Spring 1-1-2019

A Nanosecond-Level Time Accuracy Method Using Assisted-GPS Techniques with Variable Time Ambiguity Analysis

Ryan Christopher Blay
University of Colorado at Boulder, rcblay@gmail.com

A Nanosecond-level Time Accuracy Method using Assisted-GPS Techniques with Variable Time Ambiguity Analysis

by

R. C. Blay

B.S., University of Colorado Boulder, 2019

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Masters of Science

Department of Aerospace Engineering Sciences

2019

This thesis entitled:
A Nanosecond-level Time Accuracy Method using Assisted-GPS Techniques with Variable Time Ambiguity Analysis
written by R. C. Blay
has been approved for the Department of Aerospace Engineering Sciences

Prof. Dennis Akos

Dr. Nagaraj Channarayapatna Shivaramaiah

Date ________________

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Blay, R. C. (M.S., Aerospace Engineering Sciences)

A Nanosecond-level Time Accuracy Method using Assisted-GPS Techniques with Variable Time Ambiguity Analysis

Thesis directed by Prof. Dennis Akos

In this thesis, a brief overview of the GPS L1 signal structure and GPS position and time estimation is given. Then, van Diggelen's A-GPS method is discussed with his millisecond ambiguity approach. Next, a way to accurately use any ambiguity time to reach nanosecond-level accuracy is developed. Experiments are then run to quantify the sensitivity of A-GPS to a priori errors in position and time, as well as the performance of the first sample time error as a function of ambiguity time. It is found that the safe zone for A-GPS is less than 35 km initial position error and less than 30 seconds initial time error. It is also found that an ambiguity time of 200 ms converges to nanosecond-level accuracy 99 % of the time. Most first sample time errors are within one nanosecond while none exceed two nanoseconds as compared to a traditional GPS L1 SDR.

Acknowledgements

First and foremost, I would like to thank my advisor Dennis Akos for his constant encouragement, patience, and support for my goals.

I would also like to thank all of my fellow researchers in the lab during my time who made working a fulfilling experience: Nagaraj Channarayapatna Shivaramaiah, Esteban Garbin, Daniel Torvik, Sara Hrbek, Damian Miralles, Dong-Kyeong Lee, Nathan Levigne, Jorge Cervantes, Alexander Ray, Max Nystrom, and Luka Strizic.

Contents

Tables

Figures

Chapter 1

Introduction

These days, everyone wants the most benefit for the least work, and rightly so. Sometimes, to achieve this, a little assistance is required. In a large busy parking lot, it can take a long time to find a parking spot. However, with just a quick hint from a leaving car, the search reduces dramatically, and the empty white painted lines ensure the parking is well done. Assisted-GPS is like the quick hint, and this research will dive into the accuracy provided by the white painted lines. To further this analogy past the point of no return, the white painted lines provide a reference mark that you can use to calibrate your parking. For GPS, this reference mark is provided by the L1 C/A code, the navigation data, and the data subframe start. These three references are mile markers on the road that provide known time information when all else is lost.

1.1 Previous Work

For the GPS signal structure, and especially the receiver details, Borre et al.'s GPS receiver book [4] provided much of the details and figures shown in the initial theory sections.

Frank van Diggelen's A-GPS book [5] provided the majority of the inspiration and details of this research. In his book, van Diggelen describes the assistance data in detail and then develops the navigation equations with coarse time. He then shows his novel approach to fix the millisecond integer ambiguity. This book was also very important in understanding the sensitivity to a priori position and time estimate errors, which this research will also discuss. van Diggelen also briefly discusses the results when navigation bit synchronization is achieved but does not describe this application at length.

Besides van Diggelen's technique, there are other papers that describe different approaches to finding position without navigation bit decoding. Sirola and Syrjarinne in [8, 9] propose a range-fitting cost function over the space of position and coarse time using an iterative approach. Syrjarinne and Akopian also formulate a time-recovery method but use optimatizion and/or Extended Kalman Filters [2, 10]. Glennon and Bryant propose two different approaches for accurate time recovery with a focus on the single-stage algorithm which closely resembles the approach taken in this thesis [6]. Agarwal et al. in their review also approach the problem of time-stamp recovery in which they discuss several different approaches as well as the navigation-bit alignment that will be discussed in this thesis [1]. Muthuraman et al. also review several different time-recovery algorithms but do not approach the same level of accuracy as shown in other approaches [7]. Finally, Akopian and Syrjarinne return in 2009 to propose the time recovery approach that will be shown in this thesis for navigation bit alignment and show the performance at different signal strengths [3].

1.2 Thesis's Focus

In this thesis, the GPS signal structure and the GPS software receiver will be briefly described. Then, the A-GPS technique as described by van Diggelen will be discussed. Next, the approach to finding a much more accurate time estimate will be described. Instead of characterizing the performance of navigation bit alignment, this research will study the amount of time ambiguity required for nanosecond level time accuracy. Finally, experiments will be performed to find the allowable a priori errors and the performance of the accurate time estimation method.

Chapter 2

Background

2.1 GPS L1 Signal Structure

A brief description of the GPS L1 signal structure will be given below. However, the approach shown in this research applies to any GNSS signal with data present.

All GPS signals reside in the L band within the UHF band which ranges from 1 to 2 GHz. L1 and L2 frequencies are derived from a common frequency f_0 of 10.23 MHz and are shown below.

$$f_{L1} = 154 f_0 = 1575.42 \text{ MHz}$$

$$f_{L2} = 120 f_0 = 1227.60 \text{ MHz}$$

Both signals are comprised of the carrier, the navigation data, and a spreading sequence. For L1, the spreading sequence is the coarse acquisition code (C/A) and for L2, it is the encrypted precision code (P(Y)). The C/A code is a sequence of 1023 chips with one chip every 977 nanoseconds, giving it a chipping rate of 1.023 MHz. The P(Y) code has a longer sequence of around $2.35 * 10^{14}$ chips with a chipping rate of 10.23 MHz that restarts every week and is only a segment of the larger P(Y) code. In Fig. 2.1, the GPS signal generation is described.

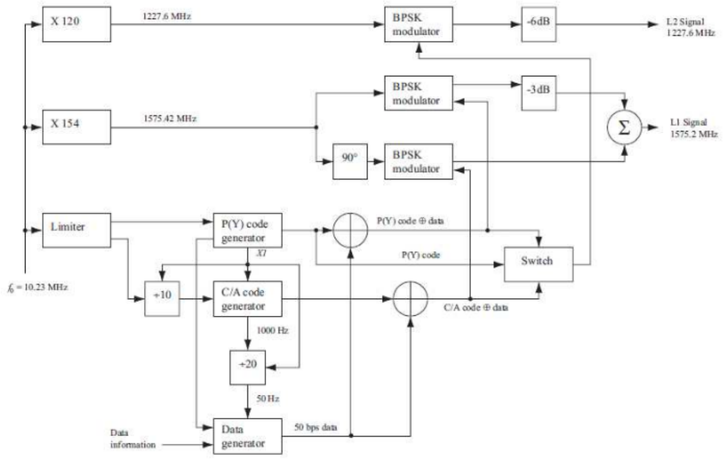


Figure 2.1: GPS signal generation diagram [4]

As shown, the C/A code is only modulated onto the L1 signal, while the P(Y) signal is modulated on L1 and L2. From this, the signal transmitted from the satellite is described as

$$s^k(t) = \sqrt{2P_C}(C^k(t) \oplus D^k(t))cos(2\pi f_{L1}t) \\ + \sqrt{2P_{PL1}}(P^k(t) \oplus D^k(t))sin(2\pi f_{L1}t) + \sqrt{2P_{PL2}}(P^k(t) \oplus D^k(t))sin(2\pi f_{L2}t) \quad (2.1)$$

where P_C, P_{PL1}, and P_{PL2} are the signal powers with C/A or P code, C_k is the C/A code sequence of satellite number k, P_k is the P(Y) code sequence for satellite number k, D_k is the navigation data, and f_{L1} and f_{L2} are the L1 and L2 carrier frequencies. Below in Fig. 2.2 are the three L1 signal components.

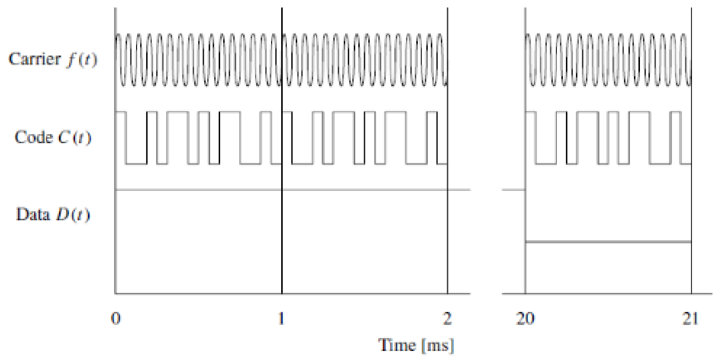


Figure 2.2: GPS signal as function of time [4]

As can be seen, the carrier signal has a much higher frequency while the code repeats every millisecond and a navigation data bit consists of 20 code periods. Both the code and the navigation bit will be very important as time ambiguities in the following A-GPS development.

2.1.1 C/A Codes

C/A codes are pseudo-random noise sequences known as Gold codes. Each satellite has a different Gold code which is created using linear feedback shift registers to result in a sequence of 1's and -1's. This process is shown below in Fig. 2.3.

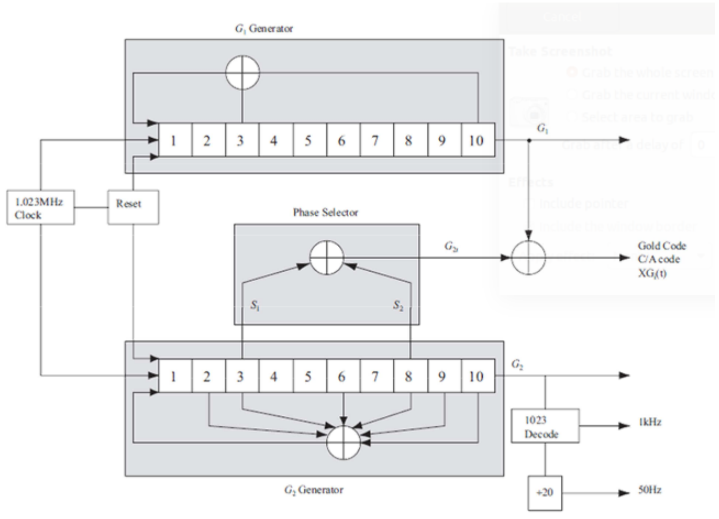


Figure 2.3: Gold sequence generation algorithm [4]

As seen above, the phase selector chooses two different phases of the G_2 generator to modulo-2 add and then this is modulo-2 added to the G_1 output to produce the unique PRN code for the satellite. Gold codes are used for their correlation properties where there is a high value at auto-correlation, and near zero correlation for any chip shift.

2.1.2 Navigation Data Bits

As previously mentioned, the navigation data is modulated onto the GPS L1 signal with a rate of 50 bits per second. Importantly, this means that every 20 milliseconds, there is a new navigation bit. However, it does not mean that every 20 milliseconds it will transition from a +1 to a -1. This will become an important detail in the following sections regarding time ambiguity. An entire navigation message is transmitted over 12.5 minutes comprised of 25 frames, each 30 seconds long. Each frame has 1500 bits and is further separated into five subframes of 300 bits. This is visualized in Fig. 2.4 below.

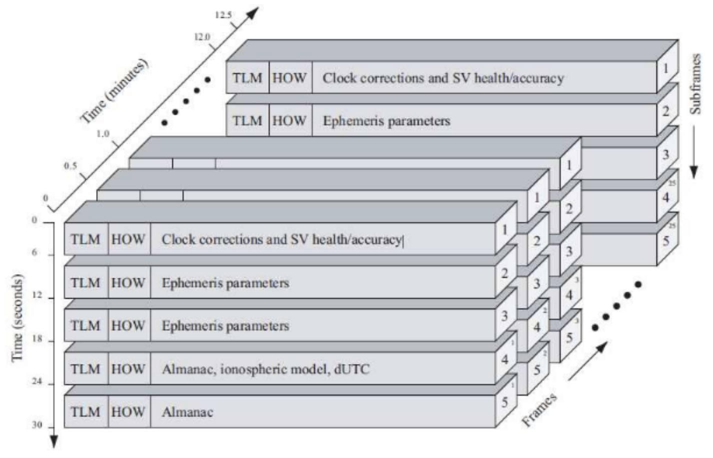


Figure 2.4: Navigation message structure [4]

Every subframe begins with a telemetry word (TLM) and handover word (HOW). The TLM word is used for frame synchronization with its 8-bit preamble. The HOW word contains a 17-bit truncated time-of-week (TOW) followed by flags and the subframe number. Each subframe consists of different details of the navigation message. The first subframe contains clock corrections and health information for the transmitting satellite. The second and third subframe contain the ephemeris parameters which give the satellite orbit and position parameters. Finally, the fourth and fifth subframes contain the almanac data which contains useful support information, such as ionospheric model parameters and UTC parameters.

2.2 GPS Position/Time Estimation

In order to calculate the position and time of a user receiver, there needs to be at least four satellites. At first glance, it would seem that only three should be required to find the receiver position but in reality the user clock is not accurate and must also be estimated. In Fig. 2.5, this is simply visualized.

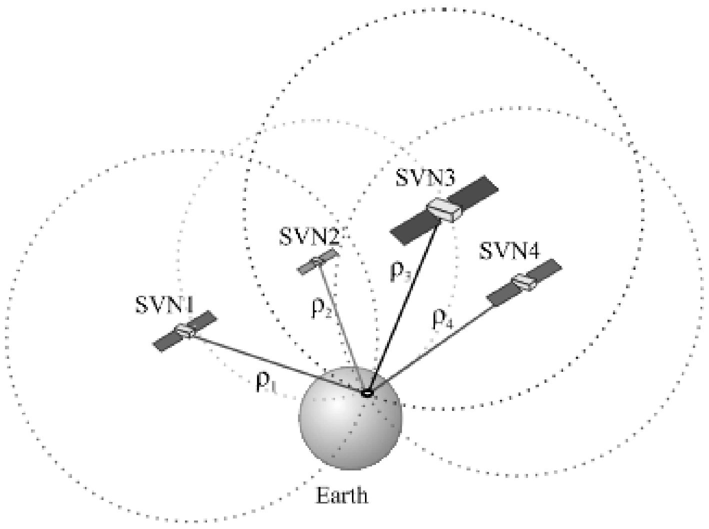


Figure 2.5: GPS solution estimation w/four satellites [4]

The calculation of the position centers around the time it takes between the signal's transmission and reception for each satellite. This time multiplied by the speed of light gives the pseudorange. In an ideal situation, this pseudorange would be the geometrical range ρ from the satellite to the receiver and the location could be triangulated. However, this is not the case, and there are multiple error sources that have to be estimated or taken care of in order to estimate the position and time.

2.2.1 Pseudorange Measurement Model

The major error sources consist of the receiver clock error dt_i, the satellite clock error dt_k, the troposphere T_i^k, the ionosphere I_i^k, and other errors, like multipath, combined into e_i^k. Below in Eq. 2.2, the relation of pseudorange P_i^k and geometric range ρ_i^k is shown with these error sources.

$$P_i^k = \rho_i^k + c(dt_i - dt^k) + T_i^k + I_i^k + e_i^k \tag{2.2}$$

where the subscript i refers to the receiver. The geometrical range is calculated as:

$$\rho_i^k = \sqrt{(X^k - X_i)^2 + (Y^k - Y_i)^2 + (Z^k - Z_i)^2} \tag{2.3}$$

where X^k, Y^k, Z^k is the satellite position and X_i, Y_i, Z_i is the receiver position.

2.2.2 Iterative Least Squares Approach

Inserting Eq. 2.3 into Eq. 2.2 and linearizing the result gives Eq. 2.4 shown below.

$$P_i^k = \rho_{i,0}^k - \frac{X^k - X_{i,0}}{\rho_{i,0}^k}\Delta X_i - \frac{Y^k - Y_{i,0}}{\rho_{i,0}^k}\Delta Y_i - \frac{Z^k - Z_{i,0}}{\rho_{i,0}^k}\Delta Z_i + c(dt_i - dt^k) + T_i^k + I_i^k + e_i^k \tag{2.4}$$

This equation has four unknowns of ΔX_i, ΔY_i, ΔZ_i, and dt_i. To minimize the error term e_i^k, an iterative least squares approach is used. Thus, Eq. 2.4 needs to be arranged in the form of Eq. 2.5.

$$A\boldsymbol{x} = \boldsymbol{b} \tag{2.5}$$

where each element of b is equal to

$$b_i^k = P_i^k - \rho_{i,0}^k + cdt^k - T_i^k - I_i^k - e_i^k \tag{2.6}$$

and is then combined into

$$A\boldsymbol{x} = \begin{bmatrix} -\frac{X^1 - X_{i,0}}{\rho_{i,0}^1} & -\frac{Y^1 - Y_{i,0}}{\rho_{i,0}^1} & -\frac{Z^1 - Z_{i,0}}{\rho_{i,0}^1} & 1 \\ -\frac{X^2 - X_{i,0}}{\rho_{i,0}^2} & -\frac{Y^2 - Y_{i,0}}{\rho_{i,0}^2} & -\frac{Z^2 - Z_{i,0}}{\rho_{i,0}^2} & 1 \\ -\frac{X^3 - X_{i,0}}{\rho_{i,0}^3} & -\frac{Y^3 - Y_{i,0}}{\rho_{i,0}^3} & -\frac{Z^3 - Z_{i,0}}{\rho_{i,0}^3} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ -\frac{X^m - X_{i,0}}{\rho_{i,0}^m} & -\frac{Y^m - Y_{i,0}}{\rho_{i,0}^m} & -\frac{Z^m - Z_{i,0}}{\rho_{i,0}^m} & 1 \end{bmatrix} \begin{bmatrix} \Delta X_{i,1} \\ \Delta Y_{i,1} \\ \Delta Z_{i,1} \\ cdt_{i,1} \end{bmatrix} = \boldsymbol{b} - \boldsymbol{e} \tag{2.7}$$

which is then iteratively solved until it converges. Convergence is usually when the state corrections' norm is within a certain tolerance, e.g. 1×10^{-8}.

2.3 GPS Software Receiver

To go from raw data to a converged position solution, the GPS software defined radio (SDR) comes into play. The software receiver is useful for having full visibility of the GPS data and its corresponding acquisition, tracking, and navigation results. It allows for much more flexibility and for modifications to be possible. The very first component of the GPS software receiver is the data input, or front-end. Each GPS satellite transmits its signal as described above for the GPS antenna to receive. The signal then goes through a multitude of filtering, amplification, and sampling to become the IF sampled signal that the GPS SDR accepts. This is visualized in a block diagram in Fig. 2.6 below.

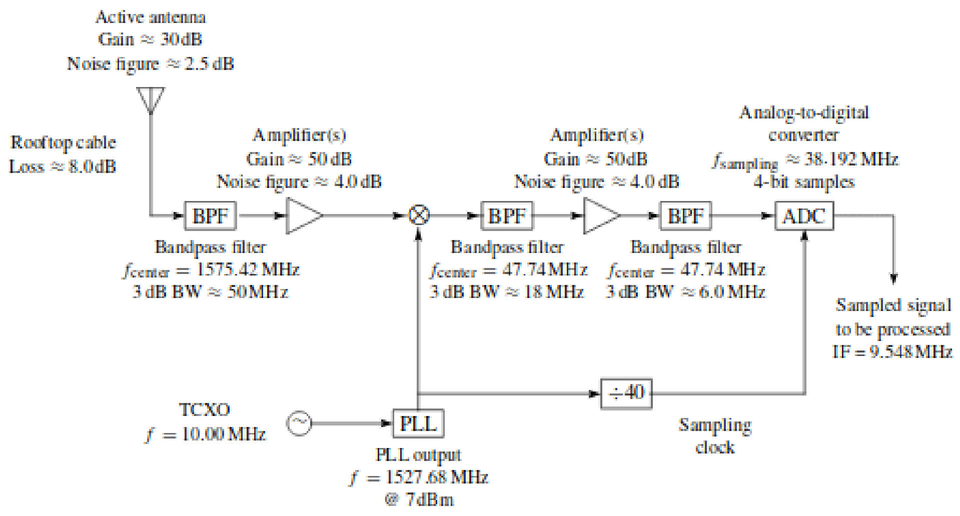


Figure 2.6: Receiver front-end block diagram [4]

The most important characteristics of the data are the sampling frequency, the intermediate frequency (IF), and whether the signal is complex or real. Once the data is passed in, the signal is acquired and tracked before calculating the position solution. Each of these steps will now be explained in further detail.

2.3.1 Acquisition

Acquisition is essentially the search for satellite PRN codes. The purpose of acquiring is to determine which satellites are visible and their coarse carrier frequencies and code phases. There are 32 different PRN sequences, each corresponding to a satellite, that acquisition searches for. The main challenge of acquiring GPS signals is that the signal power is below that of thermal noise, as seen in Fig. 2.7.

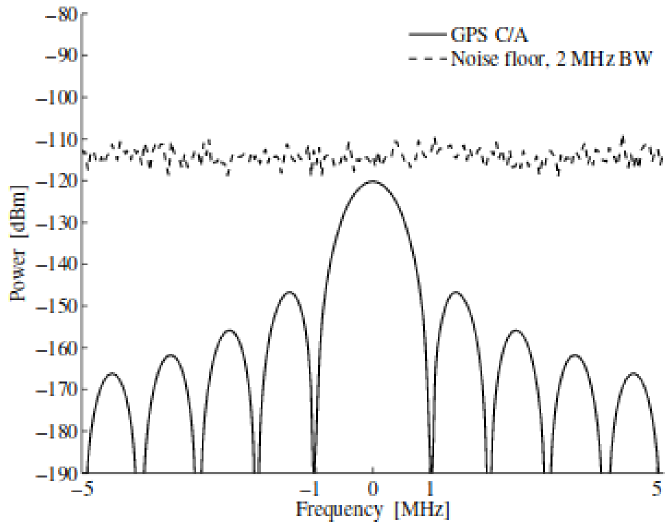


Figure 2.7: GPS signal under noise [4]

This is where the correlation properties of the PRN codes come into play. If the data is correlated with a replica of a present satellite's PRN code, then the resulting spike will be strong enough to overcome the noise (if the signal is strong enough). However, the code phase of the data's PRN code for that satellite is unknown so each different possible delay needs to be tested. Adding to that, the movement of the satellite causes the signal's frequency to change and thus the many different doppler frequencies also need to be tested for a high correlation value. This results in a search space of doppler frequency and code phase, as shown in Fig. 2.8.

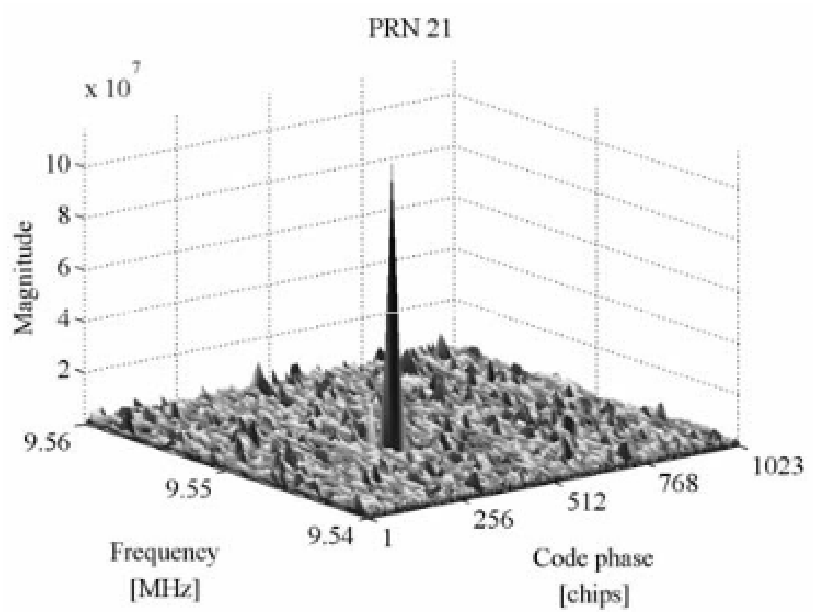


Figure 2.8: Acquisition peak [4]

If the magnitude of the peak is strong enough, then that satellite is acquired and passed to tracking. In their book, Borre et al. compare several different acquisition algorithms, finding that the parallel code phase search is the fastest but most complex [4].

2.3.2 Tracking

Given all the visible satellites and their rough carrier frequencies and code phases, it is now up to tracking to refine these values and be able to demodulate the navigation data from the satellite's signal. In order to understand how the navigation data is retrieved, the signal starting from the front-end will be examined. In the equation below, the output after filtering and down-converting is shown:

$$s^k(t) = \sqrt{2P_C}C^k(t)D^k(t)\cos(\omega_{IF}t) + \sqrt{2P_{PL1}}P^k(t)D^k(t)\sin(\omega_{IF}t) \tag{2.8}$$

This signal is then sampled by the ADC and is described by:

$$s^k(n) = C^k(n)D^k(n)\cos(\omega_{IF}n) + e(n) \tag{2.9}$$

where n $= 1/f_s$ and thus is now discrete in time. To extract the data $D^k(n)$, the signal must be modulated down to baseband by removing the IF carrier $\cos(\omega_{IF}n)$ and the code $C^k(n)$. Removing the carrier just requires multiplying the signal by a replica with same carrier frequency and applying a low-pass filter. Then the signal is

$$LHS = \frac{1}{2}C^k(n)D^k(n) \tag{2.10}$$

Then by lining up the code replica exactly, the data can be extracted as shown in the equation below.

$$\sum_{n=0}^{N-1} C^k(n)C^k(n)D^k(n) = N\sum_{n=0}^{N-1} D^k(n) \tag{2.11}$$

where $N\sum_{n=0}^{N-1} D^k(n)$ is the navigation data multiplied by the number of points N in the signal. In order to have the demodulation done accurately, the code phase and the carrier phase have to be tracked. Fig. 2.9 shows a visualization of how the code is tracked.

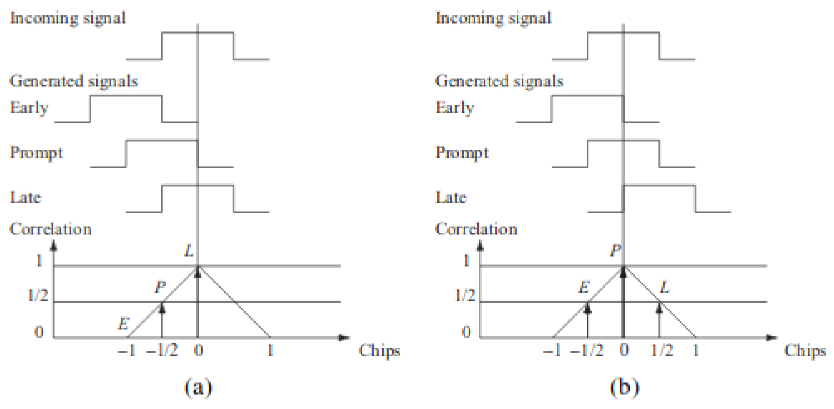


Figure 2.9: Early Prompt Late tracking diagram [4]

Essentially, there are three spaced correlators that check the correlation value at their chip location. In this case, they are spaced with half a chip distance. The goal is to have the prompt correlator match up exactly with the zero offset, while the early and late are on either side, ideally symmetric at half correlation. This is visualized simply in Fig. 2.10.

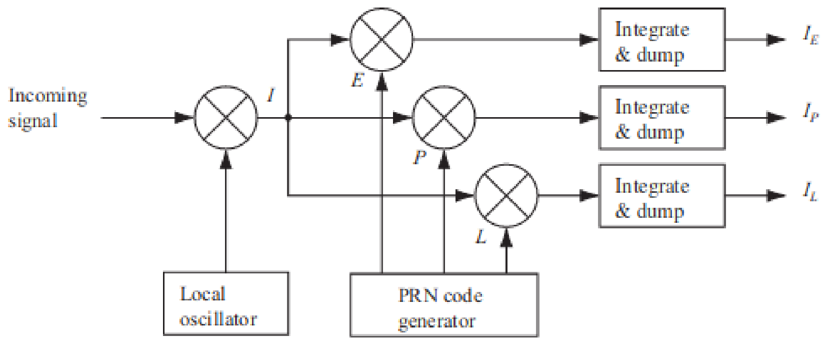


Figure 2.10: Simplified loop at EPL tracking loop [4]

As shown, the PRN code generator inputs into the early, prompt, and late correlators and is mixed with the incoming signal with the carrier removed. This would be ideal if the phase was locked but since there is phase error, it is necessary to correlate the in-phase and quadrature arms separately, as shown in Fig. 2.11.

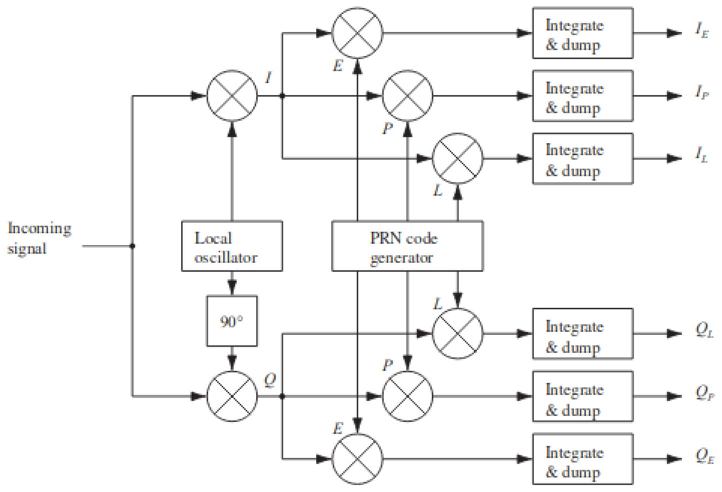


Figure 2.11: I/Q tracking loop with EPL diagram [4]

Importantly, the tracking loop needs a feedback discriminator to let it know how to adjust itself so that the prompt correlator lines up with zero offset. Borre et al. looked at several different discriminators and find that the one below is good at tracking noisy signals of different strengths.

$$D = \frac{(I_E^2 + Q_E^2) - (I_L^2 + Q_L^2)}{(I_E^2 + Q_E^2) + (I_L^2 + Q_L^2)} \tag{2.12}$$

In order to track the carrier phase/frequency, a slightly different approach is required, as shown in Fig. 2.12.

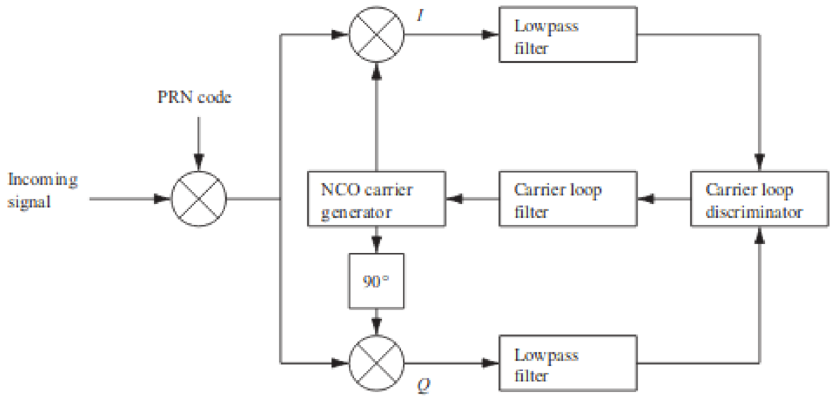


Figure 2.12: Carrier tracking diagram [4]

The above figure details a Costas loop. The goal of a Costas loop is to keep all of the energy in the in-phase arm. If the code replica is aligned perfectly then the in-phase arm results in the following sum:

$$D^k(n)cos(\omega_{IF}n)cos(\omega_{IF}n + \varphi) = \frac{1}{2}D^k(n)cos(\varphi) + \frac{1}{2}D^k(n)cos(2\omega_{IF}n + \varphi) \tag{2.13}$$

where φ is the phase offset. The quadrature arm results in the following:

$$D^k(n)cos(\omega_{IF}n)sin(\omega_{IF}n + \varphi) = \frac{1}{2}D^k(n)sin(\varphi) + \frac{1}{2}D^k(n)sin(2\omega_{IF}n + \varphi) \tag{2.14}$$

After these signals are lowpass-filtered, then the following equations result:

$$I^k = \frac{1}{2} D^k(n) cos(\varphi) \tag{2.15}$$

$$Q^k = \frac{1}{2} D^k(n) sin(\varphi) \tag{2.16}$$

Thus the feedback discriminator for the Costas loop is found by calculating the phase error as shown below.

$$\frac{Q^k}{I^k} = \frac{\frac{1}{2} D^k(n) sin(\varphi)}{\frac{1}{2} D^k(n) cos(\varphi)} = \tan\varphi \tag{2.17}$$

$$\varphi = \tan^{-1}(\frac{Q^k}{I^k}) \tag{2.18}$$

Thus, when all the energy is in the in-phase arm, the phase error is zero. Combining the carrier and code tracking loops results in the block diagram shown in Fig. 2.13.

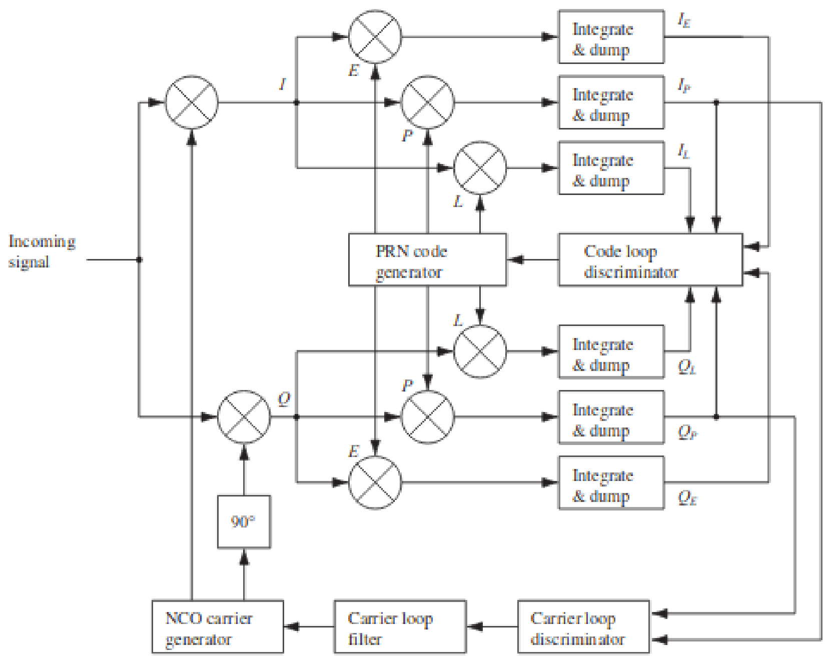


Figure 2.13: Carrier/Code combined tracking loop diagram [4]

Fig. 2.13 is the optimized combination where the multiplications are shared between the two loops. In Fig. 2.14, the flow diagram is shown in how the receiver tracks all the satellite signals.

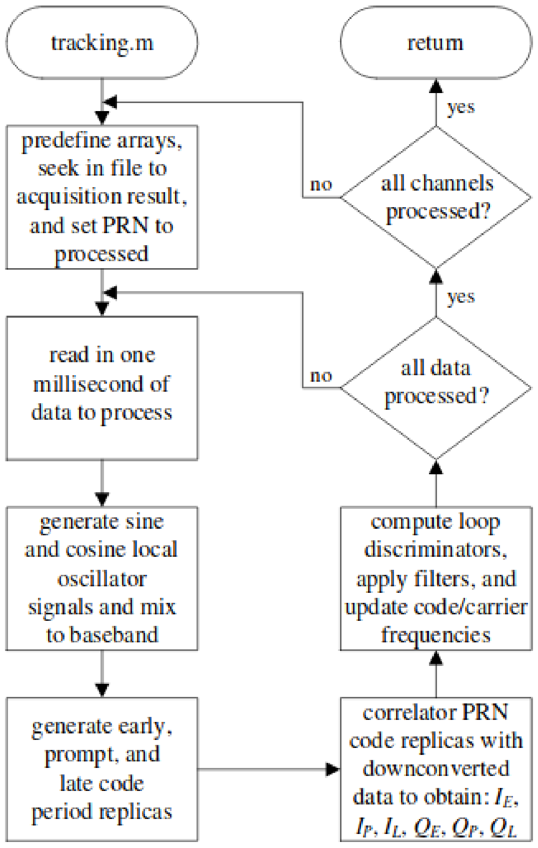

tracking.m
predefine arrays, seek in file to acquisition result, and set PRN to processed
read in one millisecond of data to process
generate sine and cosine local oscillator signals and mix to baseband
generate early, prompt, and late code period replicas
correlator PRN code replicas with downconverted data to obtain: I_E, I_P, I_L, Q_E, Q_P, Q_L
compute loop discriminators, apply filters, and update code/carrier frequencies
all data processed?
no
yes
all channels processed?
no
yes
return

Figure 2.14: Tracking block diagram [4]

With all the satellites tracked, the navigation data is successfully extracted and can now be parsed. This will lead to being able to estimate the position and other navigation parameters.

2.3.3 Navigation (Position Estimation)

Finally, the position can be estimated. First though, the navigation bits have to be correctly parsed from the prompt I correlator data. As shown in Fig. 2.15, the transitions have to be identified accurately so that there is integrity of the bit sequence where the y-axis shows the prompt-I correlation data as explained above.

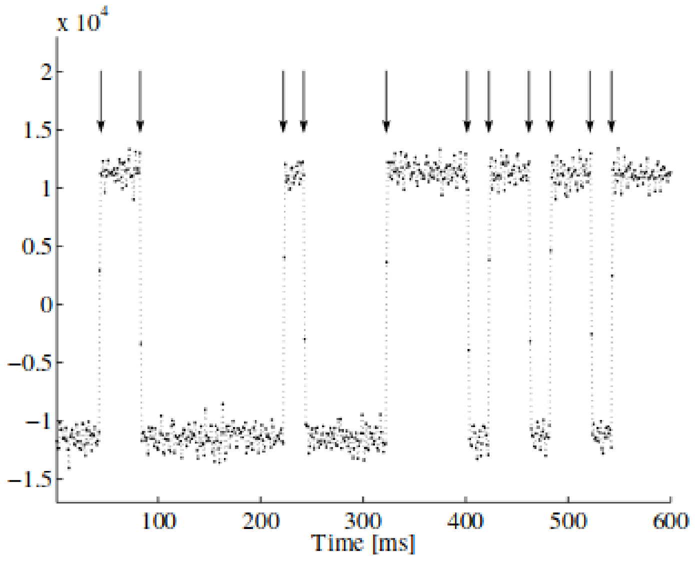


Figure 2.15: Navigation bit transition diagram [4]

Once the prompt I is converted into 1's and -1's, it can be parsed. The receiver first searches for the preamble in the TLM word, and then finds the first sub-frame location in the data and the TOW. The contents of the TLM and HOW words are shown in Fig. 2.16.

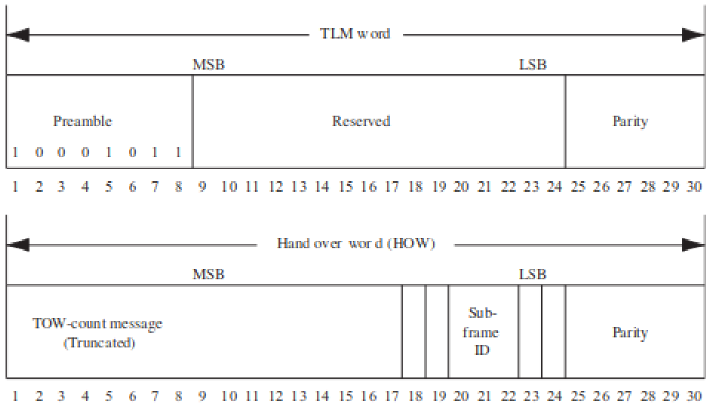


Figure 2.16: TLM and HOW words in navigation message structure [4]

After that, the TOW is used to find the time of reception. Next, the pseudoranges are calculated, the satellites are positioned, and the least squares iteratively finds the position as explained above.

Calculating the pseudoranges consists of finding the difference between the time of transmission and time of reception and is visualized in the flow diagram in Fig. 2.17.

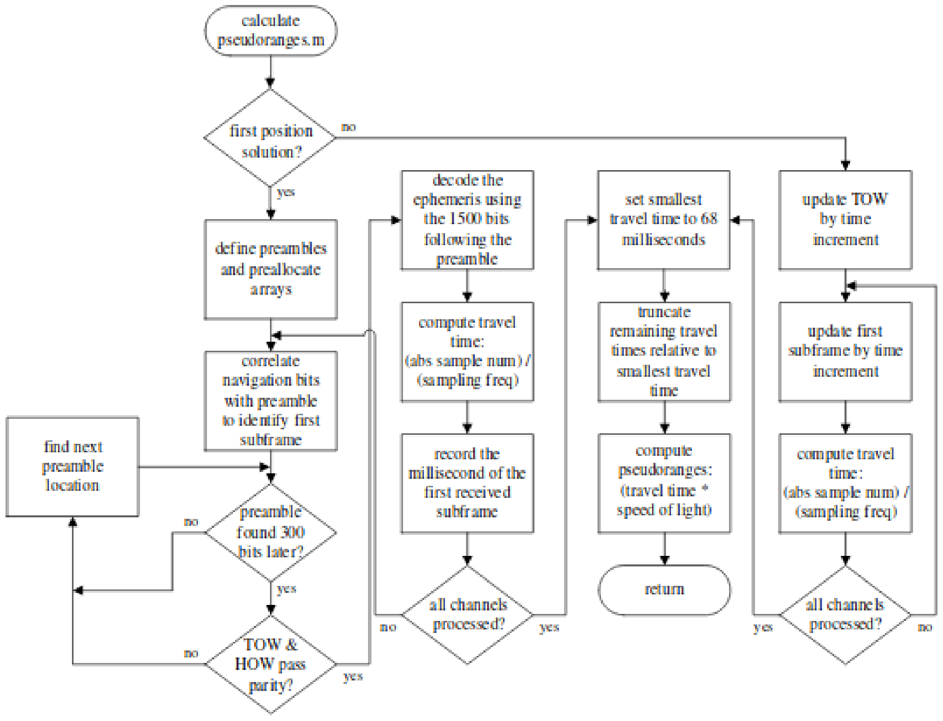

calculate pseudoranges.m
first position solution?
no
yes
define preambles and preallocate arrays
correlate navigation bits with preamble to identify first subframe
find next preamble location
preamble found 300 bits later?
no
yes
TOW & HOW pass parity?
no
yes
decode the ephemeris using the 1500 bits following the preamble
compute travel time: (abs sample num) / (sampling freq)
record the millisecond of the first received subframe
all channels processed?
no
yes
set smallest travel time to 68 milliseconds
truncate remaining travel times relative to smallest travel time
compute pseudoranges: (travel time * speed of light)
return
update TOW by time increment
update first subframe by time increment
compute travel time: (abs sample num) / (sampling freq)
all channels processed?
yes
no

Figure 2.17: Calculate Pseudoranges block diagram [4]

If the receiver is calculating the first position solution, then the first step is to search for the preamble and make sure that it is repeated every 300 bits. The TLM and HOW are decoded and passed through a parity check before decoding the entire ephemeris. The millisecond is then recorded of the first subframe and is repeated for each channel. After all channels are processed, the travel times are calculated and used to find the pseudoranges.

Finally, the least squares algorithm's flow diagram is shown below in Fig. 2.18.

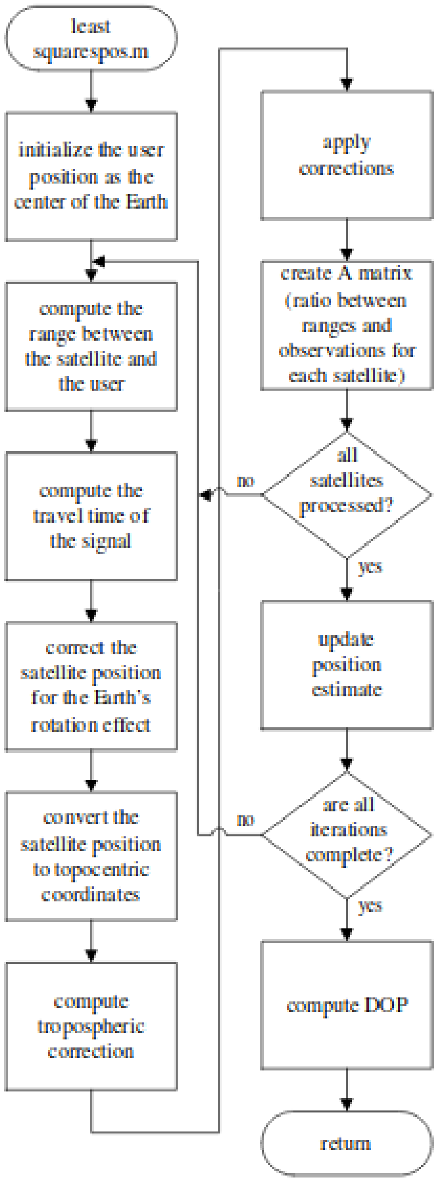

least squarespos.m
initialize the user position as the center of the Earth
compute the range between the satellite and the user
compute the travel time of the signal
correct the satellite position for the Earth's rotation effect
convert the satellite position to topocentric coordinates
compute tropospheric correction
apply corrections
create A matrix (ratio between ranges and observations for each satellite)
all satellites processed?
no
yes
update position estimate
are all iterations complete?
no
yes
compute DOP
return

Figure 2.18: Least Squares block diagram [4]

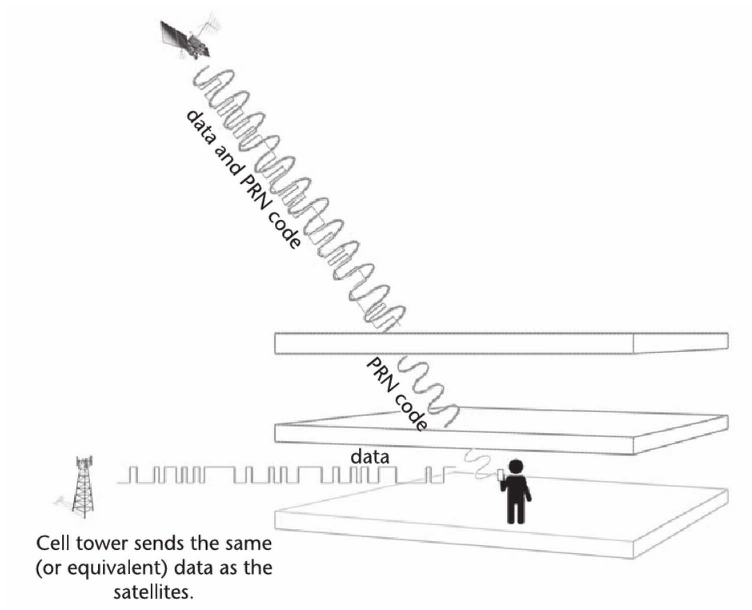
First, the user's position is initialized at the center of the Earth. The range is then computed between the user's estimated position and the satellite positions at the time of transmission. The travel time is derived from the range and used to correct the satellite position due to the Earth's rotation in that time. The atmospheric corrections are then applied. The A matrix and the b vector are then found as each satellite is processed. The position correction is calculated and the state is updated until it converges.

This is performed for every navigation measurement until a position estimate and time estimate is found. The corrected sampling frequency and time of first sample is then calculated.

Chapter 3

Assisted-GPS

The goal of Assisted-GNSS is to get accurate estimates of position and time with much less data. The receiver needs to still receive satellite measurements but can now process them in a much quicker time, or process much weaker signals. Below in Fig. 3.1, an overview of A-GPS is visualized.



data and PRN code
PRN code
data
Cell tower sends the same (or equivalent) data as the satellites.

Figure 3.1: A-GPS overview [5]

Essentially, the satellites still send the data and PRN code but because of obstruction, the receiver loses the ability to parse the data. Instead, a local cell tower sends the equivalent data to the receiver so that it can still process the satellites. More generally, A-GPS provides much more information and consists of an entire reference network as shown in Fig. 3.2.

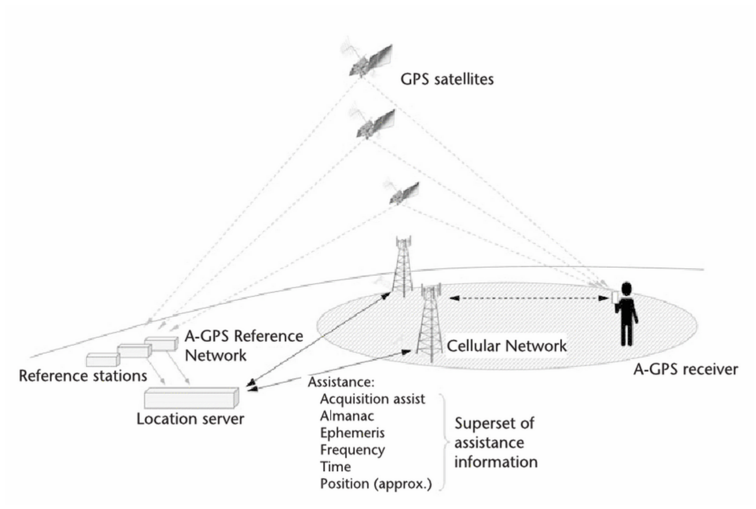


Figure 3.2: A-GPS network assistance [5]

The assistance can consist of the PRN's to search for with the acquired code phase and doppler frequencies, the ephemeris and almanac, and the approximate position and time of the receiver. The question still remains of how to use this assistance data to solve for the receiver's position and time. As explained above, the time of transmission and time of reception are needed to calculate the pseudoranges. Without either time, the pseudoranges must be estimated and improved on, as described below.

3.1 Five-State Least Squares Approach to Coarse Time Estimation

To investigate this problem, the coarse-time error shall be defined as the difference between the true receiver time and the estimated receiver time. Looking at the effect of adding a coarse-time error, first consider the pseudorange residuals as used in the least squares algorithm:

$$\boldsymbol{\delta z}^{(k)} = \boldsymbol{z}^{(k)} - \hat{\boldsymbol{z}}^{(k)} \tag{3.1}$$

where $z^{(k)}$ is the actual pseudorange for satellite k, and $\hat{z}^{(k)}$ is the predicted pseudorange and is given by

$$\hat{\boldsymbol{z}}^{(k)} = \left|\boldsymbol{x}^{(k)}(\hat{t}_{tx}) - \boldsymbol{x}_{xyz0}\right| - \boldsymbol{\delta}_t^{(k)}(\hat{t}_{tx}) + b_0 \tag{3.2}$$

and $\hat{t}_{tx}$ is the estimated time of transmission of the satellite signal we are measuring, $\boldsymbol{x}^{(k)}(\hat{t}_{tx})$ is the calculated satellite position at time $\hat{t}_{tx}$, $\boldsymbol{x}_{xyz0}$ is the a priori receiver position, $\boldsymbol{\delta}_t^{(k)}(\hat{t}_{tx})$ is the satellite clock error in units of length at time $\hat{t}_{tx}$, and b_0 is the a priori estimate of the common bias in units of length. However, the problem is that the time $\hat{t}_{tx}$ is in error, so the normal four-state least squares algorithm can not solve for the receiver position and common bias. When $\hat{t}_{tx}$ is incorrect, so is $\boldsymbol{x}^{(k)}(\hat{t}_{tx})$ and $\boldsymbol{\delta}_t^{(k)}(\hat{t}_{tx})$. For each one second of error, the pseudoranges would be incorrect on the order of 800 m [5]. There is hope though, as we know the velocities of the satellites, and thus the pseudorange rate. Below in Eq. 3.3, the difference of estimated pseudorange at the correct and incorrect transmission time is shown.

$$\begin{aligned}\hat{\boldsymbol{z}}^{(k)}(\hat{t}_{tx}) - \hat{\boldsymbol{z}}^{(k)}(t_{tx}) &= \hat{\boldsymbol{z}}^{(k)}(\hat{t}_{tx}) - \hat{\boldsymbol{z}}^{(k)}(\hat{t}_{tx} + \delta_{tc}) \\ &= -\nu^{(k)}\delta_{tc}\end{aligned} \tag{3.3}$$

where δ_{tc} is the coarse-time error, t_{tx} is the actual time of transmission, $\hat{t}_{tx}$ is the coarse-time estimate of the time of transmission, $\nu^{(k)} = (\boldsymbol{e}^{(k)} \cdot \boldsymbol{v}^{(k)} - \dot{\delta}_t^{(k)})$ is the pseudorange rate, $\boldsymbol{e}^{(k)}$ is the unit vector from the receiver to the satellite k, $\boldsymbol{v}^{(k)}$ is the satellite velocity vector, and $\dot{\delta}_t^{(k)})$ is the satellite-clock error rate in units of length/time. So now there is a term that takes care of the coarse-time error in the pseudoranges. The state-update vector is now

$$\boldsymbol{\delta x} = \begin{bmatrix} \delta_x \\ \delta_y \\ \delta_z \\ \delta_b \\ \delta_{tc} \end{bmatrix} \tag{3.4}$$

and the pseudorange residuals for each satellite are

$$\boldsymbol{\delta z}^{(k)} = \boldsymbol{z}^{(k)} - \hat{\boldsymbol{z}}^{(k)} = -\boldsymbol{e}^{(k)} \cdot \boldsymbol{\delta x}_{xyz} + \boldsymbol{\delta}_b + \nu^{(k)} \cdot \boldsymbol{\delta}_{tc} + \boldsymbol{\varepsilon}^{(k)} \tag{3.5}$$

with $\boldsymbol{\varepsilon}^{(k)}$ encapsulating all atmospheric and other errors. Stacking the pseudorange residuals for all satellites leads to the following:

$$\boldsymbol{\delta z} = \mathbf{H}\boldsymbol{\delta x} + \boldsymbol{\varepsilon} \tag{3.6}$$

with

$$\mathbf{H} = \begin{bmatrix} -\boldsymbol{e}^{(1)} & 1 & \boldsymbol{\nu}^{(1)} \\ \vdots & \vdots & \vdots \\ -\boldsymbol{e}^{(K)} & 1 & \boldsymbol{\nu}^{(K)} \end{bmatrix} \tag{3.7}$$

where K is the number of satellites. Since there are now five columns of the H matrix, there now must be at least five satellites to solve for $\boldsymbol{\delta x}$. Thus, a closed-form five-state solution has been developed to solve for position without precise time. However, there is one issue that has yet to be addressed. The measured pseudoranges for each satellite will have a millisecond integer ambiguity error. Explaining and fixing this error will be discussed in the section below. Assuming it is resolved, Fig. 3.3 shows the block diagram for the five-state least squares algorithm.

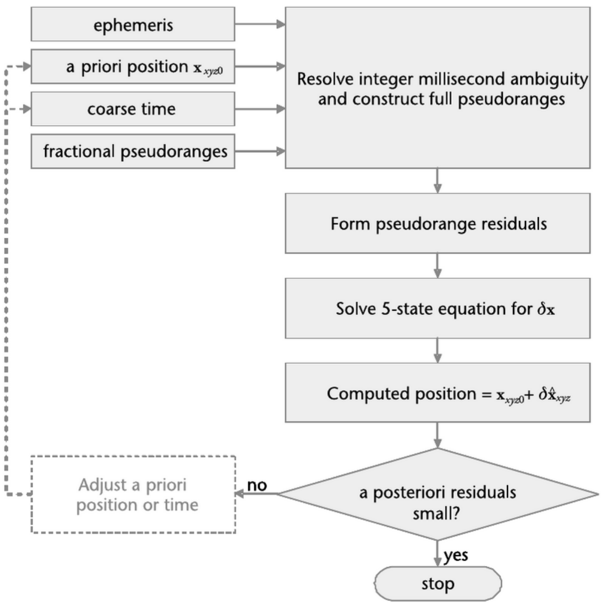

ephemeris
a priori position $\mathbf{x}_{xyz0}$
coarse time
fractional pseudoranges
Resolve integer millisecond ambiguity and construct full pseudoranges
Form pseudorange residuals
Solve 5-state equation for $\delta\mathbf{x}$
Computed position = $\mathbf{x}_{xyz0} + \delta\hat{\mathbf{x}}_{xyz}$
a posteriori residuals small?
no
yes
Adjust a priori position or time
stop

Figure 3.3: Five-state navigation solution block diagram [5]

The a posteriori residual check is to make sure that the computed position and time actually fit the given satellite positions and is usually not a concern if the a priori error is not too large.

3.2 Fixing the Millisecond Ambiguity

Since the full pseudorange is not available because the subframes are not decoded, there is a problem with correctly identifying the number of milliseconds of each pseudorange. Below in Fig. 3.4, a tape measure analogy visually explains the difference between the full pseudorange and the sub-millisecond pseudorange that is available.

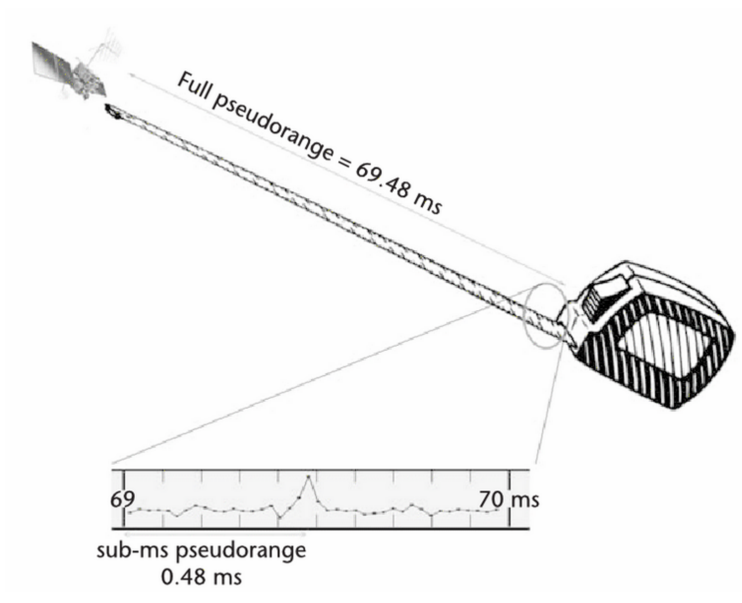


Figure 3.4: Sub-millisecond pseudorange visualization [5]

Unlike the image above, the amount of milliseconds is unclear. This is because there is uncertainty in the satellite positions, the common bias, and the satellite clock errors. The real problem occurs when the common bias causes an integer rollover. van Diggelen explores the benign and malign cases in depth [5]. So to solve for the millisecond ambiguity, the van Diggelen technique will be described. The first step is to find the sub-millisecond pseudoranges given by the code phase measurements, either from acquisition or tracking. Next, a reference satellite (usually the strongest) is chosen. Using the ephemeris and the a priori state, the full pseudorange is estimated and the millisecond integer $\boldsymbol{N}^{(0)}$ is assigned. The 0 represents the reference while k will represent any other satellite. Thus, the reference satellite has full pseudorange $\boldsymbol{N}^{(0)} + \boldsymbol{z}^{(0)}$ where $\boldsymbol{z}^{(0)}$ is the sub-millisecond pseudorange. The assigned integer $\boldsymbol{N}^{(0)}$ will be used to reconstruct the the other satellites' integers in a way that removes the possibility of integer rollover. Keep in mind, the reconstructed full pseudorange has an implicit common bias and is related to the actual geometric range by the following:

$$\begin{aligned} \boldsymbol{N}^{(0)} + \boldsymbol{z}^{(0)} &= \boldsymbol{r}^{(0)} - \boldsymbol{\delta}_t^{(0)} + b + \boldsymbol{\varepsilon}^{(0)} \\ &= \hat{\boldsymbol{r}}^{(0)} - d^{(0)} - \boldsymbol{\delta}_t^{(0)} + b + \boldsymbol{\varepsilon}^{(0)} \end{aligned} \tag{3.8}$$

where $\boldsymbol{r}^{(0)}$ is the unknown actual geometric range, $\hat{\boldsymbol{r}}^{(0)}$ is the estimated geometric range from the a priori position at the a priori time of transmission, $d^{(0)}$ is the error in $\hat{\boldsymbol{r}}^{(0)}$ caused by the error in the a priori position and time, $\boldsymbol{\delta}_t^{(0)}$ is the satellite clock error, b is the common bias, and $\boldsymbol{\varepsilon}^{(0)}$ is the measurement errors that includes atmospheric and thermal errors. The goal is to assign integers $N^{(k)}$ such that they share the same common bias. If this is possible, then for satellite k the full pseudorange would be:

$$\begin{aligned} \boldsymbol{N}^{(k)} + \boldsymbol{z}^{(k)} &= \boldsymbol{r}^{(k)} - \boldsymbol{\delta}_t^{(k)} + b + \boldsymbol{\varepsilon}^{(k)} \\ &= \hat{\boldsymbol{r}}^{(k)} - d^{(k)} - \boldsymbol{\delta}_t^{(k)} + b + \boldsymbol{\varepsilon}^{(k)} \end{aligned} \tag{3.9}$$

where b is the same common bias as for the reference satellite. Subtracting satellite k's pseudorange by the reference satellite's, the following is found:

$$(\boldsymbol{N}^{(k)}+\boldsymbol{z}^{(k)})-(\boldsymbol{N}^{(0)}+\boldsymbol{z}^{(0)}) = (\hat{\boldsymbol{r}}^{(k)}-d^{(k)}-\boldsymbol{\delta}_t^{(k)}+b+\boldsymbol{\varepsilon}^{(k)})-(\hat{\boldsymbol{r}}^{(0)}-d^{(0)}-\boldsymbol{\delta}_t^{(0)}+b+\boldsymbol{\varepsilon}^{(0)}) \tag{3.10}$$

and solving for $\boldsymbol{N}^{(k)}$:

$$\boldsymbol{N}^{(k)} = \boldsymbol{N}^{(0)}+\boldsymbol{z}^{(0)}-\boldsymbol{z}^{(k)}+(\hat{\boldsymbol{r}}^{(k)}-d^{(k)}-\boldsymbol{\delta}_t^{(k)}+b+\boldsymbol{\varepsilon}^{(k)})-(\hat{\boldsymbol{r}}^{(0)}-d^{(0)}-\boldsymbol{\delta}_t^{(0)}+b+\boldsymbol{\varepsilon}^{(0)}) \tag{3.11}$$

It is clear that the common bias b cancels exactly. And, if the magnitude of $(-d^{(k)} + \boldsymbol{\varepsilon}^{(k)} + d^{(0)} - \boldsymbol{\varepsilon}^{(0)})$ is less than 0.5 light-milliseconds (about 150 km), then the correct value of $\boldsymbol{N}^{(k)}$ is given by:

$$\boldsymbol{N}^{(k)} = round(\boldsymbol{N}^{(0)} + \boldsymbol{z}^{(0)} - \boldsymbol{z}^{(k)} + (\hat{\boldsymbol{r}}^{(k)} - \boldsymbol{\delta}_t^{(k)}) - (\hat{\boldsymbol{r}}^{(0)} - \boldsymbol{\delta}_t^{(0)})) \tag{3.12}$$

where all variables are known, and no matter the value of $\boldsymbol{N}^{(0)}$, the values for $\boldsymbol{N}^{(k)}$ will have the same common bias. A block diagram summarizing the calculation of the pseudorange milliseconds is shown in Fig. 3.5.

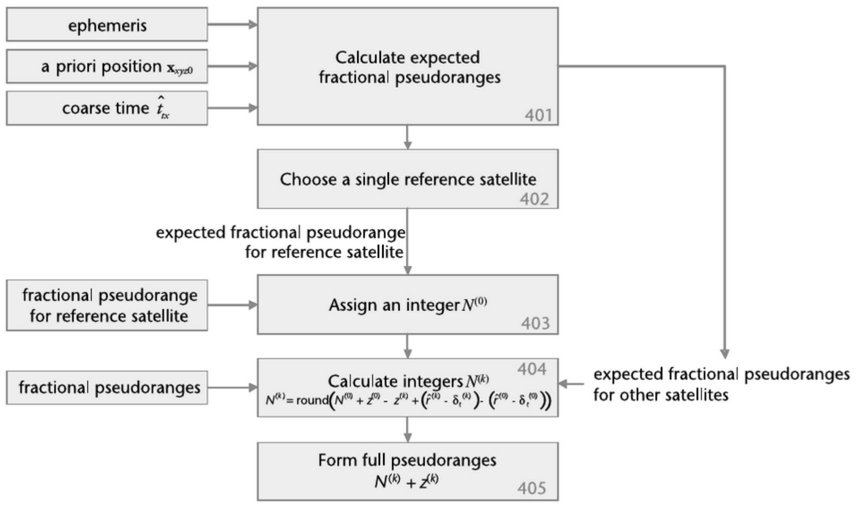

ephemeris
a priori position x_{xyz0}
coarse time $\hat{t}_{tx}$
Calculate expected fractional pseudoranges
401
Choose a single reference satellite
402
expected fractional pseudorange for reference satellite
fractional pseudorange for reference satellite
Assign an integer $N^{(0)}$
403
fractional pseudoranges
Calculate integers $N^{(k)}$
$N^{(k)} = \mathrm{round}\left(N^{(0)} + z^{(0)} - z^{(k)} + \left(\hat{r}^{(k)} - \delta_t^{(k)}\right) - \left(\hat{r}^{(0)} - \delta_t^{(0)}\right)\right)$
404
expected fractional pseudoranges for other satellites
Form full pseudoranges
$N^{(k)} + z^{(k)}$
405

Figure 3.5: Millisecond ambiguity block diagram [5]

With the full pseudoranges, the five-state navigation algorithm can be implemented as shown above in Fig. 3.3.

3.3 Changes to the GPS Software Receiver

To implement Assisted-GPS in the software receiver based off of Borre, Akos, et al.'s book, there are a few changes in acquisition and calculating PVT solutions. In acquiring the satellites, there is now no need to search for all the different PRN codes. The visible satellites are passed to the A-GPS code with the code phases and doppler frequencies in order to acquire all the visible satellites much more quickly. This is visually represented in Fig. 3.6 below.

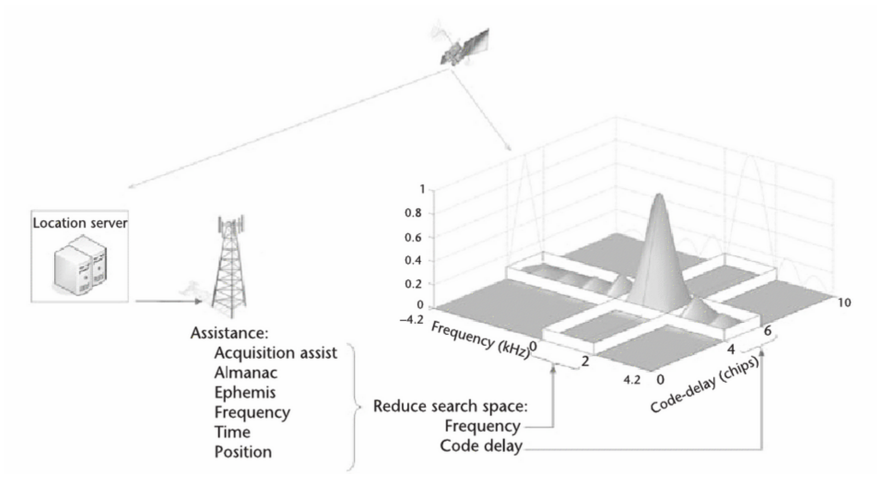


Figure 3.6: A-GPS reduced acquisition search space [5]

Even with just the visible satellites, the search time is reduced significantly. This reduced time could allow for more time to pick up weak signals instead. In calculating the navigation solutions, there is now no need to decode the ephemeris which removes the requirement to have a full frame of data before being able to calculate the receiver's position and time. Instead, the ephemeris is passed in with the a priori position and time and the five-state least squares algorithm described earlier is used. The only other detail is that the estimated transmission time is found by subtracting the estimated a priori pseudoranges from the a priori coarse time estimate.

Chapter 4

Accurate Timing Resolution

Even though the five-state least squares converges and gives a good coarse time, the timing can still be tens of milliseconds off depending on the mitigation of error sources. The problem is that it is very difficult to ascertain if the satellites sent their signals at one millisecond or the next since the pseudorange changes on the order of meters. So the least squares will converge but because of the other errors, it is most likely not the true time. Below in Fig. 4.1, the GPS time milestones are shown.

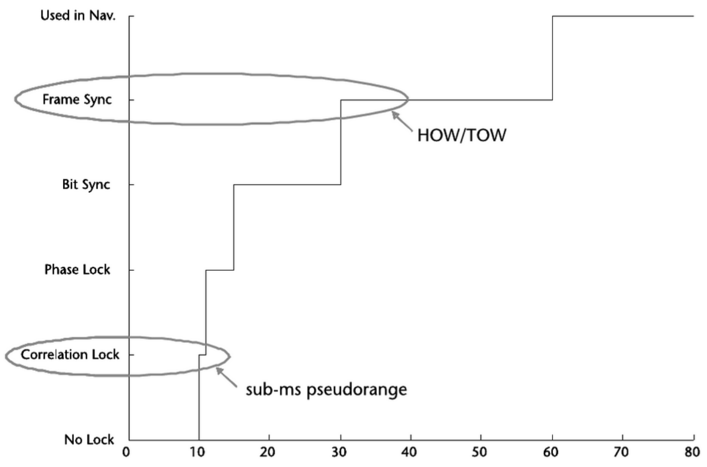


Figure 4.1: GPS time milestones as a function of time [5]

The goal is to use as little data as possible.

4.1 TOW-Decoded A-GNSS (6sec+)

One of the methods to provide good timing involves decoding the TOW word. However, this requires upwards of six seconds in order to ensure that the TOW word is in the data to decode. Because of this, this thesis will instead focus on using the data bit transition.

4.2 Data-bit Transition A-GPS

Using the data bit transitions is more challenging. First, there has to be enough data to ensure that a data-bit transition is present. This can be upwards of 100 milliseconds to have a strong probability of transition. Also required is that the five-state least squares algorithm's coarse time error from the true time is less than 10 milliseconds. This is not guaranteed for all geometries or situations, especially when atmospheric errors are not mitigated. Assuming the error is less than 10 milliseconds, the time of transmission for satellite k is estimated as:

$$\hat{t}_{tx}^{(k)} = \hat{t}_c - \left|\boldsymbol{x}^{(k)}(\hat{t}_{tx}) - \boldsymbol{x}_{xyz}\right| / c + \boldsymbol{\delta}_t^{(k)} \tag{4.1}$$

where $\hat{t}_c$ is the coarse time estimate returned by the five-state least squares algorithm, $\boldsymbol{x}^{(k)}(\hat{t}_{tx})$ is the satellite's position at time $\hat{t}_{tx}$, c is the speed of light, and $\boldsymbol{\delta}_t^{(k)}$ is the satellite clock correction.

If the navigation bit transition is known, then the number of milliseconds from each satellite's bit transition is calculated. Every bit transition occurs at what the satellite thinks is a multiple of 20 milliseconds when it transmits the signal, i.e at 1.02 s, 1.04 s, 1.06 s etc. Therefore, the number of milliseconds from this bit transition gives the amount of milliseconds from a time with a modulus of 20 milliseconds. So, if for a satellite the number of milliseconds away from a bit transition was 7, then the possible transmission times could be 1.013 s, 1.033 s, 1.053 s and likewise for every 20 milliseconds. The concern is how to pick the correct 20 millisecond time since it is ambiguous. This is where the assumption that the coarse time is correct within 10 milliseconds is important. If true, then the following equation will round to the correct 20 millisecond:

$$t_{tx_{20ms}}^{(k)} = round(\hat{t}_{tx}^{(k)}/20ms + N_m^{(k)}/20) * 20ms \tag{4.2}$$

where $\hat{t}_{tx}^{(k)}$ is the estimated transmission time given by Eq. 4.1, and $N_m^{(k)}$ is the integer number of milliseconds from the satellite k's bit transition. With the correctly rounded 20 millisecond transmission time, the number of milliseconds $N_m^{(k)}$ can now be used to find the transmission of the current millisecond (i.e. without sub-millisecond correction) as:

$$t_{tx_{1ms}}^{(k)} = t_{tx_{20ms}}^{(k)} - N_m^{(k)} \times 10^{-3} \tag{4.3}$$

Finally, to find the transmission time with the sub-millisecond correction, the code phase is used as shown below:

$$t_{tx}^{(k)} = t_{tx_{1ms}}^{(k)} + (1 - z^{(k)}) \times 10^{-3} \tag{4.4}$$

where $z^{(k)}$ is the code phase for the k$^{\text{th}}$ satellite in terms of a fraction of a chip. The calculated transmission time is now as close to the correct transmission time that can be achieved and is summarized in the equation below:

$$t_{tx}^{(k)} = round(\hat{t}_{tx}^{(k)}/20ms + N_m^{(k)}/20) * 20ms + (1 - z^{(k)} - N_m^{(k)}) \times 10^{-3} \tag{4.5}$$

With this time of transmission, the error in the previously computed transmission time $\hat{t}_{tx}^{(k)}$ can be found for each satellite:

$$\boldsymbol{\delta}_{t_{tx}}^{(k)} = t_{tx}^{(k)} - \hat{t}_{tx}^{(k)} \tag{4.6}$$

Ideally, this value would be equal for all satellites, but because of slight errors in the code phase or other sources, they have slight variations. However, they should definitely all be within one millisecond of each other. Now, the accurate coarse time estimate is found by adding the average of $\boldsymbol{\delta}_{t_{tx}}^{(k)}$ for all satellites to the estimate of coarse time, as shown in the equation below:

$$t_c = \hat{t}_c + \boldsymbol{\delta}_{t_{tx}} \tag{4.7}$$

This is the very accurate coarse time estimate. For even more accuracy, the normal GPS four-state least squares algorithm is run with the coarse time estimate and associated satellite positions to further refine it (usually on the order of nanoseconds improvement).

4.3 Accuracy as a function of Time Ambiguity

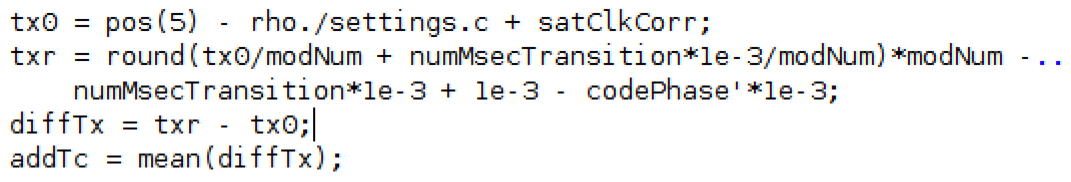
Unfortunately in many cases, the accuracy of the five-state least squares is not within 10 milliseconds. Therefore, a different time ambiguity is required to be a benchmark for the satellite transmission times. All that needs to change for the corrected time of transmission equation is to replace the 20 millisecond modulus time with a different time ambiguity as shown below:

$$t_{tx}^{(k)} = round(\hat{t}_{tx}^{(k)}/t_{mod} + N_m^{(k)}/(t_{mod}/10^{-3})) * t_{mod} + (1 - z^{(k)} - N_m^{(k)}) \times 10^{-3} \tag{4.8}$$

where t_{mod} is the time ambiguity in seconds, and $N_m^{(k)}$ is the number of milliseconds from that ambiguity. The experiments described later will investigate the time ambiguity where accurate timing can most likely be achieved.

4.4 Changes to the Software Receiver

To include time resolution using an arbitrary ambiguity, the only changes show up in the five-state least squares function of the A-GPS software receiver. Essentially, after the least squares algorithm converges, the method explained above is used to round the transmission times to the closest time ambiguity that is being used. The number of milliseconds from the next time ambiguity is passed into the function and is used with the sub-ms pseudorange to offset from the closest time ambiguity. This gives the correct time of transmission for each satellite. The difference between this time of transmission and the estimated time of transmission is found and averaged for each satellite and used to correct the coarse time as shown in Fig. 4.2.



```
tx0 = pos(5) - rho./settings.c + satClkCorr;
txr = round(tx0/modNum + numMsecTransition*1e-3/modNum)*modNum -...
    numMsecTransition*1e-3 + 1e-3 - codePhase'*1e-3;
diffTx = txr - tx0;
addTc = mean(diffTx);
```

Figure 4.2: MATLAB code for time correction

Finally, for improved accuracy the satellite positions are found at the correct time of transmissions and the calculated full pseudoranges are subtracted by the common bias found in the five-state least squares. These satellite positions and pseudoranges are then used in the four-state least squares to get a more accurate position and common bias estimate.

4.5 Code Layout

The set-up for the eniter code structure consists of a regular GPS L1 SDR (a.k.a Full SDR) and an A-GPS L1 SDR (a.k.a. A-GPS SDR). A data file is first passed to the Full SDR and goes through acquisition, tracking, and position estimation as explained above. Importantly, this implies that the file is at least 37 seconds long. After the file is analyzed, the relevant navigation solutions are saved. If time accuracy is not a worry, then only the decoded ephemeris parameters, the acquired satellites, the position, and the first sample time of the file are saved. For improved time accuracy, the number of milliseconds from the start of the file to the first subframe start for each satellite is also saved. This is explained in the section above.

The saved parameters are then passed to the AGPS SDR. The position and time are the a priori guesses for the A-GPS algorithm (subject to error offsets). If time is not a concern, then the PRN's given by the Full SDR are acquired and the five-state least squares A-GPS algorithm is implemented. For accurate time, the signals are tracked for only three seconds and then the time-corrected five-state least squares algorithm is used.

Chapter 5

Testing & Results

The experiments' main objective was to test A-GPS's ability to converge. First, without looking at the timing accuracy, the sensitivity of the A-GPS position error was examined. The initial position and time passed to the A-GPS code was varied until a good range of acceptable initial error values was found.

Then, the main experiment was performed where the accuracy of the timing was examined with several data sets.

5.1 Sensitivity to A Priori Position and Time Errors

One of the main questions that needs to be answered is: how inaccurate can your a-priori position and time be and still give accurate position results? To investigate the allowable errors, the data is simulated with several different errors. First, the effect of position and time errors are examined separately. Then, they are added in parallel to form a 2-D error space. In order to get a strong sense of the allowable combined errors, the experiment is run over 24 hours so that different satellite geometries and atmospheric conditions are included in the results.

The plot shown in Fig. 5.1 shows the position error norm with respect to the initial position bias.

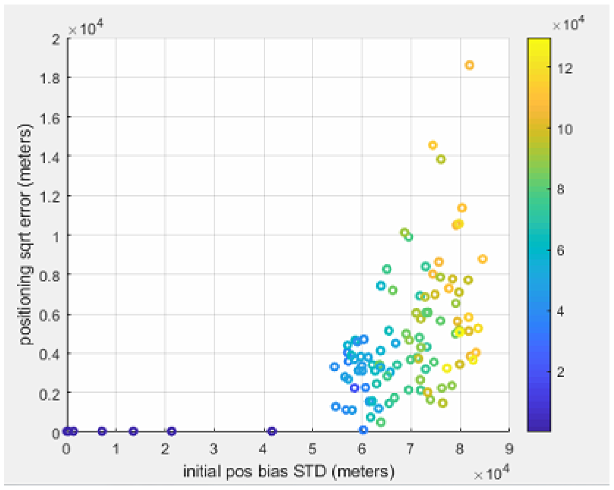


Figure 5.1: Position errors w.r.t initial position bias

After around 50 kilometers of offset, the position error starts to increase rapidly as the five-state least squares does not converge correctly.

A similar story plays out for the position errors with respect to the initial time bias as shown in Fig. 5.2.

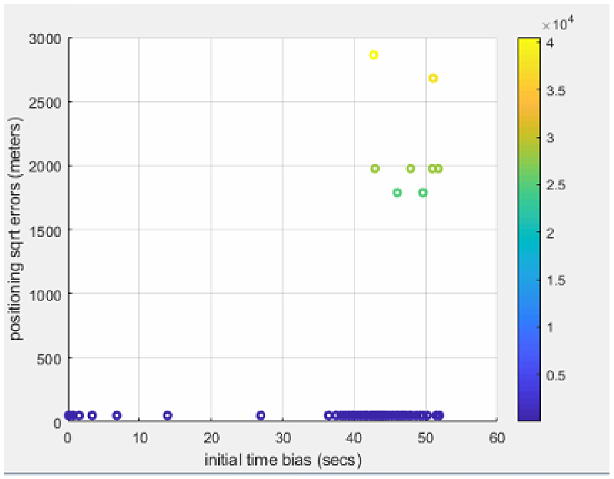


Figure 5.2: Position errors w.r.t initial time bias

After around 40 seconds, the initial time bias is too much for the least squares algorithm to reliably converge.

Varying both initial position and time, the position error over the 2-D space is shown in Fig. 5.3.

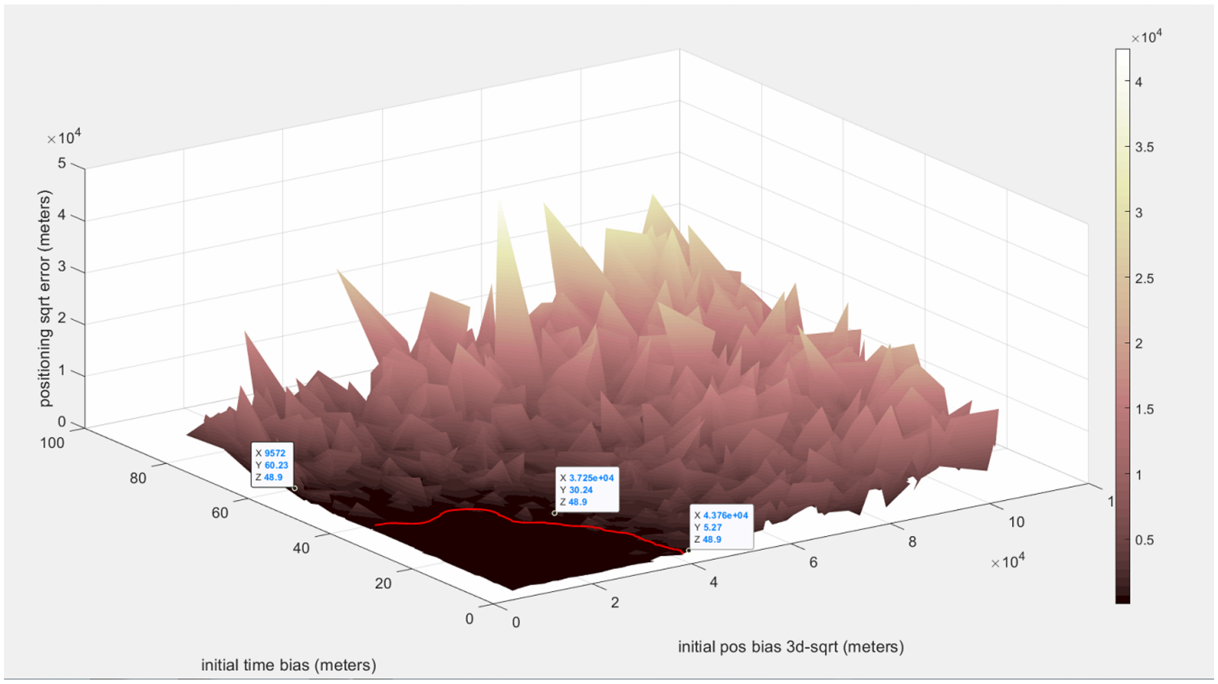


Figure 5.3: Position errors w.r.t initial position and time biases

The safe zone is roughly any position error less than 35 kilometers and any time error less than 30 seconds. van Diggelen in his A-GPS book has a safe zone of roughly 100 km position error or 1 minute time error [5].

5.2 Time Ambiguity

The next big question is: how much time ambiguity is required to ensure that the A-GPS algorithm can find time accurately? Basically it is the search for how much certainty of time there must be to then converge within nanoseconds. To understand, imagine you go on a walk with signs at specific intervals and a step-counter you reset at every sign. If the signs were a mile apart, then it would be easy to know how far you have walked since you know easily whether you are on your first or second mile, and also know how many steps you are after the mile marker. However, it is a different story if the signs were 200 feet apart. It would be very hard to distinguish how many signs you passed but you still know exactly how many steps you are from the last sign. There is a chance of guessing your distance walked correctly but it is low. With a sign every foot it would be almost impossible to guess the distance traveled. For A-GPS, the signs are the time ambiguities, and the step counter is represented by the number of milliseconds until the time ambiguity.

In the GPS L1 C/A signal, there are three time ambiguities built in. They are the millisecond code period, the 20 millisecond navigation bit, and the 6 second navigation message subframe. For this experiment, the milliseconds to the subframe start for each satellite is passed to the A-GPS code. With that, it can be modulated down to any time ambiguity that is a factor of 6000 milliseconds. This is so that at the six second subframe start, every other time ambiguity is also at that time. Then, an analysis of time accuracy as a function of time ambiguity can be tested.

To look at the time accuracy with respect to time ambiguity, data is first collected and run through the Full SDR. The necessary information for A-GPS is collected as well as the Full SDR's first sample time, corrected sampling frequency, and number of milliseconds to the first subframe for each satellite. The same data is then run through the A-GPS SDR with the milliseconds until the currently tested time ambiguity and the method described above is implemented. This is repeated for every time ambiguity that is being tested. Further, this is then repeated for 200 hundred data collections spaced every 15 minutes. This corresponds to a 50 hour time span.

An important note is that the first sample time and corrected sampling frequency calculated by the Full SDR only is calculated using the results from the first three seconds of data so that it is a better comparison to the A-GPS code.

In Fig. 5.4, the first sample time errors are shown with respect to ambiguity time.

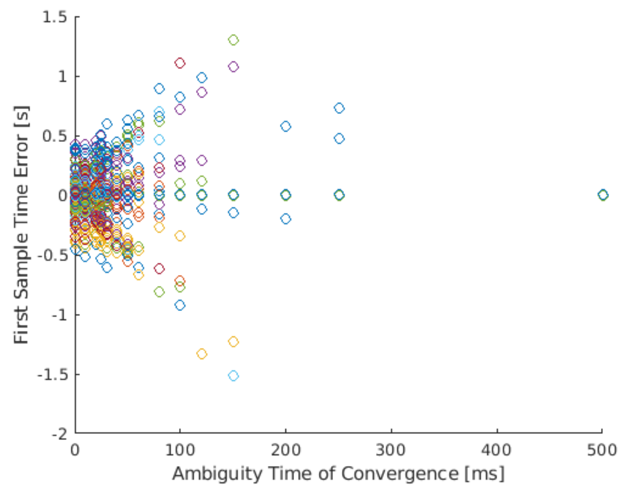


Figure 5.4: First sample time error w.r.t ambiguity time

Interestingly, the error increases in some cases as ambiguity time increases. This is likely due to the transmission time rounding to the incorrect time, which is a bigger mistake the larger the ambiguity.

In Fig. 5.5, the probability of converging given a specific ambiguity time is shown.

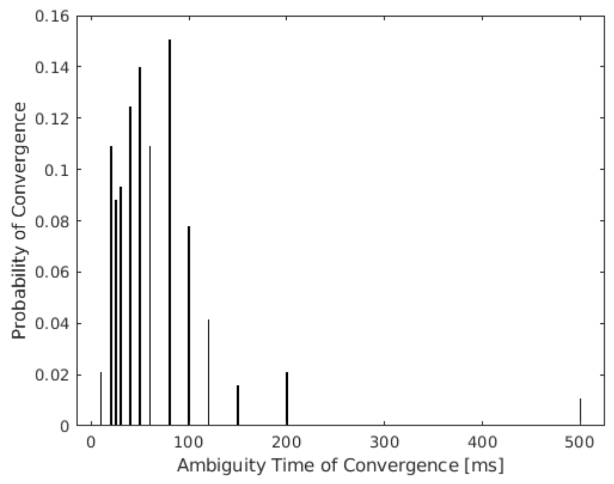


Figure 5.5: Probability distribution of ambiguity time of convergence

The probabilities look to follow a gamma distribution with most of the convergence happening before 200 milliseconds.

In Fig. 5.6, the more useful cumulative distribution is shown that gives a good visualization of convergence.

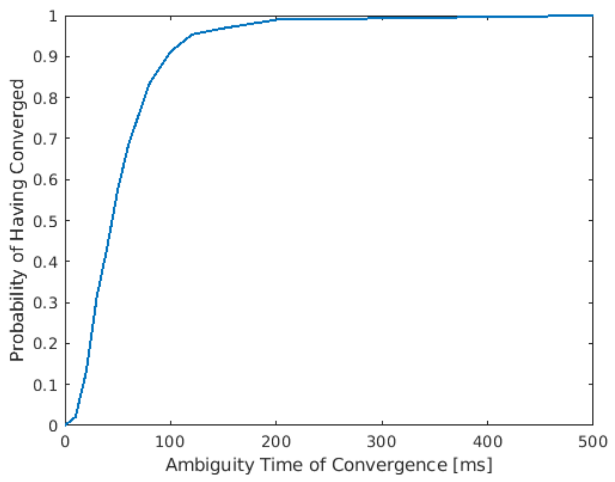


Figure 5.6: Cumulative distribution function of ambiguity time of convergence

At 100 ms, it is 91% likely to converge while at 200 ms it is 99% likely.

The results for key tested ambiguities can be seen in the table below:

Table 5.1: Probability of Having Converged given Ambiguity Time

Ambiguity Time (ms)	10	20	30	40	50	60	80	100	150	200
Convergence Prob. (%)	2.1	13.0	31.1	43.5	57.5	68.4	83.4	91.2	96.9	99.0

It is also important to point out that using the navigation bit ambiguity time of 20 ms only results in 13 % probability of convergence.

These results lead to the next question: what causes the least squares to converge at different ambiguity times? One possible metric is the dilution of precision of the fifth state, coarse time. Below in Fig. 5.7 is a plot of the coarse time dilution of precision (CTDOP) with respect to the ambiguity time of convergence.

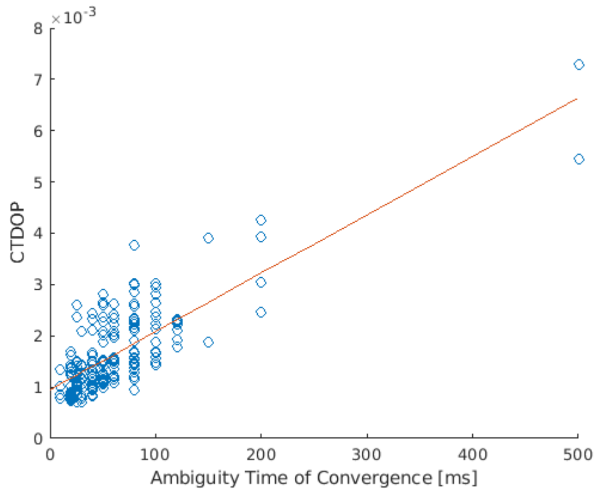


Figure 5.7: Coarse time DOP w.r.t ambiguity time of convergence

There is a clear linear relationship although the data is wide spread. At each extreme, the CTDOP does explain the ambiuity time of convergence. With only 10 ms required, the CTDOP is very small and when 500 ms is required, CTDOP is very large.

The CTDOP is more clearly visualized as a 2-D histogram in Fig. 5.8 below:

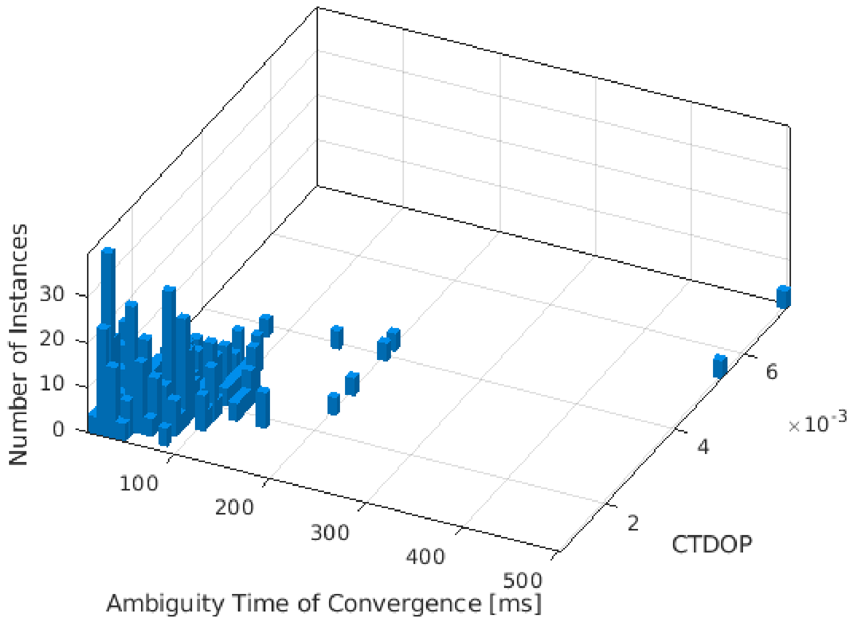


Figure 5.8: 2-D histogram of coarse time DOP and ambiguity time of convergence

Most of the distribution is within 200 ms and a CTDOP of 0.003.

Another possible metric for predicting ambiguity time of convergence is number of satellites. However, it might be important to see the relationship between number of satellites and CTDOP first. Looking at Fig. 5.9, the logarithm of the CTDOP is plotted with respect to the number of satellites.

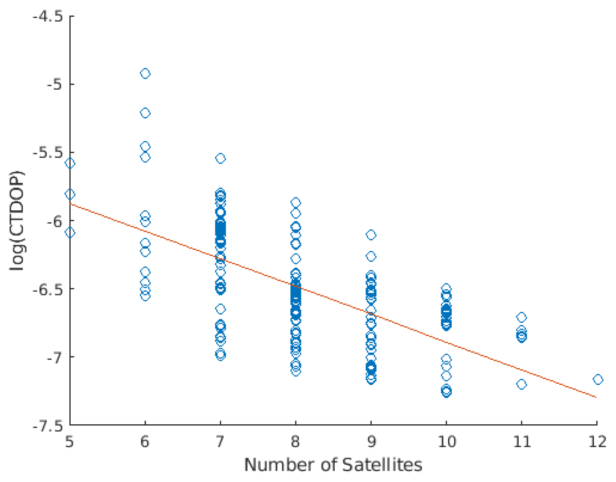


Figure 5.9: The logarithm of CTDOP w.r.t number of satellites

This shows that there is an exponential decay in CTDOP as the number of satellites increases. This is an interesting but reasonable result, since the geometry of the satellites directly impacts the coarse time result.

As an important note, it is necessary to see the distribution of number of satellites that the experiment's data contains. In Fig. 5.10, the number of satellites for the experiment is shown.

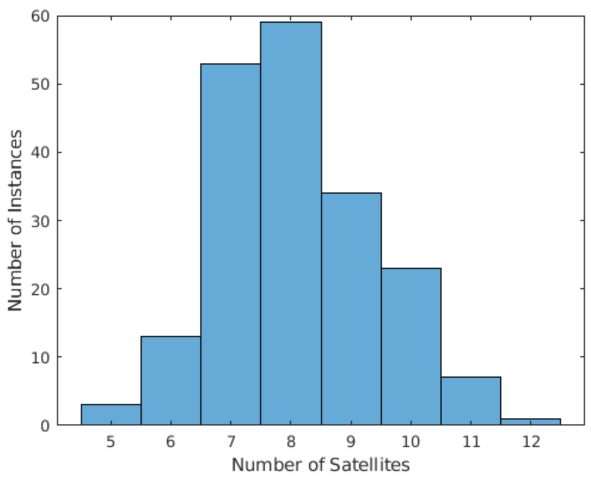


Figure 5.10: Histogram of number of satellites

This is significant in the fact that most of the cases had a good amount of satellites and thus the data does not represent the cases with five or six satellites well.

Looking at the first sample time, Fig. 5.11 shows the distribution of first sample time errors as compared to the Full SDR with a normal curve overlaid.

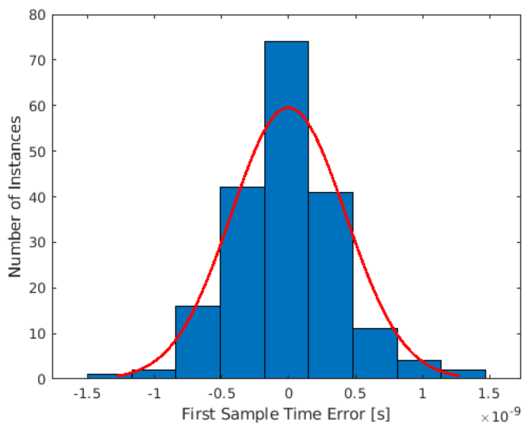


Figure 5.11: Probability distribution of first sample time error

As can be seen, most errors fall under a magnitude of one nanosecond error with none exceeding two nanoseconds. An important note is that these first sample time errors are the errors between the A-GPS SDR and the Full SDR. These are the errors once converged, and it is important to note that they do not change as ambiguity time is increased further.

In Fig. 5.12, a 2-D histogram is shown of the number of satellites and the logarithm of the first sample time error.

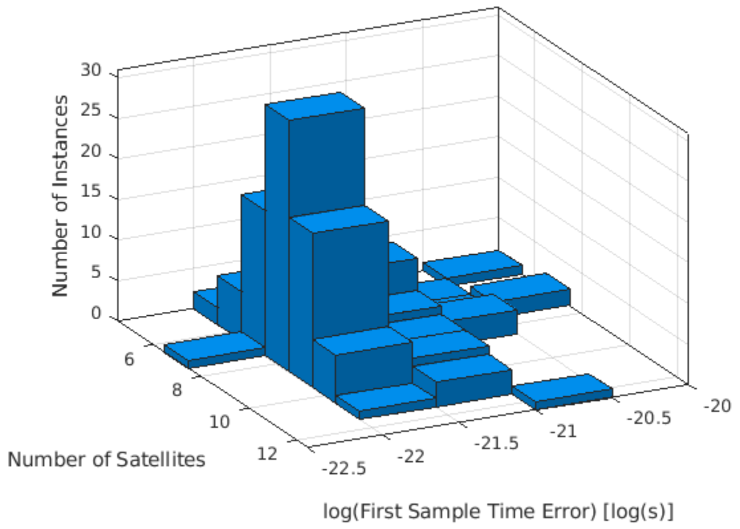


Figure 5.12: 2-D histogram of log of first sample time error and number of satellites

Not too much can be seen from this plot, although there is a possible correlation that shows having less satellites could result in a larger first sample time error.

Chapter 6

Discussion

Overall, this method works well but is linked to the performance of the least squares algorithm. The sections below dive into how to improve the performance, how other GNSS signals can provide the necessary time ambiguity, and how combining GNSS signals would help.

6.1 Improving the Accuracy of the Five-State Least-squares

If possible, the ideal performance of the code would be nanosecond-level operation with just the navigation bit data for the time ambiguity. This is not the case. However, the data shown above was collected and processed with no mitigation of atmospheric or multipath errors. Reducing these errors would improve the convexity of the five-state least squares and shift the probability of converging to be focused at an earlier time of convergence.

6.2 Other GNSS Signals

In real A-GNSS operation, arbitrary ambiguities are unfortunately not available to the receiver. All that the receiver has are the signals that are found in the different GNSS constellations. So the goal therefore is to find a time ambiguity that repeats just often enough to provide a correction that would make convergence to nanosecond-level timing guaranteed.

For example, GPS L2C has its CL code that repeats every 1.5 seconds, which would be ample enough time ambiguity. In this case, it is even more useful because the transition point can be determined with much less than 1.5 seconds of data. Another example is GLONASS with its frame repeating every two seconds. Once synced, two seconds is also more than enough time to ensure nanosecond-level accuracy in all cases.

6.3 Combining GNSS Signals

As seen, the number of satellites has an exponential effect on the CTDOP, and the CTDOP has a linear relationship with time of convergence. By combining several GNSS signals, the number of available satellites for the five-state least squares increases drastically, and is much more likely to need less ambiguity time to converge. For combining GPS L1 and GLONASS L1, one extra state would be required in the least-squares algorithm to keep track of the time difference between the two signals. However, this would be easy to handle with the extra amount of available satellites.

Chapter 7

Conclusion

In this thesis, a brief overview of the GPS L1 signal structure and GPS position and time estimation was given. Then, van Diggelen's A-GPS method was discussed with his millisecond ambiguity approach. Next, a way to accurately use any ambiguity time to reach nanosecond-level accuracy was developed. Experiments were then run to test the sensitivity of A-GPS to a priori errors in position and time, as well as the performance of the first sample time error as a function of ambiguity time. It was found that the safe zone for A-GPS is less than 35 km initial position error and less than 30 seconds initial time error. It was also found that an ambiguity time of 200 ms converged to nanosecond-level accuracy 99 % of the time. Interestingly, the coarse time dilution of precision had a linear relationship with the time of convergence, and decayed exponentially with an increase in the number of satellites. Most first sample time errors were within one nanosecond while none exceed two nanoseconds. It was then discussed that improving the accuracy of the five-state least squares would result in earlier convergence times. Also, it was discussed that other GNSS signals could provide a small enough ambiguity time to use this method and have it converge in all cases. Finally, it was discussed that other GNSS signals can be combined with GPS to provide more satellites to decrease the necessary convergence time.

Bibliography

[1] Nainesh Agarwal, Julien Basch, Paul Beckmann, Piyush Bharti, Scott Bloebaum, Stefano Casadei, Andrew Chou, Per Enge, Wungkum Fong, Neesha Hathi, and et al. Algorithms for gps operation indoors and downtown. GPS Solutions, Nov 2002.

[2] D. Akopian and J. Syrjarinne. A network aided iterated ls method for gps positioning and time recovery without navigation message decoding. 2002 IEEE Position Location and Navigation Symposium (IEEE Cat. No.02CH37284), 2002.

[3] D. Akopian and J. Syrjarinne. A fast positioning method without navigation data decoding for assisted gps receivers. IEEE Transactions on Vehicular Technology, 58(8):4640–4645, 2009.

[4] Kai Borre, Dennis M. Akos, Nicolaj Bertelsen, Søren Holdt. Jensen, and Peter Rinder. A Software-Defined GPS and Galileo Receiver: a Single-Frequency Approach. Birkhauser Boston, 2007.

[5] Frank van. Diggelen. A-GPS: Assisted GPS, GNSS, and SBAS. Artech House, 2009.

[6] Éamonn P Glennon and Roderick C Bryant. Solution of Timing Errors for AGPS.

[7] Kannan Muthuraman, Jim Brown, and Mangesh Chansarkar. Coarse Time Navigation: Equivalence of Algorithms and Reliability of Time Estimates.

[8] Niilo Sirola. A method for gps positioning without current navigation data. Master's thesis, 2001.

[9] Niilo Sirola and Jari Syrjarinne. GPS Position Can Be Computed without the Navigation Data.

[10] J. Syrjarinne. Time recovery through fusion of inaccurate network timing assistance with gps measurements. Proceedings of the Third International Conference on Information Fusion, 2000.