TE 412 TERM PROJECT

WSSUS-CHANNEL MODELS for BROADBAND MOBILE COMMUNICATION SYSTEMS

SPRING 2004

by

Çiğdem Altay
5645

Onur Sarkan
5241

Introduction:

In order to compensate the need for high data rates and large bandwidths, new communications systems have been developed and thus, in order to define these channels new propagation models have been established. In this project, we have used MATLAB to simulate and then to analyze the performance of broadband mobile communication systems using WSSUS (Wide Sense Stationary Uncorrelated Scattering) model.

The difference of WSSUS from the previously used models, such as COST 207 is that WSSUS is a model that can define high bandwidth and high data rate channels such as UMTS and DVB-T that have a bandwidth of 5MHz with a data rate of 2Mbits/sec per user [1]. As will be explained in later sections of this report, the data rates that we have mentioned are mainly affected by the service environment and the mobility characteristics of the moving station [2]. Therefore, due to these changing characteristics, the WSSUS channel has to be treated as a time-variant, multipath-fading channel [1].

In this project, we have mainly used the channel propagation model presented by [1] and WSSUS theory presented in [4]. The resulting simulation of the channel has then been analyzed by plotting a BER vs. SNR graph.

Channel Response of WSSUS Propagation Models:

The channel response of wireless broadband channels is mainly affected by the noise in the atmosphere and the paths that are subject to the environmental spectrum. Due to this fact, WSSUS propagation models are defined and characterized by two variables,

namely the path delay (τ) and the Doppler frequency (f_D). With these two variables, the scattering function $S(\tau, f_D)$ is formed, which is defined to be a function characterizing the angular distribution of scattered radiation in terms of the scattering angle. By using the Scattering Function, and integrating it with respect to the path delay or the Doppler frequency variables, we reach two other important functions, respectively the Power Delay Profile *(PDP)* and the Doppler Power Spectrum (*DPS)*, which describe the propagation effects.

$$PDP(\tau) = \int_{-f_{Dmax}}^{f_{Dmax}} S(\tau, f_D)\, df_D \qquad (1)$$

$$DPS(f_D) = \int_{0}^{\tau_{max}} S(\tau, f_D)\, d\tau \qquad (2)$$

Below in Figure 1 are examples of the above mentioned Scattering Function, Power Delay Profile and Doppler Power Spectrum.

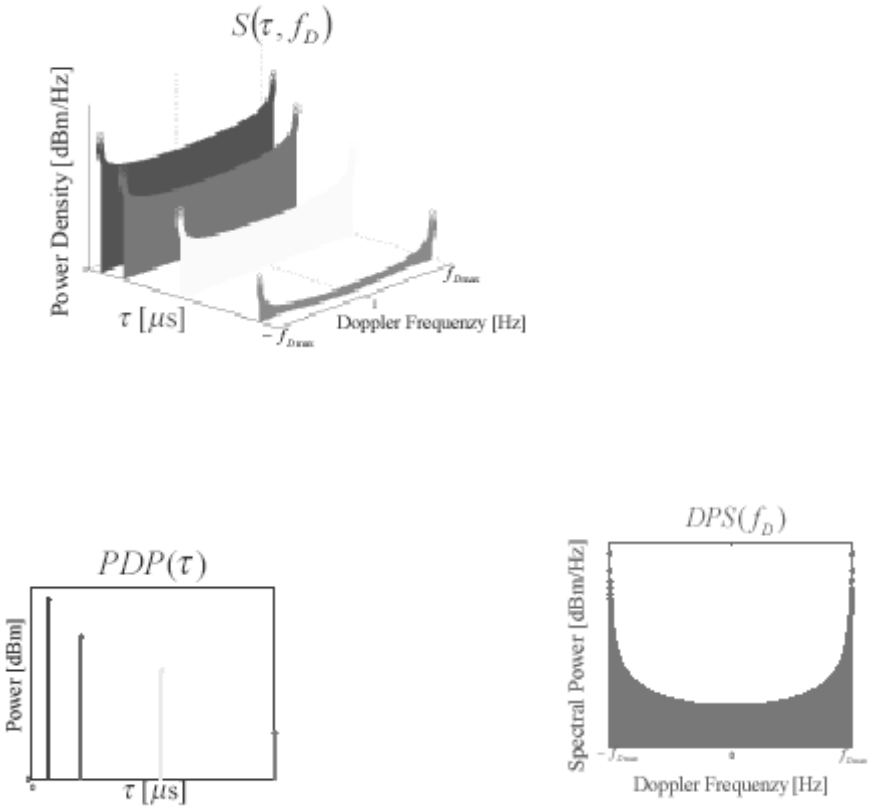


Figure 1: Example of an idealized Scattering Function with its Power Delay Profile and its Doppler Power Spectrum.

As the WSSUS-channel model is characterized by being wide sense stationary and has uncorrelated scattering as the name indicates, the autocorrelation function of the channel response is a function of only the time difference in the time domain and is only a frequency difference in the frequency domain. Thus, the autocorrelation function can be defined as:

$$R_{HH}(\Delta f, \Delta t) = \tfrac{1}{2} E\{H^*(f,t) \cdot H(f + \Delta f, t + \Delta t)\} \qquad (3)$$

and is called the spaced frequency, spaced time correlation function [1] where,

$$H(f,t) = h(\tau,t) \cdot exp\ (-j2\pi f\tau)\ d\tau$$

In the above equation, it is important to note that $h(\tau,t)$ is the time-variant equivalent lowpass response of the mobile channel.

The above defined spaced frequency, spaced time function and the scattering function are related by the following equation:

$$S(\tau, f_D) = \int_0^{\infty}\int_0^{\infty} r_{HH}(\Delta f,\Delta t) \cdot e^{j2\pi\Delta f\tau}\, e^{j2\pi f D\Delta t}\, d(\Delta f)d(\Delta t) \qquad (4)$$

In order to model the WSSUS channel in MATLAB, we have used the Typical K table in [1] which states the dedicated path delay τ, average power P (summation of the deterministic and scattered parts), Rice factor K which is the result of the summation of all the deterministic parts of the scattering function divided by the summation of all the scattered parts of the scattering function, Doppler shift f_{Dshift} / f_{Dmax} (f_{Dshift} is the Doppler frequency and f_{Dmax} is the maximum Doppler frequency), and complex AR-filter coefficients to generate the Doppler power spectra for each tap.

The z-transform of the transfer function is then defined to be:

$$H(z) = \frac{1}{A(z)} = \frac{1}{1 + \sum a_v z^{-v}} \qquad (5)$$

Table 1: Typical Suburban

#	ד[μs]	P[dB]	K[dB]	fDshift/ fDmax	AR-Coeff. (a1…a8)	
1	0.00	0.0	-7.2	0.18	0.1466-0.0785i	0.2012-0.0619i
					0.1977-0.0145i	0.1907-0.0187i
					0.1479-0.0295i	0.1407-0.0032i
					0.1075+0.0093i	-0.0043+0.0108i
2	0.58	-8.6	-4.4	-0.11	-0.1182-0.0043i	0.0130-0.0345i
					0.1970+0.0120i	0.1956+0.0321i
					0.1075+0.0303i	0.1117+0.0426i
					0.1258-0.0099i	0.0730-0.0194i
3	1.62	-8.8	-3.0	-0.11	-0.1182-0.0043i	0.0130-0.0345i
					0.1970+0.0120i	0.1956+0.0321i
					0.1075+0.0303i	0.1117+0.0426i
					0.1258-0.0099i	0.0730-0.0194i
4	2.19	-6.6	-1.2	-0.27	-0.1095+0.0193i	0.0541+0.0267i
					0.1813-0.0044i	0.1549+0.0028i
					0.1407+0.0282i	0.1293+0.0234i
					0.1127-0.0122i	0.0753+0.0198i

Table 2: Typical Urban

#	ד[µs]	P[dB]	K[dB]	fDshift/ fDmax	AR-Coeff. (a1…a8)	
1	0.00	0.0	-6.1	-0.03	-0.0394-0.0131i	0.1924+0.0054i
					0.2030+0.0188i	0.1820+0.0168i
					0.1705+0.0048i	0.1164-0.0022i
					0.1122-0.0118i	0.0658+0.0166i
2	0.35	-4.3	-7.3	-0.03	-0.0394-0.0131i	0.1924+0.0054i
					0.2030+0.0188i	0.1820+0.0168i
					0.1705+0.0048i	0.1164-0.0022i
					0.1122-0.0118i	0.0658+0.0166i
3	0.86	-5.1	-4.0	-0.35	-0.1461-0.0046i	0.2111+0.0030i
					0.1715+0.0296i	0.1524+0.0142i
					0.1546+0.0155i	0.1079+0.0118i
					0.0859-0.0156i	0.0657-0.0286i
4	1.31	-5.2	-3.0	-0.03	-0.1992-0.0461i	0.1982-0.0440i
					0.1887+0.0013i	0.2075+0.0189i
					0.1094+0.0405i	0.1560+0.0077i
					0.1180+0.0369i	0.0725-0.0159i
5	1.71	-5.4	-3.0	-0.03	-0.1992-0.0461i	0.1982-0.0440i
					0.1887+0.0013i	0.2075+0.0189i
					0.1094+0.0405i	0.1560+0.0077i
					0.1180+0.0369i	0.0725-0.0159i

Channel Model in the Simulation

The channel model that we have used in the simulation can be seen in the following two figures. The first block diagram summarizes how the filter coefficients are defined. The second diagram on the other hand shows how these coefficients are used in the FIR filter designed.

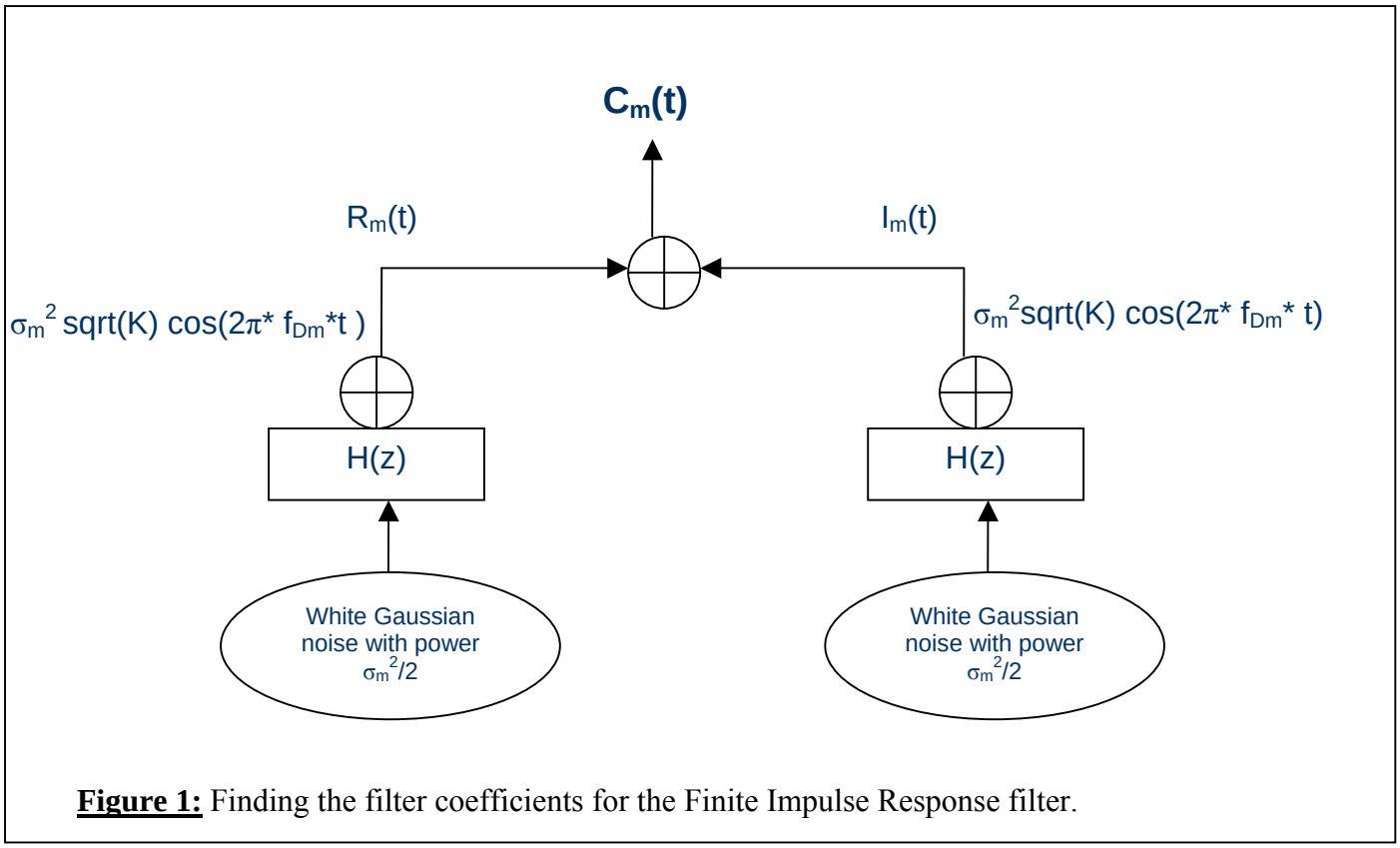


Figure 1: Finding the filter coefficients for the Finite Impulse Response filter.

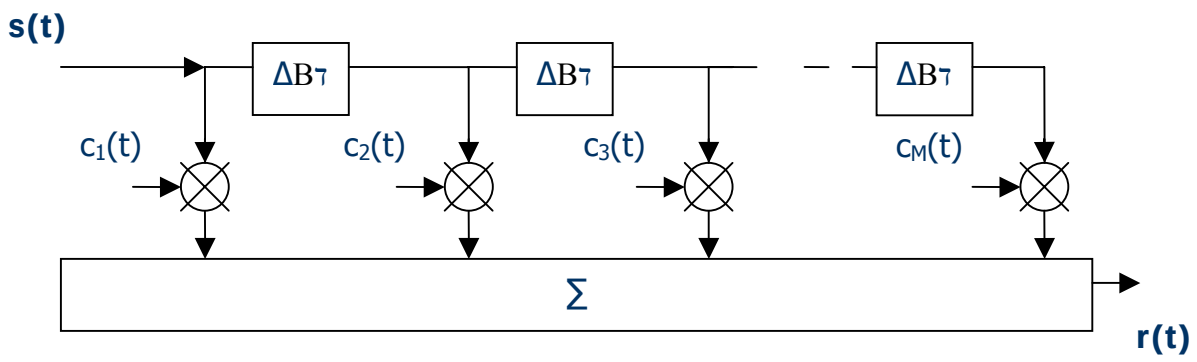


Figure 2: Modeling the mobile multipath fading channel using Finite Impulse Response filter.

Using the above defined channel model, we have reached the below graphs as channel responses:

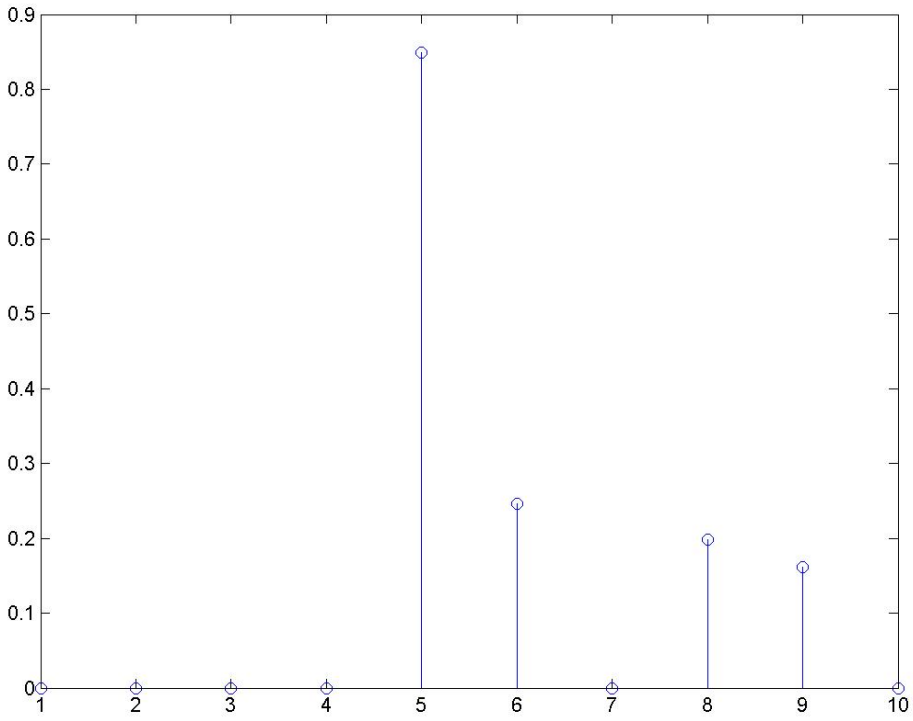


Figure3: Channel Response of Typical Suburban using f = 10Mhz

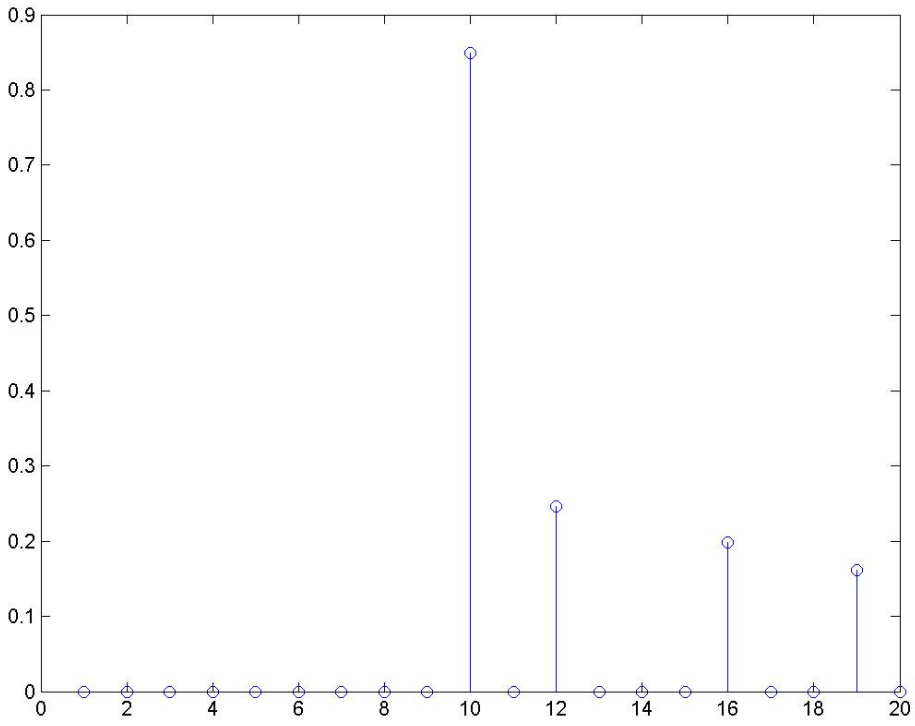


Figure4: Channel Response of Typical Suburban using f = 20Mhz

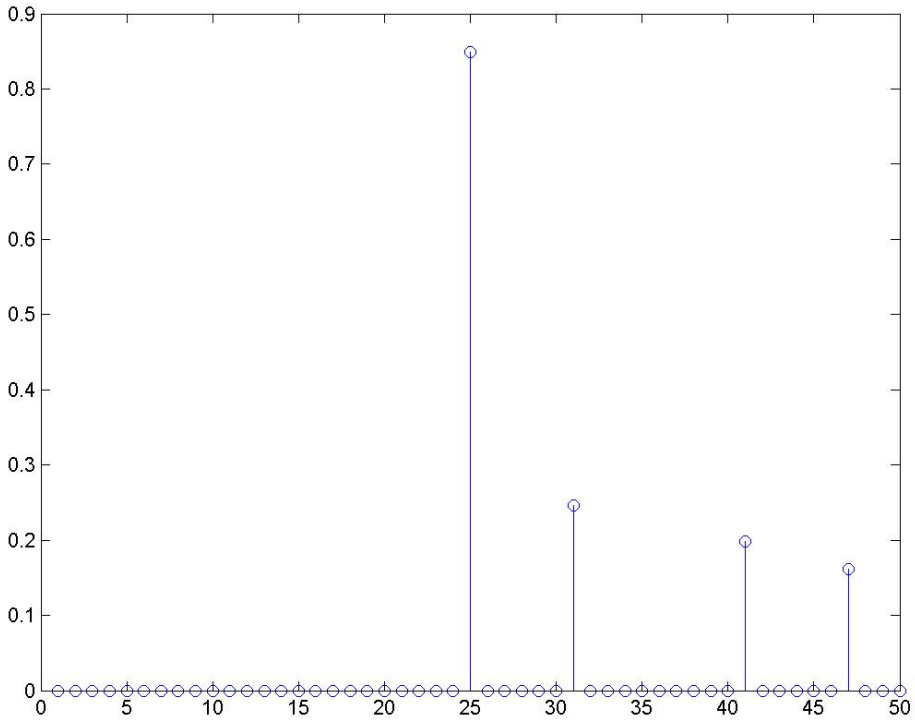


Figure5: Channel Response of Typical Suburban using f = 50Mhz

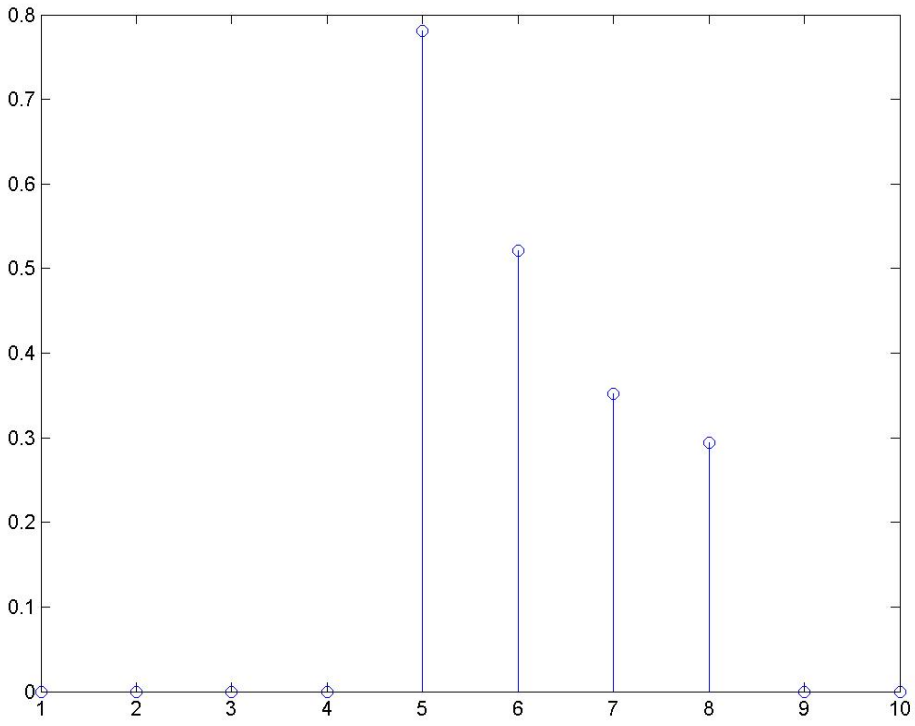


Figure6: Channel Response of Typical Urban using f = 10Mhz

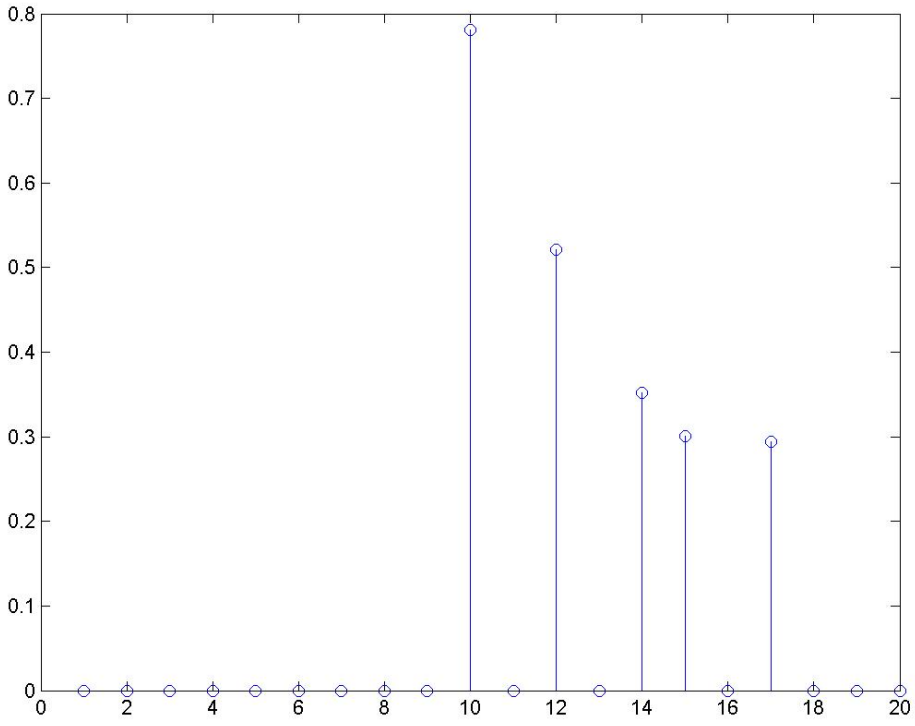


Figure7: Channel Response of Typical Urban using f = 20Mhz

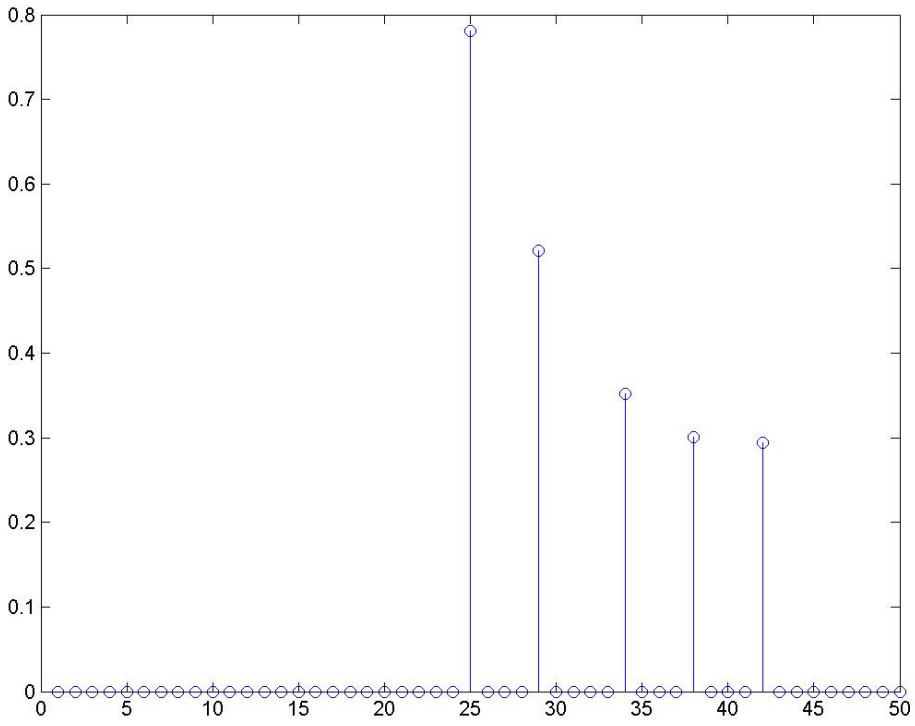


Figure8: Channel Response of Typical Urban using f = 50Mhz

Equalization:

Due to the channel characteristics such as ISI, thermal noise and channel response of the system, transmitted signal faces distortion. In order to overcome this distortion and increase the quality of the received signal, we need to have an equalizer that we will be passing our received signal through so that the signal can be recovered.

In order to maintain the total response of the system close to the impulse function, the response of the equalizer should be approximately equal to the inverse of the channel response.

In this project, we use two types of equalization techniques, which are Zero Forcing Equalizer and Minimum Mean Square Equalizer.

Zero Forcing Equalizer:

This type of equalizer aims that the total time domain impulse response of the channel and equalizer to be the impulse at time zero and zero otherwise. Therefore the frequency response of the equalizer is very approximate to the inverse of the channel response up to the equalizer length. However this type of equalizer doesn't consider the noise power and is therefore not immune to high noise power. It can boost the noise power at the frequencies where the channel response is small, therefore at high noise level and in a highly disturbed channel response the zero forcing equalizer doesn't show a high performance.

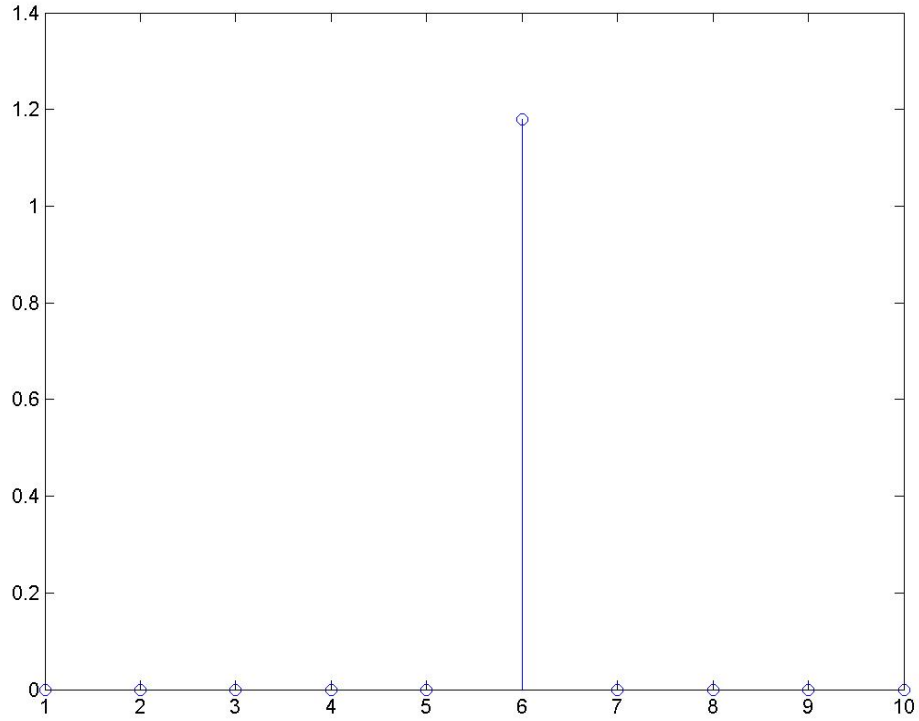


Figure9: Zero-forcing equalizer at f = 10Mhz

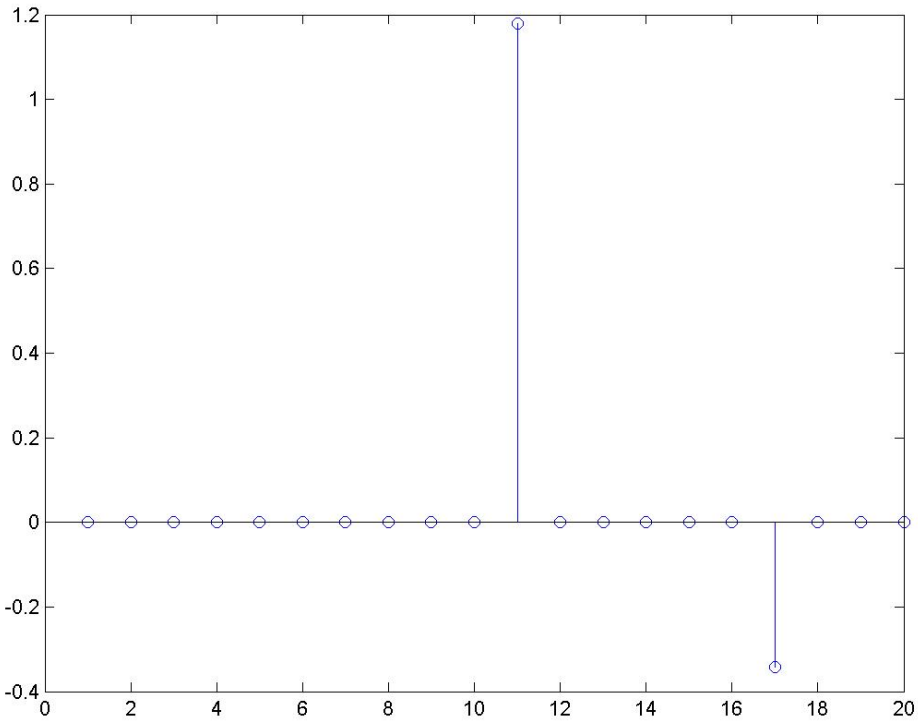


Figure10: Zero-forcing equalizer at f = 20Mhz

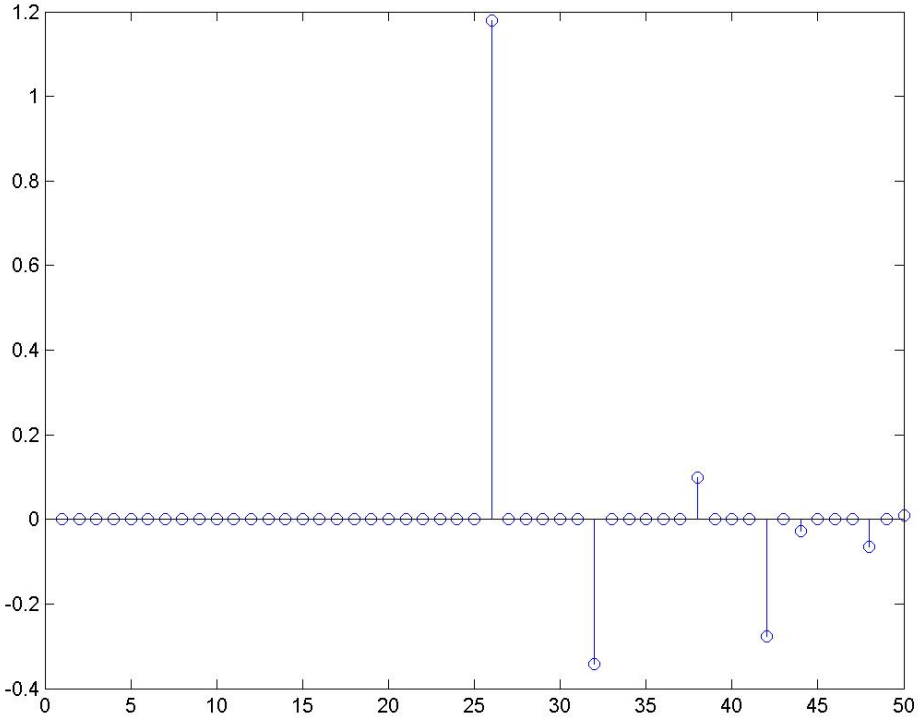


Figure11: Zero-forcing equalizer at f = 50Mhz

Minimum Mean Square Equalizer:

This type of equalizer optimizes the weight of the equalizer taps by considering the channel distortion and the noise power both. Therefore the total response of the channel and the equalizer is not equal to the impulse function in the presence of noise but the noise is not boosted in some frequencies. Therefore the equalizer is optimized due to the noise power and therefore it should have a better performance than the previously shown equalizer in the presence of noise.

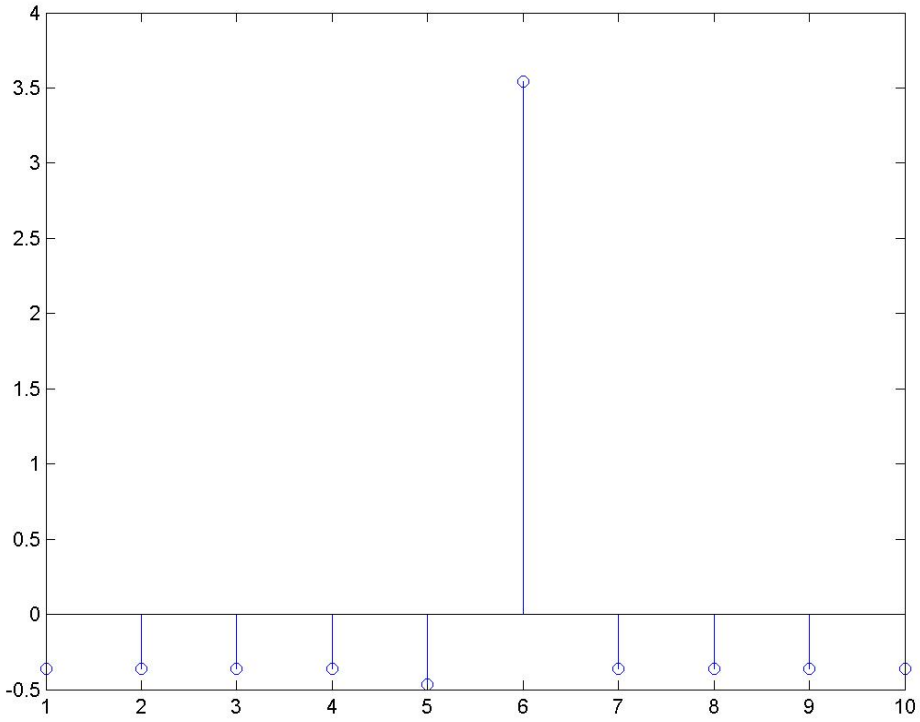


Figure12: MMSE for Suburban at f = 10Mhz

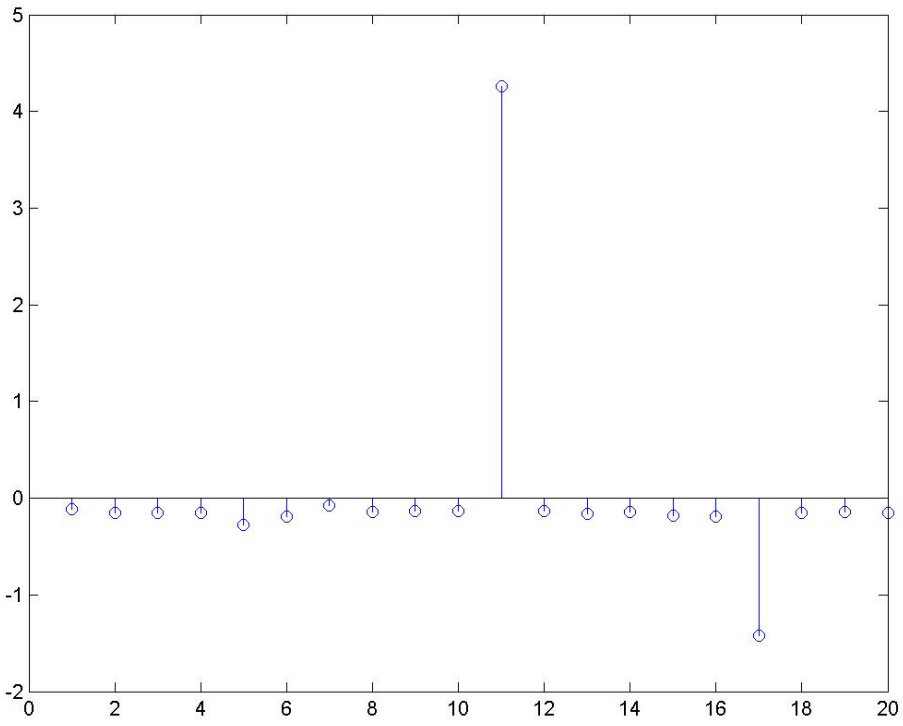


Figure13: MMSE Suburban at f = 20Mhz

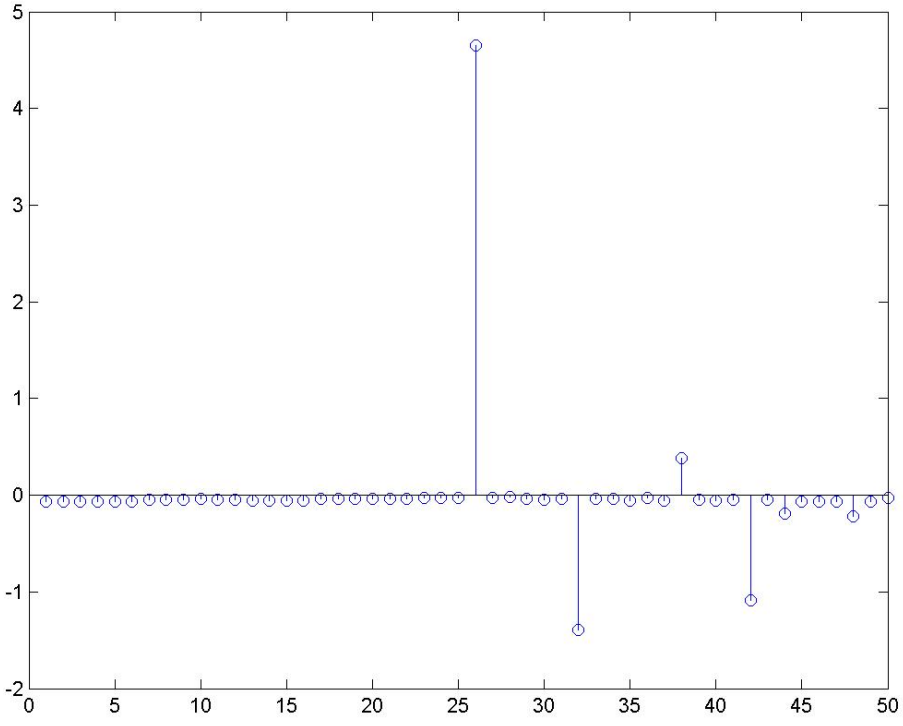


Figure14: MMSE Suburban at f = 50Mhz

Graphical User Interface (GUI):

As can be seen from the example figures, the graphical user interface has three main parts: inputs, channel response plot and BER vs SNR plot. The inputs are simply the type of the channel, the response length, the type of equalization used and the equalizer length. Then the "PLOT" button is used in order to plot the channel impulse response, the equalizer impulse response and the SNR vs BER graph. Below you can see two simulations that describe the function of the GUI.

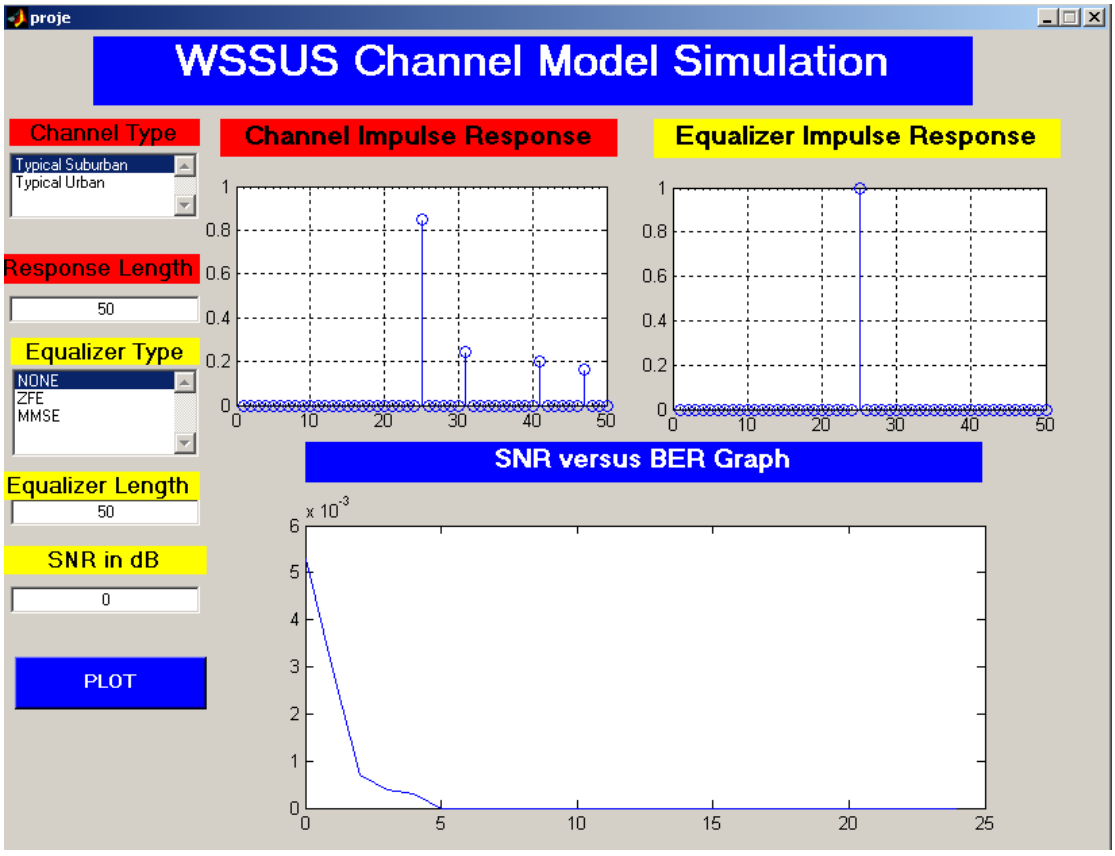
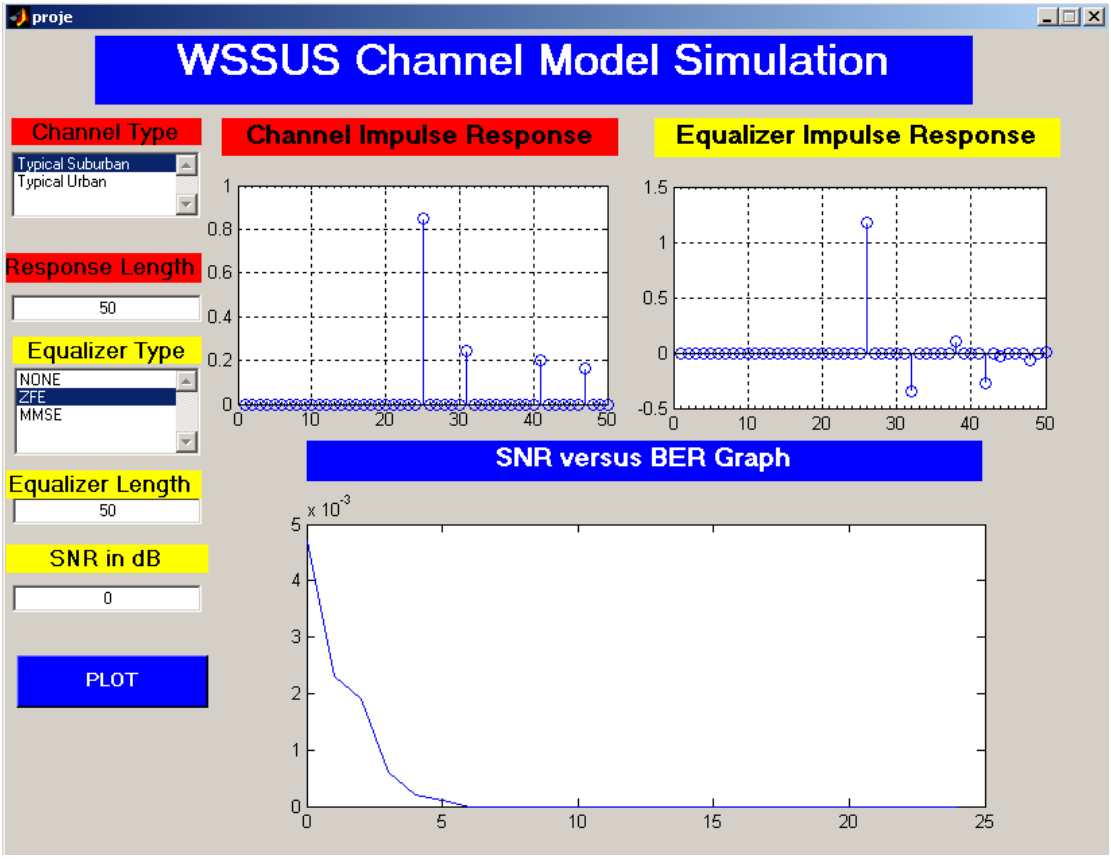


Figure15: WSSUS Channel Model Simulation where the channel type is Typical Suburban, the response length is 50, and no equalizer is used.



s

Figure16: WSSUS Channel Model Simulation where the channel type is Typical Suburban, the response length is 50, Zero Forcing equalization is used with an equalizer length of 50.

Conclusion:

According to our result, we can see that equalization improve our snr versus ber ratio. Especially, minimum mean square error equalization is better than zero forcing equalization.

References:

[1] St. Bug, Ch. Wengerter, I. Gaspard, R. Jakoby "WSSUS-Channel Models for Broadband Communication Systems," *IEEE Vehicular Technology Conference*, Birmingham, Alabama, 2002.

[2] F. Muratore, "UMTS: Mobile Communications for the Future" Wiley, USA, 2001.

[2] P.A. Bello "Characterization of randomly time-variant linear channels," *IEEE Trans. Comm. Syst.* , Bd. CS-11, no. 4, 1963, pp. 360-393.

Appendix:

Channel_response.m

```
%TE 304 Digital Communication

%Term Project :

%By Onur Sarkan & Çigdem Altay


%This function produce impulse response of channel.

%INPUTS:

%type: type of channel(like 1,2,3..)

%ntaps:Length of channel impulse response


%OUTPUT:

%h_n: Channel impulse response vector.


function h_n = new_channel_response(type,ntaps);


%Channel parameters are initialized according to type of channel.

if type=='1'

   %typical suburban ar-coefficients

   a = [1 0.1466-0.0785i 0.2012-0.0619i 0.1977-0.0145i 0.1907-0.0187i 0.1479-
0.0295i 0.1407-0.0032i 0.1075+0.0093i -0.0043+0.0108i; 1 -0.1182-0.0043i 0.0130-
0.0345i 0.1970+0.0120i 0.1956+0.0321i 0.1075+0.0303i 0.1117+0.0426i 0.1258-
0.0099i 0.0730-0.0194i; 1 -0.1182-0.0043i 0.0130-0.0345i 0.1970+0.0120i
0.1956+0.0321i 0.1075+0.0303i 0.1117+0.0426i 0.1258-0.0099i 0.0730-0.0194i; 1 -
```

```
0.1095+0.0193i   0.0541+0.0267i   0.1813-0.0044i   0.1549+0.0028i   0.1407+0.0282i
0.1293+0.0234i 0.1127-0.0122i 0.0753+0.0198i];
   k=[-7.2 -4.4 -3 -1.2];
   f=[0.18 -0.11 -0.11 -0.27];
   tao=[0 0.6 1.6 2.2];
   power=[0 -8.6 -8.8 -6.6];
   path_num=4;
elseif type=='2'
   %typical urban3 ar-coefficients
   a=[1      -0.0394-0.0131i      0.1924+0.0054i      0.2030+0.0188i      0.1820+0.0168i
0.1705+0.0048i  0.1164-0.0022i  0.1122-0.0118i  0.0658+0.0166i;  1  -0.0394-0.0131i
0.1924+0.0054i   0.2030+0.0188i   0.1820+0.0168i   0.1705+0.0048i   0.1164-0.0022i
0.1122-0.0118i  0.0658+0.0166i;  1  -0.1461-0.0046i  0.2111+0.0030i  0.1715+0.0296i
0.1524+0.0142i 0.1546+0.0155i 0.1079+0.0118i 0.0859-0.0156i 0.0657-0.0286i; 1 -
0.1992-0.0461i   0.1982-0.0440i   0.1887+0.0013i   0.2075+0.0189i   0.1094+0.0405i
0.1560+0.0077i  0.1180+0.0369i  0.0725-0.0159i;  1  -0.1992-0.0461i  0.1982-0.0440i
0.1887+0.0013i   0.2075+0.0189i   0.1094+0.0405i   0.1560+0.0077i   0.1180+0.0369i
0.0725-0.0159i];
   k=[-6.1 -7.3 -4 -3 -3];
   f=[-0.03 -0.03 -0.35 -0.03 -0.03];
   tao=[0 0.4 0.9 1.3 1.7];
   power=[0 -4.3 -5.1 -5.2 -5.4];
   path_num=5;
end
```

```
%Random noise is generated for each delay path
noise=zeros(1,path_num);
for x=1:1:path_num
   w_n=0;
   for y=1:1:100
      w_n=w_n+wgn(1,1,power(x));
   end
   w_n=w_n/100;
   noise(x)=w_n;
end
k=k/20;

%Output of H(z) is calculated.
y_noise=zeros(1,path_num);
for x=1:1:path_num
   y_noise(x)=filter(1,a(x,:),noise(x));
end

%C_m are generated.
C_m=zeros(1,path_num);
for x=1:1:path_num
   sigma_m=sqrt(2*(10^(power(x)/10)));
   C_m(x)=sigma_m*sqrt(k(x))*cos(2*pi*f(x)*tao(x)*1e-
6)+i*sigma_m*sqrt(k(x))*sin(2*pi*f(x)*tao(x)*1e-6);
end
```

```
pre_h_n=zeros(1,ntaps);
for x=1:1:path_num
   pre_h_n(round(ntaps/2+(ntaps/5)*tao(x)))=C_m(x);
end
h_n=abs(pre_h_n);
```

Zfe.m:

```
%TE 304 Digital Communication
%Term Project : Twisted pair cable transmission simulation
%By Onur Sarkan & Çigdem Altay
%Zero Forcing Equalizer:
%This function produce Zero forcing equalizer coefficients.

%Input Parameters:
%h:channel's impulse response
%ntaps:number of taps in the equalizer

%Outputs:
%weq:equalizer's tap coefficients

function weq=zfe(h,ntaps)
if (length(h)+1)/2 < ntaps
   %Zero padding from both sides.
   h=[zeros(1,ceil(ntaps-(length(h)+1)/2)) h zeros(1,floor(ntaps-(length(h)+1)/2))];
end
C=h(ceil(length(h)/2):ceil(length(h)/2+ntaps-1));
```

```
R=fliplr(h(ceil(length(h)/2-ntaps+1):ceil(length(h)/2)));
x=toeplitz(C,R); %z=xc
z=[zeros(ceil((ntaps-1)/2),1); 1; zeros(floor((ntaps-1)/2),1)];   %z
weq=(inv(x)*z).';     %weq=c=(1/x)*z
```

Mmse.m:

```
%TE 304 Digital Communication
%Term Project :
%By Onur Sarkan & Çigdem Altay
%Minimum mean square equalizer:
%This function produce minimum mean square equalizer coefficients according
%to variance of noise, bit representation, channel impulse response, and
%number of taps in the equalizer.

%Input Parameters:
%h:channel's impulse response
%ntaps:number of taps in the equalizer
%nvar:noise variance
%alpha_1:input bit 1
%alpha_0:input bit 0

%Output:
%weq:equalizer's tap coefficients

function weq = mmse(h,ntaps,nvar,alpha_1,alpha_0);
```

```
C=[h(length(h)) zeros(1,ntaps-1)];
R=[fliplr(h) zeros(1,ntaps-1)];
x=toeplitz(C,R);
response=[zeros(floor((length(h)+ntaps-1)/2),1);  1;  zeros(floor((length(h)+ntaps)/2-
1),1)];
d=((alpha_1-alpha_0)/2)^2*eye(length(h)+ntaps-1)+((alpha_1+alpha_0)/2)^2;
weq = flipud(inv((x*d*x')+(nvar*eye(ntaps)))*x*response);
```

Atd.m:

```
%Onur Sarkan 5241
%te 304 Digital Communication
%Homework 4
%Adaptive Threshold Detection

function opt_threshold = atd(type_of_channel,type_of_equalizer,snr);


h_n=channel_response(type_of_channel,50);
if type_of_equalizer=='2'
   weq=zfe(h_n,50);
elseif type_of_equalizer=='3'
   weq=mmse(h_n,50,snr,1,0);
else
   weq=zeros(1,50);
   weq(25)=1;
```

```
end
thresh=[-0.5:0.1:1];
h_total=conv(h_n,weq);
x = randsrc(1,1000,[[0 1]; [0.5 0.5]]);
yeq=conv(h_total,x);
yeq=yeq(49:1049);
thresh=thresh+0.5;
yeq=yeq(floor((length(yeq)-length(x))/2+1):floor((length(yeq)+length(x))/2));
temp=repmat(thresh',1,length(yeq));
y=repmat(yeq,length(thresh),1);
z=logical(y>temp);
z=z-repmat(x,length(thresh),1);
[t,index]=min(abs(sum(z,2)));
opt_threshold=thresh(index);
```

Snr_ber.m:

```
%TE 304 Digital Communication
%Term Project :
%By Onur Sarkan & Çigdem Altay

function [snr, ber] = snr_ber(type_of_channel,type_of_equalizer);
   h_n=channel_response(type_of_channel,50);
if type_of_equalizer=='2'
   h_n=channel_response('2',50);
   weq=zfe(h_n,50);
```

```
    h_total=conv(h_n,weq);
    message = randsrc(1,10000,[[0 1]; [0.5 0.5]]);
    signal=zeros(1,500000);
    for i=1:1:10000
       for j=1:1:50
          signal((i-1)*50+j)=message(i);
       end
    end
    out_signal=conv(h_total,signal);
    deneme=out_signal(50:500000);
    snr=zeros(1,25);
    ber=zeros(1,25);
    thresh=atd(type_of_channel,type_of_equalizer,0);
    for i=0:1:24
       noisy=awgn(out_signal,i);
       demodulated_signal=demodulate_signal(noisy,thresh);
       ber(i+1)=sum(abs((message-demodulated_signal)/10000));
       snr(i+1)=i;
    end
elseif type_of_equalizer=='3'
    h_n=channel_response(type_of_channel,50);
    message = randsrc(1,10000,[[0 1]; [0.5 0.5]]);
    signal=zeros(1,500000);
    for i=1:1:10000
       for j=1:1:50
```

```
            signal((i-1)*50+j)=message(i);
        end
    end
    snr=zeros(1,25);
    ber=zeros(1,25);
    for i=0:1:24
        thresh=atd(type_of_channel,type_of_equalizer,0);
        weq=mmse(h_n,50,0,1,0);
        h_total=conv(h_n,weq);
        out_signal=conv(h_total,signal);
        deneme=out_signal(50:500000);
        noisy=awgn(out_signal,5*i);
        demodulated_signal=demodulate_signal(noisy,thresh);
        ber(i+1)=sum(abs((message-demodulated_signal)/1000000));
        snr(i+1)=i;
    end
else
    weq=zeros(1,50);
    weq(25)=1;
    h_n=channel_response(type_of_channel,50);
    h_total=conv(h_n,weq);
    message = randsrc(1,10000,[[0 1]; [0.5 0.5]]);
    signal=zeros(1,500000);
    for i=1:1:10000
        for j=1:1:50
```

```
            signal((i-1)*50+j)=message(i);
        end
    end
    out_signal=conv(h_total,signal);
    deneme=out_signal(50:500000);
    snr=zeros(1,25);
    ber=zeros(1,25);
    thresh=atd(type_of_channel,type_of_equalizer,0);
    for i=0:1:24
        noisy=awgn(out_signal,i);
        demodulated_signal=demodulate_signal(noisy,thresh);
        ber(i+1)=sum(abs((message-demodulated_signal)/10000));
        snr(i+1)=i;
    end
end
%figure;plot(snr,ber);
```

Demodulate_signal.m:

```
%TE 304 Digital Communication
%Term Project :
%By Onur Sarkan & Çigdem Altay

function output = demodulate_signal(input,thresh);

sizes=size(input);
```

```
bit_num=floor((sizes(2)/50)-1);
pre_output=zeros(1,bit_num);
for i=1:1:bit_num
   sig=0;
   for j=1:1:50
      sig=sig+input((i-1)*50+j+50);
   end
   if floor(sig/50)>thresh
      pre_output(i)=1;
   else
      pre_output(i)=0;
   end
end
output=pre_output;
```

Proje.m:

```
function varargout = proje(varargin)
% PROJE M-file for proje.fig
%      PROJE, by itself, creates a new PROJE or raises the existing
%      singleton*.
%
%      H = PROJE returns the handle to a new PROJE or the handle to
%      the existing singleton*.
%
%      PROJE('CALLBACK',hObject,eventData,handles,...) calls the local
```

```
%      function named CALLBACK in PROJE.M with the given input arguments.
%
%      PROJE('Property','Value',...) creates a new PROJE or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before proje_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to proje_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES


% Edit the above text to modify the response to help proje


% Last Modified by GUIDE v2.5 18-Jun-2004 14:32:06


% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @proje_OpeningFcn, ...
                   'gui_OutputFcn',  @proje_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
```

```
if nargin & isstr(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before proje is made visible.
function proje_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to proje (see VARARGIN)

% Choose default command line output for proje
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
```

```
% UIWAIT makes proje wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = proje_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
```

```
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Executes on selection change in listbox1.
function listbox1_Callback(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns listbox1 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from listbox1


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
```

```
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function listbox2_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to listbox2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



% --- Executes on selection change in listbox2.
function listbox2_Callback(hObject, eventdata, handles)
% hObject    handle to listbox2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns listbox2 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from listbox2



% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function listbox3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Executes on selection change in listbox3.
function listbox3_Callback(hObject, eventdata, handles)
% hObject    handle to listbox3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: contents = get(hObject,'String') returns listbox3 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from listbox3


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
response_length = str2num(get(handles.edit1, 'String'));
channel_type = num2str(get(handles.listbox1, 'Value'));
h_n=channel_response(channel_type,response_length);
axes(handles.axes3);
stem(h_n);
set(handles.axes3,'XMinorTick','on');
grid on;

eq_type = num2str(get(handles.listbox2, 'Value'));
equalizer_length = str2num(get(handles.edit3, 'String'));
if eq_type=='2'
   weq=zfe(h_n,equalizer_length);
elseif eq_type=='3'
   SNR = str2num(get(handles.edit5, 'String'));
   weq=mmse(h_n,equalizer_length,SNR,1,0);
```

```
elseif eq_type=='1'
    weq=zeros(1,50);
    weq(25)=1;
end

axes(handles.axes4);
stem(weq);
set(handles.axes4,'XMinorTick','on');
grid on;

[a,b]=snr_ber(channel_type,eq_type);
plot(a,b,'Parent',handles.axes5);




% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
```

```
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end




function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a double
```