Design of a Robust Acoustic Positioning System for an Underwater Nuclear Reactor Vessel Inspection Robot

by

Allen B. Maples

Thesis submitted to the faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Mechanical Engineering

Approved:

C.F. Reinholtz, Chairman

A. L. Abbott

H. H. Robertshaw

May 17, 1993

Blacksburg, Virginia

LD
5655
V855
1993
M275
c.2

c.2

Design of a Robust Acoustic Positioning System for an Underwater Nuclear Reactor Vessel Inspection Robot

Abstract

The objective of this thesis is the algorithmic enhancement and initial evaluation of an underwater acoustic positioning system which is designed to determine the position and orientation of a mobile nuclear reactor vessel inspection robot.

Although a great deal of research has been done in the area of underwater acoustic positioning, this work differs from previous work in three significant ways. First, most applied acoustic positioning systems have been designed for the offshore oil drilling industry, and thus their requirements and restrictions are dictated by an oceanic environment. Second, most previous work has focused only upon acquiring the position of a point from the acoustic system. The inspection robot operation requires accurate positioning and orientation. Finally, the accuracy of acoustic positioning systems is generally dependent upon an evaluation of the speed of sound. However, this parameter is highly dependent upon water temperature. As will be discussed, the reactor vessel water temperature may not be uniform or constant, which makes the design of a precise positioning system difficult. Original methods to overcome this obstacle are discussed and evaluated. Also examined are configurations and constraints of the acoustic transceivers, the numerical solution procedures utilized, and the resulting errors associated with the developed methods.

Acknowledgments

There is quite a bit that I would like to thank Dr. Charles Reinholtz for, but this thesis is already too long. Suffice to say he is the reason I came to Virginia Tech, and much of the reason I enjoyed my stay. For additional accolades, one needs only ask anyone who has been fortunate enough to be associated with Charlie.

I didn't have the opportunity to work extensively with Dr. Harry Robertshaw, and so my appreciation is more for who he is than for what he does. However, I did learn a little digital control, and a little wrist control. I expect both will serve me in the future.

I would like to extend a special thanks to Dr. Lynn Abbott. He has bravely ventured far from his domain in order to serve on my committee, and this is greatly appreciated.

And then there is the group of students under Dr. Charles Reinholtz which I have had the chance to be a part of; definitely a most unique collection of individuals. It has been a very good two years. I would particularly like to thank Will Cordle for the comfy chair.

Finally, I would like to thank my father. None of this would have been possible without him, and my appreciation extends far beyond the obvious biological and financial restrictions of that statement. Thanks Dad.

Table of Contents

1. Introduction 1
    1.1 Motivation 1
    1.2 Approach 3
    1.3 Literature Review 3
2. The Reactor Vessel Environment 7
    2.1 Physical Dimensions and Characteristics of Vessels 7
    2.2 Radiation 7
    2.3 Temperatures 7
3. The Generalized Positioning System 11
    3.1 Definitions of Coordinate Systems 11
    3.2 System Equations 13
    3.3 Transceiver Configurations 15
    3.4 Equivalent Velocity Approximation 16
    3.5 System Equations Incorporating Equivalent Velocity Approximation 17
    3.6 Solution Process 19
        3.6.1 Effect of Velocity Approximation 19
        3.6.2 Minimization Method 20
        3.6.3 Overdetermined Systems 21
4. The Applied Global Positioning Scheme 23
    4.1 URSULA Operation Characteristics 23
    4.2 The Marquest Transceivers 24
    4.3 Fixed Transceiver Placement 25
    4.4 Addition of Inclinometers 25
    4.5 Velocity Compensation System 26
    4.6 Vessel Temperature Models 31
        4.6.1 Linear Model 32
        4.6.2 Parabolic Model 32
5. Simulation of Positioning System 33
    5.1 System Parameters 33

5.1.1 Fixed Transceiver Configuration 33
5.1.2 Mobile Transceiver Configuration 33
5.1.2.1 Number 33
5.1.2.2 Spacing 33
5.1.2.3 Orientation 34
5.1.3 Depth 34
5.1.4 Temperature Distribution 35
5.1.5 Incorporated Errors 35
5.1.5.1 Fixed Transceiver Positions 35
5.1.5.2 Mobile Transceiver Positions 35
5.1.5.3 Times of Flight 36
5.1.5.4 Inclinometers 36
5.1.6 Compensation Factor 36
5.1.7 Operating Radius 36
5.2 Simulation Results 36
5.2.1 Depth 38
5.2.2 Mobile Transceiver Array Orientation 38
5.2.3 Temperature Distribution 39
5.2.4 Error Incorporation 39
5.2.5 Compensation Factor 39
5.2.6 Transceiver Spacing 39
5.2.7 Operating Radius 39
5.2.8 Error Graphs 40
6. Future Work 55
6.1 Testing 55
6.2 System Alterations 55
6.2.1 Reduction of Fixed Transceivers 55
6.2.2 Harmonic Mean 55
6.2.3 System Combinations 57
6.3 Temperature Range Restrictions 57
6.4 Code Optimization 58
7. Conclusion 59
8. References 60

Appendix 1 : Error Trends and Results 62
Appendix 2 : Data File Conventions 65
Appendix 3 : GPS Data Files 67
Appendix 4a : GPS Code; Interpolated Velocity System 95
Appendix 4b : GPS Code; Equivalent Velocity System 117
Vita 129

List of Figures

Figure 1.1 URSULA in reactor vessel 2
Figure 1.2 Weld scan patterns 5
Figure 2.1 Speed of sound dependence on temperature 8
Figure 3.1 Coordinate systems 11
Figure 3.2 Euler rotations 12
Figure 3.3 Vector conventions 14
Figure 3.4 Configuration of 3 fixed and 3 mobile transceivers 16
Figure 3.5 Similarity of acoustic paths 17
Figure 4.1 Interpolated transmission velocity 26
Figure 4.2 Comparison of true and interpolated transmission velocities 28
Figure 4.3 Transceiver sing-around 29
Figure 5.1 Definition of pivot angular position 37
Figure 5.2 NAA(A,B,C,D,E)BA3.000 40
Figure 5.3 QAA(A,B,C,D,E)BA3i.000 40
Figure 5.4 NAADBA3.000 41
Figure 5.5 QAADBA3i.000 41
Figure 5.6 NAB(A,B,C,D,E)BA3.000 42
Figure 5.7 QAB(A,B,C,D,E)BA3i.000 42
Figure 5.8 NABDBA3.000 43
Figure 5.9 QABDBA3i.000 43
Figure 5.10 NAA(A,B,C,D,E)CA3.000 44
Figure 5.11 QAA(A,B,C,D,E)CA3i.000 44
Figure 5.12 NAADCA3.000 45
Figure 5.13 QAADCA3i.000 45
Figure 5.14 NAADDA3.000 46
Figure 5.15 QAADDA3i.000 46
Figure 5.17 NAADBB3.000 47
Figure 5.16 QAADBB3i.000 47
Figure 5.18 NAADBC3.000 48

Figure 5.19 QAADBC3i.000 48
Figure 5.20 NAADBD3.000 49
Figure 5.21 QAADBD3i.000 49
Figure 5.22 QAADBE3i.000 50
Figure 5.23 QAADBA(1,3)i.000 50
Figure 5.24 NAADBA(0,1,2,3,4).000 51
Figure 5.25 QAADBA(0,1,2,3,4)i.000 51
Figure 5.26 NBADBB3.000 52
Figure 5.27 QBADBB3i.000 52
Figure 5.28 NACDBB3.000 53
Figure 5.29 QACDBB3i.000 53
Figure 5.30 NAADBA3.rd1 54
Figure 5.31 QAADBA3i.rd1 54
Figure 6.1 Harmonic mean; velocities between transceivers 56

List of Tables

Table 3.1 Transceiver combinations for basic system 18
Table 4.1 Transceiver combinations with the use of inclinometers 26
Table 4.2 Transceiver combinations with the use of interpolated velocities 27
Table 4.3 Transceiver combinations with the use of interpolated velocities and inclinometers 27

1. Introduction

1.1 Motivation

The work presented here is in response to the need for an underwater positioning system for use with a nuclear reactor vessel inspection robot. Specifically, a mobile inspection robot was designed by the Robotics and Mechanisms Group of Virginia Polytechnic Institute and State University in conjunction with the Special Products and Inspection Services (SPIS) Division of Babcock and Wilcox Nuclear Technologies for use in reactor vessel inspections. This robot was designed to eventually replace the ARIS system, which is currently performing the reactor vessel inspections.

The ARIS (Automated Reactor Inspection System) and ARIS II manipulator are based upon a polar crane design which has a four degree-of-freedom, all-revolute, serial arm which is part of a deployable boom assembly. Although ARIS has successfully completed many past inspections, its size, weight, and complexity make installation and removal of the manipulator very difficult and time consuming. It requires several tractor trailers to transport it to site where a 20-ton crane must be used to assemble the manipulator's total 16 ton bulk. The process of assembly and calibration takes approximately three days before vessel inspections may begin. Disassembly and loading for return transport requires about the same amount of time. Since the cost incurred by the utilities due to inspection is measured in lost reactor operational time, reducing the overall inspection period is highly desirable.

URSULA (Underwater Reactor Scanner Un-Like Aris) is a design concept which greatly reduces total inspection time by decreasing the amount of assembly and disassembly required on-site. Transport requirements and personnel are similarly reduced. The URSULA design (Fig. 1.1) is based upon a mobile platform bearing a six-degree-of-freedom, all-revolute, serial manipulator arm. The platform sits atop a number of active suction pads which hold the mobile robot fixed to the vessel walls, while the manipulator arm is fitted with an ultrasonic transducer array which is used to scan the vessel welds. Upon completion of the scan pattern within the manipulator workspace, the manipulator replaces the ultrasonic head with an additional suction pad from a tool rack on the platform. This suction pad is then fixed to the vessel wall and the base pads release their hold. The manipulator arm is then used to move the platform to a new location, where the

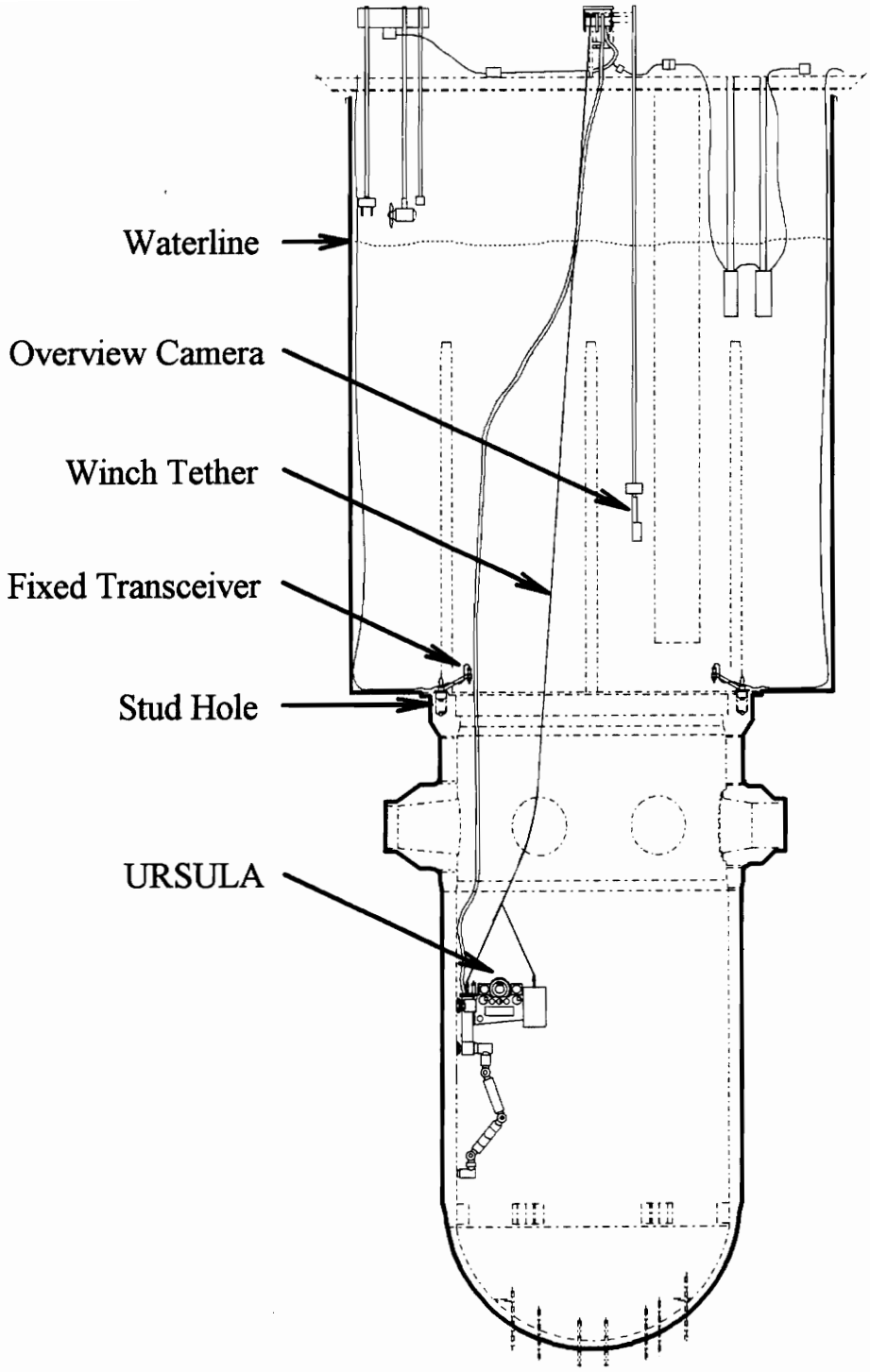


Figure 1.1 URSULA in reactor vessel.

base pads reestablish their suction. It then replaces its suction pad with the appropriate scanning tool, and the process begins anew. In this manner the arm is used for both locomotion and inspection.

The objective of the inspections is to ultrasonically scan the reactor vessel welds and evaluate their condition. It is thus necessary to accurately and consistently locate the scanning tools. Since the ARIS platform is fixed to the reactor vessel, the end-effector may be located relative to reactor vessel coordinates by applying forward and inverse kinematics to the manipulator arm. The URSULA design requires an additional step. While the forward and inverse kinematics provide a relationship between the end-effector and the mobile robot, the relationship between the mobile robot and the reactor vessel varies, and must be obtained by other means.

In its simplest form, the objective of this work is to provide the transform which links URSULA to the reactor vessel. The process by which this is accomplished can not be as succinctly stated.

1.2 Approach

The goal of this positioning system is to determine the location and orientation of the mobile robot as it maneuvers about the interior walls of the reactor vessel. The basic solution to this problem is to attach a number of ultrasonic transceivers to known locations in the reactor vessel, and to attach additional transceivers to positions on the mobile robot which are known accurately relative to the robot itself. Each of these transceivers will then send and receive acoustic signals to and from one another, and the resulting times of flight will be used to determine the three dimensional position and orientation of the underwater robot.

1.3 Literature Review

Underwater acoustic location and positioning encompasses a vast amount of applications. Everything from subsea military defense to offshore drilling work to undersea exploration has found a place for acoustics. The reason sound has been so widespread a medium for the transmission of environmental information is that water is an extremely good conductor of acoustic signals, while it is an extremely poor conductor of the more conventional electromagnetic signals. The maximum underwater range for an acoustic

signal depends upon the transmission frequency and is on the order of 5×10^4 wavelengths. In contrast, electromagnetic transmissions such as radio waves are attenuated to unusable levels after a distance of only 100 wavelengths (McCartney, 1980). As a result, most underwater positioning work has relied exclusively upon acoustics.

The most significant driving force behind the development of underwater positioning systems has been the offshore drilling industry. As the oil rigs are built increasingly larger in deeper waters the need for more precise construction techniques has provided a natural avenue for the use of acoustics. Acoustics have been used for accurate positioning of drilling templates on the ocean floor, location of remotely operated vehicles (ROVs) for underwater platform inspection, positioning of deep sea cable laying machines, and a myriad of additional applications.

However, for the most part, these systems have been designed to operate at relatively large distances for which good accuracy is measured in feet, and even this result is frequently misrepresented as an absolute accuracy when in fact it is often only a repeatability factor. It is desired that the reactor vessel positioning system for URSULA be absolutely accurate to within a few centimeters. Although a specific, set boundary for the magnitudes of acceptable inaccuracies has not been established, positioning errors will essentially amount to lost dollars. Since the purpose of the mobile robot is to scan the reactor vessel welds, its operation must ensure that the welds have been completely covered. Therefore if the positioning system can only guarantee 5 cm accuracies, then a 5 cm buffer zone must be added to the planned scanning area in order to be certain that the weld scan has been completed. This would require additional scanning time and thus an extended inspection period, resulting in further reactor down time. Additionally, the welds are scanned in a raster pattern which runs both perpendicular and parallel to the weld (Fig. 1.2). The orientation of the robot must be determined to within a few degrees in order to accurately scan in this manner.

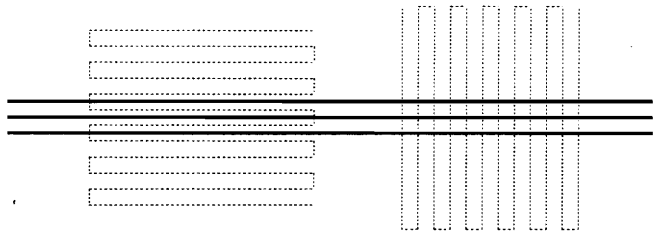


Figure 1.2 Scan patterns on reactor vessel meridian weld. Each pattern type must be performed twice over the same weld patch with the transducer head rotated 180 degrees.

Another significant difference between previous work in acoustic positioning and the work presented here concerns the transmission velocity of the acoustic signal. Almost exclusively, this parameter has first been determined before positioning work could commence. As will be discussed, this may not be possible for the reactor vessel environment.

Also, most acoustic positioning systems only address the location of an underwater object and disregard (or obtain by other means) the object's orientation, although there are exceptions. For example, Lee, Lee and Hong [1992] conducted an experimental study for the position control of an underwater ROV utilizing a system of acoustic transceivers which determined the heading as well as the position of the vehicle.

It must be noted that the reactor vessel environment is extremely specific, and the operation of remote vehicles within this system is relatively new. Thus, there is limited previous work which may be used as a resource for the development of an accurate positioning system for this application. One source of information comes from General Electric Nuclear Energy[1]. They have developed a small ROV for visual inspection of boiling water reactors (BWRs) and use an acoustic transceiver array to determine the position and heading of the vehicle. They claim positioning accuracies on the order of a few centimeters. However, GE uses a more traditional approach to evaluate the speed of sound. The robot is sent to a known location where signal time of flight information is acquired. The known distances and returned times are then used to calculate a speed of sound. The robot is then constrained to operate in that local area, and the process must be repeated periodically. Although this method seems to work for GE, repeatedly moving URSULA to a specified location is not desirable since it is not as mobile as the GE swimming robot and would consume unacceptable inspection time in the process.

[1] Information obtained from conversations with David C. Vickerman (April 1993) of GE Nuclear Energy.

Mitsubishi Heavy Industries has taken a different approach to the positioning of their nuclear reactor vessel ultrasonic scanning robot which moves about the interior surface of the vessel[2]. They have utilized a system of three laser trackers to determine the robot location and rely upon operation constraints and inclinometer data to obtain the orientation. Although they claim positioning accuracies on the order of a few millimeters, their system is extremely expensive, and complicated to operate and maintain.

[2] Information obtained from communications with John Bares and Takeo Oomichi (February 1992) of Mitsubishi Heavy Industries, Ltd., Takasago Research & Development Center.

2. The Reactor Vessel Environment

2.1 Physical Dimensions and Characteristics of Vessels

Each nuclear facility employs reactor vessels with slightly different dimensions. The distance from the core bottom to the stud hole flange is usually around 10 m, and the diameter of this area ranges from 5 to 6 m. The overall water depth varies around 20 m. The overall layout of the vessel is best illustrated by Fig. 1.1.

2.2 Radiation[3]

In the pressurized water reactors which BWNT is responsible for inspecting, the expected radiation exposure is dependent upon the position within the vessel. On the flange where the fixed transceivers will be located (as will be discussed), less than 1 R is expected. In the lower part of the reactor vessel where the fuel assembly is normally located the exposure increases. As high as 50 R may be found directly against the walls is this area, but due to the high absorption of the water this figure drops to less than 10 R only a few feet from the wall. Although GE has experienced some difficulties with their acoustic transceivers due to radiation exposure, it is expected that the lower radiation of the pressurized water reactors (PWRs) for which BWNT is responsible, and the fact that the mobile transceiver array will not be in direct contact with the vessel walls will prevent similar occurrences with the BWNT system. However, this remains to be seen.

2.3 Temperatures

Very little is known about the specific temperature distribution of the water in the reactor vessel. The vessel water may be a relatively uniform temperature, or there may be significant temperature gradients. It is possible that the water temperature varies by vertical striations, much as the temperature of the ocean. On the other hand, heat radiating from the vessel walls may produce a more radially dependent temperature profile. Convection may be playing a significant role, as may the intrusion and movement of the mobile robot. Inlet and outlet nozzle pumps may be in operation, along with the pumps associated with the operation of the robot itself which may create temperature

[3] Information supplied by S. W. Glass (May 1993) of Babcock & Wilcox Nuclear Technologies, Special Products and Inspection Services division.

currents. Most likely, all of these factors will affect the temperature characteristics of the reactor vessel water in a manner that is neither known, nor easily anticipated.

Unfortunately, the velocity of sound in water is highly dependent upon the temperature. Fig. 2.1 illustrates this dependency, as well as the effect of the water salinity level which is made only as a point of comparison. It is expected that the water in the reactor vessel will be homogeneous with respect to salinity, and only unpredictable gradients in the water properties present problems for the positioning system. The reason for this will soon become clear.

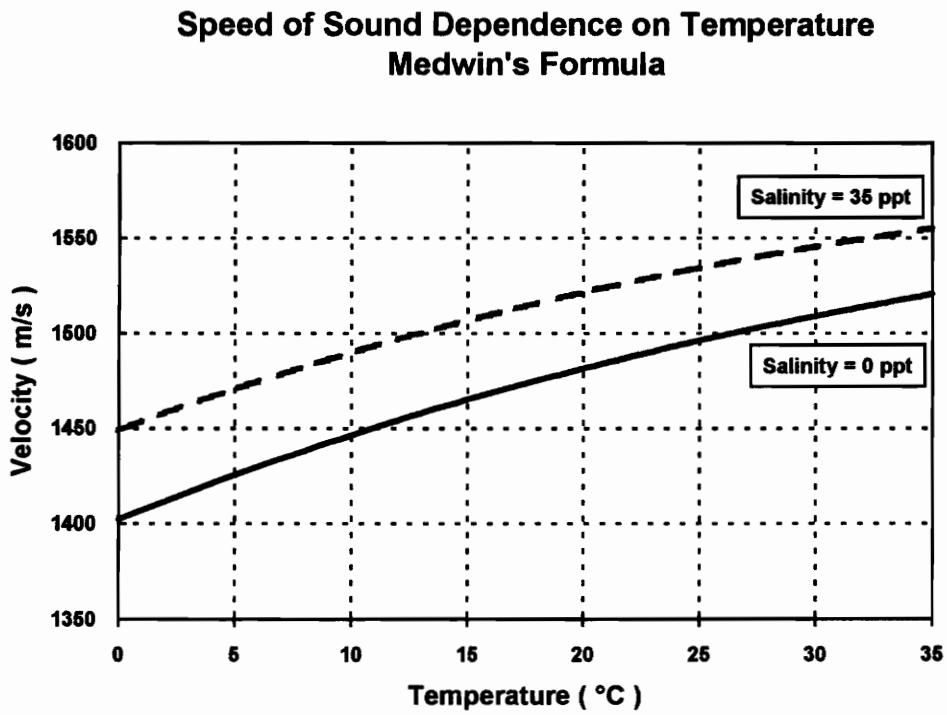


Figure 2.1 Speed of sound dependence on water temperature and effects of salinity levels at 35 parts per thousand (sea water) and 0 parts per thousand (freshwater).

This graph is based upon an empirical equation known as Medwin's formula (Eq. 2.1) which shows the dependence of the speed of sound in water (v) upon temperature (T), salinity (s), and depth (d) covering the ranges $0 < T < 35°C$, $0 < s < 45$ parts per thousand, $0 < d < 1000m$. The effects of pressure differences due to variations in depth over the operating range of the robot ($\approx$10m) is relatively insignificant.

$$v = 1449.2 + 4.6T - 0.055T^2 + 0.00029T^3 + (1.34 - 0.010T)(s - 35) + 0.016d \qquad \textbf{(2.1)}$$

It must be understood that the velocity of Eq. 2.1 is localized; it is the speed of sound at a point. As the transceiver signal travels through the water, it may experience areas of higher and lower temperatures, speeding up and slowing down accordingly. Additionally, the signal path will bend while traversing areas of different temperatures much as light is refracted at the boundaries of mediums with differing propagation velocities. Thus, the velocity of sound between two transceivers is not necessarily constant, nor does the acoustic signal necessarily follow a straight-line path. The speed of sound will only be consistent if the vessel water is uniform in temperature and composition.

Although no detailed information pertaining to the thermal distribution of the vessel water is available in the literature, the experience of personnel familiar with reactor vessel inspections (BWNT, General Electric Nuclear Energy, Nuclear Regulatory Commission) may be used to estimate the bulk mean temperatures which may be expected. During inspections, it is believed that the vessel water temperature will generally range from 60° C (140° F) to 38° C (100° F), decreasing over the period of reactor down time. Additionally, it is generally believed that the water will be a relatively uniform temperature, with the exception of the area directly above the core of BWRs. This condition would be ideal since the speed of sound could be measured at any convenient location and be applied throughout the vessel allowing a more simplistic triangulation scheme. However, there is no definitive data supporting these beliefs. Therefore the design of this positioning system is based upon a worst case assumption that there may be significant temperature gradients.

Although the velocity of the acoustic signal may vary as it travels from one transceiver to another, it is possible to define a velocity which may be treated as constant. Eq. 2.2 simply defines the *experienced velocity* between two transceivers i and j which are separated by a distance d_{ij} and a transmission time of flight t_{ij}. Unless otherwise indicated, all velocities in this discussion are experienced velocities, as opposed to localized point velocities. Experienced velocities will also be referred to as *transmission velocities*.

$$v_{ij,experienced} = \frac{d_{ij}}{t_{ij}} \tag{2.2}$$

Perhaps the best way to illustrate the effects of the speed of sound dependency on temperature is with an example. If two transceivers were separated by 10 m in water at a

uniform 30°C, the resulting velocity of sound would be approximately 1510 m/s. However, a miscalculation of only 1°C would cause a 2.4 m/s error in the transmission velocity value, resulting in approximately a 1.6 cm positioning error. Note that this error is only in one dimension; along the path between the transceivers. The errors associated with the full three-dimensional positioning system will be a product of many pathways, but this example does serve to demonstrate the sensitivity of the positioning system to the temperature variations which may be experienced. It can be seen that measuring the temperature or calculating the velocity at one point within the reactor vessel will not be sufficient if there are temperature differences in the water.

3. The Generalized Positioning System

3.1 Definitions of Coordinate Systems

The position and orientation of the mobile robot may be defined by relating a local coordinate system which is fixed to the robot, to a *global* coordinate system which is fixed to the reactor vessel. In three-dimensional space, this requires the definition of six variables. These values will be described by a vector $\vec{P}$ (which has three components) to define the global position of the local frame, and three rotation angles θ, φ, μ to define the orientation of the local frame relative to the global frame. The global system will be characterized by the coordinates *xyz* and the vector superscript *G*, while the local robot system will be characterized by the coordinates *qrs* and the vector superscript *R* . Note that in this work, *global* is related to the reactor vessel as opposed to the terrestrial reference of the *global* satellite positioning system.

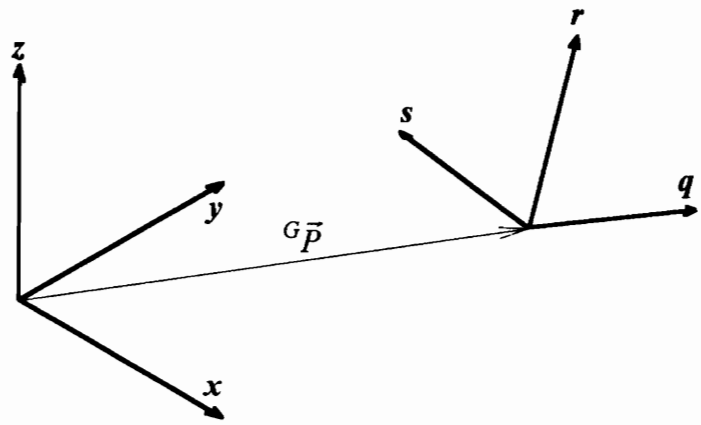
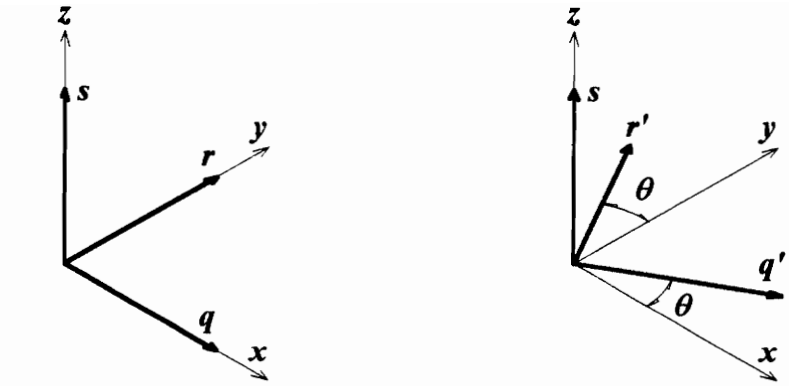


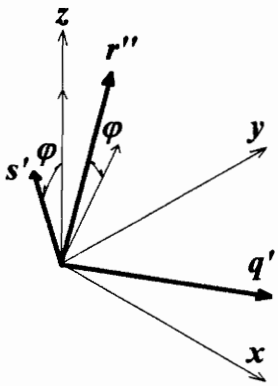
Figure 3.1 Conventions and relationship of the global and local coordinate systems.

Although the position of the robot as defined by $^G\vec{P}$ may be clear, the orientation of the local frame in relationship to the global system requires further discussion. This relationship may be understood by initially envisioning the *qrs* frame coincident with the *xyz* frame (Fig. 3.2a), then rotating it such that it could match any arbitrary orientation. This is done by first rotating the local frame about its *s*-axis (which is equivalent to a rotation about the global *z*-axis at this point) by angle θ (Fig. 3.2b). This results in a new orientation of the local frame which is designated $q'r's$. The local frame is next rotated an angle φ about q' which results in the next orientation $q'r''s'$ (Fig. 3.2c). The final rotation

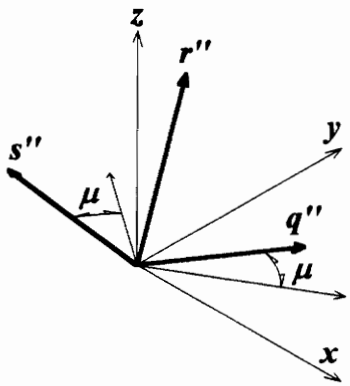


(a) Initially coincident axis.

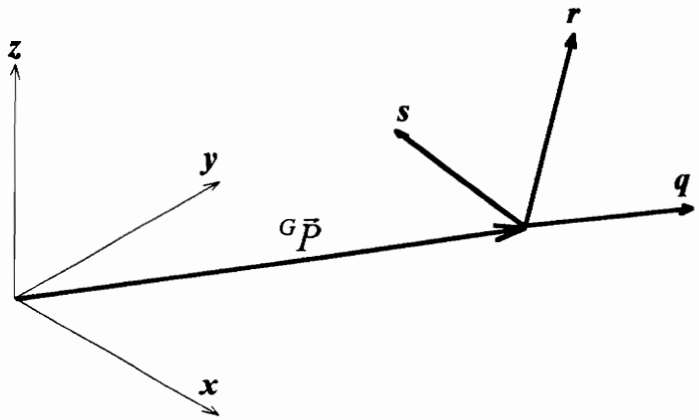
(b) Theta rotation about s.



(c) Phi rotation about q'.



(d) Mu rotation about r".



(e) Translation of oriented system.

Figure 3.2 Euler rotations of mobile robot coordinate frame.

is about r'' by angle μ (Fig. 3.2d). By defining the values of these rotation angles, the transceiver plane may be oriented in any manner.

Now, any point on the mobile robot (${}^{R}\vec{M}$) may be described in the global coordinate system (${}^{G}\vec{M}$) by defining the values of ${}^{G}\vec{P}$ and the rotation angles θ, φ, μ as in Eqs. 3.1 and 3.2.

$$ {}^{G}\vec{M} = \boldsymbol{R}_{\theta\varphi\mu}\,{}^{R}\vec{M} + {}^{G}\vec{P} = \boldsymbol{R}_{\theta}\boldsymbol{R}_{\varphi}\boldsymbol{R}_{\mu}\,{}^{R}\vec{M} + {}^{G}\vec{P} \tag{3.1} $$

$$ \boldsymbol{R}_{\theta} \equiv \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{R}_{\varphi} \equiv \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi & \cos\varphi \end{bmatrix}, \quad \boldsymbol{R}_{\mu} \equiv \begin{bmatrix} \cos\mu & 0 & \sin\mu \\ 0 & 1 & 0 \\ -\sin\mu & 0 & \cos\mu \end{bmatrix} \tag{3.2} $$

Note that Euler rotations (rotations about the moving frame) were used instead of fixed rotations (rotations about the fixed frame). Both methods will uniquely define the orientation of the local frame relative to the global system, but the Euler rotations are better suited to the actual implementation of the global positioning system. The reason for this will be explained.

3.2 System Equations

The acoustic transceivers may be divided into two categories, fixed and mobile. These transceivers will be designated F and M respectively with individual transceivers numbered sequentially beginning with zero. Although this numbering system may seem unconventional and even awkward at times, it corresponds to the zero offset labeling system used by the C programming language, and is adopted in order to more easily relate the written analysis to the computer code which accompanies this work.

The basic plan for the global positioning system is to fix m acoustic transceivers at accurately known locations in the reactor vessel, and to attach n more to the robot at locations known accurately in the local robot coordinate frame. The times of flight between each of the fixed and mobile transceivers may then be utilized to determine the global position and orientation of the robot. This is done by solving a set of nonlinear system equations generated by equating the global vector differences between the fixed and mobile transceivers with the distance d between them. This distance d is simply the product of the experienced velocity of sound and the time of flight between those transceivers. That is, the number of system equations will be $m \times n$. An arbitrary pair of

transceivers and their respective vector representations are illustrated in Fig. 3.3, and the resulting form of the system equations is given by Eq. 3.3 and 3.4.

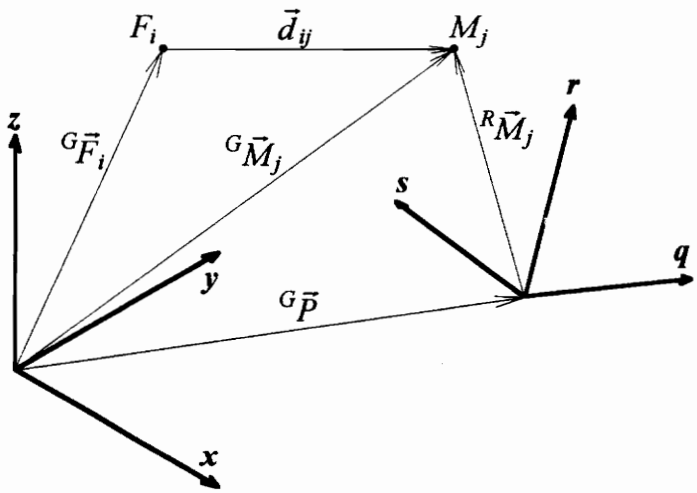


Figure 3.3 Vector conventions and arrangement for an arbitrary pair of transceivers.

$$\left|{}^G\vec{F}_i - {}^G\vec{M}_j\right| = d_{ij} \tag{3.3}$$

or

$$\left[\left(F_{ix} - M_{jx}\right)^2 + \left(F_{iy} - M_{jy}\right)^2 + \left(F_{iz} - M_{jz}\right)^2\right]^{1/2} = v_{ij}t_{ij} \tag{3.4}$$

There will be one equation in the above form for each F-M pairing. Note that the distance d, velocity v, and time t are each subscripted to indicate the fixed and mobile transceivers with which they are associated. The first subscript refers to the fixed transceiver, while the second subscript refers to the mobile transceiver. This convention is adopted throughout this work unless otherwise noted.

Each of the system equations is comprised of known and unknown values. The global positions of the m fixed transceivers (${}^G\vec{F}_i$) must be defined and known, while the times of flight (t_{ij}) are provided by the acoustic transceiver system. Thus, the unknown variables of Eq. 3.4 are the velocities (v_{ij}) and the global positions of the n mobile transceivers (${}^GM_j$). As in Eq. 3.1, these global positions are defined by three unknown rotation angles θ, φ, μ and the undefined vector ${}^G\vec{P}$ in addition to the known relative positions of the mobile transceivers.

$$ {}^{G}\bar{M}_j = \boldsymbol{R}_{\theta\varphi\mu}\,{}^{R}\bar{M}_j + {}^{G}\bar{P} \tag{3.5} $$

Hereafter, the origin of the local robot frame which is located by the vector ${}^{G}\bar{P}$ will also be described as the *pivot*.

3.3 Transceiver Configurations

The question now arises, how many fixed transceivers and how many mobile transceivers are required to determine the position and orientation of the mobile robot? Without immediately turning to the system equations for the answer, assume for the time being that the velocity of sound between any two transceivers is known, or can be determined. When coupled with the times of flight, the distances between each of the fixed and mobile transceivers would then be known.

With one transceiver fixed and a second mobile transceiver a known distance away, a spherical solution space for the three-dimensional position of the mobile transceiver is generated. This can be visualized by treating the distance between the transceivers as a rigid link of known length which is fixed to the global coordinate frame by a spherical joint. The resulting solution space is simply the set of all possible positions of the free end which is the surface of a sphere. This analogy may be extended to examine the results of utilizing multiple fixed and mobile transceivers. For instance, the solution space for a configuration of two fixed transceivers and one mobile is simply the intersection of the two resulting spheres. This intersection is generally a circle, although it can degrade to a single point if the mobile transceiver lies along the axis between the two fixed transceivers. Adding a third fixed transceiver results in a solution space of only two points, one on each side of the plane defined by the fixed array. If the workspace can be defined as existing on a specified side of this plane, then one of the solutions may be eliminated. Thus, the global position of a mobile transceiver (and thus a point on the robot) may be determined by a minimum of one mobile and three fixed transceivers.

The problem of defining the orientation of the mobile robot in addition to its position may be related to the problem of uniquely describing a plane relative to the global coordinate system. If the plane is envisioned as attached to the mobile platform, then by defining three points in the plane which are related to the robot in a known manner, both the position and orientation of the robot may be determined. If three mobile transceivers were attached to the robot at positions defined in its local coordinate frame, then the global

positions of each of those points could be determined by a fixed array of three transceivers (Fig. 3.4) thus defining the robot position and orientation. However, this result is based upon the condition of determinable velocities which in reality is not likely the case. Much of the work presented here addresses this problem and investigates possible methods to circumvent this situation.

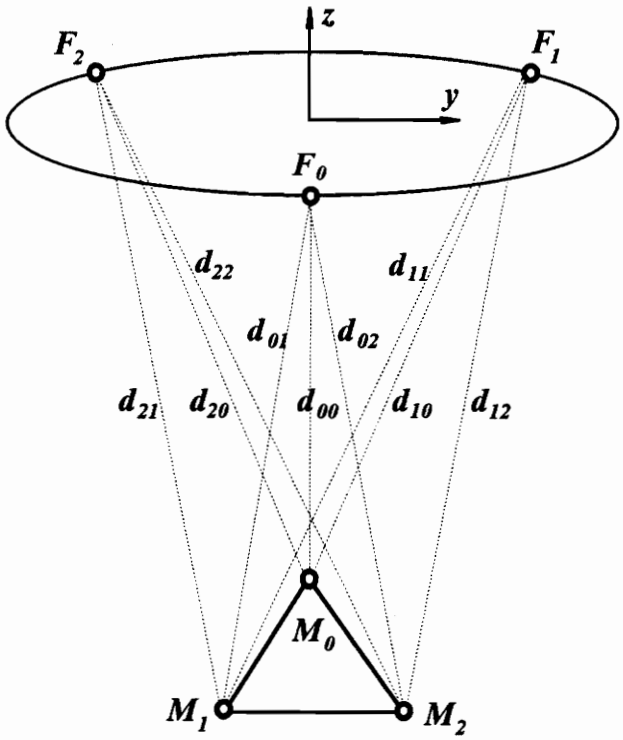


Figure 3.4 Configuration of three fixed transceivers and three mobile transceivers.

3.4 Equivalent Velocity Approximation

The system equations for m fixed transceivers and n mobile transceivers are represented by Eq. 3.6.

$$\left| {}^{G}\vec{F}_i - {}^{G}\vec{M}_j \right| = v_{ij} t_{ij} \quad \{i\colon 0 \ldots m-1\,;\, j\colon 0 \ldots n-1\} \tag{3.6}$$

As previously discussed, the three components of the pivot are unknown, as are the three rotation angles. Additionally, there are the unknown velocities $v_{00} \ldots v_{(m-1)(n-1)}$; one for each transmission path between the fixed and mobile transceivers. So for an arbitrary system configuration there are $m \times n$ system equations and $6 + m \times n$ unknowns. Even if the system equations were linear, the fact that there will always be more unknowns than

equations (regardless of the number of fixed or mobile transceivers) precludes the possibility of finding a solution. It is therefore necessary to simplify the system to reduce its dimensional degree (number of unknowns). In order to do this, it is assumed that the velocity of sound from any one fixed transceiver to any of the mobile transceivers will be approximately the same (Eq. 3.7).

$$v_{i0} \approx v_{i1} \approx \ldots \approx v_{i(n-1)} \tag{3.7}$$

As long as the distances from the fixed transceivers to the mobile platform are large compared to the separation between the mobile transceivers, this approximation should remain valid. When this is the case, the paths between a specific fixed transceiver and each of the mobile transceivers will be similar as in Fig. 3.5. Thus, the acoustic signals are traveling through similar conditions and the transmission velocities should therefore be approximately equivalent.

Figure 3.5 Similar acoustic paths between a fixed transceiver and three mobile transceivers.

3.5 System Equations Incorporating Equivalent Velocity Approximation

Eq. 3.8 incorporates the equivalent velocity approximation into the system equations. Note that now the velocities are only associated with each of the fixed transceivers, and are independent of the mobile transceivers. This reduces the total number of unknowns from $6+m\times n$ to $6+m$.

$$\left|{}^{G}\bar{F}_i - {}^{G}\bar{M}_j\right| = v_i t_{ij} \quad \{i\colon 0\ldots m-1\,;\, j\colon 0\ldots n-1\} \tag{3.8}$$

An examination of possible transceiver configurations is shown in Table 3.1. These results are obtained by applying the condition of Eq. 3.9. This restriction simply requires that the number of system equations must be greater than or equal to the number of unknowns. Note that only integer values have any real meaning, and thus the results of the last entry must be rounded up. However, these results are misleading. Only three

points are required to define a plane, and thus more than three transceivers in the fixed or mobile planes is redundant. Mathematically, the extraneous transceivers result in dependent system equations. This may be made physically clear by examining, for example, the second entry of Table 3.1 and applying the previous analogy of the path between each of the fixed and mobile transceivers being treated as a rigid link capped by spherical joints. The resulting structure would still be able to rotate about the axis between the two fixed transceivers so that the position of the mobile transceiver array could be anywhere on the resulting circle of solution space. This will occur whether there are four or four hundred mobile transceivers operating with the two fixed transceivers. Therefore Eq. 3.9 is a necessary but not sufficient condition. It must be coupled with the restriction that no more than three transceivers for m or n produce independent system equations. In this case, only the third entry of Table 3.1 is valid. It will be shown that redundant information (i.e. additional transceivers) is extremely useful, but the minimum requirements must first be met.

$$m \times n \geq 6 + m \tag{3.9}$$

Table 3.1 Transceiver combinations for basic system; $m \times n \geq 6 + m$ restriction.

Number of Fixed Transceivers m	Resulting Minimum Required Mobile Transceivers n
1	7
2	4
3	3
4	$2.5 \rightarrow 3$

Equations 3.10 to 3.18 are the expanded system equations for the configuration of three fixed and three mobile transceivers utilizing the equivalent velocity approximation. This is a system of nine equations and nine unknowns which means that a solution is now possible although not necessarily unique due to the nonlinearity of the equations.

$$\left| {}^{G}\bar{F}_0 - {}^{G}\bar{M}_0 \right| = v_0 t_{00} \tag{3.10}$$

$$\left| {}^{G}\bar{F}_0 - {}^{G}\bar{M}_1 \right| = v_0 t_{01} \tag{3.11}$$

$$\left|{}^{G}\vec{F}_0 - {}^{G}\vec{M}_2\right| = v_0 t_{02} \tag{3.12}$$

$$\left|{}^{G}\vec{F}_1 - {}^{G}\vec{M}_0\right| = v_1 t_{10} \tag{3.13}$$

$$\left|{}^{G}\vec{F}_1 - {}^{G}\vec{M}_1\right| = v_1 t_{11} \tag{3.14}$$

$$\left|{}^{G}\vec{F}_1 - {}^{G}\vec{M}_2\right| = v_1 t_{12} \tag{3.15}$$

$$\left|{}^{G}\vec{F}_2 - {}^{G}\vec{M}_0\right| = v_2 t_{20} \tag{3.16}$$

$$\left|{}^{G}\vec{F}_2 - {}^{G}\vec{M}_1\right| = v_2 t_{21} \tag{3.17}$$

$$\left|{}^{G}\vec{F}_2 - {}^{G}\vec{M}_2\right| = v_2 t_{22} \tag{3.18}$$

3.6 Solution Process

The system equations for an arbitrary transceiver configuration previously given by Eq. 3.8 may be restated in a more useful form as in Eq. 3.19. The terms in each of the previous system equations were squared, and the resulting arguments were brought to one side. The system was squared in order to simplify their derivatives which will be used later in the solution process. The reason behind bringing the terms to one side of the equations and equating to zero will soon become clear.

$$\Delta_{ij} - \left(v_i t_{ij}\right)^2 = 0 \quad \{i\text{: } 0 \ldots m-1 \text{ ; } j\text{: } 0 \ldots n-1\} \tag{3.19}$$

$$\textit{Where } \Delta_{ij} \equiv \left(\left|{}^{G}\vec{F}_i - {}^{G}\vec{M}_j\right| \right)^2$$

3.6.1 Effect of Velocity Approximation

Although the equivalent velocity approximation opens the door to a possible solution of the system unknowns, it must be recognized that now the equations may no longer be solved exactly. That is, there will be some error as a result of the simplification, and there may not be a set of variable values which exactly satisfy the system equations. This condition is represented by Eq. 3.20 which augments Eq. 3.19 to include an error factor e_{ij}.

$$\Delta_{ij} - \left(v_i t_{ij}\right)^2 = e_{ij} \quad \{i\text{: } 0 \ldots m-1 \text{ ; } j\text{: } 0 \ldots n-1\} \tag{3.20}$$

These error factors provide a means to evaluate how well any particular set of values for the unknown variables satisfy the system equations. By treating the system errors as an $m \times n$ component vector $\bar{e}$, its magnitude may be evaluated and used to represent the overall system error. The goal of the solution process is then to determine the solution which minimizes this magnitude.

3.6.2 Minimization Method

A Newtonian numerical search process is used to solve the nine nonlinear system equations. By starting at some defined point (set of values for the unknowns) the Newtonian search determines the change in each dimension (each unknown) necessary to minimize the functional values of each equation (errors $\bar{e}$). This is based upon a first order Taylor series. As an example, take the case of three arbitrary system equations in three unknowns:

$$f_0(u,v,w) = e_0 \tag{3.21}$$

$$f_1(u,v,w) = e_1 \tag{3.22}$$

$$f_2(u,v,w) = e_2 \tag{3.23}$$

Although it may not be possible to find a set of variables which makes each component of $\bar{e}$ equal to zero, the errors may be minimized by a search process which assumes that they are, and then attempts to find a solution which satisfies that condition. This is accomplished through an iterative use of a first-order Taylor series expansion. Evaluated at a particular set of variable values $(u,v,w)_q$, the first order Taylor series expansion for Eqs. 3.21-3.23 would be:

$$f_{0_q} + \left(\frac{\partial f_0}{\partial u}\right)_q \delta u + \left(\frac{\partial f_0}{\partial v}\right)_q \delta v + \left(\frac{\partial f_0}{\partial w}\right)_q \delta w = 0 \tag{3.24}$$

$$f_{1_q} + \left(\frac{\partial f_1}{\partial u}\right)_q \delta u + \left(\frac{\partial f_1}{\partial v}\right)_q \delta v + \left(\frac{\partial f_1}{\partial w}\right)_q \delta w = 0 \tag{3.25}$$

$$f_{2_q} + \left(\frac{\partial f_2}{\partial u}\right)_q \delta u + \left(\frac{\partial f_2}{\partial v}\right)_q \delta v + \left(\frac{\partial f_2}{\partial w}\right)_q \delta w = 0 \tag{3.26}$$

Equations 3.24-3.26 may be restructured as:

$$\begin{bmatrix} \frac{\partial f_0}{\partial u} & \frac{\partial f_0}{\partial v} & \frac{\partial f_0}{\partial w} \\ \frac{\partial f_1}{\partial u} & \frac{\partial f_1}{\partial v} & \frac{\partial f_1}{\partial w} \\ \frac{\partial f_2}{\partial u} & \frac{\partial f_2}{\partial v} & \frac{\partial f_2}{\partial w} \end{bmatrix}_q \begin{bmatrix} \delta u \\ \delta v \\ \delta w \end{bmatrix} = \begin{bmatrix} -f_0 \\ -f_1 \\ -f_2 \end{bmatrix}_q \tag{3.27}$$

Equation 3.27 now represents a system of linear equations, and is of the form:

$$[\mathbf{J}][\vec{x}] = [-\vec{f}] \tag{3.28}$$

In the global positioning system utilizing the equivalent velocity approximation, the Jacobian $\boldsymbol{J}$ is actually a $(m \times n)\mathrm{x}(6+m)$ matrix. The vector $\vec{x}$ represents the desired finite changes in the unknowns, and the vector $\vec{f}$ is the error vector with each term negated. The Jacobian and error vector values are calculated at a particular set of dimensional values, while $\vec{x}$ must be solved.

If the Jacobian was a square matrix, then the next step would be to solve for $\vec{x}$ using an established linear matrix solution method such as LU decomposition. However, it is possible for the number of system equations to be greater than the number of unknowns. This results in an overdetermined system, and an additional step is required before the matrix solution can be performed.

3.6.3 Overdetermined Systems

At first it may seem that an overdetermined system is undesirable, and that the best way to approach the solution of an overdetermined system would be to simply eliminate some of the system equations such that the number of remaining equations equaled the number of unknowns. If the system equations were based upon exact theoretical formulae, then this elimination approach might be plausible. However, since the equations are not exact due to the equivalent velocity approximation and the fact that their values will be a product of real-world application, a redundancy of information is extremely beneficial. In a sense, the additional number of equations serve to reduce the overall effects of the inherent inconsistencies. Although there is not a quantified correlation between increasing accuracy and degree of overdetermination, reducing the ratio of unknowns to equations will generally result in a more robust system which is less sensitive to the unavoidable discrepancies and perturbations of the applied global positioning scheme.

In order to discuss the solution method for an overdetermined system, it is first necessary to employ the concept of the residual vector. The residual vector of Eq. 3.28 would be:

$$\left[\vec{R}\right] = [\mathbf{J}][\vec{x}] + \left[\vec{f}\right] \tag{3.29}$$

For the ideal case, the residual vector $\vec{R}$ would be reduced to the zero vector with an appropriate choice of $\vec{x}$. In most cases, however, this will not be possible since there are more equations to satisfy than there are definable variable values. Thus, some method of minimizing the residual vector must be used. An approach which minimizes the sum of the squares of the residual components seems appropriate and has been utilized in past acoustical ranging systems (Refs. 5,18). This method may be demonstrated by the following example:

A system of p equations and q unknowns (p > q):

$$\begin{bmatrix} a_{11} & \cdots & a_{1q} \\ \vdots & \ddots & \vdots \\ a_{p1} & \cdots & a_{pq} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_q \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_p \end{bmatrix} \tag{3.30}$$

Converted to a normalized n by n system by least-squares method:

$$\begin{bmatrix} (\vec{a}_1 \bullet \vec{a}_1) & \cdots & (\vec{a}_1 \bullet \vec{a}_q) \\ \vdots & \ddots & \vdots \\ (\vec{a}_q \bullet \vec{a}_1) & \cdots & (\vec{a}_q \bullet \vec{a}_q) \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_q \end{bmatrix} = \begin{bmatrix} (\vec{a}_1 \bullet \vec{b}) \\ \vdots \\ (\vec{a}_q \bullet \vec{b}) \end{bmatrix} \tag{3.31}$$

Where $(\vec{a}_i \bullet \vec{a}_j) = a_{1i}a_{1j} + \cdots + a_{pi}a_{pj}$

Now that the system has been normalized, $\vec{x}$ may be solved for by LU decomposition as previously mentioned. The resulting solution is the $\vec{x}$ which minimizes the vector magnitude of $\vec{R}$, and represents the dimensional changes needed to minimize the error vector. These dimensional changes are made, and the process is then repeated. Subsequent iterations of this procedure should quickly converge to a stable answer.

4. The Applied Global Positioning Scheme

Up to this point the global positioning system has been discussed in a relatively generalized manner. Since the goal of this work is to design a specific system to be used with the BWNT URSULA robot, it is necessary that details associated with the implementation of that system be investigated and addressed. This examination will reveal new difficulties, and will lead to variations in the basic solution scheme.

4.1 URSULA Operation Characteristics

Certain characteristics of the mobile robot operation are significant to the implementation of the global positioning system. First, it is important that the mobile transceivers retain a clear line-of-sight between one another. The acoustic paths must not be obstructed by the mobile platform, manipulator arm, or other equipment. Fortunately, the orientation of the mobile robot will remain relatively constant throughout the reactor vessel inspection. By design, the manipulator arm always reaches down in the vessel, below the mobile platform. This condition greatly reduces the difficulties associated with the placement of the mobile transceivers.

Aside from the time spent repositioning the mobile platform base, the platform is always on the vessel walls. BWNT has indicated that the radial distance to the vessel walls may not be constant due to uneven cladding and surface irregularities, but these inaccuracies may prove to be acceptable as a means of determining the radial position of the robot. This factor is not utilized in the work presented here, but it is worth mentioning and possibly worth further examination.

There was some consideration given to the possibility of placing a transceiver on the manipulator arm. This might provide a direct link between the positioning system and the transformations which relate the manipulator to the robot base. However, there are situations during the inspection process which will obstruct the line of sight to the manipulator transceiver such as the examination of the inlet and outlet nozzles. Thus, for the time being, the manipulator transceiver option has not been utilized.

4.2 The Marquest Transceivers

Much of the past work relating to underwater acoustic positioning has concerned the generation and processing of ultrasonic transmissions. This aspect of acoustic positioning is not within the scope of this work. Thus, for the time being it is necessary to rely upon a commercially available system of acoustic transceivers. The Marquest Group, Inc. produces just such a system which they have designated SHARPS.

> [The Sonic High Accuracy Ranging & Positioning System] is adaptable for use in rivers, lakes, ponds, or the open ocean, but is exceptionally well-suited to high multi-path environments such as enclosed tanks... Applications include high-precision tracking and navigation of divers and remotely-operated vehicles... (SHARPS Operation and Software Manual 1992, 8)

The SHARPS transceivers utilize hemispherical piezo transducers to generate and receive acoustic pulses centered at 300 kHz. Operating at the maximum range of 100 meters, Marquest claims accuracies of ± 2 cm, and repeatability of 0.5 cm.

At first it may seem that this system is ideal, and directly employable as the reactor vessel global positioning system. However, SHARPS is only capable of accurately locating a single point and does not address the orientation requirement. Additionally, SHARPS relies upon a very accurate knowledge of the speed of sound.

> Obtaining this kind of accuracy requires very precise knowledge of the speed of sound. For instance, in order to obtain the accuracy stated above, speed of sound errors must be limited to 1 cm at 100 meters, or 1 part per 10,000. Sound velocimeters which provide this level of accuracy are commercially available. (SHARPS Operation and Software Manual 1992, 8)

As discussed, the reactor vessel water may not be a homogeneous temperature. Thus, the use of sound velocimeters would only provide localized point velocities. A localized velocity applied to the entire system which was not a uniform temperature would erroneously treat all transmission velocities as equivalent. As discussed at the end of Section 2.3, a relatively small miscalculation of these velocities can result in significant positioning errors.

It is clear that SHARPS may not be applied directly, but the transceivers it utilizes along with the signal generation and processing, seem to be ideal. This is the system which has been purchased by BWNT, and thus the specifics of the global positioning system must account for the characteristics of SHARPS. One of these considerations concerns the

number of total transceivers. The basic SHARPS system is comprised of a maximum of 6 operational transceivers, and 1 spare. The deck unit which connects the transceiver cables to the controlling computer is configured for six transceivers, and though it might be adapted for more, it is desirable to design a positioning system which only relies upon the standard six.

4.3 Fixed Transceiver Placement

It is vital that the fixed transceivers be attached to the reactor vessel at accurately known positions. Additionally, they must be located such that the paths between the fixed and mobile transceivers remain clear throughout the inspection process. Ideal locations which meets these requirements are the stud hole plugs (Fig. 1.1). These holes are used to position the top of the reactor core pressure housing (removed in Fig. 1.1) and have been located very accurately.

4.4 Addition of Inclinometers

In addition to the line-of-sight benefits, the relatively constant orientation of the mobile robot facilitates the use of inclinometers to obtain the platform rotation angles. The rotation angles φ and μ may be able to be obtained by only a pair of inclinometers; one aligned with the q-axis and one aligned with the r-axis. This is the reason Euler rotations were chosen to describe the orientation of the mobile platform. If fixed rotations were used instead, then the inclinometer angles could not be incorporated directly. An additional transformation would be necessary to relate the inclinometer readings and the orientation of the robot.

As previously discussed, three fixed transceivers and three mobile transceivers are the minimum required to uniquely determine the position and orientation of the robot. With the addition of the inclinometers to determine φ and μ, this requirement relaxes since the total number of unknowns is reduced as shown in Eq. 4.1 as compared to Eq. 3.9. The minimum requirements for transceiver configurations when the inclinometers are utilized are shown in Table 4.1. Note that a minimum of three mobile transceivers is still required, but now it is possible to solve the system with only two fixed transceivers.

$$m \times n \geq 4 + m \tag{4.1}$$

Table 4.1 Transceiver combinations with the use of inclinometers; $m \times n \geq 4 + m$ restriction.

Number of Fixed Transceivers m	Resulting Minimum Required Mobile Transceivers n
2	3
3	$2.33 \rightarrow 3$

4.5 Velocity Compensation System

The greatest obstacle to an accurate global positioning system is the varying temperature, and thus varying speed of sound characteristics of the reactor vessel water. Even with the use of the equivalent velocity approximation, the system dimensions are still partially comprised of m unknown velocities. One approach to this problem might be to closely associate a sound velocimeter with each of the fixed and each of the mobile transceivers. These transducers would then provide a local speed of sound value in their vicinity. By using a linear interpolation between these values, a speed of sound between each transceiver of the fixed array and each transceiver of the mobile array could be obtained as in Fig. 4.1 and Eq. 4.2. Here the localized speed of sound at a transceiver is subscripted with an f or an m to indicate whether it is associated with the fixed or mobile arrays respectively, followed by a numerical value to indicate the specific transceiver. The velocities between the fixed and mobile transceivers are indicated as before. For example, the speed of sound at transceiver F_0 would be designated v_{f0}, while the velocity between F_0 and M_2 would be designated v_{02}.

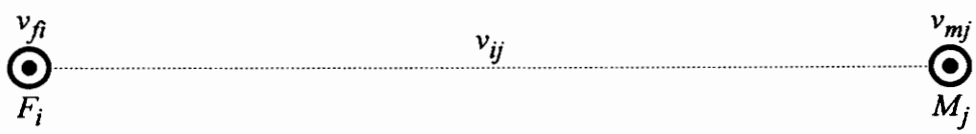


Figure 4.1 Transmission between two transceivers with known local point velocities.

$$v_{ij} \approx \frac{v_{fi} + v_{mj}}{2} \tag{4.2}$$

If this method is used, then the velocities are no longer constituents of the system dimensions. The resulting balance of system equations to unknowns is shown in Eq. 4.3, and the possible transceiver configurations are shown in Table 4.2. If the inclinometers are also utilized, then there are only four remaining unknowns in the system: P_x, P_y, P_z, θ.

As indicated by Eq. 4.4 and Table 4.3, only two mobile transceivers are required. Although this is generally true, it must be mentioned that there is a singular point in the solution when the vector between the mobile transceivers is aligned with the global z-axis. If this occurs, θ cannot be determined.

$$m \times n \geq 6 \tag{4.3}$$

Table 4.2 Transceiver combinations with the use of interpolated velocities; $m \times n \geq 6$ restriction.

Number of Fixed Transceivers m	Resulting Minimum Required Mobile Transceivers n
2	3
3	2

$$m \times n \geq 4 \tag{4.4}$$

Table 4.3 Transceiver combinations with the use of interpolated velocities and inclinometers; $m \times n \geq 4$ restriction.

Number of Fixed Transceivers m	Resulting Minimum Required Mobile Transceivers n
2	2
3	$1.33 \rightarrow 2$

There are two major drawbacks associated with this velocity interpolation approach. The first is that it requires an additional system of transducers to be integrated into the complete global positioning scheme. This means $m+n$ velocimeters and their associated hardware including cabling and data acquisition system must be accounted for physically and financially. Secondly, the interpolation between two localized velocities may not accurately represent the true transmission velocity. This discrepancy is evidenced by Fig. 4.2 which illustrates the difference between the interpolated and true velocities for an example velocity profile between two transceivers separated by 3 m. The represented

temperature distribution is parabolic and symmetric about $d = 1.0$ m where it is 30°C. At $d = 3$ m the modeled temperature is 40°C. The true experienced velocity is represented by Eq. 4.5, where Δs represents the total separation of the transceivers ($s_1 - s_0 = \Delta s = 3$ m in this case). Eq. 4.6 is the linearly interpolated velocity. The reasons a parabolic temperature distribution was chosen here as an example will be made clear in the next chapter.

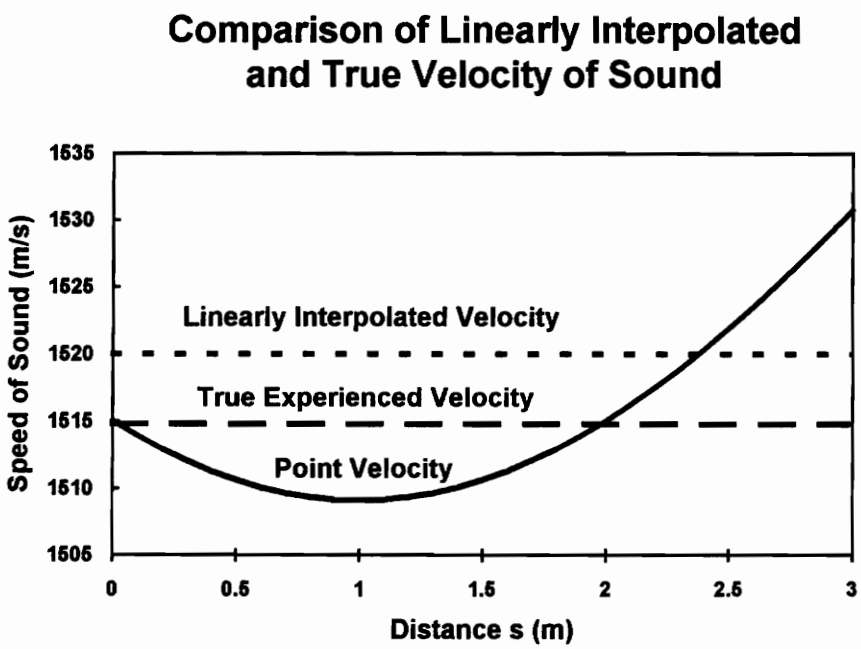


Figure 4.2 Comparison of the true transmission velocity with the linear interpolated velocity.

$$v_{experienced} = \frac{\Delta s}{\int_{s_0}^{s_1} \frac{\partial s}{v_s}} \tag{4.5}$$

$$v_{interpolated} = \frac{v_{s_0} + v_{s_1}}{2} \tag{4.6}$$

However, if the temperature profile between two transceivers is close to linear, or at least not drastically nonlinear, then this discrepancy will be less significant and may be acceptable. If this is the case, then the use of localized speeds of sound may be feasible, although there still remains the significant disadvantage of the additional hardware.

By utilizing additional information available from the transceivers, it is possible to approximate the effects of pairing velocimeters with the acoustic transceivers. So far, all discussed acoustic signals have been between the fixed array and the mobile array. It is also possible for the transceivers of each array to communicate with one another in what is termed a *sing-around*. Figure 4.3 shows an array of three transceivers (T_0, T_1, T_2) which are connected together by rigid members of known lengths. This system could represent either the fixed or mobile transceiver arrays. Since the distances between each of the transceivers is known, and the times of flight may be obtained, it is thus possible to determine an accurate speed of sound between each of the transceiver locations. These velocities are designated v_{01}, v_{12}, v_{20}. Note that here the subscripts do not represent fixed and mobile transceivers (as is the standard adopted convention), nor is their order significant. They refer only to the local transceivers as indicated by Fig. 4.3.

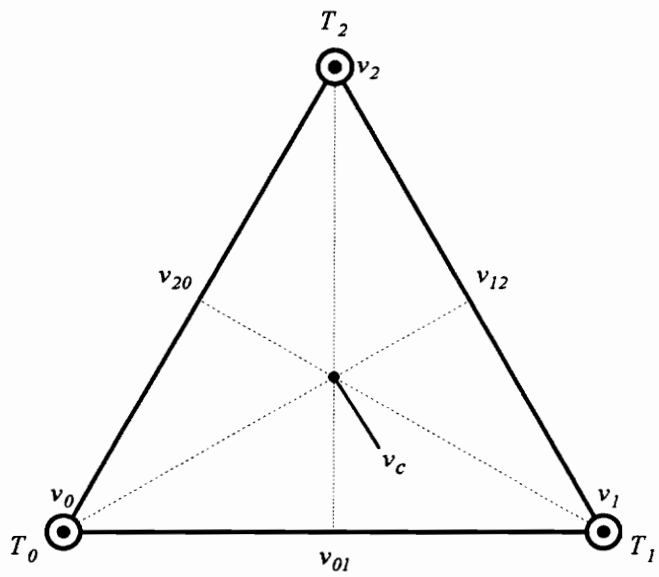


Figure 4.3 Transceiver sing-around arrangement and nomenclature for nodal velocity determination.

The immediate goal is to approximate the velocities at each of the transceivers by using the velocities between them. This is done by first defining the velocity at the center of the array as the average of each of the sing-around velocities (Eq. 4.6).

$$v_c \equiv \frac{v_{01} + v_{12} + v_{20}}{3} \tag{4.6}$$

The linear interpolation of the velocities at the vertices shown in Eq 4.7 logically follows from the fact that the distance from a side of the triangular array to the center is always one-third the distance from that side to the opposite transceiver.

$$v_n \approx v_s + 3(v_c - v_s) \tag{4.7}$$

Here v_n is the *nodal* velocity (velocity at a transceiver position), v_s is the velocity on the side opposite that transceiver, and v_c is the center velocity as previously defined. For the system of Fig. 4.3, the resulting nodal velocities are given in Eqs. 4.8-4.10.

$$v_0 \approx v_{12} + 3(v_c - v_{12}) \tag{4.8}$$

$$v_1 \approx v_{20} + 3(v_c - v_{20}) \tag{4.9}$$

$$v_2 \approx v_{01} + 3(v_c - v_{01}) \tag{4.10}$$

These nodal velocities approximate the localized speed of sound at each of the transceivers using available information, and allow the velocity interpolation between the fixed and mobile arrays without the addition of velocimeters.

Although the generation of nodal velocities and the subsequent linear interpolation to obtain the transmission velocities can be used as a stand-alone technique of reducing the system unknowns, it may also be applied to the system which utilizes the equivalent velocity approximation. As discussed, the use of the equivalent velocity approximation introduces error into the system equations. This is due to the fact that each of the transmission velocities between the mobile platform transceivers and a particular fixed transceiver may not be exactly equal. It is possible to use the nodal velocities to partially compensate for these discrepancies by augmenting the system equations with an additional scale factor as shown in Eq. 4.11.

$$\Delta_{ij} - \left(\sigma_{ij} v_i t_{ij}\right)^2 = 0 \quad \{i\text{: } 0 \ldots m-1 \text{ ; } j\text{: } 0 \ldots n-1\} \tag{4.11}$$

The equivalent velocity simplification of the system equations approximates the $m \times n$ true transmission velocities by only m velocities in order to decrease the total number of unknowns. A judicious choice of σ_{ij} will essentially reinstate the differences of the $m \times n$ velocities while retaining the reduced system dimensions. In order to do this, the nodal velocities must first be evaluated for each of the fixed and each of the mobile transceivers

as in Eq. 4.7. Eq. 4.12 then approximates the simplified velocities (v_i) of the system equations by averaging the nodal velocity of fixed transceiver F_i with the mobile array center velocity v_c. Eq. 4.13 similarly approximates the true transmission velocity between fixed transceiver F_i and mobile transceiver M_j by the linear interpolation of their associated nodal velocities v_{fi} and v_{mj}.

$$v_i \approx \frac{v_{fi} + v_c}{2} \tag{4.12}$$

$$\sigma_{ij} v_i \approx \frac{v_{fi} + v_{mj}}{2} \tag{4.13}$$

Each adjustment value σ_{ij} may now be evaluated from these two equations as in Eq. 4.14.

$$\sigma_{ij} \equiv \frac{v_{fi} + v_{mj}}{v_{fi} + v_c} \tag{4.14}$$

4.6 Vessel Temperature Models

In order to quantitatively examine different global positioning schemes before actual testing of the system it is necessary to create a model of the temperature distributions within the reactor vessel. Unfortunately, the true thermal features of the water are unknown, and an accurate model which takes into account all the possible temperature characteristics and variations would be extremely difficult if not impossible to create. However, it is possible to base a couple of simplified models on educated guesses at what might be encountered.

These models will be used to generate times of flight which in the real application would be provided by the Marquest transceivers. In the modeled system the true global positions of the fixed and mobile transceivers will be defined, thus their separations will be known. If the transmission velocities between transceivers can be determined, then the times of flight can be evaluated. This is accomplished by carrying out the integration of Eq. 4.5, which is dependent upon the point velocities. These point velocities are obtained by creating a temperature distribution model, and evaluating the speed of sound by Medwin's formula. The positioning equations are then solved utilizing the techniques previously discussed, incorporating only that information which would be available to the real system.

4.6.1 Linear Model

The linear model of the reactor vessel water assumes that the temperature decreases proportionally to increasing depth. This is similar to what is found in relatively stagnant bodies of water unexposed to environmental variations. The warmer water rises, and the denser, cooler water tends to sink. This is the basis of the linear temperature model.

The temperature at depth $z = 0$ is designated T_t to indicate the temperature at the *top* of the operating domain. The temperature at depth $z = -D$ is designated T_b to indicate the temperature at the *bottom* of the operating domain. The resulting temperature at any depth z in the vessel based on this linear model is designated T_l and is given by Eq. 4.15.

$$T_l = \frac{z}{D}(T_t - T_b) + T_t \tag{4.15}$$

4.6.2 Parabolic Model

The parabolic model of the reactor vessel water assumes that the temperature is radially dependent, being a higher temperature T_w at the reactor vessel wall due to its stored heat and radiative effects, and a lower temperature T_c at the center of the vessel. The general formula for the parabolic equation dependent on radial position is given by Eq. 4.16. The coefficients are solved for by utilizing the conditions of Eqs. 4.17-4.19, and the resulting temperature at a point due to the parabolic temperature distribution (T_p) is given by Eq. 4.20. The first condition (Eq. 4.17) simply states that the temperature at the center of the vessel must be T_c. The second condition (Eq. 4.18) requires that the temperature at the wall where the radius is R must be equal to T_w. The third condition (Eq. 4.19) utilizes the radial symmetry of the vessel to require that the slope of the temperature profile equal zero at the center.

$$T_p = ar^2 + br + c \tag{4.16}$$

$$T_{p,r=0} = c = T_c \ , \ T_{p,r=R} = aR^2 + bR + c = T_w \ , \ \left[\frac{dT_p}{dr}\right]_{r=0} = b = 0 \tag{4.17, 4.18, 4.19}$$

$$T_p = \left(\frac{T_w - T_c}{R^2}\right)r^2 + T_c \tag{4.20}$$

5. Simulation of Positioning System

5.1 System Parameters

There are essentially two approaches to the global positioning system which will be examined. The first utilizes the equivalent velocity assumption to reduce the number of unknowns, while the second utilizes the interpolated velocities to approximate the transmission velocities. Within each of these systems there are many different variations of configuration and environment which have been examined. In order to appreciate the overall effects of these differences, they must first be explained.

5.1.1 Fixed Transceiver Configuration

In all cases, three fixed transceivers have been located at the stud holes, evenly spaced around the vessel circumference. However, it is possible that the brackets which will position them there are designed such that the actual transceiver is cantilevered out towards the center of the vessel. In this way their radial location in not necessarily the same as the vessel radius, allowing variation in their overall separation.

5.1.2 Mobile Transceiver Configuration

Perhaps the most important physical aspect of the global positioning scheme concerns the relative locations of the mobile transceivers which will be fixed to the robot. Although the possible variations are almost unlimited, they may be divided into three categories.

5.1.2.1 Number

Although in some instances it is possible to use only two mobile transceivers, the best results will be obtained by the use of at least three. Since the Marquest system is configured for a total of six transceivers with three allocated to the vessel stud holes, three mobile transceivers are really all that are available.

5.1.2.2 Spacing

The specific spacing of the mobile transceivers is an important aspect of the global positioning system. The transceivers will be arranged as the vertices of an equilateral triangle, and although it may seem that the positioning will become more stable and more accurate as the transceiver separation is increased, there are counteracting considerations. First, the equivalent velocity approximation will become less accurate. As the spacing increases the transceivers are exposed to greater temperature differences. Additionally,

there are physical boundaries to separation. The transceiver array cannot become so large that the robot installation and operation are adversely affected.

5.1.2.3 Orientation

The mobile transceiver array can theoretically take on any orientation relative to the vessel. However, as previously mentioned the robot maintains a relatively consistent orientation, and thus so will the transceiver triangle. Although the choices are infinite, two basic orientations will be examined. The first orientation places the mobile transceiver plane parallel to the global xy plane in a horizontal orientation. The second is a vertical orientation such that the mobile transceiver plane is perpendicular to the global xy plane.

In all cases, slight φ and μ inclinations ($\approx 5^\circ$) have been incorporated. It was discovered early in the simulation testing that when the mobile transceiver array was perfectly horizontal, or perfectly vertical (and perpendicular to the radial vector), the results were exceptionally good. This was due to the fact that the transceivers were closely aligned with the temperature model parameters. When the transceiver plane was oriented in a perfectly horizontal manner, each of the transceivers was at exactly the same depth which was the best configuration for minimal error due to the linear temperature distribution model (Section 4.6.1). When the transceiver plane was oriented vertically and perpendicular to the vessel radial vector, each of the transceivers was at approximately the same radius. This was the best configuration for minimal error due to the parabolic temperature distribution model (Section 4.6.2). Therefore the inclinations were added in order to prevent the simulation from indicating results which might not be representative of the actual applied system, instead being a product of the model characteristics.

The third rotation angle θ is selected at each position of the mobile array such that its orientation will remain constant with respect to the vessel wall as the model is made to circle the reactor interior. That is, if the mobile array began in a vertical position with the mobile transceiver plane perpendicular to the wall, as the array moved about the interior it would continue to be perpendicular to wall.

5.1.3 Depth

The operating depth of the mobile robot will vary over the course of the inspection. Five different distances below the fixed transceiver plane have been examined: 0.0 m, 2.5 m, 5.0 m, 7.5 m, and 10.0 m.

5.1.4 Temperature Distribution

As discussed, variations in the vessel water temperature will adversely affect the global positioning system. The specific effects depend upon both the magnitudes and characteristics of the thermal gradients. Two separate temperature models have been generated, and by combining the two distributions a more complex (and hopefully realistic) system can be created. This is done by calculating the resulting speed of sound from each model and then averaging the two results. Differences in the temperature variables T_c, T_w of Eq. 4.15 and T_t, T_b of Eq. 4.20 will determine the relative effects of each model on the overall combined system.

5.1.5 Incorporated Errors

In addition to the positioning errors associated with the equivalent velocity approximation and the interpolation of transmission velocities, there will be errors due to the real world application of the global positioning system. It is important that the effects of these errors be incorporated into the full simulation in order to obtain true to life results. This has been accomplished by perturbing system parameters in a random manner and subsequently solving the system with these errors included. Since it is unclear in which manner each parameter should be disturbed in order to obtain maximum or average overall positioning errors, the random error addition is iterated, solving the system each time. The results of all solutions are then examined.

5.1.5.1 Fixed Transceiver Positions

Although the fixed transceivers will be attached to the stud holes, there may be some error in their positioning. This may be due to the machining of the retaining bracket which holds the transceiver, or a misaligned placement of the bracket in the stud hole. Additionally, there may be error associated with the assumed location of the transmission point from the piezo tip which generates the acoustic pulse. Errors on the order of a few millimeters will be examined.

5.1.5.2 Mobile Transceiver Positions

Similar to the error associated with the fixed transceiver placement, the mobile transceivers may also experience positioning inaccuracies. In this case, the error is relative to the mobile coordinate frame. Errors on the order of a few millimeters will be examined.

5.1.5.3 Times of Flight

The resolution of the Marquest transceivers is approximately 1 μs and there may be error associated with this limitation. Additionally, a consistent offset error due to calibration may exist, as well as error which increases with the separation between the fixed and mobile transceiver arrays. Each of these conditions has been incorporated into the overall system. Although various combinations and magnitudes of these errors will be examined, the total error will be on the order of a few microseconds.

5.1.5.4 Inclinometers

If inclinometers are utilized, then there will be error associated with the inherent limitations of the transducers, as well as error due to imperfect calibration. Most high precision inclinometers guarantee absolute accuracies better than 0.1 degrees.

5.1.6 Compensation Factor

Equation 4.7 may be rewritten as Eq. 5.1. In this instance, the nodal velocities will also depend upon the coefficient c which will be referred to as the *compensation factor*. This value may be altered to suit the actual experiences of the applied global positioning system, and allows limited adjustment of the system within the software which governs it.

$$v_n \approx v_s + c(v_c - v_s) \tag{5.1}$$

For the equivalent velocity system, a compensation factor value of 1 effectively results in no compensation. This is due to the fact that $c = 1$ causes each of the scale factors σ_{ij} to also be equal to 1.

5.1.7 Operating Radius

One final parameter which bears examination is the operating radius of the mobile transceiver plane as it circles the interior of the vessel. This is defined as the distance from the center axis of the reactor (z axis as defined) to the pivot.

5.2 Simulation Results

The nomenclature of each of the data files generated (Appendix 3) indicates the specific parameters of the system which was solved. These conventions are summarized in Appendix 2. The results were used to generate graphical representations which illustrate the effects of variations in different system parameters. In order to compare the

effectiveness of each of the solution approaches, errors are examined for both the equivalent velocity and the interpolated velocity systems.

Each system solution examines various positioning errors. An error in an arbitrary parameter Ω is described by Eq. 5.2.

$$\Omega_{error} \equiv \Omega_{determined} - \Omega_{actual} \tag{5.2}$$

The examined parameters are the pivot positions ${}^{G}P_x, {}^{G}P_y, {}^{G}P_z$ and the rotation angles θ, φ, μ. Also examined were the angular position of the pivot within the reactor vessel α, and the radial position of the pivot. The angular position α is defined as in Fig. 5.1.

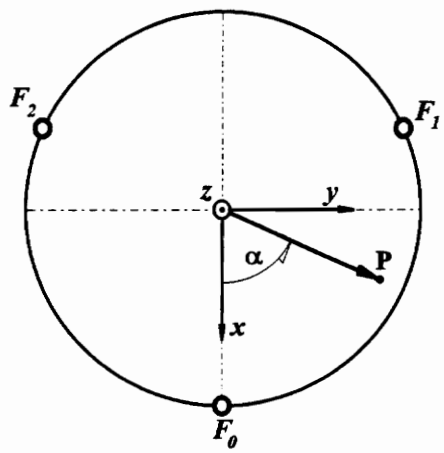


Figure 5.1 Definition of the angular position α of the pivot within the reactor vessel.

Also defined is a *total error* and a *radial error* (Eqs. 5.3,5.4). Total error is the magnitude of displacement between the actual position of the pivot and the position which was determined by the solution process. The radial error is the difference between the true pivot location radius and the determined pivot location radius with respect to the vessel center. It may turn out that in the actual operation of URSULA, the radial error is not a very significant factor due to the fact that the robot will be on the wall. This knowledge may eliminate the need for a radial measurement except as additional confirmation that the platform is against the vessel interior surface.

$$total\ error \equiv \left[\left({}^{G}P_{x,error}\right)^2 + \left({}^{G}P_{y,error}\right)^2 + \left({}^{G}P_{z,error}\right)^2\right]^{1/2} \tag{5.3}$$

$$radial\ error \equiv \left[\left({}^{G}P_{x,determined}\right)^2 + \left({}^{G}P_{y,determined}\right)^2\right]^{1/2} - \left[\left({}^{G}P_{x,actual}\right)^2 + \left({}^{G}P_{y,actual}\right)^2\right]^{1/2} \quad \textbf{(5.4)}$$

The results of the error examinations are summarized in Appendix 1. It is important to realize that these results are most likely more qualitative than quantitative. That is, the results indicate general magnitudes and error trends more than they represent exact errors which will be encountered in the applied global positioning system.

5.2.1 Depth

Figures 5.2 and 5.3 illustrate the effects of depth on the total error of the positioning system as the mobile robot circumscribes the interior of the reactor vessel wall. Note that only 180° of travel is displayed. This is simply due to the fact that the fixed transceivers are equally spaced as in Fig. 5.1, and so the results are symmetrical about the 180° position. Additionally, Figs. 5.4, 5.5 show the radial and depth components of the total error at a depth of 7.5 m. This is a standard depth which is used as a base of comparison for other error effects due to variations in other parameters. This depth was chosen as a standard since it is approximately the maximum depth below the fixed transceiver plane at which the mobile robot is expected to operate.

5.2.2 Mobile Transceiver Array Orientation

Unless otherwise noted, all examinations are based upon a standard vertical transceiver arrangement (see Appendix 2, Local Positions of Mobile Transceivers - Index A). However, Figs. 5.6-5.9 may be compared to Figs. 5.2-5.5 in order to discern the significance of mobile transceiver array orientation. The errors associated with a horizontal orientation are illustrated in the former, while errors associated with a vertical orientation are illustrated in the latter.

Also note that each examination of the equivalent velocity system incorporates inclinometers in the solution process. Without the use of at least one inclinometer, it was found that the equivalent velocity system produced poor results. This was due to the fact that the system dimensions were high (9 dimensions) and that the number of equations equaled the number of unknowns. Thus, the system was operating on the edge; there was no overdetermination which would serve to reduce the inaccuracies. Only future testing will determine whether or not this problem will manifest itself in the applied global positioning system.

5.2.3 Temperature Distribution

The temperature distribution of the reactor vessel water greatly affects the positioning accuracies. Figs. 5.10-5.13 may be compared to Figs. 5.2-5.5 to illustrate the effects of more severe temperature gradients.

Figures 5.14, 5.15 present the results of a system with the same temperature gradients as in Figs. 5.4, 5.5 but with a decreased bulk mean temperature.

5.2.4 Error Incorporation

Figures 5.16-5.22 illustrate the effects of incorporating errors which might be encountered in the real global positioning system. Figures 5.16, 5.17 illustrate the effects of adding full system errors to the solution process, while Figs. 5.18-5.22 show the relative inaccuracies which result from each of the parameter errors examined individually.

5.2.5 Compensation Factor

The results of compensation for the equivalent velocity system are shown in Fig. 5.23 which compares the error with c=3.0 and c=1.0 (no compensation). The effects of different compensation factor values for the two positioning systems are shown in Figs. 5.24, 5.25.

5.2.6 Transceiver Spacing

Figures 5.26, 5.27 may be compared to Figs. 5.4, 5.5 in order to see the effects of reducing the separation between each of the fixed transceivers. Similarly, Figs. 5.28, 5.29 may be compared to Figs. 5.4, 5.5 in to see the effects of increasing the separation between each of the mobile transceivers.

5.2.7 Operating Radius

Figures 5.30, 5.31 illustrated the effects of reducing the pivot radius to 1.0 m. These graphs may be compared to Figs. 5.4, 5.5 which show the error at the standard radius of 1.5 m.

5.2.8 Error Graphs

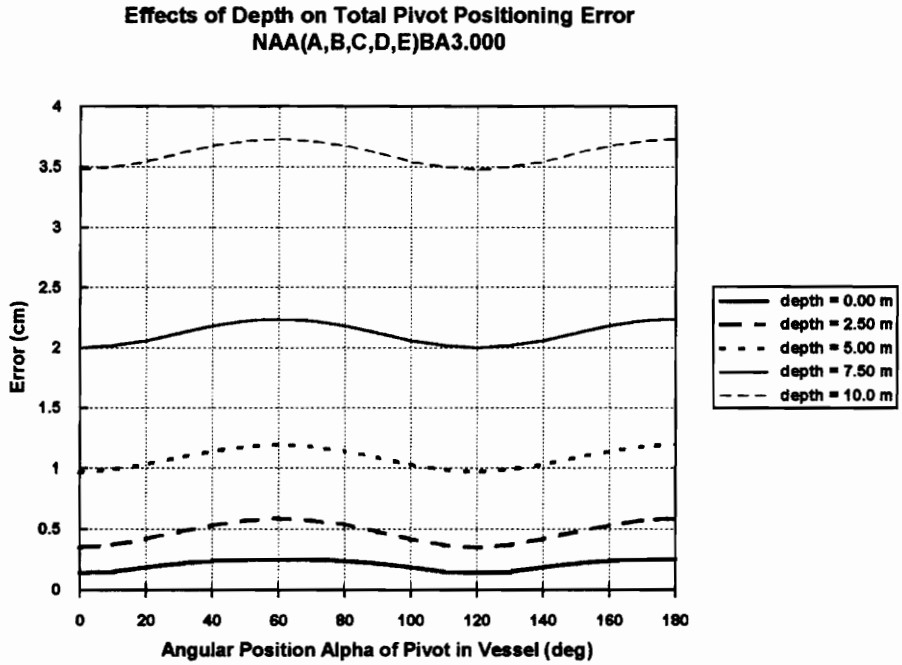


Figure 5.2 Total pivot positioning error at various operating depths for the interpolated velocity system.

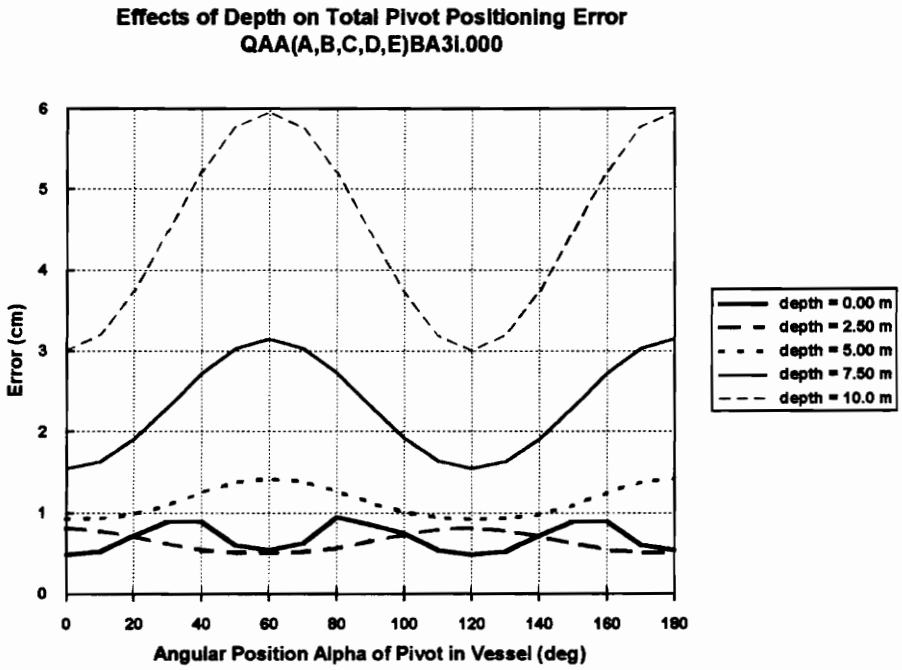


Figure 5.3 Total pivot positioning error at various operating depths for the equivalent velocity system.

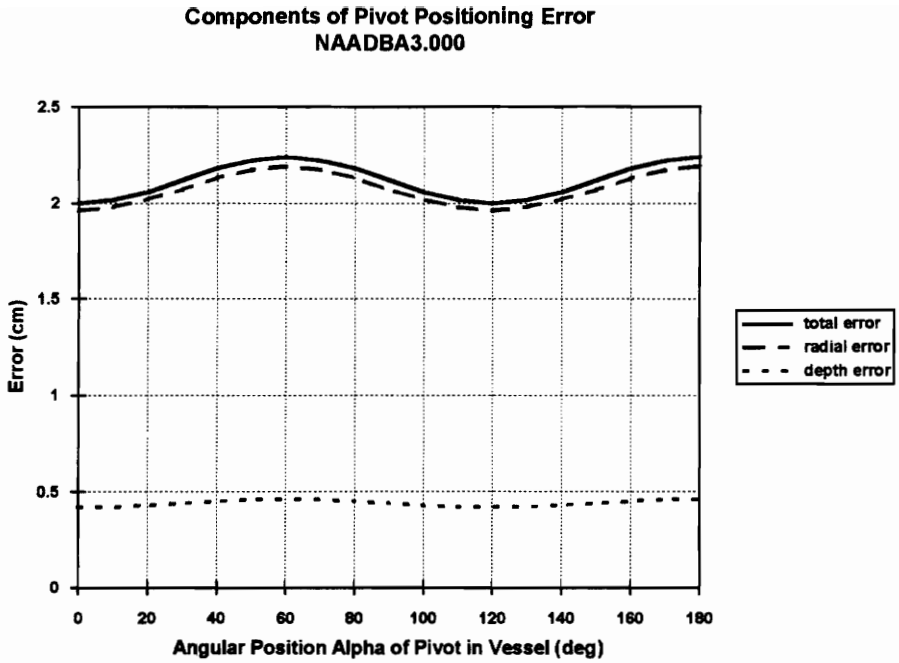


Figure 5.4 Radial and depth components of the total error at 7.5 m depth below the fixed transceiver plane. Results obtained from the interpolated velocity system.

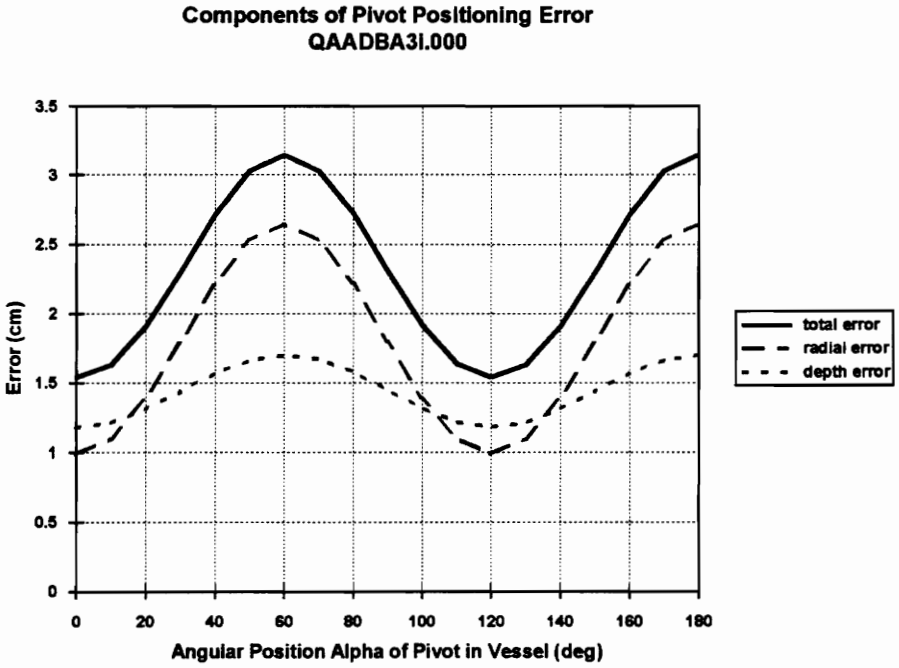


Figure 5.5 Radial and depth components of the total error at 7.5 m depth below the fixed transceiver plane. Results obtained from the equivalent velocity system.

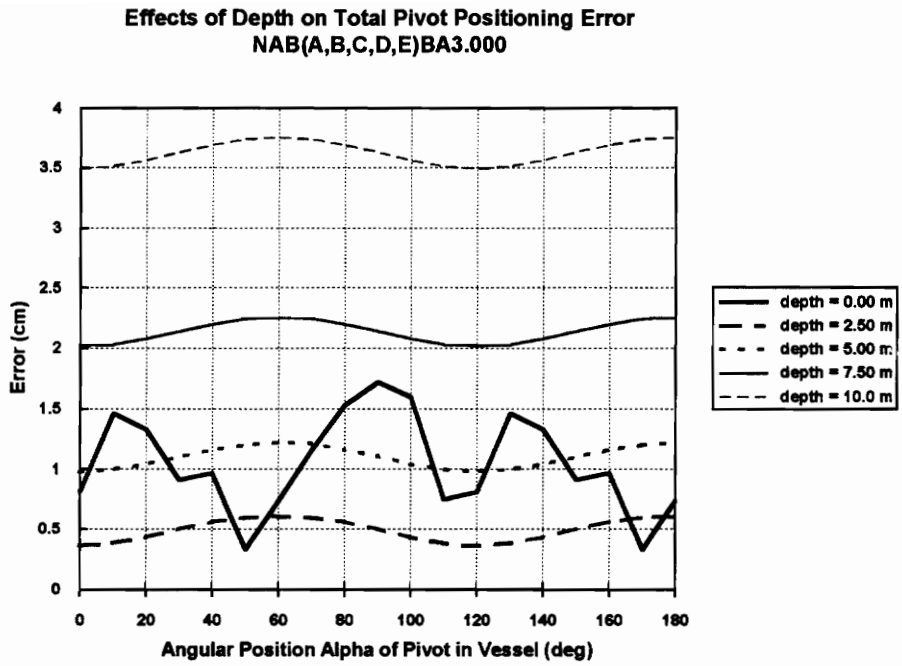


Figure 5.6 Total pivot positioning error at various operating depths of a horizontally oriented mobile transceiver plane. Results obtained from the interpolated velocity system.

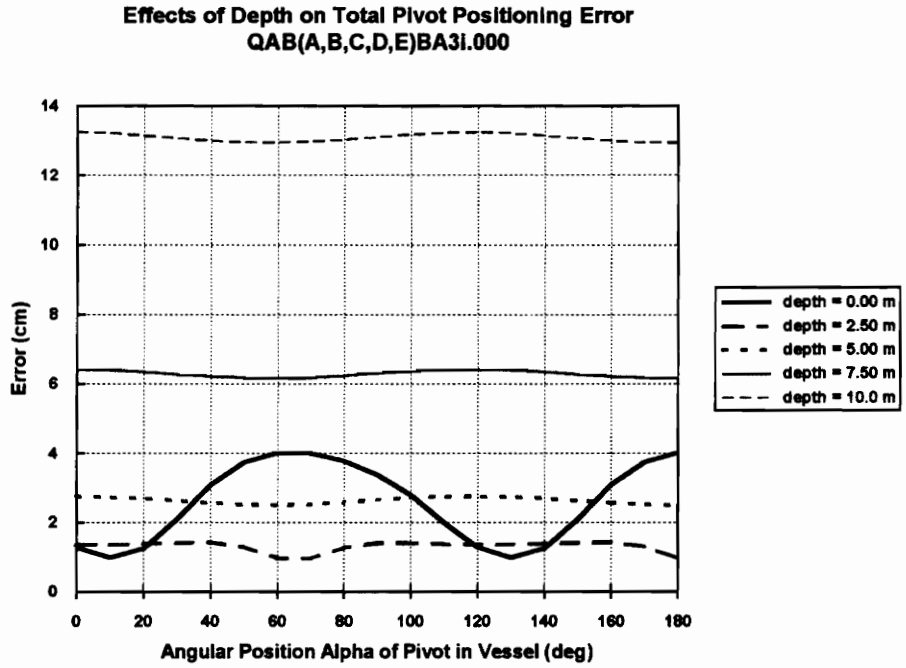


Figure 5.7 Total pivot positioning error at various operating depths of a horizontally oriented mobile transceiver plane. Results obtained from the equivalent velocity system.

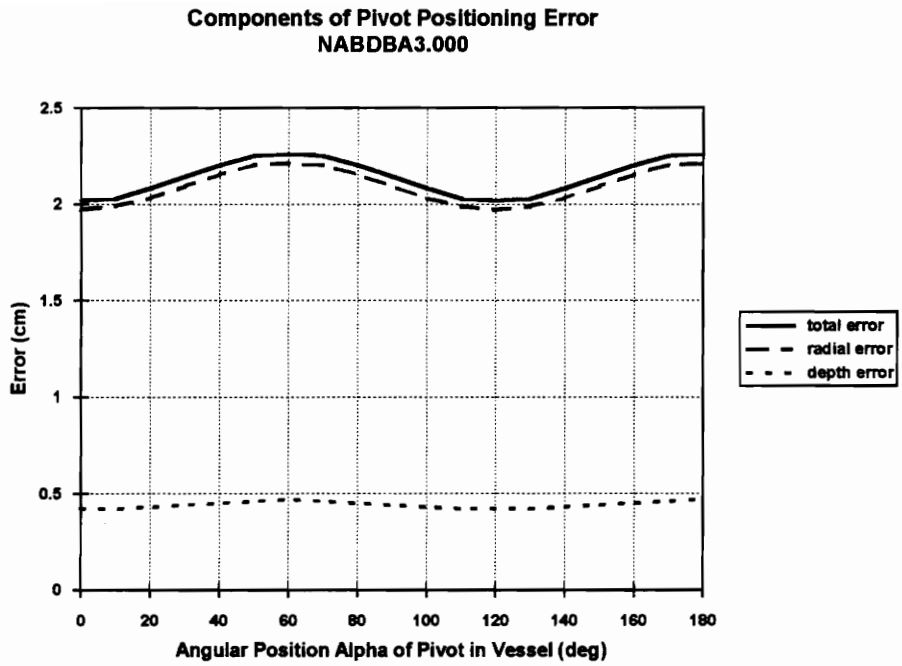


Figure 5.8 Radial and depth components of the total error at 7.5 m depth below the fixed transceiver plane for a horizontal mobile array. Results obtained from the interpolated velocity system.

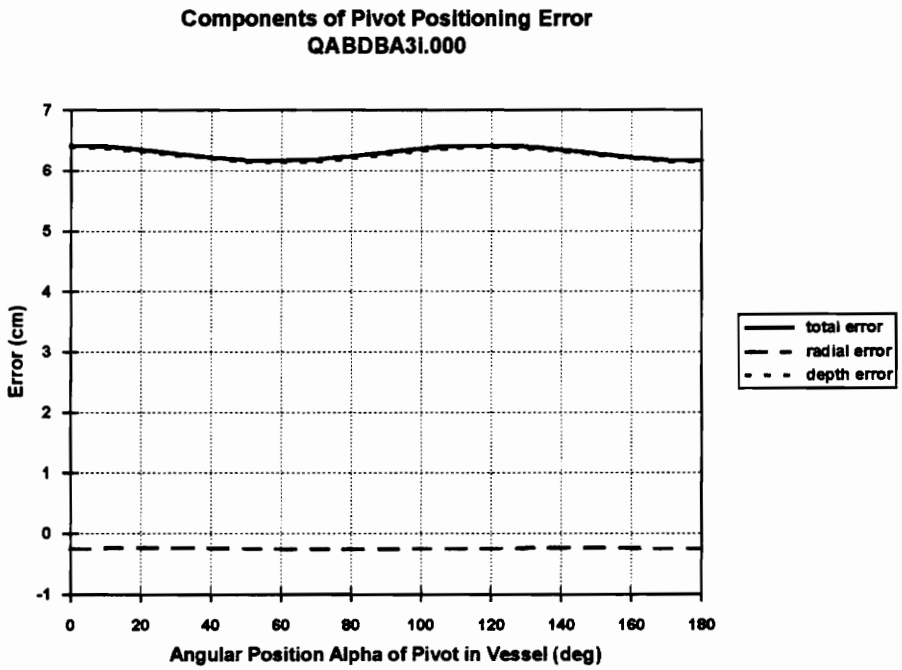


Figure 5.9 Radial and depth components of the total error at 7.5 m depth below the fixed transceiver plane for a horizontal mobile array. Results obtained from the equivalent velocity system.

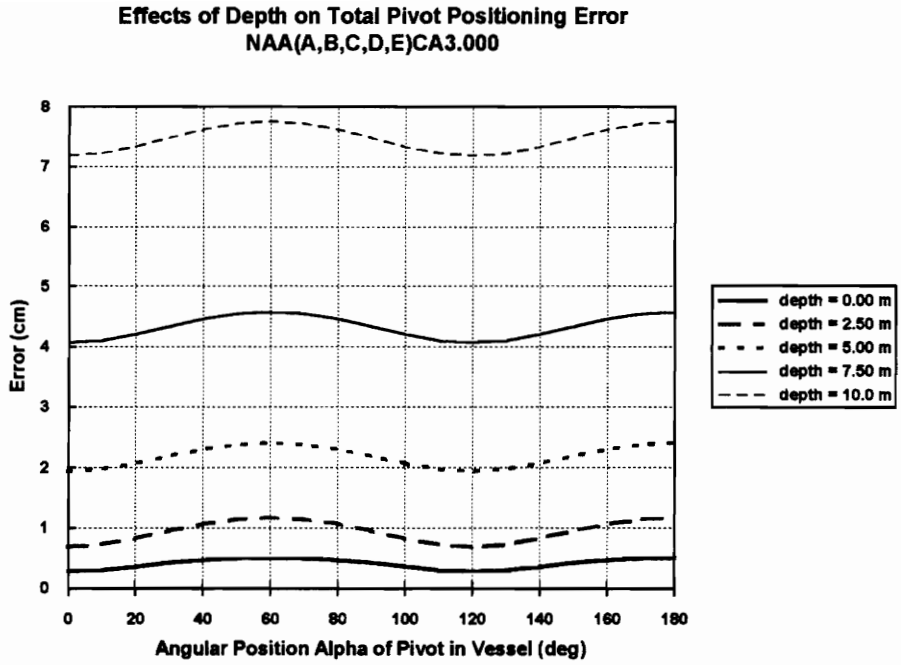


Figure 5.10 Total pivot positioning error at various operating depths from the interpolated velocity system incorporating severe temperature gradients.

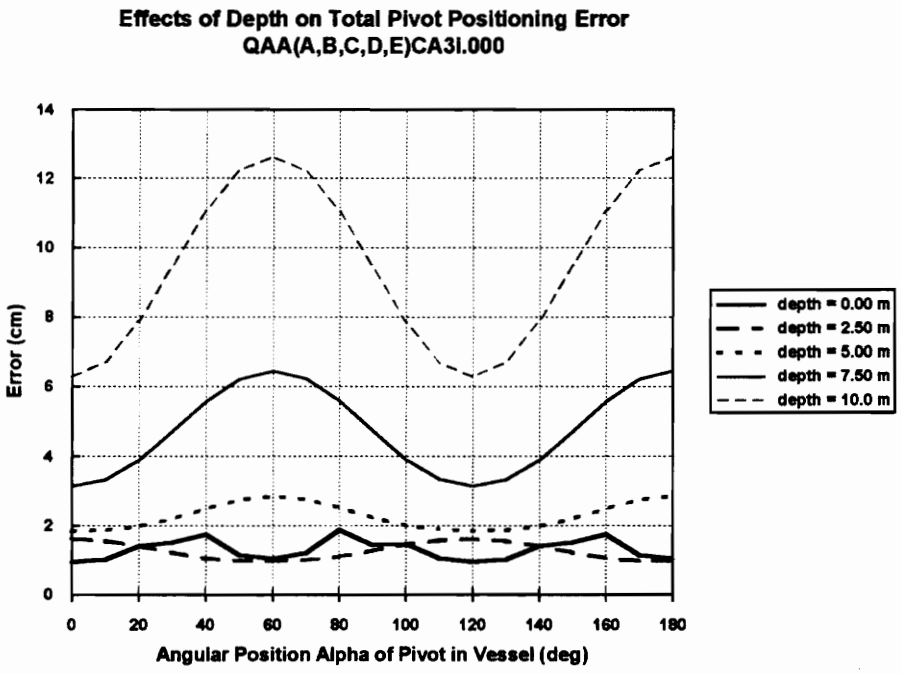


Figure 5.11 Total pivot positioning error at various operating depths from the equivalent velocity system incorporating severe temperature gradients.

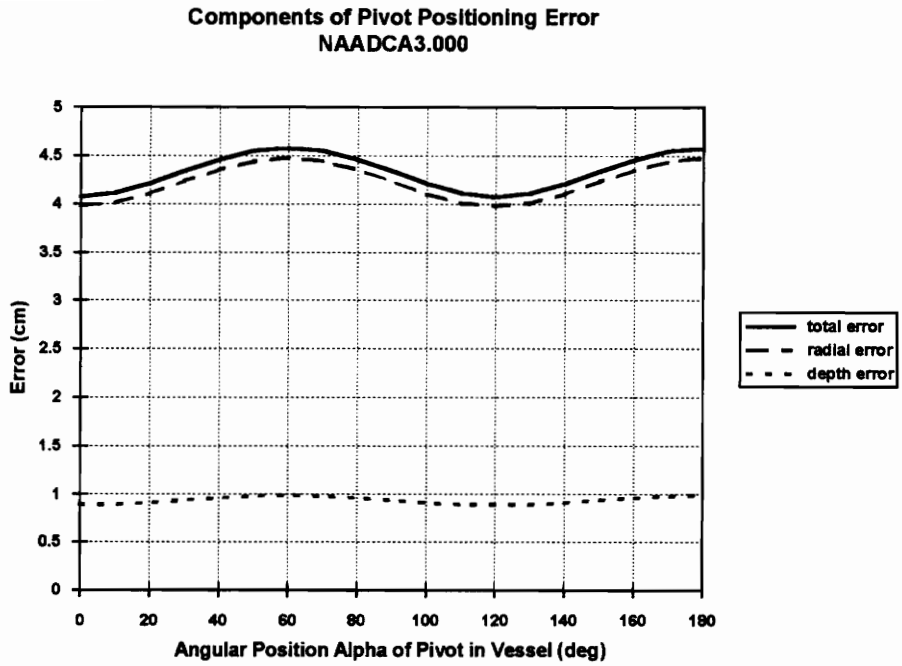


Figure 5.12 Radial and depth components of the total error at 7.5 m depth below the fixed transceiver plane. Results obtained from the interpolated velocity system incorporating severe temperature gradients.

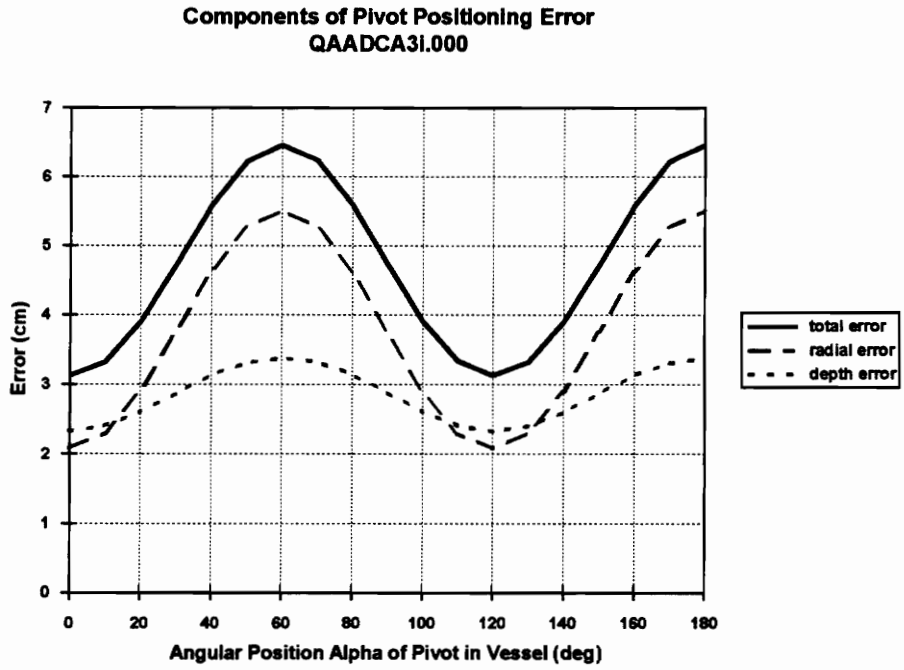


Figure 5.13 Radial and depth components of the total error at 7.5 m depth below the fixed transceiver plane. Results obtained from the equivalent velocity system incorporating severe temperature gradients.

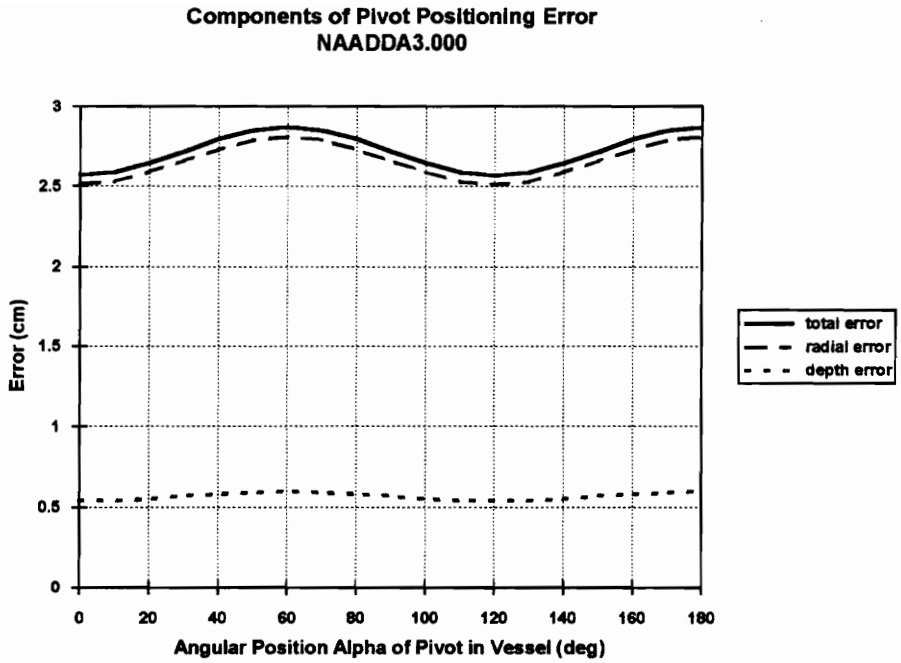


Figure 5.14 Radial and depth components of the total error at 7.5 m depth below the fixed transceiver plane. Obtained from the interpolated velocity system operating at a lower bulk mean water temperature.

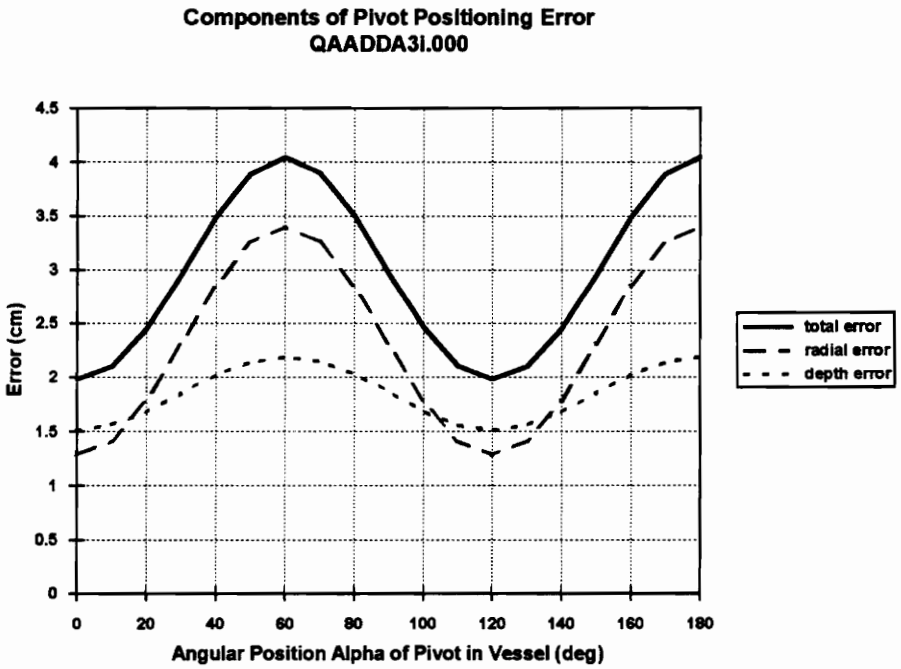


Figure 5.15 Radial and depth components of the total error at 7.5 m depth below the fixed transceiver plane. Obtained from the equivalent velocity system operating at a lower bulk mean water temperature.

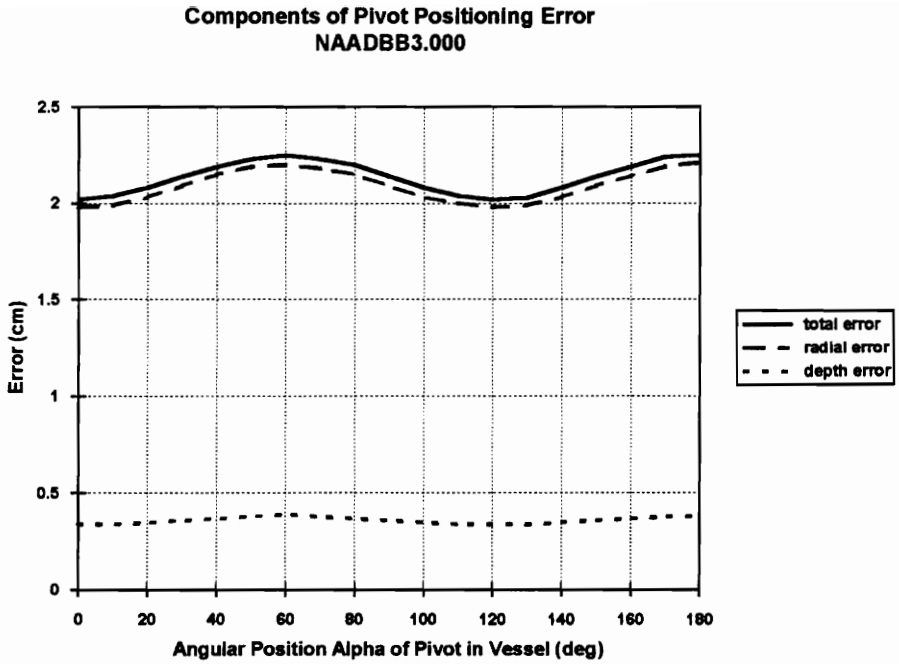


Figure 5.17 Average absolute errors after 10,000 iterations of the interpolated velocity system incorporating full system errors at 7.5 m depth.

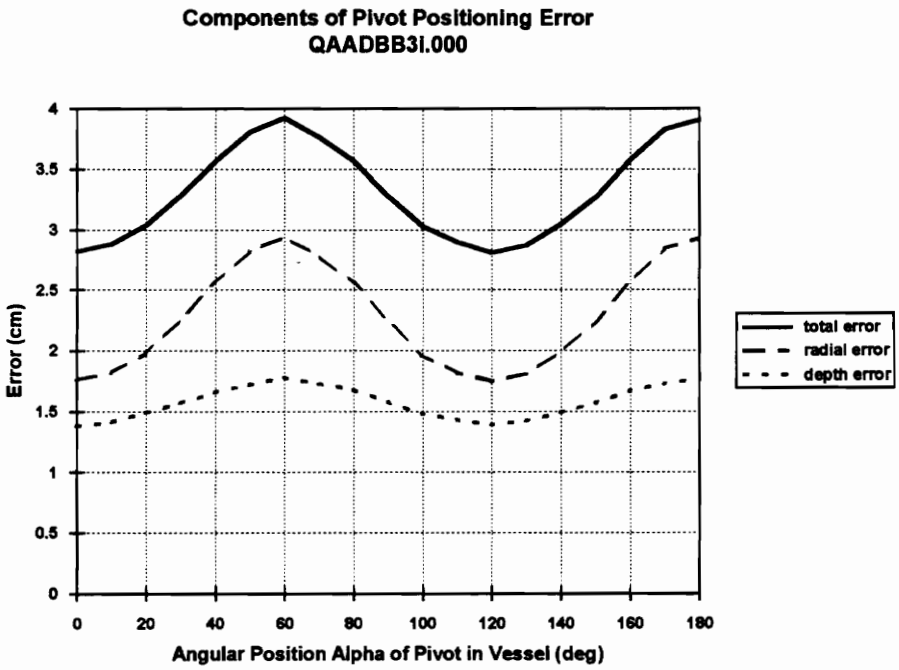


Figure 5.16 Average absolute errors after 10,000 iterations of the equivalent velocity system incorporating full system errors at 7.5 m depth.

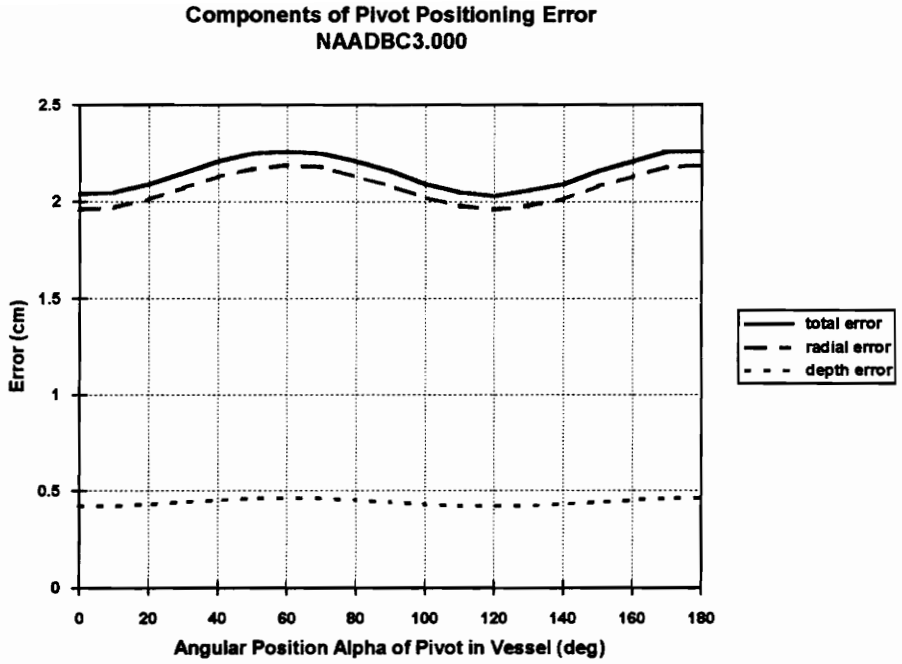


Figure 5.18 Average absolute errors after 10,000 iterations of the interpolated velocity system at 7.5 m depth. Only fixed and mobile transceiver positiong errors were incorporated into the system solution.

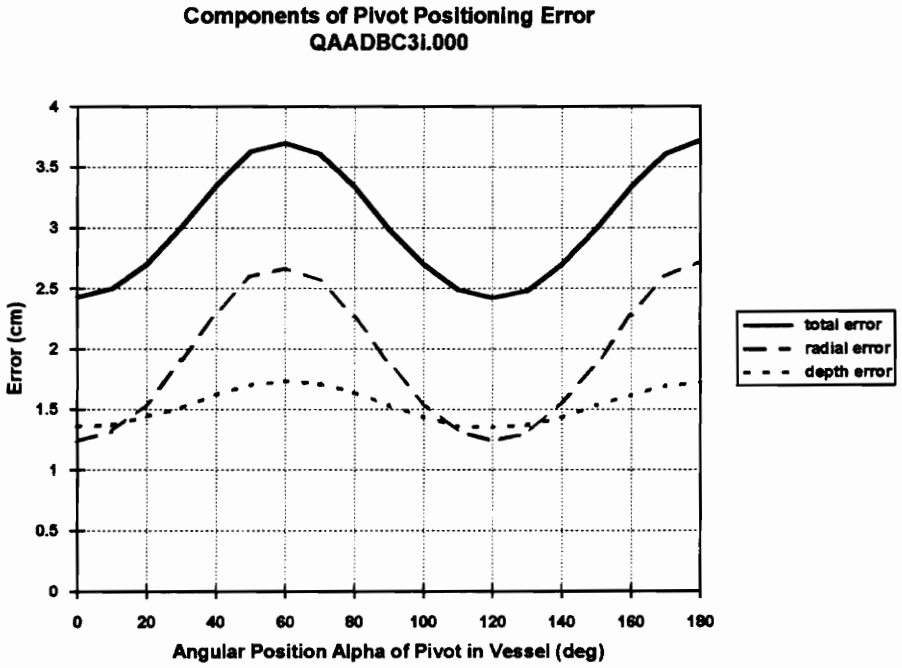


Figure 5.19 Average absolute errors after 10,000 iterations of the equivalent velocity system at 7.5 m depth. Only fixed and mobile transceiver positiong errors were incorporated into the system solution.

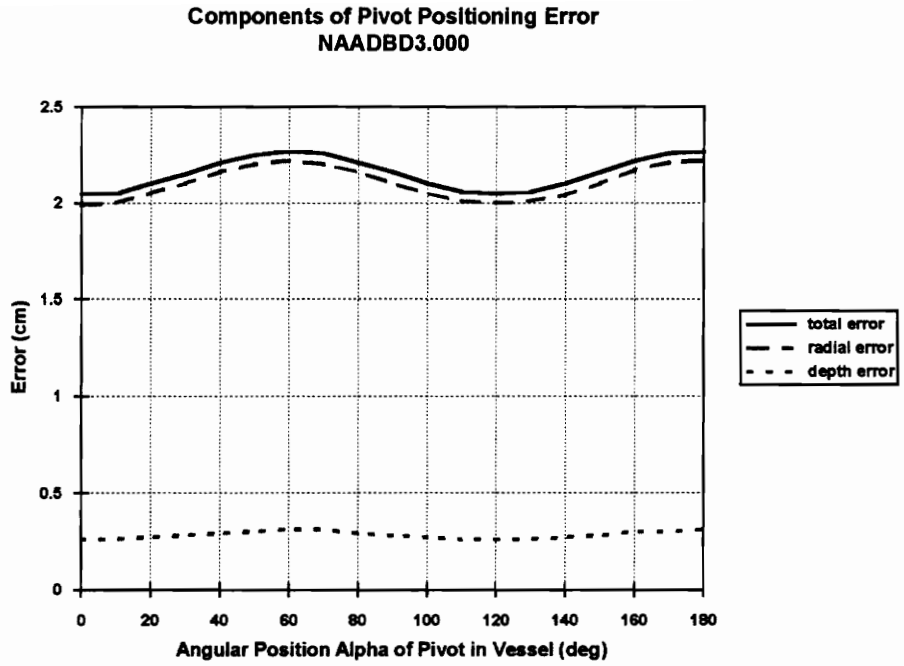


Figure 5.20 Average absolute errors after 10,000 iterations of the interpolated velocity system at 7.5 m depth. Only time of flight errors were incorporated into the system solution process.

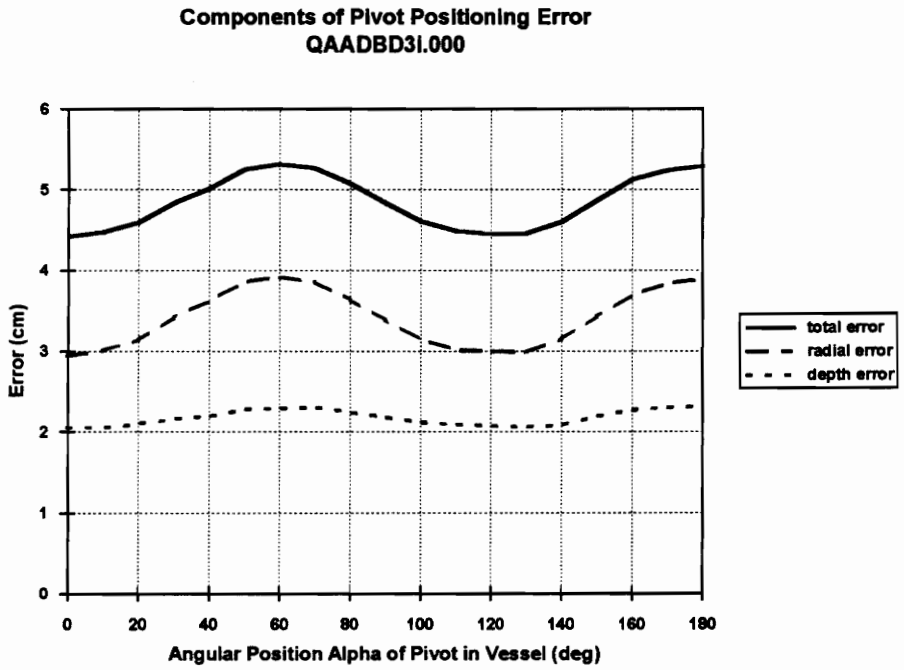


Figure 5.21 Average absolute errors after 10,000 iterations of the equivalent velocity system at 7.5 m depth. Only time of flight errors were incorporated into the system solution process.

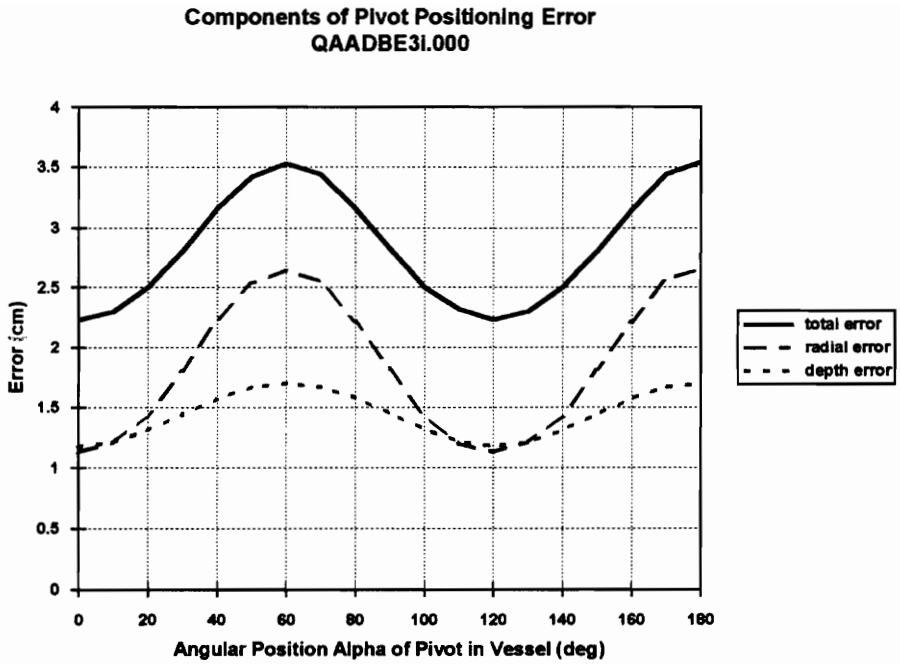


Figure 5.22 Average absolute errors after 10,000 iterations of the equivalent velocity system at 7.5 m depth. Only inclinometer errors were incorporated into the system solution process.

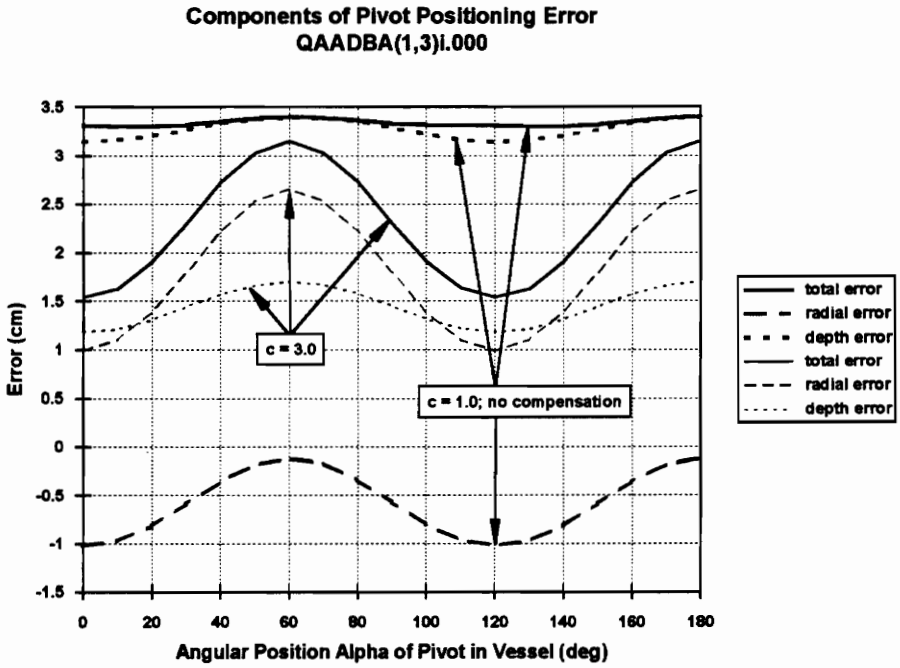


Figure 5.23 Comparison of c = 3 and no velocity compensation at a depth of 7.5 m for the equivalent velocity system.

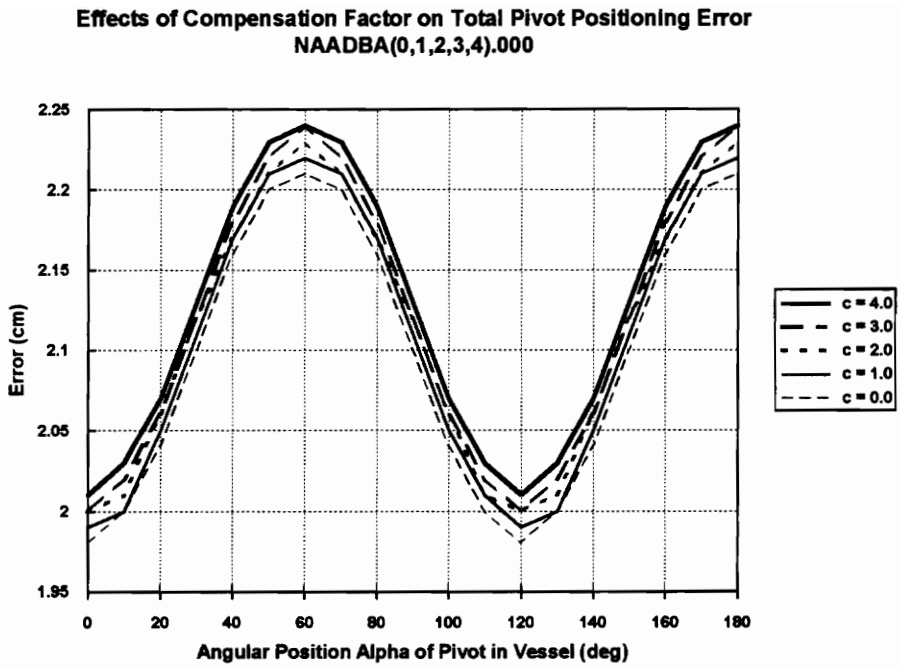


Figure 5.24 Total error at an operating depth of 7.5 m for different compensation factor values. Results obtained from the interpolated velocity system.

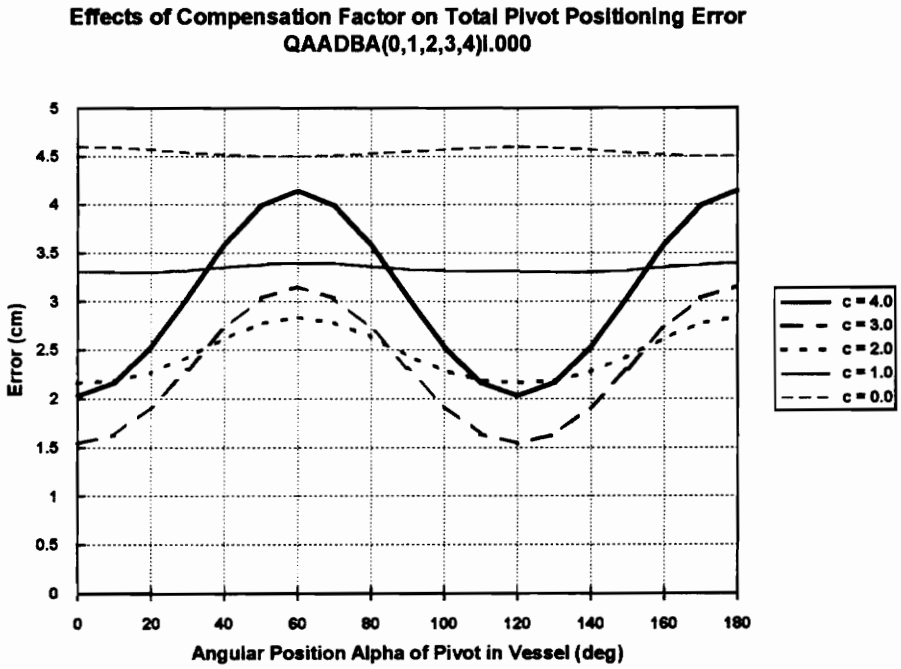


Figure 5.25 Total error at an operating depth of 7.5 m for different compensation factor values. Results obtained from the equivalent velocity system.

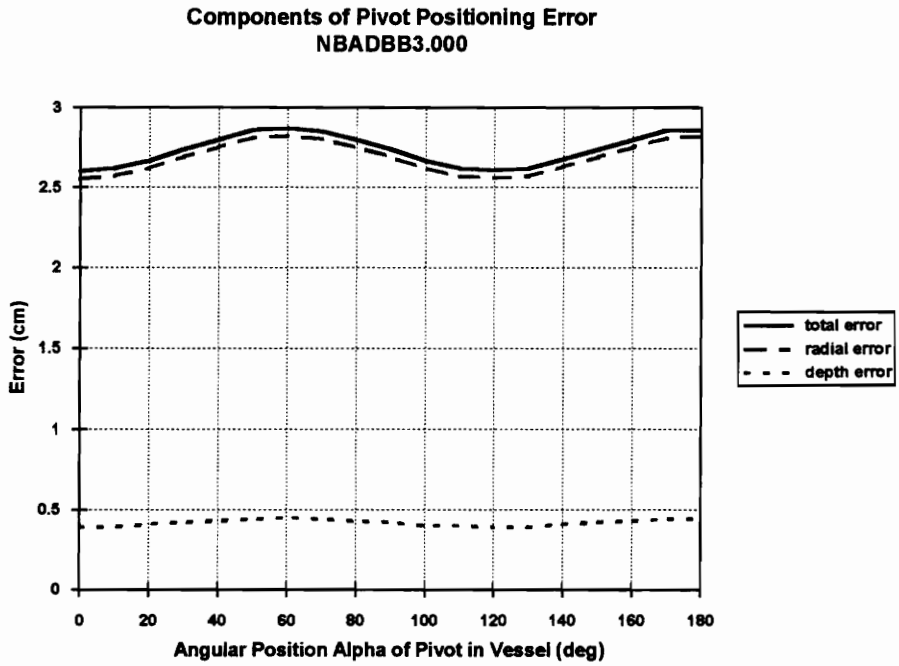


Figure 5.26 Effects of reducing the fixed transceiver separation. Average absolute errors after 10,000 iterations of the interpolated velocity system incorporating full system errors at 7.5 m depth.

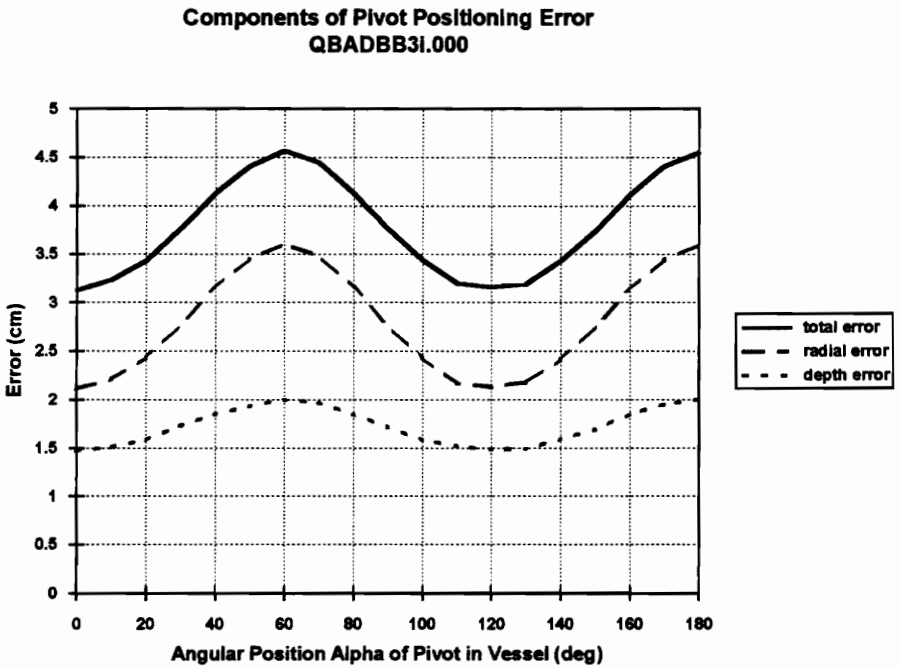


Figure 5.27 Effects of reducing the fixed transceiver separation. Average absolute errors after 10,000 iterations of the equivalent velocity system incorporating full system errors at 7.5 m depth.

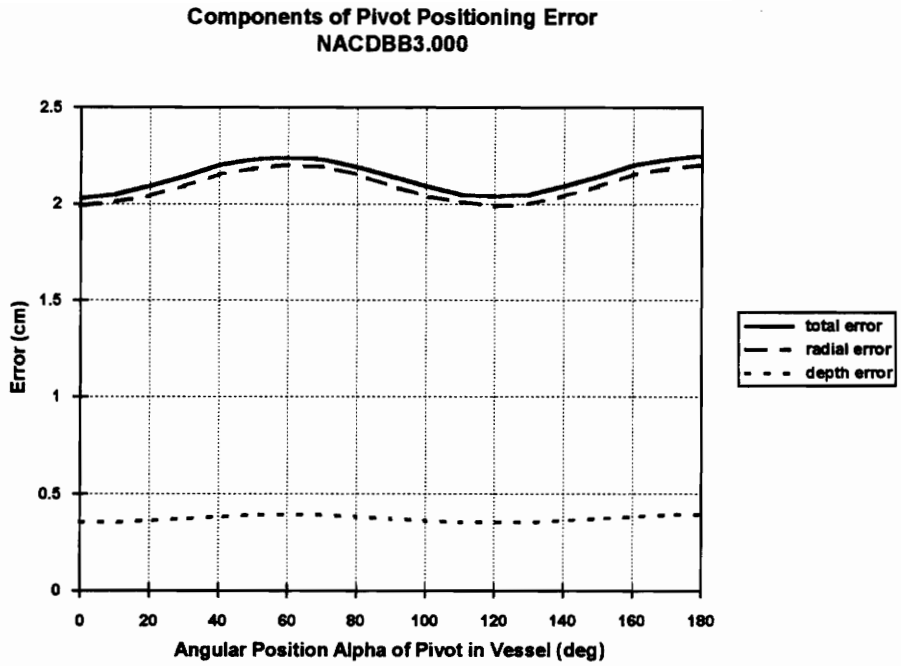


Figure 5.28 Effects of increasing the mobile transceiver separation. Average absolute errors after 10,000 iterations of the interpolated velocity system incorporating full system errors at 7.5 m depth.

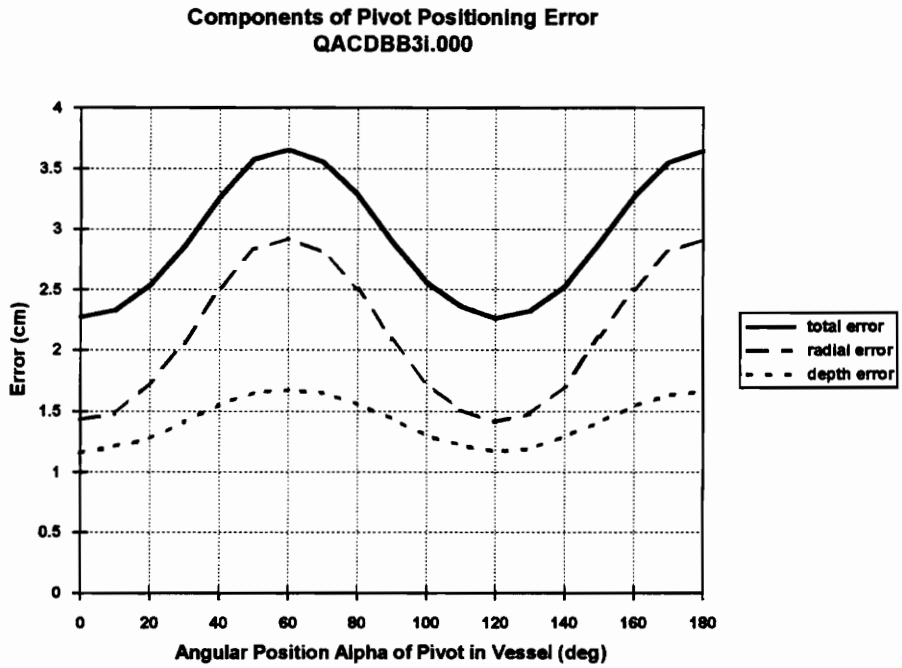


Figure 5.29 Effects of increasing the mobile transceiver separation. Average absolute errors after 10,000 iterations of the equivalent velocity system incorporating full system errors at 7.5 m depth.

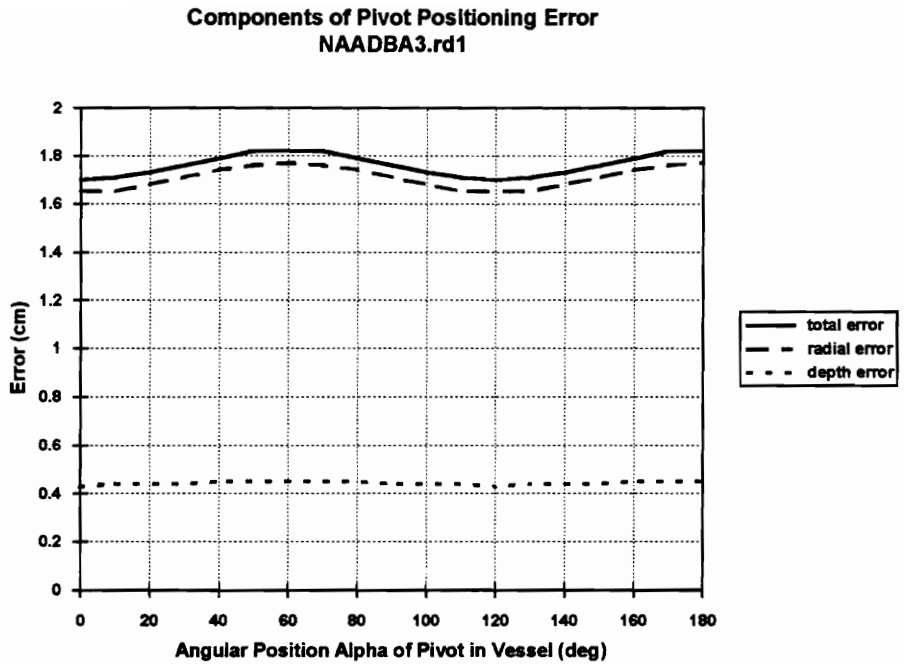


Figure 5.30 Effects of decreasing the operating radius of the mobile transceiver array from the standard 1.5 m to 1.0 m. Results obtained at a depth of 7.5 m for the interpolated velocity system.

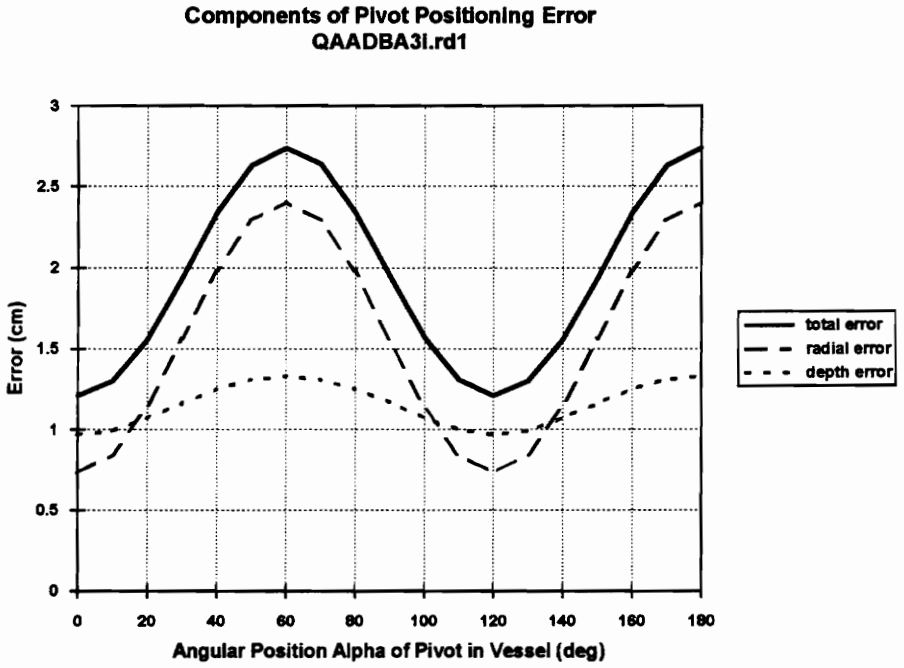


Figure 5.31 Effects of decreasing the operating radius of the mobile transceiver array from the standard 1.5 m to 1.0 m. Results obtained at a depth of 7.5 m for the equivalent velocity system.

6. Future Work

6.1 Testing

The logical next step in the creation of a functioning global positioning system is to use the methods established and discussed within this work to test the Marquest transceivers. Although significant consideration has been given to the theoretical modeling presented here, only actual testing will provide a true measure of the positioning designs' effectiveness.

6.2 System Alterations

6.2.1 Reduction of Fixed Transceivers

By examining the balance of unknowns to system equations, it was shown that less than three fixed transceivers could be used to solve the global positioning problem. This applies to both the equivalent velocity system which uses two inclinometers, and to the interpolated velocity system which solves for all three of the orientation angles. This reduced transceiver possibility may prove useful during operation if, for instance, one of the fixed transceivers should malfunction. The mobile platform might still be able to finish operating within a particular area while the defective transceiver is replaced, saving valuable inspection time.

Additionally, grouping two of the fixed transceivers at a time and then solving the system using this reduced array may prove to be a useful tactic. Each paired group of fixed transceivers could cover one third of the interior surface of the reactor, much like the Earth's global positioning satellites cover selected areas of the planet. Although the error characteristics will be different than with all three fixed transceivers, specific parameters may prove to be more accurate with this method. Initial testing indicates that radial error may be diminished slightly by this technique, particularly for the equivalent velocity system.

6.2.2 Harmonic Mean

For both the equivalent velocity system's compensation scheme and the interpolated velocity system's evaluation of transmission velocities, an averaging technique was utilized. There is some support for the use of the harmonic mean instead, as defined by Eq. 6.1.

The harmonic mean between two values x and y is defined as:

$$HM_{xy} \equiv \frac{2}{\frac{1}{x}+\frac{1}{y}} = \frac{2xy}{x+y} \tag{6.1}$$

The reason behind using the harmonic mean instead of a straight average is best illustrated by the following example:

Given two transceivers separated by a distance ab+bc *where the speed of sound is* v_{ab} *between positions* a *and* b, *and* v_{bc} *between positions* b *and* c*:*

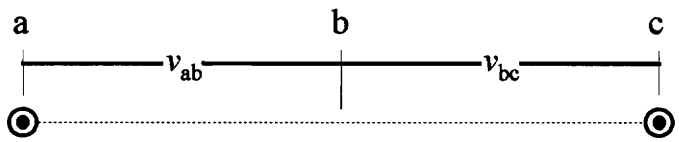


Figure 6.1 Velocities between transceivers.

The transmission velocity between the two transceivers as calculated by an average is:

$$v_{avg} = \frac{v_{ab}+v_{bc}}{2}$$

The transmission velocity calculated by the harmonic mean is:

$$v_{HM} = \frac{2v_{ab}v_{bc}}{v_{ab}+v_{bc}}$$

The true transmission velocity is:

$$v_{true} = \frac{ab+bc}{\frac{ab}{v_{ab}}+\frac{bc}{v_{bc}}} = \frac{v_{ab}v_{bc}(ab+bc)}{v_{ab}bc+v_{bc}ab}$$

If ab = bc *then* $v_{HM} = v_{true}$.

As a numerical example, if ab=bc=4 m, v_{ab}= 1400 m/s, and v_{bc}= 1500 m/s, then the average evaluated transmission velocity would be 1450.00, while the harmonic mean and true transmission velocities would be 1448.28 m/s.

This method was examined for both the equivalent velocity and interpolated velocity systems. The differences between the harmonic mean error results and the averaging

method results were insignificant. However, this may simply be a product of the model itself. The use of the harmonic mean should be included in the future testing of the actual system, and the necessary alterations are incorporated in the computer code of Appendices 4a and 4b to facilitate this examination.

6.2.3 System Combinations

In some cases the error associated with a component from the equivalent velocity system solution may be less than from the interpolated velocity system. This situation may be used to create an overall solution process with is an amalgamation of the two basic systems. For instance, it is possible to solve for positioning utilizing the interpolated velocity scheme, retaining the rotation angles and the depth results. These values could then be fed into the equivalent velocity system (significantly reducing the unknowns), which might then return an angular position and radius that is more accurate than would be obtained from either system alone.

This sequential process may also be carried out within one system. For instance, as is evidenced by Fig. 5.23, the depth error associated with a compensation factor of 3 is relatively small. Also, the radial error associated with a compensation factor of 1 is the smallest contributor of error for that solution. So the system could be run with $c = 3$, retaining the depth results, and then run again with $c = 1$ to obtain a radial measurement.

Initial testing of these methods has shown promise, and should be investigated further.

6.3 Temperature Range Restrictions

Most previous underwater acoustic work has focused on the natural environment, and thus the empirical formulae which calculate the sound velocity (such as Medwin's) have only been accurately defined within the scope of naturally occurring temperatures. Until recently, this restriction has been inconsequential. Now, however, an empirical formula which relates the velocity of sound to temperatures above 35°C is desirable, and has an application.

Some of the system examinations of this work were done at temperatures beyond the specified boundary of Medwin's formula in order to more closely model the temperatures which are expected within the reactor vessel. Therefore, the resulting velocity distributions of the examined systems may not represent the effects of exact linear and parabolic temperature distributions. However, since the goal was simply to approximate

the vessel water characteristics in order to create temperature gradients which would be detrimental to the positioning system, the end result is basically the same.

6.4 Code Optimization

Although considerable effort has been made to produce code which is versatile and easily applied to the functioning global positioning system, it is by no means optimal. There are a number of areas which might benefit from the scrutinization of a more experienced computer programmer. The algorithm which seeds the Newtonian search with an initial guess is one example. It is currently designed for simulation testing, and will need work before it is completely prepared for real operation of the positioning system.

7. Conclusion

Essentially two separate but related positioning methods have been presented in this work. The first was the equivalent velocity scheme which reduced the system unknowns by approximating similar transmission velocities as equal. The second was the interpolated velocity scheme which used calculated nodal velocities to estimate the transmission velocities. Both utilize original approaches to the design of a global positioning system.

The processes which were used to solve these systems were discussed, and the resulting associated errors were examined. In general, the interpolated velocity approach proved to be more accurate and less sensitive to errors. However, as presented in Section 6.2.3 there is promise for the equivalent velocity approach, especially when operating in conjunction with the interpolated velocity system.

Although this work has been largely theoretical, the processes designed here are intended for actual implementation in an industrial environment. To this end, every effort has been made to model the effects of reality and to prepare the computer code for use within the applied system. However, it must be stressed that only actual testing of these global positioning systems will ultimately determine the best techniques and the resulting errors. This testing is slated to begin at the end of May, 1993.

8. References

1. Burdic, William S., 1984, *"Underwater Acoustic System Analysis"*, Prentice-Hall Inc., NJ.

2. Catipovic, Josko A., Freitag, Lee E., 1990, "High Data Rate Acoustic Telemetry for Moving ROVs in a Fading Multipath Shallow Water Enviroment", *Proceedings of the Symposium on Autonomous Underwater Vehicle Technology"*, IEEE, Washington, DC.

3. Cook, Peter L., 1980, "Monitoring the 3-Dimensional Position and 3-Axis Rotation of Submerged Structures", *Proceedings of the European Offshore Petroleum Conference and Exhibition*, Vol. 1, Earls Court, London.

4. Gordon, Alan, 1992, "Use of Laser Scanning System on Mobile Underwater Platforms", *Proceedings of the Symposium on Autonomous Underwater Vehicle Technology*, IEEE, Washington, DC.

5. Hicks, J. A., Mettert, P. M., Morgan, M. J., 1978, "Acoustics Technology for Ocean Engineering", *Ocean Resources Engineering*, Vol. 12, No. 3, pp. 11-22.

6. Himmelblau, David M., 1972, *"Applied Nonlinear Programming"*, McGraw-Hill, Inc.

7. Lee, P.M., Lee J.S., Hong, S.W., 1992, "Experimental Study of a Position Control System for ROV", *Proceedings of the Second International Offshore and Polar Engineering Conference*, San Francisco, CA.

8. McCartney, B. S., 1981, "Underwater Acoustic Positioning Systems: State of the Art and Applications in Deep Water", *International Hydrographic Review*, LVIII (1), Monaco.

9. Medwin, Herman, 1975, "Speed of Sound in Water: A Simple Equation for Realistic Parameters", *The Journal of the Acoustical Society of America*, Vol. 58, No. 4, pp. 1317,1318.

10. Milne, P.H., 1983, *"Underwater Acoustic Positioning Systems"*, E. & F. N. Spon Ltd., London.

11. Milne, P. H., 1986, "Offshore Oil Applications of Sonar Technology", *Marine Technology Society Journal*, Vol. 20, No. 4, pp. 28-37.

12. Mitsubishi Heavy Industries, Ltd., 1991, "Advanced Ultrasonic Testing Machine for Reactor Vessel", Sales and Information Brochure, 5-1, Marunouchi 2-chrome, Chiyoda-ku, Tokyo 100, Japan.

13. Newman, R. V., 1981, "The Status of Underwater Acoustic Positioning", *Offshore Services and Technologies*, Vol. 14, No. 7, pp. 32,34-36.

14. "New Subsea Inspection System Claims Acoustic Positioning Breakthrough", 1981, *Offshore Engineer*, No.34, pp.34.

15. Press, W. H., Flannery, B. P., Teukolsky, S. A., Vetterling, W. T., 1988, *"Numerical Recipes in C, The Art of Scientific Computing"*, Cambridge University Press.

16. Polvani, David G., 1984, "Harmonic Mean Sound Speed is Adequate for Transponder Navigation", *Oceans 84 Proceedings*, IEEE, Vol. 1, pp. 64-68.

17. Roberts, J. L., DeSilvis, D. J., 1986, "New Multimode Acoustic Position-Reference System", *Ocean Industry,* Vol. 21, No. 5, pp. 30-34.

18. Saxena, N. K., Tucker, S. P., Ezequiel, A. M., Kumar, M., 1987, "Intercomparison of Various GPS Receivers for Marine Positioning", *Oceans 87 Proceedings: The Ocean, an International Workplace,* IEEE, Vol. 3, pp. 998-1002.

19. Scheid, Francis, 1968, *"Schaum's Outline of Theory and Problems of Numerical Analysis"*, McGraw-Hill.

20. "Sonic High Accuracy and Positioning System Operation and Software Manual", June 1992, Marquest Group, Inc.

21. Somers, Tagore, 1992, "Mobil / Marquest ROV Dynamic Positioning Trials", *Intervention/ROV 92: A Decade of Achievement,* MTS, pp. 369-373, San Diego, CA.

22. Somers, Tagore, 1993, "Development of an Underwater Vehicle for Nuclear Applications", *Proceedings of the ANS Fifth Topical Meeting on Robots and Remote Systems*, Vol. 2, pp. 707-711, Knoxville, TN.

23. Urick, Robert J., 1975, *"Principles of Underwater Sound"*, Revised Edition, McGraw-Hill, Inc.

24. Vickerman, David C., 1993, "Nuclear Energy and ROVs", *Proceedings of the ANS Fifth Topical Meeting on Robots and Remote Systems*, Vol. 2, pp. 713-723, Knoxville, TN.

25. Wood, Raymond, 1986, "An Underwater Acoustic Position Fixing System for Precise Low Range Use", *Proceedings of the Twelth International Congress on Acoustics"*, Halifax, Nova Scotia, Canada.

Appendix 1
Error Trends and Results

This is a compilation of the observed error trends and characteristics for both the interpolated velocity and the equivalent velocity systems. These results were obtained from an examination of the graphs of chapter 5 and the data files of Appendix 3.

General Trends

1. Largest errors occur when the mobile transceiver array is situated between each of the fixed transceivers: $\alpha = 60^\circ, 180^\circ, 300^\circ$.
2. Smallest errors occur when the mobile transceiver array is situated directly below each of the fixed transceivers: $\alpha = 0^\circ, 120^\circ, 240^\circ$
3. Average Q error is greater than the average N error.
   - N error is more consistent; less variation in error values as the mobile transceiver array circles the interior of the reactor vessel.
   - Q error is occasionally less than N error around $\alpha = 0^\circ, 120^\circ, 240^\circ$.
4. There is less error associated with a vertical orientation of the mobile transceiver array than there is with a horizontal orientation.
   - The horizontal orientation is much worse for Q than for N.
   - Horizontal orientation is especially poor as depth approaches zero (mobile and fixed transceiver planes become aligned) for both Q and N.
5. The positioning error is reduced as the mobile transceiver array operates further from the vessel wall, nearer the vessel center.

Radial and Depth Errors

1. Errors increase with increasing separation between the fixed and mobile transceiver arrays. Q error increases more drastically than the N error.
2. Radial error accounts for the bulk of the N error; N depth positioning is good.
3. Distribution of total error between radial and depth is more even for Q than for N.
   - Radial Q error is usually greater than Q depth error.
   - In the case where the mobile array is oriented horizontally, the total Q error is then mostly due to depth error.

α (Angular Position of Pivot in Vessel) Errors

1. The angular position of the pivot in the vessel was very accurate. The errors were small (on the order of 0.1°) for both N and Q.
2. α error is relatively unaffected by variations in depth.
3. For both N and Q, α error is directly proportional to the temperature gradients; doubling the gradients approximately doubles the error.
4. Relatively unaffected by compensation factor.
5. Only Q α error is significantly affected by mobile transceiver array orientation.

φ and μ Errors

1. Generally small errors (on the order of 0.1°) for N (examined Q systems used inclinometers and so φ and μ were not part of the solution process).
2. Angular errors were much worse (on the order of 1.0°) for horizontal system at zero depth, but small for all other depths.
3. φ error was linearly dependent upon the compensation factor, ranging from -0.8° to 0.2° as c ranged from 0 to 4. Best compensation factor calculated to be 3.2.

θ Errors

1. Not significantly affected by compensation factor or depth for either N or Q.
2. θ errors directly proportional to magnitudes of temperature gradients.
3. Error incorporation affected both N and Q equally, raising the error from less than 0.1° to around 0.25°. Errors in times of flight had the most effect.

Effects of Compensation Factor

1. Total N error is relatively insensitive to compensation factor value.
2. Total Q error is significantly affected by changes in compensation factor.
   - c = 3 resulted in least average error.
   - c = 2 was slightly better for minimizing the maximum Q error.
   - c = 1 (no compensation) results in most of the Q error being attributed to depth instead of radial position as in most cases.

Effects of Added Error

1. N positioning was extremely insensitive to the incorporated errors.
2. The addition of the full system errors decreased the Q positioning error amplitude by 33% but increased its average error value by 43%. The examined individual errors showed that the system is most affected by errors in the time of flight information.
3. Although the average absolute error was not vastly different between the N and Q systems, the standard deviation and worst case errors were significantly better for N.

Effects of Transceiver Spacing

1. Total positioning error increased for N and Q with a decrease in the separation between the fixed transceivers.
2. Total positioning error decreased for Q with an increase in the separation between the mobile transceivers, but remained relatively constant for N.

Appendix 2
Data File Conventions

Each of the data files contains the results of the error analysis for a particular system of acoustic transceivers. Individual systems are characterized by variations in fixed and mobile transceiver configurations, vessel temperature distributions, incorporated errors, etc. These differences are indicated by the data file name (each position in the file name corresponds to a specific system characteristic) and are defined in the following tables:

File Nomenclature

Index Position	Index Reference
1st	Program
2nd	Fixed Transceiver Configuration
3rd	Mobile Transceiver Configuration
4th	Depth of Pivot Model
5th	Temperature Distribution Model
6th	Error Model
7th	Compensation Factor (*0,1,2,3,4* etc.)
8th	Use of inclinometers φ and μ (i = used; blank = unused, angles solved)
Suffix	.000 : normal .hmn : uses harmonic mean instead of average to obtain interpolated velocities .rd1 : operating radius of robot reduced to 1.0 m (from 1.5 m standard)

Programs

Index	Program	Program Description
N	GPSNDVEL	Utilizes evaluated nodal velocities to approximate transmission velocities.
Q	GPSEQVEL	Utilizes equivalent velocity approximation and solves for resulting transmission velocities.

Global Positions of Fixed Transceivers ($s30 = sin30^\circ$, $c30 = cos30^\circ$)

Index	R (m)	F_{0x}	F_{0y}	F_{0z}	F_{1x}	F_{1y}	F_{1z}	F_{2x}	F_{2y}	F_{2z}
A	2.286	R	0.0	0.0	-Rs30	Rc30	0.0	-Rs30	-Rc30	0.0
B	2.000	R	0.0	0.0	-Rs30	Rc30	0.0	-Rs30	-Rc30	0.0

Local Positions of Mobile Transceivers ($s30 = sin30^\circ$, $c30 = cos30^\circ$)

Index	R (m)	M_{0q}	M_{0r}	M_{0s}	M_{1q}	M_{1r}	M_{1s}	M_{2q}	M_{2r}	M_{2s}
A	0.5	0.0	-Rc30	-Rs30	0.0	Rc30	-Rs30	0.0	0.0	R
B	0.5	Rs30	-Rc30	0.0	Rs30	Rc30	0.0	-R	0.0	0.0
C	0.75	0.0	-Rc30	-Rs30	0.0	Rc30	-Rs30	0.0	0.0	R

Depth Models

Index	Depth of Pivot (m)
A	0.0
B	2.5
C	5.0
D	7.5
E	10.0

Temperature Distribution Models

Index	Wall (°C)	Center (°C)	Top (°C)	Bottom (°C)
A	40	40	40	40
B	40	35	40	30
C	45	30	50	30
D	30	25	30	20

Incorporated Error Models

Index	*F* Positions (+/- mm)	*M* Positions (+/- mm)	ToF, Offset (+/- μs)	ToF, Random (+/- μs)	ToF, Depth (+/- μs/m)	Inclinometers (+/- deg)
A	0.0	0.0	0.0	0.0	0.0	0.0
B	1.0	1.0	0.5	1.0	0.1	0.1
C	2.0	2.0	0.0	0.0	0.0	0.0
D	0.0	0.0	1.0	2.0	0.2	0.0
E	0.0	0.0	0.0	0.0	0.0	0.2

Appendix 3
GPS Data Files

The global positioning data files are output from programs GPSNDVEL.CPP (Appendix 4a) and GPSEQVEL.CPP (Appendix 4b). Aside from the first character of the data file name (which indicates whether the data is from GPSNDVEL or GPSEQVEL), the files are listed alphabetically. Each file name represents the specific characteristics of the system which is being evaluated (see Appendix 2), with both files on one page resulting from similar characteristics. Some of these parameters are listed along with the files in order to facilitate comparisons.

File: NAAABA3.000
Program: GPSNDVEL.CPP
GPS utilizing interpolated transmission velocities.
Solving for Px,Py,Pz,Theta,Phi,Mu.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 0.00 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	0.14	0.14	-0.00	0.02	-0.00	0.00	0.01	-0.00	0.14
10.00	0.15	0.14	0.07	0.01	-0.01	-0.01	0.01	0.02	0.15
20.00	0.19	0.13	0.13	-0.00	-0.02	-0.01	0.02	0.03	0.17
30.00	0.22	0.12	0.18	-0.02	-0.03	-0.02	0.03	0.04	0.19
40.00	0.24	0.12	0.20	-0.05	-0.02	-0.01	0.03	0.03	0.22
50.00	0.25	0.12	0.22	-0.06	-0.01	-0.01	0.04	0.02	0.24
60.00	0.26	0.12	0.22	-0.07	0.00	0.00	0.04	-0.00	0.25
70.00	0.25	0.13	0.21	-0.06	0.01	0.01	0.04	-0.02	0.24
80.00	0.24	0.12	0.20	-0.05	0.02	0.02	0.03	-0.03	0.22
90.00	0.22	0.09	0.19	-0.03	0.03	0.02	0.03	-0.04	0.19
100.00	0.19	0.05	0.18	-0.00	0.02	0.02	0.02	-0.03	0.17
110.00	0.15	-0.00	0.15	0.01	0.01	0.01	0.01	-0.02	0.15
120.00	0.14	-0.07	0.12	0.02	-0.00	0.00	0.01	-0.00	0.14
130.00	0.15	-0.13	0.08	0.01	-0.01	-0.01	0.01	0.02	0.15
140.00	0.19	-0.18	0.04	-0.00	-0.02	-0.01	0.02	0.03	0.17
150.00	0.22	-0.22	0.02	-0.02	-0.03	-0.02	0.03	0.04	0.19
160.00	0.24	-0.24	-0.00	-0.05	-0.02	-0.01	0.03	0.03	0.22
170.00	0.25	-0.25	-0.00	-0.06	-0.01	-0.01	0.04	0.02	0.24
180.00	0.26	-0.25	0.00	-0.07	0.00	0.00	0.04	-0.00	0.25

File: QAAABA3i.000
Program: GPSEQVEL.CPP
GPS utilizing equivalent velocity approximation.
Solving for Px,Py,Pz,Theta.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 0.00 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	0.47	-0.47	-0.00	0.03	0.00	0.00	0.00	-0.00	-0.47
10.00	0.51	-0.45	0.25	0.03	0.02	0.00	0.00	0.12	-0.40
20.00	0.71	-0.38	0.60	0.03	0.04	0.00	0.00	0.27	-0.15
30.00	0.89	-0.29	-0.84	0.08	-0.05	0.00	0.00	-0.22	-0.67
40.00	0.89	-0.33	-0.83	0.08	-0.04	0.00	0.00	-0.16	-0.79
50.00	0.59	-0.28	-0.52	0.07	-0.01	0.00	0.00	-0.05	-0.58
60.00	0.53	-0.26	-0.46	0.07	0.00	0.00	0.00	-0.00	-0.53
70.00	0.62	-0.32	-0.52	0.07	0.02	0.00	0.00	0.05	-0.60
80.00	0.95	-0.60	-0.74	0.08	0.05	0.00	0.00	0.18	-0.83
90.00	0.85	-0.54	-0.65	0.07	0.05	0.00	0.00	0.21	-0.65
100.00	0.74	0.74	-0.04	0.02	-0.04	0.00	0.00	-0.28	-0.16
110.00	0.53	0.45	-0.27	0.03	-0.01	0.00	0.00	-0.13	-0.41
120.00	0.47	0.24	-0.41	0.03	0.00	0.00	0.00	-0.00	-0.47
130.00	0.51	0.01	-0.51	0.03	0.02	0.00	0.00	0.12	-0.40
140.00	0.71	-0.33	-0.63	0.03	0.04	0.00	0.00	0.27	-0.15
150.00	0.89	0.87	0.17	0.08	-0.05	0.00	0.00	-0.22	-0.67
160.00	0.89	0.88	0.13	0.08	-0.04	0.00	0.00	-0.16	-0.79
170.00	0.59	0.59	0.02	0.07	-0.01	0.00	0.00	-0.05	-0.58
180.00	0.53	0.53	0.00	0.07	0.00	0.00	0.00	-0.00	-0.53

File: NAAACA3.000
Program: GPSNDVEL.CPP
GPS utilizing interpolated transmission velocities.
Solving for Px,Py,Pz,Theta,Phi,Mu.
Temperatures: Tw = 40.00, Tc = 30.00, Tt = 45.00, Tb = 25.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 0.00 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	0.28	0.27	-0.00	0.03	0.00	0.00	0.02	-0.00	0.27
10.00	0.30	0.27	0.14	0.02	-0.03	-0.02	0.03	0.03	0.29
20.00	0.36	0.25	0.26	-0.01	-0.05	-0.03	0.04	0.06	0.33
30.00	0.43	0.24	0.35	-0.05	-0.05	-0.03	0.05	0.07	0.38
40.00	0.47	0.23	0.40	-0.09	-0.05	-0.03	0.07	0.06	0.44
50.00	0.50	0.24	0.42	-0.12	-0.03	-0.02	0.08	0.04	0.48
60.00	0.51	0.25	0.43	-0.13	0.00	0.00	0.08	-0.00	0.49
70.00	0.50	0.25	0.42	-0.12	0.03	0.02	0.08	-0.04	0.48
80.00	0.47	0.24	0.40	-0.09	0.05	0.03	0.07	-0.06	0.44
90.00	0.43	0.19	0.38	-0.05	0.06	0.03	0.05	-0.07	0.38
100.00	0.37	0.10	0.35	-0.01	0.05	0.03	0.04	-0.06	0.33
110.00	0.30	-0.01	0.30	0.02	0.03	0.02	0.03	-0.04	0.29
120.00	0.28	-0.14	0.24	0.03	0.00	0.00	0.02	-0.00	0.27
130.00	0.30	-0.26	0.16	0.02	-0.03	-0.02	0.03	0.03	0.29
140.00	0.36	-0.35	0.09	-0.01	-0.05	-0.03	0.04	0.06	0.33
150.00	0.43	-0.42	0.03	-0.05	-0.05	-0.03	0.05	0.07	0.38
160.00	0.47	-0.47	-0.00	-0.09	-0.05	-0.03	0.07	0.06	0.44
170.00	0.50	-0.49	-0.01	-0.12	-0.03	-0.02	0.08	0.04	0.48
180.00	0.51	-0.49	0.00	-0.13	0.00	0.00	0.08	-0.00	0.49

File: QAAACA3i.000
Program: GPSEQVEL.CPP
GPS utilizing equivalent velocity approximation.
Solving for Px,Py,Pz,Theta.
Temperatures: Tw = 40.00, Tc = 30.00, Tt = 45.00, Tb = 25.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 0.00 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	0.93	-0.93	-0.00	0.06	0.00	0.00	0.00	-0.00	-0.93
10.00	1.01	-0.88	0.49	0.06	0.04	0.00	0.00	0.24	-0.78
20.00	1.40	-0.74	1.19	0.05	0.08	0.00	0.00	0.53	-0.28
30.00	1.52	-0.56	-1.40	0.14	-0.09	0.00	0.00	-0.36	-1.18
40.00	1.76	-0.65	-1.63	0.15	-0.08	0.00	0.00	-0.32	-1.54
50.00	1.15	-0.55	-1.01	0.13	-0.03	0.00	0.00	-0.09	-1.12
60.00	1.03	-0.50	-0.89	0.13	0.01	0.00	0.00	-0.00	-1.02
70.00	1.20	-0.62	-1.02	0.13	0.04	0.00	0.00	0.09	-1.17
80.00	1.88	-1.18	-1.45	0.15	0.10	0.00	0.00	0.35	-1.63
90.00	1.44	-0.86	-1.15	0.13	0.09	0.00	0.00	0.33	-1.14
100.00	1.48	1.47	-0.06	0.04	-0.08	0.00	0.00	-0.55	-0.31
110.00	1.04	0.89	-0.53	0.06	-0.03	0.00	0.00	-0.25	-0.80
120.00	0.93	0.47	-0.80	0.06	0.00	0.00	0.00	-0.00	-0.93
130.00	1.01	0.01	-1.01	0.06	0.04	0.00	0.00	0.24	-0.78
140.00	1.40	-0.66	-1.24	0.05	0.08	0.00	0.00	0.53	-0.28
150.00	1.52	1.50	0.22	0.14	-0.09	0.00	0.00	-0.36	-1.18
160.00	1.76	1.73	0.25	0.15	-0.08	0.00	0.00	-0.32	-1.54
170.00	1.15	1.15	0.03	0.13	-0.03	0.00	0.00	-0.09	-1.12
180.00	1.03	1.02	0.01	0.13	0.01	0.00	0.00	-0.00	-1.02

File: NAABBA3.000
Program: GPSNDVEL.CPP
GPS utilizing interpolated transmission velocities.
Solving for Px,Py,Pz,Theta,Phi,Mu.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 2.50 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	0.35	0.33	-0.00	0.12	0.01	0.00	-0.04	-0.00	0.33
10.00	0.37	0.33	0.11	0.13	-0.02	0.02	-0.04	0.02	0.34
20.00	0.42	0.33	0.21	0.15	-0.03	0.03	-0.04	0.03	0.38
30.00	0.48	0.32	0.30	0.18	-0.04	0.03	-0.04	0.04	0.43
40.00	0.53	0.32	0.38	0.21	-0.03	0.03	-0.04	0.03	0.48
50.00	0.57	0.30	0.43	0.23	-0.01	0.01	-0.05	0.02	0.52
60.00	0.59	0.27	0.46	0.24	0.01	-0.00	-0.05	-0.00	0.54
70.00	0.57	0.23	0.47	0.23	0.04	-0.02	-0.05	-0.02	0.52
80.00	0.54	0.17	0.46	0.21	0.05	-0.03	-0.05	-0.03	0.48
90.00	0.48	0.10	0.43	0.18	0.06	-0.03	-0.04	-0.04	0.43
100.00	0.42	0.02	0.39	0.15	0.05	-0.03	-0.04	-0.03	0.38
110.00	0.37	-0.07	0.34	0.13	0.03	-0.02	-0.04	-0.02	0.34
120.00	0.35	-0.16	0.29	0.12	0.01	0.00	-0.04	-0.00	0.33
130.00	0.37	-0.26	0.23	0.13	-0.02	0.02	-0.04	0.02	0.34
140.00	0.42	-0.35	0.18	0.15	-0.03	0.03	-0.04	0.03	0.38
150.00	0.48	-0.42	0.13	0.18	-0.04	0.03	-0.04	0.04	0.43
160.00	0.53	-0.48	0.08	0.21	-0.03	0.03	-0.04	0.03	0.48
170.00	0.57	-0.52	0.04	0.23	-0.01	0.01	-0.05	0.02	0.52
180.00	0.59	-0.54	0.00	0.24	0.01	-0.00	-0.05	-0.00	0.54

File: QAABBA3i.000
Program: GPSEQVEL.CPP
GPS utilizing equivalent velocity approximation.
Solving for Px,Py,Pz,Theta.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 2.50 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	0.81	-0.41	-0.01	0.69	0.00	0.00	0.00	-0.00	-0.41
10.00	0.77	-0.39	-0.06	0.67	-0.01	0.00	0.00	0.00	-0.39
20.00	0.70	-0.32	-0.08	0.62	-0.02	0.00	0.00	0.01	-0.33
30.00	0.61	-0.22	-0.06	0.56	-0.02	0.00	0.00	0.02	-0.22
40.00	0.53	-0.12	0.00	0.52	-0.01	0.00	0.00	0.03	-0.09
50.00	0.50	-0.03	0.06	0.49	-0.00	0.00	0.00	0.02	0.03
60.00	0.49	0.02	0.07	0.49	0.01	0.00	0.00	0.00	0.07
70.00	0.51	0.04	0.00	0.50	0.02	0.00	0.00	-0.01	0.02
80.00	0.55	0.04	-0.11	0.54	0.03	0.00	0.00	-0.02	-0.10
90.00	0.64	0.05	-0.23	0.59	0.03	0.00	0.00	-0.02	-0.23
100.00	0.72	0.09	-0.32	0.64	0.02	0.00	0.00	-0.01	-0.33
110.00	0.79	0.15	-0.36	0.68	0.02	0.00	0.00	-0.01	-0.39
120.00	0.81	0.21	-0.35	0.69	0.00	0.00	0.00	-0.00	-0.41
130.00	0.77	0.25	-0.30	0.67	-0.01	0.00	0.00	0.00	-0.39
140.00	0.70	0.23	-0.24	0.62	-0.02	0.00	0.00	0.01	-0.33
150.00	0.61	0.16	-0.17	0.56	-0.02	0.00	0.00	0.02	-0.22
160.00	0.53	0.06	-0.11	0.52	-0.01	0.00	0.00	0.03	-0.09
170.00	0.50	-0.04	-0.06	0.49	-0.00	0.00	0.00	0.02	0.03
180.00	0.49	-0.07	-0.01	0.49	0.01	0.00	0.00	0.00	0.07

File: NAABCA3.000
Program: GPSNDVEL.CPP
GPS utilizing interpolated transmission velocities.
Solving for Px,Py,Pz,Theta,Phi,Mu.
Temperatures: Tw = 40.00, Tc = 30.00, Tt = 45.00, Tb = 25.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 2.50 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	0.69	0.65	-0.00	0.23	0.01	0.00	-0.07	-0.00	0.65
10.00	0.73	0.65	0.22	0.25	-0.04	0.03	-0.07	0.04	0.68
20.00	0.83	0.65	0.42	0.30	-0.07	0.05	-0.08	0.07	0.76
30.00	0.95	0.64	0.60	0.36	-0.08	0.06	-0.08	0.08	0.86
40.00	1.06	0.63	0.75	0.42	-0.06	0.05	-0.09	0.07	0.96
50.00	1.14	0.59	0.86	0.46	-0.02	0.03	-0.09	0.04	1.04
60.00	1.17	0.53	0.92	0.48	0.03	-0.00	-0.10	-0.00	1.07
70.00	1.14	0.45	0.94	0.46	0.07	-0.03	-0.09	-0.04	1.04
80.00	1.07	0.34	0.92	0.42	0.11	-0.06	-0.09	-0.07	0.96
90.00	0.95	0.20	0.86	0.36	0.12	-0.06	-0.08	-0.08	0.86
100.00	0.83	0.04	0.78	0.30	0.10	-0.05	-0.08	-0.07	0.76
110.00	0.73	-0.14	0.68	0.25	0.06	-0.03	-0.07	-0.04	0.68
120.00	0.69	-0.32	0.57	0.23	0.01	0.00	-0.07	-0.00	0.65
130.00	0.73	-0.51	0.46	0.25	-0.04	0.03	-0.07	0.04	0.68
140.00	0.83	-0.69	0.35	0.30	-0.07	0.05	-0.08	0.07	0.76
150.00	0.95	-0.84	0.26	0.36	-0.08	0.06	-0.08	0.08	0.86
160.00	1.06	-0.96	0.17	0.42	-0.06	0.05	-0.09	0.07	0.96
170.00	1.14	-1.04	0.08	0.46	-0.02	0.03	-0.09	0.04	1.04
180.00	1.17	-1.07	0.00	0.48	0.03	-0.00	-0.10	-0.00	1.07

File: QAABCA3i.000
Program: GPSEQVEL.CPP
GPS utilizing equivalent velocity approximation.
Solving for Px,Py,Pz,Theta.
Temperatures: Tw = 40.00, Tc = 30.00, Tt = 45.00, Tb = 25.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 2.50 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	1.61	-0.82	-0.01	1.38	0.01	0.00	0.00	-0.00	-0.82
10.00	1.55	-0.77	-0.12	1.33	-0.02	0.00	0.00	0.01	-0.78
20.00	1.40	-0.64	-0.17	1.23	-0.03	0.00	0.00	0.02	-0.66
30.00	1.21	-0.45	-0.12	1.11	-0.03	0.00	0.00	0.05	-0.45
40.00	1.04	-0.24	0.00	1.02	-0.02	0.00	0.00	0.06	-0.18
50.00	0.97	-0.06	0.12	0.97	-0.00	0.00	0.00	0.05	0.05
60.00	0.97	0.05	0.14	0.96	0.02	0.00	0.00	0.01	0.14
70.00	0.99	0.08	0.01	0.99	0.04	0.00	0.00	-0.03	0.03
80.00	1.09	0.08	-0.22	1.06	0.05	0.00	0.00	-0.04	-0.21
90.00	1.26	0.10	-0.47	1.17	0.06	0.00	0.00	-0.04	-0.47
100.00	1.44	0.18	-0.65	1.28	0.05	0.00	0.00	-0.02	-0.67
110.00	1.57	0.30	-0.73	1.36	0.03	0.00	0.00	-0.01	-0.79
120.00	1.61	0.42	-0.71	1.38	0.01	0.00	0.00	-0.00	-0.82
130.00	1.55	0.49	-0.61	1.33	-0.02	0.00	0.00	0.01	-0.78
140.00	1.40	0.46	-0.47	1.23	-0.03	0.00	0.00	0.02	-0.66
150.00	1.21	0.33	-0.33	1.11	-0.03	0.00	0.00	0.05	-0.45
160.00	1.04	0.12	-0.21	1.02	-0.02	0.00	0.00	0.06	-0.18
170.00	0.97	-0.07	-0.11	0.97	-0.00	0.00	0.00	0.05	0.05
180.00	0.97	-0.14	-0.02	0.96	0.02	0.00	0.00	0.01	0.14

File: NAACBA3.000
Program: GPSNDVEL.CPP
GPS utilizing interpolated transmission velocities.
Solving for Px,Py,Pz,Theta,Phi,Mu.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 5.00 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	0.97	0.93	-0.00	0.26	0.01	0.00	-0.06	-0.00	0.93
10.00	0.99	0.92	0.22	0.27	-0.02	0.02	-0.06	0.02	0.95
20.00	1.03	0.90	0.42	0.28	-0.05	0.03	-0.06	0.03	0.99
30.00	1.09	0.85	0.61	0.30	-0.05	0.03	-0.06	0.04	1.04
40.00	1.14	0.78	0.77	0.31	-0.04	0.03	-0.06	0.03	1.10
50.00	1.18	0.69	0.90	0.32	-0.02	0.02	-0.07	0.02	1.13
60.00	1.20	0.58	0.99	0.33	0.02	0.00	-0.07	-0.00	1.15
70.00	1.18	0.44	1.05	0.32	0.05	-0.01	-0.07	-0.02	1.13
80.00	1.14	0.28	1.06	0.31	0.07	-0.02	-0.06	-0.04	1.10
90.00	1.09	0.11	1.04	0.30	0.08	-0.02	-0.06	-0.04	1.04
100.00	1.03	-0.08	0.99	0.28	0.07	-0.02	-0.06	-0.04	0.99
110.00	0.99	-0.27	0.91	0.27	0.04	-0.01	-0.06	-0.02	0.95
120.00	0.97	-0.47	0.81	0.26	0.01	0.00	-0.06	-0.00	0.93
130.00	0.99	-0.65	0.69	0.27	-0.02	0.02	-0.06	0.02	0.95
140.00	1.03	-0.81	0.56	0.28	-0.05	0.03	-0.06	0.03	0.99
150.00	1.09	-0.95	0.43	0.30	-0.05	0.03	-0.06	0.04	1.04
160.00	1.14	-1.06	0.29	0.31	-0.04	0.03	-0.06	0.03	1.10
170.00	1.18	-1.13	0.15	0.32	-0.02	0.02	-0.07	0.02	1.13
180.00	1.20	-1.15	0.00	0.33	0.02	0.00	-0.07	-0.00	1.15

File: QAACBA3i.000
Program: GPSEQVEL.CPP
GPS utilizing equivalent velocity approximation.
Solving for Px,Py,Pz,Theta.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 5.00 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	0.92	0.08	-0.02	0.91	0.01	0.00	0.00	-0.01	0.08
10.00	0.93	0.12	0.05	0.92	-0.00	0.00	0.00	0.01	0.13
20.00	0.98	0.22	0.16	0.95	-0.01	0.00	0.00	0.03	0.26
30.00	1.09	0.36	0.31	0.98	-0.02	0.00	0.00	0.03	0.47
40.00	1.24	0.48	0.51	1.02	-0.01	0.00	0.00	0.03	0.69
50.00	1.37	0.52	0.70	1.06	-0.00	0.00	0.00	0.02	0.87
60.00	1.42	0.46	0.81	1.07	0.01	0.00	0.00	0.00	0.94
70.00	1.38	0.34	0.80	1.07	0.03	0.00	0.00	-0.02	0.87
80.00	1.26	0.20	0.67	1.04	0.04	0.00	0.00	-0.03	0.70
90.00	1.12	0.10	0.47	1.01	0.03	0.00	0.00	-0.04	0.47
100.00	1.01	0.05	0.28	0.96	0.03	0.00	0.00	-0.04	0.27
110.00	0.94	0.02	0.15	0.93	0.02	0.00	0.00	-0.03	0.13
120.00	0.92	-0.02	0.08	0.91	0.01	0.00	0.00	-0.01	0.08
130.00	0.93	-0.11	0.08	0.92	-0.00	0.00	0.00	0.01	0.13
140.00	0.98	-0.25	0.12	0.95	-0.01	0.00	0.00	0.03	0.26
150.00	1.09	-0.45	0.16	0.98	-0.02	0.00	0.00	0.03	0.47
160.00	1.24	-0.68	0.16	1.02	-0.01	0.00	0.00	0.03	0.69
170.00	1.37	-0.86	0.10	1.06	-0.00	0.00	0.00	0.02	0.87
180.00	1.42	-0.94	-0.01	1.07	0.01	0.00	0.00	0.00	0.94

File: NAACCA3.000
Program: GPSNDVEL.CPP
GPS utilizing interpolated transmission velocities.
Solving for Px,Py,Pz,Theta,Phi,Mu.
Temperatures: Tw = 40.00, Tc = 30.00, Tt = 45.00, Tb = 25.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 5.00 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	1.94	1.87	-0.00	0.53	0.02	0.01	-0.12	-0.00	1.87
10.00	1.98	1.85	0.44	0.54	-0.05	0.03	-0.12	0.04	1.90
20.00	2.07	1.80	0.86	0.57	-0.10	0.05	-0.12	0.07	1.98
30.00	2.19	1.70	1.24	0.60	-0.11	0.06	-0.12	0.08	2.09
40.00	2.30	1.57	1.56	0.63	-0.09	0.05	-0.12	0.07	2.21
50.00	2.38	1.39	1.82	0.66	-0.04	0.03	-0.13	0.04	2.29
60.00	2.41	1.16	2.01	0.67	0.04	0.00	-0.13	-0.00	2.32
70.00	2.38	0.89	2.11	0.66	0.10	-0.03	-0.13	-0.04	2.29
80.00	2.30	0.57	2.14	0.63	0.15	-0.05	-0.12	-0.07	2.21
90.00	2.19	0.22	2.09	0.60	0.17	-0.05	-0.12	-0.08	2.09
100.00	2.07	-0.15	1.98	0.57	0.15	-0.04	-0.12	-0.07	1.98
110.00	1.98	-0.54	1.82	0.54	0.09	-0.02	-0.12	-0.04	1.90
120.00	1.94	-0.93	1.62	0.53	0.02	0.01	-0.12	-0.00	1.87
130.00	1.98	-1.30	1.38	0.54	-0.05	0.03	-0.12	0.04	1.90
140.00	2.07	-1.64	1.13	0.57	-0.10	0.05	-0.12	0.07	1.98
150.00	2.19	-1.92	0.86	0.60	-0.11	0.06	-0.12	0.08	2.09
160.00	2.30	-2.14	0.58	0.63	-0.09	0.05	-0.12	0.07	2.21
170.00	2.38	-2.27	0.29	0.66	-0.04	0.03	-0.13	0.04	2.29
180.00	2.41	-2.32	0.00	0.67	0.04	0.00	-0.13	-0.00	2.32

File: QAACCA3i.000
Program: GPSEQVEL.CPP
GPS utilizing equivalent velocity approximation.
Solving for Px,Py,Pz,Theta.
Temperatures: Tw = 40.00, Tc = 30.00, Tt = 45.00, Tb = 25.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 5.00 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	1.84	0.17	-0.04	1.83	0.01	0.00	0.00	-0.01	0.17
10.00	1.86	0.25	0.11	1.84	-0.01	0.00	0.00	0.02	0.26
20.00	1.97	0.46	0.32	1.89	-0.03	0.00	0.00	0.05	0.54
30.00	2.19	0.74	0.63	1.96	-0.04	0.00	0.00	0.07	0.96
40.00	2.48	0.98	1.03	2.03	-0.03	0.00	0.00	0.06	1.41
50.00	2.74	1.06	1.42	2.09	-0.01	0.00	0.00	0.04	1.77
60.00	2.85	0.95	1.66	2.12	0.03	0.00	0.00	0.00	1.91
70.00	2.76	0.69	1.64	2.11	0.06	0.00	0.00	-0.03	1.78
80.00	2.52	0.41	1.37	2.07	0.08	0.00	0.00	-0.06	1.42
90.00	2.23	0.21	0.97	2.00	0.08	0.00	0.00	-0.08	0.97
100.00	2.01	0.09	0.58	1.92	0.06	0.00	0.00	-0.07	0.55
110.00	1.89	0.03	0.30	1.86	0.04	0.00	0.00	-0.05	0.27
120.00	1.84	-0.05	0.16	1.83	0.01	0.00	0.00	-0.01	0.17
130.00	1.86	-0.22	0.16	1.84	-0.01	0.00	0.00	0.02	0.26
140.00	1.97	-0.50	0.24	1.89	-0.03	0.00	0.00	0.05	0.54
150.00	2.19	-0.92	0.33	1.96	-0.04	0.00	0.00	0.07	0.96
160.00	2.48	-1.38	0.33	2.03	-0.03	0.00	0.00	0.06	1.41
170.00	2.74	-1.76	0.21	2.09	-0.01	0.00	0.00	0.04	1.77
180.00	2.85	-1.91	-0.01	2.12	0.03	0.00	0.00	0.00	1.91

File: NAADBA0.000
Program: GPSNDVEL.CPP
GPS utilizing interpolated transmission velocities.
Solving for Px,Py,Pz,Theta,Phi,Mu.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 0.00
Platform pivot at radius of 1.50 m and depth of 7.50 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	1.98	1.94	-0.00	0.39	0.08	0.01	-0.79	-0.00	1.94
10.00	2.00	1.92	0.39	0.40	0.04	0.02	-0.79	0.02	1.96
20.00	2.04	1.84	0.77	0.41	0.02	0.03	-0.79	0.04	2.00
30.00	2.10	1.72	1.12	0.42	0.01	0.03	-0.79	0.04	2.06
40.00	2.16	1.56	1.43	0.43	0.02	0.03	-0.79	0.04	2.11
50.00	2.20	1.34	1.69	0.44	0.05	0.02	-0.79	0.02	2.15
60.00	2.21	1.09	1.88	0.44	0.09	0.01	-0.79	-0.00	2.17
70.00	2.20	0.79	2.00	0.44	0.13	-0.00	-0.79	-0.02	2.15
80.00	2.16	0.46	2.06	0.43	0.16	-0.01	-0.79	-0.04	2.11
90.00	2.10	0.11	2.06	0.42	0.16	-0.01	-0.79	-0.04	2.06
100.00	2.04	-0.25	1.99	0.41	0.15	-0.01	-0.79	-0.04	2.00
110.00	2.00	-0.61	1.86	0.40	0.12	-0.00	-0.79	-0.02	1.96
120.00	1.98	-0.97	1.68	0.39	0.08	0.01	-0.79	-0.00	1.94
130.00	2.00	-1.30	1.46	0.40	0.04	0.02	-0.79	0.02	1.96
140.00	2.04	-1.59	1.21	0.41	0.02	0.03	-0.79	0.04	2.00
150.00	2.10	-1.84	0.93	0.42	0.01	0.03	-0.79	0.04	2.06
160.00	2.16	-2.02	0.63	0.43	0.02	0.03	-0.79	0.04	2.11
170.00	2.20	-2.13	0.32	0.44	0.05	0.02	-0.79	0.02	2.15
180.00	2.21	-2.17	0.00	0.44	0.09	0.01	-0.79	-0.00	2.17

File: QAADBA0i.000
Program: GPSEQVEL.CPP
GPS utilizing equivalent velocity approximation.
Solving for Px,Py,Pz,Theta.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 0.00
Platform pivot at radius of 1.50 m and depth of 7.50 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	4.60	-2.02	-0.05	4.13	0.01	0.00	0.00	-0.02	-2.02
10.00	4.59	-1.95	-0.39	4.13	-0.00	0.00	0.00	-0.02	-1.99
20.00	4.57	-1.77	-0.68	4.15	-0.01	0.00	0.00	-0.01	-1.90
30.00	4.54	-1.52	-0.91	4.18	-0.01	0.00	0.00	-0.01	-1.77
40.00	4.52	-1.23	-1.07	4.21	-0.01	0.00	0.00	-0.01	-1.63
50.00	4.50	-0.96	-1.19	4.24	-0.00	0.00	0.00	-0.01	-1.53
60.00	4.50	-0.72	-1.30	4.25	0.01	0.00	0.00	-0.01	-1.48
70.00	4.51	-0.49	-1.43	4.25	0.03	0.00	0.00	-0.01	-1.51
80.00	4.53	-0.25	-1.58	4.23	0.03	0.00	0.00	-0.01	-1.60
90.00	4.55	0.04	-1.73	4.21	0.03	0.00	0.00	-0.02	-1.73
100.00	4.57	0.38	-1.83	4.17	0.03	0.00	0.00	-0.02	-1.87
110.00	4.59	0.73	-1.84	4.14	0.02	0.00	0.00	-0.02	-1.97
120.00	4.60	1.05	-1.72	4.13	0.01	0.00	0.00	-0.02	-2.02
130.00	4.59	1.31	-1.50	4.13	-0.00	0.00	0.00	-0.02	-1.99
140.00	4.57	1.48	-1.19	4.15	-0.01	0.00	0.00	-0.01	-1.90
150.00	4.54	1.55	-0.86	4.18	-0.01	0.00	0.00	-0.01	-1.77
160.00	4.52	1.55	-0.53	4.21	-0.01	0.00	0.00	-0.01	-1.63
170.00	4.50	1.51	-0.24	4.24	-0.00	0.00	0.00	-0.01	-1.53
180.00	4.50	1.48	0.03	4.25	0.01	0.00	0.00	-0.01	-1.48

File: NAADBA1.000
Program: GPSNDVEL.CPP
GPS utilizing interpolated transmission velocities.
Solving for Px,Py,Pz,Theta,Phi,Mu.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 1.00
Platform pivot at radius of 1.50 m and depth of 7.50 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	1.99	1.95	-0.00	0.40	0.06	0.01	-0.55	-0.00	1.95
10.00	2.00	1.92	0.40	0.40	0.02	0.02	-0.55	0.02	1.96
20.00	2.05	1.85	0.78	0.41	-0.01	0.03	-0.55	0.04	2.00
30.00	2.11	1.73	1.13	0.42	-0.01	0.03	-0.55	0.04	2.06
40.00	2.17	1.56	1.44	0.44	-0.00	0.03	-0.55	0.04	2.12
50.00	2.21	1.35	1.69	0.44	0.03	0.02	-0.55	0.02	2.16
60.00	2.22	1.09	1.88	0.45	0.07	0.01	-0.55	-0.00	2.18
70.00	2.21	0.79	2.01	0.44	0.10	-0.00	-0.55	-0.02	2.16
80.00	2.17	0.47	2.07	0.44	0.13	-0.01	-0.55	-0.04	2.12
90.00	2.11	0.11	2.06	0.42	0.14	-0.01	-0.55	-0.04	2.06
100.00	2.05	-0.25	1.99	0.41	0.13	-0.01	-0.55	-0.04	2.00
110.00	2.01	-0.62	1.86	0.40	0.10	-0.00	-0.55	-0.02	1.96
120.00	1.99	-0.97	1.69	0.40	0.06	0.01	-0.55	-0.00	1.95
130.00	2.00	-1.30	1.47	0.40	0.02	0.02	-0.55	0.02	1.96
140.00	2.05	-1.60	1.21	0.41	-0.01	0.03	-0.55	0.04	2.00
150.00	2.11	-1.84	0.93	0.42	-0.01	0.03	-0.55	0.04	2.06
160.00	2.17	-2.02	0.63	0.44	-0.00	0.03	-0.55	0.04	2.12
170.00	2.21	-2.14	0.32	0.44	0.03	0.02	-0.55	0.02	2.16
180.00	2.22	-2.18	0.00	0.45	0.07	0.01	-0.55	-0.00	2.18

File: QAADBA1i.000
Program: GPSEQVEL.CPP
GPS utilizing equivalent velocity approximation.
Solving for Px,Py,Pz,Theta.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 1.00
Platform pivot at radius of 1.50 m and depth of 7.50 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	3.31	-1.02	-0.05	3.14	0.01	0.00	0.00	-0.02	-1.02
10.00	3.30	-0.95	-0.20	3.16	-0.00	0.00	0.00	-0.01	-0.97
20.00	3.30	-0.76	-0.29	3.20	-0.01	0.00	0.00	-0.01	-0.82
30.00	3.32	-0.51	-0.31	3.26	-0.02	0.00	0.00	-0.00	-0.59
40.00	3.35	-0.27	-0.24	3.33	-0.01	0.00	0.00	-0.01	-0.36
50.00	3.38	-0.11	-0.16	3.37	-0.00	0.00	0.00	-0.01	-0.19
60.00	3.40	-0.04	-0.12	3.39	0.01	0.00	0.00	-0.01	-0.12
70.00	3.39	-0.02	-0.18	3.38	0.03	0.00	0.00	-0.01	-0.17
80.00	3.36	-0.01	-0.34	3.35	0.04	0.00	0.00	-0.02	-0.34
90.00	3.33	0.07	-0.57	3.28	0.04	0.00	0.00	-0.03	-0.57
100.00	3.32	0.22	-0.77	3.22	0.03	0.00	0.00	-0.03	-0.80
110.00	3.31	0.40	-0.88	3.16	0.02	0.00	0.00	-0.03	-0.96
120.00	3.31	0.56	-0.86	3.14	0.01	0.00	0.00	-0.02	-1.02
130.00	3.30	0.65	-0.72	3.16	-0.00	0.00	0.00	-0.01	-0.97
140.00	3.30	0.63	-0.51	3.20	-0.01	0.00	0.00	-0.01	-0.82
150.00	3.32	0.52	-0.29	3.26	-0.02	0.00	0.00	-0.00	-0.59
160.00	3.35	0.35	-0.11	3.33	-0.01	0.00	0.00	-0.01	-0.36
170.00	3.38	0.19	-0.01	3.37	-0.00	0.00	0.00	-0.01	-0.19
180.00	3.40	0.12	0.03	3.39	0.01	0.00	0.00	-0.01	-0.12

File: NAADBA2.000
Program: GPSNDVEL.CPP
GPS utilizing interpolated transmission velocities.
Solving for Px,Py,Pz,Theta,Phi,Mu.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 2.00
Platform pivot at radius of 1.50 m and depth of 7.50 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	2.00	1.95	-0.00	0.41	0.03	0.01	-0.30	-0.00	1.95
10.00	2.01	1.93	0.40	0.41	-0.00	0.02	-0.30	0.02	1.97
20.00	2.06	1.86	0.78	0.42	-0.03	0.03	-0.30	0.04	2.01
30.00	2.12	1.73	1.13	0.43	-0.04	0.03	-0.30	0.04	2.07
40.00	2.17	1.57	1.44	0.44	-0.03	0.03	-0.30	0.04	2.12
50.00	2.21	1.35	1.70	0.45	0.00	0.02	-0.30	0.02	2.17
60.00	2.23	1.09	1.89	0.46	0.04	0.01	-0.30	-0.00	2.18
70.00	2.21	0.80	2.02	0.45	0.08	-0.00	-0.30	-0.02	2.17
80.00	2.17	0.47	2.08	0.44	0.11	-0.01	-0.30	-0.04	2.13
90.00	2.12	0.11	2.07	0.43	0.12	-0.02	-0.30	-0.04	2.07
100.00	2.06	-0.25	2.00	0.42	0.10	-0.01	-0.30	-0.04	2.01
110.00	2.01	-0.62	1.87	0.41	0.07	-0.00	-0.30	-0.02	1.97
120.00	2.00	-0.98	1.69	0.41	0.03	0.01	-0.30	-0.00	1.95
130.00	2.01	-1.31	1.47	0.41	-0.00	0.02	-0.30	0.02	1.97
140.00	2.06	-1.60	1.22	0.42	-0.03	0.03	-0.30	0.04	2.01
150.00	2.12	-1.85	0.94	0.43	-0.04	0.03	-0.30	0.04	2.07
160.00	2.17	-2.03	0.64	0.44	-0.03	0.03	-0.30	0.04	2.12
170.00	2.21	-2.14	0.32	0.45	0.00	0.02	-0.30	0.02	2.17
180.00	2.23	-2.18	0.00	0.46	0.04	0.01	-0.30	-0.00	2.18

File: QAADBA2I.000
Program: GPSEQVEL.CPP
GPS utilizing equivalent velocity approximation.
Solving for Px,Py,Pz,Theta.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 2.00
Platform pivot at radius of 1.50 m and depth of 7.50 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	2.16	-0.02	-0.06	2.16	0.01	0.00	0.00	-0.02	-0.02
10.00	2.18	0.06	-0.01	2.18	-0.00	0.00	0.00	-0.01	0.06
20.00	2.27	0.26	0.10	2.25	-0.01	0.00	0.00	0.00	0.28
30.00	2.43	0.51	0.31	2.35	-0.02	0.00	0.00	0.00	0.59
40.00	2.61	0.70	0.59	2.45	-0.02	0.00	0.00	0.00	0.92
50.00	2.77	0.76	0.89	2.52	-0.00	0.00	0.00	-0.00	1.16
60.00	2.84	0.65	1.08	2.54	0.02	0.00	0.00	-0.01	1.26
70.00	2.78	0.45	1.09	2.52	0.03	0.00	0.00	-0.02	1.17
80.00	2.63	0.24	0.91	2.46	0.04	0.00	0.00	-0.03	0.93
90.00	2.44	0.10	0.61	2.37	0.04	0.00	0.00	-0.04	0.61
100.00	2.29	0.05	0.30	2.27	0.03	0.00	0.00	-0.04	0.29
110.00	2.19	0.06	0.09	2.19	0.02	0.00	0.00	-0.03	0.06
120.00	2.16	0.06	0.01	2.16	0.01	0.00	0.00	-0.02	-0.02
130.00	2.18	-0.02	0.06	2.18	-0.00	0.00	0.00	-0.01	0.06
140.00	2.27	-0.22	0.18	2.25	-0.01	0.00	0.00	0.00	0.28
150.00	2.43	-0.52	0.29	2.35	-0.02	0.00	0.00	0.00	0.59
160.00	2.61	-0.87	0.31	2.45	-0.02	0.00	0.00	0.00	0.92
170.00	2.77	-1.15	0.21	2.52	-0.00	0.00	0.00	-0.00	1.16
180.00	2.84	-1.26	0.03	2.54	0.02	0.00	0.00	-0.01	1.26

File: NAADBA3.000
Program: GPSNDVEL.CPP
GPS utilizing interpolated transmission velocities.
Solving for Px,Py,Pz,Theta,Phi,Mu.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 7.50 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	2.00	1.96	-0.00	0.42	0.01	0.01	-0.06	-0.00	1.96
10.00	2.02	1.94	0.40	0.42	-0.03	0.02	-0.06	0.02	1.98
20.00	2.06	1.86	0.78	0.43	-0.05	0.02	-0.06	0.04	2.02
30.00	2.12	1.74	1.13	0.44	-0.06	0.03	-0.06	0.04	2.07
40.00	2.18	1.57	1.44	0.45	-0.05	0.02	-0.06	0.04	2.13
50.00	2.22	1.35	1.70	0.46	-0.02	0.02	-0.06	0.02	2.17
60.00	2.24	1.10	1.89	0.46	0.02	0.00	-0.06	-0.00	2.19
70.00	2.22	0.80	2.02	0.46	0.06	-0.01	-0.06	-0.02	2.17
80.00	2.18	0.47	2.08	0.45	0.08	-0.01	-0.06	-0.04	2.13
90.00	2.12	0.11	2.07	0.44	0.09	-0.02	-0.06	-0.04	2.07
100.00	2.06	-0.25	2.00	0.43	0.08	-0.01	-0.06	-0.04	2.02
110.00	2.02	-0.62	1.88	0.42	0.05	-0.00	-0.06	-0.02	1.98
120.00	2.00	-0.98	1.70	0.42	0.01	0.01	-0.06	-0.00	1.96
130.00	2.02	-1.31	1.48	0.42	-0.03	0.02	-0.06	0.02	1.98
140.00	2.06	-1.61	1.22	0.43	-0.05	0.02	-0.06	0.04	2.02
150.00	2.12	-1.85	0.94	0.44	-0.06	0.03	-0.06	0.04	2.07
160.00	2.18	-2.04	0.64	0.45	-0.05	0.02	-0.06	0.04	2.13
170.00	2.22	-2.15	0.32	0.46	-0.02	0.02	-0.06	0.02	2.17
180.00	2.24	-2.19	0.00	0.46	0.02	0.00	-0.06	-0.00	2.19

File: QAADBA3i.000
Program: GPSEQVEL.CPP
GPS utilizing equivalent velocity approximation.
Solving for Px,Py,Pz,Theta.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 7.50 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	1.54	0.99	-0.06	1.18	0.01	0.00	0.00	-0.02	0.99
10.00	1.63	1.08	0.18	1.21	-0.00	0.00	0.00	-0.00	1.10
20.00	1.91	1.29	0.50	1.31	-0.01	0.00	0.00	0.01	1.39
30.00	2.30	1.54	0.93	1.44	-0.02	0.00	0.00	0.01	1.79
40.00	2.72	1.68	1.44	1.57	-0.02	0.00	0.00	0.01	2.21
50.00	3.03	1.63	1.94	1.66	-0.01	0.00	0.00	-0.00	2.53
60.00	3.15	1.35	2.28	1.70	0.02	0.00	0.00	-0.01	2.65
70.00	3.03	0.92	2.36	1.67	0.04	0.00	0.00	-0.02	2.53
80.00	2.73	0.48	2.17	1.58	0.05	0.00	0.00	-0.04	2.22
90.00	2.31	0.12	1.80	1.45	0.05	0.00	0.00	-0.05	1.80
100.00	1.92	-0.11	1.39	1.32	0.04	0.00	0.00	-0.05	1.39
110.00	1.64	-0.28	1.07	1.22	0.02	0.00	0.00	-0.04	1.10
120.00	1.54	-0.45	0.89	1.18	0.01	0.00	0.00	-0.02	0.99
130.00	1.63	-0.70	0.85	1.21	-0.00	0.00	0.00	-0.00	1.10
140.00	1.91	-1.08	0.87	1.31	-0.01	0.00	0.00	0.01	1.39
150.00	2.30	-1.57	0.87	1.44	-0.02	0.00	0.00	0.01	1.79
160.00	2.72	-2.09	0.74	1.57	-0.02	0.00	0.00	0.01	2.21
170.00	3.03	-2.49	0.44	1.66	-0.01	0.00	0.00	-0.00	2.53
180.00	3.15	-2.65	0.03	1.70	0.02	0.00	0.00	-0.01	2.65

File: NAADBA4.000
Program: GPSNDVEL.CPP
GPS utilizing interpolated transmission velocities.
Solving for Px,Py,Pz,Theta,Phi,Mu.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 4.00
Platform pivot at radius of 1.50 m and depth of 7.50 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	2.01	1.97	-0.00	0.42	-0.01	0.00	0.19	-0.00	1.97
10.00	2.03	1.94	0.40	0.43	-0.05	0.02	0.19	0.02	1.98
20.00	2.07	1.87	0.78	0.44	-0.08	0.02	0.19	0.04	2.02
30.00	2.13	1.75	1.14	0.45	-0.09	0.03	0.19	0.04	2.08
40.00	2.19	1.58	1.45	0.46	-0.07	0.02	0.18	0.04	2.14
50.00	2.23	1.36	1.71	0.47	-0.04	0.01	0.18	0.02	2.18
60.00	2.24	1.10	1.90	0.47	-0.01	0.00	0.18	-0.00	2.19
70.00	2.23	0.80	2.03	0.47	0.03	-0.01	0.18	-0.02	2.18
80.00	2.19	0.47	2.09	0.46	0.06	-0.02	0.18	-0.04	2.14
90.00	2.13	0.11	2.08	0.45	0.07	-0.02	0.18	-0.04	2.08
100.00	2.07	-0.25	2.01	0.44	0.06	-0.01	0.19	-0.04	2.02
110.00	2.03	-0.62	1.88	0.43	0.03	-0.01	0.19	-0.02	1.98
120.00	2.01	-0.98	1.70	0.42	-0.01	0.00	0.19	-0.00	1.97
130.00	2.03	-1.32	1.48	0.43	-0.05	0.02	0.19	0.02	1.98
140.00	2.07	-1.61	1.23	0.44	-0.08	0.02	0.19	0.04	2.02
150.00	2.13	-1.86	0.94	0.45	-0.09	0.03	0.19	0.04	2.08
160.00	2.19	-2.04	0.64	0.46	-0.07	0.02	0.18	0.04	2.14
170.00	2.23	-2.16	0.32	0.47	-0.04	0.01	0.18	0.02	2.18
180.00	2.24	-2.19	0.00	0.47	-0.01	0.00	0.18	-0.00	2.19

File: QAADBA4i.000
Program: GPSEQVEL.CPP
GPS utilizing equivalent velocity approximation.
Solving for Px,Py,Pz,Theta.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 4.00
Platform pivot at radius of 1.50 m and depth of 7.50 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	2.03	2.02	-0.06	0.20	0.01	0.00	0.00	-0.02	2.02
10.00	2.16	2.11	0.37	0.24	-0.00	0.00	0.00	0.00	2.15
20.00	2.53	2.34	0.90	0.37	-0.02	0.00	0.00	0.02	2.50
30.00	3.05	2.57	1.55	0.54	-0.02	0.00	0.00	0.02	3.00
40.00	3.59	2.67	2.29	0.70	-0.02	0.00	0.00	0.02	3.52
50.00	3.99	2.50	3.00	0.82	-0.01	0.00	0.00	0.00	3.91
60.00	4.14	2.05	3.49	0.86	0.02	0.00	0.00	-0.01	4.05
70.00	3.99	1.40	3.64	0.82	0.04	0.00	0.00	-0.03	3.90
80.00	3.59	0.72	3.44	0.71	0.05	0.00	0.00	-0.04	3.51
90.00	3.05	0.15	3.00	0.54	0.05	0.00	0.00	-0.06	3.00
100.00	2.53	-0.28	2.48	0.37	0.04	0.00	0.00	-0.06	2.49
110.00	2.16	-0.62	2.05	0.25	0.02	0.00	0.00	-0.05	2.14
120.00	2.03	-0.96	1.78	0.20	0.01	0.00	0.00	-0.02	2.02
130.00	2.16	-1.38	1.64	0.24	-0.00	0.00	0.00	0.00	2.15
140.00	2.53	-1.95	1.57	0.37	-0.02	0.00	0.00	0.02	2.50
150.00	3.05	-2.63	1.45	0.54	-0.02	0.00	0.00	0.02	3.00
160.00	3.59	-3.32	1.17	0.70	-0.02	0.00	0.00	0.02	3.52
170.00	3.99	-3.85	0.67	0.82	-0.01	0.00	0.00	0.00	3.91
180.00	4.14	-4.05	0.03	0.86	0.02	0.00	0.00	-0.01	4.05

File: NAADBB3.000 Program: GPSNDVEL.CPP
GPS utilizing interpolated transmission velocities.
Solving for Px,Py,Pz,Theta,Phi,Mu.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 7.50 m.
Error incorporation over 10000 iterations.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	2.02	1.98	0.20	0.34	0.23	0.08	0.18	0.08	1.98
10.00	2.04	1.95	0.42	0.34	0.23	0.08	0.18	0.08	1.99
20.00	2.08	1.88	0.79	0.35	0.24	0.08	0.17	0.08	2.03
30.00	2.14	1.76	1.14	0.36	0.24	0.08	0.18	0.08	2.09
40.00	2.19	1.58	1.45	0.37	0.24	0.08	0.18	0.08	2.15
50.00	2.23	1.36	1.71	0.38	0.24	0.08	0.18	0.07	2.19
60.00	2.25	1.10	1.91	0.39	0.24	0.08	0.18	0.07	2.20
70.00	2.23	0.80	2.03	0.38	0.25	0.08	0.17	0.07	2.18
80.00	2.20	0.47	2.10	0.37	0.25	0.08	0.18	0.08	2.15
90.00	2.14	0.22	2.09	0.36	0.25	0.08	0.18	0.08	2.09
100.00	2.08	0.30	2.02	0.35	0.24	0.08	0.18	0.08	2.03
110.00	2.04	0.63	1.90	0.34	0.24	0.08	0.18	0.08	2.00
120.00	2.02	0.99	1.71	0.34	0.23	0.08	0.18	0.08	1.98
130.00	2.03	1.32	1.49	0.34	0.23	0.08	0.17	0.08	1.99
140.00	2.08	1.62	1.23	0.35	0.24	0.08	0.17	0.08	2.03
150.00	2.14	1.87	0.94	0.36	0.24	0.08	0.18	0.08	2.09
160.00	2.19	2.05	0.65	0.37	0.24	0.08	0.18	0.08	2.14
170.00	2.24	2.16	0.34	0.38	0.24	0.08	0.18	0.08	2.19
180.00	2.25	2.21	0.19	0.38	0.25	0.08	0.18	0.07	2.21
Maximum Error:									
60.00	3.31	1.85	2.69	0.57	0.10	-0.06	0.18	-0.10	3.25

Standard deviation of total error at alpha = 60.0 degrees was 0.26 cm.

File: QAADBB3i.000 Program: GPSEQVEL.CPP
GPS utilizing equivalent velocity approximation.
Solving for Px,Py,Pz,Theta.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 7.50 m.
Error incorporation over 10000 iterations.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	2.82	1.75	1.12	1.38	0.22	0.00	0.00	0.42	1.76
10.00	2.88	1.80	1.15	1.41	0.22	0.00	0.00	0.42	1.82
20.00	3.04	1.87	1.28	1.49	0.23	0.00	0.00	0.42	1.98
30.00	3.29	2.01	1.50	1.57	0.23	0.00	0.00	0.42	2.26
40.00	3.57	2.05	1.87	1.66	0.24	0.00	0.00	0.42	2.56
50.00	3.81	1.95	2.29	1.72	0.24	0.00	0.00	0.43	2.82
60.00	3.93	1.75	2.59	1.78	0.24	0.00	0.00	0.43	2.94
70.00	3.77	1.44	2.61	1.73	0.24	0.00	0.00	0.42	2.78
80.00	3.57	1.20	2.51	1.68	0.24	0.00	0.00	0.42	2.56
90.00	3.28	1.12	2.24	1.58	0.23	0.00	0.00	0.42	2.25
100.00	3.03	1.15	1.95	1.48	0.23	0.00	0.00	0.43	1.96
110.00	2.90	1.20	1.76	1.43	0.22	0.00	0.00	0.43	1.82
120.00	2.81	1.28	1.62	1.39	0.22	0.00	0.00	0.42	1.75
130.00	2.87	1.46	1.54	1.42	0.22	0.00	0.00	0.42	1.81
140.00	3.05	1.70	1.50	1.49	0.23	0.00	0.00	0.43	1.99
150.00	3.27	2.03	1.43	1.57	0.23	0.00	0.00	0.42	2.24
160.00	3.58	2.45	1.34	1.67	0.23	0.00	0.00	0.43	2.56
170.00	3.83	2.80	1.22	1.73	0.24	0.00	0.00	0.43	2.84
180.00	3.91	2.93	1.14	1.77	0.24	0.00	0.00	0.43	2.93
Maximum error:									
60.00	11.41	4.41	9.57	4.39	-0.28	0.00	0.00	0.34	10.49

Standard deviation of total error at alpha = 60.0 degrees was 1.78 cm.

File: NAADBC3.000 Program: GPSNDVEL.CPP
GPS utilizing interpolated transmission velocities.
Solving for Px,Py,Pz,Theta,Phi,Mu.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 7.50 m.
Error incorporation over 10000 iterations.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	2.04	1.96	0.28	0.42	0.10	0.08	0.21	0.10	1.96
10.00	2.05	1.93	0.44	0.42	0.10	0.09	0.21	0.11	1.97
20.00	2.09	1.86	0.78	0.43	0.11	0.09	0.21	0.11	2.01
30.00	2.15	1.74	1.14	0.44	0.11	0.09	0.21	0.11	2.07
40.00	2.21	1.57	1.44	0.45	0.10	0.09	0.21	0.11	2.13
50.00	2.25	1.35	1.70	0.46	0.10	0.09	0.21	0.10	2.17
60.00	2.26	1.09	1.89	0.46	0.10	0.08	0.21	0.10	2.19
70.00	2.25	0.80	2.02	0.46	0.11	0.08	0.21	0.10	2.18
80.00	2.21	0.49	2.08	0.45	0.12	0.09	0.21	0.11	2.13
90.00	2.16	0.29	2.08	0.44	0.12	0.09	0.21	0.11	2.08
100.00	2.09	0.34	2.00	0.43	0.11	0.09	0.21	0.11	2.02
110.00	2.05	0.64	1.87	0.42	0.10	0.08	0.21	0.11	1.98
120.00	2.03	0.98	1.70	0.42	0.10	0.08	0.21	0.11	1.96
130.00	2.06	1.32	1.48	0.42	0.10	0.09	0.21	0.11	1.98
140.00	2.09	1.60	1.22	0.43	0.11	0.09	0.21	0.11	2.01
150.00	2.16	1.86	0.94	0.44	0.11	0.09	0.21	0.11	2.08
160.00	2.21	2.03	0.64	0.45	0.10	0.09	0.21	0.10	2.13
170.00	2.26	2.16	0.38	0.46	0.10	0.09	0.21	0.10	2.18
180.00	2.26	2.19	0.26	0.46	0.10	0.08	0.21	0.10	2.19
Maximum Error:									
170.00	3.37	-3.29	0.28	0.68	-0.14	-0.13	0.10	0.11	3.28

Standard deviation of total error at alpha = 170.0 degrees was 0.35 cm.

File: QAADBC3i.000 Program: GPSEQVEL.CPP
GPS utilizing equivalent velocity approximation.
Solving for Px,Py,Pz,Theta.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 7.50 m.
Error incorporation over 10000 iterations.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	2.43	1.23	1.00	1.36	0.11	0.00	0.00	0.38	1.24
10.00	2.50	1.31	1.02	1.37	0.11	0.00	0.00	0.38	1.32
20.00	2.70	1.45	1.11	1.44	0.11	0.00	0.00	0.38	1.54
30.00	3.01	1.66	1.33	1.51	0.11	0.00	0.00	0.37	1.91
40.00	3.35	1.79	1.65	1.62	0.11	0.00	0.00	0.37	2.29
50.00	3.63	1.76	2.06	1.70	0.11	0.00	0.00	0.37	2.60
60.00	3.70	1.53	2.32	1.73	0.11	0.00	0.00	0.37	2.67
70.00	3.61	1.27	2.41	1.71	0.11	0.00	0.00	0.37	2.57
80.00	3.34	1.07	2.22	1.64	0.11	0.00	0.00	0.37	2.27
90.00	2.99	0.98	1.88	1.53	0.12	0.00	0.00	0.37	1.89
100.00	2.70	0.99	1.56	1.44	0.12	0.00	0.00	0.38	1.55
110.00	2.49	1.00	1.31	1.36	0.11	0.00	0.00	0.38	1.32
120.00	2.42	1.04	1.19	1.35	0.11	0.00	0.00	0.38	1.24
130.00	2.48	1.14	1.14	1.37	0.11	0.00	0.00	0.37	1.30
140.00	2.70	1.39	1.18	1.43	0.11	0.00	0.00	0.38	1.56
150.00	3.00	1.71	1.18	1.53	0.11	0.00	0.00	0.38	1.87
160.00	3.34	2.17	1.14	1.61	0.11	0.00	0.00	0.37	2.28
170.00	3.61	2.57	1.04	1.69	0.11	0.00	0.00	0.37	2.60
180.00	3.72	2.72	0.98	1.72	0.10	0.00	0.00	0.37	2.72
Maximum error:									
180.00	8.32	-8.28	0.72	-0.30	0.11	0.00	0.00	-0.26	8.28

Standard deviation of total error at alpha = 180.0 degrees was 0.93 cm.

File: NAADBD3.000 Program: GPSNDVEL.CPP
GPS utilizing interpolated transmission velocities.
Solving for Px,Py,Pz,Theta,Phi,Mu.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 7.50 m.
Error incorporation over 10000 iterations.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	2.05	1.99	0.31	0.26	0.45	0.13	0.28	0.12	1.99
10.00	2.05	1.96	0.45	0.26	0.45	0.13	0.28	0.11	2.00
20.00	2.10	1.89	0.79	0.27	0.46	0.14	0.28	0.11	2.05
30.00	2.15	1.76	1.14	0.28	0.47	0.13	0.28	0.12	2.10
40.00	2.21	1.59	1.46	0.29	0.47	0.14	0.28	0.11	2.16
50.00	2.25	1.37	1.73	0.30	0.47	0.13	0.28	0.11	2.20
60.00	2.27	1.11	1.92	0.31	0.48	0.13	0.28	0.11	2.22
70.00	2.26	0.81	2.05	0.31	0.47	0.13	0.29	0.11	2.20
80.00	2.21	0.50	2.11	0.29	0.47	0.13	0.28	0.11	2.16
90.00	2.16	0.31	2.10	0.28	0.47	0.13	0.28	0.12	2.10
100.00	2.10	0.36	2.04	0.27	0.47	0.13	0.28	0.12	2.05
110.00	2.06	0.65	1.90	0.26	0.46	0.13	0.28	0.11	2.01
120.00	2.05	1.00	1.73	0.26	0.46	0.13	0.28	0.11	2.00
130.00	2.06	1.33	1.51	0.26	0.45	0.13	0.28	0.11	2.01
140.00	2.10	1.63	1.24	0.27	0.47	0.13	0.28	0.12	2.04
150.00	2.16	1.88	0.95	0.28	0.48	0.13	0.28	0.11	2.10
160.00	2.22	2.07	0.66	0.30	0.49	0.14	0.28	0.11	2.17
170.00	2.26	2.18	0.40	0.30	0.47	0.13	0.28	0.11	2.21
180.00	2.27	2.22	0.28	0.31	0.48	0.14	0.29	0.11	2.22
Maximum Error:									
90.00	3.84	0.54	3.75	0.64	0.08	-0.03	-0.22	-0.20	3.75

Standard deviation of total error at alpha = 90.0 degrees was 0.37 cm.

File: QAADBD3i.000 Program: GPSEQVEL.CPP
GPS utilizing equivalent velocity approximation.
Solving for Px,Py,Pz,Theta.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 7.50 m.
Error incorporation over 10000 iterations.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	4.42	2.94	1.63	2.05	0.43	0.00	0.00	0.62	2.94
10.00	4.47	2.97	1.68	2.05	0.43	0.00	0.00	0.61	3.01
20.00	4.59	2.99	1.88	2.10	0.44	0.00	0.00	0.61	3.14
30.00	4.84	3.04	2.25	2.16	0.45	0.00	0.00	0.63	3.42
40.00	5.02	2.95	2.66	2.19	0.45	0.00	0.00	0.63	3.62
50.00	5.26	2.78	3.14	2.28	0.47	0.00	0.00	0.64	3.85
60.00	5.32	2.43	3.50	2.29	0.46	0.00	0.00	0.63	3.92
70.00	5.27	2.05	3.65	2.31	0.46	0.00	0.00	0.64	3.85
80.00	5.08	1.79	3.58	2.25	0.45	0.00	0.00	0.64	3.64
90.00	4.84	1.66	3.39	2.18	0.44	0.00	0.00	0.63	3.39
100.00	4.61	1.70	3.11	2.12	0.43	0.00	0.00	0.62	3.15
110.00	4.49	1.83	2.90	2.09	0.43	0.00	0.00	0.62	3.02
120.00	4.45	2.04	2.72	2.07	0.43	0.00	0.00	0.61	3.00
130.00	4.46	2.31	2.50	2.06	0.43	0.00	0.00	0.62	2.99
140.00	4.60	2.66	2.36	2.08	0.44	0.00	0.00	0.62	3.15
150.00	4.87	3.08	2.23	2.19	0.45	0.00	0.00	0.63	3.43
160.00	5.13	3.51	2.01	2.27	0.46	0.00	0.00	0.63	3.69
170.00	5.24	3.78	1.76	2.30	0.46	0.00	0.00	0.63	3.83
180.00	5.30	3.89	1.69	2.31	0.46	0.00	0.00	0.64	3.89
Maximum error:									
60.00	19.56	6.45	15.15	10.56	-0.16	0.00	0.00	0.68	16.35

Standard deviation of total error at alpha = 60.0 degrees was 2.75 cm.

File: NAADCA3.000
Program: GPSNDVEL.CPP
GPS utilizing interpolated transmission velocities.
Solving for Px,Py,Pz,Theta,Phi,Mu.
Temperatures: Tw = 40.00, Tc = 30.00, Tt = 45.00, Tb = 25.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 7.50 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	4.07	3.98	-0.00	0.89	0.02	0.01	-0.09	-0.00	3.98
10.00	4.11	3.93	0.82	0.89	-0.06	0.04	-0.09	0.05	4.01
20.00	4.21	3.78	1.60	0.91	-0.12	0.05	-0.09	0.08	4.10
30.00	4.34	3.54	2.33	0.94	-0.14	0.06	-0.09	0.09	4.23
40.00	4.46	3.20	2.96	0.96	-0.11	0.05	-0.09	0.08	4.35
50.00	4.55	2.76	3.48	0.98	-0.05	0.03	-0.09	0.05	4.44
60.00	4.58	2.24	3.88	0.99	0.04	0.01	-0.09	-0.00	4.48
70.00	4.55	1.64	4.13	0.98	0.12	-0.01	-0.09	-0.05	4.44
80.00	4.46	0.97	4.25	0.96	0.18	-0.03	-0.09	-0.08	4.35
90.00	4.34	0.25	4.23	0.94	0.20	-0.04	-0.09	-0.09	4.23
100.00	4.21	-0.50	4.08	0.91	0.17	-0.03	-0.09	-0.08	4.10
110.00	4.11	-1.25	3.81	0.89	0.10	-0.01	-0.09	-0.05	4.01
120.00	4.07	-1.99	3.45	0.89	0.02	0.01	-0.09	-0.00	3.98
130.00	4.11	-2.67	2.99	0.89	-0.06	0.04	-0.09	0.05	4.01
140.00	4.21	-3.28	2.47	0.91	-0.12	0.05	-0.09	0.08	4.10
150.00	4.34	-3.78	1.90	0.94	-0.14	0.06	-0.09	0.09	4.23
160.00	4.46	-4.16	1.29	0.96	-0.11	0.05	-0.09	0.08	4.35
170.00	4.55	-4.40	0.65	0.98	-0.05	0.03	-0.09	0.05	4.44
180.00	4.58	-4.48	0.00	0.99	0.04	0.01	-0.09	-0.00	4.48

File: QAADCA3i.000
Program: GPSEQVEL.CPP
GPS utilizing equivalent velocity approximation.
Solving for Px,Py,Pz,Theta.
Temperatures: Tw = 40.00, Tc = 30.00, Tt = 45.00, Tb = 25.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 7.50 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	3.13	2.08	-0.12	2.33	0.02	0.00	0.00	-0.05	2.08
10.00	3.32	2.27	0.37	2.40	-0.01	0.00	0.00	-0.01	2.30
20.00	3.90	2.71	1.03	2.60	-0.03	0.00	0.00	0.02	2.90
30.00	4.72	3.22	1.93	2.87	-0.05	0.00	0.00	0.02	3.75
40.00	5.58	3.52	3.00	3.13	-0.05	0.00	0.00	0.01	4.62
50.00	6.23	3.40	4.03	3.31	-0.02	0.00	0.00	-0.00	5.27
60.00	6.47	2.81	4.75	3.38	0.03	0.00	0.00	-0.02	5.51
70.00	6.24	1.91	4.92	3.33	0.08	0.00	0.00	-0.04	5.28
80.00	5.60	0.99	4.52	3.15	0.11	0.00	0.00	-0.07	4.63
90.00	4.74	0.24	3.75	2.89	0.10	0.00	0.00	-0.09	3.75
100.00	3.91	-0.25	2.90	2.62	0.08	0.00	0.00	-0.10	2.90
110.00	3.34	-0.59	2.23	2.41	0.05	0.00	0.00	-0.08	2.30
120.00	3.13	-0.93	1.86	2.33	0.02	0.00	0.00	-0.05	2.08
130.00	3.32	-1.46	1.78	2.40	-0.01	0.00	0.00	-0.01	2.30
140.00	3.90	-2.25	1.83	2.60	-0.03	0.00	0.00	0.02	2.90
150.00	4.72	-3.28	1.82	2.87	-0.05	0.00	0.00	0.02	3.75
160.00	5.58	-4.36	1.55	3.13	-0.05	0.00	0.00	0.01	4.62
170.00	6.23	-5.19	0.92	3.31	-0.02	0.00	0.00	-0.00	5.27
180.00	6.47	-5.51	0.06	3.38	0.03	0.00	0.00	-0.02	5.51

File: NAADDA3.000
Program: GPSNDVEL.CPP
GPS utilizing interpolated transmission velocities.
Solving for Px,Py,Pz,Theta,Phi,Mu.
Temperatures: Tw = 30.00, Tc = 25.00, Tt = 30.00, Tb = 20.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 7.50 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	2.57	2.51	-0.00	0.54	0.01	0.01	-0.07	-0.00	2.51
10.00	2.59	2.48	0.51	0.54	-0.04	0.02	-0.07	0.03	2.53
20.00	2.65	2.39	1.00	0.55	-0.07	0.03	-0.07	0.05	2.59
30.00	2.72	2.23	1.46	0.57	-0.08	0.04	-0.07	0.05	2.66
40.00	2.80	2.01	1.85	0.58	-0.07	0.03	-0.07	0.05	2.73
50.00	2.85	1.74	2.18	0.59	-0.03	0.02	-0.07	0.03	2.79
60.00	2.87	1.41	2.43	0.60	0.02	0.01	-0.07	-0.00	2.81
70.00	2.85	1.02	2.59	0.59	0.07	-0.01	-0.07	-0.03	2.79
80.00	2.80	0.60	2.67	0.58	0.11	-0.02	-0.07	-0.05	2.73
90.00	2.72	0.15	2.66	0.57	0.12	-0.02	-0.07	-0.06	2.66
100.00	2.65	-0.32	2.57	0.55	0.10	-0.02	-0.07	-0.05	2.59
110.00	2.59	-0.80	2.41	0.54	0.06	-0.01	-0.07	-0.03	2.53
120.00	2.57	-1.25	2.18	0.54	0.01	0.01	-0.07	-0.00	2.51
130.00	2.59	-1.68	1.89	0.54	-0.04	0.02	-0.07	0.03	2.53
140.00	2.65	-2.06	1.57	0.55	-0.07	0.03	-0.07	0.05	2.59
150.00	2.72	-2.38	1.20	0.57	-0.08	0.04	-0.07	0.05	2.66
160.00	2.80	-2.61	0.82	0.58	-0.07	0.03	-0.07	0.05	2.73
170.00	2.85	-2.76	0.41	0.59	-0.03	0.02	-0.07	0.03	2.79
180.00	2.87	-2.81	0.00	0.60	0.02	0.01	-0.07	-0.00	2.81

File: QAADDA3i.000
Program: GPSEQVEL.CPP
GPS utilizing equivalent velocity approximation.
Solving for Px,Py,Pz,Theta.
Temperatures: Tw = 30.00, Tc = 25.00, Tt = 30.00, Tb = 20.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 7.50 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	1.98	1.28	-0.07	1.51	0.01	0.00	0.00	-0.03	1.28
10.00	2.10	1.39	0.23	1.56	-0.00	0.00	0.00	-0.00	1.41
20.00	2.45	1.66	0.64	1.68	-0.02	0.00	0.00	0.01	1.78
30.00	2.96	1.98	1.19	1.85	-0.03	0.00	0.00	0.02	2.31
40.00	3.49	2.16	1.85	2.02	-0.03	0.00	0.00	0.01	2.85
50.00	3.89	2.09	2.49	2.14	-0.01	0.00	0.00	-0.00	3.25
60.00	4.05	1.73	2.93	2.19	0.02	0.00	0.00	-0.01	3.40
70.00	3.90	1.19	3.03	2.15	0.05	0.00	0.00	-0.03	3.26
80.00	3.51	0.62	2.79	2.03	0.06	0.00	0.00	-0.05	2.85
90.00	2.97	0.16	2.31	1.87	0.06	0.00	0.00	-0.06	2.31
100.00	2.47	-0.15	1.79	1.69	0.05	0.00	0.00	-0.06	1.79
110.00	2.11	-0.36	1.37	1.56	0.03	0.00	0.00	-0.05	1.41
120.00	1.98	-0.57	1.14	1.51	0.01	0.00	0.00	-0.03	1.28
130.00	2.10	-0.90	1.09	1.56	-0.00	0.00	0.00	-0.00	1.41
140.00	2.45	-1.39	1.12	1.68	-0.02	0.00	0.00	0.01	1.78
150.00	2.96	-2.02	1.12	1.85	-0.03	0.00	0.00	0.02	2.31
160.00	3.49	-2.69	0.95	2.02	-0.03	0.00	0.00	0.01	2.85
170.00	3.89	-3.20	0.57	2.14	-0.01	0.00	0.00	-0.00	3.25
180.00	4.05	-3.40	0.04	2.19	0.02	0.00	0.00	-0.01	3.40

File: NAAEBA3.000
Program: GPSNDVEL.CPP
GPS utilizing interpolated transmission velocities.
Solving for Px,Py,Pz,Theta,Phi,Mu.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 10.00 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	3.48	3.43	-0.00	0.59	0.01	0.01	-0.03	-0.00	3.43
10.00	3.50	3.38	0.66	0.59	-0.03	0.02	-0.03	0.02	3.45
20.00	3.54	3.24	1.29	0.60	-0.06	0.02	-0.03	0.04	3.49
30.00	3.61	3.02	1.88	0.61	-0.07	0.03	-0.03	0.04	3.55
40.00	3.67	2.70	2.40	0.62	-0.06	0.02	-0.03	0.04	3.61
50.00	3.71	2.30	2.84	0.63	-0.03	0.02	-0.03	0.02	3.66
60.00	3.73	1.84	3.18	0.63	0.02	0.01	-0.03	-0.00	3.67
70.00	3.71	1.31	3.42	0.63	0.06	-0.00	-0.03	-0.02	3.66
80.00	3.67	0.73	3.54	0.62	0.09	-0.01	-0.03	-0.04	3.61
90.00	3.61	0.12	3.55	0.61	0.10	-0.01	-0.03	-0.05	3.55
100.00	3.54	-0.50	3.46	0.60	0.08	-0.01	-0.03	-0.04	3.49
110.00	3.50	-1.12	3.26	0.59	0.05	-0.00	-0.03	-0.02	3.45
120.00	3.48	-1.71	2.97	0.59	0.01	0.01	-0.03	-0.00	3.43
130.00	3.50	-2.26	2.60	0.59	-0.03	0.02	-0.03	0.02	3.45
140.00	3.54	-2.74	2.16	0.60	-0.06	0.02	-0.03	0.04	3.49
150.00	3.61	-3.14	1.67	0.61	-0.07	0.03	-0.03	0.04	3.55
160.00	3.67	-3.43	1.14	0.62	-0.06	0.02	-0.03	0.04	3.61
170.00	3.71	-3.61	0.58	0.63	-0.03	0.02	-0.03	0.02	3.66
180.00	3.73	-3.67	0.00	0.63	0.02	0.01	-0.03	-0.00	3.67

File: QAAEBA3i.000
Program: GPSEQVEL.CPP
GPS utilizing equivalent velocity approximation.
Solving for Px,Py,Pz,Theta.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 10.00 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	3.00	2.60	-0.12	1.49	0.01	0.00	0.00	-0.05	2.60
10.00	3.20	2.76	0.40	1.55	-0.00	0.00	0.00	-0.03	2.79
20.00	3.73	3.13	1.08	1.72	-0.02	0.00	0.00	-0.02	3.31
30.00	4.48	3.51	1.97	1.96	-0.03	0.00	0.00	-0.02	4.03
40.00	5.22	3.68	3.00	2.18	-0.03	0.00	0.00	-0.02	4.74
50.00	5.77	3.44	3.99	2.34	-0.01	0.00	0.00	-0.03	5.27
60.00	5.96	2.80	4.68	2.40	0.02	0.00	0.00	-0.03	5.46
70.00	5.76	1.89	4.91	2.34	0.05	0.00	0.00	-0.03	5.26
80.00	5.21	0.94	4.64	2.18	0.06	0.00	0.00	-0.05	4.73
90.00	4.46	0.15	4.01	1.95	0.06	0.00	0.00	-0.06	4.01
100.00	3.72	-0.40	3.27	1.72	0.05	0.00	0.00	-0.06	3.29
110.00	3.19	-0.80	2.67	1.55	0.03	0.00	0.00	-0.06	2.78
120.00	3.00	-1.19	2.31	1.49	0.01	0.00	0.00	-0.05	2.60
130.00	3.20	-1.73	2.19	1.55	-0.00	0.00	0.00	-0.03	2.79
140.00	3.73	-2.50	2.17	1.72	-0.02	0.00	0.00	-0.02	3.31
150.00	4.48	-3.46	2.06	1.96	-0.03	0.00	0.00	-0.02	4.03
160.00	5.22	-4.44	1.68	2.18	-0.03	0.00	0.00	-0.02	4.74
170.00	5.77	-5.18	0.99	2.34	-0.01	0.00	0.00	-0.03	5.27
180.00	5.96	-5.46	0.08	2.40	0.02	0.00	0.00	-0.03	5.46

File: NAAECA3.000
Program: GPSNDVEL.CPP
GPS utilizing interpolated transmission velocities.
Solving for Px,Py,Pz,Theta,Phi,Mu.
Temperatures: Tw = 40.00, Tc = 30.00, Tt = 45.00, Tb = 25.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 10.00 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	7.19	7.06	-0.00	1.33	0.01	0.02	0.02	-0.00	7.06
10.00	7.22	6.97	1.37	1.34	-0.09	0.04	0.02	0.05	7.10
20.00	7.33	6.69	2.69	1.35	-0.15	0.06	0.02	0.09	7.20
30.00	7.48	6.22	3.91	1.38	-0.18	0.06	0.02	0.10	7.34
40.00	7.62	5.58	5.00	1.40	-0.14	0.05	0.02	0.09	7.49
50.00	7.72	4.77	5.90	1.41	-0.07	0.04	0.02	0.05	7.59
60.00	7.76	3.82	6.61	1.42	0.03	0.02	0.02	-0.00	7.63
70.00	7.72	2.73	7.08	1.41	0.13	-0.01	0.02	-0.05	7.59
80.00	7.62	1.54	7.33	1.40	0.19	-0.02	0.02	-0.09	7.49
90.00	7.48	0.29	7.35	1.38	0.22	-0.03	0.02	-0.10	7.35
100.00	7.33	-1.01	7.14	1.35	0.18	-0.02	0.02	-0.09	7.20
110.00	7.23	-2.29	6.72	1.34	0.11	-0.00	0.02	-0.05	7.10
120.00	7.19	-3.53	6.12	1.33	0.01	0.02	0.02	-0.00	7.06
130.00	7.22	-4.67	5.35	1.34	-0.09	0.04	0.02	0.05	7.10
140.00	7.33	-5.67	4.44	1.35	-0.15	0.06	0.02	0.09	7.20
150.00	7.48	-6.50	3.43	1.38	-0.18	0.06	0.02	0.10	7.34
160.00	7.62	-7.12	2.33	1.40	-0.14	0.05	0.02	0.09	7.49
170.00	7.72	-7.50	1.18	1.41	-0.07	0.04	0.02	0.05	7.59
180.00	7.76	-7.63	0.00	1.42	0.03	0.02	0.02	-0.00	7.63

File: QAAECA3i.000
Program: GPSEQVEL.CPP
GPS utilizing equivalent velocity approximation.
Solving for Px,Py,Pz,Theta.
Temperatures: Tw = 40.00, Tc = 30.00, Tt = 45.00, Tb = 25.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 10.00 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	6.29	5.57	-0.27	2.89	0.03	0.00	0.00	-0.10	5.57
10.00	6.72	5.93	0.85	3.03	-0.02	0.00	0.00	-0.07	5.99
20.00	7.88	6.73	2.29	3.39	-0.05	0.00	0.00	-0.05	7.11
30.00	9.48	7.56	4.20	3.88	-0.08	0.00	0.00	-0.05	8.65
40.00	11.07	7.90	6.41	4.36	-0.07	0.00	0.00	-0.06	10.17
50.00	12.22	7.38	8.53	4.70	-0.03	0.00	0.00	-0.06	11.28
60.00	12.63	5.98	10.02	4.82	0.04	0.00	0.00	-0.06	11.67
70.00	12.20	4.02	10.52	4.70	0.11	0.00	0.00	-0.06	11.26
80.00	11.04	1.99	9.94	4.36	0.14	0.00	0.00	-0.08	10.14
90.00	9.44	0.30	8.60	3.88	0.14	0.00	0.00	-0.11	8.60
100.00	7.85	-0.89	7.02	3.39	0.11	0.00	0.00	-0.12	7.07
110.00	6.70	-1.73	5.72	3.02	0.07	0.00	0.00	-0.12	5.97
120.00	6.29	-2.56	4.96	2.89	0.03	0.00	0.00	-0.10	5.57
130.00	6.72	-3.70	4.71	3.03	-0.02	0.00	0.00	-0.07	5.99
140.00	7.88	-5.35	4.69	3.39	-0.05	0.00	0.00	-0.05	7.11
150.00	9.48	-7.41	4.45	3.88	-0.08	0.00	0.00	-0.05	8.65
160.00	11.07	-9.50	3.64	4.36	-0.07	0.00	0.00	-0.06	10.17
170.00	12.22	-11.08	2.13	4.70	-0.03	0.00	0.00	-0.06	11.28
180.00	12.63	-11.67	0.17	4.82	0.04	0.00	0.00	-0.06	11.67

File: NABABA3.000
Program: GPSNDVEL.CPP
GPS utilizing interpolated transmission velocities.
Solving for Px,Py,Pz,Theta,Phi,Mu.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 0.00 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	0.81	0.15	0.01	-0.80	0.05	-0.99	-0.62	0.00	0.15
10.00	1.46	0.14	0.09	-1.45	0.03	-0.72	-0.44	0.02	0.16
20.00	1.33	0.13	0.15	-1.31	0.02	-0.35	-0.49	0.04	0.18
30.00	0.91	0.13	0.20	-0.88	0.02	0.00	-0.67	0.04	0.21
40.00	0.97	0.12	0.23	-0.94	0.01	0.26	-0.82	0.04	0.24
50.00	0.33	0.12	0.23	0.20	0.04	0.43	-1.25	0.02	0.26
60.00	0.74	0.13	0.23	0.69	0.05	0.40	-1.52	0.00	0.26
70.00	1.16	0.13	0.22	1.14	0.07	0.18	-1.71	-0.02	0.26
80.00	1.53	0.13	0.21	1.51	0.08	-0.18	-1.76	-0.03	0.23
90.00	1.72	0.10	0.20	1.71	0.09	-0.60	-1.65	-0.04	0.20
100.00	1.60	0.05	0.19	1.59	0.09	-0.94	-1.42	-0.03	0.18
110.00	0.75	-0.01	0.17	0.73	0.07	-1.09	-1.05	-0.02	0.16
120.00	0.81	-0.08	0.13	-0.79	0.05	-0.99	-0.62	0.00	0.15
130.00	1.46	-0.15	0.08	-1.45	0.03	-0.72	-0.44	0.02	0.16
140.00	1.33	-0.20	0.04	-1.31	0.02	-0.35	-0.49	0.04	0.18
150.00	0.91	-0.23	0.01	-0.88	0.02	0.00	-0.67	0.04	0.21
160.00	0.97	-0.25	-0.01	-0.94	0.01	0.26	-0.82	0.04	0.24
170.00	0.33	-0.26	-0.01	0.20	0.04	0.43	-1.25	0.02	0.26
180.00	0.74	-0.26	-0.00	0.69	0.05	0.40	-1.52	0.00	0.26

File: QABABA3i.000
Program: GPSEQVEL.CPP
GPS utilizing equivalent velocity approximation.
Solving for Px,Py,Pz,Theta.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 0.00 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	1.27	-0.60	0.04	-1.12	-0.01	0.00	0.00	0.02	-0.60
10.00	0.95	-0.56	0.02	-0.77	-0.00	0.00	0.00	0.05	-0.54
20.00	1.23	-0.43	-0.08	-1.14	-0.01	0.00	0.00	0.03	-0.43
30.00	2.11	-0.26	-0.26	-2.08	-0.04	0.00	0.00	-0.04	-0.36
40.00	3.08	-0.10	-0.43	-3.05	-0.07	0.00	0.00	-0.10	-0.36
50.00	3.74	0.02	-0.54	-3.71	-0.09	0.00	0.00	-0.14	-0.40
60.00	4.03	0.09	-0.58	-3.99	-0.09	0.00	0.00	-0.14	-0.46
70.00	4.01	0.12	-0.59	-3.97	-0.08	0.00	0.00	-0.12	-0.51
80.00	3.78	0.13	-0.59	-3.73	-0.06	0.00	0.00	-0.09	-0.55
90.00	3.38	0.14	-0.58	-3.32	-0.04	0.00	0.00	-0.05	-0.58
100.00	2.79	0.16	-0.56	-2.72	-0.03	0.00	0.00	-0.02	-0.58
110.00	2.00	0.22	-0.54	-1.92	-0.02	0.00	0.00	-0.01	-0.59
120.00	1.27	0.26	-0.54	-1.12	-0.01	0.00	0.00	0.02	-0.60
130.00	0.95	0.26	-0.49	-0.77	-0.00	0.00	0.00	0.05	-0.54
140.00	1.23	0.28	-0.34	-1.14	-0.01	0.00	0.00	0.03	-0.43
150.00	2.11	0.36	-0.10	-2.08	-0.04	0.00	0.00	-0.04	-0.36
160.00	3.08	0.43	0.13	-3.05	-0.07	0.00	0.00	-0.10	-0.36
170.00	3.74	0.46	0.28	-3.71	-0.09	0.00	0.00	-0.14	-0.40
180.00	4.03	0.46	0.36	-3.99	-0.09	0.00	0.00	-0.14	-0.46

File: NABBBA3.000
Program: GPSNDVEL.CPP
GPS utilizing interpolated transmission velocities.
Solving for Px,Py,Pz,Theta,Phi,Mu.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 2.50 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	0.36	0.34	-0.00	0.11	0.01	-0.02	0.09	-0.00	0.34
10.00	0.38	0.34	0.11	0.12	0.00	0.00	0.09	0.02	0.35
20.00	0.43	0.34	0.22	0.15	-0.01	0.02	0.08	0.04	0.39
30.00	0.50	0.33	0.32	0.18	-0.01	0.03	0.07	0.04	0.45
40.00	0.56	0.33	0.40	0.21	-0.01	0.03	0.05	0.04	0.50
50.00	0.60	0.31	0.45	0.24	-0.00	0.02	0.04	0.02	0.55
60.00	0.61	0.28	0.48	0.25	0.00	-0.00	0.03	-0.00	0.56
70.00	0.60	0.24	0.49	0.24	0.01	-0.02	0.04	-0.02	0.55
80.00	0.56	0.19	0.48	0.21	0.02	-0.04	0.04	-0.04	0.51
90.00	0.50	0.11	0.45	0.18	0.03	-0.05	0.06	-0.04	0.45
100.00	0.43	0.03	0.41	0.15	0.02	-0.05	0.07	-0.04	0.39
110.00	0.38	-0.07	0.35	0.12	0.02	-0.03	0.08	-0.02	0.35
120.00	0.36	-0.17	0.29	0.11	0.01	-0.02	0.09	-0.00	0.34
130.00	0.38	-0.27	0.24	0.12	0.00	0.00	0.09	0.02	0.35
140.00	0.43	-0.36	0.18	0.15	-0.01	0.02	0.08	0.04	0.39
150.00	0.50	-0.44	0.13	0.18	-0.01	0.03	0.07	0.04	0.45
160.00	0.56	-0.51	0.08	0.21	-0.01	0.03	0.05	0.04	0.50
170.00	0.60	-0.55	0.04	0.24	-0.00	0.02	0.04	0.02	0.55
180.00	0.61	-0.56	0.00	0.25	0.00	-0.00	0.03	-0.00	0.56

File: QABBBA3i.000
Program: GPSEQVEL.CPP
GPS utilizing equivalent velocity approximation.
Solving for Px,Py,Pz,Theta.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 2.50 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	1.33	0.22	0.11	1.31	0.02	0.00	0.00	0.04	0.22
10.00	1.34	0.24	-0.11	1.31	-0.01	0.00	0.00	-0.06	0.22
20.00	1.37	0.33	-0.32	1.29	-0.05	0.00	0.00	-0.16	0.20
30.00	1.40	0.47	-0.49	1.23	-0.08	0.00	0.00	-0.25	0.16
40.00	1.41	0.60	-0.59	1.13	-0.11	0.00	0.00	-0.32	0.08
50.00	1.28	0.59	-0.55	0.99	-0.10	0.00	0.00	-0.31	-0.04
60.00	0.94	0.24	-0.32	0.85	-0.05	0.00	0.00	-0.14	-0.15
70.00	0.92	-0.38	0.00	0.84	0.05	0.00	0.00	0.13	-0.13
80.00	1.25	-0.78	0.15	0.96	0.11	0.00	0.00	0.30	0.01
90.00	1.38	-0.82	0.13	1.11	0.12	0.00	0.00	0.31	0.13
100.00	1.38	-0.65	0.08	1.22	0.10	0.00	0.00	0.24	0.20
110.00	1.35	-0.42	0.08	1.28	0.06	0.00	0.00	0.14	0.22
120.00	1.33	-0.21	0.14	1.31	0.02	0.00	0.00	0.04	0.22
130.00	1.34	-0.02	0.26	1.31	-0.01	0.00	0.00	-0.06	0.22
140.00	1.37	0.11	0.44	1.29	-0.05	0.00	0.00	-0.16	0.20
150.00	1.40	0.19	0.65	1.23	-0.08	0.00	0.00	-0.25	0.16
160.00	1.41	0.21	0.81	1.13	-0.11	0.00	0.00	-0.32	0.08
170.00	1.28	0.18	0.79	0.99	-0.10	0.00	0.00	-0.31	-0.04
180.00	0.94	0.15	0.37	0.85	-0.05	0.00	0.00	-0.14	-0.15

File: NABCBA3.000
Program: GPSNDVEL.CPP
GPS utilizing interpolated transmission velocities.
Solving for Px,Py,Pz,Theta,Phi,Mu.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 5.00 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	0.98	0.94	-0.00	0.26	0.02	-0.01	0.05	-0.00	0.94
10.00	1.00	0.94	0.22	0.27	0.01	0.00	0.05	0.02	0.96
20.00	1.04	0.91	0.43	0.28	-0.00	0.01	0.04	0.04	1.00
30.00	1.10	0.86	0.63	0.30	-0.00	0.01	0.04	0.04	1.06
40.00	1.16	0.79	0.79	0.31	-0.00	0.01	0.03	0.04	1.12
50.00	1.20	0.70	0.92	0.33	0.00	0.01	0.03	0.02	1.16
60.00	1.22	0.59	1.01	0.33	0.01	-0.00	0.02	-0.00	1.17
70.00	1.21	0.45	1.07	0.33	0.02	-0.01	0.02	-0.02	1.16
80.00	1.16	0.30	1.08	0.31	0.03	-0.02	0.03	-0.04	1.12
90.00	1.11	0.12	1.06	0.30	0.03	-0.02	0.04	-0.05	1.06
100.00	1.04	-0.07	1.00	0.28	0.03	-0.02	0.04	-0.04	1.00
110.00	1.00	-0.27	0.92	0.27	0.02	-0.02	0.05	-0.02	0.96
120.00	0.98	-0.47	0.82	0.26	0.02	-0.01	0.05	-0.00	0.94
130.00	1.00	-0.66	0.70	0.27	0.01	0.00	0.05	0.02	0.96
140.00	1.04	-0.83	0.57	0.28	-0.00	0.01	0.04	0.04	1.00
150.00	1.10	-0.97	0.43	0.30	-0.00	0.01	0.04	0.04	1.06
160.00	1.16	-1.08	0.29	0.31	-0.00	0.01	0.03	0.04	1.12
170.00	1.20	-1.15	0.15	0.33	0.00	0.01	0.03	0.02	1.16
180.00	1.22	-1.17	0.00	0.33	0.01	-0.00	0.02	-0.00	1.17

File: QABCBA3i.000
Program: GPSEQVEL.CPP
GPS utilizing equivalent velocity approximation.
Solving for Px,Py,Pz,Theta.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 5.00 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	2.74	0.02	0.24	2.73	0.02	0.00	0.00	0.09	0.02
10.00	2.72	-0.01	0.12	2.72	0.00	0.00	0.00	0.05	0.02
20.00	2.68	-0.00	0.02	2.68	-0.02	0.00	0.00	0.01	0.00
30.00	2.62	0.00	-0.04	2.62	-0.02	0.00	0.00	-0.01	-0.02
40.00	2.56	-0.01	-0.06	2.55	-0.02	0.00	0.00	-0.01	-0.05
50.00	2.50	-0.07	-0.03	2.50	-0.01	0.00	0.00	0.01	-0.07
60.00	2.48	-0.18	-0.00	2.48	0.01	0.00	0.00	0.06	-0.09
70.00	2.51	-0.31	0.02	2.49	0.03	0.00	0.00	0.11	-0.09
80.00	2.57	-0.41	0.00	2.53	0.05	0.00	0.00	0.16	-0.07
90.00	2.64	-0.46	-0.04	2.60	0.06	0.00	0.00	0.17	-0.04
100.00	2.70	-0.43	-0.09	2.66	0.05	0.00	0.00	0.17	-0.01
110.00	2.73	-0.34	-0.12	2.71	0.04	0.00	0.00	0.14	0.01
120.00	2.74	-0.22	-0.11	2.73	0.02	0.00	0.00	0.09	0.02
130.00	2.72	-0.11	-0.07	2.72	0.00	0.00	0.00	0.05	0.02
140.00	2.68	-0.02	-0.01	2.68	-0.02	0.00	0.00	0.01	0.00
150.00	2.62	0.03	0.02	2.62	-0.02	0.00	0.00	-0.01	-0.02
160.00	2.56	0.06	0.02	2.55	-0.02	0.00	0.00	-0.01	-0.05
170.00	2.50	0.07	-0.05	2.50	-0.01	0.00	0.00	0.01	-0.07
180.00	2.48	0.09	-0.16	2.48	0.01	0.00	0.00	0.06	-0.09

File: NABDBA3.000
Program: GPSNDVEL.CPP
GPS utilizing interpolated transmission velocities.
Solving for Px,Py,Pz,Theta,Phi,Mu.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 7.50 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	2.02	1.97	-0.01	0.42	0.02	-0.01	0.04	-0.00	1.97
10.00	2.03	1.95	0.40	0.42	0.01	0.00	0.04	0.02	1.99
20.00	2.08	1.88	0.79	0.43	0.01	0.01	0.04	0.04	2.03
30.00	2.14	1.76	1.15	0.44	0.00	0.01	0.03	0.04	2.09
40.00	2.20	1.59	1.46	0.45	0.01	0.01	0.03	0.04	2.15
50.00	2.25	1.37	1.72	0.46	0.01	0.00	0.02	0.02	2.20
60.00	2.26	1.11	1.91	0.47	0.02	-0.00	0.02	-0.00	2.21
70.00	2.25	0.81	2.04	0.46	0.03	-0.01	0.02	-0.03	2.20
80.00	2.20	0.48	2.10	0.45	0.03	-0.02	0.03	-0.04	2.15
90.00	2.14	0.13	2.09	0.44	0.04	-0.02	0.03	-0.05	2.09
100.00	2.08	-0.24	2.02	0.43	0.04	-0.02	0.04	-0.04	2.03
110.00	2.03	-0.62	1.89	0.42	0.03	-0.01	0.04	-0.03	1.99
120.00	2.02	-0.98	1.71	0.42	0.02	-0.01	0.04	-0.00	1.97
130.00	2.03	-1.32	1.49	0.42	0.01	0.00	0.04	0.02	1.99
140.00	2.08	-1.62	1.23	0.43	0.01	0.01	0.04	0.04	2.03
150.00	2.14	-1.87	0.95	0.44	0.00	0.01	0.03	0.04	2.09
160.00	2.20	-2.06	0.64	0.45	0.01	0.01	0.03	0.04	2.15
170.00	2.25	-2.17	0.33	0.46	0.01	0.00	0.02	0.02	2.20
180.00	2.26	-2.21	0.01	0.47	0.02	-0.00	0.02	-0.00	2.21

File: QABDBA3i.000
Program: GPSEQVEL.CPP
GPS utilizing equivalent velocity approximation.
Solving for Px,Py,Pz,Theta.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 7.50 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	6.42	-0.25	0.60	6.39	0.03	0.00	0.00	0.23	-0.25
10.00	6.40	-0.33	0.48	6.37	0.01	0.00	0.00	0.20	-0.24
20.00	6.35	-0.39	0.36	6.32	0.00	0.00	0.00	0.18	-0.24
30.00	6.28	-0.43	0.26	6.26	-0.00	0.00	0.00	0.17	-0.24
40.00	6.22	-0.47	0.18	6.20	-0.00	0.00	0.00	0.17	-0.24
50.00	6.18	-0.53	0.12	6.15	0.01	0.00	0.00	0.18	-0.25
60.00	6.17	-0.60	0.06	6.14	0.02	0.00	0.00	0.21	-0.25
70.00	6.19	-0.67	-0.03	6.15	0.04	0.00	0.00	0.24	-0.26
80.00	6.24	-0.72	-0.14	6.20	0.05	0.00	0.00	0.26	-0.26
90.00	6.31	-0.72	-0.26	6.26	0.05	0.00	0.00	0.27	-0.26
100.00	6.37	-0.65	-0.38	6.33	0.05	0.00	0.00	0.27	-0.25
110.00	6.41	-0.54	-0.47	6.37	0.04	0.00	0.00	0.26	-0.25
120.00	6.42	-0.40	-0.52	6.39	0.03	0.00	0.00	0.23	-0.25
130.00	6.40	-0.25	-0.53	6.37	0.01	0.00	0.00	0.20	-0.24
140.00	6.35	-0.12	-0.52	6.32	0.00	0.00	0.00	0.18	-0.24
150.00	6.28	-0.01	-0.50	6.26	-0.00	0.00	0.00	0.17	-0.24
160.00	6.22	0.08	-0.50	6.20	-0.00	0.00	0.00	0.17	-0.24
170.00	6.18	0.16	-0.52	6.15	0.01	0.00	0.00	0.18	-0.25
180.00	6.17	0.25	-0.55	6.14	0.02	0.00	0.00	0.21	-0.25

File: NABEBA3.000
Program: GPSNDVEL.CPP
GPS utilizing interpolated transmission velocities.
Solving for Px,Py,Pz,Theta,Phi,Mu.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 10.00 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	3.49	3.44	-0.01	0.59	0.03	-0.01	0.04	-0.00	3.44
10.00	3.51	3.40	0.66	0.60	0.02	-0.00	0.04	0.02	3.46
20.00	3.56	3.26	1.30	0.60	0.01	0.00	0.04	0.04	3.51
30.00	3.63	3.03	1.89	0.61	0.01	0.01	0.04	0.05	3.57
40.00	3.69	2.72	2.42	0.62	0.01	0.00	0.03	0.04	3.64
50.00	3.74	2.32	2.86	0.63	0.02	0.00	0.03	0.02	3.68
60.00	3.75	1.86	3.20	0.63	0.03	-0.00	0.03	-0.00	3.70
70.00	3.74	1.33	3.44	0.63	0.04	-0.01	0.03	-0.03	3.68
80.00	3.69	0.75	3.56	0.62	0.04	-0.01	0.03	-0.04	3.64
90.00	3.63	0.14	3.57	0.61	0.04	-0.02	0.04	-0.05	3.57
100.00	3.56	-0.49	3.47	0.60	0.04	-0.01	0.04	-0.05	3.51
110.00	3.51	-1.11	3.28	0.60	0.04	-0.01	0.04	-0.03	3.46
120.00	3.49	-1.71	2.99	0.59	0.03	-0.01	0.04	-0.00	3.44
130.00	3.51	-2.27	2.61	0.60	0.02	-0.00	0.04	0.02	3.46
140.00	3.56	-2.75	2.17	0.60	0.01	0.00	0.04	0.04	3.51
150.00	3.63	-3.15	1.68	0.61	0.01	0.01	0.04	0.05	3.57
160.00	3.69	-3.45	1.15	0.62	0.01	0.00	0.03	0.04	3.64
170.00	3.74	-3.64	0.58	0.63	0.02	0.00	0.03	0.02	3.68
180.00	3.75	-3.70	0.01	0.63	0.03	-0.00	0.03	-0.00	3.70

File: QABEBA3i.000
Program: GPSEQVEL.CPP
GPS utilizing equivalent velocity approximation.
Solving for Px,Py,Pz,Theta.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 10.00 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	13.24	-0.70	1.27	13.16	0.03	0.00	0.00	0.49	-0.69
10.00	13.22	-0.89	1.08	13.14	0.02	0.00	0.00	0.47	-0.69
20.00	13.15	-1.04	0.87	13.08	0.01	0.00	0.00	0.45	-0.68
30.00	13.08	-1.15	0.66	13.01	0.01	0.00	0.00	0.44	-0.66
40.00	13.00	-1.24	0.46	12.93	0.01	0.00	0.00	0.44	-0.65
50.00	12.95	-1.32	0.26	12.88	0.02	0.00	0.00	0.45	-0.65
60.00	12.94	-1.39	0.05	12.86	0.03	0.00	0.00	0.47	-0.65
70.00	12.97	-1.43	-0.18	12.89	0.04	0.00	0.00	0.49	-0.66
80.00	13.03	-1.42	-0.43	12.94	0.05	0.00	0.00	0.51	-0.67
90.00	13.11	-1.35	-0.68	13.02	0.06	0.00	0.00	0.52	-0.68
100.00	13.18	-1.21	-0.92	13.09	0.05	0.00	0.00	0.52	-0.69
110.00	13.23	-1.00	-1.11	13.15	0.05	0.00	0.00	0.51	-0.69
120.00	13.24	-0.75	-1.24	13.16	0.03	0.00	0.00	0.49	-0.69
130.00	13.22	-0.49	-1.31	13.14	0.02	0.00	0.00	0.47	-0.69
140.00	13.15	-0.23	-1.34	13.08	0.01	0.00	0.00	0.45	-0.68
150.00	13.08	0.00	-1.33	13.01	0.01	0.00	0.00	0.44	-0.66
160.00	13.00	0.22	-1.31	12.93	0.01	0.00	0.00	0.44	-0.65
170.00	12.95	0.44	-1.27	12.88	0.02	0.00	0.00	0.45	-0.65
180.00	12.94	0.65	-1.23	12.86	0.03	0.00	0.00	0.47	-0.65

File: NACDBB3.000 Program: GPSNDVEL.CPP
GPS utilizing interpolated transmission velocities.
Solving for Px,Py,Pz,Theta.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 7.50 m.
Error incorporation over 10000 iterations.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	2.03	1.99	0.19	0.35	0.15	0.00	0.00	0.07	1.99
10.00	2.05	1.97	0.39	0.35	0.14	0.00	0.00	0.07	2.01
20.00	2.09	1.89	0.76	0.36	0.15	0.00	0.00	0.08	2.04
30.00	2.14	1.77	1.12	0.37	0.15	0.00	0.00	0.08	2.09
40.00	2.20	1.60	1.43	0.38	0.15	0.00	0.00	0.07	2.15
50.00	2.23	1.38	1.70	0.39	0.15	0.00	0.00	0.07	2.18
60.00	2.24	1.11	1.89	0.39	0.15	0.00	0.00	0.07	2.20
70.00	2.23	0.80	2.04	0.39	0.15	0.00	0.00	0.07	2.19
80.00	2.19	0.46	2.10	0.38	0.15	0.00	0.00	0.08	2.15
90.00	2.14	0.21	2.09	0.37	0.15	0.00	0.00	0.08	2.09
100.00	2.09	0.30	2.02	0.36	0.15	0.00	0.00	0.08	2.04
110.00	2.05	0.63	1.90	0.35	0.15	0.00	0.00	0.08	2.01
120.00	2.04	0.99	1.73	0.35	0.15	0.00	0.00	0.07	1.99
130.00	2.05	1.31	1.51	0.35	0.15	0.00	0.00	0.07	2.00
140.00	2.09	1.60	1.26	0.36	0.15	0.00	0.00	0.07	2.04
150.00	2.14	1.85	0.98	0.37	0.15	0.00	0.00	0.08	2.09
160.00	2.20	2.04	0.67	0.38	0.15	0.00	0.00	0.07	2.15
170.00	2.23	2.16	0.36	0.39	0.15	0.00	0.00	0.07	2.18
180.00	2.25	2.20	0.19	0.39	0.15	0.00	0.00	0.07	2.20
Maximum error:									
180.00	3.22	-3.17	0.00	0.59	-0.03	0.00	0.00	-0.00	3.17

Standard deviation of total error at alpha = 180.0 degrees was 0.26 cm.

File: QACDBB3i.000 Program: GPSEQVEL.CPP
GPS utilizing equivalent velocity approximation.
Solving for Px,Py,Pz,Theta.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 7.50 m.
Error incorporation over 10000 iterations.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	2.27	1.43	0.87	1.16	0.15	0.00	0.00	0.33	1.43
10.00	2.33	1.47	0.89	1.21	0.15	0.00	0.00	0.33	1.48
20.00	2.54	1.63	1.02	1.27	0.15	0.00	0.00	0.33	1.73
30.00	2.86	1.81	1.26	1.41	0.16	0.00	0.00	0.33	2.06
40.00	3.26	1.96	1.70	1.55	0.16	0.00	0.00	0.33	2.50
50.00	3.58	1.89	2.19	1.65	0.16	0.00	0.00	0.34	2.83
60.00	3.66	1.61	2.52	1.67	0.16	0.00	0.00	0.34	2.92
70.00	3.56	1.26	2.61	1.65	0.16	0.00	0.00	0.33	2.81
80.00	3.29	1.00	2.46	1.56	0.16	0.00	0.00	0.33	2.51
90.00	2.90	0.88	2.09	1.44	0.16	0.00	0.00	0.33	2.09
100.00	2.56	0.88	1.73	1.30	0.16	0.00	0.00	0.33	1.73
110.00	2.36	0.90	1.48	1.22	0.15	0.00	0.00	0.33	1.51
120.00	2.26	1.00	1.30	1.17	0.15	0.00	0.00	0.33	1.41
130.00	2.32	1.15	1.26	1.19	0.15	0.00	0.00	0.33	1.48
140.00	2.53	1.41	1.24	1.29	0.15	0.00	0.00	0.33	1.70
150.00	2.89	1.87	1.24	1.41	0.16	0.00	0.00	0.33	2.11
160.00	3.27	2.37	1.12	1.54	0.16	0.00	0.00	0.34	2.50
170.00	3.55	2.76	0.98	1.63	0.16	0.00	0.00	0.33	2.81
180.00	3.65	2.90	0.90	1.66	0.16	0.00	0.00	0.34	2.91
Maximum error:									
50.00	9.50	6.84	5.16	4.10	0.04	0.00	0.00	-0.70	8.37

Standard deviation of total error at alpha = 50.0 degrees was 1.34 cm.

File: NBADBB3.000 Program: GPSNDVEL.CPP
GPS utilizing interpolated transmission velocities.
Solving for Px,Py,Pz,Theta.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 7.50 m.
Error incorporation over 10000 iterations.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	2.60	2.55	0.22	0.39	0.23	0.00	0.00	0.08	2.55
10.00	2.62	2.52	0.51	0.39	0.23	0.00	0.00	0.09	2.57
20.00	2.67	2.43	0.99	0.41	0.23	0.00	0.00	0.09	2.62
30.00	2.74	2.27	1.45	0.42	0.23	0.00	0.00	0.09	2.69
40.00	2.80	2.04	1.85	0.43	0.23	0.00	0.00	0.09	2.75
50.00	2.86	1.76	2.19	0.44	0.23	0.00	0.00	0.08	2.81
60.00	2.87	1.41	2.45	0.45	0.23	0.00	0.00	0.08	2.82
70.00	2.85	1.02	2.61	0.44	0.23	0.00	0.00	0.08	2.80
80.00	2.80	0.59	2.69	0.43	0.24	0.00	0.00	0.09	2.75
90.00	2.74	0.24	2.69	0.42	0.23	0.00	0.00	0.09	2.69
100.00	2.67	0.37	2.60	0.40	0.23	0.00	0.00	0.09	2.62
110.00	2.62	0.81	2.44	0.40	0.23	0.00	0.00	0.09	2.57
120.00	2.61	1.28	2.22	0.39	0.23	0.00	0.00	0.08	2.56
130.00	2.62	1.70	1.93	0.39	0.23	0.00	0.00	0.09	2.57
140.00	2.68	2.08	1.61	0.41	0.23	0.00	0.00	0.09	2.63
150.00	2.74	2.38	1.24	0.42	0.23	0.00	0.00	0.09	2.69
160.00	2.80	2.62	0.85	0.43	0.23	0.00	0.00	0.09	2.75
170.00	2.86	2.78	0.45	0.44	0.24	0.00	0.00	0.08	2.81
180.00	2.86	2.82	0.21	0.44	0.23	0.00	0.00	0.08	2.82
Maximum error:									
80.00	3.98	0.60	3.89	0.60	0.27	0.00	0.00	0.03	3.94

Standard deviation of total error at alpha = 80.0 degrees was 0.29 cm.

File: QBADBB3i.000 Program: GPSEQVEL.CPP
GPS utilizing equivalent velocity approximation.
Solving for Px,Py,Pz,Theta.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 7.50 m.
Error incorporation over 10000 iterations.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	3.13	2.11	1.14	1.47	0.25	0.00	0.00	0.43	2.11
10.00	3.23	2.18	1.18	1.51	0.25	0.00	0.00	0.43	2.21
20.00	3.43	2.30	1.37	1.58	0.25	0.00	0.00	0.43	2.43
30.00	3.77	2.45	1.71	1.73	0.25	0.00	0.00	0.44	2.77
40.00	4.13	2.50	2.18	1.85	0.26	0.00	0.00	0.44	3.15
50.00	4.41	2.35	2.71	1.93	0.26	0.00	0.00	0.44	3.45
60.00	4.57	2.04	3.14	2.00	0.26	0.00	0.00	0.44	3.60
70.00	4.45	1.60	3.27	1.97	0.27	0.00	0.00	0.44	3.48
80.00	4.13	1.28	3.10	1.85	0.27	0.00	0.00	0.44	3.16
90.00	3.76	1.15	2.76	1.72	0.26	0.00	0.00	0.43	2.76
100.00	3.44	1.19	2.41	1.58	0.25	0.00	0.00	0.43	2.43
110.00	3.20	1.26	2.09	1.52	0.25	0.00	0.00	0.43	2.17
120.00	3.16	1.41	1.95	1.48	0.25	0.00	0.00	0.43	2.13
130.00	3.19	1.64	1.81	1.49	0.25	0.00	0.00	0.43	2.18
140.00	3.43	1.98	1.76	1.59	0.25	0.00	0.00	0.44	2.41
150.00	3.74	2.44	1.67	1.69	0.25	0.00	0.00	0.44	2.75
160.00	4.12	2.96	1.49	1.85	0.26	0.00	0.00	0.44	3.13
170.00	4.41	3.38	1.28	1.95	0.26	0.00	0.00	0.44	3.44
180.00	4.55	3.58	1.18	2.00	0.26	0.00	0.00	0.44	3.59
Maximum error:									
60.00	14.08	5.75	11.42	5.90	0.12	0.00	0.00	0.26	12.77

Standard deviation of total error at alpha = 60.0 degrees was 2.07 cm.

File: QAADBE3i.000
Program: GPSEQVEL.CPP
GPS utilizing equivalent velocity approximation.
Solving for Px,Py,Pz,Theta.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 3.00
Platform pivot at radius of 1.50 m and depth of 7.50 m.
Error incorporation over 10000 iterations.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	2.23	1.13	1.24	1.18	0.04	0.00	0.00	0.47	1.13
10.00	2.30	1.20	1.24	1.21	0.04	0.00	0.00	0.47	1.21
20.00	2.50	1.37	1.27	1.31	0.04	0.00	0.00	0.47	1.43
30.00	2.81	1.59	1.39	1.44	0.03	0.00	0.00	0.47	1.80
40.00	3.16	1.76	1.62	1.57	0.03	0.00	0.00	0.48	2.23
50.00	3.42	1.75	1.98	1.67	0.02	0.00	0.00	0.47	2.53
60.00	3.53	1.58	2.27	1.70	0.02	0.00	0.00	0.48	2.64
70.00	3.44	1.43	2.36	1.67	0.04	0.00	0.00	0.48	2.55
80.00	3.16	1.30	2.17	1.58	0.05	0.00	0.00	0.47	2.22
90.00	2.83	1.27	1.82	1.45	0.05	0.00	0.00	0.48	1.82
100.00	2.50	1.21	1.45	1.32	0.05	0.00	0.00	0.47	1.43
110.00	2.32	1.20	1.25	1.22	0.05	0.00	0.00	0.48	1.20
120.00	2.23	1.16	1.16	1.18	0.04	0.00	0.00	0.47	1.13
130.00	2.30	1.19	1.18	1.21	0.04	0.00	0.00	0.47	1.21
140.00	2.50	1.34	1.23	1.31	0.04	0.00	0.00	0.47	1.42
150.00	2.81	1.67	1.28	1.44	0.03	0.00	0.00	0.47	1.81
160.00	3.15	2.09	1.31	1.57	0.03	0.00	0.00	0.48	2.21
170.00	3.44	2.51	1.29	1.67	0.02	0.00	0.00	0.48	2.56
180.00	3.54	2.64	1.29	1.70	0.02	0.00	0.00	0.48	2.65
Maximum error:									
60.00	5.46	4.48	2.61	1.71	0.07	0.00	0.00	-0.95	4.52

Standard deviation of total error at alpha = 60.0 degrees was 0.84 cm.

File: NAADBA3.rd1
Program: GPSNDVEL.CPP
GPS utilizing interpolated transmission velocities.
Solving for Px,Py,Pz,Theta,Phi,Mu.
Temperatures: Tw = 30.00, Tc = 25.00, Tt = 30.00, Tb = 20.00.
Compensation factor = 3.00
Platform pivot at radius of 1.00 m and depth of 7.50 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	1.70	1.65	-0.00	0.43	0.00	0.01	0.01	-0.00	1.65
10.00	1.71	1.62	0.32	0.44	-0.03	0.01	0.01	0.02	1.65
20.00	1.73	1.56	0.62	0.44	-0.06	0.02	0.02	0.03	1.68
30.00	1.76	1.45	0.90	0.44	-0.06	0.02	0.02	0.03	1.71
40.00	1.79	1.30	1.16	0.45	-0.05	0.02	0.02	0.03	1.74
50.00	1.82	1.11	1.37	0.45	-0.03	0.01	0.02	0.02	1.76
60.00	1.82	0.89	1.53	0.45	0.01	0.01	0.02	-0.00	1.77
70.00	1.82	0.63	1.64	0.45	0.04	0.00	0.02	-0.02	1.76
80.00	1.79	0.35	1.70	0.45	0.07	-0.00	0.02	-0.03	1.74
90.00	1.76	0.06	1.71	0.44	0.07	-0.01	0.02	-0.03	1.71
100.00	1.73	-0.24	1.66	0.44	0.06	-0.00	0.02	-0.03	1.68
110.00	1.71	-0.54	1.57	0.44	0.04	0.00	0.01	-0.02	1.65
120.00	1.70	-0.82	1.43	0.43	0.00	0.01	0.01	-0.00	1.65
130.00	1.71	-1.09	1.25	0.44	-0.03	0.01	0.01	0.02	1.65
140.00	1.73	-1.32	1.04	0.44	-0.06	0.02	0.02	0.03	1.68
150.00	1.76	-1.51	0.80	0.44	-0.06	0.02	0.02	0.03	1.71
160.00	1.79	-1.65	0.55	0.45	-0.05	0.02	0.02	0.03	1.74
170.00	1.82	-1.74	0.28	0.45	-0.03	0.01	0.02	0.02	1.76
180.00	1.82	-1.77	0.00	0.45	0.01	0.01	0.02	-0.00	1.77

File: QAADBA3i.rd1
Program: GPSEQVEL.CPP
GPS utilizing equivalent velocity approximation.
Solving for Px,Py,Pz,Theta.
Temperatures: Tw = 40.00, Tc = 35.00, Tt = 40.00, Tb = 30.00.
Compensation factor = 3.00
Platform pivot at radius of 1.00 m and depth of 7.50 m.
No added error to system.

Alpha Position (deg)	Total Error (cm)	x Error (cm)	y Error (cm)	z Error (cm)	Theta Error (deg)	Phi Error (deg)	Mu Error (deg)	Alpha error (deg)	radial error (cm)
0.00	1.21	0.73	-0.07	0.97	0.01	0.00	0.00	-0.04	0.73
10.00	1.30	0.83	0.14	0.99	-0.00	0.00	0.00	-0.00	0.84
20.00	1.56	1.06	0.42	1.07	-0.02	0.00	0.00	0.02	1.14
30.00	1.94	1.33	0.81	1.16	-0.02	0.00	0.00	0.02	1.56
40.00	2.34	1.50	1.29	1.25	-0.02	0.00	0.00	0.02	1.98
50.00	2.63	1.47	1.75	1.31	-0.01	0.00	0.00	-0.00	2.29
60.00	2.74	1.24	2.06	1.33	0.01	0.00	0.00	-0.02	2.40
70.00	2.64	0.86	2.12	1.31	0.03	0.00	0.00	-0.05	2.29
80.00	2.34	0.47	1.92	1.25	0.05	0.00	0.00	-0.07	1.98
90.00	1.95	0.15	1.55	1.17	0.04	0.00	0.00	-0.09	1.55
100.00	1.57	-0.05	1.14	1.07	0.03	0.00	0.00	-0.09	1.14
110.00	1.31	-0.17	0.83	1.00	0.02	0.00	0.00	-0.07	0.84
120.00	1.21	-0.31	0.67	0.97	0.01	0.00	0.00	-0.04	0.73
130.00	1.30	-0.53	0.65	0.99	-0.00	0.00	0.00	-0.00	0.84
140.00	1.56	-0.89	0.71	1.07	-0.02	0.00	0.00	0.02	1.14
150.00	1.94	-1.37	0.74	1.16	-0.02	0.00	0.00	0.02	1.56
160.00	2.34	-1.87	0.65	1.25	-0.02	0.00	0.00	0.02	1.98
170.00	2.63	-2.25	0.40	1.31	-0.01	0.00	0.00	-0.00	2.29
180.00	2.74	-2.40	0.04	1.33	0.01	0.00	0.00	-0.02	2.40

Appendix 4a
GPS Code; Interpolated Velocity System

This code is designed to examine the effectiveness of the global positioning system which utilizes interpolated nodal velocities to obtain the transmission velocities. It is configured such that any of the system parameters may be altered without requiring significant changes to the code.

```
/*==================================================================*/
/* GPSNDVEL.CPP                                                     */
/* Program designed to evaluate the use of acoustic transceivers in the  */
/* location of an underwater mobile platform.  It has been designed to allow */
/* many different schemes to be examined without significant changes to the  */
/* program code.                                                    */
/* Program utilizes interpolated velocities to reduce system unknowns.  */
/* Programmer: Allen Maples                                         */
/* Program code: C                                                  */
/* Development completed: 4/93                                      */
/*==================================================================*/

#include <math.h>
#include <dos.h>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>

/* Defined Constants */
const int N_FXD = 3; /* Total number of fixed xcvrs. */
const int N_MBL = 3; /* Total number of mobile xcvrs. */
const int DIMS = 6; /* Max number of system dimensions. */
const int SYS_EQS = N_FXD*N_MBL; /* Max number of system equations. */
const double EPS = 0.00001; /* Tolerance value. */
const double TINY = 1e-20; /* Small value. */
const long double PI = (4.01*atan(1.01));
const double RAD_TO_DEG = 180.0/PI; /* Conversion factor. */

/* Functions */
void   define_known();
void   aquire_inclns( int,int );
void   aquire_ToF();
double point_vel( double[] );
void   evaluate_rtn_mtx( double[] );
double eval_tof( double[],double[] );
void   aquire_F_Sing();
void   aquire_M_Sing();
void   determine_nodal_vels();
void   determine_interpolated_vels();
void   adjust_active();
void   establish_approx( double[] );
void   tetra_solve( double*,double[][3],double,double,double,int );
void   store_true( double[][3],double[][N_MBL],double[],double[][3],double[] );
void   incorporate_errors( double[][N_MBL] );
void   solve_system( double[],double[] );
void   reset_true( double[][3],double[][N_MBL],double[],double[][3],double[] );
double objective_function( double[] );
void   jacobian_evaluation( double[][DIMS],double[] );
void   functions_evaluation( double[],double[] );
int    least_squares_solution( double*,double* );
double dot_vector( double*,double*,int,int,int );
double step_evaluation( double[],double[],double );
```

```
double  pattern_search( double[] );
int     matrix_solve( double*, double* );
int     lu_dcmp( double*,int* );
void    lu_bksb( double*,int*,double*);
double  convert_angle( double );
void    move( double[],double[],double[],double );
double  vector_magnitude( double[],int );
void    normalize_vector( double[],int );

/* Global Variables */
double
Fxyz[N_FXD][3], /* Fixed transceiver global locations. */
Mxyz[N_MBL][3], /* Mobile transceiver global locations. */
Mqrs[N_MBL][3], /* Locations of mobile transceivers relative to Pivot. */
Pxyz[3], /* Global location of Pivot. */
Model_Vel[N_FXD][N_MBL], /* F <--> M model transmission velocities. */
ToF[N_FXD][N_MBL], /* F <--> M times of flight (sound). */
F_Sing_Vel[N_FXD], /* Sing-around velocities; {F0-F1,F1-F2,F2-F0}. */
M_Sing_Vel[N_MBL], /* Sing-around velocities; {M0-M1,M1-M2,M2-M0}. */
F_Sing_ToF[N_FXD], /* Sing-around tof; {F0-F1,F1-F2,F2-F0}. */
M_Sing_ToF[N_MBL], /* Sing-around tof; {M0-M1,M1-M2,M2-M0}. */
F_Node_Vel[N_FXD], /* Nodal velocities of fixed xcvrs. */
M_Node_Vel[N_MBL], /* Nodal velocities of mobile xcvrs. */
Intrp_Vel[N_FXD][N_MBL], /* Approx of transmission vels using nodal vels. */
Rotation[3][3], /* Theta,Phi,Mu rotation matrix. */
Tht_Phi_Mu[3], /* Platform rotation angles. */
Comp_Fctr = 3.0, /* Compensation factor. */
Alpha = 0.0, /* Angular position of pivot in vessel. */

/* Vessel Model Characteristics */
Salinity = 0.0, /* Water salenity. */
Wall_Temp = 30.0, /* Parabolic/radial temperature distribution value. */
Center_Temp = 25.0, /* Parabolic/radial temperature distribution value. */
Top_Temp = 30.0, /* Linear/depth temperature distribution value. */
Bottom_Temp = 20.0, /* Linear/depth temperature distribution value. */
Default_Temp = 30.0, /* Default temperature of vessel water. */
Vessel_Depth = 10.0, /* Distance from F plane to Bottom_Temp location. */
Rbt_Rad = 1.5, /* Radius that robot pivot resides at. */
Vsl_Rad = 2.286; /* Radius of reactor vessel in meters. */

int
Active_F[N_FXD] = { 1,1,1 }, /* Indicates which fixed xcvrs are active. */
Solve_Angles[3] = { 1,0,0 }, /* Which angles (theta,phi,mu) must be solved. */
N_Fxd, /* Number of fixed transceivers active in the solution process. */
Dims, /* Number of dimensions for the system being solved. */
Sys_Eqs, /* Number of system equations for the system being solved. */
Add_Error = 1; /* Indicates whether or not error is to be incorporated. */

/*==================================================================*/
/*==================================================================*/
void main()
{

double
guess[DIMS],
answer[DIMS],
true_fxyz[N_FXD][3],
true_tof[N_FXD][N_MBL],
true_pivot[3],
true_mqrs[N_MBL][3],
true_angles[3],
err_mtx[2*N_FXD+N_MBL+1][N_MBL],
revs,
division = PI/18.0,
t_err,
p_err,
m_err,
tot_err,
```

```
x_err,
y_err,
z_err,
r_err,
a_err,
t_tot_err = 0.0,
t_x_err = 0.0,
t_t_err = 0.0,
t_p_err = 0.0,
t_m_err = 0.0,
t_y_err = 0.0,
t_r_err = 0.0,
t_a_err = 0.0,
t_z_err = 0.0,
max_err[10] = {0,0,0,0,0,0,0,0,0,0},
std_dev,
std_dev_at_max,
temp,
radius,
depth;

int
i,
locations=19,
rndm_iters=5;

if (!Add_Error) rndm_iters = 1;

clrscr();

define_known();

printf( "\nProgram: GPSNDVEL.CPP" );
printf( "\nGPS utilizing interpolated transmission velocities." );
printf( "\nSolving for Px,Py,Pz" );
if (Solve_Angles[0]) printf( ",Theta" );
if (Solve_Angles[1]) printf( ",Phi" );
if (Solve_Angles[2]) printf( ",Mu" );
printf( ".\nTemperatures: Tw = %5.2lf, Tc = %5.2lf, Tt = %5.2lf, Tb = %5.2lf.",
        Wall_Temp,Center_Temp,Top_Temp,Bottom_Temp );
printf( "\nCompensation factor = %4.2lf", Comp_Fctr );
printf( "\nPlatform pivot at radius of %4.2lf m and depth of %4.2lf m.",
        Rbt_Rad,-Pxyz[2] );
if (Add_Error) printf( "\nError incorporation over %d iterations.", rndm_iters );
else printf( "\nNo added error to system." );
printf( "\n\n\n" );
/*==================================================================*/

printf( "  Alpha | Total |   x   |   y   |   z   | Theta|  Phi |  Mu  | Alpha| radial" );
printf( "\n" );
printf( "Position| Error | Error | Error | Error | Error| Error| Error| error| error" );
printf( "\n" );
printf( "  (deg) |  (cm) |  (cm) |  (cm) |  (cm) | (deg)| (deg)| (deg)| (deg)|  (cm)\n" );
printf( "--------|-------|-------|-------|-------|------|------|------|------|-------" );

Alpha = 0.0;

for (i=0; i<locations; i++) {

        define_known();
        /* aquire_inclns( 1,1 ); */
        aquire_ToF();
        aquire_M_Sing();
        aquire_F_Sing();
        determine_nodal_vels();
        determine_interpolated_vels();
        adjust_active();
        establish_approx( guess );
```

```
    t_tot_err=0.0;
    t_x_err=0.0;
    t_t_err=0.0;
    t_p_err=0.0;
    t_m_err=0.0;
    t_y_err=0.0;
    t_r_err=0.0;
    t_a_err=0.0;
    t_z_err=0.0;

    randomize();

std_dev = 0.0;

    for (int jj=0; jj<rndm_iters; jj++) {

    if (Add_Error) {
            store_true( true_fxyz,true_tof,true_pivot,true_mqrs,true_angles );
            incorporate_errors( err_mtx );
    }

    solve_system( guess,answer );

    if (Add_Error)
            reset_true( true_fxyz,true_tof,true_pivot,true_mqrs,true_angles );

    x_err = answer[0]-Pxyz[0];
    y_err = answer[1]-Pxyz[1];
    z_err = answer[2]-Pxyz[2];
    r_err = sqrt( answer[0]*answer[0] + answer[1]*answer[1]) -
            sqrt(Pxyz[0]*Pxyz[0] + Pxyz[1]*Pxyz[1]);
    if (Solve_Angles[0])
            t_err = convert_angle( answer[3]-Tht_Phi_Mu[0] );
    else t_err = 0.0;
    if (Solve_Angles[1])
            p_err = convert_angle( answer[4]-Tht_Phi_Mu[1] );
    else p_err = 0.0;
    if (Solve_Angles[2])
            m_err = convert_angle( answer[5]-Tht_Phi_Mu[2] );
    else m_err = 0.0;
    a_err = convert_angle( atan2(answer[1],answer[0])-atan2(Pxyz[1],Pxyz[0]) );
    tot_err = sqrt(x_err*x_err + y_err*y_err + z_err*z_err);

    if (Add_Error) {
            t_tot_err += tot_err;
            t_t_err += fabs(t_err);
            t_p_err += fabs(p_err);
            t_m_err += fabs(m_err);
            t_x_err += fabs(x_err);
            t_y_err += fabs(y_err);
            t_r_err += fabs(r_err);
            t_a_err += fabs(a_err);
            t_z_err += fabs(z_err);

            std_dev += tot_err*tot_err;

            if (tot_err > max_err[1]) {
                    max_err[0] = Alpha;
                    max_err[1] = tot_err;
                    max_err[2] = x_err;
                    max_err[3] = y_err;
                    max_err[4] = z_err;
                    max_err[5] = t_err;
                    max_err[6] = p_err;
                    max_err[7] = m_err;
                    max_err[8] = a_err;
                    max_err[9] = r_err;
```

```
        }
        }
        else {
                t_tot_err += tot_err;
                t_t_err += t_err;
                t_p_err += p_err;
                t_m_err += m_err;
                t_x_err += x_err;
                t_y_err += y_err;
                t_r_err += r_err;
                t_a_err += a_err;
                t_z_err += z_err;
        }

        if (jj==(rndm_iters-1)) {
                if (Add_Error) {
                        std_dev = sqrt( (std_dev - t_tot_err*t_tot_err/
                                        double(rndm_iters))/double(jj) );
                        if (int(RAD_TO_DEG*Alpha)==int(RAD_TO_DEG*max_err[0]))
                                std_dev_at_max = std_dev;
                }

                t_err = t_t_err/rndm_iters;
                p_err = t_p_err/rndm_iters;
                m_err = t_m_err/rndm_iters;
                tot_err = t_tot_err/rndm_iters;
                x_err = t_x_err/rndm_iters;
                y_err = t_y_err/rndm_iters;
                a_err = t_a_err/rndm_iters;
                z_err = t_z_err/rndm_iters;
                r_err = t_r_err/rndm_iters;

                printf( "\n" );
                printf( " %6.2lf |%6.2lf |%6.2lf |%6.2lf |%6.2lf |",
                (Alpha)*RAD_TO_DEG,
                1.0e2*tot_err,
                1.0e2*x_err,
                1.0e2*y_err,
                1.0e2*z_err );
                printf( "%5.2lf |%5.2lf |%5.2lf |%5.2lf |%6.2lf",
                t_err*RAD_TO_DEG,
                p_err*RAD_TO_DEG,
                m_err*RAD_TO_DEG,
                a_err*RAD_TO_DEG,
                1.0e2*r_err );
        }
}

Alpha += division;

}

if (Add_Error) {
        printf( "\n\nMaximum error:\n" );
        printf( " %6.2lf |%6.2lf |%6.2lf |%6.2lf |%6.2lf ",
        max_err[0]*RAD_TO_DEG,
        1.0e2*max_err[1],
        1.0e2*max_err[2],
        1.0e2*max_err[3],
        1.0e2*max_err[4] );
        printf( "|%5.2lf |%5.2lf |%5.2lf |%5.2lf |%6.2lf",
        max_err[5]*RAD_TO_DEG,
        max_err[6]*RAD_TO_DEG,
        max_err[7]*RAD_TO_DEG,
        max_err[8]*RAD_TO_DEG,
        1.0e2*max_err[9] );
        printf( "\nStandard deviation of total error at alpha = " );
        printf( "%4.1lf degrees was %5.2lf cm.",
```

```
                max_err[0]*RAD_TO_DEG,std_dev_at_max*1e2 );
}

}
/*=================================================================*/
/*=================================================================*/
void adjust_active()
{

/* Adjusts global system parameters N_Fxd,Dims,and Sys_Eqs to correspond to
/* the current situation.  */

int i,xcvr = 0;

for (i=0; i<N_FXD; i++) if (Active_F[i]) xcvr++;

N_Fxd = xcvr;
Dims = 3 + Solve_Angles[0] + Solve_Angles[1] + Solve_Angles[2];
Sys_Eqs = N_Fxd*N_MBL;

}
/*=================================================================*/
void define_known()
{

double mbl_rad = 0.5; /* Determines separation of mobile transceivers. */

/* Fixed transceiver global positions. */
Fxyz[0][0] = Vsl_Rad;
Fxyz[0][1] = 0.0;
Fxyz[0][2] = 0.0;
Fxyz[1][0] = -Vsl_Rad*sin(PI/6.);
Fxyz[1][1] = Vsl_Rad*cos(PI/6.0);
Fxyz[1][2] = 0.0;
Fxyz[2][0] = -Vsl_Rad*sin(PI/6.);
Fxyz[2][1] = -Vsl_Rad*cos(PI/6.0);
Fxyz[2][2] = 0.0;

/* Mobile transceiver positions relative to pivot. */
Mqrs[0][0] = 0.0;
Mqrs[0][1] = -mbl_rad*cos(PI/6.0);
Mqrs[0][2] = -mbl_rad*sin(PI/6.0);
Mqrs[1][0] = 0.0;
Mqrs[1][1] = mbl_rad*cos(PI/6.0);
Mqrs[1][2] = -mbl_rad*sin(PI/6.0);
Mqrs[2][0] = 0.0;
Mqrs[2][1] = 0.0;
Mqrs[2][2] = mbl_rad;

/* These values aren't really known but are needed for the model. */
Pxyz[0] = Rbt_Rad*cos(Alpha);
Pxyz[1] = Rbt_Rad*sin(Alpha);
Pxyz[2] = -7.5;

Tht_Phi_Mu[0] = convert_angle( Alpha );
Tht_Phi_Mu[1] = 0.1;
Tht_Phi_Mu[2] = 0.1;

}
/*=================================================================*/
void aquire_inclns(int p,int m)
{
/* In the real system, this function would be used to aquire the inclinometer
/* data if inclinometers were utilized. */

if (p) Tht_Phi_Mu[1] = 0.1;
if (m) Tht_Phi_Mu[2] = 0.1;
```

```
}
/*==========================================================================*/
void determine_interpolated_vels()
{

/* Evaluating the interpolated velocities between transceivers. */

int
i,j;

for (i=0; i<N_FXD; i++) {
        for (j=0; j<N_MBL; j++) {
        /* If averaging: */
                Intrp_Vel[i][j] = (F_Node_Vel[i] + M_Node_Vel[j])/2.0;
                /* If using Harmonic Mean: */
                /* Intrp_Vel[i][j] = 2.0*F_Node_Vel[i]*M_Node_Vel[j]/
                /*              (F_Node_Vel[i] + M_Node_Vel[j]); */
        }
}

}
/*==========================================================================*/
void determine_nodal_vels()
{

/* Evaluating the local velocities at each of the transceivers. */

double
vf_center = 0.0,
vm_center = 0.0;

int
i,j;

for (i=0; i<3; i++) {
        vf_center += F_Sing_Vel[i];
        vm_center += M_Sing_Vel[i];
}
vf_center /= 3.0;
vm_center /= 3.0;

for (i=0; i<3; i++) {
        if (i==0) j=1;
        else if (i==1) j=2;
        else if (i==2) j=0;
        F_Node_Vel[i] = F_Sing_Vel[j] + Comp_Fctr*(vf_center - F_Sing_Vel[j]);
        M_Node_Vel[i] = M_Sing_Vel[j] + Comp_Fctr*(vm_center - M_Sing_Vel[j]);
}

}
/*==========================================================================*/
void aquire_ToF()
{

/* This function would, in practice, fire each of the transceivers in some
/* logical manner in order to aquire a time of flight between each.
/* This is a model of the results. */

int
i,j,k;

evaluate_rtn_mtx( Tht_Phi_Mu );

/* Calculating the true global (xyz) positions of mobile transceivers by
/* rotating their relative coordinates (qrs) by the indicated theta,phi,mu
/* and translating the system by the position of the pivot.  Thus, the mobile
/* platform is positioned by choosing the pivot and the rotation angles. */
for ( i=0; i<N_MBL; i++) {
```

```
        for ( j=0; j<3; j++) {
                Mxyz[i][j]=0.0;
                for ( k=0; k<3; k++) Mxyz[i][j] += Rotation[j][k]*Mqrs[i][k];
                Mxyz[i][j] += Pxyz[j];
        }
}

for (i=0; i<N_FXD; i++)
        for (j=0; j<N_MBL; j++)
                ToF[i][j] = eval_tof( &Fxyz[i][0],&Mxyz[j][0] );

}
/*==================================================================================*/
double eval_tof( double pt_a[3],double pt_b[3] )
{

/* Numerical integration (trapazoidal) process to determine the transmission
/* velocity between two points. */

double
delta = 0.0,
inc,
slope[3],
pos[3],
tof=0.0;

int
divisions = 50, /* Integration resolution. */
i,j,k;

for (i=0; i<3; i++) {
        pos[i] = pt_a[i];
        slope[i] = pt_b[i]-pt_a[i];
        delta += slope[i]*slope[i];
}
delta = sqrt( delta );
for (i=0; i<3; i++) slope[i] /= delta;
inc = delta/double(divisions);
for (i=0; i<=divisions; i++) {
        if ((i!=0)&&(i!=divisions))
                tof += 1.0/point_vel( pos );
        else tof += 0.5/point_vel( pos );
        for (j=0; j<3; j++) pos[j] += inc*slope[j];
}
tof *= inc;

return( tof );

}
/*==================================================================================*/
double point_vel( double pt[3] )
{

/* Evaluates the velocity of sound at a point based on the active temperature
/* distribution models. */

double
tmp=0.0,
tmp_par,
tmp_lin,
r;

int
i=0;

r = sqrt( pt[0]*pt[0]+pt[1]*pt[1] );
tmp_par = ((Wall_Temp-Center_Temp)*r*r)/(Vsl_Rad*Vsl_Rad) + Center_Temp;
tmp_lin = ((Top_Temp-Bottom_Temp)*pt[2])/Vessel_Depth + Top_Temp;
```

```
if ( tmp_par>EPS ) { tmp += tmp_par; i++; }
if ( tmp_lin>EPS ) { tmp += tmp_lin; i++; }
if ( tmp>EPS ) tmp /= double(i);
else tmp = Default_Temp;

return( 1449.2 + 4.6*tmp - 0.055*tmp*tmp + 0.00029*tmp*tmp*tmp +
                (1.34 - 0.01*tmp)*(Salinity - 35.0) + 0.016*fabs(pt[2]) );

}
/*==================================================================*/
void aquire_F_Sing()
{

/* Aquires tof among each of the fixed transceivers. */
/* This is a model of those results. */

double
delta;

int
i,j,k;

for (i=0; i<3; i++) {
        if (i==0) j=1;
        else if (i==1) j=2;
        else if (i==2) j=0;

    delta = 0.0;
        for (k=0; k<3; k++)
                delta += (Fxyz[i][k] - Fxyz[j][k])*(Fxyz[i][k] - Fxyz[j][k]);
        delta = sqrt( delta );

        F_Sing_ToF[i] = eval_tof( &Fxyz[i][0],&Fxyz[j][0] );
    F_Sing_Vel[i] = delta/F_Sing_ToF[i];
}

}
/*==================================================================*/
void aquire_M_Sing()
{

/* Aquires tof among each of the mobile transceivers. */
/* This is a model of those results. */

double
delta;

int
i,j,k;

evaluate_rtn_mtx( Tht_Phi_Mu );

/* Calculating the true global (xyz) positions of mobile transceivers by
/* rotating their relative coordinates (qrs) by the indicated theta,phi,mu
/* and translating the system by the position of the pivot.  Thus, the mobile
/* platform is positioned by choosing the pivot and the rotation angles. */
for ( i=0; i<N_MBL; i++) {
        for ( j=0; j<3; j++) {
                Mxyz[i][j]=0.0;
                for ( k=0; k<3; k++) Mxyz[i][j] += Rotation[j][k]*Mqrs[i][k];
                Mxyz[i][j] += Pxyz[j];
        }
}

for (i=0; i<3; i++) {
        if (i==0) j=1;
        else if (i==1) j=2;
        else if (i==2) j=0;
```

```
    delta = 0.0;
        for (k=0; k<3; k++)
                delta += (Mxyz[i][k] - Mxyz[j][k])*(Mxyz[i][k] - Mxyz[j][k]);
        delta = sqrt( delta );

        M_Sing_ToF[i] = eval_tof( &Mxyz[i][0],&Mxyz[j][0] );
    M_Sing_Vel[i] = delta/M_Sing_ToF[i];
}

}
/*==================================================================================*/
void establish_approx( double guess[DIMS] )
{

/* Algorithm which works in conjunction with tetra_solve to calculate an
/* initial "guess" at the system unknowns.  It utilizes a closed form solution
/* of tetrahedrons with three vertices at known locations (fixed transceivers)
/* and solves for the fourth vertex position (mobile transceiver).  Although it
/* serves its purpose for this program, this algorithm must be improved if it
/* is to be used in the applied global positioning system.  (An alternative for
/* the applied global positioning system would be to use the last indicated
/* platform position and a rough approximation of the change in position to
/* obtain the initial guess). */

int
i,j,
sgn=1;

double
m_guess[N_MBL][3],
p_guess[3],
d[3];

for (i=0; i<3; i++) {
        d[0] = Intrp_Vel[0][i]*ToF[0][i];
        d[1] = Intrp_Vel[1][i]*ToF[1][i];
        d[2] = Intrp_Vel[2][i]*ToF[2][i];
        tetra_solve( &m_guess[i][0],Fxyz,d[0],d[1],d[2],0 );

        if ( fabs(m_guess[i][0]+m_guess[i][1]+m_guess[i][2])<EPS )
                for (j=0; j<3; j++) m_guess[i][j] = Mxyz[i][j];
}

for (i=0; i<3; i++) {
        d[i] = 0.0;
        for (j=0; j<3; j++) d[i] += Mqrs[i][j]*Mqrs[i][j];
        d[i] = sqrt( d[i] );
}

tetra_solve( &p_guess[0],m_guess,d[0],d[1],d[2],sgn );

guess[0] = p_guess[0];
guess[1] = p_guess[1];
guess[2] = p_guess[2];

if (Solve_Angles[0]) guess[3] = Tht_Phi_Mu[0];
if (Solve_Angles[1]) guess[4] = Tht_Phi_Mu[1];
if (Solve_Angles[2]) guess[5] = Tht_Phi_Mu[2];

}
/*==================================================================================*/
void tetra_solve( double *ans,double base[3][3],double r0,double r1,
                double r2,int side )
{

/* Tetrahedron vertex solution.  Given the positions of three vertices and
/* the lengths of the sides from each of those vertices to the fourth vertex,
```

```
/* the position of the forth vertex is determined. */

double
x0 = base[0][0],
x1 = base[1][0],
x2 = base[2][0],
y0 = base[0][1],
y1 = base[1][1],
y2 = base[2][1],
z0 = base[0][2],
z1 = base[1][2],
z2 = base[2][2],
a01,a02,
b01,b02,
c01,c02,
d01,d02,
e0,e1,
f0,f1,
a,b,c,d,
sgn;

a01 = 2.0*(x0 - x1);
a02 = 2.0*(x0 - x2);
b01 = 2.0*(y0 - y1);
b02 = 2.0*(y0 - y2);
c01 = 2.0*(z0 - z1);
c02 = 2.0*(z0 - z2);
d01 = x0*x0 - x1*x1 + y0*y0 - y1*y1 + z0*z0 - z1*z1 - r0*r0 + r1*r1;
d02 = x0*x0 - x2*x2 + y0*y0 - y2*y2 + z0*z0 - z2*z2 - r0*r0 + r2*r2;
e0 = (d01*b02 - d02*b01)/(a01*b02 - a02*b01);
f0 = -(c01*b02 - c02*b01)/(a01*b02 - a02*b01);
e1 = (d01*a02 - d02*a01)/(b01*a02 - b02*a01);
f1 = -(c01*a02 - c02*a01)/(b01*a02 - b02*a01);

a = f0*f0 + f1*f1 + 1.0;
b = 2.0*(e0*f0 - x0*f0 + e1*f1 - y0*f1 - z0);
c = x0*x0 - 2.0*e0*x0 + e0*e0 + y0*y0 - 2.0*y0*e1 + e1*e1 + z0*z0 - r0*r0;
d = b*b - 4.0*a*c;

if (side) sgn = 1.0;
else sgn = -1.0;

if (d >= 0.0) {
        *(ans+2) = (-b + sgn*sqrt(d))/(2.0*a);
        *(ans) = e0 + f0*(*(ans+2));
        *(ans+1) = e1 + f1*(*(ans+2));
}
else {
        /* Problem with solution. */
        *(ans) = (x0 + x1 + x2)/3.0;
        *(ans+1) = (y0 + y1 + y2)/3.0;
        *(ans+2) = (z0 + z1 + z2)/3.0;
}

}
/*==========================================================================*/
void store_true( double t_fxyz[N_FXD][3],double t_tof[N_FXD][N_MBL],
                double t_pivot[3],double t_mqrs[N_MBL][3],double t_angs[3] )
{

/* Storing true values so system can be reset after error incorporation. */

int i,j;

for (i=0; i<N_FXD; i++)
        for (j=0; j<3; j++)
                t_fxyz[i][j] = Fxyz[i][j];
for (i=0; i<N_FXD; i++)
```

```
        for (j=0; j<N_MBL;j++)
                t_tof[i][j] = ToF[i][j];
for (i=0; i<N_MBL; i++)
        for (j=0; j<3; j++)
                t_mqrs[i][j] = Mqrs[i][j];
t_pivot[0] = Pxyz[0];
t_pivot[1] = Pxyz[1];
t_pivot[2] = Pxyz[2];
t_angs[0] = Tht_Phi_Mu[0];
t_angs[1] = Tht_Phi_Mu[1];
t_angs[2] = Tht_Phi_Mu[2];

}
/*==================================================================================*/
void incorporate_errors( double err_mtx[2*N_FXD+N_MBL+1][N_MBL] )
{

double
f_err, /* Maximum xyz dimensional errors for fixed xcvr positioning. */
offset_err, /* Offset error for time of flight data. */
t_r_err, /* Maximum pure random error in time of flight data. */
t_d_err, /* Maximum tof random error related to depth of xcvr. */
m_err, /* Maximum qrs dimensional errors for mobile xcvr positioning. */
phi_err, /* Maximum inclinometer angular error for phi measurement. */
mu_err, /* Maximum inclinometer angular error for mu measurement. */
random_err, /* Resulting tof random error. */
depth_factor, /* Depth factor for calculating depth related error. */
depth_related_err; /* Resulting depth related tof error. */

int
i,j;


/* Adding error to flange tranceiver locations. */
f_err = 1.0; /* In +/- millimeters. */
for (i=0; i<N_FXD; i++) {
        for (j=0; j<3; j++) {
                err_mtx[i][j] = (double)(f_err*(random(2000) - 1000)/1.e6);
                Fxyz[i][j] += err_mtx[i][j];
        }
}

/* Adding error to time array. */
offset_err = 0.5; /* In microseconds. */
t_r_err = 1.0; /* In +/- microseconds. */
t_d_err = 0.10; /* In +/- microseconds per meter. */
for (i=0; i<N_FXD; i++) {
        for (j=0; j<N_MBL; j++) {
                random_err = (double)(t_r_err*(random(2000) - 1000)/1.e9);
                depth_factor = (int)(Mxyz[j][2]*1000);
                depth_related_err = (double)(t_d_err*((random(2*depth_factor)
                        - depth_factor)/1.e9));
                err_mtx[i+N_FXD][j] = 1e-6*offset_err + random_err
                        + depth_related_err;
                ToF[i][j] += err_mtx[i+N_FXD][j];
        }
}

/* Adding error to assumed distances between M transceivers. */
m_err = 1.0; /* In +/- millimeters. */
for (i=0; i<N_MBL; i++) {
        for (j=0; j<3; j++) {
                err_mtx[i+2*N_FXD][j] = (double)(m_err*(random(2000) - 1000)/1.e6);
                Mqrs[i][j] += err_mtx[i+2*N_FXD][j];
        }
}

/* Adding error to inclinometer angles Phi and Mu. */
```

```
/*phi_err = 0.10; /* In +/- degrees. */
/*err_mtx[N_MBL+2*N_FXD][0] = (double)(phi_err*(random(2000)-1000)*PI/18.e4);
/*Tht_Phi_Mu[1] += err_mtx[N_MBL+2*N_FXD][0];
/*mu_err = 0.10; /* In +/- degrees. */
/*err_mtx[N_MBL+2*N_FXD][1] = (double)(mu_err*(random(2000)-1000)*PI/18.e4);
/*Tht_Phi_Mu[2] += err_mtx[N_MBL+2*N_FXD][1]; */

}
/*==================================================================================*/
void reset_true( double t_fxyz[N_FXD][3],double t_tof[N_FXD][N_MBL],
                 double t_pivot[3],double t_mqrs[N_MBL][3],double t_angs[3] )
{

/* Resets true system values after error incorporation. */

int i,j;

for (i=0; i<N_FXD; i++)
        for (j=0; j<3; j++)
                Fxyz[i][j] = t_fxyz[i][j];
for (i=0; i<N_FXD; i++)
        for (j=0; j<N_MBL;j++)
                ToF[i][j] = t_tof[i][j];
for (i=0; i<N_MBL; i++)
        for (j=0; j<3; j++)
                Mqrs[i][j] = t_mqrs[i][j];
Pxyz[0] = t_pivot[0];
Pxyz[1] = t_pivot[1];
Pxyz[2] = t_pivot[2];
Tht_Phi_Mu[0] = t_angs[0];
Tht_Phi_Mu[1] = t_angs[1];
Tht_Phi_Mu[2] = t_angs[2];

}
/*==================================================================================*/
double convert_angle( double ang )
{

/* Converts angle such that -PI < ang < PI. */

double revs;

revs = int( ang/(2.0*PI) );
ang -= revs*2.0*PI;
if (ang > PI) ang -= 2.0*PI;
else if (ang < -PI) ang += 2.0*PI;

return( ang );

}
/*==================================================================================*/
void evaluate_rtn_mtx( double angs[3] )
{

/*      Euler rotations.
/*      1. Rotation about z by tht.
/*      2. Rotation about new x by phi.
/*      3. Rotation about new y by mu. */

double st,sp,sm,ct,cp,cm;

st = sin(angs[0]);      ct = cos(angs[0]);
sp = sin(angs[1]);      cp = cos(angs[1]);
sm = sin(angs[2]);      cm = cos(angs[2]);

Rotation[0][0] = ct*cm-st*sp*sm;
Rotation[0][1] = -st*cp;
Rotation[0][2] = ct*sm+st*sp*cm;
```

```
Rotation[1][0] = st*cm+ct*sp*sm;
Rotation[1][1] = ct*cp;
Rotation[1][2] = st*sm-ct*sp*cm;
Rotation[2][0] = -cp*sm;
Rotation[2][1] = sp;
Rotation[2][2] = cp*cm;

}
/*==========================================================================*/
double objective_function( double xvec[] )
{

/* The objective function is simply the vector magnitude of the values of the
/* system equations. */

double
fvec[SYS_EQS];

functions_evaluation( fvec,xvec );
return( vector_magnitude( fvec,Sys_Eqs ) );

}
/*==========================================================================*/
void functions_evaluation( double fvec[],double xvec[] )
{

/* System equations. */

double
mxyz_eval[N_MBL][3],
delta,
vel,
rot_angs[3];

int
i,j,k,
cmpnt = 0;

j = 0;
for (i=0; i<3; i++) {
        if (Solve_Angles[i]) {
                rot_angs[i] = xvec[3+j];
                j++;
        }
        else rot_angs[i] = Tht_Phi_Mu[i];
}

evaluate_rtn_mtx( rot_angs );

/* Evaluating the positions of the mobile transceivers based on the current
/* determined theta and position of the pivot.  These evaluated M positions
/* at this point in the search are stored in the 2D array mxyz_eval. */
for (i=0; i<N_MBL; i++) {
        for (j=0; j<3; j++) {
                mxyz_eval[i][j]=0.0;
                for (k=0; k<3; k++) mxyz_eval[i][j] += Rotation[j][k]*Mqrs[i][k];
                mxyz_eval[i][j] += xvec[j];
        }
}

/* Evaluating the system equations in the vector "fvec." */
for (i=0; i<N_FXD; i++) if (Active_F[i]) {
        for (j=0; j<N_MBL; j++ ) {
                delta = 0.0;
                for (k=0; k<3; k++ ) {
                        delta += (Fxyz[i][k]-mxyz_eval[j][k])*
                                        (Fxyz[i][k]-mxyz_eval[j][k]);
                }
```

```
                vel = Intrp_Vel[i][j];
                fvec[cmpnt] = delta - (vel*ToF[i][j])*(vel*ToF[i][j]);
                cmpnt++;
        }
}

}
/*==================================================================================*/
void jacobian_evaluation( double jcbn[][DIMS],double xvec[] )
{

/* Jacobian of linearized system. */

double
st,ct,sp,cp,sm,cm,
temp,
dR_dtht[3][3], /* Partial derivatives of Rotation matrix wrt Theta. */
dR_dphi[3][3], /* Partial derivatives of Rotation matrix wrt Phi. */
dR_dmu[3][3], /* Partial derivatives of Rotation matrix wrt Mu. */
rot_angs[3],
mxyz_eval[N_MBL][3],
dM_dtht[N_MBL][3], /* Partial derivatives of M positions wrt Theta. */
dM_dphi[N_MBL][3], /* Partial derivatives of M positions wrt Phi. */
dM_dmu[N_MBL][3]; /* Partial derivatives of M positions wrt Mu. */

int
i,j,k,kk,
indx,
ang_pos[3],
row = 0;

j = 0;
for (i=0; i<3; i++) {
        if (Solve_Angles[i]) {
                rot_angs[i] = xvec[3+j];
                j++;
        }
        else rot_angs[i] = Tht_Phi_Mu[i];
}

evaluate_rtn_mtx( rot_angs );

/* Evaluating postions of mobile transceivers. */
for (i=0; i<N_MBL; i++) {
        for (j=0; j<3; j++) {
                mxyz_eval[i][j]=0.0;
                for (k=0; k<3; k++) mxyz_eval[i][j] += Rotation[j][k]*Mqrs[i][k];
                mxyz_eval[i][j] += xvec[j];
        }
}

st=sin(rot_angs[0]);   ct=cos(rot_angs[0]);
sp=sin(rot_angs[1]);   cp=cos(rot_angs[1]);
sm=sin(rot_angs[2]);   cm=cos(rot_angs[2]);

if (Solve_Angles[0]) {
        dR_dtht[0][0] = -ct*sp*sm-st*cm;
        dR_dtht[0][1] = -ct*cp;
        dR_dtht[0][2] = ct*sp*cm-st*sm;
        dR_dtht[1][0] = -st*sp*sm+ct*cm;
        dR_dtht[1][1] = -st*cp;
        dR_dtht[1][2] = st*sp*cm+ct*sm;
        dR_dtht[2][0] = 0.0;
        dR_dtht[2][1] = 0.0;
        dR_dtht[2][2] = 0.0;
        for (i=0; i<N_MBL; i++) {
                for (j=0; j<3; j++) {
                        dM_dtht[i][j] = 0.0;
```

```
                        for (k=0; k<3; k++)
                                dM_dtht[i][j] += dR_dtht[j][k]*Mqrs[i][k];
                }
        }
}
if (Solve_Angles[1]) {
        dR_dphi[0][0] = -st*cp*sm;
        dR_dphi[0][1] = st*sp;
        dR_dphi[0][2] = st*cp*cm;
        dR_dphi[1][0] = ct*cp*sm;
        dR_dphi[1][1] = -ct*sp;
        dR_dphi[1][2] = -ct*cp*cm;
        dR_dphi[2][0] = sp*sm;
        dR_dphi[2][1] = cp;
        dR_dphi[2][2] = -sp*cm;
        for (i=0; i<N_MBL; i++) {
                for (j=0; j<3; j++) {
                        dM_dphi[i][j] = 0.0;
                        for (k=0; k<3; k++)
                                dM_dphi[i][j] += dR_dphi[j][k]*Mqrs[i][k];
                }
        }
}
if (Solve_Angles[2]) {
        dR_dmu[0][0] = -st*sp*cm-ct*sm;
        dR_dmu[0][1] = 0.0;
        dR_dmu[0][2] = -st*sp*sm+ct*cm;
        dR_dmu[1][0] = ct*sp*cm-st*sm;
        dR_dmu[1][1] = 0.0;
        dR_dmu[1][2] = ct*sp*sm+st*cm;
        dR_dmu[2][0] = -cp*cm;
        dR_dmu[2][1] = 0.0;
        dR_dmu[2][2] = -cp*sm;
        for (i=0; i<N_MBL; i++) {
                for (j=0; j<3; j++) {
                        dM_dmu[i][j] = 0.0;
                        for (k=0; k<3; k++)
                                dM_dmu[i][j] += dR_dmu[j][k]*Mqrs[i][k];
                }
        }
}

/* Evaluating jacobian elements. */
for (i=0; i<N_FXD; i++) if (Active_F[i]) {
        for (j=0; j<N_MBL; j++) {
        kk=0;
         for (k=0; k<3; k++)
                        if (Solve_Angles[k]) { jcbn[row][3+kk] = 0.0; ang_pos[k] = kk;
kk++; }
                for (k=0; k<3; k++) {
                        jcbn[row][k] = -2.0*(Fxyz[i][k]-mxyz_eval[j][k]);
                        if (Solve_Angles[0])
                                jcbn[row][3+ang_pos[0]] +=
                                        -2.0*(Fxyz[i][k]-mxyz_eval[j][k])*dM_dtht[j][k];
                        if (Solve_Angles[1])
                                jcbn[row][3+ang_pos[1]] +=
                                        -2.0*(Fxyz[i][k]-mxyz_eval[j][k])*dM_dphi[j][k];
                        if (Solve_Angles[2])
                                jcbn[row][3+ang_pos[2]] +=
                                        -2.0*(Fxyz[i][k]-mxyz_eval[j][k])*dM_dmu[j][k];
                }
                row++;
        }
}

}
/*==================================================================================*/
void solve_system( double org_pnt[DIMS],double sln_pnt[DIMS] )
```

```
{
/* This function relies on the evaluation of the system jacobian to find a
/* solution, but if this should fail it falls back on a dimensional pattern
/* search.  The solution is the set of dimensional variables which best
/* satisfy the system equations.  Additionally, the alogorith may also utilize
/* a univariate minimization routine.  Both the pattern search and univariate
/* search may be removed and this algorithm appropriately altered.  They are
/* included to ensure a minimum is determined.
/* The initial point (dimensional values) is org_pt, and the solution is
/* returned in sln_pt. */

double
current_point[DIMS],
jacobian[SYS_EQS][DIMS],
new_obj_fnc_value,
last_obj_fnc_value,
search_vector[DIMS],
f_vector[SYS_EQS],
test_point[DIMS],
difference,
step;

int
i,j,k,
maxiter=20,
count=0,
same_count=0,
ok,
stop = 0;

for (i=0; i<DIMS; i++) current_point[i] = org_pnt[i];
last_obj_fnc_value = objective_function( current_point );
do      {
        jacobian_evaluation( jacobian,current_point );
        functions_evaluation( f_vector,current_point );
        for (i=0; i<Sys_Eqs; i++) f_vector[i] = -f_vector[i];
        ok = least_squares_solution( &jacobian[0][0],f_vector );
        if (ok) {
                for (i=0; i<Dims; i++) search_vector[i] = f_vector[i];
                        move( test_point,current_point,search_vector,1.0 );
                new_obj_fnc_value = objective_function( test_point );
                if (new_obj_fnc_value<last_obj_fnc_value) {
                        for (i=0; i<Dims; i++) current_point[i] = test_point[i];
                        difference = fabs(new_obj_fnc_value - last_obj_fnc_value);
                        if (difference<EPS) ++same_count;
                        last_obj_fnc_value = new_obj_fnc_value;
                }
                else if (new_obj_fnc_value>last_obj_fnc_value) {
                        step = vector_magnitude( search_vector,Dims );
                        if (step<EPS) step = 0.0001;
                        normalize_vector( search_vector,Dims );
                        step = step_evaluation( current_point,search_vector,step );
                        move( test_point,current_point,search_vector,step );
                        new_obj_fnc_value = objective_function( test_point );
                        if (new_obj_fnc_value<last_obj_fnc_value) {
                                for (i=0; i<Dims; i++) current_point[i] = test_point[i];
                                difference = fabs(new_obj_fnc_value - last_obj_fnc_value);
                                if (difference<EPS) ++same_count;
                                last_obj_fnc_value = new_obj_fnc_value;
                        }
                        else if (new_obj_fnc_value==last_obj_fnc_value) same_count++;
                        else stop = 1;
                }
                else same_count++;
        }
        else {
                printf( "\nMatrix Solve Failure : Pattern Search Implemented \n" );
                new_obj_fnc_value = pattern_search( current_point );
```

```
                difference = fabs(new_obj_fnc_value - last_obj_fnc_value);
                if (difference<EPS) ++same_count;
                if (new_obj_fnc_value<last_obj_fnc_value)
                        last_obj_fnc_value = new_obj_fnc_value;
        }
        ++count;
        if (same_count>=2) stop = 1;
        else if (new_obj_fnc_value<0.0001) stop = 1;
        else if (count>maxiter) stop = 1;
        if (stop) {
                new_obj_fnc_value = pattern_search( current_point );
                if ((new_obj_fnc_value-last_obj_fnc_value)<(-EPS)) stop = 0;
        }
} while ( !stop );

for (i=0; i<Dims; i++) sln_pnt[i] = current_point[i];

}
/*==================================================================================*/
double step_evaluation( double *xvec,double *svec,double step )
{

/* Univariate minimization routine which utilizes a quadradic minimum.
/* This function searches in the +/- direction of the dimensional vector
/* svec for the minimum functional value.  The function returns the signed
/* distance from the initial position vector xvec to this minimum.
/* Based upon the DSC-Powell unidimensional search process outlined in
/* "Applied Nonlinear Programming", David M. Himmelblau, 1972, pp. 42-50.*/

double
f,fa,fb,fc,da,db,dc,
alpha,alpha_old,
new_xvec[DIMS],
temp[DIMS],
branch;

int
i,k,
got_alpha,
iter,
itermax;

f = objective_function( xvec );
fa=fb=fc=f;
da=db=dc=0.0;
alpha_old=0.0;
got_alpha=0;
k=-2;
iter=0;
itermax=20;

do      {
        move( new_xvec,xvec,svec,step );
        f = objective_function( new_xvec );
        k+=1;
        if ( f<fa ) {
                dc=db; db=da; da=step;
                fc=fb; fb=fa; fa=f;
                step*=2.0;
        }
        else if ( f>fa ) {
                if ( k<0 )      { fb=f; db=step; step=-step; }
                else break;
        }
        else if ( f==fa ) {
                alpha=0.5*(da+step);
                got_alpha=1;
        }
```

```
}       while ( !got_alpha );
if ( !got_alpha ) {
        fc=fb; fb=fa; fa=f;
        dc=db; db=da; da=step;
}
while ( !got_alpha ) {
        alpha = 0.5*((db*db-dc*dc)*fa+(dc*dc-da*da)*fb+(da*da-db*db)*fc);
        alpha /= (fa*(db-dc)+fb*(dc-da)+fc*(da-db));
        iter+=1;
        step=alpha;
        if ( fabs(fabs(step)-alpha_old)<EPS ) { got_alpha=1; break; }
        else if ( fabs(step)<EPS ) { got_alpha=1; break; }
        else if ( iter>itermax ) { got_alpha=1; break; }
        alpha_old=fabs(alpha);
        move(new_xvec,xvec,svec,step);
        f = objective_function( new_xvec );
        branch=(dc-step)*(step-db);
        if ( branch>0 ) {
                if ( f>fb )                     { fc=f;dc=step; }
                else if ( f<fb )        { da=db;db=step;fa=fb;fb=f; }
                else                            { alpha=0.5*(step+db);break; }
        }
        else if ( branch<0 ) {
                if ( f>fb )                     { fa=f;da=step; }
                else if ( f<fb )        { dc=db;db=step;fc=fb;fb=f; }
                else                            { alpha=0.5*(step+db);break; }
        }
        else break;
}

return( alpha );

}
/*==================================================================*/
double pattern_search( double point[DIMS] )
{

/* Searches in each dimension for minimum functional value.  It returns the
/* resulting objective function value and the minimum point. */

double
temp,
objfnc, /* Objective function value. */
test_point[DIMS],
search[DIMS], /* Search vector. */
step[DIMS]; /* Initial steps. */
int i,j;

step[0] = step[1] = step[2] = 0.01;
for (i=3; i<Dims; i++) step[i] = 1.0;
j=0;
for (i=0; i<3; i++)
        if (Solve_Angles[i]) { step[3+j] = 0.05; j++; }
objfnc = objective_function( point );
search[0] = 1.0;
for (i=1; i<Dims; i++) search[i] = 0.0;
for (i=0; i<Dims; i++) {
        step[i] = step_evaluation( point,search,step[i] );
        move( test_point,point,search,step[i] );
        temp = objective_function( test_point );
        if (temp<objfnc) {
                objfnc = temp;
                for (i=0; i<Dims; i++) point[i] = test_point[i];
        }
        if (i!=(Dims-1)) {     search[i] = 0.0;search[i+1] = 1.0; }
}

return( objfnc );
```

```
}
/*==========================================================================*/
void move( double result[],double current[],double search[],double step )
{

int i;

for (i=0; i<Dims; i++) result[i] = current[i] + search[i]*step;

}
/*==========================================================================*/
double vector_magnitude( double vec[],int number_of_components )
{

double mag=0.0;
int i;

for (i=0; i<number_of_components; i++) mag += vec[i]*vec[i];
mag = sqrt(mag);

return( mag );

}
/*==========================================================================*/
void normalize_vector( double vec[],int number_of_components )
{

double mag = vector_magnitude(vec,number_of_components);
int i;

for (i=0; i<number_of_components; i++) vec[i] /= mag;

}
/*==========================================================================*/
int least_squares_solution( double *overdetermined_a,double *sol_vector_b )
{

/* Determines the least square solution (residual squares) of the equation
/* Ax=B where A is an p by q matrix (p = # of system equations defined as
/* SYS_EQS and q = # of unknowns or dimensions of system defined as DIMS),
/* and B is a vector of m components.  It creates a normal q by q A matrix
/* called normal_a_matrix and a normal B vector of q components called
/* normal_b_vector and then solves this system.  The resulting x vector of
/* dimensional solutions is stored (and returned) at the address of the
/* pointer sol_vector_b.  Thus this variable serves as the input B and the
/* output solution x.  Note that q of the original values of sol_vector_b are
/* replaced by the x vector solution and thus if they need to be retained the
/* values should be copied before sending its address to this function.
/* Note also that this function returns a flag.  A returned value of 1 is an
/* indication that all is well.  A returned value of 0 is an indication that
/* there may have been a problem, i.e. matrix singularity. */

double normal_a_matrix[DIMS][DIMS],normal_b_vector[DIMS];
int i,j,ok;

for (i=0; i<Dims; i++) {
        for (j=i; j<Dims; j++) {
                if (i!=j)
                        normal_a_matrix[i][j] = normal_a_matrix[j][i] =
                        dot_vector( overdetermined_a+i,overdetermined_a+j,
                                DIMS,DIMS,Sys_Eqs );
                else
                        normal_a_matrix[i][j] =
                        dot_vector( overdetermined_a+i,overdetermined_a+j,
                                DIMS,DIMS,Sys_Eqs );
        }
        normal_b_vector[i] = dot_vector( overdetermined_a+i,sol_vector_b,
```

```
                DIMS,1,Sys_Eqs );
}
ok = matrix_solve( &normal_a_matrix[0][0],&normal_b_vector[0] );
for (i=0; i<Dims; i++) *(sol_vector_b+i) = normal_b_vector[i];

return( ok );

}
/*=================================================================================*/
double dot_vector( double *a,double *b,int num_of_a_clmns,int num_of_b_clmns,
        int length )
{

/* This function takes in the memory address of the first component of each
/* vector a and b, and calculates the dot product of the vectors.  Note that
/* the two vectors may by columns of a two-dimensional array.  This is why
/* the function requires the number of columns in each matrix. */

double dot_product = 0.0;
int i;

for (i=0; i<length; i++)
        dot_product += (*(a+num_of_a_clmns*i))*(*(b+num_of_b_clmns*i));

return( dot_product );

}
/*=================================================================================*/
int matrix_solve( double *a,double *b )
{

/* This function simply joins lu_dcmp and lu_bksb to solve Ax=B where A is a
/* DIMS by DIMS matrix and B is a vector of DIMS components.  It returns a
/* flag value of 0 is there was a failure in the solution process.
/* matrix_solve, lu_dcmp, lu_bksb based upon the LU Decomposition described in
/* "Numerical Recipes in C", Willeam H. Press et. al., 1988, pp. 39-45. */

int     indx[DIMS],flag;

flag = lu_dcmp( a,&indx[0]);
lu_bksb(a,&indx[0],b);

return( flag );

}
/*=================================================================================*/
int lu_dcmp( double *a,int *indx )
{

/* Performs the LU decomposition of a DIMS by DIMS matrix.  It uses a
/* pivoting scheme to increase its robustness and its pivoting history is
/* stored in "indx".  It is designed to be used with lu_bksb.  Note that "a"
/* is replaced by the LU scheme, thus if it is needed elsewhere it should be
/* copied before sending its address to this function.  It also returns a
/* flag for which a 0 value indicates that a singular value matrix was sent
/* to it and it could not be decomposed. */

double  big,sum,temp1,temp2,rowscale[DIMS];
int     i,j,k,imax,flag=1;

for (i=0; i<Dims; i++) {
        big=0.0;
        for (j=0; j<Dims; j++)
                if ((temp1 = fabs(*(a+DIMS*i+j))) > big) big=temp1;
        if (big==0.0) flag = 0;
        else rowscale[i] = 1.0/big;
}
if (flag) {
```

```
        for (j=0; j<Dims; j++) {
                for (i=0; i<j; i++) {
                        sum = *(a+DIMS*i+j);
                        for (k=0; k<i; k++) sum -= (*(a+DIMS*i+k))*(*(a+DIMS*k+j));
                        *(a+DIMS*i+j) = sum;
                }
                big = 0.0;
                for (i=j; i<Dims; i++) {
                        sum = *(a+DIMS*i+j);
                        for (k=0; k<j; k++) sum -= (*(a+DIMS*i+k))*(*(a+DIMS*k+j));
                        *(a+DIMS*i+j) = sum;
                        if ((temp2 = rowscale[i]*fabs(sum)) >= big) {
                                big = temp2;
                                imax = i;
                        }
                }
                if (j != imax) {
                        for (k=0; k<Dims; k++) {
                                temp2 = *(a+DIMS*imax+k);
                                *(a+DIMS*imax+k) = *(a+DIMS*j+k);
                                *(a+DIMS*j+k) = temp2;
                        }
                        rowscale[imax] = rowscale[j];
                }
                *(indx+j) = imax;
                if (*(a+DIMS*j+j) == 0.0) *(a+DIMS*j+j) = TINY;
                if(j != (Dims-1)) {
                        temp2 = 1.0/(*(a+DIMS*j+j));
                        for (i=j+1; i<Dims; i++) *(a+DIMS*i+j) *= temp2;
                }
        }
}

return( flag );

}
/*==========================================================================*/
void lu_bksb( double *a,int *indx,double *b )
{

/* This function solves the system ax=b (where a has been previously LU
/* decomposed), and returns the x vector solution in "b".  It is to be used
/* in conjunction with lu_dcmp. */

double sum;
int     i,j,ii=-1,ip;

for (i=0; i<Dims; i++) {
        ip = *(indx+i);
        sum = *(b+ip);
        (*(b+ip))=(*(b+i));
        if (ii+1)
                for (j=ii; j<=i-1; j++) sum -= (*(a+DIMS*i+j))*(*(b+j));
        else if (sum) ii=i;
        (*(b+i))=sum;
}
for (i=Dims-1; i>=0; i--) {
        sum = (*(b+i));
        for (j=i+1; j<Dims; j++) sum -= (*(a+DIMS*i+j))*(*(b+j));
        (*(b+i)) = sum/(*(a+DIMS*i+i));
}

}
/*==========================================================================*/
/*=============================  End of Program  ============================*/
/*==========================================================================*/
```

Appendix 4b
GPS Code; Equivalent Velocity System

This code is designed to examine the effectiveness of the global positioning system which utilizes the equivalent velocity approximation. It is configured such that any of the system parameters may be altered without requiring significant changes to the code. This program uses most of the code of Appendix 4a (as indicated by the function prototypes), and so only those sections which are different (or are not a part of the interpolated velocity solution process) are included here.

```
/*=================================================================*/
/* GPSEQVEL.CPP                                                    */
/* Program designed to evaluate the use of acoustic transceivers in the  */
/* location of an underwater mobile platform.  It has been designed to allow */
/* many different schemes to be examined without significant changes to the  */
/* program code.                                                   */
/* Program utilizes the equivalent velocity approximation to reduce system  */
/* unknowns.                                                       */
/* Programmer: Allen Maples                                        */
/* Program code: C                                                 */
/* Development completed: 4/93                                     */
/*=================================================================*/

#include <math.h>
#include <dos.h>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>

/* Defined Constants */
const int N_FXD = 3; /* Total number of fixed xcvrs. */
const int N_MBL = 3; /* Total number of mobile xcvrs. */
const int DIMS = 9; /* Max number of system dimensions. */
const int SYS_EQS = N_FXD*N_MBL; /* Max number of system equations. */
const double EPS = 0.00001; /* Tolerance value. */
const double TINY = 1e-20; /* Small value. */
const long double PI = (4.01*atan(1.01));
const double RAD_TO_DEG = 180.0/PI; /* Conversion factor. */

/* Functions */
void    define_known();
void    aquire_inclns( int,int );
void    aquire_ToF();
double  point_vel( double[] );
void    evaluate_rtn_mtx( double[] );
double  eval_tof( double[],double[] );
void    aquire_F_Sing();
void    aquire_M_Sing();
void    determine_nodal_vels();
void    determine_interpolated_vels();
void    determine_Sigmas();
void    adjust_active();
void    establish_approx( double[] );
void    tetra_solve( double*,double[][3],double,double,double,int );
void    store_true( double[][3],double[][N_MBL],double[],double[][3],double[] );
void    incorporate_errors( double[][N_MBL] );
```

```
void    solve_system( double[],double[] );
void    reset_true( double[][3],double[][N_MBL],double[],double[][3],double[] );
double  objective_function( double[] );
void    jacobian_evaluation( double[][DIMS],double[] );
void    functions_evaluation( double[],double[] );
int     least_squares_solution( double*,double* );
double  dot_vector( double*,double*,int,int,int );
double  step_evaluation( double[],double[],double );
double  pattern_search( double[] );
int     matrix_solve( double*, double* );
int     lu_dcmp( double*,int* );
void    lu_bksb( double*,int*,double*);
double  convert_angle( double );
void    move( double[],double[],double[],double );
double  vector_magnitude( double[],int );
void    normalize_vector( double[],int );

/* Global Variables */
double
Fxyz[N_FXD][3], /* Fixed transceiver global locations. */
Mxyz[N_MBL][3], /* Mobile transceiver global locations. */
Mqrs[N_MBL][3], /* Locations of mobile transceivers relative to Pivot. */
Pxyz[3], /* Global location of Pivot. */
Model_Vel[N_FXD][N_MBL], /* F <--> M model transmission velocities. */
ToF[N_FXD][N_MBL], /* F <--> M times of flight (sound). */
F_Sing_Vel[N_FXD], /* Sing-around velocities; {F0-F1,F1-F2,F2-F0}. */
M_Sing_Vel[N_MBL], /* Sing-around velocities; {M0-M1,M1-M2,M2-M0}. */
F_Sing_ToF[N_FXD], /* Sing-around tof; {F0-F1,F1-F2,F2-F0}. */
M_Sing_ToF[N_MBL], /* Sing-around tof; {M0-M1,M1-M2,M2-M0}. */
F_Node_Vel[N_FXD], /* Nodal velocities of fixed xcvrs. */
M_Node_Vel[N_MBL], /* Nodal velocities of mobile xcvrs. */
Intrp_Vel[N_FXD][N_MBL], /* Approx of transmission vels using nodal vels. */
Sigma[N_FXD][N_MBL], /* Equivalent Vel. Approx. compensation factors. */
Rotation[3][3],        /* Theta,Phi,Mu rotation matrix. */
Tht_Phi_Mu[3], /* Platform rotation angles. */
Comp_Fctr = 3.0, /* Compensation factor. */
Alpha = 0.0, /* Angular position of pivot in vessel. */

/* Vessel Model Characteristics */
Salinity = 0.0, /* Water salenity. */
Wall_Temp = 30.0, /* Parabolic/radial temperature distribution value. */
Center_Temp = 25.0, /* Parabolic/radial temperature distribution value. */
Top_Temp = 30.0, /* Linear/depth temperature distribution value. */
Bottom_Temp = 20.0, /* Linear/depth temperature distribution value. */
Default_Temp = 30.0, /* Default temperature of vessel water. */
Vessel_Depth = 10.0, /* Distance from F plane to Bottom_Temp location. */
Rbt_Rad = 1.5, /* Radius (m) that robot pivot resides at. */
Vsl_Rad = 2.286; /* Radius (m) of reactor vessel. */

int
Active_F[N_FXD] = { 1,1,1 }, /* Indicates which fixed xcvrs are active. */
Solve_Angles[3] = { 1,0,0 }, /* Which angles (theta,phi,mu) must be solved. */
N_Fxd, /* Number of fixed transceivers active in the solution process. */
Dims, /* Number of dimensions for the system being solved. */
Sys_Eqs, /* Number of system equations for the system being solved. */
Add_Error = 1, /* Indicates whether or not error is to be incorporated. */
Use_Comp = 1; /* Indicates whether or not vel compensation is to be used. */

/*==================================================================*/
/*==================================================================*/
void main()
{

double
guess[DIMS],
answer[DIMS],
true_fxyz[N_FXD][3],
true_tof[N_FXD][N_MBL],
```

```
true_pivot[3],
true_mqrs[N_MBL][3],
true_angles[3],
err_mtx[2*N_FXD+N_MBL+1][N_MBL],
revs,
division = PI/18.0,
t_err,
p_err,
m_err,
tot_err,
x_err,
y_err,
z_err,
r_err,
a_err,
t_tot_err=0.0,
t_x_err=0.0,
t_t_err=0.0,
t_p_err=0.0,
t_m_err=0.0,
t_y_err=0.0,
t_r_err=0.0,
t_a_err=0.0,
t_z_err=0.0,
max_err[10] = {0,0,0,0,0,0,0,0,0,0},
std_dev,
std_dev_at_max,
temp,
radius,
depth;

int
i,
locations=19,
rndm_iters=5;

if (!Add_Error) rndm_iters=1;

clrscr();

define_known();

printf( "\nProgram: GPSEQVEL.CPP" );
printf( "\nGPS utilizing equivalent velocity approximation." );
printf( "\nSolving for Px,Py,Pz" );
if (Solve_Angles[0]) printf( ",Theta" );
if (Solve_Angles[1]) printf( ",Phi" );
if (Solve_Angles[2]) printf( ",Mu" );
printf( ".\nTemperatures: Tw = %5.2lf, Tc = %5.2lf, Tt = %5.2lf, Tb = %5.2lf.",
        Wall_Temp,Center_Temp,Top_Temp,Bottom_Temp );
printf( "\nCompensation factor = %4.2lf", Comp_Fctr );
printf( "\nPlatform pivot at radius of %4.2lf m and depth of %4.2lf m.",
        Rbt_Rad,-Pxyz[2] );
if (Add_Error)
        printf( "\nError incorporation over %d iterations.", rndm_iters );
else printf( "\nNo added error to system." );
printf( "\n\n\n" );
printf( "  Alpha | Total |   x   |   y   |   z   " );
printf( "| Theta|  Phi |  Mu  | Alpha| radial" );
printf( "\n" );
printf( "Position| Error | Error | Error | Error " );
printf( "| Error| Error| Error| error| error" );
printf( "\n" );
printf( "   (deg) |  (cm) |  (cm) |  (cm) |  (cm) " );
printf( "| (deg)| (deg)| (deg)| (deg)|  (cm)  \n" );
printf( "--------|-------|-------|-------|-------" );
printf( "|------|------|------|------|-------" );
```

```
Alpha = 0.0;

for (i=0; i<locations; i++) {

        define_known();
        aquire_inclns( 1,1 );
        aquire_ToF();
        aquire_M_Sing();
        aquire_F_Sing();
        determine_nodal_vels();
        determine_interpolated_vels();
        determine_Sigmas();
        adjust_active();
        establish_approx( guess );


        t_tot_err=0.0;
        t_x_err=0.0;
        t_t_err=0.0;
        t_p_err=0.0;
        t_m_err=0.0;
        t_y_err=0.0;
        t_r_err=0.0;
        t_a_err=0.0;
        t_z_err=0.0;

        randomize();

    std_dev = 0.0;

        for (int jj=0; jj<rndm_iters; jj++) {

        if (Add_Error) {
                store_true( true_fxyz,true_tof,true_pivot,true_mqrs,true_angles );
                incorporate_errors( err_mtx );
        }

        solve_system( guess,answer );

        if (Add_Error)
                reset_true( true_fxyz,true_tof,true_pivot,true_mqrs,true_angles );

        x_err = answer[0]-Pxyz[0];
        y_err = answer[1]-Pxyz[1];
        z_err = answer[2]-Pxyz[2];
        r_err = sqrt( answer[0]*answer[0] + answer[1]*answer[1]) -
                        sqrt(Pxyz[0]*Pxyz[0] + Pxyz[1]*Pxyz[1]);
        if (Solve_Angles[0])
                t_err = convert_angle( answer[3]-Tht_Phi_Mu[0] );
        else t_err = 0.0;
        if (Solve_Angles[1])
                p_err = convert_angle( answer[4]-Tht_Phi_Mu[1] );
        else p_err = 0.0;
        if (Solve_Angles[2])
                m_err = convert_angle( answer[5]-Tht_Phi_Mu[2] );
        else m_err = 0.0;
        a_err = convert_angle( atan2(answer[1],answer[0])-atan2(Pxyz[1],Pxyz[0]) );
        tot_err = sqrt(x_err*x_err + y_err*y_err + z_err*z_err);

        if (Add_Error) {
                t_tot_err += tot_err;
                t_t_err += fabs(t_err);
                t_p_err += fabs(p_err);
                t_m_err += fabs(m_err);
                t_x_err += fabs(x_err);
                t_y_err += fabs(y_err);
                t_r_err += fabs(r_err);
                t_a_err += fabs(a_err);
```

```
                t_z_err += fabs(z_err);

                std_dev += tot_err*tot_err;

                if (tot_err > max_err[1]) {
                        max_err[0] = Alpha;
                        max_err[1] = tot_err;
                        max_err[2] = x_err;
                        max_err[3] = y_err;
                        max_err[4] = z_err;
                        max_err[5] = t_err;
                        max_err[6] = p_err;
                        max_err[7] = m_err;
                        max_err[8] = a_err;
                        max_err[9] = r_err;
        }
    }
        else {
                t_tot_err += tot_err;
                t_t_err += t_err;
                t_p_err += p_err;
                t_m_err += m_err;
                t_x_err += x_err;
                t_y_err += y_err;
                t_r_err += r_err;
                t_a_err += a_err;
                t_z_err += z_err;
        }

        if (jj==(rndm_iters-1)) {

                if (Add_Error) {
                        std_dev = sqrt( (std_dev - t_tot_err*t_tot_err/
                                        double(rndm_iters))/double(jj) );
                        if (int(RAD_TO_DEG*Alpha)==int(RAD_TO_DEG*max_err[0]))
                                std_dev_at_max = std_dev;
                }

                t_err = t_t_err/rndm_iters;
                p_err = t_p_err/rndm_iters;
                m_err = t_m_err/rndm_iters;
                tot_err = t_tot_err/rndm_iters;
                x_err = t_x_err/rndm_iters;
                y_err = t_y_err/rndm_iters;
                a_err = t_a_err/rndm_iters;
                z_err = t_z_err/rndm_iters;
                r_err = t_r_err/rndm_iters;

                printf( "\n" );
                printf( " %6.2lf |%6.2lf |%6.2lf |%6.2lf |%6.2lf |",
                (Alpha)*RAD_TO_DEG,
                1.0e2*tot_err,
                1.0e2*x_err,
                1.0e2*y_err,
                1.0e2*z_err );
                printf( "%5.2lf |%5.2lf |%5.2lf |%5.2lf |%6.2lf",
                t_err*RAD_TO_DEG,
                p_err*RAD_TO_DEG,
                m_err*RAD_TO_DEG,
                a_err*RAD_TO_DEG,
                1.0e2*r_err );
        }
}

Alpha += division;

}
```

```
if (Add_Error) {
        printf( "\n\nMaximum error:\n" );
        printf( " %6.2lf |%6.2lf |%6.2lf |%6.2lf |%6.2lf ",
        max_err[0]*RAD_TO_DEG,
        1.0e2*max_err[1],
        1.0e2*max_err[2],
        1.0e2*max_err[3],
        1.0e2*max_err[4] );
        printf( "|%5.2lf |%5.2lf |%5.2lf |%5.2lf |%6.2lf",
        max_err[5]*RAD_TO_DEG,
        max_err[6]*RAD_TO_DEG,
        max_err[7]*RAD_TO_DEG,
        max_err[8]*RAD_TO_DEG,
        1.0e2*max_err[9] );
        printf( "\nStandard deviation of total error at alpha = " );
        printf( "%4.1lf degrees was %5.2lf cm.",
                max_err[0]*RAD_TO_DEG,std_dev_at_max*1e2 );
}

}
/*==========================================================================*/
/*==========================================================================*/
void adjust_active()
{

/* Adjusts global system parameters N_Fxd,Dims,and Sys_Eqs to correspond to
/* the current situation.  */

int i,xcvr = 0;

for (i=0; i<N_FXD; i++) if (Active_F[i]) xcvr++;

N_Fxd = xcvr;
Dims = 3+Solve_Angles[0]+Solve_Angles[1]+Solve_Angles[2]+N_Fxd;
Sys_Eqs = N_Fxd*N_MBL;

}
/*==========================================================================*/
void define_known()
{

double mbl_rad = 0.5; /* Determines separation of mobile transceivers. */

/* Fixed transceiver global positions. */
Fxyz[0][0] = Vsl_Rad;
Fxyz[0][1] = 0.0;
Fxyz[0][2] = 0.0;
Fxyz[1][0] = -Vsl_Rad*sin(PI/6.);
Fxyz[1][1] = Vsl_Rad*cos(PI/6.0);
Fxyz[1][2] = 0.0;
Fxyz[2][0] = -Vsl_Rad*sin(PI/6.);
Fxyz[2][1] = -Vsl_Rad*cos(PI/6.0);
Fxyz[2][2] = 0.0;

/* Mobile transceiver positions relative to pivot. */
Mqrs[0][0] = 0.0;
Mqrs[0][1] = -mbl_rad*cos(PI/6.0);
Mqrs[0][2] = -mbl_rad*sin(PI/6.0);
Mqrs[1][0] = 0.0;
Mqrs[1][1] = mbl_rad*cos(PI/6.0);
Mqrs[1][2] = -mbl_rad*sin(PI/6.0);
Mqrs[2][0] = 0.0;
Mqrs[2][1] = 0.0;
Mqrs[2][2] = mbl_rad;

/* These values aren't really known but are needed for the model. */
Pxyz[0] = Rbt_Rad*cos(Alpha);
Pxyz[1] = Rbt_Rad*sin(Alpha);
```

```
Pxyz[2] = -7.5;

Tht_Phi_Mu[0] = convert_angle( Alpha );
Tht_Phi_Mu[1] = 0.1;
Tht_Phi_Mu[2] = 0.1;

}
/*==================================================================*/
void aquire_inclns(int p,int m)
{

/* In the real system, this function would be used to aquire the
/* inclinometer data when inclinometers are utilized. */

if (p) Tht_Phi_Mu[1] = 0.1;
if (m) Tht_Phi_Mu[2] = 0.1;

}
/*==================================================================*/
void establish_approx( double guess[DIMS] )
{

/* Algorithm which works in conjunction with tetra_solve to calculate an
/* initial "guess" at the system unknowns.  It utilizes a closed form solution
/* of tetrahedrons with three vertices at known locations (fixed transceivers)
/* and solves for the fourth vertex position (mobile transceiver).  Although it
/* serves its purpose for this program, this algorithm must be improved if it
/* is to be used in the applied global positioning system.  (An alternative for
/* the applied global positioning system would be to use the last indicated
/* platform position and a rough approximation of the change in position to
/* obtain the initial guess). */

int
i,j,
sgn=1;

double
m_guess[N_MBL][3],
p_guess[3],
d[3];

for (i=0; i<3; i++) {
        d[0] = Intrp_Vel[0][i]*ToF[0][i];
        d[1] = Intrp_Vel[1][i]*ToF[1][i];
        d[2] = Intrp_Vel[2][i]*ToF[2][i];
        tetra_solve( &m_guess[i][0],Fxyz,d[0],d[1],d[2],0 );

        if ( fabs(m_guess[i][0]+m_guess[i][1]+m_guess[i][2])<EPS )
                for (j=0; j<3; j++) m_guess[i][j] = Mxyz[i][j];
}

for (i=0; i<3; i++) {
        d[i] = 0.0;
        for (j=0; j<3; j++) d[i] += Mqrs[i][j]*Mqrs[i][j];
        d[i] = sqrt( d[i] );
}

tetra_solve( &p_guess[0],m_guess,d[0],d[1],d[2],sgn );

guess[0] = p_guess[0];
guess[1] = p_guess[1];
guess[2] = p_guess[2];

if (Solve_Angles[0]) guess[3] = Tht_Phi_Mu[0];
if (Solve_Angles[1]) guess[4] = Tht_Phi_Mu[1];
if (Solve_Angles[2]) guess[5] = Tht_Phi_Mu[2];

for (i=0; i<N_Fxd; i++)
```

```
        guess[3+Solve_Angles[0]+Solve_Angles[1]+Solve_Angles[2]+i] = 1500.0;

}
/*==================================================================*/
void determine_Sigmas()
{

/* Evaluating velocity compensation variables. */

double
c_vel;

int
i,j;

c_vel  = (M_Node_Vel[0] + M_Node_Vel[1] + M_Node_Vel[2])/3.0;
for (i=0; i<N_FXD; i++) {
        for (j=0; j<N_MBL; j++) {
                if (Use_Comp) {
                        /* If averaging: */
                        Sigma[i][j] = (F_Node_Vel[i] + M_Node_Vel[j])/(F_Node_Vel[i] +
c_vel);
                        /* If using Harmonic Mean: */
                        /* Sigma[i][j] = ((F_Node_Vel[i] + c_vel)*M_Node_Vel[j])/
                        /*              ((F_Node_Vel[i] + M_Node_Vel[j])*c_vel); */
        }
                else Sigma[i][j] = 1.0;
        }
}

}
/*==================================================================*/
void incorporate_errors( double err_mtx[2*N_FXD+N_MBL+1][N_MBL] )
{

double
f_err, /* Maximum xyz dimensional errors for fixed xcvr positioning. */
offset_err, /* Offset error for time of flight data. */
t_r_err, /* Maximum pure random error in time of flight data. */
t_d_err, /* Maximum tof random error related to depth of xcvr. */
m_err, /* Maximum qrs dimensional errors for mobile xcvr positioning. */
phi_err, /* Maximum inclinometer angular error for phi measurement. */
mu_err, /* Maximum inclinometer angular error for mu measurement. */
random_err, /* Resulting tof random error. */
depth_factor, /* Depth factor for calculating depth related error. */
depth_related_err; /* Resulting depth related tof error. */

int
i,j;


/* Adding error to flange tranceiver locations. */
f_err = 1.0; /* In +/- millimeters. */
for (i=0; i<N_FXD; i++) {
        for (j=0; j<3; j++) {
                err_mtx[i][j] = (double)(f_err*(random(2000) - 1000)/1.e6);
                Fxyz[i][j] += err_mtx[i][j];
        }
}

/* Adding error to time array. */
offset_err = 0.5; /* In microseconds. */
t_r_err = 1.0; /* In +/- microseconds. */
t_d_err = 0.10; /* In +/- microseconds per meter. */
for (i=0; i<N_FXD; i++) {
        for (j=0; j<N_MBL; j++) {
                random_err = (double)(t_r_err*(random(2000) - 1000)/1.e9);
                depth_factor = (int)(Mxyz[j][2]*1000);
```

```
                depth_related_err = (double)(t_d_err*((random(2*depth_factor)
                        - depth_factor)/1.e9));
                err_mtx[i+N_FXD][j] = 1e-6*offset_err + random_err
                        + depth_related_err;
                ToF[i][j] += err_mtx[i+N_FXD][j];
        }
}

/* Adding error to assumed distances between M transceivers. */
m_err = 1.0; /* In +/- millimeters. */
for (i=0; i<N_MBL; i++) {
        for (j=0; j<3; j++) {
                err_mtx[i+2*N_FXD][j] = (double)(m_err*(random(2000) - 1000)/1.e6);
                Mqrs[i][j] += err_mtx[i+2*N_FXD][j];
        }
}

/* Adding error to inclinometer angles Phi and Mu. */
phi_err = 0.10; /* In +/- degrees. */
err_mtx[N_MBL+2*N_FXD][0] = (double)(phi_err*(random(2000)-1000)*PI/18.e4);
Tht_Phi_Mu[1] += err_mtx[N_MBL+2*N_FXD][0];
mu_err = 0.10; /* In +/- degrees. */
err_mtx[N_MBL+2*N_FXD][1] = (double)(mu_err*(random(2000)-1000)*PI/18.e4);
Tht_Phi_Mu[2] += err_mtx[N_MBL+2*N_FXD][1];

}
/*==================================================================================*/
void functions_evaluation( double fvec[],double xvec[] )
{

/* System equations. */

double
mxyz_eval[N_MBL][3],
delta,
vel,
rot_angs[3];

int
i,ii=0,j,k,
cmpnt = 0;

j = 0;
for (i=0; i<3; i++) {
        if (Solve_Angles[i]) {
                rot_angs[i] = xvec[3+j];
                j++;
        }
        else rot_angs[i] = Tht_Phi_Mu[i];
}

evaluate_rtn_mtx( rot_angs );

/* Evaluating the positions of the mobile transceivers based on the current
/* determined position and orientation of robot.  These evaluated M positions
/* at this point in the search are stored in the 2D array mxyz_eval. */
for (i=0; i<N_MBL; i++) {
        for (j=0; j<3; j++) {
                mxyz_eval[i][j]=0.0;
                for (k=0; k<3; k++)    mxyz_eval[i][j] += Rotation[j][k]*Mqrs[i][k];
                mxyz_eval[i][j] += xvec[j];
        }
}

/* Evaluating the system equations in the vector fvec. */
for (i=0; i<N_FXD; i++) if (Active_F[i]) {
        for (j=0; j<N_MBL; j++ ) {
                delta = 0.0;
```

```
                for (k=0; k<3; k++ ) {
                        delta += (Fxyz[i][k]-mxyz_eval[j][k])*
                                        (Fxyz[i][k]-mxyz_eval[j][k]);
                }
                vel = xvec[3 + Solve_Angles[0] + Solve_Angles[1] + Solve_Angles[2] + ii];
                fvec[cmpnt] = delta - (vel*Sigma[i][j]*ToF[i][j])*
                        (vel*Sigma[i][j]*ToF[i][j]);
                cmpnt++;
        }
    ii++;
}

}
/*==================================================================================*/
void jacobian_evaluation( double jcbn[][DIMS],double xvec[] )
{

/* Jacobian of linearized system. */

double
st,ct,sp,cp,sm,cm,
temp,
dR_dtht[3][3], /* Partial derivatives of Rotation matrix wrt Theta. */
dR_dphi[3][3], /* Partial derivatives of Rotation matrix wrt Phi. */
dR_dmu[3][3], /* Partial derivatives of Rotation matrix wrt Mu. */
rot_angs[3],
mxyz_eval[N_MBL][3],
dM_dtht[N_MBL][3], /* Partial derivatives of M positions wrt Theta. */
dM_dphi[N_MBL][3], /* Partial derivatives of M positions wrt Phi. */
dM_dmu[N_MBL][3]; /* Partial derivatives of M positions wrt Mu. */

int
i,ii=0,j,k,kk,
ang_pos[3],
ang_sum = Solve_Angles[0]+Solve_Angles[1]+Solve_Angles[2],
row = 0;

j = 0;
for (i=0; i<3; i++) {
        if (Solve_Angles[i]) {
                rot_angs[i] = xvec[3+j];
                j++;
        }
        else rot_angs[i] = Tht_Phi_Mu[i];
}

evaluate_rtn_mtx( rot_angs );

/* Evaluating positions of mobile transceivers. */
for (i=0; i<N_MBL; i++) {
        for (j=0; j<3; j++) {
                mxyz_eval[i][j]=0.0;
                for (k=0; k<3; k++) mxyz_eval[i][j] += Rotation[j][k]*Mqrs[i][k];
                mxyz_eval[i][j] += xvec[j];
        }
}

st=sin(rot_angs[0]);   ct=cos(rot_angs[0]);
sp=sin(rot_angs[1]);   cp=cos(rot_angs[1]);
sm=sin(rot_angs[2]);   cm=cos(rot_angs[2]);

if (Solve_Angles[0]) {
        dR_dtht[0][0] = -ct*sp*sm-st*cm;
        dR_dtht[0][1] = -ct*cp;
        dR_dtht[0][2] = ct*sp*cm-st*sm;
        dR_dtht[1][0] = -st*sp*sm+ct*cm;
        dR_dtht[1][1] = -st*cp;
        dR_dtht[1][2] = st*sp*cm+ct*sm;
```

```
        dR_dtht[2][0] = 0.0;
        dR_dtht[2][1] = 0.0;
        dR_dtht[2][2] = 0.0;
        for (i=0; i<N_MBL; i++) {
                for (j=0; j<3; j++) {
                        dM_dtht[i][j] = 0.0;
                        for (k=0; k<3; k++)
                                dM_dtht[i][j] += dR_dtht[j][k]*Mqrs[i][k];
                }
        }
}
if (Solve_Angles[1]) {
        dR_dphi[0][0] = -st*cp*sm;
        dR_dphi[0][1] = st*sp;
        dR_dphi[0][2] = st*cp*cm;
        dR_dphi[1][0] = ct*cp*sm;
        dR_dphi[1][1] = -ct*sp;
        dR_dphi[1][2] = -ct*cp*cm;
        dR_dphi[2][0] = sp*sm;
        dR_dphi[2][1] = cp;
        dR_dphi[2][2] = -sp*cm;
        for (i=0; i<N_MBL; i++) {
                for (j=0; j<3; j++) {
                        dM_dphi[i][j] = 0.0;
                        for (k=0; k<3; k++)
                                dM_dphi[i][j] += dR_dphi[j][k]*Mqrs[i][k];
                }
        }
}
if (Solve_Angles[2]) {
        dR_dmu[0][0] = -st*sp*cm-ct*sm;
        dR_dmu[0][1] = 0.0;
        dR_dmu[0][2] = -st*sp*sm+ct*cm;
        dR_dmu[1][0] = ct*sp*cm-st*sm;
        dR_dmu[1][1] = 0.0;
        dR_dmu[1][2] = ct*sp*sm+st*cm;
        dR_dmu[2][0] = -cp*cm;
        dR_dmu[2][1] = 0.0;
        dR_dmu[2][2] = -cp*sm;
        for (i=0; i<N_MBL; i++) {
                for (j=0; j<3; j++) {
                        dM_dmu[i][j] = 0.0;
                        for (k=0; k<3; k++)
                                dM_dmu[i][j] += dR_dmu[j][k]*Mqrs[i][k];
                }
        }
}

/* Evaluating jacobian elements. */
for (i=0; i<N_FXD; i++) if (Active_F[i]) {
        for (j=0; j<N_MBL; j++) {
        kk=0;
        for (k=0; k<3; k++)
                        if (Solve_Angles[k])
                                { jcbn[row][3+kk] = 0.0; ang_pos[k] = kk; kk++; }
                for (k=0; k<3; k++) {
                        jcbn[row][k] = -2.0*(Fxyz[i][k]-mxyz_eval[j][k]);
                        if (Solve_Angles[0])
                                jcbn[row][3+ang_pos[0]] +=
                                        -2.0*(Fxyz[i][k]-mxyz_eval[j][k])*dM_dtht[j][k];
                        if (Solve_Angles[1])
                                jcbn[row][3+ang_pos[1]] +=
                                        -2.0*(Fxyz[i][k]-mxyz_eval[j][k])*dM_dphi[j][k];
                        if (Solve_Angles[2])
                                jcbn[row][3+ang_pos[2]] +=
                                        -2.0*(Fxyz[i][k]-mxyz_eval[j][k])*dM_dmu[j][k];
                }
                for (k=0; k<N_Fxd; k++) {
```

```
                        if (k==ii)
                                jcbn[row][3+ang_sum+ii] =
                                        -2.0*xvec[3+ang_sum+k]*Sigma[i][j]*Sigma[i][j]*
                                        ToF[i][j]*ToF[i][j];
                        else jcbn[row][3+ang_sum+k] = 0.0;
        }
                row++;
        }
    ii++;
}

}
/*==================================================================================*/
/*=============================== End of Program ==================================*/
/*==================================================================================*/
```

Vita

Following an early childhood inspiration spawned by Legos™, Allen Maples went on to receive a Bachelor's Degree in Mechanical Engineering from Duke University in 1991. For reasons unknown even to himself, he also received an additional B.S. in the unrelated field of Electrical Engineering. He then went on to Virginia Tech where he became a Master's student under the guidance of Dr. Charles Reinholtz. Upon completion of a Master of Science Degree in Mechanical Engineering, Allen plans to work for Gore (makers of Gortex™) and never again write about himself in third person.