# Modern Digital Communications: A Hands-On Approach

## GPS: Extracting the Bits

**Dr. Nicolae Chiurtu**

**- Course material of Prof. Bixio Rimoldi -**

*Last revision: Oct. 19, 2021*

# Goal

The goal is to decode the bit sequence sent by a GPS satellite.

In this lecture we go over the structure of the satellite signal, derive the channel model, find the structure of the received signal, and finally learn how to estimate the various signal parameters.

Once those parameters are estimated, the signal will look to the receiver as assumed in PDC (Principles of Digital Communications), namely a pulse train modulated by (binary) symbols plus white Gaussian noise.

# THE SATELLITE SIGNAL

# The Data Structure

The data sent by a satellite is organized in pages, pages are made of subframes, subframes of words, words of bits, bits of C/A codes, and C/A codes of chips. Specifically:

- a C/A code consists of 1023 chips. It takes 1 ms to send a C/A code.
- a bit contains 20 repetitions of the C/A code (20 ms)
- 30 bits make one word (600 ms)
- 10 words make a subframe (6 s)
- 5 subframes make one page (30 s)
- 25 pages make a complete data set, also called superframe (12.5 min)

To decode the bits we need to understand the details about the top two bullets. First a pictorial overview.

# Pictorially

C/A code made of $1023$ chips: $1$ms total

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | $\cdots$ | 1018 | 1019 | 1020 | 1021 | 1022 | 1023 |

Bit made of $20$ C/A codes: $20$ms total

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | $\cdots$ | 15 | 16 | 17 | 18 | 19 | 20 |

Word made of $30$ bits: $600$ms total

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | $\cdots$ | 25 | 26 | 27 | 28 | 29 | 30 |

Subframe made of $10$ words: $6$s total

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

Page made of $5$ subframes: 30s total

| | | | | |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |

# How Much Data do we Need

The information contained in the first 3 subframes of each page suffices to obtain what we need for positioning. (Other frames contain information such as the satellite "health," ionospheric conditions, etc.)

Hence, at the very least, it takes 18 sec worth of data for each satellite. If you collect 30 sec (1 page) worth of data you are sure to have the first three subframes of each satellite.

You will be given more than 30 sec worth of decoded bits for each satellite.

Your will reconstruct at least one page of four satellites.

# Clocks

There are 24 satellites (plus a few extra).

Each satellite has a clock and the receiver has a clock.

The time shown by the clock of satellite $s \in \{1, \ldots, 24\}$ is denoted by $t^s$. The time shown by the receiver is denoted by $t_r$.

There is a GPS-wide reference clock, with time denoted by $t$. The various clocks are related the following way:

$$t^s = t + \delta t^s;$$
$$t_r = t + \delta t_r.$$

In particular,

$$t^s - t_r = \delta t^s - \delta t_r.$$

# The Transmitted Signal

The transmitted signal consists of the superposition of a civilian and a military part.

We can't decode the military part since we don't know some of its key parameters. We will simply treat it as additive noise. As usual, the receiver front end will project the received signal onto the signal space of interest (the one spanned by the civilian signal). Since the signals are of spread-spectrum type, the projection of the neglected component will simply show up as additive white Gaussian noise.

The (civilian) signal at the satellite antenna as a function of the satellite clock $t^s$ has the form

$$s(t^s) = \Re\left\{ e^{j2\pi f_c t^s} \sum_i b_i p_b(t^s - iT_b) \right\}$$

where $f_c$ is the carrier frequency and $p_b(t^s - iT_b)$ is the pulse of duration $T_b$ that carries the $i$-th bit $b_i \in \{\pm 1\}$. The pulse $p_b$ consists of a sequence of spread-spectrum chips (more on this later).

The baseband-equivalent signal is

$$s_E(t^s) = \sum_i b_i p_b(t^s - iT_b).$$

The bit-carrying pulse $p_b(t)$ consists of 20 repetitions of a pulse $p_a(t)$ which is determined by a satellite-dependent C/A code. Specifically

$$p_b(\xi) = \sum_{i=0}^{19} p_a(\xi - iT_a)$$

where $20T_a = T_b$ and

$$p_a(\xi) = \sum_{i=0}^{1022} c_i u(\xi - iT_c)$$

where $c_i$ is the $i$-th component of the C/A code (satellite dependent) and $u(\xi)$ is a rectangular pulse (called chip in spread-spectrum jargon) defined by

$$u(\xi) = \begin{cases} 1, & \xi \in [0, T_c) \\ 0, & \text{otherwise,} \end{cases}$$

where $T_a = 1023T_c$.

It will also be useful to think of the baseband-equivalent signal as a train of $p_a$ pulses, i.e.,

$$s_E(t^s) = \sum_n a_n p_a(t^s - nT_a)$$

where $a_n = b_{\lfloor \frac{n}{20} \rfloor}$.

It will be useful to think of the above signal in the following terms: (i) each bit $b_k$ is mapped into a codeword $(a_{20k}, a_{20k+1}, \ldots, a_{20k+19})$ that consists of $20$ repetitions of $b_k$; (ii) the $\{a_k\}$ sequence is used as information sequence to modulate the $p_a$ pulse train.

The following diagram reminds us of the relationship between the $\{a_k\}$ and the $\{b_k\}$ sequences:

$$b_k \longrightarrow (a_{20k}, a_{20k+1}, \ldots, a_{20k+19}) \coloneqq (b_k, b_k, \ldots, b_k).$$

# THE RECEIVED SIGNAL

We derive the channel model and then the received signal model. First we need to talk about the various clocks of the GPS system.

# Timing Issues

The GPS *system* consists of *satellites*, *ground control stations*, and *GPS receivers*. Each of these subsystems has its own clock.

*Ground stations* are important to supervise and control satellites but are not fundamental to understand how the system works. For the purpose of our discussion we may assume that they are just the holder of a GPS-wide reference time called *the GPS time.* We will refer to it by the letter $t$.

*Satellites* keep track of the passage of time by means of precise clocks. However, due to relativistic effects (accelerations), satellite times are not aligned with the GPS time. Specifically, we assume that at GPS time $t$, the time $t^s$ shown by the clock of satellite $s$ is of the form

$$t^s = t + \delta t^s$$

for some offset $\delta t^s$ (which depends on the position of the satellite within the satellite's orbit).

*GPS receivers* have inexpensive clocks. Nevertheless, they may also be considered as advancing in synchrony over the interval of time of interest to us. Hence we may assume that the time $t_r$ shown by a receiver clock at GPS time $t$ is

$$t_r = t + \delta t_r.$$

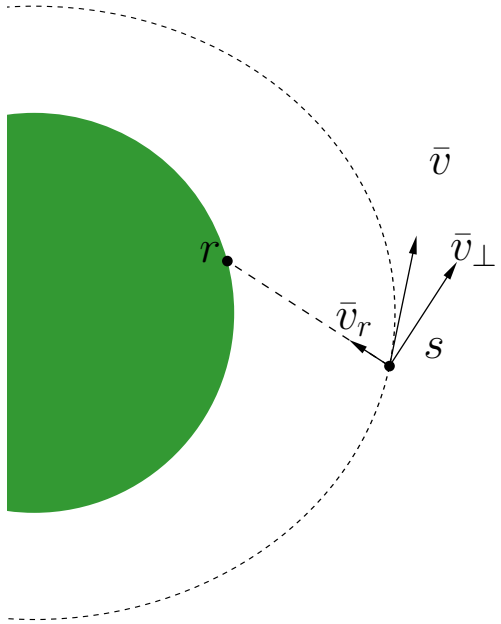
The fact that satellites have precise clocks just makes it easier, for the control station, to keep track of $\delta t^s$ and send it back to the satellite. The satellite will then send it down to the receiver along with other information necessary to determine the satellite position.

When we consider several satellites and possibly several receivers, we will write $t^{(i)}$ and $t_j$ to denote the time at satellite $i$ and GPS receiver $j$, respectively. For now we consider only one satellite and one receiver.

A note regarding notation: superscripts are used for satellites (up in the sky) whereas subscripts are used for the receiver (down on earth).

# The Received Signal

Over a short time interval, we can assume that a satellite is moving towards the receiver at some constant speed $v$. ($v$ is the magnitude of the radial velocity vector $\bar{v}_r$ depicted in the figure.) A negative value of $v$ means that the satellite is moving away from the receiver.



Hence, at GPS time $t$, the distance between the satellite and the receiver behaves as

$$d(t) = d_0 - vt$$

for some $d_0$.

A signal transmitted at GPS time $t_1$, reaches the receiver at GPS time $t_2$

$$t_2 = t_1 + \frac{d(t_1)}{c} = t_1(1-\nu) + \frac{d_0}{c}, \tag{1}$$

where $c$ is the speed of light and $\nu = \frac{v}{c}$.

At $t_1$, the satellite clock shows $t^s = t_1 + \delta t^s$ and at $t_2$, the receiver clock shows $t_r = t_2 + \delta t_r$. If we plug this into (1) to eliminate $t_1$ and $t_2$ we obtain

$$t_r - \delta t_r = (t^s - \delta t^s)(1-\nu) + \frac{d_0}{c}.$$

By solving for $t^s$, we find the time shown by the satellite when it sent a signal received at receiver time $t_r$, namely

$$t^s = \frac{t_r}{1-\nu} - \tau_0,$$

where $\tau_0 = \frac{\delta t_r + d_0/c}{1-\nu} - \delta t^s$.

Hence, neglecting the noise, the received signal at receiver time $t_r$ is

$$r(t_r) = s\Big(\frac{t_r}{1-\nu} - \tau_0\Big),$$

which is a delayed and time-scaled version of the transmitted signal.

The scaling is by $1/(1-\nu)$, which can be bigger (signal compression) or smaller (signal expansion) than 1.

Since

$$\frac{1}{1-\nu} = 1 + \nu + \nu^2 + \cdots$$

and $\nu \ll 1$, it is convenient to approximate $\frac{1}{1-\nu}$ by $1+\nu$. Using this approximation, from now on we consider the received signal to be

$$r(t_r) = s\Big(t_r(1+\nu) - \tau_0\Big)$$

plus AWGN noise.

# The Baseband-Equivalent Signal and the Doppler Effect

The Doppler effect shows up when we look at the baseband-equivalent received signal.

First we write the satellite signal $s(t)$ in terms of its baseband-equivalent $s_E(t)$

$$s(\xi) = \Re\{e^{j2\pi f_c \xi} s_E(\xi)\}.$$

Then

$$r(t_r) = s(t_r(1+\nu) - \tau_0) = \Re\{e^{j2\pi f_c t_r} e^{j2\pi f_c \nu t_r} e^{j\phi_0} s_E(t_r(1+\nu) - \tau_0)\}$$

where $\phi_0 = -2\pi f_c \tau_0$.

The baseband-equivalent received signal is thus

$$r_E(t_r) = e^{j\phi_0} e^{j2\pi f_c \nu t_r} s_E(t_r(1+\nu) - \tau_0).$$

We see that the baseband-equivalent signal has a residual carrier frequency $f_c\nu$ referred to as the "Doppler shift".

The parameters $\nu$, $\tau_0$ and $\phi_0 = -2\pi f_c \tau_0$ are the result of a linearisation of the time of flight (from satellite to receiver) with respect to some fixed time $\bar{t}_r$ in the neighborhood of the values of $t_r$ of interest.

They may be considered constant over a few bits but we need to be aware that they vary over longer intervals. For instance, while a satellite approaches a receiver $\nu$ is positive and is negative while the satellite is moving away.

# Summarizing the Received Signal

The received signal (this time including noise) looks like

$$y(t_r) = r_E(t_r, \phi_0, \nu, \tau_0) + z(t_r)$$

where

$$r_E(t_r, \phi_0, \nu, \tau_0) = e^{j\phi_0} e^{j2\pi f_c \nu t_r} \sum_i b_i p_b(t_r(1+\nu) - \tau_0 - iT_b). \tag{2}$$

To bring the received signal in a more familiar form we define the time-scaled bit-carrying pulse

$$p_\nu(\xi) = p_b(\xi(1+\nu))$$

and its duration $T_\nu = \frac{T_b}{1+\nu}$.

Now

$$r_E(t_r, \phi_0, \nu, \tau_0) = e^{j2\pi f_c \nu t_r} \sum_i e^{j\phi_0} b_i p_\nu(t_r - iT_\nu - \tau_1). \tag{3}$$

We see the following:

- the baseband-equivalent signal has a residual carrier frequency $\nu$ that needs to be tracked and the term $e^{j2\pi f_c \nu t_r}$ removed.
- there is a delay $\tau_1 = \frac{\tau_0}{1+\nu}$ that needs to be estimated so as to know the exact position of $p_\nu(t - \tau_1)$ and its shifted versions, shifted by multiples of $T_\nu = \frac{T_b}{1+\nu}$.
- the pulse itself has been time-scaled by a factor $(1 + \nu)$.
- the effect of $e^{j\phi_0}$ is to rotate the symbols $b_i$.

# Decoding Strategy

We will first estimate $f_c\nu$ and $\tau_1$ and remove the residual carrier by multiplying with $e^{-j2\pi f_c \nu t_r}$.

Then we implement the inner products. In the absence of noise the results will be $e^{j\phi_0}b_i$ times a scaling constant due to attenuation. Assuming that the estimates of $f_c\nu$ and $\tau_1$ are correct, the $i$-th inner product constitutes a sufficient statistic for the rotated symbol $e^{j\phi_0}b_i$. (The unknown scaling constant is immaterial for the slicer of antipodal symbols.)

At this point we could try to estimate $\phi_0$, remove its effect by multiplying the inner product result by $e^{-j\phi_0}$, and then estimate $b_i$ based on the sign of what we obtain. Unless we use information contained in the bit-sequence, the best we can do is estimate $\phi_0$ modulo $\pi$. If the estimate of $\phi_0$ is off by $\pi$ then (assuming no error induced by the noise) all the decoded bits will have "flipped". We will be able to detect if this is the case when we analyze the decoded bits sequence (more on this later).

Instead of estimating $\phi_0$, we adopt a suboptimal decision rule that consists of arbitrarily deciding that the first bit is a $1$ and then decide on the remaining bits based on the phase of the inner product results: if the phase of two subsequent inner product results is essentially the same then also the corresponding bits are assumed to have equal value. If the phase difference is essentially $\pi$ then the corresponding bits have different value. Once again, by analyzing the decoded bit-sequence we will be able to tell if the first decision was incorrect, in which case we "flip" all the bits.

# Discussion

How does the GPS channel relate to other channel models? Let us see what they are:

- The PDC channel accounts for additive white Gaussian noise (AWGN) and nothing else. It is not realistic for wireless communication since it does not account for attenuation and propagation delay. Dealing with attenuation and propagation delay is straightforward if there is a single path.
- A channel model that consists of a linear time-invariant filter followed by the AWGN channel is a bit more realistic. The filter can model a continuum of delayed and attenuated paths.
- If we allow the linear filter to be time-varying, the channel adequately models situations where there is multipath and the sender and/or the receiver and/or some of the obstacles in between are in movement.

The GPS channel model that we are using is arguably a simplest example of this kind: It has a single path and the delay varies linearly with time. It also has an attenuation but we are not concerned with it since the symbols are antipodal which implies that the slicer just compares its input with the threshold set at zero.

# DECODING

# Decoding The Satellite Signal

Decoding the satellite signal means estimating the bits sent by the satellite. Now we see how to do this in detail. First let us summarize.

The receiver has "captured" a piece of signal and has transformed it to baseband. The result is a function $y$ of the receiver clock

$$y : [T_{start}, T_{end}] \to \mathbb{C}.$$

The receiver knows that the signal has the following structure

$$y(t_r) = \sum_{i=0}^{M-1} (e^{j\phi_0} b_i) e^{j2\pi f_c \nu t_r} p_\nu \left( t_r - i\frac{T_b}{1+\nu} - T_1 \right) + z(t_r)$$

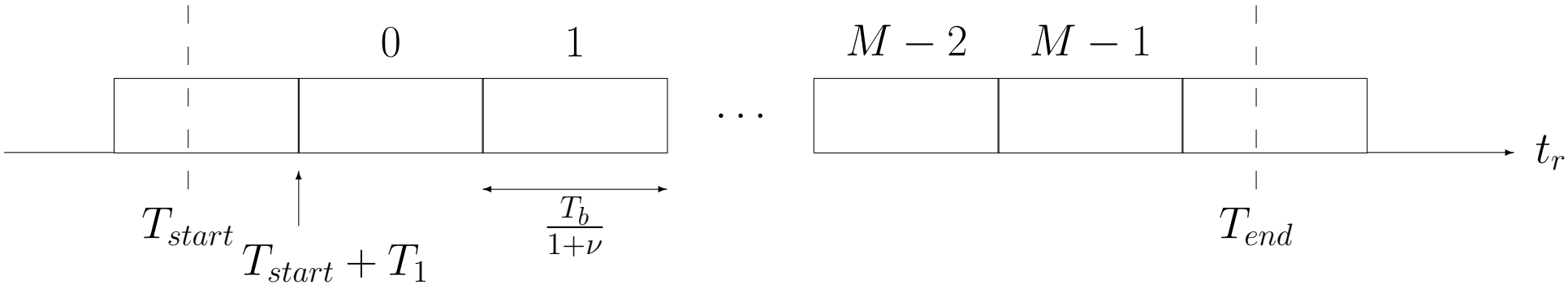
where $\phi_0$, $\nu$, $b_i$ ($b_i \in \{\pm 1\}$), $T_1$ ($0 \leq T_1 \leq \frac{T_b}{(1+\nu)}$) are all unknown and $z(t_r)$ is a realization of Gaussian noise.[1]

[1] There is a relationship between $\tau_1$, which can be quite large, and $T_1$ which is in $0 \leq T_1 \leq \frac{T_b}{(1+\nu)}$, but figuring out this relationship will not help us. All we need to know is that there is an uncertainty about the position of the pulses within the received signal and it suffices to know that uncertainty modulo the length of a pulse. The purpose of $T_1$ is to model that uncertainty.

With some abuse of notation, in the above expression we have re-indexed the symbols so that those in the interval $[T_{start}, T_{end}]$ are indexed by $0, 1, \ldots, M-1$ for some integer $M$.

Even if not relevant for what follows, for completeness we also point out that $T_1$ is $\tau_1$ modulo the symbol duration ($\frac{T_b}{(1+\nu)}$).

Pictorially, the bit-boundaries within $y$ look as following



Once we have determined (an estimate of) $f_c\nu$ and $T_1$, we pretend that they are correct, we remove the effect of the Doppler by multiplying the received signal by $e^{-j2\pi f_c \nu t_r}$, we neglect the portion of the signal prior to $T_{start} + T_1$ (see above figure) and from now on we are back to the situation of a modulated pulse train as studied in PDC, where the symbols have the form $e^{j\phi_0} b_i$ and the pulse has the form $p_\nu(t_r - T_1)$ (or $p_\nu(\xi)$ if we see the signal as a function of the new time variable $\xi$ which has value $0$ where the first pulse starts, i.e. $\xi = t_r - (T_{start} + T_1)$.)

We start by estimating $f_c\nu$ as well as the position of the first complete pulse $p_a$ within $y$.

To do so we use the fact that the inner product between the time functions $e^{j2\pi\tilde{f}t_r}p_\nu(t_r)$ and $e^{j2\pi f t_r}p_{\tilde{\nu}}(t_r+\tau)$ has a sharp peak when $f=\tilde{f}$ and $\tau=0$.

In fact, the sharpness as we vary $\tau$ is due to the fact that $p_a(\xi)$ has a sharp self-similarity function. (You may want to use MATLAB/Python to check this out).

The sharpness as a function of $\tilde{\nu}$ comes from the fact that the inner product between two truncated complex exponentials is maximum when they have the same frequency and decays rapidly as the frequency gap increases.

To implement the above idea, we take the inner products of $y(\xi+\tau)$ and $e^{j2\pi f\xi}p_a(\xi(1+\nu))$ and find the value of $f$ and $\tau$ for which the absolute value of the result is largest. The absolute value is needed since when we introduce $\tau$ we not only shift the pulse $p_\nu$ within $y$ but we also change $e^{j2\pi f_c\nu t_r}$ into $e^{j2\pi f_c\nu\tau}e^{j2\pi f_c\nu t_r}$ thereby introducing a rotation by $e^{j2\pi f_c\nu\tau}$.

The sequence of inner products with different values of $\tau$ is obtained in one shot by correlating $y(\xi)$ and $e^{j2\pi f_c\nu\xi}p_a(\xi(1+\nu))$.

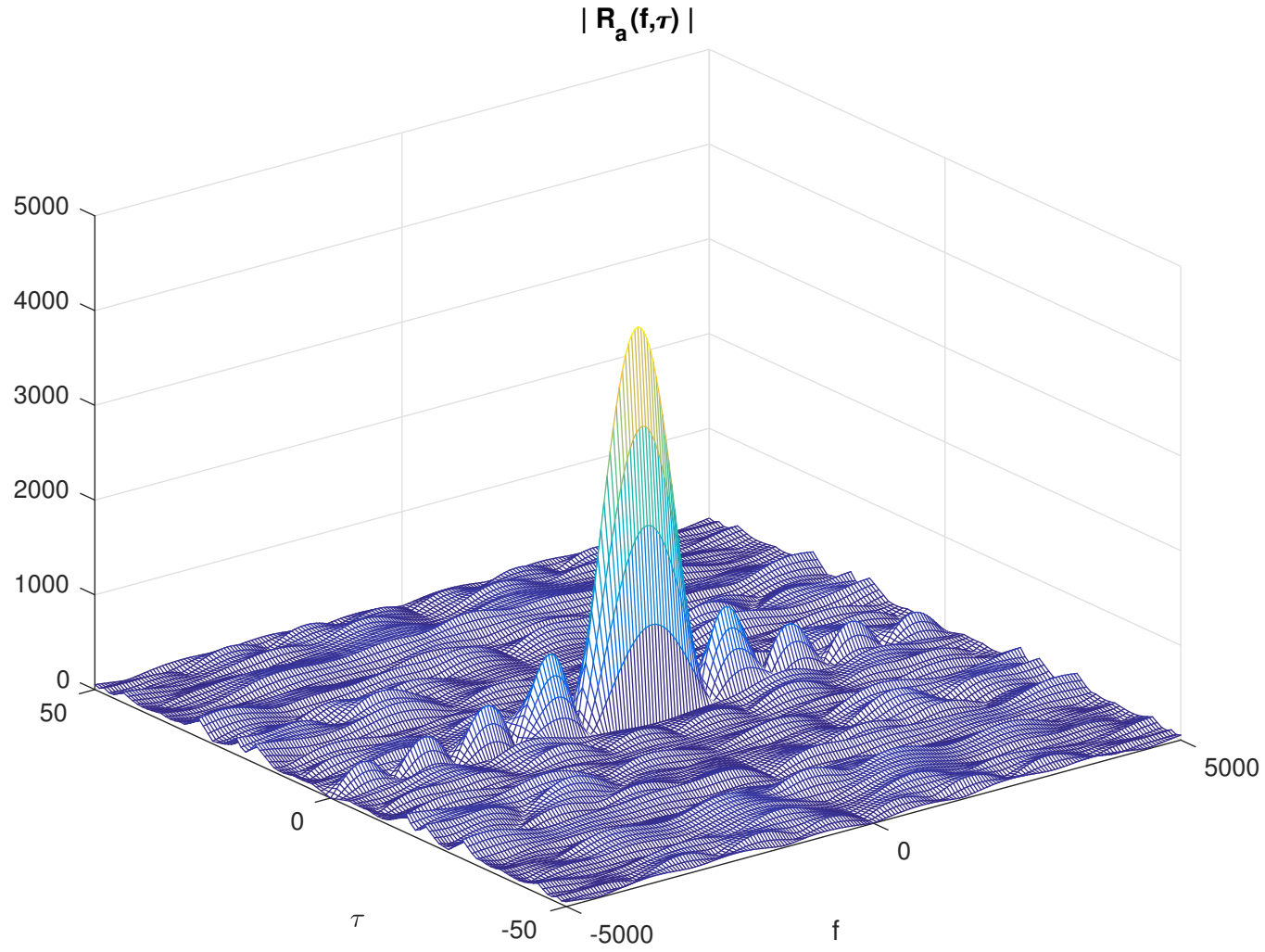
We can increase the speed of our implementation with a negligible loss in performance if we correlate with $e^{j2\pi f\xi}p_a(\xi)$ instead of $e^{j2\pi f\xi}p_a(\xi(1+\nu))$. The result is essentially the same since $\nu \ll 1$. We gain in speed since we don't have to sample $p_a(\xi(1+\nu))$ for every value of $\nu$.

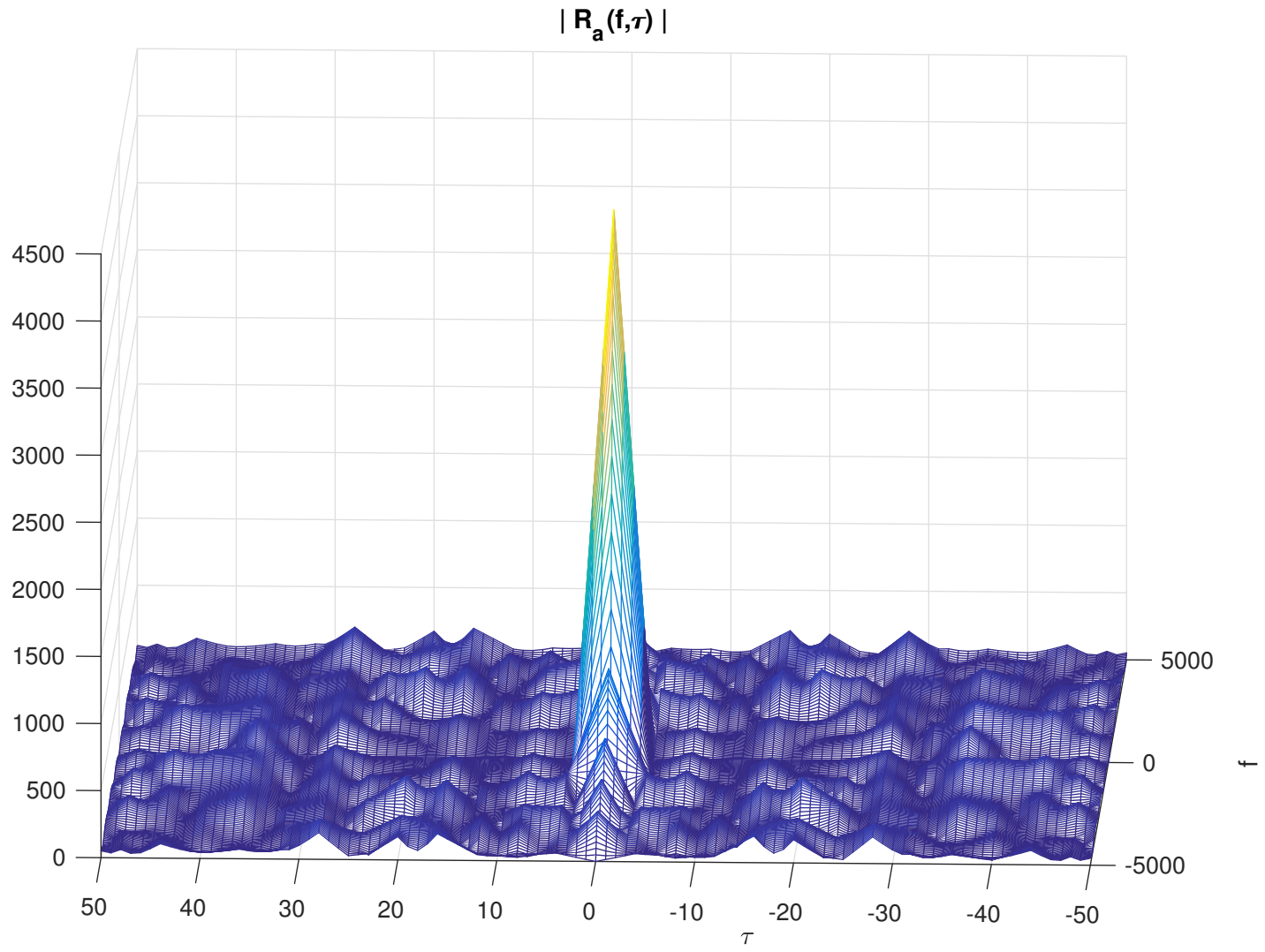
According to the above discussion, define

$$R_a(f,\tau)=\int_0^{T_a} y(\alpha+\tau)e^{-j2\pi f\alpha}p_a^*(\alpha)d\alpha,$$

where $p_a^*(\alpha)$ is the complex conjugate of $p_a(\alpha)$, and let $\hat{f}_d$ and $\hat{\tau}$ be the maximizers of $|R_a(f,\tau)|$ in the specified range.

The following plots illustrate the result of $|R_a(f,\tau)|$ when we use $p_a$ instead of $y$.




| R_a(f,τ) |
4500
4000
3500
3000
2500
2000
1500
1000
500
0
50
40
30
20
10
0
-10
-20
-30
-40
-50
τ
5000
0
-5000
f

The result of maximizing $|R_a(f,\tau)|$ gives us an estimate $\hat{f}_d$ of $f_d = \nu f_c$ and an estimate $\hat{\tau}$ of the position where the first shifted $p_a$ starts. The search for the maximizer $\hat{f}_d$ will take place over the range determined by the possible Doppler values, namely in the range $\pm 5$ [KHz].

It turns out that the precision of the estimate $\hat{f}_d$ obtained by correlating with one C/A code is not sufficient. To increase the precision we refine the estimate of $f_d$ by correlating with the concatenation of $10$ C/A codes[2]. With a slight abuse of notation let us also denote by $\hat{f}_d$ the fine estimate. (The coarse estimate will no longer be used.)

The search for $\hat{\tau}$ should be in the range $[0, \frac{T_a}{1+\nu}]$ but, once again, it makes essentially no difference if we search within the interval $[0, T_a]$.

At this point we may assume that the first $p_a$ pulse within $y$ starts at $t_r = \hat{\tau}$, and we can remove the Doppler for the data starting from that point on. Recall that, once in a while, the parameter $\hat{f}_d$ needs to be adjusted[3].

[2]We are choosing 10 since this is the largest integer $n$ for which the following in true: from any position at the boundary of a C/A code in the received signal, either the previous or the next $n$ C/A codes fall within one bit.

[3]In our implementation we do this every 5 bits.

Since we work with chunks of data (it would be unwise to load all the samples into a very large MATLAB/Python vector), in removing the Doppler one has to ensure phase continuity. Specifically, if in chunk $i$ we multiply the received data by $e^{j\theta_i}e^{-j2\pi f_i\xi}$ where $\xi$ runs from $0$ to $T$, and in chunk $i+1$ we multiply by $e^{j\theta_{i+1}}e^{-j2\pi f_{i+1}\xi}$, then we have to choose $\theta_{i+1} = \theta_i - 2\pi f_i T$. For the first chunk we may choose $\theta_1 = 0$.

Next we need to find the beginning of a bit. Recall that a bit-carrying pulse $p_b$ contains 20 repetitions of the $p_a$ pulse. To find the beginning of a bit we inspect $R_a(0, \hat{\tau} + kT_a)$ $k = 1, 2, \ldots$ (we are using $f = 0$ since we assume that the Doppler has been removed) until we detect a change in phase by roughly $\pi$ (exactly $\pi$ if we did not have noise and the parameters stayed fixed). Such a change denotes the beginning of a bit.

Let us say that it occurs when $k = \hat{k}$. Accordingly, we set $\hat{T}_1 = \hat{\tau} + \hat{k}T_a$. This marks the beginning of a bit.

Now it is convenient to shift our view from a train of $p_a$ pulses to a train of $p_b$ pulses. As a function of $p_b$, the Doppler-corrected signal $\bar{y}(t_r)$ looks like

$$\bar{y}(t_r) = \sum_{i=0}^{M-1} e^{j\phi_0} b_i p_b \left( (1+\hat{\nu}) \left( t_r - i\frac{T_b}{1+\hat{\nu}} - \hat{T}_1 \right) \right) + z(t_r).$$

Using the approximation $(1+\hat{\nu}) \approx 1$, it is convenient to write the above signal in the familiar form

$$\bar{y}(t_r) \approx \sum_{i=0}^{M-1} e^{j\phi_0} b_i p_b \left( t_r - iT_b - \hat{T}_1 \right) + z(t_r).$$

It should be clear how to estimate the bit sequence from the above signal.

To update $\hat{f}_d$ by means of the pulse that carries the $i$-th bit we let the next $\hat{f}_d$ be the $f$ that maximizes $|R_b(f, \hat{T}_i)|$, where $R_b$ is defined as $R_a$ with the pulse $p_b$ instead of $p_a$ and $\hat{T}_i = \hat{T}_{i-1} + T_b$.

The above updating rule for $\hat{T}_i$ is used over short spans. Once in a while[4], also $\hat{T}_i$ needs to be updated. We do so by choosing it as the $\tau$ around the current $\hat{T}_i$ that maximizes $|R_b(\hat{f}_d, \tau)|$.

[4] again every 5 bits in our implementation

# What Next

Once we have decoded the bits of the visible satellites (at least 4 of them), for each such satellite we will

- Find the beginning of the first subframe. This is done by looking for a certain bit pattern. If we find the negative of the bit pattern it means that we need to "flip" all bits. (The first decoded bit, arbitrarily declared as a $1$, is indeed a $-1$.)
- Identify the five pages. (The first three pages are actually sufficient.)
- Verify that the parities are fulfilled.
- Extract the ephemeris information from the decoded bit sequence.
- Determine the pseudoranges.

We will give you routines to do some of the above and will give additional information to implement the missing parts.