EE 6383

Advanced Topic In Communications

Final Exam Assignment

Software Radio Simulation in Matlab

Submitted by

S M Azharul Karim

UTSA ID: dhf227

Banner ID: @01519097

Note: All matlab codes are submitted online. It is located in the matlab codes folder. The file named- "final\_dhf227.m" is the main code. You can get all the outputs by running it in matlab.

Matlab Code:

% projcmtd.m

clc

clear all

close all

% This parameter is used

% to simulate the frequency shifts

% e.g., due to a Doppler, i.e., relative movement

% it is quite close to 1 in this project.

freq\_mult = 1/.994;

% This is the sampling frequency after downsampling

% by 6 block just before the carrier recovery

% The block diagram is simulating the analog part

% which is then sampled. This corresponds to the output

% of the downsample by 6 block. From that point

% the processing is digital and its sampling frequency

% is defined below. This corresponds to 1/(T\_symbol/4).

fs = 2000\*freq\_mult;

% This is the carrier frequency which should be

% compensated by the carrier recovery algorithm

% the shifted frequency, e.g., due to a Doppler

fc = 500\*freq\_mult;

% This the carrier frequency which should be

% compensated by the carrier recovery algorithm

% the non-shifted version

fc\_prime = 500;

% This the ansolute bandwidth of the

% raised cosine filter

B = 400;

% This the value of the frequency error

% before the carrier recovery

freq\_off = fc - fc\_prime;

% This is the sampling frequency T\_symbol/24

% which is used to simulate the analog signal

fsH = 6\*fs;

% This is the carrier frequency at which

% the signal is modulated and sent through

% the channel

fcH = 5/12\*fsH;

% This is the intermediate frequency

% to which the initially modulated

% signal should be downconverted after

% the first bandpass filter BPF1

fiH = 2.5/12\*fsH;

% A scaling factor is used in filter specifications

% to define frequency edges

fscal = .9;

% Half the sampling frequency of the signal

% transmitted through the channel

fNH = fsH/2;

% A phase offset which is used to define

% the possible shift in phase due to arrival

% delays

phase\_offsetH = 1.2;

%

fi\_prime = 500;

% Sampling interval before time recovery

% after downsampling by 6 block

Ts = 1/fs;

% Sampling interval of the transmitted signal

TsH = 1/fsH;

% half of the number of symbols spanned by the

% transmit and receive filters

L = 4;

% oversampling factor in receiver algorithms

%

M = 4;

% oversampling factor data-to-transmit data

MH = 24;

% parameter of the raised cosine

alpha = .1;

%

t\_offset = .3;

t\_offsetH = 3\*pi/2;

% one way of composing a message

msg = letters\_to\_pam4('asdlf asd;flk jasdkdfl;kdfl;kasdf;kasdfk;asdfka sbd;fkj d;kfj asdkfj a;dkfj a;dkj a;sldkj asl;dfk a;dkfj a;dkfj ;akldj a;sdkfj asl;dkja d;fkj s;fkFour score and seven years ago could not hit the broad side of a barn ECE hell bent for leather. yeah, down with Microsoft© ECE4953 the project is long! stuff... what you might not like even more is that homework 4 will be worse...');

% another way of composing a message

% by defining components

% and combining them. Note that training is used

% for the equalization and header is used to find the

% message edges

header = letters\_to\_pam4('ECE4953 header sequence');

training = letters\_to\_pam4('ECE4953 training signal ECE4953 training signal ECE4953 training signal sdlkfjasdfkljas;fkljaslalk m,,mn,cvior,mnilg,knfviln,dkfvkl,ndkfglkdfgdfg,mnfdfg,khdgd;kjsd;kalsd;as;ld The quick brown fox jumped over the lazy dog.');

% english\_text = ' down with Microsoft© ECE4953 header signal the project is hard! stuff...';

english\_text = 'S M Azharul';

projmsg = letters\_to\_pam4(english\_text);

randmsg1 = pam4(12000);

randmsg2 = pam4(1000);

randmsg3 = pam4(1000);

msg = [randmsg1 training randmsg2 header projmsg randmsg3];

orig = msg;

data\_symbols = length(projmsg);

% This filter is used to simulate distortions in a channel

% normally it should be used after the upconversion

% but here we may skip many zeros and still get a simulation

% of an intersymbol interference ISI

b = [.5 1 -.6];

%b = [.2 1 -.3];

% Apply the channel ISI distortions directly to our message

msg\_prime= filter(b,1,msg);

% This pulse is generated to be used as a received filter

pulse = SRRC(L,alpha,M,t\_offset);

% This pulse is generated to be used as a transmit filter

% its the same as "pulse" only sampling is different as they operate

% at different sampling frequencies. The important thing

% is that they span 8 symbol periods

pulseH = SRRC(L,alpha,MH,t\_offsetH);

matched = pulse;

matchedH = pulseH;

% Pulse shaping for transmitted signal. Its implemented

% at highest sampling rate with period T\_symbol/24

upsampled\_msgH = zeros(1,MH\*length(msg\_prime));

upsampled\_msgH(1:MH:end) = msg\_prime;

pulse\_shapedH = filter(pulseH,1,upsampled\_msgH) \* max(pulse)/max(pulseH);

figure;

subplot(2,1,1),nfspec(pulse\_shapedH,fsH);

title('pulse-shaped signal (T/24)');

% The modulation of the pulse shaped signal

% simply multiply to a sinusoid at a carrier frequency

% Add phase offset distortion by the channel

t = TsH\*[1:length(pulse\_shapedH)];

upconverterH = cos(2\*pi\*fcH\*t + phase\_offsetH);

modulatedH = 2\*pulse\_shapedH.\*upconverterH;

subplot(2,1,2),nfspec(modulatedH,fsH);

title('modulated signal (T/24)');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%% From this point we work with the receiver

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Bandpass filter 1 is used to filter out undesired signals at frequencies

% other than expected. BPF1 only retains the band of the expected signal

% One can see that the bandwidth of size "fcH-B fcH+B" is retained

% it is slightly expanded by the factor fscal to accomodate

% possible imperfections of the designed filters

bpf1 = remez(100,[0 fcH-2\*B fcH-B\*fscal fcH+B\*fscal fcH+2\*B fNH]/fNH,[0 0 1 1 0 0]);

figure;

nfspec(bpf1,fsH);

title('BPF1'); xlabel('frequency');

% This line applies the BPF1 to the signal

bandpassedH = filter(bpf1,1,modulatedH);

% Here we downconvert the signal to intermediate

% frequency fiH

% For this purpose we first multiply to a

% proper sinusoid and then filter by a bandpass

% filter BPF2

IFconverterH = cos(2\*pi\*(fcH-fiH)\*t);

IFconvertedH = 2\*bandpassedH.\*IFconverterH;

% This filter is filtering out images other than those

% located around required intermediate frequency 2500Hz

bpf2 = remez(100,[0 fiH-2\*B fiH-B\*fscal fiH+B\*fscal fiH+2\*B fNH]/fNH,[0 0 1 1 0 0]);

figure;

nfspec(bpf2,fsH);

title('BPF2');

% Applying designed BPF2

IFbandpassedH = filter(bpf2,1,IFconvertedH);

figure;

subplot(3,1,1),nfspec(IFbandpassedH,fsH);

title('IF bandpassed signal (T/24)');

% Downsample the signal to process

% by carrier recovery algorithm

% Here the signal spaced around (+/-) 2500Hz

% IF frequency will be replicated and

% spaced at (+/-)500Hz

for k = 2:6

IFbandpassedH(k:MH/M:end) = 0;

end

subplot(3,1,2),nfspec(IFbandpassedH,fsH);

title('IF downsampled but undecimated signal (T/24)');

IFsampledH = IFbandpassedH(1:MH/M:end);

subplot(3,1,3),nfspec(IFsampledH,fs);

title('IF sampled signal (T/4)');

%%% The following lines before "save" are used

%%% to save a signal which can be used by digital

%%% processing part of the receiver, starting from the

%%% carrier recovery. Until this place

%%% we had a simulation of the analog signal (T\_symbol/24 spacing) which was

%%% then sampled at T\_symbol/4 spacing (using downsampling by 6 block)

modulated = IFsampledH;

rolloff\_factor = alpha;

bandwidth = B;

training\_signal = training;

%save test01.mat modulated fs fc\_prime bandwidth rolloff\_factor training\_signal header msg data\_symbols english\_text;

% This is an implementation of a carrier recovery algorithm

% it is the analog of the one used in your HW but using Costas

% loops instead of PLL

% The next three lines find the corrected carrier

% The correction is obtained in "theta"

t = Ts\*[1:length(modulated)];

%% My dual costas

% I have used the costas loop function twice. I've fed the output of one

% costas loop to another.

% first costas

theta=costas\_loop(modulated,fs,fc\_prime,B);

downconv1=cos(2\*pi\*fc\_prime\*t + theta);

%second costas

theta2=costas\_loop(downconv1,fs,fc\_prime,B);

figure;

plot(-theta2);title('theta'); ylabel('theta');

xlabel('iteration');

downconverter = cos(2\*pi\*fc\_prime\*t + theta2);

%%

% Using a carrier found at the carrier recovery stage

% demodulate the signal. First multiply to the found sinusoid

% then filter out images. Note here that the sampling rate is

% 2000Hz and 1 corresponds to 1000Hz.

demodulated = modulated.\*downconverter;

LPF = remez(50,[0 .25 .7 1],[1 1 0 0]);

reconstructed = filter(LPF,1,demodulated);

figure;

nfspec(reconstructed,fs);

title('reconstructed signal (T/4)');

% TODO:

% Apply time-recovery algorithm combined with the receive filter, i.e.

% Apply matched filter to signal "reconstructed" using "matched" array.

% Apply time recovery algorithm similar to the textbook version, e.g. it

% may look like this:

% match\_filtered = trecvry(previous\_step\_array,L,alpha,M);

% TODO: comment the following line which is a combined matched filter and

% time recovery which is not studied by us.

match\_filtered = opm(reconstructed,L,alpha,M);

% Apply equalizer to compensate the ISI distortion

% In LMS equalizer the training message location is found

% and the adaptation procedure is performed by comparing

% distorted training message with the known training

% message. The equalizer filter coefficients are found

% and the filter is applied to get equalized output

[equalized feq] = lmseq(match\_filtered,training);

%quantized\_msg = quantize4(match\_filtered);

quantized\_msg = quantize4(equalized);

figure;

plot(equalized,'.'); title('equalizer output level diagram');

% After finding the symbols the next task is to read

% the message by finding its beginning. This is achieved

% by correlating the signal with the known header

% Find the index which defines the start of the message

% header=[header zeros(1,length(quantized\_msg)-length(header))];

xc=conv(quantized\_msg,fliplr(header));

xc2=conv(msg,fliplr(header));

[dummy index] = max(abs(xc));

index

if xc(index) < 0

xc = -xc;

quantized\_msg = -quantized\_msg;

end

offset = length(quantized\_msg) - length(header);

offset

figure;

subplot(2,1,1),plot(1:length(xc(offset:end)),xc(offset:end)); ylabel('conv(received,header)');

subplot(2,1,2),plot(1:length(xc2(offset:end)),xc2(offset:end)); ylabel('conv(original,header)');

figure;

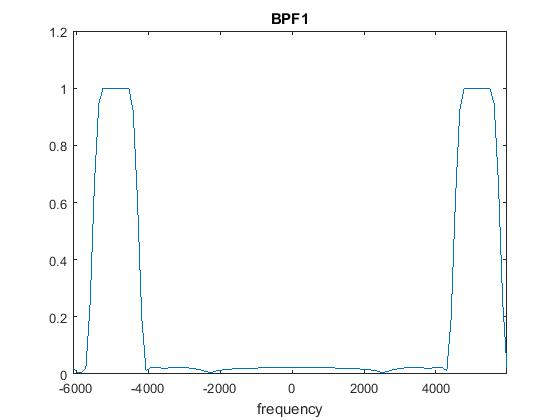
stem(conv(b,feq));

title('combined channel-equalizer response');

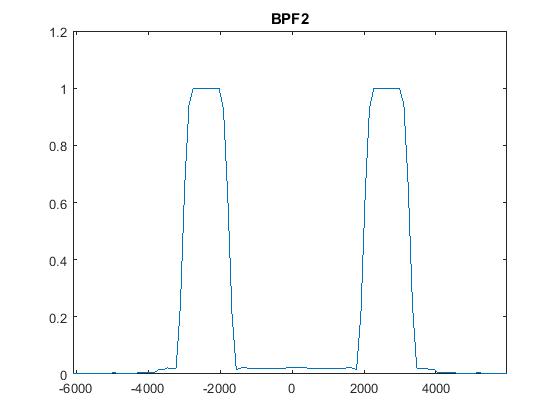
% Read the message from the identified "start of the message" index

decoded\_msg = pam4\_to\_letters(quantized\_msg(index+1:index+data\_symbols))

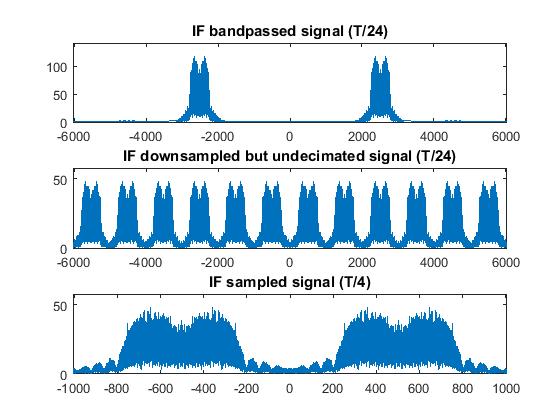
Output figures:



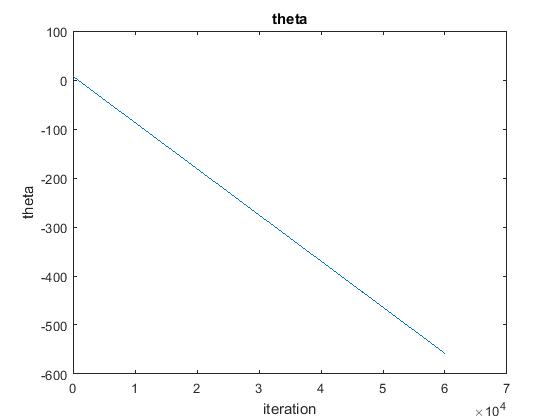
BPF 1 is the first bandpass filter used in the receiver.



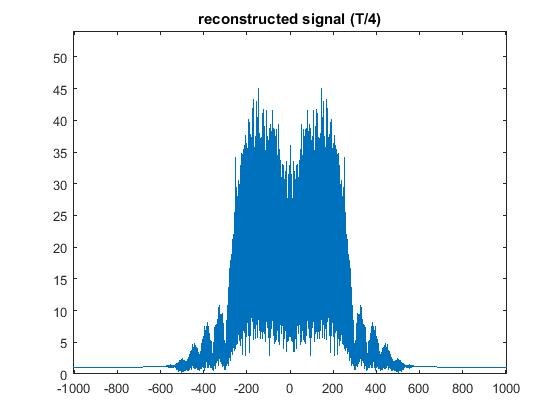
BPF 2 is the second band pass filter. This filter is filtering out images other than those located around required intermediate frequency



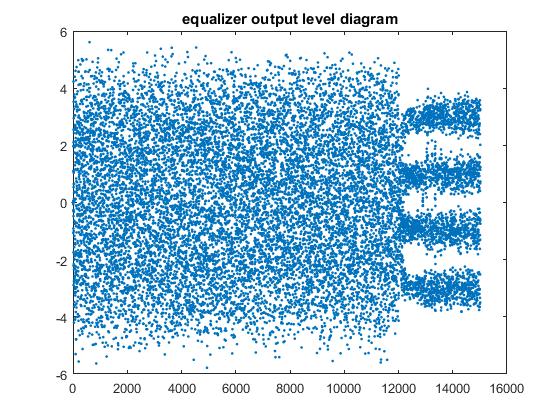
Down sampling the signal by a factor of 6



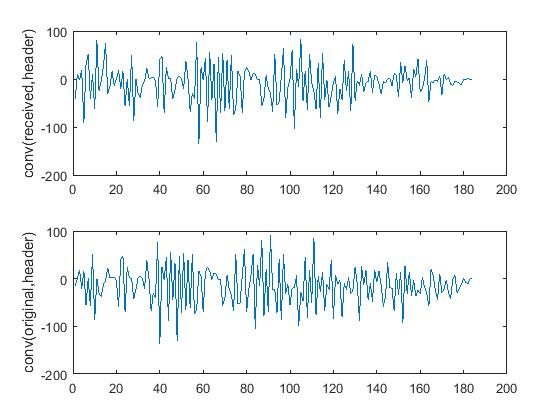
This the output of the dual costas loop that I designed. It is used for carrier recovery.



The recovered carrier is multiplied with the modulated signal and low pass filtered to obtain the demodulated signal



This is the level diagram of the equalizer output.



Convolution is done to find out the beginning of the message sequence

Receiver steps and their brief description:

1. Bandpass filter 1 is used to filter out undesired signals at frequencies other than expected.

2. Then we down convert the signal to intermediate frequency. For this purpose we first multiply to a proper sinusoid and then filter by another band pass filter BPF2. This filter is filtering out images other than those located around required intermediate frequency.

3. Then we have down sampled the signal by a factor of 6. In the transmitter, the signal was up sampled by a factor of 24. after this step, the signal is up sampled by a factor of 4.

4. After that we implement the carrier recovery part. Dual costas loop is used for carrier recovery. I have implemented the conventional costas loop twice to make a chained costas loop. I have fed output of the first costas loop to the second one. My diagram for the costas loop is given below-

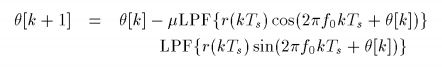
Costas Loop 2

Costas Loop 1

x[k] θ θ2 downconverter

= cos(2\*pi\*fc\*t+theta2)

The iteration formula for the costas loop is



5. The recovered carrier is then multiplied with the modulated signal and low pass filtered to obtain the demodulated signal.

6. For time recovery, a matched filter is designed. The function opm does the time recovery plus the matched filter generation part. In LMS equalizer the training message location is found using the matched filter and the adaptation procedure is performed by comparing distorted training message with the known training message. The equalizer filter coefficients are found and the filter is applied to get equalized output. The iteration formula for the LMS equalizer is given below:

lms eq.JPG

7. After finding the symbols the next task is to read the message by finding its beginning. This is achieved by correlating the signal with the known header. Convolution is used to find the index which defines the start of the message.

8. Finally, using the index, and the message signal length, the location of the message signal is found in the quantized message. It is then converted to letters to recover the original message signal.

Changes I have made in the Code:

1. I have included my first name in the transmitted message

english\_text = 'S M Azharul';

projmsg = letters\_to\_pam4(english\_text);

2. I have replaced the dual\_costas\_loop.m function provided by Dr. David Akopian With two chained conventional costas loop. My conventional costas loop function code is given below. I have changed the filter length in the conventional costas loop to 40. It is because as the length of the message signal is smaller than the previous one(the one that was initially given), if we use filter length=100, a length mismatch occurs. I have also changed the value of µ to 0.3.

%% This is a conentional costas loop

function f = costas\_loop(r\_if\_filt,fs,fi,B);

fN = fs/2;

td = (1:1:length(r\_if\_filt))/fs;

y\_c = r\_if\_filt.\*cos(2\*pi\*fi\*td);

y\_s = r\_if\_filt.\*sin(2\*pi\*fi\*td);

LPF = remez(40,[0 B 3\*B/2 fN]/fN,[1 1 0 0]);

x\_c = filter(LPF,1,y\_c);

x\_s = filter(LPF,1,y\_s);

theta=zeros(1,length(x\_c));

v1=zeros(1,length(x\_c));

v2=zeros(1,length(x\_s));

mphi = 0:pi/8:2\*pi;

guess = 2;

theta(1)=mphi(guess)+2\*pi;

mu1=0.3;

for k=1:length(x\_c)

v1(k)=x\_c(k)\*cos(theta(k))+x\_s(k)\*sin(theta(k));

v2(k)=x\_c(k)\*sin(theta(k))-x\_s(k)\*cos(theta(k));

theta(k+1)=theta(k)-mu1\*v1(k)\*v2(k);

end

f= -theta(2:end);

In the main program, I have called the costas\_loop function twice. I have fed output of the first one to the input of the second. the code segment is given below:

%% My dual costas

% I have used the costas loop function twice. I've fed the output of one

% costas loop to another.

% first costas

theta=costas\_loop(modulated,fs,fc\_prime,B);

downconv1=cos(2\*pi\*fc\_prime\*t + theta);

%second costas

theta2=costas\_loop(downconv1,fs,fc\_prime,B);

figure;

plot(-theta2);title('theta'); ylabel('theta');

xlabel('iteration');

downconverter = cos(2\*pi\*fc\_prime\*t + theta2);

3. I have replaced the built in "xcorr" function with the built in "conv" function. The below code segment shows how I have done that-

% this is how I have replaced xcorr by conv

xc=conv(quantized\_msg,fliplr(header));

xc2=conv(msg,fliplr(header));

We know that, cross correlation between two functions can be done using convolution, if one of the functions can be flipped (rotated 1800). Their mathematical expressions are given below-

x[n]∗h[n]=∑ h[k]x[n−k]

corr(x[n],h[n])=∑ h[k]x[n+k]

I have flipped one of the inputs and convoluted it with the other signal. That is the same as doing correlation. That is how I have implemented correlation using convolution.