CORNELL ECE/MAE 4150

Limited Satellite Navigation Solution

Finding a location with fewer than 4 satellites

Karl Gluck, Jasper Schneider, Rui Rick Wu

Fall 2008

FINAL CLASS PROJECT

Abstract

The standard methods used by Global Positioning System (GPS) receivers require at least four satellites to determine the receiver's location. Receivers in urban, forested or other environments with limited sky visibility would benefit from being able to find a reliable and accurate navigation solution with fewer satellites. This paper develops a method combining both satellite Doppler shift and pseudorange solutions to create a navigation solution for a stationary receiver requiring at minimum two satellites. It shows that not only can a reasonably accurate receiver location be determined with fewer than four satellites, but also that the method developed can improve the N-satellite solution found by the pseudorange method.

Contents

1. Introduction

1.1. GPS Basics

GPS, operated by the United States Air Force, is currently the world's only functional positioning system. Between twenty-four and thirty-two satellites orbit Earth twice each sidereal day at an average altitude of 20200 km. These are evenly spaced in six orbital frames to ensure maximum visibility from most points on Earth.

Each satellite transmits orbital information, known as ephemerides, on a specific carrier frequency. From this data, a receiver can determine where the satellites are located in space and measure a pseudorange for each visible satellite. This pseudorange is the addition of the signal distance from receiver to satellite and an error term, common to all satellites tracked by a receiver, created by the imperfect receiver clock. By simultaneously measuring the pseudorange to at least four satellites, the receiver can calculate its 3D position and the clock error term.

There also exists a second, less commonly-used method of navigation based on Doppler shift. In order to receive a satellite's signal, the receiver must tune in to the frequency at which the signal is arriving. Because the receiver and satellite are moving relative to one another, this frequency is Doppler shifted from the L1 carrier frequency on which all civilian GPS transmissions are made. When first turned on, this is done by trial-and-error; however, the result is that once a satellite is detected, it is assigned a value of the internal numerically controlled oscillator (NCO) that is used to receive the signal. This value is, like the pseudorange, a combination of the true frequency and an error term that accounts for receiver oscillator imperfection. By simultaneously measuring the NCO to at least four satellites, the receiver can perform a similar computation as for pseudorange to calculate its location and the error value.

1.2. Motivation

Pseudorange positioning is very effective in suitable environments. Ideally, the antenna has a clear view of the sky, is located above all reflecting surfaces, and is tracking more than four satellites spread far apart in the sky relative to the receiver. In reality, GPS devices are commonly used in big cities or wooded areas where objects obscuring the sky may prevent the receiver from tracking a sufficient number of satellites, or where the satellites available may be in a geometric configuration that magnifies their common errors. Our approach, combining both pseudorange and Doppler shift methods, not only allows a navigation solution to be found with only two satellites, but also creates a more accurate solution for a given number of satellites than pseudorange or Doppler alone.

1.3. Background

1.3.1. Pseudorange Solution

The most common technique implemented in today's civilian GPS receivers for determining the navigation solution is the pseudorange method. The pseudorange is measured by the speed of light multiplied by the difference in transmitted and receive times. The pseudorange can be expressed in terms of the true range in terms of receiver and satellite clock corrections:

$$P^j(t) = \rho^j(t) + c\delta^j(t) - c\delta_R(t) \tag{1.1}$$

where $c\delta^j(t)$ is the satellite clock offset and $c\delta_R(t)$ is the receiver clock offset. Using the ephemerides to find the location of the satellite, we can construct spheres centered at the satellite with the radius equal to the pseudorange. All measurements are made in the ECEF (Earth Centered Earth Fixed) frame. The equation of each sphere can be expressed in terms of the true range:

$$\rho^j = \sqrt{(X^j - X)^2 + (Y^j - Y)^2 + (Z^j - Z)^2} \quad (1.2)$$

where (X^j, Y^j, Z^j) are the positions of the *j*th satellite in the ECEF Frame. (X, Y, Z) is the ECEF position of the receiver. In terms of the measurable quantity pseudorange:

$$P^j(t) - c\delta^j(t) = \sqrt{(X^j - X)^2 + (Y^j - Y)^2 + (Z^j - Z)^2} - c\delta_R(t) \quad (1.3)$$

Pseudorange, satellite clock offset (given in ephemerides), and the satellite positions are known. This leaves four variables: X, Y, Z, and $c\delta_R$. If the receiver can obtain at least four pseudorange measurements, these unknowns can be found. With the Newton-Raphson method, the solution to these non-linear equations is found through iteration. ΔX, ΔY, ΔZ, and δ_R correction values are found after every iteration and update the previous values of X, Y, and Z. ΔX, ΔY, ΔZ, and δ_R are computed as follows:

The delta x vector is found using $\Delta x = A^{-1}\underline{F}$. $\underline{F}$ is a column vector containing (1.3) for each satellite used in the navigation solution. The i[th] row of the A matrix contains the derivatives of the ith row in the F vector. The A matrix is constructed as shown below:

$$A = \begin{pmatrix} \frac{d\rho^1}{dx} & \frac{d\rho^1}{dy} & \frac{d\rho^1}{dz} & -1 \\ \frac{d\rho^2}{dx} & \frac{d\rho^2}{dy} & \frac{d\rho^2}{dz} & -1 \\ \cdots & \cdots & \cdots & \cdots \\ \frac{d\rho^n}{dx} & \frac{d\rho^n}{dy} & \frac{d\rho^n}{dz} & -1 \end{pmatrix} \quad (1.4)$$

Applying this to 1.3, we have:

$$A = \begin{pmatrix} \frac{-(X^1 - X)}{\rho^1} & \frac{-(Y^1 - Y)}{\rho^1} & \frac{-(Z^1 - Z)}{\rho^1} & -1 \\ \frac{-(X^2 - X)}{\rho^2} & \frac{-(Y^2 - Y)}{\rho^2} & \frac{-(Z^2 - Z)}{\rho^2} & -1 \\ \cdots & \cdots & \cdots & \cdots \\ \frac{-(X^n - X)}{\rho^n} & \frac{-(Y^n - Y)}{\rho^n} & \frac{-(Z^n - Z)}{\rho^n} & -1 \end{pmatrix} \quad (1.5)$$

Now, Δx is defined as:

$$\begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \\ c\delta_R \end{pmatrix} = \begin{pmatrix} \frac{-(X^1 - X)}{\rho^1} & \frac{-(Y^1 - Y)}{\rho^1} & \frac{-(Z^1 - Z)}{\rho^1} & -1 \\ \frac{-(X^2 - X)}{\rho^2} & \frac{-(Y^2 - Y)}{\rho^2} & \frac{-(Z^2 - Z)}{\rho^2} & -1 \\ \cdots & \cdots & \cdots & -1 \\ \frac{-(X^n - X)}{\rho^n} & \frac{-(Y^n - Y)}{\rho^n} & \frac{-(Z^n - Z)}{\rho^n} & -1 \end{pmatrix}^{-1} \begin{pmatrix} P^1(t) - c\delta^1 - {\rho^1}_0(t) \\ P^2(t) - c\delta^2 - {\rho^2}_0(t) \\ \cdots \\ P^n(t) - c\delta^n - {\rho^n}_0(t) \end{pmatrix} \quad (1.6)$$

where ${\rho^n}_0$ is the range calculated from the initial guess at the receiver location given by the user. When more than four satellites are used, there will not exist a consistent solution to the system of equations. Thus, we must look for a solution that minimizes the error--the least squares projection. Δx can be computed as follows:

$$\Delta x = (A^T A)^{-1} A^T \mathrm{F} \quad (1.7)$$

Δx is then added to the original guess, giving successive approximations to the true location. Once the magnitude of this difference vector is sufficiently small, i.e. on the order of 10^{-6}, the method terminates and returns the solution (Course Packet, Chapter 7: Navigation Solution).

1.3.2. Doppler Solution

In order to lock on to a satellite signal, it is necessary for all receivers to find the satellite's Doppler shift; thus, no hardware modifications would be necessary to employ it. However, Doppler positioning is not widely used by GPS receivers for several reasons. Primarily, the Doppler shift solution suffers from lack of precision because Doppler shift values are not measured to the same degree of precision as pseudorange. Additionally, the Doppler solution requires a stationary receiver.

The Doppler effect is the rise or fall in the frequency of a propagating wave depending on the relative motion of the transmitting object with respect to the receiver. If the transmitter and receiver are moving toward one another, the frequency received is higher than the frequency transmitted. Likewise, if the pair are getting further apart, the received frequency falls. For GPS satellites and receivers, we can form the Doppler equation as follows:

$$f_R - f^T = \frac{-v_{rad}}{\lambda_{L1}} \tag{1.8}$$

$$\begin{cases} \lambda_{L1} \text{ is the wavelength of the L1 signal (approximately 0.19m)} \\ v_{rad} \text{ is the radial velocity of the satellite with respect to the receiver} \\ f_R \text{ is the received frequency} \\ f^T \text{ is the transmitted frequency} \end{cases}$$

Now we can manipulate this equation and put it in the form of known values and desired unknowns:

$$D = \frac{-v_{rad}}{\lambda_{L1}} \tag{1.9}$$

$$D * \lambda_{L1} = -v^i \cdot \frac{r^i - r_u}{||r^i - r_u||} \tag{1.10}$$

D is the Doppler shift ($f_R - f^T$)

v^i is the velocity vector of the ith satellite

$\left[\frac{(r^i - r_u)}{||r^i - r_u||}\right]$ is the unit vector from the receiver (r_u)to the satellite

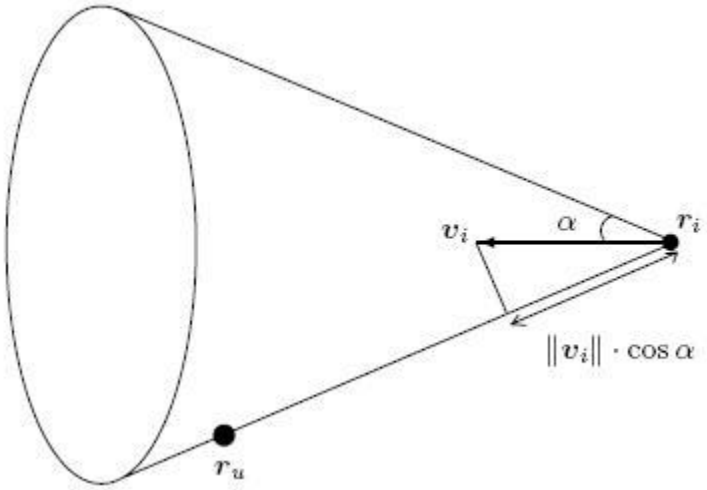
Doppler shift is measurable, the satellite position and velocity vectors can be found from the ephemerides, and λ_{L1} is a known constant. Thus, only the position of the receiver, r_u is unknown. Using cos(x) = x • y / |x||y|, we can rewrite (1.10) as:

$$D * \lambda_{L1} = \cos(\alpha)\, ||v^i|| \tag{1.11}$$

$$a = \arccos\left(\frac{D * \lambda_{L1}}{||v^i||}\right) \tag{1.12}$$

α is the angle between the satellite's velocity vector and the vector between the satellite and the receiver. The angle is a known quantity at any given time, as all other quantities in that equation can be found. This places a constraint on the possible locations: the receiver can only be located where the user to satellite vector forms an angle α with the velocity vector. Geometrically, this represents an infinite cone centered about the velocity vector, with the vertex at the satellite position.

Figure 1.1: Doppler Cone. (Lehtinen 20)



Like the receiver's clock, the oscillator that is used to generate the signal frequency is imperfect. Knowing this, we introduce a new variable for the oscillator error, denoted d. Including this error term, we can find the system of equations to solve for our navigation solution. Factoring in the error and rewriting D * λ_{L1} as $-v_{rad}$, (1.10) becomes.

$$v^i \cdot \frac{r^i - r_u}{||r^i - r_u||} + d - v^i{}_{rad} = 0 \qquad 1.13$$

v_i refers to the i^{th} satellite, where i = 1,2...n. With n >= 4, the system of equations can be solved for x, y, z, and d. Again, this nonlinear system of equations has no closed-form solution and must be solved with an iterative technique such as the Newton-Raphson method.

Congruous to the pseudorange case, we have $\Delta x = A^{-1}F$. The A matrix is defined in the exact same manner as (1.4), except F now contains $v^i \cdot \frac{r^i - r_u}{||r^i - r_u||} + d - v^i{}_{rad} = 0$

1.The derivatives in the A matrix are now significantly more complex than in the pseudorange method; however, (Lehtinen 20) shows that they can be reduced to the following form:

$$A = \begin{pmatrix} \frac{1}{||r^1 - r_u||}\{(r^1 - r_u) \times [(r^1 - r_u) \times v^1]\} & 1 \\ \frac{1}{||r^2 - r_u||}\{(r^2 - r_u) \times [(r^2 - r_u) \times v^2]\} & 1 \\ \cdots & \cdots \\ \frac{1}{||r^n - r_u||}\{(r^n - r_u) \times [(r^n - r_u) \times v^n]\} & 1 \end{pmatrix} \qquad 1.14$$

The Δx vector is then defined as:

$$\begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \\ d \end{pmatrix} = \begin{pmatrix} \frac{1}{||r^1 - r_u||}\{(r^1 - r_u) \times [(r^1 - r_u) \times v^1]\} & 1 \\ \frac{1}{||r^2 - r_u||}\{(r^2 - r_u) \times [(r^2 - r_u) \times v^2]\} & 1 \\ \cdots & 1 \\ \frac{1}{||r^n - r_u||}\{(r^n - r_u) \times [(r^n - r_u) \times v^n]\} & 1 \end{pmatrix}^{-1} \begin{pmatrix} v^1 \cdot \frac{r^1 - r_u}{||r^1 - r_u||} + d - v^1{}_{rad} \\ v^2 \cdot \frac{r^2 - r_u}{||r^2 - r_u||} + d - v^2{}_{rad} \\ \cdots \\ v^n \cdot \frac{r^n - r_u}{||r^n - r_u||} + d - v^n{}_{rad} \end{pmatrix} \qquad 1.25$$

Once again, to find the solution, we simply update the x position vector with Δx until the magnitude of Δx is sufficiently small (Lehtinen 16-29)

.

1.4. Combined Approach

We now have two unique expressions that give us information relating the satellite's position to the receiver position: (1.13) for the Doppler shift and (1.3) for pseudorange. Thus, one satellite can provide two independent nonlinear equations for the receiver solution. For the Newton - Raphson iterative solution, each satellite's contribution to the F vector is:

$$\begin{cases} P^j(t) - c\delta^j - \rho^j{}_0(t) \\ v^j \cdot \dfrac{r^j - r_u}{||r^j - r_u||} + d - v^j_{rad} \end{cases} \quad 1.36$$

Each satellite's contribution to the A matrix is:

$$\begin{cases} \dfrac{-(X^j - X)}{\rho_j} & \dfrac{-(Y^j - Y)}{\rho_j} & \dfrac{-(Z^j - Z)}{\rho_j} & -1 & 0 \\ \dfrac{1}{||r^j - r_u||}\{(r^j - r_u) \times [(r^j - r_u) \times v^j]\} & & & 0 & 1 \end{cases} \quad 1.47$$

The total solution vector that describes the receiver's position, clock error and oscillator drift is:

$$\begin{pmatrix} x \\ y \\ z \\ c\delta_R \\ d \end{pmatrix}$$

The complete terms for the iterative solution are formed by the vertical concatenation of each satellite's contribution to the F vector and A matrix. The solution is then found using 1.7, the expression for the overdetermined solution.

2. Methodology

2.1. Key Points in Solution Formulation

2.1.1. Error Correction

Applying the ideal mathematical solution to a real situation requires that several adjustments be made.

- *Satellite clock offset*: An imperfect satellite clock reports an incorrect GPS timestamp. This can be corrected using satellite ephemerides (by pseudoca.m)
- *Finite signal propagation time*: When the receiver calculates the position of the satellite from ephemerides, it needs to use a time. This time is first obtained using the timestamp reported from the ephemerides; however, once an approximate solution for the receiver's location is found, the propagation delay from satellite to receiver can be used to compute where the satellite was when the satellite actually transmitted the signal, not when the measurement was made by the receiver. calcdiffprop.m calculates the propagation time given a guess of the receiver location, and then ephemposvel.m (an adapted version of findsat.m that also calculates uncorrected satellite velocities) calculates the satellite position at the corrected time.
- *Rotation of the ECEF frame*: During signal propagation, the ECEF frame rotates with the Earth, so the satellites' positions must be corrected for this. ephemposvel.m calculates this change in position due to Earth's rotation.
- *Surface velocity of the receiver due to Earth's rotation*: Because the ECEF frame is rotating, the velocity of the receiver due to Earth's rotation must be accounted for. surfacevelocity.m calculates the receiver's velocity based on the current guess of the receiver location, and this velocity is subtracted from the uncorrected velocity returned by ephemposvel to determine the satellite velocity in the ECEF frame.

2.1.2. Reducing the Number of Variables

As described above, the Doppler shift and pseudorange solutions each have an independent error term. As a result, a complete solution has 5 variables: three for spatial position and one for both the frequency drift of the receiver's internal oscillator for Doppler shift and the clock offset of the receiver's internal timer. In order to attempt a two-satellite solution, it was necessary to reduce this to only four variables.

If the same clock is physically used for both functions, the frequency drift is the derivative of the internal clock offset and these could be related to each other mathematically to reduce 5 unknowns to 4 given more than 1 time sample. However, once the receiver unit used in this investigation begins finding a navigation solution, it continuously attempts to correct its internal clock offset; therefore, the derivative of this reported value is meaningless and this approach cannot be used.

We determined by trial and error that reasonably accurate 2-satellite solutions could be obtained by replacing the frequency drift term by the clock offset times a constant scale factor. As a result the frequency drift is eliminated from the solution vector.

By experimenting with overdetermined solutions (three or more satellites), we found that using this scale factor actually improved the accuracy of the solution over the full 5-variable solution. Although the scale factor's actual value has not been determined to be meaningful, we have some insight into its approximate function and appropriate magnitude. Very small values of the scale factor eliminate the receiver clock offset from the Doppler shift equation and cause the system to become less rigidly defined—a higher range of offsets cause less error during the solution process.

Large values of the scale factor (greater than 10^6) marginally improve the solution's accuracy with increasing order; however, near the solution point the design matrix begins to contain values with very large differences in magnitude. As a result, the floating-point operations performed by the computer lose precision and the matrix becomes almost completely singular.

Changing our method to use the scale factor, the full F vector and A matrix become:

$$F = \begin{pmatrix} P^1(t) - c\delta^1 - \rho^1{}_0(t) \\ v^1 \cdot \dfrac{r^1 - r_u}{||r^1 - r_u||} + d \\ \dots \\ P^n(t) - c\delta^n - \rho^n{}_0(t) \\ v^n \cdot \dfrac{r^n - r_u}{||r^n - r_u||} + d \end{pmatrix} \qquad A = \begin{pmatrix} \dfrac{-(X^1 - X)}{\rho^1} & \dfrac{-(Y^1 - Y)}{\rho^1} & \dfrac{-(Z^1 - Z)}{\rho^1} & -1 \\ \dfrac{1}{||r^1 - r_u||}\{(r^1 - r_u) \times [(r^1 - r_u) \times v^1]\} & & & scale \\ \dots & & & \dots \\ \dfrac{-(X^n - X)}{\rho^n} & \dfrac{-(Y^n - Y)}{\rho^n} & \dfrac{-(Z^n - Z)}{\rho^n} & -1 \\ \dfrac{1}{||r^n - r_u||}\{(r^n - r_u) \times [(r^n - r_u) \times v^n]\} & & & scale \end{pmatrix}^{-1} \quad \textbf{2.1}$$

2.2. Final Algorithm

The algorithm implemented by the MATLAB code to find the solution for a set of satellites using our combined method is as follows:

- Guess at the receiver's location in latitude, longitude and altitude coordinates and assume a zero initial receiver clock offset.
- Record the ephemerides, pseudorange and Doppler shift at a given GPS time known by the receiver clock. Correct the pseudoranges by the satellite clock offset from the ephemerides. Determine the magnitude of the radial velocity of each satellite by its measured Doppler shift (*dopplerradialvel.m*).
- For the measurement time, calculates each satellite's position and velocity in the ECEF frame at the GPS time using the satellite ephemerides (*ephemposvel.m*).
- Iterate the following steps ten times or until the magnitude of the difference in receiver location between iterations is less than 1 µm. For the *i*th iteration:
- Find the receiver's location in ECEF coordinates based on the WGS-84 model of an ellipsoidal Earth from the current latitude, longitude altitude guess using the (*ecef.m*).
- Using the guess at the receiver's ECEF location and each satellite's ECEF position, find the approximate propagation delay by dividing distance by the speed of light, and calculate new values for the satellites' position and velocities at the received GPS time minus this delay (*ephemposvel.m*, *calcpropdelays.m*).
- Find the velocity of the receiver in the ECEF frame by estimating the velocity of the surface of the Earth due to its rotation (*surfacevelocity.m*). Find the relative velocity of each satellite by subtracting this value from the calculated velocity.
- Calculate the F vector and A matrix using Equation 2.1
- Find the next guess at the receiver's coordinates by evaluating:
- $$\begin{pmatrix} x_{i+1} \\ y_{i+1} \\ z_{i+1} \\ c\delta_{R\,i+1} \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \\ z_i \\ c\delta_{R\,i} \end{pmatrix} + [A^T A]^{-1} A^T F$$
- Solve for the receiver's WGS-84 coordinates from the ECEF coordinates (*latlong.m*)

2.3. Antenna/Receiver Details and Data Collection

The receiver used to obtain data from the GPS satellites was a modified version of the Real-Time Linux PCI card developed by Aerospace Innovations. The receiver was controlled by a program called Cascade. The Cascade software is a 12-channel system that measures both integrated carrier phase and code range, and it employs the industrial standard RINEX-2.10 data format (Course packet: Chapter 1). The receiver was connected to an antenna on top of Upson Hall at Cornell University with a known position determined by survey measurements. This position, taken to be the correct position for error calculations later, was 42.444007 degrees latitude, -76.482229 degrees longitude, and 236.548 meters altitude (Course website).

The first data set was collected on Monday, December 1st, 2008. The data collection ran from GPS time 167731 to GPS time 168701. A total of 98 measurements were made, each 10 seconds apart. The satellites tracked during this time were the satellites with SVIDs 2, 4, 5, 7, 10, 12, 24, 25, 26, 27, 29, and 30. The second data set was collected on Wednesday, December 3rd, 2008. The data collection ran from GPS time 328835 to GPS time 330665. A total of 62 measurements were made, each 30 seconds apart. The satellites tracked during this time were the satellites with SVIDs 2, 4, 5, 9, 10, 12, 13, 17, 20, 23, 28, and 30.

The RINEX .nav and .obs files created by the Cascade software were formatted into matrices usable by MATLAB programs by using the MATLAB file gps.m and the function formatda.m, both written by Dr. Paul Kintner. Specifically, three matrices were created for calculations. One matrix contained ephemeris data for all satellites tracked. Another matrix contained the pseudorange measurements for each satellite, and a final matrix contained the measured NCO values for each satellite. The NCO values are simply the Doppler shift values with a constant offset caused by the difference between the local oscillator frequency and the satellite transmission frequency.

3. Results

3.1. Overview

3.1.1. Navigation Solution

We computed the navigation solution for our combined method using two, three, and four satellites, for two separate data sets. We tabulated the error in each set of solutions by taking the mean of the difference in calculated locations and the survey-defined antenna location over all time samples in a recorded data set.

3.1.2. Exponential Weighting

Early in this project, we noticed that the two-satellite solution demonstrated high-frequency error. To improve the solution by mitigating this error, we computed a weighted moving average solution set by modifying a time-independent solution set with the following algorithm.

The parameter was derived by a trial-error approach.

$$\begin{cases} \begin{pmatrix} x_t \\ y_t \\ z_t \end{pmatrix} = (1-\alpha)\begin{pmatrix} x_{t-1} \\ y_{t-1} \\ z_{t-1} \end{pmatrix} + (\alpha)\begin{pmatrix} x_t \\ y_t \\ z_t \end{pmatrix} \\ \alpha = 0.01 \end{cases}$$

3.1.3. Dilution of Precision Effect on Error

For the pure pseudorange solution, dilution of precision is calculated by evaluating the square root of the trace of the matrix $(A^TA)^{-1}$. In the derivation of dilution of precision, it is assumed that the pseudorange measurements from each satellite are independent, and that they all have the same variance σ_p. Therefore, the covariance matrix of the pseudorange measurements is σ_p times the identity matrix and can be treated as a scalar (Class Packet, Chapter 9: Ranging Errors). A similar derivation assuming a diagonal covariance matrix is used to derive DOP for the Doppler navigation solution and the dilution of precision is calculated in the same way (Lehtinen 29).

Our method involves both pseudorange and Doppler shift measurements. The error in these measurements are not necessarily independent, so the covariance matrix of both Doppler and pseudorange measurements cannot be assumed to be diagonal. However, the foundation of our method is that the pseudorange and Doppler methods produce independent receiver solutions; therefore, we propose that evaluating the square root of the trace of the matrix $(A^TA)^{-1}$ provides a good qualitative estimate of dilution of precision—enough to rank the error inherent in satellite combinations—even if it is not mathematically rigorous. This was validated by plotting the error for each satellite combination next to its DOP calculation and calculating the correlation coefficient.

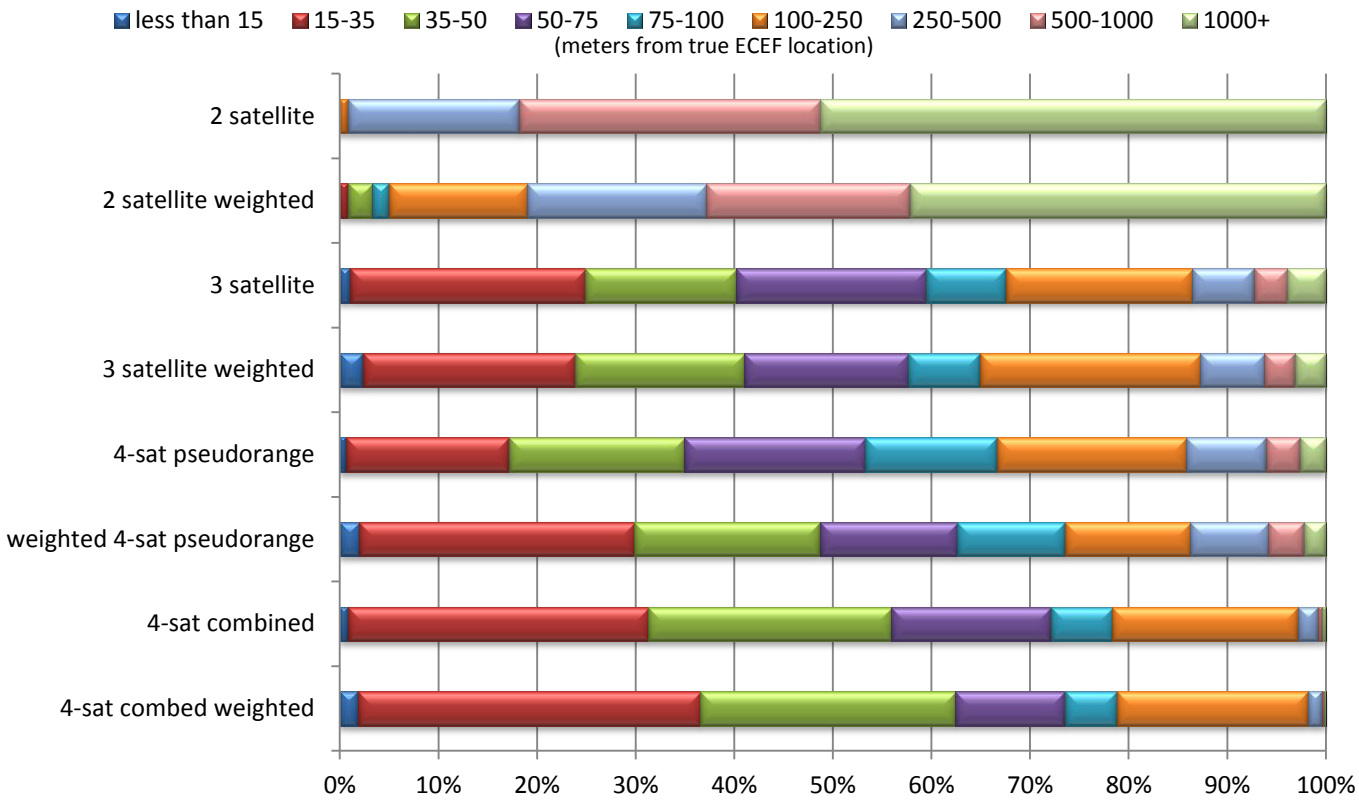
3.1.4. Four Satellite Comparison

To determine whether our combined pseudorange and Doppler shift navigation solution provided any benefit over the traditional method used by standard GPS receivers using only pseudoranges, the error from our combined method was compared to the error of the pseudorange navigation solution using 4 satellites. To eliminate the solution code and conditions as a source of error, the pure pseudorange solution for four satellites was calculated by removing (commenting out) the Doppler-shift contributions of the satellites from our navigation solution program.

3.2. Solution Method Average Error Comparison

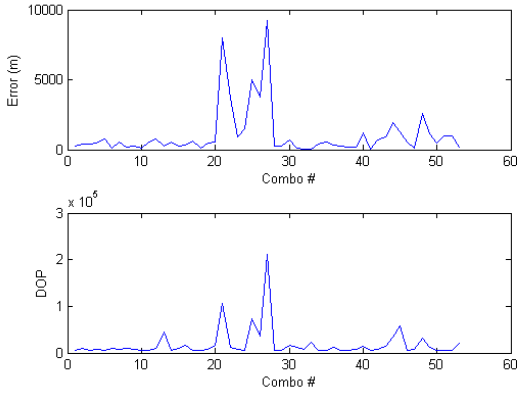
This graph is a stacked histogram that gives a visual representation of the overall errors exhibited by our navigation solution. Each of the methods analyzed is listed on the vertical axis, sorted by improving overall solution. Each method's bar is split into colored blocks whose length represents the percent of solutions with errors in the range matched to that block's color.

Fig. 3.1: Comparison of Solution Method Average Error



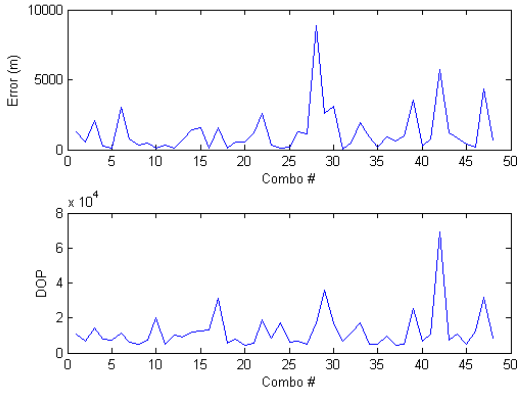
3.3. Two Satellite Solution

Figure 3.2: Two Satellite Error and Dilution of Precision, Plotted for All Satellite Combinations for time samples (1 - 10). Data Set 1 cutoff at 10,000 meters for graph readability. scale = 1000, exp_weight = 0.01



correlation coefficient between DOP and error = 0.843693511978030
bad (satellite combinations with error above 10,000 meters)=
[2 4], [2 5], [2 12], [2 24], [2 30], [4 5], [4 12], [4 24], [5 12], [10 27] [12 30], [24 26], [25 26]

Figure 3.3: Two Satellite Error and Dilution of Precision, Plotted for All Satellite Combinations for 10 time samples (1 to last available time in intervals of 10), excluding satellite 30 which had unusable data. Taken from data set 2 with error cutoff of 10,000m. Scale = 1000.



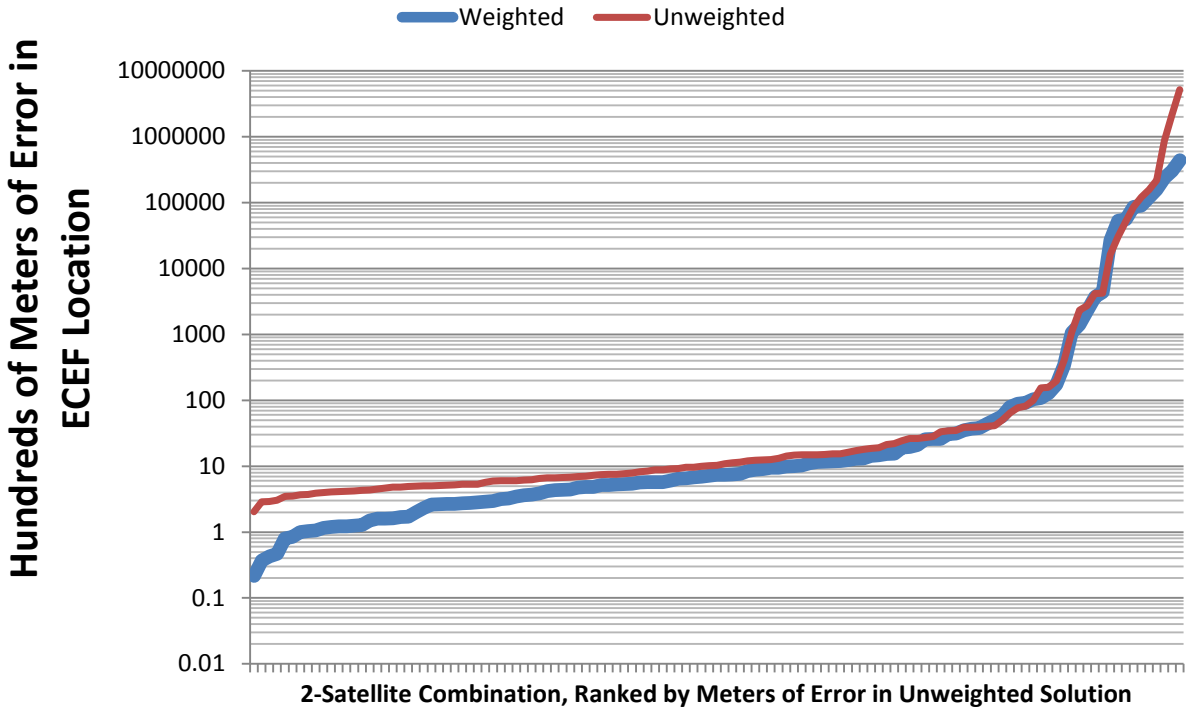
correlation coefficient between DOP and error = 0.89661512662354
bad (satellite combinations cut off by 10,000 meter limit)=
[2 4], [2 10], [2 13], [4 13], [5 12], [9 17], [17 28]

From these graphs, it can be seen that more than half of the two satellite navigation solutions provide solutions within 1 kilometer. In both figures, the error plot and DOP have the same overall shape, supporting the assertion that our method of calculating DOP preserves the quality of the actual dilution of precision. The maxima and minima mostly occur for the same satellite combinations, and have similar relative magnitudes.

3.4. Two Satellite Solution - Exponential Weighting

Weighting the 2-satellite solution's calculated receiver location over time improved the location's accuracy in almost every case. The result was especially apparent for the combinations that produced the highest accuracy, increasing the accuracy by up to half an order of magnitude.

Fig. 3.4: Comparison of 2-Satellite Solution Average Error



3.5. Three Satellite Solution

Figure 3.5: Three Satellite Error and Dilution of Precision: Data Set 1, Time Samples [1 - 10], cutoff of 500 meters. Scale = 1000, exp_weight = 0.01.

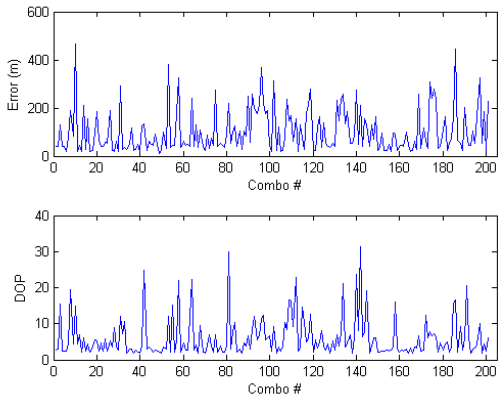
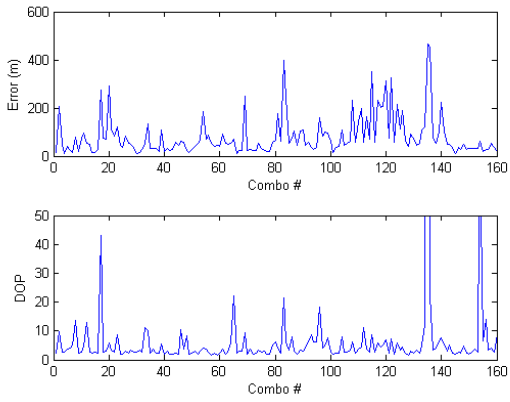


Figure 1

correlation coefficient between DOP and error = 0.924016416985303
not graphed (above 500 meters error)=
[2 4 24], [2 12 25], [2 27 30], [4 5 27], [4 7 12], [4 7 26], [4 12 26], [4 12 27], [4 26 27], [5 24 25], [5 25 29], [5 26 27], [7 12 26], [7 24 30], [7 29 30], [10 12 25], [10 27 30], [12 26 27], [24 27 29]

bad combinations (more than 10,000 m error) = [2 27 30], [4 5 27]

Figure 3.6: Three Satellite Error and Dilution of Precision: Data Set 2, Time Samples [1 - 10], cutoff of 500 meters. Scale = 1000. exp_weight = 0.01.



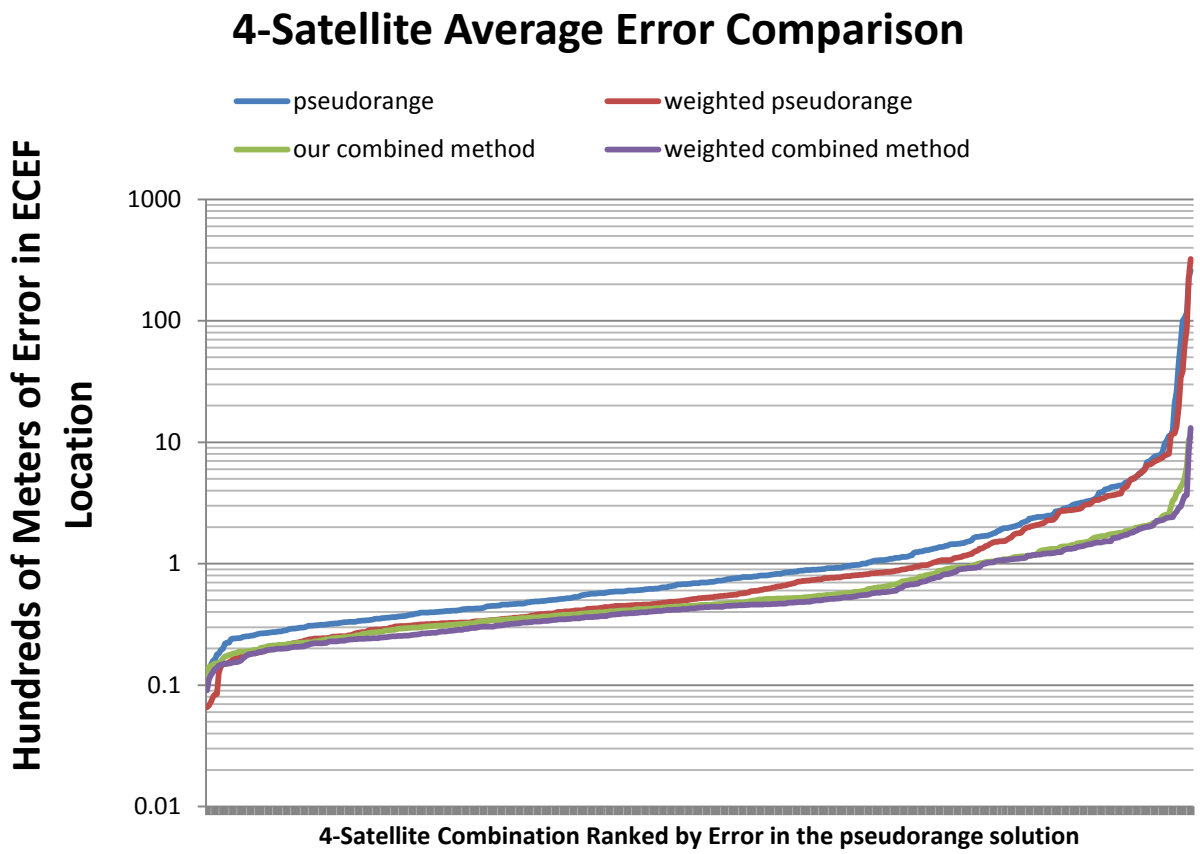
correlation coefficient between DOP and error = 0.89661512662354
not graphed (above 500 meters error) = [2 9 17], [2 9 23], [4 9 23], [9 12 20], [10 23 28]
no bad combinations (none more than 10,000 m error)
-Note that satellite 30 was excluded from this analysis: every combination involving satellite 30 would generate an error over 10,000 meters--at the starting time, satellite 30 has an elevation of about zero, and hence gives unsuable data.

From the above figures, it can be seen that using three satellites to calculate a navigation solution significantly reduces the error compared to two satellites. Most 3 satellite combinations provided solutions within 200 meters of the actual receiver location. It is more difficult to visually determine the relationship between error and DOP, but calculation shows that the two are more than 90% correlated.

3.6. Four Satellite Solution: Comparison with Pseudorange

The effect of combining pseudorange and Doppler shift measurements to determine a navigation solution, compared to the traditional method that only involves pseudorange, was examined by comparing the errors in the two navigation solutions using 4 satellites. Using data set 1, the error in both combined and pseudorange only navigation solutions were found, both with and without an exponential weighting of 0.01.

Figure 3.7. Comparison of the sorted error in the traditional pseudorange navigation solution to the combined pseudorange and Doppler shift solution using all combinations of 4 satellites.



Although the lowest errors returned by both solutions are approximately the same, the combined solution has much lower errors than the pseudorange solution, especially for high error satellite combinations. The number of satellite combinations with an error greater than 500 meters decreased from 29 to 2 when the Doppler shift equations were added to the Newton-Raphson matrices. This supports our assumption that combining Doppler shift and pseudorange measurements can improve the navigation solution when using 4 satellites.

4. Discussion

4.1. Data Analysis

4.1.1. Discussion of Two, Three, and Four Satellite Errors

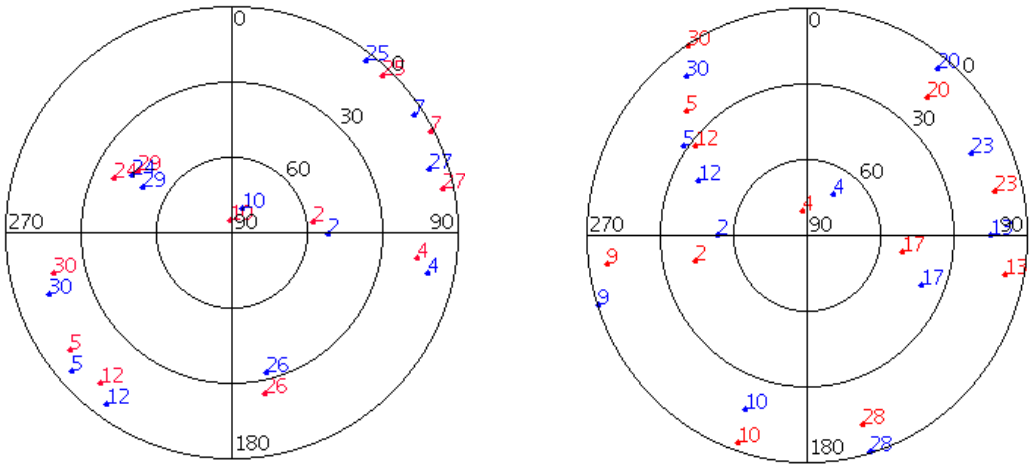
Beginning with the two satellite case, it can be seen that the errors for the navigation solution are generally greater than 100 meters (Figure 3.1). Nevertheless, there are a small percentage of combinations that returned errors of less than 50 meters, which could be considered an acceptable error for some applications. The two satellite case is more susceptible to larger errors, on the order of one kilometer or more. Nearly half of the combinations returned such errors.

For three satellites, errors are significantly reduced. Approximately 2/3 of the satellite combinations yield errors of less than 100 meters, and over 40% of the error is less than 50 meters. Only very particular combinations will give errors greater than 1 kilometer. The unweighted pseudorange solution, with four satellites, produces navigation solutions with a similar distribution of errors: 2/3 of the errors are less than 100 meters, which is almost identical to the combined three satellite solution; however, 1/3 of the errors are under 50 meters for the pseudorange solution. Recall that our combined solution had approximately 40% under 50 meters, a marked improvement.

The combined four satellite solution makes further improvements, with 37% of the errors under 35 meters; the unweighted pseudorange solution for four satellites generates 17% of the errors under 35 meters. Less than 5% of the four satellite combinations give errors greater than 1 kilometer, when using our combined solution.

4.1.2. Analysis of Good and Bad Combinations

Figure 4.1. Satellite Positions for Data Sets 1 and 2: Red corresponds to time 1, Blue is end time of data collection. (Data Set 1 is on the left, Data Set 2 is on the right).



It was shown earlier in the results section that the error is strongly correlated with the dilution of precision. However, there are cases where relatively low DOP corresponded to a high error. With the two satellite navigation solution, using Data Set 1, several combinations were found that satisfied this condition: SV_ids [2 24] and [4 24]. From Appendix 8.1.1, these combinations had errors above 1E6 meters, and DOP's below 10,000. A DOP of 10,000 was common to combinations that yielded errors below 250 meters.

With Doppler, there is an additional condition that will return large DOP not factored into our computations for DOP: when the user is in a plane between the two satellites, the intersection of the Doppler cones returns an entire plane of values. Thus, with two satellites, we do not converge to a solution in this special configuration. This can explain the errors for both [2 24] and [4 24]. Furthermore, this result shows that our DOP calculations are not sufficient to account for all sources of error in the combined Doppler / pseudorange solution; further research is needed to correct this deficiency.

Now considering three satellite errors for both data set 1 and 2, the correlation between DOP and error grows. From (Appendix 8.1.3) and (Appendix 8.1.4), there are no anomalous cases where DOP is low and error is high. However, there are cases with a high DOP corresponding to low error. For example, satellites [2 10 24] has an error of 17.7 meters with a DOP of 8.92. Although this DOP seems low compared to the two satellite case, the overall DOP's for the three satellites was significantly lower: generally under 5 for good combinations and over 10 for bad. We theorize that the error is low due to a low Doppler DOP and the overdetermined solution with three satellites. For future research, one can take the four satellite pseudorange solution and attempt to fix a high-DOP, high-error combination with Doppler aiding.

Overall, there is a strong correlation between high error and high DOP combinations, using the traditional DOP calculations. However, as shown above, there exist anomalous cases that the usual DOP calculation cannot account for, due to the mixing of Doppler and pseudorange in our Newton-Raphson matrices.

4.1.3. Merits of Exponential Weighting

From Figure 3.1, it can be seen that the use of an exponentially weighted moving average reduces the error in the navigation solution. This improvement is most drastic for two satellite combinations, while only providing slight improvement for three satellite and four satellite combinations. Starting with the two satellite case, the number of solutions with errors less than 500 meters increases from approximately 18% to 37%. The number of solutions with errors less than 100 meters increases from near zero to approximately 5%.

For three satellite combinations, the percentages remain essentially the same, but there is a slight improvement in the number of satellite combinations with errors less than 15 meters. We theorize that the use of an averaging filter reduces the effect of zero mean errors, such as random noise, and that the effect of zero mean errors is greater when using two satellite combinations. For further investigation, the use of other filters, like the Kalman filter, could be examined to further improve the navigation solution.

4.2. Solution Convergence and the Newton-Raphson Method

Using the Newton-Raphson method to solve the system of nonlinear equations developed for our method is convenient because of its simplicity, but a more robust tool to find the global minimum might allow better solutions to be found. There are several fundamental conditions necessary to employ the Newton-Raphson method. First, the initial guess must be near the desired local minimum and not on a saddle point or maxima. Second, to find the optimal solution, the local minimum that the initial guess tends toward must be the true solution. Finally, the first-order term must be the strongest influence on the overall shape of the solution space such that the higher-order contributors can be ignored during iteration.

The Doppler shift introduces nonlinearity into the Jacobian matrix of the Newton-Raphson method that is not present in the pseudorange solution. Additionally, in order to simplify the math, some nonlinear contributors that affect the calculated solution are not integrated into the system of equations and are instead updated between iterations. Such contributors involve signal propagation delay and surface-of-earth relative velocity correction. Correcting these terms reduces the dimension of the solution space, but in order to do so we have to make the assumption that updating the terms does not fundamentally change the shape of the remaining solution space. This may not be the case for all satellite configurations.

4.3. Future Work

Although we believe our results are solid, the ambiguity of the scale factor certainly merits further analysis. Specifically, it is important to see if the constant clock-error correction of the receiver unit used for testing is having an effect on the results of this method; however, our receiver does not allow this feature to be disabled. The following topics, some mentioned previously, could be useful in furthering the investigation started by this work:

- Kalman Filtering
- How to actually calculate the combined DOP
- How much does Doppler actually aid the overdetermined pseudorange solution?
- Analyze the impact of Doppler DOP/noise on the pseudorange solution for 2, 3, 4 satellites.
- Attempt a single-satellite solution with known altitude (via altimeter)

5. Conclusion

This paper attempted a limited satellite navigation solution using combined pseudorange and Doppler shift information. The results of this initial investigation are promising: using two satellites we are able to achieve a reasonably correct receiver location; with both three and four satellites, we could find a more accurate solution than the pure 4-satellite pseudorange method. The technique we present could be implemented in GPS receivers marketed for use in areas with limited sky visibility to improve the reliability of the units.

6. Saving the World: Sustainability & Us

If fewer satellites were required for GPS navigation than the twenty-four to thirty-two currently in use, there would be far-reaching environmental and social consequences. GPS satellites have finite lifetimes of approximately 7 to 9 years (GPS packet Chapter 1), and they must constantly be replaced. The cost of replacement could drastically be lowered if fewer satellites were necessary, allowing the government to devote this money to other programs such as health care and research on renewable energy sources. The resources required to replace each satellite would not be consumed in such large amounts, contributing to sustainability of Earth's resources. Another consideration is that the emissions from burning the fuel required to launch each satellite into space could also be greatly reduced.

7. Bibliography

Kintner, Paul. Course Documents of GPS Theory and Design: chapters 1, 7, 9. Version made available Fall 2008. *http://gps.ece.cornell.edu/ece415/*

Lehtinen, Antti. Doppler Positioning with GPS. Unpublished. 2001-2002. Available at *math.tut.fi/posgroup/**DopplerPositioningwithGPS**.pdf*, accessed November 2008.

8. Appendix Data

8.1. Satellite Errors

8.1.1. Data Set 1: Two Satellite Errors and DOP

Two Satellite exp_weight = 0.01, Scale = 1000. Data Set 1.

Sorted Error of Two Satellite Navigation Solution for Data Set 1													
Error (m)	DOP	SV IDs			Error (m)	DOP	SV IDs			Error (m)	DOP	SV IDs	
21.6120881	22074.07047	10	26		417.588235	5313.078195	10	29		1242.856693	57072.38365	25	27
42.9317138	7626.965835	10	25		431.3389502	8775.412738	2	10		1460.552757	6057.12825	7	24
46.9281718	5401.532213	24	25		441.6365158	6179.497648	26	30		1915.755705	34840.12916	24	30
80.4151967	11257.32729	10	24		484.1969929	8740.711656	2	26		2594.905149	32051.57765	26	27
100.5073783	8477.800204	25	30		486.4341649	7198.866381	5	27		3710.614086	11426.72537	7	10
119.7229169	5396.607652	4	26		515.5453596	5337.952991	5	7		3779.43478	36106.07486	7	26
124.72983	6438.385255	5	26		531.24067	6234.928888	4	27		4997.953754	73626.43547	7	25
128.2303377	9810.604453	2	29		539.2898128	15712.47553	5	29		7957.306392	105671.8975	5	30
148.9155061	9142.089049	4	10		568.725052	6818.407601	4	7		9204.545081	211059.4869	7	27
159.5532	8305.614758	12	27		571.4467586	6250.552185	10	30		10404.86394	59844.58118	12	30
160.8868534	20630.59459	29	30		573.7428257	4845.154318	25	29		10873.44736	107209.3062	5	12
199.5830912	6318.561147	12	26		604.5126296	6192.644194	5	25		12900.32521	20659.4208	2	30
260.4219414	4843.15819	12	25		693.1755228	15706.00473	10	12		17284.23033	19752.42781	10	27
265.3814806	10176.41602	5	10		716.4386787	6923.422779	24	27		106049.075	36375.35754	4	5
266.2263119	5775.467419	7	30		742.0298578	8871.776886	4	29		228609.3893	37692.75605	25	26
267.3449052	44805.45462	4	30		766.4062867	5803.193365	2	27		440333.8679	34802.25882	4	12
273.5742729	4965.562803	2	7		885.9655244	13748.31303	24	29		5327374.294	44224.66185	2	4
275.8229978	8342.27353	4	25		943.1723681	7236.771125	7	12		5596529.198	85616.15993	2	12
282.493896	4830.150015	7	29		986.9212625	4539.413485	27	30		8636529.797	7314.904942	4	24
315.0974721	11446.92078	12	24		1019.217523	4824.959594	27	29		15695107.88	8758.452322	2	24
344.4353701	15401.3353	5	24		1164.727259	12044.97361	26	29		23715968.07	41922.31732	24	26
369.6309551	4730.259123	2	25		1179.713943	14524.81869	12	29		44026517.82	114074.8808	2	5

8.1.2. Data Set 2: Two Satellite Errors and DOP

Two Satellite Errors, exp_weight = 0.01, Scale = 1000, Data Set 2.

Sorted Error of Two Satellite Navigation Solution for Data Set 2													
Error (m)	DOP	SV IDs			Error (m)	DOP	SV IDs			Error (m)	DOP	SV IDs	
37.0216365	6469.640098	10	17		658.3639806	8348.875007	23	28		3520.577951	25365.502	12	28
84.8347634	17162.37052	9	12		676.8764488	8928.837454	4	20		4378.334531	31675.76609	20	28
104.0202311	5600.461823	5	13		741.1470222	10598.57859	13	20		5714.809172	69086.07415	13	23
105.6501845	13199.67235	5	9		749.8860764	6256.865663	2	28		8836.164869	17113.24404	9	28
116.2253395	10132.99596	4	17		846.6523042	10532.62788	17	20		34046.14969	142790.9953	17	28
122.0862613	7146.292573	2	20		905.3510655	9508.021024	12	17		140716.6051	34006.35878	9	17
122.4606821	20284.44884	4	10		945.9850115	5202.030678	10	28		373565.0681	30354.1044	2	13
161.0281275	12667.27019	20	23		1006.39485	4938.698493	12	23		2703124.523	34573.56071	4	13
169.8307296	4996.60061	12	13		1113.690642	4827.118508	9	23		8993669.995	79047.15208	2	10
170.3327281	6126.977164	9	13		1174.449135	5695.412666	5	23		11871687.47	18732.8455	2	4
232.1005939	6637.642234	13	17		1195.978328	7483.657955	13	28		30409088.14	136077.6301	5	12
290.2146555	7911.276275	2	17		1301.489531	10578.38219	2	5					
296.8402391	4945.555258	4	5		1311.401671	6594.93411	9	20					
325.4523876	8418.17453	9	10		1439.125878	11856.99685	4	23					
363.8392941	5104.209008	4	12		1535.067893	12324.67145	4	28					
385.1277835	4965.702734	17	23		1546.631355	31170.57609	5	10					
438.1817142	11246.94101	10	20		1951.496701	17137.00046	10	23					
475.9348509	7394.381533	4	9		2085.967763	14195.61269	2	12					
516.9469372	7737.74391	5	17		2572.619355	35910.4152	10	12					
538.2336213	6817.570897	2	9		2603.736773	19091.84932	5	28					
570.4060061	4572.721409	5	20		3052.763591	11110.24696	2	23					
645.3371516	4511.492571	12	20		3117.807126	17153.88593	10	13					

8.1.3. Data Set 1: Three Satellite Errors and DOP

Three Satellite Errors. Exp_weight 0.01. Scale = 1000. Data Set 1

Sorted Error of Three Satellite Navigation Solution for Data Set 1																
Error (m)	DOP	SV IDs				Error (m)	DOP	SV IDs				Error (m)	DOP	SV IDs		
10.84525688	1.98776926 7	2	26	29		43.6260010 3	15.2803333 7	4	5	12		135.530533	3.59116309 3	7	12	29
17.60784594	2.10621122 6	2	5	29		43.7937267 1	1.94420399 9	25	26	29		137.059229 6	3.84310930 6	7	10	12
17.74141548	8.91993402 8	2	10	24		43.9926666 8	2.01929954 3	2	25	29		145.428260 9	3.71351555 4	7	12	24
18.02369726	1.90842569 2	2	24	26		44.2561707 1	2.69587222 2	10	25	27		146.597212	16.6251779 4	5	12	26
18.06259196	4.42114925 4	2	5	10		44.6095827 9	6.56950653 3	5	10	12		153.875084 2	7.91751316 6	7	25	30
18.19222758	2.66588333 5	10	24	26		45.2866043 9	1.81727787 2	10	26	27		155.002567 1	4.27522667 3	2	5	27
18.33771403	2.10009024 8	5	26	29		46.4382006 5	2.25462504 4	4	24	25		155.003824 9	22.8097699 9	5	12	30
18.65094362	2.81387661 4	2	10	26		46.6181817	2.09091243 4	4	12	24		163.264940 8	6.21026891 9	7	27	30
19.04487127	2.39075797	5	10	24		47.2057581 1	2.29192926 2	10	24	25		163.745861 8	4.33332869 8	25	27	30
19.3869421	2.51813715 2	5	10	29		49.6111079 5	2.94253214 3	2	25	27		164.147807 4	4.30684806 7	12	27	30
19.64420933	2.10237335 1	2	5	24		49.9597856 1	2.26951486 1	2	26	27		164.162405 8	15.6253149 4	24	25	29
19.76602672	2.28760503 4	5	24	26		50.7978806 3	8.12807322 2	7	25	26		164.303043 5	4.91267236 3	5	27	30
20.44511541	3.24756703 7	10	26	29		50.8970345 4	2.68124469	4	7	24		169.731843 6	16.4459706 2	5	12	27
20.77340095	2.37206430 3	2	4	26		51.0250285 9	1.99724137	2	24	25		174.878259 2	5.05795465 1	5	25	26
21.37983645	2.67482329 6	24	26	30		51.2070365 5	2.97190467 1	2	12	26		177.580386 2	5.50846949 5	5	7	24
21.77819501	3.19868480 5	10	24	30		51.3655079 5	5.37469622	7	10	27		178.606604 6	5.56203336 6	5	7	29
22.72870016	4.88466385 2	12	29	30		52.3148585 6	2.01189674 5	24	25	26		179.184606 7	5.19284634 6	4	5	25
23.01336852	5.63265588 1	5	26	30		52.4979792 8	2.47726836 6	2	7	29		185.736811 3	5.19943918 2	2	7	12
23.33726658	3.96973470 2	5	10	30		52.7564179 3	1.87055270 6	26	27	29		186.326450 8	4.81536898 2	7	12	30
23.45899473	2.29101510 7	26	29	30		52.8602659 5	1.98051500 8	7	24	26		186.494938 5	4.38990220 1	26	27	30
23.54825104	3.07273370 6	2	24	30		53.5336881 3	3.04582688 9	4	24	30		188.863066 9	5.27940285 1	2	7	30
23.98193062	1.86213185 1	10	26	30		53.5675084 3	6.97781897 1	7	24	25		189.571669 6	19.5686442 9	2	4	29
24.28666625	1.92082069 4	4	10	26		55.0071375 4	3.38655124 1	25	27	29		191.455121 7	6.19184563 1	5	7	30
24.46145812	1.95155846 8	5	10	26		55.0084258 4	9.35304789	24	26	29		198.783560 8	12.0708731 1	5	7	12
24.67190867	2.74091634 4	10	12	30		55.0727057 8	3.81570511 7	4	12	30		200.913025 9	5.75420702	24	27	30
24.76186968	3.71878471 2	2	5	30		56.3638428 9	2.90896731 8	4	29	30		210.730189 8	31.4022206 1	7	25	27
25.17762283	3.53865012	2	10	12		56.8900562 9	2.71361665 6	7	10	24		212.449017 8	6.00593607 4	2	5	25
25.18092403	1.91025496 4	2	12	29		59.4785734 8	5.71809337 3	2	7	27		216.124982 8	5.73294209 6	5	25	27
25.19765939	2.21617531 3	10	12	24		59.7447155 5	3.50575980 1	2	25	26		216.675579 4	6.93981210 6	5	7	25
25.27717116	2.39390361 5	10	12	29		60.2256471 9	12.3679587 4	12	24	29		218.017630 4	12.4728611 5	5	7	27
25.57756092	2.65074252 1	2	12	30		60.5422624 2	4.55740029 4	4	5	30		218.487078 3	29.9002991 3	4	24	29
25.86206211	7.27754080 2	2	10	29		61.2129603 9	9.22752325 6	5	12	29		226.249897	5.99990726 7	27	29	30
26.26423778	1.96754029	4	26	29		62.1651218 1	3.46798657 4	10	27	29		229.981878	7.03338916 8	7	10	30
26.38514746	1.88959209 7	2	12	24		63.6746755 7	3.60492361 3	2	10	25		237.859588 7	8.40034962 8	5	12	25
27.00340727	1.96552249 8	2	26	30		65.9541723 4	2.49148897 4	2	7	24		237.912191 9	6.29981278 8	12	25	30
27.91825618	10.2138288 6	5	24	30		67.1685626 8	1.99871521	24	26	27		238.766075 3	22.3549050 9	4	7	27
29.02481419	3.12837253 2	10	29	30		68.6455067 7	6.02828133 8	7	27	29		240.360326 6	7.45647719 4	7	12	25
29.58430333	2.45748700 5	2	5	26		73.6808437 8	3.60675356 8	24	25	27		240.955859	6.70615773 1	12	25	27
30.06903663	2.94889601 4	2	29	30		74.3861514 3	3.90930485 6	4	10	30		246.838893 8	6.72678129	4	27	30

30.77670427	6.106929778	12	24	30		77.32956714	4.339162597	2	4	27		257.9305581	8.312581271	5	7	10
32.38213484	2.395562359	12	24	26		78.40639529	14.54834523	5	24	29		258.0036241	6.695770217	12	24	25
32.80587155	2.231781377	12	26	29		79.11399748	10.49557116	5	12	24		259.5410323	21.18419852	7	12	27
33.07572305	6.817290842	4	10	29		81.22825432	3.851239331	4	27	29		273.581746	6.965404281	4	12	25
34.55066815	2.268773568	4	5	10		82.71344231	4.297054879	2	4	30		275.5469561	23.55331846	7	24	29
35.12613568	10.51605446	2	10	30		86.59173819	6.21098439	7	24	27		278.3687384	7.128272053	12	25	29
36.10520146	1.868023848	7	10	26		88.03085833	4.120806185	4	10	27		279.678803	12.48808638	5	25	30
36.71429883	2.164773799	4	5	29		90.06559088	3.899804049	4	24	27		289.5554936	12.01967154	2	10	27
37.03539944	1.984761831	4	10	12		91.45557002	15.97596136	10	24	29		307.509487	7.765058412	12	25	26
37.77699205	2.163115561	10	12	26		91.61599894	2.65934959	2	25	30		310.7886588	9.286540714	5	10	25
37.9774722	7.965385935	5	29	30		94.83253883	3.617115687	10	24	27		323.8661836	10.05281572	25	29	30
38.05139047	2.091036941	4	24	26		96.75591529	2.524817335	10	12	27		325.9058664	21.92690895	4	5	26
38.50407689	2.720068417	2	4	5		97.44910595	20.43095621	24	29	30		367.3139318	11.51874553	5	7	26
38.96492072	4.467962805	7	10	25		98.216833	3.552038632	2	27	29		379.9515372	12.18698517	4	5	7
38.98935608	11.85559427	4	7	25		98.36297542	5.53627355	2	7	10		442.874611	16.5625968	24	25	30
39.11381005	2.980379273	2	4	7		98.99953792	3.056090055	10	25	30		466.780459	14.92658437	2	5	7
39.40703279	2.44589418	10	25	29		100.9091898	7.73289864	4	25	26		502.679781	61.00939382	4	26	27
39.43440431	2.169615725	4	25	29		102.199191	2.935495694	25	26	30		548.6247429	32.41119249	4	7	26
39.47127608	2.46057269	2	4	12		102.458774	3.078976726	4	25	30		552.4980065	20.91628968	7	24	30
39.80678823	4.628342937	2	7	25		103.0905938	4.587024107	4	26	30		582.9967064	21.02732649	5	24	25
39.80958967	2.471745254	2	4	25		106.5067918	9.444402087	4	10	24		590.795056	37.98804916	2	4	24
40.00908711	6.360825301	7	25	29		106.8597486	2.587414433	12	27	29		847.5908661	22.51184355	10	12	25
40.04786357	2.021464949	4	10	25		116.8240529	3.066864848	2	12	27		912.1505562	27.19742358	5	26	27
40.23549842	2.518473396	2	7	26		118.9821821	2.716987708	12	24	27		963.1349896	40.76152146	4	12	27
40.37037541	1.990149891	4	12	29		120.0860401	3.192346153	5	27	29		988.091979	35.74778843	5	25	29
40.37723304	4.743902572	12	26	30		123.1180704	3.481141588	5	10	27		1036.340041	36.84529503	7	29	30
40.76394215	2.775695413	7	10	29		124.5476842	10.47130581	4	25	27		1115.447261	26.91929796	10	27	30
40.88935599	2.119653533	10	25	26		127.0089698	3.702919692	2	24	27		1221.619681	33.5446093	4	7	12
40.89979585	1.870215403	7	26	29		128.4216625	3.280627329	5	24	27		1335.617773	67.35515252	24	27	29
42.0968319	2.614746528	4	7	29		129.0184179	19.15000714	7	26	27		1336.882008	30.17266208	2	12	25
42.66267156	6.984431922	2	5	12		130.0666327	3.991378643	4	7	30		1579.962393	55.6508773	7	12	26
42.84943446	2.258733847	4	5	24		130.6679402	15.4479283	2	4	10		2311.92835	40.00229833	12	26	27
43.42998364	2.618743817	4	7	10		131.8825177	3.696011211	7	26	30		3199.135737	103.6700516	4	12	26
43.43706432	6.139062375	25	26	27		133.1987974	24.78503138	2	24	29		17091.05155	108.0734477	4	5	27
												117839.3822	414.7328282	2	27	30

8.1.4. Data Set 2: Three Satellite Errors and DOP

Three Satellite Errors, exp_weight = 0.01, Scale = 1000, Data Set 2.

Sorted Three Satellite Solution Errors for Data Set 2																
Error (m)		SV IDs				Error (m)		SV IDs				Error (m)		SV IDs		
9.178435793	2.573795605	2	12	17		38.91770932	3.778989842	2	4	13		75.10388524	2.607276288	2	9	10

11.58472983	2.768344644	2	4	12		39.11070504	2.131398734	4	10	12		76.21034309	7.851332444	5	10	12
11.81763342	1.932925636	12	17	20		39.12978034	2.465029137	5	17	28		76.47497556	3.294013093	10	17	23
11.91042718	2.270516752	4	12	20		41.47255895	2.275141054	4	10	17		79.51168325	13.62608005	2	4	23
13.07586416	2.262358768	2	5	17		41.49290295	3.092722968	4	5	28		82.2356185	2.587705154	2	9	28
14.5845833	2.037288001	5	17	20		43.56287473	3.40294243	10	12	28		83.30894247	3.085166554	2	10	20
14.64400884	2.42060794	2	4	5		44.07001281	2.634573965	5	20	28		84.90459018	3.922688921	5	13	17
14.81924019	1.998043333	4	5	20		44.24040246	2.094360165	2	10	13		85.79077928	2.831704353	4	9	20
15.0820226	2.532642357	2	5	20		45.25940152	1.754981194	4	10	13		89.9534222	3.108017863	10	12	20
15.97086984	5.182626092	2	4	20		45.50239233	1.99100147	4	5	10		91.48999316	3.738296071	4	10	20
16.5609079	3.396627973	2	12	20		45.76593008	4.102561359	5	10	28		95.55856782	7.545650128	5	13	23
17.10659592	2.149187463	2	4	28		46.02002327	1.857563901	5	10	17		97.26023046	4.757578339	2	5	10
17.87777453	1.873449387	2	17	28		46.89352065	4.889510852	12	20	23		100.177422	4.768482728	5	13	20
19.48731638	8.757861457	20	23	28		47.07872055	2.490037322	2	5	13		103.5150037	2.864641713	5	10	13
19.70189052	2.153001952	2	17	20		47.65527271	2.633509035	4	10	28		105.7118508	5.38323375	12	13	17
20.1423175	2.022589427	4	20	28		48.37918405	3.111824996	12	13	20		106.1236037	6.409590286	10	20	23
21.04744247	1.773716464	4	23	28		48.53258474	2.710969598	2	4	10		106.3812738	3.374535102	5	10	20
21.14248452	6.477770868	13	20	28		48.82618302	3.293081755	2	10	28		106.659548	2.873113676	5	10	23
21.89343	3.064817777	4	12	23		49.32243769	5.125028472	12	13	23		107.5865097	7.904395919	5	20	23
22.09394935	2.399307347	4	17	20		51.40426182	2.603130971	5	23	28		108.3626374	5.30238589	10	13	20
22.47684699	2.88445247	4	12	28		51.45046619	1.916713307	4	9	10		110.1970741	5.331687143	2	17	23
22.54666972	2.914251	4	20	23		51.93560446	2.45301008	10	13	17		111.9571711	3.394775079	2	9	20
23.21676668	1.896842212	4	13	28		52.23600533	4.071258212	17	20	28		114.1638412	3.371246809	9	20	28
23.90395575	1.989428405	2	20	28		52.63360926	1.743997667	4	9	28		120.0652788	12.11967594	10	13	23
24.58002652	2.066179894	4	13	20		53.00748476	1.981100354	4	10	23		120.1638058	8.677030735	2	10	12
24.69708846	3.918481986	2	4	17		53.30973002	3.357930295	5	9	28		133.1820418	10.07419103	2	13	17
24.82522748	8.462270949	4	5	17		53.87513609	2.142424008	2	10	23		142.7004319	3.843878938	9	10	20
25.74465623	13.91801679	13	23	28		54.36736463	11.15918591	2	12	28		161.2969048	18.21439864	5	12	28
25.95200357	3.742637951	4	13	23		54.45324039	3.492241856	4	17	23		164.2348298	3.70111153	9	12	13
26.17915827	2.192880586	12	17	28		54.53336281	12.86737723	2	5	12		177.6953206	3.914818524	5	9	13
26.43390834	2.693590265	2	12	23		54.96445508	3.90584466	10	17	28		184.6382803	4.061918254	4	9	13
27.09199907	6.129273387	5	12	20		55.14816821	5.92697703	4	12	13		187.5343549	4.332480195	9	23	28
27.20592429	2.373551698	4	5	23		55.49127077	2.212382851	9	10	17		197.06366	4.415892923	9	10	23
27.56567406	1.976320463	2	23	28		55.52844879	5.248032758	5	9	10		200.2138496	4.298084465	9	13	20
28.35069959	3.465642999	17	20	23		55.53584576	2.091361946	9	17	28		207.2074413	9.813646875	2	4	9
28.38204248	2.20007431	12	20	28		55.98184357	11.10255798	9	10	28		207.5919756	5.219488748	9	13	23
29.28148644	2.270804299	2	5	23		56.0401304	2.310877314	4	5	9		216.0518864	5.761588266	9	20	23
30.25628442	3.065615147	2	20	23		56.68708658	3.594646404	4	5	13		222.9320148	7.528767057	10	20	28
32.03439956	2.149481177	2	13	20		57.90716801	6.566831968	5	12	13		232.1271155	5.767091789	9	13	17
32.21909412	3.652489705	2	13	23		58.65464314	2.557440714	9	12	28		233.9326286	6.207187575	9	10	13
32.37260448	2.121030143	12	23	28		59.91505378	2.760846913	5	13	28		242.7738679	6.031825296	5	9	23
33.29092168	2.553881605	13	17	20		59.92433448	2.235414849	5	9	17		250.3210783	9.52601211	4	13	17
33.36043609	3.87629052	13	17	23		60.52941431	10.57796372	4	5	12		273.1228024	42.93165134	2	5	28
33.58504212	2.327494177	2	13	28		61.44363631	3.389249362	9	10	12		292.1093094	5.961934813	2	9	13
33.78779119	2.832218677	4	17	28		63.5722949	75.03041788	13	20	23		314.1023465	6.943053162	9	13	28
33.97235479	2.505338276	13	17	28		66.04569876	2.266807921	10	12	13		324.3580407	7.226913216	9	17	23
34.51854572	2.545750924	12	17	23		66.91070939	6.185903909	5	9	12		350.482272	8.690418262	9	12	23
34.7761822	2.390593853	5	17	23		71.69173927	22.03892874	4	12	17		398.3240468	21.38685107	5	9	20
35.68083545	6.040621868	5	12	23		72.01784115	2.9794434	4	9	12		452.9111499	22.02538858	10	17	20
35.89631043	2.727982374	2	12	13		72.04202176	2.5113681	9	12	17		464.6645548	657.268519	10	13	28
35.95777501	1.863898877	2	10	17		72.07254781	3.889493428	4	9	17		580.4962899	15.35968589	2	9	23
36.18601758	2.686353067	17	23	28		72.33876749	3.058582858	2	9	12		728.3770172	11.99279536	2	9	17
36.59032709	8.720183178	5	12	17		72.69109415	2.73734974	2	5	9		831.9349901	14.74957244	9	12	20
37.77338823	2.286865942	12	13	28		74.2737018	2.35984228	10	12	23		1505.693861	36.64726224	4	9	23
38.72553192	1.759904198	10	12	17		74.85838772	2.393843116	9	17	20		5101.528905	20.93877071	10	23	28

9. Appendix Code

9.1. calcpropdelays.m

```
% calculates the per-satellite, per-time propagation delays for a set of
% data.
% Input:
%   rx_pos - receiver position in ECEF
%   sats_pos - satellite positions
%   sprop - signal propagation speed
function propagation_delays = calcpropdelays(rx_pos, sats_pos, sprop)

% update the position and velocity based on propagation time that the
% signal would experience getting to the current location
satellites = (size(sats_pos,2)-1)/4;
propagation_delays = [];
for sat=1:satellites
    pos_diff = [sats_pos(:,sat*4-1) - rx_pos(1), sats_pos(:,sat*4+0) - rx_pos(2), sats_pos(:,sat*4+1) -
rx_pos(3)];
    pos_diff = sqrt(sum(pos_diff.^2,2)) ./ sprop;
    propagation_delays = [propagation_delays pos_diff];
end

end
```

9.2. constant.m (written by Dr. Paul Kintner)

```
% constant.m    (actual file name: constant.m)
%
% this GPS utility defines physical constants to be used in the
% GPS functions and utilities
%
% constants defined:
%   'mu' : G*Me, the "gravitational constant" for orbital motion
%       about the Earth
%   'AA' : the semi-major axis of the reference ellipsoid (WGS-84)
%   'BB' : the semi-minor axis of the reference ellipsoid (WGS-84)
%   'esquare' : the square of the Earth's orbital eccentricity
%   'OmegaE' : the sidereal rotation rate of the Earth (WGS-84)
%   'c' : the speed of light (meters/second)
%   'degrad' : a constant used for converting degrees to radians
%   'leapSeconds' : the number of leap seconds currently for the
%       GPS system (seconds)
%   'f0' : the fundamental frequency for the GPS system (Hertz)
%   'f' : the L1 carrier frequency (Hertz)
%   'lambda' : the L1 carrier wave length (meters)
%
    muearth = 398600.5e9;              % meters^3/second^2
    AA = 6378137.00000;                 % meters
    BB = 6356752.31425;                 % meters
    esquare=(AA^2 - BB^2) / AA^2;
    OmegaE = 7.2921151467e-5;           % radians/second
    c = 299792458;                          % meters/second
    degrad = pi/180.0;
    leapSeconds = 14;                       % seconds
    f0_freq = 10.23e6;                          % Hertz
    L1_freq = 154 * f0_freq;                            % Hertz
    L1_lambda = c / L1_freq;                        % meters
```

9.3. dopplerradialvel.m

```
% converts a Doppler matrix to the set of relative raidal velocities
% Note that this solution doesn't include the NCO, which has to be solved
% for during iteration
% Input:
%       sprop - the speed of propogation of each satellite's signal (rows)
%               if this is a scalar, it is multipled as necessary
%               set to the speed of light if unsure
%       dopp  - [GPStimes [SVID1 DOPPLERS] [SVID2 DOPPLERS] ...]
%       trans_freq - (scalar) the transmitting frequency of the signal
% Output:
```

```
%       [GPStimes [SVID1 relative radial velocity] [SVID2 relradvel] ...]
%
% For each satellite, the returned relative radial velocity is TRULY
% calculated by this formula:
%   |vrad| = (fT - fR + NCO) * sprop / trans_freq
% Where -(fT - fR) = dopp.  Note that this formula DOESNT correct for the
% clock offset
function [ relradvel ] = dopplerradialvel(sprop, dopp, trans_freq)

% find the number of satellites
satellites = (size(dopp,2)-1)/2;

% format the speed of propogation
if (size(sprop,1) == 1)
    sprop = ones(satellites,1) * sprop;
else
    if (size(sprop,1) ~= satellites)
        printf('A vector sprop must have the same # of rows as satellites');
        return;
    end
end

% copy GPS times
relradvel = dopp(:,1);


% copy each satellite
for sat = 1:satellites
    relradvel = [relradvel dopp(:,2*sat) (-(dopp(:,2*sat+1)) .* sprop(sat) ./ trans_freq)];
end

return;
end
```

9.4. dopsoln.m

```
%NOTES ON DOP: sigmaP ~ 10^-2 * 300 meters
%sigmaD ~ 10^-2 * lamda_l1

clear all;
constant;

ephem = load('ephem.asc');
% all satellites with data: 2 4 5 7 10 12 24 25 26 27 29 30
fprintf('\nEnter in satellite numbers in the form [sat1# sat2#...satn#] (minimum of 2)\n Available
Satellites: ');
fprintf('%d ',ephem(:,1));
fprintf('\n ');
SV_ids = input('');

% read in all of the satellite data
[ephem pseudo dopp] = loaddata(SV_ids);

[pos vel] = ephemposvel(ephem, pseudo(:,1));
[vrad] = dopplerradialvel(c, dopp, L1_freq);

% guess at the receiver position
%rx = [ 42.444290 -76.482126 239.47 ]; % a very exact guess
rx = [ 42 -76 220 ]; % an ok guess
%rx = [ 35 -70 0 ]; % a bad guess
rx_pos = ecef(rx);
rx_vel = [ 0 0 0 ]; % stationary receiver

[pos vel] = ephemposvel(ephem, pseudo(:,1), calcpropdelays(rx_pos,pos,c));

% Esimate the receiver clock offset using the radial velocity mismatch
mismatches = radialvelocitymismatch(rx_pos, rx_vel, surfacevelocity(rx_pos, rx), pos, vel, vrad);
rx_cdr = 0;%mean(mean(mismatches));

% Define the functional differencing matrix (F) and the design matrix (A)
% using parameters with MATLAB's symbolic math.
syms cx cy cz cdr;  % receiver location and clock offset
syms cvx cvy cvz;   % receiver relative velocity (surface + mobile)
syms sx sy sz;      % satellite location
syms svx svy svz;   % satellite velocity x/y/z
syms svrad;         % satellite radial velocity
```

```
syms P;             % pseudorange
syms nco;
syms scale;
syms p pnorm v vnrom rho;

scale = 1000;

% the Pseudorange equation: rho - (pseudorange + cdr) = 0
% the Doppler equation:     dot(rho_hat, sat_vel - rx_vel) - (vrad+cdr/L1_lambda) = 0

%distance from current guess to satellite
    rho = sqrt((cx-sx).^2+(cy-sy).^2+(cz-sz).^2);

%setting up the cross products
   p = [(sx-cx) (sy-cy) (sz-cz)];
   pnorm = p / rho;
   v = [svx-cvx svy-cvy svz-cvz];
   vnorm = v / rho;

%function vector
F = [ rho - (P + cdr);
      ((sx-cx)*(svx-cvx) + (sy-cy)*(svy-cvy) + (sz-cz)*(svz-cvz)) / rho - (svrad - scale*cdr);
      ];
%A matrix
A = [ -(sx-cx)/rho -(sy-cy)/rho -(sz-cz)/rho -1;
      cross(pnorm, cross(pnorm, vnorm)) scale;
      ];

times = size(pseudo,1); % number of times to calculate receiver location
rx_locations = [];      % tracks location of the receiver over time
rx_cdrs = [];

%Input the Times
fprintf('\nEnter initial time and end time as [starttime endtime]\n Maximum endtime: ');
fprintf('%d', times);
fprintf('\n');
t = input('');
tstart = t(1); tend = t(2);

error = [];
% repeat for all of the times that we have
    for time=tstart:tend
        satellites = size(SV_ids,2);
        pos_convergence = [];
        %feval_convergence = [];

        %initialization
        delta = 100;
        iterations = 0;
        while delta > 1e-6 && iterations < 10
            difference = [];
            design_matrix = [];
            cx = rx_pos(1); cy = rx_pos(2); cz = rx_pos(3);
            cdr = rx_cdr;
            rx_cdrs = [rx_cdrs rx_cdr];

            relvel = rx_vel + surfacevelocity(rx_pos, rx);
            cvx = relvel(1); cvy = relvel(2); cvz = relvel(3);

            %A MATRIX CONSTRUCTION
            for sat=1:satellites
                %find satellite positions and velocities
                sat_pos = pos(time,[sat*4-1:sat*4+1]);
                sat_vel = vel(time,[sat*4-1:sat*4+1]);
                sx = sat_pos(1); sy = sat_pos(2); sz = sat_pos(3);
                svx = sat_vel(1); svy = sat_vel(2); svz = sat_vel(3);
                svrad = vrad(time,sat*2+1);
                P = pseudo(time,sat*2+1);

                %setting up the cross products
                rho = sqrt((cx-sx).^2+(cy-sy).^2+(cz-sz).^2);
                p = [(sx-cx) (sy-cy) (sz-cz)];
                pnorm = p / rho;
                v = [svx-cvx svy-cvy svz-cvz];
                vnorm = v / rho;

                difference = [difference; eval(F)];
```

```
                design_matrix = [design_matrix; eval(A)];

            end %end of A matrix construction

            %%feval_convergence = [feval_convergence difference];
            %delta vector
            delta_pos = -inv(design_matrix'*design_matrix)*design_matrix'*difference;
            pos_convergence = [pos_convergence norm(delta_pos(1:3))];
            %increment our guess by delta vector
            rx_pos = rx_pos + delta_pos(1:3)';
            %set receiver offset
            rx_cdr = delta_pos(4);
            %convert to lat /long
            rx = latlong(rx_pos);
            %find magnitude of our correction term
            delta = norm(delta_pos(1:3));
            iterations = iterations + 1;
        end %end while loop for convergence
        %%plot(1:size(feval_convergence,2),(feval_convergence));


        s = size(rx_locations,2);
        exp_weight = 0.1;

        if (s>1 && time > tstart)
            rx_pos = rx_pos * exp_weight + (1-exp_weight)*rx_locations(:,s)';
        end
        rx_locations = [rx_locations rx_pos'];

        error = [error norm(ecef([42.444007 -76.482229 236.548])-[rx_pos(1) rx_pos(2) rx_pos(3)])];
        % update the position and velocity based on propagation time that the
        % signal would experience getting to the current location
        [pos vel] = ephemposvel(ephem, pseudo(:,1), calcpropdelays(rx_pos,pos,c));

    end%end for loop for times

% calculate the accuracy and precision of the locations
accuracy_meters = locdiffmag(rx_locations, ecef([ 42.444007 -76.482229 236.548 ]))
precision_meters = locdiffmag(rx_locations, mean(rx_locations,2)')

%plot([1:length(error)], error);
% plot the receiver evolution over time
%rx_locations = locdiff(rx_locations, mean(rx_locations,2));

%figure
%Dot denotes initial nav solution, * marks final nav solution
%%figure
%plot3(rx_locations(1,1),rx_locations(2,1),rx_locations(3,1),'o',rx_locations(1,tend-
tstart),rx_locations(2,tend-tstart),rx_locations(3,tend-
tstart),'m*',rx_locations(1,:),rx_locations(2,:),rx_locations(3,:));
%xlabel('xerror');
%ylabel('yerror');
%zlabel('zerror');
%plot3(rx_locations(1,:),rx_locations(2,:),rx_locations(3,:));
```

9.5. dopsolncombos.m

```
%NOTES ON DOP: sigmaP ~ 10^-2 * 300 meters
%sigmaD ~ 10^-2 * lamda_l1

clear all;
constant;

data = input('Enter which data set to use: 1 or 2 \n');

if(data == 1)
    ephem = load('ephem.asc');
elseif(data == 2)
    ephem = load('ephem_2.asc');
end

% all satellites with data: 2 4 5 7 10 12 24 25 26 27 29 30
allsat = ephem(:,1)';
%allsat = [2 4 5 9 10 12 13 17 20 23 28];

exp_weight = input('\nEnter Exponential Weighting\n');
cutoff = input('\nEnter cutoff\n');
```

```
%Create all the satellite combinations
SV_id_indices = [];
for a1 = 1:length(allsat)
    for b1 = a1+1:length(allsat)
        for c1 = b1 + 1:length(allsat)
            %for d1 = c1 + 1:length(allsat)
                SV_id_indices = [SV_id_indices; a1 b1 c1];
            %end
        end
    end
end

%Load the Data
if(data == 1)
    ephemData = load('ephem.asc');
    obsData = load('obs.asc');
    doppData = load('obsdopp.asc');
elseif(data == 2)
    ephemData = load('ephem_2.asc');
    obsData = load('obs_2.asc');
    doppData = load('obsdopp_2.asc');
end

[ephem, pseudo, dopp] = formatData(ephemData, obsData, doppData, allsat);


%Initialize
cerror = [];
bad = [];
DOP = [];
good = [];

 % guess at the receiver position
 %rx = [ 42.444290 -76.482126 239.47 ]; % a very exact guess
 %rx = [ 42 -76 220 ]; % an ok guess

 rx = [ 20 -70 0 ]; % a bad guess
 rx_guess = rx;
 rx_pos = ecef(rx);
 rx_vel = [ 0 0 0 ]; % stationary receiver

 % read in all of the satellite data
 [pos vel] = ephemposvel(ephem, pseudo(:,1));
 [vrad] = dopplerradialvel(c, dopp, L1_freq);

for combo_num = 1:size(SV_id_indices,1)
        SV_id_array = SV_id_indices(combo_num,:);
        %displays the satellite combination that is currently running
        allsat(SV_id_array)

        % Esimate the receiver clock offset using the radial velocity mismatch
        rx_cdr = 0;%mean(mean(mismatches));

        %arbitrary choice of scaling factor...has to be >100
        scale = 1000;

        times = size(pseudo,1); % number of times to calculate receiver location
        rx_locations = [];      % tracks location of the receiver over time

        error = [];
        % repeat for all of the times that we have
            for time_index=1:10
                satellites = size(SV_id_array,2);

                 % get the current time
                 time = pseudo(time_index,1);

                 % pull out the satellite measured radial velocities and pseudoranges at
                 % the current time
                 time_pseudo = pseudo(time_index,:);
                 time_vrad = vrad(time_index,:);

                 % by setting this each loop, we can be sure that the solution of one
                 % time doesn't affect the solutions at other times (this is an
                 % asynchronous method, after all)
                 rx_cdr = 0;
```

```
                rx = rx_guess;
                rx_pos = ecef(rx_guess);

                %initialization
                delta = 1;
                iterations = 0;


            while delta > 1e-6 && iterations < 10

                % update the position and velocity based on propagation time that the
                % signal would experience getting to the current location
                [pos vel] = ephemposvel(ephem, time);
                [pos vel] = ephemposvel(ephem, time, calcpropdelays(rx_pos,pos,c));
                difference = [];
                design_matrix = [];

                relvel = rx_vel + surfacevelocity(rx_pos, rx);

                %A MATRIX CONSTRUCTION
                for sat=1:satellites

                    sat_index = SV_id_array(sat);
                    %find satellite positions and velocities sat_pos = time_pos([sat_index*4-
1:sat_index*4+1]);
                    sat_pos = pos([sat_index*4-1:sat_index*4+1]);
                    sat_vel = vel([sat_index*4-1:sat_index*4+1]);
                    sat_rvel = time_vrad(sat_index*2+1);
                    sat_pseudo = time_pseudo(sat_index*2+1);

                    rho = sat_pos - rx_pos;
                    rho_mag = norm(rho);
                    rho_norm = rho ./ rho_mag;
                    vnorm = (sat_vel - relvel) ./ rho_mag;

                    difference_contribution = [
                            rho_mag - (sat_pseudo + rx_cdr);
                            dot(sat_pos-rx_pos,sat_vel-relvel) / rho_mag - (sat_rvel -
scale*rx_cdr);
                        ];

                    design_matrix_contribution = [
                        -rho_norm(1) -rho_norm(2) -rho_norm(3) -1;
                        cross(rho_norm, cross(rho_norm, vnorm)) scale;
                        ];

                    % add this satellite's contribution to the difference
                    % evaluation and design matrix matrices
                    difference = [difference; difference_contribution];
                    design_matrix = [design_matrix; design_matrix_contribution];

                end %end of A matrix construction

               %delta vector
                delta_pos = -inv(design_matrix'*design_matrix)*design_matrix'*difference;
                %increment our guess by delta vector
                rx_pos = rx_pos + delta_pos(1:3)';
                %set receiver offset
                rx_cdr = delta_pos(4);
                %convert to lat /long
                rx = latlong(rx_pos);
                %find magnitude of our correction term
                delta = norm(delta_pos(1:3));
                iterations = iterations + 1;

            end %end while loop for convergence

            s = size(rx_locations,2);

            if (s>1 && time_index > 1)
                rx_pos = rx_pos * exp_weight + (1-exp_weight)*rx_locations(:,s)';
            end
            rx_locations = [rx_locations rx_pos'];


        end%end for loop for times
```

```
        % calculate the accuracy and precision of the locations
        accuracy_meters = locdiffmag(rx_locations, ecef([ 42.444007 -76.482229 236.548 ]));

        %Update the DOP and errors - Note that the cutoff is used
        %only for plotting purposes, all errors are saved for correlation
        %computation
        Q = inv(design_matrix'*design_matrix);

        if(accuracy_meters < cutoff)
            cerror = [cerror accuracy_meters];
            DOP = [DOP sqrt(trace(Q))];
            good = [good; accuracy_meters sqrt(trace(Q)) allsat(SV_id_array)];
        else
            %bad satellite combinations are saved
            bad = [bad; allsat(SV_id_array)];
        end

end%end for loop for satellite combinations

%Plot the good combinations
subplot(2,1,1), plot(1:length(cerror), cerror), xlabel('Combo #'), ylabel('Error (m)');
subplot(2,1,2), plot(1:length(cerror), DOP), xlabel('Combo #'), ylabel('DOP');


%DATA ANALYSIS SECTION
cov = 1/length(DOP) * (cerror - mean(cerror))*(DOP - mean(DOP))';
corr_coeff = cov / (sqrt(var(cerror)) * sqrt(var(DOP)));

%Rank and Sort the satellite combinations by error
[ranked index] = sort(good(:,1));
ranked = [ranked good(index,2:size(SV_id_indices,2)+2)]
```

9.6. ecef.m (written by Dr. Paul Kintner)

```
% checked 8/2006
% ecef.m    (actual file name: ecef.m)
%
% this GPS utility converts a WGS-84 latitude-longitude-altitude
% position into ECEF coordinates
%
% input: 'location' vector which contains a position specified by
%       latitude (degrees), longitude (degrees), and altitude (meters)
%                           [ latitude longitude altitude ]
%
% output: 'ECEFxyz' vector which contains the same position
%       specified by ECEF coordinates (meters)
%                           [ ECEFx ECEFy ECEFz ]
%
function ECEFxyz = ecef(location);
% define physical constants
    constant;
% get latitude-longitude-altitude location to be converted to ECEF coordinates
    latdeg = location(1);
    londeg = location(2);
    alt = location(3);
% convert to radians
    lat = latdeg*degrad;        % convert latitude to radians
    lon = londeg*degrad;        % convert longitude to radians
% computes the ECEF coordinates
    NN =    (AA^4/((BB^2)*sin(lat)^2 + (AA^2)*cos(lat)^2))^(1/2);
    ECEFx = (NN+alt)*cos(lat)*cos(lon);     % meters
    ECEFy = (NN+alt)*cos(lat)*sin(lon);     % meters
    ECEFz = ((BB^2/AA^2)*NN + alt)*sin(lat);    % meters
% return location in ECEF coordinates
    ECEFxyz = [ ECEFx ECEFy ECEFz ];
    return
```

9.7. ephemposvel.m (based on findsat.m by Dr. Paul Kintner)

```
% Finds the positions and velocities of the satellites given in the input
% arguments.  Output coordinates are in ECEF, and output velocity is
% in a Earth-rotation-adjusted ECEF frame.
% Input:
%   ephem - standard ephemerides
```

```
%   t - GPS times (column) at which to find position/velocitiy
%   prop_delay - propagation delays for each satellite at each timestep
% Output:
%   pos - [GPSTimes [SVID1 X Y Z] [SVID2 X Y Z] ...]
%   vel - [GPSTimes [SVID1 VX VY VZ] [SVID2 VX VY VZ] ...]
function [ pos, vel ] = ephemposvel(ephem, t, prop_delay)
    constant;   % load GPS constants
    satellites = size(ephem, 1);
    times = size(t,1);
    pos = [t];
    vel = [t];
    for sat=1:satellites
    % set up the satellite identifiers
        SVs = ones(times,1) .* ephem(sat,1);
        pos = [pos SVs];
        vel = [vel SVs];
    % define orbital parameters
        t0 = ephem(sat,4);          % ephemeris reference time (seconds)
        ecc = ephem(sat,5);         % eccentricity (unitless)
        sqrta = ephem(sat,6);       % square root of semi-major axis (meters1/2)
        omega0 = ephem(sat,7);      % argument of perigee (radians)
        M0 = ephem(sat,8);          % mean anomaly at reference time (radians)
        l0 = ephem(sat,9);          % right ascension at reference (radians)
        omegaDot = ephem(sat,10);   % rate of right acension (radians/second)
        dn = ephem(sat,11);         % mean motion difference (radians/second)
        i0 = ephem(sat,12);         % inclination angle at reference time (radians)
        iDot = ephem(sat,13);       % inclination angle rate (radians/second)
        cuc = ephem(sat,14);        % latitude cosine harmonic correction (radians)
        cus = ephem(sat,15);        % latitude sine harmonic correction (radians)
        crc = ephem(sat,16);        % orbit radius cosine harmonic correction (meters)
        crs = ephem(sat,17);        % orbit radius sine harmonic correction (meters)
        cic = ephem(sat,18);        % inclination cosine harmonic correction (radians)
        cis = ephem(sat,19);        % inclination sine harmonic correction (radians)
        toc = ephem(sat,24);      % time of clock, ephemeris (seconds)
    % define time of position request and delta t from epoch; correct
    % for possible week crossovers; 604800 seconds in a GPS week
        dt = t - t0;
        idx = find(dt > 302400);    % if into the next week
        dt(idx) = dt(idx) - 604800;
        idx = find(dt < -302400);   % if into the previous week
        dt(idx) = dt(idx) + 604800;
    % adjust the time for the propagation delays
        if (nargin == 3)
            dt = dt - prop_delay(:,sat);
        end
    % calculate mean anomaly with corrections
        n = (sqrt(muearth) * (sqrta).^(-3)) + dn; % mean rotation speed
        M = M0 + n .* dt;
    % compute the eccentric anomaly from mean anomaly using
    % Newton-Raphson method to solve for 'E' in:
    %       f(E) = M - E + ecc * sin(E) = 0
        E = M;
       for i = 1:10
            f = M - E + ecc .* sin(E);
            dfdE = - 1 + ecc .* cos(E);
            dE = - f ./ dfdE;
            E = E + dE;
        end
    % calculate true anomoly from eccentric anomoly
        sinnu = sqrt(1 - ecc.^2) .* sin(E) ./ (1 - ecc .* cos(E));
        cosnu = (cos(E) - ecc) ./ (1 - ecc .* cos(E));
        nu = atan2(sinnu,cosnu);
    % calculate the argument of latitude and the argument of perigee
    % iteratively.
        omega = omega0;
       for i = 1:5
            u = omega + nu;
            cos2u = cos(2*u);
            sin2u = sin(2*u);
            omegaCorr = cuc.*cos2u + cus.*sin2u;
            omega = omega0 + omegaCorr;
       end
    % calculate longitude of ascending node with correction
        lcorr = omegaDot.*dt;
        l = l0 - OmegaE .* t + lcorr;
    % calculate orbital radius with correction
        rCorr = crc.*cos2u + crs.*sin2u;
```

```
        r = (sqrta.^2) .* (1 - ecc .* cos(E)) + rCorr;
    % calculate inclination with correction
        iCorr = iDot.*dt + cic.*cos2u + cis.*sin2u;
        i = i0 + iCorr;
    % find position in orbital plane
        u = omega + nu;
        xp = r .* cos(u);
        yp = r .* sin(u);
    % find satellite position in ECEF coordinates
        ECEFx = (xp .* cos(l)) - (yp .* cos(i) .* sin(l));
        ECEFy = (xp .* sin(l)) + (yp .* cos(i) .* cos(l));
        ECEFz = (yp .* sin(i));
    % append satellite locations
        pos = [ pos ECEFx ECEFy ECEFz ];

    % calculate velocity in the orbital frame
        % n is defined earliner (~line 47)
        vel_orbital_coef = ((n .* (sqrta .^ 4))./r);
        vel_orbital = [vel_orbital_coef .* -sin(E), vel_orbital_coef .* sqrt(1-ecc.^2) .* cos(E),
zeros(times,1)];
    % transform velocity at each time to the ECEF frame
        % this is the very ugly expansion of three rotation matrices:
        % rotZ(-l) * rotX(-i) * rotZ(-omega)
        vel_x_ecef = (cos(-l).*cos(-omega)-sin(-l).*cos(-i).*sin(-omega)).*vel_orbital(:,1)+(cos(-
l).*sin(-omega)+sin(-l).*cos(-i).*cos(-omega)).*vel_orbital(:,2)+sin(-l).*sin(-i).*vel_orbital(:,3);
        vel_y_ecef = (-sin(-l).*cos(-omega)-cos(-l).*cos(-i).*sin(-omega)).*vel_orbital(:,1)+(-sin(-
l).*sin(-omega)+cos(-l).*cos(-i).*cos(-omega)).*vel_orbital(:,2)+cos(-l).*sin(-i).*vel_orbital(:,3);
        vel_z_ecef = sin(-i).*sin(-omega).*vel_orbital(:,1)-sin(-i).*cos(-omega).*vel_orbital(:,2)+cos(-
i).*vel_orbital(:,3);
    % append the satellite velocity
        vel = [vel vel_x_ecef vel_y_ecef vel_z_ecef];
    end
    return;
end
```

9.8. formatData.m (written by Dr. Paul Kintner)

```
% checked 8/2006
% formatdata.m  (actual file name: formatda.m)
%
% this utility transforms the data contained in 'ephemData' and
% 'obsData' into more convenient structures: 'ephem' and 'obs'
%
% input: 'ephemData' matrix which rows contain satellite orbital
%       ephemerides
%           'obsData' matrix which rows contain a sample number. a GPS
%       time, and SV ids along with corresponding observables; either
%       pseudo-range or phase
%           'SV_ids' vector contains a list of SV ids which determine
%           which satellites data will be formated for
%
% output: 'ephem' matrix which rows contain satellite orbital
%       ephemerides; the rows are sorted by SV id order based on the
%       input 'SV_ids'; the following is a description of the 'ephem'
%       fields :
%           ephem(:,1)  SV number
%           ephem(:,2)  ephemeris reference week number
%           ephem(:,3)  ephemeris GPS reference time (seconds)
%           ephem(:,4)  ephemeris reference time of week (seconds)
%           ephem(:,5)  eccentricity
%           ephem(:,6)  square root of semi-major axis (meters1/2)
%           ephem(:,7)  argument of perigee (radians)
%           ephem(:,8)  mean anomaly at reference time (radians)
%           ephem(:,9)  right ascension at reference time (radians)
%           ephem(:,10) rate of right ascension (radians/second)
%           ephem(:,11) mean motion difference (radians/second)
%           ephem(:,12) inclination angle at reference time (radians)
%           ephem(:,13) inclination angle rate (radians/second)
%           ephem(:,14) latitude cosine harmonic correction (radians)
%           ephem(:,15) latitude sine harmonic correction (radians)
%           ephem(:,16) orbit radius cosine harmonic correction (meters)
%           ephem(:,17) orbit radius sine harmonic correction (meters)
%           ephem(:,18) inclination cosine correction (radians)
%           ephem(:,19) inclination sine correction (radians)
%           ephem(:,20) af0 clock correction (seconds)
%           ephem(:,21) af1 clock correction (seconds/second)
%           ephem(:,22) af2 clock correction (seconds/second2)
```

```
%           ephem(:,23) tgd clock correction (seconds)
%           ephem(:,24) toc time of clock ephemeris reference time
%           (seconds)
%            'obs' martix which rows contain a GPS time and then SV ids
%       followed by corresponding observables, either pseudo-range
%       (meters) or phase (cycles); SV ids are sorted according to the
%       input 'SV_ids'; the following is a description of the 'obs'
%       fields :
%           obs(:,1)    GPS time for sample (seconds)
%           obs(:,2)    SV identification
%           obs(:,3)    raw observable corresponding to SV id
%           obs(:,4)    SV identification
%           obs(:,5)    raw observable corresponding to SV id
%                           .
%                           .
%                           .
%
function [ephem, pseudo, dopp] = formatData(ephemData, obsData, obsDoppData, SV_ids)
% determine the number of satellites
    satellites = size(SV_ids,2);
% create 'ephem'
    for SV = 1:satellites
        r = find(ephemData(:,1) == SV_ids(SV));
        if (~isempty(r))
            ephem(SV,:) = ephemData(r,:);
        else
            ephem(SV,:) = zeros(1,24);
        end
    end
% remove all satellites with no ephemerides supplied
    idx = find(ephem(:,1) ~= 0);
    ephem = ephem(idx,:);
    SV_ids = SV_ids(idx);
    satellites = size(SV_ids,2);
% check if observable data has been supplied
    if (isempty(obsData))
        obs = [ ];
        return;
    end
% determine sample GPS times
    GPStime = obsData(:,2);
% determine number of samples
    samples = size(GPStime,1);
% create 'pseudo'
    SV_cols = [ ];
    for i = 1:((size(obsData,2) - 1) / 2)
                SV_cols = [ SV_cols 2 * i + 1];
    end
    pseudo = zeros(samples,2 * satellites + 1);
    for t = 1:samples
        pseudo(t,1) = GPStime(t);
        for SV = 1:satellites
            c = find(obsData(t,SV_cols) == SV_ids(SV)) * 2;
            pseudo(t,2 * SV) = SV_ids(SV);
            if (~isempty(c))
                pseudo(t,2 * SV + 1) = obsData(t,c + 2);
            end
        end
    end
    pseudo = pseudocalc(ephem,pseudo);
% create 'dopp'
    SV_cols = [ ];
    for i = 1:((size(obsDoppData,2) - 1) / 2)
                SV_cols = [ SV_cols 2 * i + 1];
    end
    dopp = zeros(samples,2 * satellites + 1);
    for t = 1:samples
        dopp(t,1) = GPStime(t);
        for SV = 1:satellites
            c = find(obsDoppData(t,SV_cols) == SV_ids(SV)) * 2;
            dopp(t,2 * SV) = SV_ids(SV);
            if (~isempty(c))
                dopp(t,2 * SV + 1) = obsDoppData(t,c + 2);
            end
        end
    end
% return 'ephem', 'pseudo', 'dopp'
```

```
    return
```

9.9. latlong.m (written by Dr. Paul Kintner)

```
% checked 8/2006
% latlong.m (actual file name: latlong.m)
%
% this GPS utility converts a position specified in ECEF coordinates
% into WGS-84 latitude-longitude-altitude coordinates
%
% input: 'location' vector which contains a position specified by
%       ECEF coordinates (meters)
%                           [ ECEFx ECEFy ECEFz ]
%
% output: 'ecoord' vector which contains the same position specified
%       by WGS-84 latitude (degrees), longitude (degrees), and
%       altitude (meters)
%                           [ latitude longitude altitude ]
%
function ecoord = latlong(location);
% define physical constants
    constant;
% get ECEF location to be converted to latitude-longitude-altitude
% coordinates
    ECEFx = location(1);
    ECEFy = location(2);
    ECEFz = location(3);
% compute the longitude which is an exact calculation
    long = atan2( ECEFy, ECEFx );   % radians
% compute the latitude using iteration
    p = (ECEFx^2 + ECEFy^2)^(1/2);
    % compute approximate latitude
    lat0 = atan( (ECEFz/p)/(1-esquare) ); % guess using h=0 approximation
    stop = 0;
    while (stop == 0)
        N0 = (AA^4/((BB^2)*sin(lat0)^2 + (AA^2)*cos(lat0)^2))^(1/2);
        altitude = p/cos(lat0) - N0;        % meters
        % calculate improved latitude
        term = 0;
        lat = atan( (ECEFz/p) * ((1 - (esquare*N0)/(N0+altitude)) ^ (-1)) );
        % check if result is close enough,
        if (abs(lat - lat0) < 1e-12)
            stop = 1;
        end
        lat0 = lat;
    end
% convert the latitude and longitude to degrees
    latitude = lat0 * 180/pi;       % degrees
    longitude = long * 180/pi;      % degrees
% return location in latitude-longitude-altitude coordinates
    ecoord = [ latitude longitude altitude ];
    return
```

9.10. loaddata.m

```
function [ephem, pseudo, dopp] = loaddata(SV_ids)

ephemData = load('ephem.asc');
obsData = load('obs.asc');
doppData = load('obsdopp.asc');
[ephem, pseudo, dopp] = formatData(ephemData, obsData, doppData, SV_ids);

end
```

9.11. locdiff.m

```
function locations = locdiff(locations, center)
% difference each location from the first so that we get relative locations
length = size(locations,2);
locations = locations - [ones(1,length) * center(1);
                         ones(1,length) * center(2);
                         ones(1,length) * center(3); ];
locations = locations(:,[2:length]);

end
```

9.12. locdiffmag.m

```
function result = locdiffmag(locations, center)
result = sqrt(mean(sum(locdiff(locations, center) .^ 2,1)));
end
```

9.13. pseudoCalc.m (written by Dr. Paul Kintner)

```
%tested on 9/2006
% pseudoCa.m    (actual file name: pseudoCa.m)
%
% < DOES NOT INCLUDE IONOSPHERIC CORRECTIONS >
%
% this function calculates the pseudo-ranges based on raw
% pseudo-ranges with clock corrections applied
%
% input: 'ephem' matrix which rows contain orbital ephemerides for
%       a given satellite; including satellite time correction terms
%                   < see formatData.m for description >
%           'pseudo' matrix which rows contain pseudo-range samples
%       for a given time
%                   < see formatData.m for description >
%                   < in this case, psuedo = obs structure >
%
% output: 'pseudo-range' matrix which rows contain a GPS time
%       (seconds), and then pairs of SV id numbers with corresponding
%       corrected pseudo-ranges (meters)
%                       [ GPStime svID pr svID pr ... ;
%                         GPStime svID pr svID pr ... ;
%                                        ...
%                         GPStime svID pr svID pr ... ]
%
% < DOES NOT INCLUDE IONOSPHERIC CORRECTIONS >
%
function pseudo_range = pseudocalc(ephem,pseudo)
% define physical constants
    constant;
% clear variable 'pseudo_range'
    pseudo_range = [ ];
% determine time samples
    GPStime = pseudo(:,1);
% determine number of samples taken
    samples = size(pseudo,1);
% determine number of satellites being used
    satellites = size(ephem,1);
% get clock correction parameters from 'ephem'
    refTime = ephem(:,24);
    af0 = ephem(:,20);
    af1 = ephem(:,21);
    af2 = ephem(:,22);
    tgd = ephem(:,23);
% create 'pseudo_range' by correcting pseudo-ranges of each raw
% pseudo-range measurement for each time sample
    for t = 1:samples
      % determine pseudo-range corrections due to satellite clock
      % corrections calculate time offset from satellite reference
      % time
            timeOffset = GPStime(t) - refTime;%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % calculate clock corrections 'cc'
        %%%%% NOTE: this is the satellite clock correction
            cc =  af0 + af1.*(timeOffset) + af2.*(timeOffset.^2) - tgd;%%%
      % calculate change in raw pseudo-range due to clock
      % corrections
            clockCorr = c.*cc;
        % calculate total pseudo-range correction
            pseudoCorr = clockCorr;
      % apply corrections to pseudo-range measurements and add
      % samples into 'pseudo_range' structure
            sample = GPStime(t);
            for i = 1:satellites
                pseudoR = pseudo(t,2 * i + 1);
                if (pseudoR ~= 0)
                    corrPseudoR = pseudoR + pseudoCorr(i);
                else
                    corrPseudoR = 0;
                end
                sample = [ sample ephem(i,1) corrPseudoR ];
```

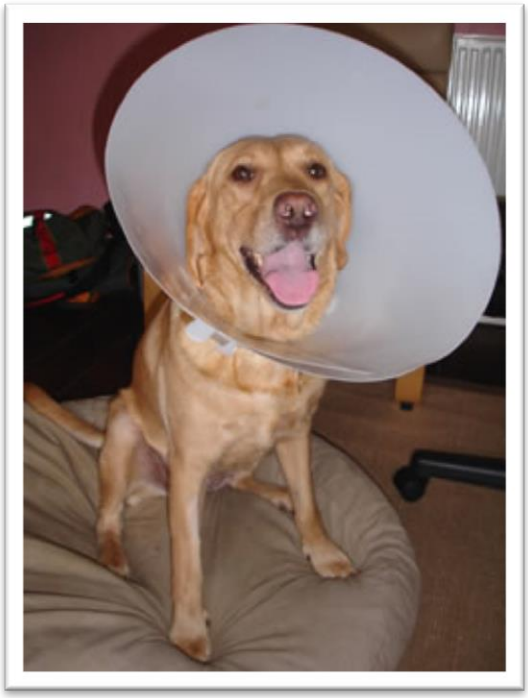
```
            end
            pseudo_range = [ pseudo_range; sample ];
    end
% return corrected pseudo-ranges
    return;
```

9.14. surfacevelocity.m

```
% Calculates the correction to apply to a receiver's observation
% of satellite velocity due to Earth's rotation.
% Input:
%   position - the position of the receiver relative to the earth's center
%   latlongalt - the position of the receiver in latitude/longitude relative
%             to the center of the spherical earth model.  coordinates in
%             degrees and altitude in meters.
function [ vel ] = surfacevelocity(position, latlongalt)
constant;
N = latlongalt(3) + sqrt(AA.^4/(BB.^2*sin(latlongalt(1)*degrad).^2+AA.^2*cos(latlongalt(1)*degrad).^2));
vel = OmegaE * cos(latlongalt(1)*degrad) * N * [-sin(latlongalt(2)*degrad), cos(latlongalt(2)*degrad),
0];
end
```

10. The Dogpler Cone

http://www.diaryofawebsite.com/photos/conehead.jpg



11. Pseudoramster

http://farm3.static.flickr.com/2172/2203720240_9fed55ae90_o.jpg