NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

ANALYSIS OF THE POSITIONAL ACCURACY OF A RANGE DIFFERENCE MISSILE POSITION MEASURING SYSTEM

by

Robert A. Klaszky

September 2000

Thesis Advisor:	D. Curtis Schleher
Second Reader:	David C. Jenn

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2000	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE: Analysis of the Positional Accuracy of a Range Difference Missile Position Measuring System	**5. FUNDING NUMBERS**
6. AUTHOR(S) Klaszky, Robert A.	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES 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.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	**12b. DISTRIBUTION CODE**

13. ABSTRACT ***(maximum 200 words)***

The Telemetry (TM) Tracker system is designed to determine time, space, and position information (TSPI) of an airborne missile by detecting its telemetry signal at a number of receiver sites. Doppler frequency measurements are converted to range differences between the missile and receiver sites, whose locations are known in three dimensions. An algorithm then utilizes these range differences to obtain missile TSPI with 1-meter accuracy. The TM Tracker was fielded during live missile firings and measurements indicated that the desired TSPI accuracy was not attained.

This thesis examines system requirements and limitations of the TM Tracker to obtain TSPI with 1-meter accuracy. The theory of operation and components of the TM Tracker are introduced. Algorithms used in computing position of a radiating source from range differences are analyzed. MATLAB simulations are conducted with missile trajectory data to determine the required measurement precision and signal-to-noise ratio (SNR) at the receiver sites to obtain 1-meter TSPI. The receivers' 45-degree, 3-dB beam widths are then implemented to observe their effects on TSPI accuracy. Simulations reveal that the TM Tracker system is capable of producing TSPI with 1-meter accuracy provided that precise frequency measurements and adequate SNR values are available at the receiver sites.

14. SUBJECT TERMS Telemetry (TM) Tracker, TSPI, Range Difference, Time Difference of Arrival (TDOA), Frequency Difference of Arrival (FDOA)	**15. NUMBER OF PAGES** 126
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

ANALYSIS OF THE POSITIONAL ACCURACY OF A RANGE DIFFERENCE MISSILE POSITION MEASURING SYSTEM

Robert A. Klaszky
Lieutenant, United States Navy
B.S., Illinois Institute of Technology, 1992

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2000

Author: [illegible]
Robert A. Klaszky

Approved by: D. Curtis Schleher
D. Curtis Schleher, Thesis Advisor

David C. Jenn
David C. Jenn, Second Reader

Dan C. Boger
Dan C. Boger, Chairman
Computer and Information Sciences & Operations

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The Telemetry (TM) Tracker system is designed to determine time, space, and position information (TSPI) of an airborne missile by detecting its telemetry signal at a number of receiver sites. Doppler frequency measurements are converted to range differences between the missile and receiver sites, whose locations are known in three dimensions. An algorithm then utilizes these range differences to obtain missile TSPI with 1-meter accuracy. The TM Tracker was fielded during live missile firings and measurements indicated that the desired TSPI accuracy was not attained.

This thesis examines system requirements and limitations of the TM Tracker to obtain TSPI with 1-meter accuracy. The theory of operation and components of the TM Tracker are introduced. Algorithms used in computing position of a radiating source from range differences are analyzed. MATLAB simulations are conducted with missile trajectory data to determine the required measurement precision and signal-to-noise ratio (SNR) at the receiver sites to obtain 1-meter TSPI. The receivers' 45-degree, 3-dB beam widths are then implemented to observe their effects on TSPI accuracy. Simulations reveal that the TM Tracker system is capable of producing TSPI with 1-meter accuracy provided that precise frequency measurements and adequate SNR values are available at the receiver sites.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLDEGEMENT

I wish to express my sincere gratitude to those who contributed to the successful completion of this project.

I would like to thank Greg Velicer and the personnel at the Naval Air Warfare Center – Weapons Division, China Lake, CA for the use of their TM Tracker final report and the expert guidance and support they provided throughout the past nine months. I would also like to thank Professor David Jenn for proofreading my thesis. Additionally, many thanks go out to my thesis advisor, Professor Curtis Scheleher, who provided me with the knowledge and guidance to program the simulations in MATLAB.

Finally, I would like to thank my wife Robin for her support and understanding during the many long hours that I spent at the campus computer lab in completing this project.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

The accurate determination of time, space, and position information (TSPI) of an airborne vehicle such as a missile has been a valuable asset on Department of Defense (DOD) test ranges for the evaluation and testing of new weapon systems. Four commonly used methods in determining TSPI include high-speed photography, use of the Global Positioning System (GPS), laser tracking, and radar tracking. While the spectrum of TSPI accuracy ranges from inches to several feet with the use of these methods, they are unable to meet the variance of field test requirements over all DOD test ranges.

High-speed photography uses multiple cameras along the missile's trajectory with high-speed film imagery. These cameras are coupled with a tracking mount that is used by an operator to visually track the airborne missile. TSPI is calculated by correlating the angular positions of the cameras with the missile's location in each frame of film. This method can produce TSPI data with several feet of accuracy, but is limited in the performance of its operators against high-speed vehicles such as supersonic missiles, the capability to be used over water, and its use with multiple targets. GPS uses a constellation of satellites to determine position, but is limited by satellite availability at the location of the test range. Laser tracking is able to produce TSPI with inches of accuracy but is extremely expensive to use and is not readily portable for use on all test ranges. Radar tracking is a common method used on DOD test ranges today. Depending on the platform, TSPI accuracy ranges between several and hundreds of feet. Numerous missile ranges over water utilize the U.S. Navy's SPY-1 phased-array radar for TSPI calculation, but are limited by its availability within the fleet of Navy ships. [Ref. 1]

The Telemetry (TM) Tracker system was developed to counter the deficiencies within these TSPI collection methods. Its theory of operation allows for providing TSPI data in a wide range of field test conditions, including the ability to be used over water and land ranges, and in all weather conditions that allow such a test to be conducted. The TM Tracker offers portability for use on all DOD test ranges and has the ability to track multiple test vehicles. The objective in its development and design was TSPI accuracy within 1 meter.

The TM Tracker determines TSPI of an airborne missile by detecting its telemetry signal at a number of remote receiver sites whose positions are known in three dimensions. The Doppler shifted signals are received and converted to range differences between the remote receiver sites. In its post processing system, an algorithm utilizes these range differences to determine the telemetry source location and hence, missile TSPI. The TM Tracker was field tested at White Sands Missile Range during two testing programs in August and September 1999 and yielded unfavorable TSPI data.

This thesis examines the system requirements and limitations of the TM Tracker to obtain TSPI with 1-meter accuracy. MATLAB was extensively used in the simulations of the TM Tracker system along with missile data obtained from the laser tracker on the range.

There are five chapters and two appendices within this thesis. In Chapter II, the theory of operation and components of the TM Tracker are introduced and discussed. Range difference algorithms used in simulations for determining source location are analyzed in Chapter III. The heart of this project is contained in Chapter IV. Results of MATLAB simulations of source location algorithms with frequency measurement errors and noise are presented. Simulations of the operation of the TM Tracker with missile trajectory data and all remote receivers are discussed. These simulations are then modified to implement the receivers' 45-degree, 3-dB antenna beam widths, which limit their ability to detect the telemetry signal along the missile's trajectory and ultimately, the accuracy of TSPI. Chapter V then summarizes and concludes this work and discusses ongoing and future research. MATLAB source codes used for simulations are listed in Appendix A. Plots of azimuth and elevation angles vs. time for the nine receivers on the missile range are contained in Appendix B.

THIS PAGE INTENTIONALLY LEFT BLANK

II. TELEMETRY TRACKER SYSTEM

A. THEORY OF OPERATION

The Telemetry (TM) Tracker determines time, space and position information (TSPI) of an airborne missile equipped with a telemetry transmitter by measuring range differences with respect to time between the missile and the receiver sites utilized during missile flight whose three-dimensional positions are already known prior to launch. Range differences are measured between receivers utilized along the missile's trajectory and a reference receiver within the constellation. For N receivers being utilized, there exists $N - 1$ range differences that are used to compute missile TSPI. Range differences are computed using Doppler frequencies that are obtained at the receiver sites along with the telemetry signal wavelength and sample time. By utilizing the Doppler difference between receivers and the initial receiver location, the range difference of arrival (RDOA) is continuously determined. This RDOA is then used within an algorithm to determine the missile's position.

The range difference determined between two receivers restricts the source location to a hyperbola with the two receivers as foci. By using four receiver sites, three independent hyperbolas are generated from three range differences and their intersection defines the source location. Figure 2.1 illustrates three intersecting hyperbolas that represent the missile's estimated position using four receiver sites in the same plane. Each of the three hyperbolas has two branches. If the receivers are in different planes, the estimated position is defined by intersecting hyperboloids.

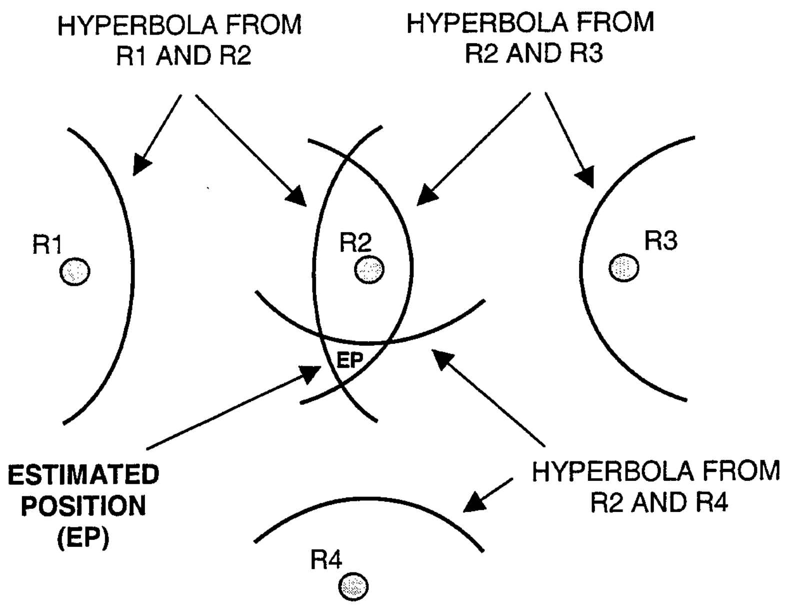

Figure 2.1. Intersecting Hyperbolas Using Four Receivers

The four hyperbolas in Figure 2.1 show a favorable geometric configuration of the receivers for position estimation using range differences. Their intersection results in minimal source location ambiguity as the hyperbolas become orthogonal to one another. The geometric configuration of the receivers, thus, can affect the amount of ambiguity in determining missile position, particularly when the hyperbolas are mutually parallel, representing a worst-case scenario. Noise present within the receivers and frequency measurement inaccuracy can also introduce errors in the estimated position.

An incremental approach is used to compute receiver range tracks for later conversion to range differences. Initial range measurements are obtained by using the missile's initial location and the surveyed locations of each receiver site, both of which are already known. A GPS receiver is also contained at each receiver site for a time and frequency reference. During missile flight the range to the receivers varies, causing a Doppler frequency shift in the received TM signal at each receiver. The Doppler frequency shift is realized at each receiver by the difference between the received TM signal from the missile and the frequency reference derived from GPS for each time period. Using the known wavelength of the TM transmitter, this Doppler shift is converted to a change in range. Range tracks for each receiver site are obtained by integrating the changes in range over time from the initial ranges obtained before launch. These individual range tracks are converted to range differences with respect to a reference receiver from which TSPI is obtained using an algorithm that requires at least four range differences from five receivers. [Ref. 1] This algorithm is discussed and analyzed in Chapter III.

Data processing is conducted upon completion of missile flight. Data from each receiver site is transmitted to the control and processing site over an RF modem data link. The modulation is removed from the data to obtain the carrier frequency with Doppler shift for conversion to range tracks and then range differences to obtain missile TSPI. [Ref. 1] Overall operation of the TM Tracker is shown in Figure 2.2.

B. SYSTEM DESCRIPTION

Remote receiving sites and a control and processing site are used for signal reception and data processing within the TM Tracker system. Each site contains an RF modem which is slaved to the master modem at the control and processing site for system timing and inter-site communication. The control and processing site controls the TM Tracker system and contains a Pentium-II class workstation with software for remotely controlling receiver sites and downloading data from these sites for processing and conversion to missile TSPI. [Ref. 1]

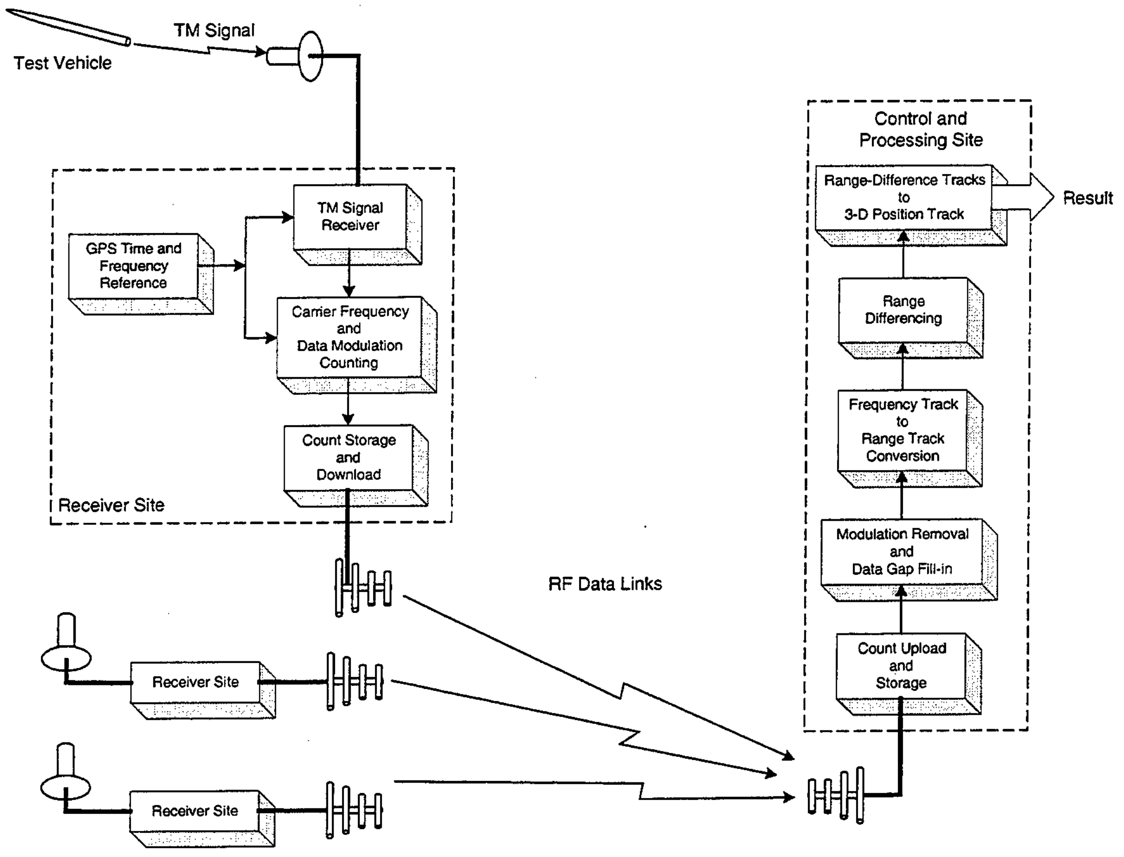
Test Vehicle
TM Signal
Receiver Site
GPS Time and Frequency Reference
TM Signal Receiver
Carrier Frequency and Data Modulation Counting
Count Storage and Download
RF Data Links
Receiver Site
Receiver Site
Control and Processing Site
Range-Difference Tracks to 3-D Position Track
Result
Range Differencing
Frequency Track to Range Track Conversion
Modulation Removal and Data Gap Fill-in
Count Upload and Storage

Figure 2.2. TM Tracker Operation Block Diagram From Ref. [1]

Prior to the test, the TM signal receiver within each receiving site is tuned to the TM frequency of 2.3 GHz with a command from the control and processing computer. The received TM signal is heterodyned to an intermediate frequency (IF) of 10 MHz, and drives the IF phase detector in the receiver while preserving any Doppler shift within the signal. The frequency shift keyed (FSK) modulation of the TM signal is also measured by the receiver with a "1" bit transmitted when the frequency is higher than the carrier, and a "0" bit transmitted when the frequency is lower than the carrier. The ratio of "1" bits and "0" bits is used to remove modulation shifts in the IF, leaving only the carrier. [Ref. 1]

All receivers within the TM Tracker system must have time and frequency synchronization to prevent errors in determining the position of the missile. Each receiver site contains a GPS receiver to provide a 10-MHz frequency reference and a one pulse-per-second time reference to the GPS satellite atomic clocks to obtain the precision necessary for measuring the missile's Doppler frequency. Each receiver is also phase-locked to the frequency reference to maintain frequency synchronization within the system. [Ref. 1]

C. FUNCTIONAL COMPONENTS

The TM Tracker system's primary functional components for signal reception and conversion to TSPI include the control and processing site, antennas, IF receiver, GPS time and frequency reference, and post processing.

1. Control and Processing Site

The control and processing site contains a computer with control and processing software, an uninterruptible power supply (UPS), and an RF modem for communication with remote receiver sites. The computer controls the primary settings and functions of all remote receivers, such as operating frequency, system status, and data collection. It also receives and processes data from these receivers for TSPI generation. High-speed computing power is not a requirement for sufficient operation of the system. A 300-MHz Pentium-II desktop and a 233-MHz Pentium-MMX laptop were both used with satisfactory system operation. [Ref. 1]

The control and processing site's distance from the remote receivers is limited by the operating range of the RF modem, approximately 20 miles. Line-of-sight visibility between the RF modem antenna and each receiver site is required for proper operation of the communication link. The link operates in the 902 - 928 MHz frequency band and has a 115.2 kilobaud data rate. [Ref. 1] Figure 2.3 shows the control and processing site utilized with a laptop computer.

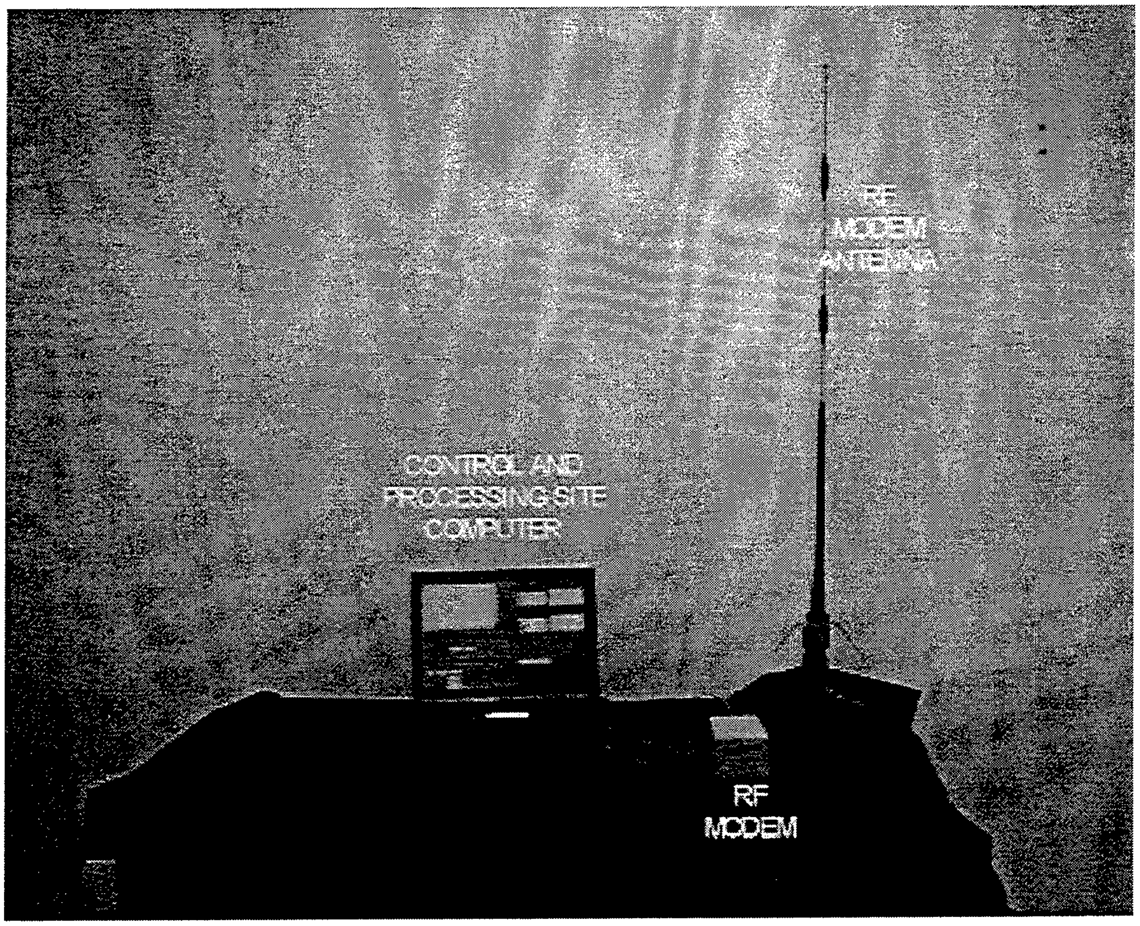

Figure 2.3. Control and Processing Site From Ref. [1]

2. Antennas

Four different antennas are used within the TM Tracker system. A 6-dB gain, S-Band helical antenna with right-hand circular polarization is used for TM signal reception. At remote receiver sites, a 10-dB gain, Yagi antenna is used for RF modem signal reception and transmission. A 5-dB gain vertical monopole antenna is utilized at the control and processing site (see Figure 2.3) for the RF modem link because of its omni-directional radiation pattern, allowing communication with all remote receiver sites. The GPS receivers at each remote site have an integrated patch antenna and low-noise amplifier combination. [Ref. 1] Figure 2.4 shows the antennas used at each remote receiver site. Figure 2.5 shows the antenna pattern for the S-Band helical antenna for TM signal reception with a 3-dB beam width of approximately 45°.

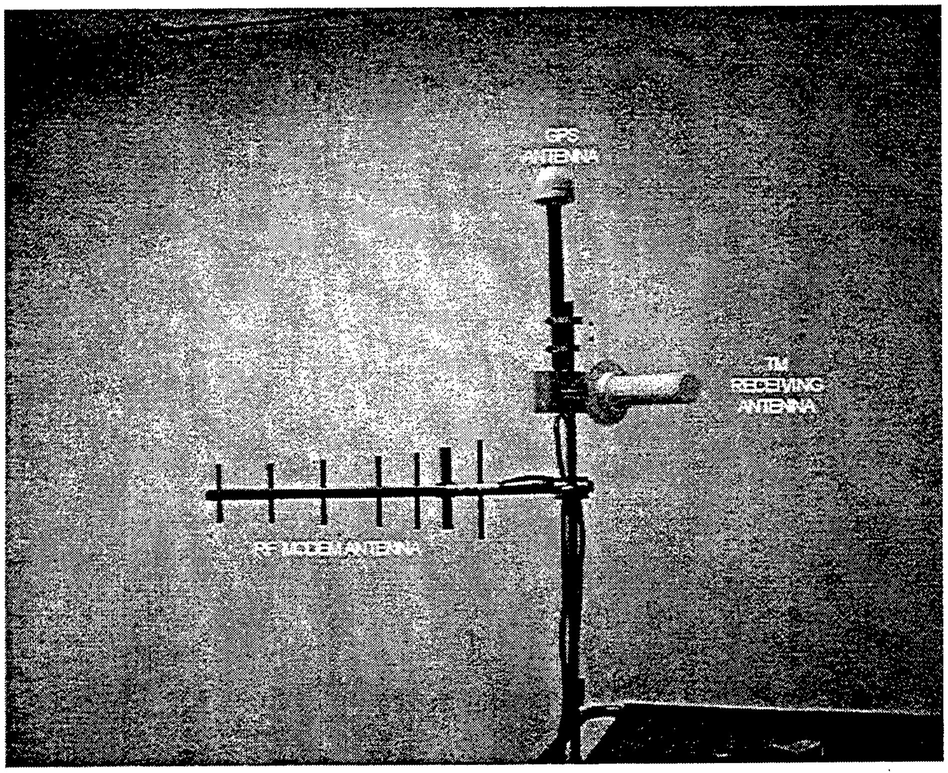

Figure 2.4. TM Tracker Antennas From Ref. [1]

3. IF Receiver

A zero-crossing counter approach of the received frequency modulated (FM) TM signal is used within the IF receiver. Because of low transmitter power, possible terrain blockage of the signal, and multipath effects, high receiver sensitivity is required. A down-converter method is utilized with an independently tunable second frequency conversion for an IF receiver passband of 1.5 MHz. Using two separate second IF channels allows for the capability to track two different transmitters with a required frequency separation of 13 MHz. [Ref. 1] Figure 2.6 shows the single channel block diagram of the receiver.

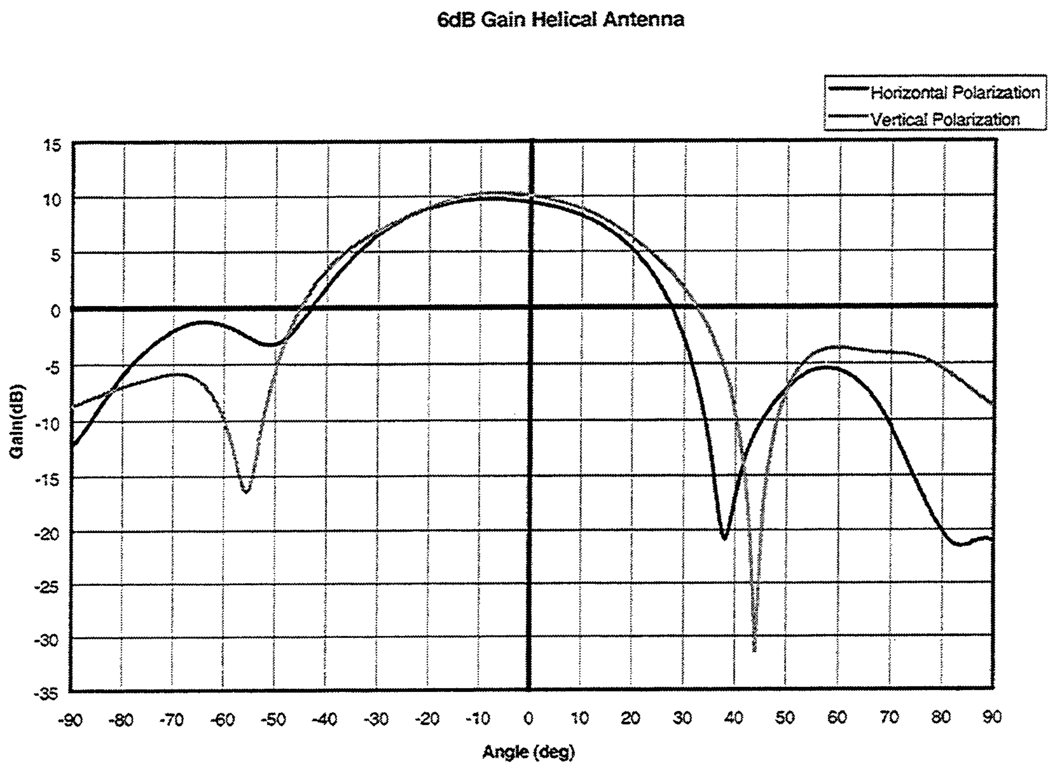

Figure 2.5. Antenna Pattern for TM Receiving Antenna From Ref [1]

A fixed 2.5-GHz local oscillator (LO) converts the 2.2 - 2.3 GHz TM band to a 200 - 300 MHz first IF band. A second LO signal is provided by a 210 - 310 MHz synthesizer, which is tunable in 20-kHz increments and forms the second IF channel centered at 10 MHz. The GPS 10-MHz reference signal is used by all LOs to provide the required synchronization between each receiver. Detailed specifications of receiver components can be found in Ref. [1].

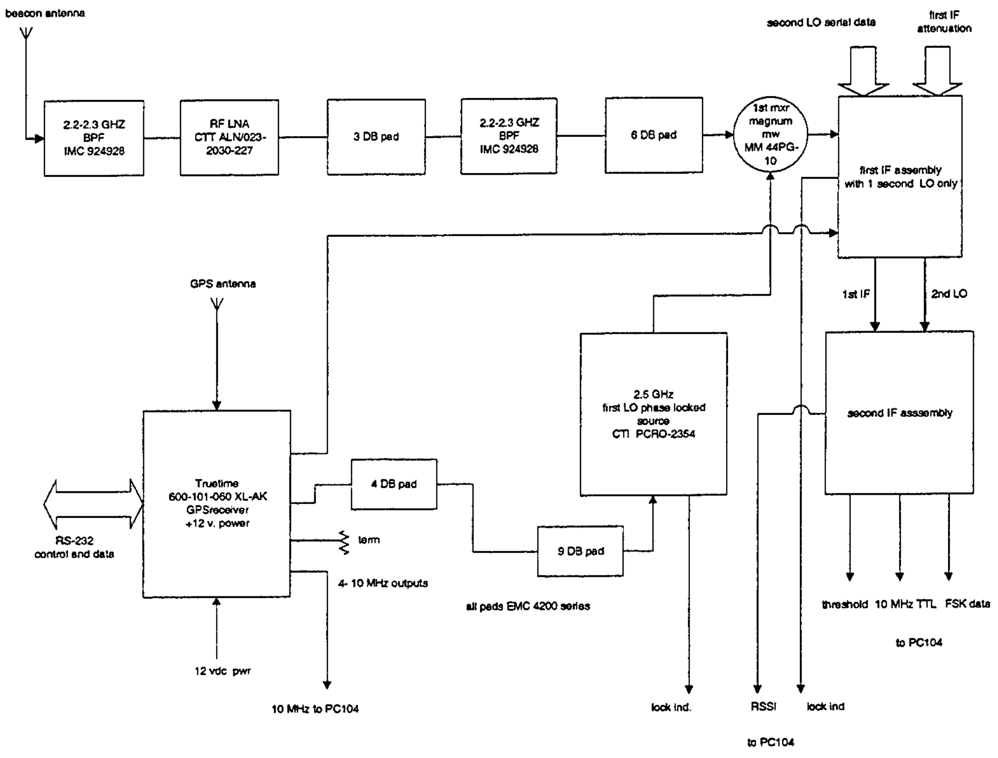
beacon antenna
2.2-2.3 GHZ BPF IMC 924928
RF LNA CTT ALN/023-2030-227
3 DB pad
2.2-2.3 GHZ BPF IMC 924928
6 DB pad
1st mxr magnum mw MM 44PG-10
second LO serial data
first IF attenuation
first IF assembly with 1 second LO only
1st IF
2nd LO
GPS antenna
2.5 GHz first LO phase locked source CTI PCRO-2354
second IF assembly
Truetime 600-101-060 XL-AK GPSreceiver +12 v. power
RS-232 control and data
4 DB pad
term
9 DB pad
4- 10 MHz outputs
all pads EMC 4200 series
threshold 10 MHz TTL FSK data
to PC104
12 vdc pwr
10 MHz to PC104
lock ind.
RSSI
lock ind
to PC104

Figure 2.6. Single Channel Receiver Block Diagram From Ref. [1]

4. GPS Time and Frequency Reference

Four 10-MHz frequency reference signals and a communications connection are provided by the TrueTime Model XL-AK GPS receiver. When power is first applied to the unit, it determines its current location and then stabilizes its 10-MHz internal oscillator to the GPS system clock. Whether tracking one or six satellites, the oscillator remains close to its stability specification of 5×10^{-11}, but experiences a large phase drift of 50 cycles when a receiver changes a satellite in the tracked constellation. During testing, drifts of one cycle between receivers were experienced between 30 and 45 seconds, translating into a frequency offset of approximately 7.5 Hz between receivers. [Ref. 1]

5. Post Processing

Post processing consists of converting the IF receiver data to range data and then converting this range data to TSPI. Each receiver's data file is analyzed to remove duplicate and unreliable data. Duplicate data is determined by analyzing each record's time stamp. Unreliable data is determined by the received signal strength indication (RSSI) voltage for each IF count within each receiver. Incremental counts with RSSI voltage levels below a specified threshold are removed, along with the previous and subsequent counts. Linear interpolation is then used to fill in the missing sample points. [Ref. 1]

Ranges from the receiver site to the missile are obtained by integrating the incremental ranges from the initial range from launch point to each receiver, requiring knowledge of the launch time. Range data from all receiver sites are then combined to compute missile TSPI. Because of the TM transmitter drift error within all range computations, a range difference algorithm is utilized for obtaining the missile's position, causing these drift errors to cancel out. The range difference algorithm for location estimation developed by Julius O. Smith and Jonathan S. Abel, explained in Chapter III, is used to compute missile TSPI.

D. TM TRACKER FIELD TESTING

The TM Tracker was field tested at White Sands Missile Range during two testing programs in August and September 1999. In the first test a TM transmitter was mounted on a cable car rolling on the aerial cable of the Aerial Cable Range (ACR). Because of the transmitter location, several receiver sites were unable to receive the TM signal resulting in unreliable data. [Ref. 1]

The second test occurred during September and October 1999. Missiles with TM sections were fired against targets attached to the cable car on the ACR. Two launch points were utilized, with approximate ranges of 3.0 km and 4.7 km from the target. Nine receiver sites were used during this test. Three receiver sites, R1a, R1b, and R1p, were placed near the launch site and shared the same receiver site enclosure. The other six receivers were placed at other positions on the test site. [Ref. 1] The two-dimensional test-site layout and the layout for the primary receiver (R1) and 3-km launch point are shown in Figures 2.7 and 2.8, respectively. Specific coordinates of each receiver are listed in Table 2.1.

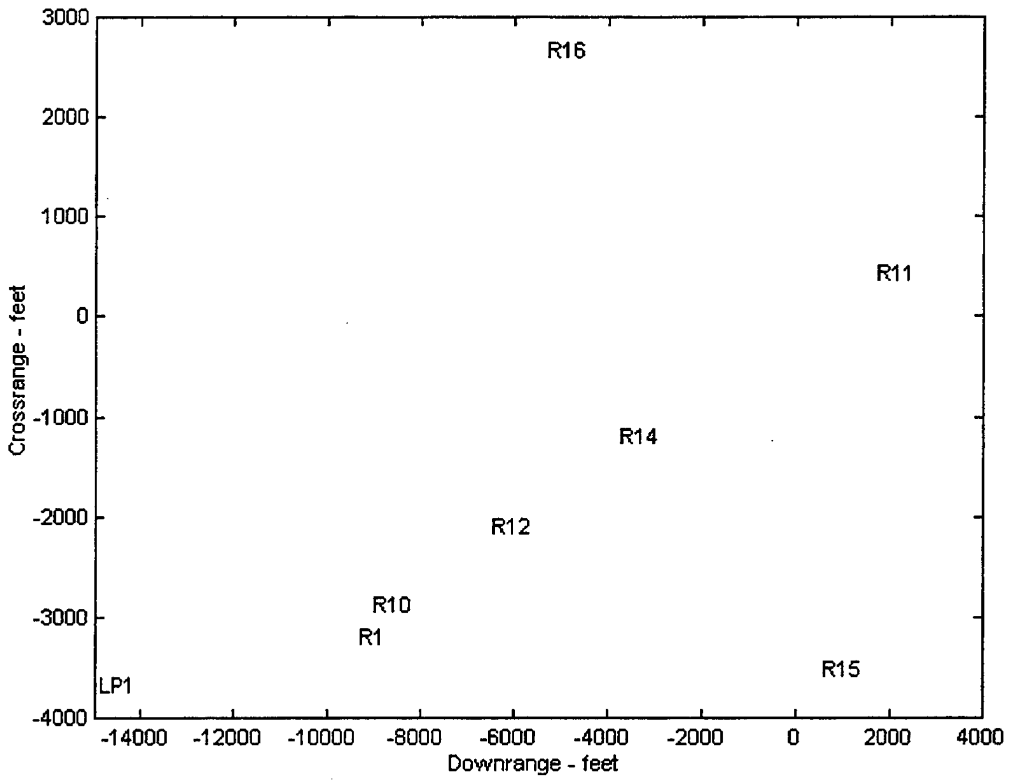

Figure 2.7. Receiver Locations From Ref. [1]

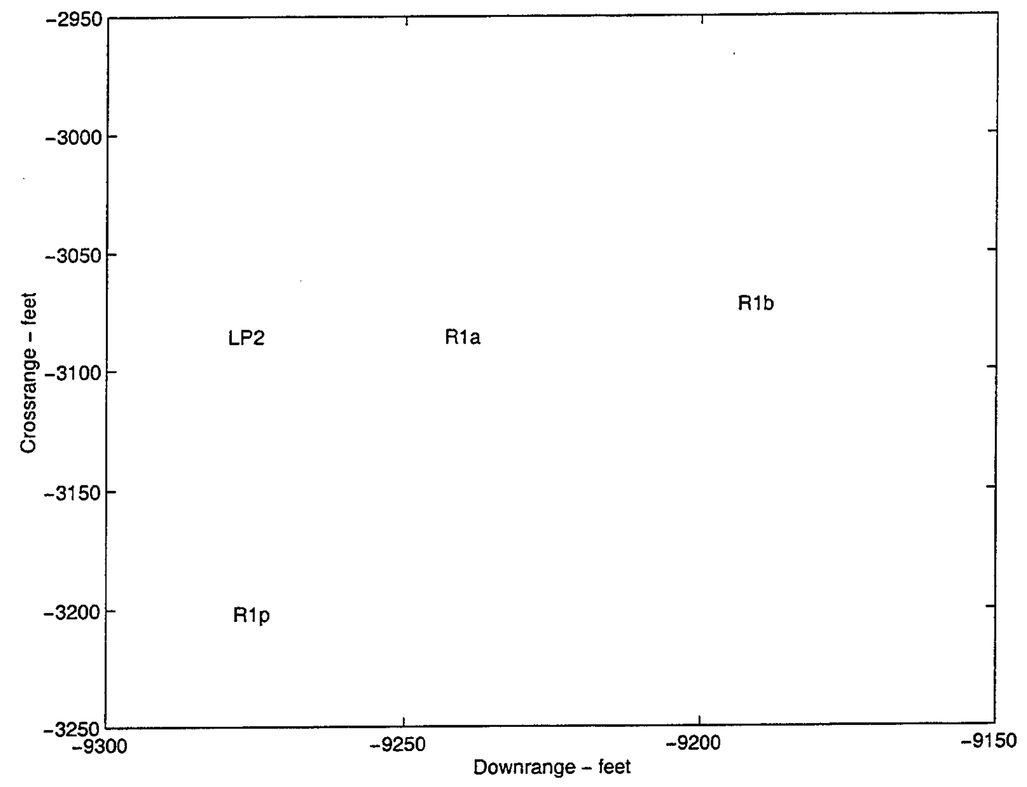

Figure 2.8. Detailed Primary Receiver and Launcher Locations From Ref. [1]

	X (downrange) feet	Y (crossrange) feet	Z (elevation) feet
Launch Point 1 (4.7 km)	-14863.3	-3676.5	-250.2
Launch Point 2 (3.0 km)	-9279.7	-3085.7	-122.3
R1 a	-9243.4	-3085.7	-127.0
R1 b	-9193.6	-3072.3	-125.0
R1 p	-9278.7	-3201.8	-92.1
R10	-8992.9	-2879.8	-109.3
R11	1757.0	430.4	40.4
R12	-6458.5	-2112.3	-39.3
R14	-3704.2	-1214.7	-33.6
R15	607.0	-3517.8	208.3
R16	-5305.4	2653.0	-153.6

Table 2.1. Receiver Site Antennas and Launch Locations From Ref. [1]

Receivers near the launcher, R1a, R1b, R1p, and R10, which provide the initial missile TSPI, experienced interference with the received TM signal around launch time. Several sites also did not receive a strong enough signal at launch time, preventing the determination of the TM receiver frequency offset at these sites. TSPI was only calculated using receiver sites R1p, R11, R12, and R16, after the interference ended, while the laser tracker data was used for initial missile TSPI. [Ref. 1]

A 2-second interval of data with minimal receiver dropouts was examined. To compare missile TSPI accuracy, laser-tracking data of the missile from the Sandia Labs' Laser Tracker was utilized. Figure 2.9 illustrates the total position error for *x* (downrange) and *y* (crossrange) dimensions during the 2-second interval. TSPI data of altitude (*z*-dimension) was unreliable and is not shown. [Ref. 1]

The results of the live fire tests were far from optimal, with a mean position error of approximately 70 feet. The TM Tracker's ability to obtain 1-meter position accuracy requires the understanding and implementation of a source estimation algorithm within its system components. Two algorithms that utilize range differences to estimate source location are discussed and analyzed in the next chapter, one of which is used within the TM Tracker's post processing system.

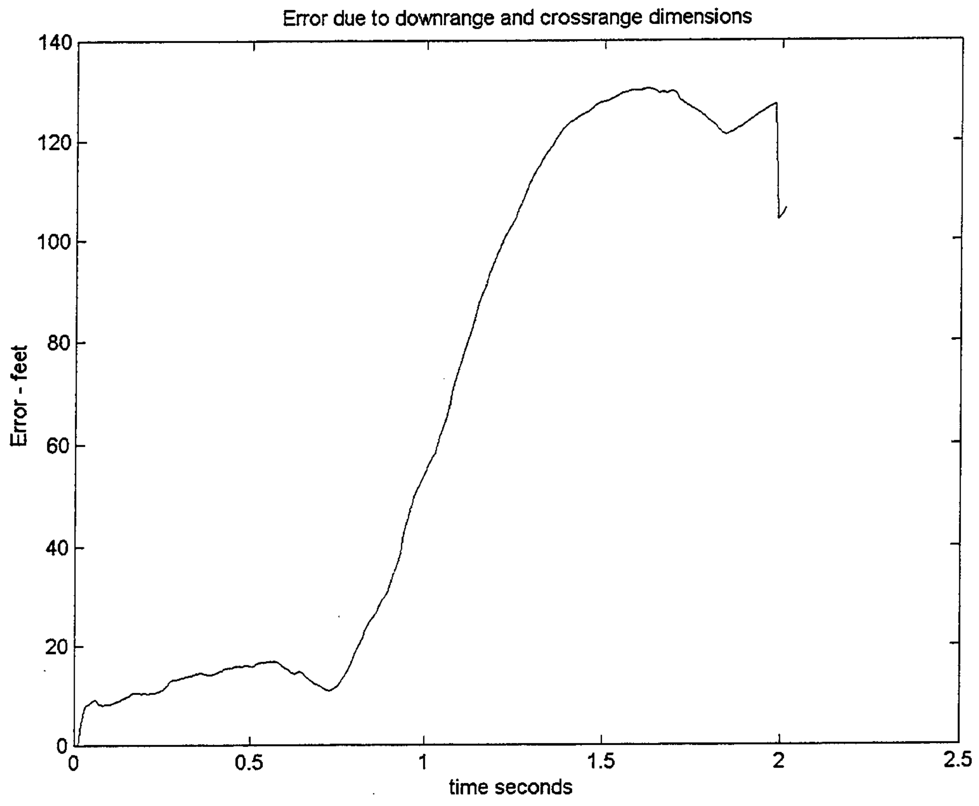

Figure 2.9. TSPI Accuracy From Ref. [1]

III. SOURCE ESTIMATION USING RANGE DIFFERENCES

A. INTRODUCTION

Position estimation of a radiating source by analyzing the signals received at a number of arbitrarily spatially separated receiving stations using the range differences of the propagating signals has been extensively studied [Refs. 2-5]. Signals propagating from the source arrive at the receiving stations at different times depending on the transmitter-to-receiver geometry and the medium characteristics through which the signal is propagating. To obtain range, signal arrival times are developed at all receiver stations relative to a designated reference receiver station. These time difference of arrival (TDOA) measurements are then transformed into range differences between receiver stations, resulting in a set of nonlinear hyperbolic equations whose solution is the estimated position of the radiating source.

Many algorithms have been developed to solve the nonlinear hyperbolic equations [Refs. 2-5]. Algorithms developed by Bertrand Fang [Ref. 2] and Julius Smith and Jonathan Abel [Ref. 3] were used in MATLAB simulations (Chapter IV) and are discussed in this chapter. Regardless of the method utilized, reliable TDOA measurements mandate precisely synchronized clocks at the receiver stations, hence, the use of GPS receivers in the TM Tracker system [Ref. 1].

B. FANG'S ALGORITHM

The source estimation algorithm developed by Fang [Ref. 2] was utilized during preliminary feasibility studies for 1-meter position accuracy of the TM Tracker system. A system with three receiver stations was used, with the origin centered on one of the stations as shown in Figure 3.1.

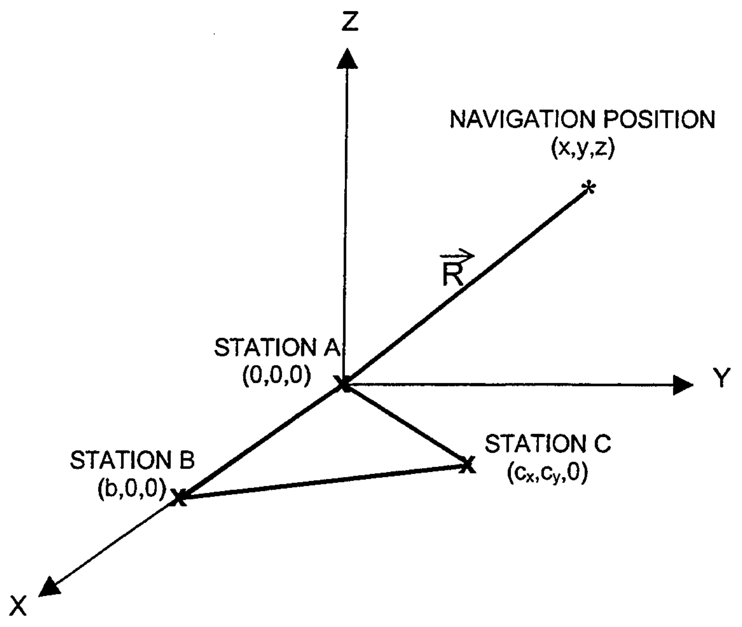

Figure 3.1. Receiver Stations and Estimated Position

From Figure 3.1 the range differences from the navigation position to station pairs A, B and A, C, are R_{ab} and R_{ac}, respectively, and are calculated as

$$\sqrt{x^2+y^2+z^2}-\sqrt{(x-b)^2+y^2+z^2}=R_{ab} \tag{3.1}$$

$$\sqrt{x^2+y^2+z^2}-\sqrt{(x-c_x)^2+(y-c_y)^2+z^2}=R_{ac} \tag{3.2}$$

Squaring and simplifying (3.1) and (3.2) yields

$$R_{ab}^2-b^2+2b*x=2R_{ab}\sqrt{x^2+y^2+z^2} \tag{3.3}$$

$$R_{ac}^2-c^2+2c_x*x+2c_y*y=2R_{ac}\sqrt{x^2+y^2+z^2} \tag{3.4}$$

where b and c are the station ranges, $c=\sqrt{c_x^2+c_y^2}$, and $*$ denotes multiplication.

Equating (3.3) and (3.4) and simplifying results in

$$y=g*x+h \tag{3.5}$$

where

$$g = \frac{R_{ac} * (b / R_{ab}) - c_x}{c_y} \tag{3.6}$$

$$h = \frac{c^2 - R_{ac}^2 + R_{ac} * R_{ab}\left(1 - (b / R_{ab})^2\right)}{2c_y} \tag{3.7}$$

Substituting (3.5) into (3.3) results in

$$d * x^2 + e * x + f = z^2 \tag{3.8}$$

where

$$d = -\left\{1 - (b / R_{ab})^2 + g^2\right\} \tag{3.9}$$

$$e = b * \left\{1 - (b / R_{ab})^2\right\} - 2g * h \tag{3.10}$$

$$f = (R_{ab}^2 / 4) * \left\{1 - (b / R_{ab})^2\right\}^2 - h^2 \tag{3.11}$$

The navigation position vector can now be written as a function of *x* and defines a hyperbola (*d* > 0) or an ellipse (*d* < 0) with mirror symmetry with respect to the receiver station plane

$$\vec{R} = x * \vec{i} + (g * x + h)\vec{j} \pm \sqrt{d * x^2 + e * x + f}\,\vec{k} \tag{3.12}$$

By restricting the navigation position to be on a different plane than the base stations results in the position becoming the solution to a quadratic equation. For position estimation in two dimensions *x* and *y*, the solution to (3.8) for *x* is found by applying the Quadratic Formula

$$x = \frac{-e \pm \sqrt{e^2 - 4df}}{2d} \tag{3.12}$$

The ambiguity of the two roots in (3.12) can be resolved if knowledge of the general location is known. The solution for *y* is found using (3.5).

Another method used to obtain the navigation position is by utilizing a fourth base station, Station C′ (c_x', c_y', *0*), which results in a navigation position that is at the intersection of three hyperboloids or ellipsoids depending on the value of *d*. Computation follows as before with the addition of another range difference from the added base station to the reference station centered at the origin

$$\sqrt{x^2+y^2+z^2}-\sqrt{(x-c_x')^2+(y-c_y')^2+z^2}=R_{ac}' \tag{3.13}$$

Two instances of (3.8) result for Station C and Station C′

$$d*x^2+e*x+f=z^2 \tag{3.14}$$

$$d'*x^2+e'*x+f'=z^2 \tag{3.15}$$

where *d′*, *e′*, and *f′* are computed the same way as *d*, *e*, and *f* in (3.9) - (3.11), substituting R'_{ac}, c_x', and c_y' for R_{ac}, c_x, and c_y for calculating *g* and *h* in (3.6) and (3.7). Subtracting (3.15) from (3.14) results in an equation in terms of *x*

$$(d-d')*x^2+(e-e')*x+(f-f')=0 \tag{3.16}$$

Solving for *x* results in

$$\frac{-(e-e')\pm\sqrt{(e-e')^2-4*(d-d')*(f-f')}}{2(d-d')} \tag{3.17}$$

Solutions for *y* and *z* follow from (3.5) and (3.8).

Simulations using Fang's algorithm with four receiver stations are discussed in Chapter IV. While Fang's algorithm provides an exact solution to the hyperbolic equations, it cannot take advantage of utilizing range differences from additional receivers. A maximum of three range differences can be used with four receivers. The TM Tracker, however, has the potential of eight range differences available from nine receiver sites. Therefore, a range difference algorithm that can use all available range differences in determining position must be used.

C. SMITH – ABEL ALGORITHM

The algorithm for source location developed by Julius Smith and Jonathan Abel [Ref. 3] uses a spherical interpolation method with linear least-squares error minimization. It is a closed-form method that uses range differences referenced to a single receiver (reference receiver) with $N-1$ range differences available within a constellation of N receivers. This algorithm is used within the TM Tracker's post-processing system.

Let N denote the number of receivers and let d_{ij} denote the range difference between the ith and jth receivers and the source. The vectors of (x, y, z) spatial coordinates for the ith receiver and the source are denoted $\underline{x}_i$ and $\underline{x}_j$ respectively. The distance between the ith receiver and the source is defined by

$$D_i = |\underline{x}_i - \underline{x}_s| \tag{3.18}$$

and the distances from the origin to the points $\underline{x}_i$ and $\underline{x}_j$ are R_i and R_s.

The reference receiver is arbitrarily chosen as Receiver 1 and the coordinate system origin is translated to this receiver giving

$$\underline{x}_1 = \underline{0} \Rightarrow R_1 = 0 \text{ and } D_1 = |\underline{x}_s| = R_s \tag{3.19}$$

Figure 3.2 illustrates the notations and geometric relations between the reference receiver (Receiver 1), the ith receiver (Receiver i), and the source. The distance between the ith receiver and the source, D_i, is found by adding the range difference between the reference receiver and the *i*th receiver to distance from the reference receiver to the source.

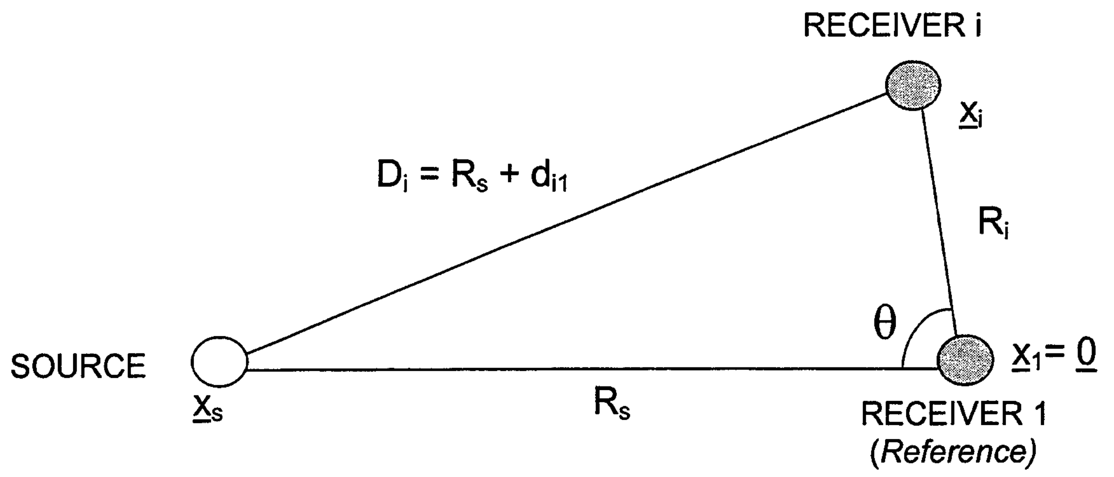

Figure 3.2. Notation and Geometric Relations for *i*th Receiver

Using the law of cosines, D_i can be expressed as

$$D_i^2 = (R_s + d_{i1})^2 = R_i^2 + R_s^2 - 2 * |\underline{x}_i| * |\underline{x}_s| cos\theta \tag{3.20}$$

where θ is the angle between the vectors $\underline{x}_i$ and $\underline{x}_j$. The angle θ between the two vectors $\underline{x}_i$ and $\underline{x}_s$ is defined by

$$cos\theta = \frac{\underline{x}_i^T \underline{x}_s}{|\underline{x}_i||\underline{x}_s|} \tag{3.21}$$

Substituting (3.21) into (3.20) and cancelling out terms results in

$$(R_s + d_{i1})^2 = R_i^2 + R_s^2 - 2\underline{x}_i^T \underline{x}_s \tag{3.22}$$

Expanding and shifting terms to the right-hand side of the equation yields

$$\varepsilon_i = R_i^2 - d_{i1}^2 - 2R_s d_{i1} - 2\underline{x}_i^T \underline{x}_s \tag{3.23}$$

where ε_i is the introduced equation error due to imprecise measurements and is minimized in a least-square sense to provide an estimate of the true source position. Equation (3.23) can be written in matrix form for $N - 1$ range differences as

$$\underline{\varepsilon} = \underline{\delta} - 2R_s\underline{d} - 2\boldsymbol{S}\underline{x}_s \tag{3.24}$$

where

$$\underline{\delta} \equiv \begin{bmatrix} R_2^2 - d_{21}^2 \\ R_3^2 - d_{31}^2 \\ \vdots \\ R_N^2 - d_{N1}^2 \end{bmatrix}, \quad \underline{d} = \begin{bmatrix} d_{21} \\ d_{31} \\ \vdots \\ d_{N1} \end{bmatrix}, \quad \boldsymbol{S} = \begin{bmatrix} x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \\ \vdots & \vdots & \vdots \\ x_N & y_N & z_N \end{bmatrix}$$

Here $\boldsymbol{S}$ represents the matrix of receiver spatial coordinates after translation of the reference receiver to the origin. The least-squares solution for $\underline{x}_s$ given R_s is

$$\underline{x}_s = \frac{1}{2}\boldsymbol{S}_W^*\left(\underline{\delta} - 2R_s\underline{d}\right) \tag{3.25}$$

where * denotes complex conjugation and complex conjugate

$$\boldsymbol{S}_W^* = \left(\boldsymbol{S}^T\boldsymbol{S}\right)^{-1}\boldsymbol{S}^T \tag{3.26}$$

yields the minimizer of $\underline{\varepsilon}^T\underline{\varepsilon}$. The range differences can be weighted according to the relative confidence of each range difference. Weights of zero can be used when certain range differences are not utilized. This technique was used to simulate when the receiver does not have the missile within its antenna beam (Chapter IV). This weighted equation error is $\underline{\varepsilon}^T\boldsymbol{W}\underline{\varepsilon}$ and is minimized for

$$\boldsymbol{S}_W^* = \left(\boldsymbol{S}^T\boldsymbol{W}\boldsymbol{S}\right)^{-1}\boldsymbol{S}^T\boldsymbol{W} \tag{3.27}$$

To obtain the source position $\underline{x}_s$ by minimizing $\underline{\varepsilon}^T\boldsymbol{W}\underline{\varepsilon}$, R_s must be allowed to vary while maintaining the relation $R_s = |\underline{x}_s|$ because the range from the reference receiver to the source is not typically known in advance, resulting in nonlinear minimization. This nonlinearity can be approximately eliminated by the application of the spherical interpolation method.

The goal of the spherical interpolation method is to minimize the equation error with respect to R_s. Substitution of (3.25) into (3.24) results in a linear least-squares problem for finding R_s due to the fact that the least-squares estimate of $\underline{x}_s$ given R_s in (3.25) is linear in R_s.

$$\underline{\varepsilon}' = \underline{\delta} - 2R_s\underline{d} - \boldsymbol{SS}_W^*\left(\underline{\delta} - 2R_s\underline{d}\right) \tag{3.28}$$

where ε' is the new equation error which is linear in R_s. Factoring out the common term $\underline{\delta}$ $-2R_s\underline{d}$ results in

$$\underline{\varepsilon}' = \left(\boldsymbol{I} - \boldsymbol{SS}_W^*\right)\left(\underline{\delta} - 2R_s\underline{d}\right) \tag{3.29}$$

where $\boldsymbol{I}$ is the identity matrix.

It follows from (3.27) that

$$\mathbf{SS}_W^* = \mathbf{S}\left(\mathbf{S}^T\mathbf{WS}\right)^{-1}\mathbf{S}^T\mathbf{W} = \mathbf{P}_S \tag{3.30}$$

and

$$\mathbf{P}_S^\perp = \mathbf{I} - \mathbf{P}_S = \mathbf{I} - \mathbf{SS}_W^* \tag{3.31}$$

where $\boldsymbol{SS}_W^*$ defines the N - 1 by N - 1 symmetric matrices, and $\boldsymbol{P}_S$ and $\boldsymbol{P}_S^\perp$ are rank 3 projection matrices that remove both orthogonal components and components within the space spanned by the columns of $\boldsymbol{S}$, respectively.

If four receivers are utilized, three range differences are produced and $\boldsymbol{P}_S = \boldsymbol{I}$. The equation error $\underline{\varepsilon}'$ becomes zero for all values of R_s resulting in ambiguity for the source location. Because of this limitation, a minimum of five receivers producing four range differences must be used for this method to be effective in non-linear minimization.

Assuming the use of at least five receivers and four range differences, substitution of (3.31) into (3.29) results in

$$\underline{\varepsilon}' = \boldsymbol{P}_S^{\perp}(\underline{\delta} - 2R_s\underline{d}) \tag{3.32}$$

Utilizing the weighting matrix $\boldsymbol{W}$ produces

$$\underline{\varepsilon}'^T\mathbf{W}\underline{\varepsilon}' = (\underline{\delta} - 2R_s\underline{d})^T \boldsymbol{P}_S^{\perp}\boldsymbol{W}\boldsymbol{P}_S^{\perp}(\underline{\delta} - 2R_s\underline{d}) \tag{3.33}$$

A form of weighted least squares results when minimizing (3.33) with respect to R_s where the weighting matrix $\mathbf{P}_S^{\perp}\mathbf{W}\mathbf{P}_S^{\perp}$ is of rank *N - 4*. The three missing dimensions within this matrix result from the degrees of freedom removed by substituting (3.25) in (3.24) for the spatial coordinates of the source $\underline{x}_s$. The minimizing value of R_s for (3.33) is

$$\tilde{R}_s = \frac{\underline{d}^T \boldsymbol{P}_S^{\perp}\boldsymbol{W}\boldsymbol{P}_S^{\perp}\underline{\delta}}{2\underline{d}^T \boldsymbol{P}_S^{\perp}\boldsymbol{W}\boldsymbol{P}_S^{\perp}\underline{d}} \tag{3.34}$$

Substituting (3.34) into (3.25) results in the source location estimate as

$$\underline{x}_s = \frac{1}{2}\boldsymbol{S}_W^*\left(\underline{\delta} - 2\tilde{R}_s\underline{d}\right) = \frac{1}{2}\left(\boldsymbol{S}^T\boldsymbol{W}\boldsymbol{S}\right)^{-1}\boldsymbol{S}^T\boldsymbol{W}\left(\underline{\delta} - 2\tilde{R}_s\underline{d}\right) \tag{3.35}$$

The source's coordinates are based on a coordinate system with the reference receiver at the origin. To obtain the source's true spatial coordinates, the coordinate system must be shifted back by translation.

The Smith-Abel algorithm for source location estimation was utilized in all MATLAB simulations involving the TM Tracker. It was programmed into MATLAB and used as the function *smith_abel.m*. The MATLAB source code for this function is listed in Appendix A.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. MATLAB SIMULATIONS

A. BACKGROUND

Several questions were addressed in evaluating the potential for achieving 1-meter TSPI accuracy with the TM Tracker in its current configuration, including:

1. What range difference measurement accuracy is required at each remote receiver site to provide 1-meter missile TSPI accuracy?
2. What signal-to-noise ratio (SNR) is required at the remote receiver sites to obtain 1-meter missile TSPI accuracy when all receivers are used during the entire missile trajectory?
3. What are the effects of the receivers' antenna beam widths on TSPI accuracy?
4. How can the receivers be utilized so that during the entire missile trajectory at least five receivers are able to receive the telemetry signal?
5. What SNR is required at the receivers to achieve 1-meter TSPI when only those that can receive the telemetry signal within their respective 3-dB beam widths are utilized?

These questions were answered by modeling and conducting simulations of the TM Tracker system. MATLAB source codes for all simulations are listed in Appendix A.

B. PRELIMINARY SIMULATIONS

Simulations in MATLAB using Fang's algorithm [Ref. 2] were conducted to evaluate the required measurement accuracy and limitations of the TM Tracker system. The initial approach utilized four receiver sites in the *X-Y* plane that were placed behind the missile trajectory for continuous TM signal reception throughout missile flight. A simple missile trajectory consisting of a Mach 1 missile flying at a constant altitude of 1 km was used. Figure 4.1 shows the receiver locations and the missile trajectory used in preliminary simulations.

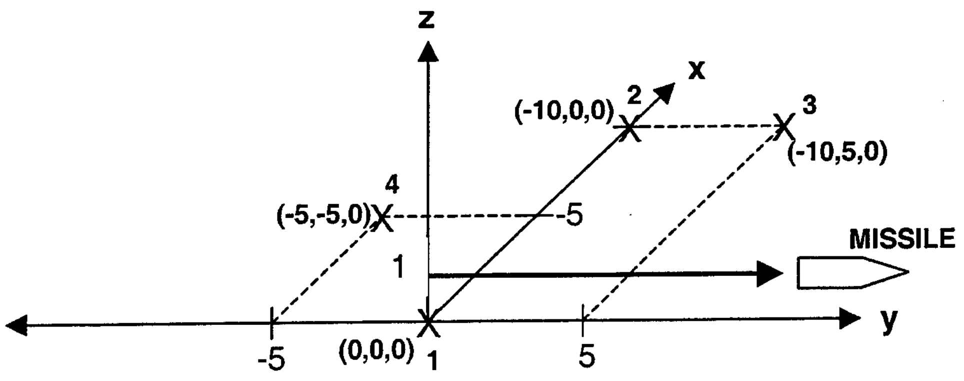

Figure 4.1. Simple Missile Trajectory with Four Receiver Sites

TDOA and resulting range differences were obtained using the frequency difference of arrival (FDOA) method by Chestnut. [Ref. 5] The theoretical lower bound on Doppler frequency measurement was used, described by the following equation:

$$f = \frac{\sqrt{\frac{3}{SNR}}}{\pi \cdot T} \tag{4.1}$$

where *SNR* is the signal-to-noise ratio of the receiver at which the frequency measurement is made and *T* is the sampling period. [Ref. 6] The theoretical position accuracy (mean error) for a given value of SNR was computed using this model. Monte Carlo simulation was utilized due to the random noise added to the TDOA measurements within the algorithm, simulating real world conditions (*fdoafang4nmc.m*). Figure 4.2 shows the mean error for SNR values from 15 – 40 dB along with a curve fit of the data. The results from this model established the feasibility of obtaining 1-meter position accuracy using TDOA and range differences, provided that sufficient SNR values are obtained at each receiver.

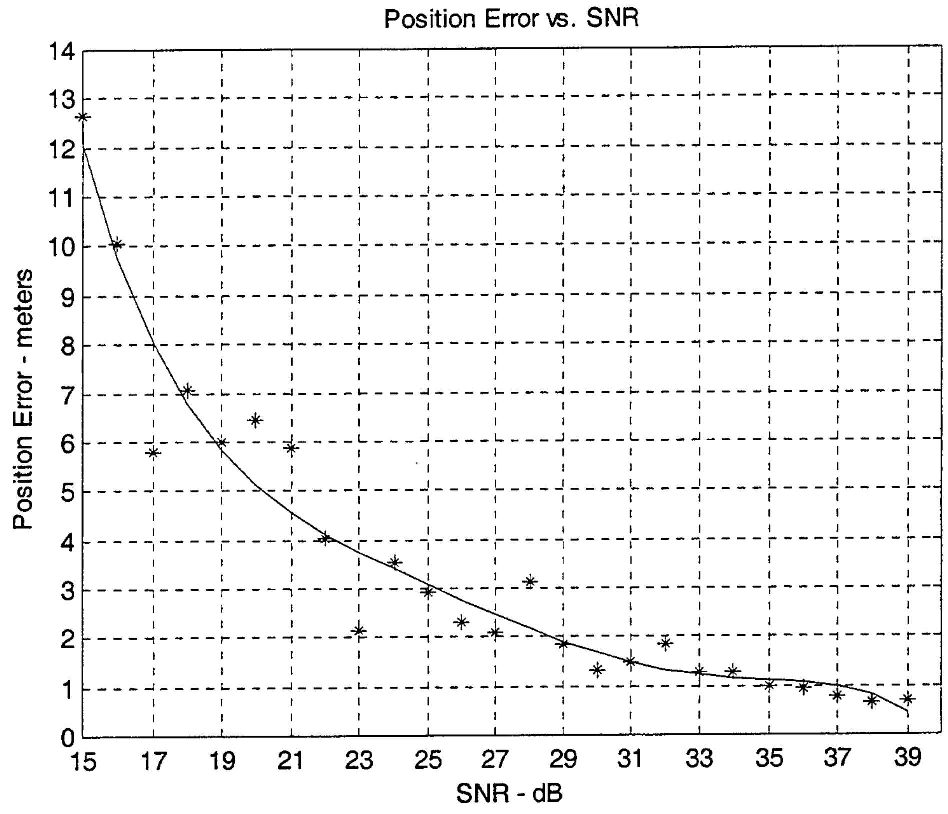

Figure 4.2. Position Error vs. SNR

The effects of TDOA measurement accuracy on position accuracy were also computed using Fang's algorithm and the receiver site configuration in Figure 4.1. A Monte Carlo simulation using 10,000 iterations for each value of root-mean-squared (RMS) TDOA measurement error was used to compute *x*-position, *y*-position, *z*-position, and total position error (*tdoafang4nmc.m)*, illustrated in Figures 4.3 - 4.6, respectively. Introduced errors in TDOA produced range difference inaccuracies which, when utilized in Fang's algorithm, resulted in increased errors in position.

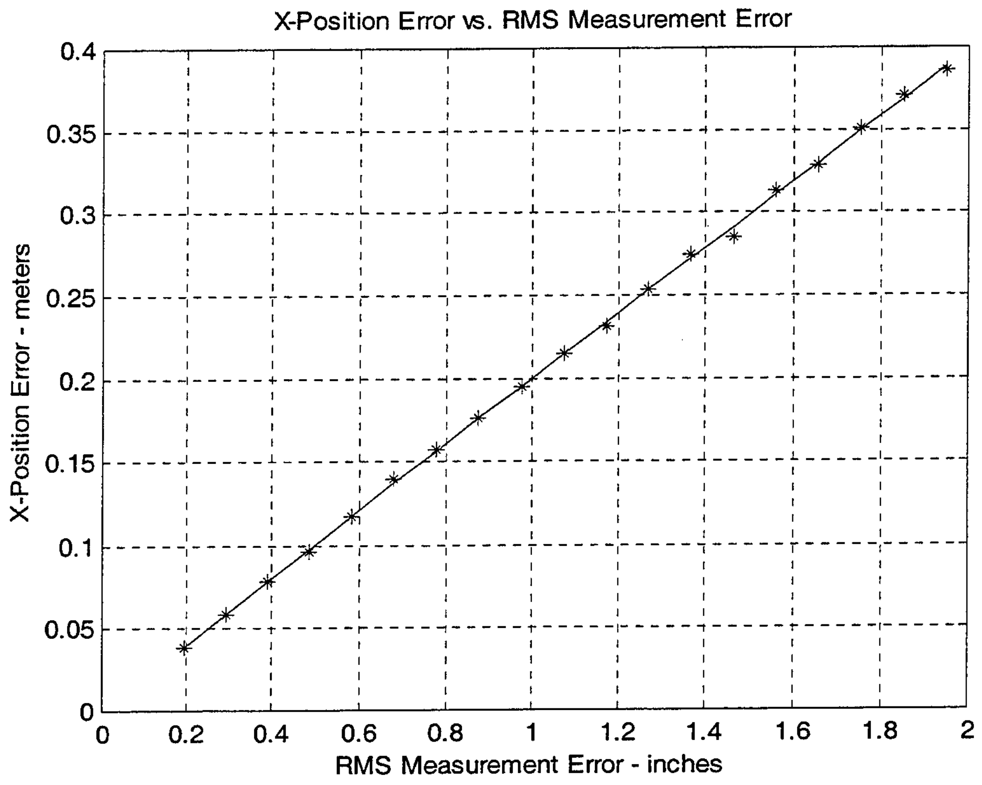

Figure 4.3. *X*-Position Error vs. RMS Measurement Error

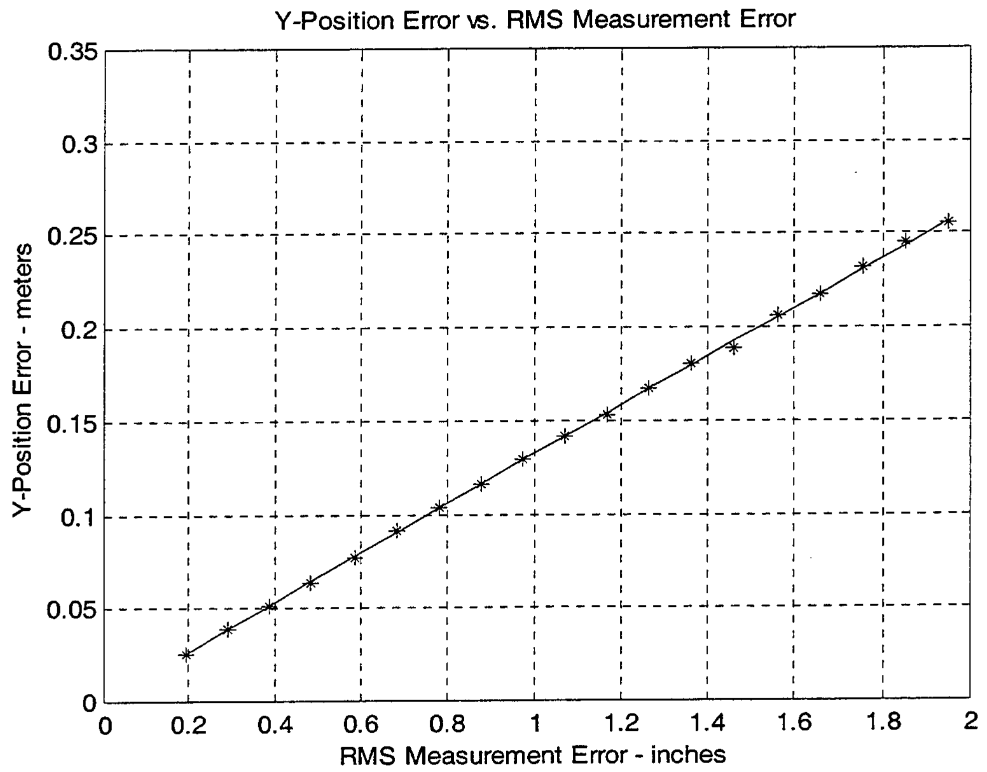

Figure 4.4. *Y*-Position Error vs. RMS Measurement Error

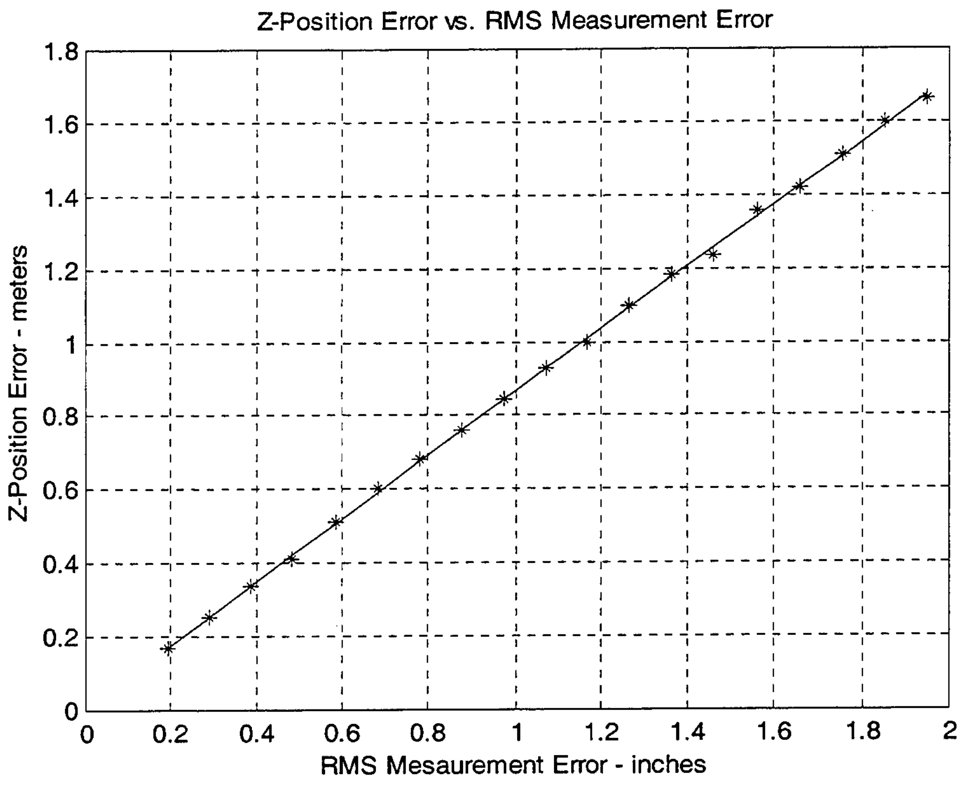

Figure 4.5. *Z*-Position Error vs. RMS Measurement Error

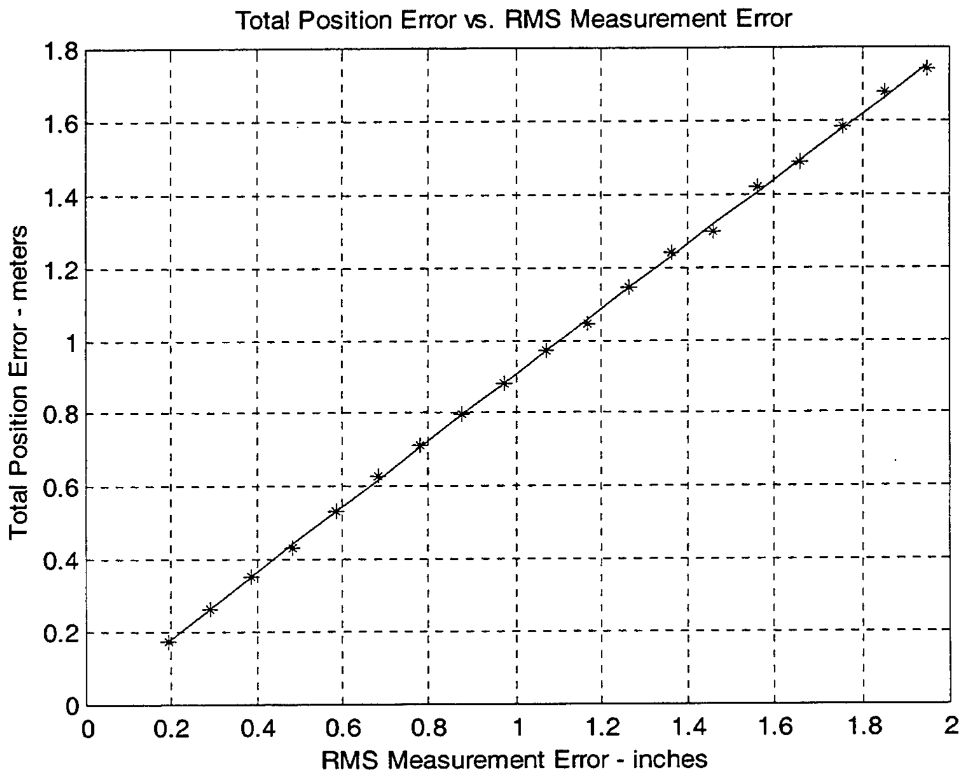

Figure 4.6. Total Position Error vs. RMS Measurement Error

As shown in the previous figures, TDOA measurement accuracy must be on the order of 1.2 inches to provide 1-meter position accuracy, with the *z*-position accuracy being the most critical measurement for this trajectory. The Fang algorithm, however, can only be used with a four-receiver configuration and does not allow the addition of more receivers for possible improved TSPI accuracy. Because of this limitation, the Smith - Abel algorithm was utilized for all subsequent simulations of the TM Tracker, allowing for the use of nine available receiver sites that were on the missile range.

C. TM TRACKER SIMULATIONS

1. Missile Trajectory and Remote Receiver Locations

The remote receiver configuration and the missile trajectory used during TM Tracker testing at White Sands Missile Range were utilized for all simulations and are depicted in Figure 4.7. Receivers are numbered corresponding to Figure 2.8.

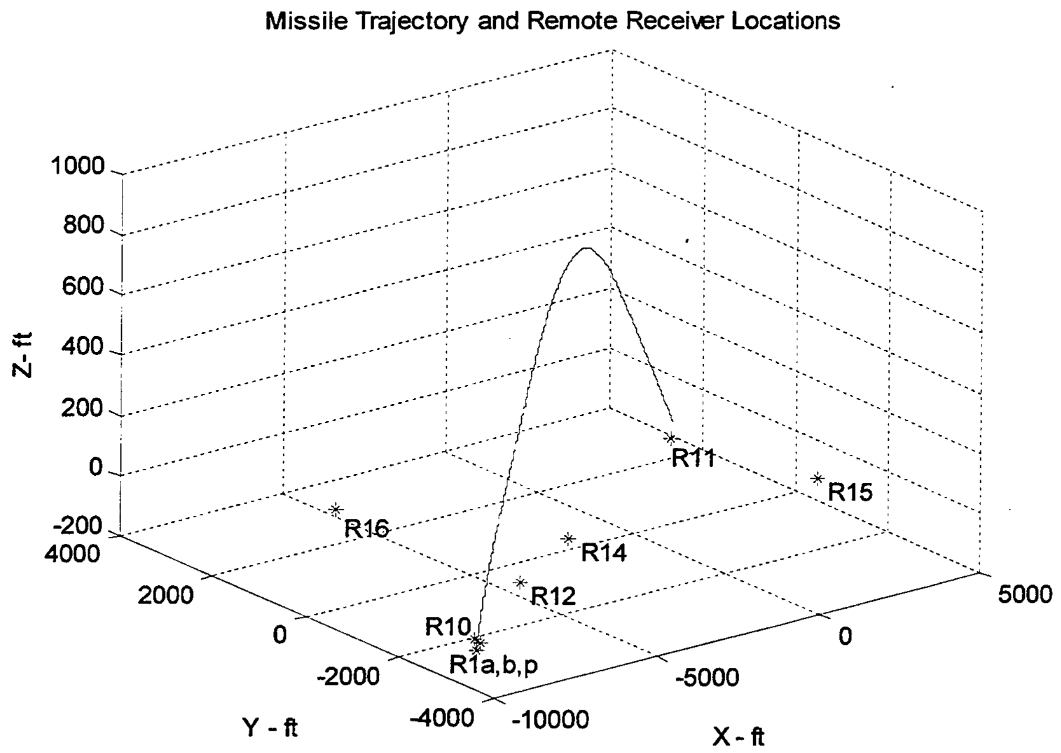

Figure 4.7. Missile Trajectory and Remote Receiver Locations

2. Computer Algorithm Implementation

a. Trajectory Data

The heart of the project was to simulate operation of the TM Tracker using missile trajectory data provided from the laser tracker on the missile range. Spatial coordinates and time data were resolved by the laser tracker at intervals of 0.01 sec. The trajectory data from the laser tracker was stored in a 4×681 matrix. The four rows within this matrix represent the time t in seconds and the missile's spatial coordinates x, y, and z, in feet. The 681 columns represent time and position data of the missile, with the first and last columns representing $t = 0.01$ sec and $t = 6.81$ sec, respectively. Data up through $t = 6.42$ sec was used in all TM Tracker simulations due to ambiguities encountered in the data near the end of the missile's trajectory.

b. Missile Velocity Components

The trajectory data provided by the laser tracker on the White Sands Missile Range was processed through a SIMULINK model and revealed a 16-Hz modulation component in the x, y, and z-component velocities, relating to the missile's roll rate. Because Doppler differential measurements were to be used for obtaining TDOA and range differences, the velocity components had to be smoothed to eliminate missile roll-rate modulation effects. Figures 4.8 and 4.9 show missile velocity components before and after a 14th degree polynomial least-mean-squares fit was applied to the trajectory data. Velocity components were found by computing the derivatives of the three equations for missile position with respect to time

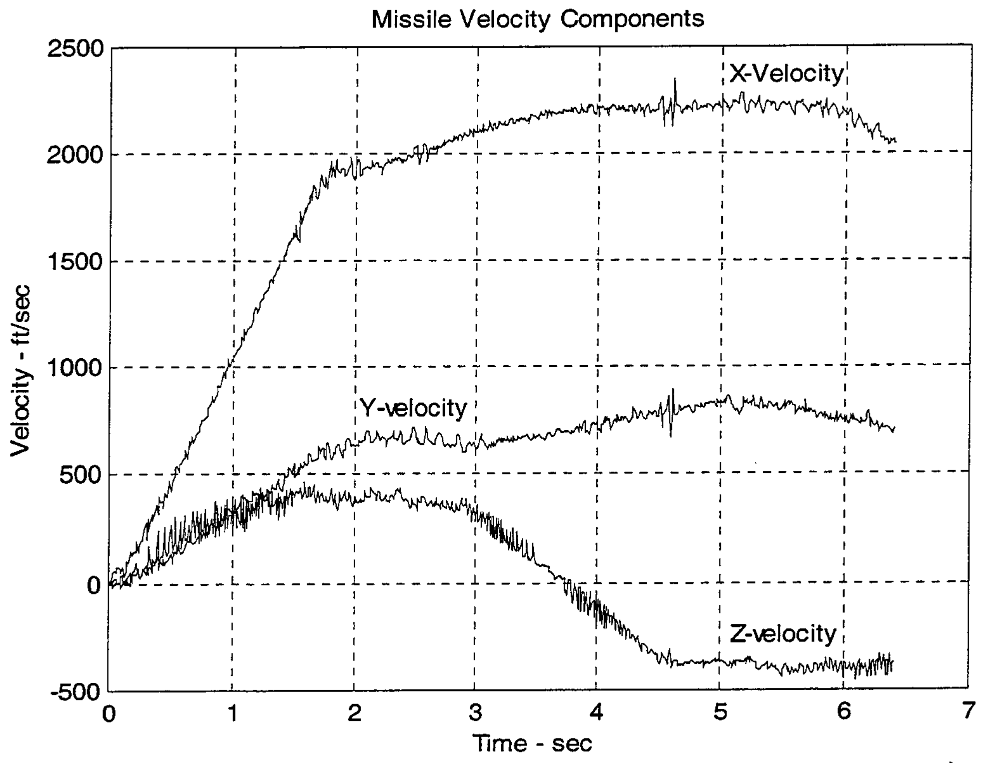

Figure 4.8. Missile Velocity Components with Roll-Rate Modulation

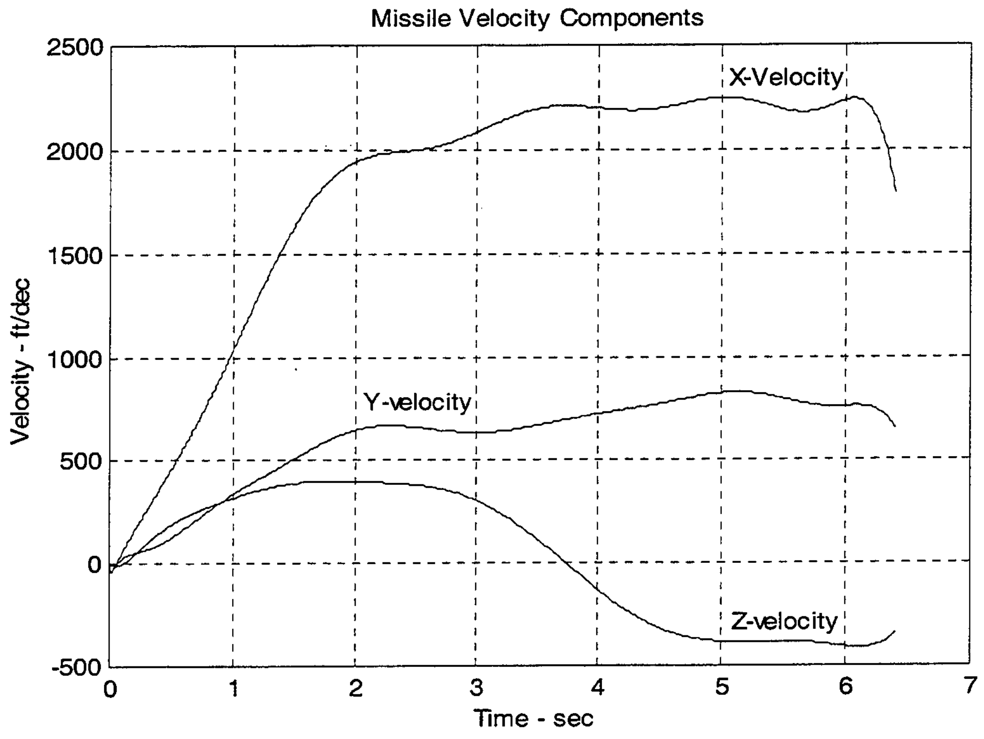

Figure 4.9. Missile Velocity Components with Modulation Removed

c. *TM Tracker Simulation Block Diagram*

The flow of calculations for simulation of the TM Tracker in MATLAB is shown in the block diagram of Figure 4.10. Subsequent sections discuss the calculations depicted in this figure.

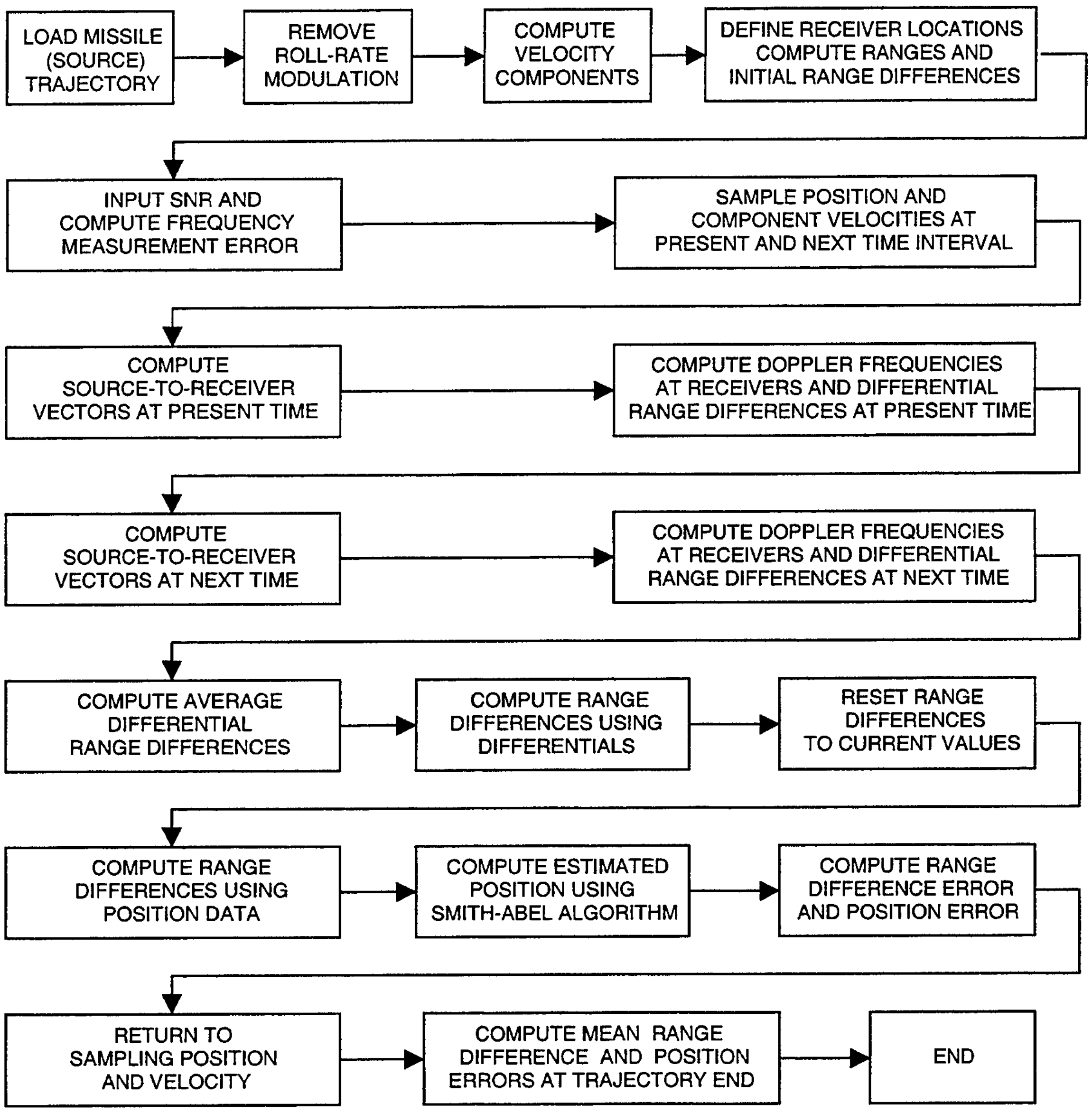
LOAD MISSILE (SOURCE) TRAJECTORY
REMOVE ROLL-RATE MODULATION
COMPUTE VELOCITY COMPONENTS
DEFINE RECEIVER LOCATIONS COMPUTE RANGES AND INITIAL RANGE DIFFERENCES
INPUT SNR AND COMPUTE FREQUENCY MEASUREMENT ERROR
SAMPLE POSITION AND COMPONENT VELOCITIES AT PRESENT AND NEXT TIME INTERVAL
COMPUTE SOURCE-TO-RECEIVER VECTORS AT PRESENT TIME
COMPUTE DOPPLER FREQUENCIES AT RECEIVERS AND DIFFERENTIAL RANGE DIFFERENCES AT PRESENT TIME
COMPUTE SOURCE-TO-RECEIVER VECTORS AT NEXT TIME
COMPUTE DOPPLER FREQUENCIES AT RECEIVERS AND DIFFERENTIAL RANGE DIFFERENCES AT NEXT TIME
COMPUTE AVERAGE DIFFERENTIAL RANGE DIFFERENCES
COMPUTE RANGE DIFFERENCES USING DIFFERENTIALS
RESET RANGE DIFFERENCES TO CURRENT VALUES
COMPUTE RANGE DIFFERENCES USING POSITION DATA
COMPUTE ESTIMATED POSITION USING SMITH-ABEL ALGORITHM
COMPUTE RANGE DIFFERENCE ERROR AND POSITION ERROR
RETURN TO SAMPLING POSITION AND VELOCITY
COMPUTE MEAN RANGE DIFFERENCE AND POSITION ERRORS AT TRAJECTORY END
END

Figure 4.10. Block Diagram of TM Tracker Simulation

d. Ranges and Initial Range Differences

The ranges from each receiver to the reference are computed before missile flight begins because these values remain constant and are one of the inputs to the Smith-Abel algorithm. Range differences are found using an incremental integrating approach. The initial range differences are calculated prior to missile launch to establish a base to which average differential range differences are added at each time interval.

e. Doppler Frequency / Differential Range Difference Calculation

A phase-detector approach for frequency measurement was used for all TM Tracker simulations along with the theoretical lower bound on frequency measurement error due to noise (see Equation 4.1) because of the inherent limitations of the zero-crossing counter method for determining frequency. This method, implemented within the TM Tracker, requires that the receivers must be in full-view of the missile throughout the entire trajectory because the zero-crossing count begins at time zero. This also introduces a uniform phase error over ± 360 degrees, equivalent to 0.130 m error in range. Even with no phase measurement noise within the system, position error due to the zero-crossing counter method alone was on the order of 10 feet. This error only became worse when phase measurement noise and receiver noise were added to the system. [Ref. 7]

The Doppler frequency vector $\vec{f}_d$ detected by a stationary receiver from a source with a wavelength λ and a velocity vector $\vec{v}$ is defined as

$$\vec{f}_d = \frac{\vec{v}}{\lambda} \qquad (4.2)$$

To compute the Doppler frequencies at each receiver, $f_{d\,rec}$, the scalar component of $\vec{f}_d$ in the direction of each receiver is found by using the dot product

$$f_{d\,rec} = \vec{f} \cdot \frac{\vec{B}}{|\vec{B}|} \tag{4.3}$$

where $\vec{B}$ is the vector from the source to the receiver and $|\vec{B}|$ is the vector magnitude. Each scalar Doppler frequency is converted to a differential range difference (RD). The RD between the reference and *i*th receivers is defined by

$$RD_i = -\lambda \cdot T \cdot (f_{d1} - f_{di}) \tag{4.4}$$

where λ is the source wavelength, T is the sample time (0.01 sec), and f_{d1} and f_{di} are the respective Doppler frequencies at the reference receiver and *i*th receiver. The negative sign ensures a positive RD value; $(f_{d1} - f_{di})$ will always be negative because the missile is moving away from the reference receiver through its entire trajectory, producing lower relative Doppler frequencies at the reference receiver.

Two differential RDs are then computed for each receiver used. The first RD is based on missile position and velocity at the present sample time. The second RD is based on missile position and velocity at the next sample time (0.01 sec later). These sets of RDs are used to determine average differential RDs which are added to the base RDs determined previously. The base RDs are then reset to these values and the process is repeated through the missile's trajectory. Geometric RDs are also calculated to determine the error in RD computation using missile position data for range calculations.

f. Smith-Abel Algorithm Position Estimation

The missile's estimated position at the next time interval is determined using the Smith-Abel algorithm (*smith_abel.m*). Inputs to this algorithm include receiver location coordinates, RDs, ranges from receivers to the reference receiver, and weight values. For all simulations, weights were only assigned values of one or zero for receiver RDs that were used or not used during the missile's trajectory.

g. Range Difference and Position Error Calculation

The range difference error is determined by computing the magnitude of the vector from the geometric RD to the RD obtained using incremental RDs from Doppler frequencies. Position error is defined as the magnitude of the vector from the Smith-Abel position estimation to the missile position coordinates at the next time interval. These errors are computed for each sample point along the missile's trajectory and stored. Mean values for these errors are then calculated at the end of missile flight.

3. Algorithm Results Using All Receivers

Simulations of the TM Tracker were first conducted using all nine receivers available on the missile range (*trajall.m*). Ideal conditions with the absence of noise were implemented in these simulations. Because there was no noise present within the system, any value of SNR used within the program yielded the same results. Figures 4.11 and 4.12 show the range difference and position errors that occurred during the missile's trajectory.

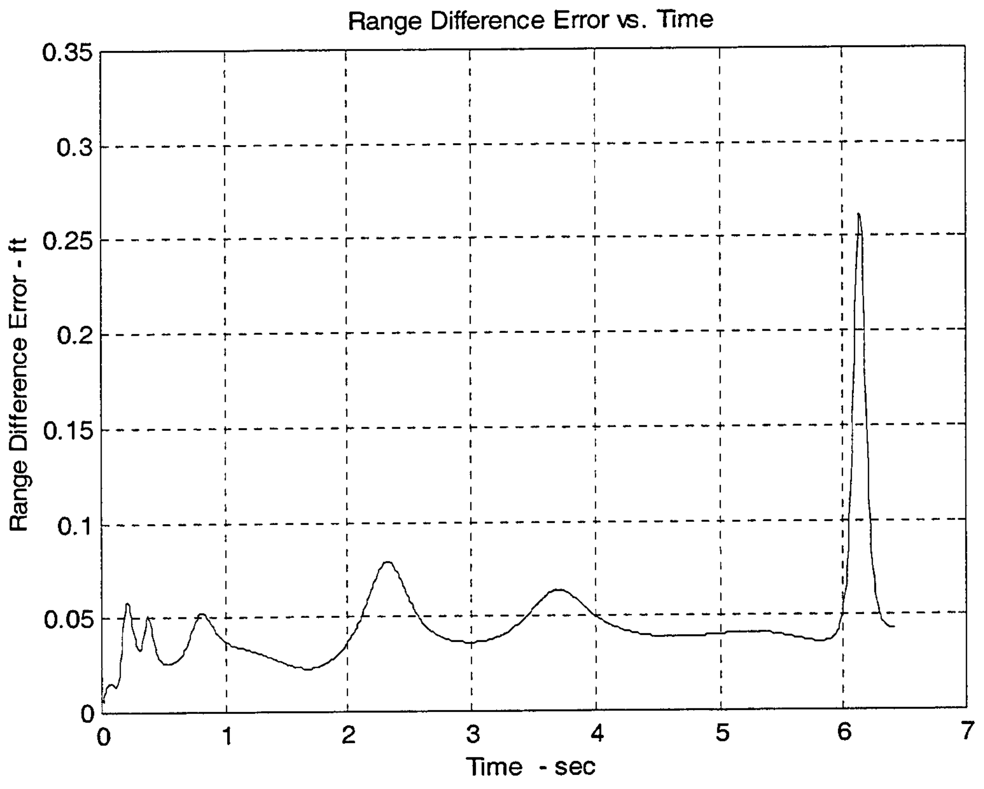

Figure 4.11. Range Difference Error vs. Time – Ideal Conditions

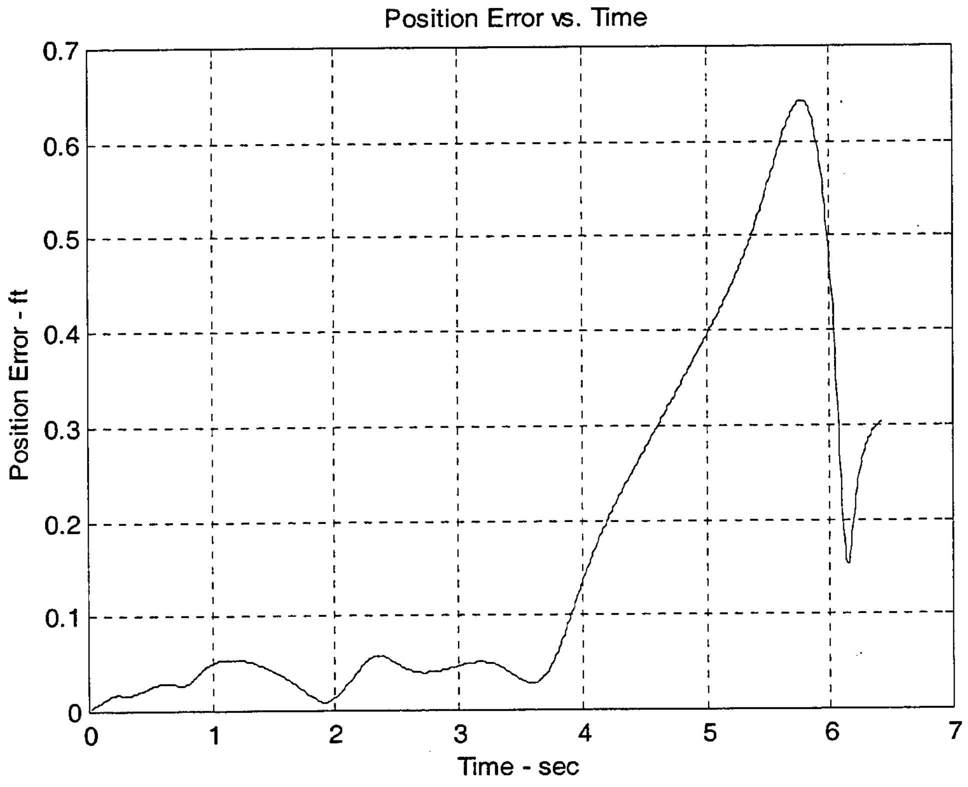

Figure 4.12. Position Error vs. Time – Ideal Conditions

Mean range difference and position errors were quite low, 0.046 ft and 0.168 ft, respectively. These errors represent the best obtainable accuracy of the TM Tracker under perfect conditions. Real-world conditions with random white Gaussian noise present within the Doppler frequency measurements were simulated to determine effects on range difference and position errors (*trajalln.m*). The resulting errors with receiver SNR of 35 dB are shown in Figures 4.13 and 4.14.

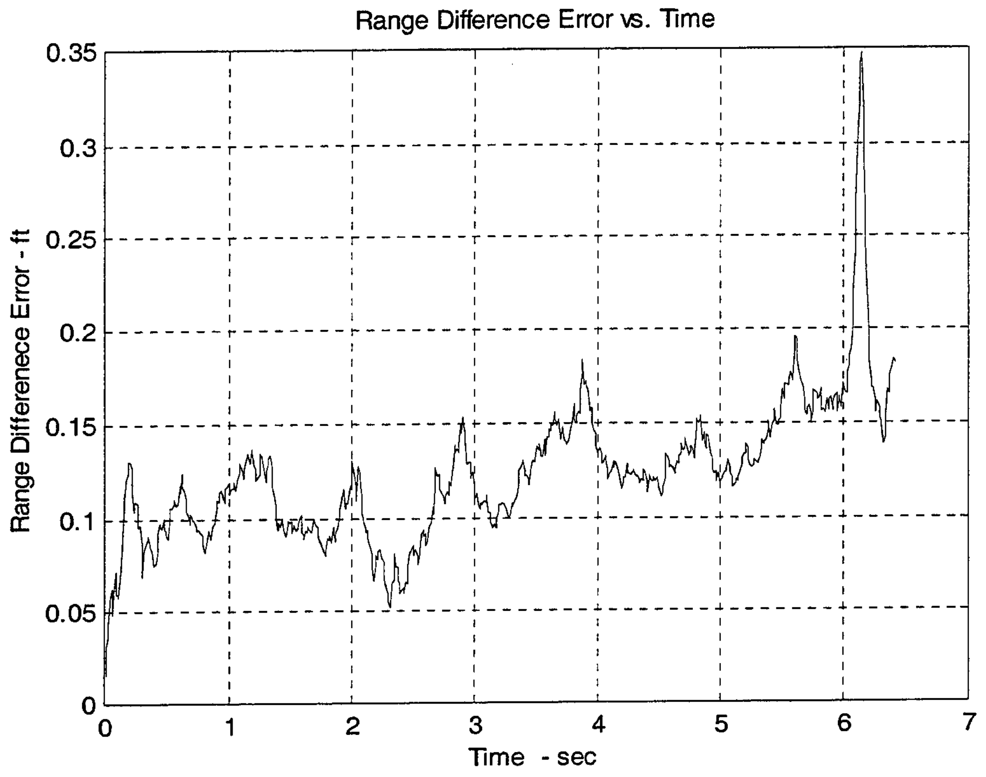

Figure 4.13. Range Difference Error vs. Time – 35 dB SNR

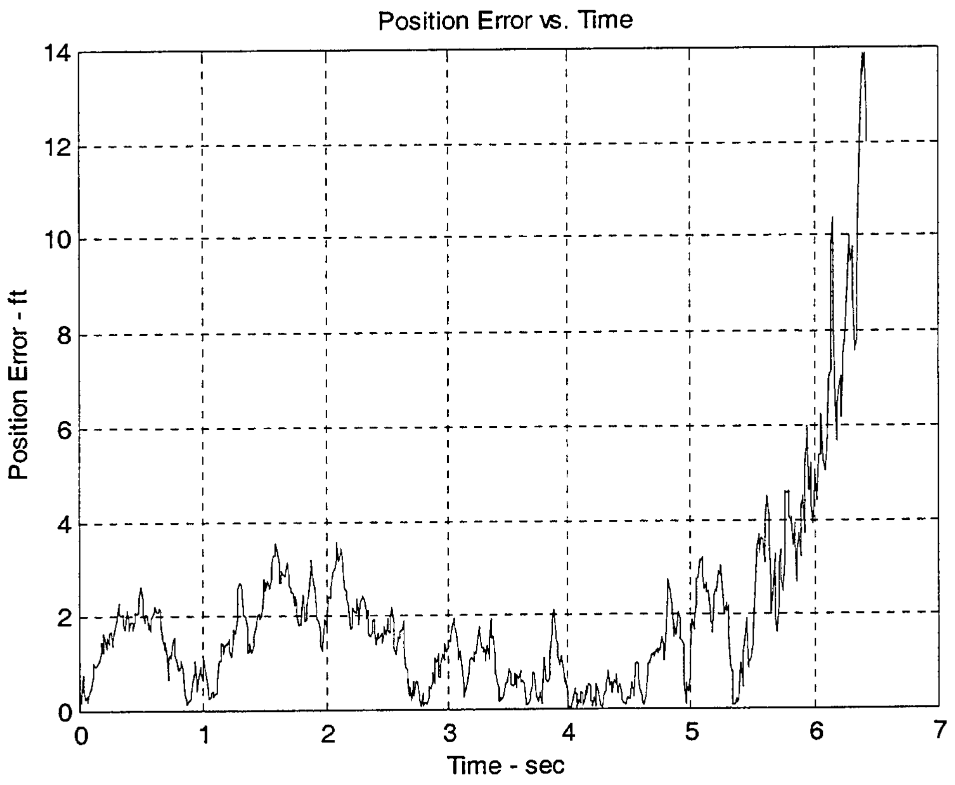

Figure 4.14. Position Error vs. Time – 35 dB SNR

The values were a 0.123-ft mean error in range difference and a 1.975-ft mean error in position. Because of the randomness of the noise, these errors would vary with each subsequent run of the program using the same SNR value. Monte Carlo simulation with 600 iterations for each value of SNR from 15 – 40 dB was conducted to determine the range difference and position errors over a wider sample of data (*trajallnmc.m*). The results are shown in Figures 4.15 and 4.16.

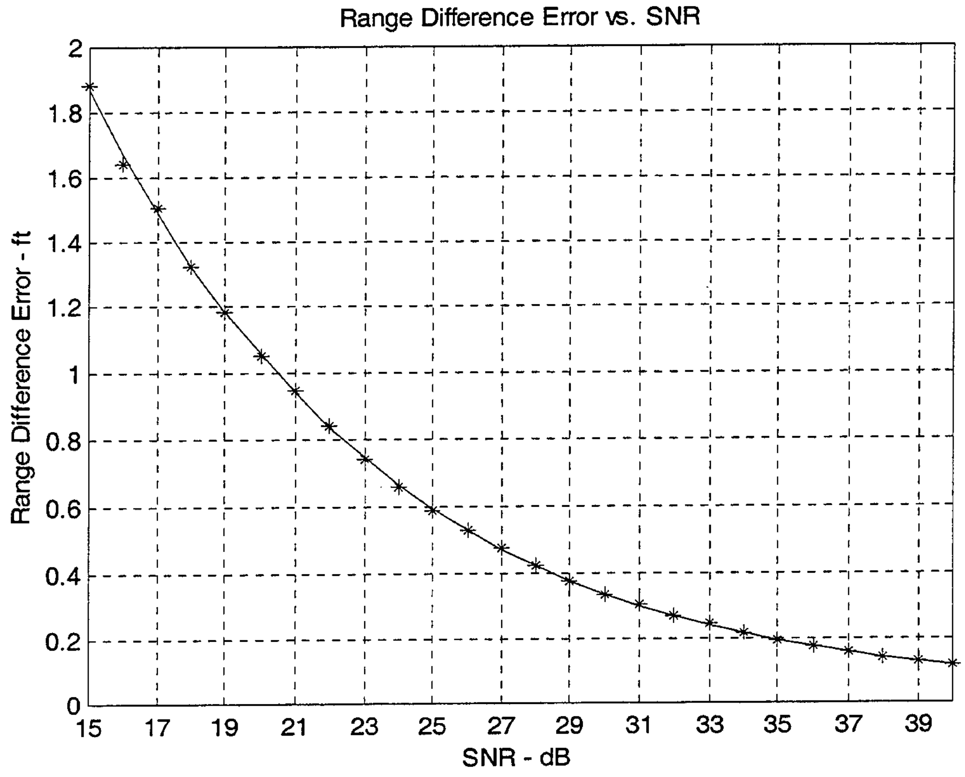

Figure 4.15. Range Difference Error vs. SNR

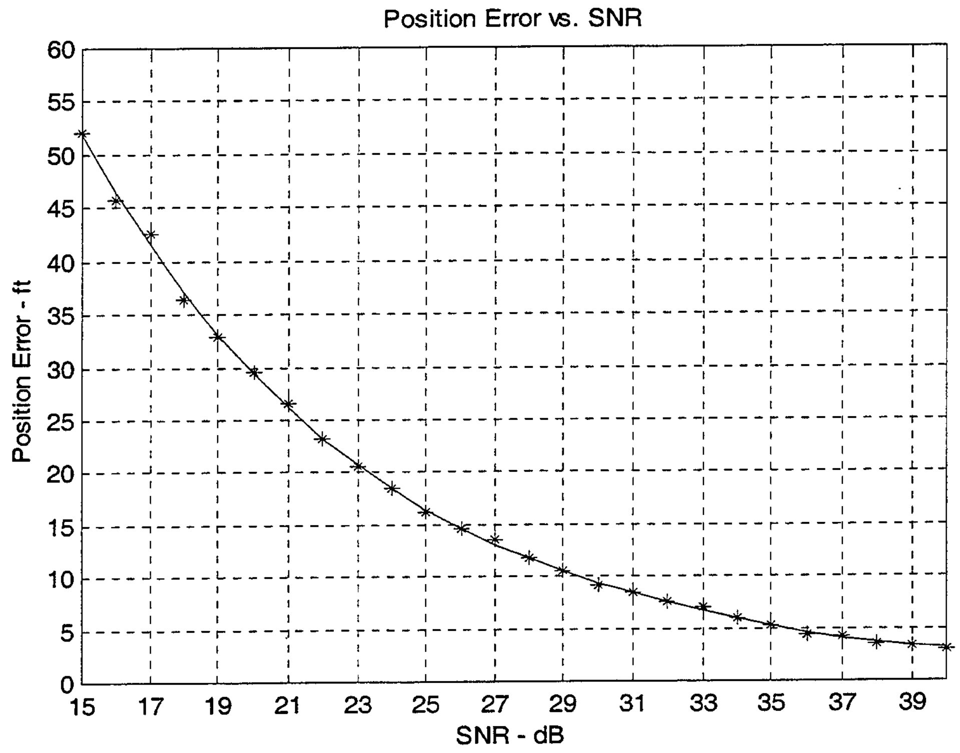

Figure 4.16. Position Error vs. SNR

The results from these figures indicate that the remote receivers require an SNR value of approximately 39 – 40 dB to obtain TSPI accuracy within 1 meter (about 3.28 ft). These simulations, however, assume that all receivers can detect the missile's telemetry signal throughout its entire trajectory. Because of the receivers' locations on the range and their antenna beam pattern limitations, this is not possible.

4. Algorithm Results Using Available Receivers

a. Antenna Beam Width Considerations

All receivers have a 3-dB beam width of approximately 45° (see Figure 2.6). To determine each receiver's ability to obtain the telemetry signal, the missile's trajectory had to be converted to spherical coordinates and referenced to each receiver in terms of azimuth and elevation. Range was not used because the trajectory did not pose any range limitations on the receivers. MATLAB was used to calculate values for azimuth (theta) and elevation (phi) for each receiver during the missile's trajectory (*angles.m*). Because MATLAB's conversion from Cartesian to spherical coordinates is referenced to the origin, the coordinate system was translated such that the origin was centered on each receiver, resulting in also translating the missile's trajectory so that the relative geometry between the receiver and the missile was maintained.

b. Receiver Time Utilization

Plots of azimuth and elevation versus time were then used to determine what period of time the receivers could receive the telemetry signal in their respective 3-dB beam widths. Since the antennas are stationary, only one time period could be utilized for each receiver. This time period is limited by a 45° change in either azimuth or elevation, whichever occurs first. Figure 4.17 shows a plot of these angles vs. trajectory time for Receiver 14 and how its time period was determined.

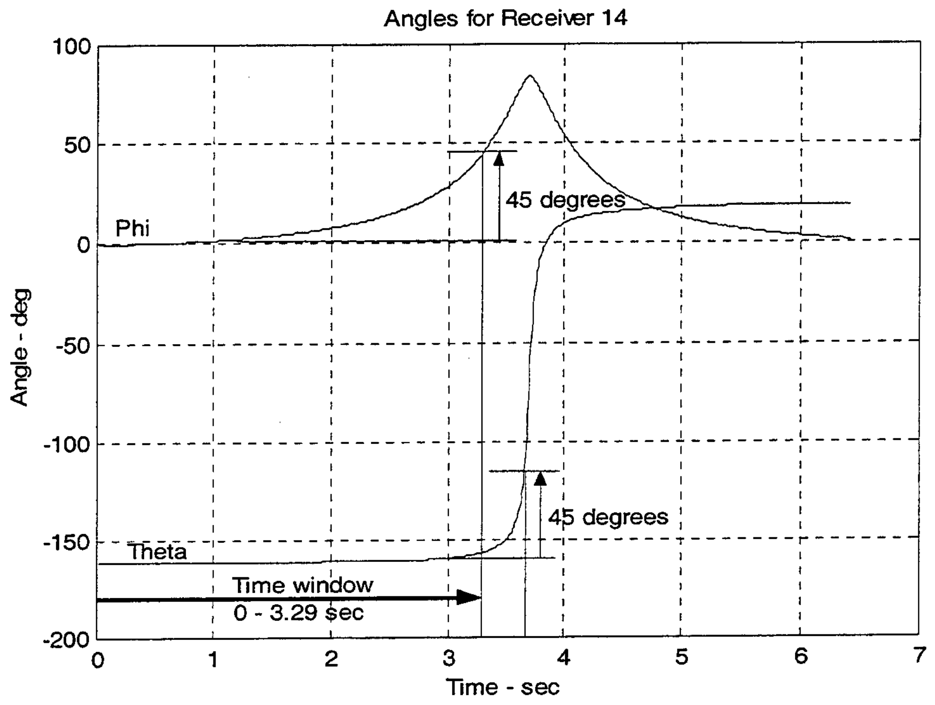

Figure 4.17. Azimuth and Elevation Angles for Receiver 14

In the case of Receiver 14, the time period was chosen at the start of the missile's trajectory and the elevation angle phi was the first to reach its 45° deviation limit at about 3.29 sec. Four receivers, 1a, 1b, 1p, and 10, were able to retain the telemetry signal through the entire trajectory because azimuth and elevation angles never changed by more than 45° during missile flight. Many different scenarios were conducted for the other five receivers to determine the best way to implement their respective time periods within the algorithm to achieve the lowest range difference and position errors. Plots of azimuth and elevation angles vs. time for each receiver are listed in Appendix B. The time period utilization of receivers for subsequent simulations is shown in Table 4.1.

Time	Receivers Used or Lost
Start – 2.11 sec	Use Receivers 1a, 1b, 1p, 10, 11, 12, 14, 15
2.12 sec	Lose Receiver 12
2.12 sec – 3.28 sec	Use Receivers 1a, 1b, 1p, 10, 11, 14, 15
3.29 sec	Lose Receiver 14
3.29 sec – 4.48 sec	Use Receivers 1a, 1b, 1p, 10, 11, 15
4.49 sec	Add Receiver 16, Lose Receiver 15
4.49 sec – 5.94 sec	Use Receivers 1a, 1b, 1p, 10, 11, 16
5.95 sec	Lose Receiver 11
5.95 sec – End	Use Receivers 1a, 1b, 1p, 10, 16

Table 4.1. Time Utilization of Receivers During Missile Trajectory

c. Initial Range Differences for Added Receivers

The weights within the Smith-Abel algorithm allow for changing the range differences used to estimate missile position during its trajectory. Values of one are applied to range differences in use, values of zero are applied to range differences not used for position estimation. Removing receivers during the trajectory poses no problems in the simulation as long as a minimum of five receivers (four range differences) is maintained. Adding receivers after time start, however, requires special consideration in terms of computing the initial base range differences to which the incremental range differences are added in subsequent time intervals.

When a receiver is added after missile flight has begun, an initial base range difference must be calculated at that instant based on the missile's *estimated* position determined from the Smith-Abel algorithm. This is due to the fact that an initial range difference was not computed prior to missile launch for this receiver because it could not receive the telemetry signal at that time. Incremental range differences are then added to this base as in the previous simulations. Because this base is derived from an estimated position that contains some error, position errors after this added receiver will become progressively worse, therefore, it is best to add receivers as late in the trajectory as feasible to prevent the rapid accumulation of error during missile flight.

d. Simulation Results

Receivers were utilized in MATLAB as shown in Table 4.1 in ideal conditions with the absence of noise (*trajgroup.m*). Figures 4.18 and 4.19 show the range difference and position errors that occurred during missile flight.

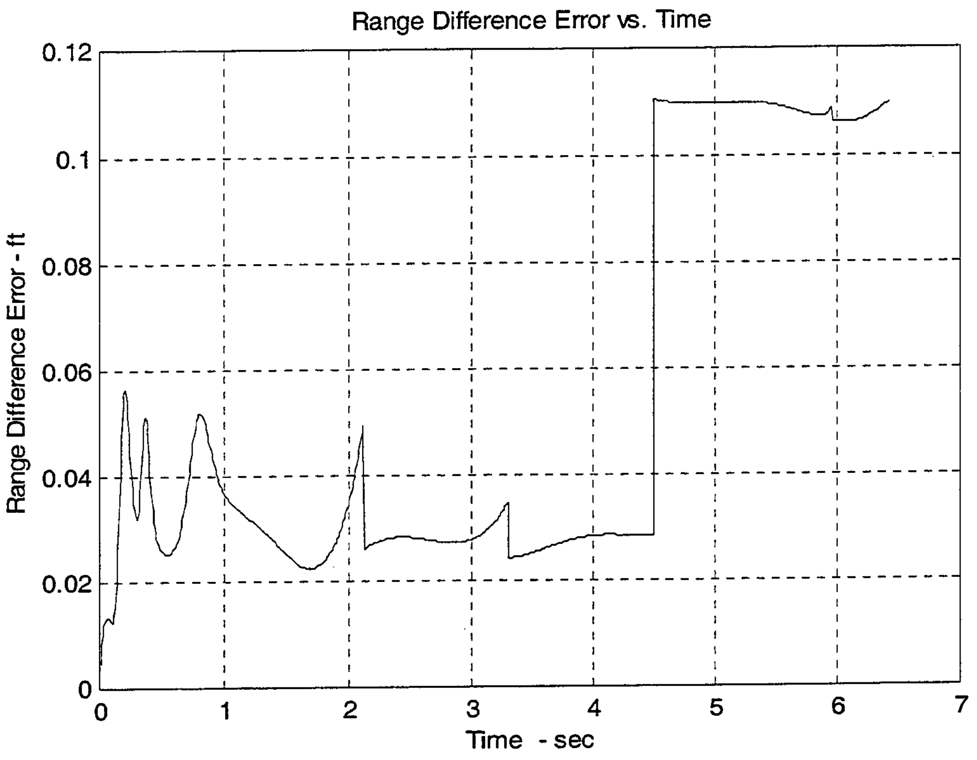

Figure 4.18. Range Difference Error vs. Time – Ideal Conditions

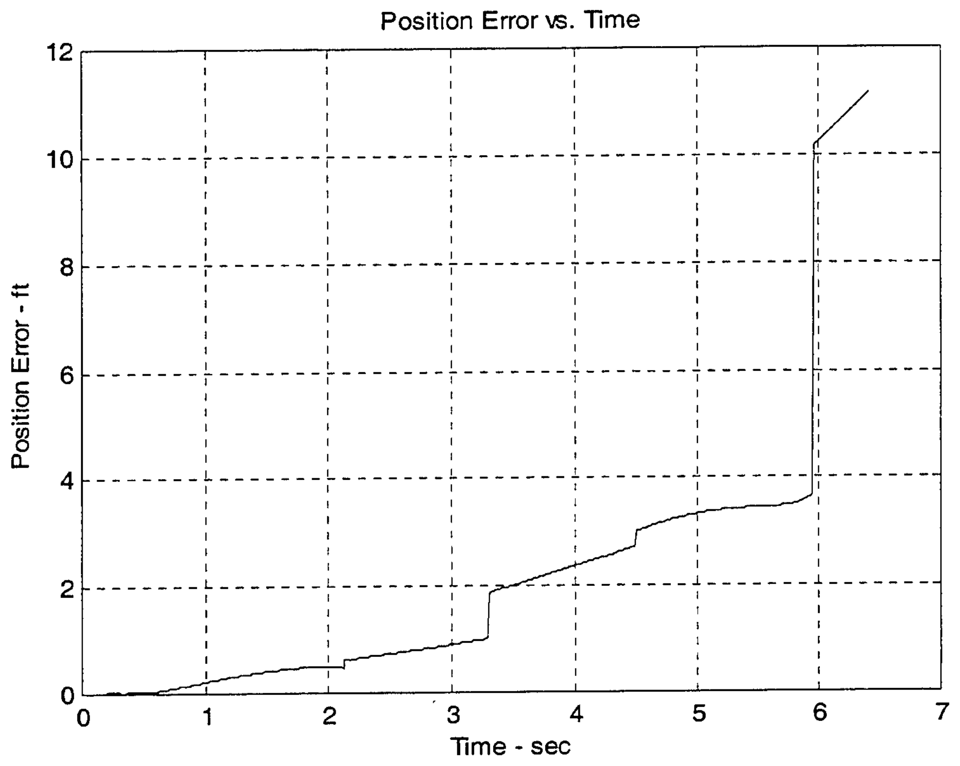

Figure 4.19. Position Error vs. Time – Ideal Conditions

Mean values for range difference and position error were 0.054 ft and 2.200 ft, respectively. A jump in range difference error can be noticed in Figure 4.18 at 4.49 sec when Receiver 16 was added due to the base range difference being calculated from the missile's estimated position at that point. Range difference error then remains stable from that point until the end of the trajectory. Figure 4.19 shows slight increases in positional error as receivers were lost during the trajectory, but a drastic increase occurred when Receiver 11 dropped out at 5.95 sec. Large increases in position error also occurred at the end of trajectory during testing of other receiver utilization methods. White Gaussian noise was added to Doppler frequency measurements in this simulation to represent real-world conditions (*trajgroupn.m*). Figures 4.20 and 4.21 show resulting range difference and position errors with receiver SNR of 35 dB.

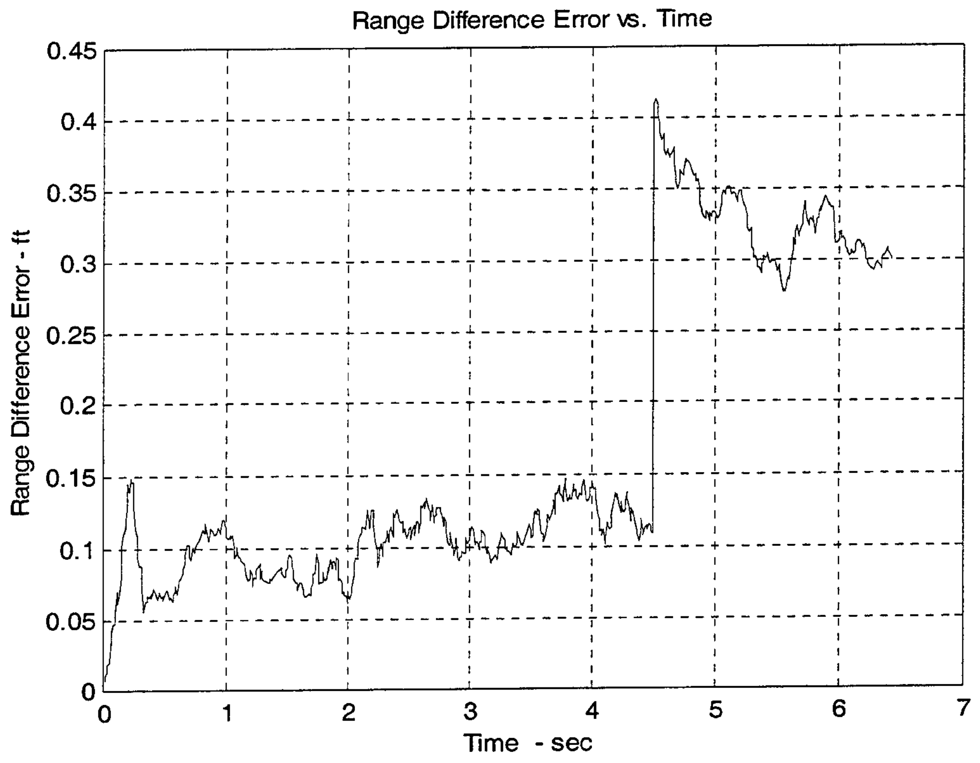

Figure 4.20. Range Difference Error vs. Time – 35 dB SNR

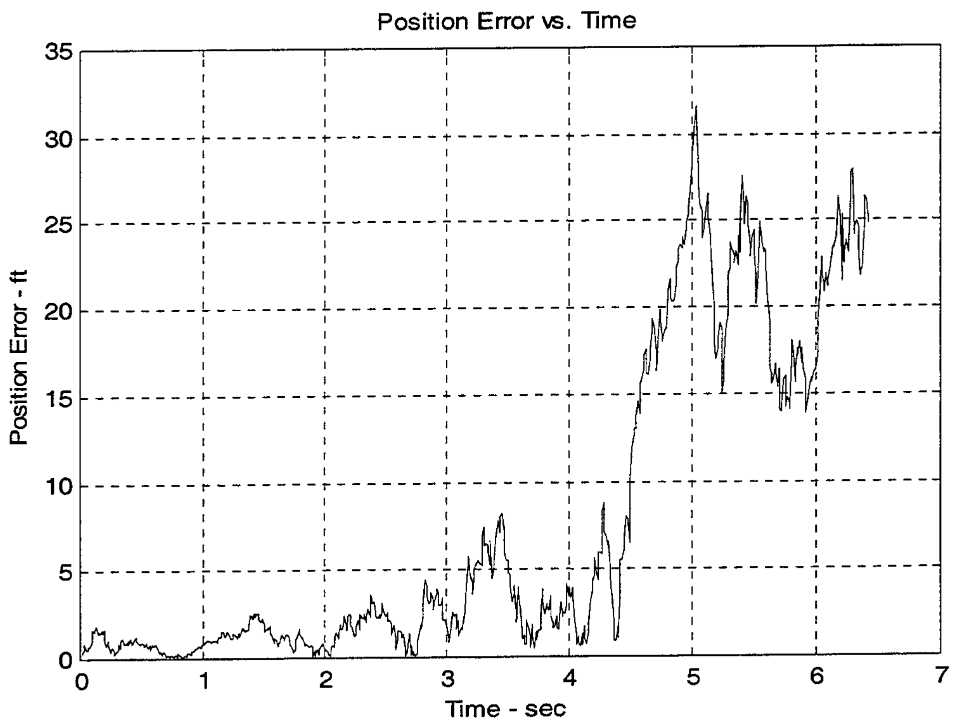

Figure 4.21. Position Error vs. Time – 35 dB SNR

Added noise caused a slight increase in mean range difference error, 0.169 ft, and a larger increase in mean position error, 7.681 ft. Even with the randomness of the noise, large increases in position error tend to occur after approximately 4.5 sec. Monte Carlo simulation was conducted with 600 iterations for each value of SNR from 15 – 40 dB to better estimate the effects using a larger sample of data (*trajgroupnmc.m*). Figures 4.22 and 4.23 depict the errors that occurred.

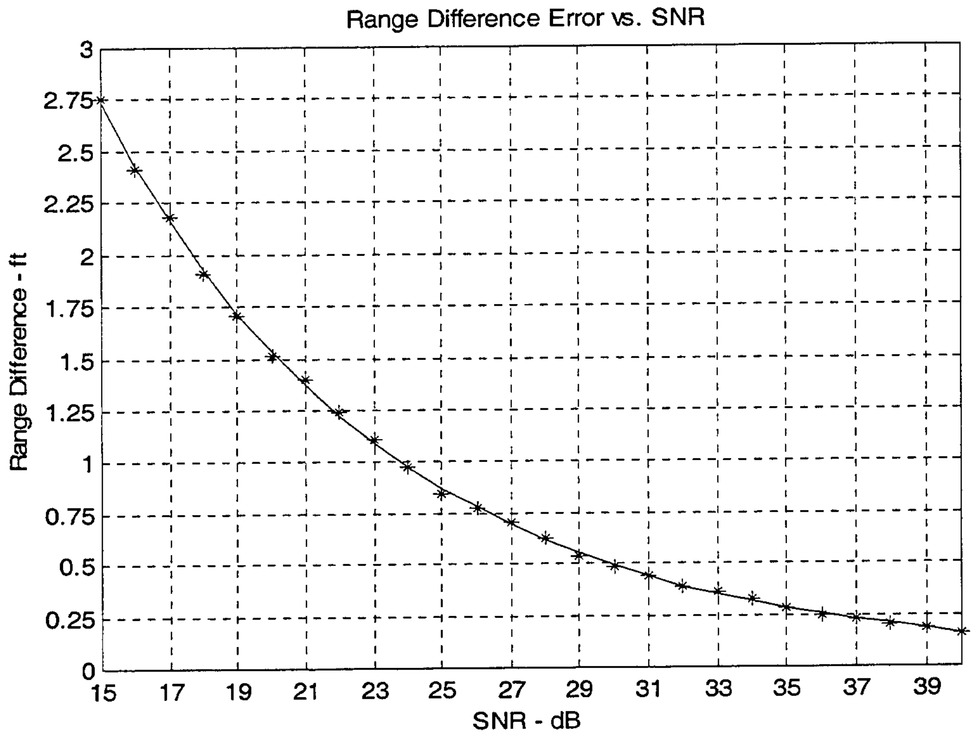

Figure 4.22. Range Difference Error vs. SNR

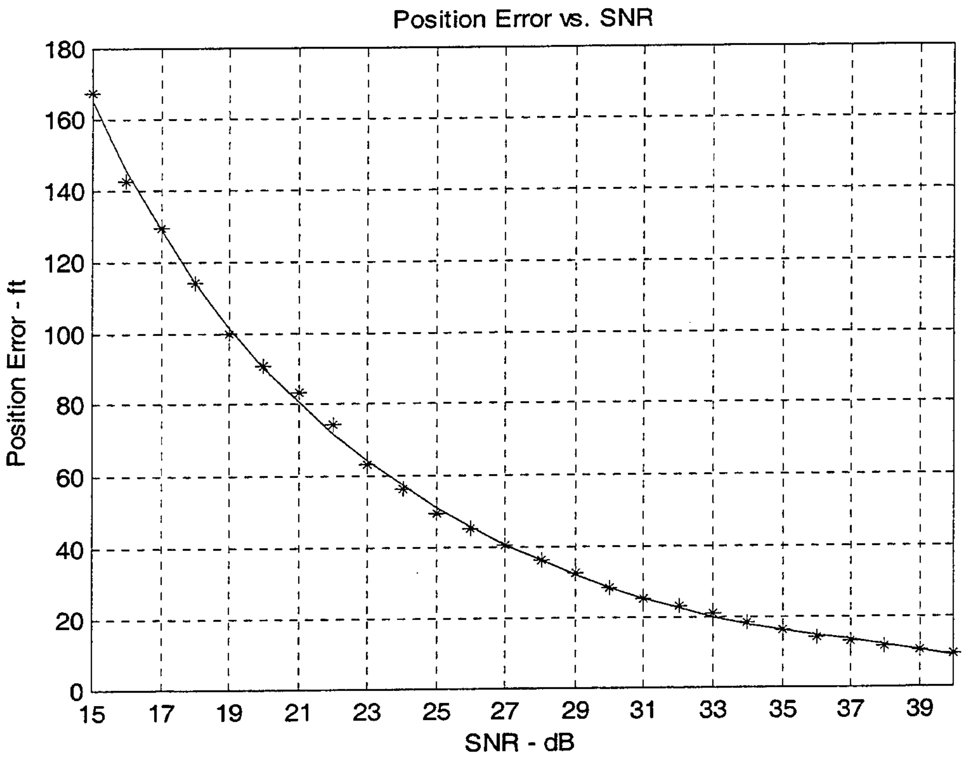

Figure 4.23. Position Error vs. SNR

Figures 4.22 and 4.23 show that receiver SNR values need to be above 40 dB to obtain 1-meter TSPI accuracy for the method used in Table 4.1. It is important to note that an enormous amount of methods to utilize the receivers during the missile's trajectory exists. The receiver time windows used in these simulations were chosen after many different combinations of time windows were tested. During the testing of these combinations, position errors increased when receivers were added near the beginning of the missile's trajectory due to the rapid accumulation of error that occurs from estimating the initial range difference of the added receiver. If this accumulation is allowed to start early in the missile's trajectory, a larger error will result by the time the end of the trajectory is reached. Adding receivers later in the trajectory minimizes this error, but at the same time, a minimum of five receivers must always be maintained which could limit the use of this method. The best implementation with the least amount of error is when all nine receivers are utilized throughout the entire trajectory as shown in Figures 4.15 and 4.16. Errors in computing added receivers' initial range differences are eliminated because all receivers are used from time start until the end of the trajectory. Relocating receivers along with the geographic conditions on the range, however, may preclude the use of this option.

A table was developed to summarize the results of TM Tracker simulations. Table 4.2 lists the simulation description and resulting mean range differences and position errors that were produced.

TM Tracker Simulation Description	**MATLAB File Name**	**Mean Range Difference Error (ft)**	**Mean Position Error (ft)**
All receivers used with no noise	*trajall.m*	0.046	0.168
All receivers used with noise, 35 dB SNR	*trajalln.m*	0.123	1.975
All receivers used with noise, 15-40 dB SNR, Monte Carlo simulation (600 iterations)	*trajallnmc.m*	SNR dependent See Figure 4.15	SNR dependent See Figure 4.16
Available receivers used with no noise	*trajgroup.m*	0.054	2.200
Available receivers used with noise, 35 dB SNR	*trajgroupn.m*	0.169	7.681
Available receivers used with noise, 15-40 dB SNR, Monte Carlo simulation (600 iterations)	*trajgroupnmc.m*	SNR dependent See Figure 4.23	SNR dependent See Figure 4.24

Table 4.2. Summary of TM Tracker Simulations

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSION

A. SUMMARY

The objective of this thesis was to determine system requirements and limitations of the TM Tracker to obtain TSPI with 1-meter accuracy. By simulating the system under ideal conditions with no noise present, a theoretical lower limit of TSPI accuracy was established using all remote receivers on the missile range. Noise was then added to Doppler frequency measurements to simulate real-world conditions and determine effects on TSPI accuracy. Based on Monte Carlo simulations, if all nine receivers are used throughout the entire missile trajectory with SNRs above approximately 39 dB, the TM Tracker is capable of producing TSPI accuracy within 1 meter.

Simulations were then conducted with the implementation of the receivers' 45-degree, 3-dB beam widths. Only those receivers that could obtain the telemetry signal within their respective beam widths were utilized at any given time during the trajectory. Because the receiver antennas are stationary, only one window of time could be used for signal reception within each receiver's beam width. Special consideration had to be given to the computation of the initial range differences when a receiver was added after missile launch. Because its initial range difference is based on an estimated position, to which all subsequent differential range differences are added, position errors after the addition of a receiver during the trajectory became progressively worse.

The receiver addition was accomplished late in the missile's trajectory to prevent the rapid accumulation of errors. Monte Carlo simulations revealed that receivers required SNRs in the 45-dB range to obtain 1-meter TSPI accuracy. Other methods of receiver utilization that result in lower errors for the same values of SNR are quite possible, but an enormous number of combinations exists, and each Monte Carlo simulation takes about 22 hours to complete on a Pentium-III class computer.

B. FUTURE RESEARCH

A zero-crossing counter approach was used in the field-tested TM Tracker for frequency measurement, which was then converted to TDOA and a range difference. Sample measurements were made every 0.01 sec and consisted of the number of zero crossings from time start to the measurement point. Because the count started at time zero, each receiver had to have the missile within full view during its entire trajectory. MATLAB simulations of the TM Tracker assumed frequency measurements were limited by the theoretical lower bound due to noise and did not take into account the implications of using the zero-crossing counter approach in determining frequency, which quantizes range difference measurements to the order of a wavelength, (0.130 m). [Ref. 7] Because of the inherent requirements and limitations of the zero-crossing counter approach and its effects on TSPI accuracy, a phase detector system was utilized in all simulations. The implementation of a phase detector system for improved frequency measurement and a continuous wave (CW) telemetry signal for band limiting the noise within the remote receivers is currently being investigated for use in future testing of the TM Tracker system.

APPENDIX A. MATLAB SOURCE CODES

The MATLAB files shown in Table A.1 were used for simulations in support of this thesis. Their respective source codes are listed within this appendix.

MATLAB File	Purpose
fdoafang4nmc.m	Monte Carlo simulation of Fang's algorithm for determining position error vs. SNR for simple trajectory
tdoafang4nmc.m	Monte Carlo simulation of Fang's algorithm for determining X, Y, Z, and total position error vs. RMS measurement error
smith_abel.m	Smith – Abel algorithm for estimating source location
trajall.m	TM Tracker simulation using all receivers with no noise
trajalln.m	TM Tracker simulation using all receivers with noise
trajallnmc.m	Monte Carlo simulation of TM Tracker using all receivers with noise
angles.m	Azimuth and elevation angle calculation for all receivers
trajgroup.m	TM Tracker simulation using available receivers with no noise
trajgroupn.m	TM Tracker simulation using available receivers with noise
trajgroupnmc.m	Monte Carlo simulation of TM Tracker using available with noise

Table A.1. MATLAB Files

fdoafang4nmc.m

```
%Position from FDOA using Fang method
%Four receivers at (0,0,0),(X2,0,0),(X3,Y3,0),(X4,Y4,0)
%Inputs source position and calculates
%Position error with Monte Carlo simulation
%fdoafang4nmc.m
clear;clc;clf;

%Receiver locations
X2=-10;                 %km
X3=-10;                 %km
X4=-5;                  %km
Y3=5;                   %km
Y4=-5;                  %km

%Input source location
xp=input('x position=');
yp=input('y position=');
zp=input('z position=');

%Telemetry and missile parameters
fo=3e9;             %Hz
v=341.6;            %m/s
wl=3e8/fo;          %m
fdm=v/wl;           %Hz
T=1e-3;             %s

%Generate SNR loop (15-40 dB)
for snrdb=15:40;
   snr=10^(snrdb/10);

%Theoretical minimum frequency RMS measurement error
   dfrms=sqrt(3/snr)/(pi*T);

%Generate Monte Carlo loop (10 iterations)
   for mc=1:10;

%Compute initial ranges from receivers to source
      R1=sqrt(xp^2+yp^2+zp^2);
      R2=sqrt((xp-X2)^2+yp^2+zp^2);
      R3=sqrt((xp-X3)^2+(yp-Y3)^2+zp^2);
      R4=sqrt((xp-X4)^2+(yp-Y4)^2+zp^2);

%Compute initial RDs
      R21i=R2-R1;R21=R21i;
      R31i=R3-R1;R31=R31i;
      R41i=R4-R1;R41=R41i;

%Define initial positions
      x=xp;xx=x;
      y=yp;yy=y;
      z=zp;zz=z;

%Define time base for trajectory
      for t=0:T:10;
```

```
        if t==0;tx=0;
        else tx=[tx t];end;

%Find present missile position
        xp1=xp+(v*t/1000);
        yp1=yp;
        zp1=zp;
        if t==0;xpx=xp1;ypx=yp1;zpx=zp1;
        else xpx=[xpx xp1];ypx=[ypx yp1];zpx=[zpx zp1];end;

%Find next missile position
        xm=xp1+v*T/1000;
        ym=yp1;
        zm=zp1;

%Find Doppler shifts at receivers
%Present missile position
        phi0=atan2(zp1,sqrt(xp1^2+yp1^2));
        beta=atan2(yp1,xp1);
        phi1=atan2(zp1,sqrt((xp1-X2)^2+yp1^2));
        alpha=atan2(yp1,(xp1-X2));
        phi2=atan2((yp1-Y3),(xp1-X3));
        gamma=atan2((yp1-Y4),(xp1-X4));
        phi4=atan2(zp1,sqrt((xp1-X3)^2+(yp1-Y3)^2));
        phi5=atan2(zp1,sqrt((xp1-X4)^2+(yp1-Y4)^2));

%Find Doppler frequencies at present time
        fd1=fdm*cos(phi0)*cos(beta)+dfrms*randn;
        fd2=fdm*cos(phi1)*cos(alpha)+dfrms*randn;
        fd3=fdm*cos(phi2)*cos(phi4)+dfrms*randn;
        fd4=fdm*cos(gamma)*cos(phi5)+dfrms*randn;

%Find differential RDOAs at present time
        drdoa3a=-1e-4*T*(fd1-fd3);
        drdoa2a=-1e-4*T*(fd1-fd2);
        drdoa4a=-1e-4*T*(fd1-fd4);

%Find Doppler shifts at receivers
%Next missile position
        phi0=atan2(zm,sqrt(xm^2+ym^2));
        beta=atan2(ym,xm);
        phi1=atan2(zm,sqrt((xm-X2)^2+ym^2));
        alpha=atan2(ym,(xm-X2));
        phi2=atan2((ym-Y3),(xm-X3));
        gamma=atan2((ym-Y4),(xm-X4));
        phi4=atan2(zm,sqrt((xm-X3)^2+(ym-Y3)^2));
        phi5=atan2(zm,sqrt((xm-X4)^2+(ym-Y4)^2));

%Find Doppler frequencies at next time
        fd1=fdm*cos(phi0)*cos(beta)+dfrms*randn;
        fd2=fdm*cos(phi1)*cos(alpha)+dfrms*randn;
        fd3=fdm*cos(phi2)*cos(phi4)+dfrms*randn;
        fd4=fdm*cos(gamma)*cos(phi5)+dfrms*randn;

%Find differential RDOAs at next time
        drdoa3b=-1e-4*T*(fd1-fd3);
        drdoa2b=-1e-4*T*(fd1-fd2);
```

```
         drdoa4b=-1e-4*T*(fd1-fd4);

%Find average differential RDOAs
         drdoa2=(drdoa2a+drdoa2b)/2;
         drdoa3=(drdoa3a+drdoa3b)/2;
         drdoa4=(drdoa4a+drdoa4b)/2;

%Find RDOAs
         R21a=R21+drdoa2;
         R31a=R31+drdoa3;
         R41a=R41+drdoa4;

%Find coefficients
         g1=(R31a*X2/R21a-X3)/Y3;
         g2=(R41a*X2/R21a-X4)/Y4;
         g=g1-g2;
         h1=(X3^2+Y3^2-R31a^2+R31a*R21a*(1-(X2/R21a)^2))/(2*Y3);
         h2=(X4^2+Y4^2-R41a^2+R41a*R21a*(1-(X2/R21a)^2))/(2*Y4);
         h=h1-h2;
         d1=-((1-(X2/R21a)^2)+g1*g1);
         d2=-((1-(X2/R21a)^2)+g2*g2);
         d=d1-d2;
         e1=X2*((1-(X2/R21a)^2))-2*g1*h1;
         e2=X2*((1-(X2/R21a)^2))-2*g2*h2;
         e=e1-e2;
         f1=((R21a^2)/4)*(1-(X2/R21a)^2)^2-h1*h1;
         f2=((R21a^2)/4)*(1-(X2/R21a)^2)^2-h2*h2;
         f=f1-f2;

%Reset range differences
         R21=R21a;
         R31=R31a;
         R41=R41a;

%Find new positions
         x=(-e+sqrt(e*e-4*d*f))/(2*d);
         y=g1*x+h1;
         za=sqrt(d1*x*x+e1*x+f1);
         z=abs(za);

%Find position error in meters
         err=1000*sqrt((x-xm)^2+(y-ym)^2+(z-zm)^2);

%Fill error matrix for missile trajectory
         if t==0;xe=err;
         else xe=[xe err];end

%Return to trajectory loop
      end;

%Calculate mean error for missile trajectory
      merr=mean(xe);

%Fill mean error matrix for Monte Carlo simulation
      if mc==1;mcxe=merr;
      else mcxe=[mcxe merr];end
```

```
%Return to Monte Carlo loop
   end;

%Calculate mean error for Monte Carlo iterations
   mcmerr=mean(mcxe);

%Fill mean error matrix for each value of SNR
   if snrdb==15;snmcmerr=mcmerr;snrdbm=snrdb;
   else snmcmerr=[snmcmerr mcmerr];snrdbm=[snrdbm snrdb];end

%Output results
   fprintf(' SNR = %g\n' , snrdb);
   fprintf(' x = %g \n' , x);
   fprintf(' y = %g \n' , y);
   fprintf(' z = %g \n' , z);
   fprintf(' Mean error for 10 iterations = %g \n' , mcmerr);

%Return to SNR loop
end;

%Plot the graph
figure(1);
plot(snrdbm,snmcmerr);grid;
title('Position Error vs. SNR');
xlabel('SNR - dB');ylabel('Position Error - meters');
```

tdoafang4nmc.m

```
%Position from TDOA using Fang method
%Four receivers at (0,0,0),(X2,0,0),(X3,Y3,0),(X4,Y4,0)
%Inputs source position, adds TDOA measurement errors, and calculates
%Position error with Monte Carlo simulation
%tdoafang4nmc.m
clear;clc;clf;

%Receiver locations
X2=-10;                 %km
X3=-10;                 %km
X4=-5;                  %km
Y3=5;                   %km
Y4=-5;                  %km

%Input source location
xp=input('x position=');
yp=input('y position=');
zp=input('z position=');

%Compute initial ranges from receivers to source
R1=sqrt(xp^2+yp^2+zp^2);
R2=sqrt((xp-X2)^2+yp^2+zp^2);
R3=sqrt((xp-X3)^2+(yp-Y3)^2+zp^2);
R4=sqrt((xp-X4)^2+(yp-Y4)^2+zp^2);

%Generate TDOA error loop
for j=.000001:.0000005:.00001;
```

```
%Generate Monte Carlo loop (10,000 iterations)
   for i=1:10000;

%Compute TDOA
      R21=(R2-R1)*(1+j*randn);
      R31=(R3-R1)*(1+j*randn);
      R41=(R4-R1)*(1+j*randn);

%Find coefficients
      g1=(R31*X2/R21-X3)/Y3;
      g2=(R41*X2/R21-X4)/Y4;
      g=g1-g2;
      h1=(X3^2+Y3^2-R31^2+R31*R21*(1-(X2/R21)^2))/(2*Y3);
      h2=(X4^2+Y4^2-R41^2+R41*R21*(1-(X2/R21)^2))/(2*Y4);
      h=h1-h2;
      d1=-((1-(X2/R21)^2)+g1*g1);
      d2=-((1-(X2/R21)^2)+g2*g2);
      d=d1-d2;
      e1=X2*((1-(X2/R21)^2))-2*g1*h1;
      e2=X2*((1-(X2/R21)^2))-2*g2*h2;
      e=e1-e2;
      f1=((R21^2)/4)*(1-(X2/R21)^2)^2-h1*h1;
      f2=((R21^2)/4)*(1-(X2/R21)^2)^2-h2*h2;
      f=f1-f2;

%Find positions
      x(i)=(-e+sqrt(e*e-4*d*f))/(2*d);
      y(i)=g1*x(i)+h1;
      z(i)=sqrt(d1*x(i)*x(i)+e1*x(i)+f1);

%Find position errors in meters
      xerr(i) = 1000*abs(xp-x(i));
      yerr(i) = 1000*abs(yp-y(i));
      zerr(i) = 1000*abs(zp-z(i));
      err(i)=1000*sqrt((x(i)-xp)^2+(y(i)-yp)^2+(z(i)-zp)^2);

%Return to Monte Carlo loop
   end;

%Find measurement errors in inches
   R21p=(R2-R1)*j*1000*39.37;
   R31p=(R3-R1)*j*1000*39.37;
   R41p=(R4-R1)*j*1000*39.37;

%Find and calculate smallest RMS measurement error
   if R21p<R31p & R21p<R41p;
      RMS=sqrt((R21p)^2);
   elseif R31p<R21p & R31p<R41p;
      RMS=sqrt((R31p)^2);
   else RMS=sqrt((R41p)^2);
   end;

%Output results
   fprintf('Introduced TDOA Error (km) = %g\n' , j);
   fprintf('RMS TDOA Measurement Error - inches = %g \n' , RMS);
   fprintf('Mean x-error - meters = %g \n' , mean(xerr));
   fprintf('Mean y-error - meters = %g \n' , mean(yerr));
   fprintf('Mean z-error - meters = %g \n' , mean(zerr));
   fprintf('Mean total error - meters = %g \n' , mean(err));
   fprintf('\n');

%Fill matrices for plotting values
   if j==.000001;
      nn=RMS;xx=mean(xerr);yy=mean(yerr);zz=mean(zerr);ee=mean(err);
   else nn=[nn RMS];xx=[xx mean(xerr)];yy=[yy mean(yerr)];
      zz=[zz mean(zerr)];ee=[ee mean(err)];
   end;

%Return to TDOA error loop
end;

%Plot the graphs
figure(1);
plot(nn,xx);grid;
title('X-Position Error vs. RMS Measurement Error');
xlabel('RMS Measurement Error - inches');
ylabel('X-Position Error - meters');

figure(2);
plot(nn,yy);grid;
title('Y-Position Error vs. RMS Measurement Error');
xlabel('RMS Measurement Error - inches');
ylabel('Y-Position Error - meters');

figure(3);
plot(nn,zz);grid;
title('Z-Position Error vs. RMS Measurement Error');
xlabel('RMS Measurement Error - inches');
ylabel('Z-Position Error - meters');

figure(4);
plot(nn,ee);grid;
title('Total Position Error vs. RMS Measurement Error');
xlabel('RMS Measurement Error - inches');
ylabel('Total Position Error - meters');
```

smith_abel.m

```
%Smith-Abel algorithm used as a function
%Computes source location estimate
%Inputs are receiver locations, range differences,
%Weights, and receiver ranges to reference receiver
%Algorithm is tailored for nine receivers
%smith_abel.m
function [source_estimate]=smith_abel(RL,d,w,R)

%Source locations of receivers other than reference receiver
S=RL(2:9,:);

%Translate coordinate system so reference receiver is at origin
S(:,1)=S(:,1)-RL(1,1);    %x-dimension translated
S(:,2)=S(:,2)-RL(1,2);    %y-dimension translated
```

```
S(:,3)=S(:,3)-RL(1,3);     %z-dimension translated

%Define delta matrix
delta=R.^2-d.^2;

%Define 8x8 diagonal matrix
W=diag(w);

%Utilize spherical interpolation
%Define Ps and Pso matrices
Ps = S*inv(S'*W*S)*S'*W;
Pso=eye(8) - Ps;

%Compute minimizing value for Rs
Rshat=(d'*Pso*W*Pso*delta)/(2*d'*Pso*W*Pso*d);

%Compute source location estimate
estimate=.5*inv(S'*W*S)*S'*W*(delta-2*Rshat*d);

%Translate back to orignial coordinate system
source_estimate=estimate'+RL(1,:);
```

trajall.m

```
%Position from FDOA using Smith-Abel algorithm
%Inputs missile trajectory and receiver SNR
%Calculates RDOA from FDOA and position with SNR measurement error
%All receivers utilized throughout entire trajectory
%trajall.m

%Load White Sands missile trajectory
load traj;
t=traj(1,:);xm=traj(2,:);ym=traj(3,:);zm=traj(4,:);

%Smooth missile trajectory
xms=polyval(polyfit(t,xm,14),t);
yms=polyval(polyfit(t,ym,14),t);
zms=polyval(polyfit(t,zm,14),t);

%Obtain polynomial coefficients for trajectory
xpc=polyfit(t,xm,14);
ypc=polyfit(t,ym,14);
zpc=polyfit(t,zm,14);

%Define coefficients for velocity (first derivative of polynomials)
i=1:14;
xvc=(15-i).*xpc(i);
yvc=(15-i).*ypc(i);
zvc=(15-i).*zpc(i);

%Evaluate velocity polynomials to obtain velocity component vectors
vxs=polyval(xvc,t);
vys=polyval(yvc,t);
vzs=polyval(zvc,t);
```

```
%White Sands remote receiver positions (feet)
RL=[-9243.4 -3085.7 -127; -9193.6 -3072.3 -125; -9278.7 -3201.8
    -92.1; -8992.9 -2879.8 -109.3; 1757 430.4 40.4; -6458.5 -2112.3
    -39.3; -3704.2 -1214.7 -33.6; 607 -3517.8 208.3; -5305.4 2653
    -153.6];

%Initialize range difference vectors
d=zeros(8,1);
da=zeros(8,1);
di=zeros(8,1);
db=zeros(8,1);

%Range from non-reference receiver to reference receiver
R=zeros(8,1);
R(1) = norm(RL(1,:) - RL(2,:));
R(2) = norm(RL(1,:) - RL(3,:));
R(3) = norm(RL(1,:) - RL(4,:));
R(4) = norm(RL(1,:) - RL(5,:));
R(5) = norm(RL(1,:) - RL(6,:));
R(6) = norm(RL(1,:) - RL(7,:));
R(7) = norm(RL(1,:) - RL(8,:));
R(8) = norm(RL(1,:) - RL(9,:));

%Set all weights equal to one (using all receivers)
w=ones(8,1);

%Telemetry and missile data
fo=2.3e9;               % Hz
wl=(3e8/fo)/.3048;      % feet
T=1e-2;                 % sec
Tmax=6.41;              % sec

%Initial range differences, with Receiver 1 as the reference
sourcei=[xms(1) yms(1) zms(1)];

di(1) = norm(sourcei - RL(2,:)) - norm(sourcei - RL(1,:));
di(2) = norm(sourcei - RL(3,:)) - norm(sourcei - RL(1,:));
di(3) = norm(sourcei - RL(4,:)) - norm(sourcei - RL(1,:));
di(4) = norm(sourcei - RL(5,:)) - norm(sourcei - RL(1,:));
di(5) = norm(sourcei - RL(6,:)) - norm(sourcei - RL(1,:));
di(6) = norm(sourcei - RL(7,:)) - norm(sourcei - RL(1,:));
di(7) = norm(sourcei - RL(8,:)) - norm(sourcei - RL(1,:));
di(8) = norm(sourcei - RL(9,:)) - norm(sourcei - RL(1,:));

d=di;

%Input receiver SNR
snrdb=input('snr(dB)=');
snr=10^(snrdb/10);

%Theoretical minimum frequency RMS measurement error
dfrms=sqrt(3/snr)/(pi*T);

%Generate loop for sampling trajectory points
%Read positions and velocities at each sample point (every .01 sec)
for ts=0:T:Tmax;
   sample=ts/T+1;
   sample=round(sample);      %Sample must be an integer value
   x=xms(1,sample);y=yms(1,sample);z=zms(1,sample);
   x1=xms(1,sample+1);y1=yms(1,sample+1);z1=zms(1,sample+1);
   v=[vxs(1,sample) vys(1,sample) vzs(1,sample)];
   v1=[vxs(1,sample+1) vys(1,sample+1) vzs(1,sample+1)];

%Find source position at present time
   source=[x y z];

%Find source position at next time
   source1=[x1 y1 z1];

%Find source-to-receiver vectors at present time
   B1=source-RL(1,:);
   B2=source-RL(2,:);
   B3=source-RL(3,:);
   B4=source-RL(4,:);
   B5=source-RL(5,:);
   B6=source-RL(6,:);
   B7=source-RL(7,:);
   B8=source-RL(8,:);
   B9=source-RL(9,:);

%Find Doppler frequency vector at present time
   F=v/wl;

%Find Doppler frequencies at receivers at present time
   fd1=dot(B1,F)/norm(B1)+dfrms;
   fd2=dot(B2,F)/norm(B2)+dfrms;
   fd3=dot(B3,F)/norm(B3)+dfrms;
   fd4=dot(B4,F)/norm(B4)+dfrms;
   fd5=dot(B5,F)/norm(B5)+dfrms;
   fd6=dot(B6,F)/norm(B6)+dfrms;
   fd7=dot(B7,F)/norm(B7)+dfrms;
   fd8=dot(B8,F)/norm(B8)+dfrms;
   fd9=dot(B9,F)/norm(B9)+dfrms;

%Find differential RDOAs at present time
   drdoa2a=-wl*T*(fd1-fd2);
   drdoa3a=-wl*T*(fd1-fd3);
   drdoa4a=-wl*T*(fd1-fd4);
   drdoa5a=-wl*T*(fd1-fd5);
   drdoa6a=-wl*T*(fd1-fd6);
   drdoa7a=-wl*T*(fd1-fd7);
   drdoa8a=-wl*T*(fd1-fd8);
   drdoa9a=-wl*T*(fd1-fd9);

%Find source-to-receiver vectors at next time
   B1=source1-RL(1,:);
   B2=source1-RL(2,:);
   B3=source1-RL(3,:);
   B4=source1-RL(4,:);
   B5=source1-RL(5,:);
   B6=source1-RL(6,:);
   B7=source1-RL(7,:);
   B8=source1-RL(8,:);
   B9=source1-RL(9,:);
```

```
%Find Doppler frequency vector at next time
   F1=v1/wl;

%Find Doppler frequencies at receivers at next time
   fd1=dot(B1,F1)/norm(B1)+dfrms;
   fd2=dot(B2,F1)/norm(B2)+dfrms;
   fd3=dot(B3,F1)/norm(B3)+dfrms;
   fd4=dot(B4,F1)/norm(B4)+dfrms;
   fd5=dot(B5,F1)/norm(B5)+dfrms;
   fd6=dot(B6,F1)/norm(B6)+dfrms;
   fd7=dot(B7,F1)/norm(B7)+dfrms;
   fd8=dot(B8,F1)/norm(B8)+dfrms;
   fd9=dot(B9,F1)/norm(B9)+dfrms;

%Find differential RDOAs at next time
   drdoa2b=-wl*T*(fd1-fd2);
   drdoa3b=-wl*T*(fd1-fd3);
   drdoa4b=-wl*T*(fd1-fd4);
   drdoa5b=-wl*T*(fd1-fd5);
   drdoa6b=-wl*T*(fd1-fd6);
   drdoa7b=-wl*T*(fd1-fd7);
   drdoa8b=-wl*T*(fd1-fd8);
   drdoa9b=-wl*T*(fd1-fd9);

%Find average differential RDOAs
   drdoa2=(drdoa2a+drdoa2b)/2;
   drdoa3=(drdoa3a+drdoa3b)/2;
   drdoa4=(drdoa4a+drdoa4b)/2;
   drdoa5=(drdoa5a+drdoa5b)/2;
   drdoa6=(drdoa6a+drdoa6b)/2;
   drdoa7=(drdoa7a+drdoa7b)/2;
   drdoa8=(drdoa8a+drdoa8b)/2;
   drdoa9=(drdoa9a+drdoa9b)/2;

%Find RDOAs
   da(1)=d(1)+drdoa2;
   da(2)=d(2)+drdoa3;
   da(3)=d(3)+drdoa4;
   da(4)=d(4)+drdoa5;
   da(5)=d(5)+drdoa6;
   da(6)=d(6)+drdoa7;
   da(7)=d(7)+drdoa8;
   da(8)=d(8)+drdoa9;

%Reset range differences
   d=da;

%Find RDOAs from geometry
   db(1) = norm(source1 - RL(2,:)) - norm(source1 - RL(1,:));
   db(2) = norm(source1 - RL(3,:)) - norm(source1 - RL(1,:));
   db(3) = norm(source1 - RL(4,:)) - norm(source1 - RL(1,:));
   db(4) = norm(source1 - RL(5,:)) - norm(source1 - RL(1,:));
   db(5) = norm(source1 - RL(6,:)) - norm(source1 - RL(1,:));
   db(6) = norm(source1 - RL(7,:)) - norm(source1 - RL(1,:));
   db(7) = norm(source1 - RL(8,:)) - norm(source1 - RL(1,:));
   db(8) = norm(source1 - RL(9,:)) - norm(source1 - RL(1,:));
```

```
%Fill matrices for time, position, and velocity components
   if ts==0;t=ts+T;xx=x;yy=y;zz=z;vx=v(1);vy=v(2);vz=v(3);
   else xx=[xx x];yy=[yy y];zz=[zz z];t=[t ts+T];...
        vx=[vx v(1)];vy=[vy v(2)];vz=[vz v(3)];
   end;

%Estimated RDOA error
   errt=norm(d - db);

%Fill matrix for estimated RDOA error
   if ts==0;xet=errt;
   else xet=[xet errt];
   end;

%Apply Smith-Abel algorithm to find estimated position from RDOAs
   source_est=smith_abel(RL,d,w,R);

%Estimated position error
   errp=norm(source1 - source_est);

%Fill matrix for estimated position error
   if ts==0;xep=errp;
   else xep=[xep errp];
   end

%Return to trajectory loop
end;

%Obtain position and velocity components vectors
pos=[t' xx' yy' zz'];
pos=pos';
v=[t' vx' vy' vz'];
v=v';

%Output results to screen
merrt=mean(xet);
fprintf('Mean RDOA Error (ft) = %g \n' ,merrt);

merrp=mean(xep);
fprintf('Mean Position Error (ft)  = %g \n' ,merrp);

%Plot graphs
figure(1)
plot(t,xet);grid;
title('RDOA Error vs Time');
xlabel('Time  - sec');ylabel('RDOA error - ft');

figure(2)
plot(t,xep);grid;
title('Position Error vs Time');
xlabel('Time - sec');ylabel('Position Error - ft');

figure(3)
plot3(xx,yy,zz);grid;
hold on
plot3(RL(:,1),RL(:,2),RL(:,3),'*');
hold off
```

```
title('Missile Trajectory and Remote Receiver Locations');
xlabel('ft');ylabel('ft');zlabel('ft');
```

trajalln.m

```
%Position from FDOA using Smith-Abel algorithm
%Inputs missile trajectory and receiver SNR and calculates
%RDOA from FDOA and position with SNR measurement error
%Random white Gaussian noise added to signal
%All receivers utilized throughout entire trajectory
%trajalln.m

%Load White Sands missile trajectory
load traj;
t=traj(1,:);xm=traj(2,:);ym=traj(3,:);zm=traj(4,:);

%Smooth missile trajectory
xms=polyval(polyfit(t,xm,14),t);
yms=polyval(polyfit(t,ym,14),t);
zms=polyval(polyfit(t,zm,14),t);

%Obtain polynomial coefficients for trajectory
xpc=polyfit(t,xm,14);
ypc=polyfit(t,ym,14);
zpc=polyfit(t,zm,14);

%Define coefficients for velocity (first derivative of polynomials)
i=1:14;
xvc=(15-i).*xpc(i);
yvc=(15-i).*ypc(i);
zvc=(15-i).*zpc(i);

%Evaluate velocity polynomials to obtain velocity component vectors
vxs=polyval(xvc,t);
vys=polyval(yvc,t);
vzs=polyval(zvc,t);

%White Sands remote receiver positions (feet)
RL=[-9243.4 -3085.7 -127; -9193.6 -3072.3 -125; -9278.7 -3201.8
    -92.1; -8992.9 -2879.8 -109.3; 1757 430.4 40.4; -6458.5 -2112.3
    -39.3; -3704.2 -1214.7 -33.6; 607 -3517.8 208.3; -5305.4 2653
    -153.6];

%Initialize range difference vectors
d=zeros(8,1);
da=zeros(8,1);
di=zeros(8,1);
db=zeros(8,1);

%Range from non-reference receiver to reference receiver
R=zeros(8,1);
R(1) = norm(RL(1,:) - RL(2,:));
R(2) = norm(RL(1,:) - RL(3,:));
R(3) = norm(RL(1,:) - RL(4,:));
R(4) = norm(RL(1,:) - RL(5,:));
```

```
R(5) = norm(RL(1,:) - RL(6,:));
R(6) = norm(RL(1,:) - RL(7,:));
R(7) = norm(RL(1,:) - RL(8,:));
R(8) = norm(RL(1,:) - RL(9,:));

%Set all weights equal to one (using all receivers)
w=ones(8,1);

%Telemetry and sampling data
fo=2.3e9;             % Hz
wl=(3e8/fo)/.3048;    % feet
T=1e-2;               % sec
Tmax=6.41;            % sec

%Initial range differences, with Receiver 1 as the reference
sourcei=[xms(1) yms(1) zms(1)];

di(1) = norm(sourcei - RL(2,:)) - norm(sourcei - RL(1,:));
di(2) = norm(sourcei - RL(3,:)) - norm(sourcei - RL(1,:));
di(3) = norm(sourcei - RL(4,:)) - norm(sourcei - RL(1,:));
di(4) = norm(sourcei - RL(5,:)) - norm(sourcei - RL(1,:));
di(5) = norm(sourcei - RL(6,:)) - norm(sourcei - RL(1,:));
di(6) = norm(sourcei - RL(7,:)) - norm(sourcei - RL(1,:));
di(7) = norm(sourcei - RL(8,:)) - norm(sourcei - RL(1,:));
di(8) = norm(sourcei - RL(9,:)) - norm(sourcei - RL(1,:));

d=di;

%Input receiver SNR
snrdb=input('snr(dB)=');
snr=10^(snrdb/10);

%Theoretical minimum frequency RMS measurement error
dfrms=sqrt(3/snr)/(pi*T);

%Generate loop for sampling trajectory points
%Read positions and velocities at each sample point (every .01 sec)
for ts=0:T:Tmax;
   sample=ts/T+1;
   sample=round(sample);      %Sample must be an integer value
   x=xms(1,sample);y=yms(1,sample);z=zms(1,sample);
   x1=xms(1,sample+1);y1=yms(1,sample+1);z1=zms(1,sample+1);
   v=[vxs(1,sample) vys(1,sample) vzs(1,sample)];
   v1=[vxs(1,sample+1) vys(1,sample+1) vzs(1,sample+1)];

%Find source position at present time
   source=[x y z];

%Find source position at next time
   source1=[x1 y1 z1];

%Find source-to-receiver vectors at present time
   B1=source-RL(1,:);
   B2=source-RL(2,:);
   B3=source-RL(3,:);
   B4=source-RL(4,:);
   B5=source-RL(5,:);
   B6=source-RL(6,:);
   B7=source-RL(7,:);
   B8=source-RL(8,:);
   B9=source-RL(9,:);

%Find Doppler frequency vector at present time
   F=v/wl;

%Find Doppler frequencies at receivers at present time (with noise)
   fd1=dot(B1,F)/norm(B1)+dfrms*randn;
   fd2=dot(B2,F)/norm(B2)+dfrms*randn;
   fd3=dot(B3,F)/norm(B3)+dfrms*randn;
   fd4=dot(B4,F)/norm(B4)+dfrms*randn;
   fd5=dot(B5,F)/norm(B5)+dfrms*randn;
   fd6=dot(B6,F)/norm(B6)+dfrms*randn;
   fd7=dot(B7,F)/norm(B7)+dfrms*randn;
   fd8=dot(B8,F)/norm(B8)+dfrms*randn;
   fd9=dot(B9,F)/norm(B9)+dfrms*randn;

%Find differential RDOAs at present time
   drdoa2a=-wl*T*(fd1-fd2);
   drdoa3a=-wl*T*(fd1-fd3);
   drdoa4a=-wl*T*(fd1-fd4);
   drdoa5a=-wl*T*(fd1-fd5);
   drdoa6a=-wl*T*(fd1-fd6);
   drdoa7a=-wl*T*(fd1-fd7);
   drdoa8a=-wl*T*(fd1-fd8);
   drdoa9a=-wl*T*(fd1-fd9);

%Find source-to-receiver vectors at next time
   B1=source1-RL(1,:);
   B2=source1-RL(2,:);
   B3=source1-RL(3,:);
   B4=source1-RL(4,:);
   B5=source1-RL(5,:);
   B6=source1-RL(6,:);
   B7=source1-RL(7,:);
   B8=source1-RL(8,:);
   B9=source1-RL(9,:);

%Find Doppler frequency vector at next time
   F1=v1/wl;

%Find Doppler frequencies at receivers at next time (with noise)
   fd1=dot(B1,F1)/norm(B1)+dfrms*randn;
   fd2=dot(B2,F1)/norm(B2)+dfrms*randn;
   fd3=dot(B3,F1)/norm(B3)+dfrms*randn;
   fd4=dot(B4,F1)/norm(B4)+dfrms*randn;
   fd5=dot(B5,F1)/norm(B5)+dfrms*randn;
   fd6=dot(B6,F1)/norm(B6)+dfrms*randn;
   fd7=dot(B7,F1)/norm(B7)+dfrms*randn;
   fd8=dot(B8,F1)/norm(B8)+dfrms*randn;
   fd9=dot(B9,F1)/norm(B9)+dfrms*randn;

%Find differential RDOAs at next time
   drdoa2b=-wl*T*(fd1-fd2);
   drdoa3b=-wl*T*(fd1-fd3);
   drdoa4b=-wl*T*(fd1-fd4);
   drdoa5b=-wl*T*(fd1-fd5);
   drdoa6b=-wl*T*(fd1-fd6);
   drdoa7b=-wl*T*(fd1-fd7);
   drdoa8b=-wl*T*(fd1-fd8);
   drdoa9b=-wl*T*(fd1-fd9);

%Find average differential RDOAs
   drdoa2=(drdoa2a+drdoa2b)/2;
   drdoa3=(drdoa3a+drdoa3b)/2;
   drdoa4=(drdoa4a+drdoa4b)/2;
   drdoa5=(drdoa5a+drdoa5b)/2;
   drdoa6=(drdoa6a+drdoa6b)/2;
   drdoa7=(drdoa7a+drdoa7b)/2;
   drdoa8=(drdoa8a+drdoa8b)/2;
   drdoa9=(drdoa9a+drdoa9b)/2;

%Find RDOAs
   da(1)=d(1)+drdoa2;
   da(2)=d(2)+drdoa3;
   da(3)=d(3)+drdoa4;
   da(4)=d(4)+drdoa5;
   da(5)=d(5)+drdoa6;
   da(6)=d(6)+drdoa7;
   da(7)=d(7)+drdoa8;
   da(8)=d(8)+drdoa9;

%Reset range differences
   d=da;

%Find RDOAs from geometry
   db(1) = norm(source1 - RL(2,:)) - norm(source1 - RL(1,:));
   db(2) = norm(source1 - RL(3,:)) - norm(source1 - RL(1,:));
   db(3) = norm(source1 - RL(4,:)) - norm(source1 - RL(1,:));
   db(4) = norm(source1 - RL(5,:)) - norm(source1 - RL(1,:));
   db(5) = norm(source1 - RL(6,:)) - norm(source1 - RL(1,:));
   db(6) = norm(source1 - RL(7,:)) - norm(source1 - RL(1,:));
   db(7) = norm(source1 - RL(8,:)) - norm(source1 - RL(1,:));
   db(8) = norm(source1 - RL(9,:)) - norm(source1 - RL(1,:));

%Fill matrices for time, position, and velocity components
   if ts==0;t=ts+T;xx=x;yy=y;zz=z;vx=v(1);vy=v(2);vz=v(3);
   else xx=[xx x];yy=[yy y];zz=[zz z];t=[t ts+T];...
        vx=[vx v(1)];vy=[vy v(2)];vz=[vz v(3)];
   end;

%Estimated RDOA error
   errt=norm(d - db);

%Fill matrix for estimated RDOA error
   if ts==0;xet=errt;
   else xet=[xet errt];
   end;

%Apply Smith-Abel algorithm to find estimated position from RDOAs
   source_est=smith_abel(RL,d,w,R);
```

```
%Estimated position error
   errp=norm(source1 - source_est);

%Fill matrix for estimated position error
   if ts==0;xep=errp;
   else xep=[xep errp];
   end

%Return to trajectory loop
end;

%Obtain position and velocity components vectors
pos=[t' xx' yy' zz'];
pos=pos';
v=[t' vx' vy' vz'];
v=v';

%Output results to screen
merrt=mean(xet);
fprintf('Mean RDOA Error (ft) = %g \n' ,merrt);

merrp=mean(xep);
fprintf('Mean Position Error (ft)  = %g \n' ,merrp);

%Plot graphs
figure(1)
plot(t,xet);grid;
title('RDOA Error vs Time');
xlabel('Time  - sec');ylabel('RDOA error - ft');

figure(2)
plot(t,xep);grid;
title('Position Error vs Time');
xlabel('Time - sec');ylabel('Position Error - ft');

figure(3)
plot3(xx,yy,zz);grid;
hold on
plot3(RL(:,1),RL(:,2),RL(:,3),'*');
hold off
title('Missile Trajectory and Remote Receiver Locations');
xlabel('ft');ylabel('ft');zlabel('ft');
```

trajallnmc.m

```
%Position from FDOA using Smith-Abel method
%Inputs missile trajectory and receiver SNR and calculates
%RDOA from FDOA and position with SNR measurement error
%Random white Gaussian noise added to signal
%Monte Carlo simulation, 15-40 dB, 600 iterations
%All receivers utilized throughout entire trajectory
%trajallnmc.m

%Load White Sands missile trajectory
load traj;
```

```
t=traj(1,:);xm=traj(2,:);ym=traj(3,:);zm=traj(4,:);

%Smooth missile trajectory
xms=polyval(polyfit(t,xm,14),t);
yms=polyval(polyfit(t,ym,14),t);
zms=polyval(polyfit(t,zm,14),t);

%Obtain polynomial coefficients for trajectory
xpc=polyfit(t,xm,14);
ypc=polyfit(t,ym,14);
zpc=polyfit(t,zm,14);

%Define coefficients for velocity (first derivative of polynomials)
i=1:14;
xvc=(15-i).*xpc(i);
yvc=(15-i).*ypc(i);
zvc=(15-i).*zpc(i);

%Evaluate velocity polynomials to obtain velocity component vectors
vxs=polyval(xvc,t);
vys=polyval(yvc,t);
vzs=polyval(zvc,t);

%White Sands remote receiver positions (feet)
RL=[-9243.4 -3085.7 -127; -9193.6 -3072.3 -125; -9278.7 -3201.8
    -92.1; -8992.9 -2879.8 -109.3; 1757 430.4 40.4; -6458.5 -2112.3
    -39.3; -3704.2 -1214.7 -33.6; 607 -3517.8 208.3; -5305.4 2653
    -153.6];

%Initialize range difference vectors
d=zeros(8,1);
da=zeros(8,1);
di=zeros(8,1);
db=zeros(8,1);

%Range from non-reference receiver to reference receiver
R=zeros(8,1);
R(1) = norm(RL(1,:) - RL(2,:));
R(2) = norm(RL(1,:) - RL(3,:));
R(3) = norm(RL(1,:) - RL(4,:));
R(4) = norm(RL(1,:) - RL(5,:));
R(5) = norm(RL(1,:) - RL(6,:));
R(6) = norm(RL(1,:) - RL(7,:));
R(7) = norm(RL(1,:) - RL(8,:));
R(8) = norm(RL(1,:) - RL(9,:));

%Set all weights equal to one (using all receivers)
w=ones(8,1);

%Telemetry and sampling data
fo=2.3e9;              % Hz
wl=(3e8/fo)/.3048;     % feet
T=1e-2;                % sec
Tmax=6.41;             % sec

%Start SNR loop for receiver SNR values (15-40 dB)
for snrdb=15:40;
```

```
   snr=10^(snrdb/10);

%Theoretical minimum frequency RMS measurement error
   dfrms=sqrt(3/snr)/(pi*T);

%Start Monte Carlo loop (600 iterations)
   for count=1:600;

%Initial range differences, with Receiver 1 as the reference
      sourcei=[xms(1) yms(1) zms(1)];

      di(1) = norm(sourcei - RL(2,:)) - norm(sourcei - RL(1,:));
      di(2) = norm(sourcei - RL(3,:)) - norm(sourcei - RL(1,:));
      di(3) = norm(sourcei - RL(4,:)) - norm(sourcei - RL(1,:));
      di(4) = norm(sourcei - RL(5,:)) - norm(sourcei - RL(1,:));
      di(5) = norm(sourcei - RL(6,:)) - norm(sourcei - RL(1,:));
      di(6) = norm(sourcei - RL(7,:)) - norm(sourcei - RL(1,:));
      di(7) = norm(sourcei - RL(8,:)) - norm(sourcei - RL(1,:));
      di(8) = norm(sourcei - RL(9,:)) - norm(sourcei - RL(1,:));

      d=di;

%Generate loop for sampling trajectory points
%Read positions and velocities at each sample point (every .01 sec)
      for ts=0:T:Tmax;
         sample=ts/T+1;
         sample=round(sample);      %Sample must be an integer value
         x=xms(1,sample);y=yms(1,sample);z=zms(1,sample);
         x1=xms(1,sample+1);y1=yms(1,sample+1);z1=zms(1,sample+1);
         v=[vxs(1,sample) vys(1,sample) vzs(1,sample)];
         v1=[vxs(1,sample+1) vys(1,sample+1) vzs(1,sample+1)];

%Find source position at present time
         source=[x y z];

%Find source position at next time
         source1=[x1 y1 z1];

%Find source-to-receiver vectors at present time
         B1=source-RL(1,:);
         B2=source-RL(2,:);
         B3=source-RL(3,:);
         B4=source-RL(4,:);
         B5=source-RL(5,:);
         B6=source-RL(6,:);
         B7=source-RL(7,:);
         B8=source-RL(8,:);
         B9=source-RL(9,:);

%Find Doppler frequency vector at present time
         F=v/wl;

%Find Doppler frequencies at receivers at present time (with noise)
         fd1=dot(B1,F)/norm(B1)+dfrms*randn;
         fd2=dot(B2,F)/norm(B2)+dfrms*randn;
         fd3=dot(B3,F)/norm(B3)+dfrms*randn;
         fd4=dot(B4,F)/norm(B4)+dfrms*randn;
         fd5=dot(B5,F)/norm(B5)+dfrms*randn;
         fd6=dot(B6,F)/norm(B6)+dfrms*randn;
         fd7=dot(B7,F)/norm(B7)+dfrms*randn;
         fd8=dot(B8,F)/norm(B8)+dfrms*randn;
         fd9=dot(B9,F)/norm(B9)+dfrms*randn;

%Find differential RDOAs at present time
         drdoa2a=-wl*T*(fd1-fd2);
         drdoa3a=-wl*T*(fd1-fd3);
         drdoa4a=-wl*T*(fd1-fd4);
         drdoa5a=-wl*T*(fd1-fd5);
         drdoa6a=-wl*T*(fd1-fd6);
         drdoa7a=-wl*T*(fd1-fd7);
         drdoa8a=-wl*T*(fd1-fd8);
         drdoa9a=-wl*T*(fd1-fd9);

%Find source-to-receiver vectors at next time
         B1=source1-RL(1,:);
         B2=source1-RL(2,:);
         B3=source1-RL(3,:);
         B4=source1-RL(4,:);
         B5=source1-RL(5,:);
         B6=source1-RL(6,:);
         B7=source1-RL(7,:);
         B8=source1-RL(8,:);
         B9=source1-RL(9,:);

%Find Doppler frequency vector at next time
         F1=v1/wl;

%Find Doppler frequencies at receivers at next time (with noise)
         fd1=dot(B1,F1)/norm(B1)+dfrms*randn;
         fd2=dot(B2,F1)/norm(B2)+dfrms*randn;
         fd3=dot(B3,F1)/norm(B3)+dfrms*randn;
         fd4=dot(B4,F1)/norm(B4)+dfrms*randn;
         fd5=dot(B5,F1)/norm(B5)+dfrms*randn;
         fd6=dot(B6,F1)/norm(B6)+dfrms*randn;
         fd7=dot(B7,F1)/norm(B7)+dfrms*randn;
         fd8=dot(B8,F1)/norm(B8)+dfrms*randn;
         fd9=dot(B9,F1)/norm(B9)+dfrms*randn;

%Find differential RDOAs at next time
         drdoa2b=-wl*T*(fd1-fd2);
         drdoa3b=-wl*T*(fd1-fd3);
         drdoa4b=-wl*T*(fd1-fd4);
         drdoa5b=-wl*T*(fd1-fd5);
         drdoa6b=-wl*T*(fd1-fd6);
         drdoa7b=-wl*T*(fd1-fd7);
         drdoa8b=-wl*T*(fd1-fd8);
         drdoa9b=-wl*T*(fd1-fd9);

%Find average differential RDOAs
         drdoa2=(drdoa2a+drdoa2b)/2;
         drdoa3=(drdoa3a+drdoa3b)/2;
         drdoa4=(drdoa4a+drdoa4b)/2;
         drdoa5=(drdoa5a+drdoa5b)/2;
         drdoa6=(drdoa6a+drdoa6b)/2;
         drdoa7=(drdoa7a+drdoa7b)/2;
         drdoa8=(drdoa8a+drdoa8b)/2;
         drdoa9=(drdoa9a+drdoa9b)/2;

%Find RDOAs
         da(1)=d(1)+drdoa2;
         da(2)=d(2)+drdoa3;
         da(3)=d(3)+drdoa4;
         da(4)=d(4)+drdoa5;
         da(5)=d(5)+drdoa6;
         da(6)=d(6)+drdoa7;
         da(7)=d(7)+drdoa8;
         da(8)=d(8)+drdoa9;

%Reset range differences
         d=da;

%Find RDOAs from geometry
         db(1) = norm(source1 - RL(2,:)) - norm(source1 - RL(1,:));
         db(2) = norm(source1 - RL(3,:)) - norm(source1 - RL(1,:));
         db(3) = norm(source1 - RL(4,:)) - norm(source1 - RL(1,:));
         db(4) = norm(source1 - RL(5,:)) - norm(source1 - RL(1,:));
         db(5) = norm(source1 - RL(6,:)) - norm(source1 - RL(1,:));
         db(6) = norm(source1 - RL(7,:)) - norm(source1 - RL(1,:));
         db(7) = norm(source1 - RL(8,:)) - norm(source1 - RL(1,:));
         db(8) = norm(source1 - RL(9,:)) - norm(source1 - RL(1,:));

%Estimated RDOA error
         errt=norm(d - db);

%Fill matrix for estimated RDOA error
         if ts==0;xet=errt;
         else xet=[xet errt];
         end;

%Apply Smith-Abel algorithm to find estimated position from RDOAs
         source_est=smith_abel(RL,d,w,R);

%Estimated position error
         errp=norm(source1 - source_est);

%Fill matrix for estimated position error
         if ts==0;xep=errp;
         else xep=[xep errp];
         end

%Return to trajectory loop
      end;

%Calculate mean RDOA and position errors for missile trajectory
%Fill matrices for these values
      merrt=mean(xet);
      merrp=mean(xep);
      if count==1;meanxet=merrt;meanxep=merrp;
      else meanxet=[meanxet merrt];meanxep=[meanxep merrp];end
```

```
%Return to Monte Carlo loop
   end;

%Calculate and print mean values for Monte Carlo iterations
   mcet=mean(meanxet);
   mce=mean(meanxep);
   fprintf('SNR(dB) = %g\n', snrdb);
   fprintf('Error in RDOA (ft) = %g\n', mcet);
   fprintf('Error in Position (ft) = %g\n', mce);

%Fill SNR, RDOA, and position error matrices for plotting
   if snrdb==15;snrdbm=snrdb;mcetm=mcet;mcem=mce;
   else snrdbm=[snrdbm snrdb];mcetm=[mcetm mcet];mcem=[mcem mce];end;

%Return to SNR loop
end;

%Save data to workspace
save data snrdbm mcetm mcem;

%Plot graphs
figure(1)
plot(snrdbm,mcetm);grid;
title('Range Difference Error vs. SNR');
xlabel('SNR - dB');ylabel('Range Difference Error - ft');

figure(2)
plot(snrdbm,mcem);grid;
title('Position Error vs. SNR');
xlabel('SNR - dB');ylabel('Position Error - ft');
```

angles.m

```
%Calculates azimuth and elevation angles for each receiver vs. time
%The matrix "pos" obtained from executing trajall.m or trajalln.m
%angles.m

%Load trajectory data (4x642 matrix)
%First row is time, other rows are x,y,z positions
load pos;

%Define receiver location vectors
bs1a=[-9243.4 -3085.7 -127.0];
bs1b=[-9193.6 -3072.3 -125.0];
bs1p=[-9278.7 -3201.8 -92.1];
bs10=[-8992.9 -2879.8 -109.3];
bs11=[1757.0 430.4 40.4];
bs12=[-6458.5 -2112.3 -39.3];
bs14=[-3704.2 -1214.7 -33.6];
bs15=[607.0 -3517.8 208.3];
bs16=[-5305.4 2653.0 -153.6];

%Translate trajectory such that receivers are at origin
traj1a=[pos(2,:)-bs1a(1);pos(3,:)-bs1a(2);pos(4,:)-bs1a(3)];
traj1b=[pos(2,:)-bs1b(1);pos(3,:)-bs1b(2);pos(4,:)-bs1b(3)];
```

```
traj1p=[pos(2,:)-bs1p(1);pos(3,:)-bs1p(2);pos(4,:)-bs1p(3)];
traj10=[pos(2,:)-bs10(1);pos(3,:)-bs10(2);pos(4,:)-bs10(3)];
traj11=[pos(2,:)-bs11(1);pos(3,:)-bs11(2);pos(4,:)-bs11(3)];
traj12=[pos(2,:)-bs12(1);pos(3,:)-bs12(2);pos(4,:)-bs12(3)];
traj14=[pos(2,:)-bs14(1);pos(3,:)-bs14(2);pos(4,:)-bs14(3)];
traj15=[pos(2,:)-bs15(1);pos(3,:)-bs15(2);pos(4,:)-bs15(3)];
traj16=[pos(2,:)-bs16(1);pos(3,:)-bs16(2);pos(4,:)-bs16(3)];

%Calculate azimuth (theta) and elevation (phi) angles
%Convert radians to degrees
[theta1a phi1a r1a]=cart2sph(traj1a(1,:),traj1a(2,:),traj1a(3,:));
theta1a=theta1a*180/pi;
phi1a=phi1a*180/pi;

[theta1b phi1b r1b]=cart2sph(traj1b(1,:),traj1b(2,:),traj1b(3,:));
theta1b=theta1b*180/pi;
phi1b=phi1b*180/pi;

[theta1p phi1p r1p]=cart2sph(traj1p(1,:),traj1p(2,:),traj1p(3,:));
theta1p=theta1p*180/pi;
phi1p=phi1p*180/pi;

[theta10 phi10 r10]=cart2sph(traj10(1,:),traj10(2,:),traj10(3,:));
theta10=theta10*180/pi;
phi10=phi10*180/pi;

[theta11 phi11 r11]=cart2sph(traj11(1,:),traj11(2,:),traj11(3,:));
theta11=theta11*180/pi;
phi11=phi11*180/pi;

[theta12 phi12 r12]=cart2sph(traj12(1,:),traj12(2,:),traj12(3,:));
theta12=theta12*180/pi;
phi12=phi12*180/pi;

[theta14 phi14 r14]=cart2sph(traj14(1,:),traj14(2,:),traj14(3,:));
theta14=theta14*180/pi;
phi14=phi14*180/pi;

[theta15 phi15 r15]=cart2sph(traj15(1,:),traj15(2,:),traj15(3,:));
theta15=theta15*180/pi;
phi15=phi15*180/pi;

[theta16 phi16 r16]=cart2sph(traj16(1,:),traj16(2,:),traj16(3,:));
theta16=theta16*180/pi;
phi16=phi16*180/pi;

%Plot graphs of theta and phi versus trajectory time
figure(1);
plot(pos(1,:),theta1a);
hold on;
plot(pos(1,:),phi1a,'r');grid;
hold off;
title('Angles for Receiver 1a - Reference Receiver');
xlabel('Time - sec');ylabel('Angle - deg');

figure(2);
plot(pos(1,:),theta1b);
```

```
hold on;
plot(pos(1,:),phi1b,'r');grid;
hold off;
title('Angles for Receiver 1b');
xlabel('Time - sec');ylabel('Angle - deg');

figure(3);
plot(pos(1,:),theta1p);
hold on;
plot(pos(1,:),phi1p,'r');grid;
hold off;
title('Angles for Receiver 1p');
xlabel('Time - sec');ylabel('Angle - deg');

figure(4);
plot(pos(1,:),theta10);
hold on;
plot(pos(1,:),phi10,'r');grid;
hold off;
title('Angles for Receiver 10');
xlabel('Time - sec');ylabel('Angle - deg');

figure(5);
plot(pos(1,:),theta11);
hold on;
plot(pos(1,:),phi11,'r');grid;
hold off;
title('Angles for Receiver 11');
xlabel('Time - sec');ylabel('Angle - deg');

figure(6);
plot(pos(1,:),theta12);
hold on;
plot(pos(1,:),phi12,'r');grid;
hold off;
title('Angles for Receiver 12');
xlabel('Time - sec');ylabel('Angle - deg');

figure(7);
plot(pos(1,:),theta14);
hold on;
plot(pos(1,:),phi14,'r');grid;
hold off;
title('Angles for Receiver 14');
xlabel('Time - sec');ylabel('Angle - deg');

figure(8);
plot(pos(1,:),theta15);
hold on;
plot(pos(1,:),phi15,'r');grid;
hold off;
title('Angles for Receiver 15');
xlabel('Time - sec');ylabel('Angle - deg');

figure(9);
plot(pos(1,:),theta16);
hold on;
```

```
plot(pos(1,:),phi16,'r');grid;
hold off;
title('Angles for Receiver 16');
xlabel('Time - sec');ylabel('Angle - deg');
```

trajgroup.m

```
%Position from FDOA using Smith-Abel algorithm
%Inputs missile trajectory and receiver SNR and calculates
%RDOA from FDOA and position with SNR measurement error
%Only receivers with telemetry in 3-dB beam width utilized
%trajgroup.m

%Load White Sands missile trajectory
load traj;
t=traj(1,:);xm=traj(2,:);ym=traj(3,:);zm=traj(4,:);

%Smooth missile trajectory
xms=polyval(polyfit(t,xm,14),t);
yms=polyval(polyfit(t,ym,14),t);
zms=polyval(polyfit(t,zm,14),t);

%Obtain polynomial coefficients for trajectory
xpc=polyfit(t,xm,14);
ypc=polyfit(t,ym,14);
zpc=polyfit(t,zm,14);

%Define coefficients for velocity (first derivative of polynomials)
i=1:14;
xvc=(15-i).*xpc(i);
yvc=(15-i).*ypc(i);
zvc=(15-i).*zpc(i);

%Evaluate velocity polynomials to obtain velocity component vectors
vxs=polyval(xvc,t);
vys=polyval(yvc,t);
vzs=polyval(zvc,t);

%White Sands remote receiver positions (feet)
RL=[-9243.4 -3085.7 -127; -9193.6 -3072.3 -125; -9278.7 -3201.8
    -92.1; -8992.9 -2879.8 -109.3; 1757 430.4 40.4; -6458.5 -2112.3
    -39.3; -3704.2 -1214.7 -33.6; 607 -3517.8 208.3; -5305.4 2653
    -153.6];

%Initialize range difference vectors
d=zeros(8,1);
da=zeros(8,1);
di=zeros(8,1);
db=zeros(8,1);

%Range from non-reference receiver to reference receiver
R=zeros(8,1);
R(1) = norm(RL(1,:) - RL(2,:));
R(2) = norm(RL(1,:) - RL(3,:));
R(3) = norm(RL(1,:) - RL(4,:));
```

```
R(4) = norm(RL(1,:) - RL(5,:));
R(5) = norm(RL(1,:) - RL(6,:));
R(6) = norm(RL(1,:) - RL(7,:));
R(7) = norm(RL(1,:) - RL(8,:));
R(8) = norm(RL(1,:) - RL(9,:));

%Telemetry and sampling data
fo=2.3e9;              % Hz
wl=(3e8/fo)/.3048;     % feet
T=1e-2;                % sec
Tmax=6.41;             % sec

%Initial range differences, with Receiver 1 as the reference
sourcei=[xms(1) yms(1) zms(1)];

di(1) = norm(sourcei - RL(2,:)) - norm(sourcei - RL(1,:));
di(2) = norm(sourcei - RL(3,:)) - norm(sourcei - RL(1,:));
di(3) = norm(sourcei - RL(4,:)) - norm(sourcei - RL(1,:));
di(4) = norm(sourcei - RL(5,:)) - norm(sourcei - RL(1,:));
di(5) = norm(sourcei - RL(6,:)) - norm(sourcei - RL(1,:));
di(6) = norm(sourcei - RL(7,:)) - norm(sourcei - RL(1,:));
di(7) = norm(sourcei - RL(8,:)) - norm(sourcei - RL(1,:));
di(8) = norm(sourcei - RL(9,:)) - norm(sourcei - RL(1,:));

d=di;

%Input receiver SNR
snrdb=input('snr(dB)=');
snr=10^(snrdb/10);

%Theoretical minimum frequency RMS measurement error
dfrms=sqrt(3/snr)/(pi*T);

%Generate loop for sampling trajectory points
%Read positions and velocities at each sample point (every .01 sec)
for ts=0:T:Tmax;
   sample=ts/T+1;
   sample=round(sample);      %Sample must be an integer value
   x=xms(1,sample);y=yms(1,sample);z=zms(1,sample);
   x1=xms(1,sample+1);y1=yms(1,sample+1);z1=zms(1,sample+1);
   v=[vxs(1,sample) vys(1,sample) vzs(1,sample)];
   v1=[vxs(1,sample+1) vys(1,sample+1) vzs(1,sample+1)];

%Find source position at present time
   source=[x y z];

%Find source position at next time
   source1=[x1 y1 z1];

%Find source-to-receiver vectors at present time
   B1=source-RL(1,:);
   B2=source-RL(2,:);
   B3=source-RL(3,:);
   B4=source-RL(4,:);
   B5=source-RL(5,:);
   B6=source-RL(6,:);
   B7=source-RL(7,:);
   B8=source-RL(8,:);
   B9=source-RL(9,:);

%Find Doppler frequency vector at present time
   F=v/wl;

%Find Doppler frequencies at receivers at present time
   fd1=dot(B1,F)/norm(B1)+dfrms;
   fd2=dot(B2,F)/norm(B2)+dfrms;
   fd3=dot(B3,F)/norm(B3)+dfrms;
   fd4=dot(B4,F)/norm(B4)+dfrms;
   fd5=dot(B5,F)/norm(B5)+dfrms;
   fd6=dot(B6,F)/norm(B6)+dfrms;
   fd7=dot(B7,F)/norm(B7)+dfrms;
   fd8=dot(B8,F)/norm(B8)+dfrms;
   fd9=dot(B9,F)/norm(B9)+dfrms;

%Find differential RDOAs at present time
   drdoa2a=-wl*T*(fd1-fd2);
   drdoa3a=-wl*T*(fd1-fd3);
   drdoa4a=-wl*T*(fd1-fd4);
   drdoa5a=-wl*T*(fd1-fd5);
   drdoa6a=-wl*T*(fd1-fd6);
   drdoa7a=-wl*T*(fd1-fd7);
   drdoa8a=-wl*T*(fd1-fd8);
   drdoa9a=-wl*T*(fd1-fd9);

%Find source-to-receiver vectors at next time
   B1=source1-RL(1,:);
   B2=source1-RL(2,:);
   B3=source1-RL(3,:);
   B4=source1-RL(4,:);
   B5=source1-RL(5,:);
   B6=source1-RL(6,:);
   B7=source1-RL(7,:);
   B8=source1-RL(8,:);
   B9=source1-RL(9,:);

%Find Doppler frequency vector at next time
   F1=v1/wl;

%Find Doppler frequencies at receivers at next time
   fd1=dot(B1,F1)/norm(B1)+dfrms;
   fd2=dot(B2,F1)/norm(B2)+dfrms;
   fd3=dot(B3,F1)/norm(B3)+dfrms;
   fd4=dot(B4,F1)/norm(B4)+dfrms;
   fd5=dot(B5,F1)/norm(B5)+dfrms;
   fd6=dot(B6,F1)/norm(B6)+dfrms;
   fd7=dot(B7,F1)/norm(B7)+dfrms;
   fd8=dot(B8,F1)/norm(B8)+dfrms;
   fd9=dot(B9,F1)/norm(B9)+dfrms;

%Find differential RDOAs at next time
   drdoa2b=-wl*T*(fd1-fd2);
   drdoa3b=-wl*T*(fd1-fd3);
   drdoa4b=-wl*T*(fd1-fd4);
   drdoa5b=-wl*T*(fd1-fd5);
   drdoa6b=-wl*T*(fd1-fd6);
   drdoa7b=-wl*T*(fd1-fd7);
   drdoa8b=-wl*T*(fd1-fd8);
   drdoa9b=-wl*T*(fd1-fd9);

%Find average differential RDOAs
   drdoa2=(drdoa2a+drdoa2b)/2;
   drdoa3=(drdoa3a+drdoa3b)/2;
   drdoa4=(drdoa4a+drdoa4b)/2;
   drdoa5=(drdoa5a+drdoa5b)/2;
   drdoa6=(drdoa6a+drdoa6b)/2;
   drdoa7=(drdoa7a+drdoa7b)/2;
   drdoa8=(drdoa8a+drdoa8b)/2;
   drdoa9=(drdoa9a+drdoa9b)/2;

%Find RDOAs
   da(1)=d(1)+drdoa2;
   da(2)=d(2)+drdoa3;
   da(3)=d(3)+drdoa4;
   da(4)=d(4)+drdoa5;
   da(5)=d(5)+drdoa6;
   da(6)=d(6)+drdoa7;
   da(7)=d(7)+drdoa8;
   da(8)=d(8)+drdoa9;

%Reset range differences
   d=da;

%Find RDOAs from geometry
   db(1) = norm(source1 - RL(2,:)) - norm(source1 - RL(1,:));
   db(2) = norm(source1 - RL(3,:)) - norm(source1 - RL(1,:));
   db(3) = norm(source1 - RL(4,:)) - norm(source1 - RL(1,:));
   db(4) = norm(source1 - RL(5,:)) - norm(source1 - RL(1,:));
   db(5) = norm(source1 - RL(6,:)) - norm(source1 - RL(1,:));
   db(6) = norm(source1 - RL(7,:)) - norm(source1 - RL(1,:));
   db(7) = norm(source1 - RL(8,:)) - norm(source1 - RL(1,:));
   db(8) = norm(source1 - RL(9,:)) - norm(source1 - RL(1,:));

%Fill matrices for time, position, and velocity components
   if ts==0;t=ts+T;xx=x;yy=y;zz=z;vx=v(1);vy=v(2);vz=v(3);
   else xx=[xx x];yy=[yy y];zz=[zz z];t=[t ts+T];...
        vx=[vx v(1)];vy=[vy v(2)];vz=[vz v(3)];
   end;

%Use Smith-Abel algorithm to obtain estimated position from RDOAs
%Apply 45 deg 3-dB beam width of receiver antennas for signal reception
%Weights of 1 define receivers used, weights of 0 define those not used
%Receivers 1a,1b,1p,10,11,12,14 and 15 within view at time start
   if ts<2.12
      w=[1;1;1;1;1;1;1;0];
      du=[d(1);d(2);d(3);d(4);d(5);d(6);d(7)];
      dbn=[db(1);db(2);db(3);db(4);db(5);db(6);db(7)];
      source_est=smith_abel(RL,d,w,R);
   end;

%Lose Receiver 12 at 2.12 sec
   if ts>=2.12 & ts<3.29
```

```
      w=[1;1;1;1;0;1;1;0];
      du=[d(1);d(2);d(3);d(4);d(6);d(7)];
      dbn=[db(1);db(2);db(3);db(4);db(6);db(7)];
      source_est=smith_abel(RL,d,w,R);
   end;

%Lose Receiver 14 at 3.29 sec
   if ts>=3.29 & ts<4.49
      w=[1;1;1;1;0;0;1;0];
      du=[d(1);d(2);d(3);d(4);d(7)];
      dbn=[db(1);db(2);db(3);db(4);db(7)];
      source_est=smith_abel(RL,d,w,R);
   end;

%Obtain RDOA for added receiver from estimated position
   if ts==4.49
      d(8)=norm(source_est - RL(9,:)) - norm(source_est - RL(1,:));
      d(8)=d(8)+drdoa9;
   end;

%Lose Receiver 15 and gain Receiver 16 at 4.49 sec
   if ts>=4.49 & ts<5.95
      w=[1;1;1;1;0;0;0;1];
      du=[d(1);d(2);d(3);d(4);d(8)];
      dbn=[db(1);db(2);db(3);db(4);db(8)];
      source_est=smith_abel(RL,d,w,R);
   end;

%Lose Receiver 11 at 5.95 sec
   if ts>=5.95 & ts<=Tmax
      w=[1;1;1;0;0;0;0;1];
      du=[d(1);d(2);d(3);d(8)];
      dbn=[db(1);db(2);db(3);db(8)];
      source_est=smith_abel(RL,d,w,R);
   end;

%Estimated RDOA error of used range differences
   errt=norm(du - dbn);

%Fill matrix for estimated RDOA error
   if ts==0;xet=errt;
   else xet=[xet errt];
   end;

%Estimated position error
   errp=norm(source1 - source_est);

%Fill matrix for estimated position error
   if ts==0;xep=errp;
   else xep=[xep errp];
   end

%Return to trajectory loop
end;

%Obtain position and velocity components vectors
pos=[t' xx' yy' zz'];
```

```
pos=pos';
v=[t' vx' vy' vz'];
v=v';
%Output results to screen
merrt=mean(xet);
fprintf('Mean RDOA Error (ft) = %g \n' ,merrt);

merrp=mean(xep);
fprintf('Mean Position Error (ft)  = %g \n' ,merrp);

%Plot graphs
figure(1)
plot(t,xet);grid;
title('RDOA Error vs Time');
xlabel('Time  - sec');ylabel('RDOA error - ft');

figure(2)
plot(t,xep);grid;
title('Position Error vs Time');
xlabel('Time - sec');ylabel('Position Error - ft');

figure(3)
plot3(xx,yy,zz);grid;
hold on
plot3(RL(:,1),RL(:,2),RL(:,3),'*');
hold off
title('Missile Trajectory and Remote Receiver Locations');
xlabel('ft');ylabel('ft');zlabel('ft');
```

trajgroupn.m

```
%Position from FDOA using Smith-Abel algorithm
%Inputs missile trajectory and receiver SNR and calculates
%RDOA from FDOA and position with SNR measurement error
%Random white Gaussian noise added to signal
%Only receivers with telemetry in 3-dB beam width utilized
%trajgroupn.m

%Load White Sands missile trajectory
load traj;
t=traj(1,:);xm=traj(2,:);ym=traj(3,:);zm=traj(4,:);

%Smooth missile trajectory
xms=polyval(polyfit(t,xm,14),t);
yms=polyval(polyfit(t,ym,14),t);
zms=polyval(polyfit(t,zm,14),t);

%Obtain polynomial coefficients for trajectory
xpc=polyfit(t,xm,14);
ypc=polyfit(t,ym,14);
zpc=polyfit(t,zm,14);

%Define coefficients for velocity (first derivative of polynomials)
i=1:14;
xvc=(15-i).*xpc(i);
```

```
yvc=(15-i).*ypc(i);
zvc=(15-i).*zpc(i);

%Evaluate velocity polynomials to obtain velocity component vectors
vxs=polyval(xvc,t);
vys=polyval(yvc,t);
vzs=polyval(zvc,t);

%White Sands remote receiver positions (feet)
RL=[-9243.4 -3085.7 -127; -9193.6 -3072.3 -125; -9278.7 -3201.8
    -92.1; -8992.9 -2879.8 -109.3; 1757 430.4 40.4; -6458.5 -2112.3
    -39.3; -3704.2 -1214.7 -33.6; 607 -3517.8 208.3; -5305.4 2653
    -153.6];

%Initialize range difference vectors
d=zeros(8,1);
da=zeros(8,1);
di=zeros(8,1);
db=zeros(8,1);

%Range from non-reference receiver to reference receiver
R=zeros(8,1);
R(1) = norm(RL(1,:) - RL(2,:));
R(2) = norm(RL(1,:) - RL(3,:));
R(3) = norm(RL(1,:) - RL(4,:));
R(4) = norm(RL(1,:) - RL(5,:));
R(5) = norm(RL(1,:) - RL(6,:));
R(6) = norm(RL(1,:) - RL(7,:));
R(7) = norm(RL(1,:) - RL(8,:));
R(8) = norm(RL(1,:) - RL(9,:));

%Telemetry and sampling data
fo=2.3e9;               % Hz
wl=(3e8/fo)/.3048;      % feet
T=1e-2;                 % sec
Tmax=6.41;              % sec

%Initial range differences, with Receiver 1 as the reference
sourcei=[xms(1) yms(1) zms(1)];

di(1) = norm(sourcei - RL(2,:)) - norm(sourcei - RL(1,:));
di(2) = norm(sourcei - RL(3,:)) - norm(sourcei - RL(1,:));
di(3) = norm(sourcei - RL(4,:)) - norm(sourcei - RL(1,:));
di(4) = norm(sourcei - RL(5,:)) - norm(sourcei - RL(1,:));
di(5) = norm(sourcei - RL(6,:)) - norm(sourcei - RL(1,:));
di(6) = norm(sourcei - RL(7,:)) - norm(sourcei - RL(1,:));
di(7) = norm(sourcei - RL(8,:)) - norm(sourcei - RL(1,:));
di(8) = norm(sourcei - RL(9,:)) - norm(sourcei - RL(1,:));

d=di;

%Input receiver SNR
snrdb=input('snr(dB)=');
snr=10^(snrdb/10);

%Theoretical minimum frequency RMS measurement error
dfrms=sqrt(3/snr)/(pi*T);
```

```
%Generate loop for sampling trajectory points
%Read positions and velocities at each sample point (every .01 sec)
for ts=0:T:Tmax;
   sample=ts/T+1;
   sample=round(sample);      %Sample must be an integer value
   x=xms(1,sample);y=yms(1,sample);z=zms(1,sample);
   x1=xms(1,sample+1);y1=yms(1,sample+1);z1=zms(1,sample+1);
   v=[vxs(1,sample) vys(1,sample) vzs(1,sample)];
   v1=[vxs(1,sample+1) vys(1,sample+1) vzs(1,sample+1)];

%Find source position at present time
   source=[x y z];

%Find source position at next time
   source1=[x1 y1 z1];

%Find source-to-base station vectors at present time
   B1=source-RL(1,:);
   B2=source-RL(2,:);
   B3=source-RL(3,:);
   B4=source-RL(4,:);
   B5=source-RL(5,:);
   B6=source-RL(6,:);
   B7=source-RL(7,:);
   B8=source-RL(8,:);
   B9=source-RL(9,:);

%Find Doppler frequency vector at present time
   F=v/wl;

%Find Doppler frequencies at receivers at present time (with noise)
   fd1=dot(B1,F)/norm(B1)+dfrms*randn;
   fd2=dot(B2,F)/norm(B2)+dfrms*randn;
   fd3=dot(B3,F)/norm(B3)+dfrms*randn;
   fd4=dot(B4,F)/norm(B4)+dfrms*randn;
   fd5=dot(B5,F)/norm(B5)+dfrms*randn;
   fd6=dot(B6,F)/norm(B6)+dfrms*randn;
   fd7=dot(B7,F)/norm(B7)+dfrms*randn;
   fd8=dot(B8,F)/norm(B8)+dfrms*randn;
   fd9=dot(B9,F)/norm(B9)+dfrms*randn;

%Find differential RDOAs at present time
   drdoa2a=-wl*T*(fd1-fd2);
   drdoa3a=-wl*T*(fd1-fd3);
   drdoa4a=-wl*T*(fd1-fd4);
   drdoa5a=-wl*T*(fd1-fd5);
   drdoa6a=-wl*T*(fd1-fd6);
   drdoa7a=-wl*T*(fd1-fd7);
   drdoa8a=-wl*T*(fd1-fd8);
   drdoa9a=-wl*T*(fd1-fd9);

%Find source-to-receiver vectors at next time
   B1=source1-RL(1,:);
   B2=source1-RL(2,:);
   B3=source1-RL(3,:);
   B4=source1-RL(4,:);
   B5=source1-RL(5,:);
   B6=source1-RL(6,:);
   B7=source1-RL(7,:);
   B8=source1-RL(8,:);
   B9=source1-RL(9,:);

%Find Doppler frequency vector at next time
   F1=v1/wl;

%Find Doppler frequencies at receivers at next time (with noise)
   fd1=dot(B1,F1)/norm(B1)+dfrms*randn;
   fd2=dot(B2,F1)/norm(B2)+dfrms*randn;
   fd3=dot(B3,F1)/norm(B3)+dfrms*randn;
   fd4=dot(B4,F1)/norm(B4)+dfrms*randn;
   fd5=dot(B5,F1)/norm(B5)+dfrms*randn;
   fd6=dot(B6,F1)/norm(B6)+dfrms*randn;
   fd7=dot(B7,F1)/norm(B7)+dfrms*randn;
   fd8=dot(B8,F1)/norm(B8)+dfrms*randn;
   fd9=dot(B9,F1)/norm(B9)+dfrms*randn;

%Find differential RDOAs at next time
   drdoa2b=-wl*T*(fd1-fd2);
   drdoa3b=-wl*T*(fd1-fd3);
   drdoa4b=-wl*T*(fd1-fd4);
   drdoa5b=-wl*T*(fd1-fd5);
   drdoa6b=-wl*T*(fd1-fd6);
   drdoa7b=-wl*T*(fd1-fd7);
   drdoa8b=-wl*T*(fd1-fd8);
   drdoa9b=-wl*T*(fd1-fd9);

%Find average differential RDOAs
   drdoa2=(drdoa2a+drdoa2b)/2;
   drdoa3=(drdoa3a+drdoa3b)/2;
   drdoa4=(drdoa4a+drdoa4b)/2;
   drdoa5=(drdoa5a+drdoa5b)/2;
   drdoa6=(drdoa6a+drdoa6b)/2;
   drdoa7=(drdoa7a+drdoa7b)/2;
   drdoa8=(drdoa8a+drdoa8b)/2;
   drdoa9=(drdoa9a+drdoa9b)/2;

%Find RDOAs
   da(1)=d(1)+drdoa2;
   da(2)=d(2)+drdoa3;
   da(3)=d(3)+drdoa4;
   da(4)=d(4)+drdoa5;
   da(5)=d(5)+drdoa6;
   da(6)=d(6)+drdoa7;
   da(7)=d(7)+drdoa8;
   da(8)=d(8)+drdoa9;

%Reset range differences
   d=da;

%Find RDOAs from geometry
   db(1) = norm(source1 - RL(2,:)) - norm(source1 - RL(1,:));
   db(2) = norm(source1 - RL(3,:)) - norm(source1 - RL(1,:));
   db(3) = norm(source1 - RL(4,:)) - norm(source1 - RL(1,:));
   db(4) = norm(source1 - RL(5,:)) - norm(source1 - RL(1,:));
   db(5) = norm(source1 - RL(6,:)) - norm(source1 - RL(1,:));
   db(6) = norm(source1 - RL(7,:)) - norm(source1 - RL(1,:));
   db(7) = norm(source1 - RL(8,:)) - norm(source1 - RL(1,:));
   db(8) = norm(source1 - RL(9,:)) - norm(source1 - RL(1,:));

%Fill matrices for time, position, and velocity components
   if ts==0;t=ts+T;xx=x;yy=y;zz=z;vx=v(1);vy=v(2);vz=v(3);
   else xx=[xx x];yy=[yy y];zz=[zz z];t=[t ts+T];...
        vx=[vx v(1)];vy=[vy v(2)];vz=[vz v(3)];
   end;

%Use Smith-Abel algorithm to obtain estimated position from RDOAs
%Apply 45 deg 3-dB beam width of receiver antennas for signal reception
%Weights of 1 define receivers used, weights of 0 define those not used
%Receivers 1a,1b,1p,10,11,12,14 and 15 within view at time start
   if ts<2.12
      w=[1;1;1;1;1;1;1;0];
      du=[d(1);d(2);d(3);d(4);d(5);d(6);d(7)];
      dbn=[db(1);db(2);db(3);db(4);db(5);db(6);db(7)];
      source_est=smith_abel(RL,d,w,R);
   end;

%Lose Receiver 12 at 2.12 sec
   if ts>=2.12 & ts<3.29
      w=[1;1;1;1;0;1;1;0];
      du=[d(1);d(2);d(3);d(4);d(6);d(7)];
      dbn=[db(1);db(2);db(3);db(4);db(6);db(7)];
      source_est=smith_abel(RL,d,w,R);
   end;

%Lose Receiver 14 at 3.29 sec
   if ts>=3.29 & ts<4.49
      w=[1;1;1;1;0;0;1;0];
      du=[d(1);d(2);d(3);d(4);d(7)];
      dbn=[db(1);db(2);db(3);db(4);db(7)];
      source_est=smith_abel(RL,d,w,R);
   end;

%Obtain RDOA for added receiver from estimated position
   if ts==4.49
      d(8)=norm(source_est - RL(9,:)) - norm(source_est - RL(1,:));
      d(8)=d(8)+drdoa9;
   end;

%Lose Receiver 15 and gain Receiver 16 at 4.49 sec
   if ts>=4.49 & ts<5.95
      w=[1;1;1;1;0;0;0;1];
      du=[d(1);d(2);d(3);d(4);d(8)];
      dbn=[db(1);db(2);db(3);db(4);db(8)];
      source_est=smith_abel(RL,d,w,R);
   end;

%Lose Receiver 11 at 5.95 sec
   if ts>=5.95 & ts<=Tmax
      w=[1;1;1;0;0;0;0;1];
      du=[d(1);d(2);d(3);d(8)];
      dbn=[db(1);db(2);db(3);db(8)];
      source_est=smith_abel(RL,d,w,R);
   end;

%Estimated RDOA error from used range differences
   errt=norm(du - dbn);

%Fill matrix for estimated RDOA error
   if ts==0;xet=errt;
   else xet=[xet errt];
   end;

%Estimated position error
   errp=norm(source1 - source_est);

%Fill matrix for estimated position error
   if ts==0;xep=errp;
   else xep=[xep errp];
   end

%Return to trajectory loop
end;

%Obtain position and velocity components vectors
pos=[t' xx' yy' zz'];
pos=pos';
v=[t' vx' vy' vz'];
v=v';

%Output results to screen
merrt=mean(xet);
fprintf('Mean RDOA Error (ft) = %g \n' ,merrt);

merrp=mean(xep);
fprintf('Mean Position Error (ft)  = %g \n' ,merrp);

%Plot graphs
figure(1)
plot(t,xet);grid;
title('RDOA Error vs Time');
xlabel('Time  - sec');ylabel('RDOA error - ft');

figure(2)
plot(t,xep);grid;
title('Position Error vs Time');
xlabel('Time - sec');ylabel('Position Error - ft');

figure(3)
plot3(xx,yy,zz);grid;
hold on
plot3(RL(:,1),RL(:,2),RL(:,3),'*');
hold off
title('Missile Trajectory and Remote Receiver Locations');
xlabel('ft');ylabel('ft');zlabel('ft');
```

trajgroupnmc.m

```
%Position from FDOA using Smith-Abel algorithm
%Inputs missile trajectory and receiver SNR and calculates
%RDOA from FDOA and position with SNR measurement error
%Random white Gaussian noise added to signal
%Monte Carlo simulation, 15-40 dB, 600 iterations
%Only receivers with telemetry in 3-dB beam width utilized
%trajgroupnmc.m

%Load White Sands missile trajectory
load traj;
t=traj(1,:);xm=traj(2,:);ym=traj(3,:);zm=traj(4,:);

%Smooth missile trajectory
xms=polyval(polyfit(t,xm,14),t);
yms=polyval(polyfit(t,ym,14),t);
zms=polyval(polyfit(t,zm,14),t);

%Obtain polynomial coefficients for trajectory
xpc=polyfit(t,xm,14);
ypc=polyfit(t,ym,14);
zpc=polyfit(t,zm,14);

%Define coefficients for velocity (first derivative of polynomials)
i=1:14;
xvc=(15-i).*xpc(i);
yvc=(15-i).*ypc(i);
zvc=(15-i).*zpc(i);

%Evaluate velocity polynomials to obtain velocity component vectors
vxs=polyval(xvc,t);
vys=polyval(yvc,t);
vzs=polyval(zvc,t);

%White Sands remote receiver positions (feet)
RL=[-9243.4 -3085.7 -127; -9193.6 -3072.3 -125; -9278.7 -3201.8
    -92.1; -8992.9 -2879.8 -109.3; 1757 430.4 40.4; -6458.5 -2112.3
    -39.3; -3704.2 -1214.7 -33.6; 607 -3517.8 208.3; -5305.4 2653
    -153.6];

%Initialize range difference vectors
d=zeros(8,1);
da=zeros(8,1);
di=zeros(8,1);
db=zeros(8,1);

%Range from non-reference receiver to reference receiver
R=zeros(8,1);
R(1) = norm(RL(1,:) - RL(2,:));
R(2) = norm(RL(1,:) - RL(3,:));
R(3) = norm(RL(1,:) - RL(4,:));
R(4) = norm(RL(1,:) - RL(5,:));
R(5) = norm(RL(1,:) - RL(6,:));
R(6) = norm(RL(1,:) - RL(7,:));
R(7) = norm(RL(1,:) - RL(8,:));
```

```
R(8) = norm(RL(1,:) - RL(9,:));

%Telemetry and sampling data
fo=2.3e9;             % Hz
wl=(3e8/fo)/.3048;    % feet
T=1e-2;               % sec
Tmax=6.41;            % sec

%Start SNR loop for receiver SNR values (15-40 dB)
for snrdb=15:40;
   snr=10^(snrdb/10);

%Theoretical minimum frequency RMS measurement error
   dfrms=sqrt(3/snr)/(pi*T);

%Start Monte Carlo loop (600 iterations)
   for count=1:600;

%Initial range differences, with Receiver 1 as the reference
      sourcei=[xms(1) yms(1) zms(1)];

      di(1) = norm(sourcei - RL(2,:)) - norm(sourcei - RL(1,:));
      di(2) = norm(sourcei - RL(3,:)) - norm(sourcei - RL(1,:));
      di(3) = norm(sourcei - RL(4,:)) - norm(sourcei - RL(1,:));
      di(4) = norm(sourcei - RL(5,:)) - norm(sourcei - RL(1,:));
      di(5) = norm(sourcei - RL(6,:)) - norm(sourcei - RL(1,:));
      di(6) = norm(sourcei - RL(7,:)) - norm(sourcei - RL(1,:));
      di(7) = norm(sourcei - RL(8,:)) - norm(sourcei - RL(1,:));
      di(8) = norm(sourcei - RL(9,:)) - norm(sourcei - RL(1,:));

      d=di;

%Generate loop for sampling trajectory points
%Read positions and velocities at each sample point (every .01 sec)
      for ts=0:T:Tmax;
         sample=ts/T+1;
         sample=round(sample);      %Sample must be an integer value
         x=xms(1,sample);y=yms(1,sample);z=zms(1,sample);
         x1=xms(1,sample+1);y1=yms(1,sample+1);z1=zms(1,sample+1);
         v=[vxs(1,sample) vys(1,sample) vzs(1,sample)];
         v1=[vxs(1,sample+1) vys(1,sample+1) vzs(1,sample+1)];

%Find source position at present time
         source=[x y z];

%Find source position at next time
         source1=[x1 y1 z1];

%Find source-to-base station vectors at present time
         B1=source-RL(1,:);
         B2=source-RL(2,:);
         B3=source-RL(3,:);
         B4=source-RL(4,:);
         B5=source-RL(5,:);
         B6=source-RL(6,:);
         B7=source-RL(7,:);
         B8=source-RL(8,:);
```

```
        B9=source-RL(9,:);

%Find Doppler frequency vector at present time
        F=v/wl;

%Find Doppler frequencies at receivers at present time (with noise)
        fd1=dot(B1,F)/norm(B1)+dfrms*randn;
        fd2=dot(B2,F)/norm(B2)+dfrms*randn;
        fd3=dot(B3,F)/norm(B3)+dfrms*randn;
        fd4=dot(B4,F)/norm(B4)+dfrms*randn;
        fd5=dot(B5,F)/norm(B5)+dfrms*randn;
        fd6=dot(B6,F)/norm(B6)+dfrms*randn;
        fd7=dot(B7,F)/norm(B7)+dfrms*randn;
        fd8=dot(B8,F)/norm(B8)+dfrms*randn;
        fd9=dot(B9,F)/norm(B9)+dfrms*randn;

%Find differential RDOAs at present time
        drdoa2a=-wl*T*(fd1-fd2);
        drdoa3a=-wl*T*(fd1-fd3);
        drdoa4a=-wl*T*(fd1-fd4);
        drdoa5a=-wl*T*(fd1-fd5);
        drdoa6a=-wl*T*(fd1-fd6);
        drdoa7a=-wl*T*(fd1-fd7);
        drdoa8a=-wl*T*(fd1-fd8);
        drdoa9a=-wl*T*(fd1-fd9);

%Find source-to-receiver vectors at next time
        B1=source1-RL(1,:);
        B2=source1-RL(2,:);
        B3=source1-RL(3,:);
        B4=source1-RL(4,:);
        B5=source1-RL(5,:);
        B6=source1-RL(6,:);
        B7=source1-RL(7,:);
        B8=source1-RL(8,:);
        B9=source1-RL(9,:);

%Find Doppler frequency vector at next time
        F1=v1/wl;

%Find Doppler frequencies at receivers at next time (with noise)
        fd1=dot(B1,F1)/norm(B1)+dfrms*randn;
        fd2=dot(B2,F1)/norm(B2)+dfrms*randn;
        fd3=dot(B3,F1)/norm(B3)+dfrms*randn;
        fd4=dot(B4,F1)/norm(B4)+dfrms*randn;
        fd5=dot(B5,F1)/norm(B5)+dfrms*randn;
        fd6=dot(B6,F1)/norm(B6)+dfrms*randn;
        fd7=dot(B7,F1)/norm(B7)+dfrms*randn;
        fd8=dot(B8,F1)/norm(B8)+dfrms*randn;
        fd9=dot(B9,F1)/norm(B9)+dfrms*randn;

%Find differential RDOAs at next time
        drdoa2b=-wl*T*(fd1-fd2);
        drdoa3b=-wl*T*(fd1-fd3);
        drdoa4b=-wl*T*(fd1-fd4);
        drdoa5b=-wl*T*(fd1-fd5);
        drdoa6b=-wl*T*(fd1-fd6);
```

```
         drdoa7b=-wl*T*(fd1-fd7);
         drdoa8b=-wl*T*(fd1-fd8);
         drdoa9b=-wl*T*(fd1-fd9);

%Find average differential RDOAs
         drdoa2=(drdoa2a+drdoa2b)/2;
         drdoa3=(drdoa3a+drdoa3b)/2;
         drdoa4=(drdoa4a+drdoa4b)/2;
         drdoa5=(drdoa5a+drdoa5b)/2;
         drdoa6=(drdoa6a+drdoa6b)/2;
         drdoa7=(drdoa7a+drdoa7b)/2;
         drdoa8=(drdoa8a+drdoa8b)/2;
         drdoa9=(drdoa9a+drdoa9b)/2;

%Find RDOAs
         da(1)=d(1)+drdoa2;
         da(2)=d(2)+drdoa3;
         da(3)=d(3)+drdoa4;
         da(4)=d(4)+drdoa5;
         da(5)=d(5)+drdoa6;
         da(6)=d(6)+drdoa7;
         da(7)=d(7)+drdoa8;
         da(8)=d(8)+drdoa9;

%Reset range differences
         d=da;

%Find RDOAs from geometry
         db(1) = norm(source1 - RL(2,:)) - norm(source1 - RL(1,:));
         db(2) = norm(source1 - RL(3,:)) - norm(source1 - RL(1,:));
         db(3) = norm(source1 - RL(4,:)) - norm(source1 - RL(1,:));
         db(4) = norm(source1 - RL(5,:)) - norm(source1 - RL(1,:));
         db(5) = norm(source1 - RL(6,:)) - norm(source1 - RL(1,:));
         db(6) = norm(source1 - RL(7,:)) - norm(source1 - RL(1,:));
         db(7) = norm(source1 - RL(8,:)) - norm(source1 - RL(1,:));
         db(8) = norm(source1 - RL(9,:)) - norm(source1 - RL(1,:));

%Use Smith-Abel algorithm to obtain estimated position from RDOAs
%Apply 45 deg 3-dB beam width of receiver antennas for signal reception
%Weights of 1 define receivers used, weights of 0 define those not used
%Receivers 1a,1b,1p,10,11,12,14 and 15 within view at time start
         if ts<2.12
            w=[1;1;1;1;1;1;1;0];
            du=[d(1);d(2);d(3);d(4);d(5);d(6);d(7)];
            dbn=[db(1);db(2);db(3);db(4);db(5);db(6);db(7)];
            source_est=smith_abel(RL,d,w,R);
         end;

%Lose Receiver 12 at 2.12 sec
         if ts>=2.12 & ts<3.29
            w=[1;1;1;1;0;1;1;0];
            du=[d(1);d(2);d(3);d(4);d(6);d(7)];
            dbn=[db(1);db(2);db(3);db(4);db(6);db(7)];
            source_est=smith_abel(RL,d,w,R);
         end;
```

```
%Lose Receiver 14 at 3.29 sec
         if ts>=3.29 & ts<4.49
            w=[1;1;1;1;0;0;1;0];
            du=[d(1);d(2);d(3);d(4);d(7)];
            dbn=[db(1);db(2);db(3);db(4);db(7)];
            source_est=smith_abel(RL,d,w,R);
         end;

%Obtain RDOA for added receiver from estimated position
         if ts==4.49
            d(8)=norm(source_est-RL(9,:))-norm(source_est-RL(1,:));
            d(8)=d(8)+drdoa9;
         end;

%Lose Receiver 15 and gain Receiver 16 at 4.49 sec
         if ts>=4.49 & ts<5.95
            w=[1;1;1;1;0;0;0;1];
            du=[d(1);d(2);d(3);d(4);d(8)];
            dbn=[db(1);db(2);db(3);db(4);db(8)];
            source_est=smith_abel(RL,d,w,R);
         end;

%Lose Receiver 11 at 5.95 sec
         if ts>=5.95 & ts<=Tmax
            w=[1;1;1;0;0;0;0;1];
            du=[d(1);d(2);d(3);d(8)];
            dbn=[db(1);db(2);db(3);db(8)];
            source_est=smith_abel(RL,d,w,R);
         end;

%Estimated RDOA error from used range differences
         errt=norm(du - dbn);

%Fill matrix for estimated RDOA error
         if ts==0;xet=errt;
         else xet=[xet errt];
         end;

%Estimated position error
         errp=norm(source1 - source_est);

%Fill matrix for estimated position error
         if ts==0;xep=errp;
         else xep=[xep errp];
         end

%Return to trajectory loop
      end;

%Calculate mean RDOA and position errors for missile trajectory
%Fill matrices for these values
      merrt=mean(xet);
      merrp=mean(xep);
      if count==1;meanxet=merrt;meanxep=merrp;
      else meanxet=[meanxet merrt];meanxep=[meanxep merrp];end
```

```
%Return to Monte Carlo loop
   end;

%Calculate and print mean values for Monte Carlo iterations
   mcet=mean(meanxet);
   mce=mean(meanxep);
   fprintf('SNR(dB) = %g\n', snrdb);
   fprintf('Error in RDOA (ft) = %g\n', mcet);
   fprintf('Error in Position (ft) = %g\n', mce);

%Fill SNR, RDOA, and position error matrices for plotting
   if snrdb==15;snrdbm=snrdb;mcetm=mcet;mcem=mce;
   else snrdbm=[snrdbm snrdb];mcetm=[mcetm mcet];mcem=[mcem mce];end;

%Return to SNR loop
end;

%Save data to workspace
save snrdata2 snrdbm mcetm mcem;

%Plot graphs
figure(1)
plot(snrdbm,mcetm);grid;
title('Range Difference Error vs. SNR');
xlabel('SNR - dB');ylabel('Range Difference Error - ft');

figure(2)
plot(snrdbm,mcem);grid;
title('Position Error vs. SNR');
xlabel('SNR - dB');ylabel('Position Error - ft');
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. RECEIVER AZIMUTH AND ELEVATION ANGLES

Figures B.1 – B.9 show the azimuth (theta) and elevation (phi) angles vs. time for each of the nine receivers on the White Sands Missile Range. These plots were obtained from the MATLAB program *angles.m* whose source code is listed in Appendix A.

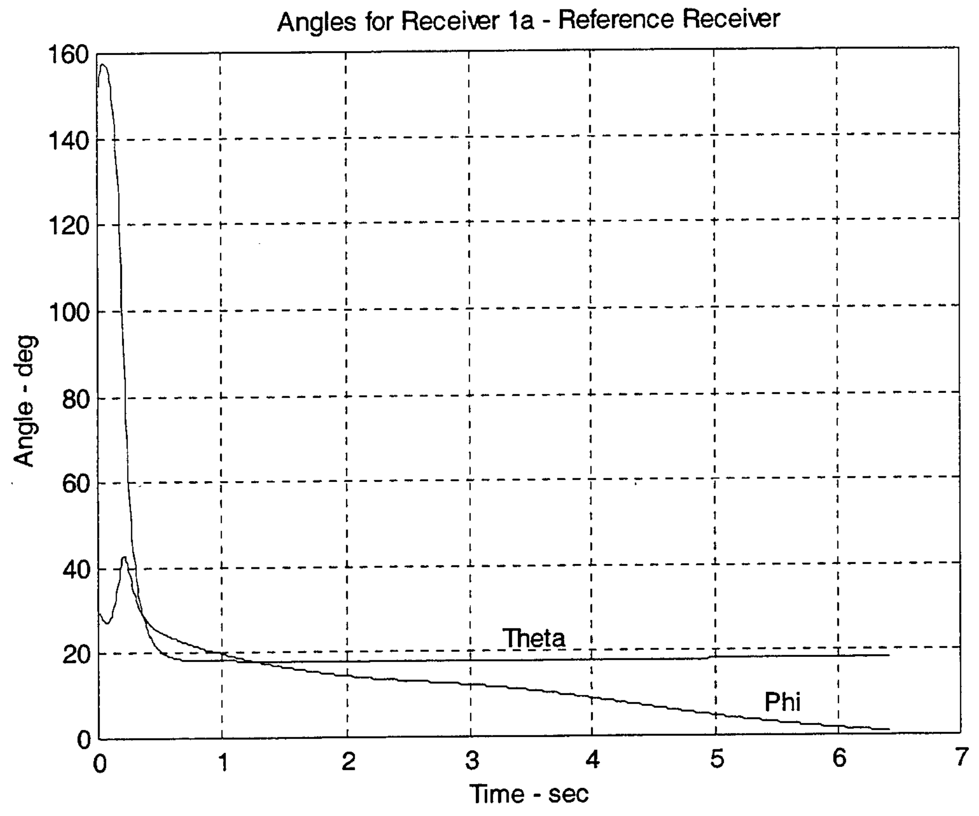

Figure B.1. Reference Receiver Azimuth and Elevation Angles

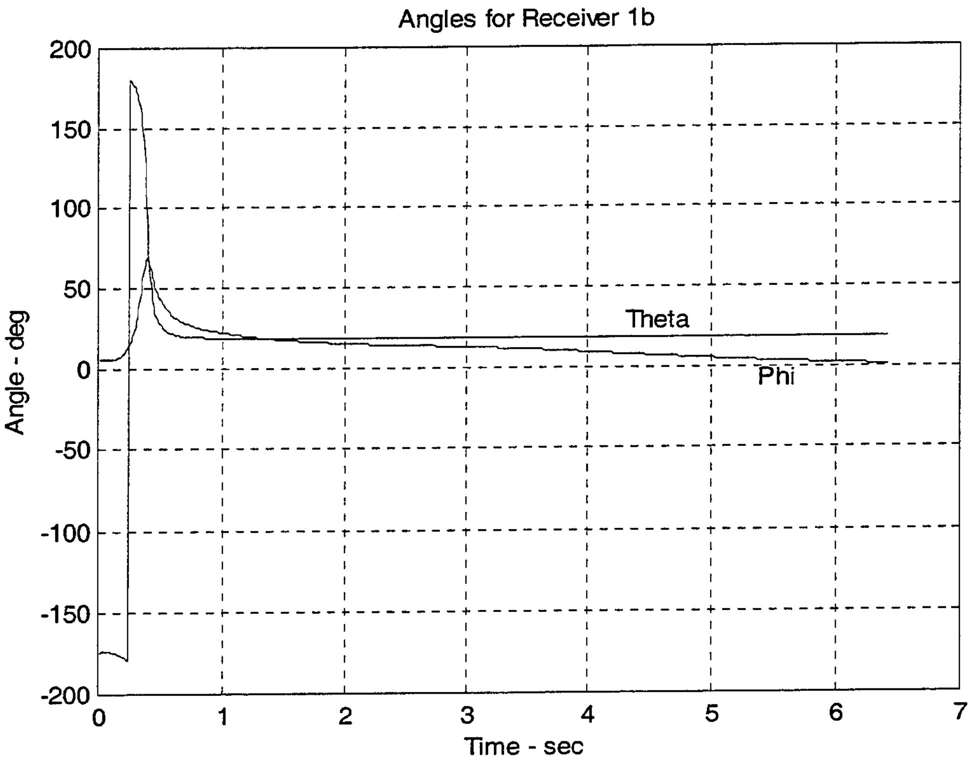

Figure B.2. Receiver 1b Azimuth and Elevation Angles

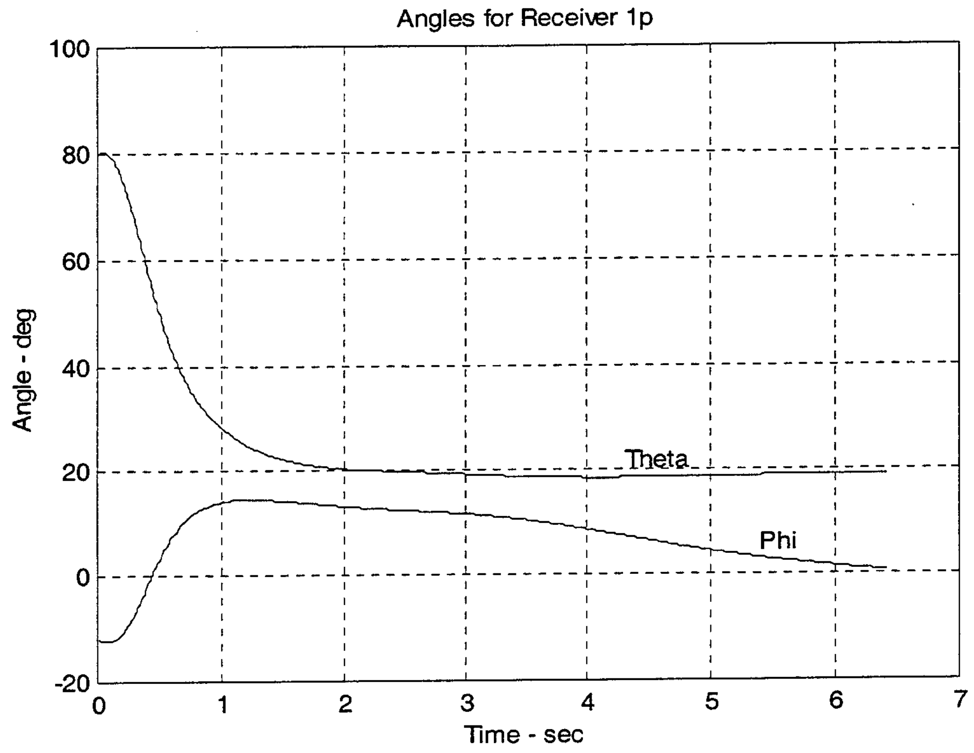

Figure B.3. Receiver 1p Azimuth and Elevation Angles

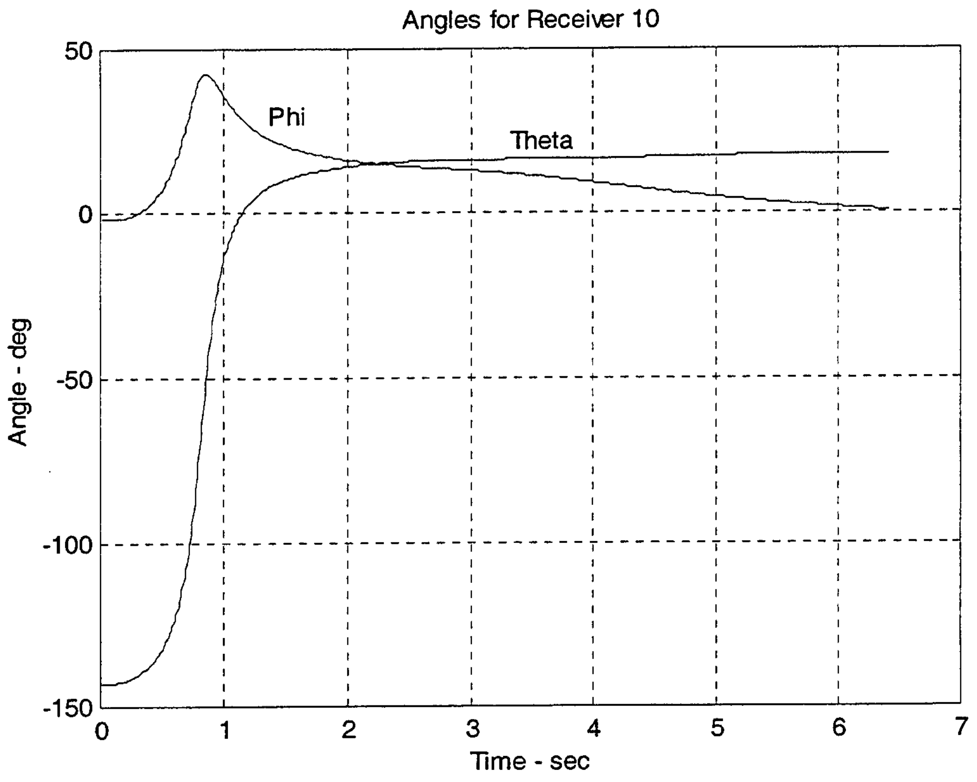

Figure B.4. Receiver 10 Azimuth and Elevation Angles

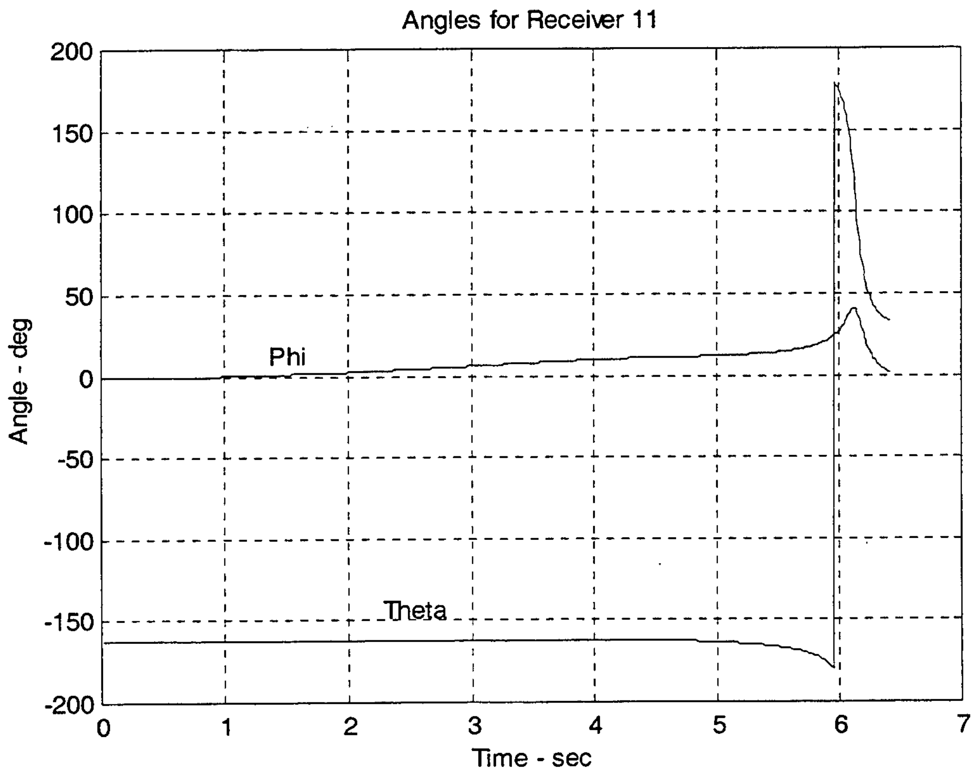

Figure B.5. Receiver 11 Azimuth and Elevation Angles

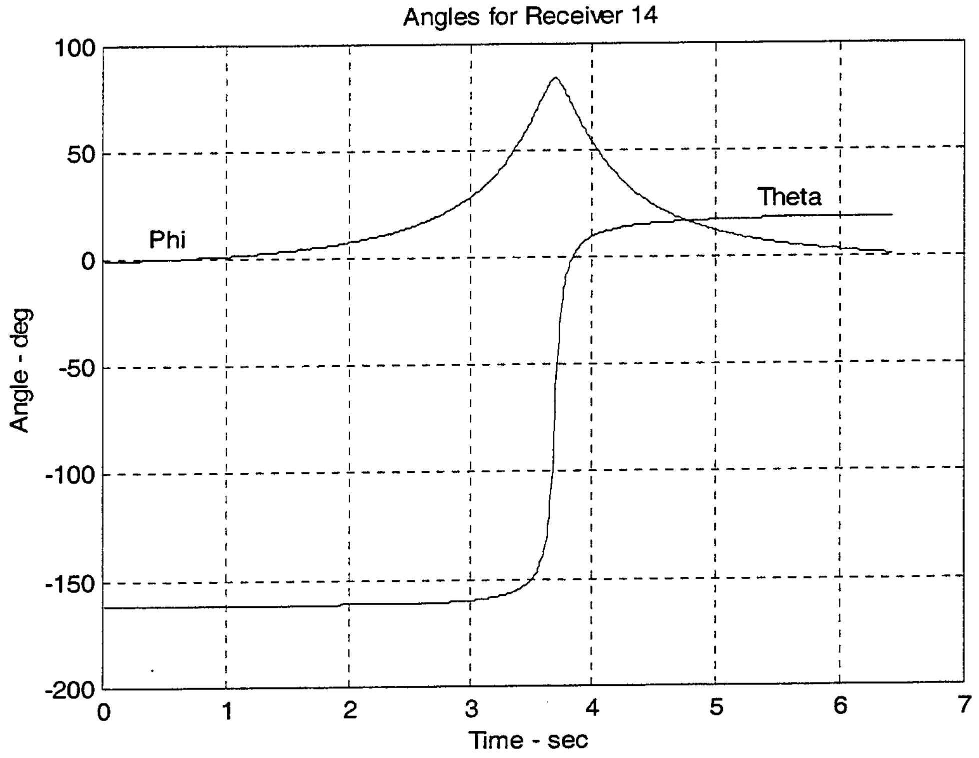

Figure B.6. Receiver 12 Azimuth and Elevation Angles

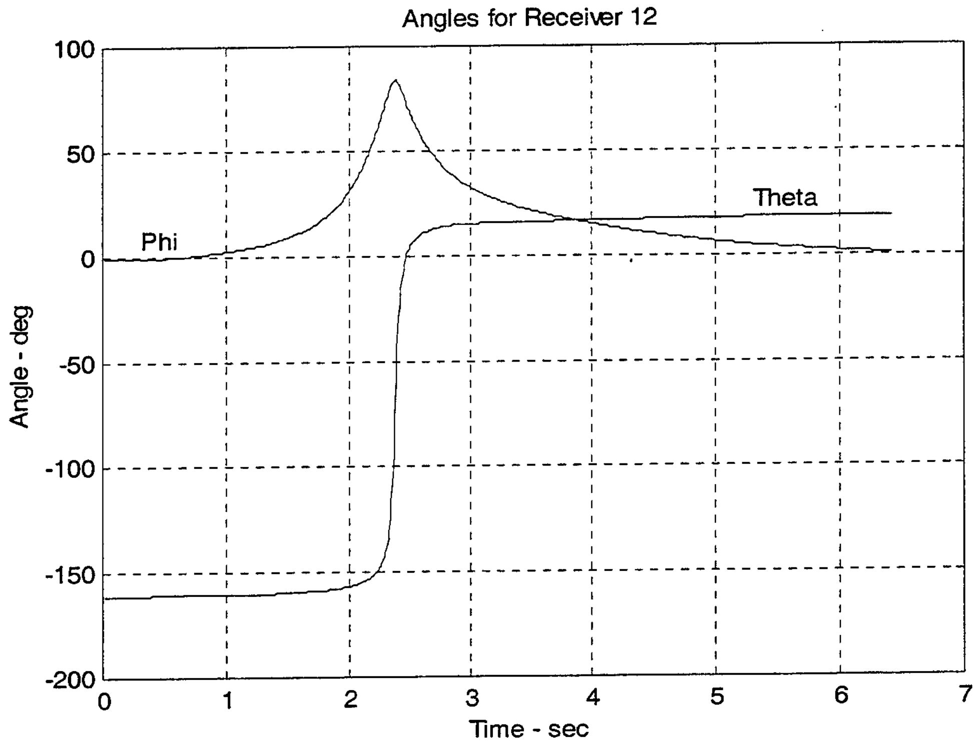

Figure B.7. Receiver 14 Azimuth and Elevation Angles

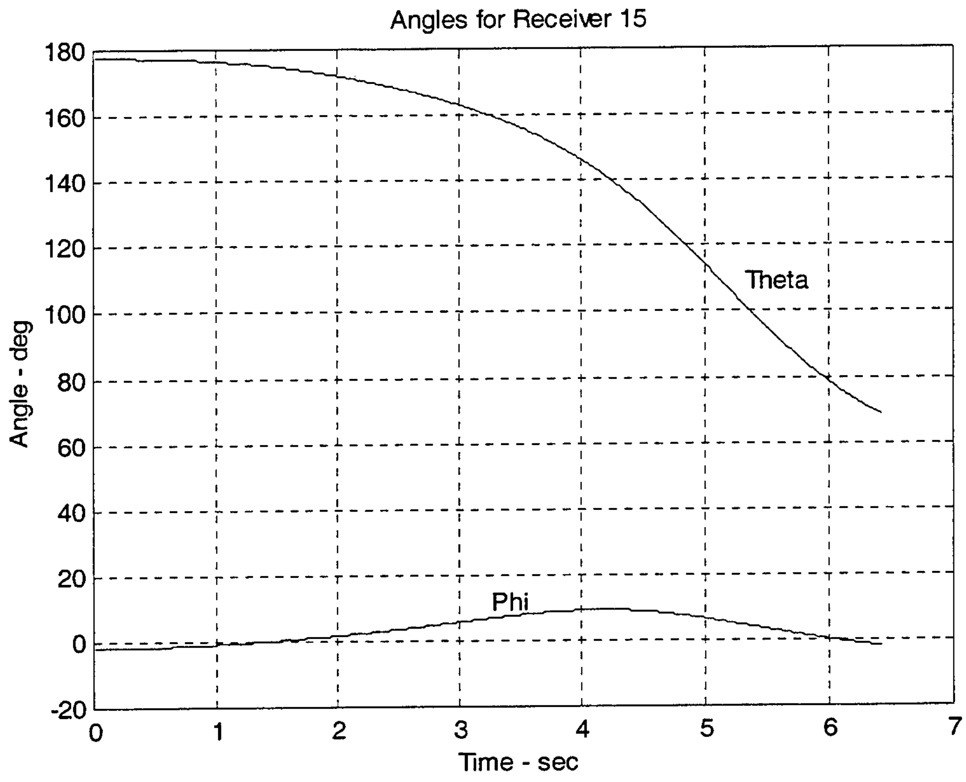

Figure B.8. Receiver 15 Azimuth and Elevation Angles

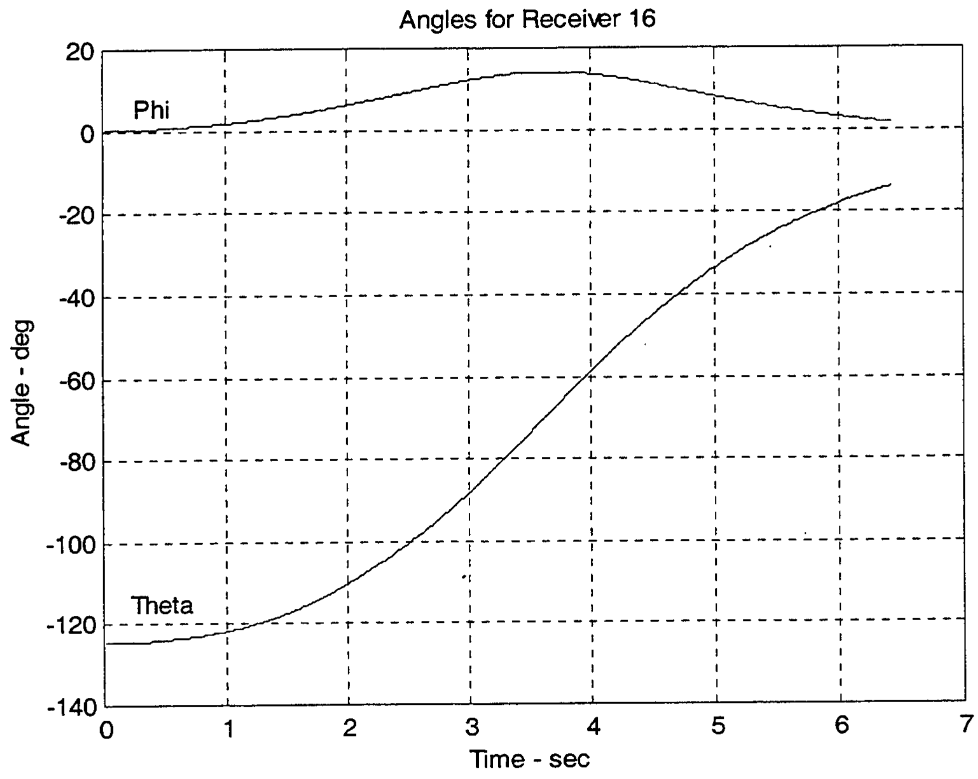

Figure B.9. Receiver 16 Azimuth and Elevation Angles

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

1. Naval Air Warfare Center – Weapons Division China Lake, CA, *TM Tracker Final Report*, Jan. 2000.

2. Fang, B.T., "Simple Solutions for Hyperbolic and Related Position Fixes," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 26, no. 5, pp. 748-753, Sep. 1990.

3. Abel, J.S. and Smith, J.O., "Closed-Form Least-Squares Source Location Estimation from Range-Difference Measurements," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-35, no. 12, pp.1661-1669, Dec. 1987.

4. Chan, Y.T. and Ho, K.C., "A Simple and Efficient Estimator for Hyperbolic Location," *IEEE Transactions on Signal Processing*, vol. 42, no. 8, pp. 1905-1915, Aug. 1994.

5. Chestnut, P.C., "Emitter Location Accuracy Using TDOA and Differential Doppler," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-18, no. 2, pp. 215-218, March 1982.

6. Schleher, D.C., *Electronic Warfare in the Information Age*, pp. 337-339, Artech House, Inc., Norwood, MA, 1999.

7. Schleher, D.C., "Preliminary Report on Positional Accuracy of TDOA Missile System," memorandum to Naval Air Warfare Center – Weapons Division China Lake, CA, 7 June 2000.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Rd., STE 0944
   Ft. Belvoir, Virginia 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, California 93943-5101

3. Professor Curtis D. Schleher 1
   Code IW/Sc
   Naval Postgraduate School
   Monterey, California 93943-5101

4. Professor David C. Jenn 1
   Code EC/Jn
   Naval Postgraduate School
   Monterey, California 93943-5101

5. Professor Dan C. Boger 1
   Chairman, Code IW
   Naval Postgraduate School
   Monterey, California 93943-5101

6. Commanding Officer 2
   Code 473400D Bldg. 31440
   Attn. Mr. Larry Rollingson / Mr. Greg Velicer
   Naval Air Weapons Center
   China Lake, California 93555

7. Robert A. Klaszky 3
   225 Worden St.
   Portsmouth, Rhode Island 02871