UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

SENSITIVITY ANALYSIS OF CODE CARRIER COHERENCE IN THE GPS SATELLITE FLEET TO REDUCE OVERBOUNDING IN WAAS TO INCREASE AVAILABILITY WHILE MAINTAINING INTEGRITY

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

DOCTOR OF PHILOSOPHY

By

CHAD S. SHERRELL
Norman, Oklahoma
2016

SENSITIVITY ANALYSIS OF CODE CARRIER COHERENCE IN THE GPS SATELLITE FLEET TO REDUCE OVERBOUNDING IN WAAS TO INCREASE AVAILABILITY WHILE MAINTAINING INTEGRITY

A DISSERTATION APPROVED FOR THE
DEPARTMENT OF ENGINEERING

BY

To my grandma, Joy Lee Green, who never gave up on me and never allowed me give up.

Acknowledgements

Eric Altshuler

Technical consultation.

Emily McCord

Prototype runs.

Table of Contents

List of Tables

List of Figures

Acronyms and Abbreviations

AMQP	Advanced Message Queuing Protocol	21
ASCII	American Standard Code for Information Interchange	22
CRC	Cyclic Redundancy Check	2
FAA	Federal Aviation Administration	ix
GEO	Geostationary Earth Orbit	2
GPS	Global Positioning System	2
GUS	GEO Uplink Subsystem	2
HDF	Hierarchical Data Format	24
NASE	National Airway Systems Engineering	ix
OLTP	On Line Transaction Processing	22
OLAP	On Line Analytical Processing	22
OSP	Operational System Parameter	1
SOG	Satellite Operation Group	21
WAAS	Wide-Area Augmentation System	ix

Abstract

SENSITIVITY ANALYSIS OF CODE CARRIER COHERENCE IN THE GPS SATELLITE FLEET TO REDUCE OVERBOUNDING IN WAAS TO INCREASE AVAILABILITY WHILE MAINTAINING INTEGRITY

Chad S. Sherrell, Ph.D.
The University of Oklahoma, 2016

Supervisors: James J. Sluss, Jr.
John W. Dyer
John E. Fagan

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. $\sin^2(\alpha) + \cos^2(\beta) = 1$. If you read this text, you will get no information $E = mc^2$. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. $\sqrt[n]{a} \cdot \sqrt[n]{b} = \sqrt[n]{ab}$. This text should contain all letters of the alphabet and it should be written in of the original language. $\frac{\sqrt[n]{a}}{\sqrt[n]{b}} = \sqrt[n]{\frac{a}{b}}$. There is no need for special content, but the length of words should match the language. $a\sqrt[n]{b} = \sqrt[n]{a^n b}$.

The Wide-Area Augmentation System (WAAS) is enhanced and maintained by National Airway Systems Engineering (NASE) at the Federal Aviation Administration (FAA).

Chapter 1

Introduction

Things I need.

PRN SVN Mapping for the last five years. Assign to Hoang.

First sentence what this dissertation has achieved. 30,000ft view of WAAS and the Problem. Pros and Cons of WAAS and the specific area I am looking at.

Paragraph 1: What is the problem?

Not more than 3-4 sentences telling the reader what the problem is, in as simple English as possible

Paragraph 2: Why is the problem hard?

What has eluded us in solving it? What does the literature say about this problem? What are the obstacles/challenges? Why is it non-trivial?

Paragraph 3: What is your approach/result to solving this problem?

How come you solved it? Think of this as your startling or sit up and take notice claims that your thesis will plan to prove/demonstrate

Paragraph 4: What is the consequence of your approach?

So, now that youve made me sit up and take notice, what is the impact? What does your approach/result enable?

WAAS has ≈2000 Operational System Parameter (OSP) values that define minimum and maximum limits, action thresholds, and timeouts. The parameters are use to control logic throughout the WAAS system. There is always a balance between usability and safety. For the FAA these are in the terms availability/continuity for usability and integrity for safety. If the system remains off then this is the highest level of integrity, meaning if the user does not use the Global Positioning System (GPS) then they are safe against all GPS related threats. This would not make a practical system, so it led the engineers developing the WAAS application to use values that would allow the system to be usable, but were significantly conservative to protect the user against GPS related threats. At the inception of WAAS there was insufficient empirical data to appropriately set many of the integrity bounding limits. At least one value was stated as being grossly overbounded and that the empirical evidence to set it correctly would require many years of data. The NASE organization has now accumulated the several years of data, but there is currently no analytical system in place to process this volume of data so that updated OSP values can be set.

Currently in this research effort the beginnings of a system have been created that can be used as an analytics platform for getting the varying data formats and elements into a common format where meaningful analysis can be performed. Every component in the system is purposely selected or designed to take data from its rawest form and process it into a format that can be utilized, manipulated and assessed. As of this writing the system can process NovAtel GUST, G-II and G-III GPS receiver binary log files. The software can identify receiver log messages, Cyclic Redundancy Check (CRC) check that the messages is valid and break each message type into its constituent elements for storing and further analysis. This has been fully demonstrated end to end on a Geostationary Earth Orbit (GEO) Satellite monitoring system in development. This system is currently monitoring one GEO satellite at one GEO Uplink Subsystem (GUS) site but will be expanded to four GEOs at 8 GUS sites in the near future. It process data from the WAAS application receiver and a second receiver used for fault isolation. The system is logging ≈1213 data elements per second from only two message types and this number will grow as more message types are recorded to the database.

Chapter 2

Background and Literature Review

2.1 GPS Overview

Chapter 1 introduced LAAS, which is based on the Global Positioning System developed by the U.S. Department of Defense (DOD). GPS is a remarkable system for finding one's location and its success is largely due to the modest needs of the average user. Standard GPS accuracy is approximately 15 meters[3] and this level of accuracy is quite satisfactory for many applications, including land navigation, which is the primary consumer application for GPS. However, when GPS accuracy is discussed it is generally in terms of horizontal accuracy. Vertical accuracy is seldom mentioned in the discussion of accuracy and most GPS manufacturers do not publish the vertical accuracy specification. Due to this, a rule-of-thumb has developed that suggests that the vertical accuracy is only half as good as the horizontal accuracy. So, if Standard GPS is accurate to within 15 meters in the horizontal, then it is considered accurate to about 30 meters in the vertical, which is unacceptable for aircraft landings. This is why LAAS is indispensable for a GPS Landing System (GLS). Many different factors affect the precision of GPS which will be explored during this overview of the Global Positioning System.

GPS is comprised of 24 well placed satellites. The satellites orbit the earth in 6 orbital planes that are at a 55^o inclination with 4 satellites per plane such that at least 4 satellites will be above the horizon at any given moment. This allows users the ability to use the system 24 hours a day anywhere on the planet. These satellites are not geo-orbital, but are at a Medium Earth Orbit (MEO) and orbit the earth twice a sidereal day with a speed of 3.9km per second. The basic principal of the

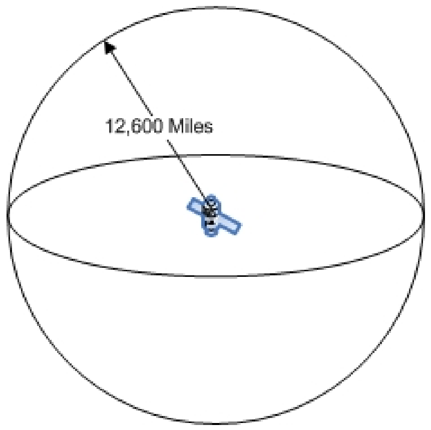


Figure 2.1: GPS Basics: Radius

GPS system is quite simple. A satellite orbits the earth at about 12,600 miles shown in Figure 2.1. The satellite acts as a reference point with a known distance which is called its *range*. How this range is known will be discussed later, but for now the user receiver knows one thing, that it lies on a sphere centered at the satellite with a known radius.

If two satellite are used then this narrows the possible location down to the circle where the two spheres centered at each satellite intersect, shown in Figure 2.2. When a third satellite is introduced it intersects the circle at two points shown in Figure 2.3. Purely speaking a fourth measurement should be used to unambiguously locate a point in space, but for GPS this is not the case. Of the two points referenced, one is normally a sensible answer and the other is either not located on Earth or is moving at an unreasonable velocity (measured in the range of thousands of kilometers per second).

The satellites are used as reference points, but how can something moving high above the earth at a high velocity be used for measuring distance? Every aspect of the GPS constellation is precisely monitored by six earth based monitor stations located throughout the world. Any variance in the orbit, position, or velocity of a satellite is compensated and adjusted for at the master control station, located

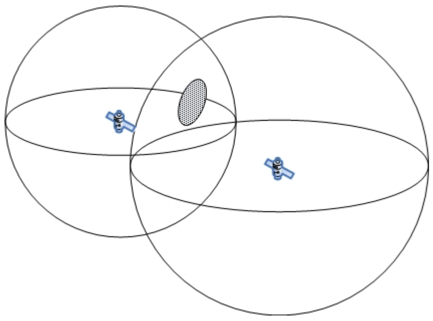


Figure 2.2: GPS Basics: Circle

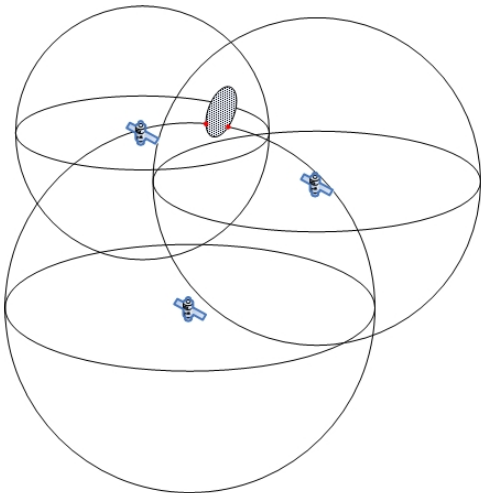


Figure 2.3: GPS Basics: Two Points

in Colorado Springs, Colorado. These constant adjustments are put into a GPS message called the ephemeris message. An *ephemeris* is a table giving the coordinates of a set of celestial bodies at a number of specific times during a given period. This data allows a GPS receiver to know the exact position of a satellite at a given time. Now a method similar to *trilateration* can be used to determine the GPS receiver's range or distance to the satellite. Trilateration uses the known locations of three or more reference points, and the measured distance between the subject and each reference point to determine the subject's location. In the case of GPS the distance is measured using the time-of-arrival (TOA) of a ranging signal called the *pseudorandom code* (PRN). The code is a very complex set of digital information that is called pseudorandom because it resembles random noise. It is repeated every millisecond. When the receiver receives the pseudorandom code it takes the signal propagation time multiplied by the speed of light to get the range to the satellite. This is simple in concept, but the timing has to be near perfect. If the time is off by even 1/1000 of a second the range could be off by 300,000 meters. The satellites actually keep track of time using atomic clocks. Each satellite is equipped with four atomic clocks to precisely track time, but atomic clocks are very costly and impractical for producing low cost GPS receivers. Instead manufacturers use a less precise, less expensive crystal clock; but now have to overcome the error introduced by the receiver clock. Thus time becomes a fourth unknown. This is resolved with an additional satellite measurement. If three measurement made with perfect time can be used to produce a position solution then four inexact measurements can be used to produce the same solution while solving for the receiver clock offset. Once the receiver's clock has been adjusted then not only will the receiver produce an accurate position, but it also works as a functional atomic clock. Theses techniques provide a more accurate range to satellites, but these ranges still contain errors; therefore they are referred to as *pseudoranges*. The errors can be categorized as follows:

- Ephemeris (< 1 meter Error)

  As mentioned earlier an ephemeris is a table of the coordinates of a celestial body at a point in time. This of course uses a model to predict where the satellite will be, but the forces acting upon the satellite don't always conform perfectly to the model which does introduce some error. Since the ephemeris data is constantly being adjusted the error introduced is small and can be removed by differential GPS between two observations of short separation.

- Satellite Clock Errors (< 1 meter Error)

  Satellites use atomic clocks to precisely monitor time, but even atomic clocks are not perfect. Since the pseudorange calculation is based on time any discrepancy in the clock will introduce error. This can also be removed with differential GPS.

- Receiver Clock Errors (1 meter $<$ Error $<$ 2 meters)

  A GPS Receiver uses a crystal clock which is much less accurate than an atomic clock; consequently additional timing errors exist. The Receiver Clock produces more error than the Satellite Clock and unfortunately can not be removed with differential GPS, but is treated as another unknown during the estimation process.

- Ionosphere / Troposphere ($\approx$4 meters)

  The atmosphere can be broken down into many layers but the two that most affect GPS are the Ionosphere and the Troposphere. The Ionosphere is a dispersive medium, therefore the GPS signal is bent and changes speed as it passes through the ionosphere. Though the bending is generally negligible, the ionosphere increases the speed of the carrier phase and slows down the pseudorandom code, so this source of error must be taken into consideration. The troposphere, on the other hand, is a non dispersive medium, so the carrier phase and pseudorandom code are delayed by the same amount increasing the measured distance to the satellite from the actual distance.[1].

- GPS Satellite Geometry

  A Satellite's geometry with respect to other satellites also plays a role in the position solution. The closer the satellites are to one another the poorer the position solution and the more spread out the better the position solution. The Dilution of Precision (DOP) is a number that represents the acceptability of the GPS geometry. It can further be broken down into a horizontal and vertical component known as the Horizontal Dilution of Precision (HDOP) and Vertical Dilution of Precision (VDOP).

- Multipath

  Multipath is a source of error that is introduced when the GPS signal is received by the GPS receiver from different paths; one signal may take the direct path and another signal be reflected off of some structure. This is the same occurrence that causes ghosting on a television set. LAAS reduces the potential of this error source by using specialized GPS Multipath Limiting Antennas (MLA) and a priori siting analysis to avoid multipath generating structures.

- Selective Availability (SA) ($>$ 10 Meters)

  The last error that will be mentioned is artificial error. The U.S. Military purposely skewed the GPS signal to artificially inflate the error in the system so that it could not be used against them. When Selective Availability was active it was the most substantial source of error, but the use of differential GPS could nearly eliminate SA's effectiveness. The use of SA was discontinued on May 1, 2000.

2.2 Differential GPS (DGPS)

Differential GPS is the method of using a GPS receiver at a fixed known reference point to determine the errors in GPS ranging. To do this, the GPS receiver's antenna position is accurately surveyed. Next, the difference is calculated between the pseudorange internally measured by the GPS receiver and the pseudorange calculated from a satellite's current GPS almanac, ephemeris information, and the surveyed location. The difference between the measured pseudorange and the calculated pseudorange is the *pseudorange correction* which is the distance that the local user should adjust a respective satellite's pseudorange by to achieve a minimum error position solution. The pseudorange correction is a lump sum adjustment for multiple sources of error which include ephemeris, satellite and receiver clocks, ionosphere and troposphere, and even Selective Availability. However, the pseudorange correction can only be computed by knowing the precise location of the reference receiver antenna. Another thing to note is that most Differential GPS systems make corrections in the range domain by adjusting the pseudorange measurement of each satellite in view. An alternative approach is to adjust the system in the position domain. This would involve calculating a position solution for the reference point then calculating a position correction vector. This is generally not done due to the fact that position solutions are calculated by manufacturers using proprietary methods, and their behavior is not known a priori.

The many different Differential GPS solutions that have been developed fall broadly into two categories: post-processed and real time. Surveyors are a primary user of the non-real time post-processing category. They collect GPS data from known reference points, which they call *benchmarks*, and the point needing to be surveyed. The distance between the benchmark and the survey point is called the *baseline*, and it is generally preferred to use the benchmark with the shortest baseline to the survey point. After about 15 minutes to an hour of collecting data at both locations they can post process the data with GPS utilities to produce a position solution accurate to within a centimeter[1]. On the other hand, real-time differential GPS systems acquire the data from the reference points, calculate the pseudorange corrections, and broadcast them out at a rate of 2 to 5 Hertz to be used in navigation applications. Since LAAS falls into the real time category, it and a few other prominent real time DGPS systems will be discussed.

One of the first systems developed actually used the acronym DGPS, but sometimes also goes by NDGPS (Nationwide Differential Global Positioning System) to distinguish DGPS the concept from DGPS the system. The NDGPS is used by the U.S. Coast Guard in harbor regions to give ships better position solutions. Even though the name implies national coverage is it limited to coastal regions. Europe and Canada also have similar solution for their heavily utilized maritime locations. There are even commercial companies like VERIPOS, StarFire, and OmniSTAR that provide Differential GPS services.

Most of the DGPS systems developed had a very specific function or operated in such a localized area that it precluded widespread use. The FAA wanted to capitalize on the benefits of DGPS while promoting a broad coverage area. To that end they designed the Wide Area Augmentation System. WAAS, as shown in Figure 2.4, established reference points throughout the contiguous United States. These fixed reference points were used to establish pseudorange corrections for most of North America. The WAAS corrections are then sent to a geo-orbital satellite and broadcast to WAAS-enabled GPS receivers. WAAS improved the accuracy of GPS to less than 2 meters in the horizontal and less than 3 meters in the vertical. This was a significant enhancement to GPS; an improved GPS position solution free for the end user. GPS manufacturers adopted this readily and now WAAS-enabled GPS receivers are commonplace. A major shortcoming with WAAS is its confinement to

[1]Utilizes GPS L1/L2 survey equipment

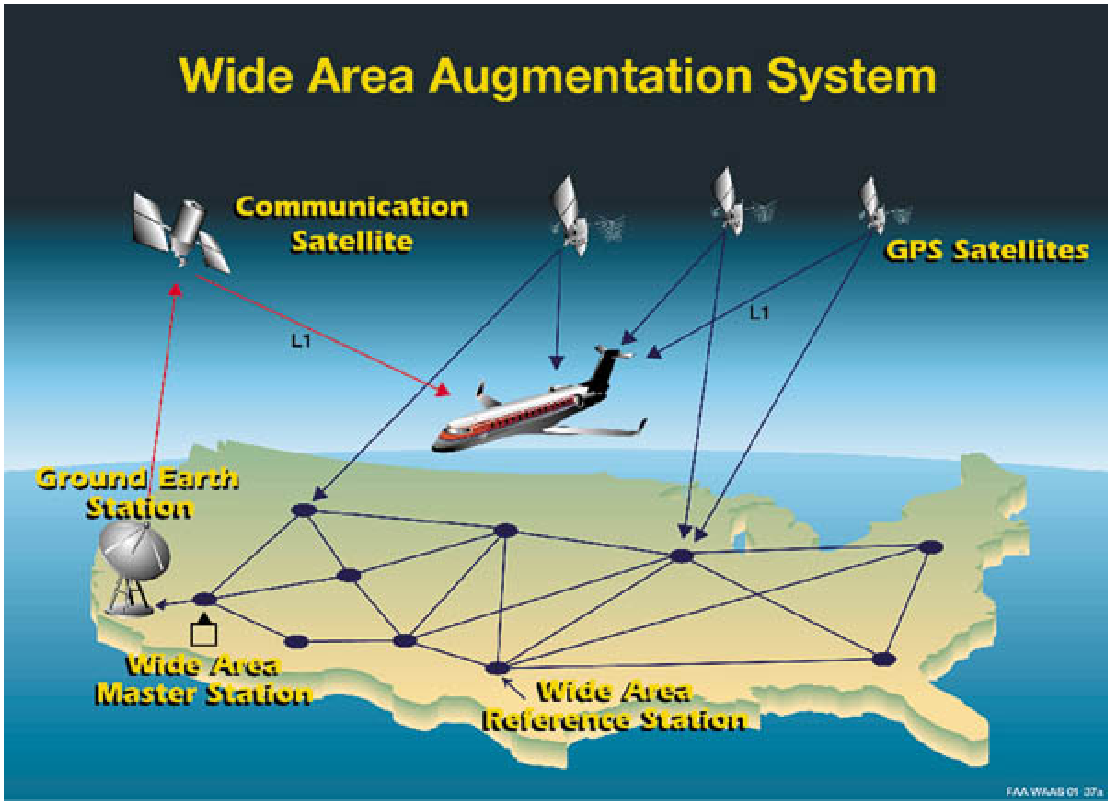


Figure 2.4: Wide Area Augmentation System[4]

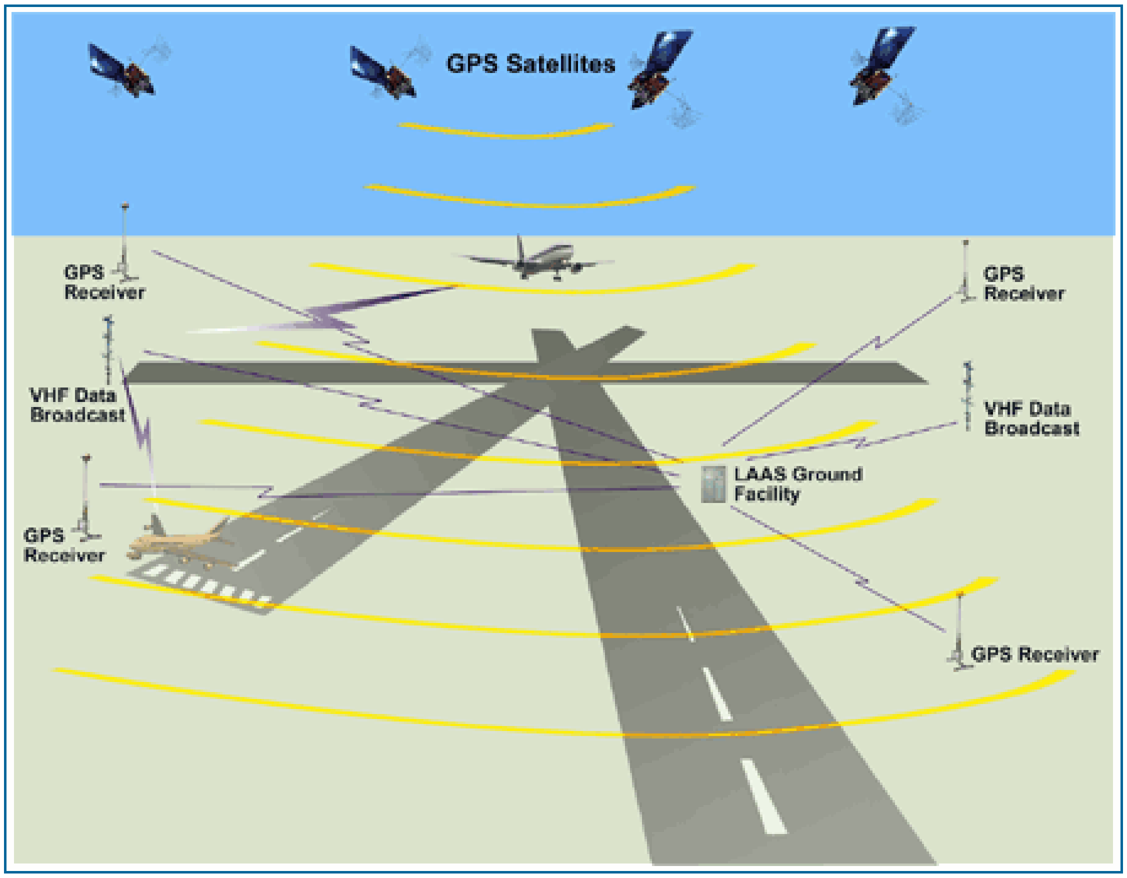


Figure 2.5: Local Area Augmentation System[2]

North America. Additional Reference Receivers are being installed in Alaska and Mexico, but this is still a United States / North America locale. Large air carriers need GPS augmentation internationally. Furthermore, WAAS accuracy is not as accurate as some other Differential GPS systems due to its broad coverage area. Because of this the FAA does not consider its accuracy and integrity acceptable for instrument landings beyond CAT I.

The Local Area Augmentation System was initiated by the FAA to provide an accurate real time differential GPS solution . The LAAS concept follows that of WAAS, but the reference receivers are generally located within the airport environment as can be seen in Figure 2.5. Further, the differential GPS corrections are only usable within the VDB broadcast range, which is about 20 to 30 miles. By limiting the operating region LAAS is able to deliver accuracy under 1 meter in both the horizontal and vertical axis[2]. A further benefit is that LAAS is only dependent on the GPS fleet; therefore it can be used anywhere in the world. The FAA formalized the interfaces and many of the algorithms that should be used in a LGF. Components of a software architecture for a Local Area Augmentation System will be examined in the remainder of this thesis, including the host architecture, message formats, and message broadcasting.

Table 2.1: WAAS Sites

City	**ICAO airport code**	**Antenna 1**	**Antenna 2**	**Antenna 3**
Bethel, Alaska	PABE	60.787916486N 161.841724416W, 52.203 m	60.787897064N 161.841663857W, 52.204 m	60.787881127N 161.841728605W, 52.198 m
Billings, Montana	KBIL	45.803707088N 108.539722283W, 1112.261 m	45.803716383N 108.539780649W, 1112.266 m	45.803756811N 108.539680968W, 1112.255 m
Barrow, Alaska	PABR	71.282765883N 156.789923397W, 15.577 m	71.282798595N 156.789965306W, 15.589 m	71.282793925N 156.789856228W, 15.577 m
Cold Bay, Alaska	PACD	55.200334771N 162.718472052W, 53.648 m	55.200394330N 162.718489390W, 53.652 m	55.200400493N 162.718623936W, 53.657 m
Fairbanks, Alaska	PAFA	64.809630987N 147.847339789W, 149.891 m	64.809681435N 147.847491409W, 149.897 m	64.809748030N 147.847379206W, 149.876 m
Honolulu, Hawaii	PHNL	21.312988930N 157.920824884W, 24.678 m	21.312645960N 157.920980760W, 25.022 m	21.312714586N 157.920825156W, 25.067 m
Juneau, Alaska	PAJN	58.362575024N 134.585705943W, 16.024 m	58.362469451N 134.585487326W, 16.029 m	58.362545895N 134.585292259W, 16.020 m
Mrida, Yucatn	MMMD	20.931909130N 089.662840352W, 29.133 m	20.931901399N 089.662887739W, 29.171 m	20.931946482N 089.662890840W, 29.168 m
Mexico City	MMMX	19.431653203N 099.068389471W, 2236.638 m	19.431676477N 099.068348099W, 2236.625 m	19.431629899N 099.068430820W, 2236.652 m
Puerto Vallarta, Jalisco	MMPR	20.679003359N 105.249202871W, 10.973 m	20.679041461N 105.249177972W, 11.269 m	20.679059454N 105.249221363W, 10.990 m
San Jos del Cabo, Baja California Sur	MMSD	23.160445938N 109.717646195W, 104.297 m	23.160383141N 109.717652895W, 104.285 m	23.160419201N 109.717704568W, 104.277 m
Tapachula, Chiapas	MMTP	14.791366074N 092.367999089W, 54.962 m	14.791334042N 092.367965119W, 54.950 m	14.791319966N 092.368009440W, 54.855 m
Kotzebue, Alaska	PAOT	66.887333160N 162.611372024W, 10.911 m	66.887368005N 162.611390215W, 10.909 m	66.887356742N 162.611304386W, 10.913 m
Iqaluit, Nunavut	CYFB	63.731490169N 068.543181586W, 10.022 m	63.731464001N 068.543402553W, 9.957 m	63.731386362N 068.543596671W, 10.014 m
Gander, Newfoundland and Labrador	CYQX	48.966489496N 054.597631164W, 146.888 m	48.966447606N 054.597532034W, 146.887 m	48.966406383N 054.597433025W, 146.899 m
Winnipeg, Manitoba	CYWG	49.900574663N 097.259396222W, 222.042 m	49.900677586N 097.259217224W, 222.051 m	49.900568446N 097.259226893W, 222.045 m
Goose Bay, Newfoundland and Labrador	CYYR	53.308646665N 060.419467188W, 37.830 m	53.308713007N 060.419365697W, 37.844 m	53.308803193N 060.419371104W, 37.853 m
Albuquerque, New Mexico	KZAB	35.173575457N 106.567349162W, 1620.117 m	35.173574799N 106.567287780W, 1620.181 m	35.173532365N 106.567287878W, 1620.164 m
Anchorage, Alaska	PAZA	61.229202467N 149.780248917W, 80.660 m	61.229118812N 149.780422686W, 80.653 m	61.229202391N 149.780423003W, 80.648 m
Aurora, Illinois	KZAU	41.782657876N 088.331335953W, 195.918 m	41.782595526N 088.331334442W, 195.921 m	41.782596464N 088.331253756W, 195.926 m
Nashua, New Hampshire	KZBW	42.735720140N 071.480425027W, 39.125 m	42.735724128N 071.480358015W, 39.151 m	42.735671312N 071.480352294W, 39.147 m
Leesburg, Virginia	KZDC	39.101595603N 077.542745736W, 80.084 m	39.101523590N 077.542730286W, 80.080 m	39.101548982N 077.542774296W, 80.092 m
Longmont, Colorado	KZDV	40.187303318N 105.127223496W, 1541.399 m	40.187303552N 105.127154188W, 1541.391 m	40.187253096N 105.127167214W, 1541.377 m
Fort Worth, Texas	KZFW	32.830649739N 097.066471191W, 155.617 m	32.830596303N 097.066523654W, 155.576 m	32.830598335N 097.066470282W, 155.620 m
Houston, Texas	KZHU	29.961896297N 095.331425748W, 10.908 m	29.961831785N 095.331449752W, 10.974 m	29.961773563N 095.331512004W, 10.958 m
Hilliard, Florida	KZJX	30.698859379N 081.908184568W, 2.149 m	30.698823791N 081.908152480W, 2.140 m	30.698791217N 081.908198025W, 2.135 m
Olathe, Kansas	KZKC	38.880159315N 094.790833106W, 305.904 m	38.880160009N 094.790643592W, 305.903 m	38.880101810N 094.790710614W, 305.636 m
Palmdale, California	KZLA	34.603517830N 118.083893947W, 763.521 m	34.603517881N 118.083828796W, 763.520 m	34.603473855N 118.083893956W, 763.598 m
Salt Lake City, Utah	KZLC	40.786043564N 111.952176782W, 1287.421 m	40.785990178N 111.952176149W, 1287.416 m	40.785990067N 111.952122320W, 1287.423 m
Miami, Florida	KZMA	25.824611968N 080.319189364W, -7.579 m	25.824659706N 080.319315758W, -8.207 m	25.824661752N 080.319234381W, -7.861 m
Memphis, Tennessee	KZME	35.067394005N 089.955369299W, 68.609 m	35.067437537N 089.955368937W, 68.883 m	35.067439374N 089.955436864W, 68.871 m
Farmington, Minnesota	KZMP	44.637463181N 093.152084552W, 262.679 m	44.637463059N 093.152011267W, 262.693 m	44.637407004N 093.152022108W, 262.628 m
Ronkonkoma, New York	KZNY	40.784328238N 073.097164869W, 6.457 m	40.784275495N 073.097154931W, 5.930 m	40.784275925N 073.097223653W, 5.936 m
Fremont, California	KZOA	37.543053122N 122.015945899W, -3.497 m	37.543025498N 122.015892540W, -3.481 m	37.542981164N 122.015929270W, -3.400 m
Oberlin, Ohio	KZOB	41.297154278N 082.206443927W, 223.689 m	41.297166589N 082.206351733W, 225.187 m	41.297086827N 082.206379312W, 223.468 m
Auburn, Washington	KZSE	47.286993478N 122.188372098W, 82.112 m	47.286907917N 122.188382169W, 82.168 m	47.286856213N 122.188363949W, 82.105 m
San Juan, Puerto Rico	TJZS	18.431335686N 065.993476761W, -28.062 m	18.431218583N 065.993514086W, -28.047 m	18.431198889N 065.993448100W, -28.108 m
Hampton, Georgia	KZTL	33.379688402N 084.296725378W, 261.138 m	33.379691546N 084.296656313W, 261.126 m	33.379634831N 084.296652682W, 261.161 m

Chapter 3

Methodology

3.1 Solution Stack

Reproducibility is one of the main principles of the scientific method and within the FAA having tools and processes that can reliably reproduce results is paramount for a safety of life application. In this section the solution stack will be introduced. And from the ground up the applications selected and process implemented are to ensure results can be reproduced by those that have access to the data store.

A solution stack or software stack is a set of software subsystems or components needed to create a complete platform such that no additional software is needed to support applications. Applications are said to ”run on” or ”run on top of” the resulting platform.

This section covers the following components of the solution stack: virtualization platform, operating system, orchestration, web server, databases, programming language and analytics environment. Further effort is required for a few remaining components of the final solution stack, mostly needed for scaling. The remaining components are orchestration, job scheduling, workload management and horizontal scaling.

3.1.1 Virtualization

Server scaling can come in two varieties: horizontal and vertical. Horizontal scaling is adding more server nodes to an application to handle the additional workload demand, while vertical scaling or scaling up is to add more resources to one node. This solution stack uses virtual machines to scale horizontally; to handle an increase in workload, as well as provide high availability, additional virtual machines can be provisioned and when the demand subsides the virtual machines can be deallocated. The use of virtual machines can streamline development since they can repeatedly be created and destroyed to test out configurations.

Vagrant with VirtualBox Provider

The first component of the solution stack is *Vagrant.* Its primary use is to create and configure virtual development environments. It also helps manage those virtual environments with a few very simple commands. The advantage of using this product is that is can be used by anyone on any platform that Vagrant supports to bring up an identical working environment. Through a Vagrant configuration file it can even constrain the virtual image to a specific version or versions of a image. It also has the capability to automatically install all the additional required applications in the solution stack so that when the system boots up the first time all required software has been installed and configurations have been completed.

This solution stack currently uses Oracle VirtualBox, but it is compatible with other virtualization software like VMware and KVM. The final virtual machines will run on VMWare in a blade cluster.

Vagrant Lifecycle Management

Command: vagrant init [box-name] [box-url]

By default Vagrant will use the containing directory of the Vagrant initialization file, called Vagrantfile, for the virtual machine name. To get started with Vagrant, first create a new directory and change into that directory.

```
mkdir WAASAnalyticsPlatform && cd WAASAnalyticsPlatform
```

The following commands initialize the current directory to be a Vagrant environment by creating the initial Vagrantfile. The Vagrant virtual image is known as a box. If a first argument is given, it configure the Vagrantfile for that type of box.

```
# Example: Create an Ubuntu 14.04 virtual machine.
vagrant init ubuntu/trusty64
# Example: Create a CoreOS virtual machine.
# Exit the WAASAnalyticsPlatform folder
cd ..
# Checkout a CoreOS Vagrantfile from github.
git clone https://github.com/coreos/coreos-vagrant/
# This will create a coreos-vagrant directory in the
# current directory. Rename this directory to the name
# that the virtual machine will be known as.
mv coreos-vagrant CoreOS-Node01
```

Note: Unless otherwise stated the following commands must be ran in the folder containing the Vagrantfile and the command will apply to that environment.

Command: vagrant up

This command creates and configures guest machines according to your Vagrantfile.

This is the single most important command in Vagrant, since it is how any Vagrant machine is created. Anyone using Vagrant must use this command on a day-to-day basis.

Command: vagrant halt

This command shuts down the running machine Vagrant is managing.

Vagrant will first attempt to gracefully shut down the machine by running the guest OS shutdown mechanism. If this fails, or if the --force flag is specified, Vagrant will effectively just shut off power to the machine.

Command: vagrant ssh

This will SSH into a running Vagrant machine and give you access to a shell.

If a -- (two hyphens) are found on the command line, any arguments after this are passed directly into the ssh executable. This allows you to pass any arbitrary commands to do things such as reverse tunneling down into the ssh program.

Command: vagrant status

This will tell you the state of the machines Vagrant is managing.

It is quite easy, especially once you get comfortable with Vagrant, to forget whether your Vagrant machine is running, suspended, not created, etc. This command tells you the state of the underlying guest machine.

Command: vagrant destroy

This command stops the running machine Vagrant is managing and destroys all resources that were created during the machine creation process. After running this command, your computer should be left at a clean state, as if you never created the guest machine in the first place.

Command: vagrant global-status and Using IDs

To interact with any of the machines, you can go to the directory with the Vagrantfile and run any of the preceding vagrant commands. Vagrant also provides a way to interact with the virtual machines without having to change into the directory. The global-status is the command that lists information and status about all known Vagrant environments on the host machine. The first column is the ID for the machine and the preceding commands all take an ID as an ending argument.

```
vagrant global-status
id       name    provider   state    directory
------------------------------------------------------------------------------------
26d0e63  default virtualbox poweroff /home/csherrell/Vagrant/v-Ubuntu-14.04-Bootstrap
d6e2bc0  default virtualbox poweroff /home/csherrell/Vagrant/v-Ubuntu-14.04-Rabbit-02
fc44416  default virtualbox poweroff /home/csherrell/Vagrant/v-Ubuntu-14.04-Rabbit-01
77c195d  default virtualbox poweroff /home/csherrell/Vagrant/v-Ubuntu-14.04-Rabbit-Cassandra-01
```

This shows that all the virtual machines are off. To start the first virtual machines run the following command.

```
vagrant up 26d0e63
...
vagrant global-status
id       name    provider   state    directory
--------------------------------------------------------------------------------------------
26d0e63  default virtualbox running  /home/csherrell/Vagrant/v-Ubuntu-14.04-Bootstrap
d6e2bc0  default virtualbox poweroff /home/csherrell/Vagrant/v-Ubuntu-14.04-Rabbit-02
fc44416  default virtualbox poweroff /home/csherrell/Vagrant/v-Ubuntu-14.04-Rabbit-01
77c195d  default virtualbox poweroff /home/csherrell/Vagrant/v-Ubuntu-14.04-Rabbit-Cassandra-01
```

To start the remaining virutal machines run the following command.

```
vagrant up d6e2bc0 fc44416 77c195d
...
vagrant global-status
id       name    provider   state   directory
-------------------------------------------------------------------------------------------
26d0e63  default virtualbox running /home/csherrell/Vagrant/v-Ubuntu-14.04-Bootstrap
d6e2bc0  default virtualbox running /home/csherrell/Vagrant/v-Ubuntu-14.04-Rabbit-02
fc44416  default virtualbox running /home/csherrell/Vagrant/v-Ubuntu-14.04-Rabbit-01
77c195d  default virtualbox running /home/csherrell/Vagrant/v-Ubuntu-14.04-Rabbit-Cassandra-01
```

Vagrant Website:

https://www.vagrantup.com/

3.1.2 Software Development

Python 3

Python version 3 is being used for this development effort. The Python programming language guidance is “if you can do exactly what you want with Python 3.x” then you should use it. During a previous phase of development a NovAtel binary message parser was written in Python 2.7 to parse log messages. This plus some other support software written in Python 2.7 was converted to version 3. The new parsing code is completely object oriented and the new classes for handling NovAtel messages where all written in Python 3 and there were no major issues. The only issue that has reoccurred was in integrating the message parsing functionality developed in Python 2.7 into the new classes. Python 3 strings are Unicode by default and now there is clean Unicode/bytes separation. For the average use case this has little to no affect, but when the purpose of the application is to read in a binary data stream and manipulate bytes this has a significant impact. In Python 2.7 strings and bytearray objects could be use interchangeably. Under Python 3 this is no longer the case. Python 3 will throw a run time exception if a string is cast to a bytearray or vica versa. In Python 3 to convert bytes or bytearray objects to ASCII strings the decode method must be called.

```
# Hello World! in Hex
input_buffer = bytearray.fromhex('48656c6c6f20576f726c6421')
ascii_string = 'Test: '
# Append the bytearray to the string
ascii_string += input_buffer[0:]

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-17-f84d039df76b> in <module>()
----> 1 ascii_string += input_buffer[0:]

TypeError: Can't convert 'bytearray' object to str implicitly


# Using the bytearray.decode() method
ascii_string += input_buffer[0:].decode('ascii')
print(ascii_string)
Test: Hello World!
```

This analysis platform uses a producer/consumer architecture. To publish data to a queuing middleware used by the downstream consumers the data must first be serialized. Serialization is the conversion of data structures to a steam of bits to be saved to disk or transported across a communication medium. Under Python 2.7 there is a built-in Python module called pickle that is used for serialization but it was not well optimized, so the msgpack module was use for initial testing. As more involved testing was performed it was discovered that the msgpack module could not serialize all the data elements. Luckily this happened about the same time as the transition to Python 3. The msgpack module has deficiencies and further it has not yet been ported to Python 3. Under Python 2.7 there was an optimized C implementation of the pickle module and under Python 3 the built-in pickle module is the C optimized implementation, so msgpack was dropped all together and pickle was used as the serialization service throughout the software.

Python Website:

https://www.python.org

https://wiki.python.org/moin/Python2orPython3

Messaging / Middleware

Middleware is the software that connects software components or enterprise applications. Middleware is the software layer that lies between the operating system and the applications on each side of a distributed computer network. Typically, it supports complex, distributed business software applications.

This analysis platform uses a distributed producer consumer architecture. Currently this is built upon the RabbitMQ middleware, but the decoupled software architecture will allow for one middleware to be swapped for another or have the ability to use multiple at once. ZeroMQ is another middleware that will be evaluated in the future as other entities in the FAA use it.

RabbitMQ

RabbitMQ is an implementation of the Advanced Message Queuing Protocol (AMQP). AMQP was designed by JPMorgan Chase and is used extensively in the financial industry. Multiple implementation of AMQP have been created, but RabbitMQ is one of the more prestigious implementation. It is used throughout the VMware product line and also used in OpenStack for messaging. It supports several different type of exchanges: Direct, Fanout, Topic, and Headers; and can support persistent queues. The Python library that supports the interaction with RabbitMQ service is called Pika and will be covered later.

ZeroMQ

ZeroMQ is a high-performance asynchronous messaging library, aimed at use in distributed or concurrent applications. It provides a message queue, but unlike message-oriented middleware, like RabbitMQ, ZeroMQ can run without a dedicated message broker. This product will be evaluated in the future and could augment the messaging capabilities in this application. The Satellite Operation Group (SOG) in the FAA use ZeroMQ so some functionality may be required to be able to store and access data in their database.

3.1.3 Databases and Storage Formats

Several databases and storage formats have already been and will be evaluated during this research effort. Databases serve a couple of different purposes. First and foremost they store and retrieve data. The advantage a database has over comma separated American Standard Code for Information Interchange (ASCII) files is that the data comes in as binary and is stored as binary there by avoiding precision being lost when floating point values are converted to ASCII. Another advantage is that databases are useful for performing data analysis. In the past databases were used primarily to do On Line Transaction Processing (OLTP), basically data entry and retrieval transaction processing and run some standard reports. Today new databases support On Line Analytical Processing (OLAP) which provides the capability for complex calculations, trend analysis, and sophisticated data modeling. This application like many modern applications will use polyglot persistence that is to say it will use the best persistence mechanism for the application's needs and that will probably utilize multiple types of storage engines. Beyond the ability to store and analyze data is the requirement to share data. Doing an analysis is one thing, but due to the integrity requirements of a safety of life system other organizations will be evaluating the results, so the input data, final solution and possibly some intermediate values should be save in such a way as to be able to hand over for independent evaluation.

Graphite's Whisper Database

Whisper is a file-based time-series database format for Graphite. Graphite is a web analytics application for time-series data. The upside is that it has good community support and designed specificity for time-series data which is the bulk of the data being used in this research. The downside is that it only support one second resolution. This is acceptable for all the acwaas data that will initially be evaluated. A more advanced time series database may need to be assessed as additional sensors are incorporated that operate faster than 1Hz. Some of the temperature sensors and accelerometers that are currently being evaluated operate at 50Hz.

Graphite Website:

http://graphite.readthedocs.org/en/latest/

https://github.com/graphite-project/whisper

Proofread Completed

PostgreSQL

PostgreSQL is a relational database. Relational databases are probably the most widespread of all database types, but their problem is that they are normally ran on one server and do not scale horizontally, but vertically. Typically this leads to a single point of failure, but there are some system administration paradigms that can support high availability. Also, there is significant overhead for indexing large datasets. Because of there use in data warehousing many of these issues do have solutions or workarounds. Relational databases have been the standard for well over 40 years, so they have there place. This application stack will use a relations database to store small datasets and will be used extensively to development models.

SQLite

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional relational database engine. The code for SQLite is in the public domain and is thus free for use for any purpose, commercial or private.

SQLite is the most widely deployed database in the world. The data is contained in a single file so this would be a solution for sharing data amongst the various people wanting to use the data.

HDF5

Hierarchical Data Format (HDF) version 5 is not a database but a unique technology suite that makes possible the management of extremely large and complex data collections. The HDF5 technology suite includes: A versatile data model that can represent very complex data objects and a wide variety of metadata. The data is structured and remains in a binary format. Matlab uses this a the base format for its MAT-File starting in R2006b. This file format is designed for very large datasets and since the dataset will be contained in one file it is being strongly considered as the format for sharing data with other organizations.

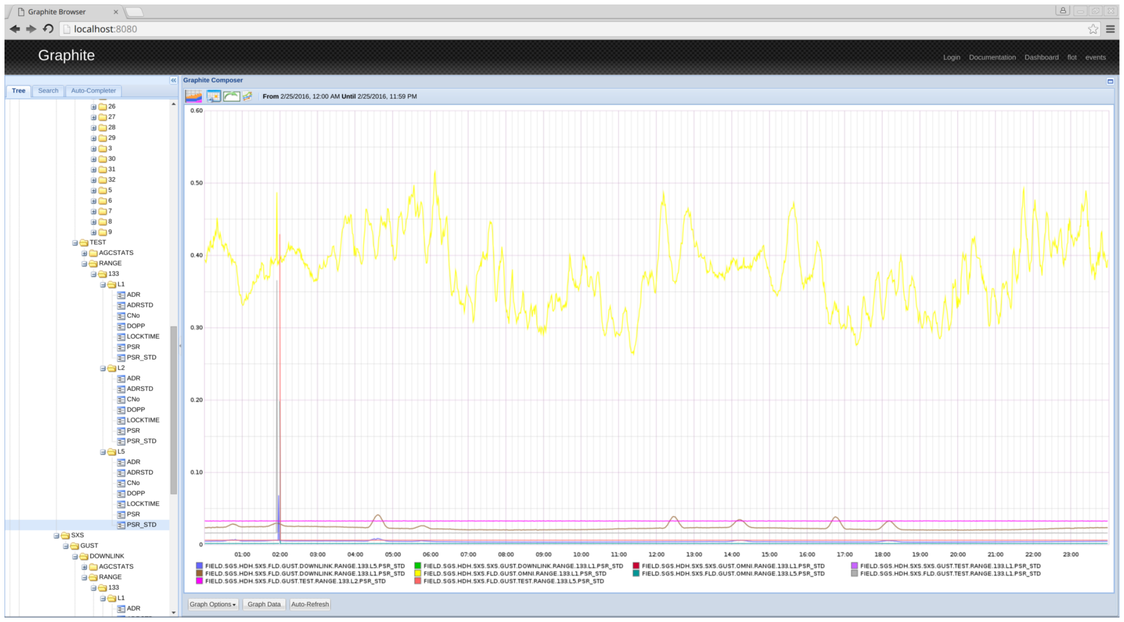
Cassandra

Cassandra is a distributed database produces by Apache designed to handle large amounts of data across many server nodes. It is a big data database.

Chapter 4

Results and Discussion

Currently the graphite database contains 1213 data element. In the plot below each data element is the leaf in the directory tree represented by a file icon. This plot for one the psuedorange and standard deviation from the GPS receiver. Each data element contains a data point for each second of the day, 86400 points. This is just a first cut at getting data loaded into the system, but the design will support many data elements and calculations done between elements.



Graphite Browser
localhost:8080
Graphite
Login Documentation Dashboard flot events
Tree Search Auto-Completer
Graphite Composer
From 2/25/2016, 12:00 AM Until 2/25/2016, 11:59 PM
26 27 28 29 3 30 31 32 5 6 7 8 9
TEST
AGCSTATS
RANGE
133
L1
ADR ADRSTD CNo DOPP LOCKTIME PSR PSR_STD
L2
ADR ADRSTD CNo DOPP LOCKTIME PSR PSR_STD
L5
ADR ADRSTD CNo DOPP LOCKTIME PSR PSR_STD
SXS
GUST
DOWNLINK
AGCSTATS
RANGE
133
L1
ADR
0.60 0.50 0.40 0.30 0.20 0.10 0
01:00 02:00 03:00 04:00 05:00 06:00 07:00 08:00 09:00 10:00 11:00 12:00 13:00 14:00 15:00 16:00 17:00 18:00 19:00 20:00 21:00 22:00 23:00
FIELD.SGS.HDH.SXS.FLD.GUST.DOWNLINK.RANGE.133.L5.PSR_STD
FIELD.SGS.HDH.SXS.FLD.GUST.DOWNLINK.RANGE.133.L1.PSR_STD
FIELD.SGS.HDH.SXS.FLD.GUST.TEST.RANGE.133.L2.PSR_STD
FIELD.SGS.HDH.SXS.SXS.GUST.DOWNLINK.RANGE.133.L1.PSR_STD
FIELD.SGS.HDH.SXS.FLD.GUST.OMNI.RANGE.133.L1.PSR_STD
FIELD.SGS.HDH.SXS.FLD.GUST.TEST.RANGE.133.L5.PSR_STD
FIELD.SGS.HDH.SXS.SXS.GUST.OMNI.RANGE.133.L1.PSR_STD
FIELD.SGS.HDH.SXS.FLD.GUST.OMNI.RANGE.133.L5.PSR_STD
FIELD.SGS.HDH.SXS.SXS.GUST.TEST.RANGE.133.L1.PSR_STD
FIELD.SGS.HDH.SXS.FLD.GUST.TEST.RANGE.133.L1.PSR_STD
Graph Options Graph Data Auto-Refresh

Chapter 5

Summary and Conclusions

There is still a lot I need to get done, but progress is going well. Currently I am having to interleave this into an already overloaded work schedule. I am hoping to take some time off to devote to more writing. I have more done that I have written here, but I just have not had a opportunity to write everything up.

Ideas for future research.

Bibliography

[1] A. El-Rabbany. *Introduction to GPS: the Global Positioning System/.* Artech House, 685 Canton Street, Norwood, MA 02062, 1st edition, 2002.

[2] http://gps.faa.gov/images/L_ARCH_2.gif.

[3] J. McNamara. *GPS For Dummies.* Wiley Publishing, Inc., 111 River Street, Hoboken, NJ 07030-5774, 2004.

[4] http://www.sti.nasa.gov/tto/spinoff1999/webimages/47.jpg.

Appendices

Appendix A

DO-178B

The software levels as defined by DO-178B are:

Level A Software whose anomalous behavior, as shown by the system safety assessment process, would cause or contribute to a failure of system function resulting in a catastrophic failure condition for the aircraft.

Level B Software whose anomalous behaviors, as shown by the system safety assessment process would cause or contribute to a failure of system function resulting in a hazardous/severe-major failure condition for the aircraft.

Level C Software whose anomalous behavior, as shown by the system safety assessment process, would cause or contribute to a failure of system function resulting in a major failure condition for the aircraft.

Level D Software whose anomalous behavior, as shown by the system safety assessment process, would cause or contribute to a failure of system function resulting in a minor failure condition for the aircraft.

Level E Software whose anomalous behavior, as shown by the system safety assessment process, would cause or contribute to a failure of system function with no effect on aircraft operational capability or pilot workload.

Appendix B

Glossary

Reference Receiver (RR) A Reference Receiver is composed of a Helibowl, Dipole, 2 GG12s, and 2 FreeWave Modems.

LAAS Ground Facility (LGF) The LAAS Ground Facility is the facility that equipment needed for LAAS.

LAAS Processing Unit (LPU) The computing platform and software used to produce the LAAS SIS.

Local Area Augmentation System (LAAS) LAAS designates all the components of this particular GBAS system. It includes the Reference Receiver, LAAS Ground Facility, and any other infrastructure requirements.