Modern Digital Communications: A Hands-On Approach

GPS: Ephemerides and Pseudoranges

Prof. Bixio Rimoldi

Last revision: Sept. 13, 2017

Goal

This week's goals are:

1. Go from bits to pages
2. Extract the ephemeris from the pages
3. Measure the pseudoranges

The Data Structure

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

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

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

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

Word made of 30 bits: 600ms total

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

Subframe made of 10 words: 6s total

1	2	3	4	5	6	7	8	9	10

Page made of 5 subframes: 30s total

1	2	3	4	5

Step-By-Step Towards A Page

We have written the following functions. You will receive them incomplete and your assignment consists in filling in the missing details.

```
function [s, ind] = removeExcessBits(bits)
%REMOVEEXCESSBITS Extracts the complete subframes in the bit sequence obta
%    [S, IDX] = REMOVEEXCESSBITS(BITS) detects the start of the first
%    subframe in the row vector BITS (values in {-1,+1}) by correlating
%    with the GPS preamble (stored in gpsc.preamble as a sequence of
%    1s and 0s). If the negative of the preamble is found all bits
%    are inverted. IDX is the index into BITS where the first subframe
%    starts; incomplete subframes at the beginning and at the end of
%    BITS are removed and the sequence of bits is converted into a
%    sequence of {0,1} values ({1,-1} <-> {0,1}) and returned into
%    vector S.
%
% Hints:
```

```
% - The preamble is stored in gpsc.preamble as a sequence of 0s and 1s.
% - It is not enough to just find one preamble: the preamble is 8 bits
%   long and it is not unlikely that this 8-bit sequence is also
%   present somewhere in the middle of the data sequence. You should
%   use the fact that there is one preamble in every subframe (every
%   300 bits).
% - Use round() on your correlation or inner product values in order to
%   eliminate MATLAB's rounding errors.

function s = establishParity(ws)
%ESTABLISHPARITY Checks the parity of the GPS words
%   S = ESTABLISHPARITY(WS) checks the parity of the subframe sequence
%   WS and flips the first 24 bits if the last bit of the previous
%   word is 1. The result is S. (The last two bits of each subframe
%   are 0 by convention.) If the parity check fails, a warning is
%   issued; otherwise a message indicating success is displayed. Both
%   the input WS and the output S are binary (0 and 1) row vectors
%   containing a multiple of subframes.
```

There are two things you need to know about parity.

The first thing to know is that the first 24 bits that form a word may have been flipped (in purpose) by an encoder. The encoder may do so to make the signal as DC-free as possible. They have been flipped if and only if the last bit of the preceding word is a 1. If it is the case you'll have to flip them back.

The other relevant information is that bits 25 through 30 are parity bits. They are obtained according to the table in the following slide.

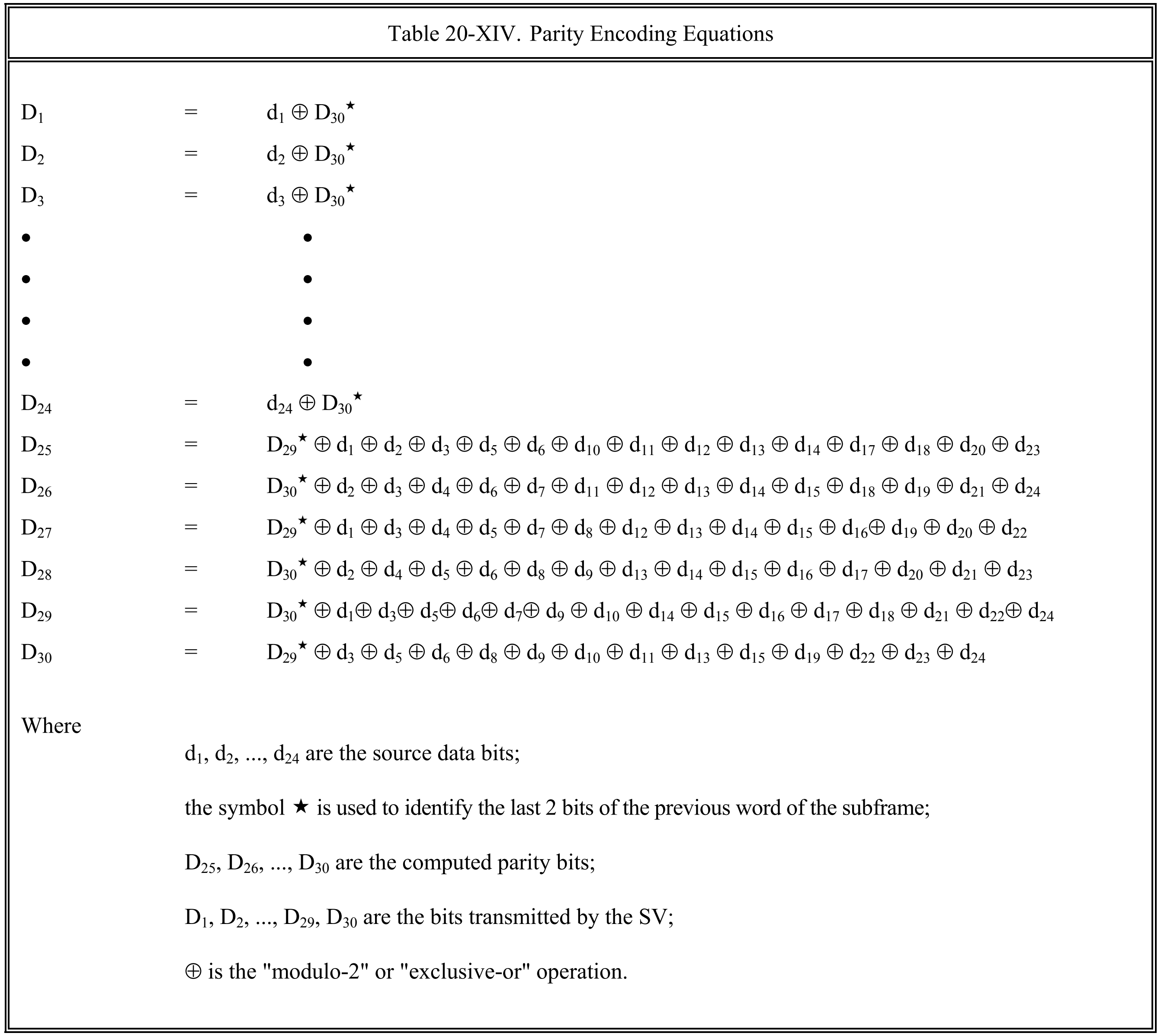


Table 20-XIV. Parity Encoding Equations

D_1	=	$d_1 \oplus D_{30}^{\star}$
D_2	=	$d_2 \oplus D_{30}^{\star}$
D_3	=	$d_3 \oplus D_{30}^{\star}$
•		•
•		•
•		•
•		•
D_{24}	=	$d_{24} \oplus D_{30}^{\star}$
D_{25}	=	$D_{29}^{\star} \oplus d_1 \oplus d_2 \oplus d_3 \oplus d_5 \oplus d_6 \oplus d_{10} \oplus d_{11} \oplus d_{12} \oplus d_{13} \oplus d_{14} \oplus d_{17} \oplus d_{18} \oplus d_{20} \oplus d_{23}$
D_{26}	=	$D_{30}^{\star} \oplus d_2 \oplus d_3 \oplus d_4 \oplus d_6 \oplus d_7 \oplus d_{11} \oplus d_{12} \oplus d_{13} \oplus d_{14} \oplus d_{15} \oplus d_{18} \oplus d_{19} \oplus d_{21} \oplus d_{24}$
D_{27}	=	$D_{29}^{\star} \oplus d_1 \oplus d_3 \oplus d_4 \oplus d_5 \oplus d_7 \oplus d_8 \oplus d_{12} \oplus d_{13} \oplus d_{14} \oplus d_{15} \oplus d_{16} \oplus d_{19} \oplus d_{20} \oplus d_{22}$
D_{28}	=	$D_{30}^{\star} \oplus d_2 \oplus d_4 \oplus d_5 \oplus d_6 \oplus d_8 \oplus d_9 \oplus d_{13} \oplus d_{14} \oplus d_{15} \oplus d_{16} \oplus d_{17} \oplus d_{20} \oplus d_{21} \oplus d_{23}$
D_{29}	=	$D_{30}^{\star} \oplus d_1 \oplus d_3 \oplus d_5 \oplus d_6 \oplus d_7 \oplus d_9 \oplus d_{10} \oplus d_{14} \oplus d_{15} \oplus d_{16} \oplus d_{17} \oplus d_{18} \oplus d_{21} \oplus d_{22} \oplus d_{24}$
D_{30}	=	$D_{29}^{\star} \oplus d_3 \oplus d_5 \oplus d_6 \oplus d_8 \oplus d_9 \oplus d_{10} \oplus d_{11} \oplus d_{13} \oplus d_{15} \oplus d_{19} \oplus d_{22} \oplus d_{23} \oplus d_{24}$

Where

$d_1, d_2, \ldots, d_{24}$ are the source data bits;

the symbol $\star$ is used to identify the last 2 bits of the previous word of the subframe;

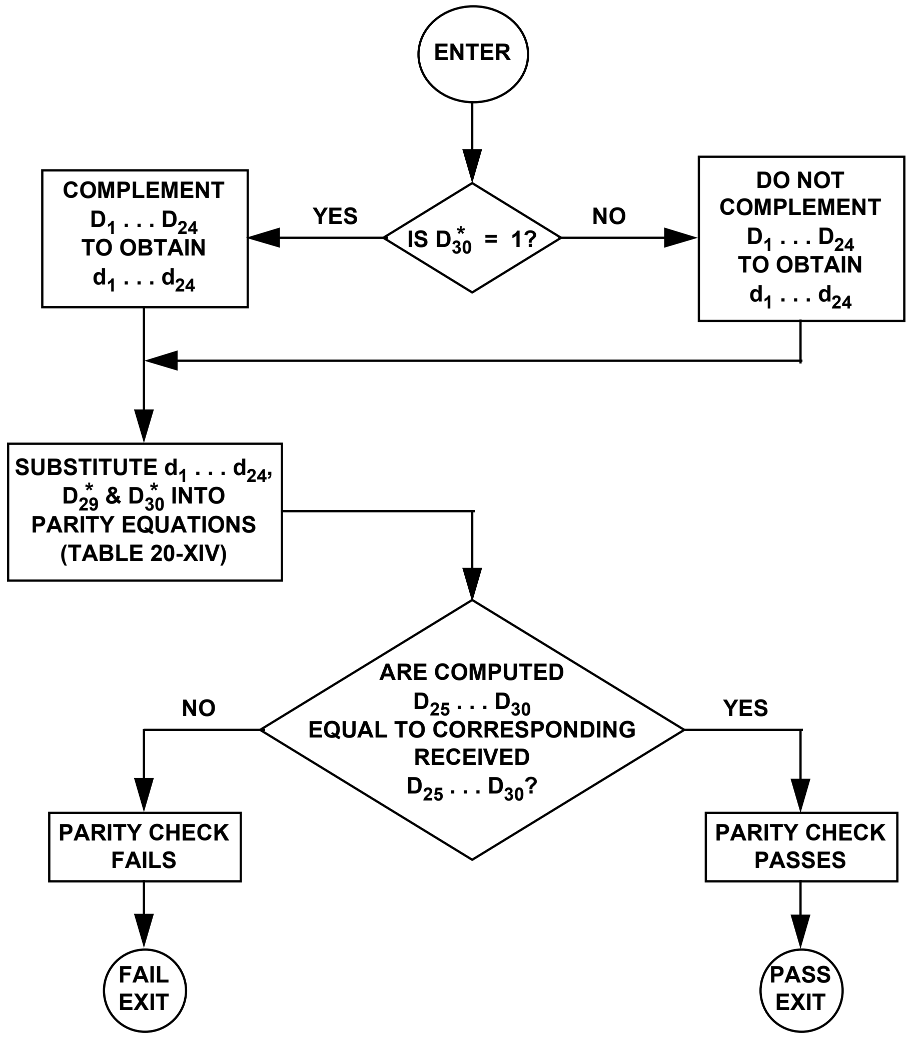
$D_{25}, D_{26}, \ldots, D_{30}$ are the computed parity bits;

$D_1, D_2, \ldots, D_{29}, D_{30}$ are the bits transmitted by the SV;

$\oplus$ is the "modulo-2" or "exclusive-or" operation.

Table 20-XIV mentioned next slide. (Courtesy of Tsui)

Here is a flowchart of what `establish_parity` does for each word.


ENTER
IS $D_{30}^* = 1$?
YES
NO
COMPLEMENT $D_1 \dots D_{24}$ TO OBTAIN $d_1 \dots d_{24}$
DO NOT COMPLEMENT $D_1 \dots D_{24}$ TO OBTAIN $d_1 \dots d_{24}$
SUBSTITUTE $d_1 \dots d_{24}$, D_{29}^* & D_{30}^* INTO PARITY EQUATIONS (TABLE 20-XIV)
ARE COMPUTED $D_{25} \dots D_{30}$ EQUAL TO CORRESPONDING RECEIVED $D_{25} \dots D_{30}$?
NO
YES
PARITY CHECK FAILS
PARITY CHECK PASSES
FAIL EXIT
PASS EXIT

(Courtesy of Tsui)

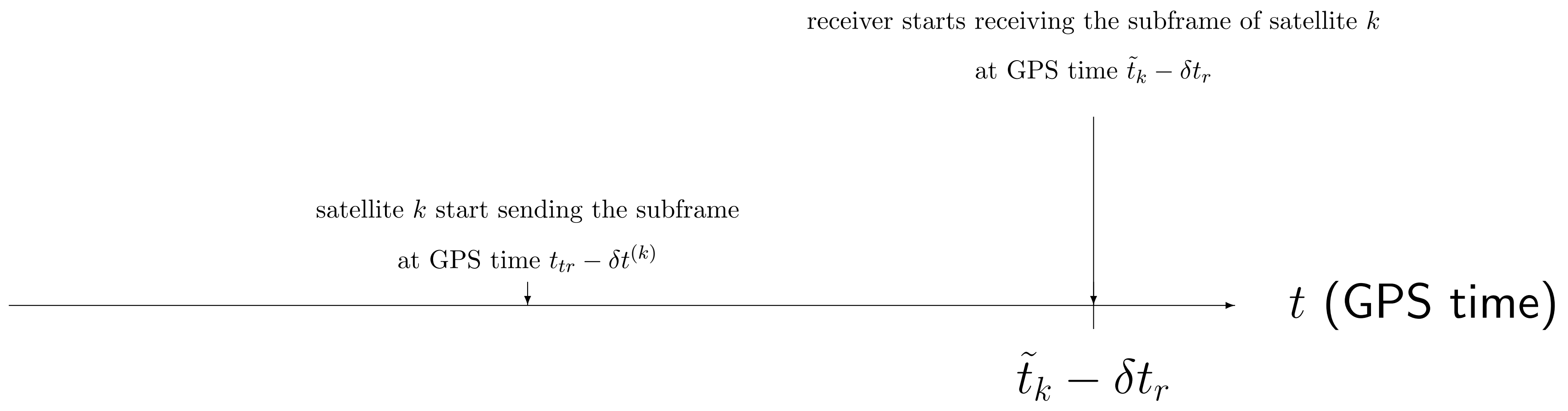
The next function creates a matrix that has, in its columns, the subframes. To do this you need to know that the subframe number is encoded in bit 50 (MSB), 51, and 52 (LSB).

```
function [subframes, subframesIDs] = bits2subframes(bits)
%BITS2SUBFRAMES Returns a matrix containing the subframes in the desired o
%   [SUBFRAMES, SUBFRAMESIDS] = BITS2SUBFRAMES(BITS) returns the matrix
%   SUBFRAMES having as its columns the subframes extracted from BITS.
%   The IDs of the subframes are returned in SUBFRAMESIDS.  BITS must
%   be a row vector containing a concatenation of subframes (0 and 1
%   elements) that have already been checked for parity.  Provided that
%   BITS is sufficiently long, SUBFRAMES contains the subframes with
%   id 1,2,3, in that order. (It might contain additional subframes.)
%   These are the subframes that we use to obtain the ephemerides.
```

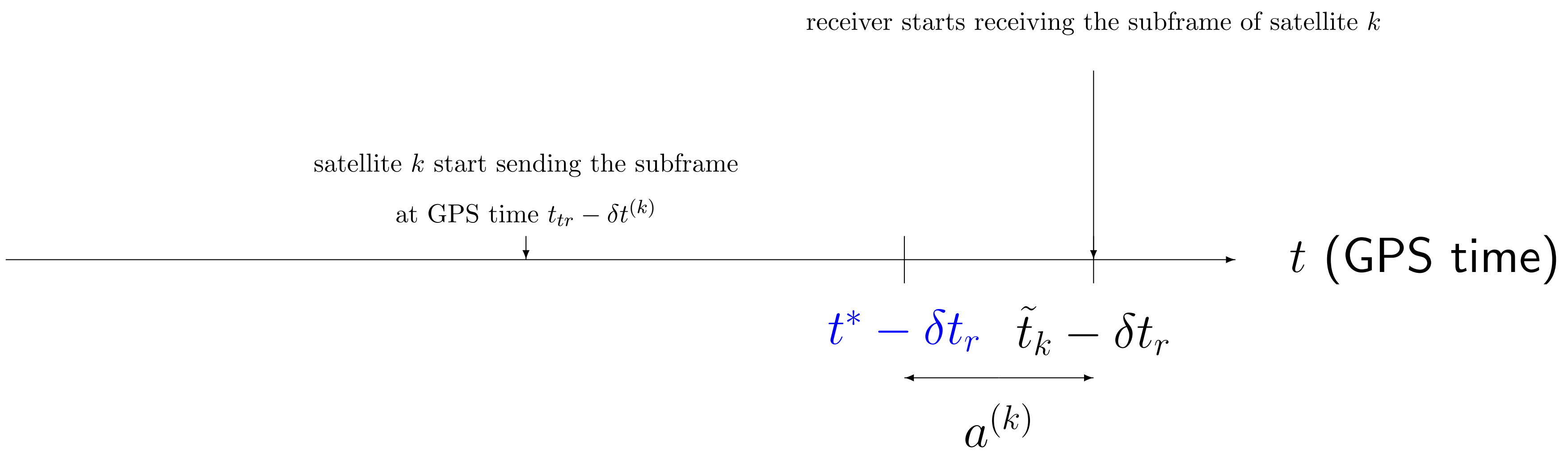
Measuring Pseudoranges

Satellite clock-errors apart, all satellites start sending a new subframe at the same time called t_{tr} (obtained from the data sent by the satellite). The actual GPS time when satellite k start sending the frame is $t_{tr} - \delta t^{(k)}$.

Subframes are numbered, so it is possible for the receiver to know the reading $\tilde{t}_k$ of its clock when it receives the start of the subframe that was sent at GPS time $t_{tr} - \delta t^{(k)}$. See the figure.

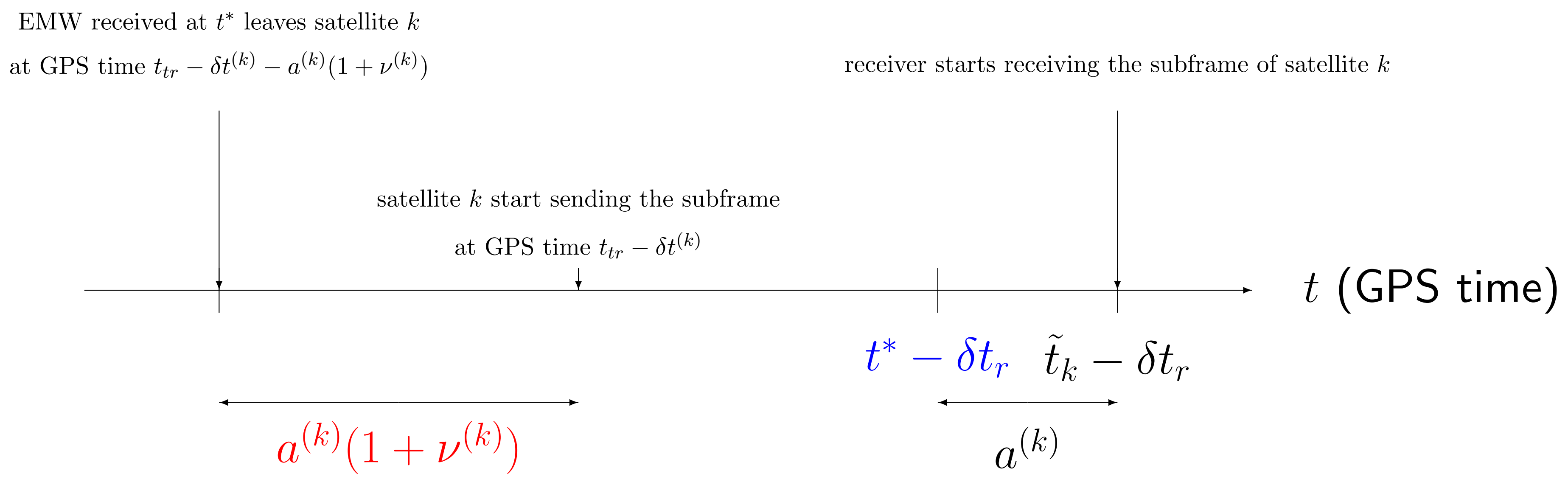

receiver starts receiving the subframe of satellite k
at GPS time $\tilde{t}_k - \delta t_r$
satellite k start sending the subframe
at GPS time $t_{tr} - \delta t^{(k)}$
t (GPS time)
$\tilde{t}_k - \delta t_r$

Fix an arbitrary time $t^* \leq \tilde{t}_k$ for all k (receiver clock). We will determine the position when the receiver clock reads t^*.


receiver starts receiving the subframe of satellite k
satellite k start sending the subframe
at GPS time $t_{tr} - \delta t^{(k)}$
t (GPS time)
$t^* - \delta t_r$
$\tilde{t}_k - \delta t_r$
$a^{(k)}$

Let $a^{(k)} \geq 0$ be the time elapsed between t^* and $\tilde{t}_k$ (see figure).

The electromagnetic wave that arrived at time t^* left the kth satellite at GPS time $t_{tr} - \delta t^{(k)} - a^{(k)}(1 + \nu^{(k)})$.


EMW received at t^* leaves satellite k
at GPS time $t_{tr} - \delta t^{(k)} - a^{(k)}(1+\nu^{(k)})$
receiver starts receiving the subframe of satellite k
satellite k start sending the subframe
at GPS time $t_{tr} - \delta t^{(k)}$
t (GPS time)
$t^* - \delta t_r$
$\tilde{t}_k - \delta t_r$
$a^{(k)}(1+\nu^{(k)})$
$a^{(k)}$

Recall that $\tau_f^{(k)}(t^*)$ is the time of flight of the electromagnetic wave sent by satellite k and received at time t^* (receiver clock reading). Hence

$$\tau_f^{(k)}(t^*) = t^* - \delta t_r - t_{tr} + \delta t^{(k)} + a^{(k)}(1+\nu^{(k)}).$$

If we remove the terms that do not depend on k, we obtain valid corrected pseudorange

$$\rho_c^{(k)} = c(\delta t^{(k)} + a^{(k)}(1+\nu^{(k)})).$$

If we remove also the satellite clock error we obtain the pseudorange

$$\rho^{(k)} = c(a^{(k)}(1+\nu^{(k)})).$$

Notice that even if not shown explicitly, both pseudoranges depend on t^* (in fact both $a^{(k)}$ and $\nu^{(k)}$ depend on t^*).

We may choose t^* to be the smallest $\tilde{t}_k$ among all visible satellites.

Notice that to obtain $a^{(k)}(1+\nu^{(k)})$ we may count the fraction of bits between t^* and $\tilde{t}_k$ and multiply the result by T_b. This will indeed give us the amount of time it took the satellite to send that chunk of signal.

Hence, to be able to determine the pseudorange, it is sufficient that we keep track of the index to the first sample of each bit.

From Page To Ephemeris

Extracting the ephemeris (and other information) from a page is now only a matter of knowing where to look.

We'll give you a function, `readEphemeris`, that extracts all the information contained in a page. GPS uses four formats for data description. `readEphemeris` uses four subfunctions, one for each of the formats. They are:

- `read_unsigned` to read unsigned integers
- `read_signed` to read integers
- `read_2part_unsigned` to read unsigned integers that occupy two chunks of bits
- `read_2part_signed` to read integers that occupy two chunks of bits

Good Luck!