Probability of Non-Detection.

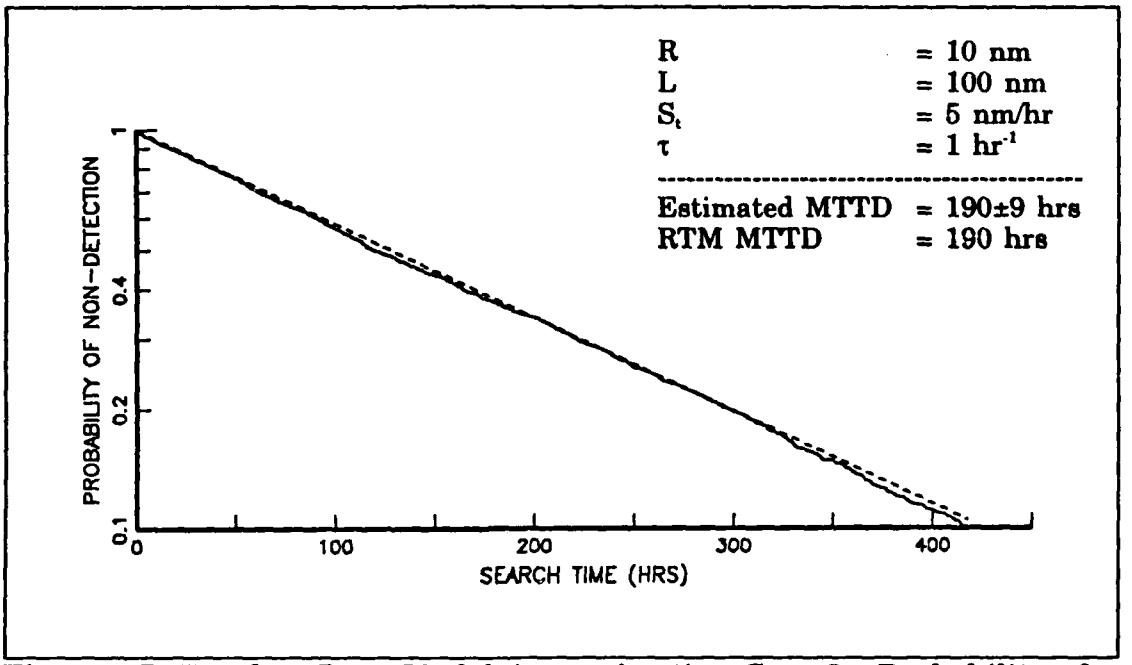

Figure 4.7. Random Tour Model Approximation Case 3 - Probability of Non-Detection.

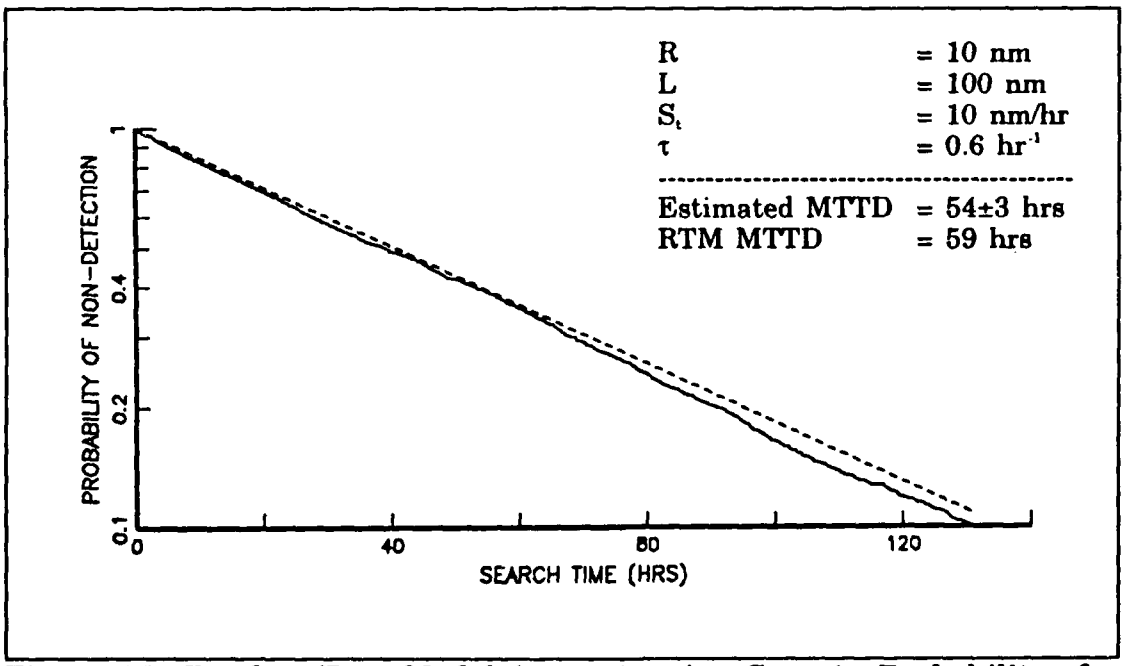

Figure 4.8. Random Tour Model Approximation Case 4 - Probability of Non-Detection.

model. These results strongly validate the modeling of target movement, target boundary reflection, and computation of the occurrence of steady-state detection.

V. EMPLOYMENT OF SUBTRAN

A. INTRODUCTION

This chapter is designed to demonstrate the employment of SUBTRAN. Five separate search scenarios involving plausible input parameters were simulated using various combinations of target and false transient occurrence rates. The output statistics of these five test cases are presented in graphical form to emphasize the relationship between mean time to detection (MTTD), mean time between target transients (MTBTT), and mean time between false transients (MTBFT). In addition, the distributions of the times to detection are analyzed to assess the assumption that they belong to the exponential class of distributions.

Two additional search scenarios, Excursion 1 and Excursion 2, were simulated to provide a further understanding of how false transient occurrences affect MTTD.

The numeric output statistics generated by SUBTRAN were used to develop the graphs displayed in this chapter.

B. INPUT PARAMETERS

The input parameters that remained common among all five test case and the two excursions are:

- Steady-state sound propagation loss curve displayed in Figure 5.1.
- Transient sound propagation loss curve equal to the steady-state sound propagation loss curve displayed in Figure 5.1.
- Transient sound figure of merit set at 90 dB.
- Target's course cumulative distribution curve displayed in Figure 5.2.

- Target's time to course change cumulative distribution curve displayed in Figure 5.3.
- Lambda-Sigma Jump model parameters. Lambda set at 3 hour^{-1}; Sigma at 6 dB.
- Track width flag set at 0.
- Number of simulation replications set at 1000.

The input parameters that varied over the five test cases and two excursions were: the mean steady-state detection range (R), the length dimension of the search area (L), the target's speed (S_t), the searcher's speed (S_s), and the maximum search time (T_{max}). The values of these parameters are summarized in Table V.1.

The target and false transient occurrence rates, converted to MTBTT and MTBFT respectively, used in each test case and excursion are:

- MTBTT's of 1000, 200, 100, 75, 50, 10, 5 (in hours).
- MTBFT's of 400, 75, 10, 5, 1 (in hours).

In addition, simulation runs were made in which no target and/or false transient sounds were allowed.

C. OUTPUT STATISTICS AND ANALYSIS

The MTTDs and associated 95% confidence intervals for the five test cases and two excursions are tabulated in Tables V.2 through V.8.

Three dimensional graphical displays of the relationship between MTTD, MTBTT, and MTBFT for the five test cases and two excursions are contained in Figures 5.4 through 5.8, 5.14, 5.16.

Graphical analyses of the probability of non-detection (PND) for the five test cases and two excursions are displayed in Figures 5.9 through 5.13, 5.15, 5.17.

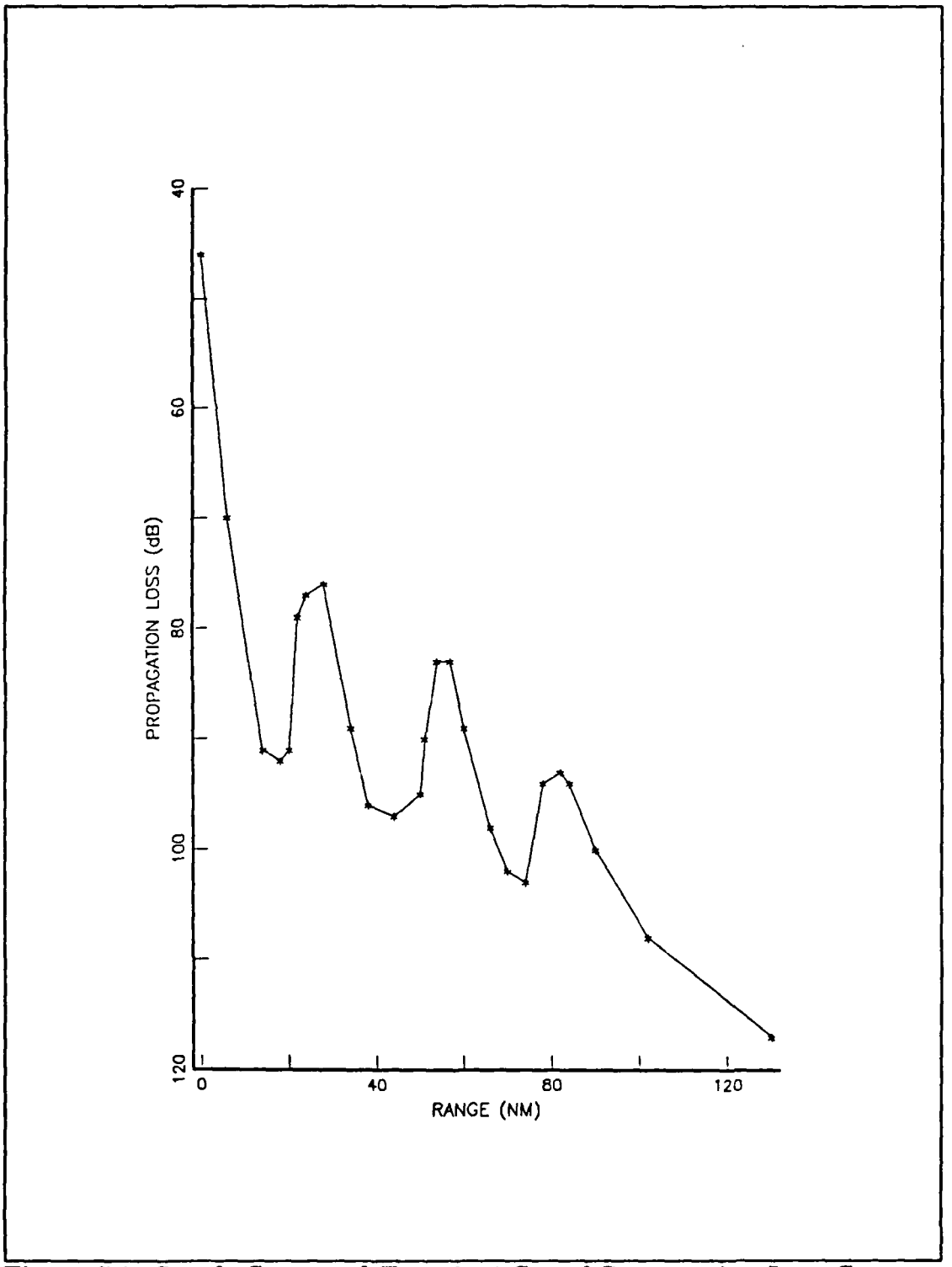

Figure 5.1. Steady-State and Transient Sound Propagation Loss Curve.

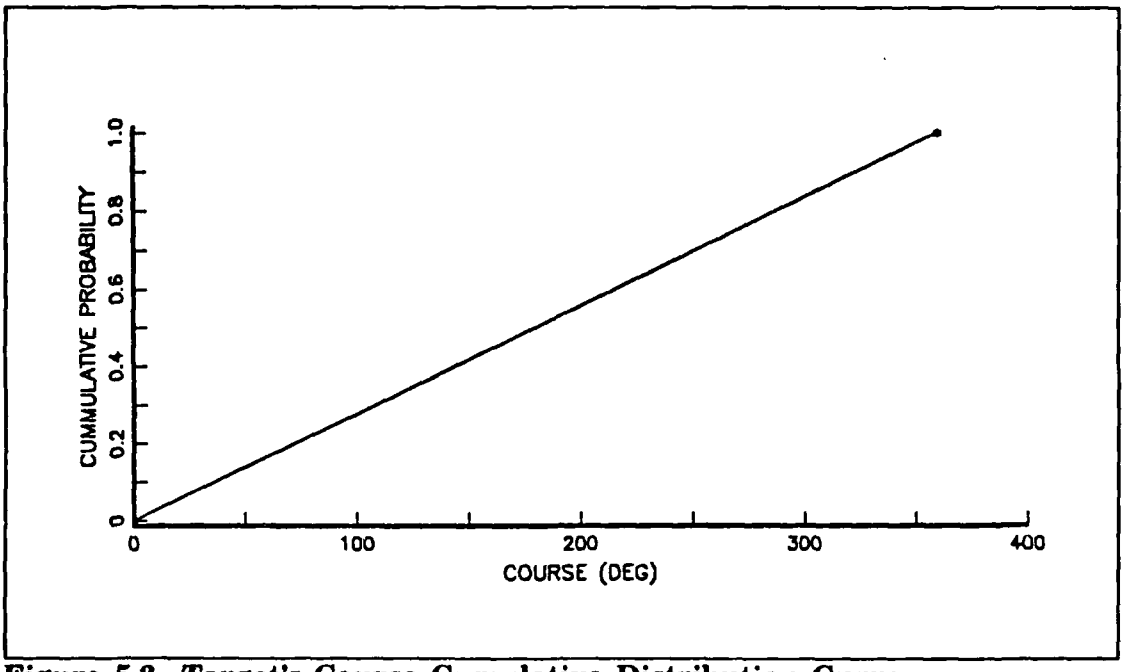

Figure 5.2. Target's Course Cumulative Distribution Curve.

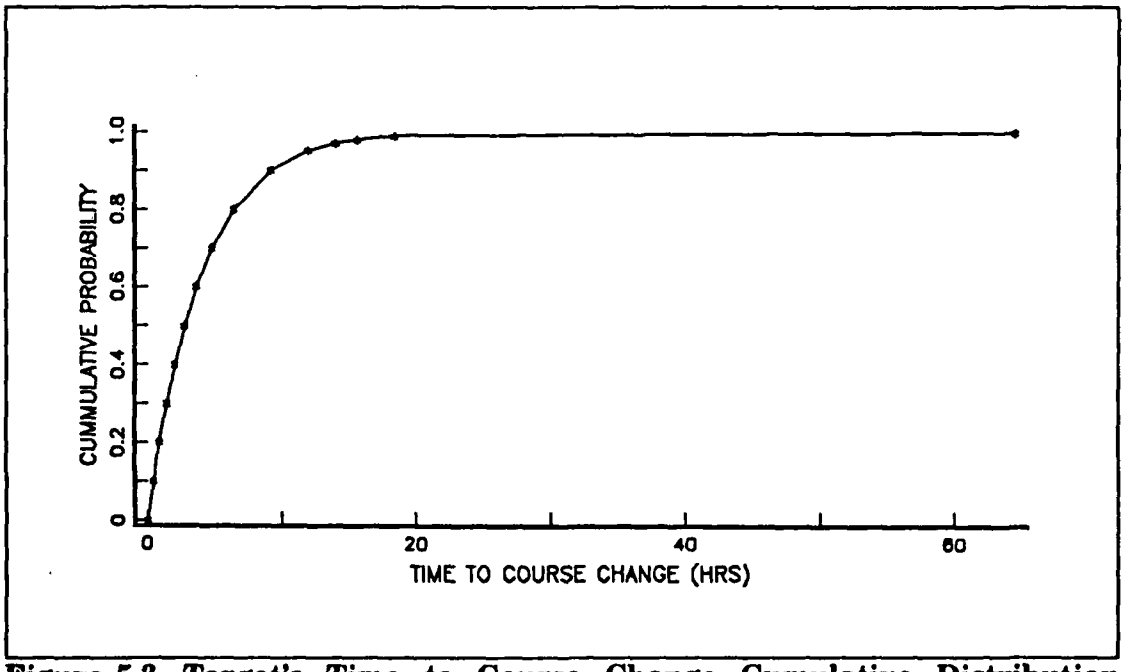

Figure 5.3. Target's Time to Course Change Cumulative Distribution Curve.

TABLE V.1. EMPLOYMENT INPUT PARAMETERS.

Test Case	R (nm)	L (nm)	S_t (nm/h)	S_s (nm/h)	T_{max} (h)
1	5.0	100	5.0	10	1000
2	2.5	100	5.0	10	1000
3	2.5	100	3.0	15	1000
4	2.5	100	3.0	5	2000
5	2.5	150	3.0	5	4000
EX 1	10.0	100	0.25	10	1000
EX 2	5.0	100	5.0	10	1000

* EX 2 involves deceptive false transient sounds.

1. Test Cases 1 Through 5

Test cases 1 through 5 involve variations in R, L, S_t, S_s,and T_{max}. The variations of R, L, S_s, and S_t were chosen to encompass plausible search scenarios. T_{max} was chosen, based on the search scenario involved, to minimize the number of simulation runs in which steady-state detection did not occur.

a. Relationship Between MTTD, MTBTT, and MTBFT

Figures 5.4 through 5.8 indicate the following three points:

- MTTD decreases as MTBTT decreases throughout the range of MTBTT's tested.
- The "decrease" trend is present in MTTD, as a function of MTBTT, for all values of MTBFT tested.
- With respect to MTBFT, MTTD is relatively constant for a given MTBTT unless a high false transient occurrence rate is present (i. e. MTBFT < 5 hours).

These three points highlight the general results indicated by Test Cases 1 through 5. Specifically, target transient sounds occurring at relatively moderate rates aid the searching submarine in the steady-state search effort,

while false transient sounds only hamper the search effort when their occurrence rate is high.

Figures 5.9 through 5.13 further emphasize the effects of target and false transient sounds on the searcher's steady-state search effort. Three lines are plotted on each graph representing the PND for the three scenarios: no target or false transient sounds present, only target transient sounds present with a MTBTT of 5 hours, and both target and false transient sounds present with a MTBTT of 5 hours and a MTBFT of 1 hour. The improvement in the time to steady-state detection is indicated by comparing the line associated with no transient sounds and the line associated with only target transient sounds. The degradation in the time to steady-state detection caused by the introduction of false transients is indicated by comparing the line associated with only target transient sounds and the line with both target and false transient sounds. In all of the test cases the magnitude of the decrease in the time to steady-state detection realized by the introduction of target transient sounds with a MTBTT of 5 hours is greater than the magnitude of the increase in the time to steady-state detection realized by adding false transient sounds with a MTBFT of 1 hour.

b. Distribution of the Times to Detection

Figures 5.9 through 5.13 can also be used to graphically assess the exponential "fit" of the estimated PNDs. Theoretical PND curves, assuming the times to detection are exponentially distributed with a mean equal to the estimated mean time to detection, are plotted as a dashed lines for comparison. It is apparent that for the five test cases, PND can be closely approximated by an exponential distribution with a mean equal to the mean time to detection.

Additional support, in favor of the exponential approximation of the PND, was provided by examining the output statistics generated by SUBTRAN. The estimated mean of the times to detection and its associated 95% confidence interval, the estimated standard deviation of the times to detection, and the mean time to detection calculated using linear regression of the times to detection (see Chapter III.C.2 for explanation) were numerically compared for agreement. If the underlying distribution is truly exponential, all three values should be approximately equal. Of the 215 simulations involved in the five test cases, 205 simulations resulted in the difference between the estimated mean of the times to detection and the estimated standard deviation to be within the 95% confidence interval associated with the mean, while 206 simulations indicated similar agreement between the estimated mean of the times to detection and the mean time to detection calculated using linear regression. The R^2 confidence factor in all simulations was greater than 0.98 with the majority of the values greater than or equal to 0.99.

2. Excursions 1 and 2

In an effort to better understand when false transients play a major role in the steady-state search effort, two excursion scenarios were simulated. The first excursion involved a slow, almost stationary (speed of 0.25 kts/hr), target and a searcher with a large mean steady-state detection range. The second excursion involved "deceptive" false transient sounds.

In the first excursion, an attempt was made to test a search scenario in which false transient sounds should noticeably degrade the search effort. As presented in Chapter IV, the times to detection are uniformly distributed in a

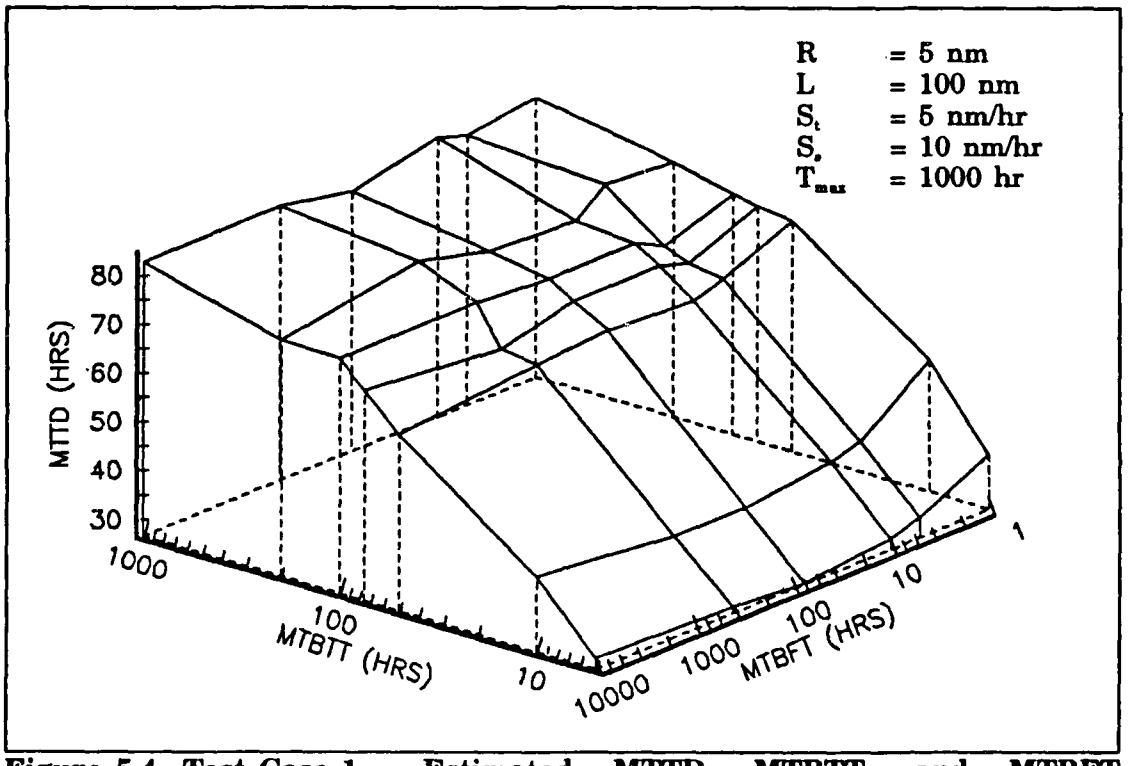

Figure 5.4. Test Case 1 - Estimated MTTD, MTBTT, and MTBFT Relationship.

TABLE V.2. TEST CASE 1 - ESTIMATED MTTD VALUES WITH 95% CI.

		MTBFT					
		1	5	10	75	400	NONE
	5	38±2	31±1	29±1	27±1	28±1	29±1
M	10	54±3	43±2	41±2	39±2	39±2	42±2
T	50	74±4	68±4	66±3	67±3	66±3	63±3
B	75	75±4	69±4	71±4	71±4	67±4	70±4
T	100	76±4	71±4	74±4	74±4	75±4	75±4
T	200	79±4	80±4	75±4	76±4	80±4	75±4
	1000	84±5	82±4	84±5	80±4	83±4	83±4
	NONE	-	-	-	-	-	85±5

* All values of MTBFT, MTBTT, and MTTD are in hours

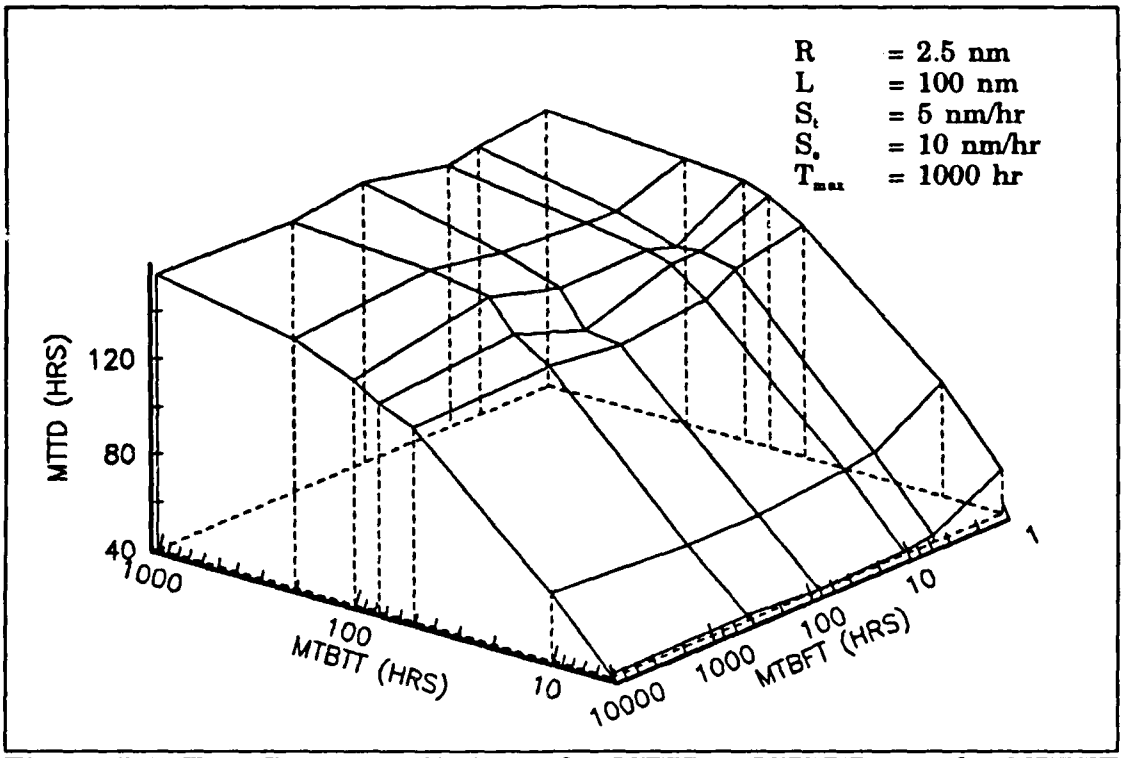

Figure 5.5. Test Case 2 - Estimated MTTD, MTBTT, and MTBFT Relationship.

TABLE V.3. TEST CASE 2 - ESTIMATED MTTD VALUES WITH 95% CI.

		MTBFT					
		1	5	10	75	400	NONE
	5	59±3	44±2	42±2	41±2	43±2	42±2
M	10	89±5	71±3	69±3	65±3	65±3	69±3
T	50	138±7	132±7	124±6	120±5	124±6	122±6
B	75	146±8	135±6	135±7	122±6	133±7	128±7
T	100	150±8	134±7	138±7	137±7	146±7	135±7
T	200	152±8	142±7	142±7	145±7	150±8	145±8
	1000	156±8	153±8	150±8	158±8	154±8	156±8
	NONE	-	-	-	-	-	157±8

* All values of MTBFT, MTBTT, and MTTD are in hours

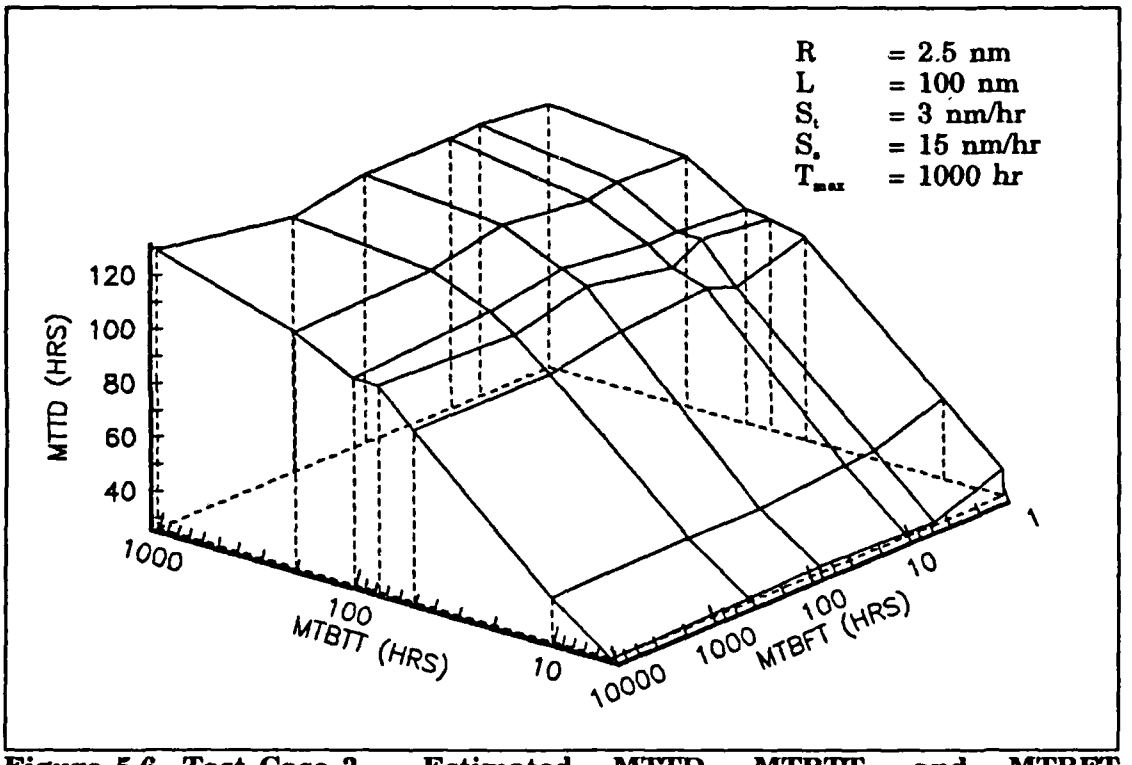

Figure 5.6. Test Case 3 - Estimated MTTD, MTBTT, and MTBFT Relationship.

TABLE V.4. TEST CASE 3 - ESTIMATED MTTD VALUES WITH 95% CI.

		MTBFT					
		1	5	10	75	400	NONE
	5	37±2	28±1	28±1	29±1	28±1	27±1
M	10	57±3	48±3	47±2	44±2	44±2	43±2
T	50	102±5	94±5	98±5	95±5	90±5	90±4
B	75	105±6	108±6	102±5	108±6	101±5	103±5
T	100	106±6	108±6	108±5	112±6	107±5	103±5
T	200	119±6	120±6	118±6	122±6	116±6	114±6
	1000	124±7	127±7	126±6	126±7	121±6	130±7
	NONE	-	-	-	-	-	127±7

* All values of MTBFT, MTBTT, and MTTD are in hours

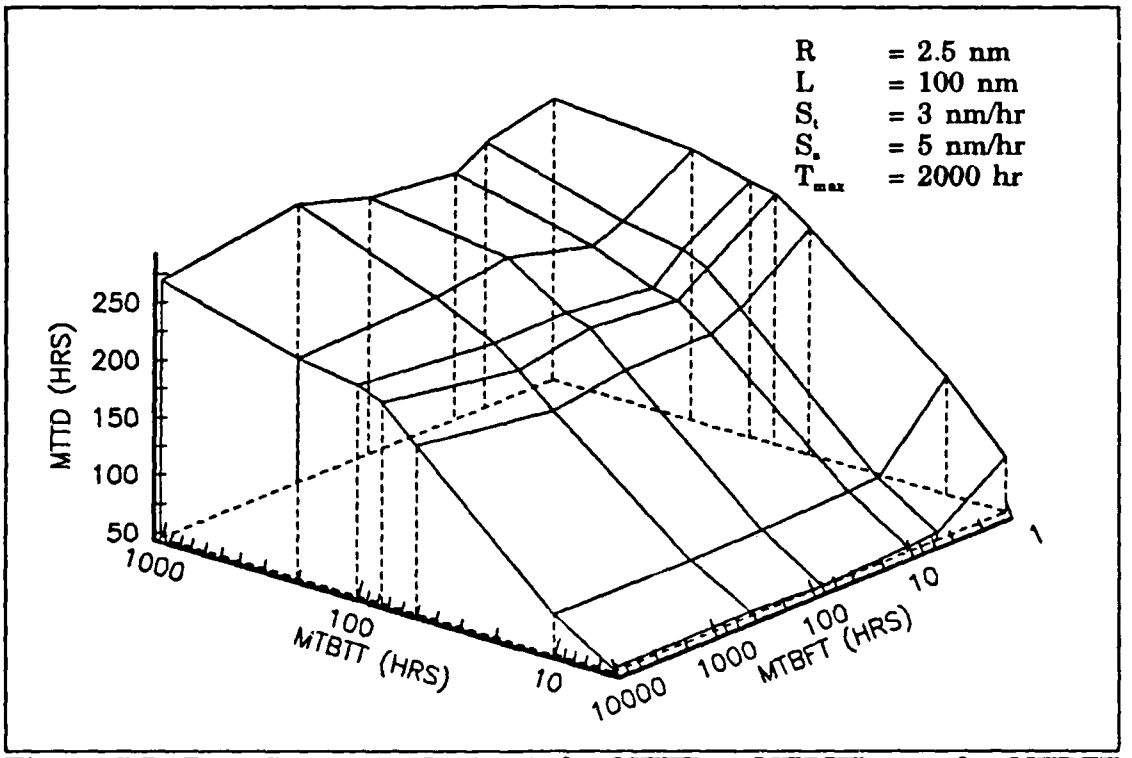

Figure 5.7. Test Case 4 - Estimated MTTD, MTBTT, and MTBFT Relationship.

TABLE V.5. TEST CASE 4 - ESTIMATED MTTD VALUES WITH 95% CI.

		MTBFT					
		1	5	10	75	400	NONE
	5	95±5	53±2	51±2	47±2	50±2	49±2
M	10	150±7	86±4	84±4	81±3	79±3	81±3
T	50	243±13	198±10	186±9	185±9	174±9	192±10
B	75	264±13	224±11	206±10	212±11	201±10	221±11
T	100	268±14	234±13	211±10	219±11	218±12	230±11
T	200	281±15	243±12	232±12	252±13	243±13	238±13
	1000	289±15	276±13	260±14	269±15	288±15	271±16
	NONE	-	-	-	-	-	273±14

* All values of MTBFT, MTBTT, and MTTD are in hours

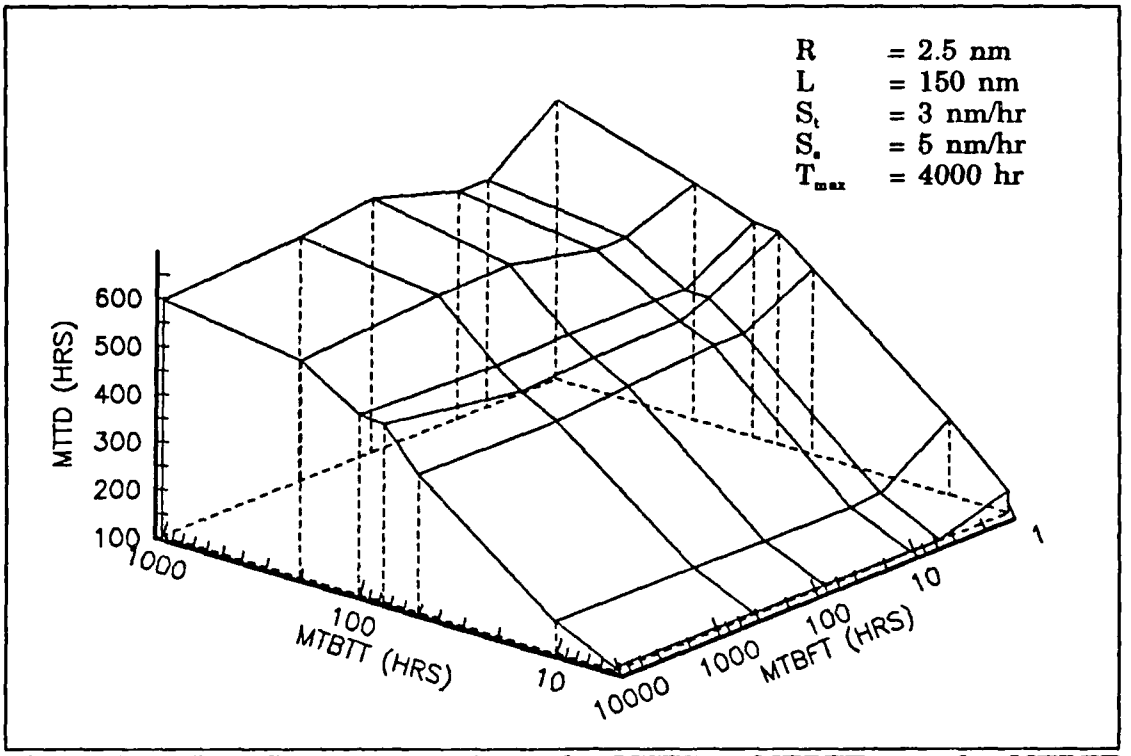

Figure 5.8. Test Case 5 - Estimated MTTD, MTBTT, and MTBFT Relationship.

TABLE V.6. TEST CASE 5 - ESTIMATED MTTD VALUES WITH 95% CI.

		MTBFT					
		1	5	10	75	400	NONE
	5	152±7	108±4	108±4	111±5	111±5	113±5
M	10	266±14	169±8	163±8	165±7	169±7	173±7
T	50	494±26	416±21	416±20	403±19	389±18	392±18
B	75	548±29	470±23	446±21	441±22	432±22	477±23
T	100	553±29	472±24	473±24	474±24	470±25	481±24
T	200	597±32	552±27	542±26	583±30	579±29	557±28
	1000	688±35	582±29	579±30	637±32	614±31	601±32
	NONE	-	-	-	-	-	604±31

* All values of MTBFT, MTBTT, and MTTD are in hours

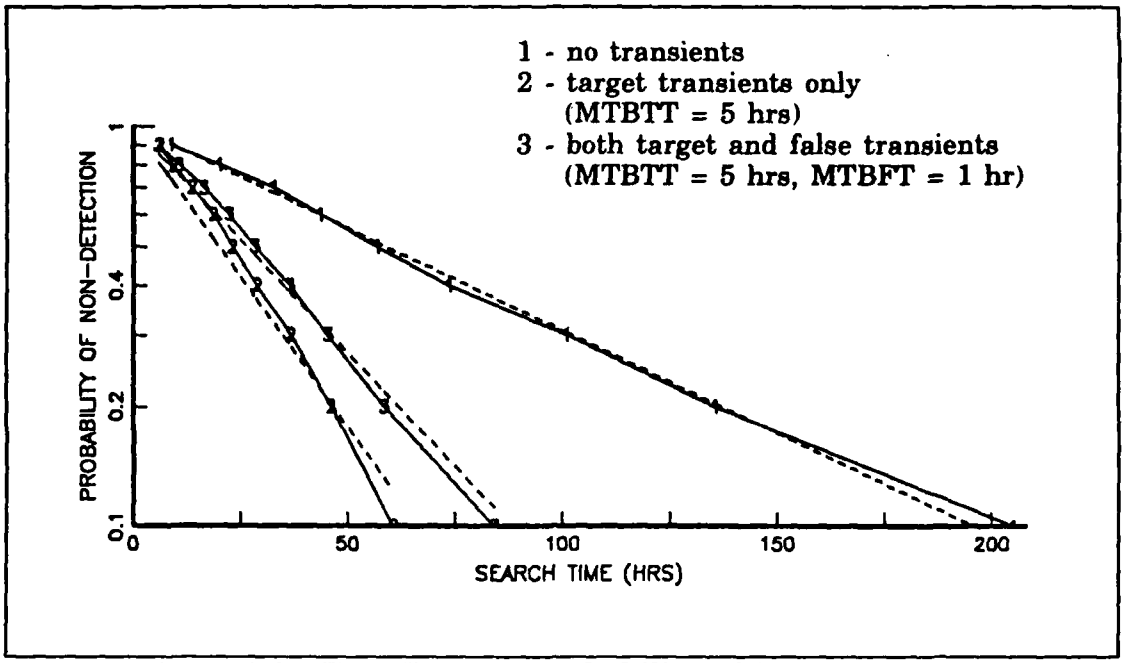

Figure 5.9. Test Case 1 - Probability of Non-Detection.

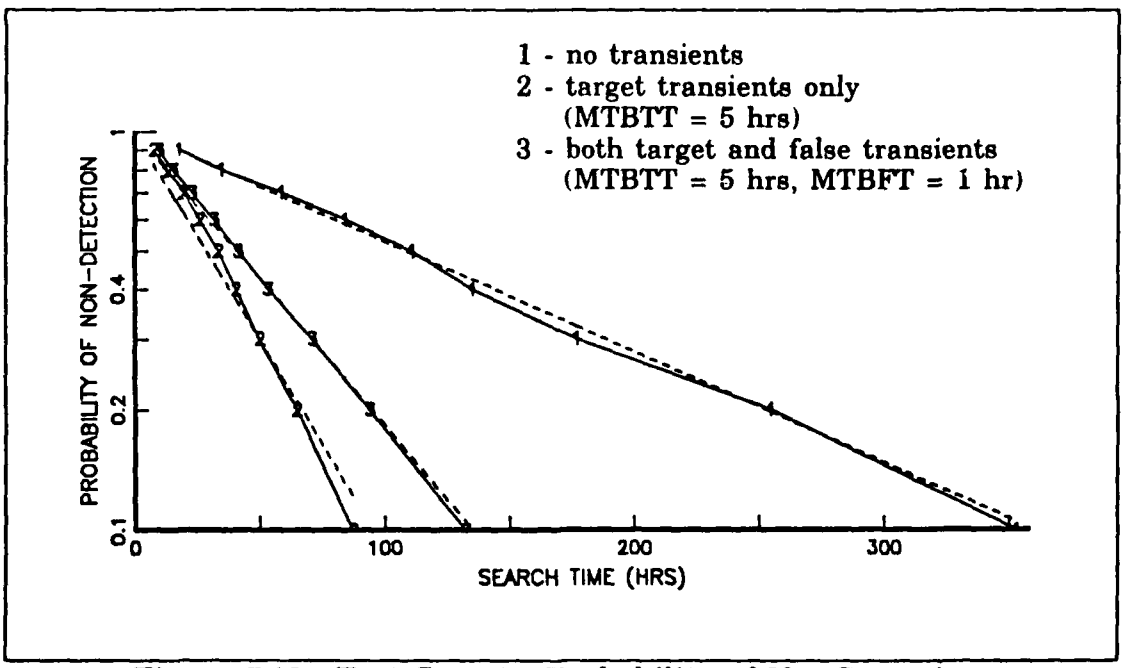

Figure 5.10. Test Case 2 - Probability of Non-Detection.

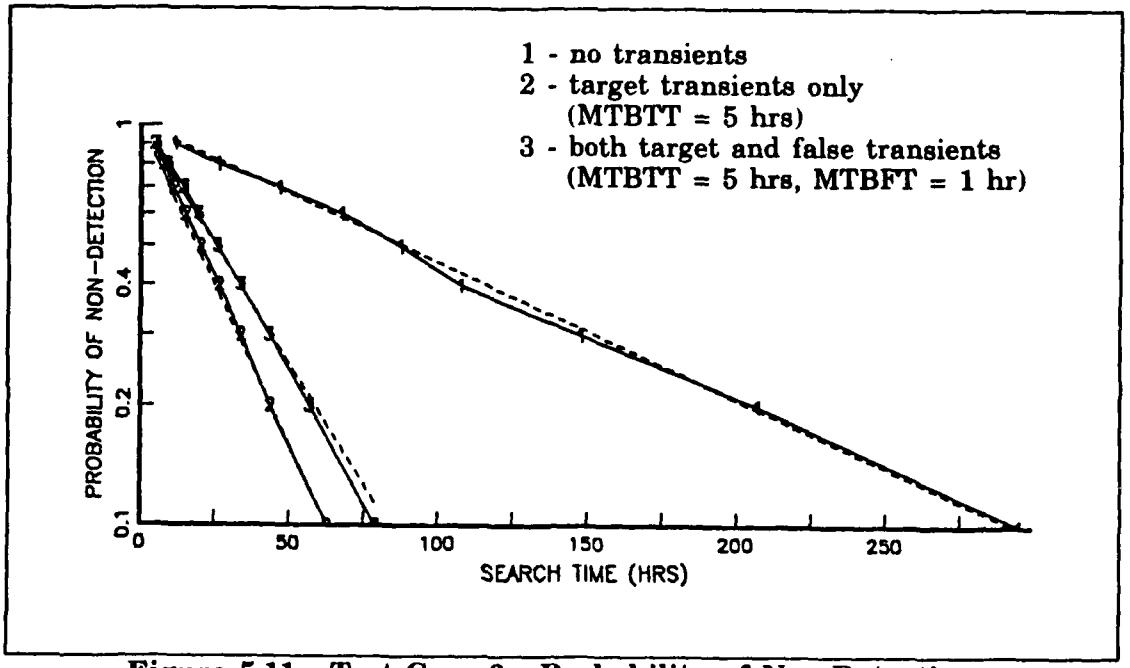

Figure 5.11. Test Case 3 - Probability of Non-Detection.

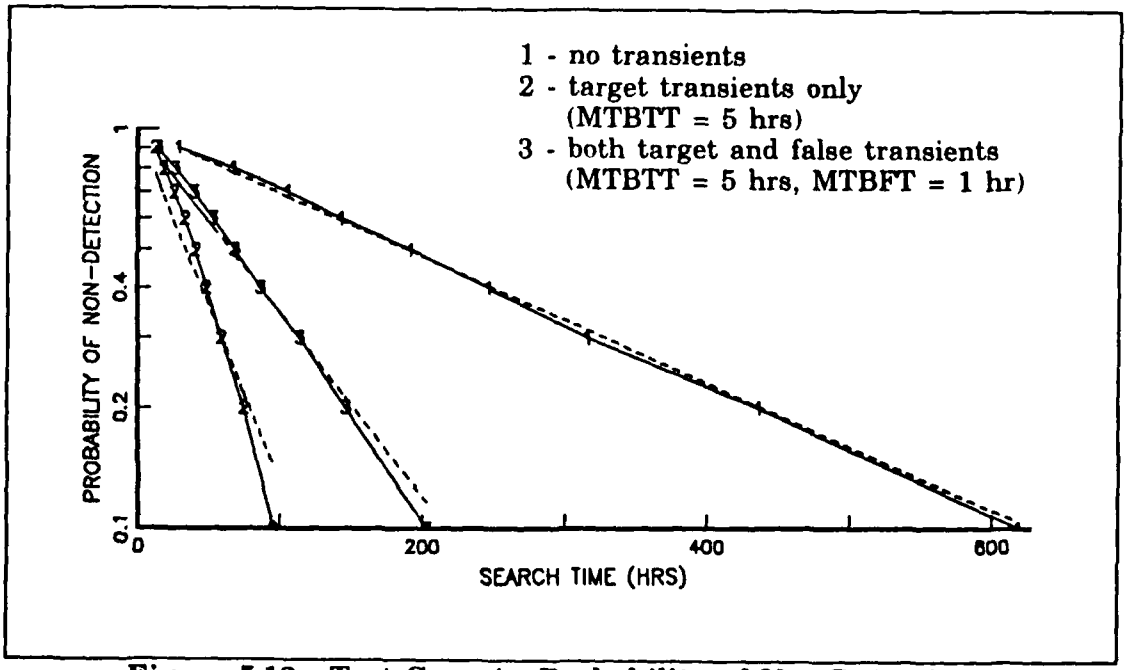

Figure 5.12. Test Case 4 - Probability of Non-Detection.

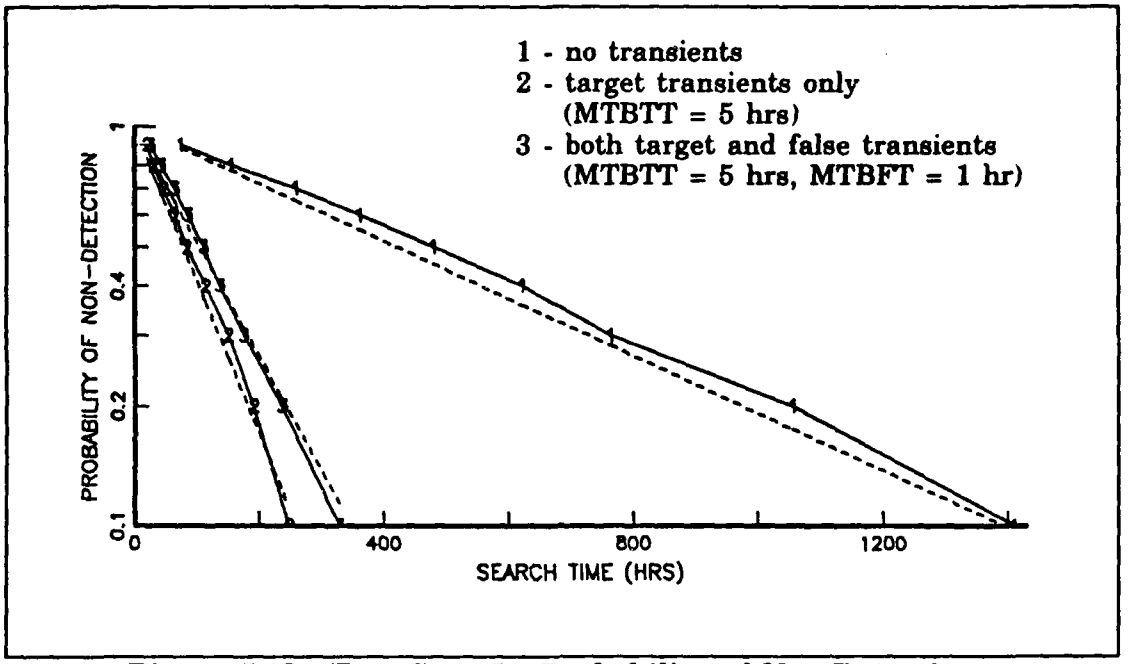

Figure 5.13. Test Case 5 - Probability of Non-Detection.

scenario involving a stationary target, a systematically moving searcher, and no transient sounds. However, by introducing target and false transient sounds to the scenario, the searcher is forced away from his systematic search by transient occurrence investigations. In essence, the search effort is degraded to a random search of the area.

Comparing the mean time to detection in the uniform distribution case to the mean time to detection in the random search case we have:

$$MTTD_{UNIFORM} = A / 4RV_E \qquad MTTD_{RANDOM} = A / 2RV_E$$

where,

A = area of search

R = radius of detection

V_E = effective search speed

Since the mean time to detection in the uniform distribution case is one half that in the random search case, the degradation in MTTD experienced with the introduction of false transient sounds should be greater than the degradation that occurred in the five test cases. Figure 5.14 indicates this contention. In addition, Figure 5.15, indicates the transition from a uniform search to a random search.

In Figure 5.15, the curve associated with no false transient sounds has deviated from the exponential fit as is indicated by the bowing of the curve on a natural log scale. Finally, Figure 5.15 also refutes the trend, established in the five test cases, that the gains achieved by target transient sounds were greater than the losses experienced from the introduction of false transient sounds at nearly equal rates of occurrence. The curve associated with both target and false transient sounds shifts back to the position of the curve associated with no transients and at higher time values surpasses it.

Excursion 2 deals with "deceptive" false transients. The main program of SUBTRAN was altered slightly to achieve a method for drawing the searcher away from the target in the event of a false transient sound emission. In Excursion 2, at the time of a false transient occurrence, the position of false transient occurrence (X_f,Y_f) was set at:

$$X_f = L - X_t$$

$$Y_f = L - Y_t$$

Using this method, the searcher was directed away from the target in most cases. The results of this excursion are graphically displayed in Figures 5.16 and 5.17. The figures clearly indicate the increased effect that false transient sounds have on the search effort. In particular, it is interesting to note

in Figure 5.17 that the curve associated with both target and false transient sounds has completely shifted past the curve associated with no transient sounds.

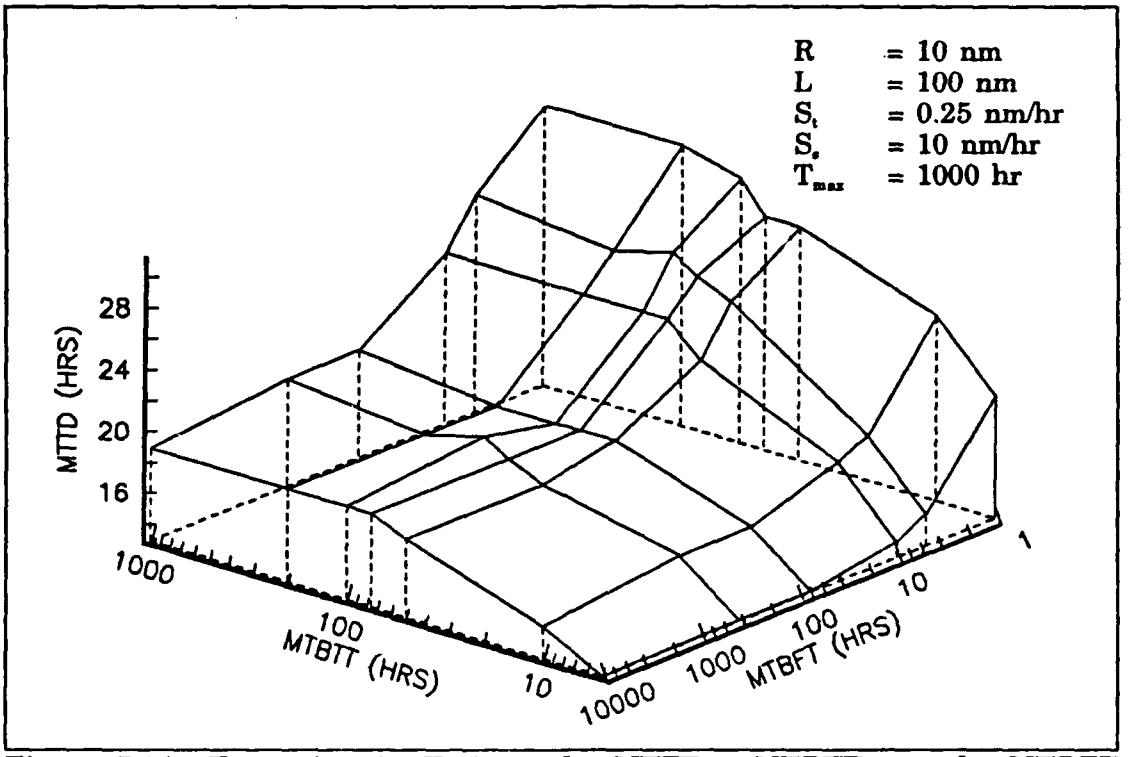

Figure 5.14. Excursion 1 - Estimated MTTD, MTBTT, and MTBFT Relationship.

TABLE V.7. EXCURSION 1 - ESTIMATED MTTD VALUES WITH 95% CI.

		MTBFT					
		1	5	10	75	400	NONE
	5	21±1	15±1	14±1	13±1	13±1	13±1
M	10	25±2	19±1	18±1	16±1	16±1	15±1
T	50	28±2	25±1	22±1	19±1	18±1	18±1
B	75	28±2	26±1	24±1	19±1	19±1	19±1
T	100	33±2	27±2	24±1	19±1	20±1	19±2
T	200	31±2	26±1	24±1	19±1	19±1	19±1
	1000	31±2	27±2	24±1	20±1	20±1	19±1
	NONE	-	-	-	-	-	19±1

* All values of MTBFT, MTBTT, and MTTD are in hours

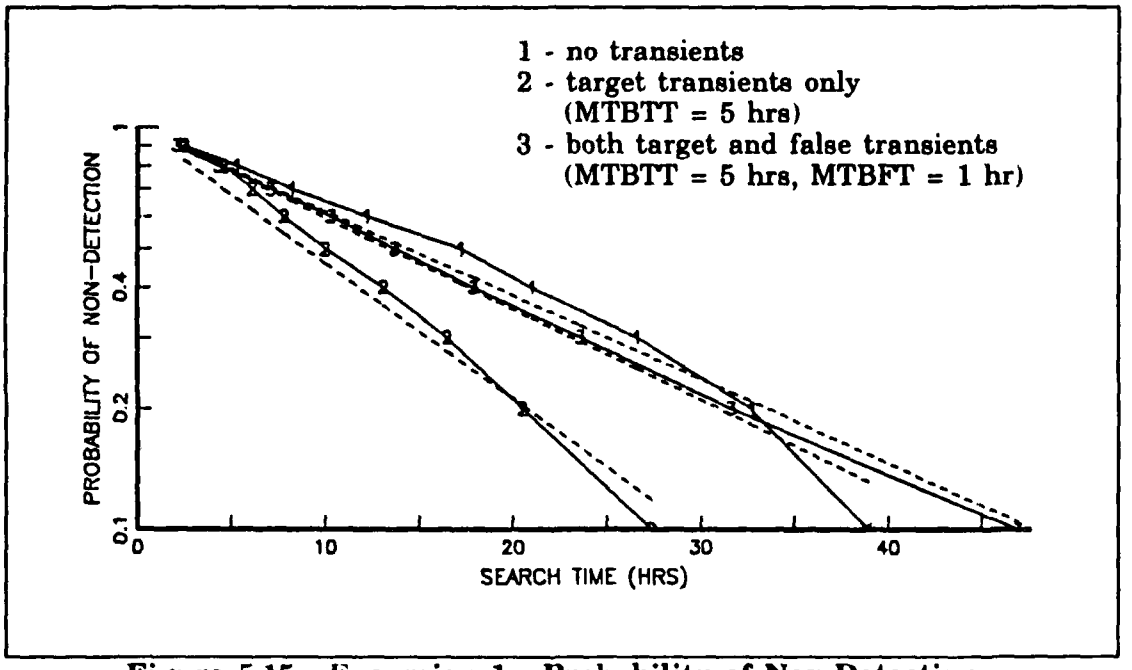

Figure 5.15. Excursion 1 - Probability of Non-Detection.

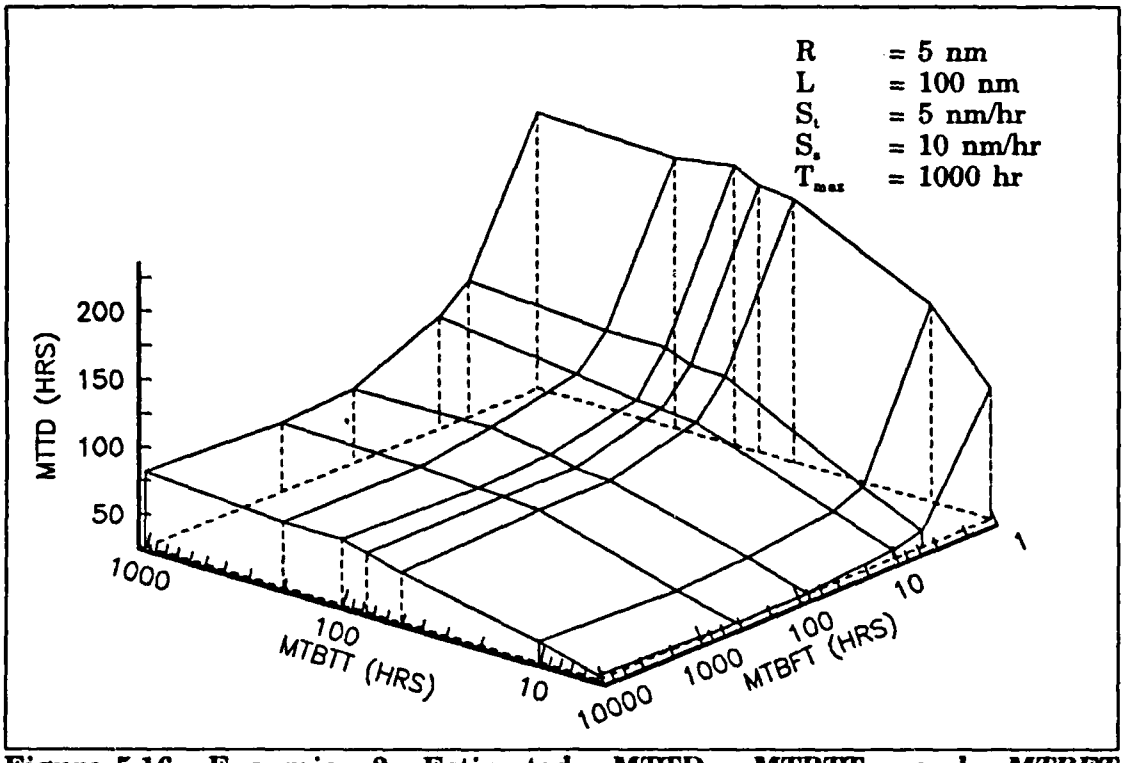

Figure 5.16. Excursion 2 - Estimated MTTD, MTBTT, and MTBFT Relationship.

TABLE V.8. EXCURSION 2 - ESTIMATED MTTD VALUES WITH 95% CI.

		MTBFT					
		1	5	10	75	400	NONE
	5	125±7	39±2	33±1	27±1	27±1	29±1
M	10	173±9	59±3	49±3	41±2	38±2	42±2
T	50	221±11	111±6	86±4	68±3	68±3	63±3
B	75	224±12	112±6	89±5	70±3	71±4	70±4
T	100	233±12	120±6	90±5	75±4	72±4	75±4
T	200	226±11	119±6	96±5	83±4	75±4	75±4
	1000	230±12	127±7	109±6	82±4	78±4	83±4
	NONE	-	-	-	-	-	85±5

* All values of MTBFT, MTBTT, and MTTD are in hours

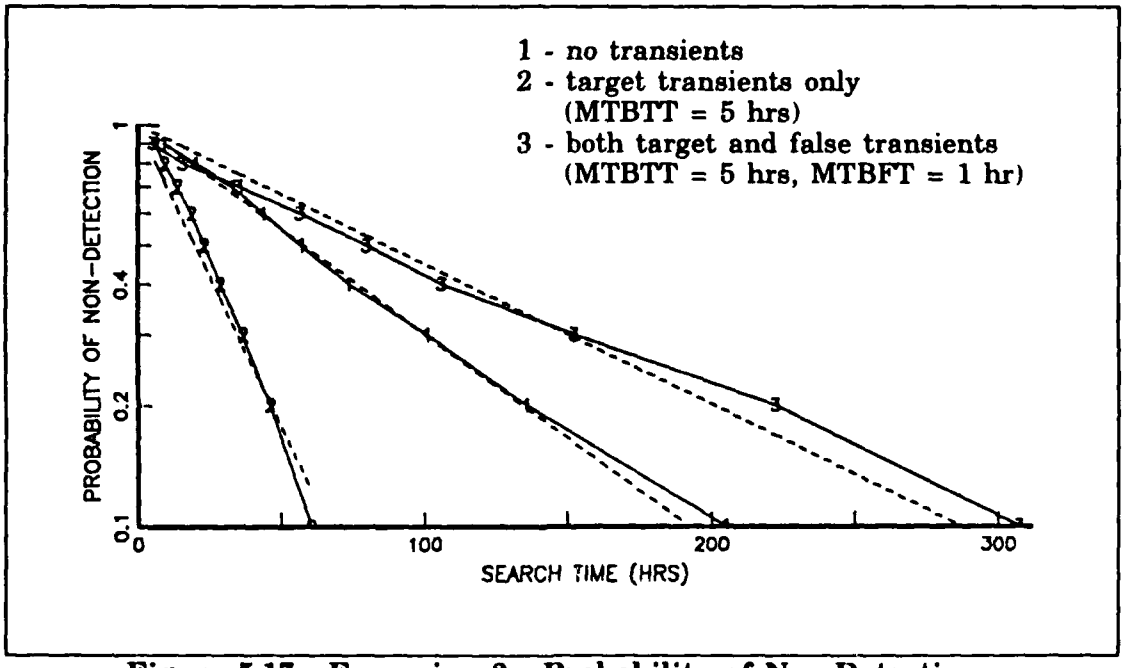

Figure 5.17. Excursion 2 - Probability of Non-Detection.

VI. CONCLUSIONS AND RECOMMENDATIONS FOR FURTHER WORK

Although this thesis does not provide an all-inclusive answer to the question of whether or not transient detection opportunities can be utilized by a searching submarine to increase its chance of gaining steady-state detection on a target submarine, it does establish a framework, through the simulation SUBTRAN, that may be useful in investigating how transient sounds affect the steady-state search effort for a given search scenario.

The modeling in SUBTRAN utilizes established methods. The target movement and transient occurrence models allow the user flexibility in defining the target's characteristics. The searcher movement model is restrictive in the sense that the searcher has no movement options in the search pattern or transient prosecution. The false transient occurrence model is designed to supplement the search scenario with randomly occurring ambient sounds that could be classified as target transient sounds by the searcher.

Major features of SUBTRAN that support it as a viable method for simulation of the transient environment search scenario presented in this thesis are:

- Considerable flexibility is available to the user in selection of search scenario input parameters.
- The simulation is designed for microcomputer implementation removing the "mainframe" requirement common among most submarine versus submarine simulation programs.
- The simulation search scenario is easy to set-up using the input data file provided.

- The output provided by SUBTRAN is concise and clearly highlights important results.

Improvements or additions SUBTRAN might include:

- Modeling target counter-detection of the searcher.
- Incorporating different types of target transients to include various duration lengths and FOMs.
- Investigating different searcher movement patterns.
- Including FOM target aspect and speed dependencies.
- Modeling target and searcher movement to include speed changes.

APPENDIX A

SUBTRAN USERS GUIDE

A. SYSTEM REQUIREMENTS

SUBTRAN requires both computer memory and disk space. The computer memory required is 126 Kbytes. The disk space, necessary for the two output files, varies with respect to the number of simulation replications performed. The disk space needed for a simulation run of 5000 is approximately 101 Kbytes, while the disk space needed for a simulation run of 1000 is approximately 27 Kbytes.

B. INPUT DATA FILE

SUBTRAN requires input parameters to be entered via an input data file. The input data file is specially formatted to aid the user in proper positioning of the input parameters. A sample of the input data file is contained in Appendix C.

SUBTRAN was not coded to check for improper input parameters. Careful attention is required when filling out the input data file to ensure input parameters are within specified range limits.

The input data file is broken up into 15 data fields. A description of the data requirements and allowed ranges follows:

1. Data Field 1

The input parameters contained in data field 1 are: random number seed (DSEED), length dimension of the search area (L), number of simulation replications (N), and the maximum search time (T_{max}).

DSEED - double precision variable range, [1,2147483646]

L - real variable / range, (0,999]

N - real variable / range, [1,5000]

T_{max} - real variable / range, [0,9999]

2. Data Field 2

The input parameter contained in data field 2 is the number of points to be entered for the steady-state propagation loss curve (NP_{pss}).

NP_{pss} - integer variable / range, [0,50]

3. Data Field 3

The input parameters contained in data field 3 are the steady-state propagation loss curve range values in increasing order (RN_{pss}) and the corresponding propagation loss values (P_{pss}). Due to the code structure for calculating the propagation loss curve, the values for any two sequential range inputs must be different. Also, the last propagation loss value must be the highest of all propagation loss values.

RN_{pss} - real array of size NP_{pss} / range, [0,999]

P_{pss} - real array of size NP_{pss} / range, [0,999]

4. Data Fields 4 and 5

The input parameters contained in data fields 4 and 5 are of the same structure as the parameters contained in data fields 2 and 3 respectively. Instead of steady-state propagation loss curve, the input corresponds to the transient propagation loss curve. The variables in data fields 4 and 5 corresponding to those in data fields 2 and 3 are:

NP_{ptr} replaces NP_{pss}

RN_{ptr} replaces RN_{pss}

P_{ptr} replaces P_{pss}

5. Data Field 6

The input parameters contained in data field 6 are the searcher's speed (S_s) and the target's speed (T_s). The speed parameters must never be set to zero.

S_s - real variable / range, (0,99]

S_t - real variable / range, (0,99]

6. Data Field 7

The input parameter contained in data field 7 is the number of points to be entered for the target's course distribution (NP_c).

NP_c - integer variable / range, [0,50]

7. Data Field 8

The input parameters contained in data field 8 are the target's course distribution values (C_c) in increasing order and the corresponding cumulative probability (P_c). The first value of the cumulative probability must be 0 and the last value of cumulative probability must be 1.

C_c - real array of size NP_c / range, [0,360]

P_c - real array of size P_c / range, [0,1]

8. Data Fields 9 and 10

The input parameters contained in data fields 9 and 10 are of the same structure as those contained in data fields 7 and 8 respectively. Instead of

target's course distribution, the input corresponds to the distribution of the time between target course changes. The variables in data fields 9 and 10 that correspond to those in data field 7 and 8 are:

NP_{tc} replaces NP_c

T_{tc} replaces C_c, range for T_{tc} is [0,99]

P_{tc} replaces P_c

9. Data Field 11

The input parameters contained in data field 11 are the transient occurrence flag (F_{tr}) and the transient occurrence rate parameter (RP_{tr}). Transient occurrence is allowed when the transient occurrence flag is set to a value of 1. Transients are not allowed when the flag is set to a value of 0. The value of the transient occurrence rate parameter must never be 0.

F_{tr} - integer variable / range, 0 or 1

RP_{tr} - real variable / range, [.000001,99]

10. Data Field 12

The input parameters contained in data field 12 are of the same structure as those input parameters contained in data field 11. Instead of transient occurrence, the input corresponds to false transient occurrence. The variables in data field 12 that correspond to those in input data field 11 are:

F_{ftr} replaces F_{tr}

RP_{ftr} replaces RP_{tr}

11. Data Field 13

The input parameters contained in input data field 13 are the Lambda-Sigma Jump error process rate parameter (RP_{lsj}) and the Lambda-Sigma Jump error process standard deviation (SD_{lsj}). The rate parameter must never be 0.

RP_{lsj} - real variable / range, [.000001,99]

SD_{lsj} - real variable / range, [0,99]

12. Data Field 14

The input parameters contained in input data field 14 are the steady-state figure-of-merit (FOM_{ss}) and the transient figure-of-merit (FOM_{tr}).

FOM_{ss} - real variable / range, [0,999]

FOM_{tr} - real variable / range, [0,999]

13. Data Field 15

The input parameters contained in input data field 15 are the track width flag (F_{tw}) and the track width (TW). The input track width is used in the simulation if the value of the track width flag is set at 1. The track width is set to the mean steady-state detection range if the value of the track width flag is set at 0.

F_{tw} - integer variable / range, 0 or 1

TW - real variable / range, [0,999]

C. PROGRAM EXECUTION

SUBTRAN is executed by running the SUBTRAN.EXE file. The user is prompted to enter the name of the input data file. The name of the file may be any proper filename and extension of total length less than or equal to 12

characters. After the input data file name is entered, the input parameters are printed at the user's terminal for verification. Simulation execution commences when the input parameters have been verified. Status of the simulation is printed at the user's terminal every time a tenth of the simulation is completed. When all replications have been run, output statistics are printed at the user's terminal and to the output data files.

D. OUTPUT DATA FILES

Two output data files are generated by SUBTRAN. The first data file, SUBTRAN.OUT, contains the input parameters and output statistics. The second data file, TIME.OUT, contains the times to detection ordered from shortest to longest. Times to detection with a value of zero are not included. In addition, simulation replications that did not result in steady-state detection before T_{max}, have time to detection set at T_{max}.

APPENDIX B

MAJOR VARIABLES USED IN SUBTRAN

The major variables used in SUBTRAN are provided in an alphabetical listing.

C_c = Course array of the targets course cumulative distribution curve in degrees.

C_r = Course the target will be on after the next boundary reflection in degrees.

C_s = Course of the searcher in degrees.

C_t = Course of the target in degrees.

DSEED = Random number seed.

E_{lsj} = Signal excess fluctuation level in dB.

F_{ftr} = False transient occurrence flag.

F_{tr} = Transient occurrence flag.

F_{tw} = Track width calculation flag.

FOM_{ss} = Steady-state figure-of-merit in dB.

FOM_{tr} = Transient figure-of-merit in dB.

L = Length dimension of the search area in nautical miles.

N = Number of simulation replications.

N_d = Number of replications in which steady-state detections occurred.

N_m = Number of replications in which no steady-state detection occurred before T_{max}.

N_z = Number of replications in which steady-state detection occurred at time zero.

NP_c = Number of points entered for the target's course cumulative distribution curve.

NP_{pss} = Number of points entered for the steady-state propagation loss curve.

NP_{ptr} = Number of points entered for the transient propagation loss curve.

NP_{tc} = Number of points entered for the target's time to course change cumulative distribution curve.

P_c = Probability array for the target's course cumulative distribution curve.

P_{pss} = Propagation loss array for the steady-state propagation loss curve in dB.

P_{ptr} = Propagation loss array for the transient propagation loss curve in dB.

P_{tc} = Probability array for the target's time to course change cumulative distribution curve.

R_d = Range array for transient detection ranges in nautical miles.

R_{ssd} = Steady-state detection range in nautical miles.

RN_{pss} = Range array for the steady-state propagation loss curve in nautical miles.

RN_{ptr} = Range array for the transient propagation loss curve in nautical miles.

RP_{ftr} = Rate parameter for the false transient occurrence distribution in hours^{-1}.

RP_{lsj} = Rate parameter for the Lambda-Sigma Jump process in hours^{-1}.

RP_{tr} = Rate parameter for the transient occurrence distribution in hours^{-1}.

S_s = Speed of the searcher in nautical miles per hour.

S_t = Speed of the target in nautical miles per hour.

SD_{lsj} = Standard deviation for the Lambda-Sigma Jump process in dB.

T_{lsj} = Time to the next signal excess fluctuation in hours.

T_{max} = The maximum search time for each replication in hours.

T_r = Time until the next target boundary reflection in hours.

T_{tc} = Time array for the target's time to course change cumulative distribution curve in hours.

TC_s = Time to the next searcher course change in hours.

TC_t = Time to the next target course change in hours.

TT_f = Time to the next false transient occurrence in hours.

TT_t = Time to the next transient occurrence in hours.

TW = Searcher's track width in nautical miles.

X_f = X component of the false transient occurrence position.

X_s = X component of the searcher's position.

X_t = X component of the target's position.

Y_f = Y component of the false transient occurrence position.

Y_s = Y component of the searcher's position.

Y_t = Y component of the target's position.

APPENDIX C

EXAMPLE SUBTRAN INPUT DATA FILE

```
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C                                                                               C
C   The following is the input file for SUBTRAN.                                C
C   All values should be right justified within                                 C
C   their designated field. Any line with a '*'                                 C
C   or a 'C' in it must not be deleted, success-                                C
C   ful reading of the input data depends on                                    C
C   it. If continued errors occur when the input                                C
C   is read by the main program, consult the                                    C
C   program documentation for the proper setup                                  C
C   of this file.                                                               C
C                                                                               C
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
*
1.      * * * * * * * * * * * * * * * * * * * *
        *     Columns          Input            *
        *     -------          -----            *
        *       3-15           DSEED      [d]   *
        *      18-30            L         [l]   *
        *      33-45            N         [n]   *
        *      48-60           Tmax       [t]   *
        * * * * * * * * * * * * * * * * * * * *
**ddddddddddddd**lllllllllll**nnnnnnnnnnnnn**ttttttttttttt**
           2855              100             1000              1000
*
2.      * * * * * * * * * * * * * * * * * * * *
        *     Columns          Input            *
        *     -------          -----            *
        *       3-15            NPpss     [n]   *
        * * * * * * * * * * * * * * * * * * * *
**nnnnnnnnnnnnn**  [NPpss cannot exceed 50]
              4
*
3.      * * * * * * * * * * * * * * * * * * * *
        *     Columns          Input            *
        *     -------          -----            *
        *       3-15         RNpss(NPp) [r]     *
        *      18-30         Ppss(NPp)  [p]     *
        * * * * * * * * * * * * * * * * * * * *
**rrrrrrrrrrrrr**ppppppppppppp**
              0               46
              6               70
             14               91
             18               92
```

```
*
4.      * * * * * * * * * * * * * * * * * * *
        *     Columns          Input          *
        *     -------          -----          *
        *       3-15            NPptr    [n] *
        * * * * * * * * * * * * * * * * * * *
**nnnnnnnnnnnnn**  [NPptr cannot exceed 50]
              2
*
5.      * * * * * * * * * * * * * * * * * * *
        *     Columns          Input          *
        *     -------          -----          *
        *       3-15          RNptr(NPp) [r] *
        *      18-30          Pptr(NPp)  [p] *
        * * * * * * * * * * * * * * * * * * *
**rrrrrrrrrrrrr**ppppppppppppp**
              0              50
             50             120
*
6.      * * * * * * * * * * * * * * * * * * *
        *     Columns          Input          *
        *     -------          -----          *
        *       3-15            St       [t] *
        *      18-30            Ss       [s] *
        * * * * * * * * * * * * * * * * * * *
**ttttttttttttt**sssssssssssss**
              5              10
*
7.      * * * * * * * * * * * * * * * * * * *
        *     Columns          Input          *
        *     -------          -----          *
        *       3-15            NPc      [n] *
        * * * * * * * * * * * * * * * * * * *
**nnnnnnnnnnnnn**  [NPc cannot exceed 50]
              6
*
8.      * * * * * * * * * * * * * * * * * * *
        *     Columns          Input          *
        *     -------          -----          *
        *       3-15          Cc(NPc)   [c] *
        *      18-30          Pc(NPc)   [p] *
        * * * * * * * * * * * * * * * * * * *
**ccccccccccccc**ppppppppppppp**
              0              0
             20             .2
             90             .5
            180             .8
            280             .9
            360              1
*
```

```
9.      * * * * * * * * * * * * * * * * * * * *
        *     Columns          Input            *
        *     -------          -----            *
        *       3-15            NPtc      [n]  *
        * * * * * * * * * * * * * * * * * * * *
**nnnnnnnnnnnnn**  [NPtc cannot exceed 50]
               5
*
10.     * * * * * * * * * * * * * * * * * * * *
        *     Columns          Input            *
        *     -------          -----            *
        *       3-15         Ttc(NPtc)   [t]  *
        *      18-30         Ptc(NPtc)   [p]  *
        * * * * * * * * * * * * * * * * * * * *
**ttttttttttttt**ppppppppppppp**
               0                0
             .42               .2
             .89               .4
            1.43               .6
            2.04                1
*
11.     * * * * * * * * * * * * * * * * * * * *
        *     Columns          Input            *
        *     -------          -----            *
        *       3-15            Ftr       [n]  *
        *      18-30            RPtr      [r]  *
        * * * * * * * * * * * * * * * * * * * *
**nnnnnnnnnnnnn**rrrrrrrrrrrrr**
               1               .1
*
12.     * * * * * * * * * * * * * * * * * * * *
        *     Columns          Input            *
        *     -------          -----            *
        *       3-15           Fftr       [n]  *
        *      18-30           RPftr      [r]  *
        * * * * * * * * * * * * * * * * * * * *
**nnnnnnnnnnnnn**rrrrrrrrrrrrr**
               0                6
*
13.     * * * * * * * * * * * * * * * * * * * *
        *     Columns          Input            *
        *     -------          -----            *
        *       3-15           RPlsj      [r]  *
        *      18-30           SDlsj      [s]  *
        * * * * * * * * * * * * * * * * * * * *
**rrrrrrrrrrrrr**sssssssssssss**
               3                6
*
```

```
14.     * * * * * * * * * * * * * * * * * * * *
        *      Columns          Input           *
        *      -------          -----           *
        *        3-15           FOMss      [s]  *
        *       18-30           FOMtr      [t]  *
        * * * * * * * * * * * * * * * * * * * *
**sssssssssssss**ttttttttttttt**
              56              90
*
15.     * * * * * * * * * * * * * * * * * * * *
        *      Columns          Input           *
        *      -------          -----           *
        *        3-15           Ftw        [f]  *
        *       18-30           TW         [s]  *
        * * * * * * * * * * * * * * * * * * * *
**fffffffffffff**sssssssssssss**
               0              10
```

APPENDIX D

EXAMPLE SUBTRAN OUTPUT STATISTICS

```
* * * * * * * * * * *
*  SUBTRAN INPUT   *
* * * * * * * * * * *
```

SIMULATION PARAMETERS

Replications = 1000
Random Seed = 2855

TRANSIENT PARAMETERS

(1 - Yes / 0 - No)

Real Transients:
  Allowed = 1
  Rate Par. = 0.1
False Transients:
  Allowed = 0
  Rate Par. = 6.0

SEARCH PARAMETERS

Search Area = 10000
Search Time = 1000
Track Width = calc.

SIGNAL ERROR PARAMETERS

Rate Par. = 3.00
Std. Dev. = 6.00

SPEED PARAMETERS

S. Speed = 10.00
T. Speed = 5.00

STEADY-STATE FIGURE OF MERIT / PROPAGATION LOSS PARAMETERS

Steady-state Figure of Merit = 56.00

Range Point :	0.00	Prop. Loss :	46.00
Range Point :	6.00	Prop. Loss :	70.00
Range Point :	14.00	Prop. Loss :	91.00
Range Point :	18.00	Prop. Loss :	92.00

TRANSIENT FIGURE OF MERIT / PROPAGATION LOSS PARAMETERS

Transient Figure of Merit = 90.00

Range Point :	0.00	Prop. Loss :	50.00
Range Point :	50.00	Prop. Loss :	120.00

TARGET COURSE PARAMETERS

Course:	0.	Prob: 0.0	Time to Change:	0.00	Prob: 0.0
Course:	20.	Prob: 0.2	Time to Change:	0.42	Prob: 0.2
Course:	90.	Prob: 0.5	Time to Change:	0.89	Prob: 0.4
Course:	180.	Prob: 0.8	Time to Change:	1.43	Prob: 0.6
Course:	280.	Prob: 0.9	Time to Change:	2.04	Prob: 1.0
Course:	360.	Prob: 1.0			

* * * * * * * * * * * *
* SUBTRAN OUTPUT DATA *
* * * * * * * * * * * *

NO. OF REPS. WITH DETECTION TIME OF ZERO 0
NO. OF REPS. WITH DETECTION TIME GREATER THAN Tmax ... 0
CALCULATED TRACK WIDTH VALUE 5.00

TIME TO DETECTION STATISTICS

Mean = 644.19 +/-31.13 (95% CI)
Std. Dev. = 598.44
1/Lambda = 602.55
R-Squared = 0.99

Prob. of Det.	By time	Prob. of Det.	By time
0.1	74.5	0.6	621.1
0.2	154.3	0.7	762.7
0.3	257.8	0.8	1054.4
0.4	360.4	0.9	1410.0
0.5	478.6	1.0	3411.8

APPENDIX E

STEADY-STATE DETECTION DERIVATION

As a result of the discrete event timing used in SUBTRAN, steady-state detection is assessed by computing the time it would take the range between the target and the searcher to decrease to the value of the steady-state detection range. The calculations used in the steady-state detection assessment computation are outlined below.

Define,

X_{to} - Starting X position of the target.

Y_{to} - Starting Y position of the target.

$X_t(t)$ - X position of the target at time t.

$Y_t(t)$ - Y position of the target at time t.

S_{tx} - X component of target speed.

S_{ty} - Y component of target speed.

R_d - Steady-state detection range.

$X_{so}, Y_{so}, X_s(t), Y_s(t), S_{sx}, S_{sy}$ - Be equivalent variables for the searcher.

At any time t, the X and Y positions of the target and searcher can be written as follows;

$$X_t(t) = X_{to} + tS_{tx} \text{ and } Y_t(t) = Y_{to} + tS_{ty} \quad \text{(E.1)}$$

$$X_s(t) = X_{so} + tS_{sx} \text{ and } Y_s(t) = Y_{so} + tS_{sy} \quad \text{(E.2)}$$

$$R(t) = ([X_s(t) - X_t(t)]^2 + [Y_s(t) - Y_t(t)]^2)^{0.5} \quad \text{(E.3)}$$

Combining equations (1), (2), (3) and setting $R(t) = R_d$ we solve for t.

$$t = \frac{-B \pm (B^2 - 4AC)^{0.5}}{2A} \quad \text{(E.4)}$$

where,

$$A = S^2_{sx} + S^2_{tx} - 2S_{sx}S_{tx} + S^2_{sy} + S^2_{ty} - 2S_{sy}S_{ty}$$

$$B = [2X_{so}S_{sx} + 2X_{to}S_{tx} - 2X_{so}S_{tx} - 2X_{to}S_{sx}] + [2Y_{so}S_{sy} + 2Y_{to}S_{ty} - 2Y_{so}S_{ty} - 2Y_{to}S_{sy}]$$

$$C = X^2_{so} + X^2_{to} - 2X_{so}X_{to} + Y^2_{so} + Y^2_{to} - 2Y_{so}Y_{to} - Rd^2$$

The time to steady-state detection is the minimum of the positive values of t obtained from equation (E.4).

Four special solution cases exist in which the time to steady-state detection must be investigated separately. They are: when both values of t are negative, when one t value is positive and the other is negative, when $(B^2 - 4AC) < 0$, and when A=0.

In the case where both t values are negative, the steady-state detection opportunity has already passed and detection does not occur.

When one t value is positive and the other is negative, the range between the searcher's and target's initial positions is less than the steady-state detection range. As a result, detection occurs and the time to detection is 0.

In the two cases $(B^2 - 4AC) < 0$ and $A = 0$ the range between the searcher and the target never decreases to the steady-state detection range. As a result, detection does not occur.

APPENDIX F

FORTRAN CODE FOR SUBTRAN

The fortran code for SUBTRAN is divided into the following parts:

- Main Program (SUBTRAN)
- Input Subroutine (IN)
- Propagation Loss Subroutine (PLOSS)
- Target Course Subroutine (TGCR)
- Target Transient Subroutine (TGTR)
- Searcher Course Subroutine (SRCR)
- Transient Course Subroutine (TRCR)
- Steady-State Detection Subroutine (SSDET)
- Transient Detection Subroutine (TRDET)
- Lambda-Sigma Jump Subroutine (LSJ)
- Position Subroutine (POSIT)
- Boundary Reflection Subroutine (REFLCT)
- Track Subroutine (TRACK)
- Output Subroutine (OUT)
- Hold Screen Subroutine (HSCRN)
- Uniform Random Number Generator Subroutine (LRNDPC)
- Normal Random Number Generator Subroutine (LNORPC)
- Gamma Random Number Generator Subroutine (LGAMPC)

```
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C                                                         C
C                    PROGRAM SUBTRAN                      C
C                                                         C
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C
      PROGRAM SUBTRAN
C
C     * SUBTRAN requires the subroutines: FALTR, HSCRN, IN,
C       LSJ, OUT, PLOSS, POSIT, REFLCT, SRCR, SSDET, TGCR,
C       TGTR, TRACK, TRCR, TRDET, LRNDPC, LNORPC, LGAMPC.
C
C.....Declare variables and dimension arrays.
C
      CHARACTER*1 ANS
      DOUBLE PRECISION DSEED
      INTEGER N, NPpss, NPptr, NPc, NPtc, Ftr, Fftr, Ftw,
     1        Nc, Nd, Nz, Nm, Fstc, Fsc, Fssd, Ftd, HFsc
      REAL L, Tmax, RNpss(50), Ppss(50), RNptr(50),
     1     Pptr(50), St, Ss, Cc(50), Pc(50), Ttc(50),
     1     Ptc(50), RPtr, RPftr, RPlsj, SDlsj, FOMss, FOMtr,
     1     TW, Rd(7), T, Xs, Ys, Xt, Yt, U(2), Ct, TCt, TTt,
     1     Cs, TCs, Xf, Yf, TTf,  Elsj, Tlsj, Rssd, Cr, Tr,
     1     Tmin, Tssd, Td(5000), HXs, HYs, HCs, HTCs, HTh
C
C.....Read input data from the user-defined input data file.
C
      CALL IN(DSEED, L, N, Tmax, NPpss, RNpss, Ppss, NPptr,
     1        RNptr, Pptr, St, Ss, NPc, Cc, Pc, NPtc, Ttc,
     1        Ptc, Ftr, RPtr, Fftr, RPftr, RPlsj, SDlsj,
     1        FOMss, FOMtr, Ftw, TW)
C
C.....Ask if input parameters are correct.
C
      WRITE(*,14)
      READ(*,15)ANS
      IF((ANS .NE. 'Y') .AND. (ANS .NE. 'y'))THEN
         WRITE(*,24)
         CALL HSCRN
         GOTO 60
      END IF
      WRITE(*,34)
C
C.....If the track width is not specified by the user
C     (Ftw=0) then compute the searcher's mean track width
C     (TW) in nautical miles using the steady state figure
C     of merit (FOMss) and the propagation loss curve.
C
      IF(Ftw .EQ. 0)THEN
         CALL PLOSS(NPpss, RNpss, Ppss, FOMss, Rd)
         TW=2*Rd(1)
      END IF
```

```
C
C.....Start counter for number of simulation replications
C     (Nc). Initialize detection counter (Nd), detection at
C     time zero counter (Nz), and detection not before Tmax
C     counter (Nm).
C
      Nc=1
      Nd=0
      Nz=0
      Nm=0
C
C.....Assign the initial value of the simulation time (T) in
C     hours to zero. This is the starting place for
C     successive runs. Set the searcher's transient course
C     flag (Fstc) to zero.
C
10    T=0
      Fstc=0
C
C.....Set the initial position of the searcher (Xs,Ys) in
C     the lower left-hand corner of the search area one-half
C     a track width's distance from the edges of the search
C     area.
C
      Xs=TW/2
      Ys=TW/2
C
C.....Set the initial position of the target (Xt,Yt)
C     uniformly distributed in the search area.
C
      CALL LRNDPC(DSEED, U, 2)
      Xt=L*U(1)
      Yt=L*U(2)
C
C.....Set initial target parameters of course in degrees
C     (Ct), time to next course change in hours (TCt), time
C     to next transient in hours (TTt). Target's speed is
C     assumed constant. If the target's transient flag (Ftr)
C     is 1, transients are allowed in the simulation.
C
      CALL TGCR(DSEED, NPc, Cc, Pc, NPtc, Ttc, Ptc, Ct, TCt)
      IF(Ftr .EQ. 1)THEN
         CALL TGTR(DSEED, RPtr, TTt)
      ELSE
         TTt=2*Tmax
      END IF
C
C.....Set the searcher's course flag (Fsc) to 1.
C     Set initial searcher parameters of course in degrees
C     (Cs), time to next course change in hours (TCs).
C     Searcher's speed is assumed constant.
C
```

```
      Fsc=1
      CALL SRCR(Fsc, Xs, Ys, Ss, TW, L, Cs, TCs)
C
C.....If false transient detections are allowed (Fftr=1),
C     then set the initial false transient parameters of
C     bearing to the transient in degrees (Cf), time to next
C     false transient (TTf).
C
      IF(Fftr .EQ. 1)THEN
         CALL FALTR(DSEED, RPftr, L  Xf, Yf, TTf)
      ELSE
         TTf=2*Tmax
      END IF
C
C.....Set initial signal error level (Elsj) and time time to
C     next error level fluctuation (Tlsj).
C
      CALL LSJ(DSEED, RPlsj, SDlsj, Elsj, Tlsj)
C
C.....Compute the steady state detection range.
C
      FOM=FOMss+Elsj
      CALL PLOSS(NPpss, RNpss, Ppss, FOM, Rd)
      Rssd=Rd(1)
C
C.....Compute the time in hours until the target must be
C     reflected off the search area boundry (Tr) if present
C     course and speed are maintained. Also compute the
C     course after reflection (Cr).
C
20    CALL REFLCT(L, Ct, St, Xt, Yt, Cr, Tr)
C
C.....Compute the minimum time (Tmin) until the next event.
C
30    Tmin=MIN((Tmax-T),TCt,TTt,TCs,TTf,Tlsj,Tr)
C
C.....Check to see if steady state detection occurs before
C     Tmin. If so stop simulation run (Fssd=1). Time to
C     steady state detection (Tssd) is computed if detection
C     occurs.
C
      CALL SSDET(Xt, Yt, St, Ct, Xs, Ys, Ss, Cs, Rssd, Fssd,
     1           Tssd)
      IF ((Fssd .EQ. 1) .AND. (Tssd .LE. Tmin))THEN
          IF((T .EQ. 0) .AND. (Tssd .EQ. 0))THEN
             Nz=Nz+1
             GOTO 50
          ELSE
             GOTO 40
          END IF
      END IF
C
```

```
C.....Calculate the new searcher and target positions based
C     on Tmin.
C
      CALL POSIT(Tmin, Xt, Yt, Ct, St, Xs, Ys, Cs, Ss)
C
C.....Decrement the time values and check to see what the
C     next event is .
C
      T=T+Tmin
      TCt=TCt-Tmin
      TTt=TTt-Tmin
      TCs=TCs-Tmin
      TTf=TTf-Tmin
      Tlsj=Tlsj-Tmin
      Tr=Tr-Tmin
C
C.....Route to various updates based on which of the
C     following events is due to occur.
C                  1. Maximum search time is up. [T >= Tmax]
C                  2. Target course change.      [TCt = 0]
C                  3. Target transient.          [TTt = 0]
C                  4. Searcher course change.    [TCs = 0]
C                  5. False transient.           [TTf = 0]
C                  6. Signal error fluctuation.  [Tlsj = 0]
C                  7. Target boundry reflection. [Tr = 0]
C
C --  Update 1.
C
      IF(T .GE. Tmax)THEN
         Nm=Nm+1
         GOTO 50
      END IF
C
C --  Update 2.
C
      IF(TCt .EQ. 0)THEN
         CALL TGCR(DSEED, NPc, Cc, Pc, NPtc, Ttc, Ptc, Ct,
     1              TCt)
         GOTO 20
      END IF
C
C --  Update 3.
C
      IF(TTt .EQ. 0)THEN
         CALL TGTR(DSEED, RPtr, TTt)
         FOM=FOMtr+Elsj
         CALL PLOSS(NPptr, RNptr, Pptr, FOM, Rd)
         CALL TRDET(Xt, Yt, Xs, Ys, Rd, Ftd)
         IF(Ftd .NE. 1)THEN
            GOTO 30
         ELSE
            Fstc=1
```

```
            CALL TRCR(Fsc, Fstc, Xt, Yt, Xs, Ys, Ss, L, TW,
     1                HFsc, HXs, HYs, HCs, HTCs, HTh, Cs,
     1                TCs)
            GOTO 30
         END IF
       END IF
C
C --   Update 4.
C
       IF(TCs .EQ. 0)THEN
         IF(Fstc .EQ. 0)THEN
            CALL SRCR(Fsc, Xs, Ys, Ss, TW, L, Cs, TCs)
            GOTO 30
         ELSE
            CALL TRCR(Fsc, Fstc, Xt, Yt, Xs, Ys, Ss, L, TW,
     1                HFsc, HXs, HYs, HCs, HTCs, HTh, Cs,
     1                TCs)
            GOTO 30
         END IF
       END IF
C
C --   Update 5.
C
       IF(TTf .EQ. 0)THEN
         FOM=FOMtr+Elsj
         CALL PLOSS(NPptr, RNptr, Pptr, FOM, Rd)
         CALL TRDET(Xf, Yf, Xs, Ys, Rd, Ftd)
         IF(Ftd .NE. 1)THEN
            CALL FALTR(DSEED, RPftr, L, Xf, Yf, TTf)
            GOTO 30
         ELSE
            Fstc=1
            CALL TRCR(Fsc, Fstc, Xf, Yf, Xs, Ys, Ss, L, TW,
     1                HFsc, HXs, HYs, HCs, HTCs, HTh, Cs,
     1                TCs)
            CALL FALTR(DSEED, RPftr, L, Xf, Yf, TTf)
            GOTO 30
         END IF
       END IF
C
C --   Update 6.
C
       IF(Tlsj .EQ. 0)THEN
         CALL LSJ(DSEED, RPlsj, SDlsj, Elsj, Tlsj)
         FOM=FOMss+Elsj
         CALL PLOSS(NPpss, RNpss, Ppss, FOM, Rd)
         Rssd=Rd(1)
         GOTO 30
       END IF
C
C --   Update 7.
C
```

```
      IF(Tr .EQ. 0)THEN
         Ct=Cr
         GOTO 20
      END IF
C
C.....Store times to detection (Td). Update detection
C     counter (Nd). Update replication counter (Nc). Route
C     for next replication run if required. Print to the
C     screen a message every 100 replications.
C
40    Nd=Nd+1
      Td(Nd)=T+Tssd
50    Nc=Nc+1
      IF((INT((Nc-1)/(N/10))*(N/10)) .EQ. (Nc-1))THEN
            WRITE(*,44)(Nc-1)
      END IF
      IF(Nc .LE. N)GOTO 10
C
C.....Compute the output statistics and print to the user's
C     terminal and output file.
C
      CALL OUT(Nd, Nz, Nm, Ftw, TW, Tmax, Td)
C
C.....Print a closing message to the user's terminal.
C
      WRITE(*,54)
C
C.....Read format statements.
C
15    FORMAT(A)
C
C.....Write format statements.
C
14    FORMAT(23(/),1X,'Are the input parameters correct?',
     1       '(Y/N)'/)
24    FORMAT(23(/),27X,'EXECUTION TERMINATED BY USER'/
     1       27X,28('-'),13(/))
34    FORMAT(23(/),1X,'Executing....'/)
44    FORMAT(5X,I4,' replications complete.')
54    FORMAT(23(/),32X,8('* ')/32X,'*   SUBTRAN   *'/
     1       32X,8('* ')///
     1       30X,'Simulation Complete',9(/))
60    END
```

```
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C                                                           C
C                       SUBROUTINE IN                       C
C                                                           C
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C
      SUBROUTINE IN(DSEED, L, N, Tmax, NPpss, RNpss, Ppss,
     1              NPptr, RNptr, Pptr, St, Ss, NPc, Cc, Pc,
     1              NPtc, Ttc, Ptc, Ftr, RPtr, Fftr, RPftr,
     1              RPlsj, SDlsj, FOMss, FOMtr, Ftw, TW)
C
C     Variables passed to INPUT are: none
C     Variables returned by INPUT are: all in subroutines
C                                      call list
C
C     * IN requires the subroutine HSCRN.
C
C.....Define variables. Dimension arrays.
C
      CHARACTER*12 FILEIN, FILEOUT
      DOUBLE PRECISION DSEED
      INTEGER N, NPpss, NPptr, NPc, NPtc, Ftr, Fftr, Ftw,
     1        IHOLD
      REAL L, Tmax, RNpss(50), Ppss(50), RNptr(50),
     1     Pptr(50), St, Ss, Cc(50), Pc(50), Ttc(50),
     1     Ptc(50), RPtr, RPftr, RPlsj, SDlsj, FOMss, FOMtr,
     1     TW
C
C.....Write to the user's terminal an intoruction message.
C
      WRITE(*,14)
      CALL HSCRN
C
C.....Read the name of the input data file from the user's
terminal. Assign
C     the name of the output file to be used to store the
input data.
C
      WRITE(*,24)
      READ(*,15)FILEIN
      FILEOUT='SUBTRAN.OUT'
C
C.....Open the user-defined input data file and the assigned
C     output data file.
C
      OPEN(10,FILE=FILEIN)
      OPEN(20,FILE=FILEOUT)
C
C.....Read the 15 data fields from the input data file.
C
C --  Data field 1. [DSEED,L,N,Tmax]
C
```

```
      READ(10,25)DSEED,L,N,Tmax
C
C --  Data field 2. [NPpss]
C
      READ(10,35)NPpss
C
C --  Data field 3. [RNpss(NPp),Ppss(NPp)]

C
      DO 10 I=1,NPpss
         READ(10,45)RNpss(I),Ppss(I)
10    CONTINUE
C
C --  Data field 4. [NPptr]
C
      READ(10,35)NPptr
C
C --  Data field 5. [RNptr(NPp),Pptr(NPp)]

C
      DO 20 I=1,NPptr
         READ(10,45)RNptr(I),Pptr(I)
20    CONTINUE
C
C --  Data field 6. [St,Ss]
C
      READ(10,55)St,Ss
C
C --  Data field 7. [NPc]
C
      READ(10,35)NPc
C
C --  Data field 8. [Cc(NPc),Pc(NPc)]
C
      DO 30 I=1,NPc
         READ(10,45)Cc(I),Pc(I)
30    CONTINUE
C
C --  Data field 9. [NPtc]
C
      READ(10,35)NPtc
C
C --  Data field 10. [Ttc(NPtc),Ptc(NPtc)]
C
      DO 40 I=1,NPtc
         READ(10,45)Ttc(I),Ptc(I)
40    CONTINUE
C
C --  Data field 11. [Ftr,RPtr]
C
      READ(10,65)Ftr,RPtr
C
```

```
C --   Data field 12. [Fftr,RPftr]
C
       READ(10,65)Fftr,RPftr
C
C --   Data field 13. [RPlsj,SDlsj]
C
       READ(10,55)RPlsj,SDlsj
C
C --   Data field 14. [FOMss,FOMtr]
C
       READ(10,55)FOMss,FOMtr
C
C --   Data field 15. [Ftw,TW]
C
       READ(10,65)Ftw,TW
C
C.....Print to the user's terminal and the output data file
C      a verification of the input. There are three screens
C      of information.
C                  1. Miscellaneous data.
C                  2. Propagation loss curve - steady state
C                                              (points).
C                  3. Propagation loss curve - transient
C                                              (points).
C                  4. Target's course and time to course
C                     change curves (points).
C
C --   Screen 1.
C
       WRITE(*,4)
       WRITE(*,34)
       WRITE(20,34)
       WRITE(*,44)N,(L**2),DSEED,Tmax
       WRITE(20,44)N,(L**2),DSEED,Tmax
       IF(Ftw .EQ. 1)THEN
          WRITE(*,54)TW
          WRITE(20,54)TW
       ELSE
          WRITE(*,64)
          WRITE(20,64)
       END IF
       WRITE(*,74)RPlsj,SDlsj,Ftr,RPtr,Fftr,Ss,RPftr,St
       WRITE(20,74)RPlsj,SDlsj,Ftr,RPtr,Fftr,Ss,RPftr,St
       CALL HSCRN
C
C --   Screen 2.
C
       WRITE(*,4)
       WRITE(*,84)FOMss
       WRITE(20,84)FOMss
       IHOLD=0
       DO 50 I=1,NPpss
```

```
            IHOLD=IHOLD+1
            WRITE(*,94)RNpss(I),Ppss(I)
            WRITE(20,94)RNpss(I),Ppss(I)
            IF((I .EQ. 15) .OR. (I .EQ. 30) .OR. (I .EQ. 45))
     1      THEN
               WRITE(*,*)
               WRITE(*,*)
               CALL HSCRN
               WRITE(*,104)
               IHOLD=0
            END IF
50       CONTINUE
         IF((I .NE. 16) .OR. (I .NE. 31) .OR. (I .NE. 46))THEN

            DO 60 I=1,19-IHOLD
               WRITE(*,*)
60          CONTINUE
            CALL HSCRN
         END IF
         WRITE(20,154)
C
C --     Screen 3.
C
         WRITE(*,4)
         WRITE(*,86)FOMtr
         WRITE(20,86)FOMtr
         IHOLD=0
         DO 70 I=1,NPptr
            IHOLD=IHOLD+1
            WRITE(*,94)RNptr(I),Pptr(I)
            WRITE(20,94)RNptr(I),Pptr(I)
            IF((I .EQ. 15) .OR. (I .EQ. 30) .OR. (I .EQ. 45))
     1      THEN
               WRITE(*,*)
               WRITE(*,*)
               CALL HSCRN
               WRITE(*,104)
               IHOLD=0
            END IF
70       CONTINUE
         IF((I .NE. 16) .OR. (I .NE. 31) .OR. (I .NE. 46))THEN
            DO 80 I=1,19-IHOLD
               WRITE(*,*)
80          CONTINUE
            CALL HSCRN
         END IF
         WRITE(20,154)
C
C --     Screen 4.
C
         WRITE(*,4)
         WRITE(*,114)
```

```
      WRITE(20,114)
      IHOLD=0
      DO 90 I=1,MAX(NPc,NPtc)
         IHOLD=IHOLD+1
         IF((I .LE. NPc) .AND. (I .LE. NPtc))THEN
            WRITE(*,124)Cc(I),Pc(I),Ttc(I),Ptc(I)
            WRITE(20,124)Cc(I),Pc(I),Ttc(I),Ptc(I)
            GOTO 100
         END IF
         IF(I .GT. NPtc)THEN
            WRITE(*,134)Cc(I),Pc(I)
            WRITE(20,134)Cc(I),Pc(I)
            GOTO 100
         END IF
         IF(I .GT. NPc)THEN
            WRITE(*,144)Ttc(I),Ptc(I)
            WRITE(20,144)Ttc(I),Ptc(I)
            GOTO 100
         END IF
100      IF((I .EQ. 18) .OR. (I .EQ. 36))THEN
            WRITE(*,*)
            CALL HSCRN
            WRITE(*,104)
            IHOLD=0
         END IF
90    CONTINUE
      IF((I .NE. 19) .OR. (I .NE. 37))THEN
         DO 110 I=1,20-IHOLD
            WRITE(*,*)
110      CONTINUE
         CALL HSCRN
      END IF
      WRITE(20,154)
C
C.....Read format statements.
C
15    FORMAT(A)
25    FORMAT(23(/),2X,D13.0,2X,F13.0,2X,I13,2X,F13.0)
35    FORMAT(7(/),2X,I13,8(/))
45    FORMAT(2X,F13.0,2X,F13.0)
55    FORMAT(8(/),2X,F13.0,2X,F13.0)
65    FORMAT(8(/),2X,I13,2X,F13.0)
C
C.....Write format statements.
C
4     FORMAT(23(/))
14    FORMAT(23(/),20X,'Submarine Transient Detection ',
     1       'Simulation',5(/),32X,8('* ')/32X,
     1       '*   SUBTRAN   *'/32X,8('* '),5(/),29X,
     1       'J. Brad Kratovil LT,USN'/35X,'26 July ',
     1       '1988',6(/))
24    FORMAT(23(/),1X,'ENTER THE NAME OF THE INPUT DATA ',
```

```
     1       'FILE.'/)
34    FORMAT(29X,10('* ')/29X,'*  SUBTRAN INPUT  *'/
     1       29X,10('* '))
44    FORMAT(//T9,'SIMULATION PARAMETERS',T50,'SEARCH ',
     1       'PARAMETERS'/ T9,21('-'),T50,17('-')/
     1       T9,'Replications = ',I6,T50,'Search Area = ',
     1       F9.2/T9,'Random Seed  = ',F9.2,T50,
     1       'Search Time = ',F9.2)
54    FORMAT(T50,'Track Width = ',F9.2)
64    FORMAT(T50,'Track Width = calc.')
74    FORMAT(T9,'TRANSIENT PARAMETERS'/T9,20('-'),T50,
     1       'SIGNAL ERROR ','PARAMETERS'/T10,
     1       '(1 - Yes / 0 - No)',T50,24('-')/T50,
     1       'Rate Par. = ',F5.2/T9,'Real Transients:',T50,
     1       'Std. Dev. = ',F5.2/T10,'Allowed   = ',I2/T10,
     1       'Rate Par. = ',F7.4,T50,'SPEED PARAMETERS'/T9,
     1       'False Transients:',T50,16('-')/T10,
     1       'Allowed   = ',I2,T50,'S. Speed = ',F5.2/T10,
     1       'Rate Par. = ',F7.4,T50,'T. Speed = ',F5.2,
     1       4(/))
84    FORMAT(12X,'STEADY-STATE FIGURE OF MERIT / ',
     1       'PROPAGATION LOSS PARAMETERS'/12X,
     1       59('-')//22X,'Steady-state Figure of ',
     1       'Merit = ',F6.2/)
86    FORMAT(13X,'TRANSIENT FIGURE OF MERIT / PROPAGATION ',
     1       'LOSS ','PARAMETERS'/13X,56('-')// 23X,
     1       'Transient Figure of ','Merit = ',F6.2/)
94    FORMAT(T13,'Range Point : ',F6.2,T50,
     1       'Prop. Loss : ',F6.2)
104   FORMAT(8(/))
114   FORMAT(28X,'TARGET COURSE PARAMETERS'/28X,24('-')//)
124   FORMAT(5X,'Course: ',F4.0,3X,'Prob: ',F4.2,T45,
     1       'Time to Change: ',F6.2,3X,'Prob: ',F4.2)
134   FORMAT(5X,'Course: ',F4.0,3X,'Prob: ',F4.2)
144   FORMAT(T45,'Time to Change: ',F6.2,3X,'Prob: ',F4.2)
154   FORMAT(3(/))
      END
```

```
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C                                                           C
C                        SUBROUTINE PLOSS                   C
C                                                           C
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C
      SUBROUTINE PLOSS(NPp, Rp, Pp, FOM, Rd)
C
C     Variables passed to PLOSS are: NPp, Rp(50), Pp(50),
C                                    FOM
C     Variables returned by PLOSS are: Rd(7)
C
C.....Declare variables. Dimension arrays.
C
      INTEGER NPp, COUNT
      REAL Rp(50), Pp(50), FOM, Rd(7), SLOPE, Rint
C
C.....Initially assign the detection ranges, Rd, to zero.
C
      DO 10 I=1,7
         Rd(I)=0
10    CONTINUE
C
C.....Compute the intersection of the propagation loss curve
C     and the figure of merit.
C
      COUNT=1
      DO 20 I=1,NPp-1
         SLOPE=(Pp(I+1)-Pp(I))/(Rp(I+1)-Rp(I))
         Rint=((FOM-Pp(I))/SLOPE)+Rp(I)
         IF((Rint .GE. Rp(I)) .AND. (Rint .LE. Rp(I+1)))THEN
            Rd(COUNT)=Rint
            COUNT=COUNT+1
            IF(COUNT .EQ. 8)GOTO 30
         END IF
20    CONTINUE
30    END
```

```
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C                                                         C
C                      SUBROUTINE TGCR                    C
C                                                         C
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C
      SUBROUTINE TGCR(DSEED, NPc, Cc, Pc, NPtc, Ttc, Ptc,
     1               Ct, TCt)
C
C     Variables passed to TGCR are: DSEED, NPc, Cc, Pc,
C                                   NPtc, Ttc, Ptc
C     Variables returned by TGCR are: Ct, TCt
C
C     * TGCR requires the subroutine LNORPC.
C
C.....Declare variables. Dimension arrays.
C
      DOUBLE PRECISION DSEED
      INTEGER NPc, NPtc
      REAL Cc(50), Pc(50), Ttc(50), Ptc(50), Ct, TCt, U(2),
     1     SLOPE
C
C.....Generate two Uniform(0,1) random variables to be used
C     in extracting the new course and time to course
C     change.
C
      CALL LRNDPC(DSEED, U, 2)
C
C.....Extract the new course value (Ct).
C
      DO 10 I=1,NPc-1
         SLOPE=(Pc(I+1)-Pc(I))/(Cc(I+1)-Cc(I))
         IF((U(1) .GE. Pc(I)) .AND. (U(1) .LE. Pc(I+1)))THEN
            Ct=((U(1)-Pc(I))/SLOPE)+Cc(I)
            GOTO 20
         END IF
10    CONTINUE
C
C.....Extract the new time to course change (TCt).
C
20    DO 30 I=1,NPtc-1
         SLOPE=(Ptc(I+1)-Ptc(I))/(Ttc(I+1)-Ttc(I))
         IF((U(2) .GE. Ptc(I)) .AND. (U(2) .LE. Ptc(I+1)))
     1   THEN
            TCt=((U(2)-Ptc(I))/SLOPE)+Ttc(I)
            GOTO 40
         END IF
30    CONTINUE
40    END
```

```
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C                                                         C
C                    SUBROUTINE TGTR                      C
C                                                         C
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C
      SUBROUTINE TGTR(DSEED, RPtr, TTt)
C
C     Variables passed to TGTR are: DSEED, RPtr
C     Variables returned by TGTR are: TTt
C
C     * TGTR requires the subroutine LGAMPC.
C
C.....Declare variables.
C
      DOUBLE PRECISION DSEED
      REAL RPtr,TTt,E
C
C.....Compute the time to next transient (TTt). The time
C     between transients is exponentially distributed with
C     mean = 1/RPtr in hours.
C
      CALL LGAMPC(DSEED,E,1,1.)
      TTt=E/RPtr
      END
```

```
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C                                                                               C
C                          SUBROUTINE SRCR                                      C
C                                                                               C
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C
      SUBROUTINE SRCR(C, Xs, Ys, Ss, TW, L, Cs, TCs)
C
C     Variables passed to SRCR are: C, Xs, Ys, Ss, TW, L
C     Variables returned by SRCR are: Cs, TCs
C
C     * The variable C is generated by SRCR when it is first
C       run and is passed back to SRCR on successive runs.
C
C.....Declare variables.
C
      INTEGER C
      REAL Xs, Ys, Ss, TW, L, Cs, TCs, PI
C
C.....Assign the value of PI.
C
      PI=3.1415926
C
C.....Set up the program routing to control the course.
C
      IF((C .EQ. 1) .OR. (C .EQ. 5))GOTO 10
      IF((C .EQ. 2) .OR. (C .EQ. 6))GOTO 20
      IF((C .EQ. 3) .OR. (C .EQ. 7))GOTO 30
      IF((C .EQ. 4) .OR. (C .EQ. 8))GOTO 40
C
C --  Searcher's course - north.
C
10    Cs=0
      TCs=(L-TW)/Ss
      IF(C .EQ. 1)THEN
         C=2
      ELSE
         C=6
      END IF
      GOTO 50
C
C --  Searcher's course - east or west (following north).
C
20    IF(C .EQ. 2)THEN
         Cs=90
         IF(Xs .GE. (L-(1.5*TW)))THEN
            TCs=((L-(TW/2))-Xs)/Ss
            C=7
         ELSE
            TCs=TW/Ss
            C=3
         END IF
```

```
      ELSE
         Cs=270
         IF(Xs .LE. (1.5*TW))THEN
            TCs=(Xs-(TW/2))/Ss
            C=3
         ELSE
            TCs=TW/Ss
            C=7
         END IF
      END IF
      GOTO 50
C
C --  Searcher's course - south.
C
30    Cs=180
      TCs=(L-TW)/Ss
      IF(C .EQ. 3)THEN
        C=4
      ELSE
        C=8
      END IF
      GOTO 50
C
C --  Searcher's course - east or west (following south).
C
40    IF(C .EQ. 4)THEN
         Cs=90
         IF(Xs .GE. (L-(1.5*TW)))THEN
            TCs=((L-(TW/2))-Xs)/Ss
            C=5
         ELSE
            TCs=TW/Ss
            C=1
         END IF
      ELSE
         Cs=270
         IF(Xs .LE. (1.5*TW))THEN
            TCs=(Xs-(TW/2))/Ss
            C=1
         ELSE
            TCs=TW/Ss
            C=5
         END IF
      END IF
      GOTO 50
50    END
```

```
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C                                                         C
C                   SUBROUTINE TRCR                       C
C                                                         C
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C
      SUBROUTINE TRCR(C, C1, Xt, Yt, Xs, Ys, Ss, L, TW, HC,
     1                HXs, HYs, HCs, HTCs, HTh, Cs, TCs)
C
C     Variables passed to TRCR are: C, C1, Xt, Yt, Xs, Ys,
C                                   Ss, L, TW
C     Variables returned by TRCR are: Cs, TCs
C
C     * The variables HC, HXs, HYs, HCs, HTCs, HTh are
C       generated by TRCR when it is first run and are
C       passed back to TRCR on successive runs.
C
C     * TRCR requires the subroutine TRACK.
C
C.....Declare variables.
C
      INTEGER C, C1, HC
      REAL Xt, Yt, Xs, Ys, Ss, L, TW, HXs, HYs, HCs, HTCs,
     1     HTh, Cs, TCs
C
C.....Set up routing depending on C1 value as follows:
C        1. Transient first detected.                [C1=1]
C        2. Return back along the transient course. [C1=2]
C        3. Return to "on hold" steady state search
C           position.                                [C1=3]
C        4. Resume steady state search.              [C1=4]
C
      IF(C1 .EQ. 1)GOTO 10
      IF(C1 .EQ. 2)GOTO 20
      IF(C1 .EQ. 3)GOTO 30
      IF(C1 .EQ. 4)GOTO 40
C
C --  Route 1.
C
10    IF(C .LE. 8)THEN
         HXs=Xs
         HYs=Ys
         HCs=Cs
         HTCs=TCs
         HC=C
         C=9
      END IF
      CALL TRACK(Xt,Yt,Xs,Ys,Ss,L,Cs,TCs)
      TCs=TCs-(TW/(2*Ss))
      HTh=TCs
      C1=2
      GOTO 50
```

```
C
C --   Route 2.
C
20     IF(Cs .LE. 180)THEN
          Cs=Cs+180
       ELSE
          Cs=Cs-180
       END IF
       TCs=HTh
       C1=3
       GOTO 50
C
C --   Route 3.
C
30     CALL TRACK(HXs, HYs, Xs, Ys, Ss, L, Cs, TCs)
       TCs=SQRT(((HYs-Ys)**2)+((HXs-Xs)**2))/Ss
       C1=4
       GOTO 50
C
C --   Route 4.
C
40     C=HC
       Cs=HCs
       TCs=HTCs
       C1=0
50     END
```

```
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C                                                           C
C                   SUBROUTINE SSDET                        C
C                                                           C
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C
      SUBROUTINE SSDET(Xt, Yt, St, Ct, Xs, Ys, Ss, Cs, Rssd,
     1                 C1, Tssd)
C
C     Variables passed to SSDET are: Xt, Yt, St, Ct, Xs, Ys,
C                                    Ss, Cs, Rssd
C     Variables returned by SSDET are: C1, Tssd
C
C.....Declare variables.
C
      INTEGER C1
      REAL Xt, Yt, St, Ct, Xs, Ys, Ss, Cs, Rssd, Tssd, PI,
     1     HOLD, Stx, Sty, Ssx, Ssy, A, B, C, Ax, Bx, Cx,
     1     Ay, By, Cy, Tssda, Tssdb
C
C.....Assign the value of PI.
C
      PI=3.1415926
C
C.....Check to see if the searcher is already within steady
C     state detection range.
C
      HOLD=SQRT(((Xs-Xt)**2)+((Ys-Yt)**2))
      IF(HOLD .LE. Rssd)THEN
         Tssd=0
         C1=1
         GOTO 20
      END IF
C
C.....Compute the target's x speed component (Stx) and y
C     speed component (Sty).
C
      Stx=St*SIN(Ct*PI/180)
      Sty=St*COS(Ct*PI/180)
C
C.....Compute the searcher's x speed component (Ssx) and y
C     speed component (Ssy).
C
      Ssx=Ss*SIN(Cs*PI/180)
      Ssy=Ss*COS(Cs*PI/180)
C
C.....Compute the constants to be used in the quadratic
C     equation for solving the time to steady state
C     detection (if it occurs).
C
      Ax=(Ssx**2)+(Stx**2)-(2*Ssx*Stx)
      Bx=(2*Xs*Ssx)+(2*Xt*Stx)-(2*Xs*Stx)-(2*Xt*Ssx)
```

```
      Cx=(Xs**2)+(Xt**2)-(2*Xs*Xt)
      Ay=(Ssy**2)+(Sty**2)-(2*Ssy*Sty)
      By=(2*Ys*Ssy)+(2*Yt*Sty)-(2*Ys*Sty)-(2*Yt*Ssy)
      Cy=(Ys**2)+(Yt**2)-(2*Ys*Yt)
      A=Ax+Ay
      B=Bx+By
      C=Cx+Cy-(Rssd**2)
C
C.....Check the conditions that yield a solution that is not
C     real or finite.
C                1. A=0
C                2. (B**2)-(4*A*C)<0
C
C --  Condition 1.
C
      IF(A .EQ. 0)GOTO 10
C
C --  Condition 2.
C
      HOLD=(B**2)-(4*A*C)
      IF(HOLD .LT. 0)GOTO 10
C
C.....Compute the values for the time to steady state
C     detection (Tssd).
C
      Tssda=(-B+SQRT(HOLD))/(2*A)
      Tssdb=(-B-SQRT(HOLD))/(2*A)
C
C.....Check for negative detection times.
C                1. Both Tssda and Tssdb negative.
C                2. Tssda positive, Tssdb negative.
C                3. Tssda negative, Tssdb positive.
C                4. Both Tssda and Tssdb positive.
C
C --  Case 1.
C
      IF((Tssda .LT. 0) .AND. (Tssdb .LT. 0))GOTO 10
C
C --  Case 2.
C
      IF((Tssda .GE. 0) .AND. (Tssdb .LT. 0))THEN
         Tssd=Tssda
         C1=1
      END IF
C
C --  Case 3.
C
      IF((Tssda .LT. 0) .AND. (Tssdb .GE. 0))THEN
         Tssd=Tssdb
         C1=1
      END IF
C
```

```
C --   Case 4.
C
       IF((Tssda .GE. 0) .AND. (Tssdb .GE. 0))THEN
          Tssd=MIN(Tssda,Tssdb)
          C1=1
       END IF
       GOTO 20
10     C1=0
20     END
```

```
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C                                                           C
C                    SUBROUTINE TRDET                       C
C                                                           C
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C
      SUBROUTINE TRDET(Xt, Yt, Xs, Ys, Rd, C)
C
C     Variables passed to TRDET are: Xt, Yt, Xs, Ys, Rd(7)
C     Variables returned by TRDET are: C
C
C.....Declare variables. Dimension arrays.
C
      INTEGER C
      REAL Xt, Yt, Xs, Ys, Rd(7), HOLD
C
C.....Compute the current range between the searcher and
C     target (HOLD).
C
      HOLD=SQRT(((Xs-Xt)**2)+((Ys-Yt)**2))
C
C.....Check to see if the current range is within transient
C     detection range. The code (C) for a detection is 1,
C     and the code for no detection is 0. Two cases exist
C     that require checking:
C                    1. Direct path detection.
C                    2. Convergence zone detection (3 max).
C
C --  Case 1.
C
      IF(HOLD .LE. Rd(1))THEN
         C=1
         GOTO 20
      END IF
C
C --  Case 2.
C
      DO 10 I=2,6,2
         IF((HOLD .GE. Rd(I)) .AND. (HOLD .LE. Rd(I+1)))THEN
            C=1
            GOTO 20
         END IF
10    CONTINUE
C
C.....Indicate that no detection occurred.
C
      C=0
20    END
```

```
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C                                                                       C
C                          SUBROUTINE LSJ                               C
C                                                                       C
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C
      SUBROUTINE LSJ(DSEED, RPlsj, SDlsj, Elsj, Tlsj)
C
C     Variables passed to LSJ are: DSEED,RPlsj,SDlsj
C     Variables returned by LSJ are: Elsj,Tlsj
C
C     * LSJ requires the subroutines: LNORPC, LGAMPC.
C
C.....Declare variables.
C
      DOUBLE PRECISION DSEED
      REAL RPlsj, SDlsj, Elsj, Tlsj, N, E
C
C.....Calculate the Lambda Sigma Jump error fluctuation
C     (Elsj). The error fluctuation is normally distributed
C     with mean zero and standard deviation = SDlsj.
C
      CALL LNORPC(DSEED, N, 1)
      Elsj=SDlsj*N
C
C.....Calculate the time to the fluctuation (Tlsj). The time
C     between flucuations is exponentially distributed with
C     mean = 1/RPlsj.
C
      CALL LGAMPC(DSEED, E, 1, 1.)
      Tlsj=E/RPlsj
      END
```

```
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C                                                                       C
C                          SUBROUTINE FALTR                             C
C                                                                       C
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C
      SUBROUTINE FALTR(DSEED, RPftr, L, Xf, Yf, TTf)
C
C     Variables passed to FALTR are: DSEED, RPftr, L
C     Variables returned by FALTR are: Xf, Yf, TTf
C
C     * FALTR requires the subroutines: LRNDPC, LGAMPC
C
C.....Declare variables. Dimension arrays.
C
      DOUBLE PRECISION DSEED
      REAL RPftr, L, Xf, Yf, TTf, U(2), E
C
C.....Compute the x and y positions of the false transient
C     (Xf,Yf). The position is uniformly distributed over
C     the search area.
C
      CALL LRNDPC(DSEED, U, 2)
      Xf=L*U(1)
      Yf=L*U(2)
C
C.....Compute the time to the next false transient (TTf).
C     The time between false transients is exponentially
C     distributed with mean = 1/RPftr in hours.
C
      CALL LGAMPC(DSEED, E, 1, 1.)
      TTf=E/RPftr
      END
```

```
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C                                                         C
C                   SUBROUTINE POSIT                      C
C                                                         C
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C
      SUBROUTINE POSIT(T, Xt, Yt, Ct, St, Xs, Ys, Cs, Ss)
C
C     Variables passed to POSIT are: T, Xt, Yt, Ct, St, Xs,
C                                    Ys, Cs, Ss
C     Variables returned by POSIT are: Xt, Yt, Xs, Ys
C
C     * POSIT updates the variables Xt, Yt, Xs, Ys
C
C.....Declare variables.
C
      REAL T, Xt, Yt, Ct, St, Xs, Ys, Cs, Ss, PI
C
C.....Assign the value for PI.
C
      PI=3.1415926
C
C.....Compute target's x position (Xt) and y position (Yt).
C
      Xt=Xt+(T*St*SIN(Ct*PI/180))
      Yt=Yt+(T*St*COS(Ct*PI/180))
C
C.....Compute searcher's x position (Xs) and y position
C     (Ys).
C
      Xs=Xs+(T*Ss*SIN(Cs*PI/180))
      Ys=Ys+(T*Ss*COS(Cs*PI/180))
      END
```

```
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C                                                         C
C                    SUBROUTINE REFLCT                    C
C                                                         C
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C
      SUBROUTINE REFLCT(L, Ct, St, Xt, Yt, Cr, Tr)
C
C     Variables passed to REFLCT are: L, Ct, St, Xt, Yt
C     Variables returned by REFLCT are: Cr, Tr
C
C.....Declare variables.
C
      REAL L, Ct, St, Xt, Yt, Cr, Tr, PI, Stx, Sty, Ty, Tx
C
C.....Assign the value of PI.
C
      PI=3.1415926
C
C.....Compute the x component of target speed (Stx). Compute
C     the y component of target speed (Sty).
C
      Stx=St*SIN(Ct*PI/180)
      Sty=St*COS(Ct*PI/180)
C
C.....Compute the time to reflection (Tr) and the reflected
C     course (Cr) for the five cases:
C         1. Special cases - Ct=0,90,180,270,360
C         2. 0<Ct<90
C         3. 90<Ct<180
C         4. 180<Ct<270
C         5. 270<Ct<360
C
C --  Special cases.
C
      IF((Ct .EQ. 0) .OR. (Ct .EQ. 360))THEN
         Tr=(L-Yt)/Sty
         Cr=180
         GOTO 10
      END IF
      IF(Ct .EQ. 90)THEN
         Tr=(L-Xt)/Stx
         Cr=270
         GOTO 10
      END IF
      IF(Ct .EQ. 180)THEN
         Tr=-Yt/Sty
         Cr=0
         GOTO 10
      END IF
      IF(Ct .EQ. 270)THEN
         Tr=-Xt/Stx
```

```
            Cr=90
            GOTO 10
         END IF
C
C --   Case 1.
C
         IF((Ct .GT. 0) .AND. (Ct .LT. 90))THEN
            Ty=(L-Yt)/Sty
            Tx=(L-Xt)/Stx
            Tr=MIN(Ty,Tx)
            IF(Tr .EQ. Ty)THEN
               Cr=180-Ct
            ELSE
               Cr=360-Ct
            END IF
            GOTO 10
         END IF
C
C --   Case 2.
C
         IF((Ct .GT. 90) .AND. (Ct .LT. 180))THEN
            Ty=-Yt/Sty
            Tx=(L-Xt)/Stx
            Tr=MIN(Ty,Tx)
            IF(Tr .EQ. Ty)THEN
               Cr=180-Ct
            ELSE
               Cr=360-Ct
            END IF
            GOTO 10
         END IF
C
C --   Case 3.
C
         IF((Ct .GT. 180) .AND. (Ct .LT. 270))THEN
            Ty=-Yt/Sty
            Tx=-Xt/Stx
            Tr=MIN(Ty,Tx)
            IF(Tr .EQ. Ty)THEN
               Cr=540-Ct
            ELSE
               Cr=360-Ct
             END IF
             GOTO 10
         END IF
C
C --   Case 4.
C
         IF((Ct .GT. 270) .AND. (Ct .LT. 360))THEN
            Ty=(L-Yt)/Sty
            Tx=-Xt/Stx
            Tr=MIN(Ty,Tx)
```

```
            IF(Tr .EQ. Ty)THEN
               Cr=540-Ct
            ELSE
                Cr=360-Ct
            END IF
            GOTO 10
         END IF
10       END
```

```
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C                                                                       C
C                        SUBROUTINE TRACK                               C
C                                                                       C
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C
      SUBROUTINE TRACK(Xt, Yt, Xs, Ys, Ss, L, Cs, TCs)
C
C     Variables passed to TRACK are: Xt, Yt, Xs, Ys, Ss, L
C     Variables returned by TRACK are: Cs, TCs
C
C     * TRACK requires the subroutine REFLCT.
C
C.....Declare variables.
C
      REAL Xt, Yt, Xs, Ys, Ss, L, Cs, TCs, PI, HOLD, Cr, Tr
C
C.....Assign the value for PI.
C
      PI=3.1415926
C
C.....Compute the approach track course (Cs) for the
C     following cases:
C          1. Special case where the y positions are equal.
C          2. (Xt .GE. Xs) and (Yt .GT. Ys)
C          3. (Xt .LE. Xs) and (Yt .LT. Ys)
C          4. (Xt .LE. Xs) and (Yt .GT. Ys)
C          5. (Xt .GE. Xs) and (Yt .LT. Ys)
C
C -- Case 1.
C
      IF(Ys .EQ. Yt)THEN
         IF(Xt .GT. Xs)THEN
            Cs=90
         ELSE
            Cs=270
         END IF
         GOTO 10
      END IF
C
C --  Compute the common value for the angle of the track
C     between the searcher and target (HOLD) to be used in
C     cases 2-5.
C
      HOLD=(ATAN((Xt-Xs)/(Yt-Ys)))*180/PI
C
C --  Case 2.
C
      IF((Xt .GE. Xs) .AND. (Yt .GT. Ys))THEN
         Cs=HOLD
         GOTO 10
      END IF
```

```
C
C --   Case 3.
C
       IF((Xt .LE. Xs) .AND. (Yt .LT. Ys))THEN
          Cs=HOLD+180
          GOTO 10
       END IF
C
C --   Case 4.
C
       IF((Xt .LE. Xs) .AND. (Yt .GT. Ys))THEN
          Cs=HOLD+360
          GOTO 10
       END IF
C
C --   Case 5.
C
       IF((Xt .GE. Xs) .AND. (Yt .LT. Ys))THEN
          Cs=HOLD+180
       END IF
C
C.....Call the REFLCT subroutine to get the time to next
C      course change (TCs).
C
10     CALL REFLCT(L,Cs,Ss,Xs,Ys,Cr,Tr)
       TCs=Tr
       END
```

```
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C                                                         C
C                     SUBROUTINE OUT                      C
C                                                         C
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C
      SUBROUTINE OUT(Nd, Nz, Nm, Ftw, TW, Tmax, Td)
C
C     Variables passed to OUT are: Nd, Nz, Nm, Ftw, TW,
C                                  Tmax, Td
C     Variables returned by OUT are: none
C
C     * OUT requires the subroutine HSCRN
C
C.....Declare variables. Dimension arrays.
C
      CHARACTER*12 FILE1
      INTEGER Nd, Nz, Nm, Ftw, ICOUNT
      REAL TW, Tmax, Td(5000), Tdor(5000), SUMd, SUMm,
     1     SUM1d, SUM1m, SUM2, SUM3, SUM4, SUM5, SUM6, SUM7,
     1     XBAR, SD, SDx, XBARd, SBAR, RPe, RS, SUR(5000),
     1     PD(10),PDT(10)
C
C.....Designate the output file. FILE1 - holds the times to
C     detection.
C
      FILE1='TIME.OUT'
C
C.....Open FILE1 for writing to. The other output file
C     (SUBTRAN.DAT) is already open from the subroutine IN
C     and is designated as device 20.
C
      OPEN(30,FILE=FILE1)

C
C.....Write to the user's screen a message indicating
C     computing output statistics.
C
      WRITE(*,14)
C
C.....Order the times to detection from smallest to largest
C     (Tord).
C
      Tdor(1)=Td(1)
      DO 10 I=1,Nd
         DO 20 J=I-1,1,-1
            IF(Td(I) .LT. Tdor(J))THEN
               Tdor(J+1)=Tdor(J)
               Tdor(J)=Td(I)
            ELSE
               Tdor(J+1)=Td(I)
               GOTO 10
```

```
            END IF
20        CONTINUE
10     CONTINUE
C
C.....Compute the mean (XBAR) and standard deviation (SD) of
C      the times to detection. For the cases where detection
C      did not occur before Tmax, assign a value of Tmax for
C      the time to detection. Cases where the time to
C      detection is zero are not included. Also include the
C      95% confidence bounds on the estimate of the mean
C      (SDx).
C
       DO 30 I=1,Nm
          Tdor(Nd+I)=Tmax
30     CONTINUE
       SUMd=0
       SUMm=0
       SUM1d=0
       SUM1m=0
       DO 40 I=1,Nd
          SUMd=SUMd+Tdor(I)
          SUM1d=SUM1d+(Tdor(I)**2)
40     CONTINUE
       DO 50 I=1,Nm
          SUMm=SUMm+Tdor(Nd+I)
          SUM1m=SUM1m+(Tdor(Nd+I)**2)
50     CONTINUE
       XBAR=(SUMd+SUMm)/(Nd+Nm)
       SD=SQRT(((SUM1d+SUM1m)-((Nd+Nm)*(XBAR**2)))/(Nd+Nm-1))
       SDx=1.645*SD/((Nd+Nm)**.5)
C
C.....Construct a empirical survival function (SUR) from the
C      ordered time to detection data. Under the assumption
C      that the data is from an exponential distribution,
C      perform a linear regression on the natural log of SUR
C      with respect to the ordered times Tdor. The slope of
C      the resultant line is an estimate for the rate
C      parameter of the exponential distribution (RPe). 1/RPe
C      is another estimate for the mean time to detection.
C      This estimate is independent of the detections that
C      occurred at time zero thus those values are not
C      included. Also computed is the R-squared value of the
C      linear regression. This value is an indication of how
C      good the exponential assumption is.
C
       SUM2=0
       SUM3=0
       SUM4=0
       ICOUNT=0
       DO 60 I=1,Nd
          SUR(I)=1-(REAL(I)/REAL(Nd+Nm))
          IF(SUR(I) .LE. 0)THEN
```

```
            ICOUNT=ICOUNT+1
            SUMd=SUMd-Tdor(I)
            SUM1d=SUM1d-(Tdor(I)**2)
            GOTO 60
         END IF
         SUM2=SUM2+ALOG(SUR(I))
         SUM3=SUM3+(Tdor(I)*ALOG(SUR(I)))
         SUM4=SUM4+(ALOG(SUR(I))**2)
60    CONTINUE
      DO 70 I=1,10
         DO 80 J=1,Nd
            IF((1-SUR(J)) .GE. (REAL(I)/10))THEN
               PD(I)=1-SUR(J)
               PDT(I)=Tdor(J)
               GOTO 70
            END IF
80       CONTINUE
70    CONTINUE
      XBARd=SUMd/(Nd-ICOUNT)
      SBAR=SUM2/(Nd-ICOUNT)
      RPe=(SUM3-(XBARd*SUM2))/
     1     (SUM1d-((Nd-ICOUNT)*(XBARd**2)))
      SUM5=0
      DO 90 I=1,Nd
         IF(SUR(I) .LE. 0)GOTO 90
         SUM5=SUM5+((Tdor(I)-XBARd)*(ALOG(SUR(I))-SBAR))
90    CONTINUE
      SUM6=SUM1d-((Nd-ICOUNT)*(XBARd**2))
      SUM7=SUM4-((Nd-ICOUNT)*(SBAR**2))
      RS=(SUM5**2)/(SUM6*SUM7)
C
C.....Write the output to the screen and the two output data
C     files.
C
      WRITE(*,114)
      WRITE(*,24)
      WRITE(20,24)
      WRITE(*,34)Nz
      WRITE(20,34)Nz
      WRITE(*,44)Nm
      WRITE(20,44)Nm
      IF(Ftw .EQ. 0)THEN
         WRITE(*,54)TW
         WRITE(20,54)TW
      ELSE
         WRITE(*,*)
      END IF
      WRITE(*,64)XBAR,SDx,SD,(-1/RPe),RS
      WRITE(20,64)XBAR,SDx,SD,(-1/RPe),RS
      WRITE (*,74)
      WRITE (20,74)
      DO 100 I=1,5
```

```
         WRITE(*,84)PD(I),PDT(I),PD(I+5),PDT(I+5)
         WRITE(20,84)PD(I),PDT(I),PD(I+5),PDT(I+5)
100   CONTINUE
      WRITE(*,*)
      CALL HSCRN
      WRITE(*,94)
      DO 110 I=1,(Nd+Nm)
         WRITE(30,*)Tdor(I)
110   CONTINUE
      WRITE(*,104)
      CALL HSCRN
C
C.....Write format statements.
C
14     FORMAT(23(/),1X,'Computing output statistics....'/)
24     FORMAT(28X,12('* ')/28X,'* SUBTRAN OUTPUT DATA *'/
     1       28X,12('* ')/)
34     FORMAT(5X,'NO. OF REPS. WITH DETECTION TIME OF ',
     1       'ZERO ..................',I4)
44     FORMAT(5X,'NO. OF REPS. WITH DETECTION TIME ',
     1       'GREATER THAN Tmax .........',I4)
54     FORMAT(5X,'CALCULATED TRACK WIDTH VALUE ',
     1       '.............................',F7.2)
64     FORMAT(/26X,'TIME TO DETECTION STATISTICS'/26X,
     1       28('-')/ 26X,'Mean      = ',F7.2,' +/-',
     1       F5.2,' (95% CI)'/26X,'Std. Dev. = ',F7.2/
     1       26X,'1/Lambda  = ',F7.2/26X,'R-Squared = ',
     1       F7.2/)
74     FORMAT(T10,'Prob. of Det.',T27,'By time',T47,
     1       'Prob. of Det.',T64,'By time'/T10,13('-'),
     1       T27,7('-'),T47,13('-'),T64,7('-'))
84     FORMAT(T15,F3.1,T28,F6.1,T52,F3.1,T65,F6.1)
94     FORMAT(23(/),1X,'Writing to output file....'/)
104    FORMAT(4X,'Output data is located in the following ',
     1       'files:'/7X,'1. SUBTRAN.DAT - input ',
     1       'parameters, time to detection ',
     1       'statistics.'/7X,'2. TIME.OUT    - times to ',
     1       'detection.'//)
114    FORMAT(23(/))
       END
```

```
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C                                                         C
C                    SUBROUTINE HSCRN                     C
C                                                         C
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C
      SUBROUTINE HSCRN
C
C.....Write to the user's terminal a "how to continue"
C     message.
C
      WRITE(*,14)
C
C.....Perform a generic READ statement to hold the screen.
C
      READ(*,*)
C
C.....Write format statement.
C
14    FORMAT(26X,'- Press <ENTER> to continue -')
      END
```

```
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C                                                           C
C                    SUBROUTINE LRNDPC                      C
C                                                           C
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C
      SUBROUTINE LRNDPC(DSEED, U, N)
C
      INTEGER*4 N, I
      REAL U(N)
      DOUBLE PRECISION D31M1, DSEED, D31
C
C     d31m1 = 2**31-1
C     d31 = 2**31
C
      DATA D31M1/2147483647.D0/
      DATA D31/2147483648.D0/
C
      DO 5 I=1,N
         DSEED = DMOD(16807.D0*DSEED,D31M1)
         U(I) = DSEED/D31
5     CONTINUE
      END
```

```
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C                                                                               C
C                          SUBROUTINE LNORPC                                    C
C                                                                               C
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C
      SUBROUTINE LNORPC(ISEED,A,N)

C     This program will generate a vector of normal random
C     variables using according to the sine-cosine method.
C
      INTEGER  N,I,IND
      REAL  A(N),U(2),S,W,U1,XSTAR
      DOUBLE PRECISION ISEED, PI
      DATA PI/3.14159265358979D0/
C
C     DATA  IND/1/
      IND = 1
      DO 100 I=1,N
      IND = -IND
      IF (IND.GE.0) GOTO 20
  10  CALL LRNDPC(ISEED, U, 2)
      S =SQRT(-2*ALOG(U(1)))
      W = 2*PI*U(2)
      XSTAR = S*COS(W)
      A(I) = S*SIN(W)
      GOTO 100
  20  A(I) = XSTAR
  100 CONTINUE
      RETURN
      END
```

```
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C                                                         C
C                    SUBROUTINE LGAMPC                    C
C                                                         C
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C
      SUBROUTINE LGAMPC(DSEED, G, N, A)
      DOUBLE PRECISION DSEED
      DIMENSION G(N), UN(2)
      DATA ASAVE /-1./,XLL/-1./
      IF(A .LE. 1.) GO TO 1500
C --- GAMMA GENERATOR A>1.  SCHMEIZER'S ALGORITHM
      IF (A .EQ. ASAVE) GO TO 50
      ASAVE = A
      X1=0
      X2=0
      F1=0
      F2=0
      X3 = A - 1.
      R3X=X3-X3*ALOG(X3)
      D = SQRT(X3)
      IF (D .GE. X3) GO TO 10
      X2 = X3 - D
      X1 = X2*(1.-1./D)
      XLL = 1. - X3/X1
      F1 = EXP(X3*ALOG(X1)  -X1+R3X)
      F2 = EXP(X3*ALOG(X2) - X2 + R3X)
 10   X4 = X3 + D
      X5 = X4*(1. + 1./D)
      XLR = 1. - X3/X5
      F4 = EXP(X3*ALOG(X4) - X4 + R3X)
      F5 = EXP(X3*ALOG(X5) - X5 + R3X)
      P1 = F2*(X3-X2)
      P2 = F4*(X4-X3) + P1
      P3 = F1*(X2-X1) + P2
      P4 = F5*(X5-X4) + P3
      P5 = X3-X2 -P1  + P4
      P6 = X4-X3-P2+P1 + P5
      P7 = (F2-F1)*(X2-X1)*.5 + P6
      P8 = (F4-F5)*(X5-X4)*.5 + P7
      P9 = -F1/XLL + P8
      P10 = F5/XLR + P9
C     WRITE(6,77) X1,X2,X4,X5,F1,F2,F4,F5,P1,P2,P3,P4,P5,P6,
C    1      P7,P8,P9,P10
77    FORMAT(5F20.8)
50    DO 1400 II=1,N
100   CALL LRNDPC(DSEED, U, 1)
      U = U*P10
      IF (U.GT.P4) GO TO 500
      IF (U.GT.P1) GO TO 200
      X = X2 + U/F2
      GO TO 1400
```

```
  200 IF (U.GT.P2) GO TO 300
      X = X3 + (U-P1)/F4
      GO TO 1400
  300 IF (U.GT.P3) GO TO 400
      X = X1 + (U-P2)/F1
      GO TO 1400
  400 X = X4 + (U-P3)/F5
      GO TO 1400
  500 CALL LRNDPC(DSEED, W, 1)
      IF (U.GT.P5) GO TO 600
      X = X2 + (X3-X2)*W
      IF ((U-P4)/(P5-P4) .LE. W) GO TO 1400
      V=F2 + (U-P4) / (X3-X2)
      GO TO 1300
  600 IF(U.GT.P6) GO TO 700
      X=X3+(X4-X3)*W
      IF ((P6-U)/(P6-P5) .GE. W) GO TO 1400
      V = F4 + (U-P5)/(X4-X3)
      GO TO 1300
  700 IF (U .GT. P8) GO TO 900
      CALL LRNDPC(DSEED, W2, 1)
      IF (W2 .GT. W) W=W2
      IF (U.GT.P7) GO TO 800
      X=X1+(X2-X1)*W
      V=F1+2.*W*(U-P6)/(X2-X1)
      IF (V.LE.F2*W) GO TO 1400
      GO TO 1300
  800 X=X5 - W*(X5-X4)
      V = F5+2.*W*(U-P7)/(X5-X4)
      IF (V.LE.F4*W) GO TO 1400
      GO TO 1300
  900 IF (U.GT.P9) GO TO 1000
      U=(P9-U)/(P9-P8)
      X=X1-ALOG(U)/XLL
      IF (X.LE.0) GO TO 100
      IF (W.LT. (XLL*(X1-X)+1.)/U) GO TO 1400
      V = W*F1*U
      GO TO 1300
 1000 U = (P10-U)/(P10-P9)
      X=X5-ALOG(U)/XLR
      IF (W.LT. (XLR*(X5-X) + 1.)/U) GO TO 1400
      V = W*F5*U
 1300 IF (ALOG(V) .GT. X3*ALOG(X) + R3X - X) GO TO 100
 1400 G(II)=X
      RETURN
C --- GAMMA GENERATOR A<=1.  ALGORITHM GS.
 1500 CONTINUE
      IF (A .EQ. ASAVE) GO TO 1550
      ASAVE = A
      AINV=1/A
      B=A*.36787944 + 1.
 1550 CONTINUE
```

```
      IF (A .EQ. 1.0) GO TO 1800
      DO 1700 I=1,N
1560  CONTINUE
      CALL LRNDPC(DSEED,UN,2)
      EXPL=-ALOG(UN(1))
      P=B*UN(2)
      IF(P .GT. 1.) GO TO 1600
      X=P**AINV
      IF(X .GT. EXPL) GO TO 1560
      G(I)=X
      GO TO 1700
1600  CONTINUE
      X=-ALOG((B-P)*AINV)
      IF( (1-A)*ALOG(X) .GT. EXPL) GO TO 1560
      G(I)=X
1700  CONTINUE
      GOTO 1900
1800  CONTINUE
      CALL LRNDPC(DSEED, G, N)
      DO 1810 I=1,N
      G(I) = -ALOG(G(I))
1810  CONTINUE
1900  CONTINUE
      RETURN
      END
```

LIST OF REFERENCES

1. Center for Naval Analyses, Report CRC 420, *An Introduction to the Analysis of Underwater Acoustic Detection*, by W. J. Hurley, July 1980.

2. Daniel H. Wagner Associates Report to the Office of Naval Research (Contract N00014-73-C-0432), *A Comparison of Detection Models Used in ASW Operations Analysis*, by B. J. McCabe and B. Belkin, 31 October 1973.

3. Operations Research Institute Report to the Office of Naval Research (Contract N00014-76-C-0479), *Model Results of the Effects of Internal Waves on Acoustic Propagation*, by E. Moses, W. Galati, and N. Nicholas, 3 August 1978.

4. Tehan, T. N., *A Survey of Parameter Estimation Methods for the Lambda-Sigma Jump Detection Model*, Master's Thesis, Naval Postgraduate School, Monterey, California, June 1979.

5. Lewis, P. A. W., and Orav E. J., *Simulation Methodology for Statisticians, Operations Analysts, and Engineers, Part I: Modeling and Crude Simulation*, Naval Postgraduate School, Monterey, California, 1987.

6. Nyhoff, L., and Leestma, S., *Fortran 77 for Engineers and Scientists*, Macmillam Publishing Company, 1985.

7. Lewis, P. A. W., Orav E. J., and Uribe, L., *Advanced Simulation and Statistics Package, IBM Professional Fortran Version*, Wadsworth and Brooks/Cole Advanced Books and Software, 1986.

8. DeGroot, M. H., *Probability and Statistics*, 2d ed., Addison-Wesely Publishing Company, 1986.

9. Chambers, J. M., and others, *Graphical Methods for Data Analysis*, Duxbury Press, Boston, Massachusetts, 1983.

10. Abd El-Fadeel, S. I., *A Mathematical Model for Calculating Non-Detection Probability of a Random Tour Target*, Master's Thesis, Naval Postgraduate School, Monterey, California, December 1985.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Professor R. N. Forrest Code 55FO Naval Postgraduate School Monterey, California 93943	1
4.	Professor James N. Eagle Code 55ER Naval Postgraduate School Monterey, California 93943	1
5.	John Hopkins University Applied Physics Laboratory John Hopkins Road Laurel, Maryland 20707-6099 Attn: Dave Restione 8-382	1