ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
School of Computer and Communication Sciences

Handout 9 Modern Digital Communications
Assignment 6 October 16, 2024 (due Oct 29, 2024)

GPS Signal Decoding

The goal of this assignment is to decode the bits sent by the visible satellites. The code framework is the same as for the previous assignment, available on the course webpage (you do not need to download it again). The list of visible satellites and their parameters (`tau`, `doppler`) are read from the file `data/foundSat.mat`. For each visible satellite, the sequence of bits that you are supposed to obtain can be found in files `data/bitsNN-short.mat`, where `NN` is satellite number, so you can check if your implementation is working as expected.

You can evaluate your code by running `decodeSatellites`. For Python, this is located within `mainProduceBits`. Make sure that the line calling `decodeSatellites` is uncommented.

Exercise 1. Your task is to implement the different functions called by `decodeSatellites()`:

- Implement `findFirstBit()` which finds the sample index corresponding to the beginning of the first complete bit in the received signal. Make your implementation according to the provided header.

  Hint. If you determine that a bit starts at position `tau`, this is not necessarily the beginning of the *first* bit. The beginning of the first bit is `tau` modulo `gpsc.spb`.

- Notice that in `decodeSatellites()` we define the parameter `numberOfBitsPerBlock ... = 5`. This is the number of bits during which we consider that our estimates of τ and ν reliable. For longer bit-sequences we need to update these estimates to track their evolution: `adjustTauAndDoppler()` takes care of this. Once you have read its body and understand its behaviour, your task is to write its two subfunctions `adjustTau()` and `adjustDoppler()`. Make your implementation according to the provided headers.

  Hint. Finding the polynomial coefficients can be done with:

  MATLAB the backslash operator (`help mldivide`)

  Python `numpy.linalg.lstsq()`

At this point we are almost in the PDC scenario (the Doppler and the delay are accounted for), with the difference that the antipodal constellation is rotated by an unknown phase. The functions `doInnerProductsBitByBit()` and `innerProductsToBits()` generate sufficient statistics and decode the bits, respectively.

- Implement `doInnerProductsBitByBit()` according to the provided header.

  Notice that when we call `doInnerProductsBitByBit`, the variable `nBits` is set to `numberOfBitsPerBlock`.

- Implement `innerProductsToBits()`, whose purpose is to obtain hard decisions about the bits, starting from the sufficient statistics computed by `doInnerProductsBitByBit()`.

  Notice that the inner products returned by `doInnerProductsBitByBit()` cannot be used directly to make the hard decisions since we have not corrected the phase ϕ_0. Instead of estimating the phase and correcting it, follow the other approach discussed in class, where we decide that two successive bits take the same value if two consecutive elements of `bitwiseInnerProductResults` have essentially the same phase. If their phase differs essentially by π, we declare that there is a bit change.
  For the first bit, we arbitrarily decide that it is a 1. We will flip all the bits at a later point if necessary. The argument `bitwiseInnerProductResults` passed to the `innerProductsToBits` is the entire vector of inner products, not just the inner products that correspond to `nBits`.