`

復旦大學

Blue Tooth Project Report

Project Name: Blue Tooth Project

Team member: Ziyu Wang; ChuangYe Xu; JinKun Han

Contents

General Introduction:

What’s Bluetooth?

Definition: ***Bluetooth*** is a wireless technology standard for exchanging data over short distances (using short-wavelength UHF radio waves in the ISM band from 2.400 to 2.485 GHz from fixed and mobile devices, and building personal area networks (PANs). It was originally conceived as a wireless alternative to RS-232 data cables

Logo: The Bluetooth logo is a bind rune merging the Younger Futhark runes ᚼ (ᚼ, Hagall) and ᛒ (ᛒ, Bjarkan), Harald's initials

Implementation: Bluetooth operates at frequencies between 2402 and 2480 MHz, or 2400 and 2483.5 MHz including guard bands 2 MHz wide at the bottom end and 3.5 MHz wide at the top.

This is in the globally unlicensed (but not unregulated) industrial, scientific and medical (ISM) 2.4 GHz short-range radio frequency band. Bluetooth uses a radio technology called frequency-hopping spread spectrum. Bluetooth divides transmitted data into packets, and transmits each packet on one of 79 designated Bluetooth channels. Each channel has a bandwidth of 1 MHz.

It usually performs 1600 hops per second, with adaptive frequency-hopping (AFH) enabled. Bluetooth Low Energy uses 2 MHz spacing, which accommodates 40 channels

Communication and connection: A master BR/EDR Bluetooth device can communicate with a maximum of seven devices in a piconet.

The Bluetooth Core Specification provides for the connection of two or more piconets to form a scatternet, in which certain devices simultaneously play the master role in one piconet and the slave role in another

At any given time, data can be transferred between the master and one other device, the master chooses which slave device to address; typically, it switches rapidly from one device to another in a round-robin fashion, Being a master of seven slaves is possible; being a slave of more than one master is possible. The specification is vague as to required behavior in scatternets

What’s technically?

Group organization: It was established by Ericsson, IBM, Intel, Toshiba and Nokia, and later joined by many other companies.

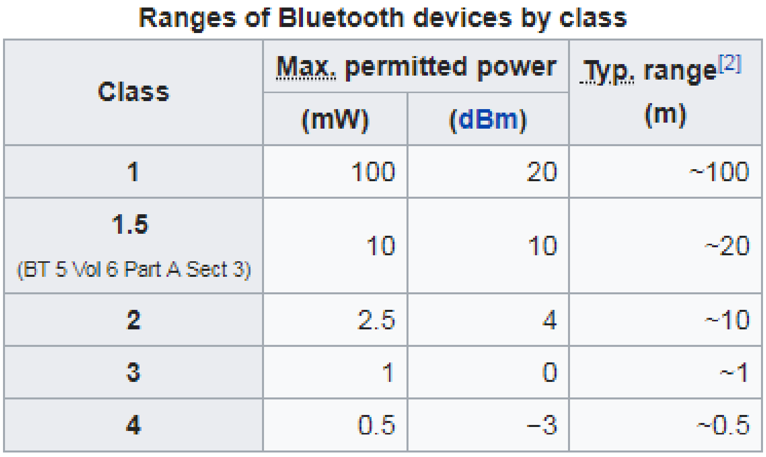
The specifications were formalized by the Bluetooth Special Interest Group (SIG) and formally announced on the 20 of May 1998.Today it has a membership of over 30,000 companies worldwide.

All versions of the Bluetooth standards support downward compatibility. That lets the latest standard cover all older versions.

The Bluetooth Core Specification Working Group (CSWG) produces mainly 4 kinds of specifications:

- The Bluetooth Core Specification, release cycle is typically a few years in between
- Core Specification Addendum (CSA), release cycle can be as tight as a few times per year
- Core Specification Supplements (CSS), can be released very quickly
- Errata (Available with a user account: Errata login)

Using Rage:



Ranges of Bluetooth devices by class

Class	Max. permitted power		Typ. range[2]
	(mW)	(dBm)	(m)
1	100	20	~100
1.5 (BT 5 Vol 6 Part A Sect 3)	10	10	~20
2	2.5	4	~10
3	1	0	~1
4	0.5	−3	~0.5

Version:

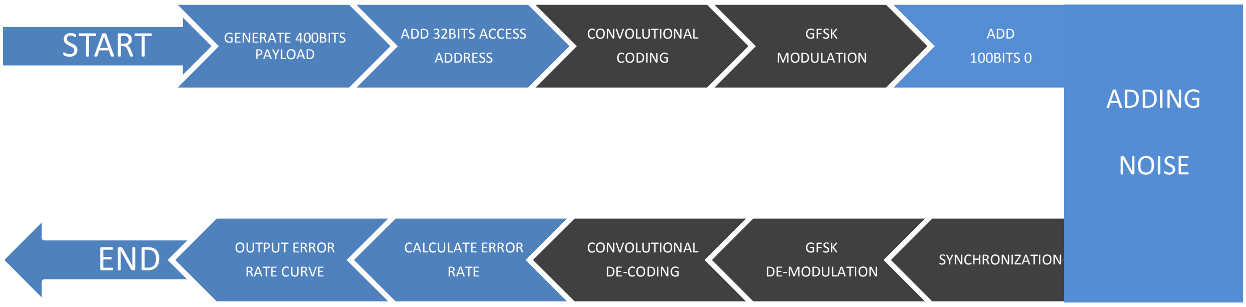
- Bluetooth 1.0 and 1.0B
- Bluetooth 1.1
- Bluetooth 1.2
- Bluetooth 2.0 + EDR
- Bluetooth 2.1 + EDR
- Bluetooth 3.0 + HS
- Bluetooth 4.0
- Bluetooth 4.1
- Bluetooth 4.2
- Bluetooth 5

Where used (From Wikipedia)?

- A typical Bluetooth mobile phone headset
- Wireless control and communication between a mobile phone and a handsfree headset.
- Wireless control of and communication between a mobile phone and a Bluetooth compatible car stereo system.
- Wireless control of and communication with iOS and Android device phones, tablets and portable wireless speakers.
- Wireless Bluetooth headset and Intercom. Idiomatically, a headset is sometimes called "a Bluetooth".
- Wireless streaming of audio to headphones with or without communication capabilities.
- Wireless streaming of data collected by Bluetooth-enabled fitness devices to phone or PC.
- Wireless networking between PCs in a confined space and where little bandwidth is required.
- Wireless communication with PC input and output devices, the most common being the mouse, keyboard and printer.
- Transfer of files, contact details, calendar appointments, and reminders between devices with OBEX.
- Replacement of previous wired RS-232 serial communications in test equipment, GPS receivers, medical equipment, bar code scanners, and traffic control devices.
- For controls where infrared was often used.
- For low bandwidth applications where higher USB bandwidth is not required and cable-free connection desired.
- Sending small advertisements from Bluetooth-enabled advertising hoardings to other, discoverable, Bluetooth devices.
- Wireless bridge between two Industrial Ethernet (e.g., PROFINET) networks.
- Seventh and eighth generation game consoles such as Nintendo's Wii, and Sony's PlayStation 3 use Bluetooth for their respective wireless controllers.
- Dial-up internet access on personal computers or PDAs using a data-capable mobile phone as a wireless modem.
- Short-range transmission of health sensor data from medical devices to mobile phone, set-top box or dedicated telehealth devices.
- Allowing a DECT phone to ring and answer calls on behalf of a nearby mobile phone.
- Real-time location systems (RTLS) are used to track and identify the location of objects in real time using "Nodes" or "tags" attached to, or embedded in, the objects tracked, and "Readers" that receive and process the wireless signals from these tags to determine their locations.
- Personal security application on mobile phones for prevention of theft or loss of items. The protected item has a Bluetooth marker (e.g., a tag) that is in constant communication with the phone. If the connection is broken (the marker is out of range of the phone) then an alarm is raised. This can also be used as a man overboard alarm. A product using this technology has been available since 2009.
- Calgary, Alberta, Canada's Roads Traffic division uses data collected from travelers' Bluetooth devices to predict travel times and road congestion for motorists.
- Wireless transmission of audio (a more reliable alternative to FM transmitters)
- Live video streaming to the visual cortical implant device by Nabeel Fattah in Newcastle university 2017.

System configuration:

Block diagram:



In our project we mainly focused on steps marked with grey in above block diagram:

That is in coding steps we have:

CONVOLUTIONAL CODING
GFSK MODULATION

In de-coding steps we have:

CONVOLUTIONAL DE-CODING
GFSK DE-MODULATION
SYNCHRONIZATION

We can mainly divide these steps to four main functions:

a. **Convolutional Coding-decoding related**
b. **GFSK Modulation and de-modulation related**
c. **Synch related**
d. **Pattern & mapping related**

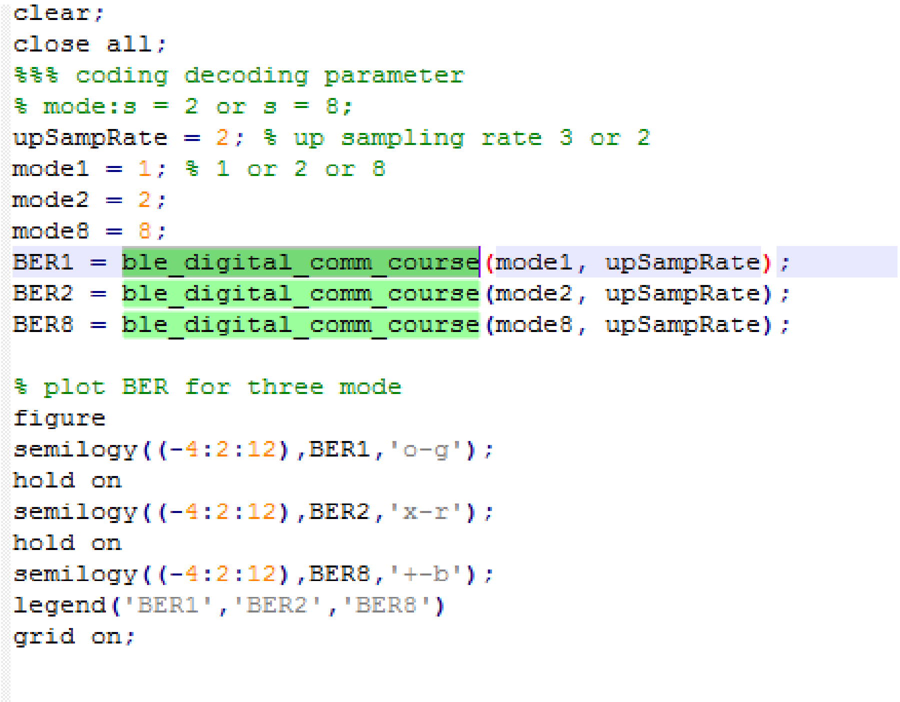
In order to do these steps, we create some functions, below section will talk details of these functions

Function explanation:

Main:

Main:

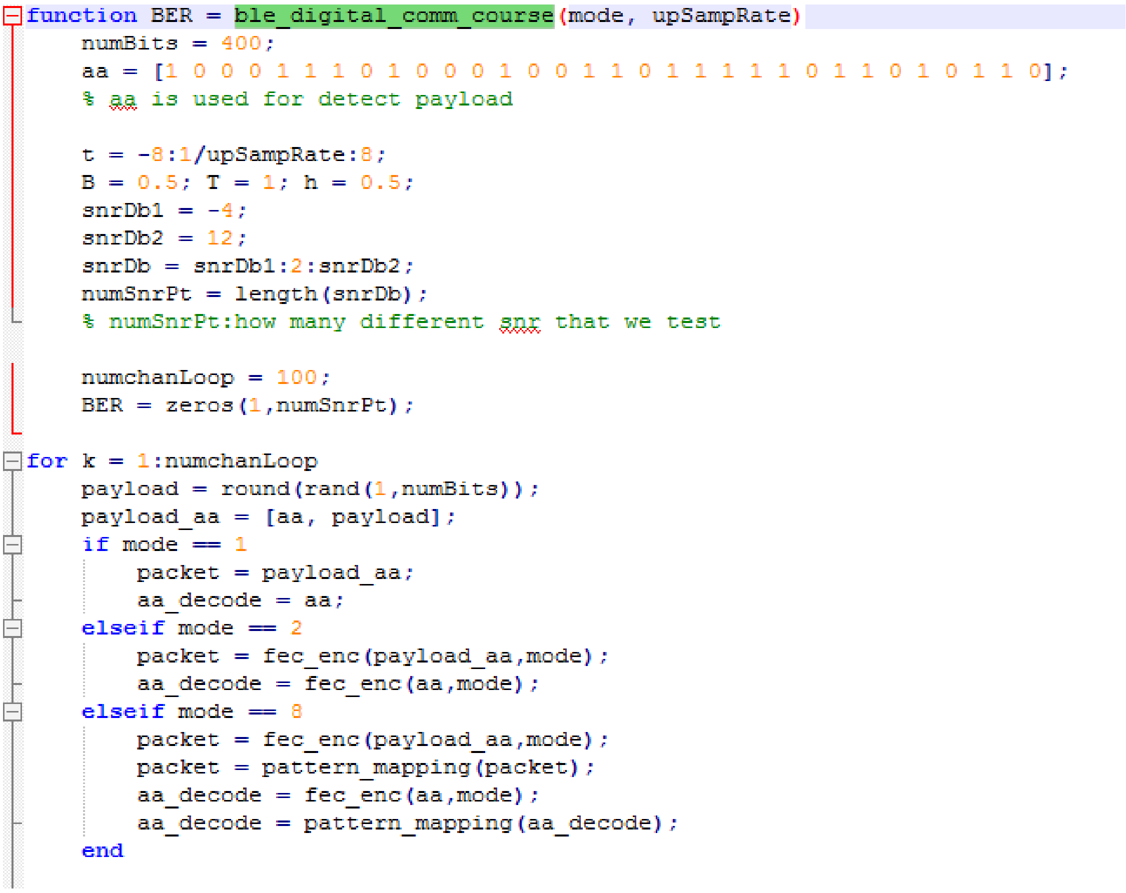
Main entrance of the function, we have three modes to test result, that is Mode=1/Mode=2/Mode=8 to select, then we will transfer Mode=1,Mode=2,Mode=8 parameters to “ble_digital_comm_course” function



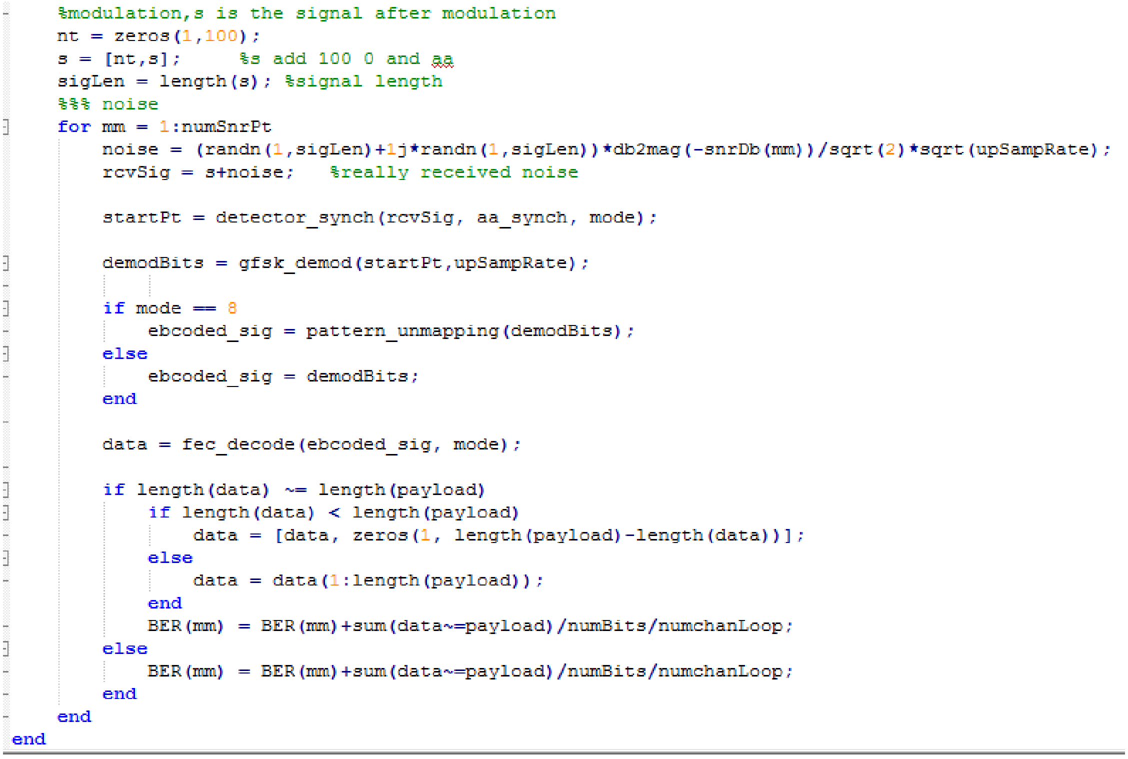
```
clear;
close all;
%%% coding decoding parameter
% mode:s = 2 or s = 8;
upSampRate = 2; % up sampling rate 3 or 2
mode1 = 1; % 1 or 2 or 8
mode2 = 2;
mode8 = 8;
BER1 = ble_digital_comm_course(mode1, upSampRate);
BER2 = ble_digital_comm_course(mode2, upSampRate);
BER8 = ble_digital_comm_course(mode8, upSampRate);

% plot BER for three mode
figure
semilogy((-4:2:12),BER1,'o-g');
hold on
semilogy((-4:2:12),BER2,'x-r');
hold on
semilogy((-4:2:12),BER8,'+-b');
legend('BER1','BER2','BER8')
grid on;
```

ble_digital_comm_course:

Deter the parameters of each mode and select function to process



```
function BER = ble_digital_comm_course(mode, upSampRate)
    numBits = 400;
    aa = [1 0 0 0 1 1 1 0 1 0 0 0 1 0 0 1 1 0 1 1 1 1 1 0 1 1 0 1 0 1 1 0];
    % aa is used for detect payload

    t = -8:1/upSampRate:8;
    B = 0.5; T = 1; h = 0.5;
    snrDb1 = -4;
    snrDb2 = 12;
    snrDb = snrDb1:2:snrDb2;
    numSnrPt = length(snrDb);
    % numSnrPt:how many different snr that we test

    numchanLoop = 100;
    BER = zeros(1,numSnrPt);

for k = 1:numchanLoop
    payload = round(rand(1,numBits));
    payload_aa = [aa, payload];
    if mode == 1
        packet = payload_aa;
        aa_decode = aa;
    elseif mode == 2
        packet = fec_enc(payload_aa,mode);
        aa_decode = fec_enc(aa,mode);
    elseif mode == 8
        packet = fec_enc(payload_aa,mode);
        packet = pattern_mapping(packet);
        aa_decode = fec_enc(aa,mode);
        aa_decode = pattern_mapping(aa_decode);
    end
```



```
    %modulation,s is the signal after modulation
    nt = zeros(1,100);
    s = [nt,s];      %s add 100 0 and aa
    sigLen = length(s); %signal length
    %%% noise
    for mm = 1:numSnrPt
        noise = (randn(1,sigLen)+1j*randn(1,sigLen))*db2mag(-snrDb(mm))/sqrt(2)*sqrt(upSampRate);
        rcvSig = s+noise;   %really received noise

        startPt = detector_synch(rcvSig, aa_synch, mode);

        demodBits = gfsk_demod(startPt,upSampRate);

        if mode == 8
            ebcoded_sig = pattern_unmapping(demodBits);
        else
            ebcoded_sig = demodBits;
        end

        data = fec_decode(ebcoded_sig, mode);

        if length(data) ~= length(payload)
            if length(data) < length(payload)
                data = [data, zeros(1, length(payload)-length(data))];
            else
                data = data(1:length(payload));
            end
            BER(mm) = BER(mm)+sum(data~=payload)/numBits/numchanLoop;
        else
            BER(mm) = BER(mm)+sum(data~=payload)/numBits/numchanLoop;
        end
    end
end
```

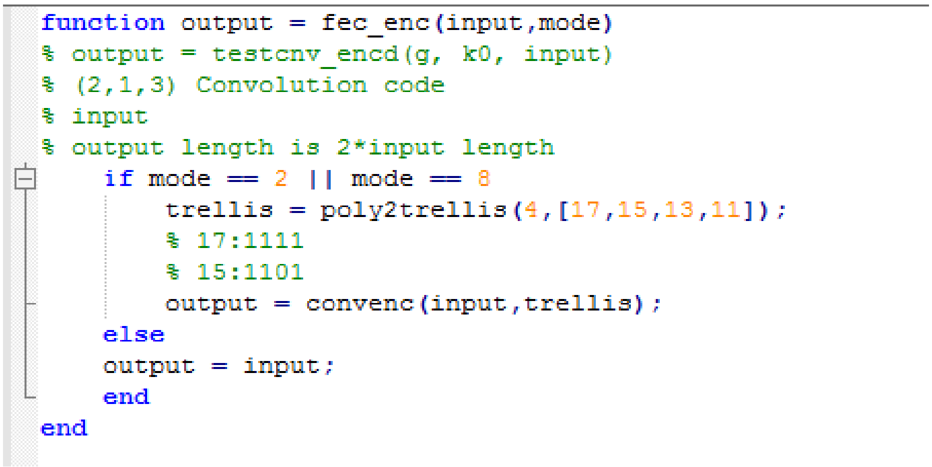
Inside this function we change “numchanLoop = 100”, this number can be changed to any positive integers, it means the result is the average of this positive integers times.

In our case we set this number to 100. The result comes from the 100 average 100 times

Convolutional Coding-decoding related:

Fec-enc:

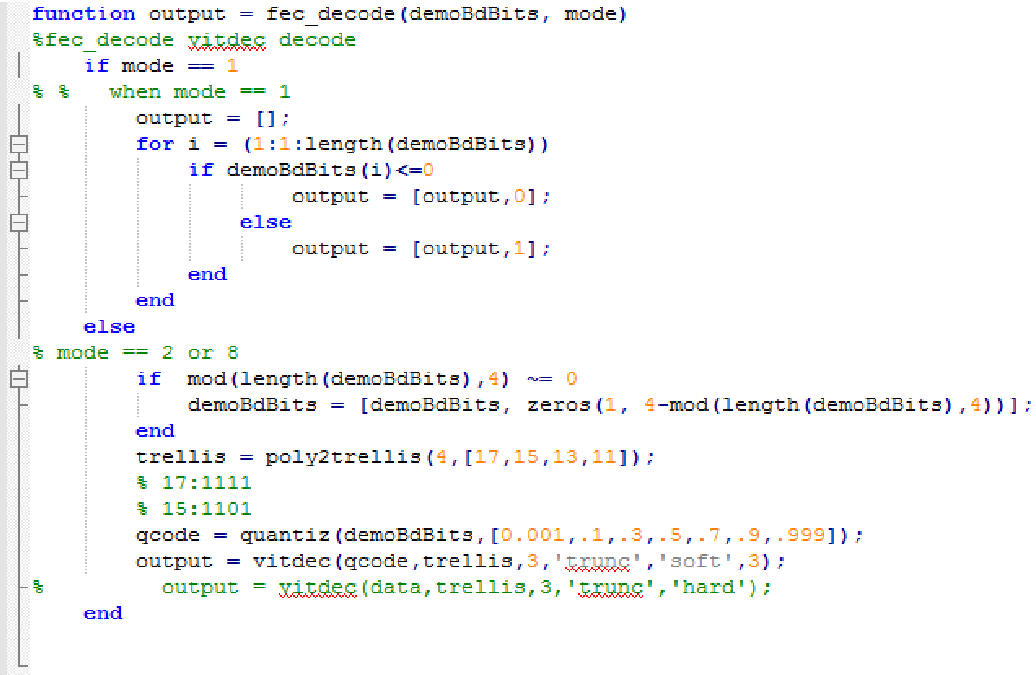
Coding:



```
function output = fec_enc(input,mode)
% output = testcnv_encd(g, k0, input)
% (2,1,3) Convolution code
% input
% output length is 2*input length
    if mode == 2 || mode == 8
        trellis = poly2trellis(4,[17,15,13,11]);
        % 17:1111
        % 15:1101
        output = convenc(input,trellis);
    else
    output = input;
    end
end
```

Fen_decode:

De-coding:



```
function output = fec_decode(demoBdBits, mode)
%fec_decode vitdec decode
    if mode == 1
% %   when mode == 1
        output = [];
        for i = (1:1:length(demoBdBits))
            if demoBdBits(i)<=0
                output = [output,0];
            else
                output = [output,1];
            end
        end
    else
% mode == 2 or 8
        if  mod(length(demoBdBits),4) ~= 0
            demoBdBits = [demoBdBits, zeros(1, 4-mod(length(demoBdBits),4))];
        end
        trellis = poly2trellis(4,[17,15,13,11]);
        % 17:1111
        % 15:1101
        qcode = quantiz(demoBdBits,[0.001,.1,.3,.5,.7,.9,.999]);
        output = vitdec(qcode,trellis,3,'trunc','soft',3);
%         output = vitdec(data,trellis,3,'trunc','hard');
    end
```

for this sentence: trellis = poly2trellis(4,[17,15,13,11])

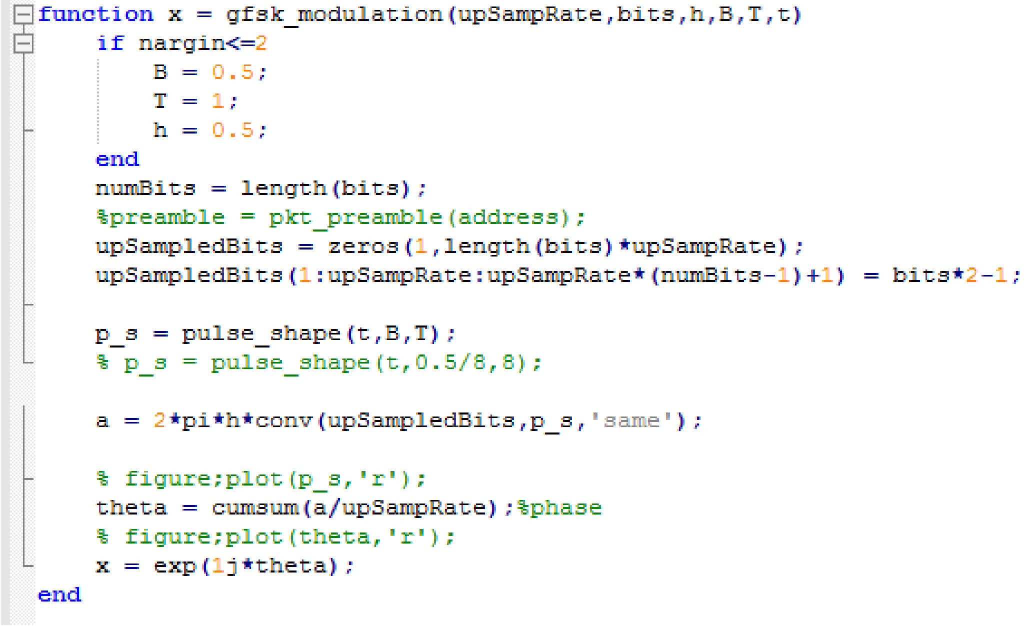
If we change the first number 4 to other number e.g, 5 or 6, it can improve our test result, but at the same time the complexility of the function also improved.

To make a trade off of complexility and test reuslt, we select constraint length as 4 and do our test best on constraint length 4.

GFSK Modulation and de-modulation related

Use GFSK method to modulation the code

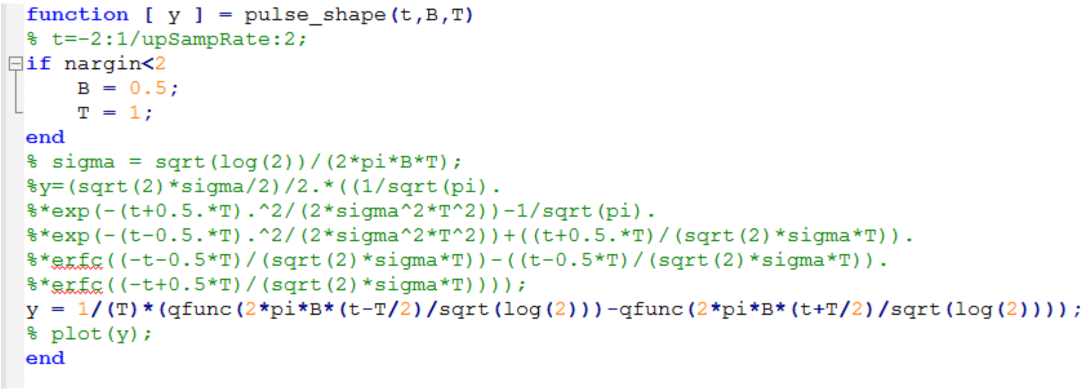
gfsk_modulation:



```
function x = gfsk_modulation(upSampRate,bits,h,B,T,t)
    if nargin<=2
        B = 0.5;
        T = 1;
        h = 0.5;
    end
    numBits = length(bits);
    %preamble = pkt_preamble(address);
    upSampledBits = zeros(1,length(bits)*upSampRate);
    upSampledBits(1:upSampRate:upSampRate*(numBits-1)+1) = bits*2-1;

    p_s = pulse_shape(t,B,T);
    % p_s = pulse_shape(t,0.5/8,8);

    a = 2*pi*h*conv(upSampledBits,p_s,'same');

    % figure;plot(p_s,'r');
    theta = cumsum(a/upSampRate);%phase
    % figure;plot(theta,'r');
    x = exp(1j*theta);
end
```

When we use GFSK to modulation the signal, we also need to shape the pulse as below:

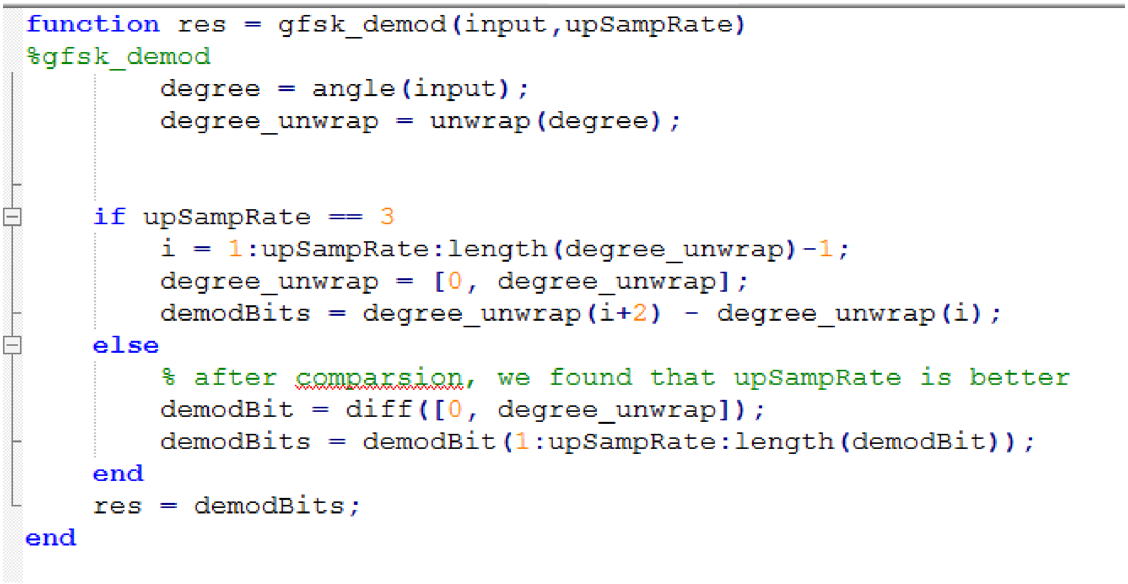
pulse_shape:



```
function [ y ] = pulse_shape(t,B,T)
% t=-2:1/upSampRate:2;
if nargin<2
    B = 0.5;
    T = 1;
end
% sigma = sqrt(log(2))/(2*pi*B*T);
%y=(sqrt(2)*sigma/2)/2.*((1/sqrt(pi).
%*exp(-(t+0.5.*T).^2/(2*sigma^2*T^2))-1/sqrt(pi).
%*exp(-(t-0.5.*T).^2/(2*sigma^2*T^2))+((t+0.5.*T)/(sqrt(2)*sigma*T)).
%*erfc((-t-0.5*T)/(sqrt(2)*sigma*T))-((t-0.5*T)/(sqrt(2)*sigma*T)).
%*erfc((-t+0.5*T)/(sqrt(2)*sigma*T))));
y = 1/(T)*(qfunc(2*pi*B*(t-T/2)/sqrt(log(2)))-qfunc(2*pi*B*(t+T/2)/sqrt(log(2))));
% plot(y);
end
```

Use GFSK method to de-modulation the code

gfsk_demod:

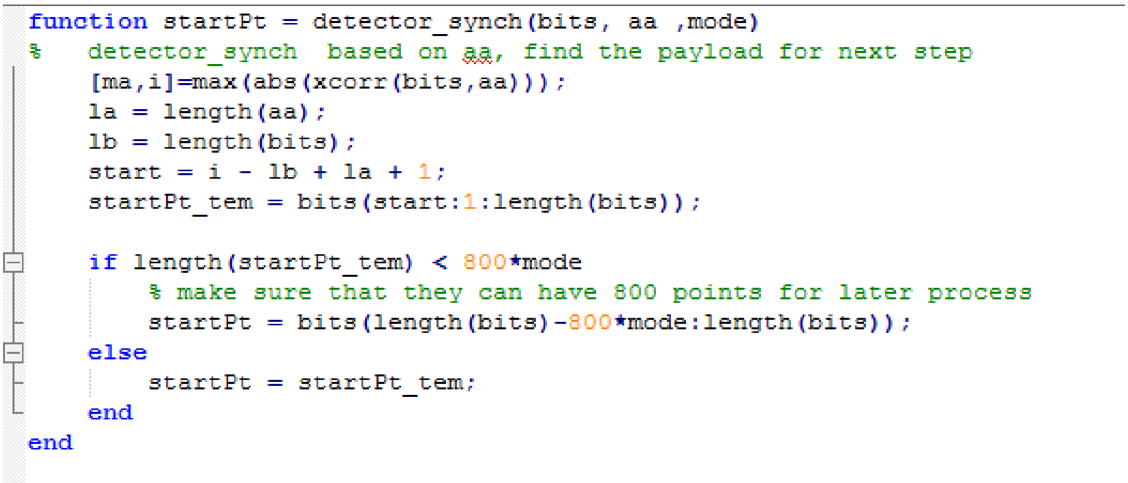


```
function res = gfsk_demod(input,upSampRate)
%gfsk_demod
        degree = angle(input);
        degree_unwrap = unwrap(degree);
        
        
    if upSampRate == 3
        i = 1:upSampRate:length(degree_unwrap)-1;
        degree_unwrap = [0, degree_unwrap];
        demodBits = degree_unwrap(i+2) - degree_unwrap(i);
    else
        % after comparsion, we found that upSampRate is better
        demodBit = diff([0, degree_unwrap]);
        demodBits = demodBit(1:upSampRate:length(demodBit));
    end
    res = demodBits;
end
```

Synch related

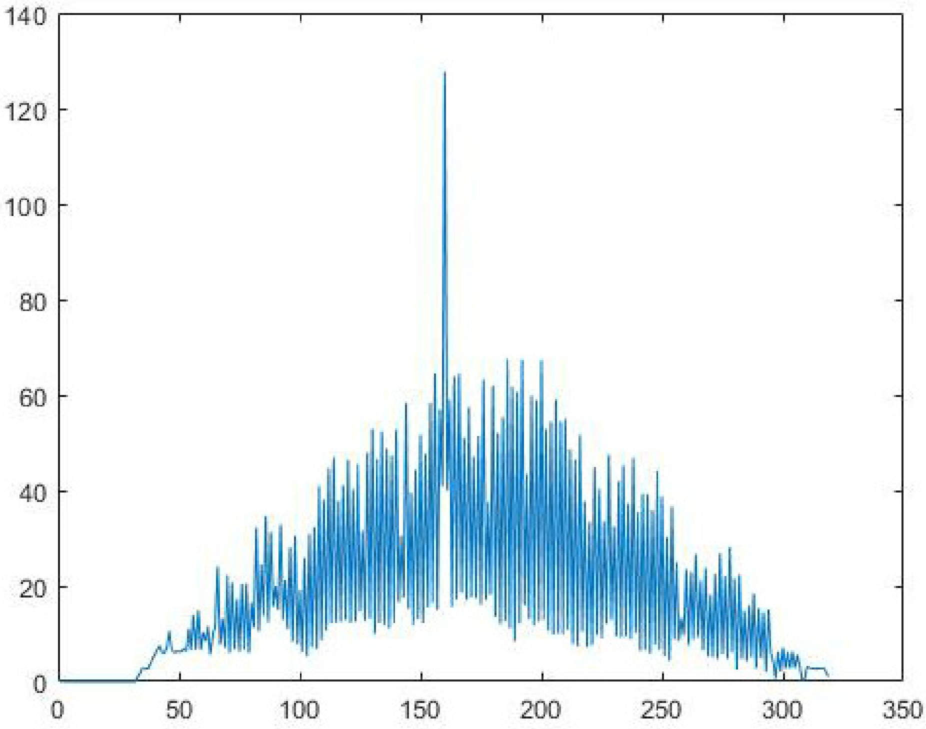
After noise added to the signal, synchronization is very important before de-coding the signal

detector_synch:

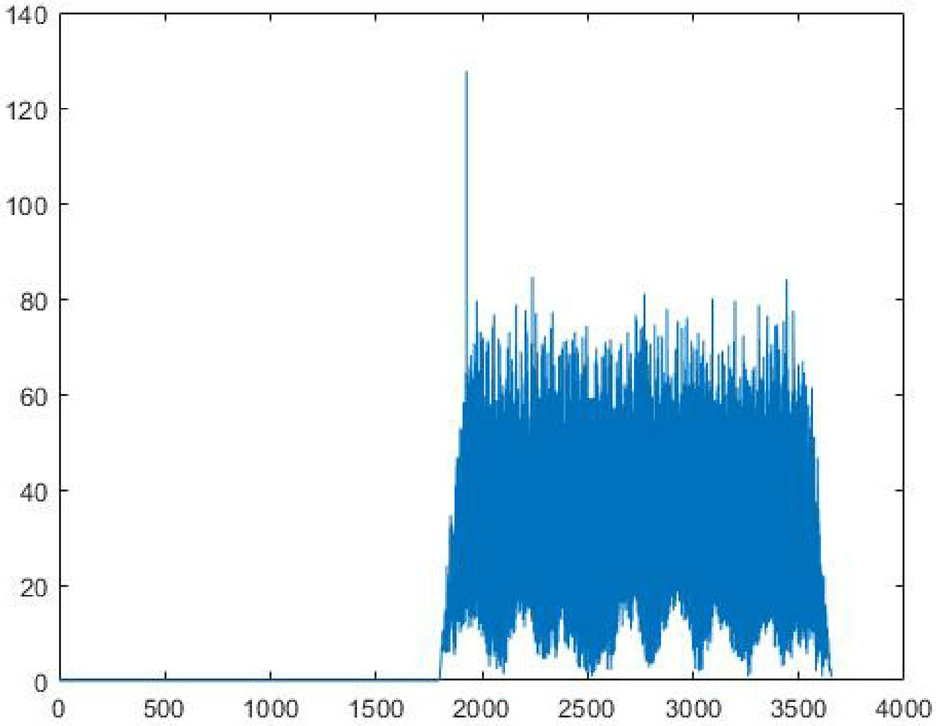


```
function startPt = detector_synch(bits, aa ,mode)
%   detector_synch  based on aa, find the payload for next step
    [ma,i]=max(abs(xcorr(bits,aa)));
    la = length(aa);
    lb = length(bits);
    start = i - lb + la + 1;
    startPt_tem = bits(start:1:length(bits));

    if length(startPt_tem) < 800*mode
        % make sure that they can have 800 points for later process
        startPt = bits(length(bits)-800*mode:length(bits));
    else
        startPt = startPt_tem;
    end
end
```

After we create this detector_synch function, we use random number to test the result shown as below:



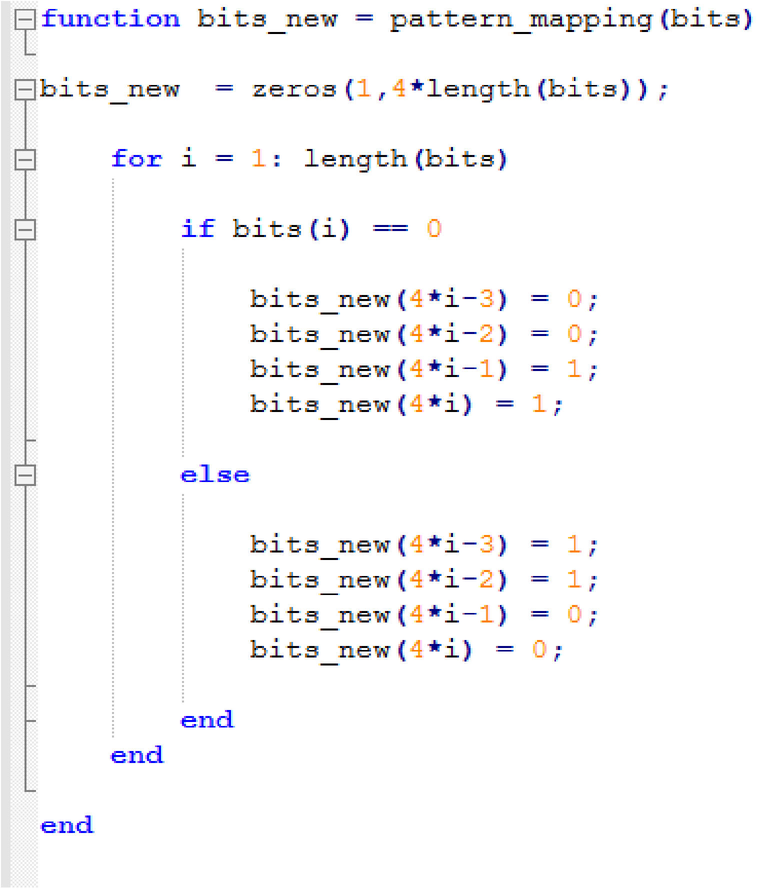
Then use this function to test out project shown as below:


140
120
100
80
60
40
20
0
0
500
1000
1500
2000
2500
3000
3500
4000

Pattern & mapping related

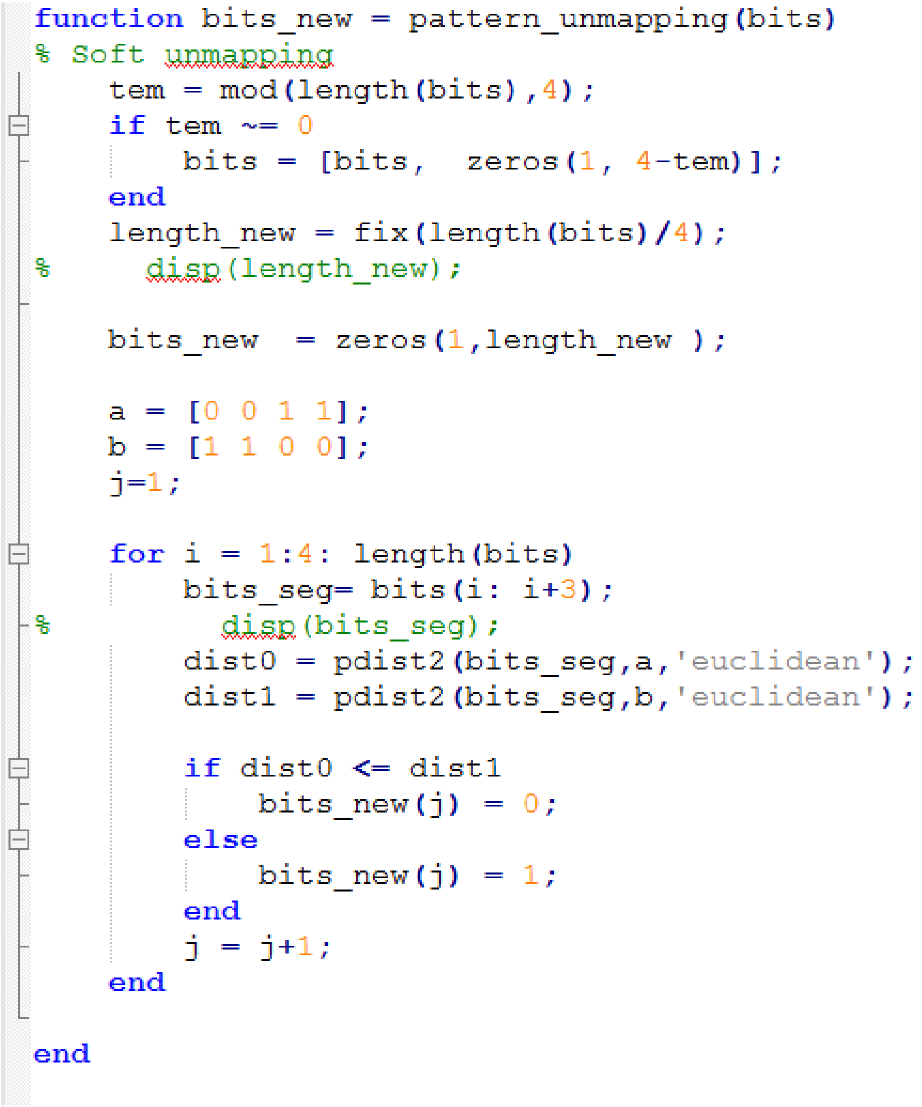
When we use Viterbi to send signals, in order to reduce noise and enlarge the capability of the channel, we need to map the signal we send, also, when we received the signal we need to unmap the signal we received from the channel and transferred them to real signal we need

pattern_mapping:



```
function bits_new = pattern_mapping(bits)

bits_new  = zeros(1,4*length(bits));

    for i = 1: length(bits)

        if bits(i) == 0

            bits_new(4*i-3) = 0;
            bits_new(4*i-2) = 0;
            bits_new(4*i-1) = 1;
            bits_new(4*i) = 1;

        else

            bits_new(4*i-3) = 1;
            bits_new(4*i-2) = 1;
            bits_new(4*i-1) = 0;
            bits_new(4*i) = 0;

        end
    end

end
```

pattern_unmapping



```
function bits_new = pattern_unmapping(bits)
% Soft unmapping
    tem = mod(length(bits),4);
    if tem ~= 0
        bits = [bits,  zeros(1, 4-tem)];
    end
    length_new = fix(length(bits)/4);
%     disp(length_new);

    bits_new  = zeros(1,length_new );

    a = [0 0 1 1];
    b = [1 1 0 0];
    j=1;

    for i = 1:4: length(bits)
        bits_seg= bits(i: i+3);
%         disp(bits_seg);
        dist0 = pdist2(bits_seg,a,'euclidean');
        dist1 = pdist2(bits_seg,b,'euclidean');

        if dist0 <= dist1
            bits_new(j) = 0;
        else
            bits_new(j) = 1;
        end
        j = j+1;
    end

end
```

Test result show:

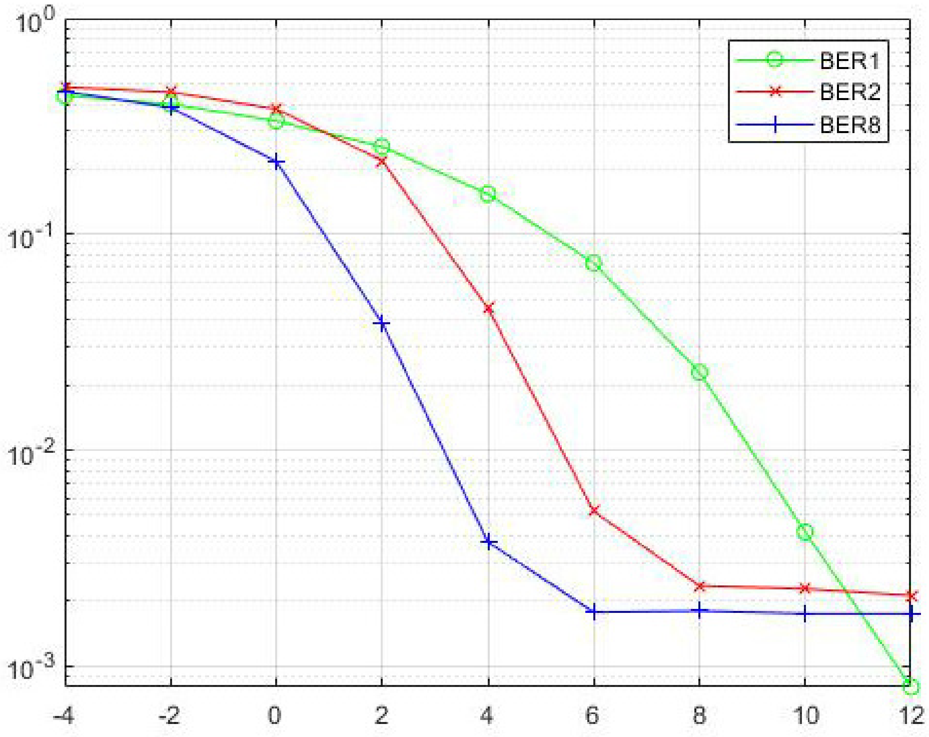
We have three modes that is mode=1, mode=2, mode=8, for each mode, after run the simulation we have below results:

In this chart:

BER1=Mode1

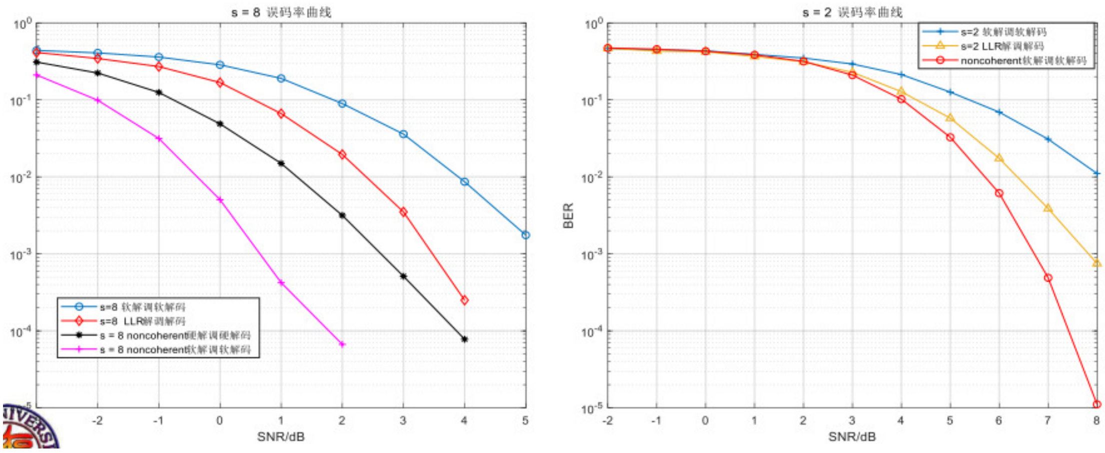
BER2=Mode2

BER3=Mode8



What we required for this project:

- For S=2, you should be no worse than 8dB @ BER=0.01
- For S=8, you get 10pts bonus for performance every dB better than 4dB @ BER = 0.01
- Submit a report with matlab code.



From above pictures we know that our result is fits what we required.