ASE 389P-7
Exam 1

Alejandro Moreno

October 4, 2022

1 Problem 1

1.1 Instruction

Write a function in Matlab that simulates the train-horn-Doppler scenario discussed in lecture. Assume that the train tracks are rectilinear.

Be sure to account for the nonzero time of flight δt_{TOF} as discussed in lecture. The effect of $\delta t_{TOF} > 0$ is that the stationary observer will discern an f_D at time t_k that relates to the train's line-of-sight velocity at time $t_k - \delta t_{TOF}$. More precisely, the apparent frequency of the train horn at the location of the observer at time t_k is given by

$$f_r(t_k) = \frac{f_c}{1 + \frac{v_{los}(t_k)}{v_s}} \tag{1}$$

where f_c is the nominal horn frequency, $v_{los}(t_k)$ is the line-of-sight velocity at t_k, and v_s is the speed of the signal in the medium. Note that the line-of-sight geometry used to calculate $v_{los}(t_k)$ is between the observer at time t_k and the horn at time $t_k - \delta t_{TOF}$.

Download the audio file trainout.wav from Canvas. This file was created with the following input argument values:

```
fh = 440;
vTrain = 20;
t0 = 0;
x0 = 0;
delt = 0.01;
N = 1000;
vs = 343;
```

Set up your simulator with these same values. Estimate the values of xObs and dObs by adjusting them in your simulation until you get an apparent received frequency profile that matches the one in the audio file.

1.2 MATLAB code

```
function [fDVec,tVec] = ...
simulateTrainDoppler(fc, vTrain, t0, x0, xObs, dObs, delt, N, vs)
% simulateTrainDoppler : Simulate the train horn Doppler shift scenario.
%
% INPUTS
%
% fc ------- train horn frequency, in Hz
%
% vTrain --- constant along-track train speed, in m/s
%
% t0 ------- time at which train passed the along-track coordinate x0, in
%            seconds
%
% x0 ------- scalar along-track coordinate of train at time t0, in meters
%
% xObs ----- scalar along-track coordinate of observer, in meters
%
% dObs ----- scalar cross-track coordinate of observer, in meters (i.e.,
%            shortest distance of observer from tracks)
%
% delt ----- measurement interval, in seconds
%
% N -------- number of measurements
%
% vs ------- speed of sound, in m/s
%
%
% OUTPUTS
%
% fDVec ---- N-by-1 vector of apparent Doppler frequency shift measurements as
%            sensed by observer at the time points in tVec
%
% tVec ----- N-by-1 vector of time points starting at t0 and spaced by delt
%            corresponding to the measurements in fDVec
%
%+---------------------------------------------------------------------------+
% References:
%
%
% Author:
%+===========================================================================+
    tVec = zeros(N,1);
    fDVec = zeros(N,1);

    t = t0;
```

```
    for h = 1:N
        % Find rRX(t) and vRX(t)
        rRX = positionRX(t, t0, xObs, dObs);
        vRX = velocityRX(t, t0);

        % Calculate TOF
        TOF = calculateTOF(t, rRX, t0, x0, vTrain, vs);

        % Find rTX(t-TOF) and vTX(t-TOF)
        rTX = positionTX(t-TOF, t0, x0, vTrain);
        vTX = velocityTX(t-TOF, t0, vTrain);

        % Find the line-of-sight
        los = (rRX - rTX) / norm(rRX - rTX);

        % Project the velocity of TX to the line-of-sight
        vTX_los =  dot(vTX, los);
        % Project the velocity of RX to the line-of-sight
        vRX_los =  dot(vRX, los);
        % Calculate the line-of-sight velocity
        v_los = vRX_los - vTX_los;

        % Calculate the apparent Doppler frequency shift as sensed by the
        % observer
        beta = v_los / vs;
        fr = fc / (1 + beta);
        fDVec(h) = fr - fc;
        tVec(h) = t;
        t = t + delt;
    end
end

function [rRX] = positionRX(t, t0, x0, y0)
    rRX = [x0, y0];
end

function [vRX] = velocityRX(t, t0)
    vRX = [0, 0];
end

function [rTX] = positionTX(t, t0, x0, v0)
    x = x0 + v0*(t-t0);
    rTX = [x, 0];
end
```

```
function [vTX] = velocityTX(t, t0, v0)
    vTX = [v0, 0];
end

function [TOF] = calculateTOF(t, rRX, t0, x0, vTrain, vs)
    % Initialize with a guess
    TOF = norm(rRX - positionTX(t, t0, x0, vTrain))/vs;
    error = vs*TOF - norm(rRX - positionTX(t-TOF, t0, x0, vTrain));

    % Iterate until convergence
    while ( abs(error) > 1e-3 )
        TOF = TOF - error/vs;
        error = vs*TOF - norm(rRX - positionTX(t-TOF, t0, x0, vTrain));
    end
end

% topSimulateTrainDoppler.m
%
% Top-level script for train Doppler simulation


clear; clc; close all;
%---- Setup
fc = 440;
vTrain = 20;
t0 = 0;
x0 = 0;
delt = 0.01;
N = 1000;
vs = 343;
xObs = 56.8;
dObs = 10;

%---- Simulate
[fDVec,tVec] = simulateTrainDoppler(fc,vTrain,t0,x0,xObs,dObs,delt,N,vs);
fApparentVec = fDVec + fc;

%---- Plot
plot(tVec,fDVec + fc, 'r');
xlabel('Time (seconds)');
ylabel('Apparent horn frequency (Hz)');
grid on;
shg;

%---- Generate a sound vector
T = delt*N;                    % simulation time (sec)
```

```
fs = 22050;                         % sample frequency (Hz)
deltSamp = 1/fs;                    % sampling interval (sec)
Ns = floor(T/deltSamp);             % number of samples
tsamphist = [0:Ns-1]'*deltSamp;
Phihist = zeros(Ns,1);
fApparentVecInterp = interp1(tVec,fApparentVec,tsamphist,'spline');
for ii=2:Ns
   fii = fApparentVecInterp(ii);
   Phihist(ii) = Phihist(ii-1) + 2*pi*fii*deltSamp;
end
soundVec = sin(Phihist);

%%------ Play the sound vector
%sound(soundVec, fs);

%------ Write to audio file
audiowrite('my_trainout.wav',soundVec,fs);

%------ Write frequency time history to output file
save trainData fApparentVec tVec

%% Extra Points
close all
% I use an spectrogram to better understand the original audio signal
[y, fs] = audioread('trainout.wav');

Nspec = 2^(nextpow2(length(y)) - 7);
wspec = hamming(Nspec);
Noverlap = Nspec/2;
figure()
% spectrogram(y, 512, 128, 128, fs, 'yaxis');
spectrogram(y, wspec, Noverlap, Nspec, fs, 'yaxis');
colormap jet
title("original signal")

set(gca,'Linewidth',1.2,'FontSize',36)
set(gcf,'Position',[2500 100 1550 800])

% Then, I manually analyze my simulated audio signal to make it look as
% similar to the original as posible by changing the position of the
% receiver
% [y, fs] = audioread('my_trainout.wav');
%
% figure()
% spectrogram(y, 512, 120, 128, fs, 'yaxis');
```

```
% colormap jet
% title("My signal")
%
% set(gca,'Linewidth',1.2,'FontSize',36)
% set(gcf,'Position',[2500 100 1550 800])
```

1.3 Result

Analizing the given signal it's possible to manually tag the moment in which the train passed in front of the receiver. Figure 1 shows how the power density is concentrated at higher frequency before 3.069 sec and drops rapidly after that. This indicates the exact moment at which the train passed infront of the receiver. Note that the frequency resolution of this spectrogram is not good but the time resolution is pretty good. This allows for better visual tuning.

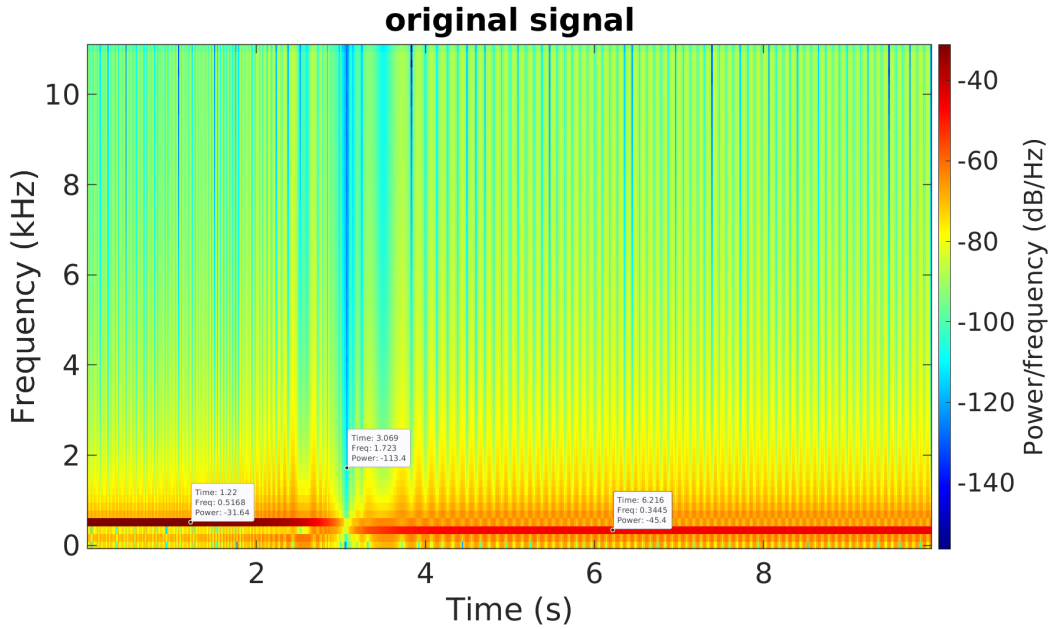

Figure 1: Spectrogram of the given signal.

This, approach lead to $xObs = 56.8$ and $dObs = 10$.

Another way to get better results would be to iteratively change the values of xObs and dObs in a neighborhood of our previous guess and try to verify if the spectrogram of our guess is equal to the spectrogram of the original signal. A very rudimentary way to do it would be to just substract the matrices that store the spectrogram and look for the iteration that minimizes that substraction maybe just using the max norm ($||A||_{max}$). Figure 2 and 3 show the type of spectrogram I would use in the aforementioned algorithm because it's has much better frequency resolution.

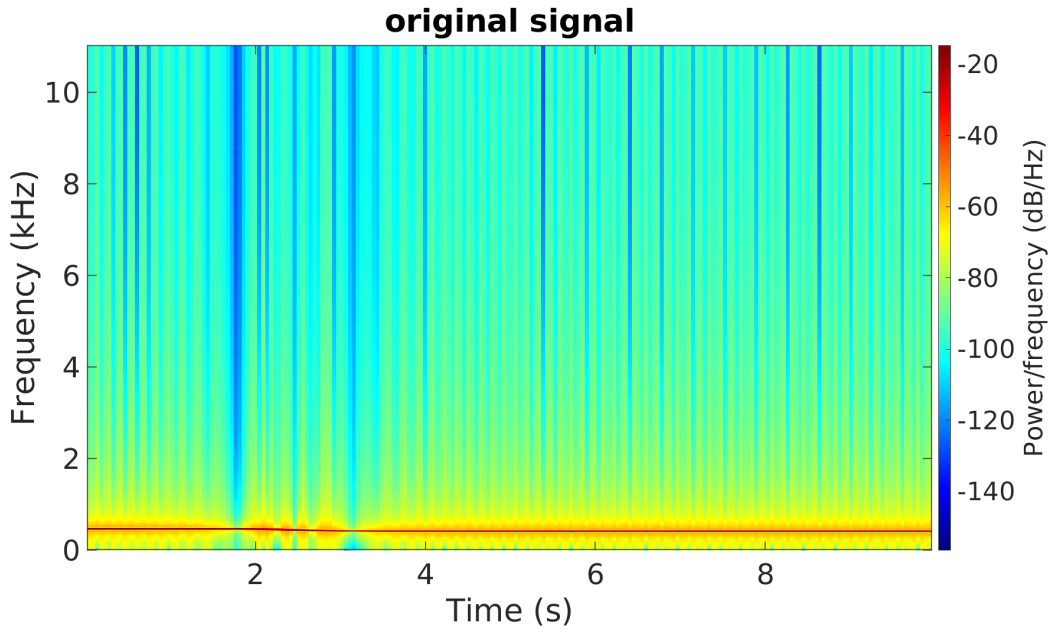

Figure 2: Spectrogram of the given signal.

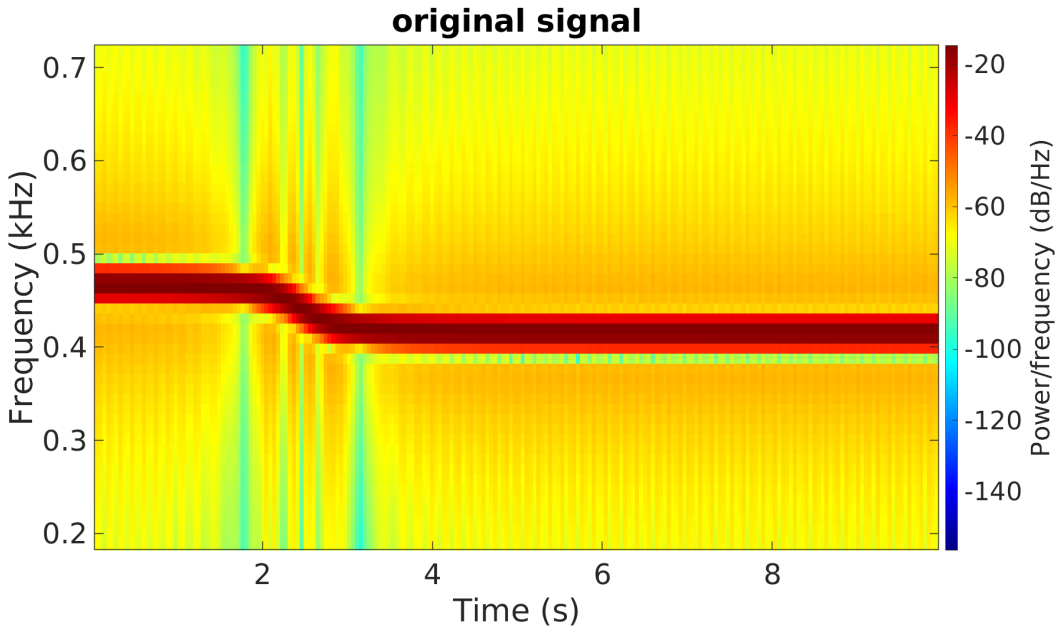

Figure 3: Zoomed spectrogram of the given signal.

2 Exercise 2

2.1 Instruction

In this problem, we'll examine the effects of clock frequency bias on Doppler derived from frequency downconversion and sampling.

Usually, we don't measure signal properties directly at the signal's incoming frequency. Instead, we convert incoming signals to a lower center frequency by a process called frequency conversion. To understand frequency conversion, recall that

$$cos(x)cos(y) = \frac{1}{2}[cos(x-y) + cos(x+y)] \tag{2}$$

Hence, by multiplying an incoming signal $x(t) = cos(2\pi f_c t)$ by a local signal $x_l(t) = 2cos(2\pi f_l t)$ and then low-pass filtering the result to eliminate the high frequency $f_c + f_l$ component, we convert the high frequency signal down to a lower frequency $f_b = f_c - f_l$. Processes such as amplification, transmission, filtering, delaying, recording, and sampling are easier to do at lower frequencies. Figure 4 shows a diagram of the frequency conversion process.

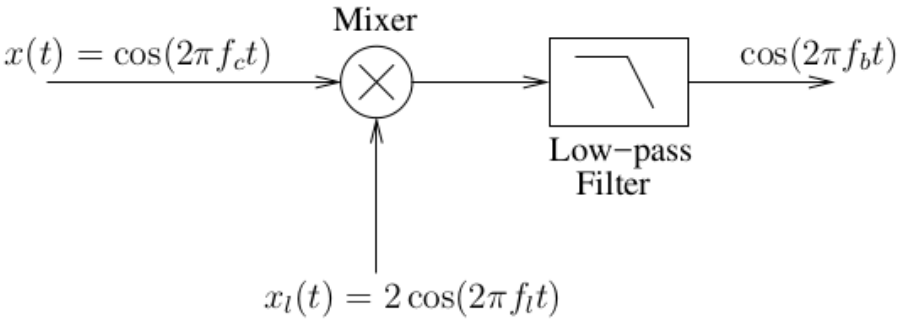

Figure 4: A single-stage frequency conversion (mixing) operation.

Assume that some oscillator with a perfect clock produces a pure sinusoid of the form $x(t) = cos(2\pi f_c t)$. Suppose that you receive this signal and make a measurement f_m of the signal's frequency by (1) performing frequency conversion to translate the signal to a nominal center frequency $f_{b,nom} = 0Hz$ and then (2) sampling the signal with nominal sampling interval Δt and observing the number of samples per period. Assume there is no motion between your receiver and the signal transmitter. Suppose that the local clock you're using to perform frequency conversion and sampling has a fractional frequency error of $\Delta f/f < 0$.

Derive an expression for the apparent Doppler f_D resulting from frequency conversion and sampling with a clock that has this fractional frequency error. Express your result in terms of f_c and $\Delta f/f$. How does this expression differ from the one derived in problem 3?

Note that if $f_l < f_c$, as will be the case for our problem because $\Delta f/f < 0$, then $f_b > 0$. This is an example of low-side mixing. But if $f_l > f_c$, then $f_b < 0$; this is high-side mixing. What would be the meaning of a signal for which $f_b < 0$? Upon sampling a signal with $f_b < 0$ to determine the period, would we measure a negative frequency?

2.2 Result

$$x(t) * x_l(t) = 2cos(2\pi f_c t)cos(2\pi f_l t)) \tag{3}$$

$$x(t) * x_l(t) = cos(2\pi(f_c - f_l)t) + cos(2\pi(f_c + f_l)t) \tag{4}$$

After applying the low-pass filter we get

$$x_{LPF}(t) = cos(2\pi(f_c - f_l)t) \tag{5}$$

Let's clarify:

$$f_b = f_c - f_l = f_c - (f_{l,nom} + \Delta f) \tag{6}$$

We know that $f_{b,nom} = 0Hz$. Therefore, $f_{l,nom} = f_c$.

Now, it's necessary to focus on the last part of the processing (sampling). For that, lets write the following relationships.

$$\frac{\Delta f}{f_{l,nom}} = \frac{f_l - f_{l,nom}}{f_{l,nom}} = \frac{f_l}{f_{l,nom}} - 1 = \frac{\Delta t_{l,nom}}{\Delta t_l} - 1 \tag{7}$$

$$\Delta t_{l,nom} = (\frac{\Delta f}{f_{l,nom}} + 1)\Delta t_l \tag{8}$$

Since the objective at this stage is to measure the frequency of the signal that comes from the low-pass filter, it is safe to say that we are trying to measure f_b. Thus, $f_b = \frac{1}{N*\Delta t_l}$.

$$f_m = \frac{1}{N * \Delta t_{l,nom}} = \frac{1}{N * (\frac{\Delta f}{f_{l,nom}} + 1)\Delta t_l} = \frac{f_b}{\frac{\Delta f}{f_{l,nom}} + 1} \tag{9}$$

Since it was stablished that $f_{l,nom} = f_c$. Then, $\Delta f = f_c \frac{\Delta f}{f_{l,nom}}$

$$f_m = \frac{\frac{\Delta f}{f_{l,nom}} f_c}{\frac{\Delta f}{f_{l,nom}} + 1} \tag{10}$$

Finally, it is possible to reason that the measured frequency is the very apparent Doppler f_D (with a changed sign). This is because the frequency conversion already applied the following substraction $f_b = f_c - f_l$, which is almost the same substraction needed for $f_D = f_l - f_c$.

$$f_D = -\frac{\frac{\Delta f}{f_{l,nom}} f_c}{\frac{\Delta f}{f_{l,nom}} + 1} \tag{11}$$

2.3 Q&A

Q: How does the expression differ from the one derived in problem 3? A: It doesn't. It's the same relationship.

Q: What would be the meaning of a signal for which $f_b < 0$? A: It would mean that the apparent Doppler would be inverted. Therefore, we would think that the TX and RX are getting further and further appart.

Q: Upon sampling a signal with $f_b < 0$, would we measure a negative frequency? A: No, we would measure the absolute value of f_b.

3 Exercise 3

3.1 Instruction

This problem will walk you through a noise analysis for the UT Radionavigation Laboratory (RNL) GNSS receiver setup. A block diagram of the setup is shown in Fig. 2. A rooftop Trimble Geodetic Zephyr II antenna is followed by cables, a bias tee, and a splitter, with parameters as follows:

- $T_A = 100K$: The temperature of the passive antenna element within the Trimble antenna.
- $L_1 = 1$ dB: Before being amplified, signals from the passive antenna element pass through a short transmission line and a low-insertion-loss bandpass filter. L_1 is the combined loss of the line and filter. Because L_1 enters before any gain is applied to the signals, it is very important to make L_1 as low as possible.
- $G_2 = 50$ dB; $F_2 = 1.5$ dB: The Trimble antenna's built-in low-noise amplifier (LNA) has extraordinary gain and a low noise figure.
- $L_3 = 6.48$ dB: Signals are routed from the active Trimble antenna to the RNL via 127 feet of C400 cable. The cable is fairly low loss: at 1.5 GHz (near the GPS L1 frequency), the cable loss is 5.1 dB per 100 feet.
- $L_4 = 0.6$ dB: A bias tee feeds the active antenna with a 5 V supply by biasing the center line of the coaxial cable coming from the antenna to 5 V DC. The DC supply voltage does not affect the high-frequency signals arriving from the antenna. From the perspective of the high-frequency signals, the bias tee looks like a passive component with a 0.6 dB insertion loss.
- $L_5 = 9.8$ dB: Signals exiting the bias tee are split by a passive 8-way splitter to provide GNSS signals for the many receivers in the RNL. For an ideal splitter, the signal power exiting any one of the 8 splitter outputs is a factor of 8 (9 dB) lower than the input power. In the RNL splitter, the input signal sees an additional 0.8 dB of insertion loss.

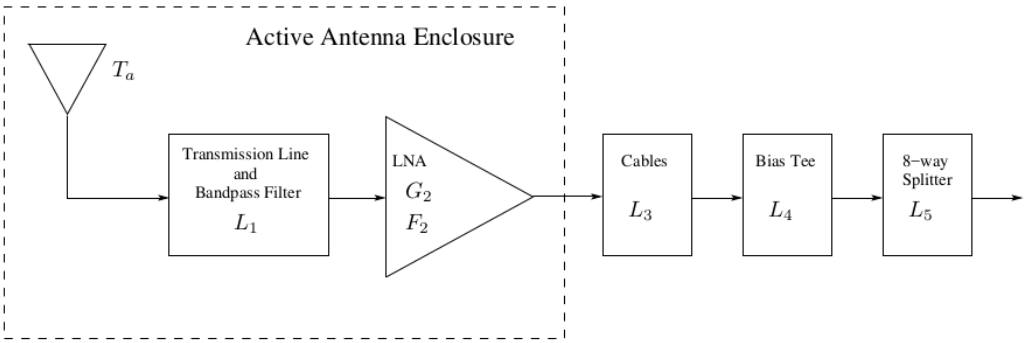

Figure 5: Radionavigation Laboratory GNSS receiver setup.

3.2 Results

3.2.1 Item a

Calculate the noise figure F_1 and effective input temperature T_1 corresponding to L_1. Assume an input temperature $T_{in} = T_0 = 290K$.

Since L_1 is a passive element

$$F = \frac{1}{G} > 1 \tag{12}$$

$$T_E = \frac{1-G}{G} T_0 \tag{13}$$

Therefore, $F_1 = 1.259$ and $T_1 = 75.11K$.

3.2.2 Item b

Calculate the effective input temperature T_2 corresponding to F_2. Assume F_2 is taken from a device data sheet that assumes $T_{in} = T_0 = 290K$.

$$T_2 = (F_2 - 1)T_0 \tag{14}$$

Therefore, since $F_2 = 1.5dB = 1.413$ the effective input temperature $T_1 = 119.64K$.

3.2.3 Item c

Lump losses L_3, L_4, and L_5 into a single loss L_{345}. Calculate an effective input temperature T_{345} corresponding to L_{345}. Assume an input temperature $T_{in} = T_0 = 290K$.

Since, $L_3 = 6.48dB$, $L_4 = 0.6dB$, and $L_5 = 9.8dB$ the lump losses are $L_{345} = 16.88dB$. The equivalent system is passive. Thus, $F = L = 48.75$.

$$T_{345} = (F_{345} - 1)T_0 \tag{15}$$

Consequently, $T_{345} = 13,848.3K$

3.2.4 Item d

Use the Friis formula to calculate the system temperature $T_S = T_A + T_R$, expressed in degrees K. How many degrees K do the combined losses L_{345} contribute to the system temperature?

$$T_R = T_1 + \frac{T_2}{G_1} + \frac{T_{345}}{G_1 G_2} \tag{16}$$

$$T_R = 75.11 + \frac{119.64}{0.794} + \frac{13,848.3}{0.794 * 100000} = 75.11 + 150.68 + 0.17 = 293.28K \tag{17}$$

Therefore, $T_S = T_A + T_R = 393.28K$. Notice that the combined losses L_{345} contribute $0.17K$ to the overall system temperature.

3.2.5 Item e

The effective noise floor of the whole cascade, N_0, is related to the system temperature T_S by $N_0 = k * T_S$, where k is Boltzmann's constant, equal to $-228.6dBW/K - Hz$. The quantity N_0 is the one used in calculating C/N_0, that all-important parameter in GNSS receiver design. For purposes of calculating C/N_0, you can think of the receiver cascade as a chain of ideal gain blocks (no internal noise) with an effective noise density N_0 at the beginning of the cascade (i.e., at the output of the passive antenna element just before the L_1 block). This is precisly the point where the signal power C is defined, so it makes sense to calculate the ratio C/N_0 here. Calculate the value of N_0 in dBW/Hz. For an expected GPS L_1 C/A received signal power C ranging from -162.5 to $-154.5dBW$, calculate the expected range of carrier-to-noise ratio C/N_0 values. Express your result in $dB - Hz$.

If one thinks the receiver cascade as a chain of ideal gain blocks (no internal noise). Then $T_S = T_A = 20dBK$

$$N_0 = k * T_S = -228.6\frac{dBW}{K - Hz} + 20dBK = -208.6\frac{dBW}{Hz} \tag{18}$$

$$46.1dB - Hz = -162.5 - (-208.6) < \frac{C}{N_0} < -154.5 - (-208.6) = 54.1dB - Hz \tag{19}$$

3.2.6 Item f

What is the noise floor $N_{0,sp}$ at the output of the 8-way splitter? Express your answer in dBW/Hz.

Since $T_S = 393.28K = 25.947dBK$

$$N_0 = k * T_S = -228.6\frac{dBW}{K - Hz} + 25.947dBK = -202.65\frac{dBW}{Hz} \tag{20}$$

4 Problem 4

4.1 Instruction

To measure the receiver temperature T_R of a GNSS receiver and antenna setup, a friend recommends placing the receiver's antenna in a RF test enclosure such as the one in the Radionavigation Laboratory (seen here https://ramseytest.com/forensic-test-enclosures), but cryogenically cooled down to 5 K, and then measuring the noise power in the raw samples generated by the receiver. The enclosure effectively isolates the antenna from environmental noise. The antenna is an active antenna consisting of a patch element, a (passive) filter, and an amplifier. Is this a valid approach for measuring TR ? Why or why not?

4.2 Result

The cryogenically cooled enclosure would only allow us to reduce the noise of the receiver but not T_A. Therefore one could measure the noise power with and without the cryogenically cooled enclosure and get an estimate of the T_R by substracting the results.

It seems like this is not a good idea though because if one would like to make the best use of their money for the general usage of the GNSS receiver. Then, buying the cryogenically cooled

enclosure would only buy you a couple dBs better noise performance. The main source of noise (the other GNSS signals one is not interested in when aquiring from a particular satellite) would not be attenuated. Therefore, one could measure the desired magnitude but that's it. No noticible benefit would come from this purchase.

5 Problem 5

5.1 Instruction

The null-to-null bandwidth B_1 of the main lobe of the power spectrum $S_X(f) = T_c sinc^2(fT_c)$ of a binary random process with chip interval T_c is $2/T_c$. So, for example, the GPS L1 C/A code, for which $T_c \approx 1us$, has $B_1 \approx 2MHz$. Because of the rounded shape of the $sinc^2(f)$ function, the spectrum within B_1 is not filled uniformly with power—in fact, there is very little power at the edges of the B_1 interval where, as it turns out, power matters most for accurate code phase (i.e., pseudorange) measurements.

Let's consider a hypothetical radionavigation system with a spreading waveform that makes more efficient use of the spectrum within B_1. Let the (equivalent baseband) power spectrum of the hypothetical system's spreading waveform be given by

$$S_X(f) = \frac{1}{W}\Pi(f/W) \tag{21}$$

This waveform has a two-sided bandwidth of W Hz.

5.2 MATLAB code

```
%% Problem 2
% GPS L1 C/A code has a Tc ~ 1us
% ==> null-to-null bandwidth of the main lobe of the power spectrum (B1)
% B1 = 2/Tc ~ 2 MHz
%
% Note that the spectrum of Sx within B1 is not filled uniformly with power.
%-------------------------------------------------------------------------
clc; close all; clear all
format short
% Since in class we use a different convention for the fourier transform
% than the one that MATLAB uses by default. Then, I change it right away
oldVal = sympref('FourierParameters',[1 -(2*sym(pi))]);

Tc = 1e-6; W = 2e6;
syms f f1 tau tau1
Sx = Tc*(sinc(f1*Tc))^2;
Sx_efficient = rectangularPulse(f/W)/W;

figure()
fplot(Sx,[-2/Tc, 2/Tc], Color='k')
hold on
```

```
fplot(Sx_efficient,[-2/Tc, 2/Tc], Color='b')
title('Power spectral density')
xlabel('f')
legend('Sx','Sx_{efficient}')
grid on

% a) Find the autocorrelation function Rx(tau) of the spreading waveform by
% taking the inverse transform of Sx(f)
Rx_efficient = ifourier(Sx_efficient,f,tau);

figure()
fplot(Rx_efficient, [-2*Tc, 2*Tc], Color='b')
title('Autocorrelation of Rx_{efficient}')
xlabel('tau')
grid on

% b) How wide is the main peak in the autocorrelation function Rx (from
% first left to first right zero-crossing)?
right_zero_crossing = vpa(solve(Rx_efficient));
main_peak_width = 2*right_zero_crossing % since the function is symmetric

% c) Compare this width to the width of the peak of the autocorrelation
% function Rx_bar for a random binary sequence with a null-to-null
% bandwidth B1 = W
Rx_bar = triangularPulse(-Tc, Tc, tau1); % from class notes

figure()
fplot(Rx_bar,[-2*Tc, 2*Tc], Color='k')
hold on
fplot(Rx_efficient, [-2*Tc, 2*Tc], Color='b')
title('Autocorrelation')
xlabel('tau')
legend('Rx','Rx_{efficient}')
grid on

main_peak_width_bar = 2*Tc

% d) The following ratio shows that the width of the main peak of the
% autocorrelation function of the more efficient waveform (in terms of
% noise) is more than 3 times wider than the original random binary
% sequence. This means that one would loose precision.
%
% My logic is the following, one would want the autocorrelation to be a
% dirac delta. Then, the achieved precision would be excellent. As far as
% one gets from this ideal situation the worse the precision gets.
```

```
ratio = main_peak_width / main_peak_width_bar
```

5.3 Results

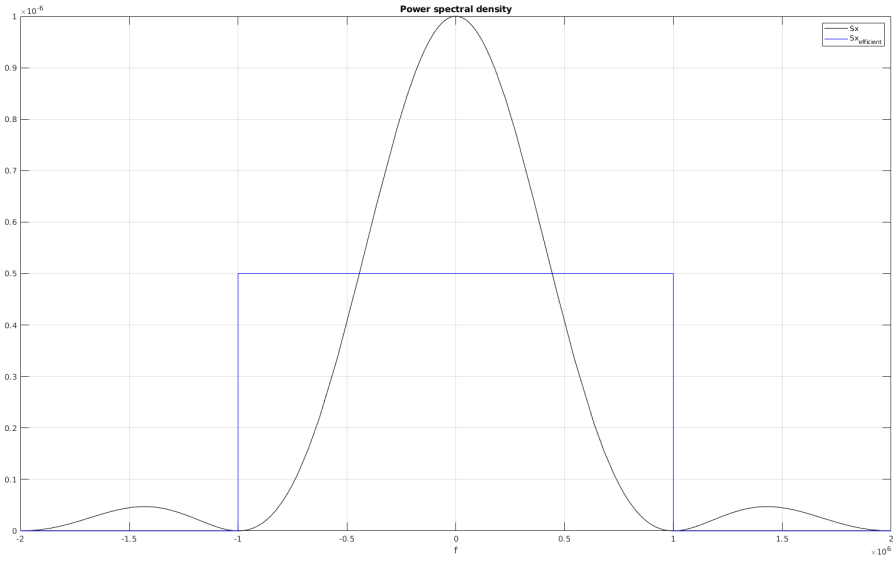

Figure 6: Power spectral density of X and X_{eff}.

5.3.1 Item a, b and c

(a) Find the autocorrelation function $R_X(\tau)$ of the spreading waveform by taking the inverse Fourier transform of $S_X(f)$.

(b) How wide is the main peak in the autocorrelation function $R_X(\tau)$ (from first left to first right zero-crossing)?

(c) Compare this width to the width of the peak of the autocorrelation function $R_X(\tau)$ for a random binary sequence with a null-to-null bandwidth $B_1 = W$

Figure 7 shows the autocorrelation function of the efficient X and the the one corresponding to a random binary sequence with null-to-null bandwidth $B_1 = W$.

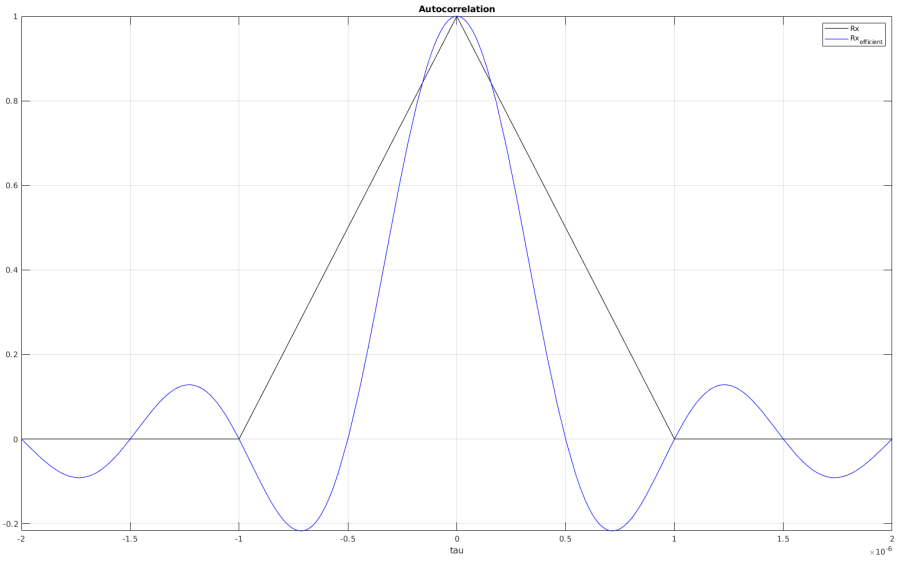

Figure 7: Autocorrelation of X and X_{eff}.

It can be seen from the graph that the main peak in the autocorrelation function $R_{Xeff}(\tau)$ is equal to T_c. On the other hand, the main peak in the autocorrelation function $R_X(\tau)$ is equal to $2 * T_c$. Thus, the width of the main peak of the autocorrelation of X is twice as wide than X_{eff}.

5.3.2 Item d

Which of the two autocorrelation functions would lend itself to a more accurate determination of $\tau peak$, the location of the autocorrelation peak, based on a single pair of autocorrelation function samples spaced by $\Delta\tau = 0.2/W seconds$ in a high-C/N0 scenario? Does you answer change for a low C/N0 scenario? Assume a receiver with a wide front-end bandwidth of 10 W Hz so that there is not any significant rounding of the triangular autocorrelation function's peak. Explain.

The wider your signal is in the time-domain, the narrower it is in the frequency domain. The narrower the frequency domain, the further away from a Dirac delta the autocorrelation gets (a Dirac delta is how we would like the autocorrelation to look like). However, the wider the signal in the time domain the better would be the receiver's ability to understand the signal under low C/N0 situations.

Therefore, a tradeoff exists and there's no right answer. I would say that in a low-C/N0 scenario one should try to slow the sequence down as much as possible to detect it because detecting it and getting low accuracy in the range measurement is better than not detecting at all. In a high-C/N0 I would by greedy with respect to accuracy and make the spreading code as fast as possible (wide as possible in freq domain).

5.3.3 Item e

Why do you suppose the random binary sequence (actually a pseudorandom approximation of it) was historically used for GPS?

The pseudo-random binary sequence was chosen because of its built-in ranging capabilities, multiple access characteristic and jamming rejection. The reason why it needs to be pseudo-random is because without knowing the actual sequence there's no way to actually "lock" to the spreading code (no way to have a local replica).

6 Problem 6

6.1 Instruction

If C(t) is a random binary ($\pm$ 1) spreading code with rectangular pulses, then, as we showed in lecture, the power spectral density of C(t) is given by

$$S_C(f) = T_C sinc^2(fT_C) \tag{22}$$

where T_C is the chipping interval.
Suppose instead that C(t) has psd

$$S_C(f) = T_C \Pi(fT_C) \tag{23}$$

where $\Pi(f)$ is the rect function introduced in lecture. Imagine a constellation of GNSS satellites broadcasting signals having this new $S_C(f)$. Assume the signals have no data bit modulation, so that the signal's baseband representation is of the form

$$r(t) = \sqrt{P}C(t)\exp[j\theta(t)] \tag{24}$$

For this case, calculate $I_0 = S_I(0)$ for a single multiple-access interference signal with power P_I, where I_0 and $S_I(f)$, the power spectrum of $I(t)$, were defined in lecture. Now consider M multiple-access signals (including the desired signal). If each of the M received signals has power P_I, at what value of M does I_0 exceed N_0 ? Express your answer in terms of N_0, T_C and P_I.

6.2 MATLAB code

```
%% Problem 4
% Suppose that C(t) has a power spectral density given by:
%
%                    Sc(f) = Tc * rectangularPulse(f*Tc)
%
% Where 'rectangularPulse' is the rect function introduced in lecture.
% Imagine a constellation of GNSS satellites broadcasting signals having
% this new Sc(f). Assume the signals have no data bit modulation, so that
% the signal's baseband representation is of the form:
%
%                    r(t) = sqrt(P)*C(t)*exp[j theta(t)]
%
clc, close all, clear all

% For this case, calculate I0 = SI(f=0) for a single multiple-access
```

```
% interference signal with power PI, where I0 and SI(f), the power spectrum
% of I(t), were defined in class.
%
% ANSWER:
% Following the same reasoning as in lecture:
% If CI(t) is a random-binary sequence "like C(t)" (same Tc), and if the
% two are uncorrelated. Then, (assuming fD=0)
%
%                     SrI(f) = PI*Sc(f)
% Thus,
%                     SI(f) = Sc(f) x SrI(f) = Sc(f) x (PI*Sc(f))
%
%                     SI(f) = PI * integral_{-inf}^{inf}(Sc(f-x)Sc(x) dx
%
% Since, Sc(f) = Sc(-f)
%
%                     SI(f) = PI * integral_{-inf}^{inf}(Sc(x-f)Sc(x) dx
%
% Now, using the fact that the receiver has a low pass filter, only the f=0
% part of the spectral density leaks through. Therefore, it's ok to assume
% I0 = SI(f=0) (REMEMBER: I0 is also a power density)
%
%     SI(0) = PI * integral_{-inf}^{inf}(Sc(x)^2) dx
%
%     SI(0) = PI * integral_{-inf}^{inf}(Tc*rectangularPulse(x*Tc)^2) dx
%
%     SI(0) = PI*Tc
I0 = PI*Tc;
I0_dBW_Hz = 10*log10(I0)

% Now consider M multiple-access signals (including the desired signal). If
% each of the M received signals has power PI, at what value of M does I0
% exceed N0? Express your answer in terms of N0, Tc, PI.
%
% ANSWER:
% The multiple-access interference is characterized by:
%
%            I0 = PI*Tc*(M-1)
%
% We want to know what M would result in I0 >= N0
%
%            PI*Tc*(M-1) >= N0
%            M >= N0/(PI*Tc) + 1, where M is integer
%
% => I0 exceeds N0 when:
```

```
%               M = ceil(N0/(PI*Tc) + 1)
%
```

7 Problem 7

In this problem we'll further analyze the data taken with the Stanford 46-meter diameter dish. The gain of the Stanford dish is so great (about 50 dB at the GPS L1 frequency) that you can actually see the transitions induced by the GPS L1 C/A code and by the GPS L1 P(Y) code. Recall that in lecture the GPS L1 signal structure was presented as

$$S_{L1}(t) = \sqrt{2P_{C1}}D(t)C(t)cos(2\pi f_{L1}t + \theta_{L1}) + \sqrt{2P_{Y1}}D(t)Y(t)sin(2\pi f_{L1}t + \theta_{L1}) \quad (25)$$

You can see from this that the GPS C/A code (spreading code C(t), chipping at 1.023 MHz) is in phase quadrature with the GPS Y code (spreading code Y (t), chipping at 10.23 MHz).

Load the "Big Dish" data into your workspace in Matlab (you can use the bigDishPSD.m script to do this). The sampling rate for these data is $fs = 46.08MHz$. The data are loaded into the Matlab complex variable Y. Each element in Y represents a complex sample of the GPS signal $S_{L1}(t)$ after mixing to baseband. The researchers who recorded the data mixed the incoming signal to baseband with a fairly accurate estimate of the signal's instantaneous center frequency (including Doppler). Hence, the baseband samples have only a small residual frequency, which gives rise to a slow rotation in the phase. If you only look at, say, 400 samples, the phase doesn't change much and you can see a pattern emerge when you plot individual samples in the complex plane. (If you look at much more than 400 samples, the slow phase rotation turns this pattern into a donut shape.) Generate a plot similar to Fig. 1 by finding a window of 400 samples of data within Y whose real and imaginary components form a square that is aligned with the plot axes. Having identified such a window, look at the real and imaginary components of Y over this window in the time domain. Identify for your plot what signal component from SL1 (t) happens to be in real(Y) and what component happens to be in imag(Y) over this window. Measure the smallest chip interval for both codes. Estimate from your data the ratio of the C/A code amplitude to the P (Y) code amplitude.

7.1 MATLAB code

```
%% Problem 6
clear all; clc; close all;
addpath("research/toolbox/")

%------ Load data
load('prn31_22apr03_01hrs40min00sec_gmt_fl1_46_08mhz_250msec.mat');

start = 8000;
Y_windowed = Y(start:start+400,:);

% Plotting the complex representation of the signal such that it forms a
% "perfect" square in the the complex plane.
```

```
figure()
plot(Y_windowed,'.')
xlabel('In_phase');
ylabel('Quadrature');
title('Complex_representation_of_the_baseband_signal');

t = XDelta*(0:400) * 1e6;
CA_code = real(Y_windowed);
Y_code = imag(Y_windowed);

% Plotting the real and imaginary parts of the signal
figure()
subplot(2,1,1)
    plot(t,CA_code)
    xlabel('time_[us]');
    ylabel('X_{In-phase}');
    title('Time-domain_representation_of_X_{In-phase}');

subplot(2,1,2)
    plot(t, Y_code)
    xlabel('time_[us]');
    ylabel('X_{Quadrature}');
    title('Time-domain_representation_of_X_{Quadrature}');

% The ratio of the C/A code amplitude to the P(Y) code amplitude
ratio = mean(sqrt(CA_code.^2)) / mean(sqrt(Y_code.^2)) % Amplitude of C/A_code is
```

7.2 Results

Figure 8 shows the complex representation of the signals in phase-quadrature.

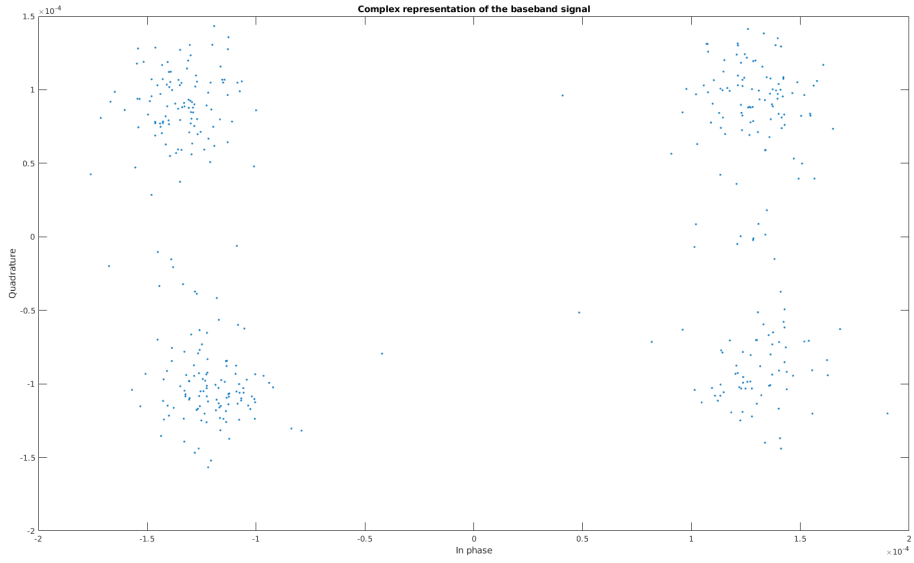

Figure 8: Individual samples of the Stanford "Big Dish" data plotted in the complex plane.

Figure 9, shows the time-domain representation of the previously shown phase-quadrature signals. It it possible to distinguish the GPS L1 C/A-code in the top plot. It can be appreciated from the graph that the period of the signal is $1\mu s$. The bottom graph exposes the P(Y)-code, characterized by its faster chipping of $0.1\mu s$.

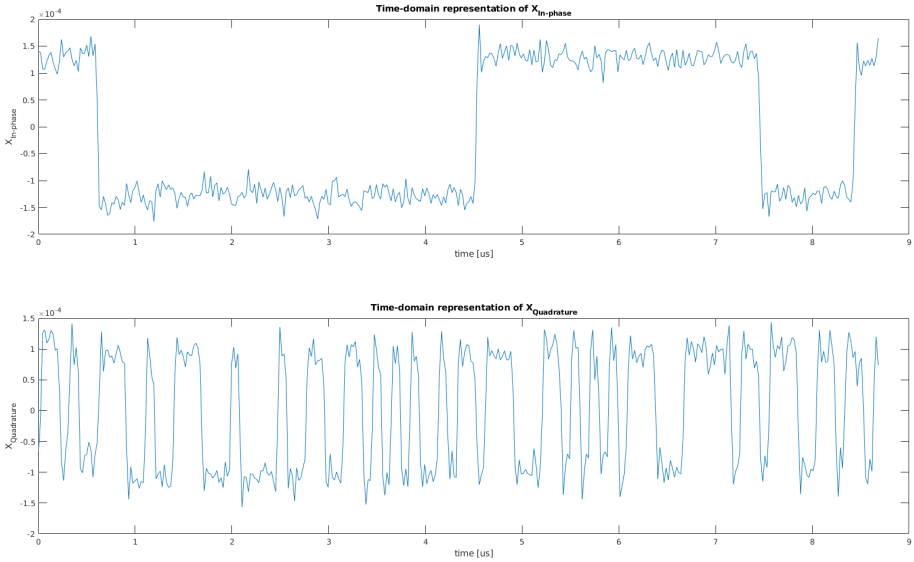

Figure 9: Time-domain representation of the signals previously shown in phase-quadrature.

Finally, the ratio of the C/A-code amplitude to the P(Y)-code amplitude reveals that the C/A-code amplitude is 1.4 times larger than P(Y)-code amplitude.

8 Problem 8

8.1 MATLAB code

```
%% Problem 9a

% correlationExperiments.m
%
% Experiment with properties of pseudorandom sequences.


clear;clc;
addpath("research/toolbox/")
%------ Setup
nStages = 10;                   % Number of stages in LFSR
Tc = 1e-3/1023;                 % Chip interval in seconds
delChip = 3/217;                % Sampling interval in chips
delOffset  = 0;                 % Offset of first sample
delt = delChip*Tc;              % Sampling interval in seconds
fs = 1/delt;                    % Sampling frequency in Hz
Np = 2^nStages - 1;             % Period of the sequence in chips
Nr = 20;                        % Number of repetitions of the sequence
Ns = round(Nr*Np/delChip);      % Number of samples of the sequence

N = 10000;
varVec = zeros(N,1);
for i=1:N
    % codeType:
    % rand ------ Sequence derived from Matlab randn function
    % pi -------- Sequence derived from the digits of pi
    % mseq ------ Maximal-length sequence with n = nStages
    codeType = 'rand';

    %------ Generate codes
    X1 = zeros(Np,1);
    X2 = zeros(Np,1);
    if(strcmp(codeType,'rand'))
      X1 = sign(sign(randn(Np,1)) + 0.1);
      X2 = sign(sign(randn(Np,1)) + 0.1);
    elseif(strcmp(codeType,'pi'))
      [sPi,vPi] = pi2str(2*Np);
      X1 = vPi(1:Np) >= 5;
```

```
      X1 = 2*X1 - 1;
      X2 = vPi(Np+1:2*Np) >= 5;
      X2 = 2*X2 - 1;
    elseif(strcmp(codeType,'mseq'))
    ciVec1 = [9, 4]';
    ciVec2 = [9, 2]';
    a0Vec1 = [1;zeros(nStages-1,1)];
    a0Vec2 = ones(nStages,1);
    X1 = generateLfsrSequence(nStages,ciVec1,a0Vec1);
    X2 = generateLfsrSequence(nStages,ciVec2,a0Vec2);
    X1 = 2*X1 - 1;
    X2 = 2*X2 - 1;
    else
      error('Unrecognized code type');
    end

    %------ Compute the sequence crosscorrelation
    [Rseq12,iiVecSeq] = ccorr(X1,X2);
    index = find(iiVecSeq==0);
    crosscorrVec(i) = Rseq12(index);
end

% The problem asks about the variance of the cross-correlation at k=0
var(crosscorrVec)

%% Problem 9b

% correlationExperiments.m
%
% Experiment with properties of pseudorandom sequences.


clear;clc; format short
addpath("research/toolbox/")
%------ Setup
nStages = 10;                   % Number of stages in LFSR
Tc = 1e-3/1023;                 % Chip interval in seconds
delChip = 3/217;                % Sampling interval in chips
delOffset  = 0;                 % Offset of first sample
delt = delChip*Tc;              % Sampling interval in seconds
fs = 1/delt;                    % Sampling frequency in Hz
Np = 2^nStages - 1;             % Period of the sequence in chips
Nr = 20;                        % Number of repetitions of the sequence
Ns = round(Nr*Np/delChip);      % Number of samples of the sequence

N = 6;
```

```
varVec = zeros(N,1);
for i=1:N
    ciVec1 = [
        [10, 9, 8, 5]',...
        [10, 9, 7, 6]',...
        [10, 9, 7, 3]',...
        [10, 9, 6, 1]',...
        [10, 9, 5, 2]',...
        [10, 9, 4, 2]'
    ];
    ciVec2 = [
        [10, 9, 8, 7, 5, 4]',...
        [10, 9, 8, 7, 4, 1]',...
        [10, 9, 8, 7, 3, 2]',...
        [10, 9, 8, 6, 5, 1]',...
        [10, 9, 8, 6, 4, 2]',...
        [10, 9, 8, 6, 4, 2]'
    ];
    a0Vec1 = [1;zeros(nStages-1,1)];
    a0Vec2 = ones(nStages,1);
    X1 = generateLfsrSequence(nStages,ciVec1(:,i),a0Vec1);
    X2 = generateLfsrSequence(nStages,ciVec2(:,i),a0Vec2);
    X1 = 2*X1 - 1;
    X2 = 2*X2 - 1;
    %------ Compute the sequence crosscorrelation
    [Rseq12,iiVecSeq] = ccorr(X1,X2);

    bound_crosscorrVec(i) = max(Rseq12);

    %------ Compute the sequence autocorrelation
    [Rseq11,iiVecSeq] = ccorr(X1,X1);
    index = find(iiVecSeq==0);
    autocorrVec(i) = Rseq11(index);
end

% The value of the autocorr at k=0 ~ (N-1)
autocorrVec

% The bound of the cross-correlation >= sqrt(N)
bound_crosscorrVec

%% Problem 9c

clear;clc;close all
addpath("research/toolbox/")
%------ Setup
```

```
nStages = 10;                   % Number of stages in LFSR
Tc = 1e-3/1023;                 % Chip interval in seconds
delChip = 3/217;                % Sampling interval in chips
delOffset  = 0;                 % Offset of first sample
delt = delChip*Tc;              % Sampling interval in seconds
fs = 1/delt;                    % Sampling frequency in Hz
Np = 2^nStages - 1;             % Period of the sequence in chips
Nr = 20;                        % Number of repetitions of the sequence
Ns = round(Nr*Np/delChip);      % Number of samples of the sequence
% codeType:
% rand ---- Sequence derived from Matlab randn function
% pi ------ Sequence derived from the digits of pi
% mseq ---- Maximal-length sequence with n = nStages
codeType = 'pi'; % Change this variable to see the different results

%---- Generate codes
X1 = zeros(Np,1);
X2 = zeros(Np,1);
if(strcmp(codeType,'rand'))
  X1 = sign(sign(randn(Np,1)) + 0.1);
  X2 = sign(sign(randn(Np,1)) + 0.1);
elseif(strcmp(codeType,'pi'))
  [sPi,vPi] = pi2str(2*Np);
  X1 = vPi(1:Np) >= 5;
  X1 = 2*X1 - 1;
  X2 = vPi(Np+1:2*Np) >= 5;
  X2 = 2*X2 - 1;
elseif(strcmp(codeType,'mseq'))
ciVec1 = [9, 4]';
ciVec2 = [9, 2]';
a0Vec1 = [1;zeros(nStages-1,1)];
a0Vec2 = ones(nStages,1);
X1 = generateLfsrSequence(nStages,ciVec1,a0Vec1);
X2 = generateLfsrSequence(nStages,ciVec2,a0Vec2);
X1 = 2*X1 - 1;
X2 = 2*X2 - 1;
else
  error('Unrecognized code type');
end

%---- Oversample code
X1os = oversampleSpreadingCode(X1,delChip,delOffset,Ns,Np);
X2os = oversampleSpreadingCode(X2,delChip,delOffset,Ns,Np);

%---- Compute autocorrelation
```

```
[R1,iiVec] = ccorr(X1os,X1os);
[R2,iiVec] = ccorr(X2os,X2os);

%------ Compute crosscorrelation
[R12,iiVec] = ccorr(X1os,X2os);

ratio = max(R1)/max(R12)

%------ Compute power spectra
% The scaling here ensures that sum(Si*delf) = 1 W, as expected, for i = 1, 2.
S1 = abs(delt*fft(X1os)).^2/(Ns*delt);
S2 = abs(delt*fft(X2os)).^2/(Ns*delt);
S12 = abs(delt*fft(R12)).^2/(Ns*delt);
delf = 1/(delt*Ns);
fVec = [0:Ns-1]'*(delf);

%------ Scale for 1-Hz delf for plotting
% We'll present the power spectra in units of dBW/Hz, so we need to scale
% the spectra so that they match a delf = 1 Hz frequency spacing.
S1 = S1*delf;
S2 = S2*delf;
S12 = S12*delf;

%------ Plot
figure(1);clf;
subplot(211)
plot(iiVec,R1/Ns);
grid on;
ylabel('R_{X1}');
title('X1 and X2 autocorrelation')
subplot(212)
plot(iiVec,R2/Ns);
grid on;
ylabel('R_{X2}');
xlabel('Lag (samples)');
figure(2);clf;
plot(iiVec,R12/Ns);
title('X1 and X2 crosscorrelation')
xlabel('Lag (samples)');
grid on;
ylabel('R_{X1,X2}');
figure(3);clf;
subplot(211)
plot(fVec/1e3,10*log10(S1));
grid on;
```

```
xlim([0  30]);
ylim([-100,0]);
title('X1 and X2 power spectral densities')
ylabel('S_{X1}(f) (dBW/Hz)');
subplot(212)
plot(fVec/1e3,10*log10(S2));
grid on;
xlim([0  30]);
ylim([-100,0]);
ylabel('S_{X2}(f) (dBW/Hz)');
xlabel('Frequency (kHz)');
```

8.2 Results

8.2.1 Item a

The result verifies the claim

8.2.2 Item b

The result verifies the claim

8.2.3 Item c

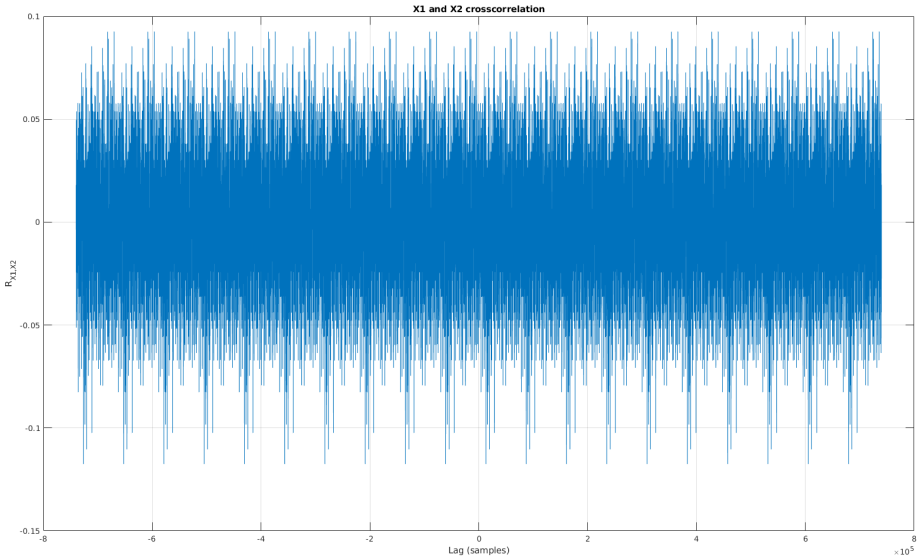

Figure 10: Crosscorrelation of X1 and X2 for the ŕandćase.

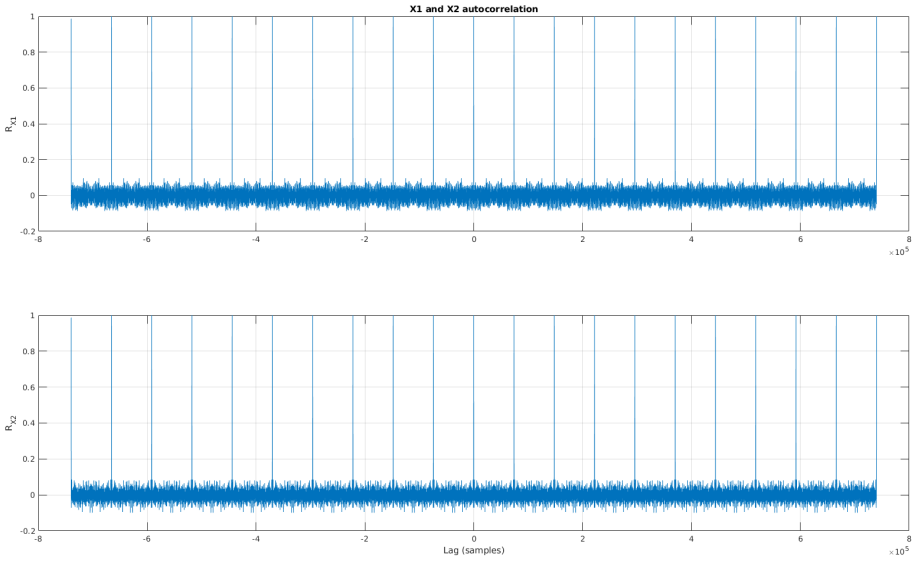

Figure 11: Autocorrelation of X1 and X2 for the 'rand' case.

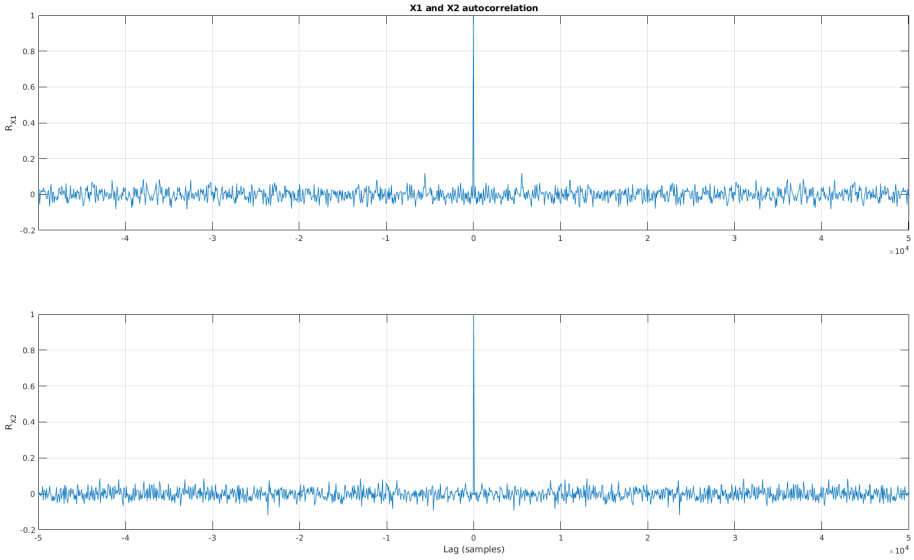

Figure 12: Zoomed autocorrelation of X1 and X2 for the 'rand' case.

The ratio of the maximum of $R_X(\tau)$ to the maximum of $R_{X1,X2}(\tau)$ is 11.54 for the case where we work with 'rand' method.

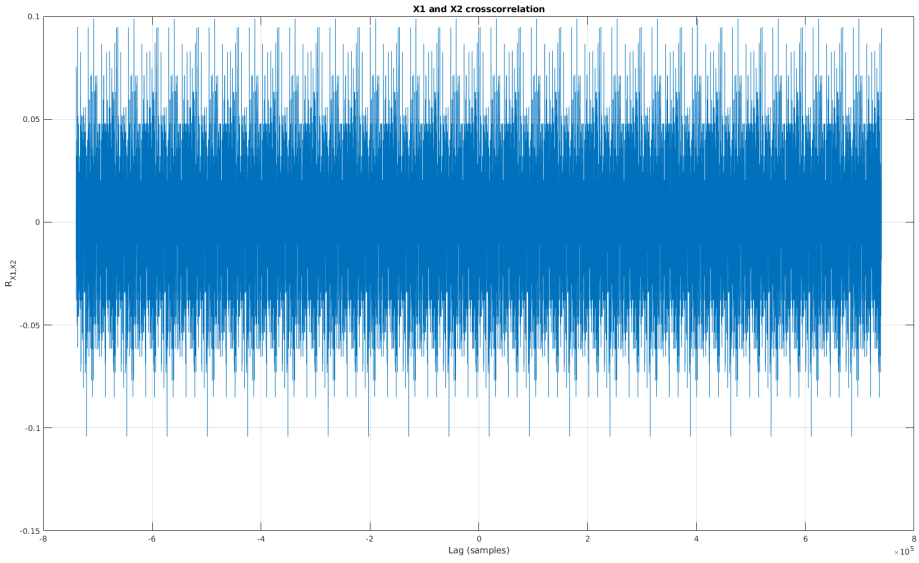

Figure 13: Crosscorrelation of X1 and X2 for the 'pi' case.

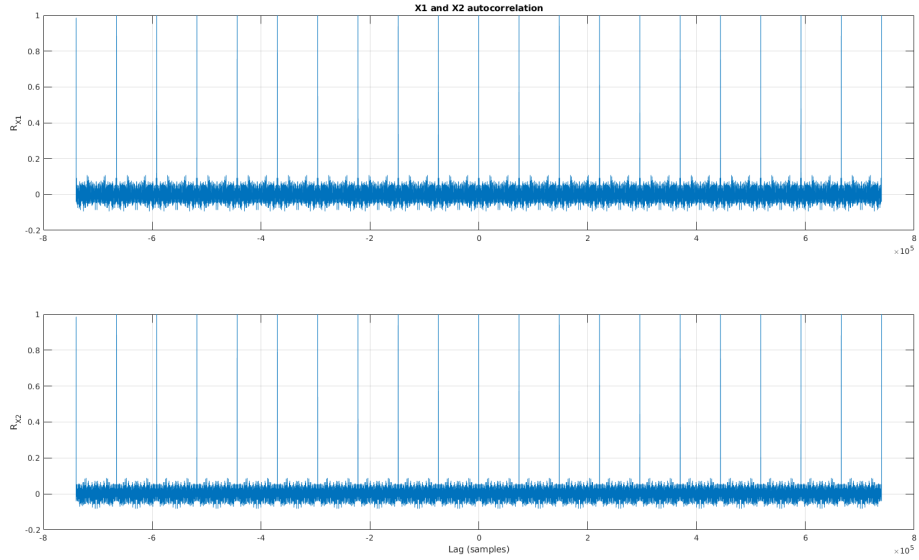

Figure 14: Autocorrelation of X1 and X2 for the 'pi' case.

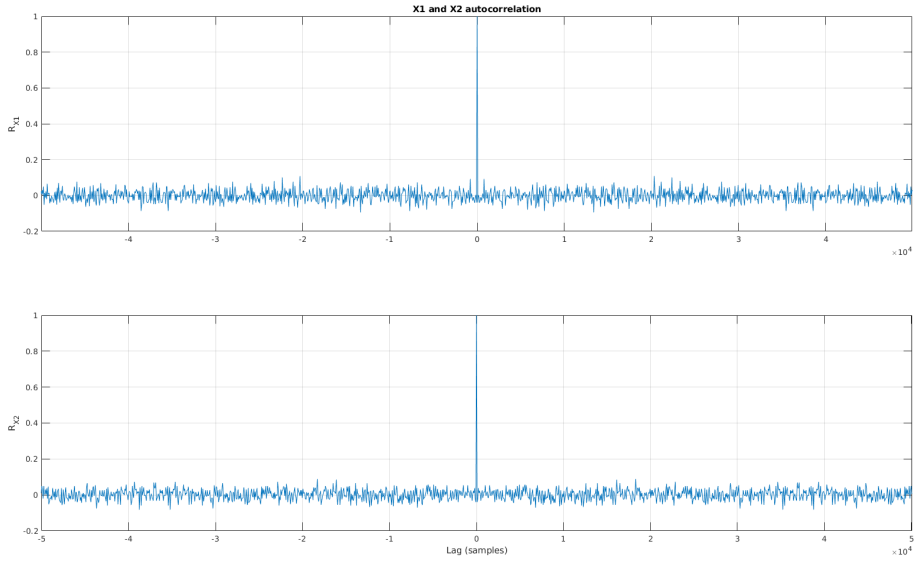

Figure 15: Zoomed autocorrelation of X1 and X2 for the 'pi' case.

The ratio of the maximum of $R_X(\tau)$ to the maximum of $R_{X1,X2}(\tau)$ is 10.12 for the case where we work with 'pi' method.

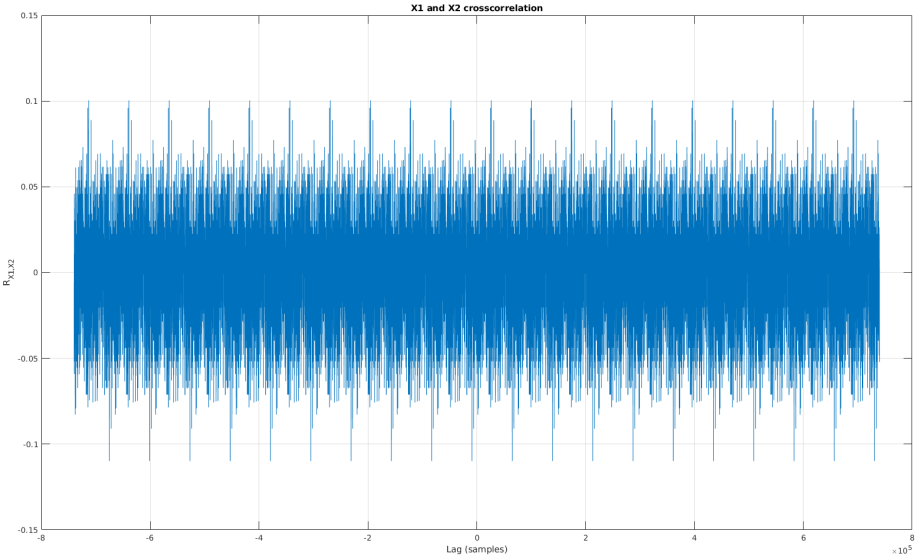

Figure 16: Crosscorrelation of X1 and X2 for the 'mseq' case.

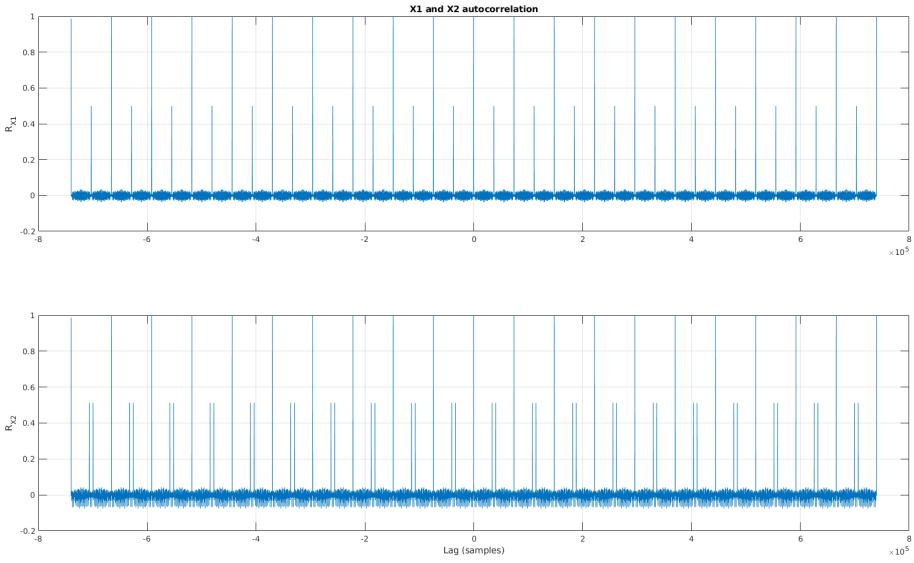

Figure 17: Autocorrelation of X1 and X2 for the 'mseq' case.

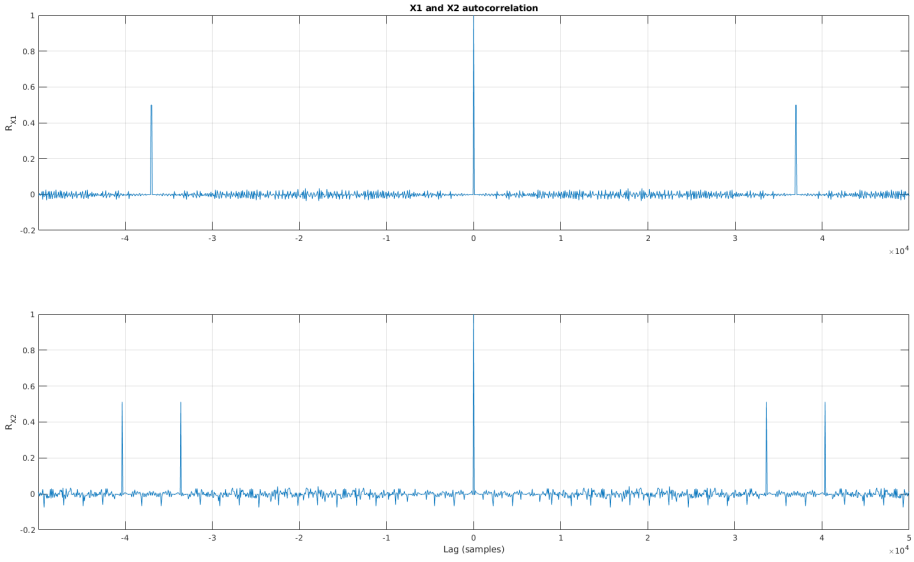

Figure 18: Zoomed autocorrelation of X1 and X2 for the 'mseq' case.

The ratio of the maximum of $R_X(\tau)$ to the maximum of $R_{X1,X2}(\tau)$ is 9.9659 for the case where we work with 'mseq' method.

Analyzing the graphs it's hard to conclude which method provides the best cross-correlation properties. They all look very similar, showing values concentrated below arround +-0.05 and

spikes upto +-0.1. The ideal behavior we are looking for is a cross-correlation equal to zero. Which would indicate that no two signal could be confused because they are perfectly orthogonal.

The analysis of the autocorrelation is a little different. The 'rand' and 'pi' methods look very similar. But the 'mseq' method is quite different. It has a much lower floor, which it's closer to the ideal case we are looking for: the Dirac delta. However, it presents smaller spikes to the sides. This could be pretty bad because one could confuse this smaller peaks with the main peak and consequently get an inaccurate range measurement due to pourly "locking" to the local replica of the code. In addition to the aforementioned, the calculated ratios show that the 'rand' method provides the greater relationship between max autocorrelation and max cross-correlation, which is a desired property since it means that there are more chances of correctly identifing and 'locking' to the desired satellite's code. This result surprised me since I was expecting the mseq to have the best properties.

9 Problem 9

9.1 Instruction

Use the generateLfsrSequence function you wrote for Problem Set 2 and your lecture notes to answer the questions.

Which of the following characteristic polynomials correspond to maximal length sequences for a 9-stage linear feedback shift register? Of those that do, which pairs of polynomials correspond to so-called preferred pairs of m-sequences, i.e., m-sequences that can be shifted and summed (modulo 2) to generate a family of Gold codes?

9.2 MATLAB code

```
function [lfsrSeq] = generateLfsrSequence(n,ciVec,a0Vec)
%
% Generate a 1/0-valued linear feedback shift register (LFSR) sequence.
%
% INPUTS
%
% n ------- Number of stages in the linear feedback shift register.
%
% ciVec -- Nc-by-1 vector whose elements give the indices of the Nc nonzero
%          connection elements. For example, if the characteristic polynomial
%          of an LFSR is f(D) = 1 + D^2 + D^3, then ciVec = [2,3] or [3,2].
%
% a0Vec -- n-by-1 1/0-valued initial state of the LFSR, where a0Vec = [a(-1),
%          a(-2), ..., a(-n)]. In defining the initial LFSR state, a
%          Fibonacci LFSR implementation is assumed.
%
% OUTPUTS
%
% lfsrSeq -- m-by-1 vector whose elements are the 1/0-valued LFSR sequence
```

```
%             corresponding to n, ciVec, and a0Vec, where m = 2^n - 1. If the
%             sequence is a maximal-length sequence, then there is no
%             repetition in the m sequence elements.

%------------------------ FIBONACCI CONFIGURATION ------------------------------

aVec = a0Vec;

m = 0;
lfsrSeq = [];
while (m < 2^n - 1)
    output = aVec(n);
    ai    = mod( sum(aVec(ciVec)), 2);
    aVec = [ai; aVec(1:end-1)] % Allows to see the state (I only care about
                               % the last one). I prefer not to alter the
                               % signature of the function.

    lfsrSeq = [lfsrSeq; output];
    m = m+1;
end

end


%% Extra Problem
clc, close all, clear all
%------ Setup
nStages = 9;                    % Number of stages in LFSR

% f1 (D) =1 + D4 + D9
% f2 (D) =1 + D3 + D5 + D6 + D9
% f3 (D) =1 + D3 + D5 + D8 + D9
% f4 (D) =1 + D3 + D4 + D6 + D9
% f5 (D) =1 + D3 + D7 + D8 + D9
% f6 (D) =1 + D2 + D9
ciVec1 = [9, 4]';
ciVec2 = [9, 6, 5, 3]';
ciVec3 = [9, 8, 5, 3]';
ciVec4 = [9, 6, 4, 3]';
ciVec5 = [9, 8. 7. 3]';
ciVec6 = [9, 2]';

a0Vec1 = ones(nStages,1);

% Examine the final state to determine if it matches the initial and
% thus conclude that it's indeed a maximal length LFSR
```

```
X1 = generateLfsrSequence(nStages,ciVec1,a0Vec1); % maximal length
X2 = generateLfsrSequence(nStages,ciVec2,a0Vec1); % maximal length
X3 = generateLfsrSequence(nStages,ciVec3,a0Vec1); % not maximal lenght
X4 = generateLfsrSequence(nStages,ciVec4,a0Vec1); % maximal length
X5 = generateLfsrSequence(nStages,ciVec5,a0Vec1); % not maximal length
X6 = generateLfsrSequence(nStages,ciVec6,a0Vec1); % not maximal length

% Now let's see if it's a Gold sequence by analizing the cross-correlation

%------ Compute crosscorrelation
[R12,iiVec] = ccorr(X1,X2);
[R13,iiVec] = ccorr(X1,X3);
[R23,iiVec] = ccorr(X2,X3);

figure()
subplot(1,3,1)
plot(R12)
title('R12')
grid on

subplot(1,3,2)
plot(R13)
title('R13')
grid on

subplot(1,3,3)
plot(R23)
title('R23')
grid on
```

9.3 Results

To determine if the characteristic polynomials correspond to maximal length sequences for a 9-stage ($n = 9$) LFSR one needs to check if the state of the LFSR (a_i) is the same as the initial state after generating a sequence of length $N = 2^n - 1$. As indicated in the MATLAB code, the characteristic polynomials 1, 2 and 4 correspond to maximal length sequence generators.

The characteristic of the Gold sequences is that their cross-correlation is defined by the following

$$R_{ab}(k) = -t(n), -1, t(n) - 2 \tag{26}$$

Where $t(n) = 1 + 2^{(n+1)/2}$ if n is odd. And $t(n) = 1 + 2^{(n+2)/2}$ if n is even. So, you have a guarantee that the cross-correlation is bounded. In our case, n=9. Then the $|R_ab(k)| < 33$. The autocorrelation follows a similar pattern as the cross-correlation except at the main peak. Figure 19 shows the cross-correlation for all the maximal-length sequences pairs. The conclusion is that the sequences are not Golden since neither respect the bound.

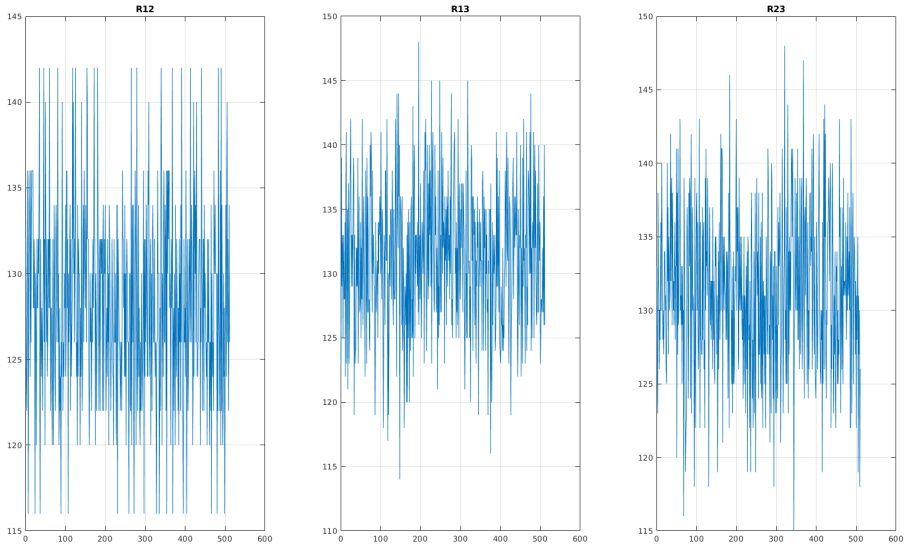

Figure 19: Cross-correlation of all the maximal length sequences.