GPS Satellite Signal Acquisition and Tracking

Fredrik Johansson, Rahman Mollaei,Jonas Thor, Jörgen Uusitalo

August 21, 1998

Abstract

This report describes the aquisition and tracking of GPS satellite signals using Matlab. Two methods for acquistion are presented - linear search and FFT search. The code tracking loop is a simple delay locked loop and the carrier tracking loop is implemented as a phase locked loop.

Contents

Chapter 1

Introduction

1.1 History

The launch of Sputnik 1957 was the start of a new era in navigation. Through monitoring the Doppler shift in radio signals broadcast from Sputnik, people determined that they could track the satellite. Soon navigators came to the conclusion that the reversed concept could help them locate their own position. This idea led to the deployment of space-based navigation systems. First, the U.S. Navy developed a satellite navigation system, called Transit and then the Russians came with their space based navigation system Tsikada. Transit as well as Tsikada were two dimensional positioning systems with high accuracy. In 1964 Transit became functional for U.S. submarines and then it came available commercial use 1967. These systems were suited for low velocity, surface vehicles, but did not work very well for high dynamic, high velocity vehicles such as aircrafts. Both systems required approximately 10 to 15 minutes of receiver processing to determine position fix. With a project called Timation, atomic precision clocks were introduced on the American navigation satellites. The satellites navigation went through further development under the 1970s which led to the GPS NAVSTAR and GLONASS systems.

1.2 Literature study

Previously there has been much work done regarding methods and techniques in the general area of this project, but books treating the totality of GPS, from a signal processing point of view are quite few. There are books about GPS, but they deal with either specific parts of GPS or other aspects of GPS like navigational, forestry, map, chart, survey etc. However there are two books that are dealing with the totality of the signal aspects of GPS. Those books are *Understanding GPS*, [1], for which Elliot E Kaplan is the editor and *Global Positioning System: Theory and Applications, Volume 1*, [2] edited by Bradford W. Parkinson and James J. Spilker Jr. There is a second volume that is dealing with differential GPS, marine usage of GPS, GPS applications in aviation among other things. *Understanding GPS* is a good source for references, because in some chapters the fundamentals of how everything works is treated and then other chapters are dealing with the subject on a higher level, for people that are more familiar to it. The authors have been describing

the spread spectrum technique, code division multiple access (CDMA), used for the PRN codes of the satellites and the receivers in the GPS system. They have also been treating serial search that is used in the acquisition stage, as well as loop filters and loops used in them, like frequency lock loops (FLL), phase lock loops (PLL), delay lock loops (DLL) and Costas loops, which were utilized in code and carrier tracking. The *Global Positioning System: Theory and Applications, Volume 1* has an even more theoretical approach than [1]. The *GPS Interface Control Document* is also a valuable source of reference [3]. Other sources of reference we would like to point out are our instructor Dr. Dennis M. Akos's Ph.D. thesis [4], that is treating the totality of GPS in the receiver aspect, *Coherent Spread Spectrum Systems* by Jack K. Holmes & Chang C. Chen [5] and *Spread Spectrum Systems with Commercial Application* by Robert C. Dixon [6], *Digital radiokommunikation* by Lars Ahlin & Jens Zander [7], that are dealing with fundamentals of spread spectrum or CDMA aspects. Further there are *Phase-Locked Loops* by Roland E. Best [8] and *Phase-Locked Loops* by John L. Stensby [9], treating PLL and Costas loops. Papers on the subject of serial search are *Analysis of Strategies for Serial Search Spread Spectrum Code Acquisition Direct Approach* by V. M. Jovanovic [10] and *Spread Spectrum Signal Acquisition: Methods and Technology* by Stephan S. Rappaport [11].

1.3   How GPS works

Today, navigation is done, worldwide, with the aid of the Global Positioning System (GPS). The satellites in the system are arranged in six orbital planes with four satellites each, in all 24 active satellites. Each satellite is transmitting its own position (in three dimensions) and time. In order for the receiver to determine its own position it will need data from four different satellites. All GPS receivers are passive, i.e. they do not transmit any signals and depend solely on the received signal for the necessary processing. The receivers make use of one way Time Of Arrival (TOA), which means that they measure the transmission time from satellites to the receivers. The condition of the satellites are being monitored by a ground control network which has its five control stations strategically placed around the earth, (on Hawaii, Ascension Island, Diego Garcia, Kwajalein and Colorado Springs). Position and time data are being updated from the network to the satellites several times every orbit. Only two L band frequencies, L1 (1575.42 MHz) and L2 (1227.6 MHz), are used for broadcast of navigation data and ranging codes from the satellites, by the use of code division multiple access (CDMA) and each satellite uses a different ranging code. The navigation data provides the receiver the satellite location at the time of transmit. The ranging code may be used by the receiver to determine the propagation delay of the satellite signal. Only the L1 carrier frequency is used by the standard positioning service (SPS), which is available to all users and it provides the users with navigation accuracies, in the horizontal plane, of 100 m and in the vertical plane with accuracies of 156 m. The precise positioning service (PPS) is only available to U.S. government and military users. The PPS is using both the L band frequencies and is providing accuracies, in the horizontal plane, of 22 m and in the vertical plane of 27.7 m. To prevent public access to the GPS PPS, two cryptographic features are used, Antispoofing (AS) and Selective Availability (SA). AS is utilized to prevent deception jamming, which is a technique for the enemy to replicate satellites PRN codes, navigation data and carrier frequencies to confuse one's own receivers. SA is utilized to diminish accuracy available to SPS users. SA decreases TOA accuracy, by distorting the satellite's time. SPS was available for use in December 1993, (the set of 24 satellites contained some prototype units) and PPS had gained full capacity with 24

production satellites in spring 1995. The facts are derived from [1].

Chapter 2

GPS SPS Signal Characteristics

In this chapter the modulation format and the ranging code for GPS SPS will be described. As previously mentioned GPS satellites transmits at two frequencies, L1 and L2. The L2 carrier is spread in frequency by a precision code called P(Y) code. The L1 carrier is modulated by two ranging codes; the P(Y) code and a coarse/acquisition (C/A) code. The P(Y) code is available for PPS users, while the C/A code also is available for SPS users. Only the L1 carrier and the C/A codes will be described here.

2.1   Frequency and Modulation Format

A simplified model of the L1 SPS modulator configuration is shown in figure 2.1. The L1 carrier is modulated by the modulo-2 sum of the C/A code and the navigation data. The modulo-2 sum is the same operation as the exclusive-or operation, denoted by $\oplus$. The nominal chipping rate of the C/A code is 1.023 MHz and one code period is 1023 chips long or 1 ms. It is important to notice that the transitions of data bits are synchronized to code periods, but at a much slower rate (50 Hz).

Before biphase shift keying (BPSK) modulation is employed, the above described modulo-2
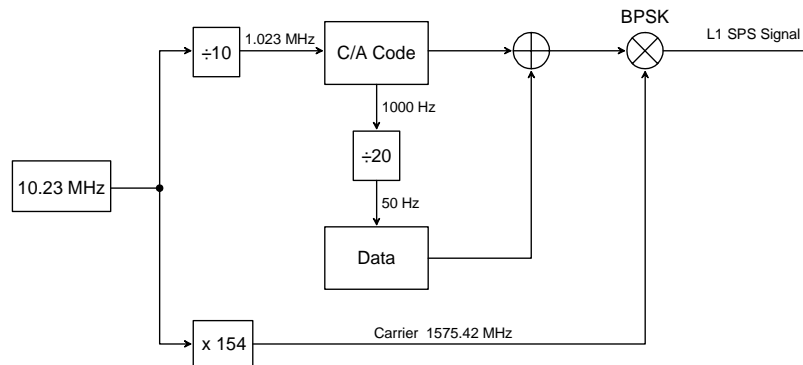

Figure 2.1: Simplified GPS SPS modulator configuration.

sum with values $\{0, 1\}$, are mapped to $\{-1, 1\}$. The transmitted L1 SPS signal is defined by

$$L_i(\omega_1 t) = A\left[CA_i(t) \oplus D_i(t)\right]\sin(\omega_1 t), \tag{2.1}$$

where: L_i broadcast signal from the ith satellite
        i indicates satellite number
        A signal amplitude
        CA_i C/A code for the ith satellite
        D navigation data for the ith satellite
        ω_1 L1 radian freqeuncy $(2\pi 1.57542 \cdot 10^9$ rad/s$)$.

2.2   C/A Code Generation and Properties

Each satellite uses a unique C/A code to implement the CDMA technique. The C/A codes belong to the family of Gold pseudorandom noise (PRN) codes. Gold codes are used because of their autocorrelation and cross-correlation properties. The architecture of a C/A code generator is shown in figure 2.2. Two 10-bit linear feedback shift registers (LFSR), G1 and G2, generate maximum length pseudo random codes with a length of $2^{10} - 1 = 1023$ bits. Initially, both G1 and G2 are all set to ones, since the all-zero state is illegal. The G1 and G2 LFSRs feedback taps are defined by the generator polynomials

$$\begin{aligned}
G1(X) &= 1 + X^3 + X^{10} \\
G2(X) &= 1 + X^2 + X^3 + X^6 + X^8 + X^9 + X^{10}.
\end{aligned} \tag{2.2}$$

The satellite specific C/A code is generated by taking the exclusive-or output of the G1 LFSR and a delayed version of the output from the G2 LFSR. The delayed effect for the G2 LFSR output is obtained by the exclusive-or of two selected stages from the G2 LFSR. This will work since adding a phase-shifted versions of a PRN sequence to itself will shift the phase of the code, while not changing the code [1]. The GPS code phase assignments are documented in [3]. See appendix A.1 for the MATLAB implementation of the C/A code generator.

The autocorrelation function of a C/A code is

$$R_{CA}(\tau) = \int\limits_{-\infty}^{\infty} CA_i(t)\,CA_i(t + \tau)\,dt, \tag{2.3}$$

where CA_i is the C/A code for the ith satellite and τ is phase of the time shift. A normalized and simplified autocorrelation function of a typical C/A code is shown in figure 2.3. The correlation peaks repeats every code period and the correlation interval is two chips. As mentioned this is a simplified model of the C/A code autocorrelation function since there are shifts which produce non-zero values between correlation peaks. The properties of the autocorrelation function is used to synchronize the receiver replicated code with the received code. It is important that the cross-correlation of any two C/A codes are minimal for any phase or Doppler shift over the entire code period. The ideal cross-correlation is defined by:

$$R_{ij}(\tau) = \int\limits_{-\infty}^{\infty} CA_i(t)\,CA_j(t + \tau)\,dt = 0 \tag{2.4}$$
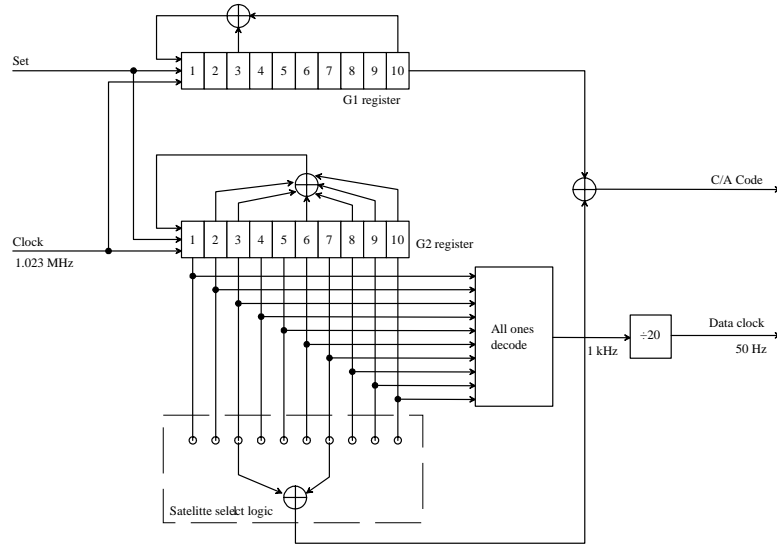
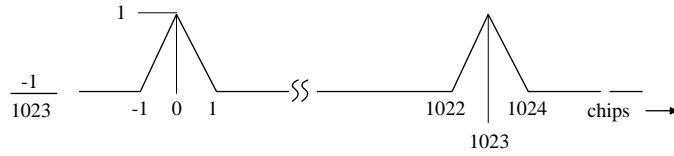Figure 2.2: C/A code generator.

Figure 2.3: Normalized and simplified autocorrelation function of a typical C/A code.

where CA_i is the C/A code for the ith satellite and CA_j C/A code for the jth satellite where $i \neq j$. Due to the fact that multiple satellites signal are received with different delays and Doppler offsets the cross-correlation properties of C/A codes are not ideal, and may under certain conditions cause false acquisitions. For further information about C/A code properties refer to [2].

Chapter 3

Satellite Signal Acquisition

The process of receiving GPS signals may be divided into three steps.

1. **Acquisition**. The signal acquisition process consist of a three-dimensional search in time (code phase), frequency and satellite-specific PRN code. In the following sections two different search methods will be described: serial search [11] and FFT search [12].

2. **Tracking.** After the acquisition process is accomplished the receiver enters the tracking mode. A delayed locked loop (DLL) is used to track the code phase (or frequency) and a phase locked loop (PLL) or a frequency locked loop (FLL) is used to track the carrier phase or frequency, respectively. The MATLAB implementation uses a simple DLL for the code tracking loop and a Costas PLL for the carrier tracking loop. Next chapter deals with tracking.

3. **Data decoding and position solution**. When both tracking loops are in lock it is possible to decode the 50 bps data and determine the receivers position. How the data bits are obtained will be shown, but as to how to implement the parity and position solution algorithms refer to [2].

Real GPS data was used to test the Matlab receiver. The L1 carrier in the data is down-converted to an intermediate frequency (IF) of 1.26 MHz and is sampled at 5 MHz.

3.1   Serial search

Serial search is the simplest and most frequently used acquisition algorithm. A non coherent correlator, shown in figure 3.1, is used since the phase of the received signal is random. First the digital IF, $x[n]$, is multiplied by the replicated CA code, $CA[n+m]$. Here n represents the nth sample and m represents the number of samples the replicated CA code is phase shifted. Since we have a sampling rate of 5 MHz the length, L, of one C/A code period is 5000 samples nominally.

After the code removal the in-phase (I) and quadrature (Q) components are generated. The I and Q components are accumulated for one or more code periods, N. The accumulated sum is squared. Next K correlations are accumulated to produce an averaged correlation
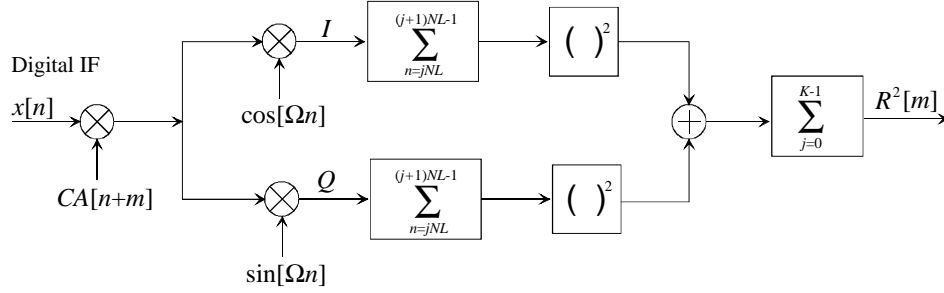
Figure 3.1: Non coherent correlator in time domain.

point. If the correlation point is larger than a certain threshold it is assumed that the satellite is acquired. The correlation is approximated by the discrete sum:

$$R^2\left[m\right] = \sum_{j=0}^{K-1} \left(\left[\sum_{n=jNL}^{(j+1)NL-1} x\left[n\right] \cdot CA\left[n\right] \cdot \cos\left[\Omega n\right] \right]^2 + \left[\sum_{n=jNL}^{(j+1)NL-1} x\left[n\right] \cdot CA\left[n\right] \cdot \sin\left[\Omega n\right] \right]^2 \right),$$
(3.1)

where Ω is radian frequency.

In the MATLAB implementation, appendix A.2, $N = 1$ and $K = 1$, since this provides the fastest acquisition. However choosing larger values for N and K will improve the acquisition of weak satellite signals and will also reduce the probability of false acquisition. One problem with choosing larger values for N is that is possible that a data signal transition can occur during the correlation process. Choosing a large N will however improve the signal to noise ratio since the accumulation is a low pass process. The data transitions are synchronized to the code periods and can occur at maximum every 20th code period. If N is even it is possible that a data bit transition will cause the dumped sum to be only noise. Choosing $K > 1$ will not be sensitive to data bit transitions, but the expected value of the noise will not be zero since the noise is squared. This must be considered when the detection threshold is selected.

Usually when serial search is performed the code phase is incremented $\frac{1}{2}$ chip[1] when one code phase has been tested. So there are $2 \times 1023 = 2046$ code phases that must be tested. The MATLAB implementation however, shifts the code phase by one sample between each dump. One sample corresponds to approximately $1/5$ chip so here 5000 code phases are tested. 10000 samples of received data is correlated by sliding the replicated code over the 10000 samples, as illustrated in figure 3.2.

Complicating things, the Doppler effects on the carrier must be taken into account. For a stationary receiver it is common to assume a ± 10 kHz Doppler offset on the carrier. The acquisition process for each code period is stepped in the Doppler range in 500 Hz frequency increments. Choosing 500 Hz steps is a compromise in accuracy and speed. If the step was 1 kHz there would be less frequency bins to test, but with the possibility that the residual carrier of the I and Q components can cancel out the correlation, since 1 kHz corresponds to the code period of 1 ms.

---

[1] The reason for chosing a 1/2 chip increment is obvious when examining the autocorrelation function of one typical C/A code
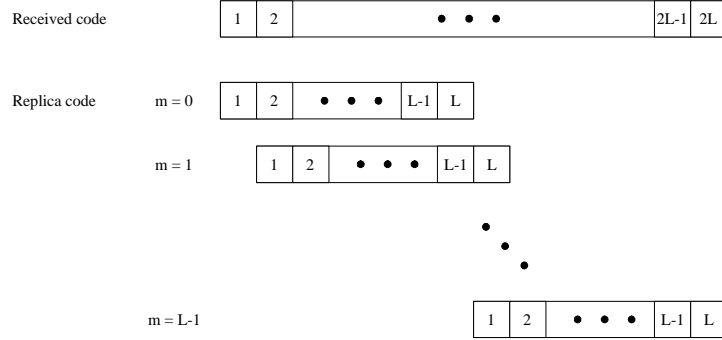
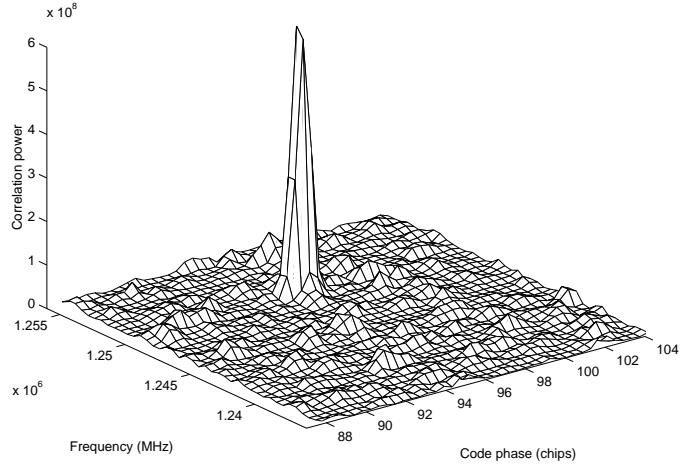Figure 3.2: Linear search algorithm.

Figure 3.3: Correlation matrix for satellite 17.

The MATLAB implementation utilizes an exhaustive search, that is, all possible combinations of code phases and Doppler offsets are tested and a correlation matrix is generated. A real receiver would set a threshold and transition to tracking when the threshold is crossed.

If there are some a priori information about either the carrier or code phase distribution it is possible to use other more efficient serial search methods [10] than, in the MATLAB implemented, linear search algorithm. Linear search simply increments in the code phase and Doppler frequency linearly. Figure 3.3 illustrates a small subset of the correlation matrix that was obtained when satellite 17 was searched for.

3.2 FFT search

The linear search algorithm can also be performed as illustrated in figure 3.4. Here 5000 samples of the received data are correlated with the replica code by circularly shifting the
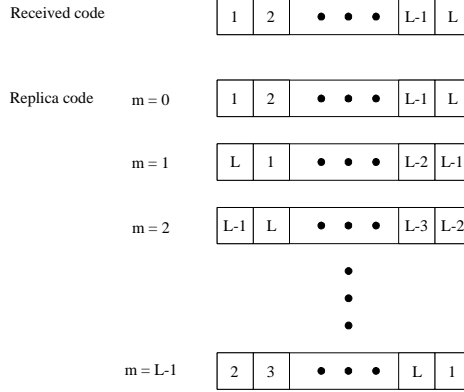
Figure 3.4: Circular search algorithm.

replica code. This resembles the circular convolution and may be expressed as

$$R\left[m\right] = \sum_{n=0}^{L-1} x\left[n\right] \cdot CA\left[\left(\left(n+m\right)\right)_{L}\right]. \tag{3.2}$$

This was not done in the serial search implementation since it is possible that a data bit transition may occur in the 5000 samples of received data. However there is one advantage with this approach. The circular convolution is a multiplication in the frequency domain. In fact we can write

$$R\left[m\right] = \underbrace{x\left[n\right] \otimes CA\left[-n\right]}_{\text{Circular convolution}} = \mathcal{F}^{-1}\left(\mathcal{F}\left(x\left[n\right]\right) \cdot \mathcal{F}\left(CA\left[n\right]\right)^{*}\right) \tag{3.3}$$

where the discrete Fourier transform (DFT) and its inverse is used to calculate R. This can be adopted to the acquisition of GPS signals as shown in figure 3.5 and the MATLAB code listed in appendix A.3. Here the incoming signal is mixed to baseband and the in-phase and quadrature components are used as the real and imaginary inputs when calculating the DFT. The result is multiplied by the complex conjugate of DFT of the C/A code. The circular convolution is obtained by taking the magnitude of the inverse DFT as seen in figure 3.5. The fast Fourier transform algorithm is used to implement the DFT and IDFT, hence the acquisition method may be called FFT search.

Nee et al [12] claims that FFT search shows a theoretical reduction in acquisition time of about 2000 times. However serial search is mostly used in commercial receivers because of the simple implementation.

The FFT search algorithm is used in the MATLAB implementation, because of its significant fast acquisition speed, compared to serial search. The parameter K, which is similar to the K described in the previous section, was first set to 3 and this resulted in that six satellites were found in the available data. When K was increased to 10, two more satellites were found as well. Figure 3.6 illustrates how the signal to noise ratio inreases as K increases. For clarity the correlation powers in figure 3.6 are normalized and the acquisition is performed at a fixed replica carrier frequency.

Appendix A.4 shows a MATLAB function that generates a correlation matrix, $\mathbb{C}$, by performing the FFT search over the Doppler range and K sequential correlations. A metric,
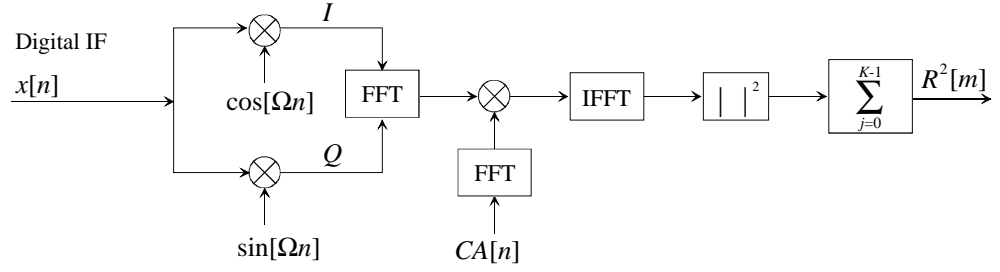
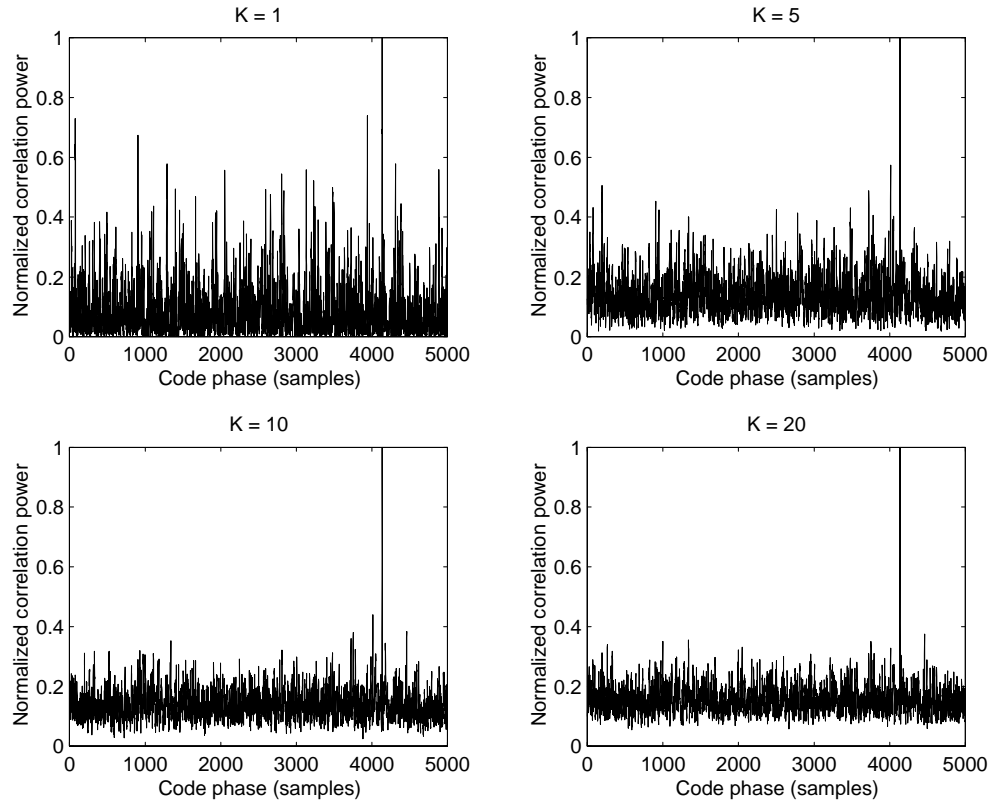Figure 3.5: Non cohorent correlator in frequency domain.

Figure 3.6: Aquisition of satellite 24 at IF 1.2445 MHz.

M, is calculated by dividing the maximum value of $\mathbb{C}$ by the avarage value of all elements in $\mathbb{C}$. This metric, M, is tested against a threshold, T, and if $M > T$ the satellite is assumed to be acquired. Simplifying things, the expected value of M is,

$$E[M] \approx \frac{K \cdot E[\mathcal{N}] + K \cdot E[\mathcal{S}]}{K \cdot E[\mathcal{N}]} = 1 + \frac{E[\mathcal{S}]}{E[\mathcal{N}]}, \tag{3.4}$$

if a satellite signal is present, and

$$E[M] \approx \frac{K \cdot E[\mathcal{N}]}{K \cdot E[\mathcal{N}]} = 1, \tag{3.5}$$

if a satellite signal is not present. $E[\mathcal{N}]$ is the expected value of the squared noise and $E[\mathcal{S}]$ is the expected value of the correlation power, when the satellite signal is approximately in phase with the replica code and carrier. As K increases the variance of M decreases, thus M can be used to detect the satellite signal.

Table 3.1 shows the code phase and carrier frequencies for the satellite signals that were found in the available data. The parameters K and M were set to 10 and 6, respectively.

Satellite	Carrier (MHz)	Code Phase (samples)
6	1.2460	2884
10	1.2450	3814
17	1.2505	469
23	1.2500	2200
24	1.2445	4136
26	1.2475	2664
27	1.2470	2844
28	1.2500	3269

Table 3.1: Satellite Acquisition

Chapter 4

Satellite Signal Tracking

As we saw in the previous chapter the *Matlab* receiver performed an acquisition search to find the frequency and the phase of the PRN code for a specified satellite. The frequency and the phase are the parameters that are needed to decode the CDMA signal and get access to the satellite data. Some problems can appear though. The most obvious problem is when using a non-stationary receiver. If the user system is moving in a considerable speed and is frequently changing direction, he will experience a Doppler shift which will alter the experienced carrier frequency of the satellite. This may cause the receiver to loose the satellite signal.

A similar, though rather more static problem is the geographical Doppler shift induced by the satellite. As the satellite moves across the sky it will induce a Doppler shift of it's own. The maximum deviation seen from this phenomena on a stationary receiver is approximately ± 10 kHz. If these frequency deviations are not corrected for, the receiver will eventually lose track of the satellite and no data will be decoded.

The other receiver problem is to keep the phase in PRN code aligned. Since the PRN code is upsampled five times, each chip consists of five equal taps. If we have a PRN code of 1011 the upsampled signal would be as illustrated in figure 4.1
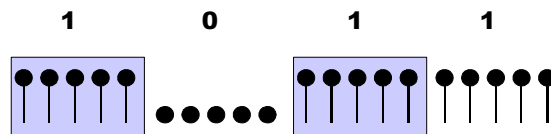

Figure 4.1: Sampled C/A code.

Now, if the phase is completely aligned we will have maximum correlation. But even if we are wrong with a few taps we will still have a considerable correlation. Therefore we have reason to believe that the phase found in the acquisition may not always give us the exact phase that should be applied to the PRN code. Still, having a perfectly aligned PRN code, the phase would soon change as a result of Doppler shift and without any corrections we could easily within 200 ms be completely out of phase. This will occur when the code difference is larger than one chip.

These two problems, the frequency shift and the phase shift has to be corrected for while receiving and leads us on to the next step - Tracking the signal.

Since the tracking part consists of dealing with two, even though dependent on each other, different problems we have divided this chapter into two parts, Code Tracking and Carrier Tracking. We will start with Code Tracking since this was the first part we implemented.

4.1 Code Tracking

In the code tracking we want to compensate the phase shift in the PRN code for any disturbances that might occur. Remember the five times upsampled PRN code that we have. Suppose that we have a total synchronous receiver that suddenly loses phase of one tap;
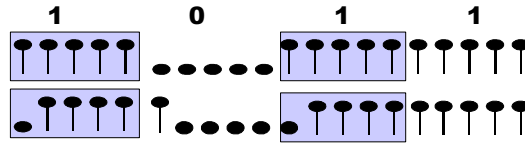

Figure 4.2: Received C/A code and replica code out of phase.

This will cause the correlation to drop, but this information alone is not enough to compensate for the phase shift that occurred.. Even if we knew that the correlation drop was only because of a phase shift we would not know whether to shift the PRN code left or right.
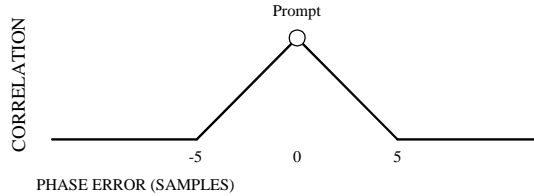

Figure 4.3: Prompt Replica Code Correlation Plot

As we can see in the figure 4.3 above, the correlation ideally slides down linearly from maximum correlation to zero correlation at five samples, that is when the phase error is one full chip. If the phase error becomes one full chip there should not be any correlation since the PRN code is supposed to act like white noise which is uncorrelated.

So if a shift occurs the mark will slide to the right or left, depending on the nature of the shift. In order to be able to determine the direction of the slide we introduce two additional PRN codes according to (1). The new PRN codes are called early and late, the original PRN code is from now on called prompt and is the one we will try to keep phase aligned.

The early PRN code is the prompt PRN code shifted 3 samples to the right and the late PRN code is shifted 3 samples to the left. This gives us a correlation/phase-error plot like
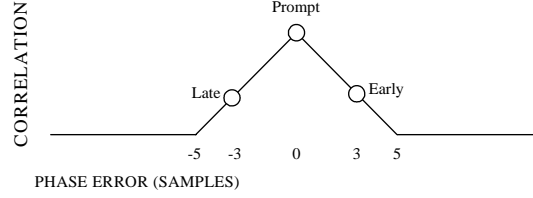
15

in figure 4.4.

Figure 4.4: Prompt/Early/Late Replica Code Correlation Plot

If the incoming PRN code shifts in any direction, as illustrated in figure 4.5, the marks will all move to the same direction and the different correlations will tell us which way to shift our prompt PRN code.
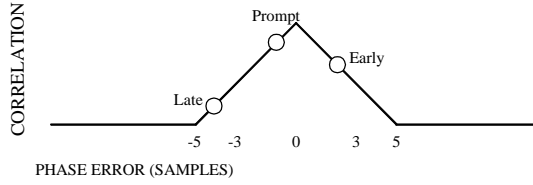

Figure 4.5: Correlation Plot With Phase Error

The discriminator algorithm is to give a calculated number on the phase error that is present in the receiver that is implemented. The discriminator output signal, ϵ, is calculated;

$$\epsilon = \frac{Early}{Late} \tag{4.1}$$

This discriminator will give us

$\epsilon = 1,$ when we are perfectly aligned,
$\epsilon > 1,$ if we should shift the code to the right,
$\epsilon < 1,$ if we should shift the code to the left.

Since we know that the the data given to us is from a stationary receiver and have a maximum doppler shift of approximatley ± 10 kHz/s, we can calculate the maximum phase deviation per code period as,

$$\Delta f_{CA\,\max} = \Delta f_{c\,\max} \frac{f_c}{f_{CA}} \approx \pm 6.5 \text{ chips/s}, \tag{4.2}$$

where $\Delta f_{c\,\max}$ is maximum Doppler shift on carrier, f_c is carrier frequency and f_{CA} is the code chipping rate. So to avoid jerky corrections, due to noice, we average over twenty code periods, which should be often enough to correlate the phase deviation before it reaches a full chip. The worst chip deviation we can obtain per twenty code periods is $0.02 \cdot 6.5 = 0.13$ chip.

16

The Structure.

We first constructed the code tracking without carrier tracking. The block schematic for this code-only tracker is shown in figure 4.6
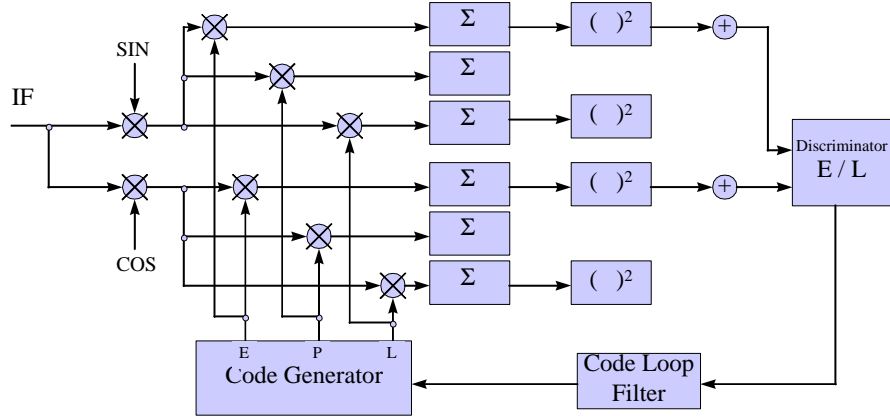

Figure 4.6: Code Tracking Schematics

The IF signal is first downconverted by the carrier frequency to the baseband in I and Q components. The I and Q are multiplied by the early, late and prompt PRN codes to get the correlation values. The values are summed and averaged over 20 code periods or 20ms. The correlation values are then fed into the discriminator algoritm which gives the early/late relationship to the code loop filter which determines if there should be a phase change in the local prompt PRN code.

The I and Q components are made by;

$$I = data \cdot \sin(f_c 2\pi t)$$
$$Q = data \cdot \cos(f_c 2\pi t)$$
(4.3)

Where fc is the carrier frequency. The prompt PRN code is generated with our own cacode generator. Then the early and late are made by shifting the prompt signal 3 samples.

The sum is first made over on code period and then the envelope of 20 periods is summed. The equations below are for early only.

$$Ie = \sum(early * I)$$
$$ee = \sqrt{(Ie^2 + Qp^2)}$$
(4.4)

The sum of the envelopes is fed into the discriminator and then the code loop filter takes the decision if a phase change should occur.

$$\epsilon = \frac{E}{L}$$
(4.5)

17

In the following figure, which is a plot from Matlab, we see how the code tracking loop keep the prompt signal at maximum correlation. Without this code tracking loop the prompt signal would quickly fall down to zero correlation. The implementation done will accept a small deviation in the phase and will not correct immediately. This is why we see in the plot a small deviation in the prompt signal, the signal is allowed to fall a certain amount before it is corrected.
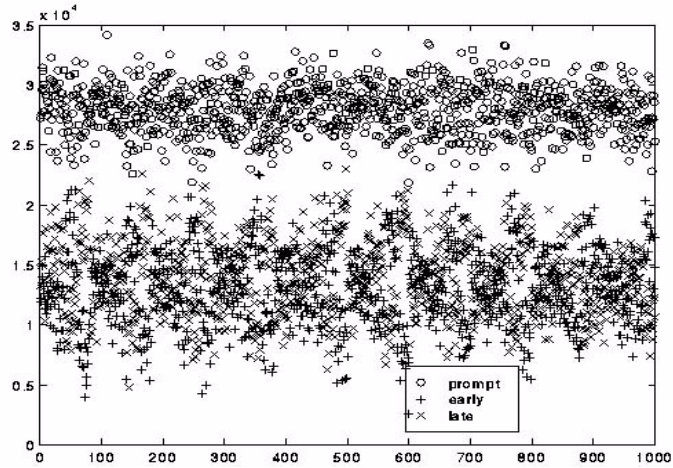

Figure 4.7: Correlation power of early, prompt and late channel with delayed locked loop.

Now that the code loop is working sufficiently, we focused on keeping the frequency locked in the Carrier Tracking.

4.2 Carrier Tracking

When the carrier frequency changes the receiver will loose track of the satellite signal. The carrier tracking loop tries to keep phase lock of the satellite as long as the frequency drift is within a given boundary. The acquisition will give the carrier frequency within ± 500 Hz, the carrier tracking must then first bring the carrier frequency into lock. We approached the carrier tracking problem by implementing a PLL, Phase Lock Loop. The basic PLL structure is shown in the figure below;

However we made some changes to the architecture. Since we have an integrator that integrates over one PRN code period we get the function of a low pass filter with a stop frequency of 1 kHz. Because of that we leave out the specified low pass filter which is only there to filter out high frequency noise anyway. This is the block schematic for the Carrier-only Tracking;

The Intermediate Frequency signal is first collapsed to the baseband in form of I-component and Q-component using the carrier frequency fc (this is the frequency that will be corrected). The baseband signals are then multiplied with our prompt replica code and are summed over one period. The I and Q signal are then fed into the discriminator which gives us the
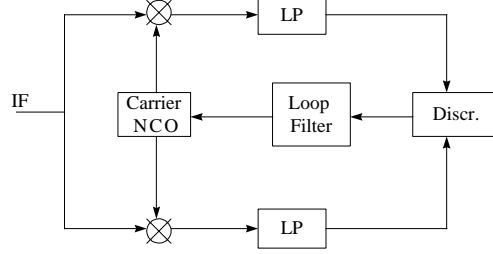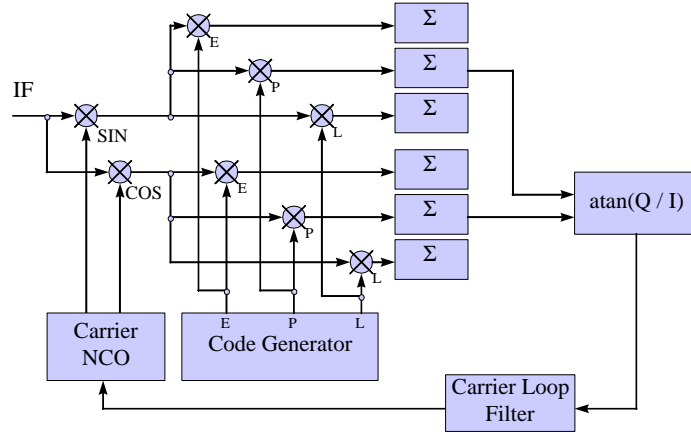
Figure 4.8: Generic PLL Schematics

Figure 4.9: Carrier Tracking Module Schematics

angle in radians between the I-component and the Q-component. What we want is to have the I-component as large as possible while keeping the Q-component as small as possible. So this yields that we should try to keep the discriminator angle as small as possible. The discriminator uses atan instead of atan2 in order to ignore the phase shifts due to the BPSK-modulation. The angle goes through the carrier loop filter to the carrier frequency NCO, in order to correct fc.

The correction of fc is made by shifting the phase in a successive way. The corrective phase is calculated from;

$$\theta_{corr} = remainder(\theta_{old} + \theta, 2\pi)$$

Which adds the phase to the last phase correction and removes any full revolutions of 2π.

The design of the loop filter was taken from the book [1]. The filter we are using is a first order filter. The time discrete description in the z-domain;

$$F(z) = \frac{(C1 + C2) - C1 * z^{-1}}{1 - z^{-1}} \tag{4.6}$$

19

The loop filter was implemented is such way that we only give the bandwidth, and damp parameters as input. Then the program calculates the coefficients $C1$ and $C2$ which are used in the calculations in the loop filters. The calculations for $C1$ and $C2$;

$$C1 = \frac{1}{K} \frac{8\zeta\omega_n\Delta T}{4 + 4\zeta\omega_n\Delta T + (\omega_n\Delta T)^2}$$
$$C2 = \frac{1}{K} \frac{4(\omega_n\Delta T)^2}{4 + 4\zeta\omega_n\Delta T + (\omega_n\Delta T)^2} \tag{4.7}$$

Where

$$\omega_n = \frac{2\beta}{\zeta + \frac{1}{4\zeta}}$$
$$\Delta T = \frac{1}{1000}s \tag{4.8}$$

We choose the parameters as:

$$\beta = 50Hz, \text{ Bandwidth}$$
$$\zeta = 0.7, \text{ Dampening}$$
$$K = 400\pi$$

Which gave us a satisfactory filter response.

This PLL works out fine with the data given to us. The only problem occurs when the acquisition is not close enough to the true carrier frequency. This is overcome by performing a linear fine grained (5 Hz) frequency search at the fixed code phase that was obtained from the acquisition process. This will have the effect that the receiver pulls in closer to carrier IF. See appendix A.5 for the MATLAB implementation. Another, more elegant, solution to this problem [2] is to close the carrier tracking loop using an FLL and a wideband loop filter. The carrier loop will transition into a Costas PLL, with a narrowband loop filter, when the FLL has pulled in the frequency close enough.

Finally we put together carrier and code tracking in one function called track.m. This file will deal with both code tracking and carrier tracking. As track.m is written as a function we can call this with the frequency and phase given from the acquisition. The schematic looks like figure 4.10.

The data from the satellite is taken from the prompt I-component before it's fed into the atan discriminator. The prompt I-component for satellite 17 in the data file is illustrated in figure 4.11.

Since the modulation is simple BPSK, we simply take the sign of the I signal.

$$Data = \begin{array}{ll} 1, & I \geq 0, \\ -1, & I \leq 0. \end{array} \tag{4.9}$$

What we do not know now is whether a 1 in the data stands for a 1 or a 0. To find this out we check the parity bits. If they do not match we invert the data.
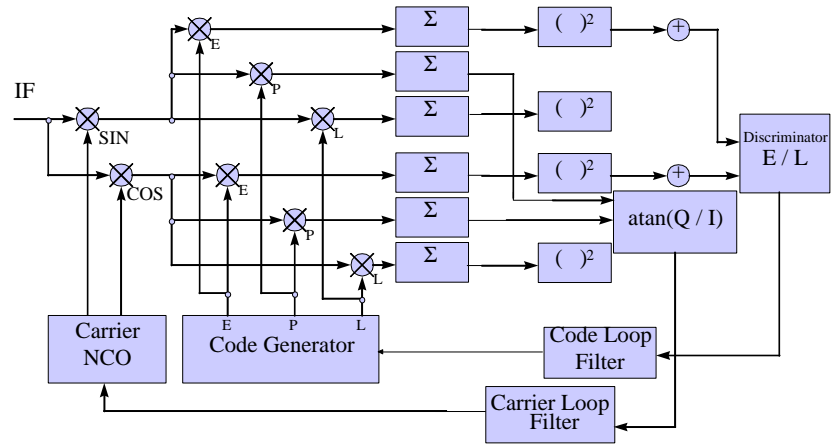
Figure 4.10: Tracking Module Schematics
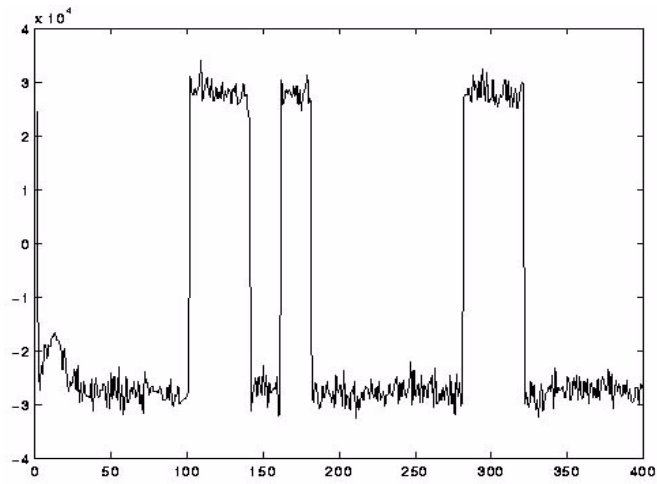
Figure 4.11: Prompt I-Component

Chapter 5

Conclusions

The Global Positioning System is a complex system, which offers many different signal processing areas of research. In this project we have limited the work to only a small subset of GPS which is the acquisition and tracking of GPS satellite signals.

We have presented two different acquisition methods - linear search and FFT search. In terms of acquisition speed, FFT search outperforms serial seach.

The code and carrier tracking methods desribed are simple, and there exist better and more accurate implementations [1, 2]. Given addition time the next step would be the decoding of the navigation data and the position solution.

Bibliography

[1] Kaplan, Elliott D.,Editor, *Understanding GPS : Principles and Applications*, Artech House, ISBN: 0890067937, March, 1996.

[2] Parkinson, Bradford W. and Spilker, James, J. Jr, Editors, *Global Positioning System: Theory and Application, Volume 1*, American Institute of Astronautics and aeronautics, ISBN: 156347106X, June 1996

[3] *ICD-GPS-200 (GPS Interface Control Document)*, GPS Joint Program Office (prepared by ARINC Research), July 1991.

[4] Akos, Dennis M. , *A Software Radio Approach to Global Navigation Satellite System Receiver Design*, Ph.D. Dissertation, Ohio University, 1997.

[5] Holmes, Jack K., *Coherent Spread Spectrum Systems*, New York, ISBN: 0-471-03301-4, 1982.

[6] Dixon, Robert C., *Spread Spectrum Systems with Commercial Application*, 3rd. ed., New York: John Wiley & Sons, ISBN: 0-471-59342-7, 1994.

[7] Ahlin, Lars and Zander, Jens, *Digital Radiokommunikation-System och Metoder*, Lund: studentlitteratur, ISBN: 91-44-34551-8, 1992.

[8] Best, Roland E., *Phase-Locked Loops: Theory, Design and Applications*, 3rd. ed., New York: McGraw-Hill, ISBN: 0-07-006051-7, 1997.

[9] Stensby, John L., *Phase-Locked Loops: Theory and Applications*, Boca Raton: CRC Press, ISBN: 0-8493-9471-6, 1997.

[10] Jovanovic, V. M., *Analysis of Strategies for Serial-Search Spread-Spectrum Code Acquisition-Direct Approach*, IEEE Transactions on Communications, Vol. COM-36, No. 11, Nov. 1988, pp. 1208-1220.

[11] Rappaport, Stephan S., *Spread Spectrum Signal Acquisition: Methods and Technology*, IEEE Communications Magazine, June 1984, Vol. 22, No. 6, pp. 6-21.

[12] van Nee, D. J. R. and Coenen, A. J. R. M., *New Fast GPS Code-Acquisition Technique using FFT*, Electronic Letters, 17th January 1991, Vol. 27 No. 2, pp. 158-160

Appendix A

Matlab Code

A.1   C/A Code Generator

```
function CA = cacode(n, chiprate, fs, n_samples)
% The funtion
%    CA = cacode(n, chiprate, fs, n_samples)
% returns the Gold code for GPS satellite ID n
% n = 1..32
% chiprate = number of chips per second.
% fs = sampling freqeuncy
% n_samples = number of samples
% The code is represented at levels : -1 for bit = 0
%                                      1 for bit = 1

% phase assignments
phase=[2 6; 3 7; 4 8; 5 9; 1 9; 2 10; 1 8; 2 9; 3 10;
       2 3; 3 4; 5 6; 6 7; 7 8; 8 9; 9 10; 1 4; 2 5;
       3 6; 4 7; 5 8; 6 9; 1 3; 4 6; 5 7;  6 8; 7 9;
       8 10; 1 6; 2 7; 3 8; 4 9];

% Intial state - all ones
G1 = -1*ones(1,10);
G2 = G1;
% Select taps for G2 delay
s1 = phase(n,1);
s2 = phase(n,2);
tmp=0;
for i=1:1023;
   % Gold-code
   G(i)=G2(s1)*G2(s2)*G1(10);
   % Generator 1 - shift reg 1
   tmp=G1(1);
   G1(1)=G1(3)*G1(10);
   G1(2:10)=[tmp G1(2:9)];
```

```
    % Generator 2 - shift reg 2
    tmp=G2(1);
    G2(1)=G2(2)*G2(3)*G2(6)*G2(8)*G2(9)*G2(10);
    G2(2:10)=[tmp G2(2:9)];
end;

% Resample - doesn't work for (i*chiprate/fs)>1023
% but replica chiprate is constant in this
% implementation

i=1:n_samples;
CA(i) = G(ceil(i*chiprate/fs));
```

A.2   Serial search

```
function C = serial(data, sat_id, ch_rate, fs, fc_lo, step, fc_hi, n_data)
%
% The function
%   C = serial(data, sat_id, ch_rate, fs, fc_lo, step, fc_hi, n_data)
% performs a serial search on the data and returns a correlation matrix.
%
%   data = the data to be processed
%   sat_id = the satellite to be searched for
%   ch_rate = the C/A code chippingrate (1.023 MHz)
%   fs = the sampling frequency
%   fc_lo = IF freq minus max Doppler offset
%   fc_hi = IF freq plus max Doppler offset
%   step = size of one Doppler bin
%   n_data = number of data samples (typically 10000 samples, must be even)

% Generate CA code
ca = cacode(sat_id, ch_rate, fs, n_data/2);
% Index for correlation matrix
doppler_index=0;
% Time vector
t = (0:(n_data-1))/fs;
% Doppler loop
for fc = fc_lo:step:fc_hi
    doppler_index=doppler_index+1;
    % Downmix to baseband
    I_comp = sin(2*pi*fc*t).*data';
    Q_comp = cos(2*pi*fc*t).*data';
    % Code phase loop
    for phase_index=1:(n_data/2)
        % Accumulate I and Q
        I = sum(I_comp(phase_index:phase_index+(n_data/2-1)).*ca);
        Q = sum(Q_comp(phase_index:phase_index+(n_data/2-1)).*ca);
        % Dump
```

```
      C(doppler_index, phase_index) = I.^2 + Q.^2;
   end  % Code phase loop
end  % Doppler loop
```

A.3   FFT Search

```
function C = fftsearch(data, sat_id, ch_rate, fs, fc, n_data)
% The function
% C = fftsearch(data, sat_id, ch_rate, fs, fc, n_data)
% performs a circular convolution by using the FFT and IFFT. A single
% carrier frequency is used and a correlation vector is returned
%
% data = The data to be processed
% sat_id = the satellite to be searched for
% ch_rate = the C/A code chippingrate (1.023 MHz)
% fs = the sampling frequency
% fc = carrier frequency
% n_data = number of data samples (typically 5000 samples)

% Generate CA code
CA = cacode(sat_id, ch_rate, fs, n_data);
% Time vector
t = (0:(n_data-1))/fs;
% In-phase component
I_comp = cos(2*pi*fc.*t).*data';
% Quadrature component
Q_comp = sin(2*pi*fc.*t).*data';
% FFT of I and Q comps
X = fft(I_comp + i*Q_comp);
% conj(FFT) of CA code
F_CA = conj(fft(CA));
% Multiply in freq domain and perform IFFT
% then get the squared magnitude
C = abs(ifft(X.*F_CA)).^2;
```

A.4   Acquisition

```
function [yes, f, ph] = acquisition(fid , K, sat_id, fc_lo, step, fc_hi, T)
% The function
% [yes, f, ph] = acquisition(fid , K, sat_id, fc_lo, step, fc_hi, T)
% perform an aquisition process for one satellite.
% RETURNED :
% yes = 1 if the satellite signal is acquired
% yes = 0 if the satellite signal is NOT acquired
% f = carrier frequency
% ph = code phase (samples)
% ARGUMENTS :
```

```
% fid = file pointer to data
% K = number of sequential correlations to be accumulated
% sat_id = satellite to search for
% fc_lo = IF freq minus max Doppler offset
% fc_hi = IF freq plus max Doppler offset
% step = size of one Doppler bin
% T = threshold (typically 6 if K = 10)

%Initialize variables
i=0;
C = zeros(length(fc_lo:step:fc_hi), 5000);
% Doppler loop
for fc = fc_lo:step:fc_hi
   i = i+1;
   % K sequential correlations is accumulated
   for j=1:K
      % Read next 5000 samples
      data=(fread(fid,5000,'schar'))';
      C(i,:)=C(i,:)+fftsearch(data, sat_id, 1.023e6, 5e6, fc, 5000);
   end
   % Go back to beginning of file
   fseek(fid,0,-1);
end

% Check threshold
yes=max(max(C))/mean(mean(C))>T;

if yes  % The satellite signal is assumed to be acquired
   fc_vector = fc_lo:step:fc_hi;
   % Find the maximum correlation
   n = find(C==max(max(C)));
   % Return the code phase
   ph = floor(n/length(fc_vector));
   % Return the carrier frequency
   n = mod(n,length(fc_vector))
   if ~n
      n = 1;
   end
   f = fc_vector(n);
else          % The satellite signal is not acquired
   % return zeros
   f = 0;
   code_phase = 0;
end
```

A.5   Carrier Pull-in

```
function f_5 = pullin(fid, f, ph, sat_id);
```

```
% The function
%    f_5 = pullin(fid, f, ph, sat_id)
% performs a linear seach at constant code phase
% and returns the IF carrier.
% The purpose is to find the IF carrier with better
% accuracy so the carrier loop can pull-in
%
% fid = file pointer to data
% f = coarse carrier IF
% ph = code phase (samples)
% sat_id = satellite
% Generate CA code
ca = cacode(sat_id, 1.023e6, 5e6, 5000);
% Time vector
t = (0:(4999))/5e6;
% Initialize variables
C=zeros(101,1);
doppler_index=0;
% Search +/- 250Hz with 5 Hz increments
for fc = (f-250):5:(f+250)
   fseek(fid, ph, -1);
   doppler_index=doppler_index+1;
   % Accumulate 10 correlations
   for n = 1:10
      % Read data
      data=(fread(fid,5000,'schar'))';
      % Despread
      data = ca.*data;
      % Downconvert
      I_comp = sin(2*pi*fc*t).*data;
      Q_comp = cos(2*pi*fc*t).*data;
      % Accumlate
      I = sum(I_comp.*ca);
      Q = sum(Q_comp.*ca);
      C(doppler_index) = C(doppler_index) + I.^2 + Q.^2;
   end
end
% Find max correlation
i = find(C==max(C));
f_5 = f + (i-51)*5;
```

A.6   Tracking

```
function Ip = track(fid, fc, ph, num_sec, sat_id)
% The function
%     Ip = track(fid, fc, ph, num_sec, sat_id)
% returns the data output from the prompt I-channel
% of the code & carrier tracker.
```

```
%
% fid =  file pointer to data
% fc = carrier IF
% ph = codephase (samples)
% num_sec = how many seconds of data to process
% sat_id = what satellite to receive

% SETUP
% # of samples between early-prompt and late-prompt channel
chip_delay=3;
% code chip rate
fcode = 1.023e6;
% # of samples in a code period
scode = 5000;
% # of codeperiods to average over
num_ava = 20;
% Set file pointer to code phase
fseek(fid,ph,-1);
% Initialize variables
y(1)=0;
pnco=0.0;
t_end = 0;

% Filter coefficients
Bl=50;
damp=.7;
wn=2*Bl/(damp+1/(4*damp));
dT=1e-3;
K=400*pi;
c1 = 1/K*8*damp*wn*dT/(4+4*damp*wn*dT*(wn*dT).^2);
c2 = 1/K*4*(wn*dT).^2/(4+4*damp*wn*dT*(wn*dT).^2);

% generate CA codes
prompt=cacode(sat_id,fcode, 5e6, scode);
early=[prompt(scode+1-chip_delay:scode) prompt(1:scode-chip_delay)];
late=[prompt(chip_delay+1:scode) prompt(1:chip_delay)];

% Loop
for i=1:round(num_sec*1000/num_ava)
   % Avaraging loop for code tracking
   for j=1:num_ava
      % recalulate time vector
      t = t_end:(1/5e6):(scode-1)/5e6+t_end;
      t_end = t(scode)+1/5e6;
      % generate I/Q (pnco = phase offset)
      Icomp_lo=sin(fc*2*pi*t+pnco);
      Qcomp_lo=cos(fc*2*pi*t+pnco);
      % read data
      data=(fread(fid,scode,'schar'))';
      % Remove carrier
```

```
        Icomp=data .* Icomp_lo;
        Qcomp=data .* Qcomp_lo;
        % Calculate index for vectors
        inda = (i-1)*num_ava+j;
        % Despread with CA code
          % In-phase
        Ie(inda)=sum(early .* Icomp);   % early
        Ip(inda)=sum(prompt .* Icomp);  % prompt
        Il(inda)=sum(late .* Icomp);    % late
          % Quadrature
        Qe(inda)=sum(early .* Qcomp);  % early
        Qp(inda)=sum(prompt .* Qcomp); % prompt
        Ql(inda)=sum(late .* Qcomp);   % late
        % Envelops
        ee(inda) = (Ie(inda).^2 + Qe(inda).^2).^.5;  % early
        pp(inda) = (Ip(inda).^2 + Qp(inda).^2).^.5;  % prompt
        ll(inda) = (Il(inda).^2 + Ql(inda).^2).^.5;  % late
        % Costas loop for carrier tracking
          % Atan discriminator
        d2(inda)=atan(Qp(inda)./Ip(inda));
          % Loop filter
        if inda<2
          y(inda)=0;
        else
          % Filtered discriminator output
          y(inda)=y(inda-1)+(c1+c2)*d2(inda)-c1*d2(inda-1);
          % Phase for Carrire NCO
          pnco=rem(K*y(inda)+pnco, 2*pi);
        end
    end
    % Code loop discriminator
    E_L = sum(ee((inda-num_ava+1):inda)./ll((inda-num_ava+1):inda))/num_ava;
    % Check for thresholds and phase shift
    % CA code if either threshold is crossed
    if E_L < .8
        prompt=[prompt(2:scode) prompt(1)];
        early=[early(2:scode) early(1)];
        late=[late(2:scode) late(1)];
    elseif  E_L > 1.5
        prompt=[prompt(scode) prompt(1:scode-1)];
        early=[early(scode) early(1:scode-1)];
        late=[late(scode) late(1:scode-1)];
    end
end
```

A.7 Acquisition and Tracking

```
function Ip = gps(fid, sat_id, K, fc_lo, fc_hi, step, T, num_sec);
```

```
% The function
%   Ip = gps(fid, sat_id, K, fc_lo, fc_hi, step, T, num_sec);
% demodulates GPS data and returns the in-phase prompt channel
% sampled at 1 kHz. If the satellite signal is NOT found Ip = 0
%
% fid = file pointer to data
% sat_id = satellite to demodulate
% K = number of sequential correlations to be accumulated
% fc_lo = IF freq minus max Doppler offset
% fc_hi = IF freq plus max Doppler offset
% step = size of one Doppler bin
% T = threshold (typically 6 if K = 10)
% num_sec = number of seconds of data to process

% Perform acquisiton
[yes, f, ph] = acquisition(fid , K, sat_id, fc_lo, step, fc_hi, T)

% Pull-in the frequency
f = pullin(fid, f, ph, sat_id);

if yes
   % Pull-in closer to IF
   f = pullin(fid, f, ph, sat_id);
   % Track the satellite
   Ip = track(fid, f, ph, num_sec, sat_id);
else
   % Not acquired
   Ip = 0;
end
```

A.8   Sample Script

```
% Sample script to demodulate signal from satellite 17
% Open data file
fid = fopen('SRVY1SF1.DAT');
% Set parameters
% Number of correlations to avarage in acquisition
K=10;
% Doppler frequencies
fc_lo = 1.25e6-10e3;
fc_hi = 1.25e6+10e3;
% Doppler bin
step = 500;
% Threshold for acquisition
T = 6;
% Guess what
sat_id = 17;
% Number of seconds to process
```

```
num_sec = 1;
% Demodulate
Ip = gps(fid, sat_id, K, fc_lo, fc_hi, step, T, num_sec);
% Plot the in-phase prompt channel
plot(Ip);
% Close file
fclose all;
```