Acquisiton of GPS L1 C/A Signals: A SDR Approach

Low & High Frequency Sampling Analysis

Carlos Jorge Alves Porrais

carlos.porrais@ist.utl.pt

***Abstract* — Software-Defined Radio (SDR) is a modern technology that allows replacing part of the hardware of a radio device with a software architecture running on a generic digital processor. Such an approach allows the development of reconfigurable terminals, thanks to the ease of creation and access to every single functional block implemented in software. This characteristic happens to be very useful for system designers that are provided with a valuable tool for testing and comparing algorithms and architectures, quickly implemented as simple software modules.**

Nowadays, SDR technology is widely used in GPS receivers. However, in the future, when High Frequency and Direct RF sampling becomes available, current SDR models will become obsolete.

Our objective is to study the potential of a SDR approach to GPS receivers and solve some of the problems that arise with the use of Direct RF sampling. We will focus on the Acquisition part of the problem

We worked with both a real life and a simulated GPS signal. We used different algorithms to extract the C/A codes and related information.

We were able to successfully perform Acquisition with all the different signals and methods, and obtained similar results to state of the art methods. We were also able to make the Direct RF sampling a competitive method, according to today's standards, via Artificial Intelligence (AI) approaches.

Keywords — SDR, GPS, Acquisition, Digital Signal Processing, AI

I. Introduction

Software receivers started by using dedicated software embedded into "GNSS Application Specific Integrated Circuits (ASIC)" [1, p. 16] to process the raw GPS RF data. With the passing of time, the need for dedicated circuits gradually disappeared.

Nowadays it is very easy to study the viability of Software Defined Radio (SDR) approaches to any suitable device, from handheld dedicated GNSS receivers to smartphones, due to the widespread of "plug and play" architecture. Many of these devices give access to the raw data, which leads to the spring of applications and open source algorithms, available online, even for devices whose main purpose is not the capture of the GPS signal.

Many are the institutions around the globe that are striving towards the use of generic computing power to conceive GNSS receivers, with remarkable contributes to further the knowledge of SDR applied to GNSS receivers [1, p. 28]. Honourable mentions are Danish GPS Center [2] [3] [4], *École Nationale de L'Aviation Civile* [5]*, Institut Supérieur de l'Aéronautique et de l'Espace* [6]*, Istituto Superiore Mario Boella* [7]*,* University of Rijeka [8], Tampere University of Technology [9], Graz University of Technology [10], Polytechnic University of Catalonia [11] and Nottingham Geospatial Institute [12].

The appearance of new markets, like the "Internet of Things", "Advanced Driver Assistance Systems", autonomous driving, unmanned ships, *etc.*, is and will keep creating a huge technological demand, which will put immense pressure on researchers to produce new hardware and software improvements to current navigation methods [1], and we aim to do our part in supplying for that demand.

Therefore, the main goal of this work is to inquire about the potential of a GPS software receiver. We will solely focus on the Acquisition part of the problem, meaning that our primary objective is the development of a GPS SDR receiver flexible enough to receive any digitized GPS signal feed and derive the correspondent satellites in line of sight and respective delays, in real time.

As a secondary objective we aim to study the impact that different signal frequencies and sample rates have on the Acquisition process. Specifically, we will critique low frequency sampling vs. high frequency sampling vs. Direct RF sampling:

- **Low frequency sampling**: A RF front-end analogically downconverts the RF signal to Intermediate Frequency (IF), and only then it is sampled and digitally downconverted to baseband. We will also refer to this approach as **Partial analogue approach**;
- **High frequency sampling**: The RF signal is sampled and digitally downconverted to IF, before baseband;
- **Direct RF sampling**: The RF signal is sampled and digitally downconverted directly to baseband.

II. Theoretical Background

In a noiseless environment, the structure of the L1 signal is mathematically described by Equation 1.

Instituto Superior Técnico / Institute of Telecommunications, May 2018

$$S_{L1} = A_P P(t) D(t) \cos(2\pi f_{L1} t + \varphi) + A_{C/A} C/A(t)\, D(t)\, sin(2\pi f_{L1} t + \varphi) \quad (1)$$

Where S_{L1} is the signal at L1 frequency, $D(t)$ is the Navigation Data code, f_{L1} is the L1 frequency, φ the phase offset, t is the time, A_P is the amplitude of the P code, $P(t)$ represents the P code, $A_{C/A}$ is the amplitude of the C/A code and $C/A(t)$ represents the C/A code [13, p. 73].

The P code can easily be extracted from the signal: Digital signal processing software, such as GNU Radio, can automatically separate the In-phase from the Quadrature component. Analyzing the S_{L1} signal, if we separate the two complex components, we find that the Quadrature component is given by:

$$S_Q = A_{C/A}\, C/A(t)\, D(t)\, sin(2\pi f_{L1} t + \varphi) \quad (2)$$

Where f_{L1} is equal to 1.57542E9 Hz [2, p. 17], $D(t)$ has a bit duration of 20 ms and $C/A(t)$ has a period of 1 ms [13, p. 85].

Regarding $D(t)$, contained in each bit/symbol of the Navigation Data there are 20 C/A full periods. Since the focus of this work is the C/A codes, our time reference is 1 ms, so these "Navigation Message bit inversions" are infrequent. In light of this, we will mostly not concern ourselves with the Navigation Message.

Regarding $C/A(t)$, it can be either +1 or -1. The product of $C/A(t)$ by $sin(2\pi f_{L1} + \varphi)$ emulates the effect of changing the phase offset by π, which corresponds to BPSK modulation [2, p. 21].

There are 32 unique C/A codes, and the importance of these codes comes from their unique properties, the most important of which being the fact that each C/A code has a high autocorrelation peak when correctly aligned, and low cross-correlation peaks regardless of alignment [13, p. 83].

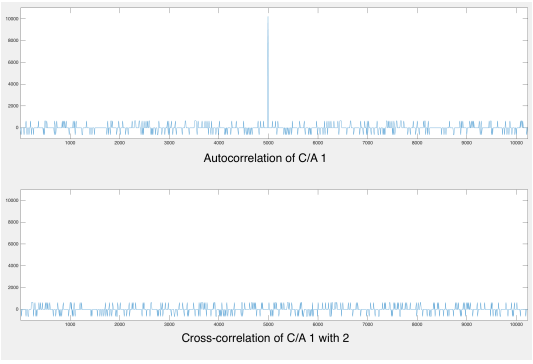


Figure II.1: Auto and cross correlation of C/A_1 with a local C/A 1 and 2. C/A codes with 10 samples per symbol. C/A_1 contains delay. Horizontal axis is the C/A code delay, in samples. Vertical axis is the value of the correlation between the two C/A codes.

Assuming a noiseless environment, to extract the C/A codes from Equation 2, first we would need to remove the sinusoidal component and only then correlate the resulting data with a locally generated C/A code with the right delay (t). If the correlation produces a well distinct peak (a correlation peak), similar to the autocorrelation peak in Figure II.1, then the code is contained in the signal, meaning we successfully extracted the C/A code.

To remove the sinusoidal component we need to know *a priori* the signal frequency and phase offset. A workaround constitutes to discretize the interval of all possible Doppler frequencies [13, p. 136] and phase offsets, and test every possible combination of the two until we find the correct pair that removes the carrier wave.

Correlating the sinusoid-less Equation 2 with a locally generated C/A boils down to:

- Discretizing the data (with any desired sample rate, so long as we are respecting the Nyquist inequality [14]);
- Generate a local C/A code with matching sample rate and matching code delay;
- Perform cross multiplication of the two.

In practice, since we do not know the delay *a priori*, we need to test the correlation for all possible delays.

Summarizing all of these steps, to extract one of the 32 C/A codes, the problem boils down to three unknowns:

- Doppler frequency f_d (simply Doppler from now on);
- Phase offset φ (simply offset from now on);
- C/A code delay t (simply delay from now on).

III. C/A Code Extraction Engine

Acquisition regards the whole process of extracting the C/A codes and the corresponding delays from the raw signal. Staying true to the SDR approach, this whole process should be automated using a coded engine. The engine has two approaches to the strip off of the carrier wave, and several approaches to the search algorithm. The "search" concerns itself with finding all the unknowns and the "strip off" concerns itself with removing the carrier wave. Regarding the "search" algorithms, they represent an Artificial Intelligence (AI) approach to the Acquisition problem, in order to improve the Acquisition speed of the whole SDR model.

First we will describe the strip off methods followed by the search algorithms, but we should keep in mind that the "strip off" needs the "search algorithm" (and vice versa), meaning there are no standalone results.

A. Downconvert to Baseband

This method is characterized by directly shifting the signal frequency from the L1 (plus Doppler) frequency to 0. Thanks to this, it is also called the "Direct Downconversion".

In the computer point of view, we receive the raw feed with a sample rate of X*1.57542 GHz, normally X being greater than 4 (4 samples per period) [13, p. 115]. From this feed is then extracted a 1 ms window, meaning best-case scenario we extract a vector of length $4 * 1.57542\,\text{GHz} * 1\,\text{ms} = 6.3017\times10\text{^}6$ samples. To successfully downconvert the signal we need locally generated sinusoids with the exact same sample rate. After multiplying the $6.3017\times10\text{^}6$ length vector by the local sinusoid with the correct signal frequency,

Doppler and offset, comes the integration step, where we can finally lower the sample rate to the MHz order.

The correlation step is done by multiplying the vectors resulting of the integration step by locally generated versions of all possible C/A codes with all possible delays (with matching sample rate).

B. Downconvert to IF

This method consists in first downconverting the signal to an intermediate frequency, and only then downconverting it to baseband.

Computing wise, again we receive the raw signal at a sample rate of X*1.57542 GHz, X being samples per period, and extract a 1 ms window. We then multiply the resulting array by a locally generated sinusoid, with a predetermined frequency [13, p. 120], downconverting the signal to an intermediate frequency (IF). We then integrate the IF signal and end up with a signal with a considerably smaller sample rate. Next we multiply the array resulting of the integration step by a another locally generated sinusoid, this time with matching signal frequency, Doppler and offset to the IF frequency signal, and filter the outcome.

Similarly to the previous case, the end result is multiplied by locally generated versions of all possible C/A codes with all possible delays.

C. Stripp off Methods End Note

As seen by the computer, every time we downconvert the original vector array, we are adding a new array of numbers on top, coming from the extra sinusoid generated (this results from the trigonometric proprieties of multiplying two sinusoids). Gradually, our desired information is immersed in a sea of sinusoidal information. With the help of filters, this effect can be attenuated, but filters themselves generate numeric waste. Regardless of approach and computer precision, with enough "number-crunching", the starting sinusoid, with its well-defined Doppler and offset, is completely lost. This phenomenon encourages the use of the minimum possible IFs, ideally 0, leading us to favoring the use of "Direct Downconversion".

An example of this numeric degradation can be seen by performing Acquisition with an increasing number of IFs between the original signal frequency and baseband. The result is shown in Figure III.1.

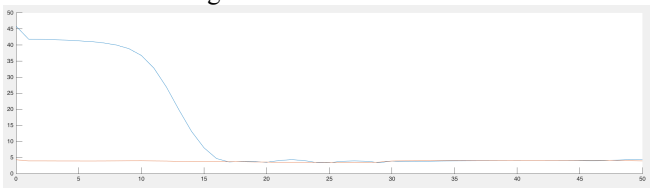


Figure III.1: Degradation of the normalized correlation peaks, due to multiple downconversions. Source signal is a simulated, single satellite, noiseless, GPS L1 signal. Vertical axis is the normalized correlation peak value. Horizontal axis is the number of IFs used. At blue, the normalized correlation peak of the transmitting satellite. At red, the average of the normalized correlation peaks of the non-transmitting satellites.

D. Bruteforce

The concept of brute force is a very intuitive one: A search algorithm that individually tests each one of the combinations of all possible discrete values for all the unknowns (C/A codes, Dopplers, offsets and delays).

E. Pruning Method: Offset

The idea behind this algorithm is to do some pre-processing to find out which offset is more likely to produce a high correlation peak. The concept is to *a priori* test all possible offsets, and disregard all but the one that is best fitting to downconvert the carrier wave. This makes it so that we spend some time before searching the tree, so we can save time during the search.

F. Pruning Method:Aliasing

The concept of this method is to force aliasing when downconverting to baseband. The result is a superimposure of the negative Doppler frequencies (and all the information contained in them) over the positive frequencies (and *vice-versa*). Thanks to this, we only have to search through half the possible Doppler frequencies, significantly reducing the time it takes to find the correct Doppler.

G. Circular Correlation

This is the state of the art model to perform Acquisition. The basic idea is to simultaneously converge to the correct Doppler frequency and C/A delay, using special proprieties of the FFTs. The mathematical and computation description of this algorithm is extensively described in [13, p. 137-144], so in this work we will not delve into it.

H. Random Mutation Hill-Climb

This algorithm chooses a possible solution (by attributing random values to the unknowns), creates small variations (what we call "Mutations") of the chosen solution, tests if any of the variations is better than the original solution, and:

- If so, jumps to the best "offspring", creates new variations (mutates), tests for the best, jumps, mutates, tests, jumps, and so on (this concept is can be visualised as climbing a mountain, in our case, the "better" the offspring, the higher in absolute the correlation, and thus the name "Hill-Climb");
- If not, tries a totally new random solution (source of the "Random" naming).

Conceptually, the "offspring" can be any variation of the "father" node: C/A code, delay, Doppler or phase.

In this work, the "offspring" regards only small variations of delay or Doppler, and the "new random solution" it can jump to refers only to a subset of possible Doppler/delay pairs.

This mentioned subset is specifically tailored so that every new "climb" climbs towards a new, not previously calculated, correlation peak of the correlation space. This results in the algorithm skipping most of the search, since it wont climb towards the same peak multiple times and it will not concern itself with finding correlation values other than maximums (either being local or absolute maximums).

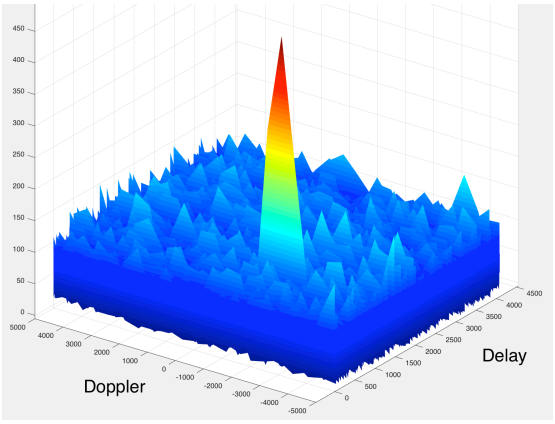


Figure III.2: Correlation space of the real signal for satellite 22. The Doppler axis is in Hz, the Delay axis in sample shifts. The vertical axis corresponds to the absolute value of the correlation.

I. Search Algorithms End-Note

All of the referred search algorithms are both **complete** and **resolution optimal** [15, p. 80], except for the Random Mutation Hill-climb algorithm which is **complete** but not **optimal** because, if by chance the noise aligns in such a way that it produces a local maximum in the middle of the climb to the absolute maximum, the algorithm will end the climb, jump to a new, not previously explored, part of the correlation space, effectively never finding the absolute maximum.

IV. Validation

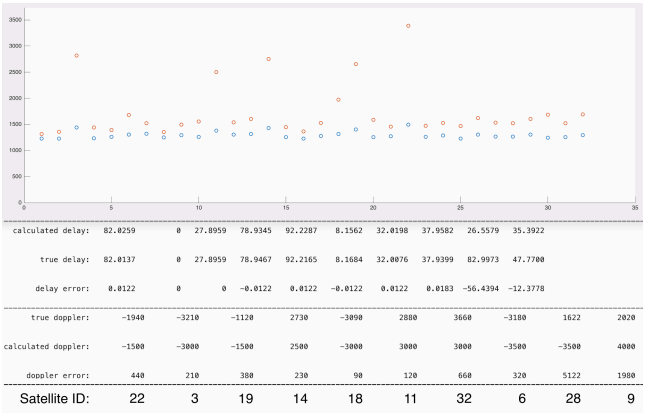
For this work, we used a real life GPS signal recording available for free to anyone online. This signal contains satellites {22,3,19,14,18,11,32,6,28,9}, from highest to lowest signal power respectively [4]. This is a recording made available by one of the researchers/developers of the "GNSS software defined receiver", at the Danish GPS Center at Aalborg University, Darius Plaušinaitis, recording which originated from the receiver used in the book "A Software-Defined GPS and Galileo Receiver: A Single-Frequency Approach" [2].

It is stated by the author that this recording is "useful for someone who wants to verify the GNSS signal Acquisition and tracking algorithms" [4], and thus it is a suitable raw feed to use as landmark for both validation and discussion.

In order to simulate the Acquisition of GPS signals at a sample rate way beyond the clock speeds of the fastest computing processors [15], we created a GPS constellation simulator that emulates our real life signal.

A. Search Algorithms Validation

With every single one of search algorithms we found the correct C/A codes, delays and Dopplers (and by "correct" we mean within our discretised variable resolution). We will just show one of the outputs, with the delay and Doppler verbose, as an example in Figure IV.1.



calculated delay:	82.0259	0	27.8959	78.9345	92.2287	8.1562	32.0198	37.9582	26.5579	35.3922
true delay:	82.0137	0	27.8959	78.9467	92.2165	8.1684	32.0076	37.9399	82.9973	47.7700
delay error:	0.0122	0	0	-0.0122	0.0122	-0.0122	0.0122	0.0183	-56.4394	-12.3778
true doppler:	-1940	-3210	-1120	2730	-3090	2880	3660	-3180	1622	2020
calculated doppler:	-1500	-3000	-1500	2500	-3000	3000	3000	-3500	-3500	4000
doppler error:	440	210	380	230	90	120	660	320	5122	1980
Satellite ID:	22	3	19	14	18	11	32	6	28	9

Figure IV.1: Output of Acquisition algorithm with the real signal, with verbose. Horizontal axis is the satellite ID. Vertical axis is the absolute of the correlation result. In red is the maximum of the absolute of the correlation result. In blue the mean of the absolute of the correlation result (only counting the delays that produce the highest correlation for each Doppler). Delays are in normalized percentages of the total possible delay. Doplers are in Hz. In the bottom half, the rows of each line correspond to satellites ordered from highest to lowest power ({22,3,19,14,18,11,32,6,28,9}).

The "calculated delay" is the delay our algorithms output. The true delay is the delay referenced by the author (after normalization). The error is the *delta* between the two. And just for the purpose of this example, we will explicitly show the calculated Doppler (using the same "calculated" and "true" reasoning).

Regarding the respective delays, the author results refer to a signal with 16 samples per symbol. As the user can define the samples per symbol as they see fit (through up-sampling or down-sampling), we normalized both ours and the author's delays to a percentage of the total samples in 1 ms. Another note is that the author gives no time reference for these "C/A Code Delay"s, so instead of blindly finding the point in the raw feed which translates into these results, we normalized all delays by the delay of satellite 3. So we end up with delays which are in percentage.

As we can see, we find similar results with our algorithms to the ones mentioned by the author: We were able to pick up 6 of the 10 satellites present in the recording, with correct delays and Dopplers. This validates the capability of performing Acquisition by the search algorithms

B. Simulator Validation

To successfully validate the simulator we recreated the real signal scenario [4] through simulation, applied the same algorithms used for the real signal, and achieved approximately the same results.

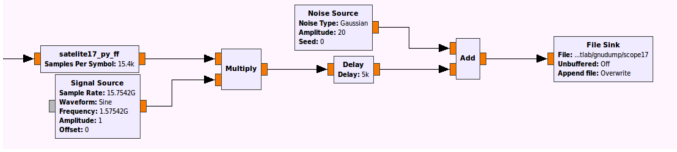


Figure IV.2: Block chain of satellite 17, developed in GNU-Radio Companion. Blocks documentation can be found in [17].

Each one of the 32 satellites is generated in GNU-Radio by a block chain similar to Figure IV.2.

The Custom block ("sateliteX_py_ff") is a custom-made python module done to simulate the C/A code of each satellite. An example of output can be seen in Figure IV.3.

The Signal Source block [18] outputs a wave of the likes of Figure IV.4.

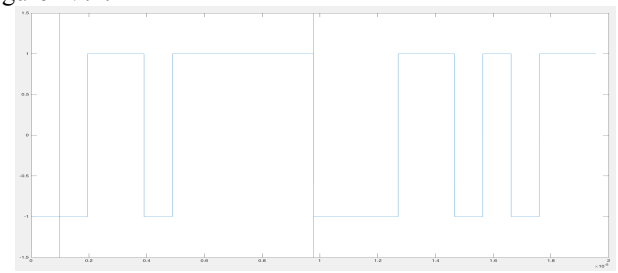


Figure IV.3: C/A 1 code. Horizontal axis is time in seconds (10^{-5} $factor$), vertical is the normalized C/A amplitude. Red vertical lines correspond to the 1 symbol mark and 10 symbols mark. Sample rate is 15.7542GHz.

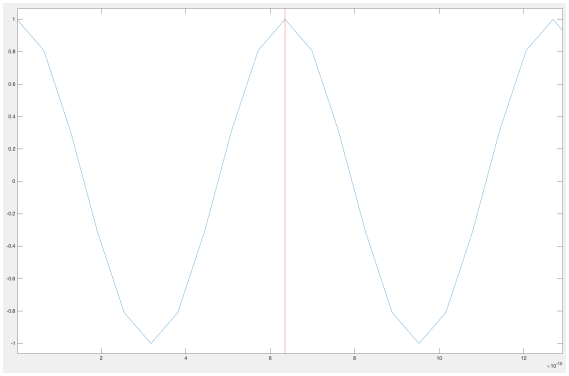


Figure IV.4: L1 frequency sine wave. Horizontal axis is time in seconds (10^{-10} $factor$), vertical axis is normalized amplitude. Red vertical line corresponds to the period mark. Sample rate is 15.7542GHz.

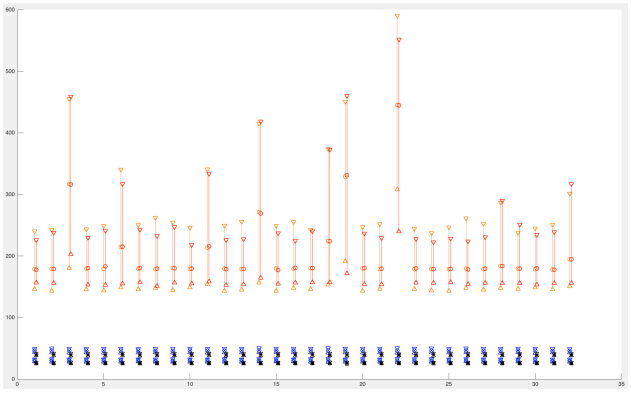


Figure IV.5: Intervals of thousands of Acquisitions (>10000) with different 1 ms windows of the real signal (red/black intervals) and simulated signal (orange/blue intervals). Horizontal axis is the satellite number. Vertical axis is the absolute of the correlation result. The pairs above the 100 mark are the maximum of the absolute of the correlation result intervals. The pairs below the 100 mark are the means of the absolute of the correlation result intervals. The arrows indicate the interval limits, the circles the averages.

The end result (for each satellite) is a signal made up of the product of a sine (Figure IV.4) by a C/A code (Figure IV.3), with added noise [19] *a posteriori*. Since making a robust noise model goes beyond the objectives of this work, we simply use white Gaussian noise. All satellite feeds are then added together.

To recreate the real life scenario we had to increase/decrease the simulated satellites signal power and noise power, to achieve a similar satellite power hierarchy to the one described in the literature [4]. We also performed as many Acquisitions as possible with both the real and synthetic signal, and strived towards a perfect match of interval averages, maximum and minimums. The end-result can be seen in Figure IV.5.

As we can see, we can not perfectly match our simulation to the real results. Despite not being a perfect match, one could argue that these are extremely good results. To put things in context, the percent errors of the maximum, average, and minimum of the simulated correlation peaks, relative to the real ones, are displayed in Table IV.1.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
minimum	6.9	8.1	11.1	5.2	6.1	4	7	2.6	7.7	4.2	2.9	6.2	5.8	4.9	7.4	5.4	7.1	2.2	11.4	7.2	4.7	28.5	6.9	7.7	8.9	4.3	7	5	6.1	2.9	6.3	3.6
average	0.6	0	0.1	0.3	2.3	0.3	0.6	0.4	0.4	0.2	1.1	0.5	0.2	0.8	1.8	0.7	0.2	0.2	0.7	0.3	0.2	0.1	0.8	0.1	0.3	0.7	0.5	0.1	0	0.8	0.1	0.1
maximum	6.4	2	0.8	6	3.2	7.4	2.9	12.8	3	13.2	2.2	10.3	12.5	0.8	4.9	13.7	0.9	0.3	2.1	4.6	10.1	7	7.5	6.8	8	16.7	9.4	1.1	5	4.4	5	4.8

Table IV.1: Table of the absolute value of the relative error (in percentage) of the minimum, average and maximum of the real signal *vs* simulated signal (Figure IV.5) correlation peaks. First row corresponds to the satellite C/A code number.

The upper bound percent error of the average of all satellites is less than 2.35%. This indicates that:

- By manipulating the satellite powers we can replicate a desired satellite power pattern/hierarchy, proven by the low average error of the present satellites (1.15%);
- By manipulating the noise power injected into our simulator, we can accurately recreate the average noise power present in the real signal, proven by the low average error of the non-present satellites (2.35%).

Regarding the present/simulated satellites, the interval limits errors, when compared to the real life scenario are shown in Table IV.2.

	3	6	9	11	14	18	19	22	28	32	
minimum	-0.78	7.43	3.01	2.22	-0.85	0.27	-2.1	6.98	-1.14	-4.84	
maximum	-11.11	-3.97	-7.7	-2.93	-4.88	-2.2	11.44	28.52	-4.98	-3.62	
	5.95	1.73	2.35	0.35	2.87	0.97	4.67	17.75	3.06	4.23	4.39

Table IV.2: Section of Table IV.1, with only the minimum and maximum error percentage of the present satellites. Third row corresponds to the absolute maximum interval drift error. The last column is the average of said drift error. In green is the best simulated interval, in red the worst.

As we can see, contrary to the low average error, we have a substantial interval shift, sometimes upwards, sometimes downwards, with an upper bound to the absolute shift error of 17.755%. For context, we need to "pull" the 22nd interval (in Figure IV.5) 17.75% downwards to actually match the real interval. This can not be done without more input variables, which could come from a robust noise model.

Despite this fact undermining the simulator credibility, we urge the reader to attend to the fact that we were able to simulate upwards of 99.98% of all possible real results: we had only 51 results outside the simulated intervals, in more than 32*10000 correlations, meaning a mismatch/match ratio inferior to 51/320000.

We need to also point out that we successfully recreated the C/A code delays and Dopplers of the real satellites [4], as seen in Figure IV.6.

```
calculated delay:   82.0137        0   27.9570   78.9834   92.2776    8.2111   32.0626   37.9277   82.9912   47.8006

     true delay:   82.0137        0   27.8959   78.9467   92.2165    8.1684   32.0076   37.9399   82.9973   47.7700

   symbol error:         0        0    0.0611    0.0367    0.0611    0.0428    0.0550    0.0122    0.0061    0.0305
    true doppler:    -1940    -3210    -1120     2730    -3090     2880     3660    -3180     1622     2020

calculated doppler:    -2000    -3000    -1500     2500    -3000     3000     3500    -3500     1500     2500

   doppler error:    60   210   380   230    90   120   160   320   122   480
```

Figure IV.6: MATLAB verbose of Doppler and delay of Acquisition performed in 1 ms window of the simulated signal. Delays are normalized similarly to Figure IV.1. Dopplers are in Hz. The "true" lines correspond to the literature data about the real signal. The "calculated" is the result of our Acquisition algorithm. The "error" is the absolute value of delta between them. The rows of each line correspond to satellites ordered from highest to lowest power ({22,3,19,14,18,11,32,6,28,9}).

We were able to, using the exact same search algorithms for both the real and simulated signal, find extremely similar results of correlation peaks, delays and Dopplers, thus validating our simulator.

Summarizing, with an interval match accuracy upwards of 99.98%, a partial overlap of the correlation averages (black and blue pairs in Figure IV.5), and a Doppler and delay match within the resolution limits, we can now safely say that our simulator is able to accurately simulate a real case scenario.

V. Results Discussion

We implemented two downconversion methods to bring a signal to baseband and five search algorithms to extract the C/A codes.

Regarding the downconversion methods, we applied both approaches to the real data and the simulation data, ending up with four different routes. A way to intuitively understand what each of the approaches represents can be seen in Figure V.1.

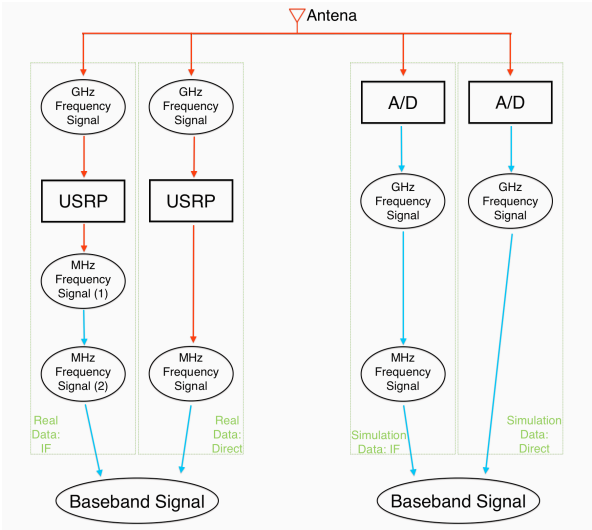


Figure V.1: Visual representation of each of the downconversion methods applied for both the real and the simulated data. "MHz Frequency Signal (1) and (2)" correspond to different IFs, both in the MHz order. Red lines correspond to analogue signal, blue lines to digital signal.

Since the algorithms are "blind" towards the input, they receive a signal and either transfer it to an IF, or to baseband, resulting in two sets of scenarios:

- The real data already comes digitized at an IF. The algorithms either downconverts it to a second IF (before baseband), or directly to baseband;
- The simulation data comes as a digitized RF signal. Appling the same logic, it is either downconverted to an IF before baseband, or directly to baseband.

Despite us using two different signals from different sources, and the four resulting approaches differing quite a lot from each other, thanks to the fact that we are emulating our real world scenario in our simulator, both "Baseband Signals" end up resembling each other.

Regarding the search algorithms, three of them were implemented both with and without IF downconversion. The reason for this is the three first search algorithms were made for the purpose of being benchmarks for the results discussion. Also, for the last two it does not make much sense to categorize them as "IF" or "Direct", since they have their own specific way of removing the carrier waves.

In a structured manner, the Acquisition times of every approach can be seen in Table V.1.

	Real Data		Simulation Data	
	IF	Direct	IF	Direct
Bruteforce	43.84	57.6	43.45	627.84
Circular Correlation	0.17	0.53	0.85	635.52
Offset Pruning	2.4	3.2	3.21	20.63
Aliasing Pruning	22.76		23.46	
Random Mutation Hill-Climb	9.632 + 0.46 = 10.092		10.72 + 18.7 = 29.42	

Table V.1: Acquisition time, in seconds, of each of the algorithms implemented. "IF" refers to downconverting to an intermediate frequency (between the frequency at which we receive the data, and baseband) before downconverting to baseband. "Direct" refers to downconverting from the frequency at which we receive the data directly to baseband.

A. *Partial Analogue Approach*

Starting with the "Real Data" numbers, they are a good illustration of the present day approaches and they answer the questions of, if we are receiving an analogically downconverted to intermediate frequency signal, what would be the time it would take to perform Acquisition using a SDR approach, if:

- (Direct) We immediately downconvert the signal to baseband?
- (IF) We downconvert the signal to a second intermediate frequency, lower the sample rate, and only then to baseband?

Taking into account that nowadays even basic processors have clock speeds around the GHz, a present day GPS SDR receiver usually goes directly from the analogue intermediate frequency to a digital baseband signal. This is the approach mainly taken in the literature, where a USRP (or similar device) takes care of analogically downconverting the signal to an intermediate frequency, and then passes it through an

A/D converter. This is also the signal genesis of our real life downloaded signal [4]. In this approach, the user only has access to the digitized raw (intermediate frequency) feed.

Further analyzing both Real Data columns uncovers the origin of the present day norm: We can see that there is no meaningful time advantage in downconverting to a second intermediate frequency (Real Data: IF *vs* Direct column), while having the extra computational hurdle of performing an additional downconversion. This further supports the approach suggested in the literature [2] [13].

All of this leads to the conclusion that the "Real Data: Direct" column really is a good estimation of Acquisition times with a GPS SDR receiver with present day technology, as stated previously.

As a side note, from the 5 algorithms, we can see that the Circular Correlation is the fastest, which corroborates the claims made in the literature [13, p. 159].

Having developed an algorithm that can simply be plugged into a digitized intermediate frequency GPS feed (both real world data or simulated) and performs Acquisition in less than 0.6 seconds, we can now safely say that an SDR approach is a viable alternative to using a fully analogue approach.

B. High Frequency Sampling

First regarding the "Simulation Data: IF" approach: Here we are downconverting the original signal (GHz) to an intermediate frequency (MHz), and only then downconverting it from there to baseband. Now however, this whole process is done digitally. This is the "high frequency sampling" analogous approach to present day generic GPS SDR receivers (which results can be seen in the "Real Data: Direct" column).

Rephrasing, where nowadays a first downconversion is done analogically (to an intermediate frequency) and then a second downconversion is done digitally (to baseband), with a high frequency sampled signal everything is done digitally.

The benefit of this "Simulation Data: IF" approach (when compared to the "Real Data: Direct") can be easily seen in Figure V.1: At the cost of having to introduce a high frequency A/D, we wouldn't need any of the extra electronic components contained in the USRP (the likes of filters, amplifiers and local oscillators [13, p. 120]) to downconvert the signal to intermediate frequencies. This may cheapen the production cost in the future, but even if that is not the case, it will certainly cheapen the maintenance, since newly found AI and mathematical approaches can simply be programmed earlier in the process of treating the GPS feed.

With the relationship between these two established ("Real Data: Direct" and "Simulation Data: IF"), we could simply gravitate towards the fastest algorithm: Circular Correlation. As alluded to in the literature [2] [13], nothing can compete with it in Acquisition speed. And as shown in the table, we achieve approximately the same Acquisition speeds using the digital approach instead of the analogue/digital one (Table V.1: 0.85 *vs* 0.53 seconds).

This simply proves that in the future, when we eventually have the technology to do high frequency sampling, we will only need a single ADC to have an equivalent receiver to present day state of the art GPS SRD Receivers.

C. Direct RF Sampling

Nowadays we are forced to analogically downconverting the signal to an intermediate frequency and only then digitally sampling it, but in the future that might not be the case. With the appearance of high frequency sampling, we are not forced to use IFs. So why blindly copy what the present day norm is?

If we could find an approach with competitive Acquisition speeds, without the extra numeric degradation coming from each digital downconversion, that would be ideal. Striving towards this ideal drove us to create all these alternative AI algorithms and the simulator to test them in a high sample frequency environment.

Inspecting Table V.1: "Simulation Data: Direct" column, we can immediately see that, if we want a robust high speed without deterioration Acquisition method, Circular Correlation is not the answer: Performing FFTs becomes slower and slower the bigger the vector we are working with, and for Direct RF sampling we are dealing with arrays with a minimum length of several millions. This is reflected in the "Simulation Data: Direct" Circular Correlation bracket, where we can see worse results than even Bruteforce.

We must therefore find alternatives, and Bruteforce surely will not be one: As one would expect, an algorithm that simply goes through every possible state, despite being good for validation, has no practical application, due to its prohibitive Acquisition times.

We are left with both Pruning methods and the Hill-Climb algorithms.

Starting with the Aliasing Pruning, the algorithm simply is not robust enough to be a contender: Despite the promising time shown in the table, folding the frequency spectrum makes it so that we the are adding both the information and the noise contained in both sections, on top of each other. With this added noise we can barely get 4 satellites most of the time.

The Offset Pruning does not suffer from additional noise, while significantly increasing the Acquisition speed (around 30 times faster than Circular Correlation). Nonetheless, 20 seconds is still too much time in this day and age.

The Hill-Climb, with a time of 29 s, seems even worse than the Offset Pruning, but upon closer inspection, we can see the algorithm is composed of two parts. But what exactly are these "10 and 18 seconds"?

The Random Mutation Hill-Climb algorithm has two major players working at the same time: We called them the "Creating the search starting points subspace" and the "Search" that works over it. Since they are separate entities, nothing stops the algorithm from being implemented in a parallelized fashion: It can be searching for the satellites while at the same time preparing the next subspace. In light of the parallelization, what once was a 29 seconds time, is now reduced to the greater of its two components: 18 seconds.

There is however many ways to code and attribute tasks to the "subspace creation" and the "search":

- In one extreme, the subspace creation could be responsible for only creating the "subset of points" where the search algorithm should start searches. This approach puts the heavy burden on the search part;
- On the opposite end, the subspace creation could have the task to:
  - Create the "subset of points";
  - Downconvert the source signal, with all possible Dopplers and offsets combinations, to baseband;
  - Pre-prepare all possible local C/A codes with all possible delays.

  This approach puts the heavy burden on the subspace creation, allowing the search to focus solely on "searching" (climbing).

These are the 2 extremes, but anything in-between can also be done. This goes to show that the Random Mutation Hill-Climb Algorithm is extremely flexible, and by working on both parts simultaneously and fine-tuning the task distribution between them, we can reach lower Acquisition times.

In our implementation of the Random Mutation Hill-Climb algorithm, we choose to task the "subspace creation" part with "Creating the map of search starting points" and "Downconverting the source signal", and the "search" part with creating "local C/A codes with all needed delays" and "climbing".

As we can see in Table V.1, the Real and the Simulated Data have a thing in common: Both have the 10 second part, plus a variable time (0.46 or 18 seconds). In our case, these "10 seconds" corresponds to the "search", and the variable part the "subspace creation and the downconversion".

It may not seem intuitive, but the explanation is simple: No matter the samples per symbol the raw signal feed has, the "subspace creation" outputs a "grid" of downconverted baseband signals, always with the same samples per symbol. The "search" always receives a grid with the same "dimensions" (the likes of Figure III.2 axes grid), so no matter the input, on average it takes the same time to go through it (proven by the similar 10 seconds time in the Real *vs*. Simulated columns in Table V.1, despite the substantial initial samples per symbol difference: 16 to 15400).

In the spirit of task redistribution to achieve greater speeds, the "subspace creation" could downconvert only promising Dopplers, and only if the "search" algorithm can not find a meaningful correlation peak, then the "search" has the task to decode the rest. This would lower the "18 seconds" time and increase the "10 seconds" one. Promising Dopplers would have to be information derived either from previous Acquisitions, or from likelihood of detecting satellites in specific Dopplers, etc. Many other forms of task redistribution can be argued for, meaning there is much room for optimization.

Ideally we would want to mess around with the flexibility of task redistribution and parallel processing to fine-tune the "Simulation Data: Random Mutation Hill-Climb" time to around 14 seconds each part (the middle point between 18 and 10).

Another important thing to state about this algorithm is that the results shown have no performance enhancements. There is nothing stopping us from fusing any of the other methods with this algorithm, making such a hybrid algorithm extremely faster. As an example:

- Merging in **Offset Pruning** concepts would lower the 18 seconds time to a little over one fourth of it, around 5 seconds;
- Incorporating the FFT properties exploited by **Circular Correlation** [13, p. 143] would greatly lower the 10 second part of the algorithm, to around 1 second, since we would go from a two dimensional climb (the algorithm climbs in a Doppler/Delay "grid" in Figure III.2) to a one dimensional climb (just Dopplers).

Again in this case it would be wise to rearrange the workload to favor equal work time, ideally leading to a 3 second computation time (middle ground between 1 and 5).

Concluding, taking parallel processing and task re-distribution into account, without performance enhancements, we would be talking about 14 seconds; with those enhancements, the best-case scenario is a 3 seconds Acquisition time. Let's take a moderate posture and assume a value of 10 seconds or less.

Less than 10 seconds is a respectable Acquisition time, with the added bonus of no extra deterioration due to IF downconversions, but still nowhere near the "less than 1 second" Acquisition times achieved nowadays (Table V.1: Real Data: Circular Correlation). Still the point made here is that where the Circular Correlation is a perfected state of the art Acquisition model, without much more room to grow, AI models have enormous growth potential.

One negative aspect must be remembered: The Random Mutation Hill-Climb algorithm is not optimal. Just to give context to the user, in all of more than 100000 Acquisitions (including test runs) with the real signal, we consistently extracted 4 or more of the highest S/N satellites. But despite seeming reliable, it may seem so just by pure luck.

As explained before, the algorithm may not find the correlation peaks due to local maximums. So either the algorithm is enhanced with other algorithms to achieve optimality, or one must be prepared to eventually fail Acquisitions as a compromise to achieve the best Acquisition speed possible.

Concluding the whole "high frequency sampling" *vs*. "Direct RF sampling" approach (Table V.1: "Simulation Data" columns), since we were able to conceive less than 10 seconds Acquisition times, with current day technology, we can safely say that in the future, when Direct RF sampling becomes viable, direct downconversion will eventually become the standard Acquisition approach.

Methods like Circular Correlation are optimized to such an extreme that they will mostly not benefit from extra computational power: For them, faster CPU clock speeds simply means going through code loops faster. So the less than 1 second Acquisition times seen in Table V.1 will not get much faster in the future, leading us to conclude that the whole "digitally downconvert to an Intermediate Frequency" method hasn't much room to grow (to put in perspective, doubling the CPU speed would at best halve the time, meaning we would go from 0.8 to 0.4, a 0.4 seconds time decrease).

On the contrary, the same percentage clock speed increase would greatly benefit direct downconversion and AI based algorithms. Not only will we be able to consume a higher and higher amount of samples without any performance backlash, but also the time savings in absolute value would be much higher. Different AI algorithms benefit differently from more computational power, and the challenge of future work will be in finding the one that can harness it the best.

VI. Conclusions

As the literature-proposed algorithms to perform Acquisition crumble when faced with GHz sample rates, our work mutated into more and more of an AI approach to GPS signal Acquisition. Different AI strategies where applied to be able to perform Acquisition in reasonable time frames. By the end of the journey we were deep inside AI territory, trying to create an AI algorithm that could process more than 10^6 samples fast enough for the results to be meaningful.

A. Partial Analogue Approach

We were able make a strong point that a SDR Acquisition model, when coupled with a RF front-end, is a viable way to perform GPS Acquisition: We successfully developed an algorithm that, feeding off a real life digitized intermediate frequency GPS feed, performs Acquisition in less than 0.6 seconds (Table V.1: Real Data: Direct: Circular Correlation). This result should be treated with caution when compared with results shown in the literature [2, p. 85], due to the different computational power used.

The importance of this result isn't how it fares against SDR results from other authors, but how it fares against nowadays receivers: If we assume that the Acquisition regards half of the Hot Start TTFF [20, p. 3-9], even for the most demanding case (TTFF <1 sec [1, p. 34]), we were able to achieve competitive results using an SDR approach.

In light of this, in the immediate future it will be harder and harder to justify using dedicated electronic components to perform a task that can be easily done by the likes of a Raspberry Pi.

B. High Frequency Sampling

As higher and higher clocking speeds are achieved, we predict that downconversion from a raw digitized feed will be the norm.

We started by creating and validating a software simulator capable of recreating a real life GPS constellation pattern. Without the sampling frequency limitations that real life receivers have, we were able to explore the consequences of high frequency sampling.

We were pleased to see that we can achieve approximately the same Acquisition times (less than 1 ms) using high frequency sampling instead of an analogue RF front-end. In both instances we extract the C/A codes after downconverting to an intermediate frequency, and the impact of doing it all in a digital form is miniscule.

When ADC become powerful enough, mass-market GNSS applications like smartphones, the priority of which is producing small devices, fast TTFF and low cost [1], will eventually transition to fully digital.

C. Direct RF Sampling

Without the hardware limitations, we were also able to study the very limits of the Acquisition process: Directly downconverting from GHz to baseband.

Immediately, the algorithms that nowadays perform Acquisition became obsolete, with Acquisition times of more than 600 seconds. By delving deep in AI, we were able to solve this problem, and by the end of our journey we were able perform 20 seconds Acquisitions, or even less if we consider fusing the proposed algorithm with other discussed performance enhancements.

Continuing again with the assumption that the Acquisition regards half the Hot Start TTFF, if we take the previously mentioned 10 sec time as a landmark, and compare it with the Hot Start TTFF of activities that require the highest precision possible (TTFF <15 s [1, p. 57]), we can safely say that a Direct RF method is competitive time wise, and simply the superior one accuracy wise (thanks to avoiding the numeric degradation associated with each downconversion).

In the distant future, with the help of AI, Direct RF sampling will simply be a better alternative to any other method mentioned in this work, in every aspect. New algorithms will be easier to test and implement, translating into cheaper cost of implementation, maintenance and update.

D. End Note

Our major contribute was to inquire about the consequences of high frequency sampling and solving one of the problems that Direct RF sampling will bring to the table.

Hinted through all of the "European GNSS Agency GNSS Technology Report, Issue 1" is: "Now is the time to invest research time in this field" [1]. With this work we matched the demand of two markets: High frequency sampling with a single IF fulfills the needs of "mass-market" applications (the likes of Smart-phones and Internet of Things), while Direct RF sampling fulfills the "industry" needs (the likes of Agriculture, Surveying and Construction).

VII. Works Cited

[1] © EUROPEAN GNSS AGENCY. GNSS User Technology Report, Issue 1. © European GNSS Agency. [S.l.]. 2016. (978-92-9206-029-9).

[2] BORRE, K. et al. A Software-Defined GPS and Galileo Receiver: A Single-Frequency Approach. 2nd. ed. [S.l.]: Birkhäuser, 2007. ISBN 978-0-8176-4390-4.

[3] ALBORG UNIVERSITY. Danish GPS Center. Available at: <http://gps.aau.dk/dgc/ dgc_activities/>. Accessed: March 2018.

[4] PLAUŠINAITIS, D. GNSS And Other Topics: Samples of GNSS Signal Records. GNSS And Other Topics. Available at: <http://gfix.dk/matlab-gnss-sdr-book/gnss-signal-records/>. Accessed: 2018.

[5] ÉCOLE NATIONALE DE L'AVIATION CIVILE. École Nationale de l'Aviation Civile: Research. Available at: <http://www.enac.fr/en/global-navigation-satellite-system>. Accessed: March 2018.

[6] DEPARTMENT OF ELECTRONICS, OPTRONICS AND SIGNAL PROCESSING - DEOS. ISAE SUPAERO. Available at: <https://www.isae-supaero.fr/en/research/departments/ department-of-electronics-optronics-and-signal-processing-deos/department-of-electronics-optronics-and-signal-processing-deos/>. Accessed: March 2018.

[7] ISTITUTO SUPERIORE MARIO BOELLA (ISMB). Research Areas: Navigation Technologies. Istituto Superiore Mario Boella (ISMB). Available at: <http://www.ismb.it/ en/Navigation_Technologies>. Accessed: March 2018.

[8] RENATO FILJAR, M. F. A. L. Software-defined GNSS receiver as a framework for GNSS-related research and education. Faculty of Maritime Studies, University of Rijeka, Croatia. [S.l.]. 2016.

[9] KINNARI, S. Tampere University of Technology. Research Fields: Signal processing for wireless positioning. Available at: <http://www.tut.fi/en/research/research-fields/ electronics-and-communications-engineering/signal-processing-for-wireless-positioning/ index.htm>. Accessed: March 2018.

[10] GRAZ UNIVERSITY OF TECHNOLOGY. Graz University of Technology. Research: Navigation. Available at: <https://www.tugraz.at/institute/ifg/research/navigation/>. Accessed: 27 March 2018.

[11] GAGE RESEARCH GROUP. Research Group of Astronomy and Geomatics. Technical University of Catalonia (UPC). Available at: <http://gage.upc.edu/>. Accessed: March 2018.

[12] UNIVERSITY OF NOTTINGHAM. Nottingham Geospatial Institute: Research. University of Nottingham. Available at: <https://www.nottingham.ac.uk/ngi/research/propagation-effects-on-gnss/index.aspx>. Accessed: March 2018.

[13] TSUI, J. B.-Y. Fundamentlas Of Global Positioning System Recievers: A Software Approach. 1st. ed. [S.l.]: John Wiley & Sons, Inc., 2000. ISBN 0-471-38154-3.

[14] WESCOTT, T. Sampling: What Nyquist Didn't Say, and What to Do About It. Wescott Design Services. [S.l.], p. 13-16. 2016.

[15] RUSSELL, S. J.; NORVIG, P. Artificial Intelligence: A Modern Approach. 3rd. ed. New Jersey: PearsonEducation,Inc., 2010. 80 p. ISBN 978-0-13-604259-4.

[16] CHIAPPETTA, M. AMD Breaks 8GHz Overclock with Upcoming FX Processor, Sets World Record. Hot Hardware, 13 September 2011. Available at: <https://hothardware.com/news/amd-breaks-frequency-record-with-upcoming-fx-processor>. Accessed: March 2018.

[17] GNU RADIO. GNU Radio Companion: Block Documentation, 2013. Available at: <http://www.ece.uvic.ca/~elec350/grc_doc/>. Accessed: March 2018.

[18] GNU RADIO. GNU Radio Companion Waveform Generators: Signal Source. GNU Radio Companion: Block Documentation, 2013. Available at: <http://www.ece.uvic.ca/~elec350/ grc_doc/ar01s02s01.html>. Accessed: March 2018.

[19] GNU RADIO. GNU Radio Companion Waveform Generators: Noise Source. GNU Radio Companion: Block Documentation, 2013. Available at: <http://www.ece.uvic.ca/~elec350/ grc_doc/ar01s02s03.html>. Accessed: March 2018.

[20] U.S. GOVERNMENT. NAVSTAR GPS USER EQUIPMENT INTRODUCTION. [S.l.]. 1996.