Northeastern University
Department of Electrical and Computer Engineering

April 10, 2018
EECE5698-ST (Spring 2018)
Homework 4

Due date: Friday 20, April 2018. Hand in at class or send a scanned copy to closas@northeastern.edu

To complete the problems, feel free to consult other sources (i.e., books, internet, etc.) besides class notes. Justify your answers!

Problem 1: Generate the Gold codes used in GPS L1 C/A signal. A template function is provided, `CAcodegen.m`, which implements code generation using the shifting of the second MLS, $G_2[n]$, according to GPS L1 C/A standard. The template contains the complete code to generate the first MLS, $G_1[n]$, but has empty spaces which you need to code. Namely,

(a) Generation of $G_2[n]$.

(b) Generation of $G_{2,i}[n]$, for the i-th satellite.

(c) Generation of $G^i[n]$, for the i-th satellite.

Problem 2: To validate your implementation of `CAcodegen.m`, you should identify the auto-/cross-correlation properties of Gold codes in your generated sequences. Create a script `EECE5698_CAcode.m` and copy the following commands:

```
%   EECE5698-ST: GNSS signal processing
%       GPS L1 C/A code generation
%
%   Pau Closas, Spring 2018

clearvars
close all
clc

% satellite(s) ID
svnum = 19;
% C/A chip rate
Rc = 1.023e6;
% chip period
Tc = 1/Rc;

%% generate code(s)
[ca_code]=CAcodegen(svnum);

[ca_code2]=CAcodegen(svnum+1);
```

that will generate two arbitrary codes (in this case 19 and 20, but you can change it). Notice that you should use your `CAcodegen.m` function in the `EECE5698_CAcode.m` script.

(a) Plot those codes over time, verify that they take values in ± 1.

(b) Implement the auto-correlation function of one of the codes, for instance `ca_code`, using 1) linear correlation and 2) circular correlation. Interpret the results.

(c) Compute the auto-correlation of the concatenation of three PRN codes `[ca_code ca_code ca_code]` with the local replica of one code `ca_code`. Comment the results.

(d) Verify the auto-correlation properties for one of the codes you generated, for instance `ca_code`. Use linear and circular correlation, as in (b). Explain the results.

(e) Verify the cross-correlation properties between `ca_code` and `ca_code2`, the codes you generated. Use linear and circular correlation, as in (b). Explain your results.

Hint: in implementing linear and circular correlation, make use of built-in Matlab functions `xcorr`, `fft`, and `ifft`

Problem 3: Load `HW4P3.mat` by typing `load HW4P3` in Matlab shell. That file contains a variable `ca_code_hidden`, a Gold code sequence from the GPS L1 C/A constellation. Determine which satellite's code corresponds to by looping over all possible values (i.e., 32, which is the maximum number of GPS satellites). Explain your answer.

Problem 4: Detect satellites from a real data recording. We will use a similar procedure as in Problem 3 (i.e., looping over possible satellites and correlating with the corresponding spreading code).

Below there is a code implementing non-coherent acquisitions over `Nnci` CAFs. Copy-paste this code into a script and place the file `realGPSL1capture.bin` in the same folder, that file contains a real data recording of L1 band signal ($f_s = 10$ MHz). You need also `computeCAF.m` and `ResampleCode.m` in the folder.

```
% signal parameters
fs = 10e6;           % Sampling frequency
fIF = 0;             % Intermediate frequency
fc = 1.023e6;        % Code rate of the code [Hz]
Tcoh = 0.001;        % Coherent integration time [s]
Nc = Tcoh * fs;      % number of samples contained in the coherent integration time

% Acquisition parameters
Nd = 81;             % Number of Doppler bins
DopStep = 125;       % Doppler bin size in Hz
secondOfData = 0.1;  % Seconds of data to read
Nnci = 1;            % number of non-coherent integrations of the CAF
SVIDs = 1:32;           % satellite vehicles (SV) to detect [7 16 19 21 22 25]

% read file with the IF capture
fid = fopen ('realGPSL1capture.bin','r');
[data, cnt_data] = fread(fid, 2 * secondOfData * fs, 'int8');
data = data(1:2:end) + 1i * data(2:2:end);

CAF_aux = zeros(Nd,Nc);
DopplerEst = -ones(1,32);
DelayEst = -ones(1,32);

% generate local replica and resample from Tc to Ts>Tc

% loop over all possible satellites
for svnum = SVIDs
```

```
    [ca_code]=CAcodegen(svnum);
    % Resample the code at data sampling frequency
    ca_code_resampled = ResampleCode( ca_code, Nc, fs, 0, fc );

    CAF = 0;          % initialized CAF to zero every time

    % loop to average over noncoherent integrations
    for ii = 1:Nnci
        y = data( (ii - 1) * Nc + (1:Nc) ).';   % use just 1 period of code at the time

        % loop over frequency bins
        for ff=1:Nd
            fdbin = fIF/fs + (ff - ceil(Nd/2))*DopStep/fs;   % normalized Doppler bin
            CAF_aux(ff,:) = computeCAF(y, ca_code_resampled, fdbin);
        end
        % integrate non-coherently the ii test statistics |CAF|^2
        CAF = CAF + abs(CAF_aux).^2;

        % plot 2D grid search
        CAF_normalized = CAF/max(max(CAF));      % normalize to 1 the maximum value

        figure(svnum)
        surf((0:(Nc - 1)) / fs, ((1:Nd) - ceil(Nd/2))*DopStep, CAF_normalized, 'EdgeColor', 'none');
        axis tight, set( gca, 'FontSize', 16 )
        xlabel('Code delay [s]'), ylabel('Doppler [Hz]')
        title(['SV ' num2str(svnum) ' , ' num2str(ii) ' non-coherent integrations'])
%         pause
    end
    pause

    % estimate Doppler (if satellite detected)
    [~, DopInd] = max(max(CAF.'));
    DopplerEst(svnum) = fIF + (DopInd - ceil(Nd/2))*DopStep;

    % estimate time-delay (if satellite detected)
    [~, codInd] = max(max(CAF));
    DelayEst(svnum) =  (codInd - 1) / fs;

end
```

Notice that you would need to generate the codes with `CAcodegen.m`. If you were not able to work that function out, you might load `CAcodes.mat`, which contains a matrix with the Gold code of each satellite (total 32) in the corresponding row (that is, use `[ca_code]=ca_code_matrix(svnum,:);` instead).

(a) Go through the code and explain (preferably supporting your discussion with math) the different steps implemented.

(b) Run the code with `Nnci=1`. Explain the processing that takes place in the receiver in this configuration. How many satellites are you able to clearly detect (i.e., distinguish from noise floor)?

(c) Gradually increase `Nnci` from 1 to 10 non-coherent integrations. Explain the processing that takes place in the receiver in this configuration. How many satellites are you able to clearly detect (i.e., distinguish from noise floor)? Explain the results.

(d) At the light of the results, which satellites are more likely to be present in the capture? Once a satellite is acquired, which information is passed on to the tracking loops?