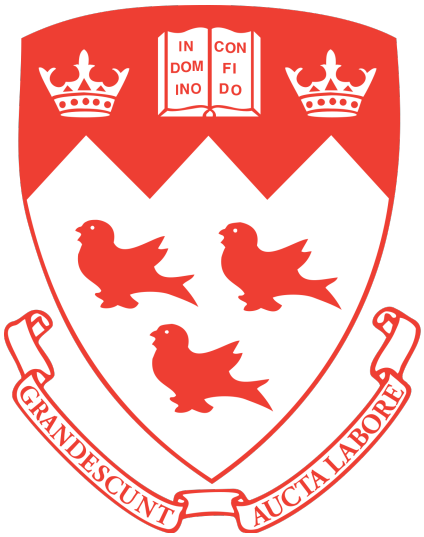
Simulink Implementation of a GPS Receiver

Alamjyot Parihar 260570908
Ahmed Sami 260568821

ECSE 457 Group 41
Project Supervisor: Prof. Harry Leib

April 17, 2018



Contents

List of Figures

List of Tables

1 Abstract

Traditionally the algorithms that make up a complete software defined GPS are done in assembly, C/C++ or Matlab. This requires a large team to write and test algorithms, which is both costly and time consuming. This project is intended to provide a basis for clear, easy to follow and modify implementation of a L1 carrier GPS receiver. Our simulation employs the use of the graphical programming environment Simulink to design and test models.This project aims to make the fundamental functionality of a GPS receiver more visible and the inner workings simpler to examine and analyze. We hope that our implementation will help pave the way for other simpler methods of implementing and understanding intricate systems.

2 List of Abbreviations

GPS - Global Positioning System

SatNav - Satellite Navigation

PRN - Pseudo-Random Noise

CDMA - Code Division Multiple Access

DSSS - Direct Sequence Spread Spectrum

BPSK - Binary Phase Shift Keying

LO - Local Oscillator

C/A - Coarse/Acquisition

DLL - Delay Locked Loop

PLL - Phase Locked Loop

SNR Signal to Noise Ratio

3 Introduction

Writing the algorithms for a software defined GPS receiver is a large task and requires mastery of the programming language in use. The goal of this project is to implement a complete single channel GPS receiver in Simulink - a graphical programming language which would facilitate the simulation, modelling and implementation of the system by simplifying the complexity and allowing for more straightforward system debugging. Working in a GUI environment will help to reduce the time and resources spent writing out complete algorithms in order to process the different of signals present in a GPS system. This project aims to investigate the methodologies and fundamental workings of the GPS receiver and allow us to be able to implement the acquisition, tracking and finally the position calculation stages of a GPS receiver. Successful use of Simulink in order to model a complex system like GPS could be used as a basis to model other complicated systems in a similar manner

Our goal for this project was to extract extract the navigation data that would be in a GPS signal from the visible satellites. This is done through the acquisition and tracking stages of a GPS

receiver. In the first semester we successfully implemented the acquisition stage which identifies the 4 strongest satellite signals and gives rough estimates of their carrier frequencies and code phase. Our objective this semester was to implement the tracking stage which takes in the results from acquisition and tracks the changes to those metrics over time, allowing for accurate navigation data extraction.

4 GPS Overview

4.1 GPS Signal

The GPS system occupies a very high frequency band (1GHz), including some carrier frequencies that are intended for military use only. The scope of this project will only be concerned with the L1 C/A band (also known as the legacy signal) which is broadcast by all satellites at a frequency of 1575.42MHz. Having the carrier at such a high frequency allows for less ionospheric interference, and reduced interference from other high power radio signal transmitters. It is worth noting that since each satellite transmits on the same carrier frequency, a captured signal will have data from every visible satellite. Methods for ensuring each of this signals can be received without cross interference will be discussed in the following sections of the project.

The GPS system signals are based on the principle of code division multiple access, which allow receivers to decode chosen satellite signals out of the aggregate of signals being broadcast on a carrier frequency. This method of multiplexing is realized by the modulation of data onto the carrier wave through binary phase shift keying (BPSK) (FIGURE XX). BPSK uses a direct sequence spread spectrum (DSSS) of binary bits which are used in an XOR operation to modulate the data. This modulated data is then used to phase shift the carrier signal by 180O at every change in binary value (denoted by a rising or falling edge). The DSSS used by the L1 carrier is the known as the coarse acquisition (C/A) code. This code runs at a rate of 1,024,000 bits/second, and modulates data which runs at a rate of 50 bits/second (denoted by C and D respectively in Figure 1). Note that another DSSS called the P code exists in the GPS system, however the sequence is not known to the public, as it is intended for military/government use only.

4.2 GPS Receiver Architecture

The receiver's purpose is to take in the incoming signal, accurately demodulate the carrier wave and C/A code and use the data bits to work out its position.

The first stage in a GPS receiver is the Acquisition stage, this receives the aggregate of the satellite signals and processes it to determine which of the 32 satellites are visible and provides rough estimates of the incoming signals carrier frequency and code phase.

The parameters estimated in Acquisition are then passed onto the the Tracking stage which tracks changes due to Doppler shifting and movements in the code phase and carrier frequency over time in the current block of data. This stage is used to refine the parameter estimation from Acquisition and outputs accurate values of these 2 parameters.

The code phase and carrier frequency values are then used to demodulate the incoming signal and the 50 Hz Navigation data bits are extracted, these data bits contain all the information the receiver requires from each satellite. The navigation data is then decoded according to US Interface standards and the information is used to work out relative time, psuedoranges and finally, receiver position.

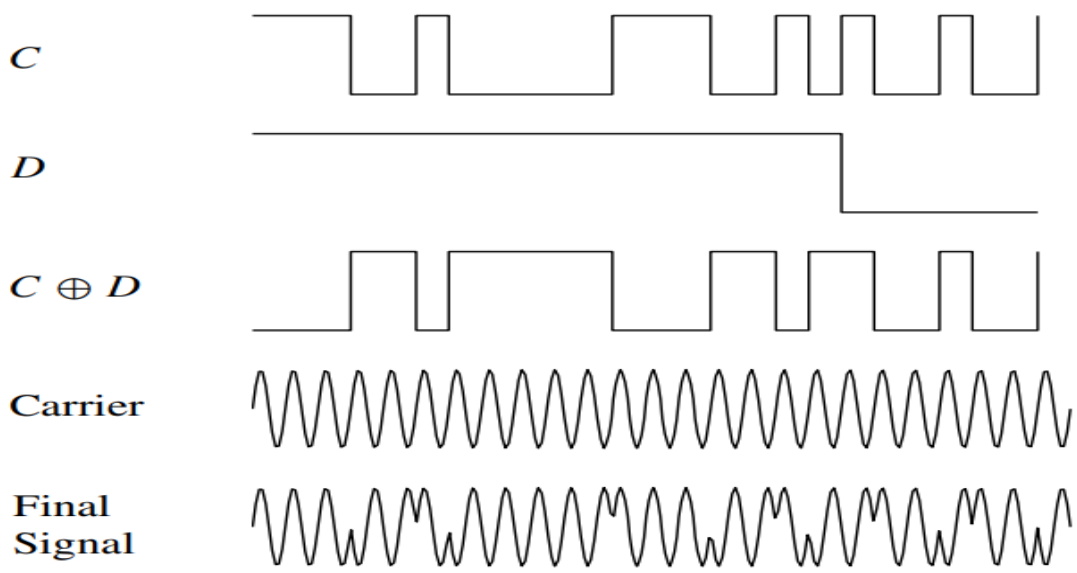


Figure 1: BPSK Modulation[1]

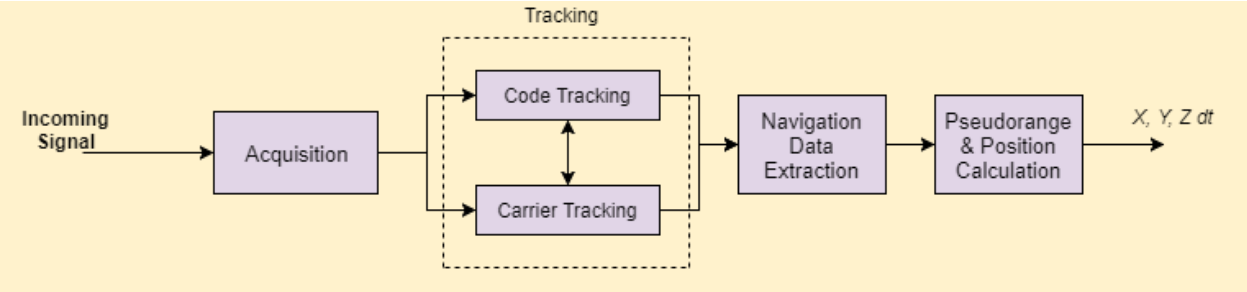


Figure 2: GPS Receiver Architecture

4.3 GPS Receiver Front End

A typical GPS Receiver front end consists of an antenna receiving the signal, which it then converts to DT. This discrete signal goes through filtering, amplifying and finally downconversion from the L1 frequency to the intermediate frequency (IF). Our project contains almost none of this as it is a software implementation and we are assuming signals will have been pre-processed appropriately for application. As a result off this assumption, when generating test signals we are able to simply modulate the data using the IF value for the carrier frequency. This allows for reduced simulation time for test signals (instead of using high GPS frequencies) as well as reduced file size for saved signals. For our generator, a 0.5 second signal file is approximately 2.5GB in size.

An example of specifications used in real hardware are:

IF 9.548 MHz

Sampling Frequency 38.192 MHz

The parameters we use are

IF: 9.207 MHz

Sampling Frequency: 32.768 MHz

The value that we have chosen for sampling frequency allows for 2^{15} samples per millisecond. This is crucial to our FFT in the acquisition stage, which requires the input be of size 2^N. The choice of IF is explained further below in our previous signal generator review.

5 Overview of Previous Work

5.1 Original Signal Generator

It was determined that the most appropriate way of creating such a simulated GPS signal would be to simply XOR a square pulse wave representing data bits with a real generated C/A code representing a satellite. This data is put into non return to zero format and modulated through multiplication onto a carrier frequency left to our choosing. With the L1 band, the carrier frequency is a multiple of the C/A code chip rate, allowing for optimal BPSK. As mentioned previously a hardware front end will down convert this carrier to an intermediate frequency, with our chosen example having a value of 9.5MHz. Using this knowledge we have chosen to modulate our test signal onto a carrier frequency of 9*(C/A chip rate), or 9.216MHz, to maintain an optimal modulation and IF range. For the purpose of describing a base frequency, we will refer to this value of 9.216MHz as our IF.

5.2 Acquisition Loop

The purpose of acquisition is to determine all visible satellites to the user. It calculates rough estimates of the Incoming signals code phase and carrier frequency by correlating the signal with locally generated PRN codes and locally generated carrier waves. The need to find the correct frequency and code phase arises from the Doppler shift experienced on received signals. After identifying a visible satellite, the parameters are passed on to the tracking stage.

The Parallel Code Phase Search method (Figure 3)implemented for acquisition parallelizes the code phase parameter, eliminating the need to sweep through all 1023 code phases for every PRN. This leaves us only the need to sweep through the 21 possible frequencies steps.This is done through circular cross-correlation between the incoming signal and the generated carrier and generated PRN code with no phase shifts.

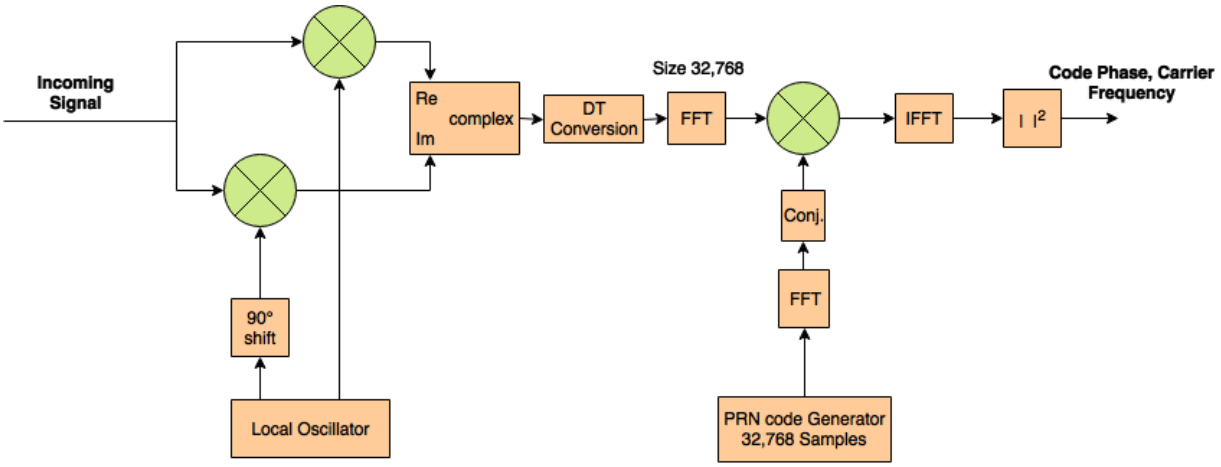


Figure 3: Parallel Code Phase Search

The incoming signal is multiplied by a locally generated carrier wave (and a 90 degree shifted version just as in the Serial Search method), sweeping through all possible 21 frequencies. The 2 LO multiplications are then combined into a complex signal and are sampled at a frequency of 32.768 MHz, over 1ms. This signal is then fed into a 32,768-point FFT.

At the same time the PRN code is generated and transformed into the frequency domain using the same method. The PRN code FFT output is complex conjugated and multiplied with the carrier FFT as part of the correlation method. The result is then taken back into the time domain using and IFFT and the absolute value is taken and squared to give the time domain correlation value between the input and the generated PRN code. Once again, if at the tested frequency and PRN code there is a peak magnitude above a certain threshold, it would indicate the visibility of a satellite and mark the code phase and confirm the carrier frequency.

6 Doppler Shift

Due to the nature of satellite navigation systems, the point of signal generation is constantly moving overhead. This movement causes a Doppler effect on the signal, leading to shifts in carrier frequency and code phase seen at the receiver. Typically, Doppler shifts to code phase can reach up to ± 3 chips/s for stationary receivers.

7 Updated Signal Generator

The signal generator for testing tracking loops (Figure 4) is similar to the previous function generator used to test the acquisition loop[1]. Several additions and modifications made to the generator are explained in the sections below.

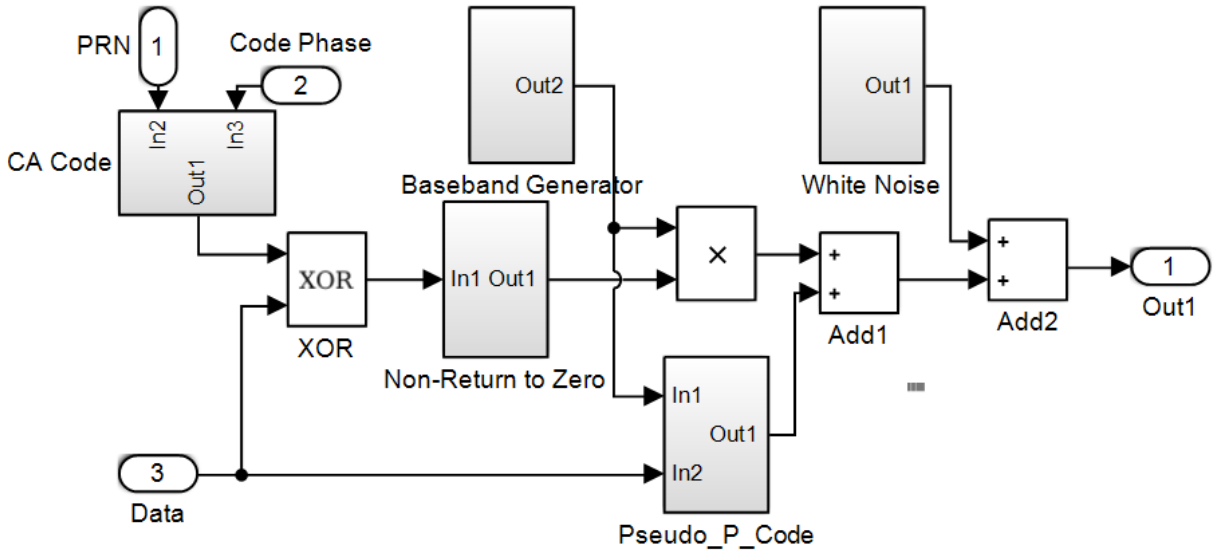


Figure 4: Signal Generator

[1]Detailed explanation of acquisition function generator available in previous semesters report

7.1 Simulated Doppler Shift

In order to properly simulate effects of the Doppler shift, a counter controlled VCO was used to generate the carrier signals (Baseband Generator block). The counter begins at 0 and counts to 2500 over a one second period. Setting the VCO input sensitivity to ±0.08,0.04, or 0.02Hz/V, allows us to achieve frequency shifts of ±200, 100, or 50Hz/s respectively. A counter was also used to shift the C/A code phase by increments of $\frac{1}{32}$ (made possible by up-sampling and shifting the original code). Adjusting the sampling period of the counter, the desired code phase shifts of ±3 to 6 chips/s can be produced.

7.2 Pseudo P-Code and Noise

A pseudo P-code was generated using a square wave with the same frequency of the actual P-code. This code is then modulated on to a 90 degree phase shifted carrier and tuned 3db down to better approximate actual GPS signal generation.

The final addition to the signal generator was that of a white noise block. This allows us to simulate noise that may occur in actual signals. We chose white noise specifically as it has equal intensity at different frequencies, providing a fairly uniform PSD. The ratio of signal to noise if varied between 10dB and 20dB

8 Received Signal Demodulation

Once a signal has been determined to be in range by the acquisition stage, the data must be demodulated. To perform demodulation, it is required to multiply the incoming signal with a carrier-frequency matched local oscillator. This effectively wipes the carrier from the signal, bringing it back to the data baseband. The last step is to multiply the signal with a locally generated C/A code, effectively decoding the data.

The values found during acquisition are used to generate the local oscillator and C/A code, however this is not enough. As explained above, the code phase and carrier frequency undergo Doppler shifts and change over time. These changes must be tracked and fed back to the generators to provide stable, coherent demodulation. The process of signal tracking is explained in the following section.

9 Tracking Loops

9.1 Introduction

The objective of the tracking phase is to refine the code phase and carrier frequency metrics estimated in the acquisition stage, track changes in these values due to doppler shifts from the motion of satellites and the receiver, and extract the navigation data from each acquired satellite. The tracking loops consist of a PLL for the carrier loop and DLL for the code phase tracking loop which generate local replicas of the incoming signals Carrier frequency and C/A code. The local replicas are correlated with the input signal and discriminators are used to adjust the locally generated metrics accordingly to be perfectly aligned with the incoming signal as it changes over time.

9.2 Carrier Tracking

In order to successfully extract navigation data, a perfect replica of the carrier of the incoming signal must be used to demodulate the incoming signal and strip the carrier. A Phase Locked Loop (PLL) with a Voltage Controlled Oscillator (VCO) is used in GPS receivers to track the incoming signal's carrier and replicate it's frequency. The PLL employs a loop discriminator which produces a function of the error in the phase between the received signal and the local replica to be fed back to control local oscillator. An IIR filter smooths this error term, minimizing oscillation and overshoot, providing stable demodulated data.

Since GPS signals are modulated using BPSK, a problem arises when using standard PLLs which are very receptive to the 180° phase shifts caused by the navigation data bit transitions. To ensure that our carrier tracking loop is insensitive to the phase transitions, we used a variation of the standard PLL called a Costas loop.

9.2.1 Costas Loop

The main feature of a Costas loop is its insensitivity to 180° shifts in phase. The input signal is multiplied with both the local carrier and a 90° phase shifted version of the carrier as shown in Fig 5. The objective of the loop is to use the carrier loop discriminator and feedback in to the VCO to keep all the energy in the I (in-phase) arm [1].

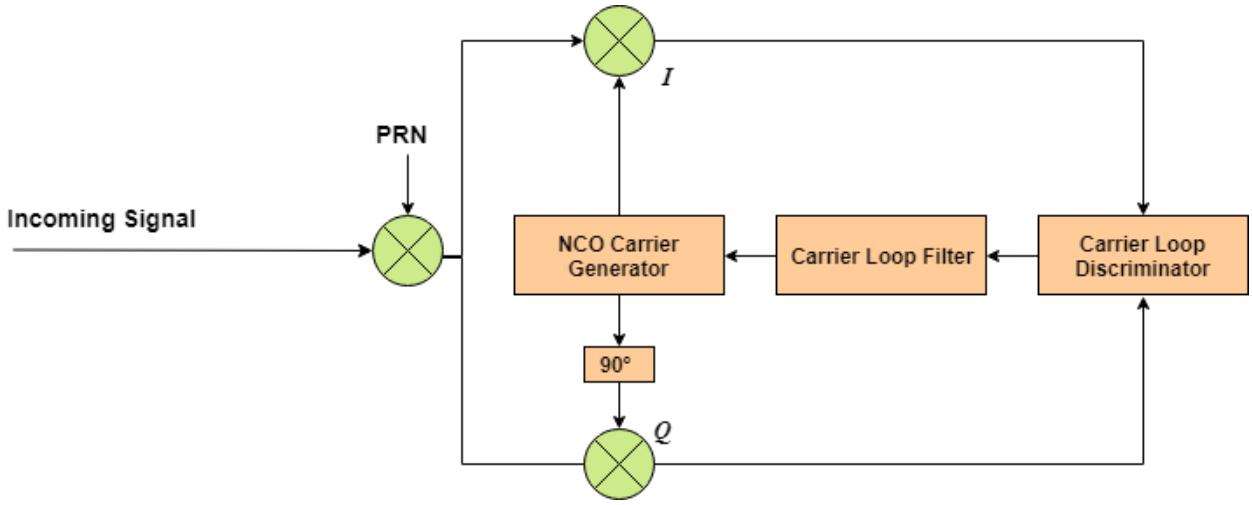


Figure 5: Costas Carrier Tracking Loop

9.2.2 Discriminators

A costas discriminator is used to keep the energy in in phase arm and outputs a function of the difference in the phase error ψ between the input signal and local carrier. There are several types of Costas PLL with their own benefits and trade-offs as shown in table 1.

From the equations in table 1 it is shown that the discriminators produce an output of 0 when the phase error of the real part is 0° or ±180°. This property renders the loop insensitive to the 180° navigation bit transition induced phase shifts. We have chosen to implement the arctan discriminator as while is it the most computationally time-consuming, it provides the most accurate phase error results. As we have mentioned, the goal of the Costas loop is to drive the signal towards

Table 1: Types of Costas PLL Discriminators [1]

Discriminator	Description	Relative Accuracy
$D = sign(I_p)Q_p$	The output of the discriminator is proportional to $\sin(\psi)$	Moderately Accurate
$D = I_pQ_p$	The discriminator output is proportional to $\sin(2\psi)$	Least Accurate
$D = tan^{-1}(\frac{Q_p}{I_p})$	The discriminator output is the phase error	Most Accurate

the In phase arm. This is reflected in the arctan discriminator equation in which the phase error is minimized when the correlation between the input signal and the with the local replica (In-Phase arm) is maximized while the correlation in the quadrature phase arm is minimized.

9.3 Code Phase Tracking

Code tracking of the phase of a particular PRN code in the incoming signal to generate a perfectly aligned code sequence is implemented using a DLL. GPS receivers use a special type of DLL known as an early-late code tracking loop. After stripping the carrier from the incoming signal with a perfectly aligned local carrier, the signal is multiplied with 3 variations of a locally generated PRN code, each $\pm\frac{1}{2}$ chip apart. These 3 local replicas correspond to the early, present (prompt) and late versions of the local PRN code (E, P, L). Following this, the multiplication results are then integrated over the number of samples, providing a correlation value between each local code replica and the incoming signal's code. If the late replica, for example, corresponds to the highest correlation then the PRN code should be delayed by $\frac{1}{2}$ a chip as shown in Figure 6.

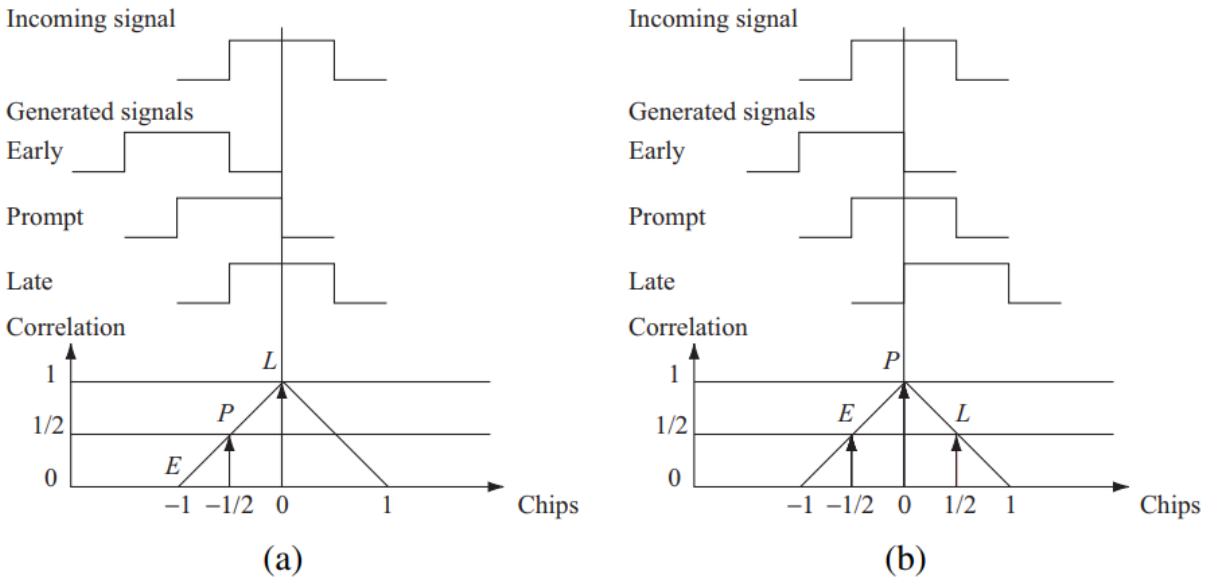


Figure 6: Code phase tracking correlation results example (a) late replica has the highest correlation (b) prompt code has the highest correlation [1]

9.3.1 Delay Locked Loop

Simple DLLs with only 3 correlators are ideal if the locally carrier wave is constant in both frequency and phase. However, cases where there are phase offsets between the local carrier wave and the incoming signal introduces noise and difficulty tracking the code phase.

We have implemented a DLL with 6 correlators, 3 of which are used for the local carrier replica and the other 3 are used for a 90° shifted version of the replica as shown in Fig 7. This allows for our code tracking loop to be insensitive and independent of the phase of the local carrier as any variations between the incoming signal and the local carrier phase will be compensated by switching the energy between the in-phase (I) and the quadrature arm (Q) [1].

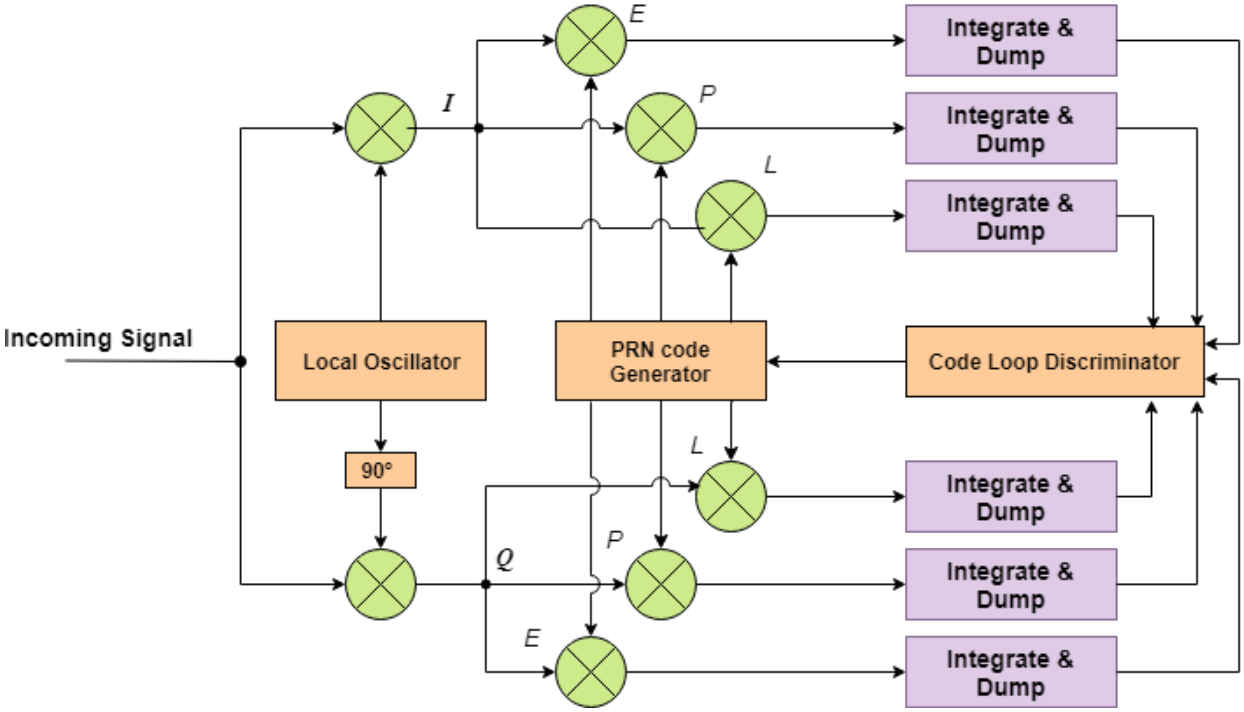


Figure 7: Code Phase Tracking DLL System Block Model

9.3.2 Discriminators

A code loop discriminator is used to feedback the correlation information to the PRN code generator in order to adjust the code phase accordingly. The specific discriminator to use depends on the desired application and expected signal noise. In our case we require the DLL to be independent to the Costas PLL of the carrier tracking, therefore we would use a discriminator which considers both the in-phase and the quadrature arms of the signal. An early-late power discriminator is adequate however we have decided to use the normalized early - late discriminator showsn in the equation below as it allows for effective performance with varying SNRs. [1]

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

Table 2: Types of DLL Discriminators [1]

Type	Discriminator	Description
Coherent	$I_E - I_L$	Does not require the Q branch but requires a good carrier tracking loop for practical functionality
Noncoherent	$(I_E^2 + Q_E^2) - (I_L^2 + Q_L^2)$	**Early - late power**. Approximately equal to the coherent one within $\frac{1}{2}$ a chip
Noncoherent	$\frac{(I_E^2+Q_E^2)-(I_L^2+Q_L^2)}{(I_E^2+Q_E^2)+(I_L^2+Q_L^2)}$	**Normalized early - late power**. Especially effective when the chip error is $> \frac{1}{2}$ a chip and tracking noisy signals
Noncoherent	$I_P(I_E-I_L)+Q_P(Q_E-Q_L)$	**Dot product**. Uses all six correlator outputs

9.4 Combined Tracking Loop

In order to reduce the computational complexity of the tracking system, the Code tracking loop is coupled with the carrier tracking loop. The prompt code from the code phase loop is multiplied with the input signal to demodule the C/A code with the resulting output being the inputs carrier with phase changes from the navigation data only. This signal is then applied to the input of the the carrier loop which generates a local replica at the carrier frequency of the input. This is then used to strip the incoming signal's carrier to produce only a C/A code with no carrier freq to be fed back into the code tracking loop. As Fig 8 shows this process consists of 11 multiplications which are the limiting factor in speed of operation.

To summarize, the code loop generates the local PRN code used to wipe the code from the incoming signal for the tracking loop which in return generates the local carrier replicas used to remove the carrier from the signal to be used in the code loop.

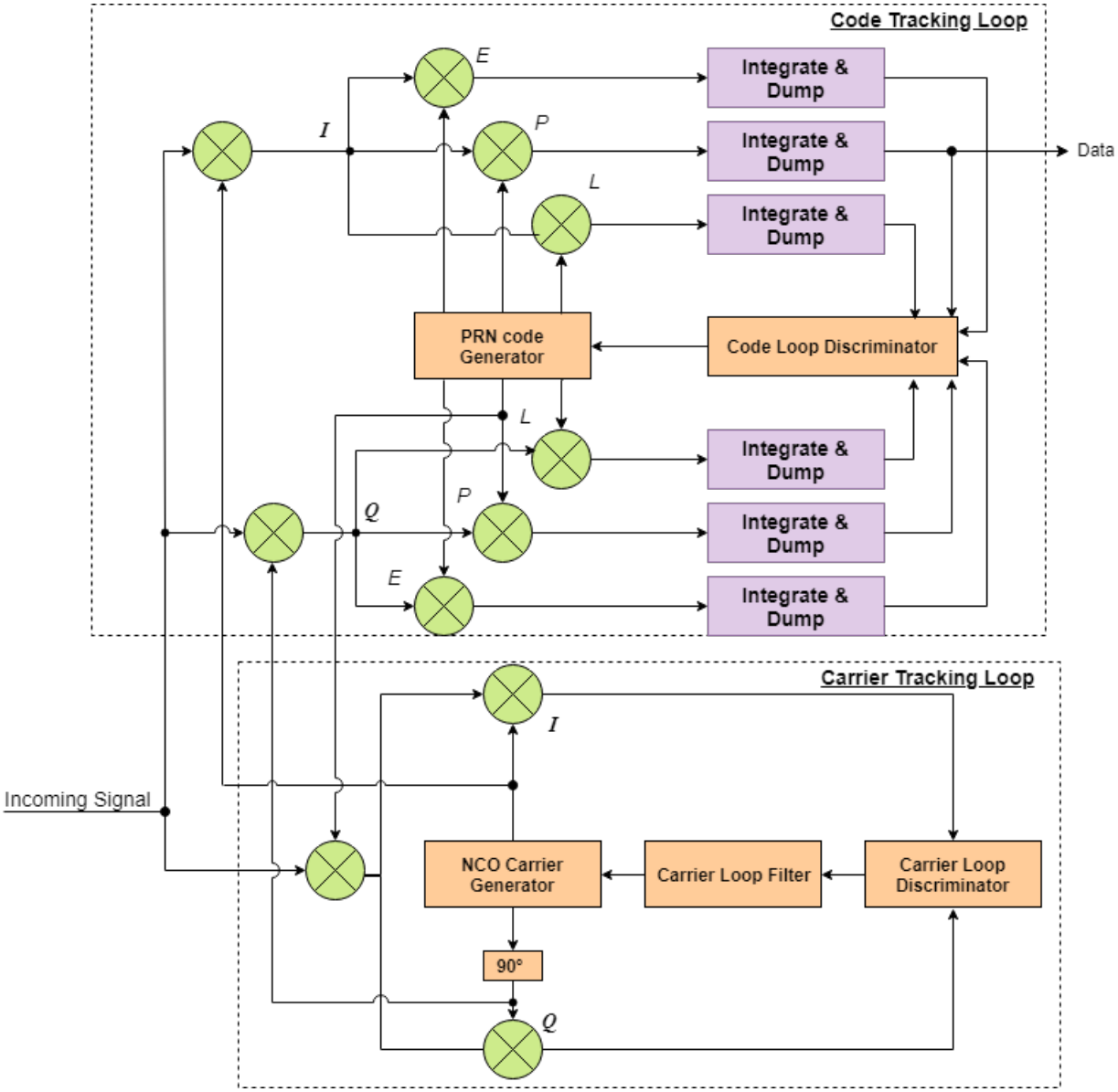


Figure 8: Combined tracking loop system block model

10 Simulink Implementation of Tracking Loop

10.1 Simulink Model

The tracking loops implemented in simulink (Figure 9) are modeled after a combined costas loop and delay locked loop mentioned in the previous section. The tracking stage block contains 4 of these loops, which track signal simultaneously

10.2 Design Parameters

When transitioning from the block model to the simulink model, there are several design choices to be made. The error term used to drive the local oscillator for carrier tracking is an arctan

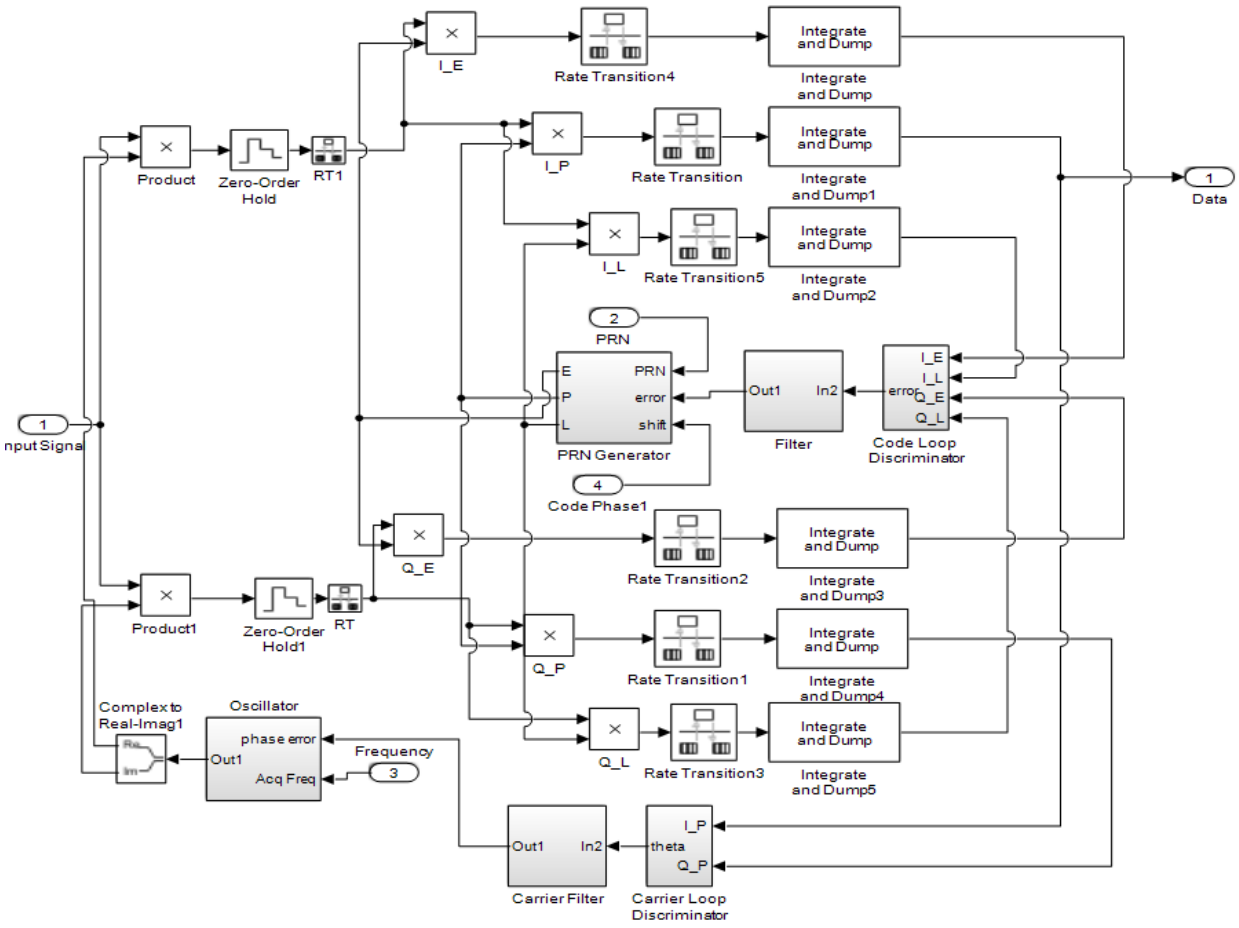


Figure 9: Combined tracking loop simulink implementation

discriminator. This provides indifference to the 180 deg phase shifts seen in BPSK signals. The error term used to drive the C/A code generator is the normalized early minus late power discriminator. This discriminator provides the best tracking when chip errors are larger than one. The integration period of the integrate and dump block is 1ms, equal to one full C/A code. In order the perform the necessary signal processing in discrete time, the continuous input signal was converted using a zero order hold block with a sampling rate equal to the sampling rate of our system (f_s=32.768MHz).

10.3 Controlled Oscillator

The controlled oscillator used to wipe the carrier from the signal is modeled after a numerically controlled oscillator. The NCO takes as input a phase error term which provides a phase increment. This increment is added to an internal phase accumulator whose value is used to output a sine wave of the correct frequency. The formulas used to control the output of the NCO (provided below) come from [2]. Note that the accumulator word length (N) is 16 for our implementation.

The frequency of the output is $f_0 = \frac{\Delta\theta f_s}{2\pi 2^N}$, where $\Delta\theta[n] = 2\pi 2^N(\frac{f_c}{f_s} + e[n])$

The output of an NCO can be written as

$$\cos[n] = \cos(2\pi[n\frac{f_c}{f_s} + \sum_{i=0}^{n} e[i]]) \tag{10.3.1}$$

Using this relationship between error term and NCO output, A simulink model for a local oscillator was created as shown in Figure 10.

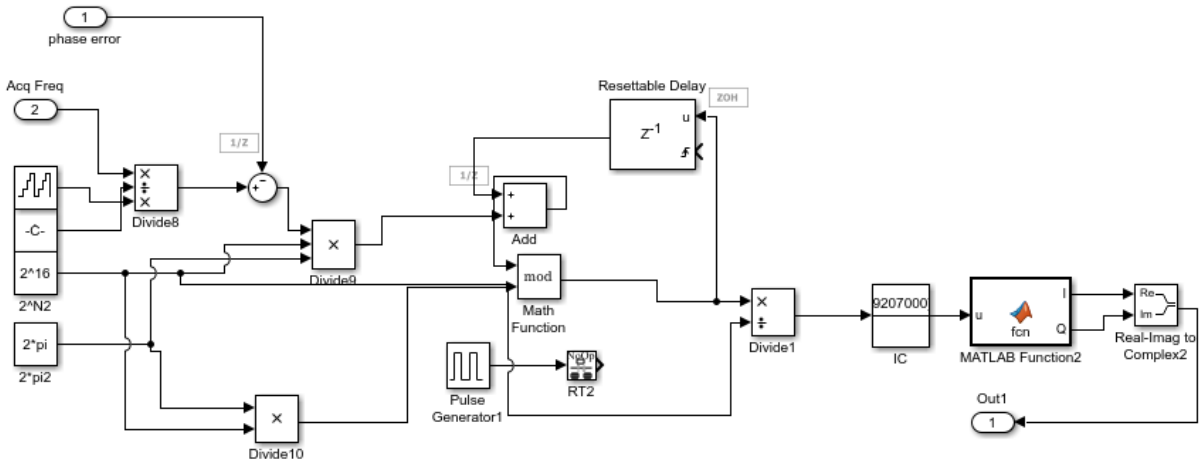


Figure 10: Tracking Loop Locally Oscillator

10.4 Controlled C/A Code Generator

The C/A code generator (Figure 11) is modeled after that of the acquisition stage, with a few adjustments. To make use of our desired discriminator, we must produce an early and late version of the C/A code (spaced half a chip apart) in addition to the prompt code. To space the chips properly, the C/A code is up sampled 32 times and shifted 16 bits away from the correct code phase, in the appropriate direction. The error term of the code phase discriminator is then added to the detected code phase from acquisition, allowing tracking of the correct code phase to be input to the local C/A code generator.

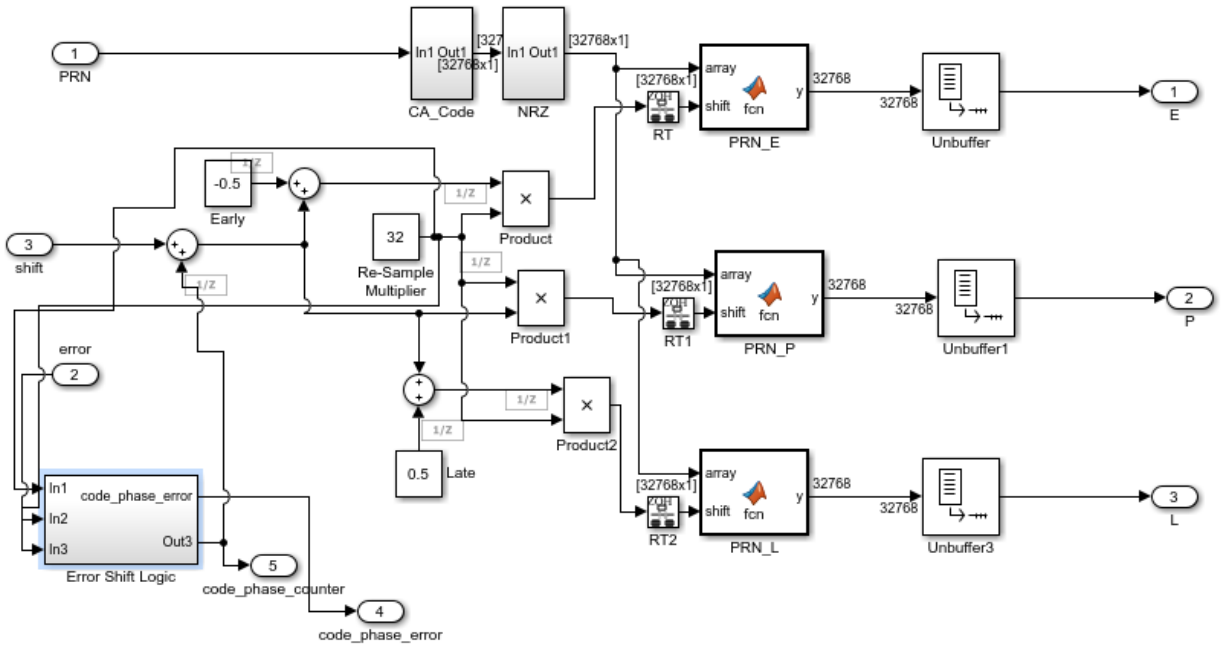


Figure 11: Tracking Loop C/A Code Generator

10.5 Discriminator Filtering

In order to provide smooth changes to the local carrier generator, the error term must be filtered. The filter implemented in simulink consist of a simple second order IIR filter. The fixed parameters used to choose appropriate coefficients are

Bandwidth 30Hz

Damping Coefficient 0.707

Loop Gain $\frac{1}{4}$

The set values of these parameters were chosen based on resulting waveforms of the carrier discriminator for several test values. The actual coefficients used in the IIR fiter block can then be calculated by using the following relationships between bandwidth, damping ratio, natural frequency and filter constants:

$BW = (\frac{W_n}{2})(\zeta + \frac{1}{4\zeta})$

$\tau_1 = K/W_n^2$

$\tau_2 = \frac{2\zeta}{W_n}$

11 Tracking Loop Testing

To test the full functionality of the tracking stage, we must first test each individual parameter. Certain parameters were tested as the loop was being built, however that is not enough to guarantee proper functionality. To thoroughly validate requires us to generate a few test signals to work with. A real GPS signal would use data bits of 50Hz, however, for the purpose of saving time on simulations we have set it to 100Hz. Data bit values are set according to a 5 bit repeating

Satellite Number	Carrier Frequency	Code Phase	Data Vector
1	9.209MHz (+50Hz/s)	0 (+3chips/s)	[1 0 0 1 1]
2	9.204Hz (-100Hz/s)	200 (+3chips/s)	[1 0 1 0 1]
3	9.207MHz (+200Hz/s)	512 (-3chips/s)	[0 1 1 0 0]
4	9.207MHz (+100Hz/s)	999 (+3chips/s)	[1 0 0 1 0]

Table 3: Test Signal Parameters

vector unique to each generated signal. The defining features of our test signals are listed below, where the value in brackets indicates the doppler shift. Note that in an a real environment the GPS signal experiences a shift of about 0.8Hz/s, however for the purpose of testing and demonstrating loop functionality in a shorter period of simulation time, we have drastically increased this value. The code phase rates we have chosen are representative of actual Doppler shift rates experienced by GPS receivers[6].

Normally the satellite PRN number, code phase and carrier frequency would be passed to the tracking loop from the acquisition stage, however for the purposes of testing, these parameters are pre-set and loaded in from a matlab array.

The very first test involving the tracking stage was simply to demodulate the data from a single signal when no other signals are present. This is fairly straightforward as the tracking loop should be able to demodulate data in the beginning without worrying about tracking changes. This beginning test was successful in recovering the generated data, and we were able to move on to the next stage of testing. In the following sections when discussing tracking results, signals are referred to by the order they appear in their respective legends.

11.1 Carrier Tracking Testing

The first carrier tracking test involved a single test signal, feeding the tracking loop with the exact values used to generate it. As the signal propagates, the frequency will changing, and we must look at a plot of the discriminator error term to determine whether it is being tracked accurately. The resulting shows two tracking loops, one where error feedback is implemented, and the other where it is now.

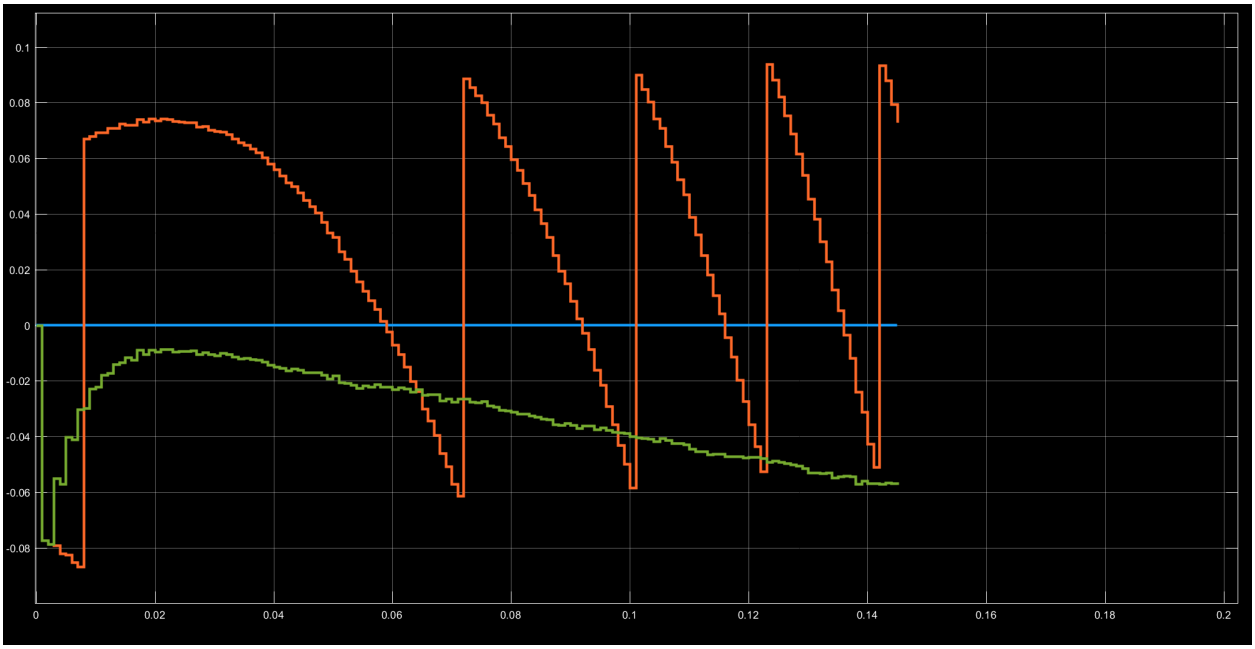


Figure 12: Carrier tracking discriminator (Y):tracked, (R):Untracked

As can be seen in Figure 12, when the error term is fed back into the carrier generator, the loop is able to accurately track its changes. When feedback is not enabled however, the loop quickly looses track of the signal.

The second carrier tracking test involved feeding four different tracking loop with the same signal, but deviating the supplied initial carrier frequency. These deviations were 1000Hz, 500Hz, and 250Hz shifts from the actual carrier frequency of 9.207MHz.

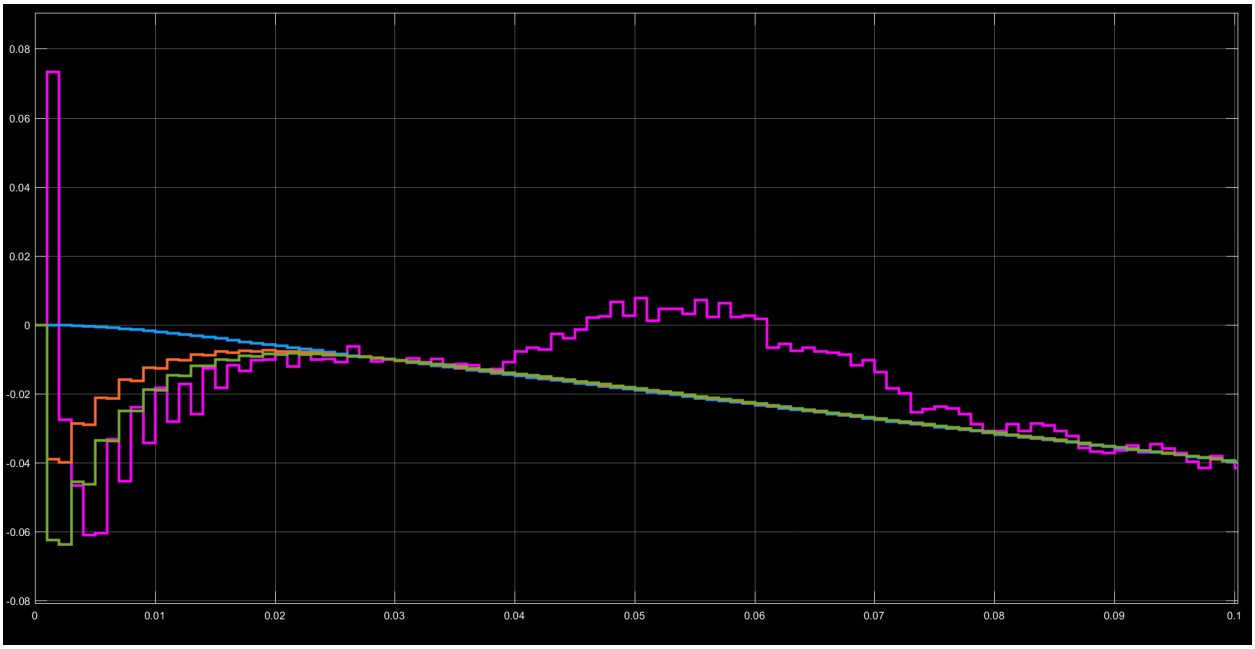


Figure 13: Carrier discriminator (P):$\Delta f = 1000$ Hz, (G):$\Delta f = 500$ Hz, (R):$\Delta f = 250$ Hz, (B):$\Delta f = 0$ Hz

As can be seen in Figure 13, the carrier loops are able to track on to the first 3 signals is under 30ms. The farther off the supplied frequency is from the actual value, the longer it takes to settle. However, for the loop being fed with a frequency off by 1000Hz, the lock in is never achieved.

The final carrier tracking test was carried out using an input signal consisting of a sum of all 4 generated signals. The purpose of this testing stage is to determine whether or not the loop can properly track the changes to four different signal being received at the same time. Achieving this will help to ensure that the system will coherently demodulate data from multiple signals in final testing.

The test signals used to generate Figure 14 are the ones listed at the beginning of this section in 3, and correspond to their order in the legend of the plot. The colors of these signals on the graph are also listed in the brackets.

Signal 1 (P)+50Hz/s

Signal 2 (B)-100Hz/s

Signal 3 (R)+200Hz/s

Signal 4 (G)100Hz/s

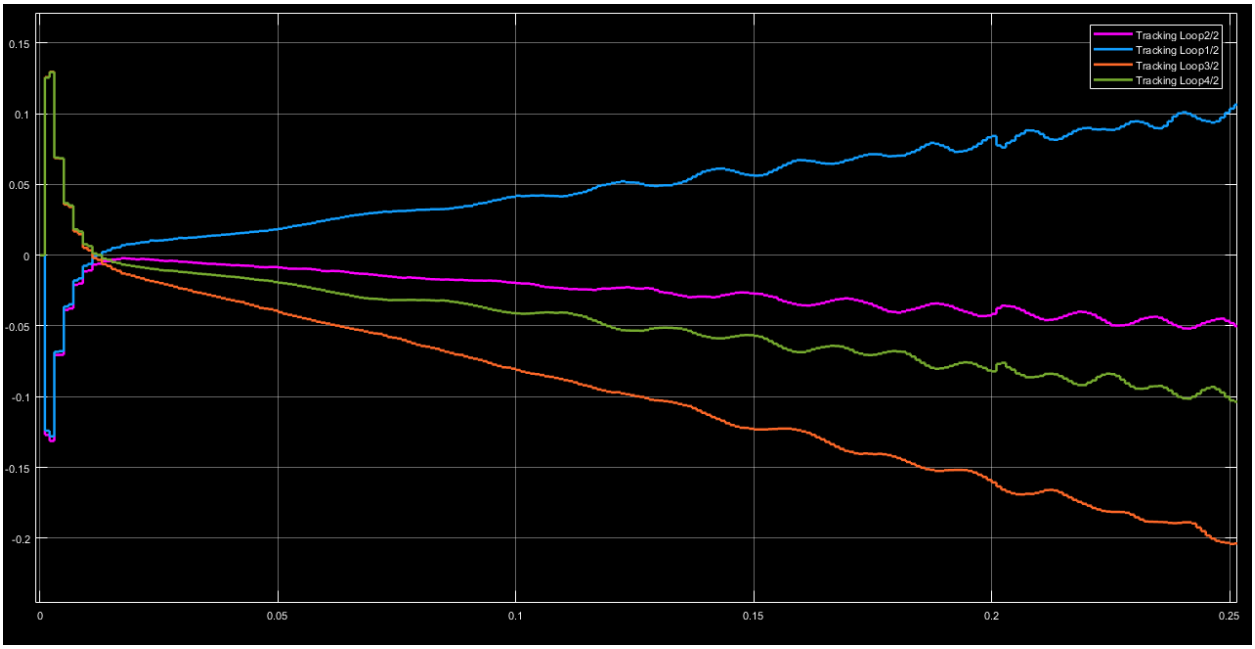


Figure 14: Carrier tracking error term for 4 different signals

To validate the function of this loop, we must observe the slope of the signals plotted in Figure14. Since the slopes of each signal match proportionately to their Doppler rate, It is evident that the tracking system is able to accurately measure changes to one signal in the presence of others. We must now test the remaining tracking loop functions.

11.2 Code Phase Tracking Testing

Code phase tracking was carried out in a manner similar to carrier tracking. The initial code tracking test involved a single signal, feeding the tracking loop with the exact code phase value used to generate the test signal. As the time passes, the code phase will changing, and we must look at a plot of the code discriminator error term to determine whether it is being tracked accurately. The plot below shows two code tracking loops, one where error feedback is implemented, and the other where it is not.

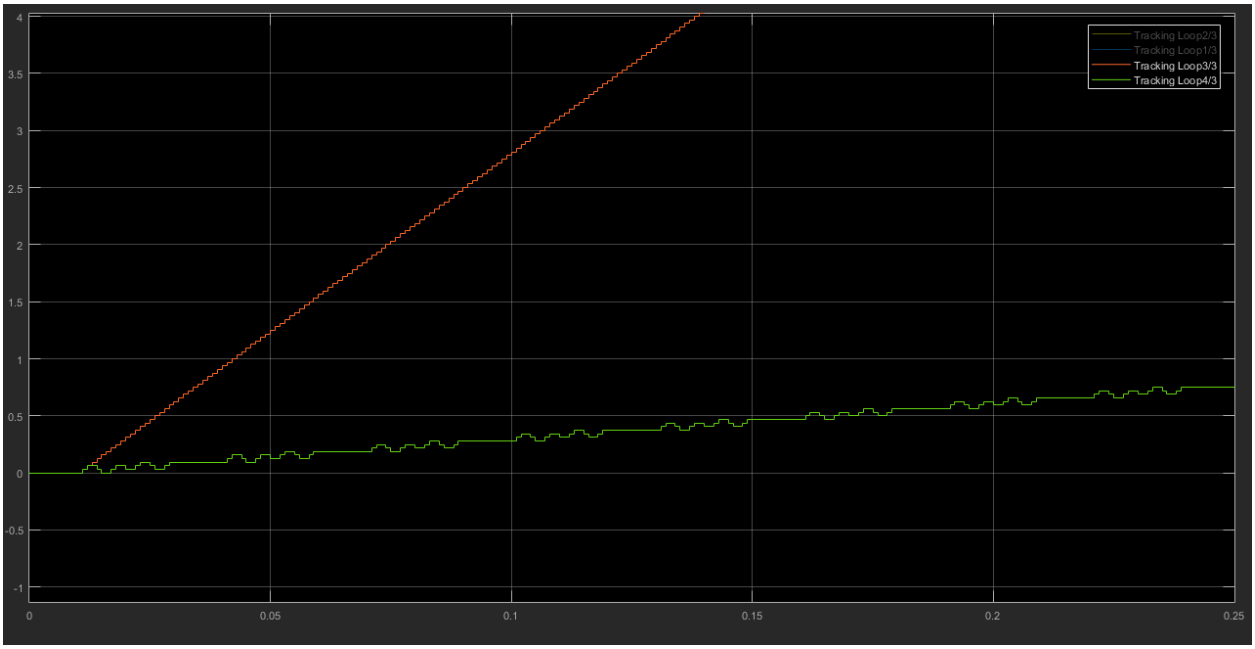


Figure 15: Code tracking error term for single signal

As observed in Figure 15, when the error term is fed back into the carrier generator, the loop is able to accurately track its changes. When feedback is not enabled however, the loop quickly looses track of the signal.

The second carrier tracking test involved feeding four different tracking loop with the same signal, but deviating the supplied initial carrier frequency. These deviations were 0 chips, 1 chip, 3 chips and 5 chips away from the actual code phase of 1000.

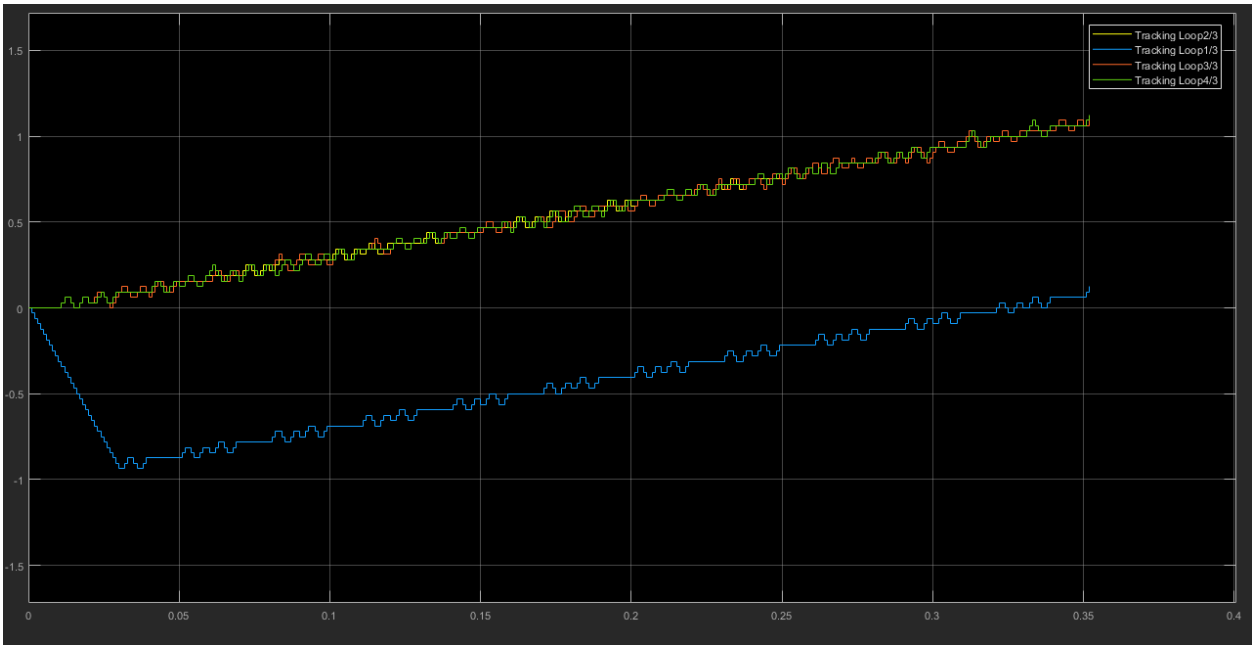


Figure 16: Code tracking discriminator (Y):Δ0 chips, (B): Δ1 Chip, (G): Δ 3 chips, (R): Δ5 chips

As can be seen in Figure 16, the code loops are able to track on to signal with a one chip shift in approximately 30ms. For code phase errors of more than two chips, the loop failed to find the original offset and tracks changes to the wrong code phase. This could be due to the correlation values of incorrect code phases being too inconsistent to provide a proper shift to the original offset. This may perhaps be improved upon with a different method of adjusting the C/A code, or better smoothing of the error term. In all acquisition test cases however, the C/A code phase was always correct so we should not have to worry about the code phase suddenly shifting more than two chips.

The final code tracking test was carried out using an input signal consisting of a sum of all 4 generated signals. The purpose of this testing stage is to determine if the loop can track changes to four different signals being received at the same time. Achieving this will help to ensure that the system will coherently demodulate data from multiple signals in final testing.

The test signals used to generate Figure 17 are the ones listed at the beginning of this section in Table 3, and correspond to the order they appear in the legend of the plot.

Signal 1 (Y)+3chips/s

Signal 2 (B)+3chips/s

Signal 3 (R)-3chips/s

Signal 4 (G)+3chips/s

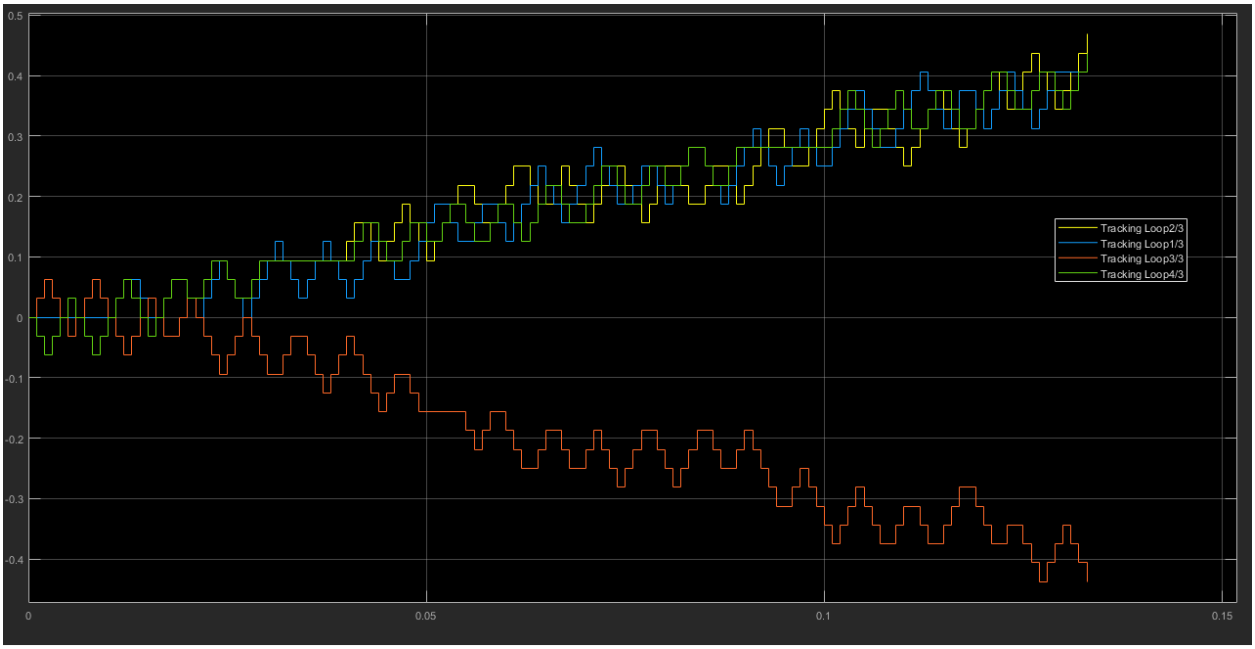


Figure 17: Code tracking discriminator for 4 different signals

To validate the function of this loop, we must observe the code phase error signals plotted in Figure17. Since the slope of the average of each signal has an approximate linear relation to the doppler shift rate, it is evident that the tracking system is able to accurately measure changes to one signal in the presence of others. We must now test the carrier and code tracking loops together.

11.3 Combined Signal Testing

The last test needed to complete the validation of our tracking stage involves summing 4 different signals as input. The signal parameters are passed to tracking from a pre-set array with initial values; These values represent the test signal parameters (as seen in Table 3 . The goal of this final stage is to recover the transmitted data bit vectors listed below.Note that the signals are numbered by their order of appearance in the plot legend, the bracket indicated the color in the graph.

Signal 1 (Y)[1 0 0 1 1]

Signal 2 (B)[[1 0 1 0 1]]

Signal 3 (R)[0 1 1 0 0]

Signal 4 (G)[1 0 0 1 0]

Figure 18 shows the resulting demodulate waveform for signals 1 and 2, while Figure 19 shows the resulting waveform for signals 3 and 4. The signals only exhibit sign (bit) changes every 0.1s (100Hz) which is the rate of data that we chose when generating the signal. This is the first sign that demodulation is being performed properly. Another good sign is the lack of correlation attenuation over time. If the loop were not performing properly this correlation value would drop over time towards zero. The last step of validation is to match the generated data vectors to their demodulated signals. Note that the demodulated waveform is in the non-return to zero format (on the set [-1,1]). Through visual comparison it is clear that the vectors do all indeed match up

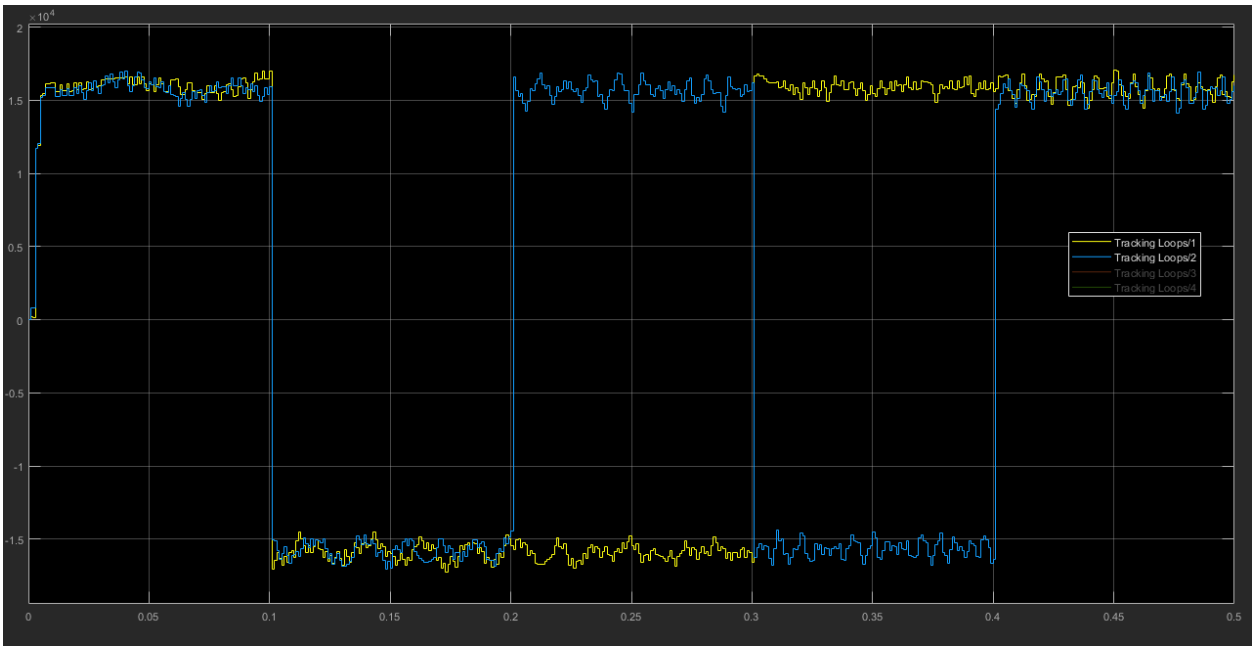


Figure 18: Final Tracking Loop Test Result Plot

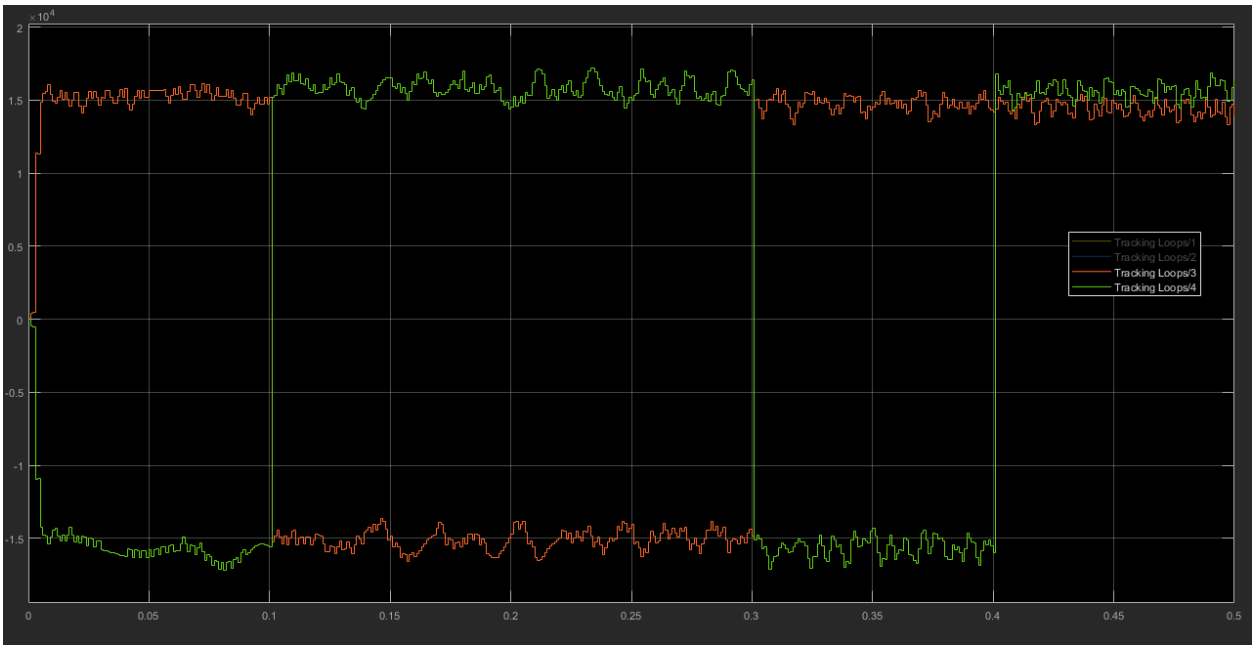


Figure 19: Final Tracking Loop Test Result Plot

correctly, however the bits have been inverted for signals 3 and 4. This is perfectly fine however, as it is a property of working with BPSK signals which are encoded with a 180 deg phase shift, and the costas loop. The bit inversion must be detected by the data navigation extraction stage before any useful data can be extracted.

12 Synthesis with Acquisition Stage

The final phase of this project was to test both the acquisition and the tracking stages together. The information used for tracking initializations must now come from the acquisition results instead of being pre-set (as in previous testing stages). To make this synthesis function as intended, some overhead logic is needed to control both models, as well as store and sort satellite parameters. Testing results and details of the adjustments required to successfully synthesize both models is provided in the following sections.

12.1 Adjustments to Acquisition and Tracking

To gather useful information from the correlation output, a matlab function block was written which stores the code phase of the highest correlation value at every frequency step. The largest value per C/A code is found in a similar manner, allowing us to locate both the code phase and frequency step of the highest correlation peak. If this peak is determined to be above our threshold ($2 * 10^7$, found through correlation data analysis) it is added to an array of acquired signals.

As the array is filled, the signals should be organized according to signal strength; This can be achieved by looking at either correlation peaks or the SNR value. The organized array is then output from the loop, where it is written to a variable via a data set memory block. This shared memory block will then be read out during tracking and used to initialize the loop parameters, as shown below.

The acquisition block controls the start of tracking stage execution through a wired boolean enable line. The acquisition loop will also terminate itself when it is no longer needed. The tracking stage of a real GPS receiver would ordinarily not start until the best suited satellites are found, which requires a full sweep through the 32 possible satellites. For the purposes of our testing

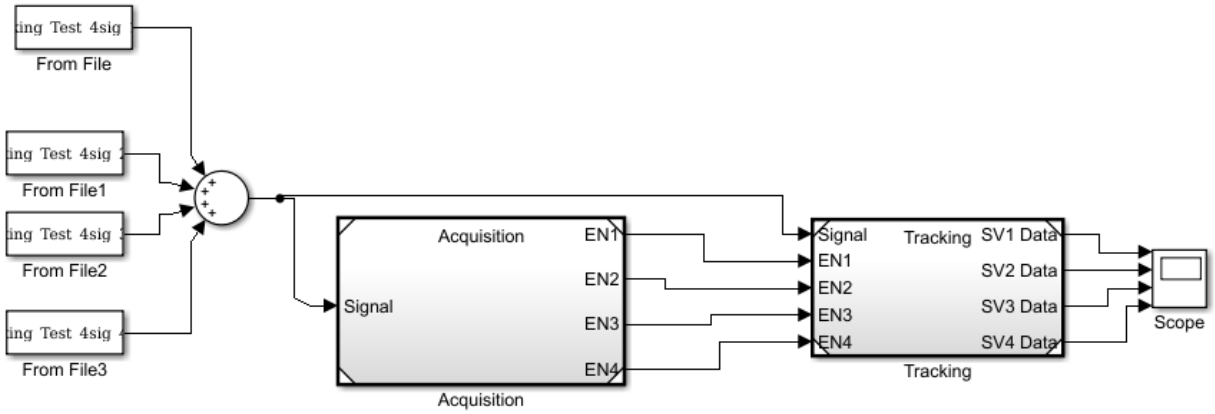


Figure 20: Synthesized receiver model

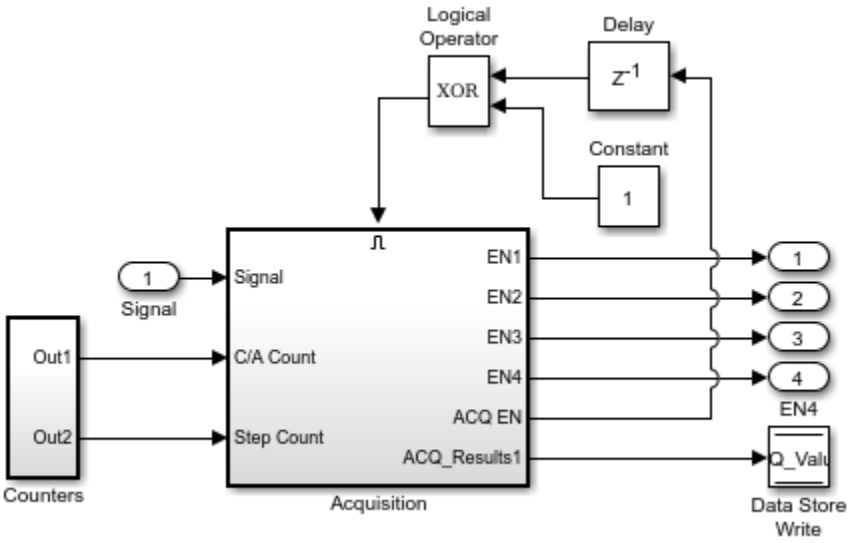


Figure 21: Overhead logic used to control acquisition model

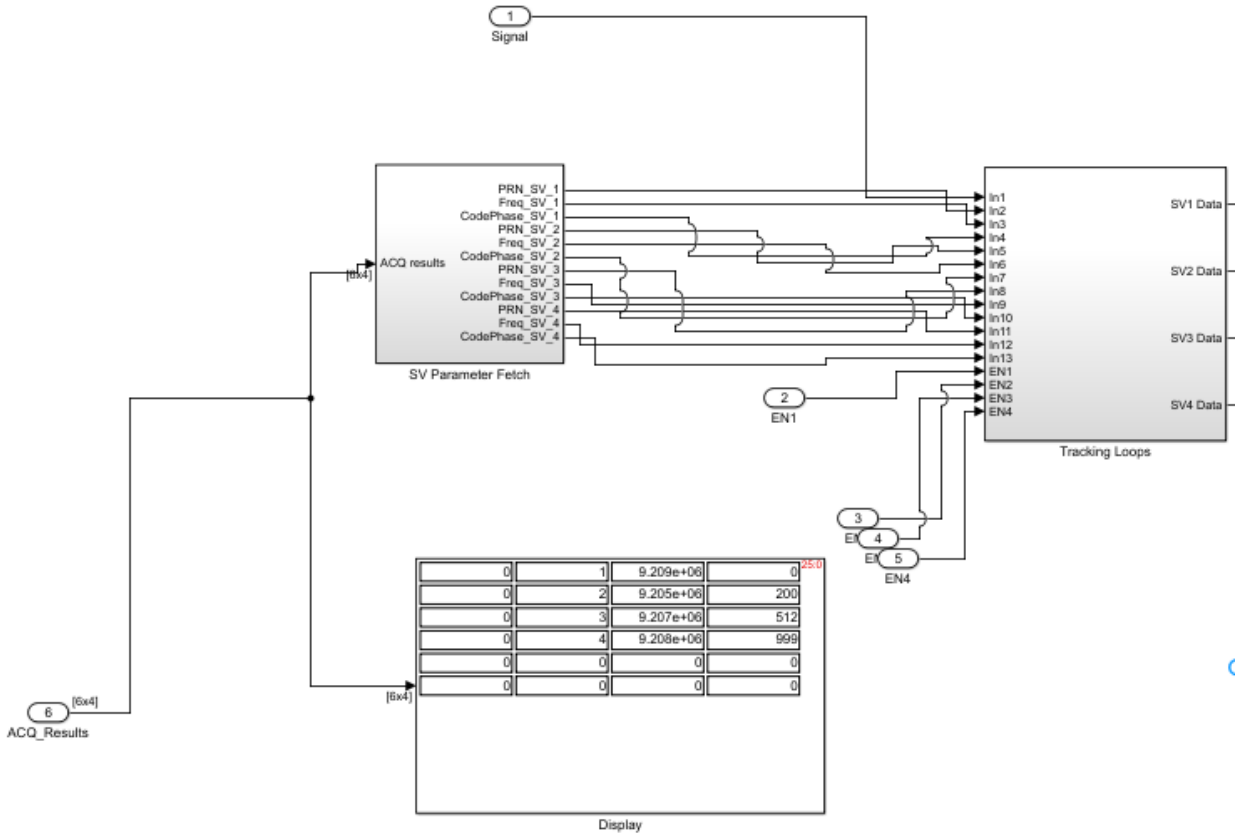


Figure 22: Tracking loop initialization

however, the acquisition stage will enable the tracking of signals as they are found. This allows us to reduce the computation time required to view the results of tracking.

12.2 Final Test

The final test was performed on the sum of 4 generated signals. Unlike previous tests, this time all of the tracking initialization parameter values come from the results of the acquisition stage. All of the signals have a data rate of 100Hz (real GPS uses 50Hz, changed to reduce required computation time) as well as different 5 bit repeating data vectors. All generated signal specifications are available in Table3 of section 11, reproduced below. The goal of this final stage testing is to simply provide an input signal and see if the recovered data bits match the generated ones. Signals are numbered according to the order they appear in their respective plot legends.

Satellite Number	Carrier Frequency	Code Phase	Data Vector	Color
1	9.209MHz (+50Hz/s)	0 (+3chips/s)	[1 0 0 1 1]	Y
2	9.204Hz (-100Hz/s)	200 (+3chips/s)	[1 0 1 0 1]	B
3	9.207MHz (+200Hz/s)	512 (-3chips/s)	[0 1 1 0 0]	R
4	9.207MHz (+100Hz/s)	999 (+3chips/s)	[1 0 0 1 0]	G

Table 4: Test Signal Parameters

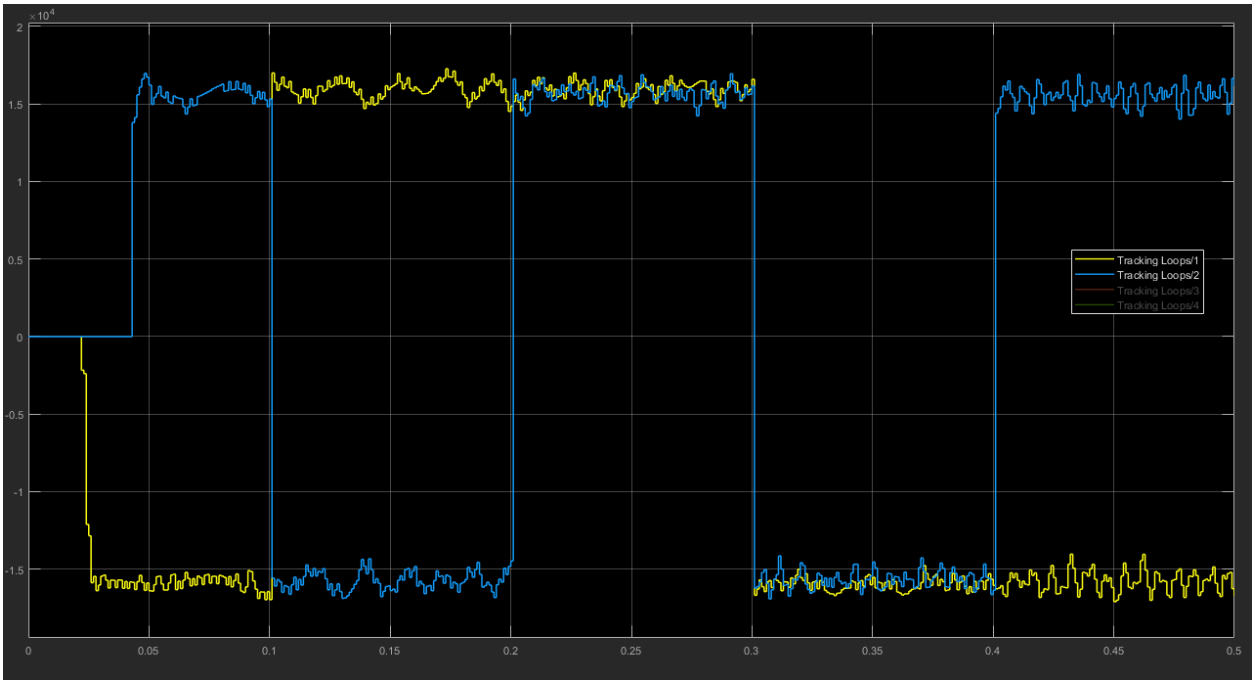


Figure 23: Receiver Model Data Results Signal 1 & 2

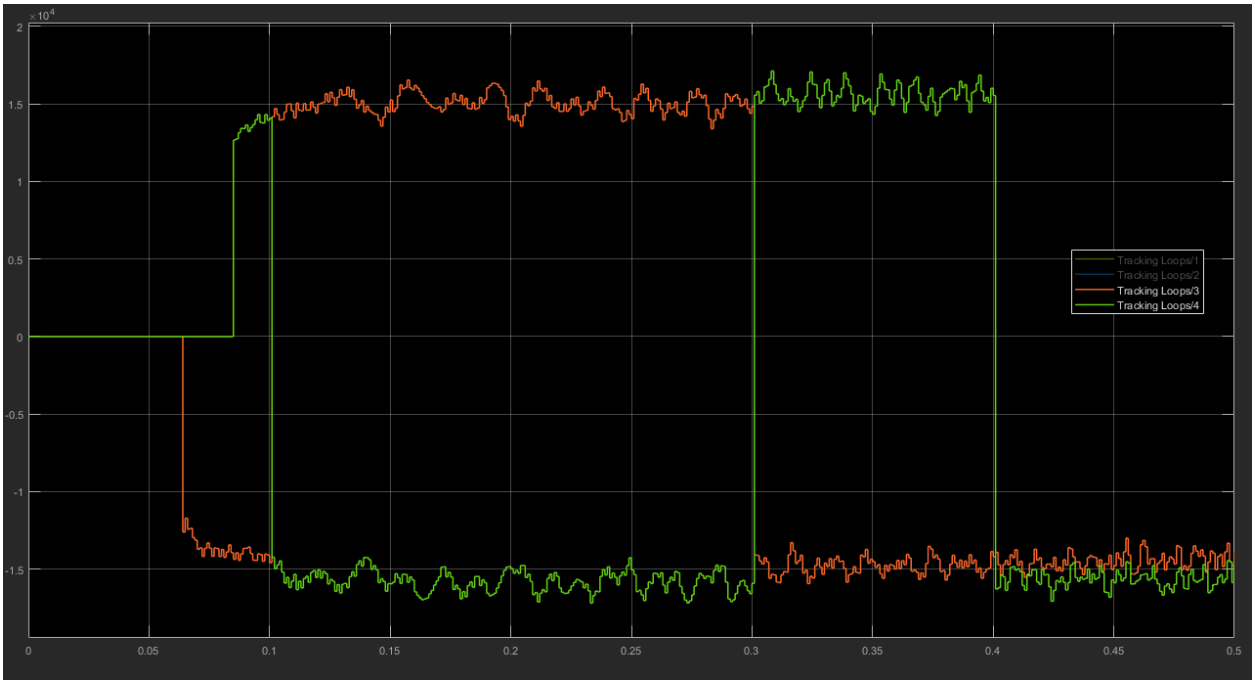


Figure 24: Receiver Model Data Results Signals 3 & 4

The resulting plots of carrier discriminator (Figure 25), code discriminator (Figure 26) and demodulated data bits (Figures 23 & 24) show a delay in signal tracking from simulation start time. This delay is caused by the time taken for acquisition to find each signal and pass on the parameters, enabling the tracking loop to begin. The demodulated data, carrier discriminator and code discriminator values agree with the previous values of tracking stage testing, as well as the generated values. Through this final simulation we have validated that our system does indeed work as designed.

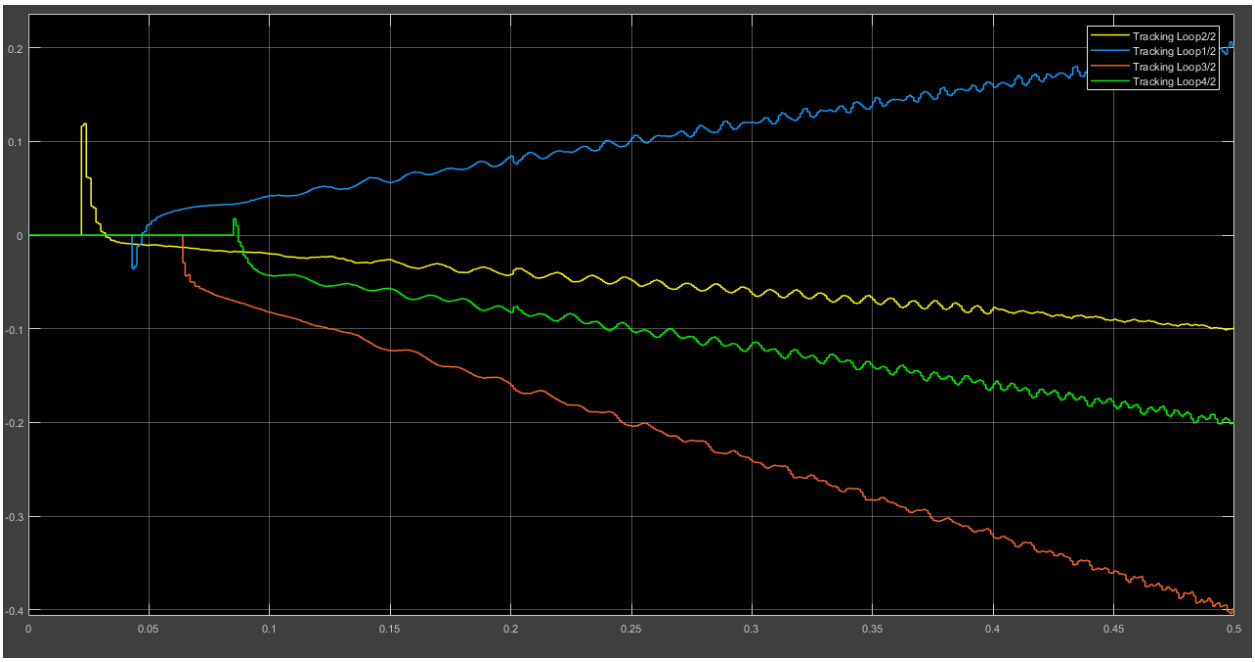


Figure 25: Receiver Model Carrier Discriminator Outputs

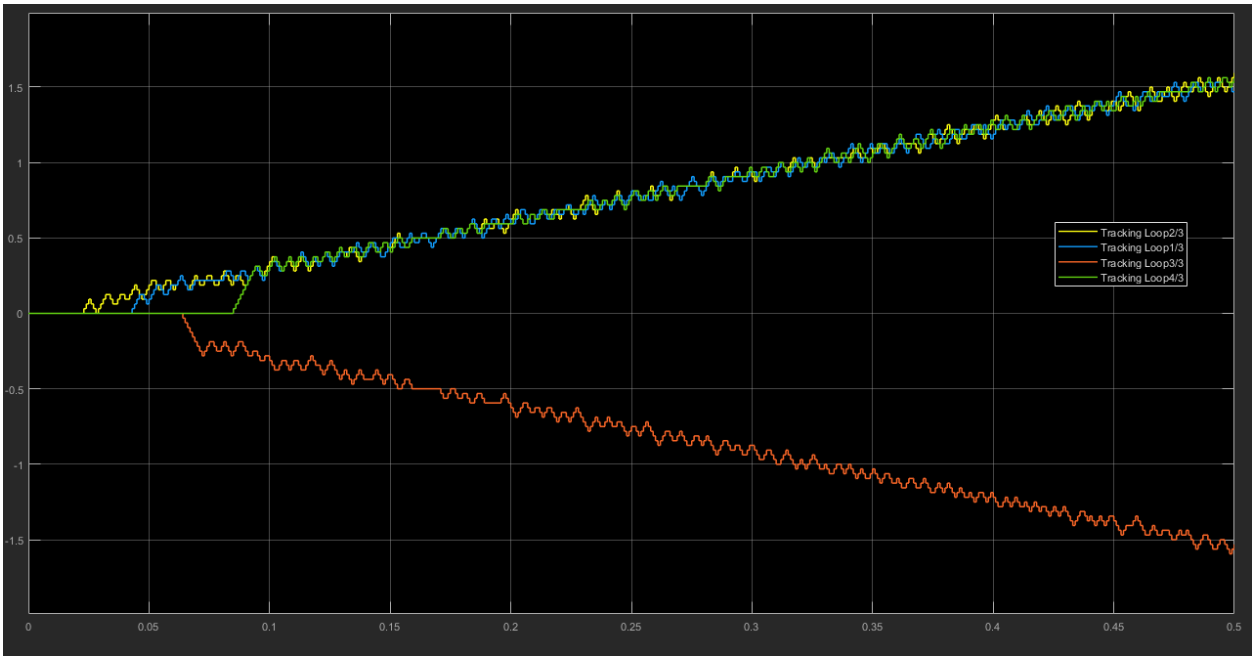


Figure 26: Receiver Model Code Discriminator Outputs

13 Data Navigation Extraction Overview

The output of the tracking stage is the in-phase arm of of the tracking block. These values need to be truncated to the set of [-1,1] (based on their sign). Due to noise or weak signals the values will be slightly distorted, so the mean value over 20ms (data rate) should be used to truncate values.The next step is to determine when a bit transition happens by detecting a zero crossings. Once the beginning of a bit transition is known, the 1000 Hz samples from tracking can be correctly down-sampled to 50Hz navigation data bits.

After the data has been properly converted, the beginning of a subframe must be located. This is marked with an 8-bit preamble of [10001011]. As mentioned earlier the costas loop is invariant to 180 deg phase shifts, leading to possible bit inverted version of [01110100]. Correlating the incoming data with this preamble will determine when the data sub frame starts, and whether the bits are inverted or not.

Each sub frame contains 300 bits divided into 10 words of 30-bits each. 24 bits of each word contain actual data, while the rest belong to a 6-bit parity. This parity is used to confirm whether data whether received data was interpreted correctly. If the parity check is successful the navigation data can be decoded using information provided in the IS-GPS-200. The decoded data can then be used for the required calculations to determine position.

14 Impact on Society and the Environment

Due to the nature of the project, there are no real or tangible impacts or costs to consider. However we can say that our project benefits society and future students as an educational tool by allowing them to visualize and dissect the inner workings of a GPS receiver. Through the use Simulink, we allow other users to analyze our systems at every stage of the receiver and allow them to easily build upon or replicate in other applications such as a real-time FPGA. We demonstrate the possibility of implementing and understanding general complex electrical systems in a graphical programming environment such as Simulink

On a more general note, GPS and Satnav systems as whole have had a profound impact on society and the environment. GPS has completely transformed the way urban planning, navigation and transportation systems are dealt with. It has contributed to an increase in quality of living and has led to better cartography and assessment of environmental changes such as deforestation. Satnav and GPS systems are heavily impacting future technologies, for example, its aid in the development of self driving cars.

15 Report on Teamwork

Teamwork, collaborating and work delegation was quite seamless as we were quite familiar with the project and what was required of us this semester. Both members had the initial task of researching and understanding the fundamentals of the tracking stage. We then split up the work needed to be done for the tracking system into the obvious division of carrier tracking and code phase tracking. Other requirements were delegated equally from updating the test signal generator, to testing with different parameters. A significant amount of testing was done together and meetings were frequent in order to update the other member on the accomplished progress, share

resources and explain when necessary.

Problems arose infrequently but were mainly due to differing or unclear understanding on some of the general concepts, this was always resolved by explaining our understanding to one another and coming to a mutual conclusion.

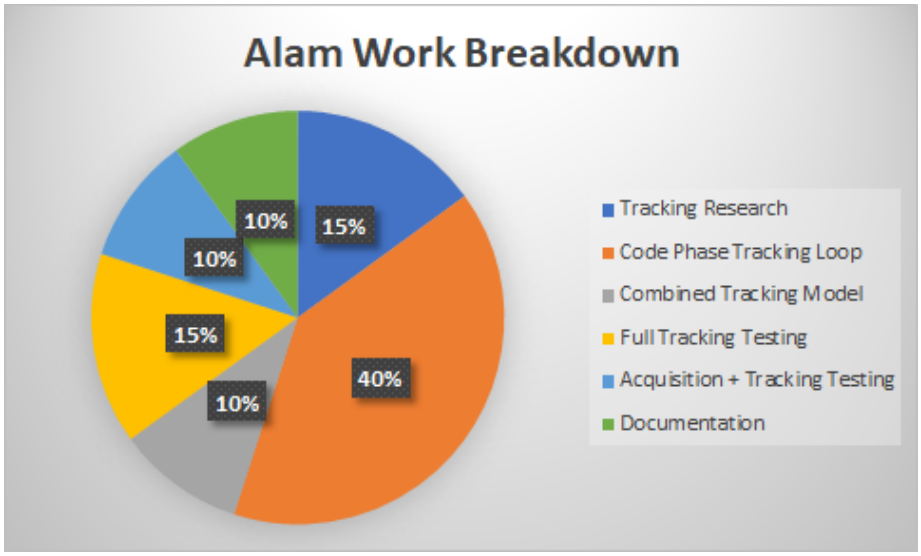


Figure 27: Alam Work Breakdown

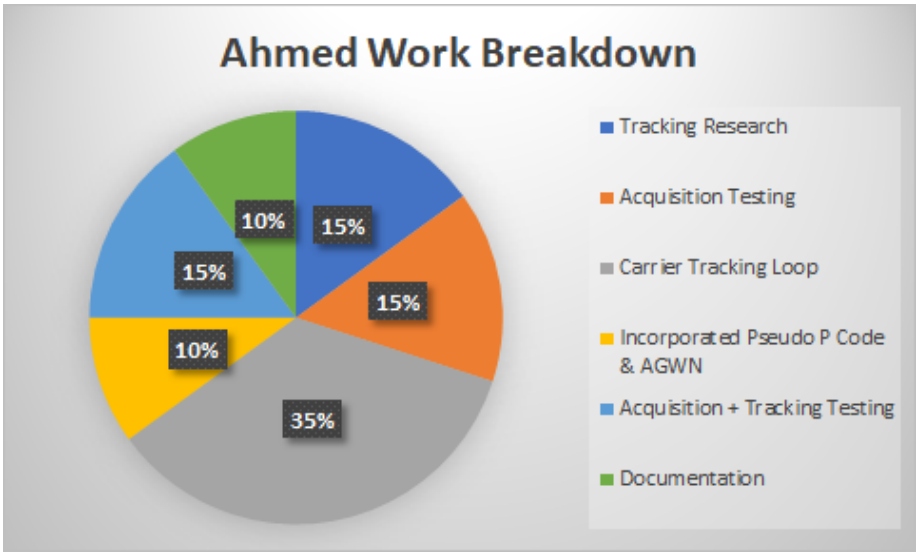


Figure 28: Ahmed Work Breakdown

16 Conclusion

The beginning of the project involved a lot of background research into the nature of GPS systems and GPS signal generation. Once these concepts were better understood we were able to begin our research into how a receiver actually works. From there we were able to work on accomplishing our main goal of implementing the receiver in a manner that makes it easier to examine and analyze.

The primary focus of this last semester was to further develop our receiver model started in semester 1, by implementing a fully functional signal tracking block. Based on the results of our rigorous testing, we confirmed the ability to simultaneously demodulate multiple simulated GPS signals from a single input source.

The focus of this project as a whole was to create a functional GPS receiver simulation model containing the acquisition and tracking stages. Through the integrated testing of both blocks, we can confidently say that we have completed this goal within our set of constraints and assumptions.

If given enough time, or perhaps if we had another group member, we would have liked to start the data navigation extraction stage. This is the last main stage before the actual positions can be calculated, and is crucial to having a full receiver model. Although we did not get to this stage, we have set up our project in a manner that will easily allow others to build upon it in the future. Our hope is that this simulink implementation could perhaps be used as a basis for a project involving an FPGA implementation of a receiver.

17 References

References

[1] K. Borre, A Software-Defined GPS and Galileo Receiver. Boston, MA: Birkhauser Boston, 2007.

[2] L. Patel and J. Patel, "Digital Implementation of Costas Loop with Carrier Recovery", International Journal of Engineering Research and Development, vol. 11, no. 2, p. 22, 2015.

[3] G. Hamza, A. Zekry and I. Motawie, "Implementation of a complete GPS receiver using simulink", IEEE Circuits and Systems Magazine, vol. 9, no. 4, pp. 43-51, 2009.

[4] J. Betz, Engineering satellite-based navigation and timing. .

[5] J.Tsui, Fundamentals of global positioning system receivers. Hoboken: Wiley, 2005.

[6] F. Van Diggelen, A-GPS. Boston: Artech House, 2009.

[7] Interface Specifications: IS-GPS-200. Navstar GPS Space Segment/Navigation User Interfaces, 2013.