SSY 135 Project part 1

Group E: Oskar Claeson Erik Tilly Yuling Zhang Haitham Babbili

I. Introduction

The objective of this paper is to provide a small basis for understanding both Rayleigh and Rician flat fading channels. This is done through simulations using two main methods, which are known as the Filter method and the Spectrum method. This paper explains how to use the methods in order to simulate a Rayleigh and Rician fading channel with a Jake's Doppler spectrum. In order to assess the channels, results from different tools are analyzed. These tools are the Power Spectral Density (PSD), Probability Density Function (PDF), Cumulative Distribution Function (CDF) and the Auto-Correlation Function (ACF). Also, the different advantages and/or disadvantages of using either method is discussed, to provide a better understanding of the wireless channels.

II. Rayleigh and Rician Flat Fading channels

This section is the documentation of how the channels were created. The base method describes the implementation of a Rayleigh channel while it then is changed according to the equation 1 to create a Rician channel for K_c larger than 0. Comparing Rayleigh and Rician channel, Rician has a line of sight(LOS) component whereas Rayleigh does not. The K-factor is used to describe a Rician fading channel which is defined as the ratio of signal power in the dominant component over the other multipath components . Thus, the K-factor is found as: $K = \frac{s^2}{2\sigma^2}$
In this project the power of the LOS component is represented by k_c. When the Rician channels are generated, it's made sure that they are normalized in order for the channel to be of unit energy $\mathbb{E}(|c(nT_s)|^2) = 1$. For $k_c = 0, 1, 10$, he K factor is thus $0, 0.5, 50$. The K factor is proportional to the value of k_c. Based on 1, when $k_c = 0$, it is exact Rayleigh channel and, actually, Rayleigh is part of the Rician channel.

$$c_{Rician}(nT_s) = c_{Rayleigh}(nTs) + k_c \tag{1}$$

The channels are implemented with a "filter method" and a "spectrum method" after the parameters are determined and checked that they full fill the requirements.

A. Determine the parameters

The Doppler frequency is calculated as:

$$f_D = \frac{v \times f_c}{c} \approx 55.3333[Hz]$$

To avoid aliasing, the value of T_s should satisfy: $1/T_s > 2f_D$. Thus the maximum tolerated T_s to avoid aliasing is $T_s < 0.009[s] = 9[ms]$. The provided $T_s = 0.1ms$ is lower than the maximum $9ms$. Therefore, $T_s = 0.1ms$ meets this requirement and can be used as sample time.

When estimating the PSD, the quality depends partly on the window size. The shorter the window size is, the poorer result and larger variation. While the window size is increased, better performance is obtained, but at the same time if the size exceed the frequency resolution some noise will occur. This resolution can be calculated accordingly:

$$f_{res} = \frac{f_s}{N_s} = \frac{1}{T_s N_s}$$

$$f_{res} = \langle 0.8 - 1 \rangle$$

A factor was calculated according to Parseval's theorem in equation II-A.

$$E = \int_{-\infty}^{+\infty} |x(t)|^2 dt = \frac{1}{2\pi} \int_{-\infty}^{+\infty} |X(\omega)|^2 dw$$

B. Filter method

To generate the channel using the Filter method, first choose T_s such that no aliasing can occur. Secondly, choose number of samples N_s and make sure this window size gives a good resolution. The filter vector $g(nT_s)$ is generated in the following way, where $n = -NT_s, ..., NT_s$ and N is a chosen length:

$$g(t) = \mathcal{F}^{-1}[\sqrt{S_c(f)}] = \begin{cases} \frac{\mathcal{J}_{1/4}(2\pi f_D|t|)}{\sqrt[4]{|t|}}, & t \neq 0 \\ \frac{\sqrt[4]{\pi f_D}}{\Gamma(5/4)}, & t = 0 \end{cases} \tag{2}$$

Where $\mathcal{J}$ is the Bessel function, Γ is the gamma function. The vector length should be carefully chosen as a small N leads to artifacts in the PSD and a large N will lead to a high complexity. The filter vector g should be normalized to make sure it has the unit energy. In order to simulate the channel this filter vector g is convolved with independent samples $x(nT_s)$ from a complex Gaussian distribution of length N_s which are randomly generated as:

$$x = (randn(N_s, 1) + j \cdot randn(N_s, 1)) \cdot a \tag{3}$$

where a is a constant to ensure that the samples of x has unit-variance. The resulting channel c has length $N_s + 2N$ due to transients following the convolution operation, which should be discarded.

C. Spectrum method

Provided that the used parameters fulfill the requirements stated in the previous section, the spectrum method makes use of the Doppler spectrum in the frequency domain, which is the Fourier transform of the ACF. Since the Doppler spectrum is band limited to $[-f_D, f_D]$ its only non zero between those frequencies and zero otherwise which is shown in 4. In this project samples for only the positive side were generated and then flipped to create the negative half of the spectrum since these should be the same.

$$S_c(f) = \begin{cases} \frac{1}{\pi f_D} \frac{1}{\sqrt{1-(f/f_D)^2}} & \mid f \mid \leq f_D \\ 0 \qquad otherwise \end{cases} \tag{4}$$

When the Doppler spectrum is constructed according to 4, with f_s as the upper limit, $G(f)$ is found by taking the square root of S_c. $G_p(f)$ is the periodic version of $G(f)$ and to create $G_p(f)$ the $G(f)$ is padded with zeros between the two halves such that the total length is N_s. The next step in the process is to generate samples from a complex Gaussian distribution denoted X created as in (3) with the same length as $G_p(f)$, i.e N_s. Then $G_p(f)$ is multiplied with X to create a vector C ,which in time domain corresponds to a convolution. In order to get the channel response $c(nT_s)$ and the vector C is inverse discrete Fourier transformed. To make sure that the channel c has unit variance a scale factor a is calculated using Parseval's theorem which is multiplied with the samples X before the multiplication.

III. Simulations of different Rician channels

In this section the simulations is presented and analysed. The different simulations that are performed as PSD, CDF, PDF and ACF. The PDF, CDF and ACF are performed with different k_c values.

A. Power Spectral Density simulation

In figure 1, the estimated PSD of the simulated channels are plotted vs the theoretical PSD. The plot show that both methods have slightly more power compared to the theoretical PSD for frequencies lower than the Doppler frequency, however in general, the form of the curves are matching. And for both of the methods, there are larger variance appearing. This is expected since the theoretical curve is calculated without any noise. When studying how the curves are behaving above the Doppler frequency the filter method and spectrum method differ with about $10dB$, whereas the theoretical is zero-valued. This indicates that the spectrum method is better at suppressing the higher frequencies since it has a steeper angle when cutting of the higher frequencies.

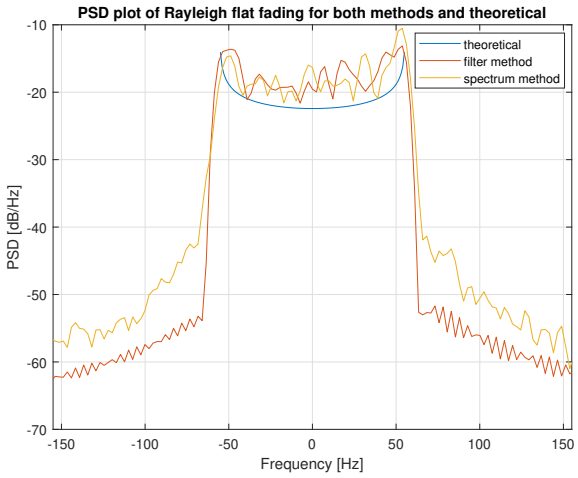

Fig. 1. Estimate PSD compared with theoretical

B. Probability Density Function

In this section, a comparison between the theoretical PDF with the estimated Rician PDF is made with 3 different values of k_c. These comparisons are plotted in figure 2.

When k_c = 0, it can be seen that the PDF is centered around 1.25, which is expected since that is how a Rayleigh distribution looks like. If k_c is increased to 1, a LOS component is present. This however, does not generate a significant change compared to the k_c = 0 channel since according to the K factor, a k_c = 1 gives that the ratio of the LOS component compared to the rest is 0.5 and thus slightly changes the shape. For the larger values of k_c, in this project, is chosen as 10, the LOS component is much more dominant over the non-LOS components which leads to that the Rician PDF resemble a Gaussian distribution centered around the value of the LOS component, in this case 10. For all values of k_c, both simulated methods seem to provide a more narrow curve compared to the theoretical curve which could be due to that these methods normalize the channels to get a unit-variance.

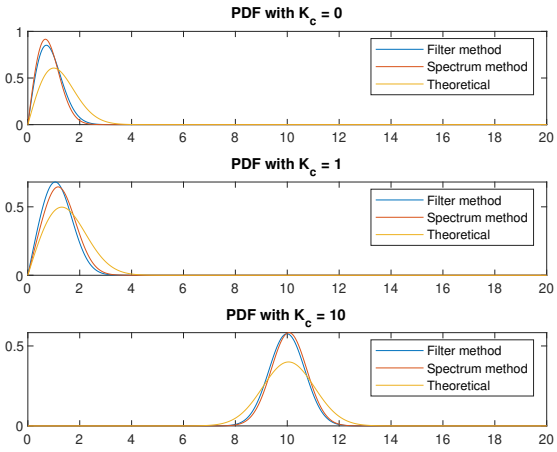

Fig. 2. Comparison of the estimated PDFs of the filter and the spectrum method compared to the theoretical channel with different k_c.

C. Cumulative Distribution Function

The CDF of the simulated channels and the theoretical are shown in figure 3.The CDF of the Rayleigh distribution i.e k_c = 0 can be seen as the Rician CDF with K = 0. The plot clearly show that the value of k_c (or the LOS component) has an impact on the CDF. As k_c is increased, the distribution shifts further to the positive direction on the x-axis which can be seen as an implication of higher SNR. For the increase of k_c from 0 to 1 the change is very slight. However, when k_c is increased to 10, the dependency on the K-factor is more obvious as this shifts the curve to be centered at 10.

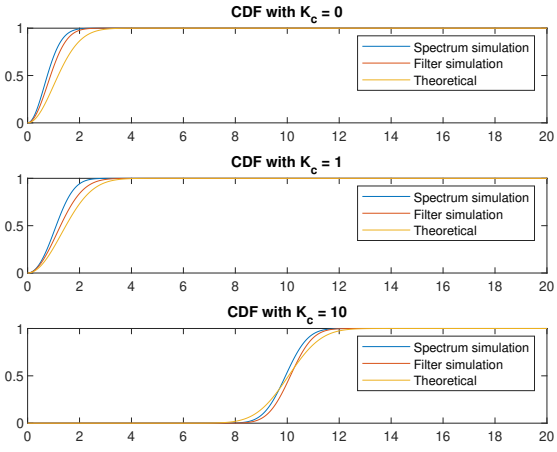

Fig. 3. Comparison of the estimated CDFs of the filter and the spectrum method compared to the theoretical channel with different k_c.

D. Auto-Correlation Function

The auto-correlation functions of both simulations and the theoretical is shown in figure4, where the theoretical ACF is given by the following equation:

$$A_c(\Delta t) = J_0(2\pi f_D \Delta t) + k_c^2 \tag{5}$$

From inspecting the plot it seems as if the shape of the ACF for both methods are more similar to the theoretical one for higher values of k_c but with a small error in amplitude as it seems like there is more correlation than expected when compared to theoretical values. The amplitude of the ACF is heavily dependant on the k_c value as the ACF contain a factor k_c^2. As can be seen in the plots, the correlation peaks or increases once again on points other than at zero, which could be a case of periodicity in both methods.

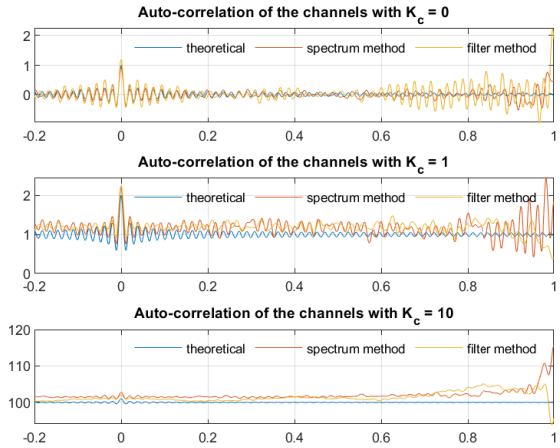

Fig. 4. Comparison of the estimated ACFs of the filter and the spectrum method compared to the theoretical channel with different k_c.

E. Discussion part 1

One advantage of using the spectrum method over the filter method is that it is less complex to perform a simple multiplication compared to the complex convolution operation used in the filter method. This makes the spectrum method quicker. Another advantage of using the spectrum method is that it is more efficient to remove unwanted frequencies, which was shown in the PSD analysis. A disadvantage of choosing the spectrum method over the filter method is that when the window size, i.e N_s is very large it takes up a lot of memory or computer capacity which leads to the decision that for very large N_s the filter method is the only viable choice albeit a much slower one.

IV. Time and Frequency-Varying Rician Fading Channels

In this task there are 18 figures plotted of different combinations of $f_d T_s$, number of paths L and values of K_c all with $N = 300$ time samples and $M = 64$ frequency samples. These figures are shown in appendix A. The delay spread can be calculated as:

$$LT_s \geq \tau_{DS} \Rightarrow \tau_{DS} = \begin{cases} 0.1ms, & \text{L=1} \\ 0.2ms, & \text{L=2} \\ 0.3ms, & \text{L=3} \end{cases}$$

To calculate the velocity, the following formula is used:

$$v = \frac{f_D T_s \cdot c}{f_c \cdot T_s}$$

which gives that

$$v = \begin{cases} 150m/s = 540km/h, & f_D T_s = 0.1 \\ 7.5m/s = 27km/h, & f_D T_s = 0.005 \end{cases}$$

The value of $f_d T_s$ corresponds to the user velocity. From figure 13 and 22, it is apparently to see that the figure, which has higher value of $f_d T_s$ is much smoother than the other under the same k_c and L.

Fixed the value of $f_d T_s$ and L, the peak value of magnitude decreases with increasing value of k_c. From figure 4, 5, 8 and 11, it can be explained that, when k_c is increased, there is much more power in the system, then, each sample will have more power. Thus, the peak value of the magnitude is shrink, however, the number of samples with higher power is increased.

L means the number of paths. The higher value of L, the faster varying will occur along the frequency domain, shown in figure 5, 6 and 7. Therefore, when $L = 1$ and do the Discreate Fourier Transform(DFT) of channel response C, it shows as constant amplitude with varying frequency. With increasing value of L, more paths are added so that the figures look like the wave. After DFT, there are different values of amplitude corresponds to different frequency value.

V. Members Contribution

The way this project was done to first have all members on their own try to figure out the MATLAB simulation part and then together discuss the solutions. Due to different levels of experience in MATLAB, the simulations were mainly done by Yuling and Oskar, where Erik helped with troubleshooting and small fixes. Haitham tried as best he could and was present in discussions of how the implementations were made and gave inputs. Everybody does effort to write report. The bulk of the texts about theory and methods were written by Yuling, Haitham and Erik, whereas Oskar fixed figures, structure, language and sections such like this one, Erik also helped with structure.

Appendix A

The 18 combinations of L, $f_D T_s$ and k_c magnitude time and frequency-varying channel responses from Section IV are plotted here.

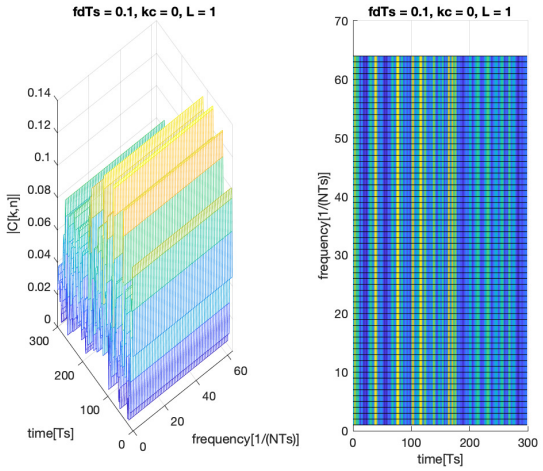

Fig. 5. $f_D T_s = 0.1$,$k_c = 0$,$L = 1$

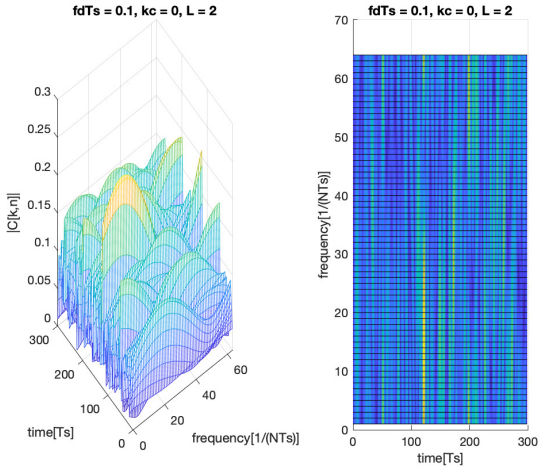

Fig. 6. $f_D T_s = 0.1$,$k_c = 0$,$L = 2$

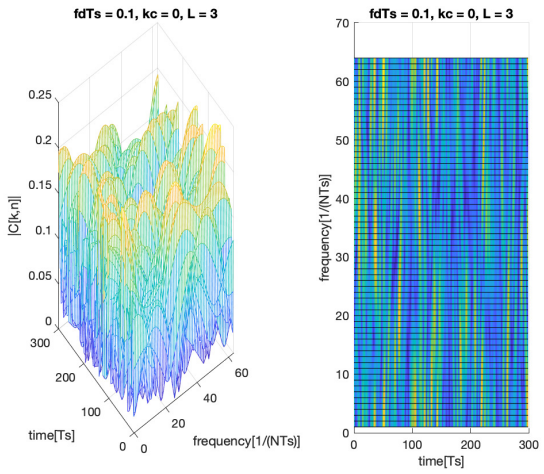

Fig. 7. $f_D T_s = 0.1$,$k_c = 0$,$L = 3$

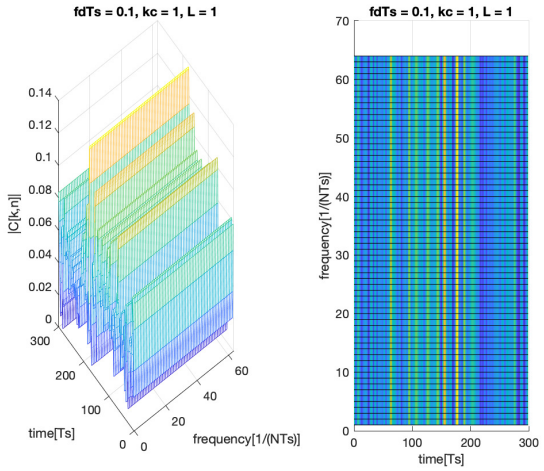

Fig. 8. $f_D T_s = 0.1$,$k_c = 1$,$L = 1$

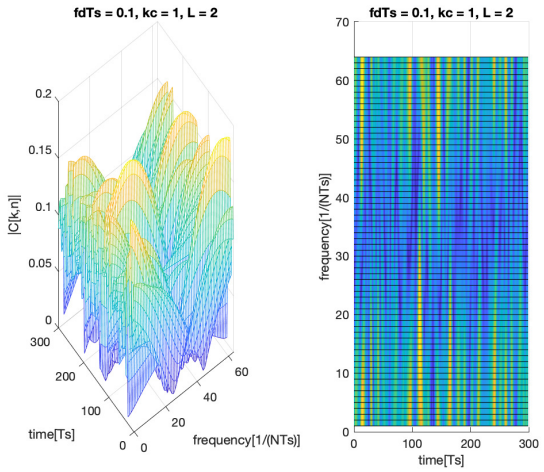

Fig. 9. $f_D T_s = 0.1$,$k_c = 1$,$L = 2$

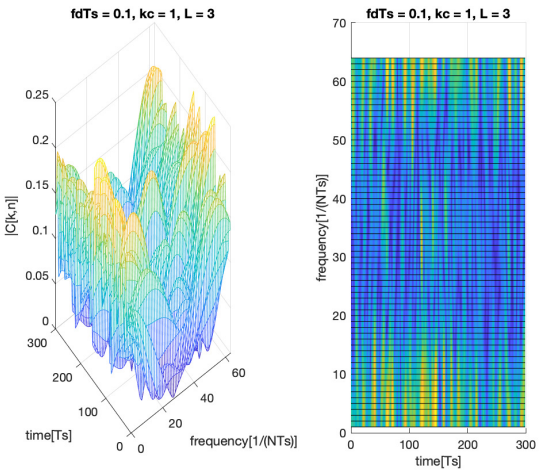

Fig. 10. $f_D T_s = 0.1$,$k_c = 1$,$L = 3$

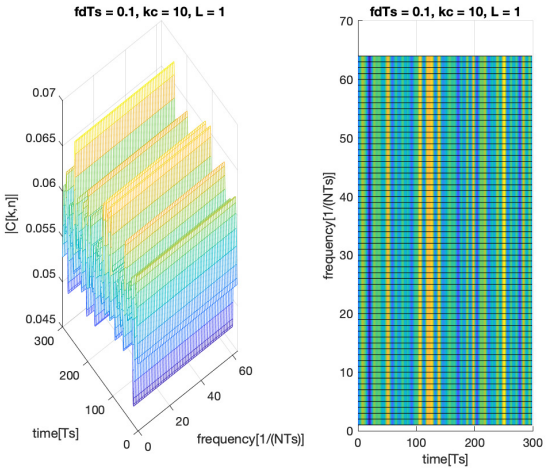

Fig. 11. $f_D T_s = 0.1$,$k_c = 10$,$L = 1$

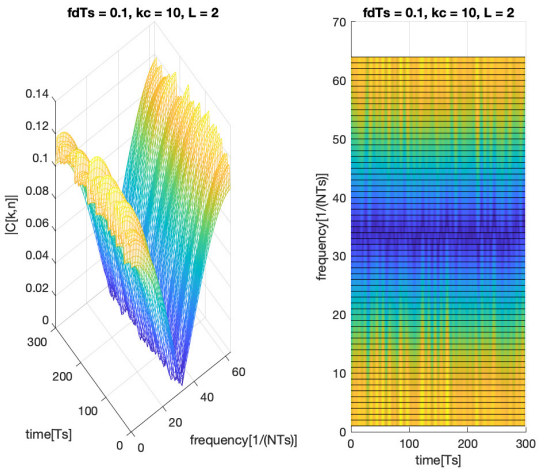

Fig. 12. $f_D T_s = 0.1$,$k_c = 10$,$L = 2$

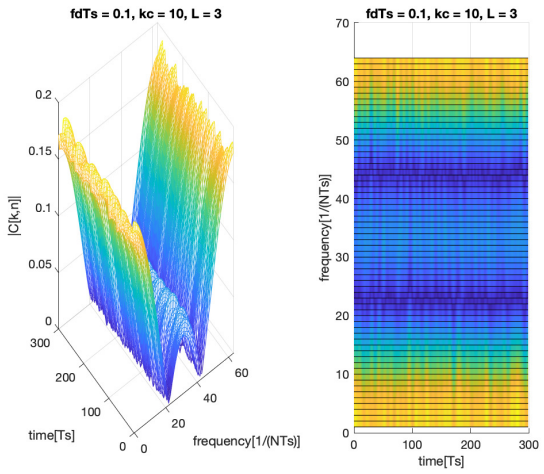

Fig. 13. $f_D T_s = 0.1, k_c = 10, L = 3$

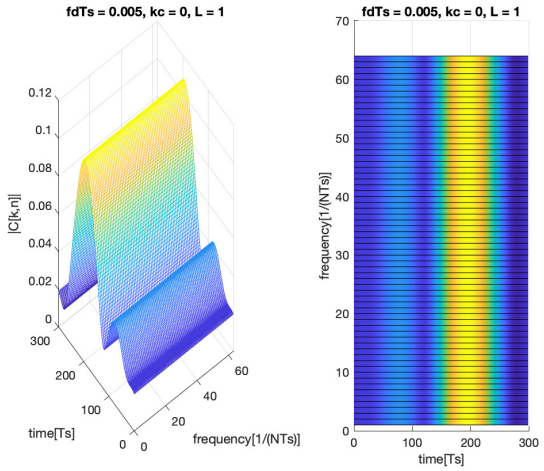

Fig. 14. $f_D T_s = 0.005, k_c = 0, L = 1$

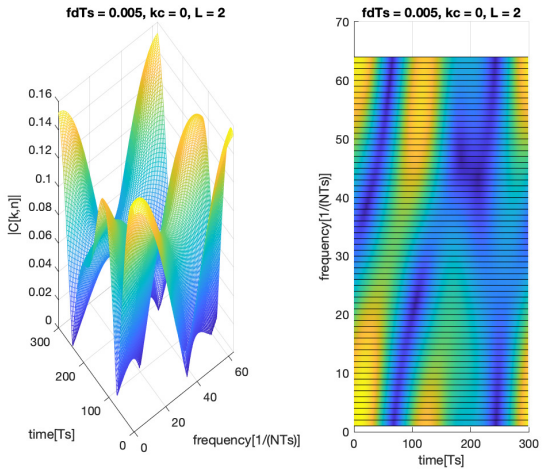

Fig. 15. $f_D T_s = 0.005, k_c = 0, L = 2$

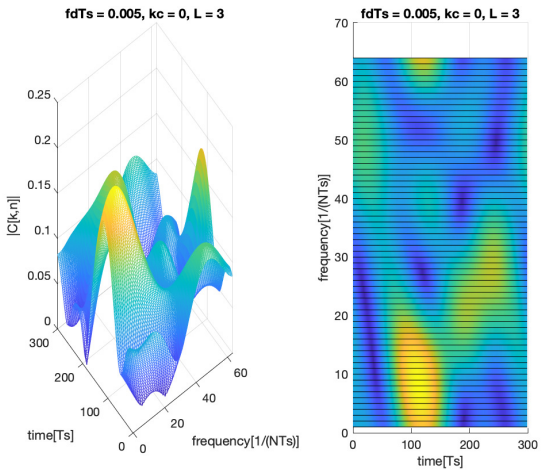

Fig. 16. $f_D T_s = 0.005, k_c = 0, L = 3$

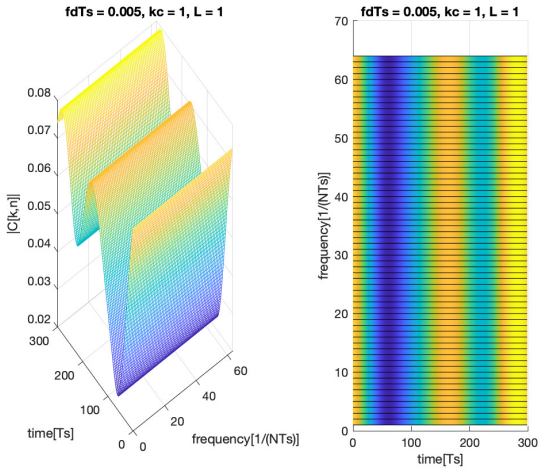

Fig. 17. $f_D T_s = 0.005, k_c = 1, L = 1$

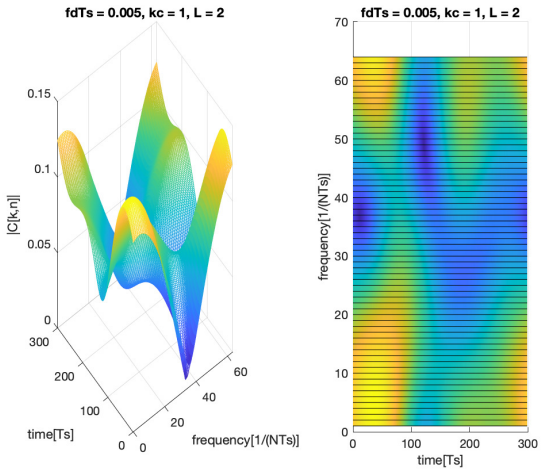

Fig. 18. $f_D T_s = 0.005, k_c = 1, L = 2$

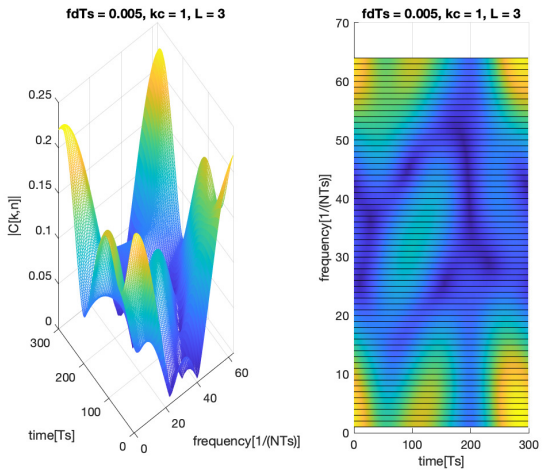

Fig. 19. $f_D T_s = 0.005$,$k_c = 1$,$L = 3$

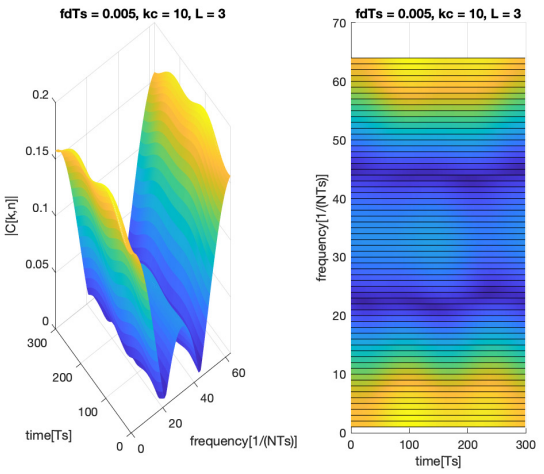

Fig. 22. $f_D T_s = 0.005$,$k_c = 10$,$L = 3$

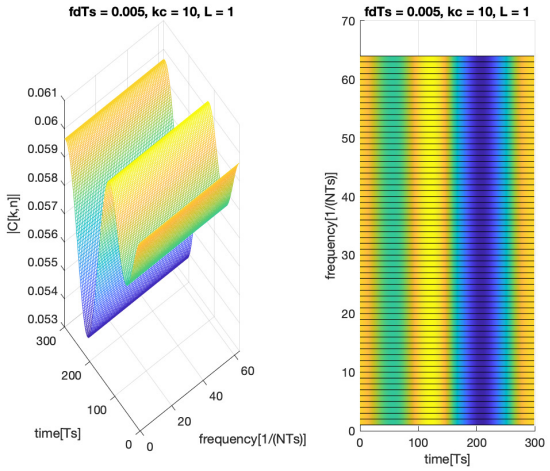

Fig. 20. $f_D T_s = 0.005$,$k_c = 10$,$L = 1$

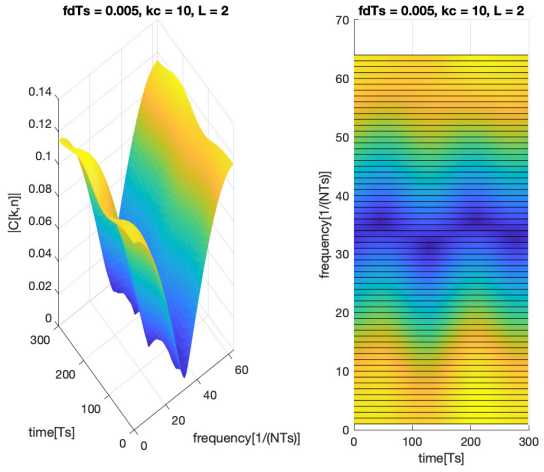

Fig. 21. $f_D T_s = 0.005$,$k_c = 10$,$L = 2$

Appendix B

Here the MATLAB code written by the authors are given.

```
%% Proj part 1
clc, clear all
% 1) simulate a frequency-flat Rayleigh fading gain process.
addpath('./functions')
% set given parameters for both filter- & spectrum method.
f_c = 2e9; % 2GHz frequency carrier
v = 30/3.6; % 30km/h velocity
f_D = v/physconst('LightSpeed')*f_c; % calculate the doppler frequency
T_s = 0.1e-3; % 0.1 ms sample interval
f_s = 1/T_s; % sampling frequency
maximumToleratedTs = 1/(2*f_D); % calculate the max Ts for no aliasing
if T_s > maximumToleratedTs % this must be satisfied
    error('The chosen sample interval time is too small, aliasing will occur')
end

% choose a reasonable N_s, not too small
N_s = 10000;

% create the Rayleigh flat fading channels.
c_spectrum = spectrumMethod(f_D,T_s,N_s,0);
expectedStuff = [mean(c_spectrum) var(c_spectrum)]
c_filter = filterMethod(f_D,T_s,N_s,0);
expectedStuff2 = [mean(c_filter) var(c_filter)]

%% Create the PSD plots
% theoretical PSD
f = (-60:f_s/N_s:60);
S = 1./(pi*f_D*(sqrt(1-(f/f_D).^2)));
S(abs(f)>=f_D) = 0; % remove parts outside the bandwidth
% simulation PSD
[PSD_spectrum,psdf1] = pwelch(c_spectrum,[],[],[],f_s,'centered');
PSD_spectrum = PSD_spectrum /(sum(PSD_spectrum)*(f_s/N_s));
[PSD_filter,psdf2] = pwelch(c_filter,[],[],[],f_s,'centered');
PSD_filter = PSD_filter /(sum(PSD_filter)*(f_s/N_s));
figure(1)
plot(f,pow2db(S)), hold on, grid on, xlim([-155,155])
plot(psdf1,pow2db(PSD_spectrum))
plot(psdf2,pow2db(PSD_filter))
title('PSD plot of Rayleigh flat fading for both methods and theoretical')
xlabel('Frequency [Hz]'), ylabel('PSD [dB/Hz]')
legend('theoretical','filter method','spectrum method');
%% PDF and CDF
K_c = [0 1 10];
xAxis = 0:0.01:20;
% initialize to store every iteration
PDF_theory = cell(3,1);
PDF_spectrum = cell(3,1);
PDF_filter = cell(3,1);
CDF_theory = cell(3,1);
CDF_spectrum = cell(3,1);
CDF_filter = cell(3,1);
for i = 1:3
    % create the Rician distributions
    theoreticalRician = makedist('Rician','s',K_c(i),'sigma',1);
    spectrumDistribution = fitdist(abs(spectrumMethod(f_D,T_s,N_s,K_c(i))'),'Rician');
    filterDistribution = fitdist(abs(filterMethod(f_D,T_s,N_s,K_c(i))),'Rician');

    % Get the pdf for these distributions
    PDF_theory{i} = pdf(theoreticalRician,xAxis);
    PDF_spectrum{i} = pdf(spectrumDistribution,xAxis);
    PDF_filter{i} = pdf(filterDistribution,xAxis);

    % Get the cdf for these distributions
    CDF_theory{i} = cdf(theoreticalRician,xAxis);
    CDF_spectrum{i} = cdf(spectrumDistribution,xAxis);
    CDF_filter{i} = cdf(filterDistribution,xAxis);
end
% Plot the PDF
figure
subplot(3,1,1)
plot(xAxis,PDF_filter{1},xAxis,PDF_spectrum{1},xAxis,PDF_theory{1})
title(['PDF with K_c = ' num2str(K_c(1))])
legend('Filter method','Spectrum method','Theoretical')

subplot(3,1,2)
```

```
plot(xAxis,PDF_filter{2},xAxis,PDF_spectrum{2},xAxis,PDF_theory{2})
title(['PDF with K_c = ' num2str(K_c(2))])
legend('Filter method','Spectrum method','Theoretical')

subplot(3,1,3)
plot(xAxis,PDF_filter{3},xAxis,PDF_spectrum{3},xAxis,PDF_theory{3})
title(['PDF with K_c = ' num2str(K_c(3))])
legend('Filter method','Spectrum method','Theoretical')
% Plot CDF
figure()
subplot(3,1,1);
plot(xAxis,CDF_spectrum{1},xAxis,CDF_filter{1},xAxis,CDF_theory{1})
title(['CDF with K_c = ' num2str(K_c(1))])
legend('Spectrum simulation','Filter simulation','Theoretical')

subplot(3,1,2);
plot(xAxis,CDF_spectrum{2},xAxis,CDF_filter{2},xAxis,CDF_theory{2})
title(['CDF with K_c = ' num2str(K_c(2))])
legend('Spectrum simulation','Filter simulation','Theoretical')

subplot(3,1,3);
plot(xAxis,CDF_spectrum{3},xAxis,CDF_filter{3},xAxis,CDF_theory{3})
title(['CDF with K_c = ' num2str(K_c(3))])
legend('Spectrum simulation','Filter simulation','Theoretical')


%% ACF
t = (-N_s*T_s:T_s:N_s*T_s);
figure(5)
for i = 1:3
    % calculate theoretical autocorr
    ACF_theory = K_c(i)^2 + besselj(0,2*pi*f_D*t);

    % calculate simulation autocorr
    [ACF_spectrum, lags_spectrum] = xcorr(spectrumMethod(f_D,T_s,N_s,K_c(i)),'unbiased');
    [ACF_filter, lags_filter] = xcorr(filterMethod(f_D,T_s,N_s,K_c(i)),'unbiased');
    subplot(3,1,i)
    plot(t,ACF_theory,lags_spectrum.*T_s,ACF_spectrum,lags_filter.*T_s,ACF_filter)
    grid on, legend('theoretical','spectrum method','filter method','Orientation','horizontal'), legend('boxoff')
    title(['Auto-correlation of the channels with K_c = ' num2str(K_c(i))])
    xlim([-0.2 1])
end
%% 2) Create Rician fading time and frequency varying channels
% given parameters
N = 300;
M = 64;
L = [1 2 3];
K_c = [0 1 10];
fDTs = [0.1 0.005];
% get fD from the normalized fDTs
Ts=T_s; % use the same T_s as before
fD = fDTs/Ts; % this is assuming fDTs = fD*Ts

% go through every 18 combinations to create the plots
for i = 1:3
    for j = 1:3
        for k = 1:2
            l = L(i);
            switch l % depending on the amount of channels L
                case 1
                    c_spectrum = spectrumMethod(fD(k),Ts,N,K_c(j));
                    c_filter = filterMethod(fD(k),Ts,N,K_c(j))';
                    c_paddedS = padWithZeros(c_spectrum,M,N);
                    c_paddedF= padWithZeros(c_filter,M,N);
                    C_S = fft(c_paddedS,M);
                    C_F = fft(c_paddedF,M);
                    figure()
                    subplot(2,2,1)
                    mesh(0:N-1,0:M-1, abs(C_S));
```

```
                title(['mesh spectrum ' 'K_c=' num2str(K_c(j)) ' fDTs=' num2str(fDTs(k)) ' L=' num2str(L(i))]);
                subplot(2,2,2)
                surf(1:N, 0:M-1,abs(C_S), 'MeshStyle','row'), view(0,90)
                title(['surf spectrum ' 'K_c=' num2str(K_c(j)) ' fDTs=' num2str(fDTs(k)) ' L=' num2str(L(i))]);
                subplot(2,2,3)
                mesh(0:N-1,0:M-1, abs(C_F));
                title(['mesh fitler ' 'K_c=' num2str(K_c(j)) ' fDTs=' num2str(fDTs(k)) ' L=' num2str(L(i))]);
                subplot(2,2,4)
                surf(1:N, 0:M-1,abs(C_F), 'MeshStyle','row'), view(0,90)
                title(['surf filter ' 'K_c=' num2str(K_c(j)) ' fDTs=' num2str(fDTs(k)) ' L=' num2str(L(i))]);
            case 2
                c_spectrum = spectrumMethod(fD(k),Ts,N,K_c(j));
                c_spectrum(2,:) = spectrumMethod(fD(k),Ts,N,K_c(j));
                c_filter = filterMethod(fD(k),Ts,N,K_c(j))';
                c_filter(2,:) = filterMethod(fD(k),Ts,N,K_c(j))';
                c_paddedS = padWithZeros(c_spectrum,M,N);
                c_paddedF= padWithZeros(c_filter,M,N);
                C_S = fft(c_paddedS,M);
                C_F = fft(c_paddedF,M);
                figure()
                subplot(2,2,1)
                mesh(0:N-1,0:M-1, abs(C_S));
                title(['mesh spectrum ' 'K_c=' num2str(K_c(j)) ' fDTs=' num2str(fDTs(k)) ' L=' num2str(L(i))])
                subplot(2,2,2)
                surf(1:N, 0:M-1,abs(C_S), 'MeshStyle', 'row'), view(0,90)
                title(['surf spectrum ' 'K_c=' num2str(K_c(j)) ' fDTs=' num2str(fDTs(k)) ' L=' num2str(L(i))])
                subplot(2,2,3)
                mesh(0:N-1,0:M-1, abs(C_F));
                title(['mesh fitler ' 'K_c=' num2str(K_c(j)) ' fDTs=' num2str(fDTs(k)) ' L=' num2str(L(i))])
                subplot(2,2,4)
                surf(1:N, 0:M-1,abs(C_F), 'MeshStyle', 'row'), view(0,90)
                title(['surf filter ' 'K_c=' num2str(K_c(j)) ' fDTs=' num2str(fDTs(k)) ' L=' num2str(L(i))])
            case 3
                c_spectrum = spectrumMethod(fD(k),Ts,N,K_c(j));
                c_spectrum(2,:) = spectrumMethod(fD(k),Ts,N,K_c(j));
                c_spectrum(3,:) = spectrumMethod(fD(k),Ts,N,K_c(j));
```

```
c_filter = filterMethod(fD(k),Ts,N,K_c(j))';
c_filter(2,:) = filterMethod(fD(k),Ts,N,K_c(j))';
c_filter(3,:) = filterMethod(fD(k),Ts,N,K_c(j))';
c_paddedS = padWithZeros(c_spectrum,M,N);
c_paddedF= padWithZeros(c_filter,M,N);
C_S = fft(c_paddedS,M);
C_F = fft(c_paddedF,M);
figure()
subplot(2,2,1)
mesh(0:N-1,0:M-1, abs(C_S))
title(['mesh spectrum' 'K_c=' num2str(K_c(j)) ' fDTs=' num2str(fDTs(k)) ' L=' num2str(L(i))])
subplot(2,2,2)
surf(1:N, 0:M-1,abs(C_S), 'MeshStyle', 'row'), view(0,90)
title(['surf spectrum' 'K_c=' num2str(K_c(j)) ' fDTs=' num2str(fDTs(k)) ' L=' num2str(L(i))])
subplot(2,2,3)
mesh(0:N-1,0:M-1, abs(C_F));
title(['mesh fitler ' 'K_c=' num2str(K_c(j)) ' fDTs=' num2str(fDTs(k)) ' L=' num2str(L(i))])
subplot(2,2,4)
surf(1:N, 0:M-1,abs(C_F), 'MeshStyle', 'row'), view(0,90)
title(['surf filter ' 'K_c=' num2str(K_c(j)) ' fDTs=' num2str(fDTs(k)) ' L=' num2str(L(i))])
            end
        end
    end
end
```

```
function c_filter = filterMethod(f_D,T_s,N_s,K_c)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
N = N_s/10; % make sure N_s is much larger than N
% initiate g vector and store g(nTs) values in it
t = (-N*T_s:T_s:N*T_s);
g = besselj(1/4,2*pi*f_D*abs(t))./nthroot(abs(t),4);
g(t==0) = nthroot(pi*f_D,4)/gamma(5/4);

% make sure g is of unit energy
g=g/norm(g);

% generate samples from zero-mean unit-variance complex Gaussian dist, we
% must divide by sqrt(2) since adding a N(0,1) with a N(0,1) gives a N(0,2)
x = (randn(N_s,1) + 1i*randn(N_s,1))/sqrt(2);

% convolve x with g and remove the transients (can be done using 'same')
c_filter = conv(x,g,'same') + K_c; % if K_c is 0 then this is Rayleigh

end
```

```
function c_spectrum = spectrumMethod(f_D,T_s,N_s,K_c)
%UNTITLED3 Summary of this function goes here
%   Detailed explanation goes here
f_s = 1/T_s;
% create the frequency vector from origin to the edge of bandwidth
f = 0:f_s/N_s:f_D; % only consider the onesided, we flip it later to create the other side
% calculate S_c according to lab PM
S_c = (1/(pi*f_D)).*(1./sqrt(1-(f./f_D).^2));
% get G as sqrt of S_c
G = sqrt(S_c);
```

```
11 if G(end) == Inf
12     G(end) = (1/(pi*f_D)).*(1./sqrt(1-(
           f_D+eps/f_D).^2));
13 end
14 % initialize the periodic version G_p
       with zeros
15 G_p = zeros(1,N_s);
16 % store the right side of S_c in the
       beggining of G_p and the left side at
17 % the end by flipping the right side
       version. They should be equal.
18 G_p(1:length(G)) = G;
19 G_p(end-length(G)+1:end) = flip(G);
20 % Generate samples from zero-mean,
       complex gaussian dist
21 a = sqrt(N_s^2/norm(G_p)^2); % find the
       constant a (using Parseval's theorem)
22 % generate samples from a complex
       gaussian distribution and multiply
       with G
23 X = (randn(N_s,1) + 1i*randn(N_s,1))*a/
       sqrt(2);
24 C = X'.*G_p; % in time-domain this is the
        convolution with the gaussian
25 c_spectrum = ifft(C)+K_c; % if K_c is 0
       then this is Rayleigh
26 end
```

```
1 function c_padded = padWithZeros(c,M,N)
2 %UNTITLED Summary of this function goes
      here
3 %   Detailed explanation goes here
4 [L, ~] = size(c);
5 padding = zeros(M-L,N);
6 c_padded = [c; padding];
7 end
```