GPS Spoofing and Anti-Spoofing Techniques

Aditya Kumar (210070003) Joel Anto Paul (210070037)

Guide: Prof. Sibiraj Pillai

Abstract

This report investigates advanced GPS spoofing methodologies that go beyond traditional techniques by focusing on signal cancellation and synthetic signal injection. We present a series of experiments based on GNSS-SDR and GPS-SDR tools, exploring the challenges in detection, correlation, fine Doppler estimation, and signal cancellation to demonstrate the viability of spoofing a GPS receiver without increased signal power. The code has been made available on Github at `https://github.com/SnarkyGoblin123/BTP.git`.

0.1. Introduction

GPS spoofing involves manipulating GPS signals to mislead a GPS receiver's position fix. While traditionally viewed as a vulnerability, this area also has significant defense applications. Understanding spoofing helps us devise anti-spoofing strategies and strengthen navigation systems against such attacks.

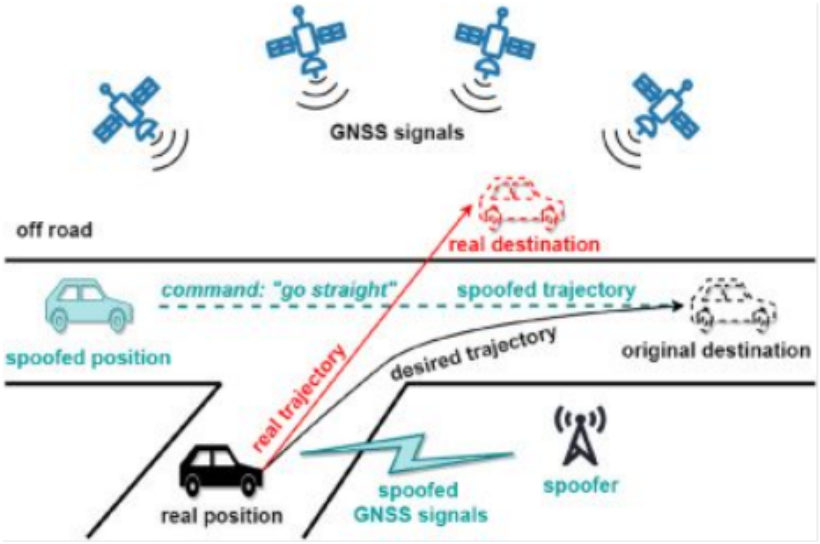

Figure 1: Illustration of GPS spoofing methodology.

0.2. Problem Setting

A GPS receiver obtains signals from multiple satellites, each transmitting synchronized signals. Let Y be the composite received signal and x_i the contribution from satellite i. The received signal can be modeled as:

$$Y = \sum_{i=1}^{N} \alpha_i x_i e^{j2\pi f_i t} + \nu(t) \tag{1}$$

Where:

- N is the number of satellites.
- α_i is the unknown amplitude of the signal from satellite i.
- f_i is the Doppler frequency shift for satellite i.
- t is time.
- $\nu(t)$ is the background noise affecting the received signal.

The goal is to estimate the α_i values for each satellite, allowing us to manipulate the received signal. Before diving into the details, we will first discuss the GNSS-SDR framework and the GPS signal structure.

0.3. GNSS-SDR Framework

GNSS-SDR 1 is an open-source, modular software-defined receiver for Global Navigation Satellite Systems (GNSS). It allows real-time processing of raw signal samples captured from radio front ends and outputs positioning and navigation data.

0.3.1. Architecture

- **Modular Design:** Composed of blocks that can be configured and connected in various ways.

- **Signal Processing Blocks:** Each block performs a specific function, such as signal acquisition, tracking, and decoding.
- **Data Flow:** Data flows through the blocks, with each block processing the signal and passing it to the next.
- **Configuration:** Users can configure the receiver to work with different GNSS constellations and signal types.
- **Real-time Processing:** Capable of processing live signals in real-time, making it suitable for various applications.
- **Open-source:** The source code is available for modification and experimentation, allowing researchers to develop new algorithms and techniques.

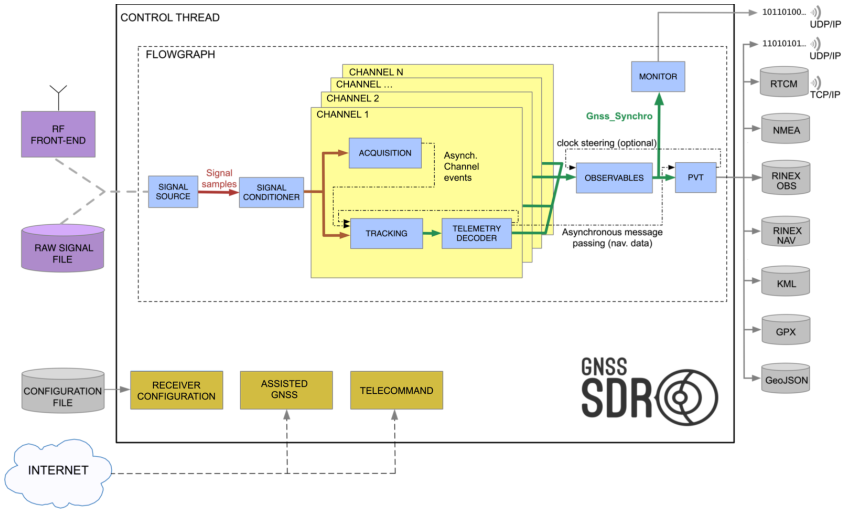

Figure 2: GNSS-SDR Framework Overview.

0.3.2. Core Components

- **Signal Source:** Interfaces with RF front ends or file-based input.
- **Acquisition Block:** Identifies visible satellites and estimates initial Doppler and code phase.
- **Tracking Block:** Maintains lock on satellite signals and refines Doppler/code phase.
- **Decoding Block:** Extracts NAV data including ephemeris and almanac.
- **PVT Computation:** Computes Position, Velocity, and Time (PVT) using decoded satellite data.

0.3.3. Advantages

- High modularity for rapid experimentation.
- Supports both GPS and Galileo constellations.
- Allows testing with both live and synthetic datasets.

In addition to the above features, GNSS-SDR also provides logging capabilities which helps us identify what went wrong, which satellite signals are present and finally the RINEX observation files provide useful data such as the doppler shifts. It also outputs the telemetry bits which can be used in exoperiments to verify the presence of the satellites.

0.4. GPS Signal Structure

0.4.1. NAV Data Hierarchy

- Each satellite transmits data in 1500-bit **frames**, repeated every 30 seconds.
- Each frame is composed of 5 **subframes**, each 300 bits and 6 seconds long.
- Each subframe consists of 10 **words** of 30 bits each.

- Each word contains various types of information, including ephemeris data, almanac data, and health status.
- The first word of each subframe contains a 8-bit preamble (10001011).

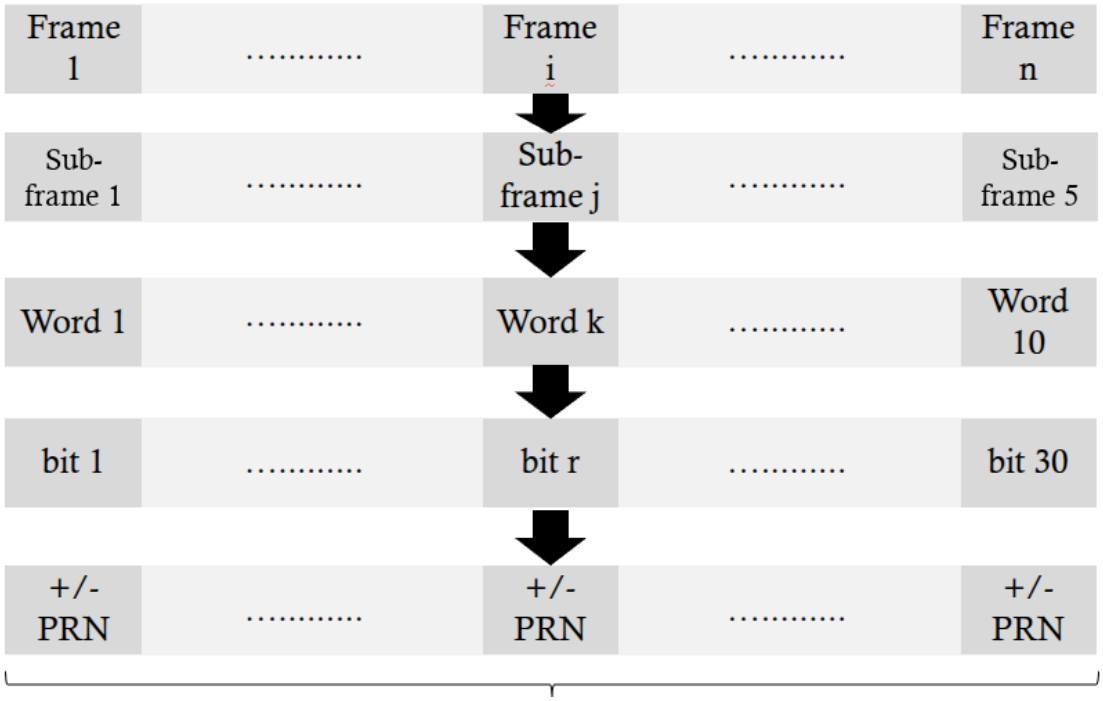

Figure 3: Structure of GPS NAV Data Hierarchy.

0.4.2. Transmission Rate

- NAV data: 50 bits per second (bps)
- Each NAV bit is modulated with a PRN sequence that repeats every millisecond.
- Thus, each bit is spread over 20 repetitions of the PRN code (20 ms).

For better understanding of the GPS signal structure, we can refer to the IS-GPS-200K document 2 which provides detailed information about the GPS signal structure.

0.5. PRN Codes

Pseudo-Random Noise (PRN) 3 codes uniquely identify each GPS satellite and allow for code division multiplexing of signals.

0.5.1. Key Characteristics

- Binary sequences of length 1023, repeating every 1 ms.
- Generated using two 10-bit Linear Feedback Shift Registers (LFSRs).
- The output is the modulo-2 sum of the outputs of the two LFSRs at specified tap positions (G1 and G2).
- The choice of G2 tap combinations determines the satellite number (SVN).

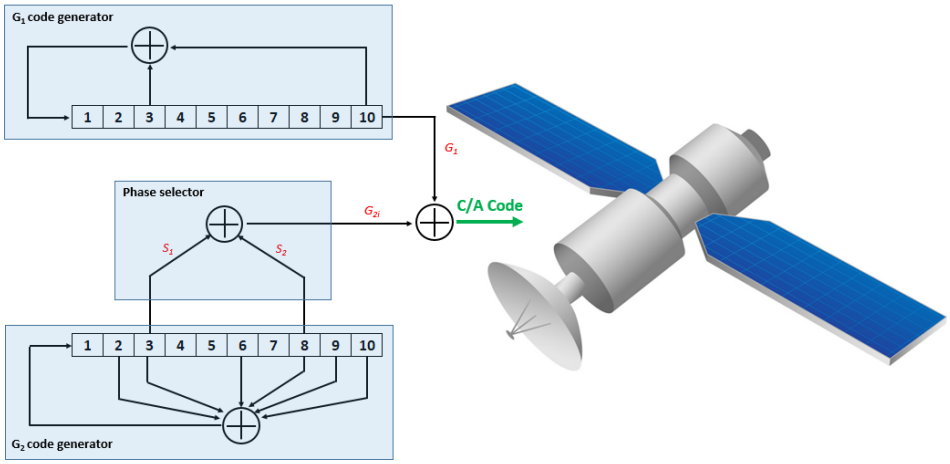

Figure 4: Generation of PRN Codes using LFSRs.

0.5.2. Orthogonality

- PRNs are approximately orthogonal — minimal cross-correlation with others, enabling simultaneous decoding.
- Orthogonality allows the receiver to correlate each code separately and isolate satellite signals.

0.5.3. Use in Correlation

During acquisition and tracking, the receiver cross-correlates incoming signals with locally generated PRNs (shifted and Doppler-adjusted). Peaks in the correlation output indicate the presence and timing of specific satellites.

0.6. Data Sources

- **Real-time data:** Captured using RF front ends and processed with GNSS-SDR or stored via GNU-Radio.
- **Real datasets:** Downloaded from public repositories.
- **Synthetic datasets:** Generated using GPS-SDR for controlled experimentation.

0.7. GPS-SDR-SIM

GPS-SDR-SIM is a software-defined GPS signal simulator that generates synthetic GPS signals for testing and research purposes. It allows users to create custom scenarios, including multiple satellites, different Doppler shifts, and various signal conditions.

0.7.1. Key Features

- Generates GPS L1 C/A signals.
- Configurable parameters: number of satellites, Doppler shifts, signal power, and noise levels.
- Outputs raw I/Q samples for further processing.
- Useful for testing GNSS receivers and algorithms in controlled environments.
- Can simulate various scenarios, including multipath effects and jamming.

0.7.2. Use in Experiments

We used GPS-SDR-SIM to generate synthetic signals for our experiments, allowing us to control the parameters and conditions of the received signals. This facilitated the testing of our signal cancellation and reconstruction techniques in a controlled environment. In our experiments, we had to control the gain of the satellite signals. To do this, in the gps-sdr-sim 4, change 'gpssim.c', at line 2301, add:

```
gain[i] = gain[i]*k;
```

where 'k' is the gain ratio we want to set. This allows us to control the gain of the satellite signals in the simulation. we can also change gains of each satellite individually.

0.8. Experiments

0.8.1. Experiment 1: Satellite Presence Verification

Aim: Verify presence of satellite 10 in the data.
Method: Upsample PRN code of satellite 10, correlate with raw data (300k samples).
Result: No visible peaks.

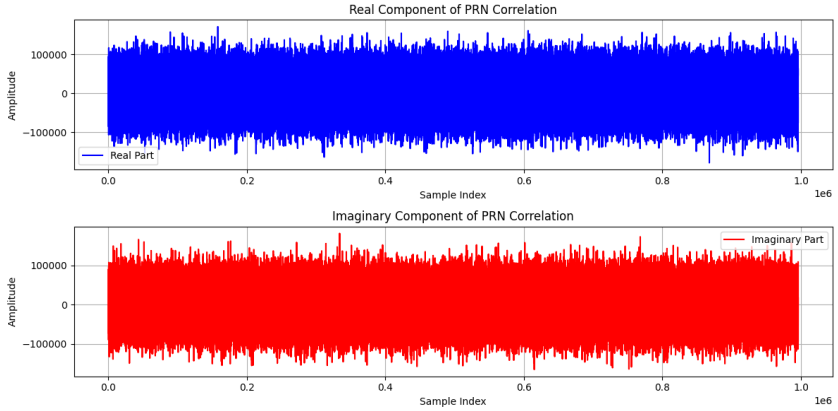

Figure 5: Correlation results for Experiment 1.

Analysis: GNSS-SDR confirmed presence of satelllite 10, but we did not see the peaks. Why? Because doppler shift was not compensated in the GPS Data while correlation, thus the correlation averaged out to just the noise.

0.8.2. Experiment 2: Doppler-Compensated Correlation

Aim: Repeat Experiment 1 with Doppler correction.
Method: Use approx. 1 million samples.
Result: Clear peaks observed, confirming satellite presence.

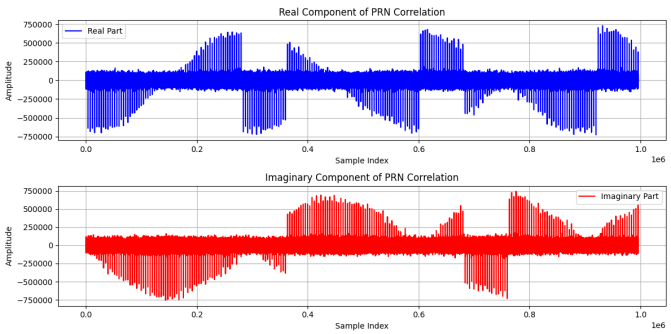

Figure 6: Correlation results for Experiment 2 with Doppler correction.

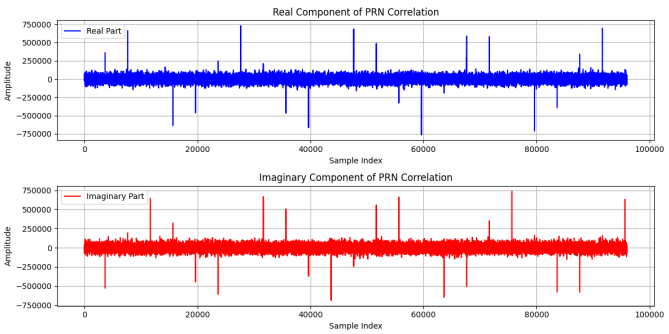

Figure 7: Doppler shift if compensated in the up-sampled PRN code.

Note: When incorporating the Doppler shift, incorporate it into the GPS Data. Do not compensate it in the upsampled PRN code

$$Y[n] = \sum_{i=1}^{N} \alpha_i x_i[n] e^{j2\pi f_d n/f_s} + \nu(n) \tag{2}$$

$$V[n] = PRN(10)_{upsampled} \tag{3}$$

where $Y[n]$ is the received signal, $x_i[n]$ is the PRN code, f_d is the Doppler frequency, and f_s is the sampling frequency. If we compensate the Doppler shift in the upsampled PRN code, we will not get the correct correlation peaks, which can be seen in these equations:

$$V[n] = V[n] e^{j2\pi f_d n/f_s} \tag{4}$$

$$Corr = \langle Y, V \rangle \tag{5}$$

Here n in (4) will correspond to the time of the PRN code which is of the length $f_s/1000$, thus giving incorrect correlation peaks, whi The incorrect results can be seen in the figure 7. Here we dont see 20 equally spaced peaks.

0.8.3. Experiment 3: Subframe Detection

Aim: Detect subframe using 8-bit preamble (10001011).
Challenges: Memory issues with large datasets (4.8 GB). Takes long time to process correlation
Solution: Used `scipy.fftconvolve` to reduce processing time from hours to few minutes. Takes more memory, therefore employed chunked processing.

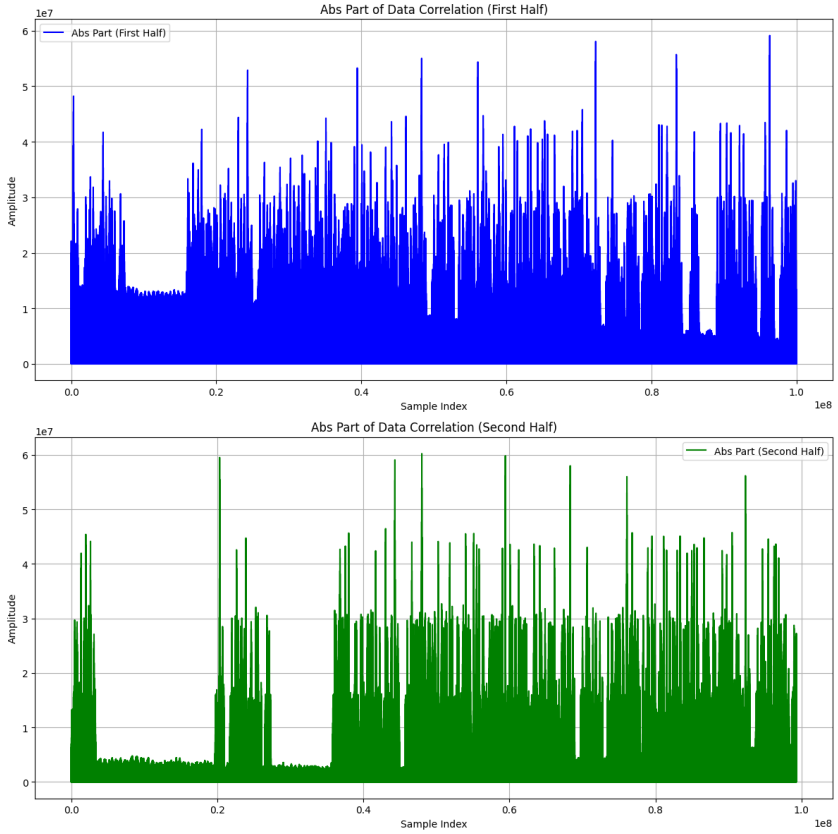

Figure 8: Subframe detection using 8-bit preamble.

0.9. Fine Doppler Correction

Ideally doppler correction should remove imaginary peaks from the correlation, however, we still see some peaks. This is because the doppler correction is not perfect. Therefore what we did is, wrote a

script which pinpoints the doppler shift to 0.01Hz accuracy. Now, we would expect to use the doppler shift which gives maximum absolute correlation, but because of noise, we will not get that, therefore what we do is keep a threshold for imaginary peaks (known apriori) and then if all imaginary values are below this threshold, take the average imaginary values and then minimize it. The pseudocode is below:

Algorithm 1 Search Doppler Shift

Require: compdata, st, chunk, fd, fs, search_param = $[f_{min}$, step, $f_{max}]$, imag_threshold
Ensure: best_shift, max_correlation

best_shift $\leftarrow 0$
max_correlation $\leftarrow 0$
prev_min $\leftarrow 10^7$
curr_min $\leftarrow 10^7$
best_corr $\leftarrow$ None
for doppler_shift from f_{min} to f_{max} with step size **do**
    shifted_data $\leftarrow$ ApplyDopplerShift(compdata, fd, fs, doppler_shift, st)
    corr $\leftarrow$ Correlate(shifted_data, chunk)
    real_corr $\leftarrow$ RealPart(corr)
    imag_corr $\leftarrow$ ImagPart(corr)
    if AllAbs(imag_corr) $<$ imag_threshold **then**
        curr_min $\leftarrow$ Mean(Abs(imag_corr))
        if curr_min $<$ prev_min **then**
            prev_min $\leftarrow$ curr_min
            best_shift $\leftarrow$ doppler_shift
            best_corr $\leftarrow$ DeepCopy(corr)
        end if
    end if
end for
if best_corr is None **then**
    Print("No valid correlation found")
    return best_shift, max_correlation
end if
return best_shift, max_correlation

The results are as follows:

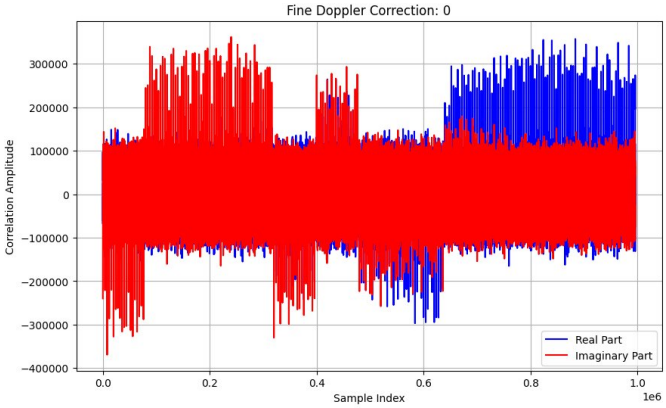

Figure 9: Correlation peaks before fine Doppler correction.

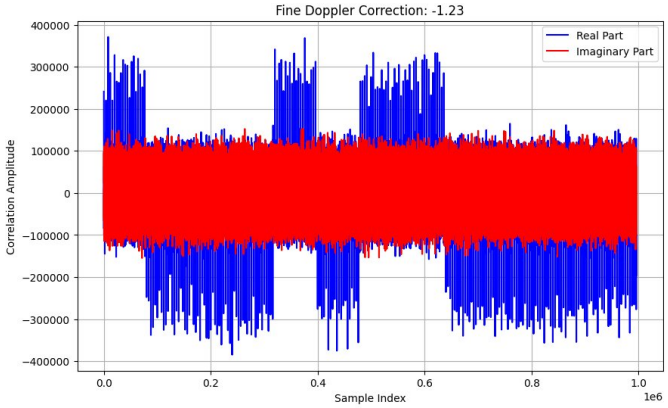

Figure 10: Fine Doppler correction: Real correlation peaks.

0.10. Signal Cancellation and Alpha Estimation

0.10.1. Signal Model

Received signal: $Y = \sum \alpha_i x_i e^{j2\pi f_i t} + \text{noise}$
Where:

- x_i: known PRN sequences
- f_i: Doppler frequencies from GNSS-SDR
- α_i: unknown amplitudes to be estimated

0.10.2. Alpha Estimation

We estimate α_i using linear regression on the correlated signal.

$$\boldsymbol{\alpha} = (\mathbf{X}^H \mathbf{X})^{-1} \mathbf{X}^H \mathbf{Y} \tag{6}$$

Here:

- $\mathbf{Y}$ is the received signal vector.
- $\mathbf{X}$ is the matrix of known PRN sequences modulated with Doppler shifts.
- $\boldsymbol{\alpha}$ is the vector of estimated amplitudes.
- H denotes the Hermitian (conjugate transpose) operator.

For satellite 10, we estimated α using the correlation output. The estimated α values were:

- **Before fine-Doppler:** $\alpha = -40.19 - 81.33j$ (unrealistic)
- **After fine-Doppler:** $\alpha = 94.96 + 8j$ (close to real)

0.11. Verification Using GPS-SDR

GPS-SDR was modified to set known gain values per subframe. The gain estimations were cross-verified against injected values. Visibility of satellite varied with gain ratio (α_r/α_{orig} from 0.25 to 1.0), showing spoofing remains effective even with ∼70% of true signal strength.

0.12. Final Results

Using the estimated α, we successfully cancelled one subframe of satellite 10. GNSS-SDR was unable to decode the NAV message post-cancellation, confirming effectiveness.

```
23  Current receiver time: 9 s
24  Current receiver time: 10 s
25  Current receiver time: 11 s
26  Current receiver time: 12 s
27  Current receiver time: 13 s
28  GPS L1 C/A tracking bit synchronization locked in channel 0 for satellite GPS PRN 01 (Block IIF)
29  GPS L1 C/A tracking bit synchronization locked in channel 4 for satellite GPS PRN 05 (Block IIR-M)
30  GPS L1 C/A tracking bit synchronization locked in channel 3 for satellite GPS PRN 13 (Block IIR)
31  GPS L1 C/A tracking bit synchronization locked in channel 1 for satellite GPS PRN 12 (Block IIR-M)
32  Current receiver time: 14 s
33  Current receiver time: 15 s
34  GPS L1 C/A tracking bit synchronization locked in channel 2 for satellite GPS PRN 10 (Block IIF)
35  Current receiver time: 16 s
36  Current receiver time: 17 s
37  Current receiver time: 18 s
38  Current receiver time: 19 s
39  New GPS NAV message received in channel 0: subframe 3 from satellite GPS PRN 01 (Block IIF)
40  New GPS NAV message received in channel 4: subframe 3 from satellite GPS PRN 05 (Block IIR-M)
41  New GPS NAV message received in channel 3: subframe 3 from satellite GPS PRN 13 (Block IIR)
42  New GPS NAV message received in channel 1: subframe 3 from satellite GPS PRN 12 (Block IIR-M)
43  Current receiver time: 20 s
44  Current receiver time: 21 s
45  Current receiver time: 22 s
46  Current receiver time: 23 s
47  Current receiver time: 24 s
48  Current receiver time: 25 s
49  New GPS NAV message received in channel 0: subframe 4 from satellite GPS PRN 01 (Block IIF)
50  New GPS NAV message received in channel 4: subframe 4 from satellite GPS PRN 05 (Block IIR-M)
51  New GPS NAV message received in channel 3: subframe 4 from satellite GPS PRN 13 (Block IIR)
52  New GPS NAV message received in channel 2: subframe 4 from satellite GPS PRN 10 (Block IIF)
53  New GPS NAV message received in channel 1: subframe 4 from satellite GPS PRN 12 (Block IIR-M)
54  Current receiver time: 26 s
55  Current receiver time: 27 s
```

Figure 11: Signal cancellation results showing the absence of satellite 10's signal.

This was the end of BTP-1. Now as of now we are able to successfully subtract the preamble of a satellite signal. Now we shift our focus to remove the entire satellite signal from the GNSS data given we know the decoded telemetry bits and the doppler shift data as recorded by the GNSS-SDR. We need to find an efficient way of doing this as correlating over the entire signal will take a lot of time. For this we introduce the sliding window subtraction where we leverage the use of the fact that there is periodicity in the signal.

0.13. Extended Methodology: Sliding Window Subtraction

Given the sampling frequency f_s, each telemetry bit takes 20 ms to transmit, therefore each bit has $k = \frac{20f_s}{1000}$ chips. Therefore, after estimating the codephase estimate, and starting from the point where GNSS-SDR gives us the first doppler value, we start subtracting the telemetry bit. Since each telemetry bit takes k chips, after subtacting one bit we can skip k chips ahead and start subtracting the next bit. This is done for all the telemetry bits. The code is present in the github repository, in BTP2 - main.py.

0.13.1. Results

After running the code, we ran GNSS-SDR on the output data. We tried to subtract satellite 10, but we still saw the peaks in the correlation. Why!?

0.13.2. Analysis

- Improper aligning between the actual bit and its estimated position
- On running the codephase estimation for preambles starting position, we observed that there is a shift in the start of the subframes. Ideally, we should see that each subframe takes 6 seconds or $6f_s$ chips which for $f_s = 4MHz$ is 24 Million samples, therefore each subframe should start after 24 million samples, but we see that each subframe starts at 24 million - 44, which is weird. To reproduce this run the code phase estimate function in BTP2/main.py. The data should be loaded from the gps-sdr-sim with the rinex navigation file provided in its respective repository. The command for reproducing the data is:

```
gps-sdr-sim -e brdc0010.22n -l 30.286502,120.032669,100 -s 4e6 -o output.dat
```

- This means that as we move into the data there is a shift in the bits, which we were not able to figure out. Once this shift is corrected, we would be able to subtract a satellite successfully
- Since this is synthetic data, we believe this should not be because of the clock correction in satellites. Nevertheless, look into gps-sdr-sim code to see if they are trying to simulate the clock correction.
- In test.ipynb, the shifting of prn symbols can be seen if you run the code, its mentioned there.
- While tackling this problem, try to think openly as our reasonings might be wrong and we might have made a mistake somewhere in the code.

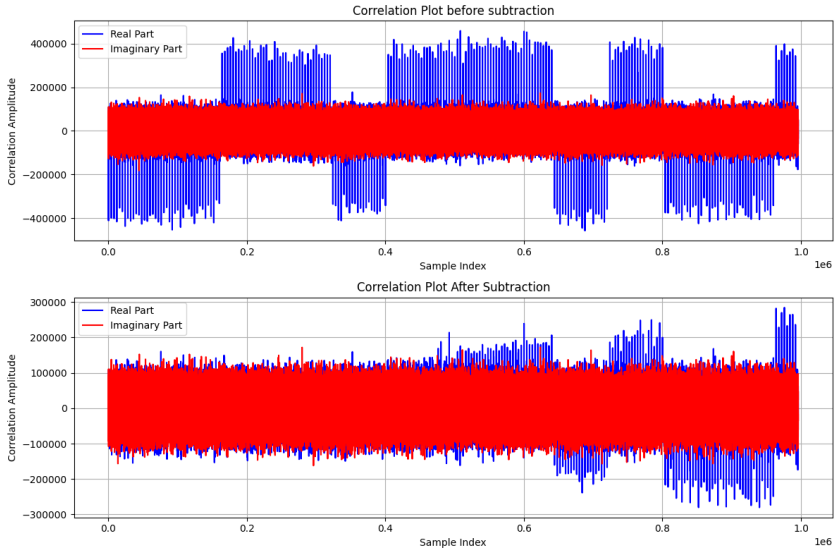

Figure 12: So basically we started subtracting from an index lets say 100, the next PRN sequence should start at 4100, or in general 4000k + 100, if we assume $f_s = 4MHz$. If there is no shift in data, after subtraction, we should ideally see no peaks, but as we move forward with time, peaks starts resurfacing

0.14. Alternate Spoofing via Signal Combination

As cancellation proved sensitive, an alternative method was explored:

- Generate two paths using `gps-sdr-sim`
- Create a combined signal: $y = \alpha x_1 + (1 - \alpha)x_2$
- Tune α until spoofed path dominates (Take minimum alpha which satisfies this condition)
- Generate: $y = x_1 + \frac{1-\alpha}{\alpha}x_2$

How did we generate paths?

- GPS-SDR-SIM's repository provides a method
- Go to satgen folder, there they ask us to install satgen v3
- Use this application to generate an NMEA facilitated
- Use google earth to generate a kml file which is fed into SatGen v3
- Finally, after producing the NMEA file, follow the steps provided in the gps-sdr-sim/satgen folder to generate a csv facilitated
- Then run gps-sdr-sim to generate data using the csv file

Figure 13: Blue is the original path, yellow is the spoofed path.

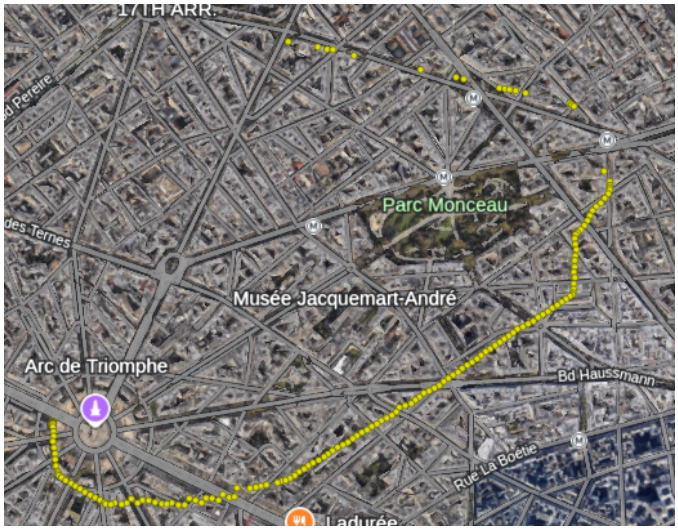

Figure 14: Spoofed Path

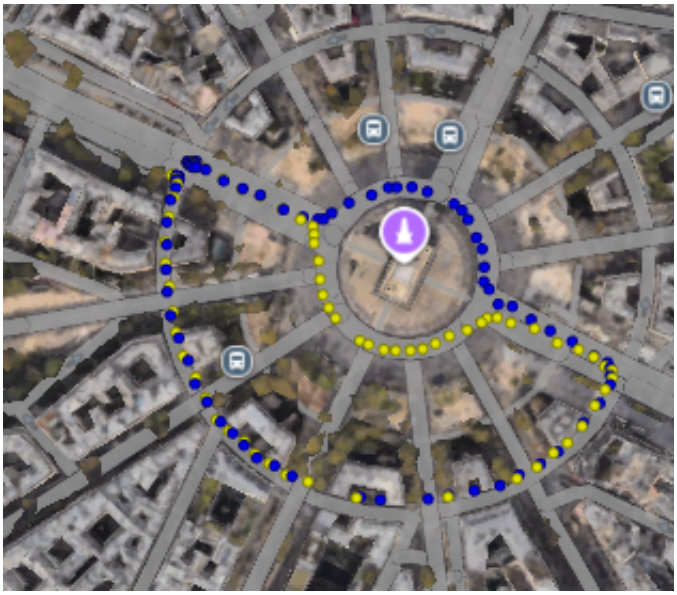

Figure 15: Blue is spoofed path, yellow is the original path.

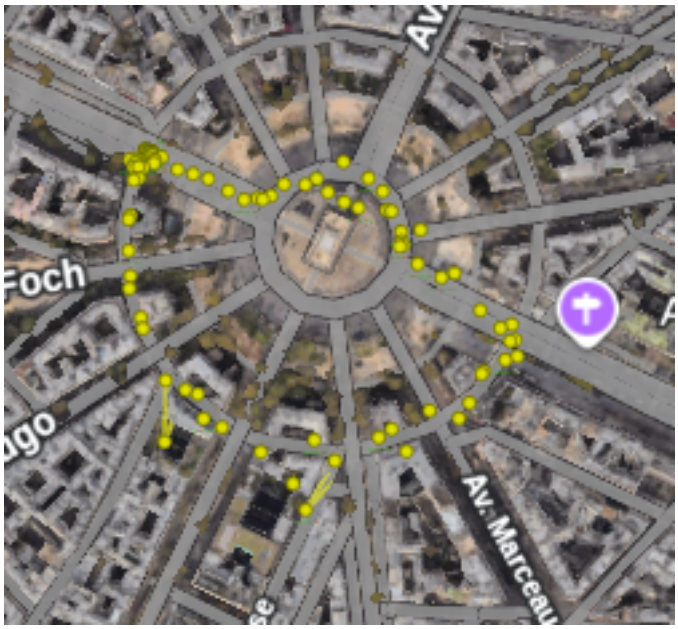

Figure 16: Spoofed Path

0.14.1. Results

- After running the above approach, we were able to spoof the receiver as GNSS-SDR locked on to the spoofed path
- Required spoofing signal to be 50% stronger than authentic

Since we are able to spoof a receiver, we now look into methods of anti-spoofing methods for this spoofing technique

0.15. Anti-Spoofing Methodology

Spoofing detection techniques We can use two methods:

- Compare live correlation with baseline magnitude
  - if correlation is atleast twice as strong as the baseline, we can assume we are being spoofed
  - Double because there are two signals and its reasonable to assume that spoofed signal is atleast as strong as the original signal
  - Here baseline correlation plot is something measured in past, so it may not be reliable
- Observe anomalies in PRN correlations:
  - Irregular peaks
  - Two imaginary or real peaks at the same delay

These methods can be used to detect spoofing for the above method of spoofing.

0.15.1. Observed Cases

Dual imaginary peaks observed at:

- $t = 135$ seconds
- $t = 180$ seconds
- $t = 225$ seconds

And on zooming in, at $t = 225$ we can see two different peaks shifted by a small amount. This is visble only if the real signal and spoofed signal are far apart in distance, or else the bit received would be same and hence indistinguishable.

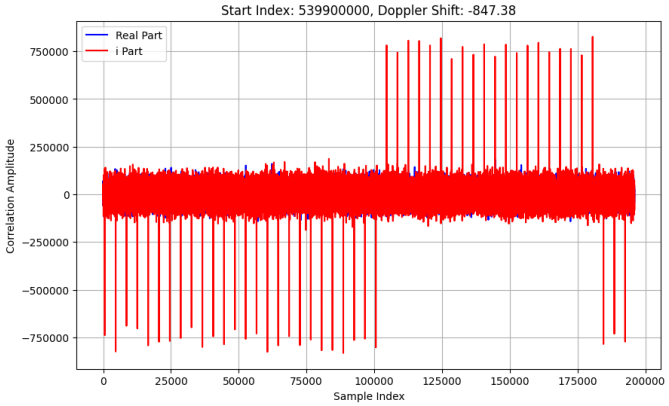

Figure 17: Correlation peaks of original data at $t = 135$ seconds.

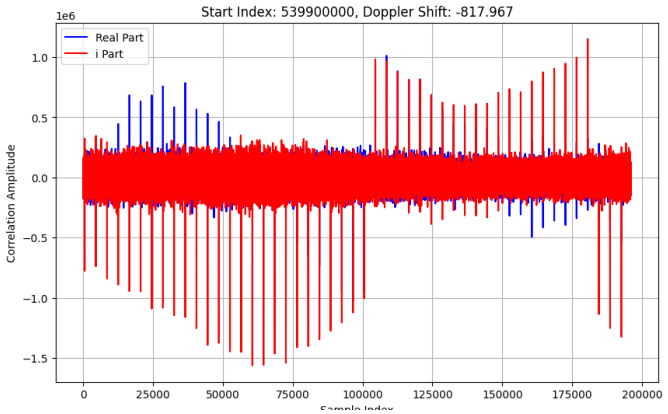

Figure 18: dual peaks at for spoofed signal $t = 135$ seconds. Not strong enough but not weak enough to be ignored.

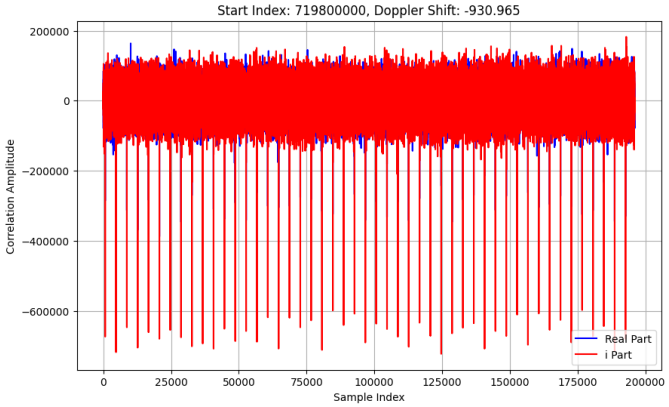

Figure 19: Correlation peaks of original data at $t = 180$ seconds.

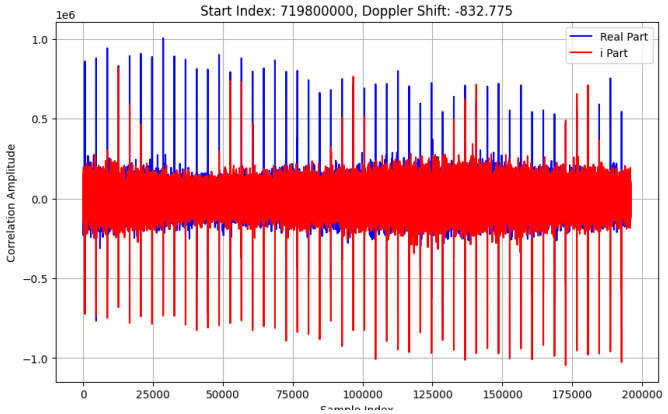

Figure 20: dual peaks at for spoofed signal $t = 180$ seconds.

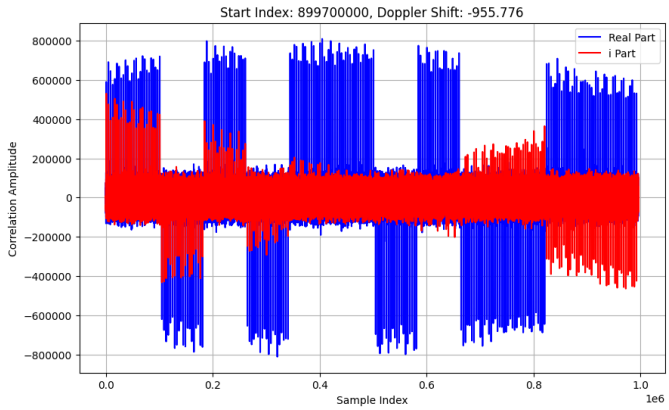

Figure 21: Correlation peaks of original data at $t = 225$ seconds.

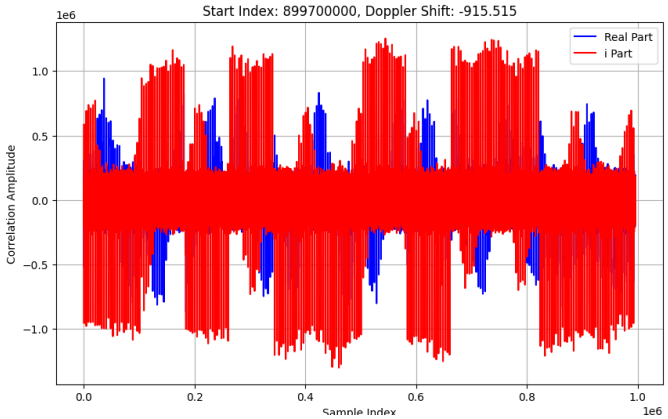

Figure 22: dual peaks at for spoofed signal $t = 225$ seconds.

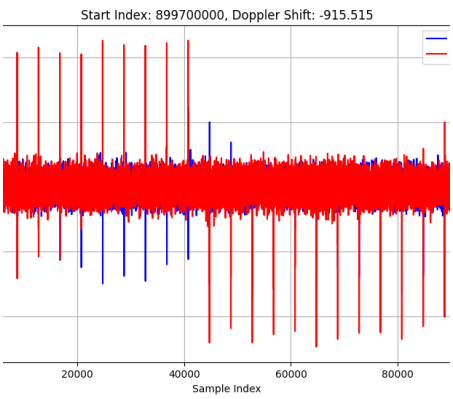

Figure 23: Zoomed-in view of correlation peaks at $t = 225$ seconds (Region 1).

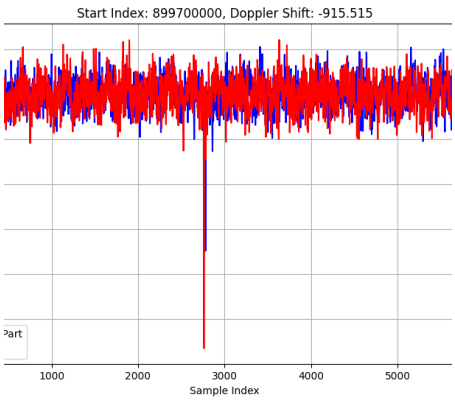

Figure 24: Zoomed-in view of correlation peaks at $t = 225$ seconds (Region 2).

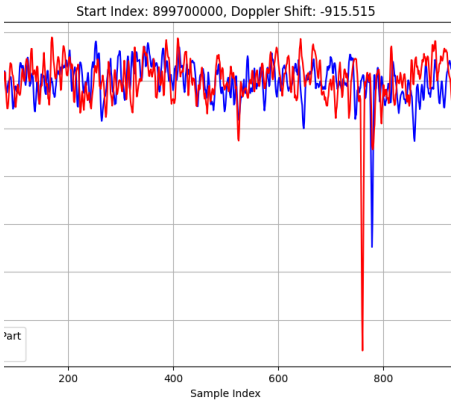

Figure 25: Zoomed-in view of correlation peaks at $t = 225$ seconds (Region 3).

These confirm presence of superimposed signals.

0.16. Miscellaneous Experiments

We plotted estimated α from a GNSS signal simulated by gps-sdr-sim, vs the actual gain values with which we simulated the signal.:

- Observed near-linear relationship
- Helps plan spoofing gain given fine Doppler info

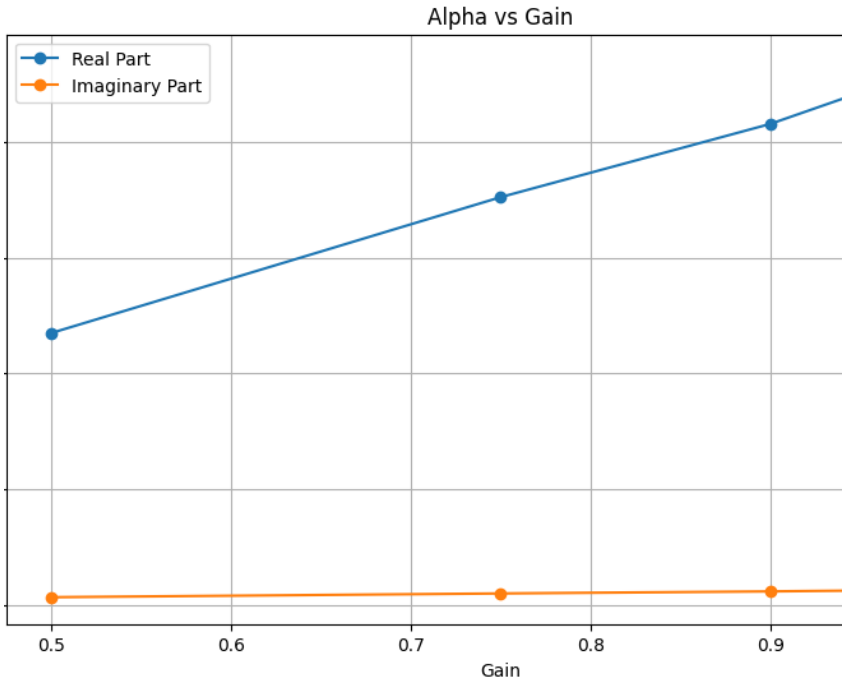

Figure 26: Estimated α vs Actual Gain Values.

0.17. Conclusions

- Satellite signals can be cancelled or overlaid for spoofing
- Fine Doppler correction is critical for alignment
- Linear combination enables spoofing with stronger secondary signals
- Anomalies in correlation allow spoof detection

0.18. Future Work

- Real-time spoofing and anti-spoofing pipelines
- Analyze long-term variations in correlation peaks
- Efficient signal subtraction over multiple frames

0.19. References

1. GNSS-SDR: An open-source GNSS software-defined receiver. Retrieved from `https://gnss-sdr.org/`
2. GPS Signal Structure: Understanding the GPS signal structure. Retrieved from `https://www.gps.gov/technical/icwg/IS-GPS-200K.pdf`
3. PRN Generator: A tool for generating PRN codes. Retrieved from `https://natronics.github.io/blag/2014/gps-prn/`
4. GPS-SDR-SIM: An open-source GPS signal simulator. Retrieved from `https://github.com/osqzss/gps-sdr-sim`

0.20. Acknowledgements

We would like to thank Prof. Sibiraj Pillai for his guidance throughout this project.