2024

Signals and Systems – CA4

روجا آقاجانی
810101380

تمرین 1-1)

```
function mapset = MapsetBuilder()
    mapset = cell(2,32);
    alphabet = 'abcdefghijklmnopqrstuvwxyz .,!";';
    for i = 1:32
        mapset{1,i} = alphabet(i);
        mapset{2,i} = dec2bin(i-1,5);
    end
end
```

در تابعی با نام MapsetBuilder، مشابه تمرین های قبلی الفبا را در یک استرینگ تعریف کرده، آنگاه در یک حلقه سطر اول را کارکتر و سطر دوم عدد باینری مربوط به آن را قرار میدهیم.

تمرین 1-2)

```
function EncodedMsg = coding_amp(inputMsg , bitRate)
    mapset = MapsetBuilder();
    Fs=100;
    ImbinarizedMsg = [];
    for char = inputMsg
        index = find([mapset{1,:}] == char, 1);
        if ~isempty(index)
            ImbinarizedMsg = [ImbinarizedMsg mapset{2, index}];
        else
            error(['Character ', char, ' does not exist in the dataset.']);
        end
    end

    stringLength = length(ImbinarizedMsg);

    loopCounter = 1;
    for i = 1 : bitRate : stringLength
        index1 = i;
        index2 = min(i + bitRate - 1, stringLength);
        outputMsg{loopCounter} = ImbinarizedMsg(index1 : index2);
        loopCounter = loopCounter + 1;
    end

    ImbinarizedMsg = outputMsg ;


    x = cell(1, 2^bitRate);

    for i=0:2^bitRate-1
        x{i+1} = dec2bin(i,bitRate);
    end

    y=zeros(1,2^bitRate);
    for i=1:2^bitRate
        y(1,i)=(i-1)/(2^bitRate-1);
    end


    [~,BMsgLength] = size(ImbinarizedMsg);
    t=zeros(BMsgLength,100);
    for i=1:100
        t(i,:)=linspace(i-1,i,Fs);
    end
    [~,BMsgLength] = size(ImbinarizedMsg);
    Coeffs=[];
    for i=1:BMsgLength
        for j=1:2^bitRate
            if strcmp(ImbinarizedMsg(1,i),x(1,j))==1
                Coeffs=[Coeffs y(1,j)];
            end
        end
    end

    [~,encMsgLen] = size(Coeffs);

    EncodedMsg=zeros(100,100);
    for i=1:encMsgLen
            EncodedMsg(i,:)=Coeffs(1,i).*sin(2*pi*t(i,:));
    end
```

ابتدا با استفاده از یک حلقه هر حرف پیام ورودی را با مپ ست مقایسه کرده، و عدد باینری متناظر با آن حرف را ذخیره میکنیم.

سپس پیام باینری را به استرینگ باینری با طول سرعت ورودی تبدیل کرده و سپس به شکل ماتریسی شامل رشته بیتها به اندازه سرعت پیام است ذخیره می کنیم و بعد در cell به نام x جایگشتهای ممکن رشته بیت با طول سرعت پیام را ذخیره می کنیم.

حال در y ضرایب مربوط به هر جایگشت در sin(2 πt) ذخیره کرده و در ماتریس t نیز نمونه های صدتایی در هر ثانیه را قرار میدهیم. در نهایت ماتریس ImbinarizedMsg را با x مقایسه کرده و ضرایب مربوط به هر جایگشت را در ماتریس ضرایب قرار میدهیم.

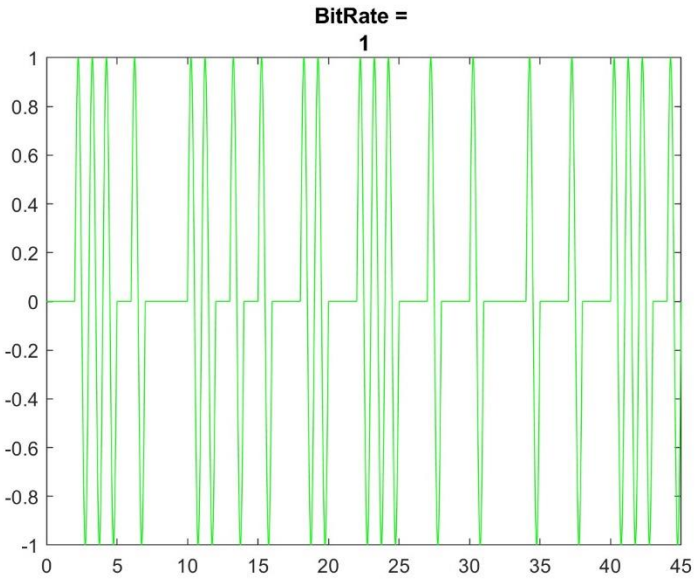
در آخر ماتریس خروجی به اندازه t تعریف کرده و هر سطر را در ضریب مربوطه sin(2 πt) ضرب کرده و بدین ترتیب خروجی حاصل میشود.

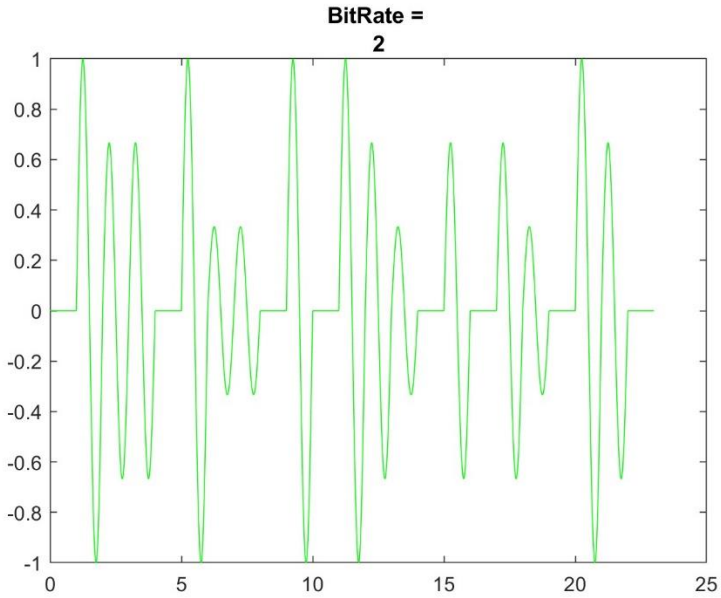
در نهایت برای رسم:

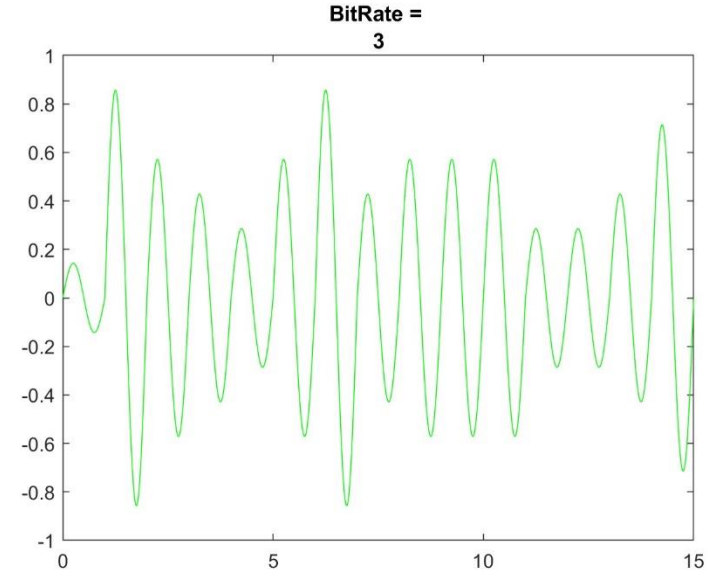
```
62        for i=1:BMsgLength
63            title(["BitRate = " , int2str(bitRate) ]);
64            plot(t(i,:),EncodedMsg(i,:),'g');
65            hold on
66        end
67    end
```

3-1) خروجی برای bitrate=1,2,3 :

```
1    test = coding_amp('hi there!' , 1);
2    test = coding_amp('hi there!' , 2);
3    test = coding_amp('hi there!' , 3);
```




BitRate =
2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
5
10
15
20
25


BitRate =
3
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
5
10
15

تمرین 1-4)

```
function dcdMsg=decoding_amp(EncodedSignal,bitRate)

    count=size(EncodedSignal);
    fs=100;
    Mapset=MapsetBuilder();

    correlationResult=[];
    t=zeros(count(1),100);

    for i=1:count(1)
        t(i,:)=linspace(i-1,i,fs);
    end

    for k=1:count(1)
        corr_2d=0.01*sum((2*sin(2*pi*t(k,:))).*(EncodedSignal(k,:)));
        correlationResult=[correlationResult corr_2d];
        correlationResult=double(correlationResult);
    end

    y=zeros(1,2^bitRate);
    for i=1:2^bitRate
        y(1,i)=(i-1)/(2^bitRate-1);
    end

    meanMatrix=zeros(1,2^bitRate-1);
    for l=1:2^bitRate-1
        meanMatrix(1,l)=(y(1,l+1)-y(1,l))/2+y(1,l);
    end

    x = cell(1, 2^bitRate);
    for i=0:2^bitRate-1
        x{i+1} = dec2bin(i,bitRate);
    end

    BinaryMsg=[];
    for i=1:count(1)
        for j=1:2^bitRate-1
            if abs(correlationResult(1,i))>meanMatrix(1,j) && abs(correlationResult(1,i))<= y(1,j+1)
                BinaryMsg=[BinaryMsg x(1,j+1)];
            end
            if abs(correlationResult(1,i))< meanMatrix(1,j) && abs(correlationResult(1,i))>= y(1,j)
                BinaryMsg=[BinaryMsg x(1,j)];
            end
            if abs(correlationResult(1,i))>1
                BinaryMsg=[BinaryMsg x(1,2^bitRate)];
            end

        end
    end
    BinaryMsg=cell2mat(BinaryMsg);

    BinMsgLen = length(BinaryMsg);
    counter = 1;
    for k = 1 : 5 : BinMsgLen
        index1 = k;
        index2 = min(k + 5 - 1, BinMsgLen);
        out{counter} = BinaryMsg(index1 : index2);
        counter = counter + 1;
    end
    BinaryMsg = out ;
```

```
    dcdMsg=[];
    MapsetLen=size(Mapset);
    msgLen=round(count(1)*bitRate/5);
    for m=1:msgLen
        for n=1:MapsetLen(1,2)
            if strcmp(BinaryMsg(1,m),Mapset(2,n))==1
                dcdMsg=[dcdMsg  Mapset(1,n)];
            end
        end
    end
    dcdMsg=strjoin(dcdMsg,'');
    disp(dcdMsg)
end
```

ابتدا تابع MapsetBuilder را فراخوانی کرده تا مپ ست مدنظر تشکیل شود. آنگاه مانند بخش قبل cell به نام x تعریف میکنیم که تمام جایگشتهای ممکن رشته های باینری به طول سرعت ارسال را در آن ذخیره میکنیم و در ماتریسی به نام y نیز مانند بخش قبل ضرایب مربوط به هرجایگشت را ذخیره میکنیم.

ماتریس دیگری به نام t تعریف میکنیم که نمونه های صدتایی در هرثانیه را در آن میریزیم.

با استفاده از یک حلقه کورولیشن گیری بین ماتریس ورودی پیام کدگذاری شده و$\sin(2\pi t)$ وضرب ضریب ۰.۰۱ در کورولیشن، نتیجه را در سلول correlationResult ذخیره میکنیم در نهایت آن را به float تغییر میدهیم.

جهت decoding: ماتریسی به نام meanMatrixبه اندازه 2 به توان bitrate تشکیل میدهیم و در آن میانگین هر دو درایه متوالی ماتریس y را ذخیره میکنیم . سپس کورولیشن meanMatrix با y را مقایسه میکنیم:

اگر بین عدد وسط و عدد کوچکتر بود، x متناظر با عدد کوچکتر را به آن نسبت میدهیم.

اگر بین عدد وسط و عدد بزرگتر بود x متناظر به عدد بزرگتر را به آن نسبت میدهیم.

و درBinaryMsg ذخیره میکنیم . درنهایت cell را به ماتریس تبدیل میکنیم.

با در دست داشتن پیام باینری، هر 5 بیت را جدا کرده و با مپ ست کورولیشنش را محاسبه میکنیم تا حرف مربوط به هر پنج بیت بدست آید. در نهایت نیز با strjoin این حروف را در یک استرینگ در کنار هم قرار میدهیم.

خروجی:

```
test = coding_amp('hi there!' , 1);
result = decoding_amp(test,1);
test = coding_amp('hi there' , 2);
result = decoding_amp(test,2);
test = coding_amp('hi there!' , 3);
result = decoding_amp(test,3);
```

Command Window

```
>> p1test
hi there!
hi there
hi there!
```

در bit-rate=2 علامت تعجب پیام حذف شد تا بخش پذیری بر بیت ریت رعایت شود.

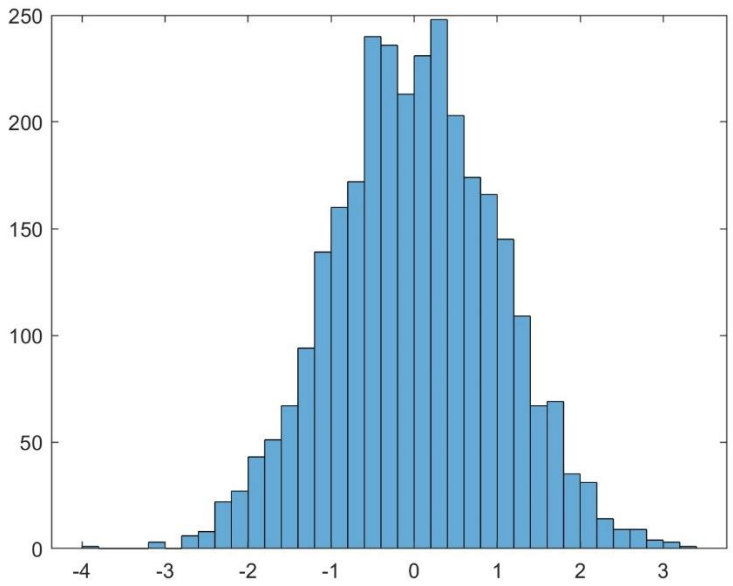
تمرین 1-5)

```
noise = randn(1 , 3000);

disp(["Mean of the Gaussian Noise is : " , mean(noise) ]);
disp(["Variance of the Gaussian Noise is : " , var(noise) ]);

disp(["Rounded mean of the Gaussian Noise is : " , round(mean(noise)) ]);
disp(["Rounded variance of the Gaussian Noise is : " , round(var(noise)) ]);

histogram(noise)
```

Command Window

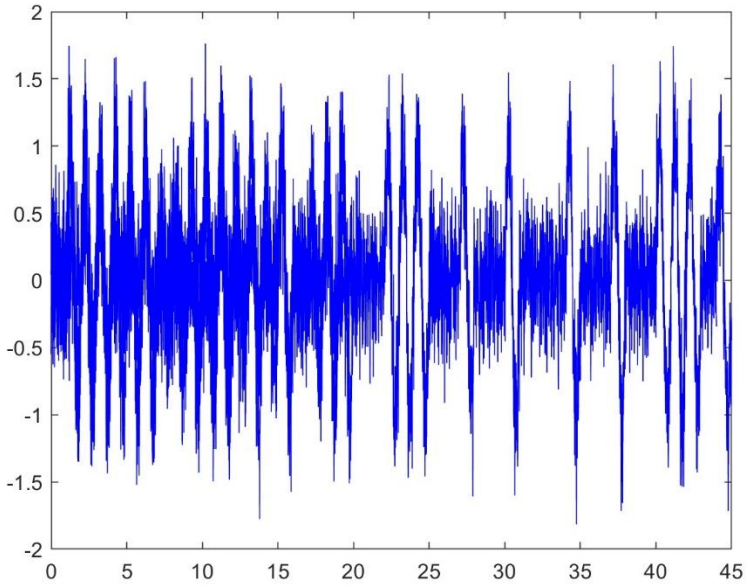
```
>> GaussianNosie
    "Mean of the Gaussian Noise is : "    "0.016254"

    "Variance of the Gaussian Noise is : "    "0.99076"

    "Rounded mean of the Gaussian Noise is : "    "0"

    "Rounded variance of the Gaussian Noise i..."    "1"
```

و تصویر هیستوگرام نویز مربوطه:



تمرین 1-6)

با افزودن نویز با سیگمای 0.3:



تصویر موج با نویز و بیت ریت 0.3

پیام دیکود شده:

```
" Sigma = "     "0.3"

"bit rate = 1, docoded: "     "hi there!"

"bit rate = 2, docoded: "     "hi;vh,,s"

"bit rate = 3, docoded: "     "hg phdre,"
```

تمرین 1-7)

با یک حلقه for برای سیگما از 0 تا 2 به صورت 0.01 پیش میرویم تا ببینیم کجا نویز پیام را غیر قابل تشخیص یا با خطا می کند.

```
for counter=0:0.01:2
    test1 = coding_amp('hi there!' , 1);
    test1wNoise = NoiseAdder(test1,counter);
    result1 = decoding_amp(test1wNoise,1);
    test2 = coding_amp('hi there' , 2);
    test2wNoise = NoiseAdder(test2,counter);
    result2 = decoding_amp(test2wNoise,2);
    test3 = coding_amp('hi there!' , 3);
    test3wNoise = NoiseAdder(test3,counter);
    result3 = decoding_amp(test3wNoise,3);
    disp([" Sigma = " , counter ]);
    disp(["bit rate = 1, docoded: " , result1]);
    disp(["bit rate = 2, docoded: " , result2]);
    disp(["bit rate = 3, docoded: " , result3]);
end
```

با اجرای کد:

```
" Sigma = "    "0.05"

"bit rate = 1, docoded: "    "hi there!"

"bit rate = 2, docoded: "    "h"rvgp,s"

"bit rate = 3, docoded: "    "hi there!"
```

در نویز با سیگمای 0.05 پیام ارسالی با بیت ریت 2 دچار نویز می شود.

```
" Sigma = "    "0.26"

"bit rate = 1, docoded: "    "hi there!"

"bit rate = 2, docoded: "    "hi t"p,s"

"bit rate = 3, docoded: "    "hizthmnev"
```

در نویز با سیگمای 0.026، پیام ارسالی با بیت ریت 3 دچار نویز می شود.

```
" Sigma = "    "1.3"

"bit rate = 1, docoded: "    "hi there!"

"bit rate = 2, docoded: "    "ev tp"r;"

"bit rate = 3, docoded: "    "dhlt;;;"n"
```

در نویز با سیگمای 1.3 پیام ارسالی با بیت ریت 1 دچار نویز می شود.

تمرین 1-8)

با توجه به نتایج بدست آمده:

Bit-rate = 3 -> maximum noise variance tolerated = 0.025^2 = 0.000625

Bit-rate = 2 -> maximum noise variance tolerated = 0.004^2 = 0.000016

Bit-rate = 1 -> maximum noise variance tolerated = 1.2^2 = 1.44

تمرین 1-9)

با قدرت فرستنده بیشتر، میتوان سیگنال با دامنه بالاتری ارسال کرد و در نتیجه threshold های انتخابی جهت تصمیم گیری بیشتر بود و لذا اثر نویز کمتر می شد.

تمرین 1-10)

بدون وجود نویز و ارسال خالص سیگنال، فرکانس نمونه برداری بیشترین نرخ بیت را معین میکند. این موضوع یعنی با فرکانس نمونه‌برداری به اندازه کافی بزرگ که با آن سیگنال را به خوبی بازنمایی شود، نیازی به نرخ بیت اضافی نیست.

در هر هرتز از فرکانس نمونه‌برداری، تقریباً دو بیت اطلاعات ذخیره می‌شود که به آن قاعده نایکیست -شانون معروف است؛ یعنی می‌گوید برای بازسازی سیگنال با فرکانس بالاتر از فرکانس نمونه‌برداری، حداقل دو برابر فرکانس نمونه‌برداری نیاز است. بنابراین، در صورت عدم وجود نویز، نرخ بیت محدودیتی ندارد و می‌توان آن را تا حد معقولی با فرکانس نمونه‌برداری افزایش داد.

همچنین:

در اوایل سال 1924، یک مهندس AT&T، هنری نایکیست، متوجه شد که حتی یک کانال عالی نیز ظرفیت انتقال محدودی دارد. او معادله ای استخراج کرد که حداکثر نرخ داده را برای یک کانال بی صدا با پهنای باند محدود بیان می کند. در سال 1948، کلود شانون کار نایکوئیست را بیشتر پیش برد و موضوع کانالی را که در معرض نویز تصادفی (یعنی ترمودینامیکی) قرار داشت را به آن گسترش داد (شانون، 1948). این مقاله مهمترین مقاله در تمام نظریه اطلاعات است.

نرخ داده بر سرعت انتقال داده ها تأثیر می گذارد. یک نکته بسیار مهم در ارتباطات داده این است که ما با چه سرعتی می توانیم داده ها را بر حسب بیت در ثانیه از طریق یک کانال ارسال کنیم. سرعت داده به 3 عامل بستگی دارد:

1. پهنای باند موجود
2. تعداد سطوح در سیگنال دیجیتال
3. کیفیت کانال - سطح نویز

دو فرمول نظری برای محاسبه نرخ داده ایجاد شد: یکی توسط نایکیست برای یک کانال بدون نویز، دیگری توسط شانون برای یک کانال نویز.

1. کانال بدون نویز: نرخ بیت نایکوئیست: برای یک کانال بدون نویز، فرمول نرخ بیت نایکوئیست حداکثر نرخ بیت نظری را تعیین می کند.

نایکیست ثابت کرد که اگر یک سیگنال دلخواه از طریق فیلتر پایین گذر پهنای باند اجرا شود، سیگنال فیلتر شده را می توان با ساختن نمونه های 2* پهنای باند (دقیق) در هر ثانیه به طور کامل بازسازی کرد. نمونه برداری از خط سریعتر از 2* پهنای باند بار در ثانیه بی معنی است زیرا اجزای فرکانس بالاتری که چنین نمونه برداری می تواند بازیابی کند قبلاً فیلتر شده اند. اگر سیگنال از سطوح گسسته L تشکیل شده باشد، قضیه نایکوئیست بیان می کند:

نرخ بیت = 2 * پهنای باند * log2 (L) بیت در ثانیه

در معادله بالا، پهنای باند پهنای باند کانال، L تعداد سطوح سیگنالی است که برای نمایش داده ها استفاده می شود، و BitRate نرخ بیت بر حسب بیت در ثانیه است.

پهنای باند یک مقدار ثابت است، بنابراین نمی توان آن را تغییر داد. از این رو، نرخ داده به طور مستقیم با تعداد سطوح سیگنال متناسب است.

تمرین 1-11)

بله .این کار عملکرد سیگنال را نسبت به نویز مقاومتر میکند .در این روش عمل پاور (قدرت) سیگنال را تقویت میکنیم؛ مثلا با ضرب کردن ۱۰ به جای۲ در قبل از سینوس مقدار بیشنیه کورووليشن از 1 به 5 تغییر میکند و همین باعث میشودکه فاصله بین threaholdهای درنظر گرفته بیشتر شود و در نتیجه خطا کمتر میشود.

تمرین 1-12)

اینترنتadsl با سرعت متوسط بین ۱ تا ۱۶ مگابیت بر ثانیه و adsl2+ با سرعت ۲۴ مگابیت بر ثانیه از سرویس های متداول در ایران هستند.

ما در این تمرین اطلاعات را با حداکثر سرعت 3 بیت بر ثانیه ارسال کردیم!