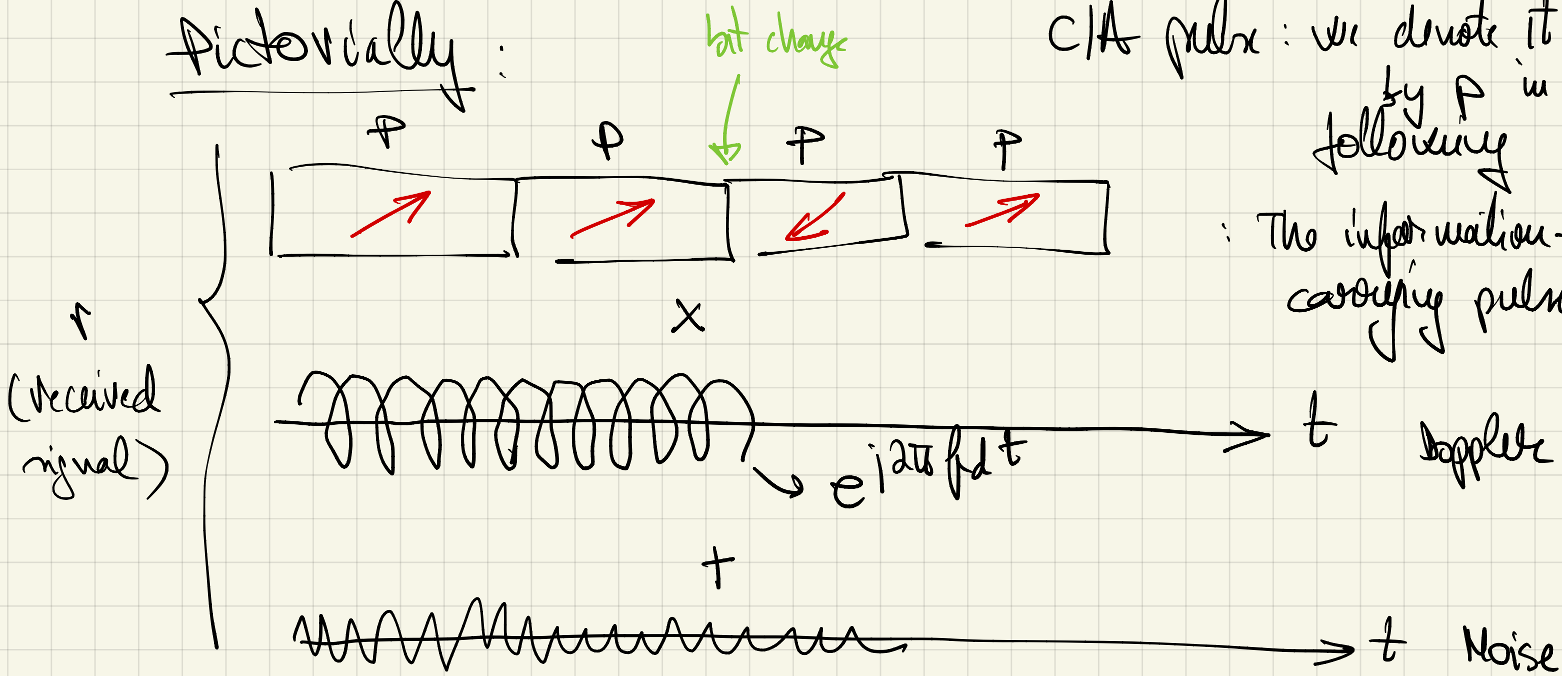
Decoding GPS - Part 2: Getting the bits

Where we are: For the visible satellites:

- we have an estimate of the Doppler frequency f_d
- we know the position (index) in the received samples where the first C/A pulse starts

Pictorially:



C/A pulse: we denote it by p in the following

: The information-carrying pulses

Next: Find the start of a bit → find the start of the first complete bit in the received sequence of samples

Basic idea

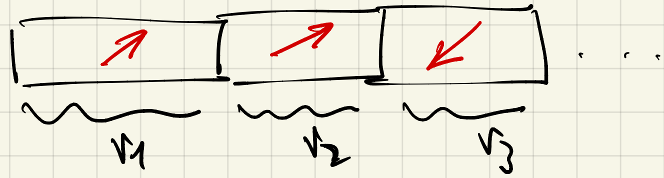
1) We correct for the Doppler: we multiply r with

$$e^{-j2\pi f_d T_s n} \qquad n = 0, 1, 2 \ldots$$

T_s: sampling time

Now

v :



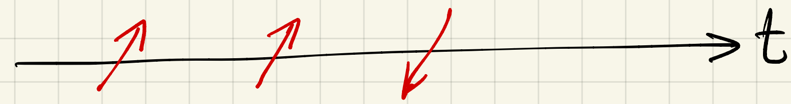
$+$

noise

2) Compute

$$\underline{y_i} = \langle v_i, \underline{p} \rangle \quad \leftarrow \text{inner product}$$

We obtain:

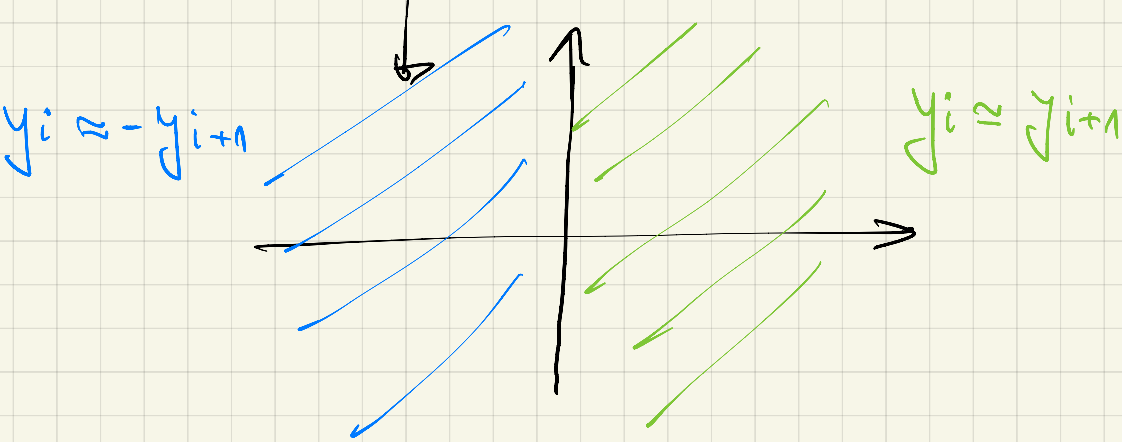


3) We look at the phases of the inner products, in particular at the transitions $y_i \to y_{i+1}$

Here is how

$$\measuredangle\left(y_i \cdot y_{i+1}^*\right) = \measuredangle y_i - \measuredangle y_{i+1}$$

$$\approx \begin{cases} 0 & \text{if } y_i \approx y_{i+1} \\ \pi & \text{if } y_i \approx -y_{i+1} \end{cases}$$



Note: Since it is worse to erroneously detect the beginning of a bit than to miss it, we decide the following way:

Steps 1) and 2) can be done more efficiently

Let $r_1, r_2, r_3 \ldots$ be as before

Let $c_1 = e^{j 2\pi f_d T_s [0, 1, \ldots L-1]}$ L: length (in samples) of a C/A

$c_2 = e^{j 2\pi f_d T_s [L, L+1, \ldots 2L-1]}$

$\vdots$

$$y_i = \langle \underbrace{r_i .* \mathrm{conj}(c_i)}_{\text{remove Doppler from } r_i}, p \rangle$$

$$= \langle r_i, p .* c_i \rangle$$

<u>Define</u>

$c = c_1$

$c_1 = c \cdot e^{j\phi_1}$ $\phi_1 = 0$

$c_2 = c \cdot e^{j\phi_2}$ $\phi_2 = \phi_1 + 2\pi f_d T_s L$

$c_3 = c \cdot e^{j\phi_3}$ $\phi_3 = \phi_2 + \underbrace{2\pi f_d T_s L}_{\text{correction of phase from } \phi_i \text{ to } \phi_{i+1}}$

$$y_i = \langle r_i, p .* c_i \rangle$$

$$= \langle r_i, p .* c \cdot e^{j\phi_i} \rangle$$

$$= \langle r_i, \underbrace{p .* c}_{\text{computed once for all}} \rangle e^{-j\phi_i}$$

Once the start of the first bit is found:

1) Compute $y_i = \langle r_i, q \rangle e^{-j\phi_i}$ $\quad$ $\underline{r_i}$: chunks of consecutive $\underline{20\, p}$

$$q = \underbrace{[p, p \dots p]}_{20 \text{ repetitions of } p} .* e^{j 2\pi f_d T_s [0,1 \dots 20L-1]}$$

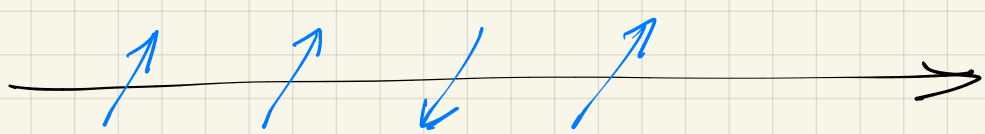
$$\phi_0 = 0$$

$$\phi_1 = \phi_0 + 2\pi f_d T_s 20L$$

$$\vdots$$

$$\phi_{i+1} = \phi_i + 2\pi f_d T_s 20L$$

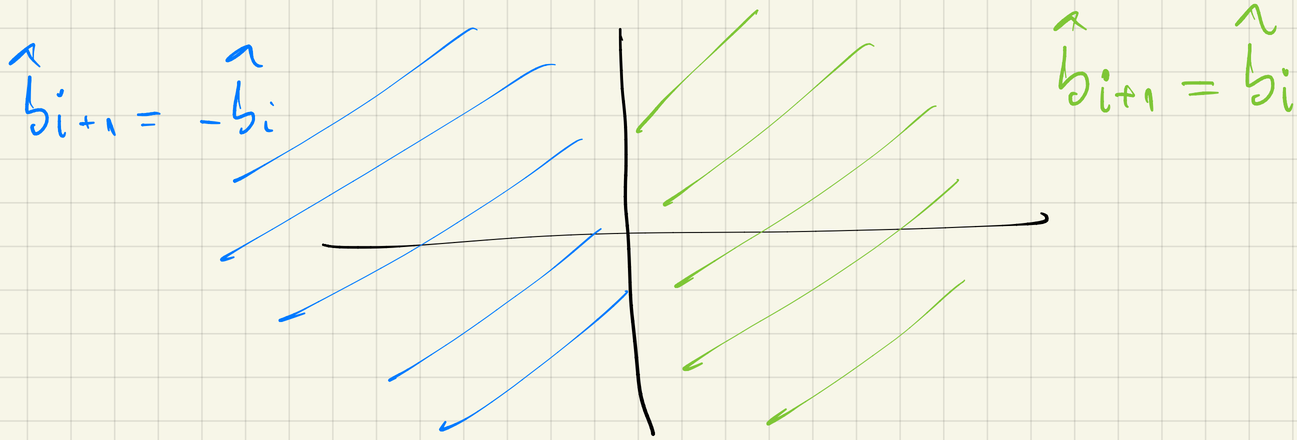
<u>what we get</u>



2) We arbitrarily decide $\hat{b}_1 = 1$

and $\hat{b}_{i+1} = \begin{cases} \hat{b}_i & \text{if } Re\{y_i * y_{i+1}^*\} \geq 0 \\ -\hat{b}_i & \text{otherwise} \end{cases}$

$b_i \in \{\pm 1\}$



If the assumption $\hat{b}_1 = 1$ is incorrect, later on we can detect it and correct.

Once in a while, we have to reassess the bit boundaries.

<u>Recall</u> T_b = duration of a bit at the satellite

At the receiver: $\frac{T_b}{1+\gamma} = \frac{T_b}{1+\frac{f_d}{f_c}} \simeq T_b$

$f_d = \nu f_c$

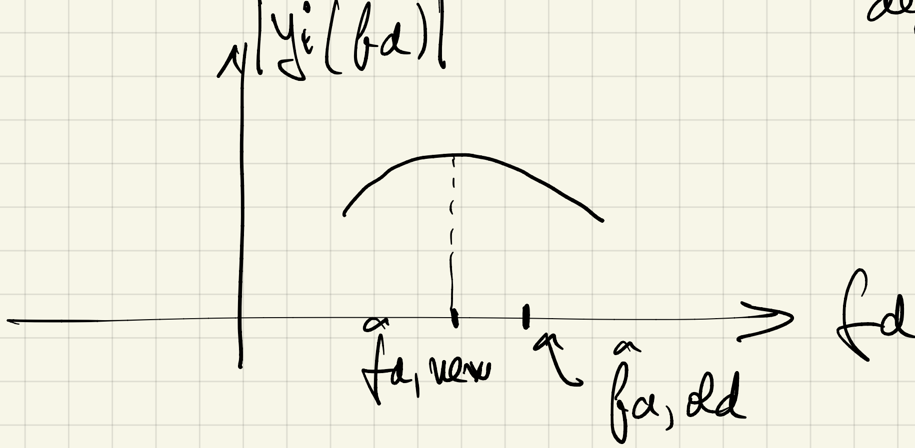
this is a good approximation over the duration of a few bits

So once in a while, we have to compute y_i using a slightly different alignment of q, and re-align by maximizing $|y_i|$.

Once in a while we have to re-estimate f_d

In the absence of noise, the function

$|y_i(\hat{f}_d)|$ is approximated by a polynomial of degree 2.



Let $f(x) = ax^2 + bx + c$ be the approximation

$$f'(x) = 2ax + b \overset{!}{=} 0 \Rightarrow \underbrace{\hat{x} = -\frac{b}{2a}}_{\text{The new Doppler estimate}}$$

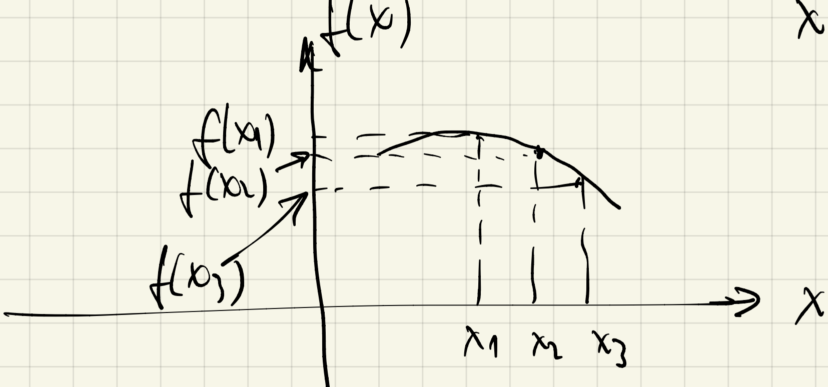
How to find a and b:

Let $f_i = f(x_i)$

$$x_1 = \hat{f}_{d,old} - f_{corr}$$

$$x_2 = \hat{f}_{d,old}$$

$$x_3 = \hat{f}_{d,old} + f_{corr}$$



$$f = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix} = \begin{pmatrix} a x_1^2 + b x_1 + c \\ a x_2^2 + b x_2 + c \\ a x_3^2 + b x_3 + c \end{pmatrix}$$

$$= \underbrace{\begin{pmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \end{pmatrix}}_{D\,:\ \text{known}} \underbrace{\begin{pmatrix} a \\ b \\ c \end{pmatrix}}_{\text{unknown}}$$

How to solve:

In Matlab use "left matrix" multiplication

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} = \underbrace{D \backslash}\, f$$

(using least-squares) to solve

Having ~~to~~ solve the equation

$$\cancel{A} \cdot x = b$$

it looks for x which minimizes

$$\| b - A \cdot x \|$$

In Python use

numpy.linalg.lstsq(...)