

2005

## Investigating the Use of Wide Area Agmented System (WAAS) as a Navigational Tool for Coast Guard and Civilian Maritime Use

Christopher M. Armstrong  
*University of Rhode Island*

Follow this and additional works at: <https://digitalcommons.uri.edu/theses>

Terms of Use

All rights reserved under copyright.

---

### Recommended Citation

Armstrong, Christopher M., "Investigating the Use of Wide Area Agmented System (WAAS) as a Navigational Tool for Coast Guard and Civilian Maritime Use" (2005). *Open Access Master's Theses*. Paper 1394.

<https://digitalcommons.uri.edu/theses/1394>

This Thesis is brought to you by the University of Rhode Island. It has been accepted for inclusion in Open Access Master's Theses by an authorized administrator of DigitalCommons@URI. For more information, please contact [digitalcommons-group@uri.edu](mailto:digitalcommons-group@uri.edu). For permission to reuse copyrighted content, contact the author directly.

INVESTIGATING THE USE OF WIDE AREA AGMENTATED SYSTEM (WAAS)  
AS A NAVIGATIONAL TOOL FOR COAST GUARD AND CIVILIAN  
MARITIME USE

BY

CHRISTOPHER M. ARMSTRONG

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE  
REQUIREMENTS FOR THE DEGREE OF  
MASTER OF SCIENCE  
IN  
ELECTRICAL ENGINEERING

UNIVERSITY OF RHODE ISLAND

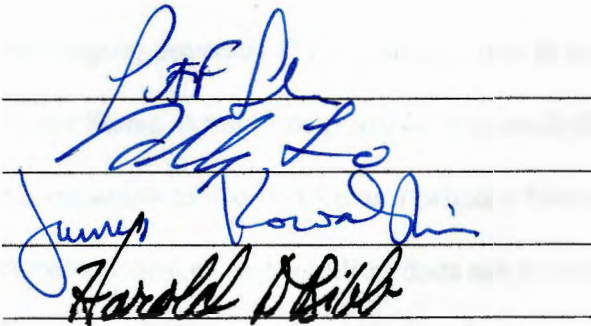
2005

MASTER OF SCIENCE THESIS  
OF  
CHRISTOPHER M. ARMSTRONG

APPROVED:

Thesis Committee:

Major Professor

The image shows four horizontal lines representing signature lines. The first line has a signature that appears to be 'P. F. D.'. The second line has a signature that appears to be 'R. L. L.'. The third line has a signature that appears to be 'James Kowalich'. The fourth line has a signature that appears to be 'Harold D. Hill'.

DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND

2005

## ABSTRACT

Since 2003, the Federal Aviation Administration (FAA) has employed WAAS for precision approaches to airports and navigational use for aircraft over the continental United States, adjacent ocean regions, and parts of Alaska. This was due in part to GPS/DGPS not meeting the FAA's strict guidelines for accuracy, integrity, and availability. Currently, the FAA rates WAAS for 250ft and above the surface of the earth; this 250-foot barrier ensures 100% coverage over the United States from two existing INMARSAT satellites. Because of this height restriction, the Coast Guard has not accepted WAAS as an individual stand-alone source of navigation for military and civilian use.

In 2005, the FAA plans on launching additional geostationary satellites to increase system redundancy and provide overlapping coverage [2]. By placing one to three more satellites due south of the United States, it might be possible to provide the coverage needed for the maritime community to use WAAS as a primary form of navigation at ground level. The current system of two satellites does not provide double or even single coverage in parts of the United States at ground level due to line of sight issues. By adding more satellites, the 250-foot barrier might be able to be brought down and double coverage for the United States might be possible in all navigable areas.

This paper reports on a project to develop software tools to predict coverage of WAAS satellites (both existing and future) at user selectable locations in the continental United States. To account for topographical features, the Digital Terrain

Elevation Data (DTED) Level 1 database with a spacing of 3 arc seconds (or 100 meter resolution) is incorporated into the tool. The results of the predictions are compared to actual field measurements made during 2004 as part of a DGPS/WAAS Accuracy and Availability Study conducted by John J. McMullen Associates in support of the U.S. Coast Guard Academy

## ACKNOWLEDGEMENTS

The author would first like to thank his major area professor, Dr. Peter Swaszek for all the time and effort put into making sure I succeeded in graduating on time and for helping produce the best possible paper. The author would also like to thank all of those who contributed to the data collection efforts and otherwise provided information used in this paper: Gregory Johnson, Ruslan Shalaev and Christian Oates, from JJMA.

<b>APPENDIX A – MAIN PROGRAM.....</b>	<b>42</b>
<b>APPENDIX B – SATEQUATIONS.....</b>	<b>71</b>
<b>APPENDIX C – ANGLETHET .....</b>	<b>72</b>
<b>APPENDIX D – ELEVATION ANGLE.....</b>	<b>73</b>
<b>APPENDIX E – RANGERINGS .....</b>	<b>74</b>
<b>APPENDIX F – TRYCATCH .....</b>	<b>75</b>
<b>APPENDIX G – DATAGRIDFIX.....</b>	<b>76</b>
<b>BIBLIOGRPAPHY .....</b>	<b>77</b>

## LIST OF FIGURES

Figure	Page
2.1 Global Positioning Satellite System Constellation .....	3
2.2 NDGPS Coverage Throughout Continental United States .....	5
2.3 Basic Differential GPS Concept Diagram.....	6
2.4 Wide Area Augmentation System Continental United States Coverage .....	8
2.5 GPS Signals Transmitted to WAAS Reference Stations Throughout CONUS .....	9
2.6 WAAS Reference Stations Transmitting to WAAS Master Stations .....	10
2.7 WAAS Master Stations relay to Uplink Stations to INMARSAT Satellites .....	10
3.1 A Planar View of the User and Satellite .....	13
3.2 Solving for the Elevation Angle.....	14
3.3 Comparison of Elevation Angles for Smooth Earth and DTED Points.....	15
3.4 Satellite Elevation Angle Versus Angle Separating the User and Satellite: User at Sea Level and User at 4.4 km Altitude.....	16
3.5 The Line of Sight Issue – Satellite Visibility Suffers due to Uneven Terrain Characteristics .....	17
3.6 DTED Elevation Data Over a 1 by 1 Degree Grid for the San Francisco Area – Mesh Plot.....	18
3.7 DTED Elevation Data Over a 1 by 1 Degree Grid for the San Francisco Area – Topographical Plot.....	19
3.8 Graphical User Interface (GUI) .....	21
4.1 Salt Lake City Utah Region .....	24
4.2 Blackout Areas to the Pacific Satellite, Salt Lake City Area.....	24
4.3 Blackout Areas to the Atlantic Satellite, Salt Lake City Area.....	25
4.4 White Areas Show Where Neither Satellite is Visible, Salt Lake City area.....	26



Figure	Page
4.5 125W Satellite for Salt Lake City Region, 3pts of Blackout .....	27
4.6 107W Satellite for Salt Lake City Region, 0 pts of Blackout .....	27
4.7 085W Satellite for Salt Lake City Region, 5 pts of Blackout .....	28
4.8 San Francisco Harbor Area .....	29
4.9 Elevation Angle to the Pacific Satellite; White Areas Indicate No Line of Sight.....	30
4.10 Elevation Angle to the Atlantic Satellite; White Areas Indicate No Line of Sight.....	30
4.11 Elevation Angle to the 125W Satellite.....	31
4.12 Elevation Angle to the 107W Satellite.....	32
4.13 Elevation Angle to the 085W Satellite.....	32
4.14 Actual Data from San Francisco Harbor .....	33
4.15 Actual Data from San Francisco Harbor, Zoomed.....	34
4.16 Antenna Location on Research Vessel.....	35
4.17 Sault Ste. Marie, Michigan.....	36
4.18 Elevation Angle to the Atlantic Satellite; No Loss of Visibility Due to Terrain.....	37
4.19 Elevation Angle to the 0125W Satellite; No Loss of Visibility Due to Terrain.....	37
4.20 Elevation Angle to the 107W Satellite; No Loss of Visibility Due to Terrain.....	38
4.21 Elevation Angle to the 085W Satellite; No Loss of Visibility Due to Terrain.....	38

## 1 INTRODUCTION

On July 10<sup>th</sup> 2003, the Wide Area Augmented System (WAAS) was put into commission by the Federal Aviation Administration (FAA) for precision approaches to airports and navigational use for aircraft over the continental United States (CONUS), adjacent ocean regions, and parts of Alaska. This was done to meet the demands of high accuracy while in flight that Differential Global Positioning System (DGPS) / Global Positioning System (GPS) could not provide. WAAS was initially able to produce a guaranteed position for the aircraft within 3-7 meters and was used from 350 feet and above the surface of the Earth. In late 2003, the height above ground limit was reduced to 250 feet. This 250-foot barrier ensures 100% coverage over the United States for all aircraft from two International Maritime Satellites (INMARSAT) [3,4].

WAAS is currently undergoing upgrades to make it more reliable. Since there are only two satellites being used right now, if one was to go down (fail) or be taken off line for any reason, one of the coasts would be affected depending upon which satellite. In 2005 and 2006, the FAA plans on launching three more geostationary satellites over the United States to provide more redundancy and overlapping coverage [1].

By placing from one to three additional satellites almost due south of the United States, it might be possible to provide the coverage needed to the maritime community to use WAAS as a primary form of navigation (in other words, extend the height limit to almost ground level). The current system of two satellites does not

provide double or even single coverage in some parts of the United States due to line-of-sight issues (blockage due to terrain, buildings, etc.). By adding more satellites, the 250-foot barrier might be able to be brought down and double coverage for the United States might be possible in all navigable areas.

### 1.1 Objective of Study

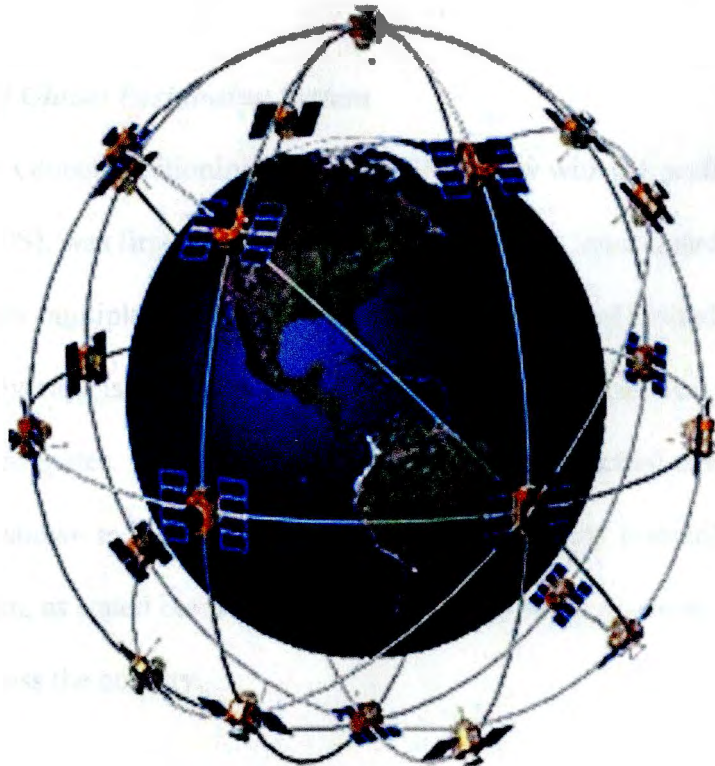
The primary focus of this project was to develop a MATLAB® Graphical User Interface (GUI) tool that would use Digital Terrain Elevation Data (DTED) Level 1 (a high level quality database of elevation data) to provide a precise theoretical model of WAAS coverage for the continental United States taking into account the effects of terrain (naked earth), both altitude itself and line of sight issues, and testing this model at selected locations within CONUS. All horizontal datum from DTED will be referenced to the World Geodetic System (WGS84) and all vertical datum will be referenced to Australian Height Datum (AHD), or Mean Sea Level outside of Australia.

## 2 BACKGROUND

In today's navigational community, there are many ways of navigating around the Continental United States. You can use Differential Global Positioning System, Loran C, or Wide Area Augmentation System. Of these, the Differential Global Positioning System and Wide Area Augmentation System are augmentations of the basic Global Positioning System. These two systems compliment GPS by providing more accurate results.

### 2.1 Global Positioning Satellite System

Global Positioning System (GPS) launched its first of many satellites back in 1978. It is a global system that relies on a full constellation of 24 satellites with a handful of backup satellites. (Figure 2.1) The full constellation was achieved in 1994.



*Figure 2.1 - Global Positioning Satellite System Constellation*

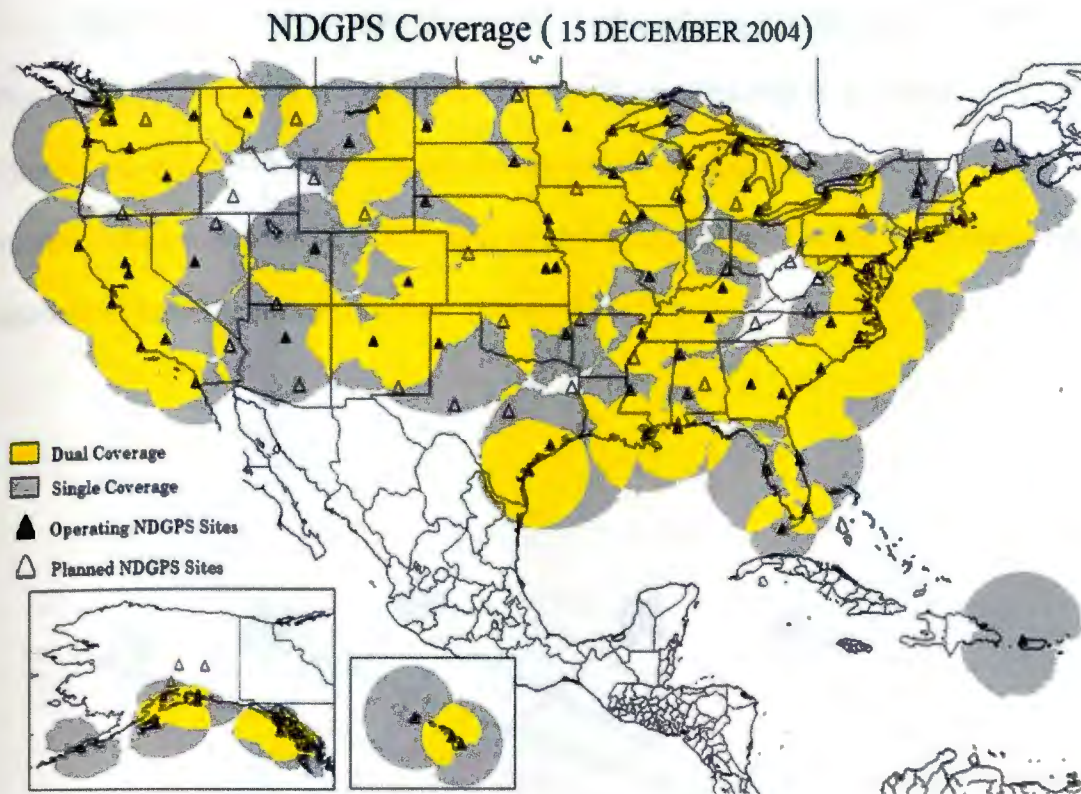
Each satellite is built to last approximately 10 years. Due to the short life span, satellites for GPS are constantly being launched into orbit. The satellites each weigh around 2,000 pounds and are approximately 17 feet across with the solar panels extended. They transmit messages on the L1, L2, and L5 bands (1575.42, 1227.60, and 1176.45 MHz respectively) with a transmitter power of only 50 watts or less.[6]

## 2.2 Augmented Global Positioning System

The two systems mentioned above can be considered augmented Global Positioning Systems. They take the primary GPS signal, and provide additional information (corrections) that provides for better accuracy. The two systems may have a similar goal in mind, but their reasons for existence are totally different.

### ***2.2.1 Differential Global Positioning System***

Differential Global Positioning System (DGPS), now with the prefix Nationwide (NDGPS), was first put into place in 1999. It is a Coast Guard affiliated system, and provides multiple coverage throughout the Continental United States (CONUS). This coverage is double along the east and west coasts as well as most of middle of the United States. Single coverage areas do exist, but actual coverage is better than what is shown in Figure 2.2. The maritime community primarily uses this system. This system, as stated before, runs off of GPS and relies on more than 84 ground stations across the country.



*Figure 2.2 - NDGPS Coverage Throughout Continental United States*

Differential GPS relies on the concept that the errors in the position at one location (a ground reference station), are similar to those for all locations within a given (local) area, i.e. a nearby ship off the coast. By recording GPS measurements at a point with known coordinates, these errors can be quantified and measured. Then corrections can be transmitted to and applied to the other locations. By applying corrections in real-time, the accuracy of GPS for instantaneous positioning is reduced from 100 meters to typically 3-5m (and even sub-meter with commercial grade GPS receivers) 95% of the time.

The first major component in this intricate system is the Reference Station. The Reference Station is at a known location with its role being to generate corrections, which will later be applied at the unknown stations/ships at sea. The reference station consists of a GPS receiver, Antenna, and transmitter (modulator and antenna) to send the corrections out. (Figure 2.3) [5]

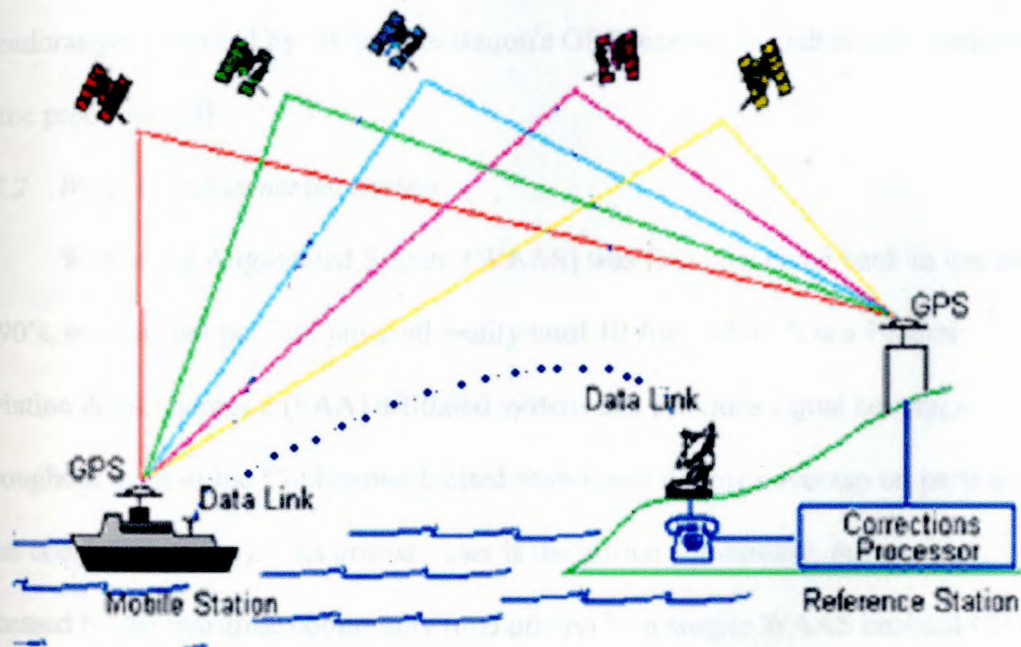


Figure 2.3– Basic Differential GPS Concept Diagram.

Second major component of the Differential GPS signal is the Data Link. This is the communications channel that the corrections travel across from the Reference Station to the Mobile Station. This Data Link must transmit the correction data fast enough to limit the age of the correction. If the correction were too old, it

might not provide an accurate correction to the Mobile Station. Once it is transmitted, the signal must also be reliable in terms of probability of good reception. [5]

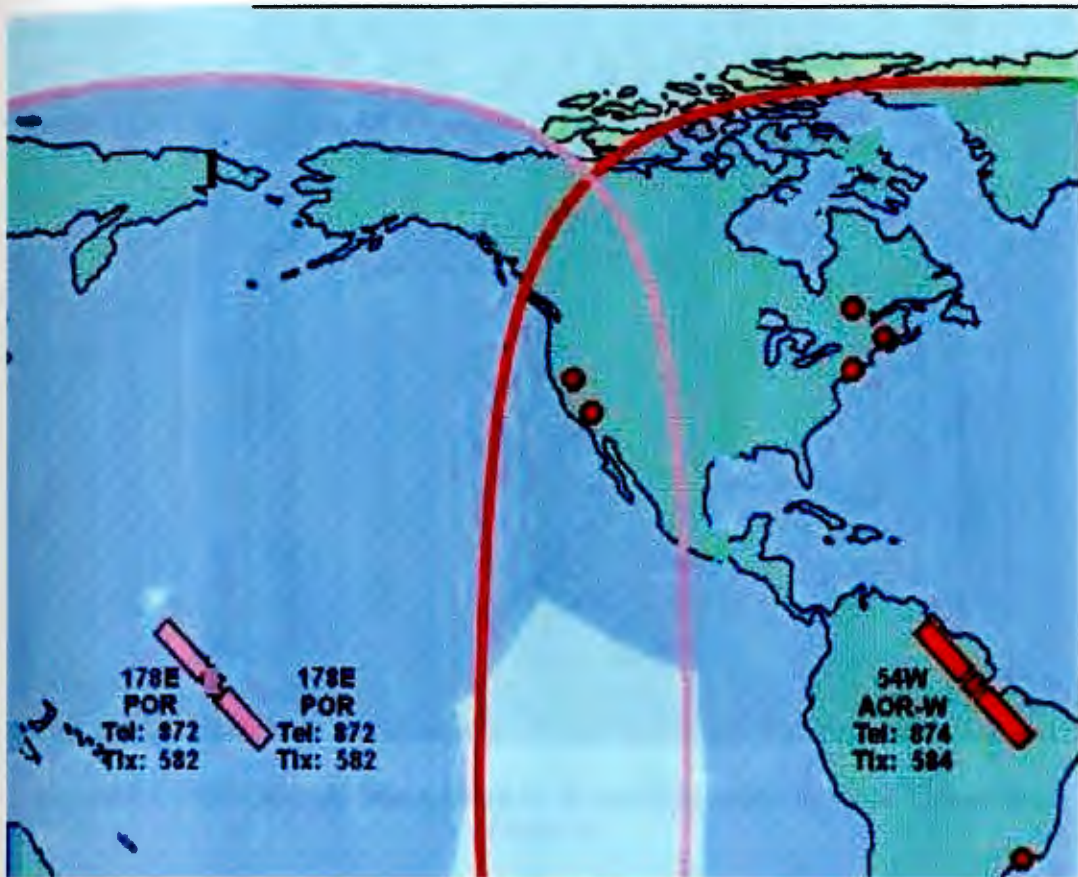
The third and final component of the Differential GPS correction path is the Mobile Station or a ship off the coast. The Mobile Station consists of a GPS receiver and antenna, Data link receiver and demodulator, and DGPS processor (computer). The pseudorange and range rate corrections are received via the data link equipment, and sent to the DGPS processor. The corrections are then applied to the individual pseudoranges observed by the mobile station's GPS receiver, which is also connected to the processor. [5]

### 2.2.2 *Wide Area Augmented System*

Wide Area Augmented System (WAAS) was first conceived back in the early 1990's, but was not put into physical reality until 10 July 2003. It is a Federal Aviation Administration (FAA) affiliated system that provides signal coverage throughout most of the Continental United States, and double coverage on parts of the west coast (Figure 2.4). Its primary user is the airline community, but can be accessed by the maritime community (and others) by a simple WAAS enabled GPS unit.

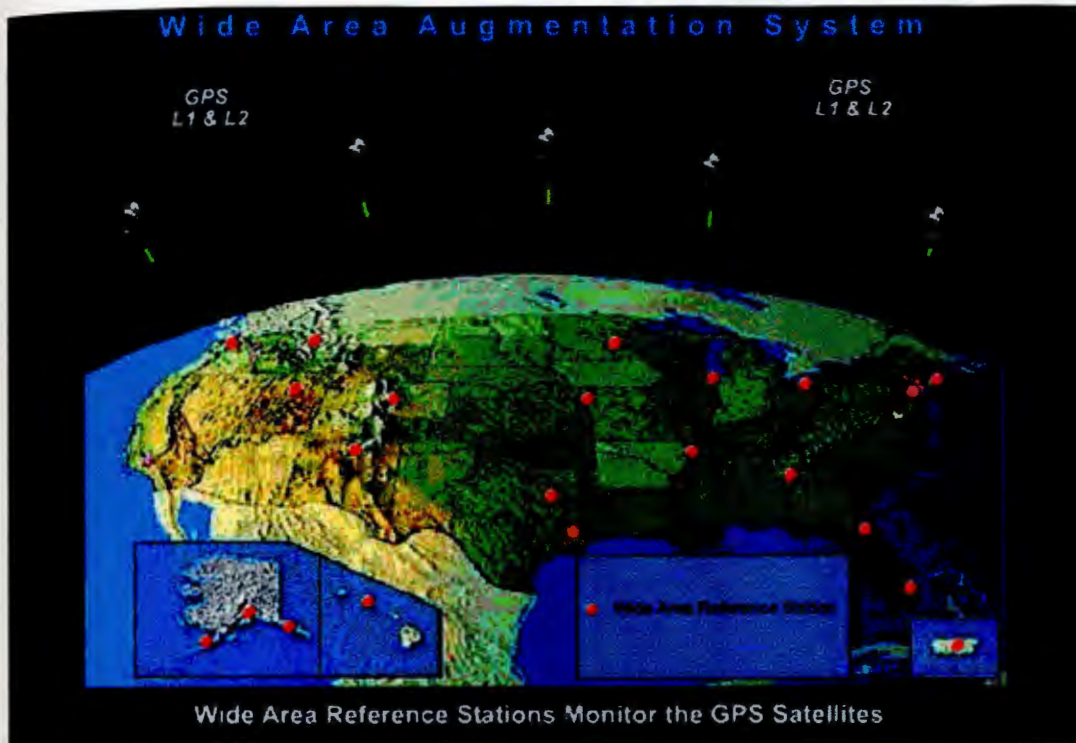
The WAAS system consists of two-leased International Maritime Satellites (INMARSAT) to provide the data link, approximately 25 ground reference stations, two master stations on either coast, and a WAAS enabled receiver.





*Figure 2.4– Wide Area Augmentation System Continental United States Coverage*

The system is similar to DGPS in that the signal originates from GPS on the L1, L2, and L5 band. The GPS signal is received at many widely spaced Wide Area Reference Stations (WRS) sites throughout CONUS. These WAAS reference stations are precisely surveyed so that any errors in the received GPS signals can be detected (Figure 2.5 ) [2]



*Figure 2.5– GPS Signals Transmitted to WAAS Reference Stations Throughout CONUS*

GPS info is now collected and forwarded from the WAAS Reference Stations to the WAAS Master Station (WMS). These Master Stations are located on each coast. At the Master Stations, a WAAS augmentation message is generated. This message contains information for GPS receivers to remove errors from the GPS signal. The messages allow for significant increase in location accuracy and reliability (Figure 2.6). [2]

The augmentation messages are sent from the WAAS Master Stations to three uplink stations, two on the west coast and one on the east coast. The messages are transmitted from these uplink stations to the two current INMARSAT geostationary communication satellites located over the equator at 178 degrees E and 054 degrees W (Figure 2.7). [2]

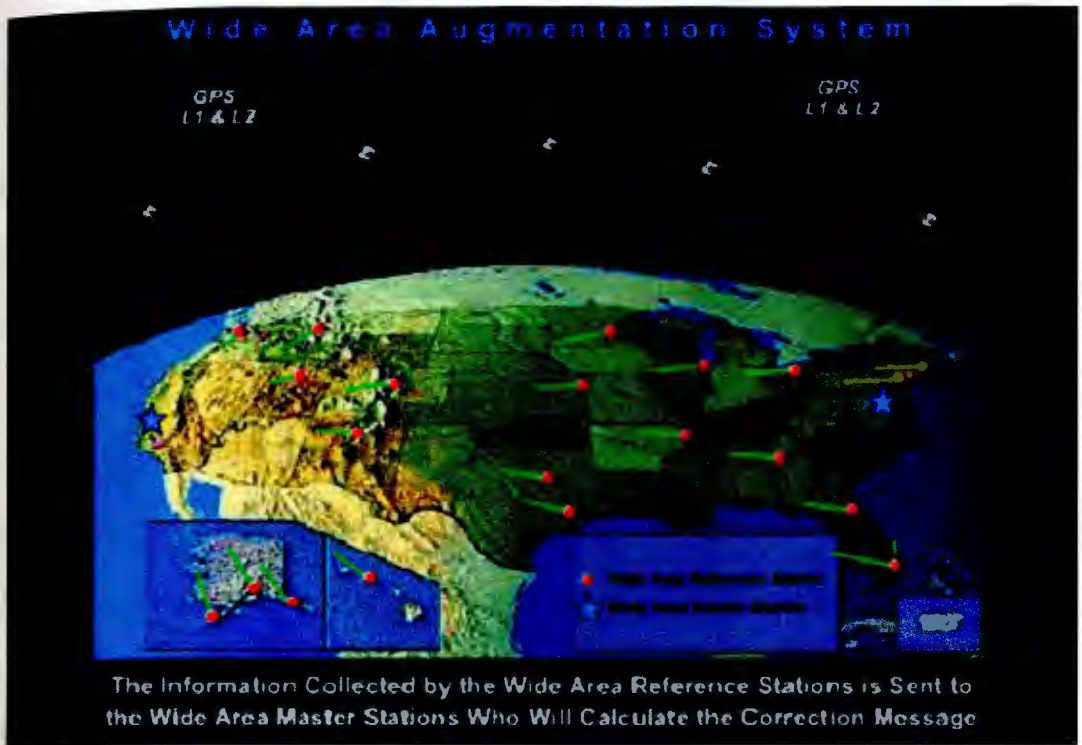


Figure 2.6– WAAS Reference Stations Transmitting to WAAS Master Stations

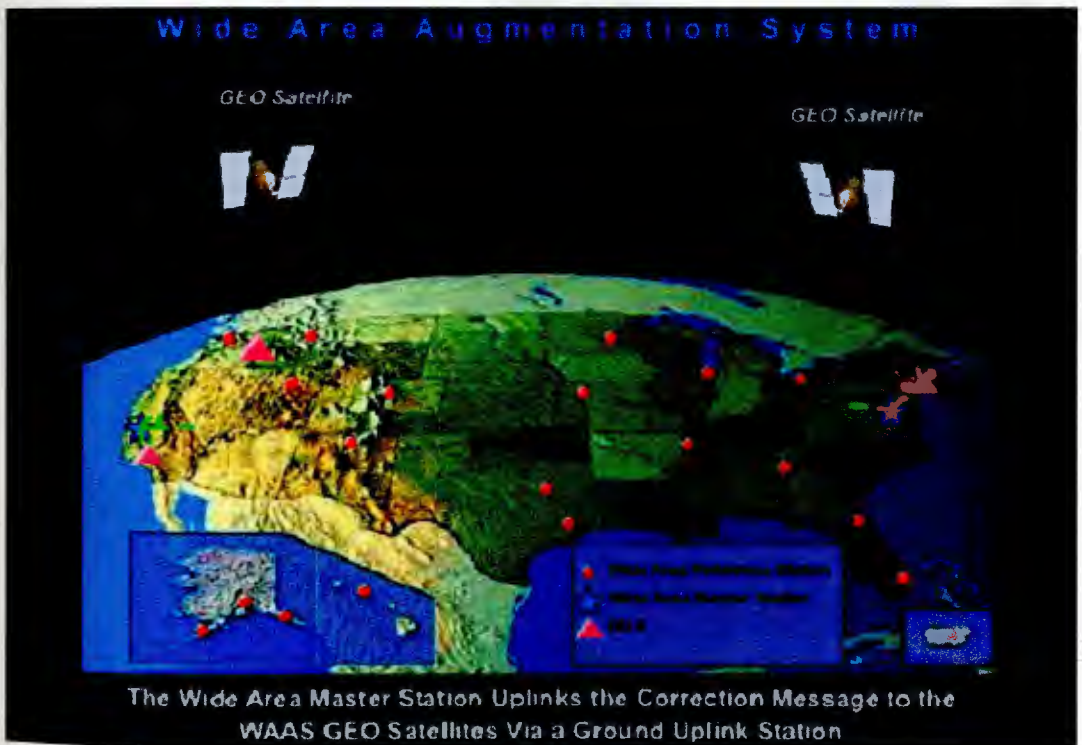


Figure 2.7– WAAS Master Stations relay to Uplink Stations to INMARSAT Satellites

The INMARSAT WAAS satellites now broadcast the correction message on a GPS-like signal to aircraft below. The GPS/WAAS receiver on board the aircraft processes the WAAS augmentation messages as part of estimating position. The GPS-like signal from the navigation transponder can be used by the receiver as an additional source for calculation of the user's position. [2]

The FAA started using WAAS because the current system at the time, GPS, did not meet its navigation requirements for accuracy, integrity, and availability. WAAS corrects the GPS signal for ionospheric disturbances, timing, and satellite orbit errors. The FAA is also able to monitor the health of the satellites through the messages being broadcast.

The original accuracy for GPS, which was subject to accuracy degradation under the government-imposed Selective Availability (SA) program, was roughly 100 meters. Today GPS is around 10 meters with Selective Availability turned off. WAAS accuracy on the other hand is less than 3 meters typically.

### 3 METHODOLOGY

The primary focus of this project is to develop a MATLAB® Graphical User Interface (GUI) tool that would use Digital Terrain Elevation Data (DTED) Level 1 to provide a precise theoretical model of WAAS coverage for the continental United States taking into account the effects of terrain, both altitude itself and line of sight issues, and testing this model at selected locations within CONUS. The line of sight issues would be looked at with the two current satellites and three proposed satellites to be launched in the near future.

#### 3.1 Trigonometry

We begin the discussion with a description of the trigonometry of the situation. Consider a plane intersecting the Earth so that it passes through the three points of the Earth's center, the user's position, and the satellite's position. The relative positions of these objects are shown in the plane diagram of Figure 1. In this diagram, the circle represents the Earth at sea level with radius  $r$ , the altitude to the satellite above sea level is marked  $h_s$ , the dotted line tangent to the Earth at the user's position shows the horizon, and the relative locations of the user and the satellite is described by the angle  $\alpha$ . Using spherical trigonometry (Napier's rules), this angle can be computed from the latitudes and longitudes of the user and satellite in which  $\Delta_{lat}$  and  $\Delta_{lon}$  are the differences in the user's and satellite's latitudes and longitudes, respectively. This is shown in Equation 1

$$\alpha = \cos^{-1}(\cos(\Delta_{lat})\cos(\Delta_{lon}))$$

Equation 1

The value of interest here is the elevation angle from the user to the satellite, marked as  $\beta$  in Figure 3.1. In other words, if  $\beta > 0$ , then the satellite is visible to the user. Using some simple trigonometry (extend the ray passing through the user, as shown in Figure 3.2, to form a right triangle with the satellite) this can be computed as in Equation 2.

$$\beta = \tan^{-1} \left( \frac{(r + h_s) \cos \alpha - r}{(r + h_s) \sin \alpha} \right)$$

Equation 2

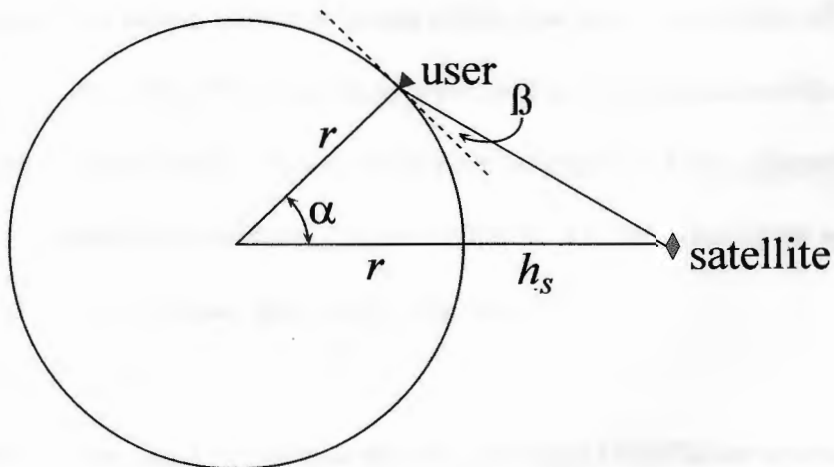


Figure 3.1 – A Planar View of the User and Satellite.

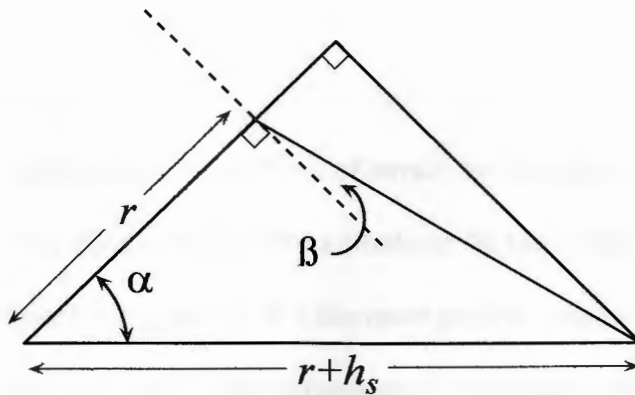


Figure 3.2 – Solving for the Elevation Angle.

### 3.2 Digital Terrain Elevation Data

For this study, Digital Terrain Elevation Data (DTED) is being used. DTED is a uniform matrix of terrain elevation values which provides basic heights of land above sea level (to within 50 ft of accuracy) for systems and applications that require terrain elevation, slope, and/or surface roughness information. It was primarily developed as a tool for the military, but has moved its way into the private sector by changing the level of accuracy available to the user. [7]

Level 5 being used by the military has a spacing of 0.0370 arc seconds or every 1-meter. The civilian version of DTED, Level 0, has 30 arc seconds of spacing or every 1000 meters. For this project DTED level 1 will be used and has 3 arc seconds of spacing or approximately 100 meters between points This is roughly 1.1 million points on a 1 by 1 degree grid of longitude and latitude.

### 3.2.1 Altitude Effects

Our first consideration of the effects of terrain on elevation to the satellite involves examining the effects of non-zero altitude of the user. Specifically, consider the situation represented in Figure 3.3. For the more general case of the user at some other altitude than sea level, say  $h_u$ , the expression for elevation angle in Equation 2 becomes Equation 3, and it is clear from Figure 3.3 that this angle decreases with increasing altitude.

$$\beta_a = \tan^{-1} \left( \frac{(r + h_s) \cos \alpha - (r + h_u)}{(r + h_s) \sin \alpha} \right)$$

Equation 3

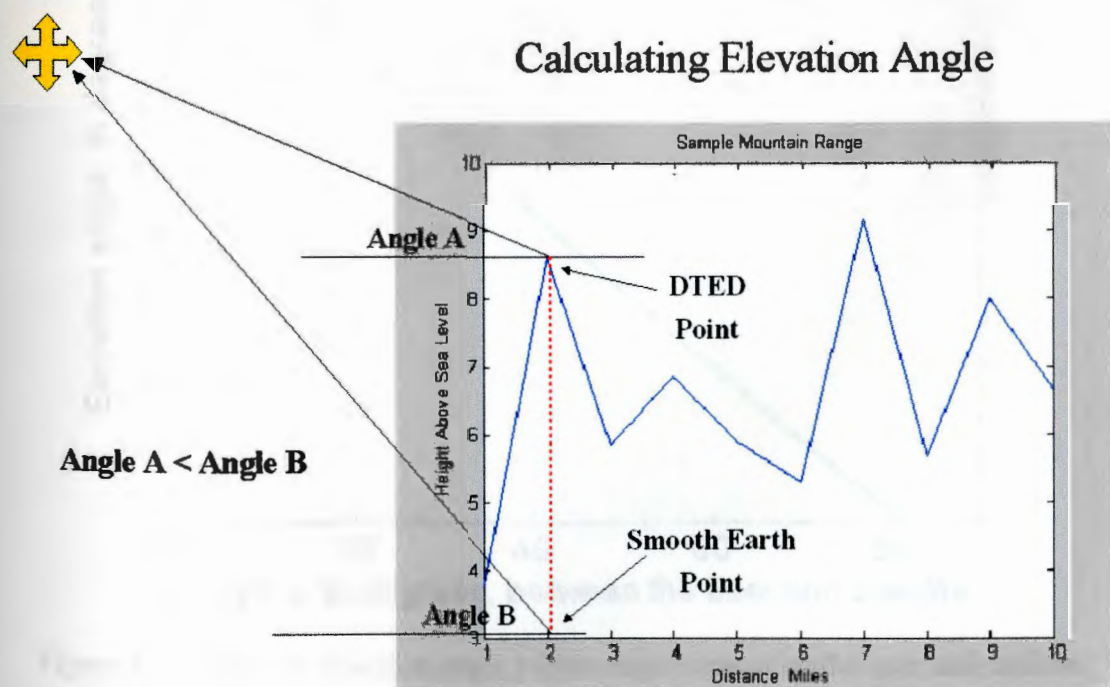


Figure 3.3 - Comparison of Elevation Angles for Smooth Earth and DTED Points



### 3.2.2 Line of Sight Effects

Since height above sea level doesn't play a significant role, we must look now to see what causes blackouts throughout CONUS. The significant effect of terrain on WAAS satellite visibility is due to shadowing in which higher terrain potentially blocks the user's line of sight to the satellite. A simple sketch of this appears in Figure 3.5. Fortunately, MATLAB contains a tool within its mapping toolbox to compute this directly from a digital terrain map. To study this effect for WAAS availability, we need an accurate model of the local terrain about the user's position.

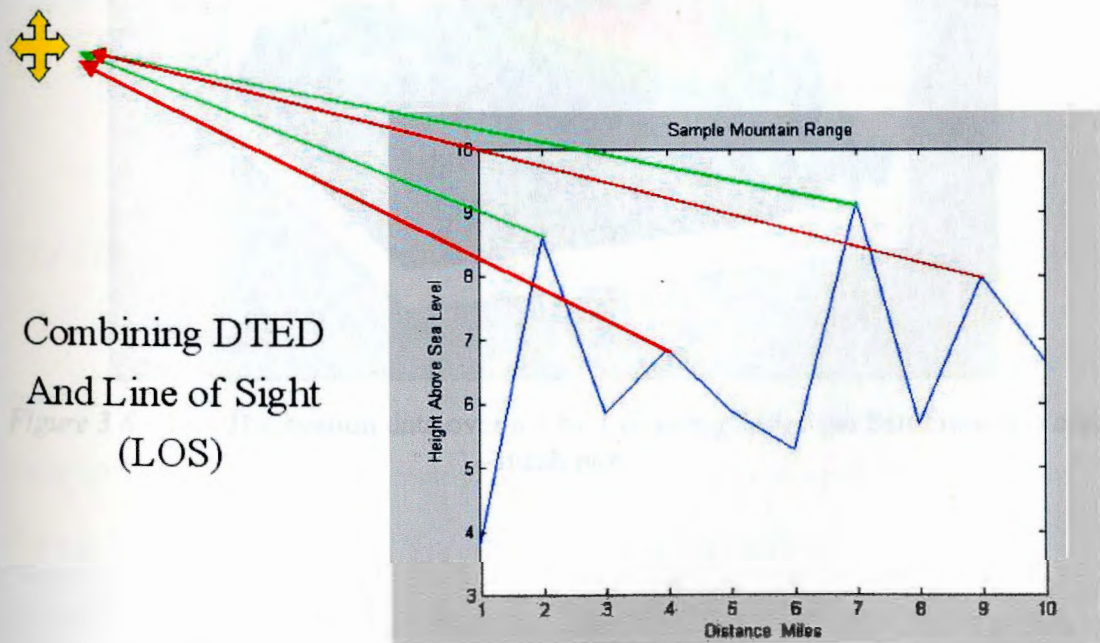


Figure 3.5 – The line of sight issue – satellite visibility suffers due to uneven terrain characteristics.

### 3.2.3 Digital Terrain Elevation Data and MATLAB

The Digital Terrain Elevation Data (DTED) database can be used to provide the required local terrain information. A 1 degree by 1 degree sample of DTED Level 1 for the San Francisco area is shown in Figures 3.6 and 3.7; Figure 3.6 shows this data as a MATLAB mesh plot, Figure 3.7 is a more typical topographical plot (logistically, DTED only contains data for square regions containing land masses – to create this map, zeros were appended for the missing ocean data).

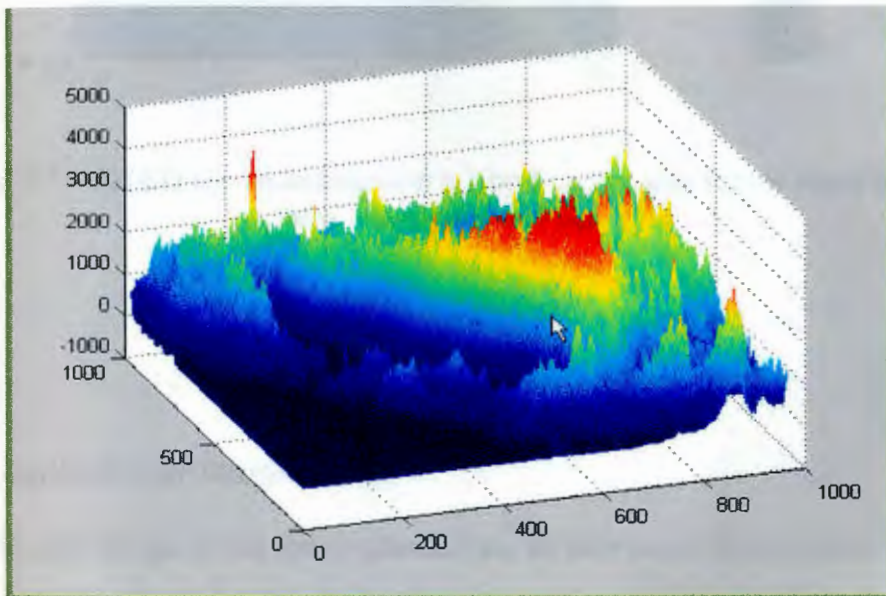
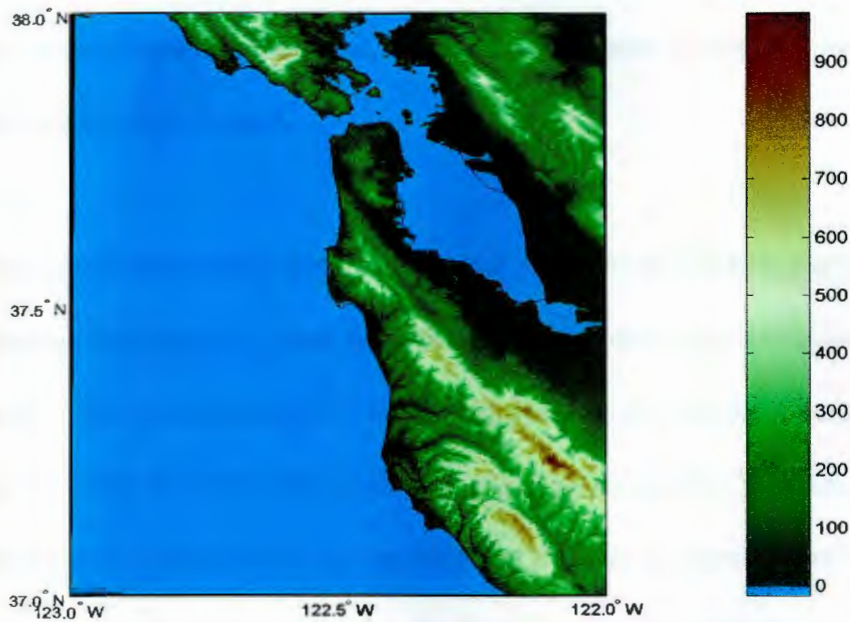


Figure 3.6 – DTED elevation data over a 1 by 1 degree grid for the San Francisco area – mesh plot.



*Figure 3.7 – DTED elevation data over a 1 by 1 degree grid for the San Francisco area – topographical plot.*

### 3.3 MATLAB

#### 3.3.1 Graphical User Interface (GUI)

To ease the use of the above calculations, an easy to use environment was needed. The GUI, created in MATLAB with an example shown in Figure 3.8, was designed to display as much information as possible without overwhelming the user. The top left hand side of the GUI contains a Mercator projection of the world. This picture initially shows the entire globe, but can be zoomed in for locating the user's position. On this projection, the user's position is displayed as well as the satellites being looked at and their range rings (using a spherical Earth model). On the upper right hand side of the GUI, computed data is displayed. The elevation angle to the user

is displayed across the top starting with the Pacific and Atlantic satellites, followed by three other possible satellite locations. Below this, the direct visible distance from the user to the satellite is displayed.

Changeable parameters are located in the center of the GUI on the right. Here you can change the range ring sizes from full horizon to minimum elevation angle of 5 to 25 degrees. When you change the range ring, you also change the minimum elevation angle that is desired when calculating blackout regions. You can also change the height of the satellite and the height of the user's antenna vertically (above the Earth's surface). In the center is also displayed the user's height above sea level in feet or meters.

On the bottom of the GUI reside entry boxes for the positions of three additional satellites, the user's position, and check boxes to enable the display of range rings for the Pacific and Atlantic satellites. The bottom left of the GUI contains a window to display a 1 by 1 degree grid of elevation angles. This picture can be displayed with line of sight taken into account or not.

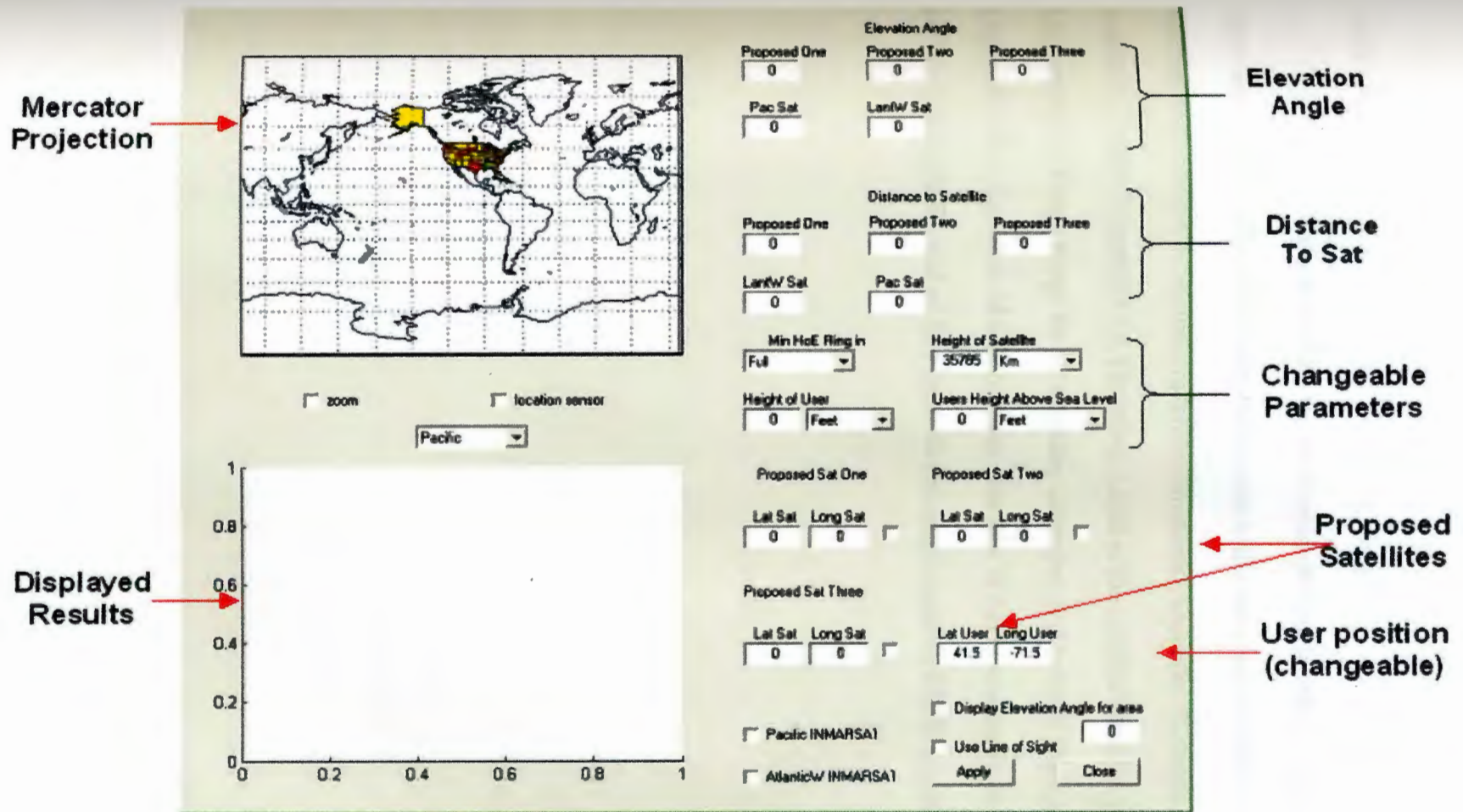


Figure 3.8 – Graphical User Interface (GUI)

### 3.3.2 Coding

Along with the main coding for the Graphical User Interface (Appendix A), there are a number of other M-Files being used in the calculations behind the scenes. M-Files were created to calculate Satellite Equations (Appendix B), angle  $\alpha$  from equation 1 above (Appendix C), Elevation Angle to the satellite from the user (Appendix D), Range Rings for all satellites displayed on Mercator Projection in GUI (Appendix E), a TryCatch M-File to make sure MATLAB is reading in the correct DTED (Appendix F), and an M-File to fix the empty spots in DTED (Appendix G),

## 4 RESULTS AND DISCUSSION

To demonstrate the computation of line of sight for various terrain situations, we present a mix of examples: near Salt Lake City (a mountainous example which exhibits significant loss of satellite visibility), San Francisco Bay (which allows us to compare theoretical results to some recent field measurements), and Sault Ste. Marie (a location with higher latitude). In each case, a 1-degree by 1-degree area is considered.

### 4.1 Salt Lake City, Utah

The area about Salt Lake City exhibits significant mountainous terrain; hence, a great opportunity for loss of visibility of a satellite. Figure 4.1 shows the region under consideration (longitude 111W to 112W, latitude 40N to 41N).

In the northwest and southwest regions of this area we have two lakes. Moving eastward we have a mountain ridge that proceeds from north to south; a valley follows this. Finally on the east side of the region we have another mountain range.

The terrain information from DTED was read and the elevation angles and line of sight were calculated for each point in the grid. Figure 4.2 shows the resulting elevation angle for the Pacific satellite. We note that this is as expected with most of the loss of visibility of the satellite (blackout – shown as white area) occurring on the eastern side of the mountain range. Figure 4.3 shows similar data for the Atlantic satellite, with the expected blackouts occurring on the western side of the mountains. There are fewer blackouts for the Atlantic satellite because of its higher nominal elevation angle ( $\sim 15.7^\circ$  versus  $\sim 6^\circ$ ).



Figure 4.1 – Salt Lake City Utah region.

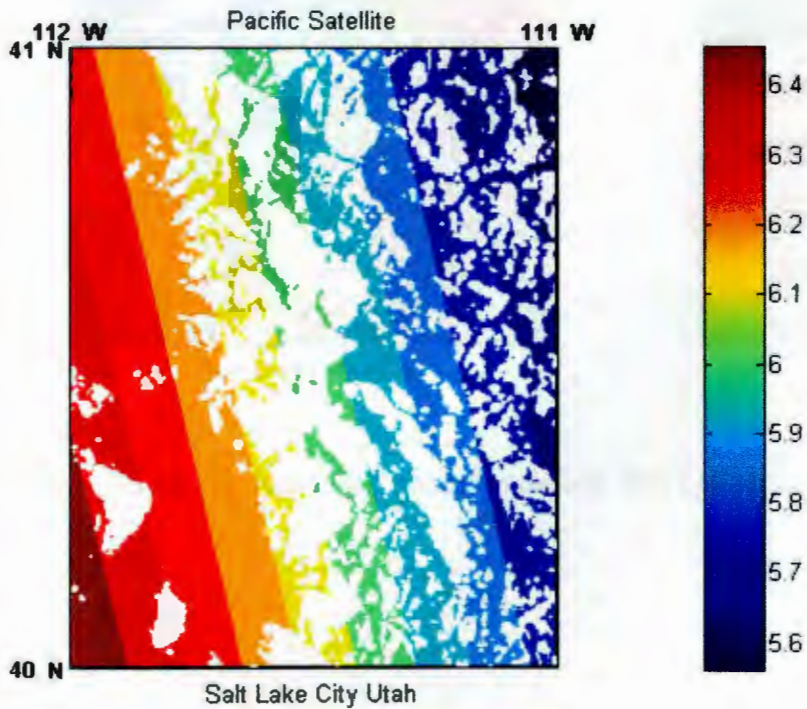


Figure 4.2 – Blackout areas to the Pacific satellite, Salt Lake City area.



Examining the data, the Pacific satellite calculation yields 27,758 points out of 59,081 points or 48 percent blackout in the Salt Lake City area; the Atlantic satellite calculation, on the other hand, yields 5,360 points out of 59,081 or 9 percent blackout. If we combine the two pictures to generate one that gives coverage based on visibility of either satellite, the result is only 1,947 points out of 59,081 or only 3.5 percent blackout in the region (shown as Figure 4.4).

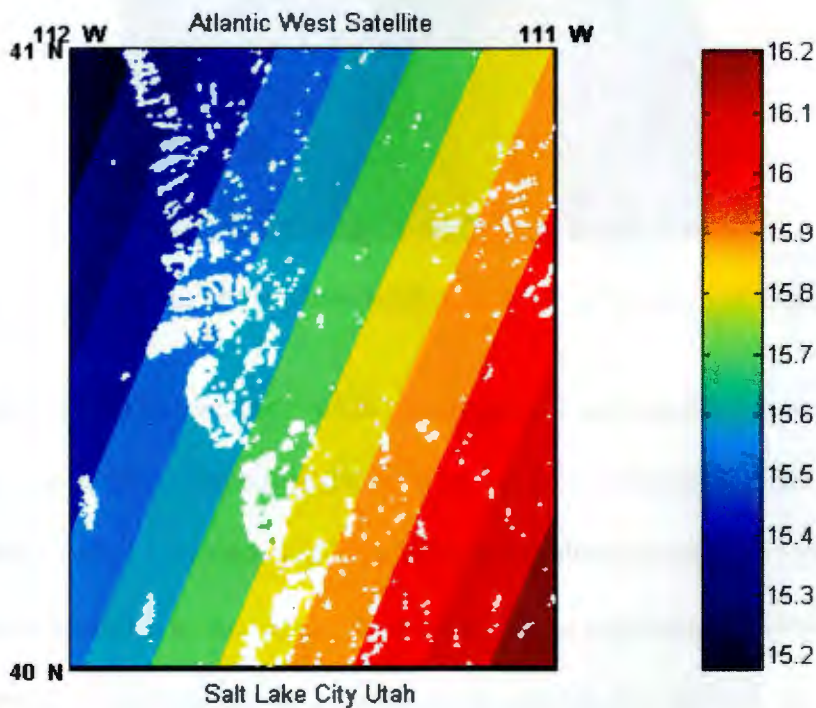


Figure 4.3 – Blackout areas to the Atlantic satellite, Salt Lake City Area.



*Figure 4.4 – White areas show where neither satellite is visible, Salt Lake City area.*

As mentioned before, the FAA is planning on launching three new satellites in the coming year or two. These satellites will be the PanAmSat Galaxy XV located at 125W, Telesat Anik F1 located at 107W, and a third satellite located at 085W. By applying these satellites to the region, we see significant increases in coverage and elevation angles to the satellite. By looking at Figures 4.5, 4.6, and 4.7, we see that coverage increases to approximately 100%, and elevation angles increase to 41 degrees, 43 degrees, and 35.8 degrees, respectively.

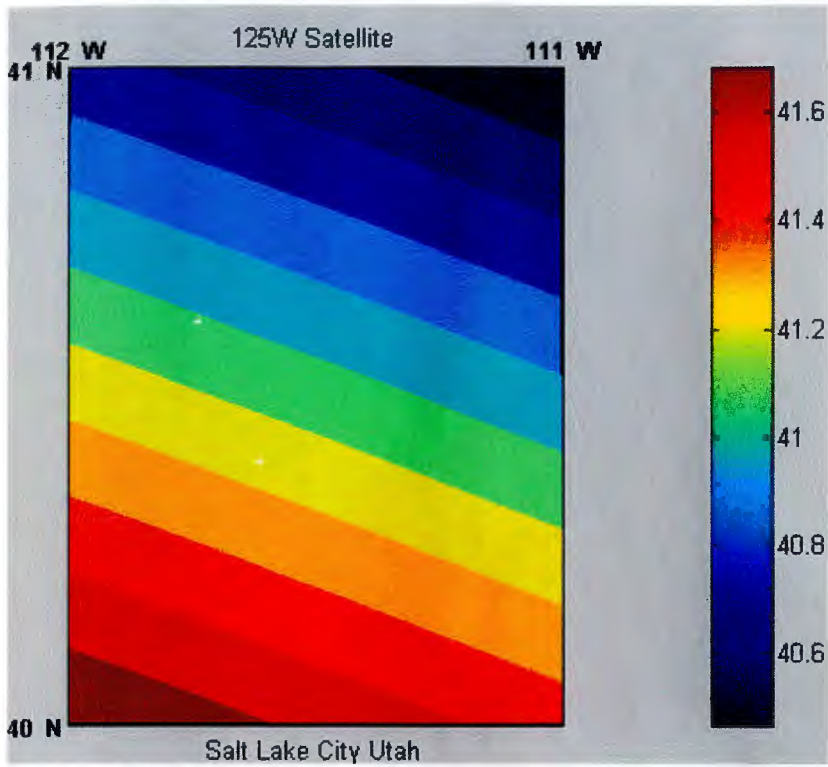


Figure 4.5 - 125W Satellite for Salt Lake City Region, 3pts of Blackout

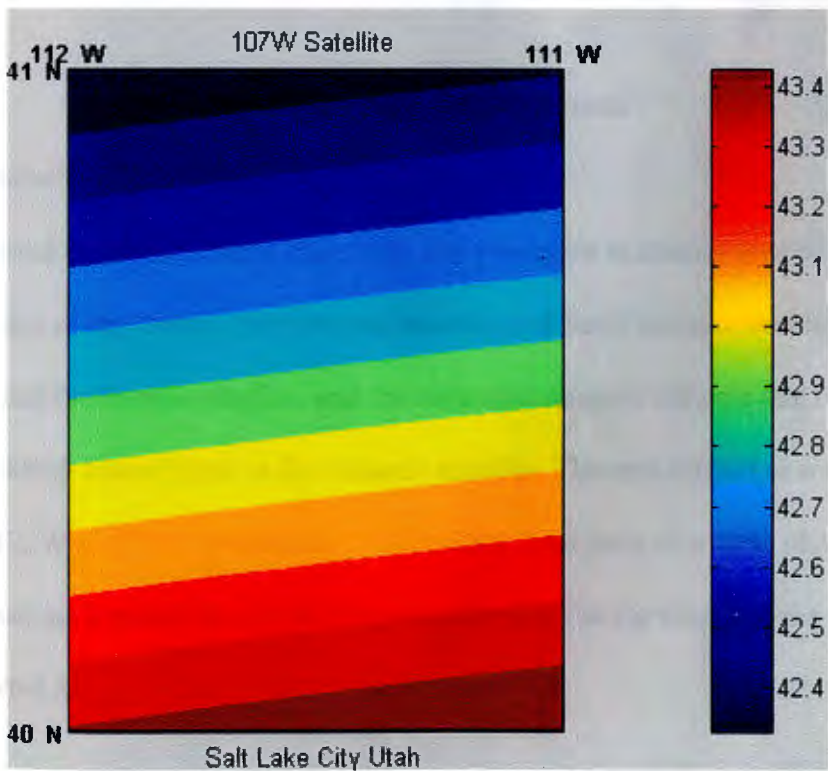


Figure 4.6 - 107W Satellite for Salt Lake City Region, 0 pts of Blackout

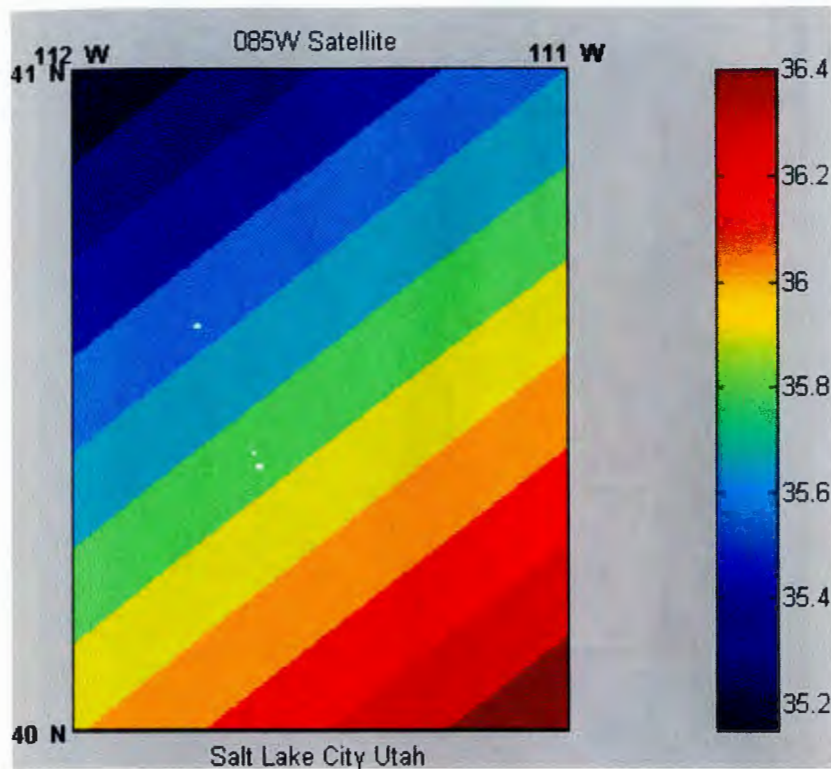


Figure 4.7 - 085W Satellite for Salt Lake City Region, 5 pts of Blackout

## 4.2 San Francisco, California

### 4.2.1 Theoretical Data

Beyond its obvious maritime value, San Francisco is also an area of interest due to the size of the harbor, the hills and buildings of San Francisco that lay between the harbor and the Pacific satellite, and the mountain range to the east that could potentially block line of sight to the Atlantic satellite. The area looked at was from longitude 122W to 123W and latitude 37N to 38N. This gave us a clear picture of the harbor as well as a partial picture to the mountain range to the east, and the city to the west (Figure 4.8).



Figure 4.8 - San Francisco Harbor Area

The line of sight calculations returned little blackout for either satellite. The Pacific satellite calculation yielded only 299 points out of 59,081, for 0.5 percent blackout; the Atlantic satellite calculation yielded 1,500 points out of 59,081, for 2.6 percent blackout. The surprising part about it was that all of the blackout locations occurred on land at this resolution (see Figures 4.9 and 4.10). Some blackout was expected along the eastern waterline of San Francisco due to the hills, but none occurred.

Our next step was to include the two new satellites going up plus the proposed third one. As stated before, the two new satellites are going up at longitudes 125W and 107W. The proposed satellite will be at 085W. By placing these three satellites in the sky, significant improvements are achieved in the elevation angles and visibility.

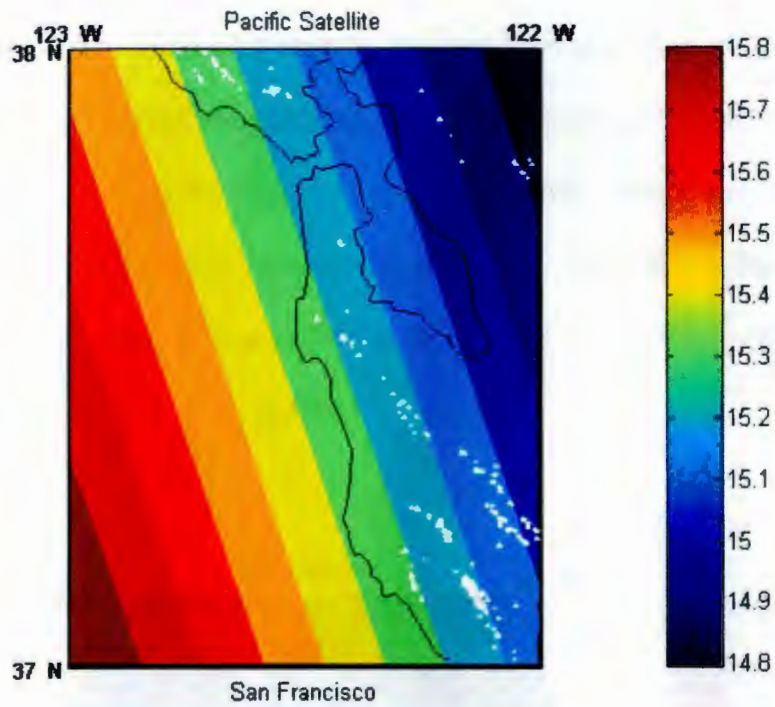


Figure 4.9 - Elevation angle to the Pacific satellite; white areas indicate no line of sight

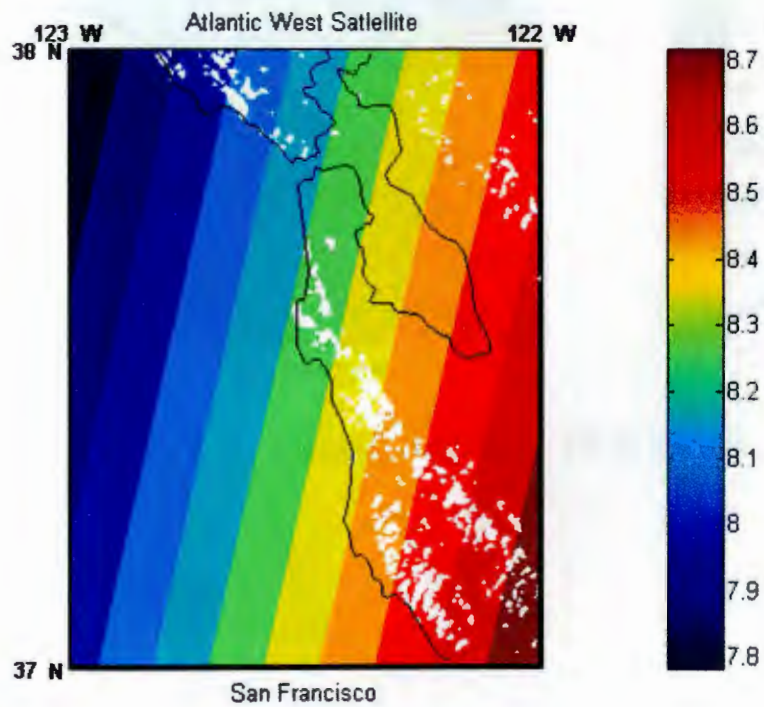


Figure 4.10 – Elevation angle to the Atlantic satellite; white areas indicate no line of sight.

The number of blackout areas in this case is reduced from approximately one-thousand to zero with the three new satellites. Elevation angles increase from 15 and 8 degrees, respectively, to 46, 43, and 31 degrees. Not only does this get rid of the blackout areas, but it provides redundancy in the system and allows for double and sometimes triple coverage in areas (elevation angle plots for these three satellites appears as Figures 4.11, 4.12, and 4.13).

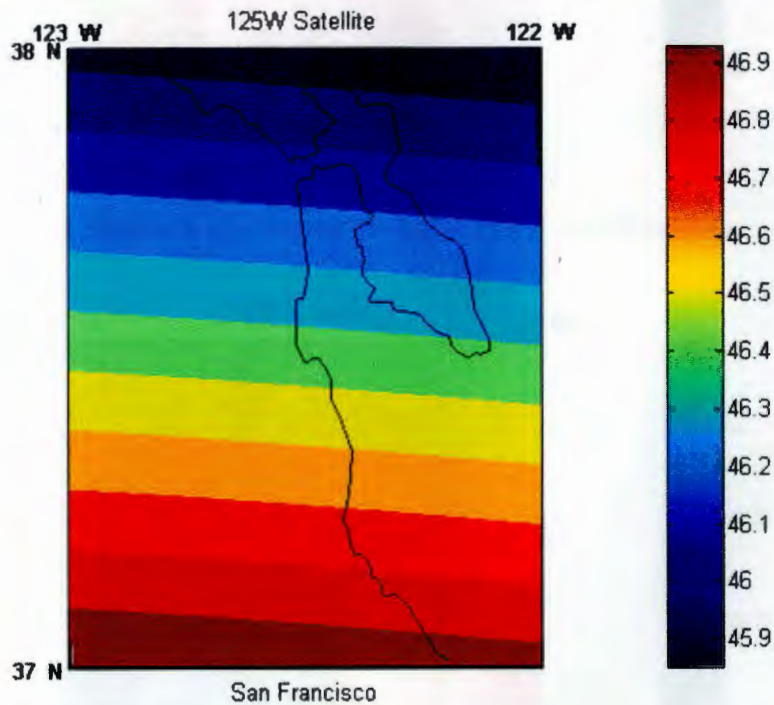


Figure 4.11 – Elevation angle to the 125W satellite.

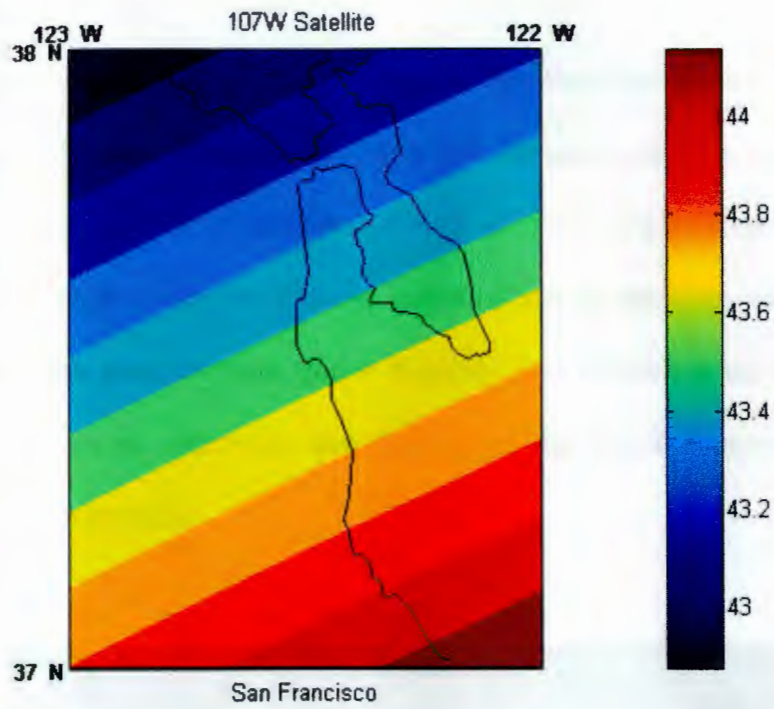


Figure 4.12 – Elevation angle to the 107W satellite.

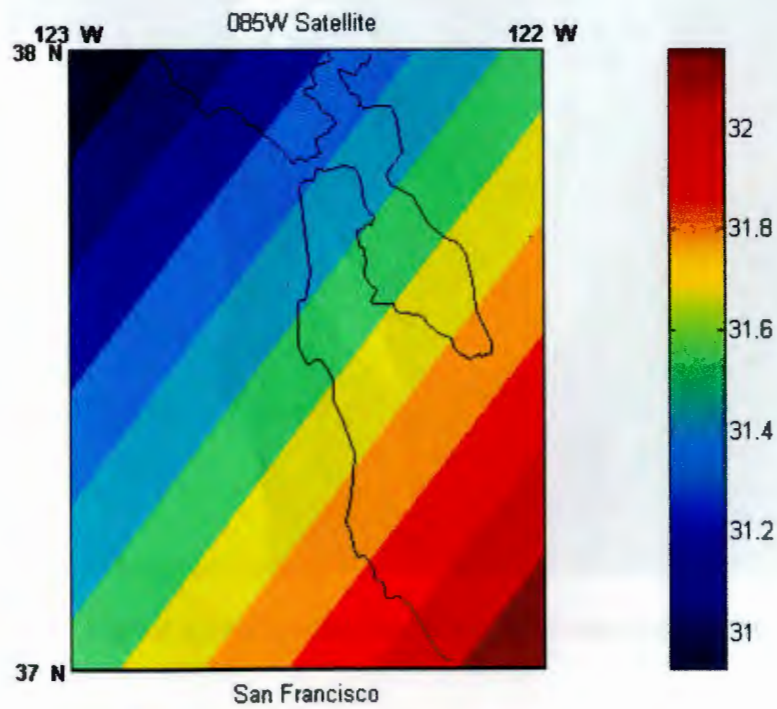
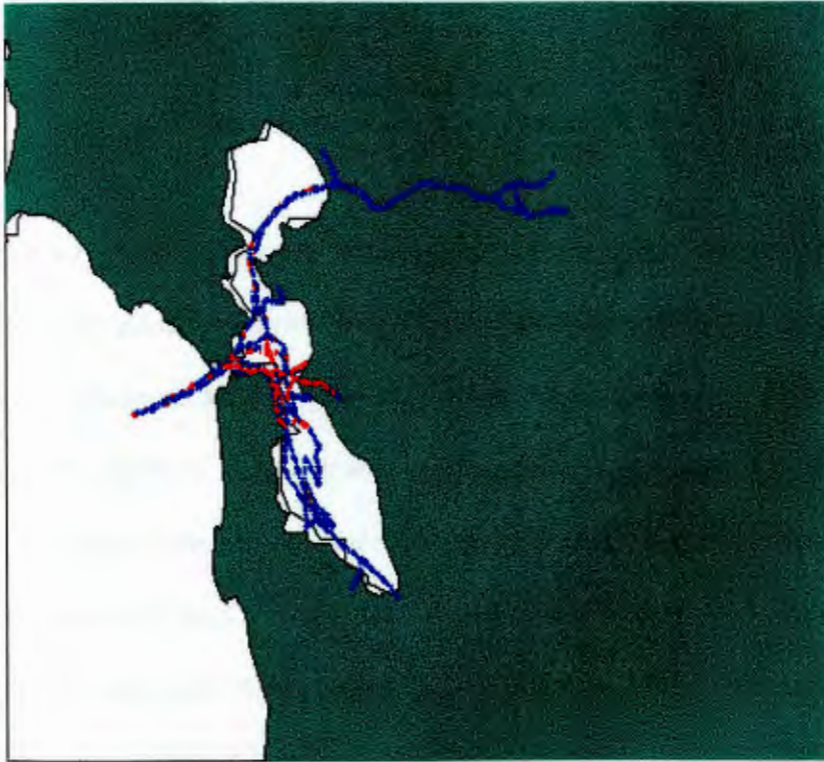


Figure 4.13 – Elevation angle to the 085W satellite.

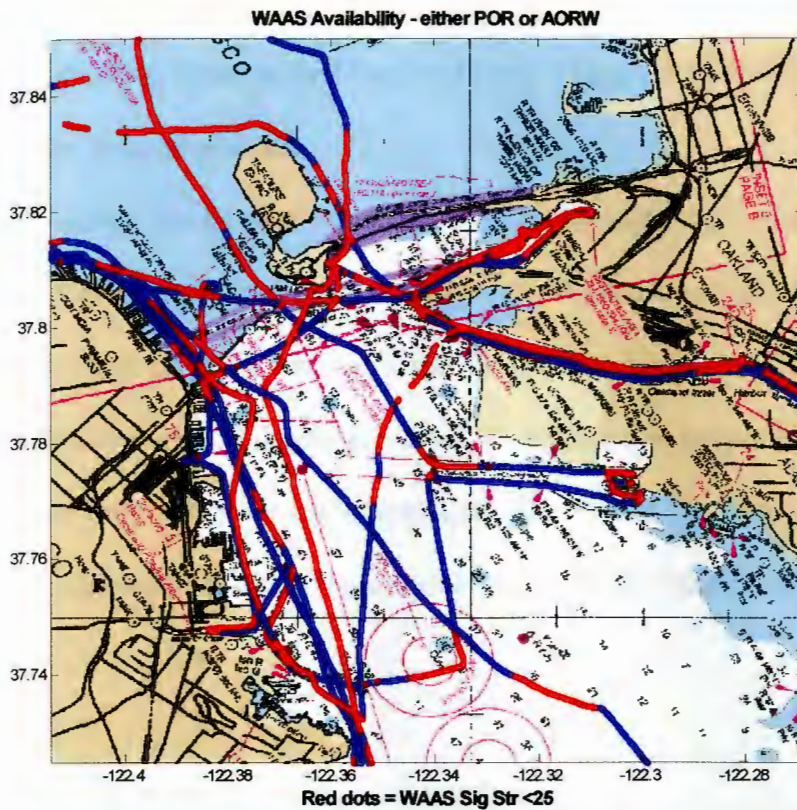


#### 4.2.2 Measured Data

Figure 4.14 shows a zoomed in area of San Francisco harbor depicting actual WAAS observations in 2004. In this picture the blue points indicate locations where at least one current WAAS satellite was visible to the user's receiver; the red points indicate those spots where the receiver indicated that no satellites were visible. One problem with this picture is that, due to the scale, it is difficult to see what is actually going on. You are not able to see that the red and blue dots are intermingled in some areas. Hence, we zoom in.



*Figure 4.14 – Actual data from San Francisco harbor*



*Figure 4.15 – Actual data from San Francisco harbor, zoomed*

Figure 4.15 zooms in on part of the harbor. In this figure we can see that the red and blue points are mixed throughout the trips that were traveled to collect the data. While this allows us a clearer picture of what was happening on the traveled legs, it also begs more questions. It is clear that east/west and north/south legs were made on this data collection mission. But, for example, on the east and west legs, you get conflicting results on WAAS availability. One leg says you have coverage from the satellite while the other says there is none. This is observed on the north and south bound legs also, and rules out terrain as a factor.

These abnormalities could be attributed to a several causes. There could have been other types of obstructions such as a bridge being passed under or a boat/tanker that passed by blocking the path. Also, the position of the antenna on the vessel could

have played a roll as the vessel rocked back and forth. (Figure 4.16 shows the vessel employed and the antenna position).

The theoretical results of elevation angle match up pretty well with the measured data. The average measured elevation to the Atlantic satellite was 8 degrees; the calculations above yielded a range of 8.1 to 8.6 degrees. For the Pacific satellite, the measured data averaged 15 degree elevation; the calculation ranges from 15 to 15.2 degrees. The only difference is the percentage of coverage. The measured data had coverage at roughly 75% of the time, while theoretical data had coverage at 100% in the harbor.



*Figure 4.16 – Antenna location on research vessel.*

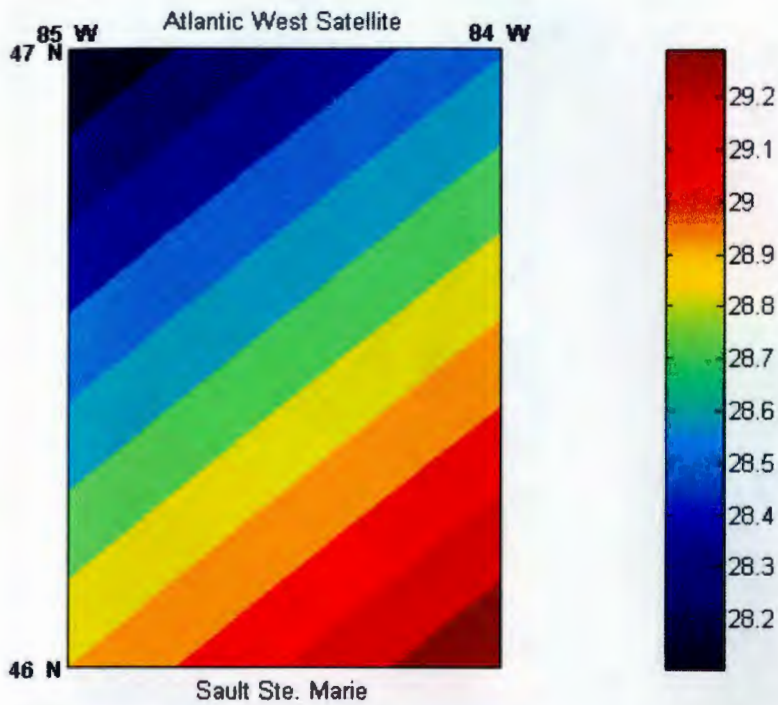


Figure 4.18 – Elevation angle to the Atlantic satellite; no loss of visibility due to terrain

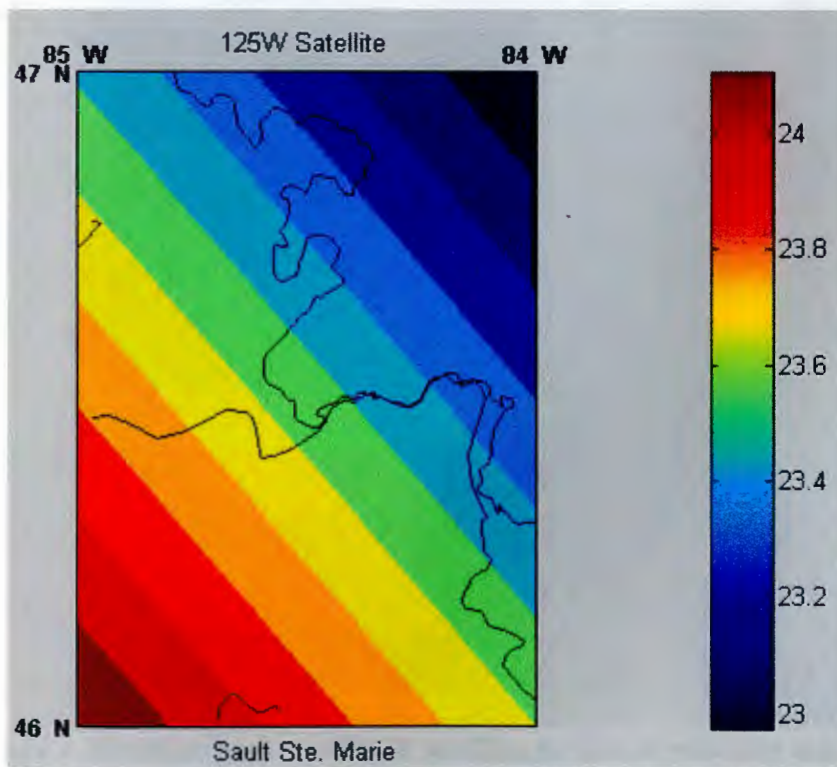


Figure 4.19 - Elevation angle to the 125W satellite, no loss of visibility due to terrain

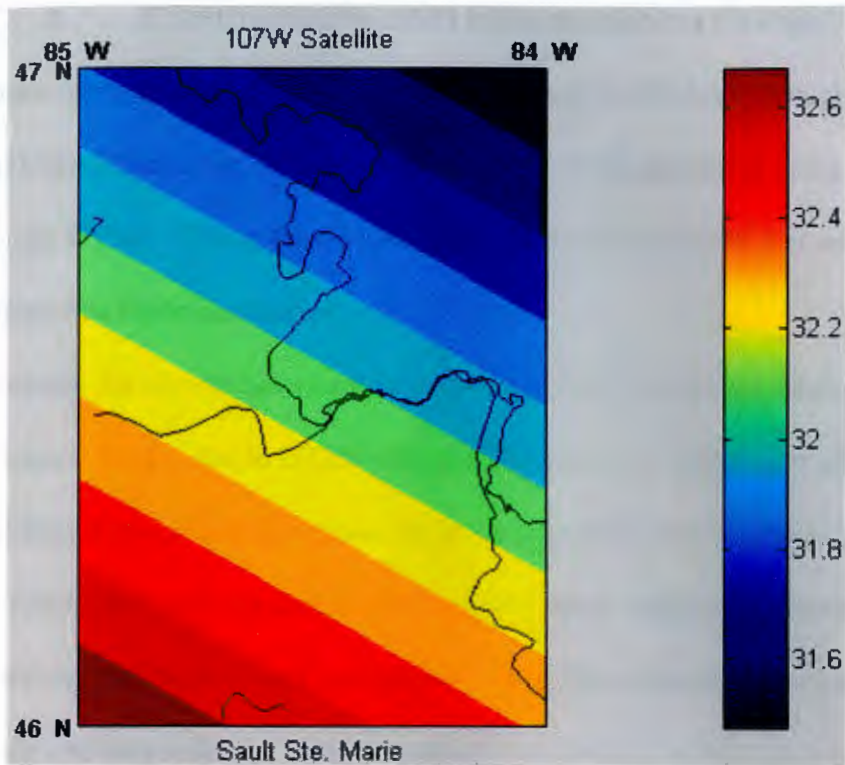


Figure 4.20 - Elevation angle to 107 W satellite, no loss of visibility due to terrain

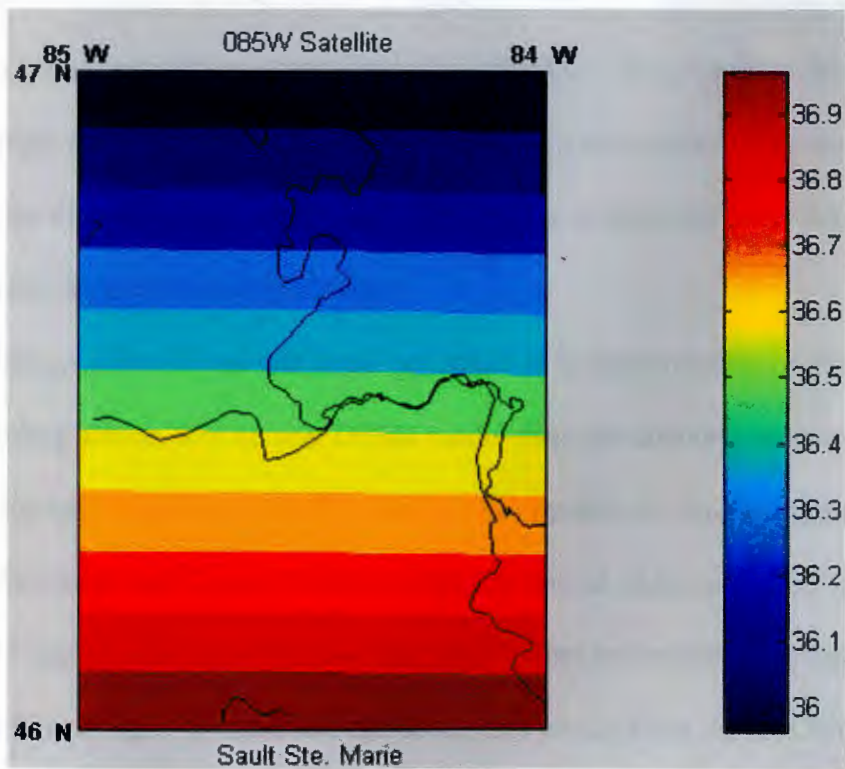


Figure 4.21 - Elevation angle to 085 W satellite, no loss of visibility due to terrain

## 5 CONCLUSIONS AND RECOMMENDATIONS

A tool for determining coverage (visibility) areas for WAAS throughout the continental United States has been developed. This was implemented using MATLAB and DTED. The system's results matched well with data that was collected from San Francisco harbor.

Currently the system has a hard time distinguishing waterways from land once you move inland. This is due to DTED only reading heights of land/water above sea level, not differentiating land and water. So, if the land surrounding a lake or river is only a few feet higher, then the data for the area will bleed together on the images created. Data can still be extracted for waterways and lakes, but you must know the exact latitude and longitude you want to look at.

The three proposed satellites to be launched later this year and early next year provide excellent redundancy to the current two satellites. They increase the average elevation angle for an area by as much as 25 degrees in some spots. Also by increasing the elevation angle, they reduce the number of blackout areas throughout the continental United States significantly.

There are a few things that could be looked at in future studies in this area. First is building affects. The current DTED model does not take into account buildings; the height above sea level in major cities ignores the buildings (e.g. compare Manhattan and Central Park). So 100-500 feet of additional "dirt" could be added to the landmasses around harbors and bays before recomputing elevation angles and testing line of sight. If blackouts did occur, this would show that the surrounding cityscape could affect the coverage of WAAS in the area.

Next, the data collected from San Francisco should be verified. The existing data, with much blackout of WAAS, was only taken once and not repeated to see if the consistent results were obtained.

A method to distinguish water from land needs to be explored for ease of reading the data results for inland locations. It would also allow for more accurate reading of the data to see if blackout areas are actually in the harbor areas or on the land. This tool that was developed is not limited to maritime navigation applications; it could be used for land use as well as on the water.

Finally, computer issues need to be addressed. It currently takes 90 minutes to calculate coverage for a one by one degree longitude/latitude grid on a 2 GHz Pentium 4 (full resolution for line of sight, but location choices decimated by a factor of 5). To run the entire one by one grid when not decimated would take a couple of hours.

## 6 REFERENCES

1. Ericson, S., Hegarty, C., & Sandhoo, K. S., "Assessment of the use of WAAS for land applications," MITRE: Center for Advanced Aviation System Development, September 1999, MITRE PRODUCT MP 98W0000083.
2. Federal Aviation Administration, "Wide Area Augmentation System," Oct. 2004, available at <http://gps.faa.gov/programs/index.htm>. (Accessed 27 April 2005).
3. Federal Aviation Administration. "WAAS is commissioned." FAA 21 Nov. 2003, available at
4. <http://gps.faa.gov/Library/Data/SatNav/satnavNovember-2003.htm> (Accessed 27 April 2005).
5. Federal Aviation Administration, "Status of Wide Area Augmentation System (WAAS)," July 2004, available at <http://gps.faa.gov/Library/Data/waas/WAASJUL2.DOC> (Accessed 27 April 2005).
6. Federal Geographic Data Committee, "What is DGPS?" Oct. 1999, available at <http://tsc.wes.army.mil/gps/dgps.htm> (Accessed 27 April 2005).
7. Garmin Ltd., "What is GPS?" 1996-2005, available at <http://www.garmin.com/aboutGPS/> (Accessed 27 April 2005).
8. Pike, J., "Digital Terrain Elevation Data [DTED]," Jan 2000, available at <http://www.fas.org/irp/program/core/dted.htm> (Accessed 27 April 2005).



## APPENDIX A MAIN PROGRAM

```
function varargout = test69(varargin)
% test69 M-file for test69.fig
% test69, by itself, creates a new test69 or raises the existing
% singleton*.
%
% H = test69 returns the handle to a new test69 or the handle to
% the existing singleton*.
%
% test69('CALLBACK', hObject,eventData,handles,...) calls the local
% function named CALLBACK in test69.M with the given input arguments.
%
% test69('Property','Value',...) creates a new test69 or raises the
% existing singleton*. Starting from the left, property value pairs are
% applied to the GUI before test69_OpeningFunction gets called. An
% unrecognized property name or invalid value makes property application
% stop. All inputs are passed to test69_OpeningFcn via varargin.
%
% *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
% instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help test69

% Last Modified by GUIDE v2.5 15-Nov-2004 12:00:18

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name', mfilename, ...
    'gui_Singleton', gui_Singleton, ...
    'gui_OpeningFcn', @test69_OpeningFcn, ...
    'gui_OutputFcn', @test69_OutputFcn, ...
    'gui_LayoutFcn', [], ...
    'gui_Callback', []);
if nargin & isstr(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before test69 is made visible.
```

```
function test69_OpeningFcn(hObject, eventdata, handles, varargin)
```

```
% This function has no output args, see OutputFcn.
```

```
% hObject handle to figure
```

```
% eventdata reserved - to be defined in a future version of MATLAB
```

```
% handles structure with handles and user data (see GUIDATA)
```

```
% varargin command line arguments to test69 (see VARARGIN)
```

```
set(handles.HOE_Ring,'Value',6);
```

```
set(handles.Height_Sat,'String',num2str(35785));
```

```
set(handles.height_of_user,'Value',1);
```

```
set(handles.Lat_User,'String',num2str(41.5));
```

```
set(handles.User_Long,'String',num2str(-71.5));
```

```
%Plots the Satellites and user on World Map with range rings
```

```
axes(handles.axis1)
```

```
%Map of World
```

```
axesm('miller','origin',[0 -110]);
```

```
framem;gridm
```

```
load coast
```

```
plotm(lat,long,'k');
```

```
h=displaym(usalo('state'));
```

```
w=1;
```

```
while w <= 51
```

```
    set(h(w),'FaceColor','y');
```

```
    w=w+1;
```

```
end
```

```
%draw and set color for great lakes
```

```
h=displaym(usalo('greatlakes'));
```

```
%set color for great lakes
```

```
set(h(1),'FaceColor','b');
```

```
set(h(2),'FaceColor','b');
```

```
set(h(3),'FaceColor','b');
```

```
for ind1=(1:size(h))
```

```
    zdatam(h(ind1),-3);
```

```
end;
```

```
axis tight;
```

```
% Choose default command line output for test69
```

```
handles.output = hObject;
```

```
% Update handles structure
```

```
guidata(hObject, handles);
```

```
% UIWAIT makes test69 wait for user response (see UIRESUME)
% uiwait(handles.test69);
```

```
% --- Outputs from this function are returned to the command line.
function varargout = test69_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
```

```
% Get default command line output from handles structure
varargout{1} = handles.output;
```

---

```
% --- Executes on button press in apply_pushbutton.
function apply_pushbutton_Callback(hObject, eventdata, handles)
% hObject handle to apply_pushbutton (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
```

```
cla
axis off;
```

---

```
%Gets all the handles that will be required for calculations
```

---

```
Latsat1 = str2num(get(handles.Lat_Sat,'String'));
Longsat1 = str2num(get(handles.Long_Sat,'String'));
Latsat2 = str2num(get(handles.Lat_Sat2,'String'));
Longsat2 = str2num(get(handles.Long_Sat2,'String'));
Latsat3 = str2num(get(handles.Lat_Sat3,'String'));
Longsat3 = str2num(get(handles.Long_Sat3,'String'));
Latuser = str2num(get(handles.Lat_User,'String'));
Longuser = str2num(get(handles.User_Long,'String'));
HeightSat = str2num(get(handles.Height_Sat,'String'));
HeightUser = str2num(get(handles.heigth_of_user,'String'));
PacSat = get(handles.Pac_Inmarsat,'Value');
LantSat = get(handles.Lant_Inmarsat,'Value');
CheckOne = get(handles.check_one,'Value');
CheckTwo = get(handles.check_two,'Value');
CheckThree = get(handles.check_three,'Value');
HoeRing = get(handles.HOE_Ring,'Value');
```

```
contourdisp = get(handles.contour_disp,'Value');
```

```
-----  
%Shows the users height above sea level from dted  
-----
```

```
Lat1=floor(Latuser);  
Lat2=ceil(Latuser);  
Long1=floor(Longuser);  
Long2=ceil(Longuser);  
[datagrid,refvec] = dted(pwd, 1, [Lat1 Lat2],[Long1 Long2]);  
latlength=length(datagrid(:,1));  
longlength=length(datagrid(1,:));  
latpos=((latlength-((Lat2-Latuser)/(1/(latlength-1)))));  
longpos=((((Long1-Longuser)*-1)/(1/(longlength-1)))+1);  
zi=interp2(datagrid,longpos,latpos);  
zz=get(handles.sealevelconversion,'Value');  
if zz==1%Feet  
    ziz=zi*3.2808399;  
elseif zz==2%meter  
    ziz=zi;  
end;  
set(handles.User_Sea_Level,'String',num2str(ziz));
```

```
-----  
%Does a check to make sure the heights are in the proper units  
-----
```

```
SatHeightUnit=get(handles.popupmenu1,'Value');  
if SatHeightUnit==1%km  
    sathi=HeightSat;  
elseif SatHeightUnit==2%nm  
    sathi=HeightSat*1.85200;  
elseif SatHeightUnit%sm  
    sathi=HeightSat*1.609433;  
end
```

```
Re=6378.1; %Distance from center of the earth in km  
UserHeightUnit=get(handles.popupmenu4,'Value');  
if UserHeightUnit==1%Feet  
    Re=Re+(zi/1000)+HeightUser*0.0003048  
elseif UserHeightUnit==2%meter  
    Re=Re+(zi/1000)+HeightUser*0.001  
end;
```

---

## %Satellite Equations

---

### %Pacific Satellite

```
%Pacific Satellite equations function
[xpac]=satequations(Re,Latuser,Longuser,0,178);
%Pacific Satellite anglex
[anglexpac]=angletheta(Re,xpac);
%Pacific Satellite height of eye function
[txpac,Bxpac]=heightofeye(Re,anglexpac,sathi);
set(handles.Pac_HOE,'String',num2str(txpac));
set(handles.Pac_Dist,'String',num2str(Bxpac));
```

### %Atlantic West Satellite

```
%Atlantic West Satellite equations function
[xlant]=satequations(Re,Latuser,Longuser,0,-54);
%Atlantic West Satellite anglex
[anglexlant]=angletheta(Re,xlant);
%Atlantic West Satellite height of eye function
[txlant,Bxlant]=heightofeye(Re,anglexlant,sathi);
set(handles.Lant_HOE,'String',num2str(txlant));
set(handles.Lant_Dist,'String',num2str(Bxlant));
```

### %Proposed Satellite One

```
if CheckOne == 1
```

```
%Proposed Satellite One equations function
[xpropone]=satequations(Re,Latuser,Longuser,Latsat1,Longsat1);
%Proposed Satellite One anglex
[anglexpropone]=angletheta(Re,xpropone);
%Proposed Satellite One height of eye function
[txone,Bxone]=heightofeye(Re,anglexpropone,sathi);
set(handles.height_of_eye,'String',num2str(txone));
set(handles.Sat_Dist,'String',num2str(Bxone));
```

```
else
```

```
set(handles.height_of_eye,'String',num2str(0));
set(handles.Sat_Dist,'String',num2str(0));
```

```
end
```

### %Proposed Satellite Two

```
if CheckTwo == 1
```

```
%Proposed Satellite Two equations function
[xproptwo]=satequations(Re,Latuser,Longuser,Latsat2,Longsat2);
%Proposed Satellite Two anglex
[anglexproptwo]=angletheta(Re,xproptwo);
%Proposed Satellite Two height of eye function
```

```

[txtwo,Bxtwo]=heightofeye(Re,anglexproptwo,sathi);
set(handles.height_of_eye2,'String',num2str(txtwo));
set(handles.Sat_Dist2,'String',num2str(Bxtwo));
else
    set(handles.height_of_eye2,'String',num2str(0));
    set(handles.Sat_Dist2,'String',num2str(0));
end

%Proposed Satellite Three
if CheckThree == 1
    %Proposed Satellite Three equations function
    [xproptthree]=satequations(Re,Latuser,Longuser,Latsat3,Longsat3);
    %Proposed Satellite Three anglex
    [anglexproptthree]=angletheta(Re,xproptthree);
    %Proposed Satellite Three height of eye function
    [txthree,Bxthree]=heightofeye(Re,anglexproptthree,sathi);
    set(handles.height_of_eye3,'String',num2str(txthree));
    set(handles.Sat_Dist3,'String',num2str(Bxthree));
else
    set(handles.height_of_eye3,'String',num2str(0));
    set(handles.Sat_Dist3,'String',num2str(0));
end

set(handles.Height_Sat,'String',num2str(HeightSat));

```

---

## %RANGE RINGS

---

```

if HoeRing==6
    %Proposed Satallite One
    if CheckOne == 1
        [Latone]=rangering11(Re,sathi,Longsat1);
        satlatone = [Latsat1]; satlongone = [Longsat1];
        plotm(satlatone,satlongone,'r*')
        [latone,longone]=scircle2(Latsat1,Longsat1,Latone,Longsat1);
        plotm(latone,longone,'b');
    else
        CheckOne ==0;
    end
    %Proposed Satallite two
    if CheckTwo == 1
        [Lattwo]=rangering11(Re,sathi,Longsat2);
        satlattwo = [Latsat2]; satlongtwo = [Longsat2];
        plotm(satlattwo,satlongtwo,'r*')
        [lattwo,longtwo]=scircle2(Latsat2,Longsat2,Lattwo,Longsat2);
        plotm(lattwo,longtwo,'b');
    end
end

```

```

else
    CheckTwo ==0;
end
%Proposed Satallite three
if CheckThree == 1
    [Latthree]=ranging11(Re,sathi,Longsat3);
    satlatthree = [Latsat3]; satlongthree = [Longsat3];
    plotm(satlatthree,satlongthree,'r*')
    [latthree,longthree]=scircle2(Latsat3,Longsat3,Latthree,Longsat3);
    plotm(latthree,longthree,'b');
else
    CheckThree ==0;
end
%Atlantic West Satellite
if LantSat == 1
    [Latlant]=ranging11(Re,sathi,-54);
    satlatlant = [0]; satlonglant = [-54];
    plotm(satlatlant,satlonglant,'r*')
    [latlant,longlant]=scircle2(0,-54,Latlant,-54);
    plotm(latlant,longlant,'b');
else
    LantSat ==0;
end
%Pacific Satellite
if PacSat == 1
    [Latpac]=ranging11(Re,sathi,178);
    satlatpac = [0]; satlongpac = [178];
    plotm(satlatpac,satlongpac,'r*')
    [latpac,longpac]=scircle2(0,178,Latpac,178);
    plotm(latpac,longpac,'b');
else
    PacSat ==0;
end
else
    %Proposed Satallite One
    if CheckOne == 1
        [Latone]=ranging2(Re,sathi,Longsat1,HoeRing);
        satlatone = [Latsat1]; satlongone = [Longsat1];
        plotm(satlatone,satlongone,'r*')
        [latone,longone]=scircle2(Latsat1,Longsat1,Latone,Longsat1);
        plotm(latone,longone,'b');
    else
        CheckOne ==0;
    end
    %Proposed Satallite two
    if CheckTwo == 1

```

```

[Lattwo]=rangering2(Re,sathi,Longsat2,HoeRing);
satlattwo = [Latsat2]; satlongtwo = [Longsat2];
plotm(satlattwo,satlongtwo,'r*')
[lattwo,longtwo]=scircle2(Latsat2,Longsat2,Lattwo,Longsat2);
plotm(lattwo,longtwo,'b');
else
    CheckTwo ==0;
end
%Proposed Satallite three
if CheckThree == 1
    [Latthree]=rangering2(Re,sathi,Longsat3,HoeRing);
    satlatthree = [Latsat3]; satlongthree = [Longsat3];
    plotm(satlatthree,satlongthree,'r*')
    [latthree,longthree]=scircle2(Latsat3,Longsat3,Latthree,Longsat3);
    plotm(latthree,longthree,'b');
else
    CheckThree ==0;
end
%Atlantic West Satellite
if LantSat == 1
    [Latlant]=rangering2(Re,sathi,-54,HoeRing);
    satlatlant = [0]; satlonglant = [-54];
    plotm(satlatlant,satlonglant,'r*')
    [latlant,longlant]=scircle2(0,-54,Latlant,-54);
    plotm(latlant,longlant,'b');
else
    LantSat ==0;
end
%Pacific Satellite
if PacSat == 1
    [Latpac]=rangering2(Re,sathi,178,HoeRing);
    satlatpac = [0]; satlongpac = [178];
    plotm(satlatpac,satlongpac,'r*')
    [latpac,longpac]=scircle2(0,178,Latpac,178);
    plotm(latpac,longpac,'b');
else
    PacSat ==0;
end
end

```

---

```

%Plots the World Map again when apply button is hit

```

---

```

%Plots the Satellites and user on World Map with range rings
axes(handles.axis1)

```



```

%Map of World
axesm('miller','origin',[0 -110]);
framem;gridm
load coast
plotm(lat,long,'k');
h=displaym(usalo('state'));
w=1;
while w <= 51
    set(h(w),'FaceColor','y');
    w=w+1;
end
%draw and set color for great lakes
h=displaym(usalo('greatlakes'));
%set color for great lakes
set(h(1),'FaceColor','b');
set(h(2),'FaceColor','b');
set(h(3),'FaceColor','b');
for ind1=(1:size(h))
    zdatam(h(ind1),-3);
end;

```

---

%Plots the positions of the Pacific and Atlantic West Satellites along with  
 %User Coordinates

---

```

%Plots Pac and Lant Satellites
satlats = [0 0]; satlongs = [178 -54];
plotm(satlats,satlongs,'r*')
%Plots User coordinates
userlats=Latuser;
userlongs=Longuser;
plotm(userlats,userlongs,'b*')

```

---

% --- Executes on button press in Pac\_Inmarsat.  
 function Pac\_Inmarsat\_Callback(hObject, eventdata, handles)  
 % hObject handle to Pac\_Inmarsat (see GCBO)  
 % eventdata reserved - to be defined in a future version of MATLAB  
 % handles structure with handles and user data (see GUIDATA)  
  
 % Hint: get(hObject,'Value') returns toggle state of Pac\_Inmarsat

---

% --- Executes on button press in Lant\_Inmarsat.

```
function Lant_Inmarsat_Callback(hObject, eventdata, handles,Re)
% hObject handle to Lant_Inmarsat (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
```

```
% Hint: get(hObject,'Value') returns toggle state of Lant_Inmarsat
```

---

```
% --- Executes during object creation, after setting all properties.
function Lat_Sat_CreateFcn(hObject, eventdata, handles)
% hObject handle to Lat_Sat (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
```

```
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function Lat_Sat_Callback(hObject, eventdata, handles)
% hObject handle to Lat_Sat (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of Lat_Sat as text
% str2double(get(hObject,'String')) returns contents of Lat_Sat as a double
```

---

```
% --- Executes during object creation, after setting all properties.
function Long_Sat_CreateFcn(hObject, eventdata, handles)
% hObject handle to Long_Sat (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
```

```
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```

function Long_Sat_Callback(hObject, eventdata, handles)
% hObject handle to Long_Sat (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Long_Sat as text
% str2double(get(hObject,'String')) returns contents of Long_Sat as a double

```

---

```

% --- Executes during object creation, after setting all properties.
function Lat_User_CreateFcn(hObject, eventdata, handles)
% hObject handle to Lat_User (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

```

```

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

```

```

function Lat_User_Callback(hObject, eventdata, handles)
% hObject handle to Lat_User (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

```

```

% Hints: get(hObject,'String') returns contents of Lat_User as text
% str2double(get(hObject,'String')) returns contents of Lat_User as a double

```

---

```

% --- Executes during object creation, after setting all properties.
function User_Long_CreateFcn(hObject, eventdata, handles)
% hObject handle to User_Long (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

```

```

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

```

```

% Hint: edit controls usually have a white background on Windows.
%   See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

```

```

function Sat_Dist_Callback(hObject, eventdata, handles)
% hObject   handle to Sat_Dist (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Sat_Dist as text
%   str2double(get(hObject,'String')) returns contents of Sat_Dist as a double

```

---

```

% --- Executes during object creation, after setting all properties.
function Height_Sat_CreateFcn(hObject, eventdata, handles)
% hObject   handle to Height_Sat (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   empty - handles not created until after all CreateFcns called

```

```

% Hint: edit controls usually have a white background on Windows.
%   See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

```

```

function Height_Sat_Callback(hObject, eventdata, handles)
% hObject   handle to Height_Sat (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Height_Sat as text
%   str2double(get(hObject,'String')) returns contents of Height_Sat as a double

```

---

```

% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject   handle to popupmenu1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   empty - handles not created until after all CreateFcns called

```

```

% Hint: popupmenu controls usually have a white background on Windows.
%   See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject   handle to popupmenu1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu1 contents as cell array
%   contents{get(hObject,'Value')} returns selected item from popupmenu1

```

```

%-----
% --- Executes during object creation, after setting all properties.
function Lant_HOE_CreateFcn(hObject, eventdata, handles)
% hObject   handle to Lant_HOE (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   empty - handles not created until after all CreateFcns called

```

```

% Hint: edit controls usually have a white background on Windows.
%   See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

```

```

function Lant_HOE_Callback(hObject, eventdata, handles)
% hObject   handle to Lant_HOE (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

```

```

% Hints: get(hObject,'String') returns contents of Lant_HOE as text
%   str2double(get(hObject,'String')) returns contents of Lant_HOE as a double

```

```

%-----
% --- Executes during object creation, after setting all properties.
function Lant_Dist_CreateFcn(hObject, eventdata, handles)
% hObject   handle to Lant_Dist (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   empty - handles not created until after all CreateFcns called

```

```

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function Lant_Dist_Callback(hObject, eventdata, handles)
% hObject handle to Lant_Dist (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Lant_Dist as text
% str2double(get(hObject,'String')) returns contents of Lant_Dist as a double

```

---

```

% --- Executes during object creation, after setting all properties.
function Pac_Dist_CreateFcn(hObject, eventdata, handles)
% hObject handle to Pac_Dist (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

```

```

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

```

```

function Pac_Dist_Callback(hObject, eventdata, handles)
% hObject handle to Pac_Dist (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Pac_Dist as text
% str2double(get(hObject,'String')) returns contents of Pac_Dist as a double

```

---

```

% --- Executes during object creation, after setting all properties.
function Pac_HOE_CreateFcn(hObject, eventdata, handles)
% hObject handle to Pac_HOE (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

```

```

% Hint: edit controls usually have a white background on Windows.
%   See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function Pac_HOE_Callback(hObject, eventdata, handles)
% hObject   handle to Pac_HOE (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Pac_HOE as text
%   str2double(get(hObject,'String')) returns contents of Pac_HOE as a double

% --- Executes during object creation, after setting all properties.
function HOE_Ring_CreateFcn(hObject, eventdata, handles)
% hObject   handle to HOE_Ring (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%   See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

% --- Executes on selection change in HOE_Ring.
function HOE_Ring_Callback(hObject, eventdata, handles)
% hObject   handle to HOE_Ring (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns HOE_Ring contents as cell array
%   contents{get(hObject,'Value')} returns selected item from HOE_Ring

% --- Executes during object creation, after setting all properties.
function height_of_eye2_CreateFcn(hObject, eventdata, handles)

```

```
% hObject handle to height_of_eye2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
```

```
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function height_of_eye2_Callback(hObject, eventdata, handles)
```

```
% hObject handle to height_of_eye2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of height_of_eye2 as text
% str2double(get(hObject,'String')) returns contents of height_of_eye2 as a double
```

```
% --- Executes during object creation, after setting all properties.
```

```
function height_of_eye3_CreateFcn(hObject, eventdata, handles)
```

```
% hObject handle to height_of_eye3 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
```

```
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function height_of_eye3_Callback(hObject, eventdata, handles)
```

```
% hObject handle to height_of_eye3 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of height_of_eye3 as text
% str2double(get(hObject,'String')) returns contents of height_of_eye3 as a double
```



```

% --- Executes during object creation, after setting all properties.
function Sat_Dist2_CreateFcn(hObject, eventdata, handles)
% hObject handle to Sat_Dist2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function Sat_Dist2_Callback(hObject, eventdata, handles)
% hObject handle to Sat_Dist2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Sat_Dist2 as text
% str2double(get(hObject,'String')) returns contents of Sat_Dist2 as a double

% --- Executes during object creation, after setting all properties.
function Sat_Dist3_CreateFcn(hObject, eventdata, handles)
% hObject handle to Sat_Dist3 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function Sat_Dist3_Callback(hObject, eventdata, handles)
% hObject handle to Sat_Dist3 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB

```

```

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Sat_Dist3 as text
% str2double(get(hObject,'String')) returns contents of Sat_Dist3 as a double

% --- Executes during object creation, after setting all properties.
function heigth_of_user_CreateFcn(hObject, eventdata, handles)
% hObject handle to heigth_of_user (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function heigth_of_user_Callback(hObject, eventdata, handles)
% hObject handle to heigth_of_user (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of heigth_of_user as text
% str2double(get(hObject,'String')) returns contents of heigth_of_user as a double

% --- Executes during object creation, after setting all properties.
function popupmenu4_CreateFcn(hObject, eventdata, handles)
% hObject handle to popupmenu4 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

```

```

% --- Executes on selection change in popupmenu4.
function popupmenu4_Callback(hObject, eventdata, handles)
% hObject handle to popupmenu4 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu4 contents as cell array
% contents{get(hObject,'Value')} returns selected item from popupmenu4

```

```

% --- Executes during object creation, after setting all properties.
function Lat_Sat2_CreateFcn(hObject, eventdata, handles)
% hObject handle to Lat_Sat2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

```

```

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

```

```

function Lat_Sat2_Callback(hObject, eventdata, handles)
% hObject handle to Lat_Sat2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

```

```

% Hints: get(hObject,'String') returns contents of Lat_Sat2 as text
% str2double(get(hObject,'String')) returns contents of Lat_Sat2 as a double

```

```

% --- Executes during object creation, after setting all properties.
function Long_Sat2_CreateFcn(hObject, eventdata, handles)
% hObject handle to Long_Sat2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

```

```

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else

```

```
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function Long_Sat2_Callback(hObject, eventdata, handles)
% hObject   handle to Long_Sat2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Long_Sat2 as text
%       str2double(get(hObject,'String')) returns contents of Long_Sat2 as a double
```

```
% --- Executes during object creation, after setting all properties.
function Lat_Sat3_CreateFcn(hObject, eventdata, handles)
% hObject   handle to Lat_Sat3 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function Lat_Sat3_Callback(hObject, eventdata, handles)
% hObject   handle to Lat_Sat3 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Lat_Sat3 as text
%       str2double(get(hObject,'String')) returns contents of Lat_Sat3 as a double
```

```
% --- Executes during object creation, after setting all properties.
function Long_Sat3_CreateFcn(hObject, eventdata, handles)
% hObject   handle to Long_Sat3 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
```

```

% See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function Long_Sat3_Callback(hObject, eventdata, handles)
% hObject handle to Long_Sat3 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Long_Sat3 as text
% str2double(get(hObject,'String')) returns contents of Long_Sat3 as a double

% --- Executes on button press in check_one.
function check_one_Callback(hObject, eventdata, handles)
% hObject handle to check_one (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of check_one

% --- Executes on button press in check_two.
function check_two_Callback(hObject, eventdata, handles)
% hObject handle to check_two (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of check_two

% --- Executes on button press in check_three.
function check_three_Callback(hObject, eventdata, handles)
% hObject handle to check_three (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of check_three

% --- Executes on button press in zoom_box.

```

```

function zoom_box_Callback(hObject, eventdata, handles)
% hObject handle to zoom_box (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

```

```

% Hint: get(hObject,'Value') returns toggle state of zoom_box
if (get(handles.zoom_box,'Value') == get(handles.zoom_box,'Max'))
    zoom on;
else
    zoom off;
end

```

```

% --- Executes on button press in location_checkbox.

```

```

function location_checkbox_Callback(hObject, eventdata, handles)
% hObject handle to location_checkbox (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

```

```

% Hint: get(hObject,'Value') returns toggle state of location_checkbox

```

```

if (get(handles.location_checkbox,'Value') == get(handles.location_checkbox,'Max'))
    ginput(1)
else
end

```

```

% --- Executes during object creation, after setting all properties.

```

```

function User_Sea_Level_CreateFcn(hObject, eventdata, handles)
% hObject handle to User_Sea_Level (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

```

```

% Hint: edit controls usually have a white background on Windows.

```

```

% See ISPC and COMPUTER.

```

```

if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

```

```

function User_Sea_Level_Callback(hObject, eventdata, handles)

```

```

% hObject handle to User_Sea_Level (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of User_Sea_Level as text
% str2double(get(hObject,'String')) returns contents of User_Sea_Level as a
double

```

```

% --- Executes on button press in contour_disp.
function contour_disp_Callback(hObject, eventdata, handles)
% hObject handle to contour_disp (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

```

```

% Hint: get(hObject,'Value') returns toggle state of contour_disp

```

```

contourdisp = get(handles.contour_disp,'Value');
satcheck = get(handles.sat_check,'Value');
Latuser = str2num(get(handles.Lat_User,'String'));
Longuser = str2num(get(handles.User_Long,'String'));
HeightSat = str2num(get(handles.Height_Sat,'String'));
HeightUser = str2num(get(handles.heigth_of_user,'String'));

```

```

%-----
%Does a check to make sure the heights are in the proper units
%-----

```

```

SatHeightUnit=get(handles.popupmenu1,'Value');
if SatHeightUnit==1%km
    sathi=HeightSat;
elseif SatHeightUnit==2%nm
    sathi=HeightSat*1.85200;
elseif SatHeightUnit%sm
    sathi=HeightSat*1.609433;
end;

```

```

Re=6378.1; %Distance from center of the earth in km
UserHeightUnit=get(handles.popupmenu4,'Value');
if UserHeightUnit==1%Feet
    Re=Re+HeightUser*0.0003048;
elseif UserHeightUnit==2%meter
    Re=Re+HeightUser*0.001;
end;

```

```

%-----

```

%DTED Stuff and second plot

```
%-----  
if contourdisp == 1  
    [datagrid,refvec,Lat1,Long1,Lat2,Long2]=trycatch(Latuser,Longuser);  
  
    [newdatagrid]=fixeddatagrid(datagrid);  
  
    Z=newdatagrid((241:481),(241:481));  
    datagridlength=length(Z(1,:))-1;  
    n=1;  
    M=1;  
    HoeMatrix=[];  
    ReRe2=[];  
    for Longuser=Long1+1:1/datagridlength:Long2-1;  
        Longuser;  
        Latsat=0;  
        if satcheck == 1  
            Longsat=178;  
        elseif satcheck == 2  
            Longsat=-54;  
        elseif satcheck == 3  
            Longsat=str2num(get(handles.Long_Sat,'String'));  
        elseif satcheck == 4  
            Longsat=str2num(get(handles.Long_Sat2,'String'));  
        else  
            Longsat=str2num(get(handles.Long_Sat3,'String'));  
        end  
        FirstHOE=[];  
        ReRe=[];  
        N=1;  
        for Latuser=(Lat1+1:1/datagridlength:Lat2-1)  
            datah=(Z(N,M));  
            Re1=(datah/1000+Re);  
            [xproptwo]=satequations(Re1,Latuser,Longuser,Latsat,Longsat);  
            [anglexproptwo]=angletheta(Re1,xproptwo);  
            [txttwo,Bxtwo]=heightofeye(Re1,anglexproptwo,sathi);  
            dtedbox=get(handles.dted_box,'Value');  
            if dtedbox == 1  
  
[latout,lonout,satheight]=bearing(Latuser,Longuser,Latsat,Longsat,txttwo);  
            vis=los2(datagrid,refvec,Latuser,Longuser,latout,lonout,0,satheight);  
            if vis==0  
                tx=NaN;  
            else  
                tx=txttwo;  
            end  
        end  
    end  
end
```



```

    else
        tx=txtwo;
    end
    FirstHOE=[FirstHOE,[tx]];
    ReRe=[ReRe,[ReRe]];
    N=N+1;
end
M=M+1
HoeMatrix=[HoeMatrix,[FirstHOE]];
ReRe2=[ReRe2,[ReRe2]];
n=n+1;
end
%figure
%mesh(datagrid)
%figure
%mesh(newdatagrid)
%figure
%mesh(Z)
%figure
%mesh(HoeMatrix)
length(HoeMatrix(1,:));
length(HoeMatrix(:,1));
axes(handles.axis2);
contourf(HoeMatrix,10);
colorbar('vert');
figure
contourf(HoeMatrix,10);
colorbar('vert');
save('HoeMatrix');

else
    contourdisp=0;
end

% --- Executes on button press in pac_check.
function pac_check_Callback(hObject, eventdata, handles)
% hObject    handle to pac_check (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of pac_check

% --- Executes on button press in lant_check.
function lant_check_Callback(hObject, eventdata, handles)
% hObject    handle to lant_check (see GCBO)

```

```
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
```

```
% Hint: get(hObject,'Value') returns toggle state of lant_check
```

```
% --- Executes on button press in one_check.
```

```
function one_check_Callback(hObject, eventdata, handles)
```

```
% hObject handle to one_check (see GCBO)
```

```
% eventdata reserved - to be defined in a future version of MATLAB
```

```
% handles structure with handles and user data (see GUIDATA)
```

```
% Hint: get(hObject,'Value') returns toggle state of one_check
```

```
% --- Executes on button press in two_check.
```

```
function two_check_Callback(hObject, eventdata, handles)
```

```
% hObject handle to two_check (see GCBO)
```

```
% eventdata reserved - to be defined in a future version of MATLAB
```

```
% handles structure with handles and user data (see GUIDATA)
```

```
% Hint: get(hObject,'Value') returns toggle state of two_check
```

```
% --- Executes on button press in three_check.
```

```
function three_check_Callback(hObject, eventdata, handles)
```

```
% hObject handle to three_check (see GCBO)
```

```
% eventdata reserved - to be defined in a future version of MATLAB
```

```
% handles structure with handles and user data (see GUIDATA)
```

```
% Hint: get(hObject,'Value') returns toggle state of three_check
```

```
% --- Executes during object creation, after setting all properties.
```

```
function sat_check_CreateFcn(hObject, eventdata, handles)
```

```
% hObject handle to sat_check (see GCBO)
```

```
% eventdata reserved - to be defined in a future version of MATLAB
```

```
% handles empty - handles not created until after all CreateFcns called
```

```
% Hint: popupmenu controls usually have a white background on Windows.
```

```
% See ISPC and COMPUTER.
```

```
if ispc
```

```
    set(hObject,'BackgroundColor','white');
```

```
else
```

```
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
```

```
end
```

```

% --- Executes on selection change in sat_check.
function sat_check_Callback(hObject, eventdata, handles)
% hObject handle to sat_check (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns sat_check contents as cell array
% contents{get(hObject,'Value')} returns selected item from sat_check

```

```

% --- Executes during object creation, after setting all properties.
function time_left_CreateFcn(hObject, eventdata, handles)
% hObject handle to time_left (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

```

```

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

```

```

function time_left_Callback(hObject, eventdata, handles)
% hObject handle to time_left (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

```

```

% Hints: get(hObject,'String') returns contents of time_left as text
% str2double(get(hObject,'String')) returns contents of time_left as a double

```

```

% --- Executes during object creation, after setting all properties.
function sealevelconversion_CreateFcn(hObject, eventdata, handles)
% hObject handle to sealevelconversion (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

```

```

% Hint: popmenu controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc

```

```

    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

% --- Executes on selection change in sealevelconversion.
function sealevelconversion_Callback(hObject, eventdata, handles)
% hObject    handle to sealevelconversion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns sealevelconversion contents as cell
array
%    contents{get(hObject,'Value')} returns selected item from sealevelconversion

% --- Executes on button press in dted_box.
function dted_box_Callback(hObject, eventdata, handles)
% hObject    handle to dted_box (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of dted_box

% --- Executes during object creation, after setting all properties.
function desired_hoe_CreateFcn(hObject, eventdata, handles)
% hObject    handle to desired_hoe (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%    See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function desired_hoe_Callback(hObject, eventdata, handles)
% hObject    handle to desired_hoe (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of desired_hoe as text
%    str2double(get(hObject,'String')) returns contents of desired_hoe as a double

```

## APPENDIX B SATEQUATIONS

```
function [x]=satequations(Re,Latuser,Longuser,Latsat,Longsat)
%Satellite equations function
Lat1=Latuser;
Long1=Longuser-Longsat;

if Long1 > 180
    Long1=360-Long1;
elseif Long1 < -180
    Long1=360+Long1;
else
    Long1=Long1;
end

%Satellite
lateq=tan(Lat1*pi/180);
lateqsq=lateq^2;
longeq=tan(Long1*pi/180);
longeqsq=longeq^2;
Re2=Re^2;
divisor1=lateqsq+lateqsq*longeqsq+1+longeqsq;
if Long1<90 & Long1>-90
    x=sqrt(Re2/(divisor1));
else
    x=sqrt(Re2/(divisor1))*(-1);
end
```

## APPENDIX C ANGLETHET

```
function [xx]=angletheta(Re,x)
%Anglex function
if x~=0
    xx=acos(x/Re)*180/pi;
else
    xx=Latuser;
end
```

## APPENDIX D ELEVATION ANGLE

```
function [tx,Bx]=heightofeye(Re,anglex,sathi)
%Function [t, BD]=heightofeye(Re,anglex,sathi)
%t=height of eye, or angle of satellite off horizon from user position
%BD= distance between user and satellite
%Re= distance from center of earth to crust in km
%anglex=angle between xaxis pointing at satellite and some xyz axis
%pointing at user position
%Height of the satellite = sathi
x1=anglex;
z1=90;
y1=90-x1;
v1=x1;
w1=y1;
q1=180-w1;
m1=x1;
%Distance from center of earth to surface
AB=Re;
%Distance from center of the earth to the satellite
AD=Re+sathi;
%Distance from user's position on earth surface to the satellite
Bx=sqrt(AB^2+AD^2-2*AB*AD*cos(x1*pi/180));
%height of eye
tx=((acos((AD^2-AB^2-Bx^2)/(2*AB*Bx))*180/pi)-180)*(-1)-90;
```

## APPENDIX E RANGERINGS

```
function [Latone1]=rangering11(Re,sathi,Longsat1);

%Will calc and give the range ring display for satellite at a certain
%lat/long
Longone=Longsat1;
%Satellite distance from center of the earth in kilometers
rone=42182;
%Distance Satellite from User Position
Qone=41694;
%Latitude for outter edge of projection
angleone=asin(Qone/rone)*180/pi;
Latone1=angleone;
Latone2=-1*angleone;
%Longituded for outter edge of projection
Longone1=Longone+angleone;
Longone2=Longone-angleone;
angleone2=angleone*60*1.85;
if Longone2 > 180
    Longone2=360-Longone2;
elseif Longone2 < -180
    Longone2=360+Longone2;
else
    Longone2=Longone2;
end
```



## Appendix F TRYCATCH

```
function [datagrid,refvec,Lat1,Long1,Lat2,Long2]=trycatch(Latuser,Longuser)

Lat1=floor(Latuser)-1;
Lat2=ceil(Latuser)+1;
Long1=floor(Longuser)-1;
Long2=ceil(Longuser)+1;

try
    [datagrid,refvec] = dted(pwd, 5, [Lat1 Lat2],[Long1 Long2]);
catch
    try
        Lat1=floor(Latuser)-1;
        Lat2=ceil(Latuser)+1;
        Long1=floor(Longuser);
        Long2=ceil(Longuser)+1;
        [datagrid,refvec] = dted(pwd, 5, [Lat1 Lat2],[Long1 Long2]);
    catch
        Lat1=floor(Latuser)-1;
        Lat2=ceil(Latuser)+1;
        Long1=floor(Longuser)-1;
        Long2=ceil(Longuser);
        [datagrid,refvec] = dted(pwd, 5, [Lat1 Lat2],[Long1 Long2]);
    end
end
```

## Appendix G DATAGRIDFIX

```
function [newdatagrid]=fixeddatagrid(datagrid)

%used to get rid of all the NaN's in the matrix datagrid

y=length(datagrid);
MM=1;
newdatagrid=[];
for datagridx=1:1:y
    NN=1;
    ydatagrid=[];
    for datagridy=1:1:y
        DATAH=(datagrid(NN,MM));
        bbb=isnan(DATAH);
        if bbb==1
            datahh=0;
        else
            datahh=DATAH;
        end
        ydatagrid=[ydatagrid,[datahh]];
        NN=NN+1;
    end
    MM=MM+1;
    newdatagrid=[newdatagrid,[ydatagrid]];
end
```

## BIBLIOGRAPHY

- Department of Transportation & Federal Aviation Administration, "Wide Area Augmentation System Geostationary Communication and Control Segment (GCCS)," Performance Specification FAA-E-2963, 27 May 2004.
- Ericson, S., Hegarty, C., & Sandhoo, K. S., "Assessment of the use of WAAS for land applications," MITRE: Center for Advanced Aviation System Development, September 1999, MITRE PRODUCT MP 98W0000083.
- Enge, Per. "WAAS Messaging System: Data Rate, Capacity, and Forward Error Correction," Stanford University, December 1996.
- Federal Aviation Administration, "Wide Area Augmentation System," Oct. 2004, available at <http://gps.faa.gov/programs/index.htm>. (Accessed 27 April 2005)
- Federal Aviation Administration. "WAAS is commissioned." FAA 21 Nov. 2003, available at <http://gps.faa.gov/Library/Data/SatNav/satnavNovember-2003.htm> (Accessed 27 April 2005)
- Federal Aviation Administration, "Status of Wide Area Augmentation System (WAAS)," July 2004, available at <http://gps.faa.gov/Library/Data/waas/WAASJUL2.DOC> (Accessed 27 April 2005)
- Federal Geographic Data Committee, "What is DGPS?" Oct. 1999, available at <http://tsc.wes.army.mil/gps/dgps.htm> (Accessed 27 April 2005)
- Garmin Ltd., "What is GPS?" 1996-2005, available at <http://www.garmin.com/aboutGPS/> (Accessed 27 April 2005)
- Johnson, G., Hartnett, R., Swaszek, P., "DGPS and WAAS Maritime Accuracy and Availability Studies." Jun 2004.
- Lejeune, R., El-Arini, M., "An Ionospheric Grid for WAAS Based on the Minimum Mean Square Error Estimator," The MITRE Corp.
- Loh, R., Wullschelger, V., "The U.S. Wide-Area Augmentation System (WAAS)," Navigation: Journal of the Institute of Navigation. Vol 42, NO 3, Fall 1995
- Mathworks, "Creating Graphical User Interfaces," 1994-2005. Available at [http://www.mathworks.com/access/helpdesk/help/techdoc/creating\\_guis/creating\\_guis.html](http://www.mathworks.com/access/helpdesk/help/techdoc/creating_guis/creating_guis.html) (Accessed 27 April 2005)
- Mathworks, "Mapping Toolbox: DTED," available at <http://www-ccs.ucsd.edu/matlab/toolbox/map/dted.html> (Accessed 27 April 2005)

METRIC, "Performance Specification - DIGITAL TERRAIN ELEVATION DATA (DTED)" MIL-PRF-89020B, 23 May 2000.

Pike, J., "Digital Terrain Elevation Data [DTED]," Jan 2000, available at <http://www.fas.org/irp/program/core/dted.htm> (Accessed 27 April 2005)

Tsui, Y., and Bao J., "Fundamentals of Global Positioning System Receivers, A Software Approach." New York, NY: John Wiley and Sons, Inc. 2000.