Wireless Communication

Project 1

Simulation of fading wireless channel

Group 17

Members:

Anton Skårbratt
antons@student.chalmers.se
Sohaib Maalik
sohaib@student.chalmers.se
Yu Cheng
chengy@student.chalmers.se

Abstract

We have simulated a Rayleigh fading wireless channel with Clarke's Doppler spectrum using two methods, filter method and spectrum method. Based on PSD, level crossing rate and average fade duration we evaluated the results obtained from the two methods. We obtained very similar results from both of them compared to the theory. The filter method is however more complex in terms of simulation time compared to the efficient spectrum method. The filter method gives better results compared to spectrum method when numbers of samples are taken very large. We have also simulated the channel for different taps and Doppler frequencies to observe time-frequency varying behaviour of a Rayleigh fading channel.

Table of Contents

1. Introduction

The aim of this project is to study and get a good understanding of wireless channel and the factors which affects the performance of such a channel. This is done by simulating a Rayleigh fading channel with Clarke's Doppler spectrum. Simulations are done in Matlab. Two different methods are used to analyze the performance of the fading channel; the filter method and the spectrum method.

2. The Filter Method

In the filter method, c(t) is considered as the output from a filter with impulse response g(t) when the incoming signal x(t) is taken as complex, white Gaussian noise with unit variance. The output of the filter will simply be the convolution of x(t) and g(t):

$$c(t) = x(t) * g(t)$$

The impulse response function of the filter is chosen as

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

where f_D is the Doppler frequency calculated as

$$f_D = \frac{v}{c_0} f_c$$

where v is the relative velocity between the transmitter and the receiver, f_c the carrier frequency and c_0 the speed of light. To compute $\hat{g}(t)$ we have chosen a time-limited window function w(n) whose value is being set equal to 1 in a finite interval, so that $\hat{g}(t)$ can be made causal and time limited. Since Matlab is being used, discrete time samples are needed. Therefore, impulse response $\hat{g}(nT_s)$ is calculated as

$$\hat{g}(nT_s) = Kg((n-N)T_s)w((n-N)T_s)$$

The factor K was chosen(was tuned such that) such that $\hat{g}(nT_s)$ has unit power. Figure 1 shows $\hat{g}(nT_s)$ for a signal with carrier frequency 2 GHz and a relative speed of 50 km/h, which gives us a Doppler frequency $f_D = 92$ Hz.

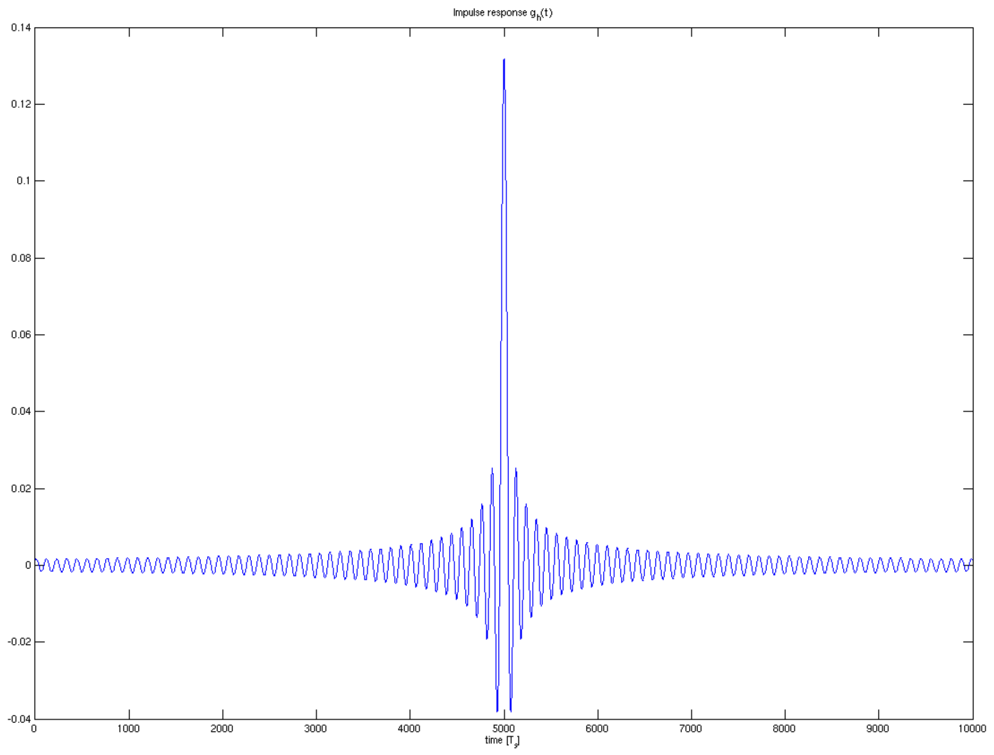

Figure 1 - Impulse response $\hat{g}(t)$

The input signal z(t) to the filter was generated with zero mean, unit variance, complex Gaussian distribution. The impulse response of the channel was finally calculated as:

$$c(nT_s) = z(nT_s) * \hat{g}(nT_s)$$

The magnitude, phase and power spectral density of the impulse response of the channel is shown in figure 2 and 3 respectively. The power spectral density of the channel was calculated by using the autocorrelation of c(t).

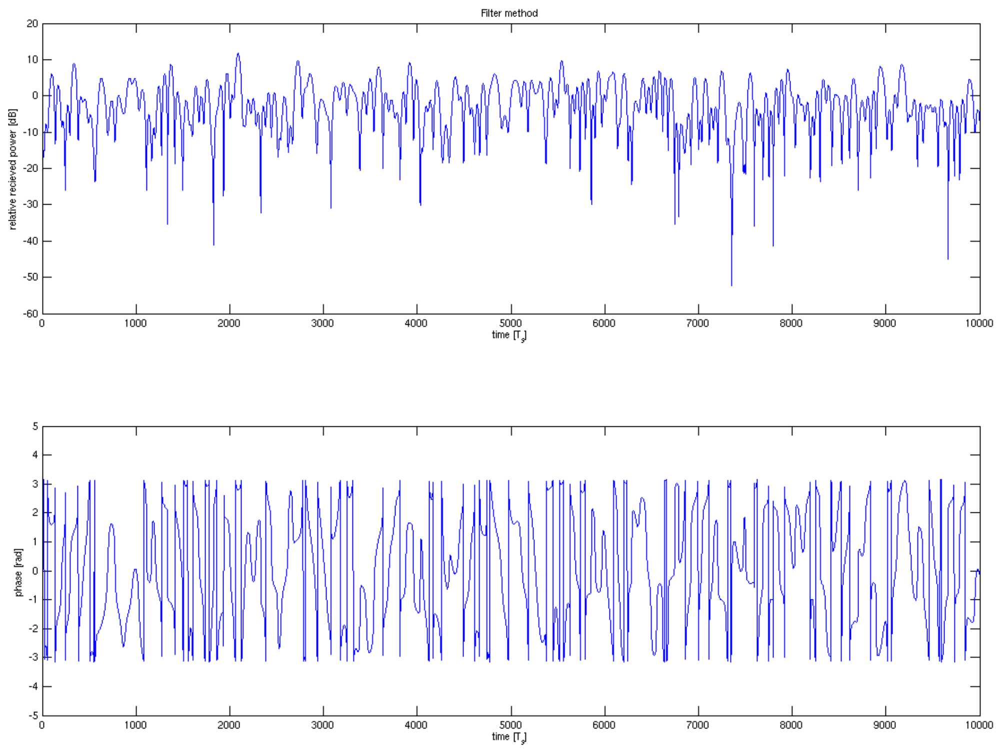

Figure 2 - Amplitude and phase of c(t)

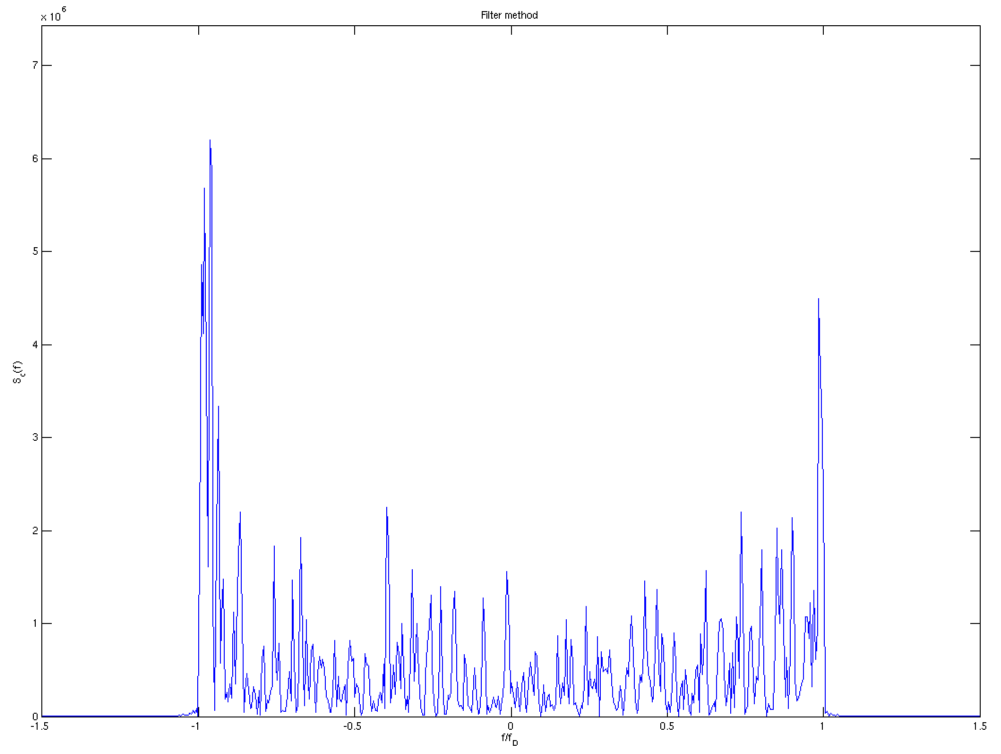

Figure 3 – Power spectral density of c(t)

3. The Spectrum Method

In the spectrum method the fading gain is sampled in the frequency domain instead of the time domain. The samples are then transformed to the time domain using an inverse FFT.

First a Clarke's Doppler spectrum was constructed with power spectral density $S_c(f)$ according to:

$$S_c(f) = \begin{cases} \dfrac{1}{\pi f_D} \dfrac{1}{\sqrt{1-(f/f_D)^2}}, & |f| \le f_D \\ 0, & \text{otherwise} \end{cases} .$$

From this spectrum$G(f)$ was easily calculated as $G(f) = \sqrt{S_c(f)}$. The function was then periodically shifted to $\tilde{G}(f) = G(f) + G(f - f_s)$ and sampled with N=10000 samples from f to f_s. This shift is illustrated in figures 4 and 5.

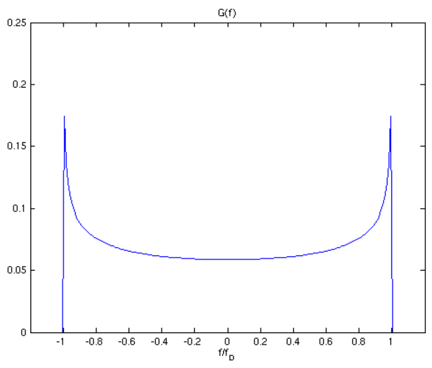

Figure 4 – $G(f)$

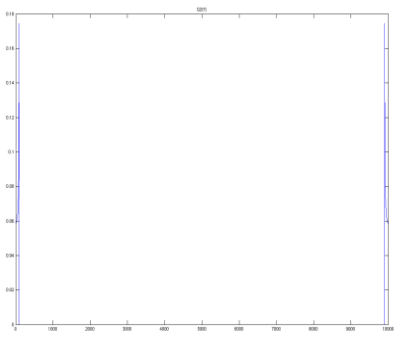

Figure 5 – $\tilde{G}(f)$

The next step was to draw N=10000 statistically independent, zero-mean, complex Gaussian random numbers with a variance that results in a unit variance impulse response c(t) at the end. The implementation of this variance was built on a trial and error approach.

Finally, c(t) was calculated according to:

$$c(nT_s) = \mathcal{F}^{-1}\left\{\tilde{G}\left(\frac{kf_s}{N}\right) \cdot Z(k)\right\}$$

where $Z(k)$ is the Gaussian noise. The resulting amplitude and phase of the impulse response is showed in figure 6. The power spectral density of the channel was calculated by using the autocorrelation of c(t), and it is showed in figure 7.

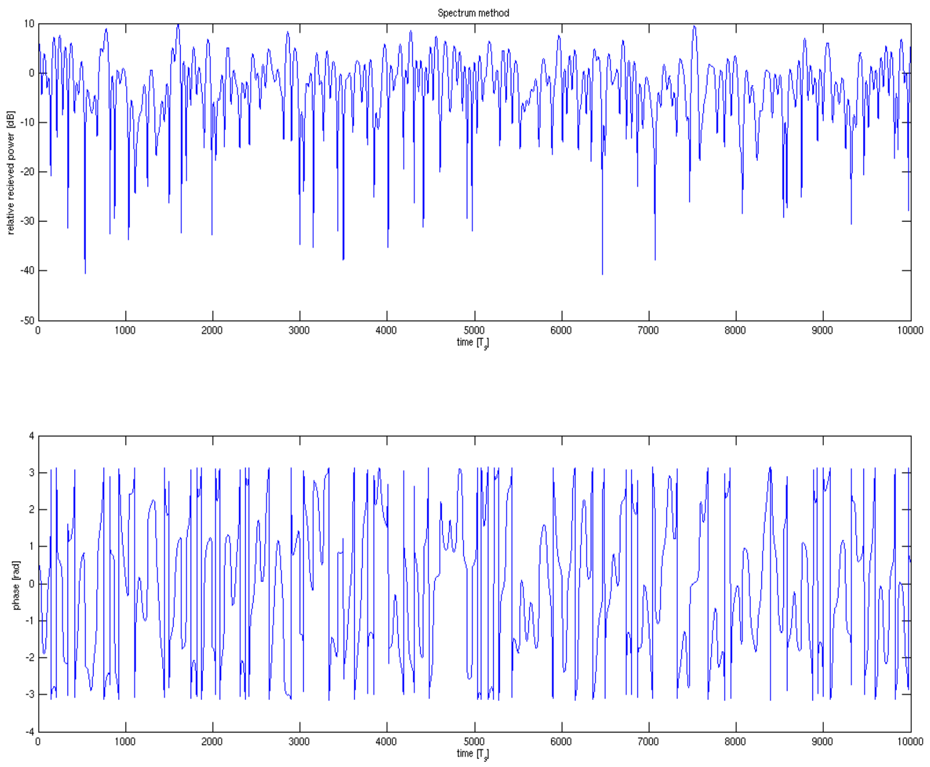

Figure 6 – Amplitude and phase of c(t)

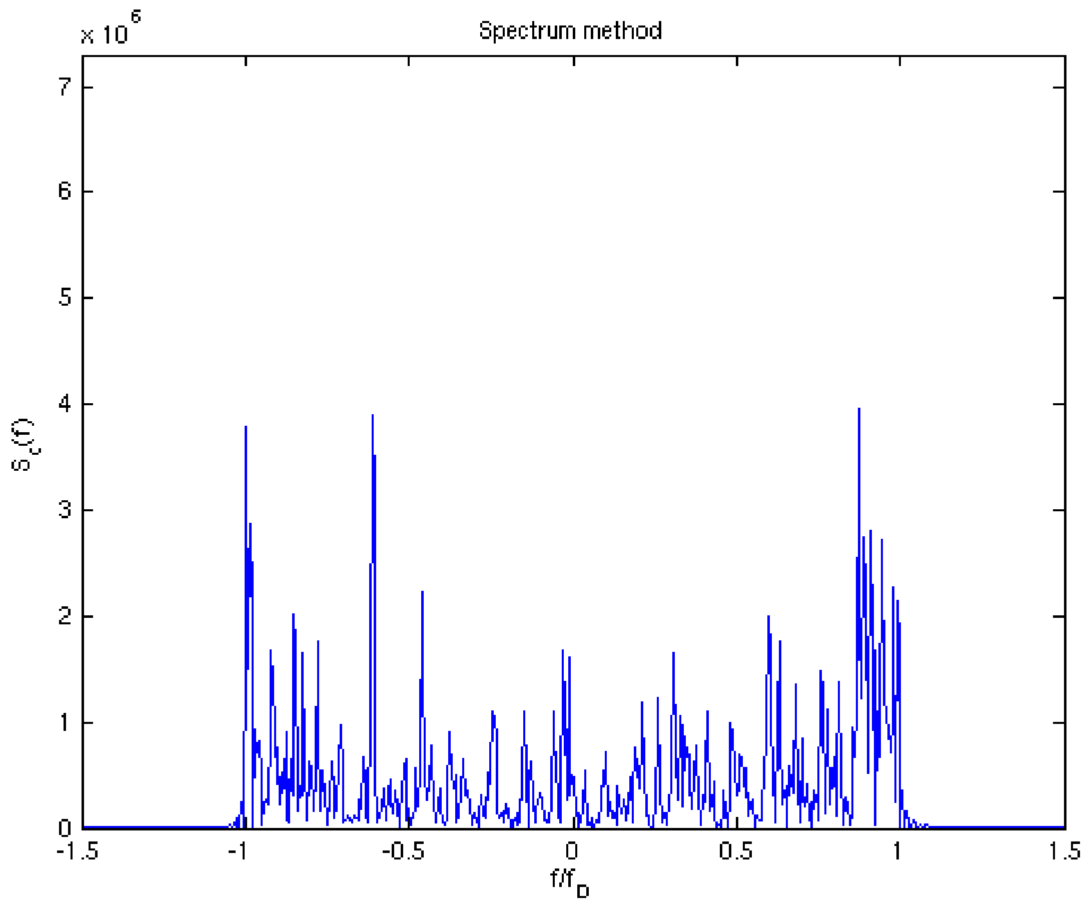

Figure 7 – Power spectral density of c(t).

4. Time and frequency-varying Channel

A study was also made on a Rayleigh fading channel with both time and frequency variations. The model we used for simulating this channel is built on L=3 taps with complex gain $c_l(t)$ where the received signal is calculated according to:

$$r(nT_s) = \sum_{l=1}^{L} c_l(nT_s)s(nT_s - \tau_l T_s)$$

The response C(f,nT_s) for such a channel is just the amplitude scaling a complex exponential with frequency f experiences when it is transmitted over the channel. To compute the time-varying frequency response we used the equation

$$C(k/(NT_s), nT_s) = \sum_{l=1}^{L} c_l(nT_s)\exp\left(-j2\pi\frac{k}{NT_s}\tau_l T_s\right)$$

where $c_l(nT_s)$ was calculated using the filter method. A plot of the response was simulated for each and one of the nine combinations for L=1,2,3 and $f_D T_s = 0.0005, 0.005, 0.05$. The behaviour of the time and frequency response of the channel for a specific case can be observed in figure 8. The rest is attached in Appendix B.

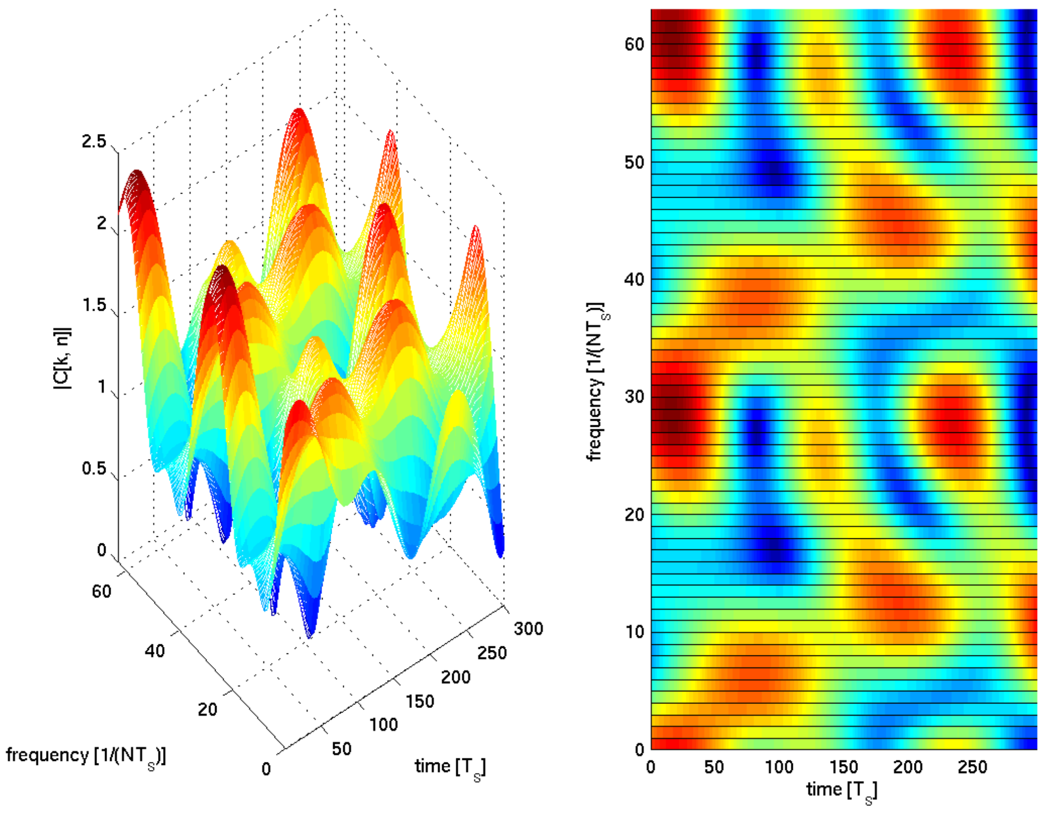

Figure 8 - Time and frequency varying channel response for L=3 and $f_D T_s = 0.005$

5. Discussion

5.1 Channel characteristics for the filter- and spectrum method

After simulating channel responses with the two methods we compared the amplitude and phase of c(t) with theory to roughly verify that they are valid. The characteristics of these plots matched the theory quite well. A more detailed analysis was made based on autocorrelation, PSD, level crossing rate, average fade duration, complexity etc.

5.1.1 Autocorrelation and power spectral density

The PSD of the channel impulse response $S_c(f)$ with the filter and spectrum method was plotted by taking Fourier transform of its autocorrelation function. The results obtained from the two methods are quite similar. By comparing the PSD and autocorrelation plots of the channel obtained from the two methods we couldn't say everything about the channel characteristics. Therefore, level crossing rate and average fade duration was calculated through simulation and theoretically as well.

5.1.2 Level crossing rate

Another parameter we used to verify the characteristics of the channel was the level crossing rate. Level crossing rate was calculated by setting different threshold levels. The values are obtained for 20,000 samples and running the program for 5 times to average it out. Table 1 reveals that both methods generate a channel with quite similar response to what theory predicts.

	LEVEL CROSSING RATE [crossings/s]		
Threshold level for abs(c)	**Filter method**	**Spectrum method**	**Theory**
0.9	92	94	93
0.7	101	103	100
0.5	90	90	90
0.3	65	65	64
0.1	21	25	23

Table 1 – Level crossing rate

5.1.3 Average fade duration

As a complement to the level crossing rate the average fade duration was calculated for both channels, see table 2. Also here the two methods generate similar results.

	AVERAGE FADE DURATION [ms]		
Threshold level for abs(c)	**Filter method**	**Spectrum method**	**Theory**
0.9	6.1	5.9	6.0
0.7	3.9	3.7	3.9
0.5	2.5	2.4	2.4
0.3	1.3	1.3	1.4
0.1	0.45	0.39	0.43

Table 2 – Average fade duration

5.1.4 Advantages and disadvantages

As one can note above from the parameters, both methods generates a channel response close to the theory. The filter method is however a bit more complex and it takes more time to simulate a channel. This was tested by increasing the number of samples while noting the time for simulation. But it gives better results compared to spectrum method when number of samples is taken very large.

By increasing the length of the input signal Ns the result would definitely be better since the input is a random process and we are going to estimate the channel which is a random process itself. In other words: the more the Ns is, the better the channel estimation would be. But the drawback of this is that by increasing Ns the simulation will take longer time. The other important factor is the length of the filter N which is a Bessel function. By getting more samples of the Bessel function with truncation method the filter would be better estimated so the result would be better.

5.2 Time and frequency-varying channel

If we have only one tap, i.e. L=1, our channel will have less delay spread, but on the other hand the channel performance will suffer. By increasing the number of taps (L=2, L=3) delay spread of the channel will increase but at the same time channel performance will get better. There is always a tradeoff between channel performance and complexity.

The factor $f_D T_s$ shows how fastly a channel is varying in time. At low value of $f_D T_s$ the channel will vary very slowly, where as for higher values of $f_D T_s$ channel variations will increase as can be seen from the plots appended in Appendix B.

When we have one tap L=1 and normalized Doppler frequency $f_D T_s$=0.0005 our channel will behave like slow, flat fading channel which is quite obvious from the plot in Appendix B. By increasing the Doppler frequency to $f_D T_s$=0.005, 0.05 channel variations will also increase.

Appendix A – MATLAB code

Filter_method.m

```
function c2 = filter_method(fDTs)

%------ Specified parameters
vkmh=50;
vms=vkmh/3.6;
f0=2*10^(9);
c=3*10^(8);
Ts=0.1*10^(-3);
Fs=1/Ts;
%fD=fDTs/Ts;

fD=vms*f0/c
%Ts=0.1*10^(-3);
P=10000;
%N=nsamples;
%------ ------ ------ ------

%------ Calculates impulse response g(nTs)
for n=1:(2*P+1)
    w((n))=1;              % The window function w(nTs)
    if n-P-1 == 0;
        g(n) = (pi*fD)^(1/4)/(gamma(5/4));
    end
    if n-P-1 ~= 0;
        g(n) = besselj(1/4,( 2*pi*fD*abs( (n-P-1)*Ts ) ) ) / (abs((n-P-
1)*Ts))^(1/4);
    end
end
%------ ------ ------ ------

%------ Calculates K
sum=0;
for n=1:(2*P+1)
    gcomplex1(n) = g(n)*w(n);
    sum=sum+(gcomplex1(n))^2;
end
K=sqrt(1/sum);
%------ ------ ------ ------

%------ Calulates the new truncated and delayed impulse response g(nTs)
with K.
for n=1:(2*P+1)
    gcomplex2(n) = K*g(n)*w(n);
end



%------ ------ ------ ------

%------ Generates N statistically independent, zero-mean, complex Gaussian
random
%       numbers with unit variance.
z = (randn(2*P+1, 1) + sqrt(-1)*randn(2*P+1, 1))/sqrt(2);

%------ ------ ------ ------
```

```
c=conv(z,gcomplex2); % Calculates the channel gain c(t)

%------ Cuts the transients, but it is the N first and N last?
x=0;
for i=P:length(c)-P
    x=x+1;
    c2(x)=c(i);
end
%------ ------ ------ ------

%------ %------ %------ %------ %------ %------ NOT FOR THE FUNCTION %-----
- %------ %------ %------ %------ %------

%------ Just a x-axis vector for the plot


for i=1:length(c2)
   x1(i)=i;
end
%------ ------ ------ ------

%------ PLOTS
% figure()
% plot(gcomplex2)
% axis([0 10000 -0.04 0.14])
% title('Impulse response g_h(t)')
% xlabel('time [T_s]')
%
% figure()
% subplot(2,1,1)
% plot(x1,10*log(abs(c2)))
% axis([0 10000 -60 20])
% title('Filter method')
% ylabel('relative recieved power [dB]')
% xlabel('time [T_s]')
% subplot(2,1,2)
% plot(x1,angle(c2))
% axis([0 10000 -5 5])
% ylabel('phase [rad]')
% xlabel('time [T_s]')
% figure()

% %------ ------ ------ ------

%----- Extra for plotting Gspectrum afterwards.
% G=fft(xcorr(c2));
% Gspectrum=abs(fftshift(G));
%
% for i=1:length(Gspectrum)
%     freq(i)=(i-length(Gspectrum)/2)*Fs/(length(Gspectrum)*fD);
% end


% plot(freq,Gspectrum)
% axis([-1.5 1.5 0 1.2*max(Gspectrum)])
% xlabel('f/f_D')
% ylabel('S_c(f)')
% title('Filter method')
```

```
%------ ------ ------ ------

%------ This is code to check that the K-value normalize it correctly
  %  sum2=0;
 %   for n=1:2*N
  %       sum2=sum2+(gcomplex2(n))^2;
  %  end
   % thesumthatshouldbe1=sum2;
    thevarianceofc=var(c2)
%------ ------ ------ ------

    %time(n)= (n-P-1)*Ts; % A time-vector used for the plots, created in
    %first loop.
    %time2(n) = (P-1)*Ts; % A time-vector used for the plots




%------ For level crossing rate and average fade duration
H=abs(c2);
Lcross=0;
Level=0.1;
for i=1:length(H)-1
    if H(i)>=Level && H(i+1)<Level
        Lcross=Lcross+1;
    end
end

r=0;

for i=1:length(H)
    if H(i)<Level
        r=r+1;
    end
end


Ltime=r/(Lcross*10000); %cause Ts=0.1 ms.. we divid with 10000
Lcross=Lcross*10000/length(H)
Ltime


%Lc=sqrt(2*pi)*fD*Level/sqrt(2)*exp(-(Level)^2/2)
Lc=sqrt(2*pi)*fD*Level*exp(-(Level)^2)


Lt=(1/( Level*fD*sqrt(2*pi) ) ) * ( exp(Level^2) - 1)
length(H)
meanofc=mean(abs(c2))

% ------ ------ ------ ------
```

Spectrum_method.m

```
function c = spectrum_method(fDTs)
%------ Specified parameters
vkmh=50;
vms=vkmh/3.6;
f0=2*10^(9);
c=3*10^(8);
Ts=0.1*10^(-3);
fD=vms*f0/c;
Fs=1/Ts;
N=10000;
%------ ------ ------ ------
%------ Calculates G(f)
for f=1:2*N+1
    freq1(f)=(Fs*(f-N-1)/(2*N));
    if abs(Fs*(f-N-1)/(2*N))<= abs(fD)
        Sc(f)= ( 1/(pi*fD) ) * 1/sqrt(1-((Fs*(f-N-1)/(2*N))/fD)^2);
    else
        Sc(f)=0;
    end
    G(f)=sqrt(Sc(f));

end
%------ ------ ------ ------
% ---- Calculates G~ (Gny)
for f=N+1:1:2*N % Note that freq1(N+1)=0, so index N+1 corresponds to f=0
    Gny(f-N)=G(f);
end

for f=1:1:N
    Gny(f+N)=G(f);
end

Ns=length(Gny)
for i=1:Ns
    freq2(i)=Fs*(i-1)/(Ns);
end
%------ ------ ------ ------
%------ Generates Ns statistically independent, zero-mean, complex Gaussian
random number with unit variance.
z = (randn(Ns, 1) + sqrt(-1)*randn(Ns, 1))/sqrt(2);
%------ ------ ------ ------
thevarianceofZ=var(z);

%------ To fix variance of c(t)
K=sqrt(var(Gny)); % K fixes the variance in the end, but if this is used
after ifft, it will be exact, now it is random

for i=1:Ns
   test(i)=K*Gny(i)*z(i);
end
%------ ------ ------ ------

c=ifft(test,Ns); % The inverse Fourier transform
c=c/sqrt(var(c));
```

```
%------ Just a x-axis vector for the plot
for i=1:length(c)
   x1(i)=i;
end
%------ ------ ------ ------


% % %------ PLOTS
%  figure()
%  plot(freq2,Gny,'-')
% title('G2(f)')
% figure()
% plot(freq1/fD,G,'-')
% title('G(f)')
% xlabel('f/f_D')
%
% axis([-1.2 1.2 0 0.25]) % 0.25 when N=50000.0014
%  figure()
%  subplot(2,1,1)
%  plot( x1,10*log(abs(c)),'-')
%  title('Spectrum method')
%  ylabel('relative recieved power [dB]')
%  xlabel('time [T_s]')
%   subplot(2,1,2)
%  plot(x1,angle(c))
%  ylabel('phase [rad]')
%  xlabel('time [T_s]')
% %------ ------ ------ ------
% % %----- Extra for plotting Gspectrum afterwards.
% G=fft(xcorr(c));
% Gspectrum=abs(fftshift(G));
%
% for i=1:length(Gspectrum)
%     freq(i)=(i-length(Gspectrum)/2-1)*Fs/(length(Gspectrum)*fD);
% end
%
% figure()
% plot(freq,Gspectrum)
% axis([-1.5 1.5 0 0.25*max(Gspectrum)])
% xlabel('f/f_D')
% ylabel('S_c(f)')
% title('Spectrum method')
% %------ ------ ------ ------
% %plot(abs(xcorr(c)))
% thevarianceofc=var(c);
% themeanofc=mean(abs(c));
% fD;
%------ For level crossing rate & Average fade duration
H=abs(c);
Lcross=0;
Level=0.1;
for i=1:length(H)-1

    if H(i)>=Level && H(i+1)<Level
        Lcross=Lcross+1;
    end


end
```

```
r=0;

for i=1:length(H)
    if H(i)<Level
        r=r+1;
    end
end

 Ltime=r/(Lcross*10000);

Lcross=Lcross*10000/length(H)
Ltime


%Lc=sqrt(2*pi)*fD*Level/sqrt(2)*exp(-(Level)^2/2)
Lc=sqrt(2*pi)*fD*Level*exp(-(Level)^2)


Lt=(1/( Level*fD*sqrt(2*pi) ) ) * ( exp(Level^2) - 1)

langd=length(H)
%------
```

timeandfreq.m

```
clear all
clc

M=300; % Time samples
N=64; % Frequency samples
fDTs=[0.0005 0.005 0.05];
tap=[0 2 4];

i=3;
C=zeros(5,M);
c3=zeros(5,M);

for l=1:2:5
      c2(l,:)=spectrum_method(fDTs(i));

    for s=1:M % Selecting M samples
    c3(l,s)=c2(l,s);
    end

end

C=fft(c3,N);

f=figure(1)
colormap('jet')
subplot(1, 2, 1)
mesh(1:M, 0:N-1, abs(C));
ylabel('frequency [1/(NT_s)]')
xlabel('time [T_s]')
zlabel('|C[k, n]|')
set(gca, 'Ylim', [0 N-1], 'Xlim', [1 M])

subplot(1, 2, 2)
surf(1:M, 0:N-1, abs(C), 'MeshStyle', 'row')
ylabel('frequency [1/(NT_s)]')
xlabel('time [T_s]')
```

```
set(gca, 'Ylim', [0 N-1], 'Xlim', [0 M-1])
view(2)

saveas(f,'spectrum_l3i3.png')
close(f)
```

Appendix B – Time and frequency varying graphs

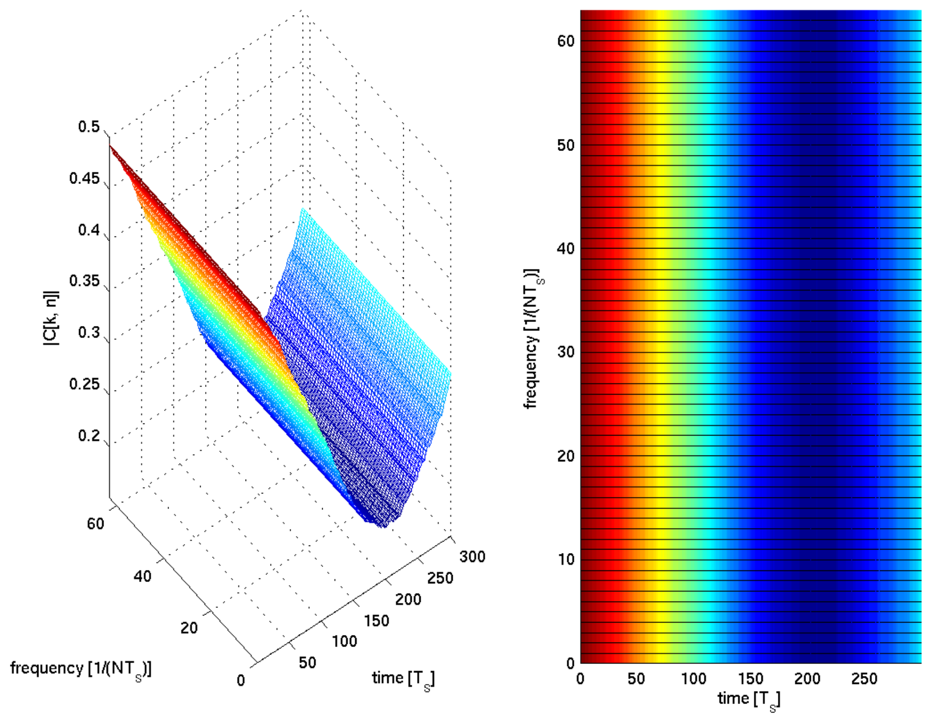

Figure B1 – Channel response with L=1, $f_D T_s = 0.0005$

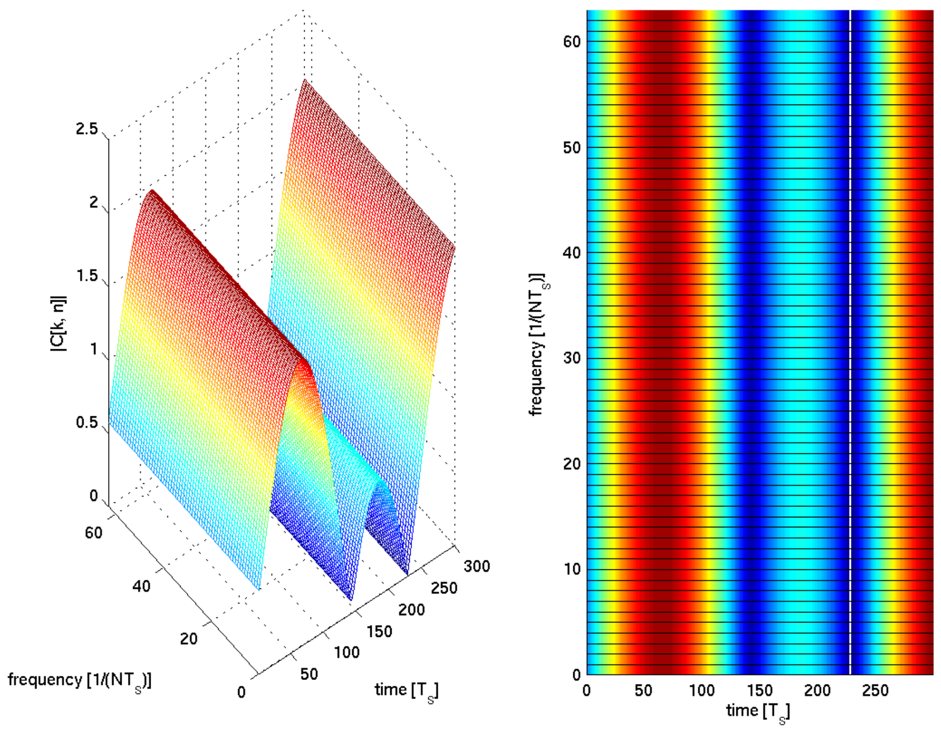

Figure B2 – Channel response with L=1, $f_D T_s = 0.005$

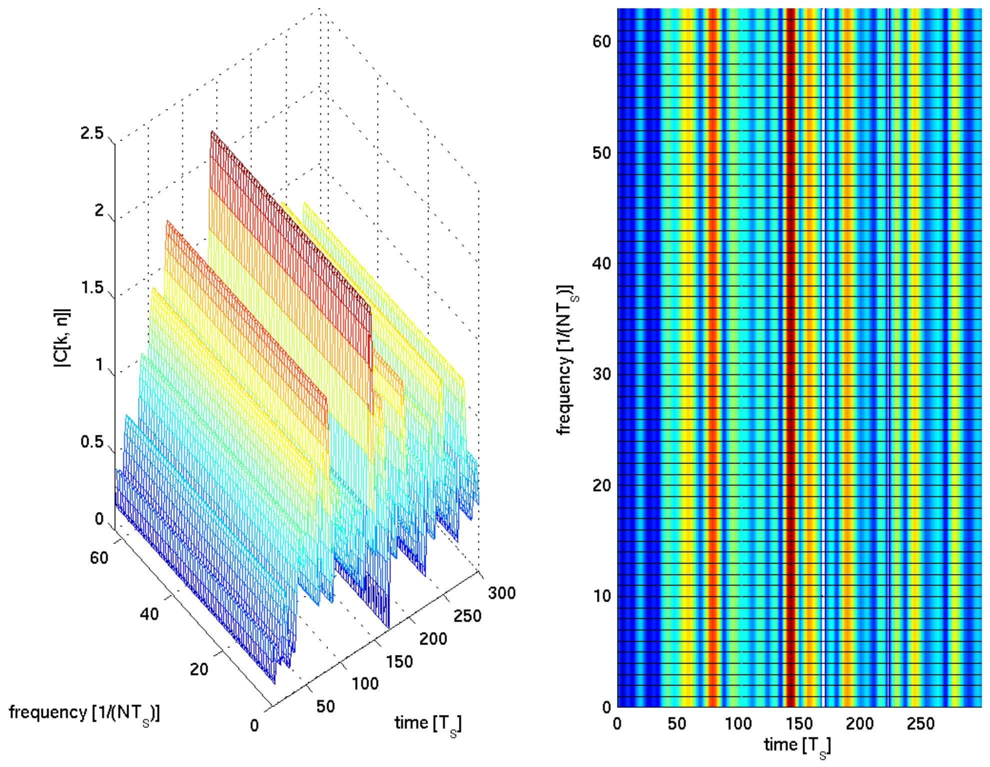

Figure B3 – Channel response with L=1, $f_D T_s = 0.05$

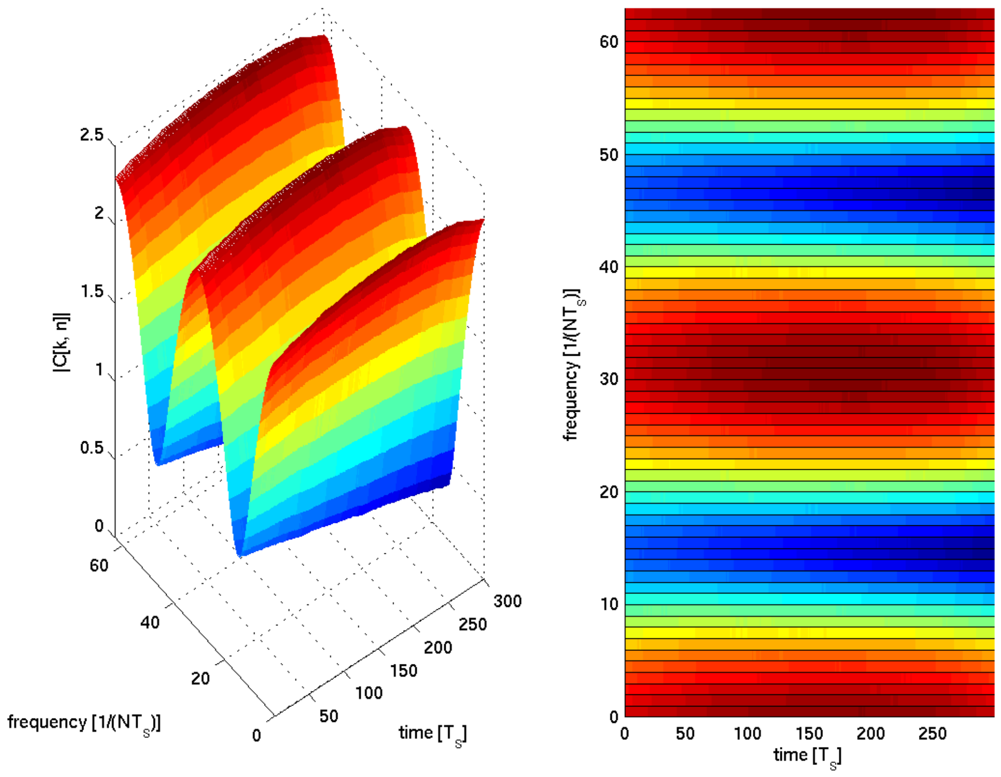

Figure B4 – Channel response with L=2, $f_D T_s = 0.0005$

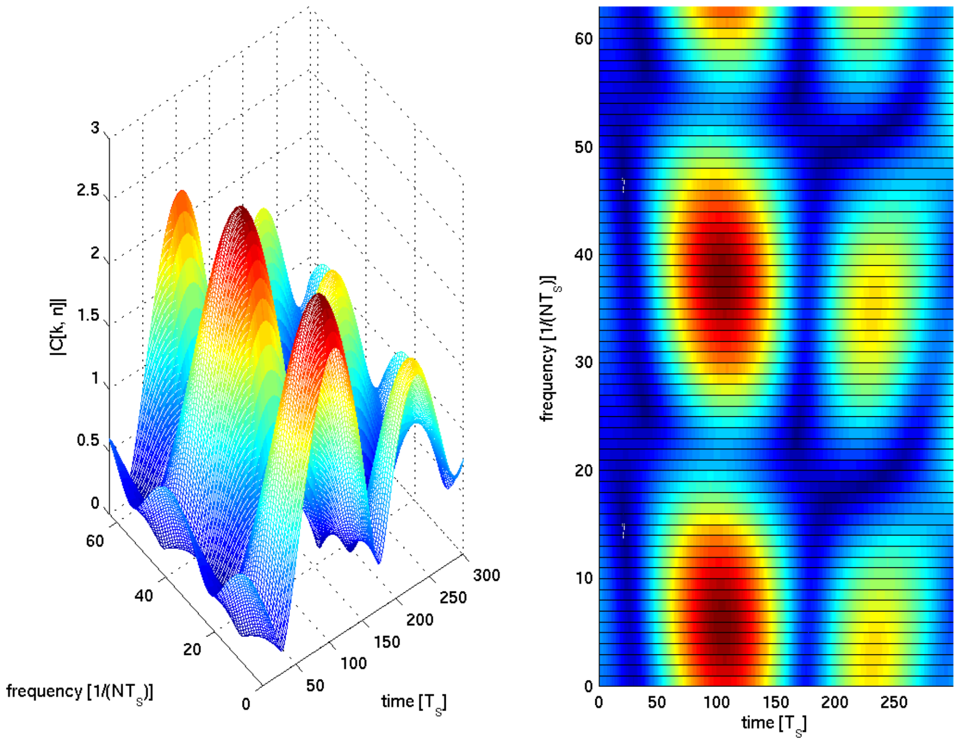

Figure B5 – Channel response with L=2, $f_D T_s = 0.005$

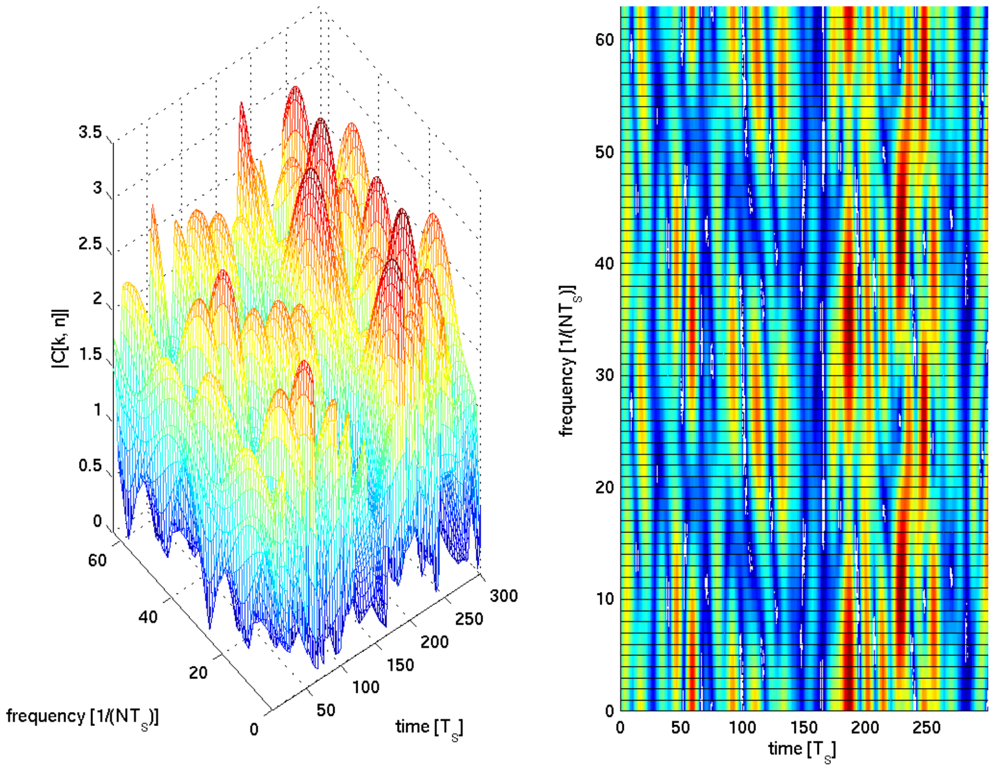

Figure B6 – Channel response with L=2, $f_D T_s = 0.05$

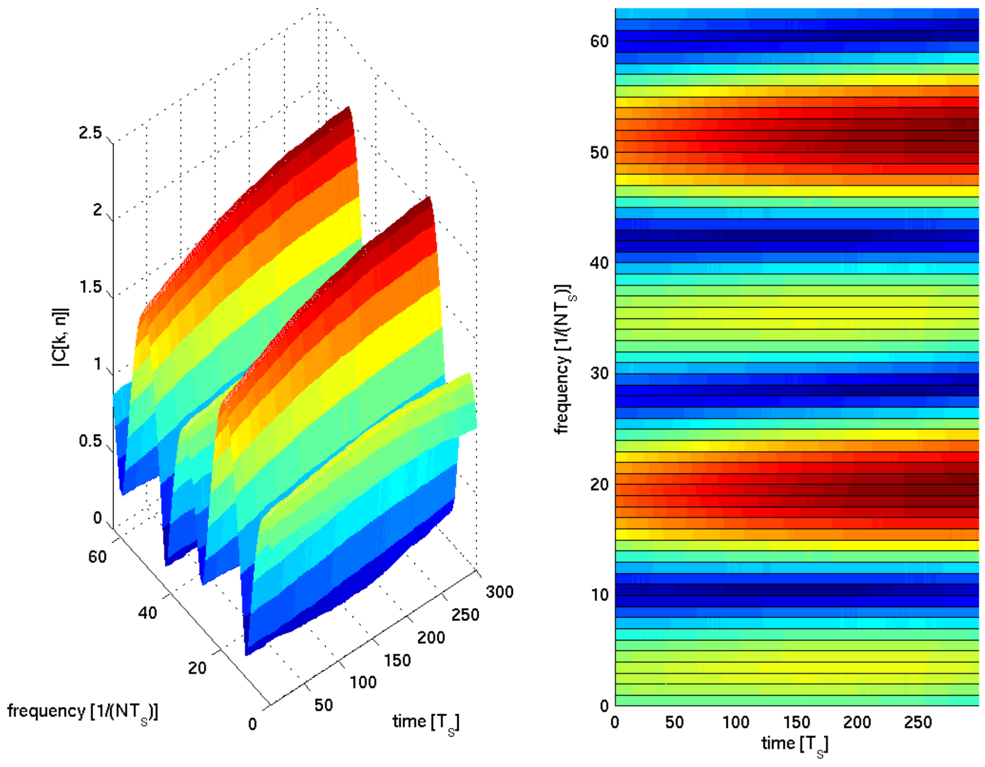

Figure B7 – Channel response with L=3, $f_D T_s = 0.0005$

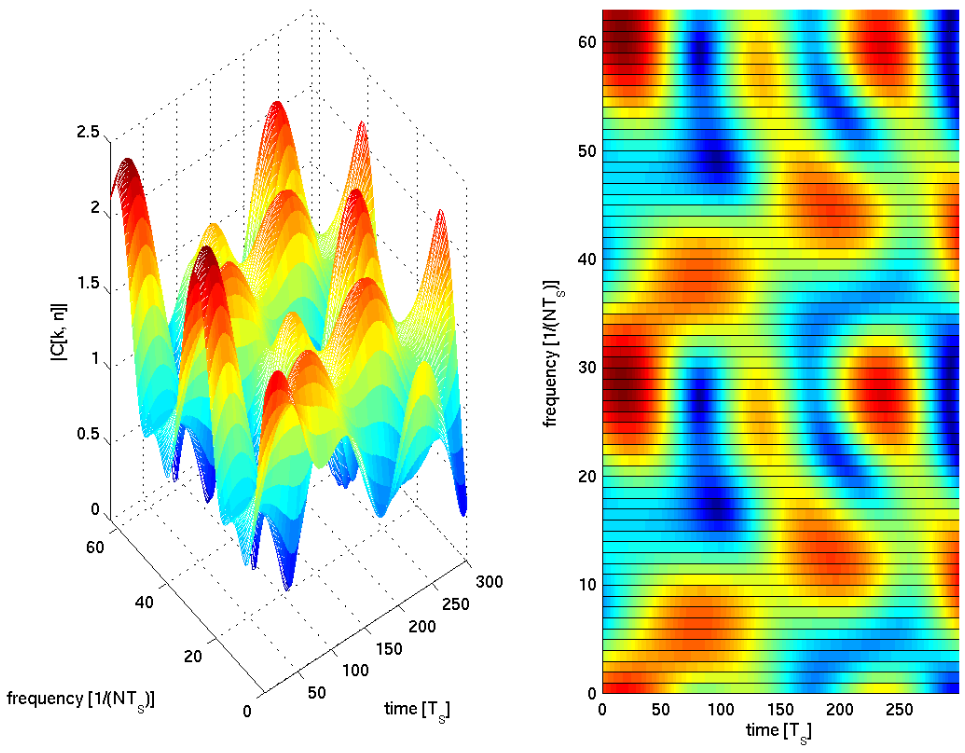

Figure B8 – Channel response with L=3, $f_D T_s = 0.005$

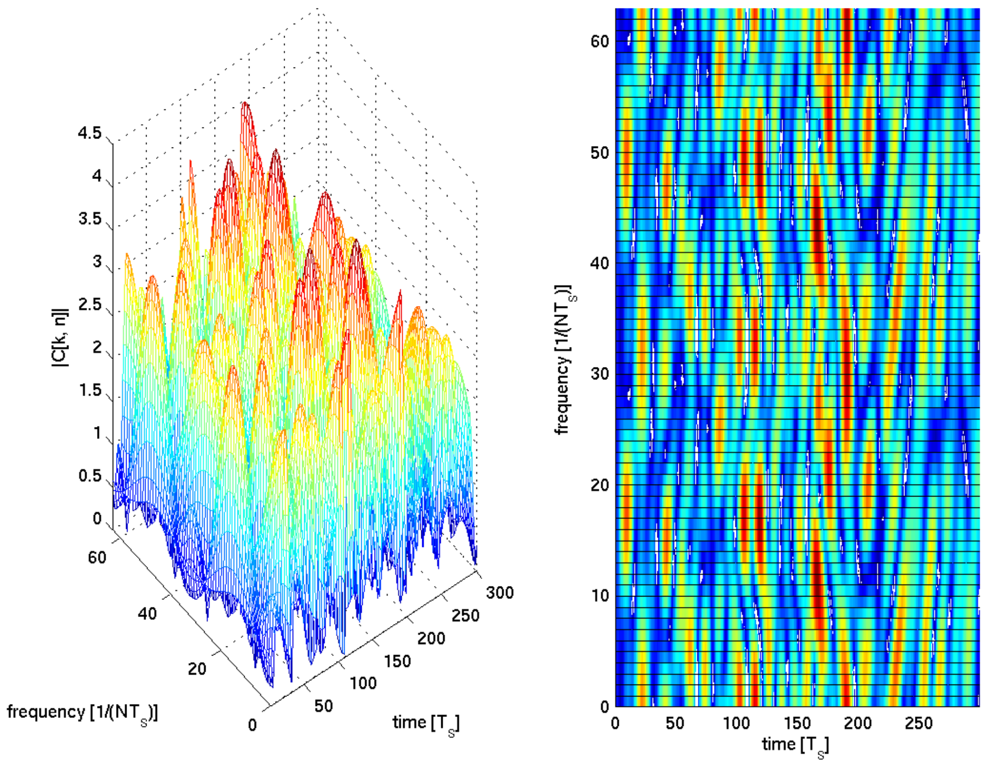

Figure B9 – Channel response with L=3, $f_D T_s = 0.05$

References

1-Wireless communications by Andrea Goldsmith
2-Simulation of fading wireless channel By Erik Ström, Nima Seifi