

SIGNAL PROCESSING FOR GPS/IRNSS RECEIVER

Dual Degree (B.Tech + M.Tech) Dissertation

Submitted in Partial Fulfillment of the Requirements
for the degree of

Master of Technology

by

Chirag C. Shetty

120070007

Supervised By

Prof. Madhav P. Desai
and
Prof. Sibi Raj B. Pillai



Department of Electrical Engineering
Indian Institute of Technology, Bombay
Powai, Mumbai - 400 076.

2016-2017

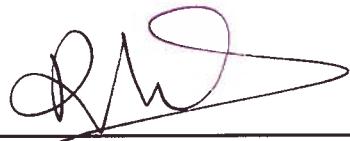
Dissertation Approval

This dissertation entitled "Signal Processing for GPS/IRNSS Receiver", submitted by Chirag C. Shetty (Roll No. 120070007), is approved for the award of the degree of Master of Technology in Electrical Engineering.

Examiners

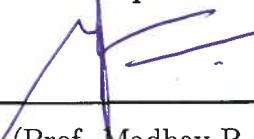


(Prof. Shalabh Gupta)



(Prof. Rajesh Zele)

Supervisor



(Prof. Madhav P. Desai)

Co-Supervisor



(Prof. Sibi Raj B. Pillai)

Chairman



(Prof. Rajesh Zele)

Date : 30/6/2017

Place : VLSI Conference Room, GG, EE dept., IITB

Declaration of the Academic Ethics

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I adequately cited and referenced the original sources. I declare that I have properly and accurately acknowledged all sources used in the production of this thesis.

I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/ data/ fact/ source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources which have not been properly cited or from whom proper permission has not been taken when needed.

Date: 3/7/2017



Chirag C. Shetty

(Roll No. 120070007)

Acknowledgment

My greatest thanks to my supervisors Prof. Madhav P. Desai and Prof. Sibi Raj B. Pillai, not only for the constant support and direction, which made me a better engineer, but also for teaching me many valuable life lessons. They encouraged me to be curious, to explore and learn, while ensuring focus and discipline. I thank Prof. Shalabh Gupta, for very patiently hearing out my doubts at every stage of the project. Discussions with him helped me greatly, to appreciate the practical engineering aspects, which are hard to find in books. I also thank Prof. Madhu Belur and Prof. Animesh Kumar for valuable inputs.

I thank my friends, Arunabh, Praveen and Vineet, for the enriching and fun time we had, working on the front-end. I would also like to acknowledge Wadhwani Electronics Laboratory(WEL), Electrical Department IIT Bombay and SAMEER Mumbai for providing me with necessary resources to carry out the project.

- Chirag C. Shetty

Abstract

Indian Regional Navigation Satellite System (IRNSS), also named as NAVIC, is India's indigenously developed satellite navigation system, completed in April 2016 with a coverage over the Indian subcontinent. Key to success of a navigation system is the capability to build accurate, cheap and low-power receivers which can be easily integrated with multiple existing platforms and devices. While IRNSS has many similarities with GPS, there are some crucial differences. In this work, we design, implement and analyze the most essential signal processing blocks in building an IRNSS receiver, namely acquisition, tracking and pseudo-range computation. The techniques are inspired by vast literature on GPS receiver design. Key engineering parameters involved in the design are identified and their inter-dependencies are studied, as a first step towards subsequent development of an ASIC receiver. An example software implementation is provided. Towards the end, an analog front-end for S-band of IRNSS using off-the-shelf is built and used to test the software receiver.

Contents

1 Satellite Navigation Systems & IRNSS	1
1.1 Introduction	1
1.2 Working Principle	2
1.2.1 Trilateration	2
1.2.2 Reference in Space	3
1.2.3 Time Reference	4
1.2.4 pseudorange Measurement	5
1.3 Indian Regional Navigation Satellite System (IRNSS)	5
1.3.1 Similarities with GPS	6
2 Signal Structure & Processing	7
2.1 Signal Structure	7
2.1.1 PRN Codes	10
2.1.2 Navigation Bits Framing & Time information	11
2.2 Signal Reception	14
2.3 Signal Processing	15
2.3.1 Calculating the pseudorange	16
2.3.2 Acquisition	20
2.3.3 Tracking	24
2.3.4 Navigation Bit Processing	30
3 Design & Analysis of the Receiver	33
3.1 Acquisition Block Design	33
3.1.1 Integration Time, T_{int} & Doppler Search Bin-Width, Δf_d	33
3.1.2 Code-Phase Search Bin-Width, $\Delta\tau_c$	34
3.1.3 Threshold V_{th} and detection decision	35
3.2 Tracking Block Design	36
3.2.1 DLL	36
3.2.2 PLL	37
3.3 Computing pseudorange	42

4 Software Implementation of GPS Receiver	45
4.1 Software Receiver pseudocode	45
4.1.1 Baseband Processing	47
4.1.2 pseudocode	49
4.2 Experimental Setup	54
4.3 Results and Validation	55
5 IRNSS Front-end: Hardware Implementation	65
5.1 GNSS receiver front-end architecture	65
5.1.1 Sampling Parameters	69
5.2 IRNSS S-band Front-End implementation	72

List of Figures

1.1	Triangulation Illustration	4
2.1	Effect of spreading	8
2.2	GPS/IRNSS Signal Structure	8
2.3	PRN Codes and Navigation bits Alignment	9
2.4	PRN code Correlation properties	11
2.5	Gold Code Generation [11]	12
2.6	IRNSS subframe structure [11]	12
2.7	GPS subframe structure	13
2.8	Raw signal properties	14
2.9	Measuring time relative to preamble beginning	17
2.10	Measuring relative delays using preamble beginnings	19
2.11	Acquisition Illustration	20
2.12	Result of Acquisition procedure	23
2.13	Effect of Residual Doppler after acquisition	24
2.14	Basic Structure of a Phase Lock Loop	26
2.15	Changing code-phase due to Doppler	28
2.16	Working of Delay Lock Loop	29
2.17	Parity encoding in GPS signal	32
3.1	Choice of T_{int} & Δf_d	35
3.2	Choice of $\Delta\tau_c$	35
3.3	PLL Loop Updates Illustration	38
3.4	PLL outputs	39
3.5	PLL oscillations	40
3.6	Noise pulls PLL out of oscillations	41
3.7	Computing relative time of arrivals	42
3.8	Using Code Phase in delay measurement	43
3.9	Result of aligning PLL T_{int} segments with incoming PRN sequence	43
4.2	Acquisition output on real data	57
4.3	PLL frequency tracking output on real data	58
4.4	DLL Code Phase tracking on real data	58

4.5	PLL frequency tracking output on real data	59
4.6	Delays between arrival of Preamble from different satellites	59
4.7	Accuracy of pseudo-range computation	60
4.8	Stability of receiver clock	61
4.9	NAV bit frames from real data	61
4.10	Measure of PLL lock	62
4.11	PLL lock measure during USRP frequency shifts	62
4.12	PLL locks at $(f_d + 500 \text{ Hz})$	63
4.13	PLL lock measure in case of false lock	63
4.14	NAV bit flips in real data	64
5.1	Complete Receiver Block Diagram	67
5.4	Oversampling incase of one-bit quantization	71
5.7	WiFi Interference seen using Spectrum Analyzer	74

Chapter 1

Satellite Navigation Systems & IRNSS

1.1 Introduction

Satellite based navigation system is one of the defining technologies of our times. Positioning and timing services of Global Navigation Satellite Systems (GNSS) are used in a wide range of applications, from critical systems like safe airplane landings, power grid time synchronization etc [9], to civilian uses like Google Maps and GNSS reflectometry for soil moisture measurements [8]. It is worth noting that GNSS receivers can act as very accurate time keeping devices as well. As of 2017, Global Positioning System (GPS) of the United States and GLONASS of Russia are fully operational satellite navigation systems with global coverage (which means, one can use these to find location anywhere on Earth). European Union's Gallileo is in the process of being made a global navigation system. Besides these, there is BeiDou of China, and IRNSS (India Regional Navigation Satellite System, named NAVIC) from India, which are regional navigation systems with coverage over and around the respective countries.

Global Positioning System (GPS) developed in the US was the first global satellite navigation system to become operational (in 1995). It was preceded by ground-based navigation systems. The birth of idea that lead to GPS dates back to 1957 when Sputnik 1, the first man-made satellite was launched by the Soviet Union. The observation that Doppler shift in the signal received from Sputnik can be used to locate the satellite along its orbit, triggered interest in the reverse problem of using satellites with precisely defined orbits to locate places on Earth or space. This led to development of the first satellite navigation system TRANSIT in 1960. It consisted of a five satellite constellation giving a position fix once every 110 minutes near the equator and had a typical accuracy of 200 meters. This was followed by the Timation satellites, which tested very accurate clocks in orbit and effects of relativity, which are an important part of any GNSS. The current GPS system became fully operational in April 1995. Since then many advancements have been made and several new types of signal transmissions were added to develop new functionalities and to improve the accuracy of old ones. GLONASS of Russia was developed in parallel with GPS.

All satellite navigation systems are made up of three segments namely:

- The Space Segment: It consists of the satellite constellation. The number of satellites in the constellation decides the coverage. In order to enable global coverage atleast 4 satellites should be visible from any point on Earth, for reasons described later. GPS and GLONASS have 32 satellite and 28 satellite constellations respectively. The satellites have synchronized clocks aboard them to have a common reference time and their orbits should be precisely known. Satellites transmit information about their orbit, time and health at all times. An unlimited number of users can receive and process these signals.
- The Control Segment: There are multiple ground-stations that constantly monitor the satellites and take actions to ensure proper functioning of the system. The control segment is responsible for updating satellite orbital parameters, onboard time keeping etc at regular intervals. The locations of ground-stations and other details for GPS can be found in [26].
- The User Segment: Users are passive receivers i.e no signal needs to be transmitted to the satellites/ground-stations to use the navigation systems. Any receiver capable of capturing the transmitted signal and processing, can compute its location coordinates. The main factor which ensured the widespread use of satellite navigation was the availability of portable and affordable, low power receivers, which can be integrated with existing devices like mobile phones. The designs typically do away with heavy computations at the receiver. Usually two kinds of services are offered, the Standard Positioning Service(SPS) for civilian use, and Precise Positioning Service(PPS), with better accuracy and encryption for military uses. The two services use orthogonal signals transmitted by the satellites. SPS receiver design is the main focus of this text.

The following section briefly describes the working principles underlying all GNSS's. It is followed by an introduction to IRNSS, highlighting its similarity to GPS.

1.2 Working Principle

To assign unique coordinates to locations in a given space we need reference points whose locations are precisely defined, and a method to uniquely identify all other points w.r.t these references. Distances can be measured from the given point to the references, and then unique coordinates can be assigned. In satellite navigation systems, loosely speaking, the satellites are treated as the references in space.

1.2.1 Trilateration

In 2D, if two reference points are fixed and distances of a point from these references are given, the point can be identified as one of the two on either side of the line joining the references. This can be done by drawing circles with references as centers and the given distances as radii. The circles will intersect at two points unique to the given distance pair. Extrapolating the

same idea to 3D, we will have spheres instead of circles. 3 spheres intersect at two points. 1.1 illustrates this concept, known as ‘trilateration’. By measuring distances at any instant, between the location of interest and three reference satellites, we can assign coordinates to the location (after eliminating the other point in space, also resulting from intersection of the three spheres). So, the primary goal is to accurately and simultaneously measure distances between satellites and the receiver on Earth. The receiver accomplishes this by estimating the time that received signal took to travel from the satellite to the receiver. Measured time is multiplied by the speed of radio waves to get the distance.

Suppose we make the time measurements. Let t_i be the transit time from satellite i to the receiver. Then the distances ρ_i for $i = 1, 2, 3$ are given by $\rho_i = ct_i$. These are known as the ‘pseudorange’ measurements. Let (x_i, y_i, z_i) be the known coordinates of the satellite i at the time of transmission and (x_r, y_r, z_r) be the receiver coordinates at the time of reception, which need to be computed. We have three unknown coordinates and three equations, $\rho_i = \sqrt{(x_i - x_r)^2 + (y_i - y_r)^2 + (z_i - z_r)^2}$ for $i = 1, 2, 3$. In principle, these equations can be simultaneously solved for (x_r, y_r, z_r) . However there are considerable challenges in making these measurements. A small error of $1\mu\text{s}$ in measurement of t_i will cause the range to be off by 300m. This and other considerations are described in the next subsection. Conventions used in the GPS system are used as examples here, which may differ in technical details from other navigation systems.

1.2.2 Reference in Space

Firstly, it is important to have a universal coordinate system that can be used to assign coordinates to locations on earth, using the orbiting satellites as references. GPS uses Earth-Centered Inertial Coordinate System (ECI) to specify satellite orbits. ECI system considers center-of-mass of the Earth as the origin and axes fixed with respect to the stars. However receiver positions have to be specified in Earth-centered Earth-Fixed system (ECEF) which can then be converted to the familiar coordinates of latitude, longitude and altitude. The receiver also needs to account for the rotation of Earth while the signal is traveling from the satellite to receiver. The details of establishing the two coordinate systems and conversions between them is non-trivial and not dealt with in this report. Secondly, knowing the orbits of satellites precisely is crucial to navigation system accuracy, since satellites are the references. Satellites transmit their locations, to be used by the receiver in fixing its own position. Hence the satellites need to have information about their location at all times. Owing to multiple non-idealities, the orbital parameters change and have to be corrected for regularly. This is done by the Control Segment ground-stations. These corrections still leave a possibility for small errors known as ‘ephemeris errors’. In a typical receiver ephemeris errors are usually large in radial direction i.e in altitude measurement as compared to ground coordinates. (This is because satellite references will be present only on one side of the receiver along radial direction, since other side is the earth.)

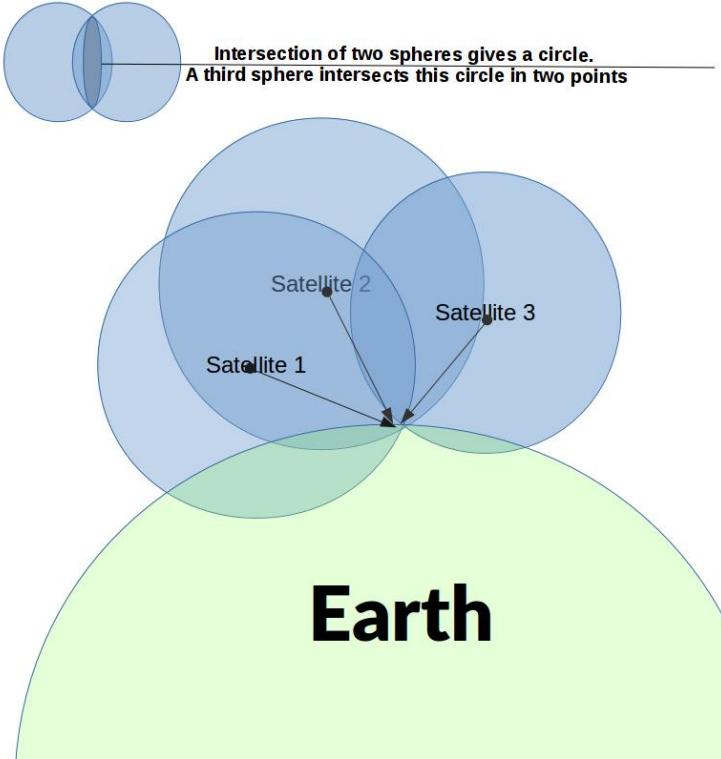


Figure 1.1: Intersection of spheres centered at 3 satellites identifies 2 points (One is shown on surface of the Earth, other would be higher above in space)

1.2.3 Time Reference

As mentioned before, computing time of arrival of the signal is key to computing position. This would have been easy if all the satellites and the receivers had a common and accurate time reference. In this case all satellites need to transmit their signature signals at a common agreed times and the receiver can measure delay of each signal on reception and estimate the ranges ρ_i 's. Three such ranges suffice to solve for receiver coordinates. To enable this, satellites carry synchronized ‘atomic clocks’. This is referred to as GPS system time. However they are subject to relativistic effects, hence 10.23MHz reference clocks (w.r.t Earth) on GPS satellites are actually set at 10.229999954326MHz [10] (about 38 seconds faster). These corrections are not fixed either; they need to be monitored by the ground-station and corrected to some extent.

But receivers cannot afford to have a synchronized accurate clock. They should work with less accurate clocks, which have time offsets as well as frequency drifts affected by environmental conditions, normally these are unknown quantities. Thus clock offset, Δt , of the receiver, w.r.t GPS time at the time of measurement becomes the fourth unknown in the system. The range equations get modified to $\rho_i = \sqrt{(x_i - x_r)^2 + (y_i - y_r)^2 + (z_i - z_r)^2} + c(\Delta t)$. Solving this necessitates 4 independent equations, and hence visibility of 4 satellites, rather than 3, to get a position fix. Value of Δt can be known as a byproduct of solving for (x_r, y_r, z_r) and the receiver time can be offset by Δt to get near atomic-clock like accuracy. In practice, the differences between pseudorange of satellites $\rho_i - \rho_j$ is measured by the receiver. One of

the satellites is taken to be at some arbitrary distance (say 20,000km) and other distances are taken relative to it by adding the differences. Once the equations are solved, the true distances can be obtained. The satellite signal has to travel through layers of atmosphere to reach the receiver. Atmospheric effects add additional time delay to this path causing errors in range measurements. Ionospheric and Tropospheric delays are the main cause of errors due to their unpredictable nature. Models have been developed to predict and compensate for these errors. The first-hand distance estimates from time of travel is actual distance plus errors due to atmosphere and receiver clock offset. Thus it is referred to as **pseudorange** in GNSS literature.

Most of these models, corrections and solvers are well-documented and one can directly use toolboxes like [19] to work out the coordinates once data bits from satellites decoded and the pseudoranges have been calculated.

1.2.4 pseudorange Measurement

Here we discuss what entails pseudorange calculation as elaborated in [1]. Let us take the GPS Time (GPST) maintained at the control segment ground station as the reference. Consider the signal received at the antenna at time t GPST. Let the travel time from the satellite be τ . Thus the signal left the satellite at time $(t - \tau)$ GPST. The clocks at the receiver and the ones on satellite may have some offset compared to the GPST. Let $t_u(t)$ and $t_s(t)$ be the time measured by the receiver and the satellite respectively when the actual time according to GPST is t . Thus time of transmission as recorded by the satellite is $t_s(t - \tau)$, and let us assume that there is a way by which the receiver can recover this information. Then the distance measured at t would be:

$$\rho(t) = c[t_u(t) - t_s(t - \tau)] + \epsilon(t)$$

Here $\epsilon(t)$ represents error due to other effects like the ionospheric delays.

$$\begin{aligned} \text{Let } t_u(t) &= t + \delta t_u(t) \quad \text{and} \quad t_s(t) = t + \delta t_s(t) \\ \therefore \rho &= c[t + \delta t_u(t) - (t - \tau) - \delta t_s(t - \tau)] + \epsilon(t) \\ \therefore \rho &= c\tau + c[\delta t_u(t) - \delta t_s(t - \tau)] + \epsilon(t) \\ \text{Let } r &= c\tau \quad \text{i.e the actual range} \\ \therefore \rho &= r + c[\delta t_u(t) - \delta t_s(t - \tau)] + \epsilon(t) \end{aligned} \tag{1.1}$$

ρ is called the pseudorange (pseudo- because it is the actual range plus the errors). This is measured using the raw signals, and models are used to estimate for the errors and obtain estimate of r from ρ .

1.3 Indian Regional Navigation Satellite System (IRNSS)

Indian Regional Navigation Satellite System (IRNSS), named as NAVIC is a satellite navigation system developed and controlled by the Indian Space Research Organization (ISRO). The last

GPS and IRNSS: Signal Structure		
Navigation System	GPS	IRNSS
Transmission Frequency	L1: 1575.42MHz (Civil, Restricted) L2: 1227.60MHz (Restricted)	L5: 1176.45MHz (SPS & RS) S : 2492.028MHz (SPS & RS)
Min. Received Power (Civilian)	-158.5dBW (L1)	-159.0dBW (L5) -162.3dBW (S)
Modulation, Coding, Polarization	BPSK, DSSS-CDMA, RHCP	BPSK, DSSS-CDMA, RHCP
Navigation Data rate	50bps	50bps
Error Correction	(32,26) Hamming Code with 6 parity bits per 30-bit word	$\frac{1}{2}$ rate convolution code, interleaving & parity coding with CRC-24Q
PRN code (Civilian Use)	1023-length Gold Codes, 1ms long	1023-length Gold Codes, 1ms long
Data Frame Structure	1 Frame = 5 subframes, each subframe 300bits long	1 Frame = 4 subframes, each subframe 600bits long
Subframe differences	Time of Week: 19bits (6 second counts) 8 bit sync code	Time of Week: 17bits (12 second counts) 16 bit sync code

Table 1.1: Comparision between GPS and IRNSS signals

satellite of the constellation was launched on 28th April, 2016. It consists of a seven satellites, with expected coverage from latitude 30° South to 50° North, longitude 30° East to 130° East [11]. Like GPS, IRNSS offers Standard Positioning Service (SPS) for civilian use and Precise Positioning Service or Restricted Service (RS) to authorized users. The system is expected to offer an accuracy of 10 meters in the India [12]. IRNSS has many features in common with GPS, except that IRNSS is a regional navigation system and all of the 7 satellites in the constellation are visible at all times over the Indian sub-continent. This would simplify some aspects of receiver design for IRNSS.

1.3.1 Similarities with GPS

Table 1.1 lists some key comparisons between GPS and IRNSS signal structures:

More details can be found in Interface specification documents of GPS [10] and IRNSS [11]. As is clear from table 1.1, GPS and IRNSS differ only in higher layer frame structure and in frequency band of transmission. Thus, front end signal processing to decode data from raw signals would be same for GPS and IRNSS. Hence the discussions in subsequent chapters use GPS as the standard, but will apply equally well for IRNSS, unless otherwise specified.



Chapter 2

Signal Structure & Processing

Global Positioning System (GPS) was the first satellite based navigation system to become operational globally and still remains the dominant one. This chapter looks at the salient features of GPS signals and methods involved in decoding all necessary information from them to compute pseudoranges. The algorithms almost exactly extend to IRNSS.

The GPS signals are transmitted at 1.57542 GHz (L1 band) and 1.2276 GHz (L2 band). The L1 signal comprises of data transmissions for civilian use of GPS known as the coarse/acquisition (C/A) code as well as the more accurate precision P(Y) code transmissions for authorized use. L2 signal is used only for P(Y) code transmission in older GPS satellites, but a new civilian signal called L2C was introduced in the L2 band with newer satellites. IRNSS employs two frequencies 1176.45 MHz (L5 band) and 2492.028MHz (S band), for both C/A as well as Precision codes. Use of two separate bands is expected to provide a better estimate of ‘ionospheric group delay’, and thus a more accurate positioning than L1-based GPS. We shall restrict to the L1 C/A signal of GPS in this chapter to illustrate the basic principles. Measurement of time of travel of the signal from the satellite to the receiver is the crucial step in reception.

At the receiver, L1 GPS signals have nominal power of -128.5 dBm, in 20.46 MHz band around the center frequency. Comparing this to typical signal strength of cellphone signals of about -80 dBm gives an idea of how low the SNR of GPS signal are. Section 1 describes the signal structure used in GPS/IRNSS transmission. The following section briefly describes the signal at reception, more details of which are described in Chapter 4. Section 3 lists the information to be extracted from these signals and introduces the basic algorithms to do so. Section 4 presents some analysis on the same.

2.1 Signal Structure

GPS/IRNSS employ Direct Sequence Spread Spectrum (DSSS) technique in signal transmission. In DSSS, the data bits intended for transmission are multiplied by a known pseudo-random bit sequence of much higher frequency than data bits, and the resulting signal is modulated and transmitted. This process deliberately ‘spreads’ the spectrum of input signal to several times the original bandwidth as shown in Figure 2.1. At reception, the signal is de-spread by match-

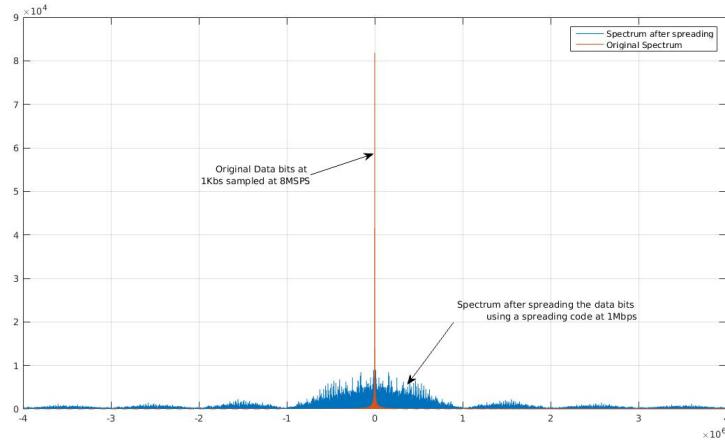


Figure 2.1: Effect of Spreading

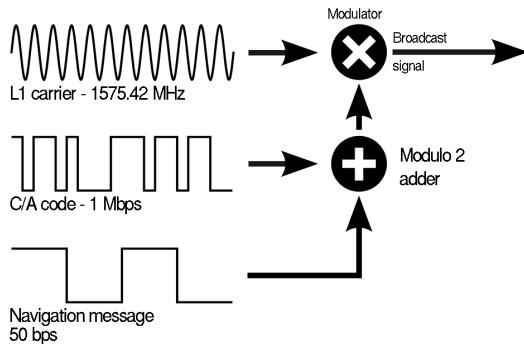
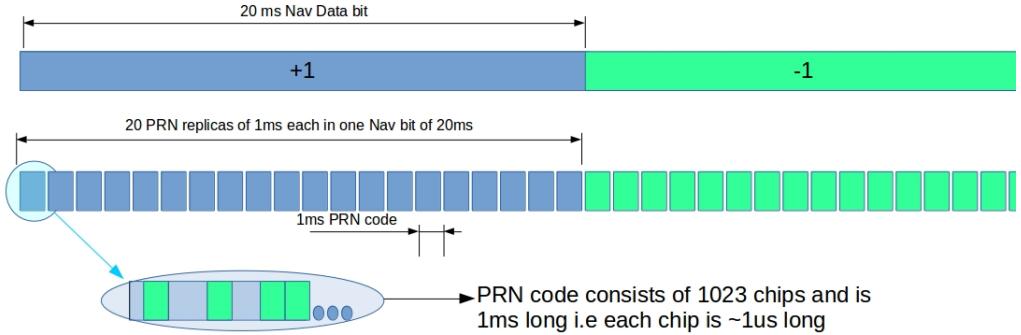


Figure 2.2: GPS/IRNSS Signal Structure (Source:Wikipedia)

filtering the received signal with the same pseudo-random sequence. To enable decoding, the spreading sequence must have the property of very low correlation with shifted versions of itself. DSSS is a form of Code Division Multiple Access (CDMA) technique, allowing multiple satellites to transmit data in exactly the same frequency band but with different, mutually nearly-orthogonal (i.e low cross-correlation) spreading sequences. Restrictions on power flux spectral density to prevent interference to microwave line-of-sight communication and radio astronomy measurements, makes DSSS a preferred way of signal transmission for GPS since it allows high power satellite transmission, but still remain within the spectral density limit due to large bandwidth [2]. Besides, DSSS provides robustness to GPS/IRNSS signals against narrow-band interference, multi-path effects and jamming attempts [13]. But more importantly, the pseudo-random sequence is what allows very accurate estimation of time of travel of the signal and hence is a key concept behind GPS/IRNSS.

The signal consists of three main components which are multiplied together to get the transmitted signal. Refer to 2.2 for a pictorial representation of the signal.

- **Navigation data bits**, $D(t)$ carry information on ephemeris (satellite position, velocity), time, satellite health data etc. at a very low rate of 50 bits per second.



The PRN codes are multiplied (modulo 2 addition in terms of bits) to obtain the final signal to be modulated onto the carrier using Binary Phase Shift Keying (BPSK)

Figure 2.3: Alignment of PRN codes with the navigation data bits

- **Ranging/Spread Spectrum codes**, $x(t)$ are 1023 bits long pseudo-random (PRN) sequences lasting 1 millisecond, uniquely assigned one each for all of the satellite in the constellation. PRN codes used in GPS/IRNSS belong to a family called Gold Codes. For a given satellite, same code sequence is appended one after the other every 1ms.
- **Carrier wave** is RF wave onto which the bit information mentioned above are modulated and transmitted. Frequency of GPS civilian signal is $f_{L1} = 1575.42\text{MHz}$ (wavelength $\approx 19\text{ cm}$). The signal is Right-Hand Circular Polarized (RHCP).

Each navigation data bit of 20ms is multiplied by 20 replicas of the PRN code lasting 1ms each. The PRN codes are aligned with data bit boundaries. Figure 2.3 illustrates the alignment. Since the chips are $T_c = \frac{1}{1023}\text{ ms}$ long, the resulting baseband signal has a first null frequency of 1.023MHz as shown in Figure 2.1. This string of chips is then BPSK modulated onto the carrier and transmitted. In GPS/IRNSS signals spectrum efficiency is not a worry. Hence the chips are sent as rectangular pulses modulated on the carrier without any pulse shaping.

Thus the signal transmitted at f_{L1} from the satellite is of the form:

$$s_{f_{L1}}(t) = \sqrt{2P_1}D_1(t)x(t)\cos(2\pi f_{L1}t) + \sqrt{2P_2}D_2(t)y(t)\sin(2\pi f_{L1}t) \quad (2.1)$$

where P_1, P_2 = transmitted signal powers ,

D_1, D_2 = Data bits

$x(t), y(t)$ = Ranging codes for C/A and P(Y) signals respectively

Note that the C/A and P(Y) signals at f_{L1} are in phase quadrature. Since we are only concerned with the C/A signal, we will drop the P(Y) signal in further analysis. Also, since we deal with only f_{L1} center frequency, we will drop the subscripts.

2.1.1 PRN Codes

pseudo-Random Noise (PRN) code is a bit sequence which is generated using a deterministic rule, but has spectral properties of a statistically random sequence (i.e a noise sequence where every bit is independent of all other bits). In particular, a PRN code has a delta-like peak in auto-correlation and very-low cross-correlation with any other PRN code. For L length sequences $\{X_n\}_{n=0}^{L-1}$ and $\{Y_n\}$, the ‘circular’ autocorrelation function $R_{XX}[m]$ and circular cross-correlation function $R_{XY}[m]$ are:

$$R_{XX}[m] = \sum_{n=0}^{n=L-1} X_n X_{((n+m) \bmod L)} \quad \forall m \in \mathbb{Z}$$

$$R_{XY}[m] = \sum_{n=0}^{n=L-1} X_n Y_{((n+m) \bmod L)} \quad \forall m \in \mathbb{Z}$$

For $\{X_n\}$ and $\{Y_n\}$ to be a good PRN codes, we need, as shown in Figure 2.4,

$$R_{XX}[0] \gg R_{XX}[m] \quad \forall m \neq 0$$

$$R_{YY}[0] \gg R_{YY}[m] \quad \forall m \neq 0$$

$$R_{XY}[0] \ll L \quad \forall m$$

Depending on the deterministic rule used to generate the PRN codes, there are different family of codes. PRN codes are usually generated using specific feedback tap combinations in a Linear Feedback Shift Registers(LFSR). **Gold Codes** is a set of PRN codes widely used as the spreading sequences in CDMA communication. 1023 length Gold Codes are used in SPS (Standard Positioning Service) of GPS/IRNSS. Each satellite is allotted a fixed Gold-Code and the allotment is known at the receiver. PRN codes serve two main functions w.r.t GPS/IRNSS as listed below:

- Sense presence of signal and satellite identification: If the signal received at the antenna correlates strongly with any shift of PRN code of a satellite, it implies the presence of signal from that particular satellite, with very high probability.
- Distance measurement: While a very coarse estimate of time of travel is obtained from the Navigation data, code phase measurements of the PRN codes, described later, provide the fine estimate used to get an accurate position fix.

Gold Code Generation

GPS/IRNSS Gold codes are generated, as XOR of two PRN sequences, using two 10-bit LFSR’s, G1 and G2. Refer to Figure 2.5a. At every clock event, the result of XOR-ing the bits at the feedback tap positions (3, 10 for G1 and 2, 3, 6, 8, 9, 10 for G2) is fed back into the shift register. And 10th bit of the two registers are XOR-ed to obtain the PRN sequence. This generates a PRN sequence with period of upto $2^{10} - 1$. Same setup is used for all satellites in GPS as well

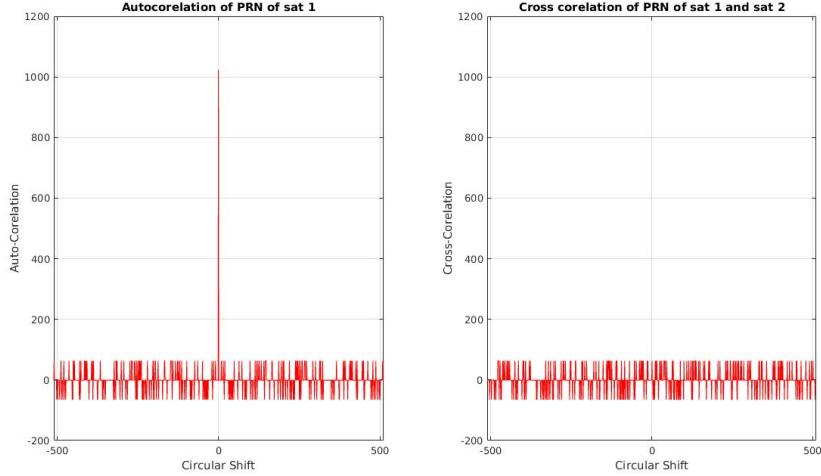


Figure 2.4: Correlation properties of PRN codes for $L = 1000$

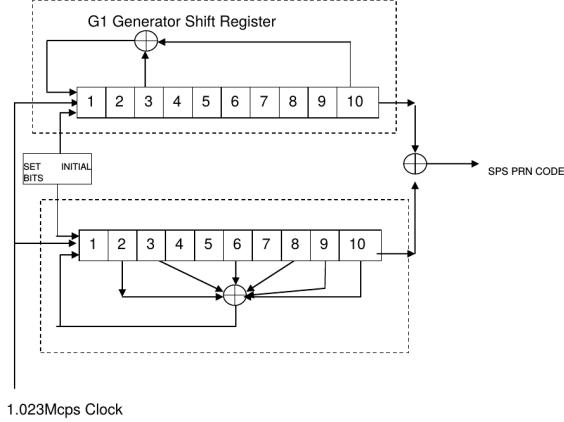
as IRNSS. What changes from satellite to satellite is the initialization of register G2. In each case, G1 is initialized to all 1's. G2 initializations for IRNSS satellites are shown in Table 2.5b. The table also gives first 10 chips in Octal2. Example, for satellite 1 of S-band, the first 10 chips are $\underbrace{1}_{1} \underbrace{100}_{4} \underbrace{010}_{2} \underbrace{000}_{0}$. For GPS, G2 is initialized to all 1's and output is taken as the XOR of last bit of G1 and selected two positions of G2. Which two positions depend on the satellite number (pseudocode to generate the codes is given in Chapter 4). Refer to Section 4.3.1.1 of [3] for details. If $\{X_n^{sv}\}_{n=0}^{1023}$ is the Gold Code corresponding to satellite sv , then the spreading sequence $x(t)$ is:

$$x(t) = \sum_{n=0}^{1023} X_n^{sv} \text{rect}\left(\frac{t - nT_c}{T_c}\right)$$

where $\text{rect}(t)$ is rectangular pulse with support $t \in [0, 1]$ ms and T_c is the chip duration, $\frac{1}{1.023} \mu\text{s}$ in case of GPS/IRNSS. The signal $x(t)$ lasts for $1023 \times T_c = 1\text{ms}$ and then repeats periodically.

2.1.2 Navigation Bits Framing & Time information

Navigation bits are transmitted at the rate of 50 bits per second, after error correction encoding. Navigation bits carry information about Ephemeris Data, satellite almanacs, time of transmission information, clock corrections, satellite health information and ionospheric models. Bits are arranged into frames and MSB is transmitted first. In GPS, a single subframe is of 1500 bits consisting of 5 sub-frames of 300 bits each, lasting 6 seconds ($20\text{ms} \times 300$), while IRNSS consists of 2400 bits long frames with 4 sub-frames of 600 bits each, lasting 12 seconds. Navigation data framing and error correction coding are the only differences between signal structure of GPS and IRNSS (Refer to Section 5 of [11] and Section 20.3 of [10] for details). We now look at parts of the navigation data that allow for frame synchronization and time computation.

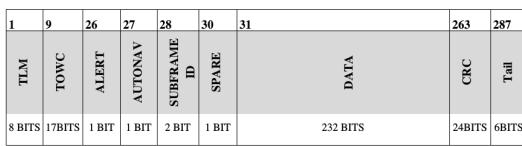


(a) Gold Code Generation LFSR setup

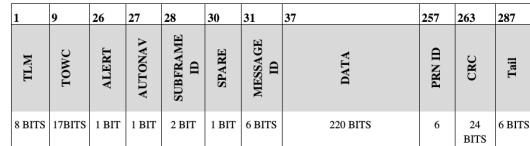
PRN ID	SV Location	L5-SPS		S-SPS	
		Initial Condition for G2 Register	First 10 Chips in Octal2	Initial Condition for G2 Register	First 10 Chips in Octal2
1	55°E	1110100111	130	0011101111	1420
2	55°E	0000100110	1731	0101111101	1202
3	83°E	1000110100	0713	1000110001	0716
4	111.75°E	0101110010	1215	0010101011	1524
5	111.75°E	1110110000	0117	1010010001	0556
6	32.5°E	0001101011	1624	0100101100	1323
7	131.5°E	0000010100	1753	0010001110	1561

(b) G2 register initializations for IRNSS

Figure 2.5: Gold Code Generation [11]



(a) IRNSS Subframes 1 & 2



(b) IRNSS Subframes 3 & 4

Figure 2.6: IRNSS subframe structure [11]

In GPS, each 300 bit subframe is divided into 10, 30-bit packets called ‘words’. Each word ends with 6 parity bits and is encoded using Hamming(32,26), where 32-bit block is formed with the current word and last two bits of the previous word. Refer to Figure 2.7. First word of every subframe is called the telemetry word (TLM), which contains an 8-bit preamble in the beginning. The preamble, 10001101, which is the same for all satellites indicates the beginning of a subframe. However, preamble-like combination may occur anywhere in the data, hence further parity checks are required to ascertain the presence of preamble in the decoded navigation bit stream (Figure 20-5 of [10]). TLM is followed by the Hand-Over Word (HOW). First 17 bits of the HOW are called Time of Week (TOW) bits and provide timing information as described below. IRNSS has a 600-bit subframe, which begins with the 16-bit synchronization sequence 1110 1011 1001 0000. The remaining 584 bits are encoded using $\frac{1}{2}$ rate convolution code and block interleaved with dimensions 8 rows \times 73 columns. After decoding, we will have 292 information bits which have a structure as shown in Figure 2.6.

Time Measurement

In case of GPS, satellite time is counted in units of 1.5 seconds. Currently, Zero time is taken to be the midnight between 21st August and 22nd August, 1999. Time is kept since the zero time point as a 29-bit number called the **Z-count**. The first 10 bits of Z-count store the number of weeks ($\text{mod } 2^{10}$) that have passed since the zero-point time, called as the Week Number (WN). First 10 bits of the Word 3 of Subframe 1 gives the WN. The last 19 bits of Z-count store

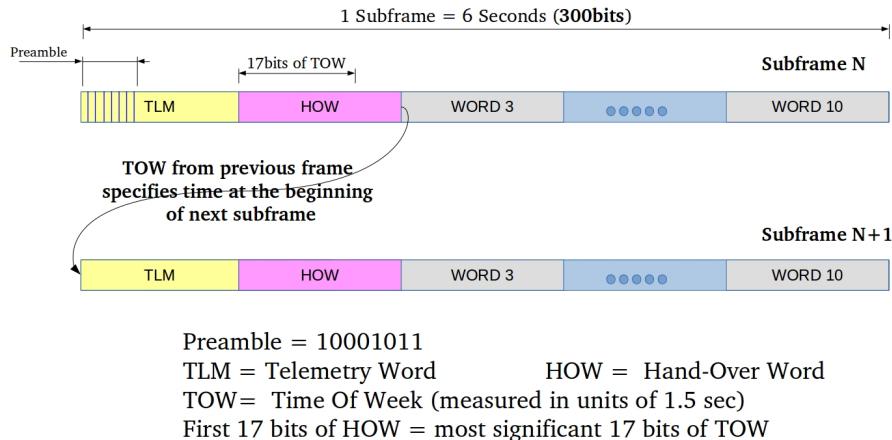


Figure 2.7: GPS subframe structure

the number of 1.5sec units that have passed since the beginning of the current week, referred to as TOW (Time-Of-Week). Note that 1 week is $\frac{3600*24*7}{1.5} = 403200$ 1.5sec units; and hence it can be stored as 19 bit number ($2^{19} = 524288$). Now, the satellite transmits the TOW of the beginning of the next subframe, as the most significant 17 bits of the HOW of the current subframe as shown in Figure 2.7. To specify the HOW of just the subframe beginnings, only 17 bits suffice because the subframes occur at an interval of 6 seconds, which is $2^2 \times 1.5$ seconds, allowing us to omit specifically mentioning them in the subframe word. They can be recovered at the receiver by simply counting the number of navigation bits since the previous HOW. **Thus the receiver can fully recover the Z-count maintained by the satellite at the time of transmission of the first bit of the current subframe being processed.** IRNSS maintains a 27-bit Z-count consisting of 10 bit WN and 17 bit TOW, as 12-second counts. The Z-count computation for IRNSS is same as that of GPS, except that IRNSS maintains 12 seconds count due to 600-bit subframe lengths as against 300 bit subframe lengths of GPS.

It should be noted that all the satellite clocks are synchronized, and hence the bit edges of all the satellites occur at the same time. In principle, a receiver can simply look for the edge of first bit of the preamble from atleast 4 satellites satellite, measure time differences between reception of each of them and hence get the 3 equations of pseudorange differences, $\rho_i - \rho_j$. This can be solved to get the user location, after substituting the satellite locations as derived from orbital parameters conveyed by the navigation bits. However, the catch is that these time differences are in the order of 10's of milliseconds and must be measured with a resolution of micro-seconds, to get reasonably good position accuracy. The PRN codes enable these finer measurements, as described in the last section.

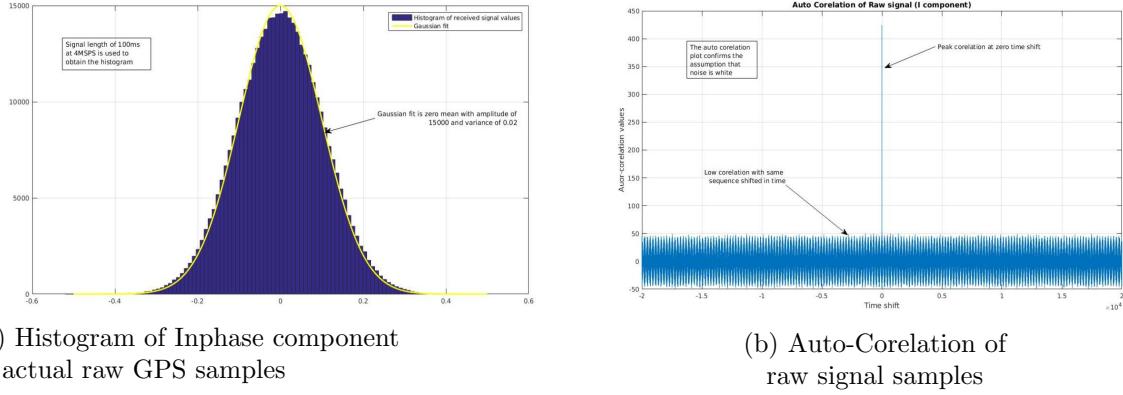


Figure 2.8: Raw signal properties

2.2 Signal Reception

Signal, as described in previous section is transmitted by the satellite orbiting earth at very high speeds (14,000 km/hour) from distance of 20,000 km to 25,000 km above the surface of Earth. This has two significant effects on the signal:

- **Doppler Shift:** Relative motion between the orbiting satellite and the receiver causes a Doppler shift (f_d) and the signal at reception will be off from center frequency of f_{L1} by f_d . The speed of the satellite and of the receiver itself, due to rotation of Earth or otherwise, contribute to the Doppler Shift. For GPS f_d varies in the range of $\pm 6\text{kHz}$ [2]. f_d changes over time depending on where the satellite is in its orbit. Absolute f_d is maximum for satellites at low elevation angles and nearly zero when the satellites are over-head. Unlike GPS, IRNSS has satellites in the geosynchronous and geostationary orbits. Thus range of f_d would be smaller for IRNSS. The S-band signal will have higher Doppler shifts than the L5 signals, owing to higher center frequency. Doppler shift also affects the PRN codes. PRN codes at a frequency of 1.023MHz will suffer a worst case shift of $\frac{6000*1.023}{1575.42} = \frac{6000}{1540} \approx 4\text{Hz}$ i.e shift of one C/A chip every 0.25 seconds. Doppler shift is unknown apriori at the the receiver and finding it is the first step of processing.
- **Noise:** Signal suffers significant path loss and additive noise due to the atmosphere and receiver electronics. As mentioned in 1.1, the typical reception SNR values of GPS signals in 20.46MHz bandwidth is -160dBW. The signal is buried about -24.5dB [14] under the thermal noise floor, and won't be visible on the frequency spectrum of raw reception. In the analysis, noise $n(t)$ is assumed to be a zero-mean additive white Gaussian process. This is a reasonable assumption as illustrated in Figure 2.8. The one-sided noise power density is N_0 .

So, let f_d be the unknown Doppler shift at f_{L1} . The received signal $s_r(t)$ and transmitted

signal $s_t(t)$, are given as (Equation 2.1):

$$\begin{aligned}s_t(t) &= \sqrt{2P_t} D(t)x(t)\cos(2\pi f_{L1}t) \\ s_r(t) &= \sqrt{2P_r} D(t-\tau)x(t-\tau)\cos(2\pi \hat{f}t + \theta) + n(t)\end{aligned}\quad (2.2)$$

where τ = net time of travel of the signal or the ‘code-phase’ at reception ,

$$\hat{f} = f_{L1} + f_d,$$

θ = unknown carrier phase at reception

$n(t)$ = additive noise

Strictly speaking, the PRN signal $x(t)$ should also have some Doppler correction, however as calculated above, the correction is small. Hence we neglect it in here. However estimating it is necessary, and it will be dealt with in a later section on tracking. $s_r(t)$ is the signal at antenna of the receiver from one satellite. The signal must be appropriately filtered, amplified and down-converted before being sampled and processed. The design details of front-end is dealt with in the next chapter. For the sake of analysis, let’s assume that the signal is down converted to frequency f_{IF} , which is sufficiently larger than signal main lobe bandwidth of 2.046MHz. Let $s(t)$ and $\tilde{n}(t)$ be the resulting signal and noise process respectively. Thus, after down-conversion, we obtain

$$s(t) = \sqrt{P_r} D(t-\tau) x(t-\tau) \cos(2\pi(f_{IF} + f_d)t + \theta) + n(t) \quad (2.3)$$

$s(t)$ is then sampled and quantized. Quantization of raw input signal in GPS receivers is typically only one- or two-bit and this suffices as long as there is no strong interference signal.

2.3 Signal Processing

The goals of signal processing on $s(t)$ are as follows:

- Identify the satellites whose signals are being received
- Estimate, code-phase shift τ , Doppler shift f_d and Carrier Phase θ (optional) for each of the satellites
- Recover Navigation Data bits $D(t)$
- Perform error check and correction
- Compute pseudoranges to each visible satellite

If at least 4 pseudoranges are simultaneously available at the receiver, coordinates can be computed. This section describes the processing involved in achieving the tasks listed above. To illustrate the concepts, we treat the signals in continuous time domain although in practice the signals are sampled before performing further operations described. Practical details are dealt with in Chapters 3 and 4.

2.3.1 Calculating the pseudorange

Suppose there are K transmitting satellites visible at the receiver, then the signal received post front-end processing is,

$$s_r(t) = \sum_{i=1}^K \sqrt{P_i} D_i(t - \tau_i) x_i(t - \tau_i) \cos(2\pi(f_{IF} + f_{d,i})t + \theta_i) + n(t)$$

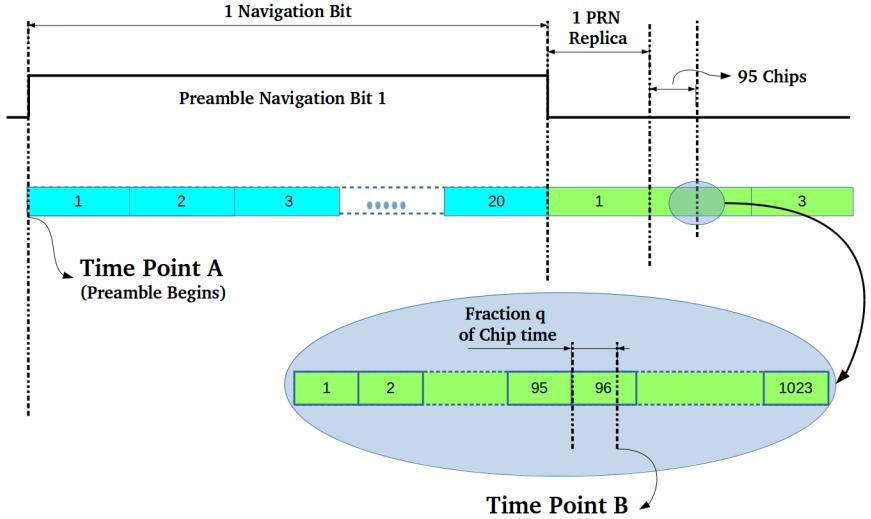
The receiver must detect the signals from these K satellites, remove the residual carrier, separate out the K signals from $s_r(t)$, say $\{s_i(t) = D_i(t - \tau_i) x_i(t - \tau_i)\}_{i=1}^{i=K}$, and compute the corresponding K pseudoranges, say $\{\rho_i\}_{i=1}^{i=K}$. Assume that the $s_i(t)$'s have been obtained. Let us now see how pseudoranges are computed. Consider $s_1(t)$ for instance, from some satellite X. The receiver detects the beginning of the first preamble in $s_1(t)$, call it time point A. Let T_A be the GPS reference time at which satellite X transmitted the point A. Consider another point B after A, as shown in Figure 2.9. Let it be transmitted at time T_B . The relative time difference $T_B - T_A$ can be precisely measured using the structure of the GPS signal as shown. In general if A and B are separated by N Navigation bits, R PRN replicas after the last NAV bit, C code chips after the last PRN replica and fraction q of the current chip, then the relative time of transmission is given by,

$$(T_B - T_A) = N \times 20\text{ms} + R \times 1\text{ms} + C \times \frac{1}{1023}\text{ms} + q \times \frac{1}{1023}\text{ms}$$

Also note that the Z-count gives the precise absolute GPS time of beginning of every preamble. Thus T_A can be found out and hence absolute time of transmission of any point within the signal can be computed. The signal therefore, not only carries the information bits, but also acts like a clock carrying information about when it was transmitted, with atomic-clock like accuracy (except for some clock offset, relativistic and other errors). The receiver on decoding a signal at any time point B knows the GPS time of its transmission. For N Navigation bits since the beginning of preamble, R PRN replicas since the last NAV bit, C code chips since the last PRN replica and fraction q of the current chip it is given by,

$$T_B(\text{in miliseconds}) = \text{Z-count} \times 1500 + N \times 20 + R \times 1 + \frac{C}{1023} + \frac{q}{1023}$$

This gives the time of transmission. However, to measure the time of travel, receiver needs to know the precise GPS time of reception as well. This is not possible unless the receiver has a clock synchronized with the satellites. Thus receiver clock offset ΔT wrt GPS time becomes the fourth unknown in the system, along with the three receiver coordinates Accounting for this, the receiver needs $K \geq 4$ pseudorange values to solve for the receiver position. There are two ways of computing the pseudoranges, which give similar result[15].



$$\text{Time between point A and B} = (1 \text{ Nav Bit}) \times (20\text{ms}) + (1 \text{ PRN}) \times (1\text{ms}) + (95 \text{ Chips}) \times (1/1023)\text{ms} + (q) \times (1/1023)\text{ms}$$

Figure 2.9: Measuring time relative to preamble beginning

Common Transmission Time

Let $K = 4$. Refer to Figure 2.10. 4 satellites transmit the Preamble, say at time T_0 . Signal from satellite X reaches the receiver after a delay of T_1 , at GPS time $T_0 + T_1$. Similarly, signals from satellites Y, Z and U reach the receiver at GPS time $(T_0 + T_2)$, $(T_0 + T_3)$ and $(T_0 + T_4)$ respectively. Let the receiver have an unknown clock offset of ΔT compared to GPS time. Receiver can compute T_0 precisely on receiving preamble from satellite X, as described above. It looks at time of reception of preambles for each of the signal using its own clock and concludes them to be $\hat{T}_i = T_0 + T_i + \Delta T$ for $i = 1, 2, 3, 4$. It computes pseudoranges as $\rho_i = c(\hat{T}_i - T_0)$, while actual ranges are $r_i = c(\hat{T}_i - T_0 - \Delta T)$ and obtains satellite locations (x_i, y_i, z_i) for $i = 1, 2, 3, 4$ from Navigation bits. Using the measured \hat{T}_i 's and T_0 it then solves the following set of equations (after some atmospheric error corrections etc), for the four unknowns namely receiver co-ordinates (x_r, y_r, z_r) and clock offset ΔT

$$r_i = c(\hat{T}_i - T_0 - \Delta T) = \sqrt{(x_i - x_r)^2 + (y_i - y_r)^2 + (z_i - z_r)^2} \text{ for } i = 1, 2, 3, 4$$

In principle, this is how a pseudorange computation using common transmission time signal works. Here we observe the starting edge of the preamble, which we know were transmitted simultaneously by all the satellites.

In implementing this procedure however we note that computing T_0 is unnecessary to get receiver

position. By eliminating T_0 and ΔT above equations can be re-written as:

$$\begin{aligned} c(\hat{T}_1 - \hat{T}_2) &= \sqrt{(x_1 - x_r)^2 + (y_1 - y_r)^2 + (z_1 - z_r)^2} - \sqrt{(x_2 - x_r)^2 + (y_2 - y_r)^2 + (z_2 - z_r)^2} \\ c(\hat{T}_1 - \hat{T}_3) &= \sqrt{(x_1 - x_r)^2 + (y_1 - y_r)^2 + (z_1 - z_r)^2} - \sqrt{(x_3 - x_r)^2 + (y_3 - y_r)^2 + (z_3 - z_r)^2} \\ c(\hat{T}_1 - \hat{T}_4) &= \sqrt{(x_1 - x_r)^2 + (y_1 - y_r)^2 + (z_1 - z_r)^2} - \sqrt{(x_4 - x_r)^2 + (y_4 - y_r)^2 + (z_4 - z_r)^2} \end{aligned} \quad (2.4)$$

We need $(T_A - T_B)$, $(T_A - T_C)$ and $(T_A - T_D)$ in Figure 2.10, to construct the equations above. These can be computed as discussed previously, using the signal from satellite X as reference and measuring delay in occurrences of preambles from other satellites w.r.t edge of preamble from satellite X.

Common Reception Time

Common reception time method considers the signal across all satellites at a give time instance, say at point D (t_D). So we have the signal points $\{s_1(t_D), s_2(t_D), s_3(t_D), s_4(t_D)\}$ which are transmitted at different instances by different satellites but received at the same time by the receiver. Now the receiver computes the times at which each of these were transmitted from the respective satellite. Since the preambles were all transmitted at the same time T_0 , measuring the time from point D to the respective preambles (S_1, S_2, S_3 in Figure 2.10) gives the transmission times. These differences are used in RHS in the pseudorange equations 2.4 and solved.

Ideally, the two methods are equivalent and $S_i = T_4 - T_i$. The common transmission time method however assumes that the ΔT is same across the various time instances at which the preambles from different satellites are received, which if not true will introduce errors in the relative pseudoranges. ΔT will change if there is a offset in receiver clock frequency. As long as this drift is small, the clock offset will not change much in the interval of measurement. The common reception time, on the other hand uses signals transmitted by the satellites at slightly different times. And given that satellites are constantly moving, the pseudoranges measured are not simultaneous. However for small durations the change in range should not be much. Discounting these two subtleties, the two methods are same. The implementation detailed in this report uses the common transmission method. It should be noted that since satellites are at a range varying from 20000 km to 25000 km, the travel times are typically $\frac{20000 \times 10^3}{3 \times 10^8} \approx 65\text{ms}$ to 80ms. And the time difference between preambles of two satellites are typically less than 15ms (less than a navigation bit length). Also, when more than 4 satellites are visible, it results in an over-determined system of equations. These can be solved for the minimum square solution and result in improved accuracy compared to having just 4 satellites.

Now consider the processing involved in obtaining $\{s_i(t) = D_i(t - \tau_i)x_i(t - \tau_i)\}_{i=1}^K$ from the front-end output $s(t) = \sum_{i=1}^K \sqrt{P_i} D_i(t - \tau_i)x_i(t - \tau_i) \cos(2\pi(f_{IF} + f_{d,i})t + \theta_i) + n(t)$

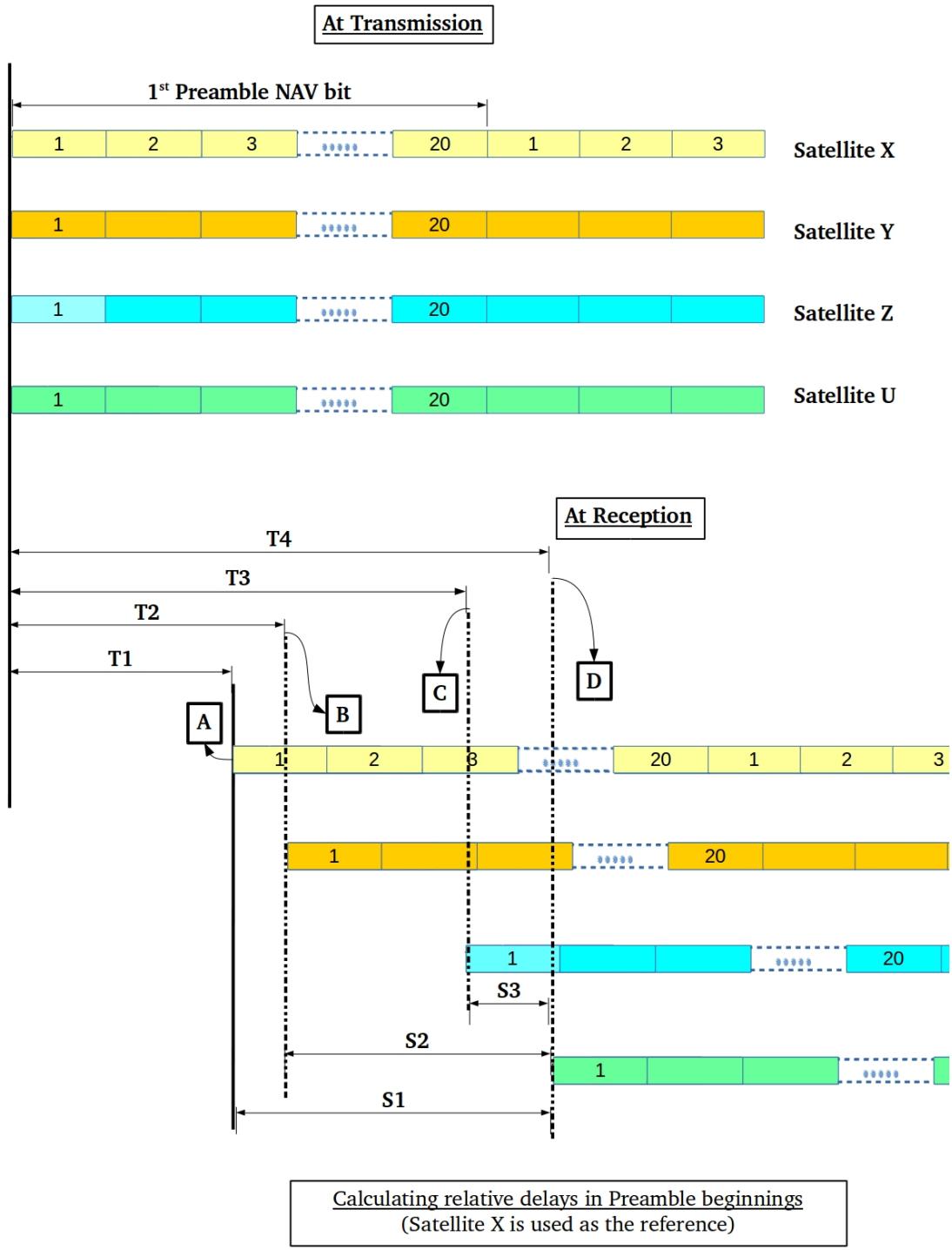


Figure 2.10: Measuring relative delays using preamble beginnings

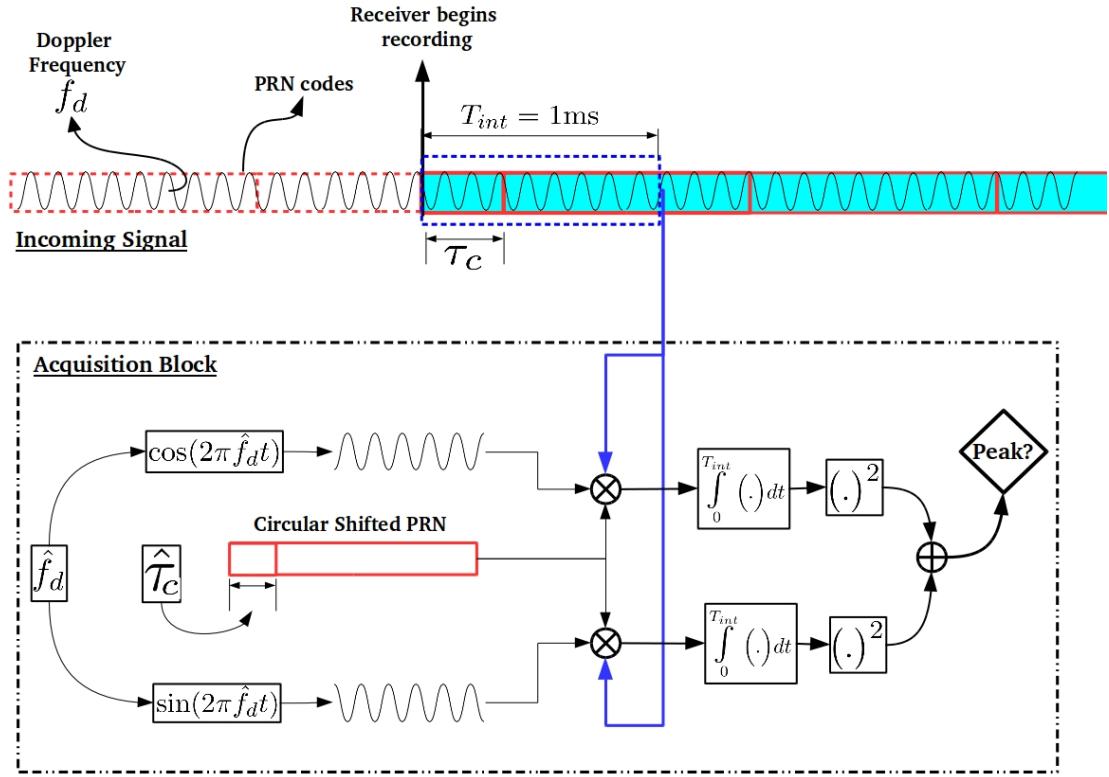


Figure 2.11: Acquisition with $T_{int} = 1\text{ms}$ (Note that f_{IF} has been taken to be 0)

2.3.2 Acquisition

The process of finding the satellites signals present and getting a coarse estimate of the Doppler frequency shift and the ‘code-phase’ for each of them is referred to as ‘Acquisition’. When cold started, the receiver starts recording the incoming signal from an unknown point of the PRN sequence and it differs from satellite to satellite since the different signals arrive out of sync depending on the distance of satellites from the receiver. To be able to de-spread the signal from the j^{th} satellite, the receiver must align with the PRN sequence $x_j(t)$ of that satellite and then decode the data bits. This requires knowledge of τ_j modulo 1ms as described below. But this operation requires the baseband signal i.e $D_j(t - \tau_j)x_j(t - \tau_j)$. So the carrier must be stripped off the signal, but it is necessary to know the Doppler shift $f_{d,j}$ to do so.. Thus if the signal from a particular satellite is present, the first step is to get a coarse estimate of the f_d and $(\tau \bmod 1)$, without knowing θ . Knowing f_d will eliminate the carrier completely and give the baseband signal. Knowing $(\tau \bmod 1)$ then will align the receiver code generator with incoming signal by introducing a code-offset of $(\tau \bmod 1)$ as described below. However given the low SNR of the CDMA signal, it requires a brute-force search over frequency and code-phase space to get the coarse estimates. Since each satellite needs to be acquired independently, in subsequent analysis, we will consider the signal only from one satellite and treat signals from

other satellites as noise. Thus

$$s_r(t) = \sqrt{P} D(t - \tau) x(t - \tau) \cos(2\pi(f_{IF} + f_d)t + \theta) + \tilde{n}(t)$$

$$\text{where } \tilde{n}(t) = n(t) + \sum_{i=1}^{K-1} \sqrt{P_i} D_i(t - \tau_i) x_i(t - \tau_i) \cos(2\pi(f_{IF} + f_{d,i})t + \theta_i)$$

Now, refer to Figure 2.11. Here, the receiver begins recording from some arbitrary point in the signal. The nearest beginning of the PRN sequence is say a τ_c ms away. We will refer to this as the acquisition code-phase. Note that actually $\tau_c = (\tau \bmod 1\text{ms})$. Consider a time interval of length T_{int} , which is a multiple of 1ms, the length of PRN codes. $s_r(t)$ is correlated with a local signal generated according to receiver's estimate, say $\{\hat{f}_d, \hat{\tau}_c\}$. Co-relation is computed as:

$$\begin{aligned} \Re_I(\hat{f}_d, \hat{\tau}_c) &= \int_0^{T_{int}} s_r(t) x(t - \hat{\tau}_c) \cos(2\pi(f_{IF} + \hat{f}_d)t) dt \\ \Re_Q(\hat{f}_d, \hat{\tau}_c) &= \int_0^{T_{int}} s_r(t) x(t - \hat{\tau}_c) \sin(2\pi(f_{IF} + \hat{f}_d)t) dt \end{aligned}$$

Let, $\Re(\hat{f}_d, \hat{\tau}_c) = \Re_I + j\Re_Q$, where $j = \sqrt{-1}$,

$$\|\Re(\hat{f}_d, \hat{\tau}_c)\|^2 = \Re_I^2 + \Re_Q^2$$

$\|\Re\|^2$ is known as the ambiguity function. The aim of acquisition is to search over the space of $\hat{f}_d, \hat{\tau}_c$ to find a distinctly high value of $\|\Re\|^2$. Since PRN code of a satellite correlates strongly only with a correctly aligned PRN code of the same satellite, presence of a peak confirms presence of signal from that satellite and also identifies the correct alignment. If such a peak occurs, the satellite is declared present, the bin $\hat{f}_d, \hat{\tau}_c$ at which it occurs is chosen as the coarse estimate and the receiver proceeds to tracking. Let $\{\tilde{f}_d, \tilde{\tau}_c\}$ be the estimates concluded from the acquisition process. Thus the output of acquisition is,

$$\{\tilde{f}_d, \tilde{\tau}_c\} = \operatorname{argmax}_{\{\hat{f}_d, \hat{\tau}_c\}} \|\Re(\hat{f}_d, \hat{\tau}_c)\|^2$$

Alternatively, \Re can be written in complex notation.

$$\Re(\hat{f}_d, \hat{\tau}_c) = \int_0^{T_{int}} s_r(t) x(t - \hat{\tau}) e^{(j2\pi\hat{f}_d t)} dt$$

Substituting for $s_r(t)$ from 2.3, we get,

$$\begin{aligned}\Re(\hat{f}_d, \hat{\tau}_c) &= \int_0^{T_{int}} \left(\sqrt{P_r} D(t-\tau) x(t-\tau) \cos(2\pi(f_{IF} + f_d)t + \theta) + \tilde{n}(t) \right) x(t - \hat{\tau}_c) e^{(j2\pi(f_{IF} + \hat{f}_d)t)} dt \\ &= \left(\sqrt{P_r} \int_0^{T_{int}} D(t-\tau) x(t-\tau) x(t - \hat{\tau}_c) \cos(2\pi(f_{IF} + f_d)t + \theta) e^{(j2\pi(f_{IF} + \hat{f}_d)t)} dt \right) + \tilde{n}\end{aligned}$$

where \tilde{n} is a term resulting from noise. If T_{int} is taken much less than 20ms, like 1 or 2ms, then $D(t)$ would be constant in the interval. So take it to be $D = \pm 1$ (neglecting the possibility a data bit boundary in the T_{int} -long segment). When $\hat{\tau}_c = \tau_c (= \tau \bmod 1)$, $x(t-\tau)x(t - \hat{\tau}_c) = x(t-\tau)^2 = 1$. Then,

$$\begin{aligned}\Re(\hat{f}_d, \tau_c) &= \sqrt{P_r} D \int_0^{T_{int}} \left(\frac{e^{j(2\pi(f_{IF} + f_d)t + \theta)} + e^{-j(2\pi(f_{IF} + f_d)t + \theta)}}{2} \right) e^{j2\pi(f_{IF} + \hat{f}_d)t} dt + \tilde{n} \\ &= \sqrt{P_r} D \int_0^{T_{int}} \left(\frac{e^{j(2\pi(2f_{IF} + f_d + \hat{f}_d)t + \theta)} + e^{j(2\pi(\hat{f}_d - f_d)t + \theta)}}{2} \right) dt + \tilde{n}\end{aligned}$$

Since $2f_{IF} \gg \frac{1}{T_{int}}$, we have $\int_0^{T_{int}} e^{j(2\pi(2f_{IF} + f_d + \hat{f}_d)t + \theta)} dt \approx 0$

$$\begin{aligned}\therefore \Re(\hat{f}_d, \tau_c) &\approx \sqrt{P_r} D \int_0^{T_{int}} \left(\frac{e^{j(2\pi(\hat{f}_d - f_d)t + \theta)}}{2} \right) dt + \tilde{n} \\ &= \frac{\sqrt{P_r} e^{j\theta}}{2} D \int_0^{T_{int}} (e^{j2\pi\delta_{f_d} t}) dt + \tilde{n} \quad (\delta_{f_d} = \hat{f}_d - f_d) \\ &= (D e^{j\theta} \sqrt{P_r}) \frac{\sin(\pi\delta_{f_d} T_{int})}{2\pi\delta_{f_d}} e^{\pi\delta_{f_d} T_{int}} + \tilde{n}\end{aligned}\tag{2.5}$$

Neglecting the effect of \tilde{n} , observe that:

- $\|\Re\|^2$ is independent of θ and D . Thus acquisition process will function the same without modification irrespective of the carrier phase on reception and the navigation data in the chosen signal segment
- $\lim_{\delta_{f_d} \rightarrow 0} \|\Re\|^2 = P_r$. For values of $\{\hat{f}_d, \hat{\tau}_c\}$ other than $\{f_d, \tau_c\}$, $\|\Re\|^2 < P_r$. Thus ideally, under low noise conditions, output of acquisition are the actual Doppler and code shifts.

The Doppler shift in frequency is in most cases known to be in within $\pm 6\text{kHz}$. Hence the frequency search space is $-6000 \leq \hat{f}_d \leq 6000$. And since the PRN sequence lasts 1ms, code-

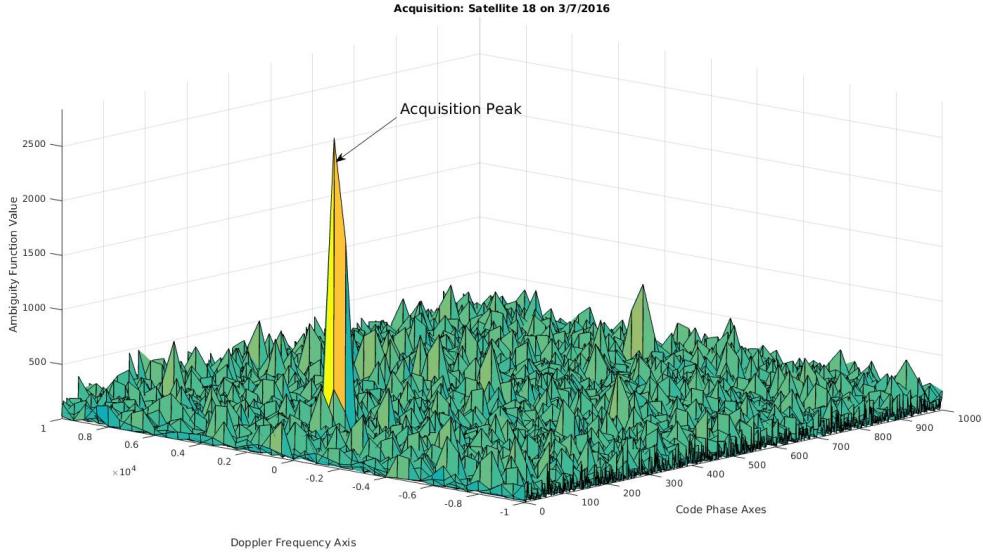


Figure 2.12: Result of Acquisition procedure as a 2D plot on frequency code-phase plane

phase search space is $0\text{ms} \leq \hat{\tau}_c < 1\text{ms}$. If we could search over all possible values in this search space, we can obtain the exact Doppler shift and align the local code generator exactly with the incoming signal. This routine could be repeated every once in a while when Doppler frequency or the code-phase has changed and realign the receiver. However, it is not possible to search over the entire search space, so we discretize the search space with a frequency bin size of say Δf_d and code-phase bin of $\Delta \tau_c$. That is, the ambiguity function is only computed for every Δf_d^{th} frequency and every $\Delta \tau_c^{\text{th}}$ code-phase shift. And hence the estimates $\{\tilde{f}_d, \tilde{\tau}_c\}$ thus obtained are ‘coarse’. To check all code shifts in the PRN sequence of length 1023, we need $\Delta \tau_c \leq \frac{1}{1023}\text{ms}$ and for reasons detailed in Chapter 3, we need $\Delta f_d \leq 500\text{Hz}$ if $T_{int} = 1\text{ms}$. If we take them as $\frac{1}{1023}\text{ms}$ and 500Hz , the output \tilde{f}_d and $\tilde{\tau}_c$ will be among $\{-6000, -5500, \dots, -500, 0, 500, \dots, 6000\}\text{Hz}$ and $\{0, \frac{1}{1023}, \frac{2}{1023}, \dots, \frac{1022}{1023}\}\text{ms}$ respectively. The number of ambiguity functions to be computed (or size of search space) is $\frac{6000 - (-6000)}{\Delta f_d} \times \frac{1}{\Delta \tau_c} \approx 24,000$. Performing 24,000 correlations is a computationally expensive task, thus even after discrete-izing the search. Besides, f_d and τ_c change only slowly and smoothly with time. So it is wasteful to perform acquisition often. In practice, acquisition is thus performed only once while the receiver is cold-started. The resulting coarse estimate is used as the initial condition for the tracking block which then settles to the a finer estimate and tracks f_d and τ_c to keep the receiver aligned with incoming signal thereafter. An example of acquisition output is shown in Figure 2.12. As long as the SNR is high enough (typically ≥ 40 dB-Hz), it is unlikely that noise generates a peak which is higher than at the nearest estimates for $\{f_d, \tau_c\}$. Let us assume so and proceed to tracking.

In GPS, on cold start, the receiver has no information about the visible satellites. Thus it performs acquisition with all satellite PRN codes till it finds 4 or more satellites. In case of IRNSS however all the 7 satellites (as of June 2017) are visible all the time over Indian

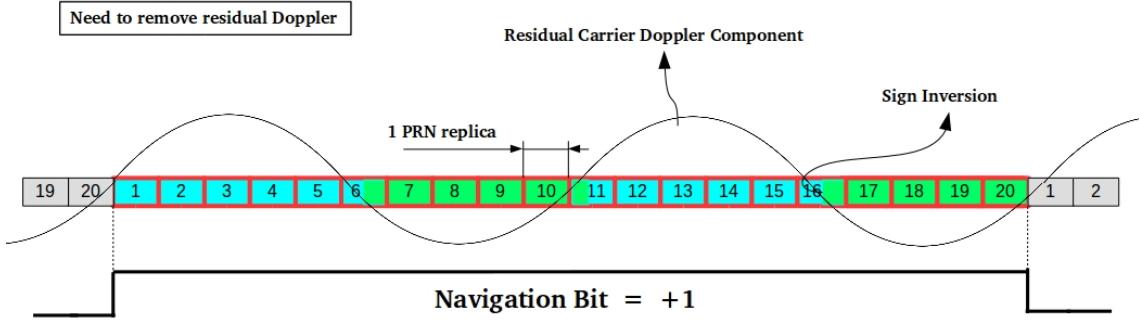


Figure 2.13: Effect of Residual Doppler after acquisition. Green implies inverted code-chips

subcontinent. The estimates $\{\tilde{f}_d, \tilde{\tau}_c\}$ for each of the satellites are used to initialize many separate tracking loops. Each such loop is called a ‘channel’ and the specification, ‘number of channels’ in a receiver refers to the maximum number of satellites that the receiver can track simultaneously. But sometimes it also simply refers to the number of correlators in the receiver. The number of channels can be as large as 22 to track multi-constellation satellites for better accuracy or simply to reduce time-to-first-fix by computing many correlations parallel.

2.3.3 Tracking

We consider the tracking block for one satellite. If acquisition detects the correct peak, the receiver concludes the estimates $\{\tilde{f}_d, \tilde{\tau}_c\}$, where we have,

$$|\tilde{f}_d - f_d| \leq \frac{\Delta f_d}{2} \quad (2.6)$$

$$|\tilde{\tau}_c - \tau_c| \leq \frac{\Delta \tau_c}{2} \quad (2.7)$$

The aim now is to not only decode the Navigation bits, but also to maintain the code-phase synchronization thereby precisely detecting bit and PRN sequence boundaries, used in pseudo-range computation. We now motivate the need and functioning of the two main components that make up the tracking block.

PLL: Remove Residual Doppler and Phase

Observe from Equation 2.5 that, if $\delta_{f_d} = 0$, $\Re = De^{j\theta}$ (ignoring P_r and noise). Thus, ignoring the effects of noise, if $\delta_{f_d} = 0$ is maintained, the data bit sequence can be obtained from the sign of real (or imaginary) part of \Re alone, it is $D \cos \theta$ (or $D \sin \theta$). Note that the bit sequence so obtained will either be D or $-D$ depending on what θ is, which can then be distinguished using NAV frame elements, like the preamble. Moreover, we can find θ from the phase of complex number \Re , and get $D = e^{-j\theta}\Re$. This is like making θ zero, in which case all the power in the signal is in the In-phase component. This is helpful because, when considering the effect of noise term $\tilde{n} = \tilde{n}_I + j\tilde{n}_Q$, taking the sign without correcting for θ actually gives $\text{sign}[D \cos(\theta) + \tilde{n}_I]$.

Clearly, greater the θ , more likely is a erroneous flip in sign due to the noise term. Multiplying by $e^{-j\theta}$ gives, $\text{sign}[D + \tilde{n}'_I]$. \tilde{n}'_I and \tilde{n}_I have same variance owing to circular symmetry of the noise. Thus, compensating for the phase θ , after ensuring $\delta_{f_d} = 0$ gives a robust method for decoding NAV bits. However, after acquisition, carrier can be removed except for a residual of $\frac{\Delta f_d}{2}$. So, if $\Delta f_d = 500\text{Hz}$, a sinusoid with $0 \leq \delta_{f_d} \leq 250 \text{ Hz}$ still modulates the baseband signal $D(t)x(t)$. And this residual keeps inverting the chip sequence as shown in Figure 2.13. If the residual is of 100 Hz, it inverts the PRN sequence in every 5th replica, and NAV bit cannot be decoded.

The tracking block of receivers have a **Phase Lock Loop (PLL)** to correct for the residual Doppler frequency and the phase, finally shifting all incoming power to the inphase component, the sign of which gives the NAV bit. Sometimes a **Frequency Lock Loop** is initially used to correct for δ_{f_d} and then followed with a PLL to correct for θ . Refer to Figure 2.14. The basic function of a PLL is to track the phase of an input signal, so that the output signal is phase-locked to input. Let the input be $s_i(t) = \cos(\omega t + \phi(t))$, where the frequency ω is known and phase $\phi(t)$ is a function varying with time at a much slower rate than ω . A frequency modulated signal is one such example. Suppose the PLL outputs $s_o(t) = \cos(\omega t + \hat{\phi}(t))$. First block in the PLL computes the error $e(t) = \phi(t) - \hat{\phi}(t)$. The function that extracts phase difference, or at least a good estimate of it, from the input and output signals is called the ‘discriminator’ function or the ‘phase detector’. A naive example would be $[\cos^{-1}(s_i) - \cos^{-1}(s_o)]$, but in practice we need something that can be computed easily. The error signal is then subject to a filter, to minimize effects of noise and achieve desirable dynamical properties which decide how fast a change can be tracked, how stable the loop is etc. The resulting feedback signal is then fed to a signal generator, which speeds up or slows down according to the feedback, to match the input phase. In hardware, it is usually a Voltage Controlled Oscillator (VCO). A FLL is similar except that the first block would use a discriminator which computes difference in frequencies irrespective of the phase and correct for it. In this report we will only describe a PLL that corrects for phase and small frequency differences like the residual Doppler. We treat $(2\pi\delta_{f_d}t + \theta)$ as the smooth slowly varying phase $\phi(t)$ and track it, not correcting for δ_{f_d} separately.

Let $\Phi_i(s), \Phi_o(s), E(s)$ and $Y(s)$ be the Laplace transforms of the the input phase, output phase, error and feedback signals respectively. Note that we directly work with phase signals i.e we assume the discriminator has given a good estimate of the input phase and that the signal generator faithfully converts the output phase to output signal. Let $K \times H(s)$ be the PLL filter transfer function. $G(s)$ is the signal generator transfer function which is usually simply an integrator (i.e $G(s) = \frac{1}{s}$). This is so because a VCO varies frequency in proportion to the feedback input. Thus phase is obtained as an integral of the feedback.

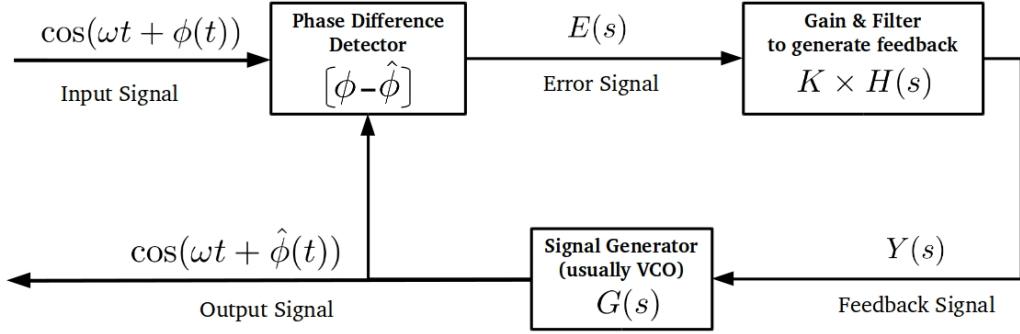


Figure 2.14: Basic Structure of a Phase Lock Loop

The PLL can thus be characterized as,

$$E(s) = \Phi_i(s) - \Phi_o(s) \quad (2.8)$$

$$Y(s) = K H(s) E(s) \quad (2.9)$$

$$\Phi_o(s) = G(s) Y(s) \quad (2.10)$$

$$\therefore \Phi_o(s) = K G(s) H(s) (\Phi_i(s) - \Phi_o(s)) \quad (2.11)$$

$$\Phi_o(s) = \frac{KG(s)H(s)}{1 + KG(s)H(s)} \quad (2.12)$$

$$\text{and } E(s) = \frac{1}{1 + KG(s)H(s)} \quad (2.13)$$

Now, K and $H(s)$ can be chosen so that the roots of $1 + KG(s)H(s)$ lies in the left half of complex plane, thus ensuring that the error signal vanishes over time and phase-lock between input and output. Note that by introducing a constant offset in the discriminator function, we can use the same concept to maintain an output at a fixed non-zero phase difference from the input as well. In particular we can generate $\sin(\omega t + \hat{\phi}(t))$, along with $\cos(\omega t + \hat{\phi}(t))$ shown in Figure 2.14. This is used the GPS receiver as described below. In a GPS receiver, the input to PLL is the raw GPS signal from the front-end after stripping it off the PRN code based on the code-phase estimate. In short it is just the residual sinusoid multiplied with NAV bits and additive noise i.e $s_i(t) = D(t - \tau) \cos(2\pi f_{IFT} t + \underbrace{(2\pi\delta f_d t + \theta)}_{\phi(t)}) + \tilde{n}(t)$. Once we obtain a signal

from PLL, phase locked to this residual and its quadrature signal (i.e both sine and cos), we can use it to remove the residual, leaving behind only the NAV bits. We simply compute a \Re -like correlation between the raw GPS input signal and the PLL outputs, and observe the sign of real part of the result. This gives the NAV bit D (or $-D$). Also, the frequency of PLL output signal actually tracks the Doppler frequency in raw GPS signal. Note however that keeping the synchronization with code-phase to remove the PRN sequence is crucial for functioning of the PLL and the next section describes how it is done using the Delay Lock Loop.

After having ensured that the PLL error converges to 0 with time, the guidelines listed

below must be considered in designing $KH(s)$ for GPS/IRNSS receiver. Although a continuous time PLL was described, the intuition behind designing a digital PLL like the one used in the implementation, remain the same.

- Noise: Since GPS signal is very noisy, the discriminator must construct a phase difference estimate which is robust to noise. The estimate can not be based on individual signal samples like $\cos^{-1}(s_i(t))$, since they would be too noisy. Rather, the estimate should be generated as an average of the phase over long times to minimize the effect of noise and then fed to PLL. Hence, as we will see in Chapter 3, the GPS/IRNSS PLL operates with an update rate of 1 kHz, while the samples are actually arriving at 4 MHz.
- Agnostic to 180° phase difference: Since the input to PLL is $D(t) \cos(\omega_{IF}t + \phi(t))$, it will have abrupt phase change of π at NAV bit boundaries. So when $D(t)$ changes from $+1$ to -1 for instance, the phase detector will see the phase change from $\phi(t)$ to $\phi(t) + \pi$. This sudden large change in discriminator may prompt the PLL to give a large corrective feedback, while infact we do not want the PLL to respond to it and preserve the Data bits. PLL discriminators are insensitive to shifts of 2π , since $\cos(\omega t + \phi(t))$ and $\cos(\omega t + \phi(t) + 2\pi)$ are the same signals. However, in GPS receiver we need discriminators insensitive to π shifts, due to NAV bits. So an ideal discriminator function for GPS should compute $(\phi(t) - \hat{\phi}(t))$ modulo π . Discriminator called *atan* or \tan^{-1} discriminator is used in the software implementation. It is discussed in Chapter 3.
- Capture Range: Capture range is the maximum frequency offset that the PLL can correct when it is started. In our case the frequency offset is δ_{f_d} which at most can be $\frac{\Delta f_d}{2}$. Thus PLL must be designed with a Capture range of atleast $\frac{\Delta f_d}{2}$.
- Lock Range: Lock range is the maximum sudden change in the input frequency that the PLL can correct for. In GPS receivers, a local oscillator (LO) is used to down convert the signal from f_{L1} to f_{IF} . LO's may have sudden drifts in their outputs which will reflect in the PLL input as offsets in ω_{IF} . The PLL should be able to compensate for it, lest the lock will be lost and receiver may have to initiate acquisition all over again. Thus the lock range should be decided based on stability of LO used.
- Dependence on input power: The SNR and hence signal input power keeps changing. If the discriminator function is dependent on the input signal power, the error signal may vary as input power varies and it is undesirable. Except in the case where signal is one-bit sampled, the receiver may need an Automatic Gain Control (AGC) in the analog front-end to maintain input power,else the PLL discriminator should measure phase difference insensitive to signal powers.

The detailed analysis of the PLL used in the implementation is given in Chapter 3.

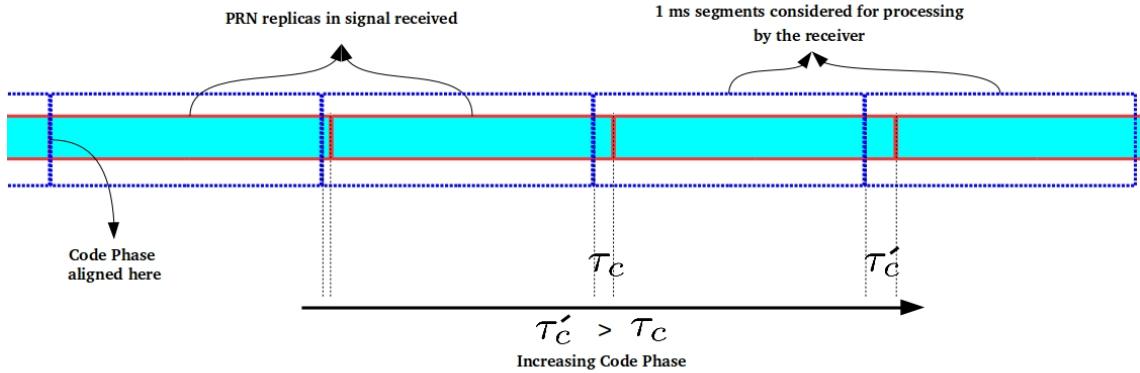


Figure 2.15: Exaggerated illustration of changing code-phase due to Doppler. Doppler shift in this case is negative. The incoming code is wider than 1 ms.

DLL: Keep the code aligned

As noted before, it is necessary that the locally generated code $x(t - \hat{\tau}_c)$ is always aligned to the PRN codes in incoming signal $x(t - \tau_c)$ to remove the codes in the signal and let the PLL lock. A **Delay Lock Loop (DLL)** performs this function, in particular:

- Acquisition gives a coarse estimate $\tilde{\tau}_c$ of the code-phase. The maximum error in this estimate can be as high as $\frac{1}{2 \times 10^{23}}$ ms from Equation 2.6, which implies a pseudo-range error of about $\frac{3 \times 10^8}{2 \times 10^{23}} \times 10^{-3} \approx 150$ meters. The DLL refines the estimate from acquisition to get exact code alignment.
- So far the effect of Doppler shift on PRN codes was ignored, but now we see how it necessitates a continuously running DLL. If the satellites and receiver were stationary with respect to one another, the receiver generated code once aligned would remain aligned with incoming code (assuming the receiver code generator clock itself does not drift). With no change in code phases the pseudoranges would all remain constant, which is as expected for stationary satellites. When there is relative motion there is Doppler shift which is simply a manifestation of changing distance between satellite and the receiver. If the satellite is moving away every next chip transmitted must travel a larger distance to reach the receiver. Thus the PRN code, 1 ms long when transmitted, will be stretched to more than 1 ms on reception. Refer Figure 2.15. The receiver on the other hand is generating the PRN sequence in 1 ms intervals. So it will see the code-phase increasing over time (in Figure 2.15, τ_c and τ'_c are code-phases of subsequent T_{int} segments. And $\tau_c > \tau'_c$). This in turn will increase the computed pseudoranges with time, which is as it should be for a satellite moving away. Similarly, for a satellite moving towards receiver, code-phase decreases with time. A DLL is required to track these changes.

The rate of change of code-phase due to Doppler is typically less than 4 chips per second, as computed before. DLL works using the smooth variation in the autocorrelation of the PRN

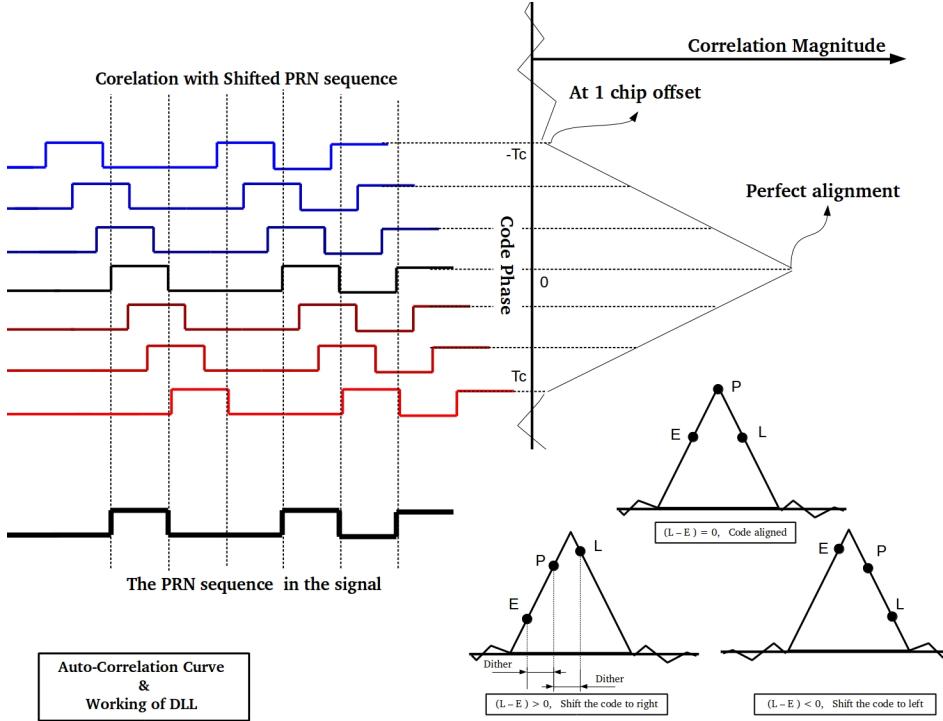


Figure 2.16: Construction of the correlation curve. Working of DLL using correlation curve is shown in lower right corner. E, P, L stand for Early, Prompt and Late correlation values respectively.

sequence $x(t)$ in under one-chip shifts about perfect alignment i.e

$$R(l) = \int_0^{1 \text{ ms}} x(t)x(t+l)dt$$

in the range $l \in (-T_c, T_c)$. Note that $x(t+l)$ is actually a circular shift, i.e $x((t+l)\bmod 1 \text{ ms})$. Figure 2.16 shows plot of $R(l)$ in region of interest. Acquisition gives code-phase within one-chip of the aligning code-phase. So by shifting the locally generate code by $\tilde{\tau}_c$ its correlation with the code in incoming signal is somewhere on the correlation peak. If it lies on the rising edge of the curve, receiver needs to shift its code slightly to the right to move towards perfect alignment. On the falling edge, it needs to move left. To identify the edge it is on w.r.t incoming signal $x(t - \tau)$ and the magnitude by which it should shift the local code, DLL does the following: Choose a dither value $\Delta_c \leq \frac{T_c}{2}$.

Step 1: Compute correlations $R_E = \int_0^1 x(t - \tilde{\tau}_c - \Delta_c)x(t - \tau)dt$ and $R_L = \int_0^1 x(t - \tilde{\tau}_c + \Delta_c)x(t - \tau)dt$.

They are called the Early and Late correlations respectively. ($R_P = \int_0^1 x(t - \tilde{\tau}_c)x(t - \tau)dt$ is the Prompt correlation)

Step 2: Find the difference $\delta_R = |R_L| - |R_E|$.

Step 3: Update code-phase $\tilde{\tau}_c = \tilde{\tau}_c - K_c \delta_R$. Go to Step 1

K_{DLL} is the feedback constant and should be chosen such that the update $K_D \delta_R$ is never more than $\frac{T_c}{2}$ for two reasons. Firstly, it is clear that the DLL will work as long as it is on the correlation curve or within d ms from it i.e $|\tau - \tilde{\tau}_c| \leq (T_c + \Delta_c)$. So we limit the feedback to ensure the prompt point is not thrown out of the curve. Secondly, to avoid oscillation where the prompt point keeps jumping from one edge to the other. With appropriate choices of K_c and Δ_c , the DLL tracks the code-phase. More generally any function of R_L and R_E that indicates their difference can be chosen as feedback. In the implementation $\delta_R = |R_L|^2 - |R_E|^2$ is used. This is to eliminate the effect of carrier phase θ as in acquisition.

It should be noted at this point, that the Doppler shift in PRN sequence is exactly related to Doppler in carrier ($\frac{f_d * 1.023}{1575.42} = \frac{f_d}{1540}$). Thus ideally a single PLL should be enough to adjust for code-phase change due to Doppler as described above. DLL would be required only to correct the acquisition estimate and would be used only in the beginning of tracking till the PLL locks. If this works, there is no need to keep the DLL running at all times. This works well in most cases. But if there is an event where PLL unlocks, then the PLL may not lock back, as the code-phase lock would be lost. Thus it is best to keep a separate DLL to maintain code-phase. Besides this, the f_d that PLL locks to may not be the exact Doppler shift and may include the frequency offset in the local oscillator used for down-conversion.

Post-acquisition, the DLL begins tracking the code-phase and facilitates the PLL to lock. Both the loops simultaneously operate to output the code-phase τ and NAV bits D as long as both are locked. Four or more such tracking blocks have to be running in a receiver, to track each of the visible satellites and compute the user position.

2.3.4 Navigation Bit Processing

Once tracking loops for atleast 4 satellites are locked, 4 NAV bits data stream can be recovered. Following are the typical steps thereafter:

Step 1: Detect beginning of a subframe by searching for preamble in each of the NAV data streams

Step 2: Once detected, the first edge of the preamble of one of the satellites can be taken as reference (usually the one to occur the earliest) and the time delay to all other preamble edges w.r.t the reference can be measured in terms of difference in PRN replicas and code chips as discussed in 2.3.1.

Step 3: The time of travel of signal for the reference satellite is taken to be a reasonable but arbitrary value, like 65 ms and multiplied with $c = 299792458$ m/s, to get a base pseudorange of 19,486 km.

Step 4: All other pseudoranges are computed as the base value plus c multiplied by the preamble delays measured in Step 2.

Step 5: Recover the entire frame (30 seconds long) and get satellite positions and other correction parameters for each of the satellites.

Step 6: Compute the user location by solving the pseudorange equations.

Step 7: Once the preambles are detected, the tracked code-phases can be used to compute changes in pseudoranges and hence user location any time thereafter using Step 5 and 6.

So, in GPS receiver, given NAV data stream $D(t)$ of one satellite, first step is to find the preamble, 10001101, with each bit lasting 20ms. $D(t)$ is correlated with preamble to identify matches. However the preamble sequence may occur anywhere in the data and hence there are many matches. Some additional steps will sieve out the actual preamble. First of this is the parity check. Table 2.17 describes the encoding scheme used in GPS words. So, we can form a 32 bit sequence by taking the 29 bits following and 2 bits preceding the first bit of the preamble matching sequence. Then parity check can be done by computing $D_{25}, D_{26}, \dots, D_{30}$ as shown in the table and match them with the last 6 bits of the 32-bit sequence. If they match, the sequence has passed the parity check, and the 30-bit word is indeed the beginning of a subframe, with high probability. However very rarely some false preamble pass the parity test too. In this case one can test for the parity of next 30 bit word. Additionally some permanent bits like, bit numbers 23, 24 (always 1) and bit number 59, 60 (always 0) in every subframe can be used to check validity of detected words.

IRNSS however uses convolution codes with interleaving, 16-bit sync preamble, and 24 CRC parity bits all of which are different from GPS. So an appropriate decoder needs to be implemented to detect, extract and validate the frames. But the essential steps of pseudorange computation remains the same.

Conclusion

This chapter outlined all the basic concepts of GPS signal processing. The special signal structure of GPS/IRNSS signals was presented in detail. This was followed by a discussion on the salient features of the signal on reception. The goals of GPS signal processing were listed and procedures involved in fulfilling them namely, acquisition, tracking and pseudorange computation were discussed. How the signal structure enables measurement of time of travel was explained. Working of acquisition and tracking blocks was discussed in detail, along with design guidelines and intuitions behind why they work. Now we look at practical details of implementing a GPS receiver.



Table 20-XIV. Parity Encoding Equations

D_1	=	$d_1 \oplus D_{30}^*$
D_2	=	$d_2 \oplus D_{30}^*$
D_3	=	$d_3 \oplus D_{30}^*$
•		•
•		•
•		•
•		•
D_{24}	=	$d_{24} \oplus D_{30}^*$
D_{25}	=	$D_{29}^* \oplus d_1 \oplus d_2 \oplus d_3 \oplus d_5 \oplus d_6 \oplus d_{10} \oplus d_{11} \oplus d_{12} \oplus d_{13} \oplus d_{14} \oplus d_{17} \oplus d_{18} \oplus d_{20} \oplus d_{23}$
D_{26}	=	$D_{30}^* \oplus d_2 \oplus d_3 \oplus d_4 \oplus d_6 \oplus d_7 \oplus d_{11} \oplus d_{12} \oplus d_{13} \oplus d_{14} \oplus d_{15} \oplus d_{18} \oplus d_{19} \oplus d_{21} \oplus d_{24}$
D_{27}	=	$D_{29}^* \oplus d_1 \oplus d_3 \oplus d_4 \oplus d_5 \oplus d_7 \oplus d_8 \oplus d_{12} \oplus d_{13} \oplus d_{14} \oplus d_{15} \oplus d_{16} \oplus d_{19} \oplus d_{20} \oplus d_{22}$
D_{28}	=	$D_{30}^* \oplus d_2 \oplus d_4 \oplus d_5 \oplus d_6 \oplus d_8 \oplus d_9 \oplus d_{13} \oplus d_{14} \oplus d_{15} \oplus d_{16} \oplus d_{17} \oplus d_{20} \oplus d_{21} \oplus d_{23}$
D_{29}	=	$D_{29}^* \oplus d_1 \oplus d_3 \oplus d_5 \oplus d_6 \oplus d_7 \oplus d_9 \oplus d_{10} \oplus d_{14} \oplus d_{15} \oplus d_{16} \oplus d_{17} \oplus d_{18} \oplus d_{21} \oplus d_{22} \oplus d_{24}$
D_{30}	=	$D_{29}^* \oplus d_3 \oplus d_5 \oplus d_6 \oplus d_8 \oplus d_9 \oplus d_{10} \oplus d_{11} \oplus d_{13} \oplus d_{15} \oplus d_{19} \oplus d_{22} \oplus d_{23} \oplus d_{24}$

Where

d_1, d_2, \dots, d_{24} are the source data bits;

the symbol $*$ is used to identify the last 2 bits of the previous word of the subframe;

$D_{25}, D_{26}, \dots, D_{30}$ are the computed parity bits;

$D_1, D_2, \dots, D_{29}, D_{30}$ are the bits transmitted by the SV;

\oplus is the "Modulo-2" or "Exclusive-Or" operation.

Figure 2.17: Parity encoding in GPS signal. Reproduced from [10]

Chapter 3

Design & Analysis of the Receiver

Having introduced the signal processing blocks involved in reception of GPS signals, we now discuss in detail the design parameters of these blocks, their interdependences and rationale behind choices made in the software receiver implementation presented in Chapter 4. Section 1 and Section 2 deal with Acquisition and Tracking parameters respectively. Section 3 describes the pseudorange computation algorithm used in the software implementation.

3.1 Acquisition Block Design

As discussed in Section 2.3.2 acquisition block has three main parameters namely, the coherent integration time, T_{int} , frequency search bin width Δf_d and code phase search bin width $\Delta \tau_c$. There is a fourth parameter, the detection threshold V_{th} . Usually, if the acquisition peak is greater than V_{th} the receiver declares the satellite found, else it either does a finer search or aborts search for the satellite and moves on to the next satellite.

Number of correlations to be performed in acquisition is inversely proportional to Δf_d and $\Delta \tau_c$. Also, computational cost increases proportional to T_{int} . Thus the aim is to find reasonable values of each of these parameters, under constraints on detection probabilities. Exact detection probabilities (detection, false positive, false negative probabilities) are not computed here and leads can be found in [2].

3.1.1 Integration Time, T_{int} & Doppler Search Bin-Width, Δf_d

With f_d as the Doppler shift, the carrier in received signal will be $\cos(2\pi(f_{IF} + f_d)t + \theta)$. To analyze, view each correlation computation in acquisition as sequence of three steps. First, multiplication by the received carrier with locally generated carrier of estimated Doppler f_d , next multiplication by shifted PRN code, and finally the integration for duration of T_{int} to obtain the correlation. In the first step, receiver multiplies $\cos(2\pi(f_{IF} + f_d)t + \theta)$ with $\cos(2\pi(f_{IF} + \hat{f}_d)t)$ and $\sin(2\pi(f_{IF} + \hat{f}_d)t)$, then the low frequency component in the resulting signal will be $\cos(2\pi(f_d - \hat{f}_d)t + \theta)$ and $-\sin(2\pi(f_d - \hat{f}_d)t + \theta)$ (we ignore the high frequency component of $(2f_{IF} + f_d + \hat{f}_d)$, since it will vanish anyway in subsequent integration). Thus a residual carrier

of $(f_d - \hat{f}_d)$ Hz will remain in the two signals, before we correlate with the PRN code to wipe-off the code. Refer to Figure 3.1. If $(f_d - \hat{f}_d)$ is small, no matter which 1ms segment we take, the PRN code will be retained (or fully inverted) in atleast one of cos and sin signals. So when PRN code is correlated with such resulting signals, atleast one of \Re_I and \Re_Q will be large resulting in a large value for \Re . However if $(f_d - \hat{f}_d)$ is large, the entire code may not be retained in either of the signals for some segments. In such cases some parts of underlying PRN codes is inverted with respect to other parts and resulting in lower values even on correlating it with PRN code of correct code phase. It is easy to see from the Figure, that to retain the code, we need residual $(f_d - \hat{f}_d) \leq 250$ Hz for $T_{int} = 1\text{ms}$. More generally we need $(f_d - \hat{f}_d) < \frac{1}{(4T_{int})}$. If we search with frequency bin width of Δf_d , even the nearest estimate may leave a residual of atmost $\frac{\Delta f_d}{2}$. For instance, let $f_d = 1740\text{Hz}$. If we take $\Delta f_d = 500\text{Hz}$, we will search at 0, 500, 100, 1500, and so on. The nearest estimate of $\hat{f}_d = 1500\text{Hz}$ will leave a residual of 240Hz. Thus for good correlation output, we need,

$$\begin{aligned}(f_d - \hat{f}_d) &\leq \frac{\Delta f_d}{2} \leq \frac{1}{(4T_{int})} \\ \Rightarrow \Delta f_d &\leq \frac{1}{(2T_{int})}\end{aligned}$$

For $T_{int} = 1\text{ms}$, we have $\Delta f_d \leq 500\text{Hz}$, so we take it to be 500Hz in software implementation. Partial correlation is not the only reason to limit value of Δf_d . After code-wipeoff and integration, the final value of correlation is given by Equation 2.5. The value is a sinc function in $(f_d - \hat{f}_d)T_{int}$, which highlights the same dependence between f_d and T_{int} as discussed above. Limiting this loss to a factor of 0.9032, for instance, gives $\Delta f_d = 500\text{Hz}$ [16]. The empirical rule usually considered in GNSS literature is $\Delta f_d \leq \frac{2}{3T_{int}}$, which is 667 Hz for $T_{int} = 1\text{ms}$, even though it causes some correlation loss due to code inversion [3]. One more consideration in choosing Δf_d is the capture range of PLL in tracking block. As discussed in Section 2.3.3, we need capture range $\geq \frac{\Delta f_d}{2}$. Else PLL may not lock or lock to the wrong frequency as discussed in next section.

It is clear that increasing T_{int} , increases the search space by that many times. On the other hand, integrating over larger T_{int} will reduce noise variance and give better correlation peaks. But beyond a point, increasing T_{int} increase the chance of having a NAV bit boundary in the T_{int} segment, resulting in cancellation of correlations on either side of the boundary and low net correlation. Thus in acquisition, T_{int} is taken as either 1ms or 2ms.

3.1.2 Code-Phase Search Bin-Width, $\Delta\tau_c$

The code phase spacing $\Delta\tau_c$ should be such that correlation with at least one code phase should fall on the correlation curve of the PRN code (Refer Figure 2.16). It is clear that if $\Delta\tau_c < 2T_c$, this is achieved. If the SNR is good, even a slight alignment of the codes at the edges of the correlation curve may result in distinguishable peak. However, if not, it increases the chance of missing the signal like in Figure 3.3 (marked in red). So we take $\Delta\tau_c \approx T_c$. Also, if the

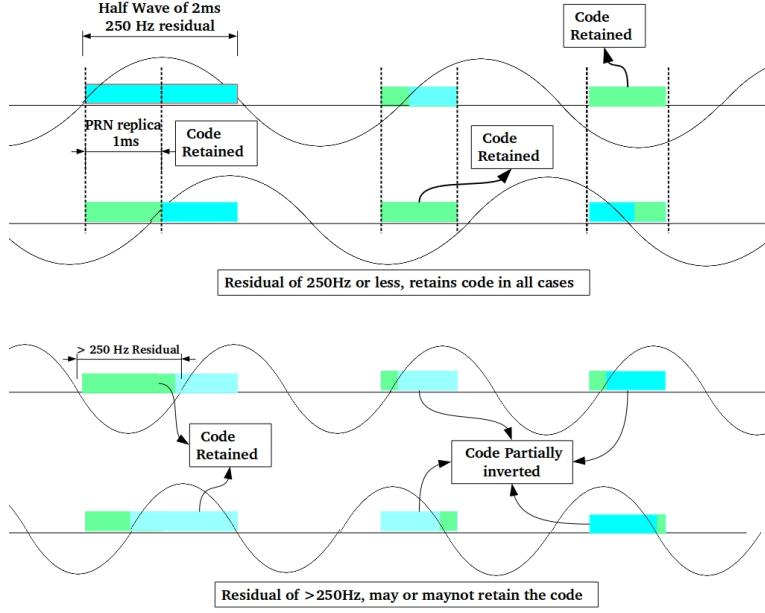


Figure 3.1: Code is retained in at least one component if $(f_d - \hat{f}_d) < \frac{1}{(4T_{int})}$. Here $T_{int} = 1\text{ms}$

sampling rate is F_s we can change code-phase only by shifting the samples i.e $\Delta\tau_c$ is a multiple of $\frac{1}{F_s}$. This gives $\Delta\tau_c = \frac{1}{F_s} \lceil \frac{T_c}{F_s} \rceil$, where $\lceil . \rceil$ is the ceiling function. With $F_s = 4\text{MHz}$, $\Delta\tau_c = 4$.

3.1.3 Threshold V_{th} and detection decision

Once the correlation over the entire search space is done, the receiver needs to find the peak and decide if the peak value is large enough to decide detection. There are many works on coherent detection problem. Refer to [4], [2]. A simple method is to have a threshold V_{th} for the peak value. A good choice would be $V_{th} \approx 3\sigma_{\eta^2}$ where σ_{η^2} is the variance of noise term η in correlation output. Note that, this however will make the threshold dependent on input signal power. So unless there is AGC (automatic gain control) in the front-end, this method requires

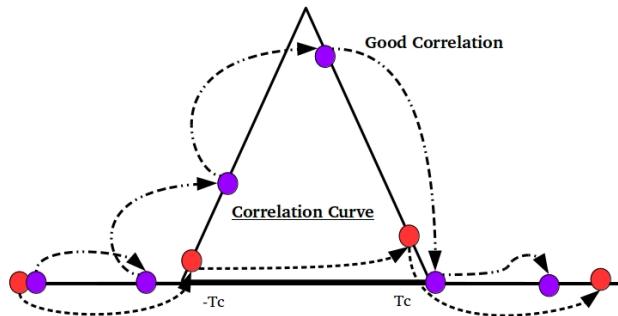


Figure 3.2: Bin width $< T_c$ gives atleast one good correlation (marked in purple)

dynamically changing the parameter V_{th} .

In the software implementation, if the satellite is not detected in the current T_{int} long segment, the receiver takes the next segment for acquisition. It does so $N_{attempt}$ times, which is currently arbitrarily chosen to be 10. At each attempt threshold is checked. Sometimes the peaks do not cross the given threshold, but the presence of signal is clear from repeated peaks in the same bin in successive segments. Thus in each attempt, the receiver checks for repeat peaks (Refer to the function ACQUISITION in Chapter 4). If either threshold is crossed or a repeated peak is found, the receiver declares acquisition successful, stops further attempts and proceeds to tracking. In many commercial receivers, coherent integration is followed with non-coherent averaging i.e the correlation for each bin, with successive segments is averaged and then threshold is checked. This is used to detect signals at very low SNR, but it requires remembering the entire search space result for previous segment unlike checking for repeated peaks. Much better methods for acquisition exist and some are discussed in [3]. What has been described here is the method ‘Serial Search’ Acquisition. FFT based parallel methods are much faster, but have significant hardware cost. These methods are best when a DSP is to be used for acquisition [16].

3.2 Tracking Block Design

We now look at parameters involved in the DLL and PLL comprising the tracking block.

3.2.1 DLL

DLL uses the early late correlator to correct for the code-phase. In practical receiver, early-late feedback is used to adjust the speed of generation of the local PRN code, which in turn adjusts code-phase. This has the advantage that the Doppler effect on the code also gets considered since the time duration of local PRN code is stretched or shrunk according to whether actual code-phase is leading or lagging the local code-phase due to Doppler. In the software receiver described here however, the PRN code is always 1ms long and only code phase is adjusted with feedback constant K_c . Thus the DLL has two parameters, K_c and the dither Δ_c . As discussed in Section 2.3.3, we need dither $\Delta_c \approx \frac{T_c}{2}$. Accordingly, take $\Delta_c = \frac{\Delta\tau_c}{2}$. With $F_s = 4\text{MHz}$, $\Delta_c = 2$. Choice of K_c is based on ensuring that the DLL remains on the correlation curve. The largest possible early-late feedback is the height of correlation curve. This happens when one of early or late correlation perfectly aligns. The feedback constant K_c should be chosen such that even in that case, the code-phase moves by half-chip. So if V_p is the peak correlation, $K_c \leq \frac{T_c}{2V_p}$. Though the peak value from acquisition will be less than, it can be taken as an estimate for V_p and K_c can be accordingly set in the DLL. However, note that this makes K_c depend on the input power-level. In practice K_c can be taken much smaller than $\frac{T_c}{2V_p}$ because the code-phase, once aligned, shifts at a very slow rate of about 1 chip in 250 ms and requires only slow-tracking. So, K_c can be kept fixed at a small value, for a given front-end. Also, the code-phase output of DLL is used in pseudorange computation. A larger K_c increases effect of noise on code-phase and thus on the pseudorange computed.

3.2.2 PLL

Continuing from Section 2.3.3, we design and analyze a PLL for GPS reception. Let the incoming signal to PLL be $r(t) = \cos(2\pi(f_{IF} + f_d)t + \theta)$. The PLL has an estimate $\{\hat{f}_d, \hat{\theta}\}$, which should eventually converge to $\{f_d, \theta\}$. First step is to reliably estimate the errors $(f_d - \hat{f}_d)$ and $\theta - \hat{\theta}$. So an appropriate discriminator must be used. So the receiver does the following:

Generate $v(t) = 2\cos(2\pi(f_{IF} + \hat{f}_d)t + \hat{\theta})$ and $u(t) = -2\sin(2\pi(f_{IF} + \hat{f}_d)t + \hat{\theta})$ and multiply with incoming signal. We get,

$$\begin{aligned} r(t)v(t) &= \cos(2\pi(f_d - \hat{f}_d)t + \theta - \hat{\theta}) + \text{high frequency component} \\ r(t)u(t) &= \sin(2\pi(f_d - \hat{f}_d)t + \theta - \hat{\theta}) + \text{high frequency component} \end{aligned}$$

Let $\Delta\omega = 2\pi(f_d - \hat{f}_d)$ and $\Delta\theta = \theta - \hat{\theta}$. To remove the high frequency component and reduce the noise variance, integrate the outputs from current time T_1 to time $T_1 + T$,

$$\begin{aligned} I &= \frac{1}{T} \int_{T_1}^{T_1+T} \cos(\Delta\omega t + \Delta\theta) dt \\ &= \frac{1}{T} \frac{2}{\Delta\omega} \sin\left(\frac{\Delta\omega T}{2}\right) \cos\left(\Delta\omega \frac{(2T_1 + T)}{2} + \Delta\theta\right) \\ Q &= \frac{1}{T} \int_{T_1}^{T_1+T} \sin(\Delta\omega t + \Delta\theta) dt \\ &= \frac{1}{T} \frac{2}{\Delta\omega} \sin\left(\frac{\Delta\omega T}{2}\right) \sin\left(\Delta\omega \frac{(2T_1 + T)}{2} + \Delta\theta\right) \end{aligned}$$

Now to extract the phase-difference, we use the \tan^{-1} discriminator,

$$e_r = \tan^{-1}\left(\frac{Q}{I}\right) = \text{sign}\left(\frac{Q}{I}\right) \left(\left| \Delta\omega \frac{(2T_1 + T)}{2} + \Delta\theta \right| \mod \frac{\pi}{2} \right)$$

If $T_1 = 0$, we get the error term $e_r = \frac{(\Delta\omega T)}{2} + \Delta\theta$. We see that if $\Delta\omega$ were 0, the signals would be locked with one update of $\hat{\theta} = \hat{\theta} + e_r$. Similarly, if $\Delta\theta = 0$, $\hat{f}_d = \hat{f}_d + \frac{1}{\pi T} e_r$ would be enough. When both are non-zero, suppose we use the update rules,

$$\begin{aligned} \hat{\theta} &= \hat{\theta} + K_1 e_r \\ \hat{f}_d &= \hat{f}_d + K_2 e_r \end{aligned}$$

We need to find suitable values of K_1 and K_2 , such that the error converges to zero eventually. We will model this as a two state digital system, with states $\omega[n]$ and $\theta[n]$, assuming that the Doppler frequency remains relatively constant in intervals of time T , and the estimates $\hat{\omega}[n]$ and $\hat{\theta}[n]$. T_1 can be always be taken 0, by updating $\theta[n]$ and $\hat{\theta}[n]$, with phase accumulated in

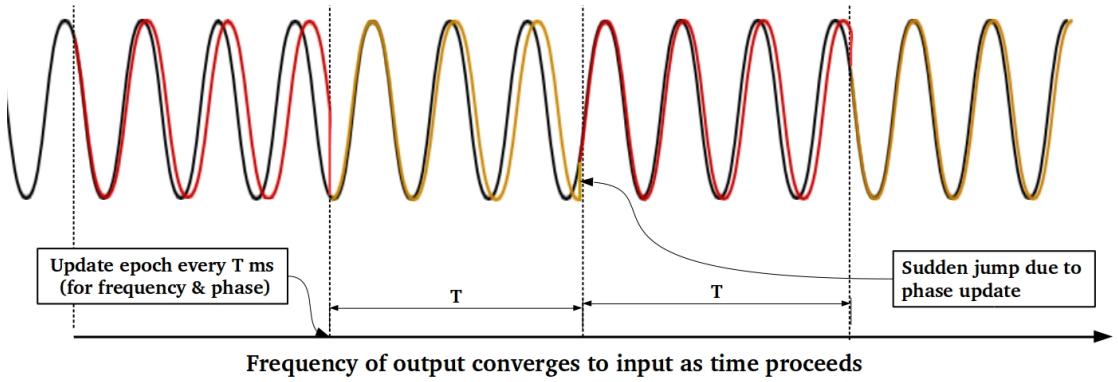


Figure 3.3: PLL Loop Updates Illustration. The colored waveform eventually locks with the input waveform. Both frequency and phase are updated every epoch

time T . It is then easy to verify,

$$e_r[n] = \frac{(\omega[n] - \hat{\omega}[n])}{2}T + \theta[n] - \hat{\theta}[n] \quad (3.1)$$

$$\theta[n] = \theta[n-1] + \omega[n-1]T \quad (3.2)$$

$$\hat{\theta}[n] = \hat{\theta}[n-1] + \hat{\omega}[n-1]T + K_1 e_r[n-1] \quad (3.3)$$

$$\hat{\omega}[n] = \hat{\omega}[n-1] + K_2 e_r[n-1] \quad (3.4)$$

Let $E_r(z), \Theta(z), \hat{\Theta}(z), \Omega(z), \hat{\Omega}(z)$ be the respective z transforms. Taking z transform and substituting, we get,

$$E_r(z) = \frac{(z^2 - 1)}{2z^2 + (2K_1 + L - 4)z + (L + 2 - 2K_1)} \Omega(z) T \quad (3.5)$$

where $L = K_2 T$. Let $H(z) = 2z^2 + (2K_1 + L - 4)z + (L + 2 - 2K_1)$. Then,

$$\hat{\Omega}(z) = \frac{L(z+1)}{H(z)} \Omega(z) \quad (3.6)$$

$$\hat{\Theta}(z) = \frac{(z+1)(L - K_1 + K_1 z)}{H(z)} \Theta(z) \quad (3.7)$$

The poles of this second order system are roots of $H(z)$, which are,

$$\frac{-(2K_1 + L - 4) \pm \sqrt{(2K_1 + L - 4)^2 - 8(L - 2K_1 + 2)}}{4}$$

For a choice of K_1, K_2 and T , as long as these roots lie within the unit circle, the system will be BIBO (bounded-input bounded-output) stable. The PLL will track the Doppler frequency and lock to the input signal. Figure 3.4 shows PLL outputs for a good choice of parameters (actually

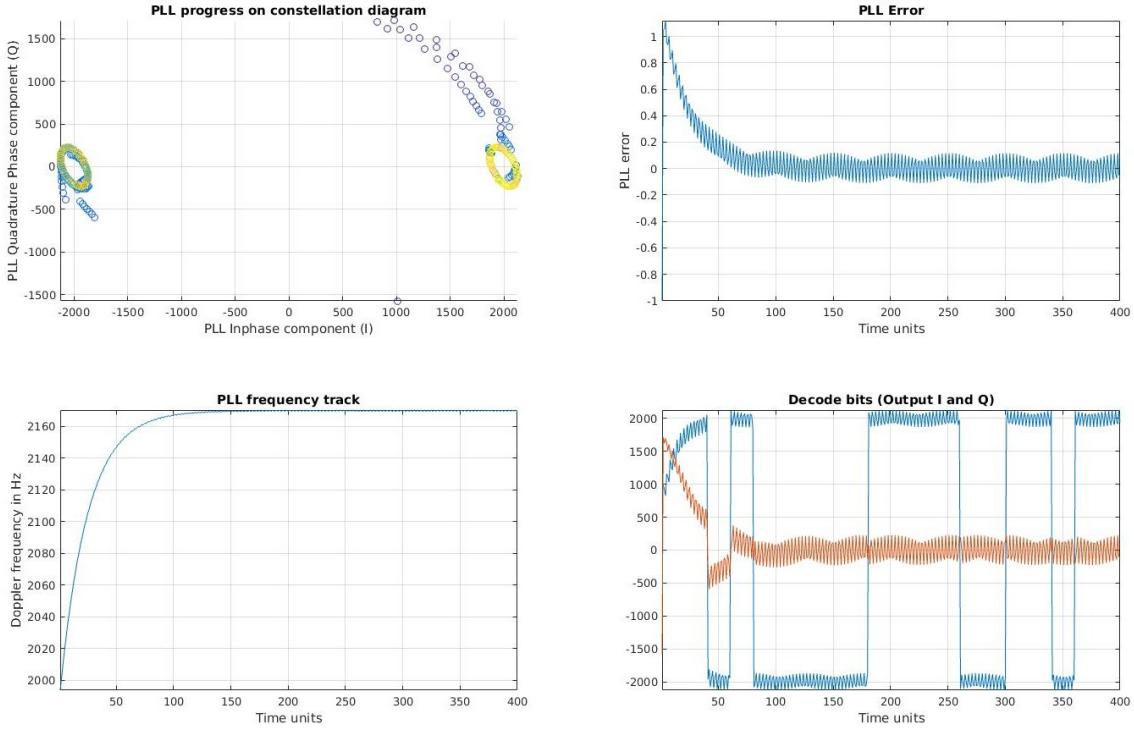


Figure 3.4: PLL outputs for $K_1 = 1$ and $K_2 = 40$. PLL is initialized to 2000 Hz, while input is at 2170 Hz. (Simulation)

the output may not be exactly locked to input, since pole-zero cancellations may happen. Even in the Figure we see the error having small oscillations and constellation is not a single point. However for all practical purposes, this does not matter much). In the implementation, K_1 is taken as 1, since that is the quickest way to correct for phase when $\Delta\omega \approx 0$ as discussed above. K_2 is taken as 40 i.e \hat{f}_d feedback coefficient of $\frac{40}{2\pi}$ after it was experimentally found to be a good trade-off between speed of the loop and its sensitivity to noise.

We now make some observations regarding the \tan^{-1} discriminator function used.

- Discriminator output is independent of input signal power. Hence it does not require an AGC in the front-end to function.
- Discriminator output is $\left(\Delta\omega \frac{T}{2} + \Delta\theta\right) \bmod \frac{\pi}{2}$. So it is immune to 180° phase shifts
- But on the other hand, the discriminator output is the same for $\Delta\omega + \frac{\pi}{T}k$ for any integer k . That means, the PLL can not distinguish between f_d and $f_d + \frac{1}{2T}$. Thus capture range of the PLL is limited $\pm \frac{1}{4T}$. In GPS receiver we need $T \leq T_{int}$ as discussed in acquisition block design. In GPS receivers generally T is taken to be 1ms and then gradually increased as PLL locks. With $T = 1\text{ms}$, the capture range is $\pm 250 \text{ Hz}$. So in acquisition with $\Delta f_d = 500\text{Hz}$, if the actual f_d is near midpoint of two frequencies searched, like 2240Hz, then noise can make acquisition block conclude the estimate as 2500Hz rather than the nearer 2000Hz. In this case the PLL will lock, but to $(2240 + 500)\text{Hz}$. This needs to be

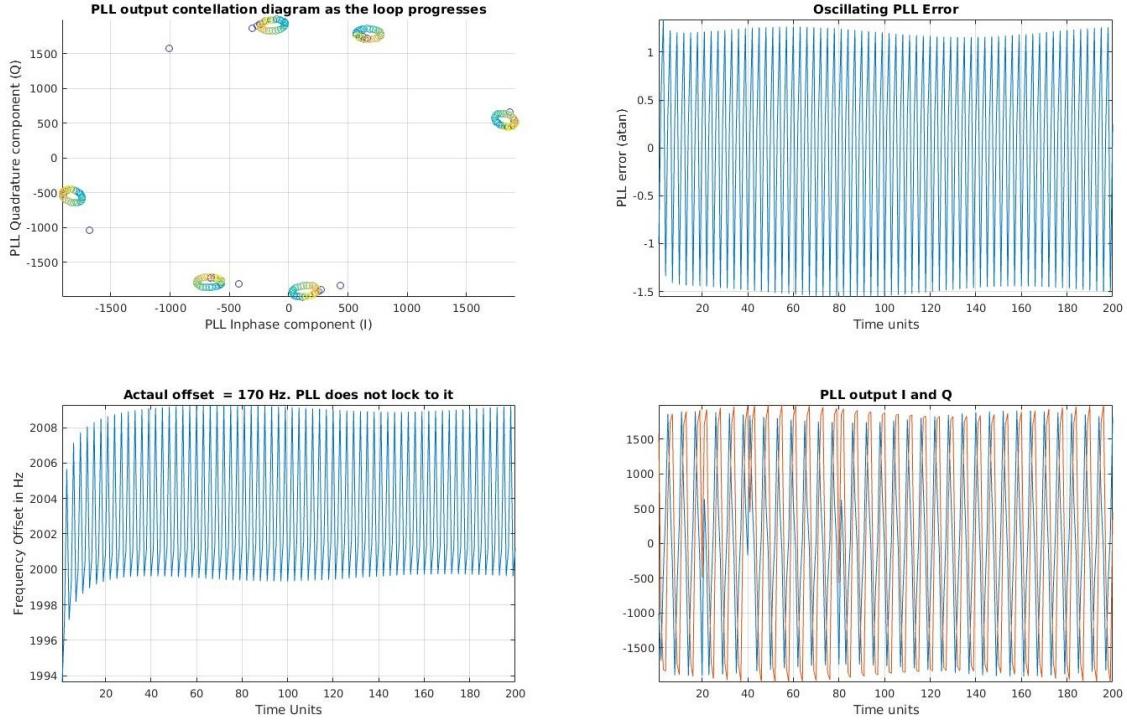


Figure 3.5: PLL outputs for $K_1 = 0.5$ and $K_2 = 40$. PLL is initialized to 2000 Hz, while input is at 2170 Hz. It does not lock. (Simulation)

separately taken care of.

- Since the \tan^{-1} discriminator outputs $\left(\frac{(\Delta\omega T)}{2} + \Delta\theta\right) \bmod \frac{\pm\pi}{2}$ rather than just $\left(\frac{(\Delta\omega T)}{2} + \Delta\theta\right)$, the wrapping of error around $\frac{\pm\pi}{2}$ may cause the PLL to oscillate. Thus it may never converge. Figure 3.5 shows an example. Whether or not the PLL gets into oscillations may also depend on the initial phase offset ($\theta[0] - \hat{\theta}[0]$). For instance, with $K_1 = 0.6$ and $K_2 = 200$ and initial frequency offset of 200 Hz, PLL locks if $(\theta[0] - \hat{\theta}[0]) = 0$, but does not if $(\theta[0] - \hat{\theta}[0]) = \frac{\pi}{3}$. However noise can actually help by disturbing the fine balance that is needed to maintain oscillations and pull the PLL out of it, as in Figure 3.6.

- The discriminators immunity to 180° phase shifts creates the problem of NAV bit flips. Consider the following example:

$$\theta = 0.4\pi, \hat{\theta} = 0 \text{ and } \omega = 510, \hat{\omega} = 500. \text{ Let } T = 1\text{ms}, K_1 = 1.$$

$$\text{Feedback} = \frac{2\pi(50T)}{2} + 0.4\pi = 0.45\pi$$

$$\therefore \hat{\theta} = 0 + 2\pi(500)T + 0.45\pi = 1.45\pi$$

$$\text{While } \theta = 0.4\pi + 2\pi(510)T = 1.5\pi.$$

Clearly, in the next few iterations the two signals will phase lock. Observe the feedback 0.45π . Values near to 0.5π result when I is small in $\tan^{-1}\left(\frac{Q}{I}\right)$. When I is small, a little noise can flip its sign and result in the feedback -0.45π instead of 0.45π . Then,
 $\hat{\theta} = 0 + 2\pi(500)T - 0.45\pi = 0.55\pi$

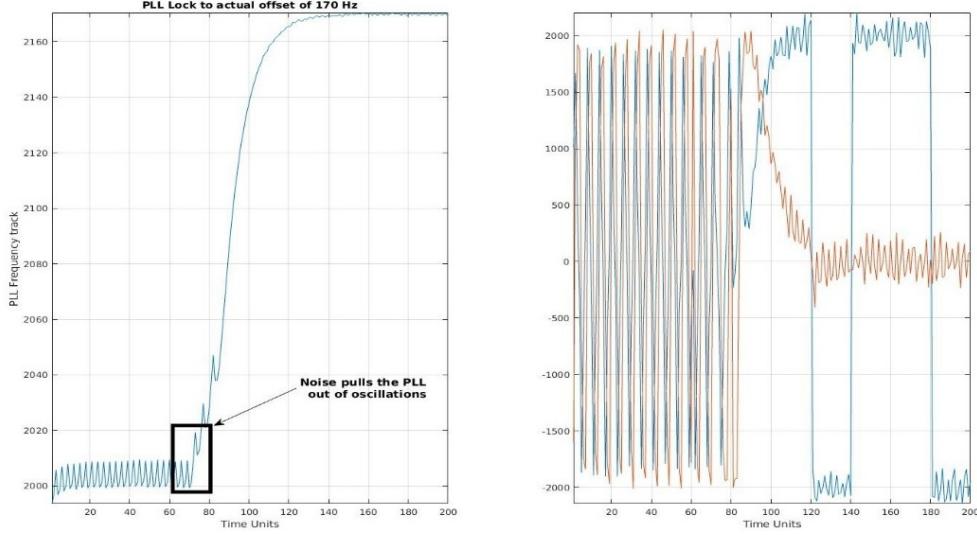


Figure 3.6: PLL outputs for $K_1 = 0.5$ and $K_2 = 40$ with 0 dB noise. PLL is pulled out of oscillations

$\theta - \hat{\theta} \approx \pi$. Now the PLL is moving towards settling with a phase difference of π . Thus a small amount of noise at the right instance can result in PLL flips. This is problematic since output sign of I is used to conclude the NAV data bits. As long as $\text{sign}(I)$ consistently gives D or $-D$, we can decode the frames. But if it randomly switches, it may lead to erroneous NAV data words. Possible solutions are:

- Use larger T to reduce the effect of noise. This can be employed once the PLL has locked. But increasing T involves additional computational costs.
- Cap the feedback. As discussed above, noise is more likely to affect when I and Q are such that they result in large feedback. So we cap the maximum absolute feedback value to say 0.25π . In this case, a switch in sign would have given -0.25π instead of 0.25π and caused a net error of 0.5π in the phase estimate, thus it can still converge back. However, even the cap will not help if two such events happen consecutively, resulting in a phase difference of π . The software implementation uses a cap at 0.25π .

We can make the bilinear transformation on the z -domain equations and derive the rise time. However it is sufficient to remember that rise time changes in inverse proportion of K_2 . This was experimentally found and is intuitively easy to see why this holds.

Measure of Lock

The receiver needs to know the extent to which the PLL is locked to decide if the $\text{sign}(I)$ output is valid for further processing. So we need a ‘lock detector’. We use the function, $\frac{(I^2-Q^2)}{(I^2+Q^2)}$ commonly used for the purpose. It is easy to see that it is actually $\cos(2 \tan^{-1}(\frac{Q}{I}))$. So as the error tends to zero, the lock measure tends to 1. However the function was found to be noisy with real GPS data. So, in the implementation, lock measure is low pass filtered with

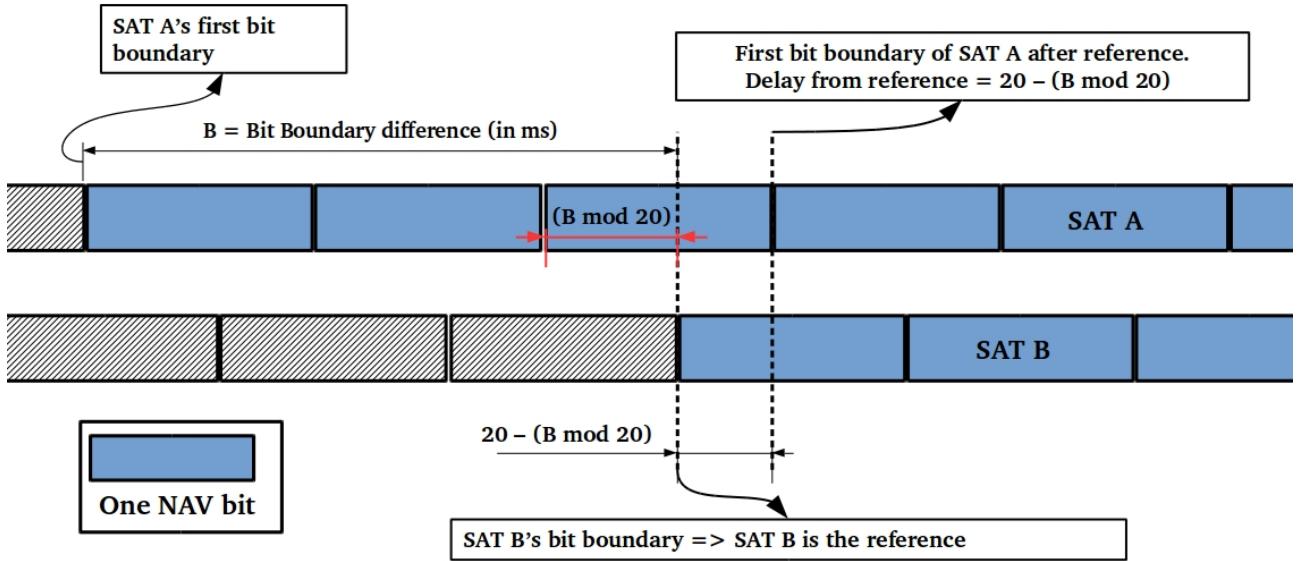


Figure 3.7: Detecting Bit Boundary and measuring relative time of arrivals

the IIR $\frac{1}{1-\alpha z^{-1}}$, which eventually settles to $\frac{1}{1-\alpha}$. With $\alpha = 0.9$, it settles to 10. So the receiver can keep a threshold of 5 on the lock measure to declare that the PLL is locked and detect when the PLL unlocks.

3.3 Computing pseudorange

As discussed before, to calculate the receiver position we need to compute the difference between pseudoranges of satellites. This can be obtained by measuring the difference in times of arrival of the signals, sent by the satellites at the same time (common transmission method, Section 2.3.1). So we measure time difference between arrival of the preamble from different satellites, as accurately as possible. The accuracy is limited to $\frac{1}{F_s}$, but may be improved by interpolating for times of arrival of signal between two samples. Here we describe how the software receiver measure this delay.

Every T_{int} , PLL outputs $\text{sign}(I)$. Suppose we collect these in to a data array $\{D[n]\}_{n=0}^{\infty}$, with elements indexed as $D[0], D[1], \dots$ and so on, $D[0]$ being the first output after PLL is switched on. There will be K such channels, one for each satellite. Let the data arrays be, D_i for $i = 1, 2, \dots, K$. First we need to wait until the PLL's lock and the output $\text{sign}(I)$ becomes valid. For the ease of explaining, we assume that the PLL once locked remain locked. After getting locked, the outputs of PLL gives the NAV data bits (or inverted ones), sampled T_{int} apart. Let us take $T_{int} = 1$ ms. So one NAV bit will have 20 samples in the array D . Once valid data begins coming through, the receiver waits for a NAV bit boundary, i.e a sign

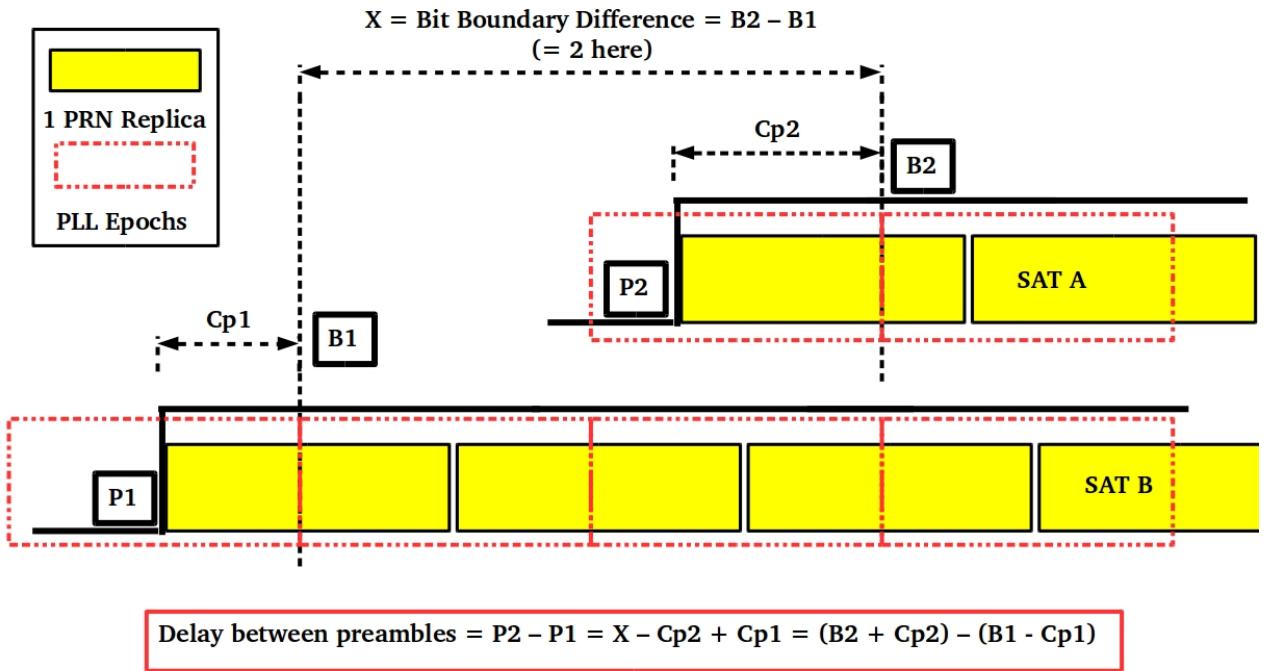


Figure 3.8: Measuring the delay between preamble arrivals using bit boundary difference and code-phase. The red dotted segments are used by the PLL in $\text{sign}(I)$ computation. Bit boundary difference ($B_2 - B_1$) gives difference between these epochs. To get the actual difference between preamble arrivals P_1 and P_2 , code-phase from DLL must be used along with $(B_2 - B_1)$

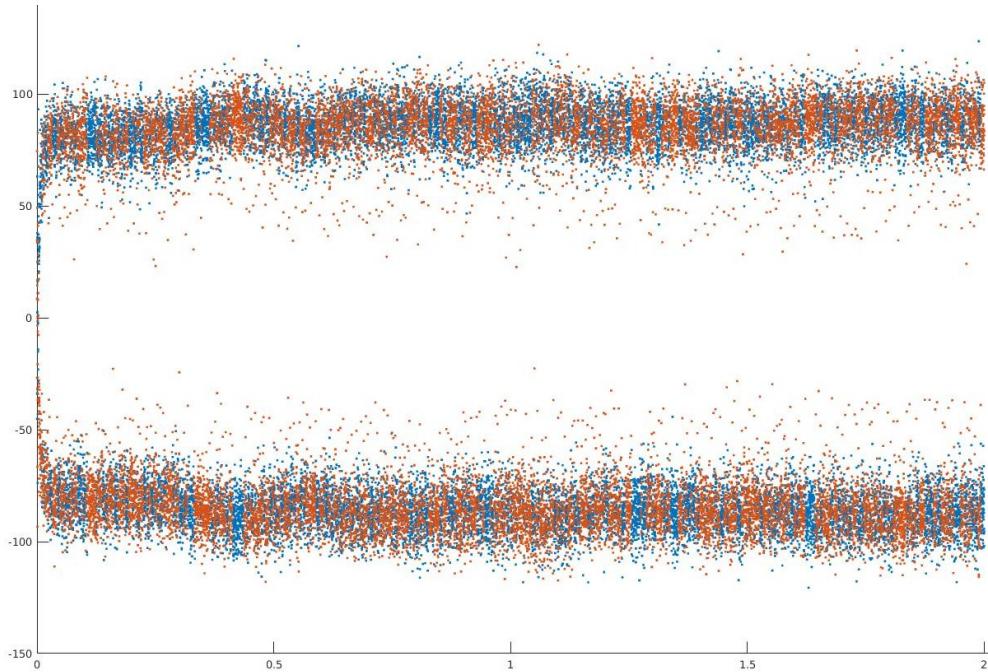


Figure 3.9: Result of aligning PLL T_{int} segments with incoming PRN sequence. Orange dots denote $\text{sign}(I)$ output without this alignment and blues ones. after alignment

change in the PLL output sign(I). Let us say the bit boundaries occur at indexes m_i in the data stream D_i for $i = 1, 2, \dots, K$. We choose the satellite with largest m_i , say m_{ref} as the reference satellite, say sv_{ref} . We know that all data streams D_i , definitely have valid data from index m_{ref} onwards (subject to our assumption that once locked, PLL remains locked). The relative offset of the NAV bits, which will be an integer between 0 and 20, can now be measured as shown in Figure 3.7. Recall that the signals arrive from different satellites with delays typically less than 15 ms, i.e within one NAV bit for GPS. So the starting edges of the preambles in D_i 's will be coarsely separated by bit boundary difference. But these differences have a resolution of 1ms. We need to use the code-phases at the preamble start bit boundaries to get finer delay measurement. Figure 3.8 illustrates how this can be done. Hence in each epoch, the PLL outputs the code-phase recorded by the DLL along with sign(I). Using the bit boundary difference and code-phases we can get delays in preamble arrivals with an accuracy of $\frac{1}{F_s}$ (because that is the accuracy of code-phase measurement). Note that in the software implementation (in the code, not included in the pseudocode), when the PLL starts, the PLL T_{int} -segments are aligned with the incoming PRN codes, by taking the segments with a delay of $\tilde{\tau}_c$ with respect to the segments taken for acquisition. Recall that $\tilde{\tau}_c$ is the code-phase estimate from acquisition. This gives sharp changes in sign(I) at NAV bit boundaries, as shown in Figure 3.9. Clearly, the outputs with alignment show better segregation into +1 and -1. However the alignment will eventually drift away due to Doppler and needs to be reset regularly.

Conclusion

This Chapter described the design parameters and methods used in the software implementation and presented the thought process behind the choices made. GPS reception being a widely studied subject, superior ways of doing the tasks described here can be found in literature. This report however outlines the most basic receiver and its characteristics, to act as a starting point in the long process of building a practical receiver for IRNSS.



Chapter 4

Software Implementation of GPS Receiver

This chapter describes the implementation of software receiver for acquisition, tracking and pseudo-range computation, based on discussions in the previous chapter. The pseudocode is documented in the Section 1. An implementation of the same in MATLAB/OCTAVE can be found in Appendix A. The Section 2 describes the experimental setup used for data collection and validation. Section 3 presents results and observations from the MATLAB implementation. It should be noted that the same codes for acquisition and tracking can be used for GPS as well as IRNSS, with the minor change of PRN codes used. In this report, 'software receiver' is used to refer to the specific implementation of the receiver described in this document unless otherwise specified.

4.1 Software Receiver pseudocode

We now define symbols and conventions used in the pseudocode description.

- $\{A[n]\}_{n=0}^{n=L-1}$ will denote array $A_{1 \times L}$ of length L , with its elements indexed from 0 to $L - 1$. The array can be initialized as $\{A[n]\}_{n=0}^{n=L-1} \leftarrow x$, which means all elements of A are set to scalar x .
- $A[k]$ will denote the $(k + 1)^{\text{th}}$ element of array A . Let $\{A\}_{k_1}^{k_2}$ denote the $k_2 - k_1 + 1$ size sub-array formed by $\{A[k_1], A[k_1 + 1], \dots, A[k_2]\}$ in same order.
- **Circular Left Shift:** For any integer N , array A_N , resulting from circularly N -left shifts of any finite n length array $A_{1 \times n}$ is defined as:

$$\begin{aligned} A_N &:= (A \ll N) \\ \implies A_N[i] &= A[(i + N) \bmod n] \quad \forall i \in [0, n - 1] \end{aligned}$$

Similarly circular right shift $A \gg N$ can be defined. It would simply be A_{-N} .

- **Dot Product:** Scalar value $D_{A,B,C}$ resulting from dot product of any finite n length arrays $A_{1 \times n}$, $B_{1 \times n}$ and $C_{1 \times n}$ is defined as:

$$D := A \odot B \odot C$$

$$\implies D_{A,B,C} = \sum_{i=0}^{n-1} A[i]B[i]C[i]$$

- **Shift Register:** For finite n length array $A_{1 \times n}$, define,

$$B_{1 \times n} := (x \Rightarrow A)$$

$$\implies B[0] = x \text{ and } B[i] = A[i-1] \quad \text{for } i = 1, \dots, n-1$$

- For a sequence $\{s[n]\}_{n=0}^{\infty}$ of samples, define the k^{th} L length **packet** as the array,

$$P_k^L(s) := \left[s[n] \right]_{n=(k-1)L}^{n=kL-1} \quad \text{for } k, L \in \mathbb{Z}^+$$

- For a real number x , $\text{sign}(x)$ will return the sign of x for a non-zero x , and 0 if $x = 0$
- For binary numbers x and y , $x \oplus y$ will denote the XOR operation

Also note, the parity check matrix used in PARITYCHECK $M_{6 \times 32}$ is [10] ,

1	0	1	1	1	0	1	1	0	0	0	1	1	1	1	0	0	1	1	0	1	0	0	0	0	0	0	0	
0	1	0	1	1	1	0	1	1	0	0	0	1	1	1	1	0	0	1	1	0	1	0	0	1	0	0	0	0
1	0	1	0	1	1	1	0	1	1	0	0	0	1	1	1	1	0	0	1	1	0	1	0	0	0	0	0	0
0	1	0	1	0	1	1	1	0	1	1	0	0	0	1	1	1	1	0	0	1	1	0	1	0	0	0	0	0
0	0	1	0	1	0	1	1	1	0	1	1	0	0	0	1	1	1	1	0	0	1	1	0	0	0	0	0	0
1	1	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0	0	0	1	1	1	0	0	0	0	0	0	0

SV	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19							
Tap 1	2	3	4	5	1	2	1	2	3	2	3	5	6	7	8	9	1	2	3							
Tap 2	6	7	8	9	9	10	8	9	10	3	4	6	7	8	9	10	4	5	6							
SV	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	-							
Tap1	4	5	6	1	4	5	6	7	8	1	2	3	4	5	4	1	2	4	-							
Tap2	7	8	9	3	6	7	8	9	10	6	7	8	9	10	10	7	8	10	-							

Table 4.1: Satellite-wise Tap Positions for C/A code generation in GPS (*TapPosition* array to be used in pseudocode CACODE)

Variable Definitions & Values		
Variable	Description	Value
f_I	Intermediate frequency to which the signal is down-converted by the front-end to be sampled	-
F_s	Sampling rate at which the ADC samples signal from front-end	4MHz
T_{int}	Time length of signal considered for code and carrier wipeoff in each iteration of acquisition or tracking	10^{-3} sec
T	Number of samples of signal considered for code and carrier wipeoff	$T_{int}F_s = 4000$
f_R	Acquisition procedure searches for Doppler shift in the range $[-f_R, f_R]$	6kHz
Δf_{bin}	Acquisition procedure searches for Doppler shift in intervals of Δf_{bin} Hz, also known as frequency bin width	$\frac{1}{(2T_{int})} = 500$ Hz
Δ_C	Acquisition procedure searches for code-shift in a given packet at intervals of Δ_C shifts	$\left\lfloor \frac{F_s}{1.023\text{MHz}} \right\rfloor = 4$
sv	Vehicle number of the satellite being searched for	1 to 32 for GPS, 1 to 7 for IRNSS
$N_{attempt}$	Number of packets acquisition procedure tests before declaring the satellite of given sv as ‘not found’	5 (arbitrary choice)
V_{th}	Threshold for the acquisition peak value, above which satellite is declared as ‘found’	3000*
Δ_D	Dither used in Early and Late correlations	$\frac{\Delta_C}{2} = 2$
θ_{cap}	The value at which the PLL feedback is capped to prevent NAV bit flips	$\frac{\pi}{4}$
α	1st Order IIR filter pole applied on PLL lock measure	0.9
K_c	Feedback Constant used in the DLL. Let V_{peak} be the peak correlation value from acquisition	$\frac{1}{(2 \times V_{peak})}$
K_1	Feedback Constant used in Phase update of the PLL	1
K_2	Feedback Constant used in angular frequency update of the PLL	40 ($\frac{20}{\pi}$ in Hz)

Table 4.2: Variable Definitions and Values

(*For full precision samples, using the 20dB GPS antenna and USRP setup with 60dB gain)

4.1.1 Baseband Processing

In all of the discussion so far and in the pseudocode description, it is assumed that incoming signal $\{s[n]\}_{n=0}^{\infty}$ is at $f_{IF} > 0$. However using a setup like the USRP as described below, will downconvert the signal directly to baseband, leading to complex signal samples or equivalently two streams of samples, the in-phase samples $\{s_I[n]\}_{n=0}^{\infty}$ and the quadrature samples $\{s_Q[n]\}_{n=0}^{\infty}$. Let P_I and P_Q represent the respective packets. Following changes must be made:

- In the correlation computation step (1) of ACQUIRE, $(P \odot I_{lo} \odot \hat{C})^2 + (P \odot Q_{lo} \odot \hat{C})^2$ should be replaced with $(P_I \odot I_{lo} \odot \hat{C} + P_Q \odot Q_{lo} \odot \hat{C})^2 + (P_I \odot Q_{lo} \odot \hat{C} - P_Q \odot I_{lo} \odot \hat{C})^2$. Same applies for steps (3) and (4) of TRACK
- In step (1) of TRACK, replace $(P \odot I_{lo} \odot \hat{C})$ with $(P_I \odot I_{lo} \odot \hat{C} + P_Q \odot Q_{lo} \odot \hat{C})$. Similarly in step (2), use $(P_I \odot Q_{lo} \odot \hat{C} - P_Q \odot I_{lo} \odot \hat{C})$

Function Names & Descriptions	
Function Name	Description
$v_p, f_p, c_p \leftarrow \text{ACQUIRE}(P, C)$	Given T samples long array P , and PRN code C of a satellite perform one iteration of acquisition. If satellite signal is found, return peak correlation v_p , acquired Doppler frequency f_p and code phase c_p . Else $v_p = 0$
$v_p, f_p, c_p \leftarrow \text{ACQUISITION}(\{s[n]\}_{n=0}^{\infty}, sv)$	For an input stream of signal samples $\{s[n]\}_{n=0}^{\infty}$ sampled at F_s , the function performs the ACQUIRE procedure $N_{attempt}$ number of times, as described in Chapter 3, for satellite sv
$\{D[n]\}_{n=0}^{\infty}, \{Z[n]\}_{n=0}^{\infty}, \text{lock} \leftarrow \text{TRACK}(\{s[n]\}_{n=0}^{\infty}, sv, f_{aq}, c_{aq})$	Given Doppler and Code-phase estimates f_{aq}, c_{aq} from acquisition, for satellite sv, the function tracks the signal to output data bits $\{D[n]\}_{n=0}^{\infty}$ and code-phase stream $\{Z[n]\}_{n=0}^{\infty}$, at the rate of one reading per T_{int} . It like one channel in a receiver. lock is a measure of extent of PLL lock. Empirically a threshold of $\frac{1}{2(1-\alpha)}$ can be set, above which PLL can be declared to be, locked.
$m \leftarrow \text{FINDBITBOUNDARY}(\{D[n]\}_{n=0}^{\infty})$	Given the readings $\{D[n]\}_{n=0}^{\infty}$ from tracking block, the function output the index of first NAV bit boundary, after the loops have locked.
$m \leftarrow \text{FINDPREAMBLE}(\{\text{NAV}[n]\}_{n=0}^{\infty})$	Given the NAV data bit stream $\{\text{NAV}[n]\}_{n=0}^{\infty}$ the function output the index of beginning of a preamble, after checking for parity
$L \leftarrow \text{PARITYCHECK}(\{A[n]\}_{n=0}^{31})$	Given a 32-bit GPS NAV bit packet $\{A[n]\}_{n=0}^{31}$, the function performs the parity check. Output $L = 1$ if the check is passed else 0
$y \leftarrow \text{CAP}(x, \theta)$	The function caps absolute value of input real x to θ
$\{C[n]\}_{n=0}^{T-1} \leftarrow \text{CACODE}(sv)$	For GPS satellite number sv, the function generates the C/A code and samples it at F_s . The resulting T length sequence is output to be used in acquisition and tracking
$\{\text{pseudoRange}[n]\}_{n=1}^K \leftarrow \text{MAIN}(\{s[n]\}_{n=0}^{\infty})$	For input stream of raw signal samples $\{s[n]\}_{n=0}^{\infty}$, the function performs acquisition, tracking and outputs pseudoranges computed for K detected satellites, at the preamble beginnings. Note that this assumes that a satellite signal once acquired, is tracked, & generates valid NAV bit stream

Table 4.3: Descriptions of functions used in the software implementation

Also note that having the signal in baseband, i.e $f_{IF} = 0$ will create problems at all places where we assumed the high frequency component to vanish on T_{int} integration. However the above procedure will ensure that this component is sample-wise removed, not relying on the integration to eliminate it.

4.1.2 pseudocode

A summary of all the functions defined can be found in Table 4.3.

Algorithm 4.1.1: CACODE(sv)

```

 $\{C[n]\}_{n=0}^{T-1} \leftarrow 0$ 
 $\{G1[n]\}_{n=1}^{10} \leftarrow 1$ 
 $\{G2[n]\}_{n=1}^{10} \leftarrow 1$ 
 $\{t_1, t_2\} \leftarrow TapPosition(sv)$ 
 $\delta \leftarrow \frac{1}{F_s}$ 
 $T_c \leftarrow \frac{1}{1023 \times 1000}$ 
 $k \leftarrow 0$ 
 $C[k] = G1[10] \oplus G2[t_1] \oplus G2[t_2]$ 
 $n_c \leftarrow 1$ 
while ( $n_c < 1024$ )
   $k \leftarrow k + 1$ 
  while ( $k\delta < n_c T_c$ )
    do
       $C[k] = C[k - 1]$ 
       $k \leftarrow k + 1$ 
  do
     $G1 \leftarrow [(G1[3] \oplus G1[10]) \Rightarrow G1]$ 
     $G2 \leftarrow [(G2[2] \oplus G2[3] \oplus G2[6] \oplus G2[8] \oplus G2[9] \oplus G2[10]) \Rightarrow G2]$ 
     $C[k] = G1[10] \oplus G2[t_1] \oplus G2[t_2]$ 
     $n_c \leftarrow n_c + 1$ 
   $C \leftarrow (2 \times C - 1)$ 
comment: Element-wise operation to convert array of {0, 1} to {-1, 1}
return ( $C$ )

```

Algorithm 4.1.2: CAP(x, θ)

comment: x is a real number, θ is a positive real number

```

if ( $|x| < \theta$ )
  then  $y = x$ 
  else  $y = \frac{x}{|x|} \theta$ 
return ( $y$ )

```

Algorithm 4.1.3: ACQUIRE(P, C)

procedure ACQUIRE(P, C)

$v_p \leftarrow 0$

$f_p \leftarrow 0$

$c_p \leftarrow 0$

$f \leftarrow f_I - f_R - \Delta f_{\text{bin}}$

while $f \leq (f_I + f_R)$

do {

$$\begin{cases} f \leftarrow f + \Delta f_{\text{bin}} \\ I_{lo} \leftarrow \left[\cos \left(\frac{2\pi f}{F_s} n \right) \right]_{n=0}^{n=T-1} \\ Q_{lo} \leftarrow \left[\sin \left(\frac{2\pi f}{F_s} n \right) \right]_{n=0}^{n=T-1} \\ \text{for } m \leftarrow 0 \text{ to } 1022 \\ \quad \text{do } \left\{ \begin{array}{l} \hat{C} \leftarrow (C \gg m\Delta_C) \\ v_{\text{corr}} \leftarrow (P \odot I_{lo} \odot \hat{C})^2 + (P \odot Q_{lo} \odot \hat{C})^2 \end{array} \right. \\ \quad \text{if } (v_{\text{corr}} > v_p) \\ \quad \quad \text{then } \left\{ \begin{array}{l} v_p \leftarrow v_{\text{corr}} \\ f_p \leftarrow f \\ c_p \leftarrow m\Delta_C \end{array} \right. \end{array} \right. \end{cases} \quad (1)$$

}

return (v_p, f_p, c_p)

Algorithm 4.1.4: ACQUISITION($\{s[n]\}_{n=0}^{\infty}, sv$)

$k \leftarrow 1$

$f_0 \leftarrow 0$

$c_0 \leftarrow 0$

$\text{found} \leftarrow 0$

$C_{sv} \leftarrow \text{CACODE}(sv)$

while ($\text{found} = 0$) **or** ($k < N_{attempt}$)

do {

$$\begin{cases} v, f_1, c_1 \leftarrow \text{ACQUIRE}(P_k^T(s), C_{sv}) \\ \text{if } (v > V_{th}) \text{ or } ((f_1 = f_0) \text{ and } (c_1 = c_0)) \\ \quad \text{then } \text{found} \leftarrow 1 \\ f_0 \leftarrow f_1 \\ c_0 \leftarrow c_1 \\ k \leftarrow k + 1 \end{cases}$$

}

return $(f_0, c_0, \text{found} \times v)$

Algorithm 4.1.5: TRACK($\{s[n]\}_{n=0}^{\infty}$, sv, f_{aq} , c_{aq})

```

 $c_p \leftarrow c_{aq}$ 
 $f_{lo} \leftarrow f_{aq}$ 
 $\phi_{lo} \leftarrow 0$ 
 $C_{sv} \leftarrow \text{CACODE(sv, } F_s)$ 
lock  $\leftarrow 0$ 
 $k \leftarrow 0$ 
while (1)
   $\hat{C} \leftarrow (C_{sv} \gg c_p)$ 
   $\hat{C}_e \leftarrow (\hat{C} \ll \Delta_D)$ 
   $\hat{C}_l \leftarrow (\hat{C} \gg \Delta_D)$ 
   $I_{lo} \leftarrow \left[ \cos\left(\frac{2\pi f_{lo}}{F_s} n + \phi_{lo}\right) \right]_{n=0}^{n=T-1}$ 
   $Q_{lo} \leftarrow \left[ \sin\left(\frac{2\pi f_{lo}}{F_s} n + \phi_{lo}\right) \right]_{n=0}^{n=T-1}$ 
   $i_p \leftarrow (P_k^T \odot I_{lo} \odot \hat{C}) \quad (1)$ 
   $q_p \leftarrow (P_k^T \odot Q_{lo} \odot \hat{C}) \quad (2)$ 
   $v_p \leftarrow (P_k^T \odot I_{lo} \odot \hat{C}_e)^2 + (P_k^T \odot Q_{lo} \odot \hat{C}_e)^2 \quad (3)$ 
   $v_l \leftarrow (P_k^T \odot I_{lo} \odot \hat{C}_l)^2 + (P_k^T \odot Q_{lo} \odot \hat{C}_l)^2 \quad (4)$ 
do {
   $d_{\text{DLL}} \leftarrow (v_l - v_e)$ 
   $d_{\text{PLL}} \leftarrow \text{CAP}(\tan^{-1}\left(\frac{q_p}{i_p}\right), \theta_{\text{cap}})$ 
   $c_p \leftarrow c_p + K_c d_{\text{DLL}}$ 
   $\phi_{lo} \leftarrow (\phi_{lo} + 2\pi T_{int} f_{lo} + K_1 d_{\text{PLL}}) \bmod 2\pi$ 
   $f_{lo} \leftarrow f_{lo} + \frac{K_2}{2\pi} d_{\text{PLL}}$ 
  lock  $\leftarrow \alpha \times \text{lock} + \frac{(i_p^2 - q_p^2)}{(i_p^2 + q_p^2)}$ 
   $D[k] \leftarrow \text{sign}(i_p)$ 
   $Z[k] \leftarrow c_p$ 
   $k \leftarrow k + 1$ 
return ( $\{D[n]\}_{n=0}^{\infty}$ ,  $\{Z[n]\}_{n=0}^{\infty}$ , lock)
  
```

Algorithm 4.1.6: FINDBITBOUNDARY($\{D[n]\}_{n=0}^{\infty}$)

```

 $A_{20} \leftarrow [1, 1, \dots, 1]_{1 \times 20}$       comment: An array of 20 ones
 $m \leftarrow 19$ 
while ( $|\{D\}_{(m-19)}^m \odot A_{20}| < 20$ ) or ( $D[m+1] \times D[m] > 0$ )
    do  $m \leftarrow m + 1$ 
return ( $m + 1$ )

```

Algorithm 4.1.7: FINDPREAMBLE($\{\text{NAV}[n]\}_{n=0}^{\infty}$)

```

found = 0
 $m \leftarrow 2$ 
 $P_{pre} \leftarrow [1, -1, -1, -1, 1, -1, 1, 1]_{1 \times 8}$       comment: GPS preamble
while (found = 0)
    while ( $|\{\text{NAV}\}_m^{m+7} \odot P_{pre}| < 8$ )
        do  $m \leftarrow m + 1$ 
    do  $L = \text{PARITYCHECK}(\{\text{NAV}\}_{m-2}^{m+29})$ 
        if ( $L = 1$ )
            then found = 1
return ( $m$ )

```

Algorithm 4.1.8: PARITYCHECK($\{A[n]\}_{n=0}^{31}$)

comment: A is an array with elements $\in \{+1, -1\}$

$\text{Word}_{(32 \times 1)} \leftarrow \{A[n]\}_{n=0}^{31}$

$\text{ParityBits}_{(6 \times 1)} \leftarrow \{A[n]\}_{n=26}^{31}$

$\text{ParityCheck}_{(6 \times 1)} \leftarrow [(M_{(6 \times 32)} \text{Word}_{(32 \times 1)}) \bmod 4] - 2$

comment: Modulo 4 and subtraction by 2 are to be done element-wise on the array

```

if ( $\text{ParityBits}_{(6 \times 1)} = \text{ParityCheck}_{(6 \times 1)}$ )
    then  $L = 1$ 
    else  $L = 0$ 
return ( $L$ )

```

Algorithm 4.1.9: MAIN($\{s[n]\}_{n=0}^{\infty}$)

```

 $K \leftarrow 0$ 
 $F_{aq}, C_{aq}, \text{SAT}_{aq}$ 
for sv  $\leftarrow 1$  to 37
  do  $\begin{cases} f_{aq}, c_{aq}, v_{aq} \leftarrow \text{ACQUISITION}(s[n], sv) \\ \text{if } v_{aq} > 0 \\ \quad \text{then} \quad \begin{cases} K \leftarrow K + 1 \\ \text{SAT}_{aq}[K] \leftarrow sv ; F_{aq}[K] \leftarrow f_{aq} ; C_{aq}[K] \leftarrow c_{aq} \end{cases} \end{cases}$ 
for i  $\leftarrow 1$  to K
  do  $\{D_i[n]\}_{n=0}^{\infty}, \{C_i[n]\}_{n=0}^{\infty}, \text{lock}_i \leftarrow \text{TRACK}(\{s[n]\}_{n=0}^{\infty}, \text{SAT}_{aq}[i], F_{aq}[i], C_{aq}[i])$ 

   $\{B[n]\}_{n=1}^K \leftarrow 0$ 
  for i  $\leftarrow 1$  to K
    do  $\begin{cases} B[i] \leftarrow \text{FINDBITBOUNDARY}(\{D_i[n]\}_{n=0}^{\infty}) \\ T_{ref} \leftarrow 0 \\ \text{sv}_{ref} \leftarrow 0 \\ \text{if } (T_{ref} < B[i]) \\ \quad \text{then} \quad \begin{cases} T_{ref} \leftarrow B[i] ; \text{sv}_{ref} \leftarrow i \end{cases} \end{cases}$ 
   $A_{1 \times 20} \leftarrow [1, 1, \dots, 1]$ 
  for i  $\leftarrow 1$  to K
    do  $\begin{cases} m \leftarrow 0 \\ \text{while } (1) \\ \quad \text{do} \quad \begin{cases} \text{NAV}_i[m] \leftarrow \text{sign}(\{D_i[n]\}_{n=B[i]+20m}^{B[i]+20m+19} \odot A) \\ m \leftarrow m + 1 \end{cases} \end{cases}$ 
   $B_{rel} \leftarrow (T_{ref} - B) \bmod 20$ 
   $B_{rel} \leftarrow 20 - B_{rel}$ 
   $B_{rel}[\text{sv}_{ref}] \leftarrow 0$ 
  for i  $\leftarrow 1$  to K
    do  $\{ \text{PrePos}[i] \leftarrow \text{FINDPREAMBLE}(\{\text{NAV}_i[n]\}_{n=0}^{\infty}) \}$ 
   $\text{PrePos}_{rel} \leftarrow \text{PrePos} - \text{PrePos}[\text{sv}_{ref}]$ 

  for i  $\leftarrow 1$  to K
    do  $\begin{cases} \text{CodePhase}[i] = C_i[1 + T_{ref} + B_{rel}[i] + 20 \times \text{PrePos}[i]] \\ \text{DelayDifference}[i] = (\text{PrePos}_{rel}[i] \times 20 \times 10^{-3} + B_{rel}[i] \times 10^{-3} + \frac{\text{CodePhase}[i]}{F_s}) \text{ seconds} \\ \text{pseudoRange}[i] = (65 \times 10^{-3} + \text{DelayDifference}[i]) \times 299792458 \text{ meters} \end{cases}$ 
  return ( $\{\text{pseudoRange}[n]\}_{n=1}^K$ )

```

4.2 Experimental Setup

Actual data from GPS satellites is recorded and then input into the software receiver, which processes it offline to obtain the pseudoranges. Recording is done using USRP + GNURadio setup. The recording is also processed with GNSS-SDR, a GNURadio based open source GNSS receiver implementation and the results are compared. Described below are each component used in recording and validation.

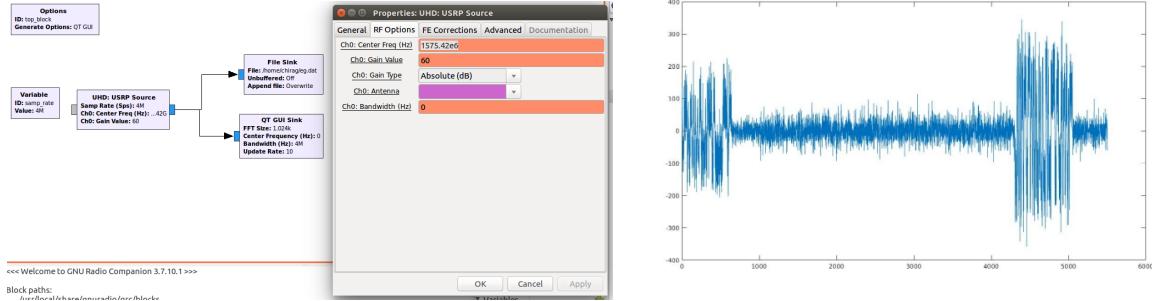
GPS active antenna

Usually GPS antennas are patch antennas designed for RHCP (right hand circular polarised) signal. Since GPS antennas have to be placed in open location for good reception, long wires are required to carry signal from the antenna to the processing unit. However the GPS signals are so weak that the cable losses can degrade SNR considerably, beyond processing unit's ability to recover it. Thus most GPS antennas come with an in-built Low-Noise Amplifier(LNA), so that the antenna output (both signal and noise, impartially) is amplified before being transmitted through the long cable. The LNA is also important to amplify the signal to be greater than the ADC quantization level. We use a readily available active GPS antenna centered at 1575.42MHz and requiring a 3.3V to 5V supply. The LNA has +20dB gain and is very wideband. The antenna must be kept under open sky and avoiding any shadow regions caused by tall structures.

USRP & GNURadio

Universal Software Radio Peripheral (USRP) with DBSRX 2 (Rev. 2) daughter board from Ettus Research is used as to record the signal from the antenna. DBSRX 2 is used since it has an onboard bias-tee to power the active antenna. G Following must be noted about using the USRP and GNURadio

- USRP front-end filters and samples the input at 100 MHz irrespective of the sampling frequency set by the user. The signal is then digitally filtered according to Nyquist, about the center frequency, and downsampled to the desired sampling rate F_s . Setting 0 for bandwidth in GNURadio will make the USRP filter the signal to maximum bandwidth allowed according to Nyquist criterion, for a given F_s .
- USRP and the setup will have a frequency range in which it operates, typically 50MHz to 2.2GHz. Thus it can be used for L1 and L5 band recordings, but not for S band. We sample GPS L1 signals at 1575.42MHz.
- USRP has a stable clock source with very little jitter. The center frequency shifts due to clock stability is around 100Hz at 1575.42MHz, which is within the lock-in range of PLL.
- Since GPS signal recording involves sampling rate in the upwards of 2MHz, USRP must have enough buffer size. Else, the samples are dropped which can render the data useless, since timing of each sample is important for pseudorange computation. Figure 4.1b shows



(a) GNURadio settings used for recording

(b) Result of insufficient buffer length. This data was recorded on USRP B2100 with $F_s = 4\text{MHz}$

the result of tracking for such a data recording. For this reason, USRP N210 is used. Note that it requires a PC/laptop with 1 Gb Ethernet port with communicate with USRP N210. Also, when GNURadio prints either 'D' or 'O' on the terminal it indicates a buffer overflow. (Refer <http://gnss-sdr.org/conf/> for a brief description).

- It was observed that the data must be recorded with a USRP gain of atleast 40dB for the software receiver and GNSS-SDR to work. The setting is to be done in GNURadio as shown in Figure 4.1a. We keep a gain of 60dB.
- The USRP down-converts the signal to baseband resulting in a complex signal. Both the inphase and quadrature components are then sampled at F_s . Each complex sample has a 32-bit real and 32-bit imaginary part, which are interleaved and stored into a file.

GNSS-SDR

GNSS-SDR [27] is an open-source implementation of Global Navigation Satellite System receiver in software, aimed at software-defined radio applications. GNSS-SDR is used to validate the experimental setup. The recorded file from the above setup is input to the GNSS-SDR program and it locks to the correct coordinates. GNSS-SDR requires a satellite signal strength of 40dB-Hz or more to detect and lock on to the satellite. The working of GNSS-SDR and of the antenna on first use were verified by comparing them with a commercially available complete GPS receiver. GNSS-SDR provides a observables file after processing the recorded data. It contains information like the Doppler shift, pseudorange, SNR etc for each of the acquired satellite. This is used to validate the outputs of the software receiver.

4.3 Results and Validation

The data samples recorded from the setup at $F_s = 4\text{MHz}$ were input into the software implementation. Described below are the results for a 3 minute data recorded on 19th September 2016, at Department of Electrical Engineering, IIT Bombay. Satellite vehicles 8, 10, 14, 18 and 32 were visible at the time of recording as validated using GNSS-SDR.

Figure 4.2 shows the result of ACQUISITION for each of the visible satellites. Acquisition result

for satellite 2 is also shown to compare the difference in acquisition result with and without presence of the signal. Note that the signal from satellite 8 is weak and hence the correlation value doesn't cross the threshold. However, the peak repeats in the same bin for two successive attempts, and thus the function declares it as detected. Prominent second and third peaks in some cases is due to strong correlations at same code-phase but $(\tilde{f}_d + 500)$ Hz and $(\tilde{f}_d - 500)$ Hz frequency bins.

The tracking block is initialized after acquisition. Figure 4.3 shows the local frequency value as the PLL settles to actual Doppler Frequency. Figure 4.5 shows the output of tracking loop revealing the navigation bits. Figure 4.6 shows the delays in reception of the preamble. These delays along with DLL code-phase tracks are computed and used in calculating the relative pseudoranges. Figure 4.4. Different rates of code phase change is indicative of different Doppler shifts or the different rates at which the satellite is moving towards (or away incase of satellite 32) the receiver.

Figure 4.7 compares the relative pseudorange of satellites 18 and 10, as calculated by the code and GNSS-SDR. The pseudoranges are computed every 6 seconds by the code, at the arrival of preambles. The pseudo-ranges computed by GNSS-SDR are around the same time though not exactly at the same instance as the code. This may contribute slightly to the error between the two computations. Moreover, the error is within 300 meters, which implies that the DLL tracks of the code and GNSS-SDR are within about 1 chip difference from one another. A better DLL implementation whether the speed of Generation of the code itself is controlled rather than just the code-phase may yield better result as discussed in 3.2.

Figure 4.8 shows similar sudden changes in the frequency tracked by the PLL for all the satellites. This is the result of center frequency shifts in the USRP. However the shifts are small and within the lock-in region of PLL, and thus the lock is not lost. Figure 4.10 and 4.11 demonstrates how the lock measure used in TRACK may be used.

False PLL Lock

Refer to Figure 4.12 and 4.13. As discussed in Section 3.2.2, the PLL can not distinguish between f_d and $f_d + k \frac{1}{2T_{int}}$ for any integer k . In Figure 4.12, the receiver operates on incoming samples to acquire data from satellite 32 (19th September 2016 recording). The top row shows that acquisition gives $\tilde{f}_d = 500$ Hz and PLL locks to correct $f_d \approx 750$ Hz, which is near the midpoint of 500 Hz and 1000 Hz frequency bins. However if the receiver starts operating a second later, the noise tips the acquisition value in favor of 1000 Hz, resulting $\tilde{f}_d = 1000$ Hz. The PLL locks, but to $f_d + 500$ Hz. Figure 4.13 shows that the lock measure will not indicate such false locks. Thus when the receiver observes that PLL has locked but the output sign(I) are not as expected, it should try initializing the PLL with an offset of $\pm \frac{1}{2T_{int}}$ Hz.

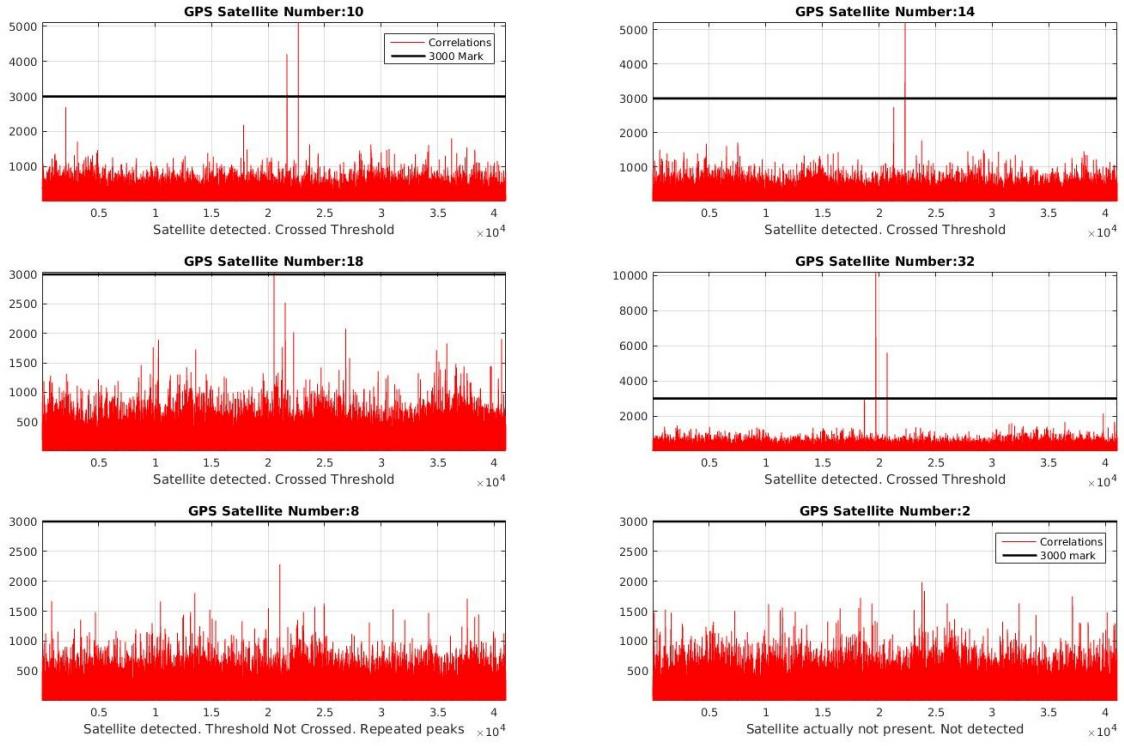


Figure 4.2: Acquisition output. The correlation values of 2D search over frequency and code-phase is plotted as a 1D array.

NAV bit flips

Refer to Figure 4.14. These are results for SV 16, for a data collected on 6th May 2017. The satellite signal had low SNR (for the software receiver given here and with $T_{int} = 1\text{ms}$) of about 40 to 42 dB-Hz, as measured in GNSS-SDR. The lock measure is noisy, indicating the rough performance of the PLL. This increases the probability of PLL undergoing the 180° phase switch as discussed in Section 3.2.2. One such event is shown in Figure 4.14.



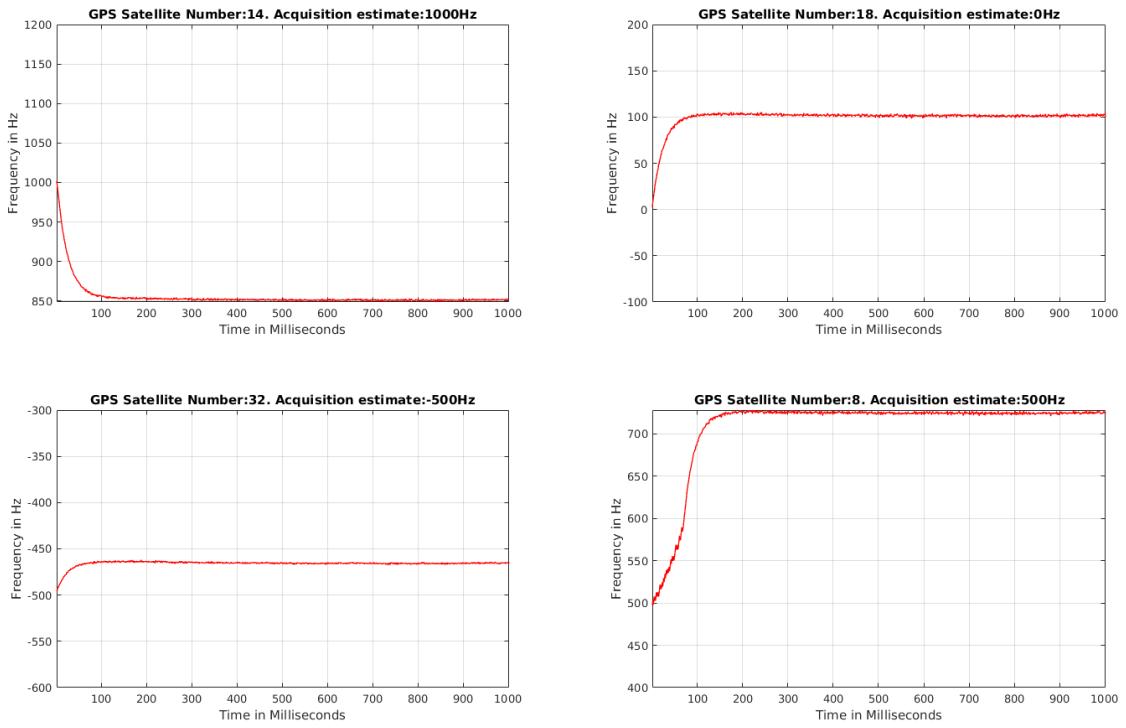


Figure 4.3: PLL frequency tracking output, beginning with the acquisition output value, over the next 1 second

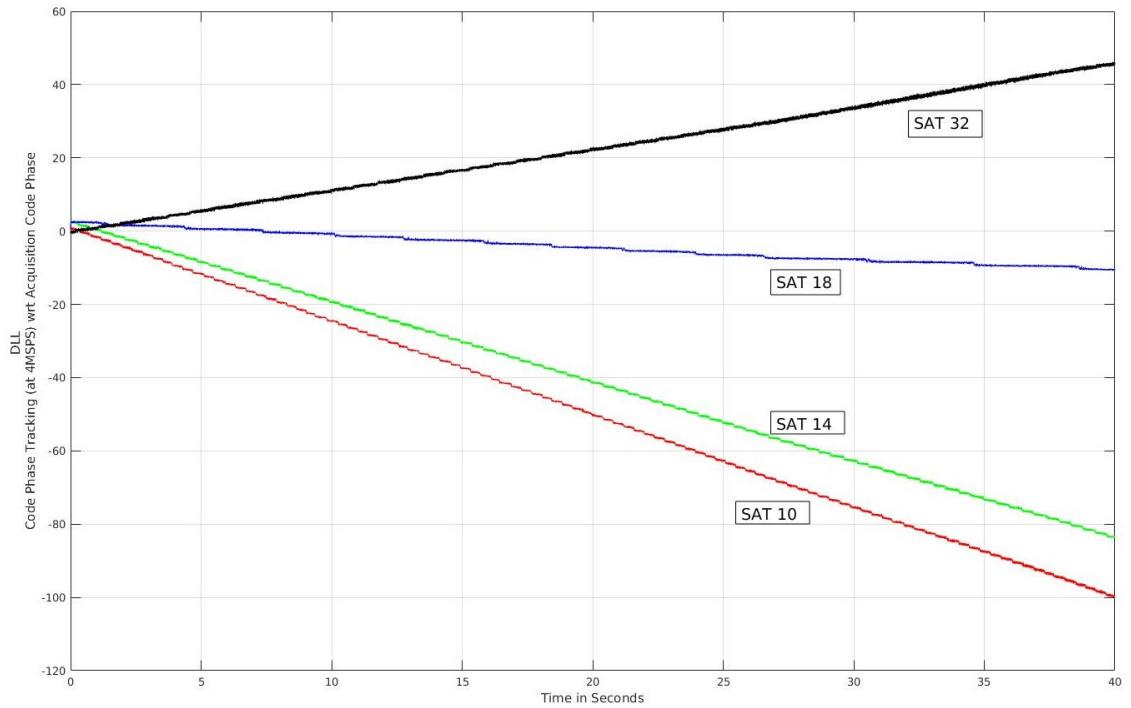


Figure 4.4: Code Phase tracked by DLL of each channel relative to their starting value (the value from acquisition)

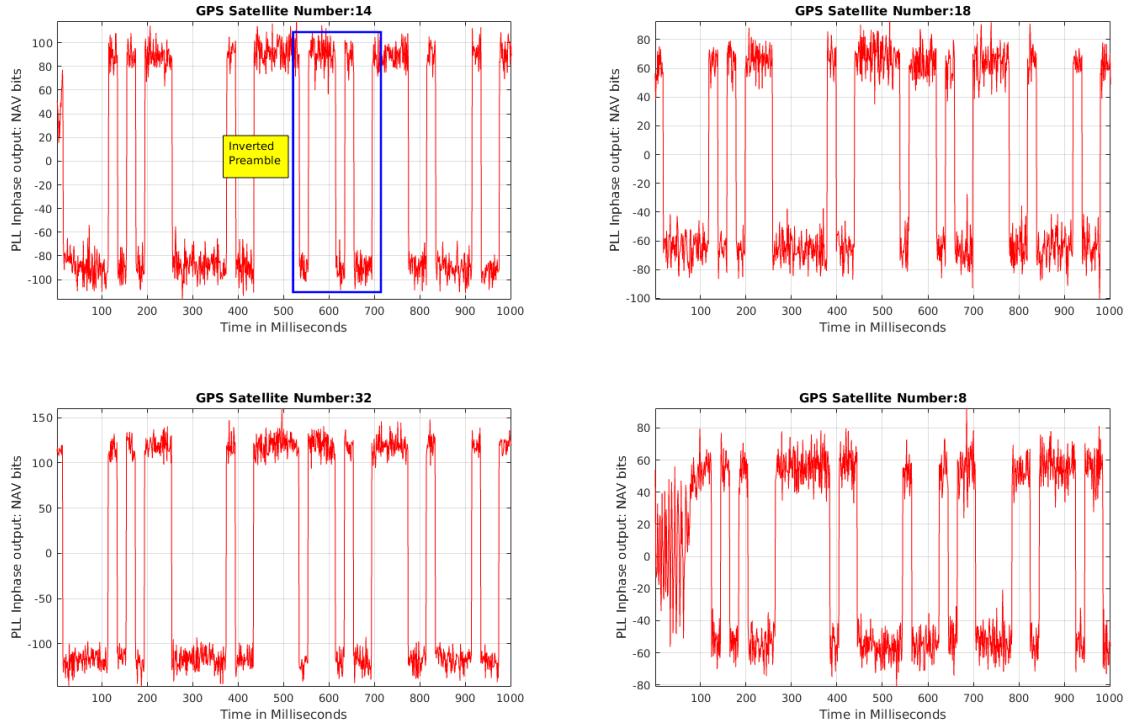


Figure 4.5: Output of the TRACK procedure settles to show the NAV bits. Preamble can be found around the same time in all the NAV bit streams (was validated using parity check)

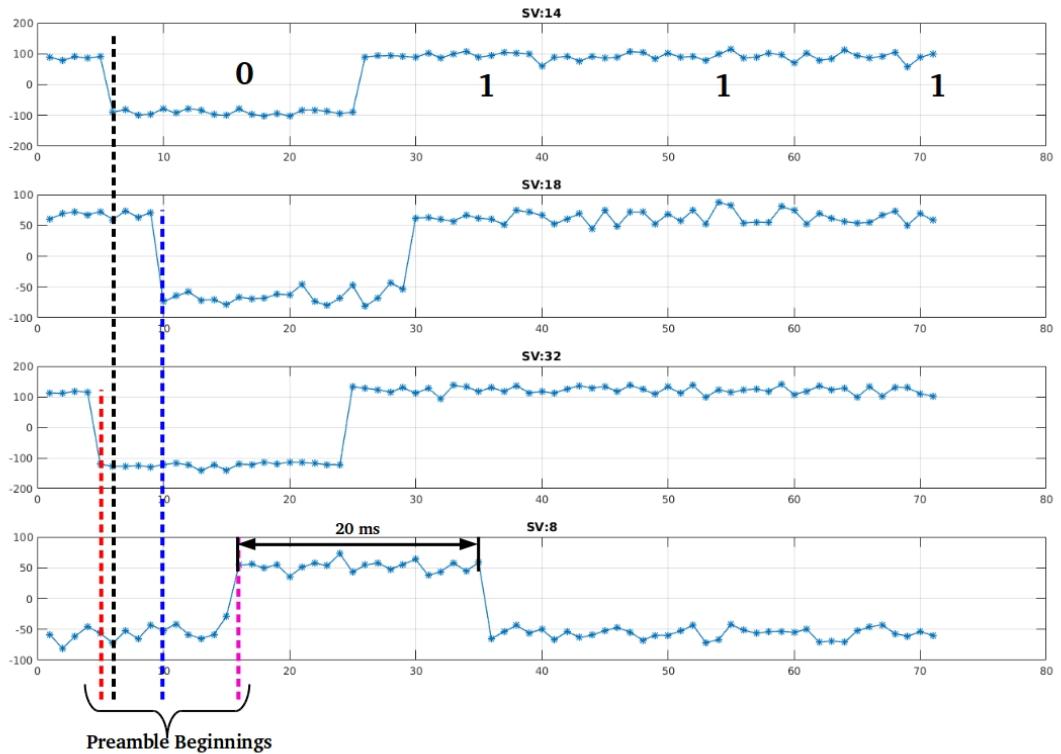


Figure 4.6: Delays between arrival of Preamble from different satellites

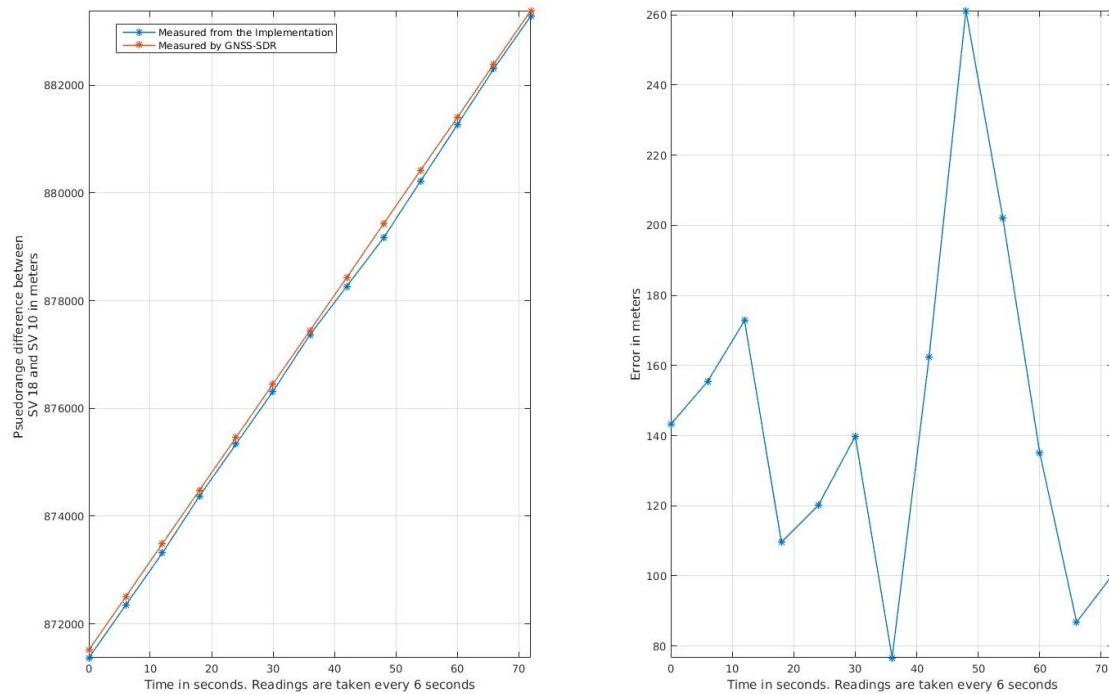


Figure 4.7: The difference between pseudorange of satellite 10 and satellite 18, as computed by the software implementation and GNSS-SDR are compared. The error for other satellites are in the same order

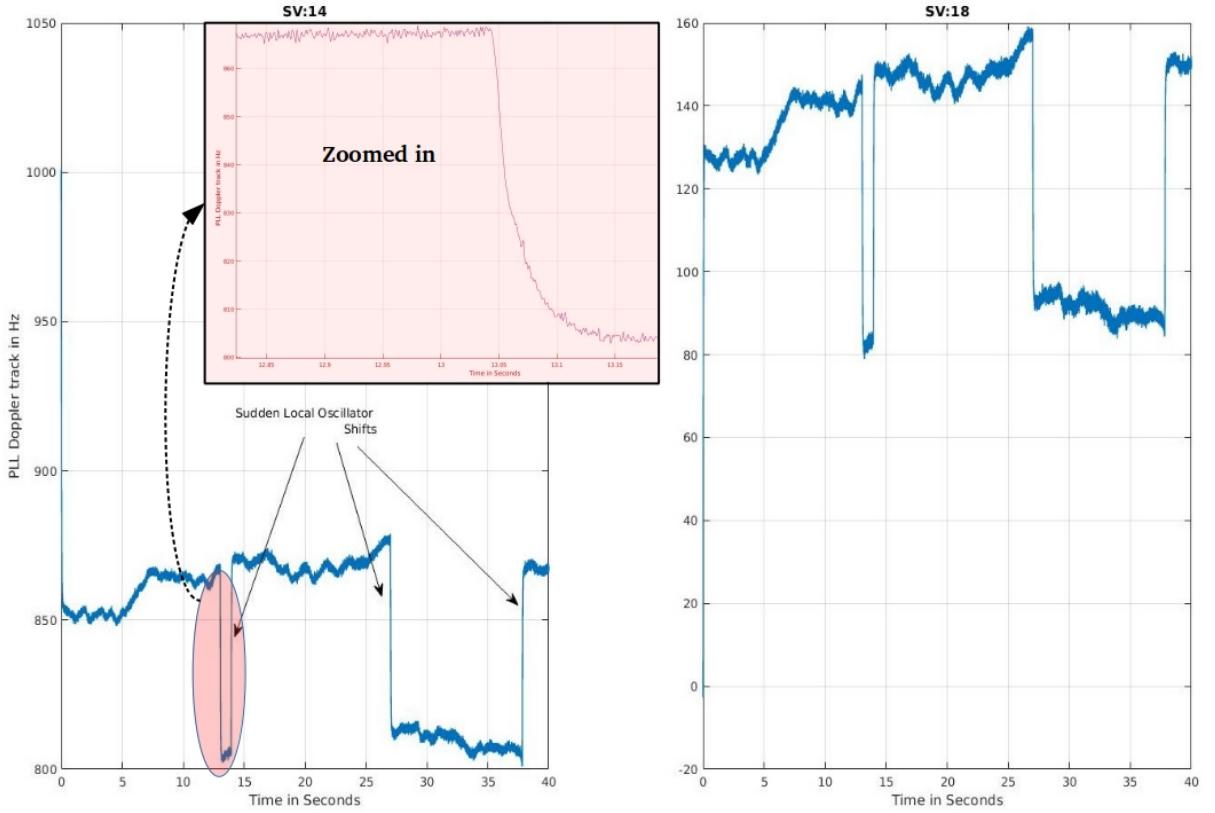


Figure 4.8: Frequency tracked by PLL over 40 seconds for two different satellites shows sudden shifts in receiver center frequency

PREAMBLE	PARITY BITS	TOW (must be XORed with 1)	SF ID (XOR with 1)	WEEK NUMBER
SF 1 1 0 0 0 1 0 1 1 0 0 0 0 1 1 0 1 1 1 1 0 1 1 0 0 0 0 1 1 1 1	1 1 0 1 0 0 1 1 1 0 1 0 0 0 0 0 1 1 1 1	1 1 0 1 0 0 1 1 1 0 1 0 0 0 0 0 1 1 1 0	1 1 0 1 1 1 0 1 1 1 0 0 0 1 0 0 0 0 0 0 0	1 1 0 1 1 1 1 1 0 1 1 0 1 0 0 0 0 0 0 0 0
SF 2 1 0 0 0 1 0 1 1 0 0 0 0 1 1 0 1 1 1 1 0 1 1 0 0 0 0 1 1 1 1	1 1 0 1 0 0 1 1 1 0 1 0 1 0 0 0 0 1 0 1 0	1 1 0 1 0 0 1 1 1 0 1 0 1 0 0 0 0 1 0 1 0	0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0	0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SF 3 1 0 0 0 1 0 1 1 0 0 0 0 1 0 1 1 0 1 1 0 0 0 0 1 1 1 1	1 1 0 1 0 0 1 1 1 0 1 0 1 0 0 0 0 0 1 1 1 0	1 1 0 1 0 0 1 1 1 0 1 0 1 0 0 0 0 0 1 1 1 0	1 1 1 0 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0	1 1
SF 4 1 0 0 0 1 0 1 1 0 0 0 0 1 1 0 1 1 1 1 0 1 1 0 0 0 0 1 1 1 1	1 1 0 1 0 0 1 1 1 0 1 0 1 0 0 0 0 0 0 1 1 1 0	1 1 0 1 0 0 1 1 1 0 1 0 1 0 0 0 0 0 0 1 1 1 0	0 1 1 1 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0	0 1 1 1 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
SF 5 1 0 0 0 1 0 1 1 0 0 0 0 1 1 0 1 1 1 1 0 1 1 0 0 0 0 1 1 1 1	1 1 0 1 0 0 1 1 1 0 0 1 1 1 1 0 0 0 0 1 1 1 1	1 1 0 1 0 0 1 1 1 0 0 1 1 1 1 0 0 0 0 1 1 1 1	0 0 1 0 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0	0 1 0 1 0 0 0 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
SF 1 1 0 0 0 1 0 1 1 0 0 0 0 1 0 1 1 1 1 0 1 1 0 0 0 0 1 1 1 1	1 1 0 1 0 0 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 0 0	1 1 0 1 0 0 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 0 0	0 0 0 0 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0	1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
SF 2 1 0 0 0 1 0 1 1 0 0 0 0 1 1 0 1 1 1 0 1 1 0 0 0 0 1 1 1 1	1 1 0 1 0 0 1 1 1 0 0 1 1 0 1 1 1 1 0 1 0 1 0	1 1 0 1 0 0 1 1 1 0 0 1 1 0 1 1 1 1 0 1 0 1 0	0 0 0 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0	0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
SF 3 1 0 0 0 1 0 1 1 0 0 0 0 1 1 0 1 1 1 0 1 1 0 0 0 0 1 1 1 1	1 1 0 1 0 0 1 1 1 0 0 1 1 0 1 1 1 1 0 1 0 0 0	1 1 0 1 0 0 1 1 1 0 0 1 1 0 1 1 1 1 0 1 0 0 0	0 0 1 0 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0	1 1
TLM WORD				
HOW WORD				

Figure 4.9: NAV bit frames from SV 32, 19th September 2016, first 70 bits of each frame are shown. Verify that the Week Number in decimal is 891 (Check here <https://www.labsat.co.uk/index.php/en/gps-time-calculator>). The SF ID and TOW increment by 1 every subframe.

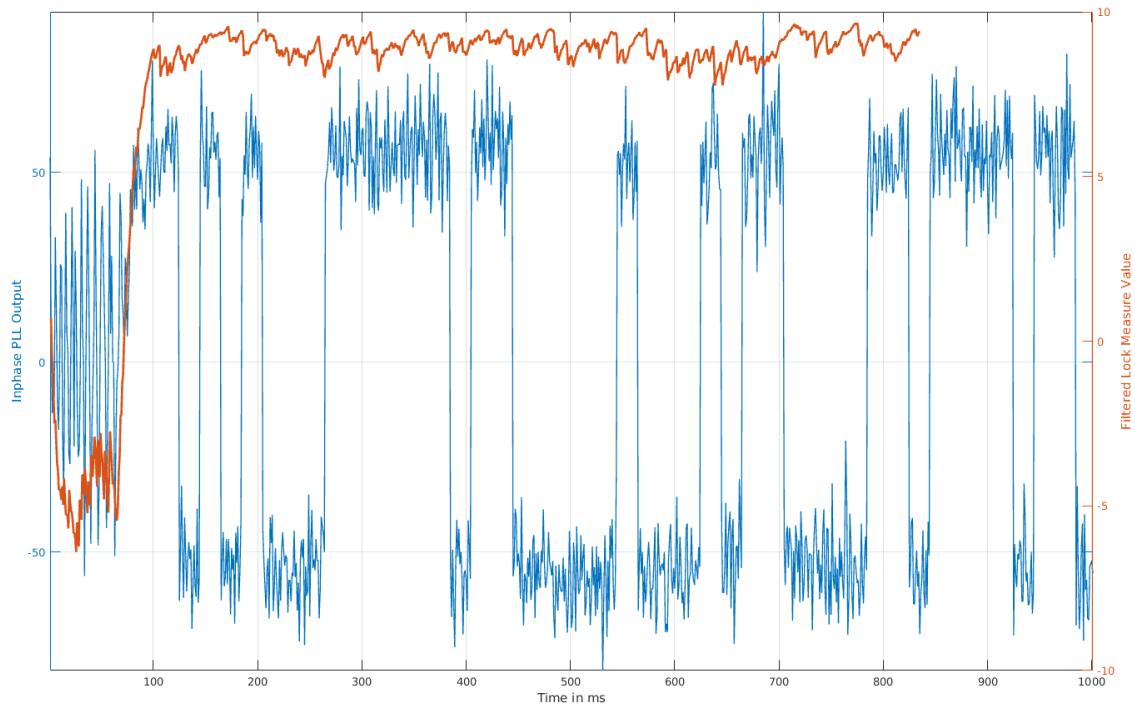


Figure 4.10: Measure of PLL lock plotted along with the PLL output. the measure of lock can be used to decide if the PLL output has settled and can be sampled to obtain NAV bits

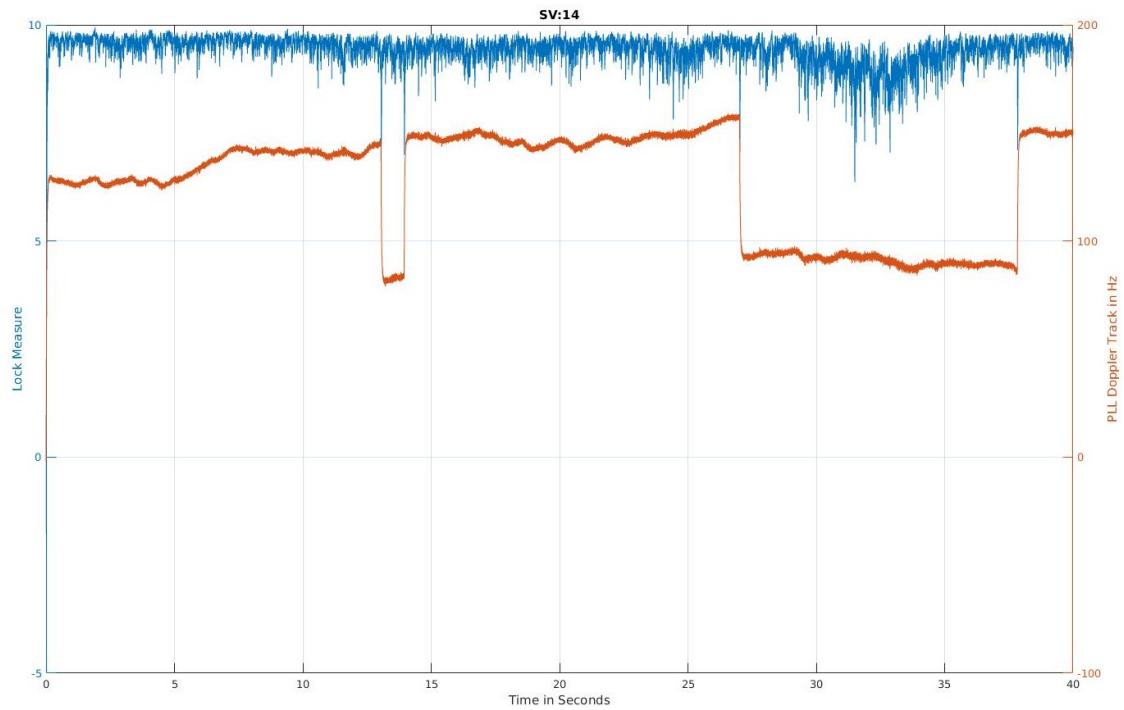


Figure 4.11: The lock measure can also be used to detect momentary deviations in the PLL and thus the sudden frequency shifts

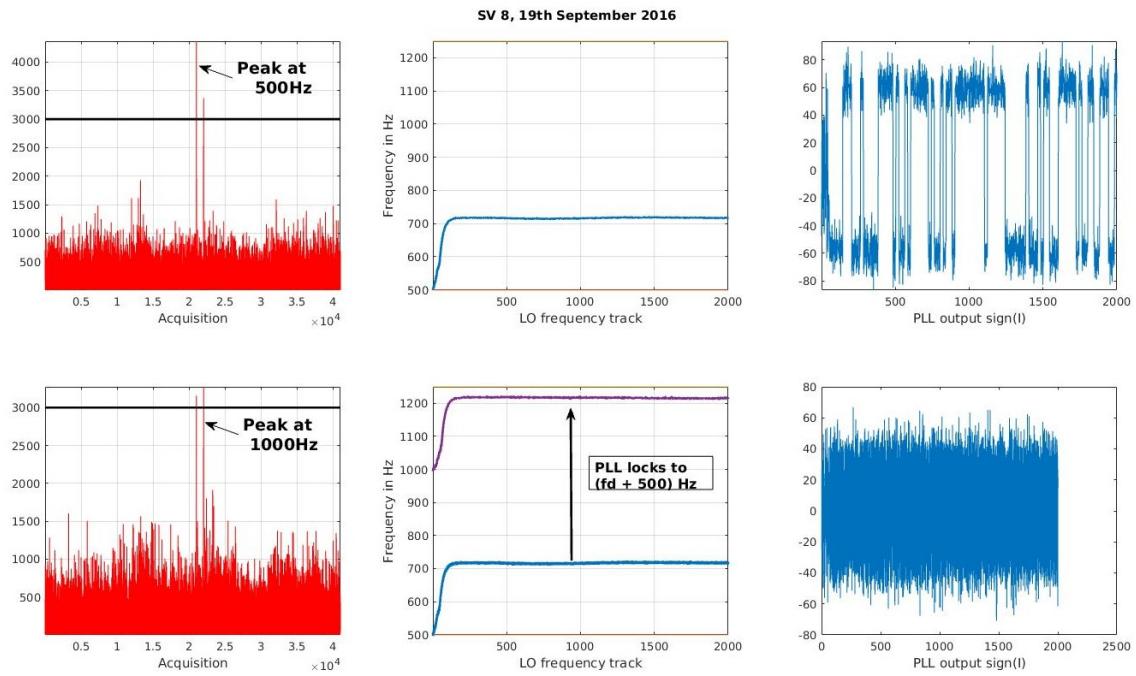


Figure 4.12: Acquisition and tracking performed on a recorded data with a separation of 1 second. PLL locks to $(f_d + 500)$ Hz due to wrong acquisition

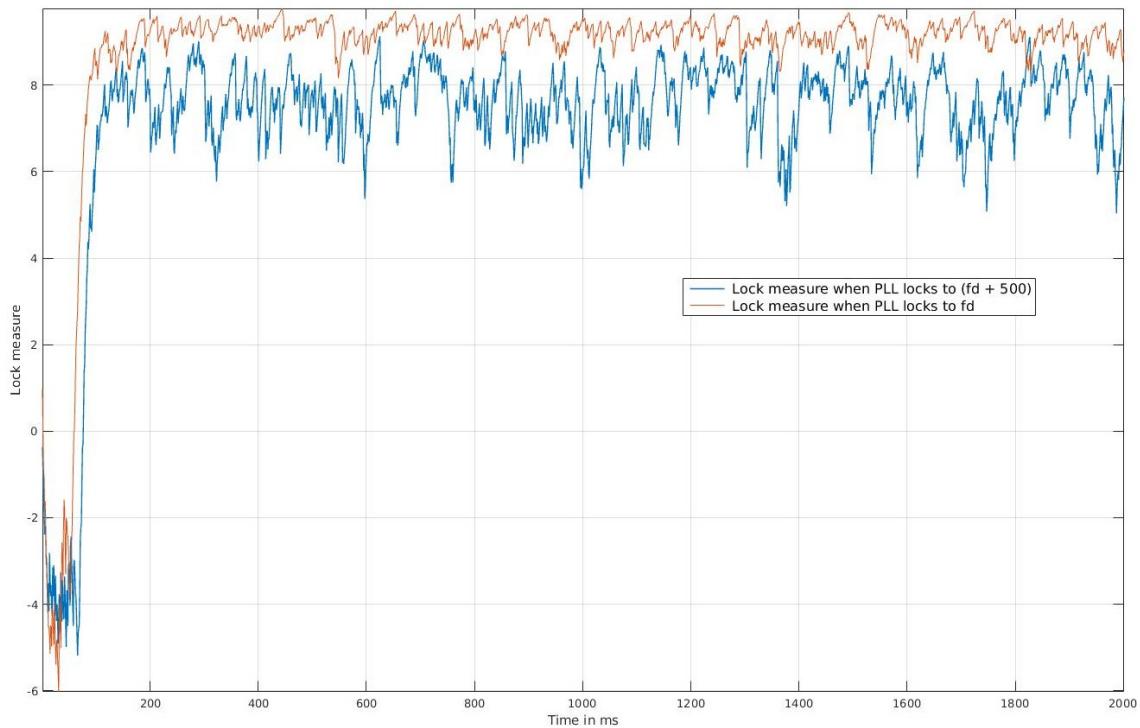


Figure 4.13: Lock measure is only slightly less when PLL locks to $(f_d + 500)$ Hz

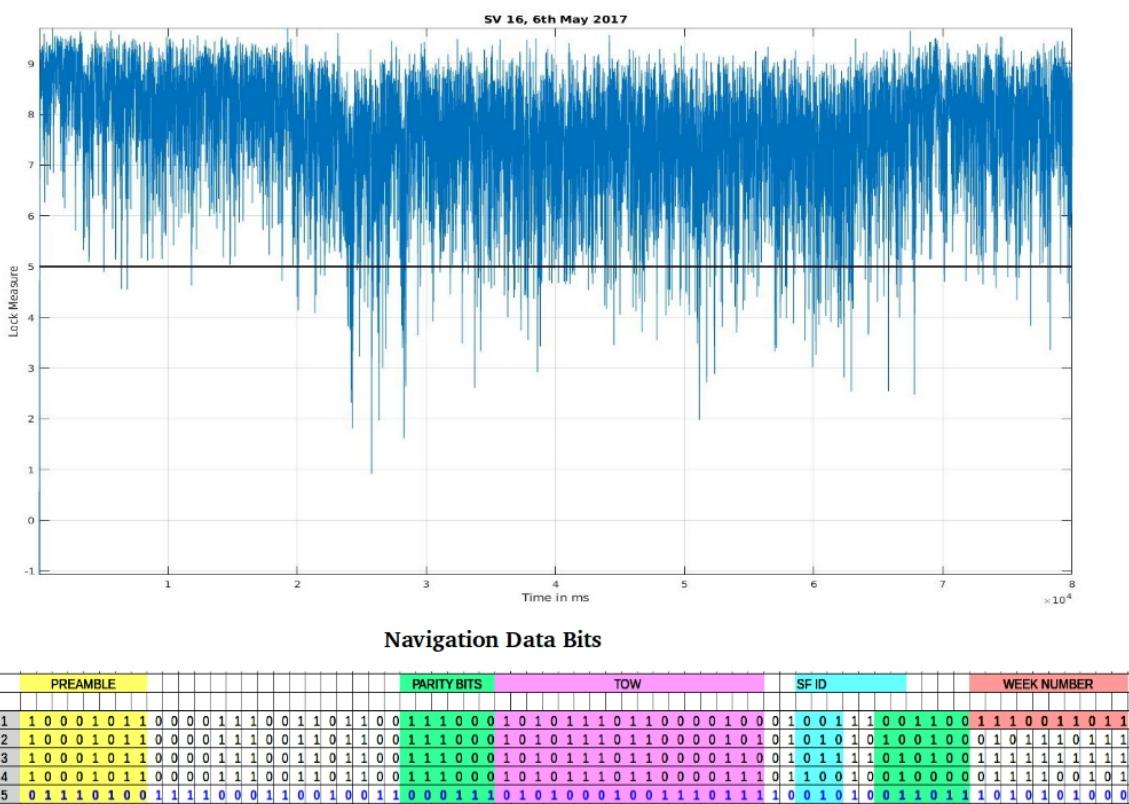


Figure 4.14: Lock measure obtained while processing data from SV 16, 6th May 2017 over 80 seconds. The signal from SV 16 was very noisy, which is reflected in the lock measure. NAV bit frames, first 70 bits of each frame are shown. Subframe 5 is inverted indicating a PLL flip somewhere in subframe 4

Chapter 5

IRNSS Front-end: Hardware Implementation

(*This Chapter is based on the work done along with Arunabh Saxena, Praveen Sriram and Vireet Moghe as their ‘EE344:Electronic Design Lab’ project under the guidance of Prof. Shalabh Gupta, EE, IITB*)

As outlined in 1.1, the main difference between GPS and IRNSS is the frequency band used for transmission. While the same signal processing programs can be used to decode both IRNSS and GPS signals with minor distinctions, front-end tuned for different frequencies are required to capture the signals. This chapter begins by introducing the generic front-end architecture common to any GNSS receiver. This is followed by the design details of a receiver front-end, built for S-band (2492.028 MHz) of IRNSS. The signal captured with the said front-end was recorded using the setup described in previous chapter. The program presented in Chapter 4 was able to acquire and track signals from satellites of IRNSS constellation, simply by replacing the C/A codes of GPS with that of IRNSS.

5.1 GNSS receiver front-end architecture

Figure 5.1 shows block diagram of a standard receiver, along with Fourier domain visualization of the signal at each stage of the front-end. Recapping the signal model, raw signal received at

the antenna ① is:

$$s_r(t) = \underbrace{\sqrt{P} D(t - \tau) x(t - \tau) \cos(2\pi(f_c + f_d)t + \theta)}_{s(t)} + n(t)$$

where, $D(t)$ = Navigation Data Bits

$x(t)$ = PRN code

τ = Signal Travel Time

f_c = Carrier Frequency

f_d = Doppler Frequency Shift

$n(t)$ = White Gaussian Noise

The amplified band-pass filtered signal about f_c can be represented as ②,

$$s_r(t) = \sqrt{P_r} D(t - \tau) x(t - \tau) \cos(2\pi(f_c + f_d)t + \theta) + n_I(t) \cos(2\pi f_c t) - n_Q(t) \sin(2\pi f_c t)$$

where, $n_I(t) + jn_Q(t)$ = Baseband equivalent of the band-pass filtered white noise

As per the ICD [11], maximum signal power received at the antenna (in 20MHz) is -153dBW (i.e 123dBm), and -139dB after first stage amplification of say 14dB . The noise power at antenna can be found as $N = kT_E B$, where k is the Boltzmann's constant ($1.3806 \times 10^{-23} JK^{-1}$), T_E kelvin is effective noise temperature (513K) and B is the concerned bandwidth (20MHz in this case). Note that the effective noise temperature represents net effect of thermal noise due to the actual temperature and the noise figure of the amplifier electronics. That gives noise density of $-201.5 dBW/Hz$ and noise power of $-201.5 + 73 = -128.5 dBW$ (-114.5dBW after LNA of 14dB). So SNR that can be expected is $-153 + 128.5 = -24.5 dB$ [14]. The signal is thus buried under the noise floor and will not be visible on a spectrum analyzer.

As the signal $s_r(t)$, received at the antenna, flows through the receiver chain, each component adds its own noise, say $n_i(t)$ (the power of which is quantified as 'noise figure' specification of the component). So the signal finally sampled will be $s(t) + n(t) + \sum_i n_i(t)$. Since the desired signal $s(t)$ is already so weak, any further degradation of SNR in the receiver should be minimized. To maintain the SNR no worse than at the antenna, amplify the signal from antenna, say by a factor of A , before any other processing. Then the final signal would be $A(s(t) + n(t)) + \sum_i n_i(t)$ and the SNR, for large A , $\frac{A^2 \times P}{(A^2 P(n) + \sum_i P(n_i))} \approx \frac{P}{P(n)} = \text{SNR}$ at the antenna ($P(n_i)$ are noise powers). The noise power added (which is inevitable) in the process of amplification itself should be very low. That is to say that the amplifier should have a very low noise figure. Such an amplifier is called a **Low Noise Amplifier**(LNA), with a noise figure of typically less than 1 dB. Noise floor is the thermal noise power plus the noise figure of the component (note that the addition is in time scale, not dB scale). So it can be said that a LNA amplifies input signal without raising the noise floor by too much. LNA is the first component after the antenna in

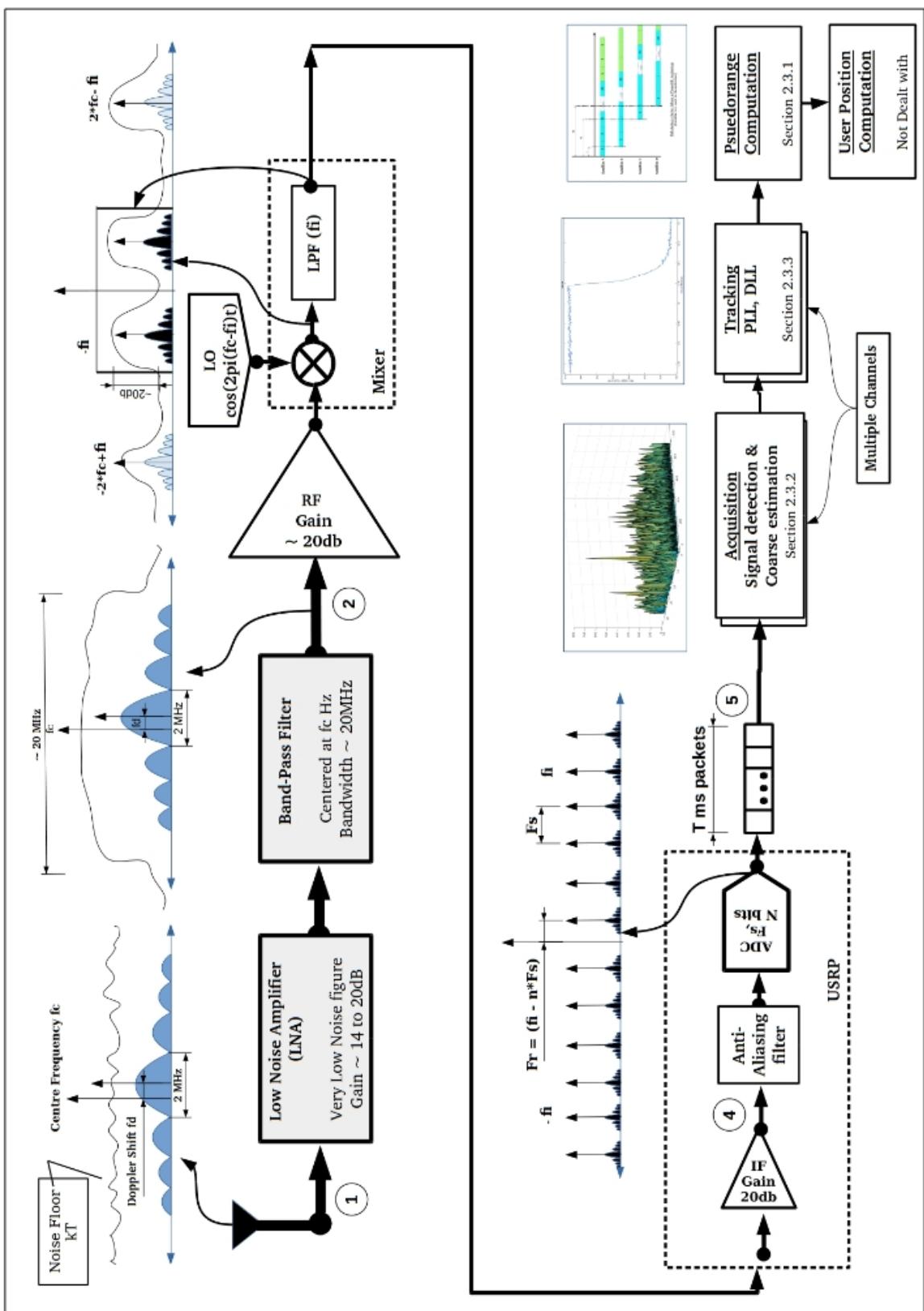


Figure 5.1: Complete Receiver Block Diagram

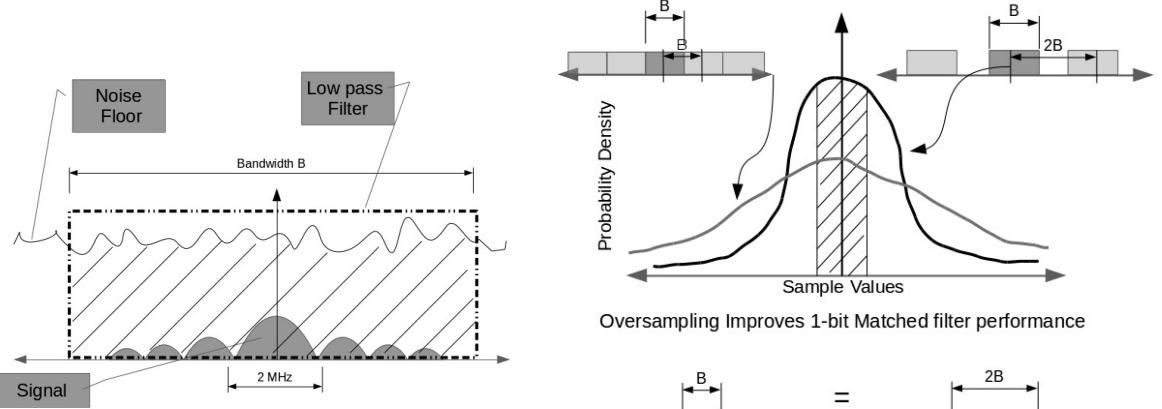
any GNSS receiver. The commercial ‘GPS antennas’ are actually the passive antenna plus an LNA. They have to be powered, usually with a 3.3V or 5V supply, provided through a bias-tee. Hence they are referred to as **active antennas**.

Having ensured considerable immunity to receiver noise, the next step is to down-convert the signal at a large carrier frequency of f_c to a reasonable intermediate frequency f_I within the analog bandwidth of ADC used. The process of down-conversion is carried out in a ‘**mixer**’, followed by a low pass filter (LPF). Mixer multiplies the input signal carrier $\cos(2\pi(f_c + f_d)t + \theta)$ with a locally generated $\cos(2\pi(f_c - f_I)t)$ and the resulting signal is passed through the LPF to eliminate the high frequency component. The result is the signal $\cos(2\pi(f_I + f_d)t + \theta)$. Down-conversion to f_I could have been achieved using $\cos(2\pi(f_c + f_I)t)$ as well. This is referred to as **higher side injection**, while the former is known **lower side injection**. In case of lower-side injection, frequency $f_c + f$, appears at $f_I + f$ in the down-converted signal, but in higher-side injection, it appears at $f_I - f$, towards the baseband. If the captured signal has unwanted signals at frequencies lower than the band of interest, then one would choose higher-side injection. This would put the unwanted signals away from baseband and a LPF can attenuate them. In practice, a mixer chip includes the signal multiplier, low pass filter and possibly gain stages as well. The local signal for the mixer is generated by a **Local Oscillator (LO)**, which generally is a digitally controlled analog PLL chip. Stability of the LO is an important consideration as discussed in Section 2.3.3 and 3.2.2.

Usually mixers have large noise figures and the input signal should be significantly above the noise floor of the mixer. As an empirical thumb rule, the input signal power should be about 20 dB more than the noise floor of the mixer. Hence, our captured signal $s_r(t)$ must be amplified by atleast [noise figure of mixer + 20] dB before the mixer. A single LNA stage may not be able to provide this gain, hence multiple amplifiers, not necessarily of the low-noise kind, may be needed before the mixer. At this point, note that the noise $n(t)$ will be very wideband (white-noise-like). Thus passing the amplified LNA output as it is to the subsequent gain stages may saturate them. This necessitates the use of a **band-pass filter (BPF)** after the LNA (since putting it before the LNA would degrade SNR) and before the gain stages. The BPF should allow f_c and a band of atleast $2 \times \frac{1}{T_c} = 2.046$ MHz about it, without much loss. T_c is the GPS/IRNSS chip time. Besides avoiding saturation, BPF also performs the following two functions:

- **Image frequency reduction:** After the mixer down-converts with LO frequency of $(f_c - f_I)$, signals at f_c and $f_c - 2f_I$, both appear at f_I . Thus we would need that the power at $f_c - 2f_I$, due to noise or otherwise, be attenuated sufficiently before entering the mixer. The BPF can be so chosen to ensure this.
- **Interference elimination:** S-band signal of IRNSS is spectrally very near to the 2.4 GHz WiFi, which is usually very powerful. A BPF with sharp roll-off on the lower frequency side can be used to reduce the impact of WiFi. If this is not ensured, strong WiFi signal may saturate the RF chain, in-spite of presence of the BPF.

The BPF will cause an unavoidable loss in the band of interest. It must be included in deciding



(a) Choosing the pre-sampling bandwidth B_p for GPS signal. Signal main lobe is 2.046 MHz wide

(b) Probability densities of sample values for different B_p and its effect on one-bit sampling.

the required RF chain gain. Also note that a large f_I is preferred for image frequency rejection. Bandwidth of the ADC used however is the upper limit.

After the signal has been amplified, filtered and down-converted, it can now be sampled. Like with mixer, we need to ensure that the signal is sufficiently higher than the noise floor of the ADC. Moreover the signal should have sufficient amplitude to occupy the dynamic range of the ADC. Thus additional gain in the intermediate frequency (IF) stage may be required. The amount of IF gain required is dictated by the ADC specifications. In GPS signal processing varying power of the input signal is undesirable since some receiver parameters must be changed accordingly, as discussed in the previous chapters. Hence the IF stage may be designed to have an **automatic gain control (AGC)** ability. The AGC unit may be inbuilt in the mixer output, or into the ADC input. AGC can also ensure that the entire dynamic range of the ADC is always used, improving accuracy of the signal samples. AGC however is not required in case the signal is one-bit sampled, as in some primitive receivers. After IF gain stages, if any, the signal is filtered to prevent aliasing and sampled. The pre-sampling bandwidth B_p , the sampling rate F_s and the number of levels of quantization L are important parameters. They are briefly discussed next.

5.1.1 Sampling Parameters

[17] is a widely cited paper with respect to sampling in GNSS literature. The paper presents simulated results on the effect of these parameters on the match-filtering process, like acquisition in our case. Here we discuss intuitions to make good choices for B_p , F_s and L . Following are some interesting theoretical considerations:

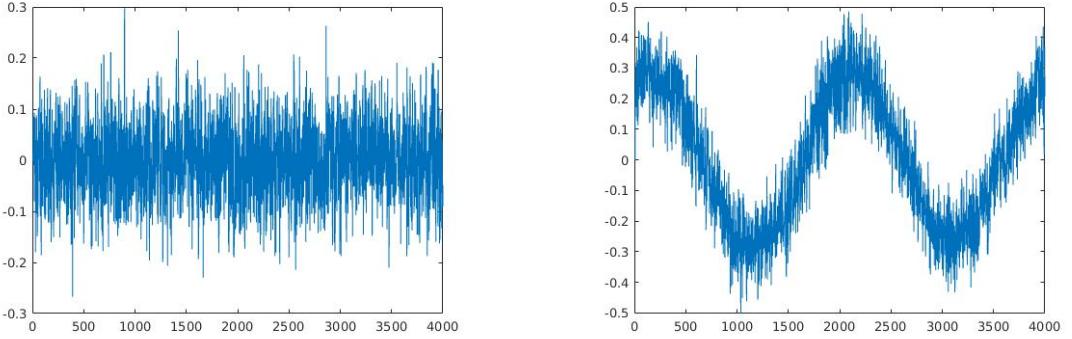
- Firstly, suppose the signal samples are not quantized and we retain the analog values (or high precision sampling more realistically), then clearly $B_p = 2F_s$. This is to satisfy the Nyquist criterion and because oversampling does not give any advantage. The question

then is: for a wideband, very low SNR signal like GPS, what value of B_p should one choose? As shown in 5.2a, as the bandwidth is increased, the receiver collects more noise than signal. Thus SNR falls with B_p . But however match-filtering analysis like in [5] tells that increasing B_p will improve acquisition peaks, though with diminishing returns. Also, increasing B_p proportionally increases computational cost since sizes of arrays involved in acquisition, tracking etc increase. With these trade-offs, B_p to cover 2 to 4 lobes of the signal seem sufficient. Thus in all our implementations, we take $F_s = 4$ MHz, $B_p = 8$ MHz (USRP does it for us). Since the USRP does high precision sampling, the software implementations were tested under this case. However, it has worked even when the samples were converted to one-bit and fed to the software receiver.

- Now we need to quantize the samples. Figure 5.3b shows how a strong narrow-band interference degrades the signal. Usually jammers sweep the entire band with such interference and the receiver needs to appropriately notch-filter them out [20]. So if the receiver intends to have anti-jamming capabilities, then it needs high precision sampling. If not then there is not much to gain from higher order sampling of a low SNR signal like GPS. Receivers usually perform one to four bit quantization. But it is known that oversampling (more than Nyquist) is helpful in mitigating the quantization loss [21]. So keeping $B_p = 8$ MHz, is there any advantage in taking $F_s > 4$ MHz?
- One-bit sampling works well for GPS and hence is of interest. One-bit quantization offers a lot of advantages. Like discussed before, one-bit sampling makes many receiver constants independent of the signal strength, thus eliminating the need for AGC. Thus from hardware design perspective one-bit sampling is highly attractive. To analyze one-bit sampling consider,

$$\begin{aligned} \text{sign}[y(t)] &= \text{sign}[x(t) + n(t)] = \text{sign}[x(t)] \\ \text{only if, } |n(t)| &< |x(t)| \end{aligned}$$

This region corresponds to the shaded part in Figure 5.2b. Only samples of $y(t)$ that lie in this region contain any information about the original signal $x(t)$. For GPS/IRNSS $x(t)$ is the PRN sequence and hence it contains information only in its sign, So the one-bit sample in shaded region contains complete information of the signal. The peak height above the wrong correlations in acquisition will depend on the fraction of area in the shaded region. With white assumption on noise, 5.2b illustrates how oversampling may help. Nyquist sampling with one-bit gives the same result, since, on going from B_p to $2B_p$ the variance of noise samples double, but the number of samples per unit time also double. Hence fraction of samples in the desired region remain the same. Figure 5.4 shows simulation results which verify advantage of oversampling. Also, analysis results from previous chapters do not directly extend to low-bit quantization. For instance, one can see that the carrier phase will matter in acquisition at one-bit sampling.



(a) Normal GPS signal capture without interference

(b) GPS signal capture with interference

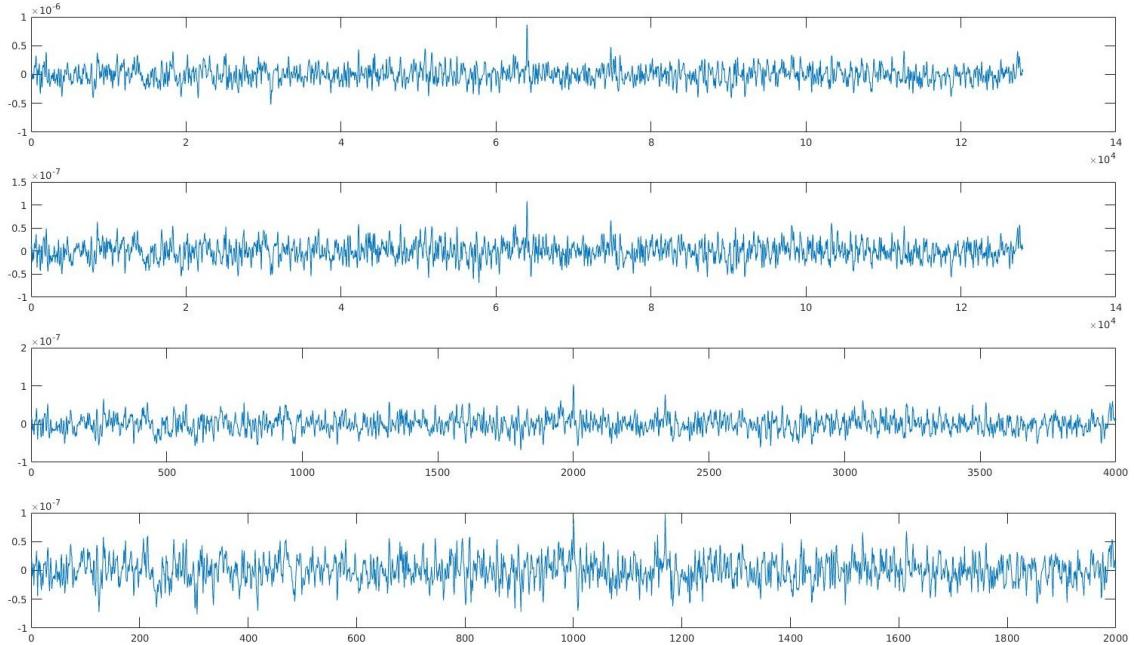
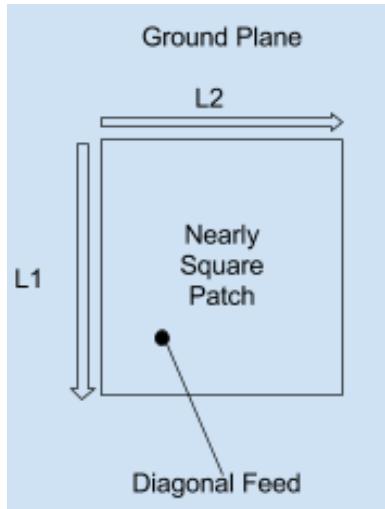


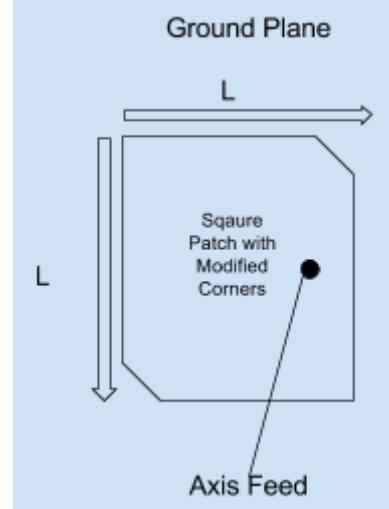
Figure 5.4: B is taken to be 2.046MHz. Acquisition for full precision, one-bit at 128MHz f_s , at 4MHz f_s , at 2MHz f_s . Noise $\sigma = 50$

- Having decided the quantization order, the optimal spacing between quantization levels must be computed. Works like [18] suggest use of non-uniform quantizers at low SNRs, but they add to hardware costs. The sampling can be done at higher precision and then reduced to enable fast computation in a FPGA. In such cases, quantizers specific to the signal can be designed, for example [16].

In conclusion, we note that GPS-like sampling involves interesting theoretical questions [22] [23], which have direct practical implications on the receiver design.



(a) Nearly Square MSA Schematic



(b) Square MSA with modified corners Schematic

5.2 IRNSS S-band Front-End implementation

This section gives details of the S-band receiver designed, implemented and tested based on discussions in the previous section. The main constituents are, the antenna, the LNA, the SAW filter, the RF gain blocks, the mixer and the LO.

Antenna

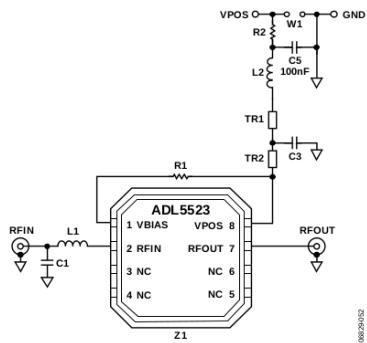
IRNSS signal, like GPS, are RHC polarized, for multiple reasons, including easy multi-path rejection. Thus the receiver must have an RHCP antenna. A linearly polarized antenna can be used as well, but it will result in loss of half the power (3dB loss). Listed below are guideline specifications for a good antenna design of S-band of IRNSS:

- Centre frequency at 2492.028 MHz
- 3 dB Bandwidth of 8 MHz around the center frequency on both sides
- Polarization: Right Handed Circular Polarization
- VSWR lesser than 2 at the center frequency
- Axial Ratio < 3 dB at the center frequency

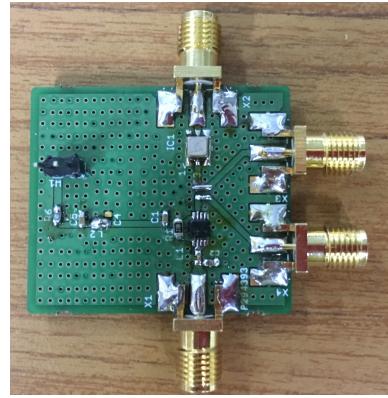
Generally receivers have patch antennas with some asymmetry to make them circularly polarized. The two Micro-Strip Antenna designs shown in Figure 5.5a and 5.5b were used.

LNA, SAW filter and the RF gain stages

Analog Devices Low Noise Amplifier ADL5523 was chosen, with a noise figure of 0.9 dB and 13.5 dB gain in the frequency range of interest. The Saw filter TA1442A which had a center frequency near 2.49 GHz and a 10dB Bandwidth of 100 MHz was used. Along with loss due to SAW filter, the first two components gave a net gain of 9.2 dB. The mixer that was used had a noise figure of 11 dB. According to the thumb-rule net RF chain gain should be about (20 +



(a) LNA typical circuit as given in the datasheet



(b) Fabricated LNA and SAW Filter board. Antenna is to be directly connected to this

11)dB. Two LNA devices of the same kind were used in the gain stage. The measured net gain was $9.2 + 13.2 + 9.2 = 30.68$ dB (though 9.2 dB + 13.5 dB + 13.5 dB was expected).

Mixer and LO

Mixer AD8347 was used. The mixer can provide upto 20 dB RF gain and 19.5 dB IF gain. The gain can be tuned. The maximum measured gain was 38.2 dB and it was used. The net measured gain of the chain until mixer output was 62 dB (includes losses due to cables and SMA connectors). AD8347 also provides AGC and base band conversion options. They were not used in the current receiver design. A Vector Signal Generator VSG25 of Signal Hound was used to provide -8dBm LO input to the mixer.

Testing

WBX-FE Simple R6.0 daughter-board on N210 USRP along with GNURadio as described in Chapter 4 was used. Signals were recorded on the terrace of GG building of EE Department, to have setup directly under the sky and the stay far from WiFi routers. Figure 5.7 shows the signal output from the receiver chain, near a WiFi router. The recorded signal was fed to the software receiver implementation, using the IRNSS C/A codes instead of GPS and $T_{int} = 2\text{ms}$ was used for the acquisition. The results are shown in Figures 5.8a and 5.8b. In some good recordings upto 6 of the 7 IRNSS satellites have been acquired and tracked.

It was observed that setting the gain given by the mixer and the USRP to 0 dB, still allowed for a good acquisition. This might be due to high quality ADC used in the USRP, capable of sensing very low amplitude signals. However, the RF gain is important, and reducing it (by removing a gain stage) results in loss of detection.

Conclusion

The software implementations of acquisition and tracking blocks have thus been tested for actual IRNSS signals as well. The chapter described general considerations involved in designing

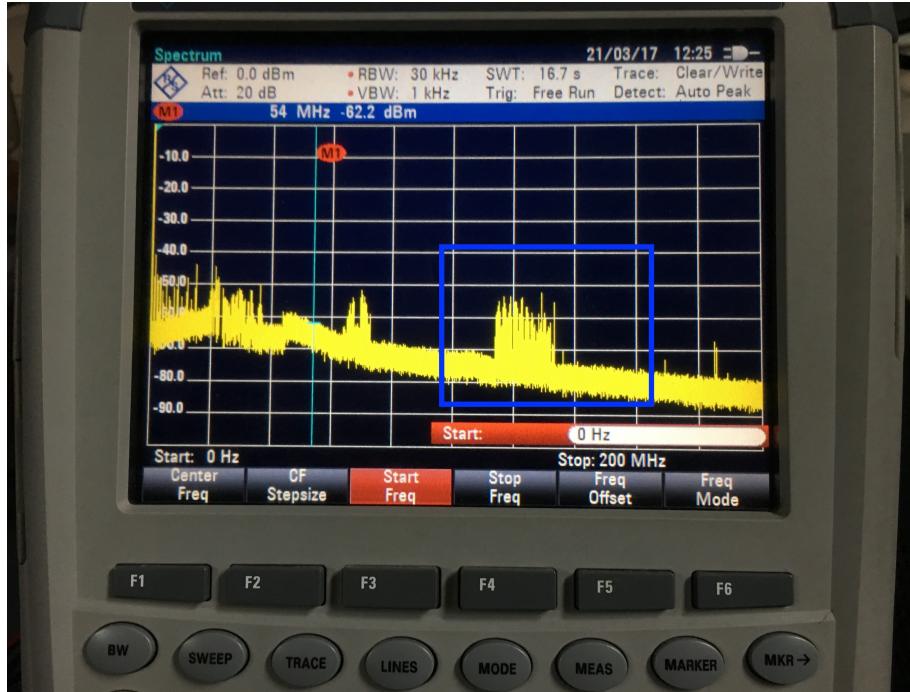
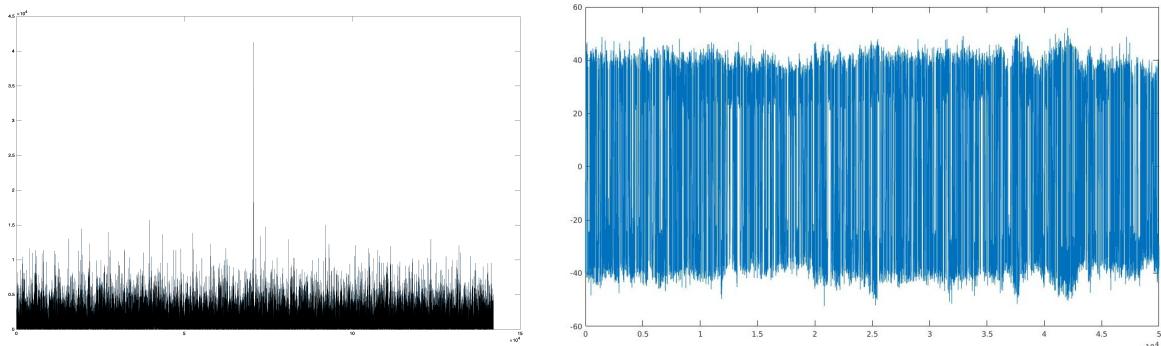


Figure 5.7: WiFi Interference seen using Spectrum Analyzer. This is with higher side injection. Marker locates the frequency to which 2.492028 GHz is down-converted



(a) Acquisition result for IRNSS satellite number 3 (b) Navigation bits from SAT 3, over 50 seconds

the front-end and presented the example of IRNSS S-band receiver front-end implementation. Possible next steps could be to replace the USRP and complete the front-end till ADC, convert the acquisition and tracking blocks into real-time engines on a FPGA and integrate it with the front-end and to add the last layer of software to compute receiver position (or more generally the PVT- position, velocity, time) from measured pseudoranges. The entire setup can be used receive L5 signals as well, which appropriate changes to antenna and the filters. On the other hand, to complete the software implementation of IRNSS receiver, NAV bit processing functions like a constitutional decoded need to be written. Finally, the parts can be put together for a receiver which can effectively use the two bands of IRNSS and the 24×7 visibility of all 7 satellites, to provide accurate positioning and timing services



Bibliography

- [1] Pratap Misra, Per Enge "Global Positioning System, Signals, Measurements and Performance", Ganga-Jamuna Press, Revised Second edition
- [2] Global Positioning System: Theory and Applications, Volume 1, Edited by: Bradford W. Parkinson, James, J. Spilker Jr.
- [3] Understanding GPS: Principles and Applications, 2nd Edition, Elliott D. Kaplan, Christopher J. Hegarty
- [4] Digital Communications by Satellite, J.J. Spilker
- [5] Communication Systems, Simon Haykin, Michael Moher
- [6] Fundamentals of Global Positioning Receivers: A software Approach, James Bao-Yen Tsui
- [7] A Software-Defined GPS and Galileo Receiver: A Single-Frequency Approach, Kai Borre, Dennis M. Akos and others
- [8] Tutorial on Remote Sensing Using GNSS Bistatic Radar of Opportunity, Valery U. Zavorotny et al., IEEE Geoscience and Remote Sensing Magazine (Volume: 2, Issue: 4, Dec. 2014)
- [9] GPS: Primary Tool for Time Transfer, W. Lewandowski et al. (Proceedings of the IEEE, Volume. 87, No. 1, January 1999)
- [10] GPS ICD, <http://www.gps.gov/technical/icwg/IS-GPS-200E.pdf>
- [11] "IRNSS signal in space ICD for Standard Positioning Service version 1.0", *ISRO-IRNSS-ICD-SPS-1.0*, <http://www.isro.gov.in/irnss-programme>
- [12] A. Bhaskaranarayana, Scientific Secretary, Indian Space Research Organisation. <http://www.unoosa.org/pdf/icg/2008/expert/2-3.pdf>
- [13] Raymond L. Pickholtz, Donald L. Schilling. "Theory of Spread-Spectrum Communications-A Tutorial," *IEEE Transactions on Communications*, VOL.COM-30, NO. 5, MAY 1982
- [14] GPS Receiver Architectures and Measurements, Michael Braasch, A.J. Van Dierendonck (1999)

- [15] SDR Joint GPS/Galileo Receiver from Theory to Practice, Marco Rao, Gianluca Falco (2012)
- [16] GNSS Receiver Implementation on a DSP: Status, Challenges, and Prospects, Todd E. Humphreys, Mark L. Psiaki, and Paul M. Kintner, Jr, Brent M. Ledvina (2006)
- [17] Presampling Filtering, Sampling and Quantization Effect on the Digital Matched Filter Performance, Horen Chang, International Telemetering Conference Proceedings (1982)
- [18] At Low SNR, Asymmetric Quantizers are Better, Tobias Koch and Amos Lapidot (2012)
- [19] MATLAB GPS Navigation toolbox <http://in.mathworks.com/matlabcentral/fileexchange/41364-gps-navigation-toolbox>
- [20] A Multi-State Notch Filter for GNSS Jamming Mitigation, Daniele Borio (2014)
- [21] The reconstruction of analog signals from the sign of their noisy samples, E. Masry (1981)
- [22] Sampling with Finite Rate of Innovation: Channel and Timing Estimation for UWB and GPS, Julius Kusuma, Irena Maravic and Martin Vetterli (2003)
- [23] I. Maravic and M. Vetterli, Digital DS-CDMA receivers working below the chip rate: Theory and design, IEEE Transactions on Communications, 2002.
- [24] Influence of Doppler Bin Width on GPS Acquisition Probabilities, Bernhard C. Geiger, Christian Vogel (2013)
- [25] A Statistical Analysis of GNSS Acquisition, Daniele Borio (2008)
- [26] GPS official website, <http://www.gps.gov/systems/gps/control/>
- [27] GNSS-SDR, open source GNSS software receiver. <http://gnss-sdr.org/>
- [28] Inside GNSS, <http://www.insidegnss.com/>
- [29] Andrew Holmes, Home-Made GPS Receiver <http://www.aholme.co.uk/GPS/Main.htm>