ARMY RESEARCH LABORATORY

The Case for Using the Spherical Model to Calculate the Interpolated Points in the Connectivity Software Deployment Module

by G. Welles Still and James F. Nealon

ARL-TR-4373 **February 2008**

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

DESTRUCTION NOTICE—Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5068

ARL-TR-4373 **February 2008**

The Case for Using the Spherical Model to Calculate the Interpolated Points in the Connectivity Software Deployment Module

G. Welles Still and James F. Nealon
Survivability/Lethality Analysis Directorate, ARL

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
February 2008	Final	October 2006 to September 2007

4. TITLE AND SUBTITLE

The Case for Using the Spherical Model to Calculate the Interpolated Points in the Connectivity Software Deployment Module

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

G. Welles Still and James F. Nealon (both of ARL)

5d. PROJECT NUMBER
84VVE1

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

U.S. Army Research Laboratory
Survivability/Lethality Analysis Directorate
Aberdeen Proving Ground, MD 21005-5068

8. PERFORMING ORGANIZATION REPORT NUMBER

ARL-TR-4373

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

A new software package undergoing development through joint efforts by the Missile Defense Branch at Aberdeen Proving Ground, Maryland, and the Communications Electronic Warfare Branch at Fort Monmouth, New Jersey, of the U.S. Army Research Laboratory's (ARL) Survivability/Lethality Analysis Directorate models and predicts the viability of communications links between moving nodes. Interpolation calculations will be needed to predict the location of the moving nodes between user-provided way points. The developers must decide which model of the earth to use as the basis of the calculations. Thus, a comparison was made between the National Geodetic Survey-provided computer programs Forward an Inverse based on the WGS84 Oblate Spheroid (OS) model, and a newly constructed program based on a Perfect Sphere (PS). The basis of the comparison was computational accuracy and speed. For a way point separation of 100 km or less, the maximum PS and WGS84 OS discrepancy was 1 meter—accurate enough for link budget applications. When Forward and Inverse were kept intact and ran from a batch file, it took 30 times longer to do the same calculations as the PS model. When Forward and Inverse were modified to compute efficiently, it took 1.5 times longer. To modify the OS codes took 2.5 times as long as it did to write the PS code. Based on these results, it is recommended that the PS model for the earth be used.

15. SUBJECT TERMS

great circle; interpolation; oblate spheroid; snapshot time; sphere; waypoint; WGS84

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			G. Welles Still
Unclassified	Unclassified	Unclassified	SAR	55	19b. TELEPHONE NUMBER (*Include area code*) 410-278-3377

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

List of Tables

Executive Summary

A new software package undergoing development through joint efforts by the Missile Defense Branch at Aberdeen Proving Ground, Maryland, and the Communications Electronic Warfare Branch at Fort Monmouth, New Jersey, of the U.S. Army Research Laboratory's (ARL) Survivability/Lethality Analysis Directorate models and predicts the viability of communications links between moving nodes. This connectivity software model consists of six modules: the Deployment Module, the Propagation Module, the Antenna Module, the Noise Module, the Link Budget Module, and the Connectivity Confidence Interval Module.

The first module to see development is the Deployment Module, which concerns itself with the location of communications platforms as a function of time. This module queries the user for locations of waypoints in terms of latitude, longitude, and altitude. This will be obtained from time and an elevation database such as the Digital Terrain Elevation Data (DTED) (*1*) for ground communications platforms or nodes. The module will then calculate the location of the moving nodes for times between the waypoint times called "snapshot times". For each snapshot time, the location of every single platform will be calculated and recorded. We can then model the node motion by playing back the interpolated snapshot positions much like rapidly projecting successive frames from a motion picture.

At this juncture in time, the code's developers face an important choice. When we make the interpolated calculations for the platforms' locations at the snapshot times, should the Deployment Module use the World Geodetic System 1984 (WGS84) model of the earth, which assumes that it is shaped like an oblate spheroid (OS) (*2*), or model of the earth as a perfect sphere (PS) (*3*)? The argument favoring the OS is that it is a model which has been extensively verified and is widely used (*2*). Software packages, such as Forward and Inverse (*4*), which are needed to do the interpolation calculations, are available free to the general public, thus bypassing the need to develop significant portions of the Development Module. Alternatively, using the PS as the basis for calculating the interpolation points would result in faster computations because the model is much simpler. The snapshot locations will be accurate enough for the connectivity calculations done by the subsequent modules.

Knowledge of just how accurate the PS-based calculations would be and how much computation time is saved would allow the developers to make an informed choice as to which model to use. Therefore, software based on the PS was written to calculate interpolated latitudes and longitudes. The accuracy and computation time were then compared to Forward and Inverse calculated longitudes and latitudes.

Three versions of Forward and Inverse were used. The first used executable C++ programs that required a batch program and auxiliary programs to maintain data format. The second used

Matrix Laboratory (MATLAB)[1] scripts, which required an additional script to oversee the process and control the data format. The third involved modifying FORTRAN (Formula Translator) source codes to combine Forward and Inverse into one program and altering some aspects of the program to make it more computationally efficient. The programs were run on three computation platforms: a 1-year-old Dell[2] laptop, a 3-year-old Dell desktop, and an 8-year-old ProGen[3] laptop.

For waypoint separation of 647 km, a maximum difference between the PS- and OS-based calculations was 29 meters. When the waypoint separation was reduced to 100 km, the maximum difference was reduced to 1 meter. Past link budget studies (*5*) show that waypoint separations of less than 100 km are typical. The best computation time for the OS-based software was the modified FORTRAN Forward and Inverse program, which took 1.5 times longer than the PS-based software on the Dell laptop. The longest computing time was the unmodified C++ programs, which took 30 times longer to compute than the PS-based program on the ProGen laptop.

Because of the much greater computation time required for the WGS84-based OS codes and the more-than-adequate accuracy of the PS-based calculations, it is recommended that the developers use the PS as the basis for calculation of the interpolated snapshot points. In this study, only two waypoints were considered. In a genuine deployment, many waypoints for many platforms will be modeled, greatly increasing the computation time. For practical problems, the accuracy of the PS will be within 1 meter, greatly exceeding the Levels 1 and 2 DTED standards (*1*). Modification of the OS WGS84 FORTRAN programs to make them compute faster took 2.5 times longer than it did to produce the PS-based program.

[1]MATLAB is a registered trademark of The MathWorks.

[2]Dell is a registered trademark of the Dell Computer Corporation.

[3]ProGen is a registered trademark of Reflex Nutrition Ltd.

1. Introduction

A new software package undergoing development through joint efforts by the Missile Defense Branch at Aberdeen Proving Ground, Maryland, and the Communications Electronic Warfare branch at Fort Monmouth, New Jersey, of the U.S. Army Research Laboratory's (ARL) Survivability/Lethality Analysis Directorate models and predicts the viability of communications links between moving nodes. The connectivity software is divided into six modules as illustrated in the hypothetical graphical user interface in figure 1.

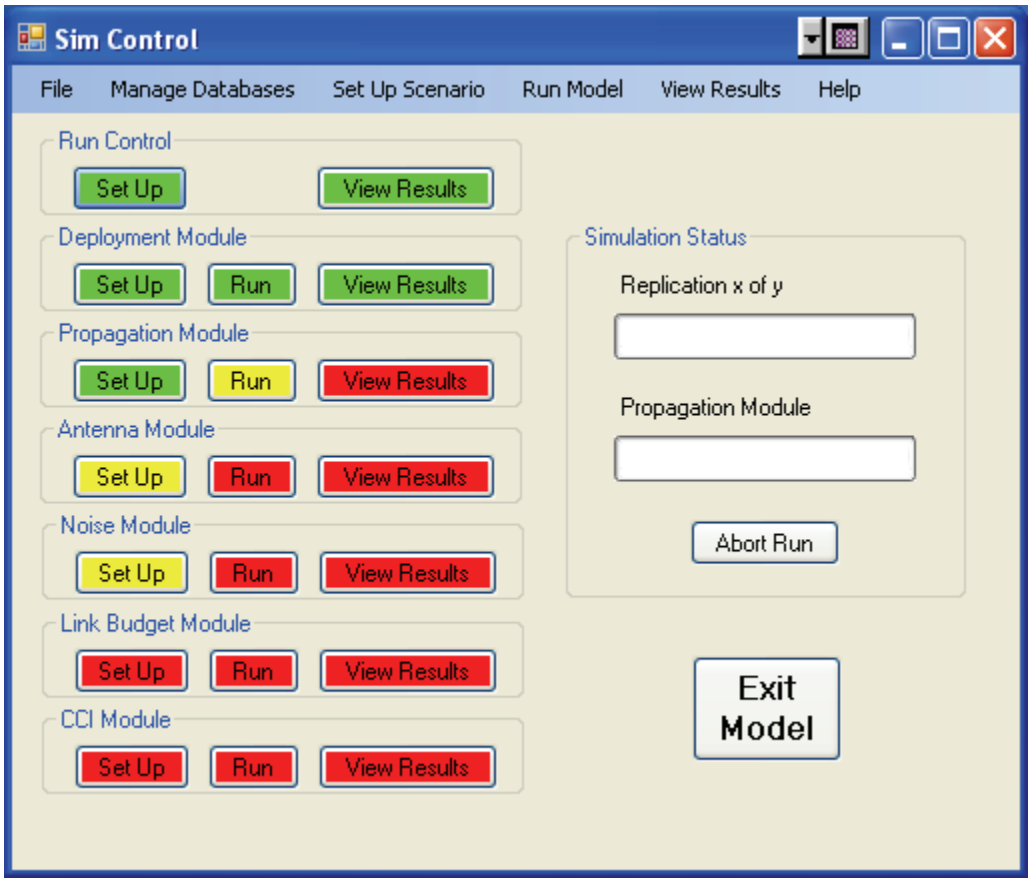

Figure 1. Modules of the connectivity software.

The Deployment Module models the locations and movements of the wireless network nodes. The Propagation Module contains data to calculate the attenuation of radio signals between the nodes attributable to atmospheric and terrain effects. The Antenna Module contains data concerning modeled antennae, including the gain, loss, and electromagnetic (EM) radiation pattern. The Noise Module computes receiver noise attributable to internal and external contributions, including EM noise which includes cosmic background radiation, the earth's thermal radiation, and jamming. The Link Budget Module takes the data from the previous modules and calculates the wireless link

mean signal-to-noise (S/N) ratio for each communications node. The Connectivity Confidence Interval (CCI) Module selects the appropriate standard deviation of the path loss associated with each link and computes the probability of successful signal reception. The model manages the six modules as they perform the calculations necessary to do the wireless network simulation.

1.1 Background

The Deployment Module is the first module undergoing development. This module tracks the motion of communications nodes. The approach is to allow the user to establish the location of a set of waypoints for each moving communication node in terms of latitude, longitude, and elevation. (Elevation is read from a database in the case of ground nodes.) The user also specifies waypoint arrival and departure information for each platform.

At certain times, the viability of the links between the moving nodes will have to be calculated. To do the calculation, the location of the platforms must be known. Invariably, these snapshot times will not correspond to the times when the platforms will be at their waypoints. Therefore, it will be necessary to interpolate the platforms' latitude, longitude, and elevation, based on a given velocity, and altitude profile. The number of interpolations may be few or could number into the hundreds for a single platform. Multiple platforms are expected to be portrayed; therefore, the time and memory space required for calculations become important.

The important question facing the developers is "which model of the earth should be used to calculate the interpolated points?" Computational time and geometric fidelity must both be considered. Basically, two geometric models are available: the World Geodetic System 1984 (WGS84) oblate spheroid (OS) model (*6*) and the perfect sphere (PS) model (*3*). The WGS84 OS is shaped like a flattened sphere; a cross section taken through a meridian line renders an ellipse with a minor radius of 6,356,752.3142 m (measured from a pole to the earth's center point), and a major a radius of 6,378,137 m (measured from the equator to the earth's center point) (*7*). Alternatively, a cross section of the perfect sphere through either a meridian or the equator produces a circle. Figure 2 shows the cross section of a constant meridian OS superimposed of the cross section of a PS. The radius of the circle in the figure is chosen so that the PS surface area will match the WGS84 OS surface area (*8*). This was done to show the difference in shape between the PS and the WGS84 OS. The OS in the figure is much more elliptical, flattened, and eccentric than a WGS84 OS to exaggerate the comparison with the sphere. In reality, the flattening of the earth is not discernible to the naked eye.

1.2 Purpose

If the developers chose to use the WGS84 OS model as the basis for interpolating the platform locations, much computation time would be required. For example, calculating the exact circumference of an ellipse requires the summation of an infinite series (*9*). To be practical, the series would have to be truncated, requiring many terms for every single snapshot point, and an algorithm would have to be developed to determine how accurate the sum must be and how many

terms need to be summed. An interpolation on the surface of an OS requires doing only a portion of a curve that will likely be more complicated than an ellipse.

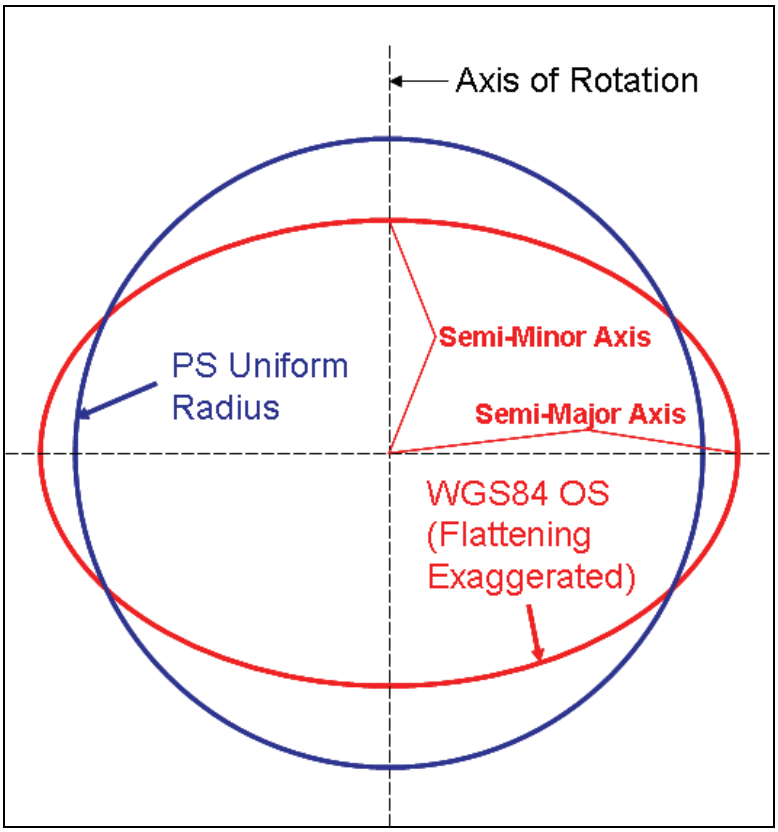

Figure 2. Cross sections of an OS (in red) and a PS (in blue). (The flattening of the OS in the figure is greatly exaggerated compared to a WGS84 OS to show the difference between the OS and PS earth models.)

Alternatively, an iterative process may be used (*10*), but it too would require many iterations per snapshot point and an algorithm to decide what is accurate enough and how many iterations are necessary. A third alternative involves using programs such as Forward and Inverse (offered as FORTRAN, C++, and MATLAB [Matrix Laboratory] listings and executable files by the National Geodetic Survey [NGS]) (*11*), which would still require multitudes of calculations for every snapshot point.

If the developers choose to use a PS as the interpolation basis, the calculation would be much faster; each snapshot point requires the calculation of only a single expression as opposed to a many-termed or multi-stepped iteration. Furthermore, a member of the development team has used the PS model in a previous analytical effort (*5*). Should any difficulty encountered in the development of modules beyond the Deployment Module be traced to that module, a resident expert in the PS model would be available as a problem-solving resource.

Therefore, if the OS is used, the location of the interpolated points will be more accurate, but if the PS is used, less computation time will be required. This is an important consideration in that the Deployment Module will be the first executed, so the less computation time used, the better. The importance of saving computation time becomes apparent in that the interpolated latitude/ longitude/altitude points must be computed many times for a multitude of waypoints, for a multitude of platforms.

Invariably, this discussion raises four specific questions on the part of the developers deciding between the OS and PS models: 1) How much more computation time would be required to use the WGS84 model? 2) How much accuracy is lost if the PS model is used? 3) How much time would it take to do the coding to accommodate the OS and PS model? 4) is there a particular problem that either of these models can solve which the other cannot? The purpose of performing this study is to perform a typical interpolation problem with the use of the two models so that quantitative data will be available to support the developers' choice of an earth model.

1.3 Scope

The comparison between the models was done at median sea level. The extant two-dimensional (2-D) versions of Forward and Inverse were used to allow for compatibility with the older C++ version of the codes. Furthermore, no C++ version of Forward and Inverse in three dimensions (3-D) is available at this time. These programs concern locations on the surface of an OS. Newer versions of Forward and Inverse (FORTRAN source code) address the 3-D solutions, only as line-of-sight or straight-line distances and sea-level distances, which is insufficient for modeling non-sea-level plat-form motion. Although the PS could address motion at any elevation, sea level was used to make the comparison consistent. The earlier 2-D versions of Forward and Inverse compute great circle distances at sea level. The 3-D versions also compute this distance, in addition to the line-of-sight distance for points that are higher than sea level.

2. Methodology

The process used is to calculate a set of interpolated points between two waypoints with both models. Then, the time needed to write or modify the software to do the calculations, the computation time needed for various computer platforms to do the computations, and the latitude/ longitude results calculated for the OS and PS models are compared. Several sets of waypoints with various distances between them were considered for three latitudes. The C++ version of Forward and Inverse can only calculate distances along the surface of an WGS84 OS, so only non-elevated paths were compared. The particular details of the methodology are enumerated next.

2.1 Process

The process represented is shown in figure 3. The assumptions that govern the motion are that the platform 1) moves at sea level, 2) moves along the shortest path between the two points according to the model (PS or OS) used to calculate the interpolated points, and 3) starts at the first point at 0 meters per second and accelerates at a constant rate until it reaches the second point traveling at 20 meters per second (or 48 mph). The final assumption is that the position between the two points is calculated every 10 seconds. Comparison is then made of the resultant interpolated points predicted by the OS and PS models.

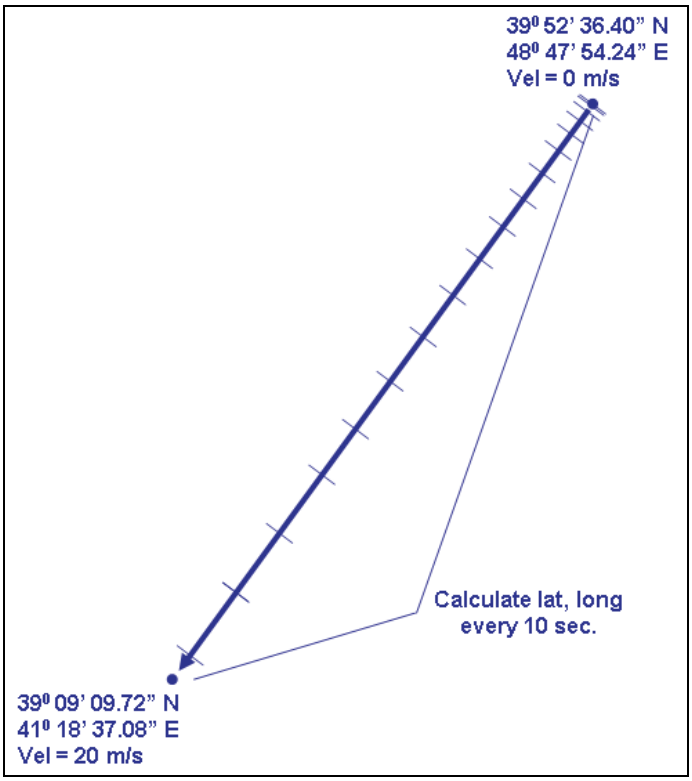

Figure 3. Platform movement between two waypoints with snapshot times.

2.2 Procedure

It is assumed that the path traveled by the platform is a great circle (*12*) as distinct from a rhumb line or loxodrome (*13*). A rhumb line crosses all meridians at the same angle, whereas a great circle crosses different meridians with different angles. Although used extensively in navigation, a rhumb line is not (unlike a great circle) the shortest distance between two points. Although nearly indistinct from a great circle for short distances near the equator, a rhumb line's spiral curvature becomes evident for long distances near the poles. Alternatively, a great circle traces the shortest path between two points for all latitudes and all distances.

The procedure involved the method used to incorporate the OS model and the PS model to do the calculations. The input file for all processes (way.dat which is listed in appendix A) is an ASCII

(American Standard Code for Information Interchange) file containing the latitudes, longitudes, elevations, and times for the two waypoints. Four processes took the information from way.dat and calculated the interpolated points.

2.2.1 FORTRAN Code Using the PS

The first involves using the PS model and is illustrated in figure 4. Test.exe is a FORTRAN program written by the author to calculate the interpolated points from the points in way.dat. It works by using the fact that a great circle is the shortest path on the surface of the sphere between the two points.

Figure 4. Procedure for calculating the interpolated points with the PS model.

The equation for a great circle on a sphere is

$$\tan \phi = \tan \phi_{max} \cos (\theta - \theta_0) \tag{1}$$

where ϕ is the latitude, θ is the longitude, and ϕ_{max} and θ_0 are respectively the latitude and longitude of the point on the great circle which is farthest north from the equator. By knowing two points on the great circle (ϕ_1, θ_1) and (ϕ_2, θ_2), one can calculate ϕ_{max} and θ_0.

Substituting each set of points into equation 1 to get two equations, then eliminating ϕ_{max}, the value for θ_0 is found to be

$$\theta_0 = \text{Arc tan } [(\tan \phi_1 \cos \theta_2 - \tan \phi_2 \cos \theta_1) / (\tan \phi_2 \sin \theta_1 - \tan \phi_1 \sin \theta_2)]. \tag{2}$$

By substituting (ϕ_2, θ_2) into equation 1 and using the value for θ_0 obtained in equation 2, we find the value for ϕ_{max}

$$\phi_{max} = \text{Arc tan } [\tan \phi_2 / \cos (\theta_2 - \theta_0)]. \tag{3}$$

ϕ_{max} and θ_0 are then used to transform the waypoints (ϕ_1, θ_1) and (ϕ_2, θ_2) to a primed coordinate system so that they are both on the primed equator:

$$\theta_1' = \text{Arc cos } [\cos (\theta_1 - \theta_0) \cos \phi_1 / \cos \phi_{max}] \tag{4a}$$

$$\theta_2' = \text{Arc cos } [\cos (\theta_2 - \theta_0) \cos \phi_2 / \cos \phi_{max}] . \tag{4b}$$

Having read the times t_2 and t_1 corresponding to the waypoints from the input data file way.dat and using the fact that the waypoints are both on the primed equator, we calculate the arc acceleration α with this equation:

$$\alpha = 2 (\theta_2' - \theta_1') / (t_2 - t_1)^2. \tag{5}$$

Since the interpolated times t_i have already been selected as occurring once every 10 seconds, the corresponding primed interpolated latitude θ_i' is calculated with

$$\theta_i' = \theta_1' + \alpha\,(t_i - t_1)^2 / 2. \tag{6}$$

The primed interpolated longitude is then converted back to the original coordinate system longitude and latitude with $\phi_{\mathbf{max}}$ and θ_0 once again:

$$\phi_i = \text{Arc sin}\,(\cos\theta_i' \sin\phi_{\mathbf{max}}), \tag{7a}$$

$$\theta_i = \text{Arc cos}\,(\cos\theta_i' \cos\phi_{\mathbf{max}} / \cos\phi_i') + \theta_0. \tag{7b}$$

The result, 6,473 interpolated sets of latitudes, longitudes, elevations (kept at 0 for this study), and times, were then stored in a second ASCII file called "snap.out". The time for computation was then measured with a stopwatch as indicated by the clock in figure 4.

We calculated the arrival time of the platform (64717.85189 seconds) in figure 3 by assuming that 1) the platforms starts from rest at time t = 0, 2) the platform accelerates at a constant rate a, and 3) the arrival velocity is 20 meters per second. The distance between the points was calculated from their latitudes and longitudes given in figure 3. Two equations of motion, $D = a\,t^2/2$ and $v_{final} = a\,t$, were then used to solve for the acceleration a and the time t.

2.2.2 C++ Codes Using the OS

The second procedure used two executable C++ files, Forward and Inverse, both available from the U.S. Army Topographic Engineering Center (TEC) since November 1995 (*14*). Forward calculates the latitude and longitude at sea level, given the starting point latitude and longitude, a bearing, and a sea-level distance. Inverse calculates the bearing and distance between two points, given their latitudes and longitudes. Both offer the user a choice of several models of the earth; we chose the WGS84 OS.

These particular executable files originated as C++ programs. The way they were used is outlined in figure 5. We begin with the same ASCII file (way.dat) that the PS FORTRAN file test.exe used. Although the waypoint latitudes and longitudes were contained in way.dat, the information had to be rendered in a format that Inverse.exe could understand. The short FORTRAN file Test1.exe took the waypoint latitude and longitude information from wat.dat and wrote it to a file in a format that Inverse.exe understood called "way1.crd". Inverse.exe then calculated the distance and bearing between the two waypoints and recorded them in a file called "way1.out". Sections of these files are included in appendices A through F.

The process used was to take the beginning and ending times, t_i and t_f from way.dat, the distance d from way1.out, and use them to calculate the acceleration constant "a" via the equation

$$a = 2\,d / (t_f - t_1)^2. \tag{8}$$

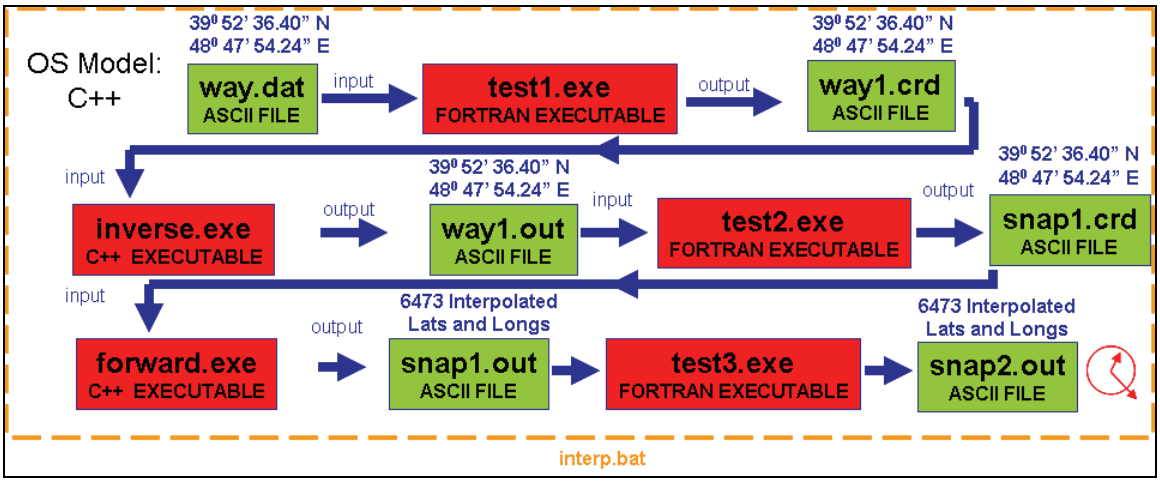

Figure 5. How the C++ executable files forward.exe and inverse.exe were used to do the interpolation.

Then, using the fact that the interpolation time t_i occurred every 10 seconds, we calculated the acceleration a interpolated distances d_i with the formula

$$d_i = a\,(t_i - t_1)^2 / 2. \qquad (9)$$

The bearing was read from way1.out, with the interpolated distances, and the latitude and longitude of the first waypoint and stored in the file snap1.crd. Forward.exe then used the first waypoint longitude, latitude, bearing, and interpolated distances to calculate 6,473 interpolated points and store the information in a file called "snap1.out". Test3.exe then read snap1.out and put the interpolated latitudes, longitudes, elevations (0 meters for this study) and times in a file called "snap2.out". This last step ensured that the format of the interpolated points matched the format of the file snap.out, calculated with the PS model. All the executables were run with the use of a batch file called interp.bat. The calculation time was measured with a stopwatch. For readers wishing to know more details of the format of the ASCII files described in figure 5, excerpts of the files way.dat, way1.crd, way1.out, snap1.crd, snap1.out, and snap2.out are included in the appendices.

2.2.3 MATLAB Scripts Using the OS

A similar process that employed downloaded MATLAB scripts of Forward (*15*) and Inverse (*16*) based on the OS WGS84 model were also used. Figure 6 shows the process. It was necessary for the authors to construct an additional script for flow control and to produce the output file path_conta.txt in a format identical to snap.out and snap2.out. Then forward was run once to get the forward bearing and the distance between the waypoints. Using the azimuth, we re-calculated the distance to correspond to the interpolated time under constant acceleration, and we calculated the latitude and longitude by calling Inverse repeatedly. Additional scripts placed the data in the output file path_conta.txt and ensured that the data in path_conta.txt data had a format identical to snap.out and snap2.out. This operation was timed with MATLAB timing routines and was also timed with a stopwatch.

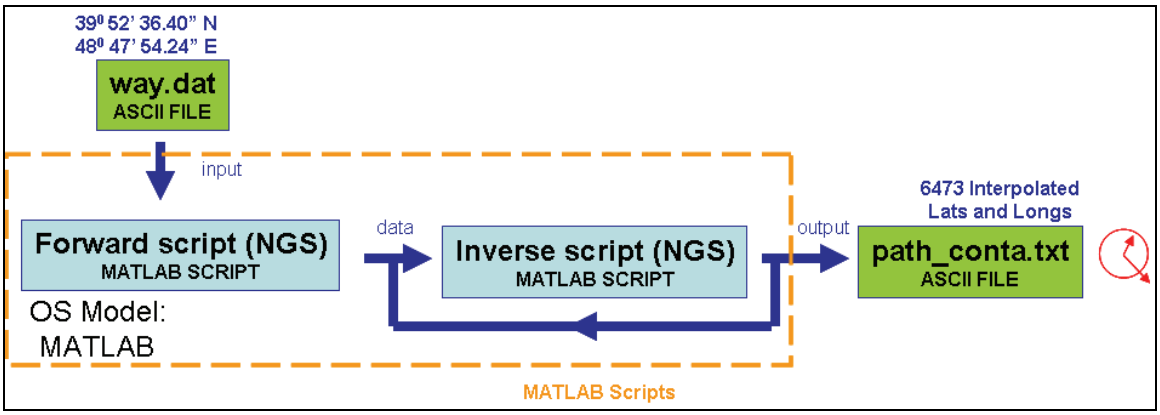

Figure 6. The process used in calculating interpolated latitude and longitude points via downloaded Forward and Inverse MATLAB scripts.

2.2.4 FORTRAN Program Using the OS

NGS provides FORTRAN listings for Forward and Inverse (*17*). Rather than use the programs intact, the approach this time was to modify the listings and splice the two programs together to form one executable file, as shown in figure 7. Additionally, the "do" loops for repeatedly using Inverse were chosen so as to define the bearing only once, and the number of output was reduced so as to minimize the computation time needed to calculate the latitudes and longitudes. The program was also modified to calculate the distance, with an assumed constant acceleration. The two listings were combined and then compiled to form a single executable test4.exe. The results were placed in the file snap3.out. The timing for this operation was also done with a stopwatch.

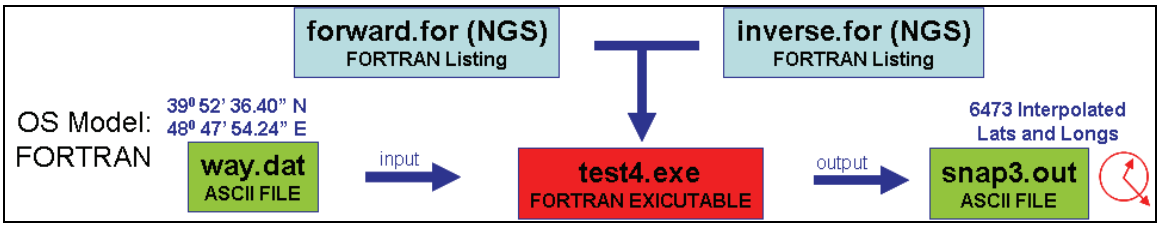

Figure 7. The FORTRAN listings were downloaded for Forward and Inverse and combined to form a single executable program, test4.exe.

2.3 Platforms

The interpolation programs were also ran on a variety of platforms (see table 1). They consisted of an 8-year-old ProGen laptop running Windows[4] 98, A 3-year-old desktop running Windows 2000, and a 1-year-old Dell laptop running Windows XP. Only the Desktop ran MATLAB, while all three machines ran the FORTRAN and C++-based programs. The reason why three platforms were used was to establish the effect that processor speed and model difference (OS and PS) had on computation time.

[4]Windows is a trademark of the Microsoft Corporation.

Table 1. The three different computing platforms used.

Computer Manufacturer	Age (years)	Operating System	RAM	Processor
ProGen	8	Windows 98	64 MB	Pentium II
Dell[a]	3	Windows 2000	1 GB	2.2-GHz single
Dell	1	Windows XP	1 GB	T2400 @ 1.83 GHz

[a]MATLAB platform

3. Analysis

The analysis consisted of tracking the running time of the calculations done with the algorithms, based on the OS and PS models. It also consisted of recording the time required to code the algorithms. The bulk of the analysis consisted of comparing the spatial difference between the interpolated longitudes and latitudes and considering a special problem where the waypoints are on the equator but nearly 180 degrees apart.

3.1 Analysis for the Spatial Difference Between the OS- and PS-based Algorithms

We begin with a short derivation of the equations used for the analysis. Figure 8 shows the superposition of a latitude/longitude coordinate system with a rectilinear coordinate system; ϕ is the latitude, which varies from -90 to 90 degrees. The designation "north latitude" would indicate a positive value, while "south latitude" indicates a negative value; θ is the longitude, varying from -180 to + 180 degrees. "East" corresponds to positive latitude, while "West" connotes negative latitudes. R is the distance from the earth's center and is the sum of the mean sea level and the elevation. R is always positive and can have a value between zero and infinity.

The superimposition of a 3-D rectilinear coordinate system on the latitude and longitude system aids the calculation of distance. The x axis starts at the earth's center point and passes through the point where the prime meridian (0 degrees longitude) and equator (0 degrees latitude) intersect. Likewise, the y axis starts at the center point but goes through the point at the intersection of the equator (0 degrees latitude) and 90 degrees east longitude. The z axis begins at the earth's center as well but passes through the north pole (90 degrees latitude).

To convert from the latitude/longitude coordinate system to the rectilinear system involves employing simple geometric relationships. In the vertical right triangle, the side opposite the latitude angle ϕ in figure 8 is the z coordinate.

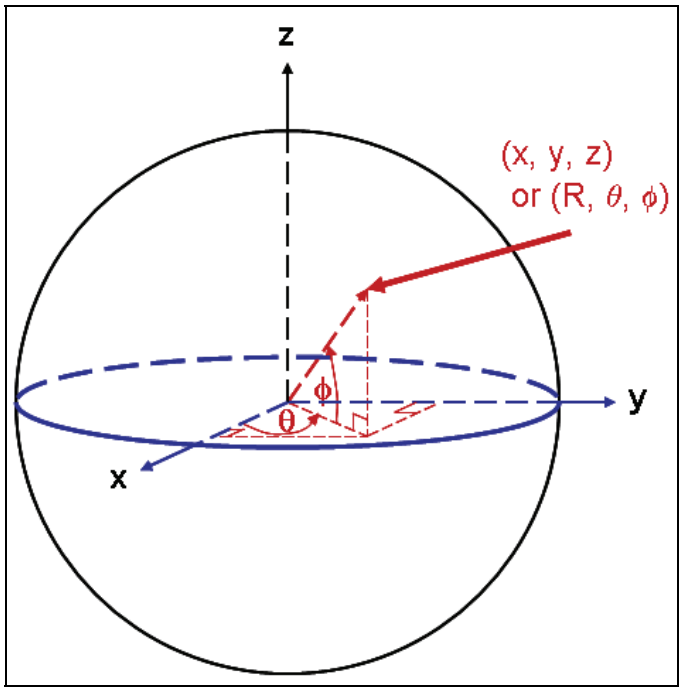

Figure 8. The latitude-longitude coordinate system superimposed on the rectilinear coordinate system.

Expressed in terms of the latitude-longitude system, this becomes

$$z = R \sin \phi. \tag{10}$$

The side adjacent to the latitude angle ϕ becomes the hypotenuse for the two right triangles in the x-y plane shown in figure 1, with a value of $R \cos \phi$. The side opposite the longitude angle θ is the y coordinate, which in terms of R, ϕ, and θ is

$$y = R \cos \phi \sin \theta. \tag{11}$$

The side adjacent to the latitude angle ϕ is the x coordinate, rendered

$$x = R \cos \phi \cos \theta. \tag{12}$$

Figure 9 shows the platform paths that were computed based on the two models. At a given time, t, the location of the moving platform will be computed at two different points. We wish to compute the distance between them. Figure 10 shows a cross-sectional wedge showing the straight line distance Δ and the angle δ between the points computed by the OS and PS models at time t. As is evident by figure 10, the straight line distance Δ is related to the great circle distance by way of the angle δ by $\sin(\delta/2) = \Delta/(2R)$. The great circle distance is $R\,\delta$. We suspect that the distance between the points is on the order of hundreds of kilometers. The radius of the earth is about 6300 kilometers. Since the radius of the earth is about 1000 times larger than Δ or $R\,\delta$, δ becomes very small, so we may employ the well-known asymptotic relationship $\sin(\delta/2) \approx \delta/2 = \Delta/(2R)$ or put another way, the great circle distance $\delta R = \Delta$, the straight line distance. If we find the straight line

distance, we have found the great circle distance. Using data presented from the Results section, we will justify this assumption.

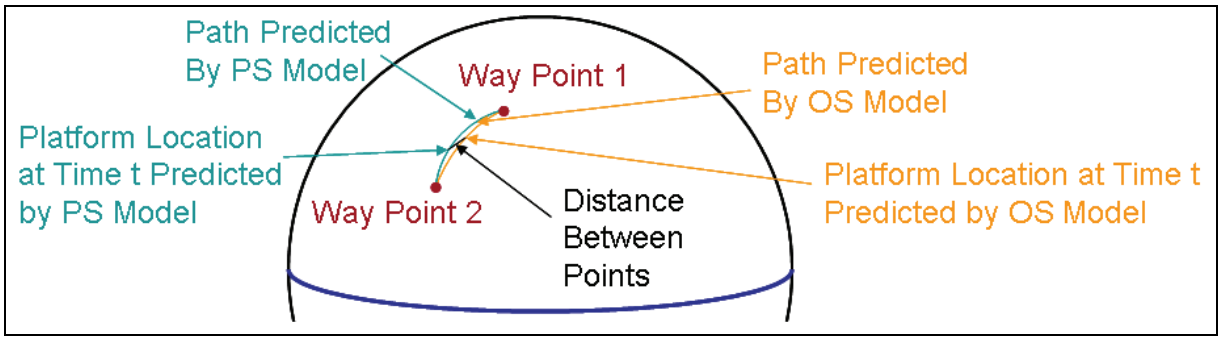

Figure 9. The platform paths and the distance between the points calculated at time t.

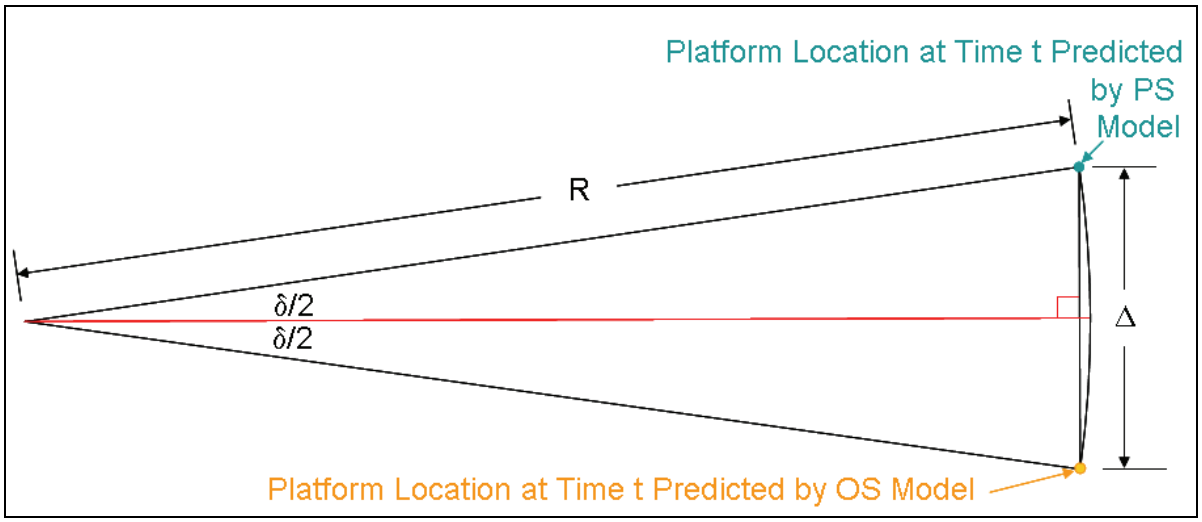

Figure 10. Distance and angle between points at time t predicted by the OS and PS models.

If we know the rectilinear coordinates of the two points, the distance between them is calculated with the sum of the squares of the differences of the x, y, and z coordinates:

$$\Delta^2 = (x_{OS} - x_{PS})^2 + (y_{OS} - y_{PS})^2 + (z_{OS} - z_{PS})^2. \tag{13}$$

We do not have the rectilinear coordinates, but we do have the latitudes, longitudes, and the radii of the OS and PS. With equations 10, 11, and 12, 13 becomes

$$\begin{aligned} \Delta^2 = &\left(R_{OS}\cos\phi_{OS}\cos\theta_{OS} - R_{PS}\cos\phi_{PS}\cos\theta_{OS}\right)^2 + \\ &\left(R_{OS}\cos\phi_{OS}\sin\theta_{OS} - R_{PS}\cos\phi_{PS}\sin\theta_{OS}\right)^2 + \\ &\left(R_{OS}\sin\phi_{OS} - R_{PS}\sin\phi_{PS}\right)^2. \end{aligned} \tag{14}$$

Next, the squares for each term are evaluated and listed, rendering equation 14 as

$$\Delta^2 = {R_{OS}}^2 \cos^2\phi_{OS} \cos^2\theta_{OS} - 2R_{OS}\cos\phi_{OS}\cos\theta_{OS}\,R_{PS}\cos\phi_{PS}\cos\theta_{PS} + {R_{PS}}^2\cos^2\phi_{PS}\cos^2\theta_{PS} + {R_{OS}}^2\cos^2\phi_{OS}\sin^2\theta_{OS} - 2R_{OS}\cos\phi_{OS}\sin\theta_{OS}\;R_{PS}\cos\phi_{PS}\sin\phi_{PS} + {R_{PS}}^2\cos^2\phi_{PS}\sin^2\theta_{PS} + {R_{OS}}^2\sin^2\phi_{OS} - 2R_{OS}\sin\phi_{OS}\;R_{PS}\sin\phi_{PS} + {R_{PS}}^2\sin^2\phi_{PS}. \quad (15)$$

Applying the well-known trigonometric identity $\sin^2\alpha + \cos^2\alpha = 1$ (*18, p 433*) twice, it is easy to show that $\cos^2\phi\cos^2\theta + \cos^2\phi\sin^2\theta + \sin^2\phi = 1$. This reduces six terms of equation 15 to two. With some re-grouping of the remaining three terms, equation 15 becomes

$$\Delta^2 = {R_{OS}}^2 + {R_{PS}}^2 - 2R_{OS}R_{PS}\left[\cos\phi_{OS}\cos\phi_{PS}\left(\cos\theta_{OS}\cos\theta_{PS} + \sin\theta_{OS}\sin\theta_{PS}\right) + \sin\phi_{OS}\sin\phi_{PS}\right] \quad (16)$$

Next, we employ another well-known trigonometric identity: $\cos(A + B) = \cos A\cos B - \sin A\sin B$ (*18, p 434*). Application of this identity further simplifies equation 16 to read

$$\Delta^2 = {R_{OS}}^2 + {R_{PS}}^2 - 2R_{OS}R_{PS}\left[\cos\phi_{OS}\cos\phi_{PS}\cos\left(\theta_{OS} - \theta_{PS}\right) + \sin\phi_{OS}\sin\phi_{PS}\right]. \quad (17)$$

From figure 10, we reasoned that if Δ is small compared to the R_{OS}, or R_{PS}, then δ must be small, so that $\sin\delta \approx \delta$ (*19*). Since $\theta_{OS} - \theta_{PS}$ is the longitudinal component of δ, it must be smaller than δ. Then the angle $\theta_{OS} - \theta_{PS}$ must also be a very small angle. Expanding the term cos A into a Taylor series, if A is a small angle, we may limit the expansion to the first two terms and ignore the rest of the series with little loss in accuracy, so that $\cos A \approx 1 - A^2/2$ (*18, p 736*). Applied to equation 17,

$$\Delta^2 \approx {R_{OS}}^2 + {R_{PS}}^2 - 2R_{OS}R_{PS}\left[\cos\phi_{OS}\cos\phi_{PS} - \frac{\cos\phi_{OS}\cos\phi_{PS}\left(\theta_{OS} - \theta_{PS}\right)^2}{2} + \sin\phi_{OS}\sin\phi_{PS}\right]. \quad (18)$$

Again, using the identity $\cos(A + B) = \cos A\cos B - \sin A\sin B$, equation 18 becomes

$$\Delta^2 \approx {R_{OS}}^2 + {R_{PS}}^2 - 2R_{OS}R_{PS}\left[\cos\left(\phi_{OS} - \phi_{PS}\right) - \frac{\cos\phi_{OS}\cos\phi_{PS}\left(\theta_{OS} - \theta_{PS}\right)^2}{2}\right]. \quad (19)$$

If δ is small, then the vertical component $\phi_{OS} - \phi_{PS}$ must be smaller. Re-applying the relationship $\cos A \approx 1 - A^2/2$, equation 19 simplifies a bit more:

$$\Delta^2 \approx {R_{OS}}^2 + {R_{PS}}^2 - 2R_{OS}R_{PS}\left[1 - \frac{\left(\phi_{OS} - \phi_{PS}\right)^2}{2} - \frac{\cos\phi_{OS}\cos\phi_{PS}\left(\theta_{OS} - \theta_{PS}\right)^2}{2}\right]. \quad (20)$$

The polar radius and the equatorial radius of the earth differ by 0.3%. Since the PS radius is between the equatorial and polar radius of the OS, it follows that R_{OS} and R_{PS} differ by even less (see figure 2). The OS radius at the latitude for which this study was performed (39 degrees north latitude), the OS and PS radius must be very nearly equal, so that $R_{OS} \approx R_{PS}$. Applying this to equation 20, we reach the equation with which the difference calculation was performed:

$$\Delta^2 = {R_{OS}}^2 \left[\cos^2 \phi_{OS} \left(\theta_{OS} - \theta_{PS} \right)^2 + \left(\phi_{OS} - \phi_{PS} \right)^2 \right]. \quad (21)$$

After the codes calculated the latitude and longitude corresponding to platform location at a particular interpolated time, the distance between the results predicted by the PS and OS models were calculated and then plotted via equation 21. Δ is the difference in meters between the two interpolations; ϕ_{OS} and θ_{OS} are the latitude and longitude reported for the OS model, and ϕ_{PS} and θ_{PS} are the PS latitude and longitude. For equation 21 to report the correct results, ϕ_{OS}, ϕ_{PS}, θ_{OS}, and θ_{PS} must be in radians.

The cross section of the WGS84 OS model of the earth is an ellipse (*21*). Thus, R_{OS} becomes a function of the latitude ϕ_{OS}. Therefore, the exact equation for the radius must be the equation for an ellipse in polar coordinates, namely,

$$\frac{1}{{R_{OS}}^2} = \frac{\cos^2 \phi_{OS}}{A^2} + \frac{\sin^2 \phi_{OS}}{B^2} \quad (22)$$

where R_{OS} is the radius of the earth at latitude ϕ_{OS}, A is the equatorial radius in meters (6378137), and B is the polar radius in meters (6356752.3142) (*7*).

The difference between the OS and PS calculated points of figure 9 may be extended to a flat surface. Because the distance Δ is small compared to the earth's radius, the area around the path difference line may be approximated as a flat surface, as has been done with figure 11. To obtain the distance between points Δ, we employ the Pythagorean theorem, $h^2 + v^2 = \Delta^2$ (*20*). The latitude distance, v, is simply the earth's radius at the point times when the angle latitude changes, or R_{OS} (ϕ_{OS} - ϕ_{PS}). The longitude distance h is also the earth's radius times the longitude change, but it must be multiplied by the cosine of the latitude. That is because the distance for a given longitude difference increases as the latitude decreases. (Consult figure 8 and equations 11 and 12.) Therefore, $h = R_{OS} (\theta_{OS} - \theta_{PS}) \cos \phi_{OS}$. Making the substitutions for h and v in the Pythagorean theorem, we arrive at an expression for the distance Δ:

$$\Delta^2 = {R_{OS}}^2 \left[\cos^2 \phi_{OS} \left(\theta_{OS} - \theta_{PS} \right)^2 + \left(\phi_{OS} - \phi_{PS} \right)^2 \right]. \quad (23)$$

This expression is identical to equation 21, giving us confidence that this formula correctly calculates Δ^2.

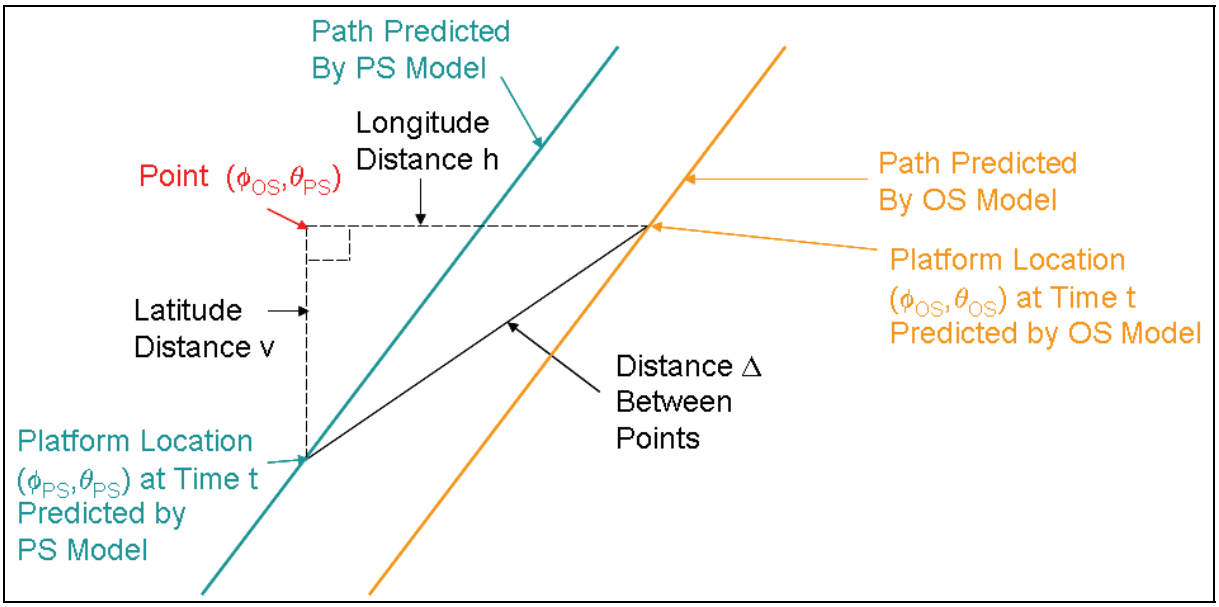

Figure 11. Figure 9 extended to a flat surface.

3.2 Analysis for the Special Problem

The greatest difference between the interpolated longitude and latitude points predicted by the OS and PS models should be for the special case where the two waypoints are on the equator and very nearly 180 degrees apart. This situation is depicted in figure 12. The two waypoints on the equator are where the line $y = y_0$ and $z = 0$ intersects the OS. This line through the OS connecting the two waypoints on the OS surface is parallel to the x axis a consistent perpendicular distance y_0 from the x axis, and lies in the xy plane. It is also perpendicular to the y axis and skew to the z axis. Because the OS is flattened at the poles, the arc of shortest distance on the OS which connects the points travels closer to the pole than to the equator. To see just how close, the situation should be viewed in cross section of the yz plane, as shown in figure 13.

The cross section taken through the poles is an ellipse (*21*) with equation 24 (*22*):

$$1 = y^2 / a^2 + z^2 / c^2 \tag{24}$$

where a is the equatorial radius and c is the polar radius. The shortest path line on the surface of the OS intersects the cross-sectional ellipse at a point fairly near the pole. The straight line connecting the waypoints intersects the y axis at the point $(0,y_0,0)$. These two ponts are separated by a distance D. The square of the distance D is represented by the equation

$$D^2 = (y - y_0)^2 + z^2, \tag{25}$$

which can be rendered in terms of a single variable after we solve the elliptical cross-sectional equation (equation 24) for z^2 and substitute into equation 25:

$$D^2 = (y - y_0)^2 + c^2 (1 - y^2 / a^2). \tag{26}$$

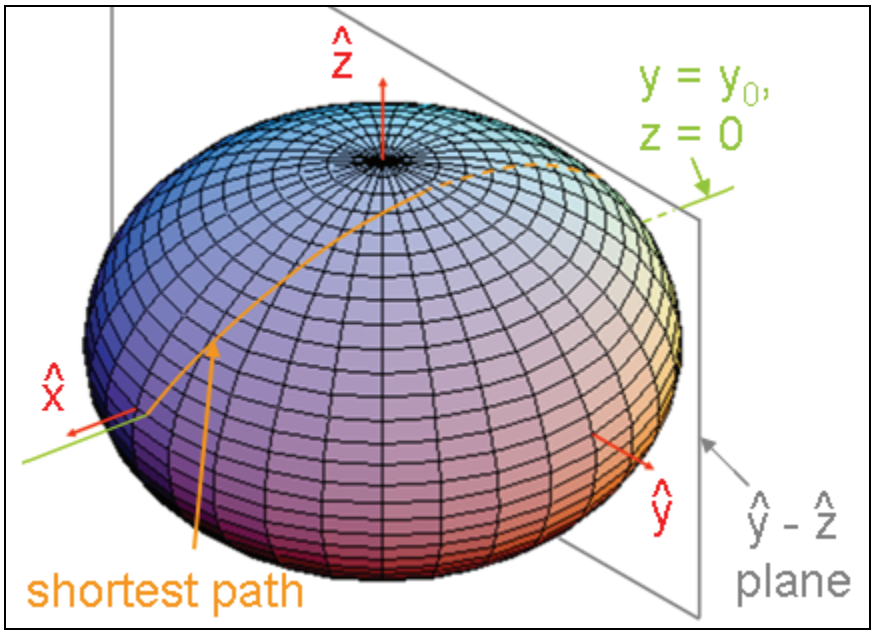

Figure 12. The shortest path on the OS connecting two equatorial waypoints nearly opposite each other goes closer to the poles than to the equator.

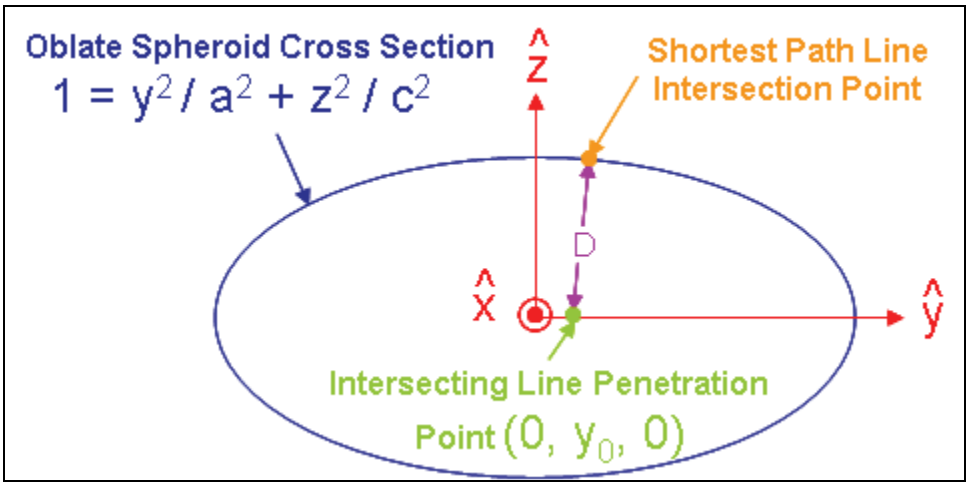

Figure 13. The xy cross section of the OS of figure 12.

It follows that the shortest path line along the surface of the OS should be the line closest to the straight line intersecting the waypoints. Therefore, the point on the ellipse where the shortest mean sea level path intersects the cross section should be a point on the ellipse that is closest to $(0, y_0, 0)$. This point should have a value for y so that the distance squared D^2 is minimized. The value for y is easily found if we take the y derivative of equation 26, and set it to zero. The answer is

$$y = y_0 / (1 - c^2 / a^2). \qquad (27)$$

The greatest value y can have is the equatorial radius a, so substituting that value in equation 27, and solving for y_0 renders the maximum value possible for y_0.

$$y_0 = a\,(1 - c^2 / a^2\,). \tag{28}$$

For two points on the equator, the line connecting them on the surface of an OS does NOT follow the equator if they are within 2 y_0 of being 180 degrees opposite each other. If they are closer than that, the closest path lies on the equator. Plugging in the WGS84 values for the earth, for a = 6,378,137.0 meters, and c = 6,356,752.314 245 m (*6*), we find y_0 = 42,697.67 meters.

Therefore, if two waypoints on the equator are between 180 and 179.2329 degrees apart in longitude, the shortest path connecting them is NOT along the equator. Any software calculating the interpolated points for this special circumstance should provide interpolated points that are not on the equator. Note that by symmetry, the shortest mean sea level path on an OS also may pass by the southern pole as well as the northern pole. Conversely, the path connecting two points on the equator, which are less than 179.2329 degrees apart, has a shortest mean sea level path on the equator, and software calculating the interpolated points should provide points that are on the equator.

4. Results

The results are presented in three sections. The first section compares the spatial differences for the interpolated points calculated with the use of algorithms based on the WGS84 OS model and the PS model. The second section compares the computation time required for the various pieces of software on the available platforms and the time required for the coders to write the code. The final section is devoted to the results calculated for two waypoints on the equator but nearly opposite each other, as outlined in section 3.2.

4.1 Spatial Differences Between the OS- and PS-Based Algorithms

The results for the calculations made between the C++ OS software, the FORTRAN OS software, the MATLAB OS software, and the FORTRAN PS software are shown in figure 14. Even though only one line is shown in the graph, the four plots are so close to each other that they appear as a single line. Note that the bearings of all the plots change with distance, consistent with a shortest path sea level route.

At a resolution allowing visibility of the entire path, all four processes agree. The small separation between the PS FORTRAN calculations and the three sets of OS calculations can be seen when the vertical axis is expanded by a factor of 80 and the horizontal scale by a factor of 160. Figure 15 shows the results. The portion of the graph that displays the greatest difference between the three OS-based software calculations and the PS-based calculation is displayed in this rather limited window. The three WGS84-based calculations appear identical, even with this very high level of magnification because the same base algorithms were used for the three OS-based codes.

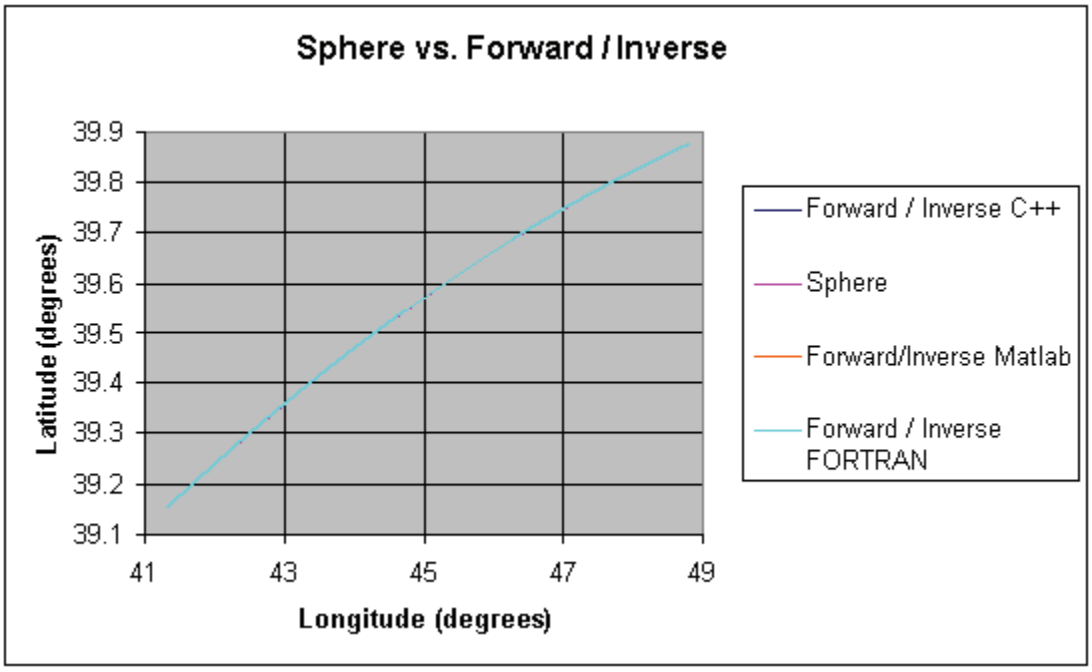

Figure 14. Results for interpolated points as calculated with the PS, the C++ version, the MATLAB version, and the FORTRAN version of Forward and Inverse. (The graph shows only one line because the differences between them are too small to be seen at this scale.)

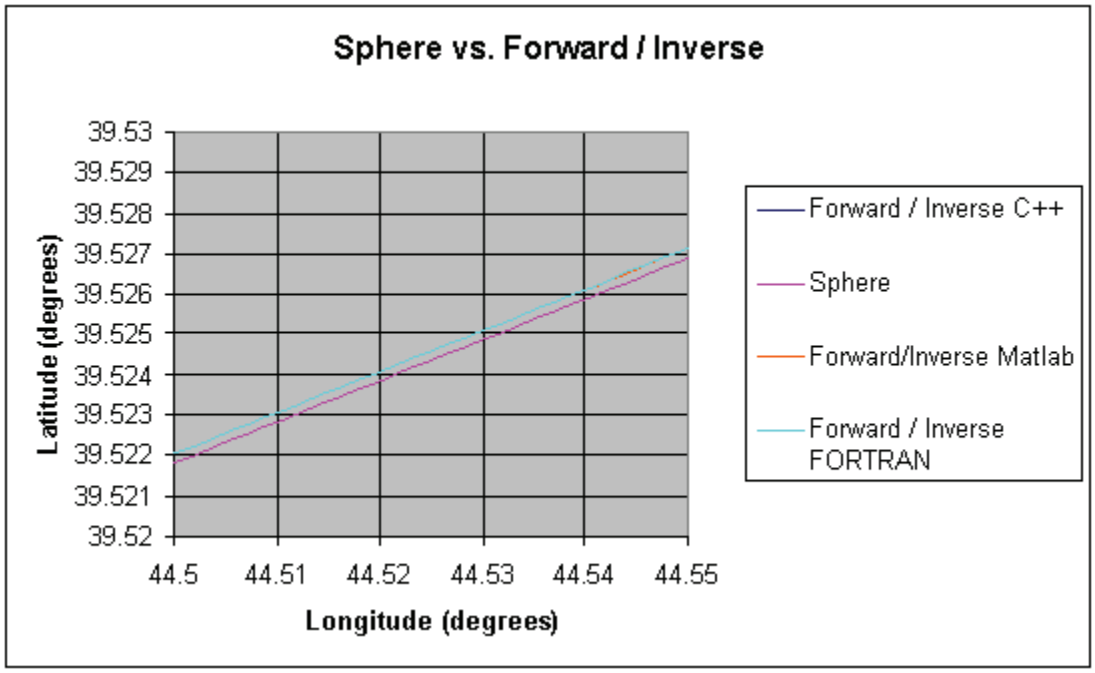

Figure 15. Difference between the spherically based calculations and the WGS84-based calculations with the horizontal axis of figure 14 multiplied by 160 and the vertical axis by 80. (A window is placed over those parts of curve where the differences are greatest.)

Some slight difference between the three OS-based calculations becomes more apparent if the horizontal axis is expanded by an additional factor of 8-1/3 and the vertical axis by an additional factor of 1-2/3. This has been done in figure 16, where a slight difference between the MATLAB and FORTRAN versions of Forward and Inverse begins to emerge in the upper right corner and center of the graph.

To report the difference between the various OS- and PS-based codes in figures 14, 15, and 16, it is not enough to give the separation distance between the curves. Instead, the distance between each snapshot location point for identical interpolation times must be made on a point-by-point basis. Thus, the differences presented in figures 17 and 18 are shown as a function of time. Therefore, a comparison between the differences of the latitudes and longitudes as predicted with the C++ and FORTRAN versions of Forward and Inverse with WGS84 parameters using equations 21 and 22 reveals that the maximum difference is very close to 1 meter. The comparison is shown in figure 17. The jagged nature of the graph suggests that the difference between the C++ and FORTRAN programs is attributable to rounding error.

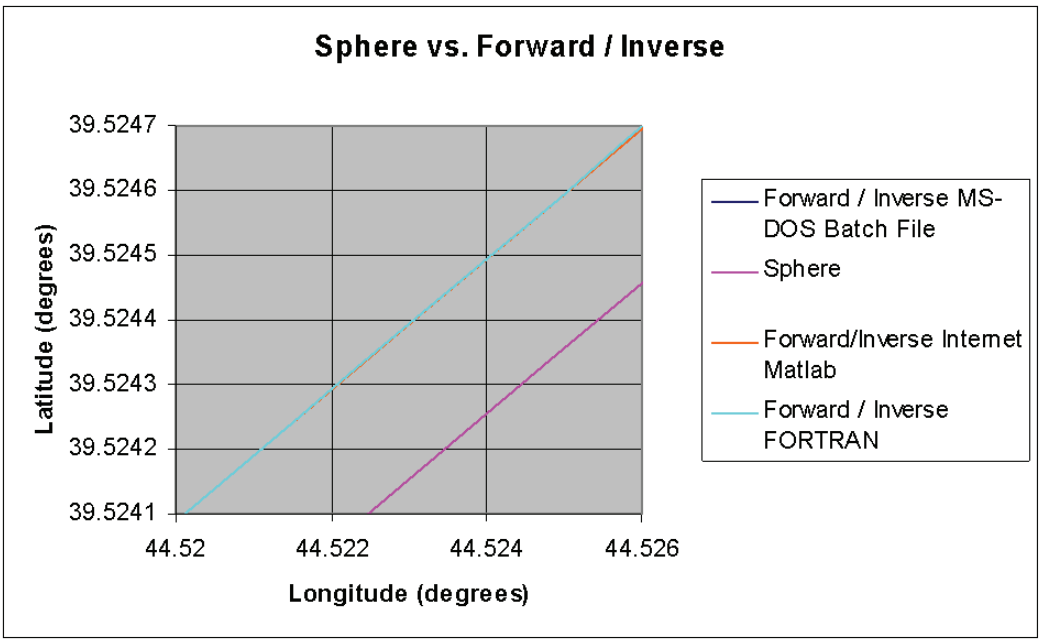

Figure 16. Slight difference between the WGS84-based MATLAB and FORTRAN begins to reveal itself after extreme magnification of figure 15.

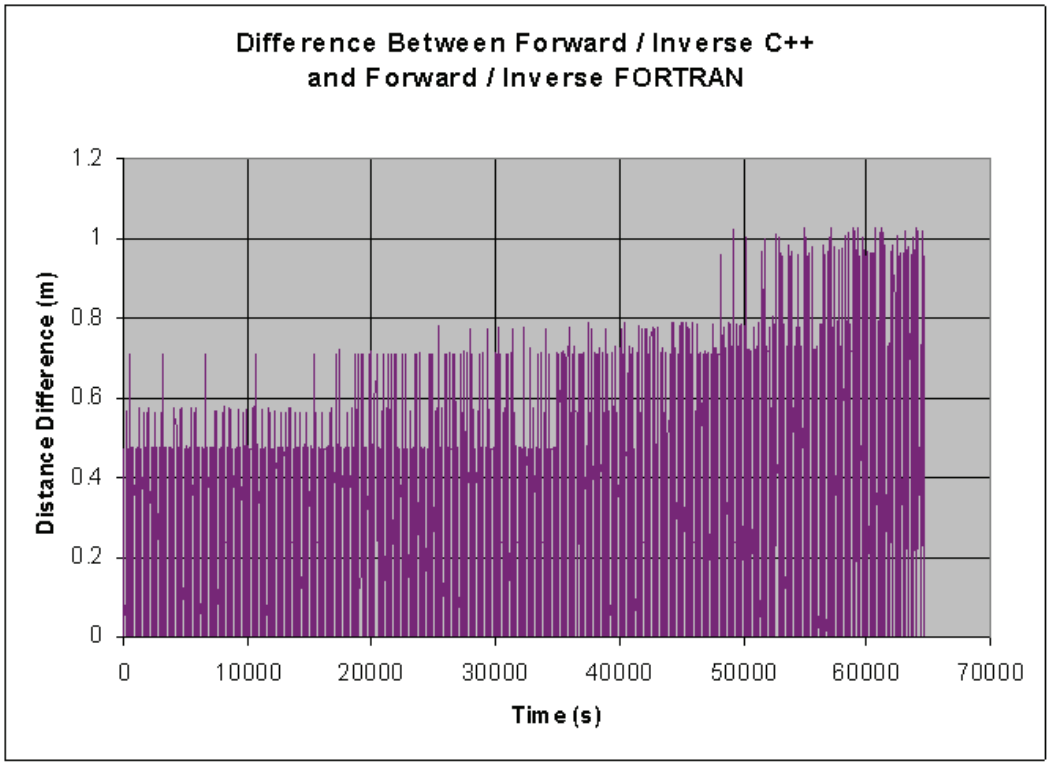

Figure 17. Slight difference between the WGS84-based MATLAB and FORTRAN. (Graph's jaggedness suggests that small difference is attributable to rounding error.)

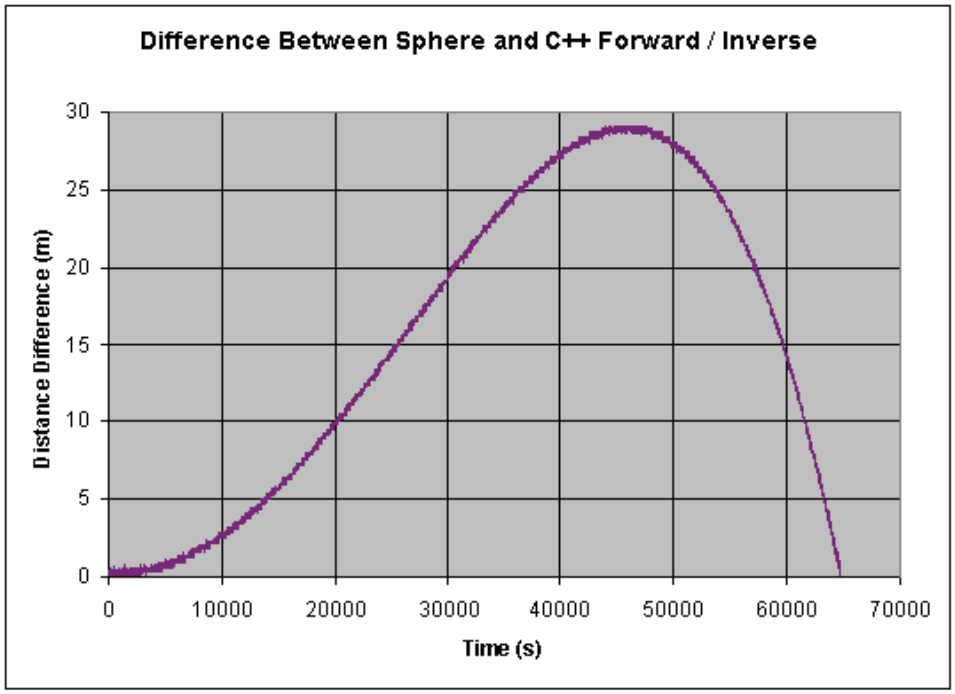

Figure 18. Difference between results of the OS C++ computation and PS FORTRAN computation. (Differences reported are time based.)

The differences between the C++ WGS84 OS and the FORTRAN PS results are much greater and are shown in figure 18. The smoothness of the overall envelope of the curve suggests that the difference between the two (calculated via equations 21 and 22) is attributable to the genuine difference in algorithm. Rounding plays some, although a greatly reduced, role in the difference of figure 18, as seen by the slight saw-tooth jaggedness superimposed on the smooth waveform.

The maximum difference between paths shown in figure 18 is less than 30 meters. The ratio of 30 meters to the earth's radius (6.3×10^6 meters) is a very, very small number—about 4.8×10^{-6}. Thus, the assumption (that the distance between paths is small compared to the earth's radius) used to derive equations 21 and 23 is justified.

The 29-meter difference between the OS and PS model is far greater than the 1-meter difference between the OS FORTRAN and C++ models, both of which employ an algorithm based on the WGS84 OS model. Because the difference between the PS- and OS-based software is large compared to the difference between the WGS84-based OS models and because our purpose is to compare the OS and PS models, our inclination is to regard all the differences between the various OS models as zero.

The distance between the two waypoints represented by figure 18 is roughly 647 km. More comparison runs were made comparing the difference between the PS- and OS-based algorithms at distances of 400 km, 200 km, 100 km, 80 km, 60 km, 40 km, 30 km, 20 km, and 10 km. The greatest distance between two points and the average distance (the distance difference totaled for all points and divided by the number of points) between the interpolated longitudes and latitudes are shown in figure 19.

The curve in figure 19 shows that the longer the path, the greater the average and maximum difference between the OS and PS algorithms. The parabolic shape of the plot indicates that the difference between the OS and PS algorithms as a fraction of the distance between the waypoints also increases with distance. This is expected since the shorter the path along the OS, the closer it comes to being the arc of a circle.

The difference should be reduced if the comparison of the plot in figure 19 is made at a higher latitude. Consider that the cross section of an OS is an ellipse; the change in radius along the arc length of an ellipse is zero at the points where the major and minor radii intersect the ellipse. A circle (the cross section of a perfect sphere) has no change in radius along the path. So the closer to the poles and equator the path, the more it should resemble a circle and the less the difference between the latitudes and longitudes predicted between the OS and PS models. Figure 20 confirms our suspicions for the very high latitude of 80 degrees: the maximum of 16 meters is somewhat over half the 600-meter path length difference of 26 meters at the latitude of 40 degrees.

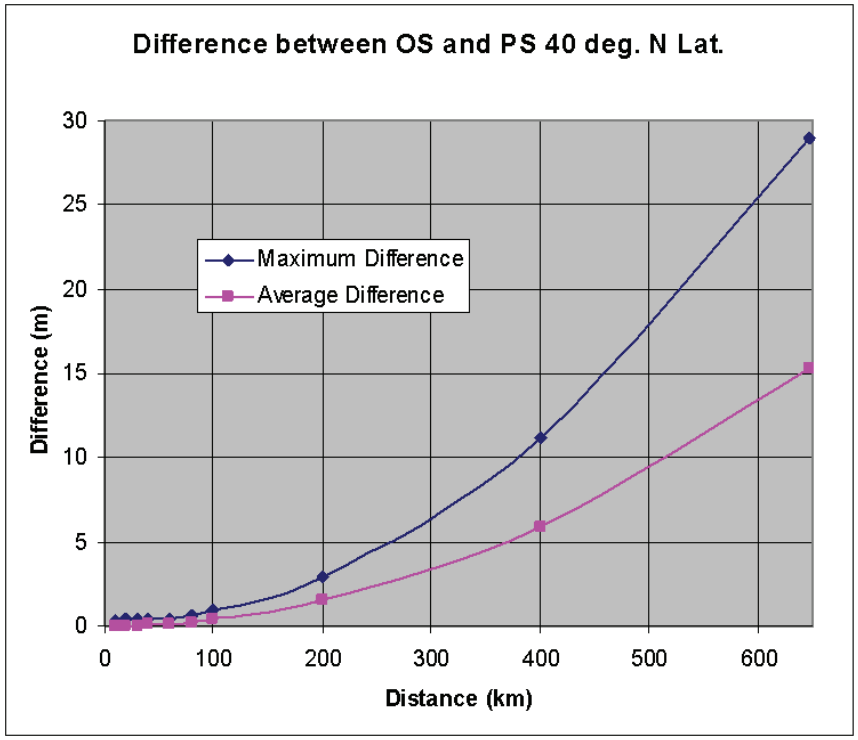

Figure 19. Maximum and average difference between the OS and PS algorithm as a function of distance at 40 degrees north latitude.

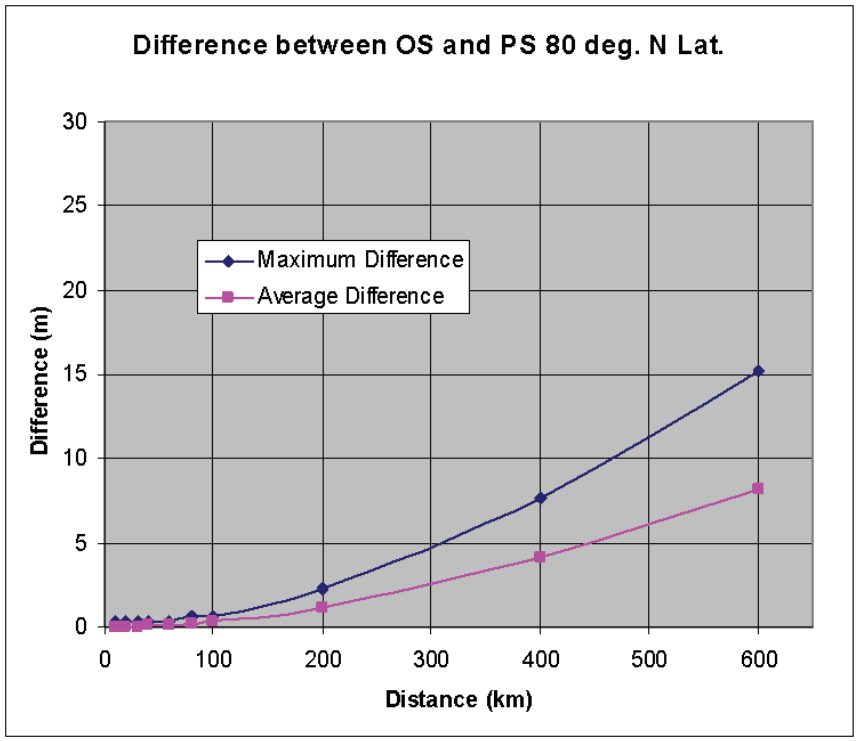

Figure 20. The maximum and average difference between the OS and PS algorithm as a function of distance at 80 degrees north latitude.

At the equator, the difference should be even smaller because unlike the pole, the difference in curvature is not near zero but exactly equal to zero for movement in the direction of the equator. As we move along a line of constant latitude, the change in radius is very nearly zero at the equator, so the region close to the equator on an OS is most like that of a PS. The graph of figure 21 shows the least amount of difference between the sets of interpolated latitudes and longitudes calculated according to the OS and PS models. For a waypoint distance of 600 km, the maximum difference is a mere 3 meters.

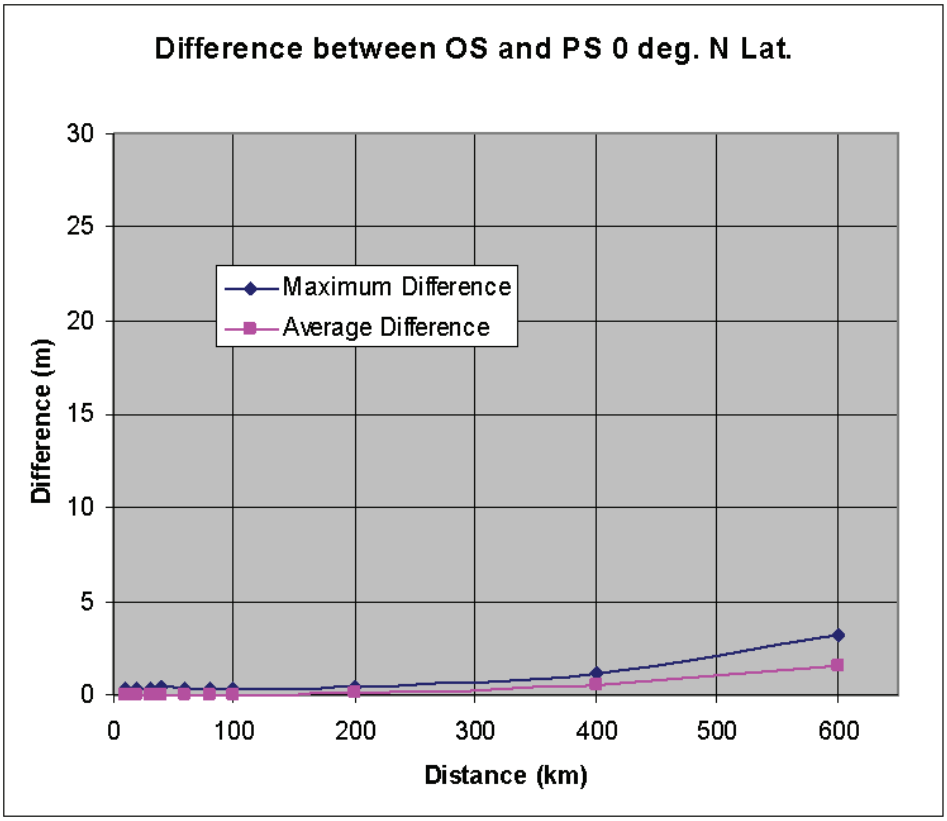

Figure 21. The maximum and average difference between the OS and PS algorithm as function of distance at 0 degrees north latitude.

4.2 Computation Time

The computation times required for the various software packages on the available platforms are shown in table 2. The times reported in this table were the times necessary to compute the data shown in the graphs in figures 15 to 19. Interpolated points (6,473) were calculated via the three platforms: a Dell desktop, a Dell laptop, and a ProGen laptop. All the platforms calculated the interpolated latitudes and longitudes using the two OS-based software packages: the C++ programs Forward and Inverse, and the FORTRAN programs Forward and Inverse. All the platforms calculated the PS-based FORTRAN program. Only the Dell desktop had the MATLAB software installed, so it was the only platform that did the computations with the downloaded OS MATLAB versions of Forward and Inverse.

Table 2. Computation times for OS- and PS-based algorithms on three platforms.

Model / Computer	Forward/Inverse (OS)			Sphere (PS) FORTRAN
	C++ Batch File	MATLAB	FORTRAN	
ProGen	49	--	3.39	1.5
Dell Desktop	6	3.7	0.94	0.61
Dell Laptop	6	--	0.91	<0.5

Times are given in seconds. The entry under "Sphere (PS) FORTRAN" and "Dell Laptop" was too short to obtain an accurate measurement.

As expected, the ProGen laptop, the oldest platform, reported the slowest speeds. The C++ batch file took 49 seconds, the FORTRAN versions of Forward and Inverse took 3.39 seconds, while the PS-based FORTRAN code needed only 1.5 seconds. The next newest machine, the Dell desktop, took 6 seconds to run the C++ batch file, 3.7 seconds to run the MATLAB scripts, 0.94 second to run the OS FORTRAN Forward and Inverse files, and only 0.61 second to run the PS FORTRAN file. The newest computer, the Dell laptop, took the least amount of time to run the programs: 6 seconds for the C++ batch file, 0.91 second for the FORTRAN Forward and Inverse files, and less than 0.5 second to run the PS-based FORTRAN program. The time to run the PS-based program (as were all the times) was measured with a stopwatch. The PS-based FORTRAN program ran so quickly, there was not quite enough time to start and then stop the stopwatch, so the upper boundary time is reported. Consistency in the ranking of the running time was also required by each program. On all the platforms, the C++ batch file took the longest, the MATLAB scripts were the next quickest, then the OS FORTRAN version of Forward and Inverse, while the FORTRAN version of the PS-based program ran the fastest.

One point to be noted in the time comparison is that three calculating modes are in use. The modes are a batch file for executable Windows/DOS programs; interpreted files inside a MATLAB environment run as an additional operation and execution of a single program. It may be possible to accelerate the run times for the C++ batch files if we perform the same type of recompilation as was done for FORTRAN source code. The MATLAB codes may also be combined and could be exported as a separate executable program. These may be useful for ensuing investigations.

The time required to write/modify the programs to calculate the interpolated points is presented in table 3. The MATLAB modification of Forward and Inverse took the least amount of time (1/4 of 1 day) because the scripts for Forward and Inverse were imported from the internet. The manipulating and modifying scripts were also MATLAB scripts, so the MATLAB version was the easiest to assemble. The C++ batch file had to use executable versions of Forward and Inverse, which made it necessary to write a comprehensive, albeit simple, manipulative batch file. The batch file was the next longest to write at 1 day. The PS FORTRAN code had to be written from scratch. Even though no scripts were available to make any of the computations, the PS model was simple enough that it required only 2 days to write and de-bug the program. Using the FORTRAN listings of Forward and Inverse required the most time: 5 days. This was because attempts were made to modify the programs so that they would run efficiently. Extraneous output lines were

"commented out" of the listing; the "do" loop controlling the interpolation calculations was placed so that the entered data were used only once. Forward and Inverse were modified and compiled to make one large program, but the process that likely took most of the time was the addition of comment lines. This was done so that the coders could keep track of the modifications made. Because Forward and Inverse were extensive, extensive comment lines had to be included in the revised program. They were so extensive that it took a week to make the necessary alterations.

Table 3. The time necessary to write/modify codes so that they were able to calculate needed interpolation points.

	Forward/Inverse (OS)			Sphere (PS) FORTRAN
	C++ Batch File	MATLAB	FORTRAN	
Coding Time (days)	1	0.25	5	2

4.3 Special Problem

The attempts to calculate the interpolated points for the special problem (two points on the equator separated by more than 179.2329 degrees but less than 180 degrees) are summarized in table 4. All the software packages calculated a path that was on the equator. Only the C++ batch file froze and refused to perform any computations. The remaining packages calculated interpolated points along the equator. Further investigation of the FORTRAN source codes and the vintage of the C++ code indicated a potential source for the problem with the C++ code. Comments in the FORTRAN code showed several improvements, and "divide-by-zero" traps were introduced in versions dating back to 1998. One of the reasons cited for including these traps was to avoid program failure at near-antipodal conditions. The C++ code was available in 1995 and may not have included the antipodal trap.

Table 4. Summary of the results for special problem outlined in section 3.2

	Forward/Inverse (OS)			Sphere (PS) FORTRAN
	C++ Batch File	MATLAB	FORTRAN	
Results	No computation	On equator	On equator	On equator

5. Conclusions

Conclusions fall into four categories. The categories are computation time for the OS versus the PS models, accuracy for the PS model compared to the WGS84-based OS model, time needed to modify and write the various OS- and PS-based codes, and how well the OS- and PS-based programs managed the special problem outlined in section 3.2. The conclusions are presented for each category.

5.1 Computation Time

For any platform and for any software package used, the time needed to calculate the interpolated latitudes and longitudes for the WGS84-based models took anywhere from 1.5 times as long (as on the Dell laptop) to 30 times as long (as on the ProGen laptop) as for the PS model. The software package that came closest to matching the PS FORTRAN code was the FORTRAN listings of Forward and Inverse provided by the NGS (*11*) modified and re-compiled into one code in such a way as to minimize the computation time.

Although the OS model only took 1.5 times longer to do the calculations on the Dell laptop and the Dell desktop computers using the modified NGS codes than with the PS FORTRAN code, we believe that the more realistic comparison between these codes is the result from the ProGen laptop computer (see table 2). Attempting to accurately measure times of less than 1 second is very difficult with a stopwatch; making the measurements on the newer machines is somewhat less reliable because the programs took less than 1 second to run. The results on the older ProGen laptop were easier to measure because the programs took longer than 1 second to run on this machine. Assuming that the results for processing speed can be linearly extracted to the newer, faster machines and that the inconsistency of measurement increases with the machine speed, it is not unreasonable to realize that the true ratio of the OS processing time to the PS processing time could well be closer to 2.25.

5.2 Latitude and Longitude Accuracy

The greatest difference between any of the OS models and the PS model in interpolating a set of latitudes and longitudes was 29 meters. This was for the 647-km separation of waypoints at about 40 degrees north latitude (see figure 19). One way to characterize these numbers is to note that the ratio of the disagreement distance to the waypoint separation distance is less than 0.0044%—very small indeed.

Another way is to see it in terms of EM propagation time: we will be concerned with a time resolution of approximately 1 second for the connectivity software. Yet an EM wave can travel 29 meters in less than 97 nanoseconds—instantaneous compared to 1 second. For practical applications, this is ignorable.

Another way to characterize the difference is in the additional free space attenuation between stations that are 29 meters farther separated than we assume. If we have a sending and receiving platform separated by 10 km of free space, then the free space distance attenuation will increase by 0.025 dB, which is very small compared to most phenomena that degrade communication signals. Typical of these is insertion loss, which is typically 1 dB (*5*).

5.3 Code Writing and Code Modification Times

Table 3 outlines the time needed to write or adapt the codes needed to do the comparison for this study. The shortest time needed was to use the MATLAB scripts: only one quarter of a day. The

next shortest time was for use of the C++ executable programs Forward and Inverse and to write a few additional programs to modify the format of the input and output files so as to put them in a useful format that Forward could read: one day. The next was to write the FORTRAN program test.exe based on the PS model: two days. By far, the longest time required was to consolidate the FORTRAN listings for Forward and Inverse: five days. The long time was needed to examine and understand the programs and to make non-harmful modifications to minimize the computation time. This was also because of the far more complex math needed to calculate the WGS84 OS model than with the PS model. Note that the FORTRAN program based on the PS model was the only code that was not based on the WGS84 OS model. For the OS models, the tendency was that the more time spent on producing the code, the faster it ran.

5.4 The Special Problem of Section 3.2

As table 4 shows, none of the codes calculated the problem correctly. The MATLAB and FORTRAN OS codes gave the same mistaken output as the PS FORTRAN code, while the C++ code simply did not run. It would be possible to have the Deployment Module (when faced with two waypoints that are on the equator and separated by 180 to 179.2329 degrees of latitude) fix a phantom waypoint between the two equatorial waypoints (as outlined in section 3.2) and then use any of the processes outlined to interpolate latitude and longitude points from the first waypoint to the phantom waypoint, then to the second waypoint.

---

6. Recommendations

We recommend that the PS model be used to interpolate the latitudes and longitudes for the snap-shot points. The primary reason is that the computation time is much shorter. By the data in table 2, it takes anywhere from 1.8 to 32 times longer to generate the same number of interpolated points, depending on the process used to make the calculations with the WGS84 OS model. (For the reasons outlined in section 5.1, we believe that the lower boundary is closer to 2.25 times.) This study calculated the interpolation points for one platform between two waypoints. Performing a genuine link study will require doing many waypoints for many platforms so that the impact of the longer time will be magnified. The Deployment Module is the first of six modules to be written, so we should strive to make the first module as computationally efficient as possible to minimize the impact of inefficiencies that appear in any of the remaining modules.

The strongest argument for using the WGS84 OS-based software is that it is more accurate than the PS model. The discourse of section 5.2 shows that for waypoints 647 km apart, the PS model will be accurate enough. Yet doing past link budgets for moving platforms, waypoint separations are likely to be less than 100 km apart (5). The comparison of the PS to the WGS84 OS for 100 km shows a maximum disagreement of only 1 meter (see figure 19) for the most inaccurate latitude surveyed in the present work. Lower and higher latitudes agree even more (see figures 20 and 21).

The DTED (from which the Deployment Module will be drawing its elevation data) Level 1 specifies a linear error of less than 30 meters and a horizontal accuracy of less than 50 meters at least 90% of the time (*1*). Level 2 is even more restrictive than level 1, requiring a linear error of less than 18 meters, and a horizontal error of less than 23 meters at least 90% of the time (*1*). For waypoint distances of 100 km or less, the PS has surpassed the Level 1 and Level 2 DTED specifications.

Other arguments in favor of using the WGS84 OS models point to the coding time saved in using software that has already been written. Table 3 indicates that the time needed to apply the codes Forward and Inverse for MATLAB (0.25 day), and C++ (1 day) was indeed less than the time needed to write the PS FORTRAN code (2 days). However, these two codes required 6 and 10 times longer to run on the Dell desktop (see table 2), more than outweighing the coding time advantage. Modifying the FORTRAN OS program to cause it to run faster increased the coding time dramatically (5 days). The WGS84-based FORTRAN code still took about twice as long as the PS FORTRAN code to do the calculation. Further modification of the WGS84-based code to improve the computation time would require even more coding time, which has already exceeded the coding time for the PS FORTRAN code. Investing more coding to improve calculation time for the WGS84-based computer programs is a losing proposition.

Finally, if any difficulty should arise in the development of the next modules, which requires detailed knowledge of the interpolation process used in the Deployment Module, there is no one in the group who has a detailed understanding of how to calculate great circle snapshot points using the OS model. In contrast, a team member wrote the PS-based FORTRAN code for this study and has used the PS model in past link budget analyses (*5*).

7. References

1. Web site: http://www.mission-planning.com/DTED_Part2.htm June 28, 2007.
2. Web site: http://en.wikipedia.org/wiki/ED50 June 29, 2007.
3. Web site: http://en.wikipedia.org/wiki/Spherical_Earth (June 7, 2007).
4. Web site: http://www.ngs.noaa.gov/PC_PROD/Inv_Fwd/ June 29, 2007.
5. Bevec, A.; Still, G. W. *WIN-T Link Budget Analysis for the Ground to Unmanned Aerial Vehicles (U)*; Unpublished Document; U.S. Army Research Laboratory: Aberdeen Proving Ground, MD, September 2005 (SECRET).
6. Web site: http://en.wikipedia.org/wiki/WGS84 (June 7, 2007)
7. Web site: http://en.wikipedia.org/wiki/Figure_of_the_Earth (June 7, 2007)
8. Web site: http://mathworld.wolfram.com/OblateSpheroid.html (June 7, 2007)
9. Web site: http://mathforum.org/dr.math/faq/formulas/faq. ellipse.circumference.html (June 11, 2007)
10. Vincenty, T. Direct and Inverse Soloutions of Geodesics on the Ellipsoid with Application of Nested Equations, *Survey Review* **April 1975**, *XXII* (176), 88-93.
11. Web site: http://www.ngs.noaa.gov/TOOLS/Inv_Fwd/Inv_ Fwd.html (June 11, 2007)
12. Web site: http://en.wikipedia.org/wiki/Great_circle, September 11, 2007.
13. Web site: http://en.wikipedia.org/wiki/Rhumb_line, September 11, 2007.
14. U.S. Army Toporaphic Engineering Center. *REUSER's Manual*, page 2, REUSERS.DOC, August 12, 1996.
15. Web site: http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=10821&ref=rssfeed&id=mostDownloadedFiles, June 18, 2007.
16. Web site: http://www.mathworks.com/MATLABcentral/fileexchange/loadFile.do?objectId=5379&objectType=file, June 18, 2007.
17. Web site: http://www.ngs.noaa.gov/TOOLS/Inv_Fwd/Inv_ Fwd.html June 18, 2007.
18. Leithold, L. *The Calculus with Analytic Geometry Third Edition*, pp 433-434, 736; Harper & Row Publishers: New York, 1976.

19. Sears, F. W.; Zemansky, M. W.; Young, H. D. *Fifth Edition University Physics, Pages 204-205*; Addison-Wesley Publishing Company: Reading. MA, March 1977.

20. Web site: http://en.wikipedia.org/wiki/Pythagorean_ theorem, December 10, 2007.

21. Web site: http://williams.best.vwh.net/ellipsoid/ node1.html, December 10, 2007.

22. Web site: http://www.analyzemath.com/EllipseEq/ EllipseEq.html, December 10, 2007.

Appendix A. Listing of the Contents of the File Way.dat.

Lat.(d m s)	Long.(d m s)	Altitude(m)	Time(s)
039 52 36.40	048 47 54.24	00000.00000000000	000000.0000000000
039 09 09.72	041 18 37.08	00000.00000000000	064717.8518900000

INTENTIONALLY LEFT BLANK

Appendix B. Listing of the Contents of the File Way1.crd.

```
COORD_FMT       DMS
LON_FMT         360
ACCURACY        .01_SECOND
ELLIPSOID       WE
LATITUDE_FROM    39 52 36.40 N
LONGITUDE_FROM   48 47 54.24 E
LATITUDE_TO      39  9  9.72 N
LONGITUDE_TO     41 18 37.08 E
CONVERT
;
```

INTENTIONALLY LEFT BLANK

Appendix C. Listing of the Contents of the File Way1.out.

```
;   COORD_FMT       DMS
;   LON_FMT         360
;   ACCURACY        .01_SECOND
;   ELLIPSOID       WE


ELLIPSOID : WGS 84
INPUT
================================================================================
==========
FROM LATITUDE       FROM LONGITUDE       TO LATITUDE         TO LONGITUDE
================================================================================
==========
39 52 36.40 N       048 47 54.24 E       39 09 9.72 N        041 18 37.08 E


OUTPUT
================================================================================
==========
FORWARD AZIMUTH     BACK AZIMUTH         DISTANCE
================================================================================
==========
265 16 41.91        080 30 34.76         648742
```

INTENTIONALLY LEFT BLANK

Appendix D. Partial Listing of the Contents of the File Snap1.crd.

```
COORD_FMT  DMS
LON_FMT    360
ACCURACY   .01_SECOND
ELLIPSOID  WE
LATITUDE   39 52 36.40 N
LONGITUDE   48 47 54.24 E
AZIMUTH    265 16 41.91
DISTANCE         0
CONVERT
;
COORD_FMT  DMS
LON_FMT    360
ACCURACY   .01_SECOND
ELLIPSOID  WE
LATITUDE   39 52 36.40 N
LONGITUDE   48 47 54.24 E
AZIMUTH    265 16 41.91
DISTANCE         0
CONVERT
;
COORD_FMT  DMS
LON_FMT    360
ACCURACY   .01_SECOND
ELLIPSOID  WE
LATITUDE   39 52 36.40 N
LONGITUDE   48 47 54.24 E
AZIMUTH    265 16 41.91
DISTANCE         0
CONVERT
;
COORD_FMT  DMS
LON_FMT    360
ACCURACY   .01_SECOND
ELLIPSOID  WE
LATITUDE   39 52 36.40 N
LONGITUDE   48 47 54.24 E
AZIMUTH    265 16 41.91
DISTANCE         0
CONVERT
;
.
.
```

INTENTIONALLY LEFT BLANK

Appendix E. Partial Listing of the Contents of the File Snap1.out.

```
;    COORD_FMT  DMS
;    LON_FMT    360
;    ACCURACY   .01_SECOND
;    ELLIPSOID  WE
;    DISTANCE        0


ELLIPSOID : WGS 84
================================================================================
==========
STATION       LATITUDE        LONGITUDE       AZIMUTH        DISTANCE
================================================================================
==========
FROM STATION   39 52 36.40 N     048 47 54.24 E    265 16 41.91     0
TO STATION     39 52 36.40 N     048 47 54.24 E    085 16 41.91

;    ;
;    COORD_FMT  DMS
;    LON_FMT    360
;    ACCURACY   .01_SECOND
;    ELLIPSOID  WE
;    DISTANCE        0


ELLIPSOID : WGS 84
================================================================================
==========
STATION       LATITUDE        LONGITUDE       AZIMUTH        DISTANCE
================================================================================
==========
FROM STATION   39 52 36.40 N     048 47 54.24 E    265 16 41.91     0
TO STATION     39 52 36.40 N     048 47 54.24 E    085 16 41.91

;    ;
.
.
.
.
.
.
.
.
```

INTENTIONALLY LEFT BLANK

Appendix F. Partial Listing of the Contents of the File Snap2.out.

Lat.(d m s)	Long.(d m s)	Altitude(m)	Time(s)
39 9 9.72	41 18 37.08	.00000000000	.0000000000
39 9 9.72	41 18 37.08	.00000000000	10.0000000000
39 9 9.72	41 18 37.08	.00000000000	20.0000000000
39 9 9.72	41 18 37.08	.00000000000	30.0000000000
39 9 9.72	41 18 37.08	.00000000000	40.0000000000
39 9 9.72	41 18 37.08	.00000000000	50.0000000000
39 9 9.72	41 18 37.08	.00000000000	60.0000000000
39 9 9.72	41 18 37.08	.00000000000	70.0000000000
39 9 9.73	41 18 37.12	.00000000000	80.0000000000
39 9 9.73	41 18 37.12	.00000000000	90.0000000000
39 9 9.73	41 18 37.12	.00000000000	100.0000000000
39 9 9.73	41 18 37.12	.00000000000	110.0000000000
39 9 9.73	41 18 37.12	.00000000000	120.0000000000
39 9 9.73	41 18 37.16	.00000000000	130.0000000000
39 9 9.73	41 18 37.16	.00000000000	140.0000000000
39 9 9.73	41 18 37.16	.00000000000	150.0000000000
39 9 9.74	41 18 37.20	.00000000000	160.0000000000
39 9 9.74	41 18 37.20	.00000000000	170.0000000000
39 9 9.74	41 18 37.20	.00000000000	180.0000000000
39 9 9.74	41 18 37.24	.00000000000	190.0000000000
39 9 9.74	41 18 37.24	.00000000000	200.0000000000
39 9 9.74	41 18 37.24	.00000000000	210.0000000000
39 9 9.75	41 18 37.29	.00000000000	220.0000000000
39 9 9.75	41 18 37.29	.00000000000	230.0000000000
39 9 9.75	41 18 37.33	.00000000000	240.0000000000
39 9 9.75	41 18 37.33	.00000000000	250.0000000000
39 9 9.76	41 18 37.37	.00000000000	260.0000000000
39 9 9.76	41 18 37.37	.00000000000	270.0000000000
39 9 9.76	41 18 37.41	.00000000000	280.0000000000
39 9 9.76	41 18 37.41	.00000000000	290.0000000000
39 9 9.77	41 18 37.45	.00000000000	300.0000000000
39 9 9.77	41 18 37.49	.00000000000	310.0000000000
39 9 9.77	41 18 37.49	.00000000000	320.0000000000
39 9 9.78	41 18 37.53	.00000000000	330.0000000000
39 9 9.78	41 18 37.57	.00000000000	340.0000000000
39 9 9.78	41 18 37.57	.00000000000	350.0000000000
39 9 9.79	41 18 37.61	.00000000000	360.0000000000
39 9 9.79	41 18 37.66	.00000000000	370.0000000000
39 9 9.79	41 18 37.66	.00000000000	380.0000000000
39 9 9.80	41 18 37.70	.00000000000	390.0000000000

.

INTENTIONALLY LEFT BLANK

Acronyms

2-D	Two dimensional
3-D	Three dimensional
APG	Aberdeen Proving Ground
ARL	Army Research Laboratory
ASCII	American Standard Code for Information Interchange
DTED	Digital Terrain Elevation Data
EM	electromagnetic
EW	electromagnetic warfare
FORTRAN	Formula Translator
IEPD	Information & Electronic Protection Division
MATLAB	Matrix Laboratory
NGS	National Geodetic Survey
OS	oblate spheroid
PS	perfect sphere
SLAD	Survivability/Lethality Analysis Directorate
S/N	signal to noise (ratio)
WGS84	World Geodetic System 1984
TEC	Topographic Engineering Center

NO. OF COPIES ORGANIZATION

1 (PDF ONLY) DEFENSE TECHNICAL INFORMATION CTR DTIC OCA 8725 JOHN J KINGMAN RD STE 0944 FORT BELVOIR VA 22060-6218

1 US ARMY RSRCH DEV & ENGRG CMD SYSTEMS OF SYSTEMS INTEGRATION AMSRD SS T 6000 6TH ST STE 100 FORT BELVOIR VA 22060-5608

1 DIRECTOR US ARMY RESEARCH LAB IMNE ALC IMS 2800 POWDER MILL RD ADELPHI MD 20783-1197

1 DIRECTOR US ARMY RESEARCH LAB AMSRD ARL CI OK TL 2800 POWDER MILL RD ADELPHI MD 20783-1197

2 DIRECTOR US ARMY RESEARCH LAB AMSRD ARL CS OK T 2800 POWDER MILL RD ADELPHI MD 20783-1197

1 US ARMY RSCH LAB ATTN AMSRD ARL SL EG L ANDERSON WSMR NM 88002-5502

2 US ARMY RSCH LAB ATTN AMSRD ARL SL E R FLORES AMSRD ARL SL EM T MCDONALD WSMR NM 88002-5513

5 US ARMY RSCH LAB ATTN AMSRD ARL SL EW P BOTHNER FORT MONMOUTH NJ 07703-5602

1 US ARMY RDECOM/TARDEC ATTN S CAMPBELL MS 263 6501 E 11 MILE ROAD WARREN MI 48397-5000

NO. OF COPIES ORGANIZATION

ABERDEEN PROVING GROUND

1 DIRECTOR US ARMY RSCH LABORATORY ATTN AMSRD ARL CI OK (TECH LIB) BLDG 4600

6 DIRECTOR US ARMY RSCH LABORATORY ATTN AMSRD ARL SL D BAYLOR J BEILFUSS J FRANZ J FEENEY M STARKS P TANENBAUM BLDG 328

1 DIRECTOR US ARMY RSCH LABORATORY ATTN AMSRD ARL SL BE L ROACH BLDG 328

6 DIRECTOR US ARMY RSCH LABORATORY ATTN AMSRD ARL SL EM J NEALON G W STILL (5 CYS) BLDG 309A