ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
School of Computer and Communication Sciences

Handout 8 Modern Digital Communications
Assignment 5 October 8, 2025 (due Oct 14, 2025)

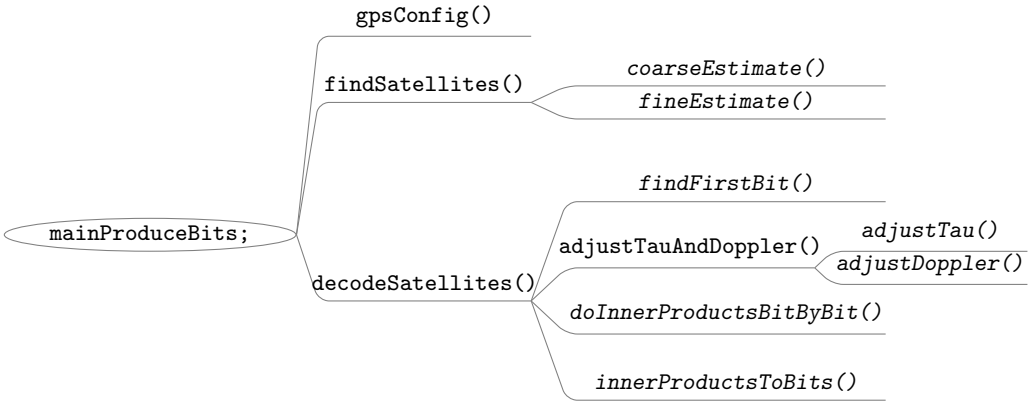
GPS Signal Synchronization

The goal of this assignment is to identify the visible satellites and to estimate their delays and Doppler shifts.

First download the code framework from the course webpage[1]. Extract the archive in a directory of your choice, and move the file `01.mat` in the folder `data/`. This file contains the samples of a received GPS signal. The code framework contains the functions for GPS synchronization and decoding, and will be used both for this assignment and for the following one.

Code Organization

To get an overall idea of how the program works, read through the provided code before you start writing your own code. At the same time, have a look at following graph that reflects the program structure[2]



- The script `mainProduceBits` is the main file that puts everything together to achieve the goal of the next assignment.
- The structure/module `gpsc` contains all the GPS system parameters, such as sampling rate, chip rate, maximum Doppler frequency, etc.

  MATLAB The initialization routine `gpsConfig()` creates and initializes the data structure `gpsc`. Type `help gpsConfig` to see a full list of the parameters and their meaning. To have access to the fields of this structure, you should declare `gpsc` as a global variable in every function that you write. To do this, simply include the statement `global gpsc;` after the function header. If you need to use `gpsc` in the command line, you should also declare it global.

  Python One just needs to import the `gpsc` module to have access to its entries.
- After the initialization step, `findSatellites()` and `decodeSatellites()` (called in this order) take care of the signal processing. We will describe `findSatellites()` in more detail below. `decodeSatellites()` will be the focus of the next assignment.
- At this point, you should read the descriptions of `getData()` and `satCode()` since you will be using these two functions several times throughout this assignment:
  - `getData()`: given the big amount of data we need to process, we will not load it all at once. Instead we will use this function to read a subset of the samples of the received GPS signal stored in `data/01.mat`. To do this, we call `getData()` specifying the initial and final indices of the samples for the fragment we want to read.

[1] `gpsSynchroDecoding_assignment.zip`
[2] Names ending by `()` denote functions, and names ending by `;` denote scripts. The functions in italic font are those that you will write as part of the assignment.

– `satCode()`: returns the C/A code (PRN sequence) for any satellite of the GPS constellation. You should use the option `'fs'` in order to get the C/A sampled at the sampling rate (4 samples per chip).

NOTES

- All the functions that you write must be saved in the `decoding/` directory. In Python, they are located within `my_utilGpsDecoding.py`.
- We provide the solutions for every function that you should write. You can use these solutions to test a whole working system, maybe just substituting one function at a time with your implementation. In MATLAB, the script `function_mapper` is used to select for each function whether to use the provided solution or your own implementation. In Python, you can just import `utilGpsDecoding.py` from the solutions folder.

EXERCISE 1. In this exercise, you will implement the function that produces estimates of ν and τ, using the method seen in class.

Read through `findSatellites()`, whose purpose is to analyze the received GPS signal looking for visible satellites. You will notice that this is a two-pass process, since there are two loops. In the first loop, the received signal is correlated with the C/A code of every satellite in the GPS constellation. The goal here is to find τ and an approximate ν. Only the first N (default $N = 6$) "strongest" satellites are kept for the second loop, where we refine our estimate of ν, by doing inner products (rather than correlations) with multiple repetitions (10) of the C/A codes.

Your task is to code `coarseEstimate()` and `fineEstimate()`, responsible for the first and the second pass, respectively. Read the function headers that we provide and make your implementation accordingly. You should load the minimum amount of data needed.

Hint. Notice that the function $R_a(\hat{\nu}, \hat{\tau})$, as a function of $\hat{\tau}$ with $\hat{\nu}$ fixed, is a correlation. You should know from the previous assignment how to implement that operation.

Hint. For MATLAB, as we have seen in class, if you do `xcorr(y,p)`, where `y` is a vector of data, and `p` is a C/A code at sampling rate (shorter than `y`), then whatever is contained in the first `length(y)-1` positions of the result is part of the ramp-up transient. Nevertheless, it is possible that, due to noise, the maximum of the absolute value will be reached in that part. Hence, you should remove that part before you look for the max. For the same reason, it is safer to remove also the ramp-down transient at the end. In Python, if you use `scipy.signal.corr`, the option `'valid'` takes care of removing both transients. If you use the convolution, (`numpy.convolve` in Python, `conv` in MATLAB), you can set as well the option `'valid'` in order to remove both transients. We advise you to carefully read through the documentation of the above mentioned functions.

When implementing `fineEstimate()`, keep in mind that you do not know where the bit transitions are. Also, you should load the minimum amount of data needed. Implement your function efficiently, otherwise you'll spend a lot of time waiting for `findSatellites()` to complete. If your two codes work correctly, `findSatellites.m` (`mainProduceBits.py` for Python) will produce a summary that looks like this:

```
Sat R     fd       tau
-----------------------
 29  9038 -2730.00  890
  4  8014   810.00 4001
 26  7643  1730.00  280
 21  6941  1190.00 3346
 25  6350 -3600.00 3536
 31  5537 -1800.00 3082

findSatellites: Successful.
```

For MATLAB, do the necessary modifications in `function_mapper.m` to make sure that you are running your functions and not our solution!

The table above is also saved in file `data/foundSat.mat`. This file needs to be read by `decodeSatellites()` (next assignment).