باسمه تعالی

دانشگاه صنعتی شریف

دانشکده مهندسی برق

درس سیستم‌های مخابراتی

پروژه‌ی پایانی درس: شبیه‌سازی یک سیستم مخابرات دیجیتال

استاد درس: دکتر پاکروان

امیرحسین افشارراد

۹۵۱۰۱۰۷۷

۱۱ بهمن ۱۳۹۷

۱ پیاده‌سازی بلوک‌ها به صورت مجزّا

در این قسمت به توضیح توابع نوشته‌شده به منظور شبیه‌سازی کامل یک سیستم مخابرات دیجیتال می‌پردازیم.

۱.۱ تابع Divide

وظیفه‌ی این تابع آن است که یک دنباله را به عنوان ورودی دریافت کند و در خروجی دو دنباله با طول نصف دنباله‌ی اولیه ارائه دهد که اعضای دنباله‌ی اولیه به صورت یکی‌درمیان در آن‌ها قرار گرفته‌اند. کد این تابع در ادامه قابل مشاهده می‌باشد.

```
function [out1, out2] = Divide(inp)
out1 = inp(1:2:end);
out2 = inp(2:2:end);
```

۲.۱ تابع PulseShaping

این تابع در ورودی یک دنباله از صفر و یک و دو شکل موج دریافت می‌کند که یکی متناظر با بیت صفر و دیگری متناظر با بیت یک می‌باشد. خروجی این دنباله شکل موج پیوسته‌ای است که به ازای هر کدام از بیت‌های صفر یا یک، شکل موج متناظر با آن را دارا می‌باشد. همچنین ضروری است طول دو شکل موج متناظر با صفر و یک با هم برابر باشد، لذا چنان‌چه این شرط برقرار نباشد تابع پیغام خطا نمایش می‌دهد. در ادامه کد تابع نیز قابل مشاهده است.

```
function out = PulseShaping(sequence, ZeroWaveform, OneWaveform)
if (length(ZeroWaveform) ~= length(OneWaveform))
    disp ('Error! The two waveforms must have equal lengths!')
    return
end
n = length(sequence);
L = length(ZeroWaveform);
out = zeros(1,n*L);
for i = 1 : n
    if (sequence(i))
        out((i-1)*L+1 : i*L) = OneWaveform;
    else
        out((i-1)*L+1 : i*L) = ZeroWaveform;
    end
end
```

۳.۱ تابع AnalogMod

این تابع وظیفه‌ی آن را دارد که پس از مدولاسیون دیجیتال (توسط تابع PulseShaping) عملیات مدولاسیون آنالوگ را روی شکل موج حاصل از قسمت قبل انجام دهد. برای این کار دو ورودی x_1 و x_2 به همراه فرکانس نمونه برداری f_s و فرکانس حامل f_c در ورودی دریافت می‌شود؛ دو سیگنال ورودی، یکی در حامل سینوسی و دیگری در حامل کسینوسی ضرب شده و در نهایت جمع دو سیگنال حاصل به عنوان خروجی مدوله‌شده ارائه می‌شود. دقت کنید که با توجه به این که

در شبیه‌سازی سیگنال‌های پیوسته از نمونه‌های آن‌ها استفاده می‌شود، نیاز به فرکانس نمونه‌برداری داریم تا محور زمان (t) را در نقاطی با فاصله‌های $\frac{1}{fs}$ شبیه‌سازی کنیم. در ادامه کد متلب این تابع نیز قابل مشاهده است.

```
function out = AnalogMod(x1, x2, fs, fc)
T = (max(length(x1), length(x2))-1)/fs;
t = 0 : 1/fs : T;
out = x1.*cos(2*pi*fc*t) + x2.*sin(2*pi*fc*t);
```

۴.۱ تابع Channel

این تابع شبیه‌ساز یک کانال مخابراتی است. این کانال بدون اعوجاج است و پهنای باند معینی از سیگنال را عبور می‌دهد؛ بنابراین عملاً یک فیلتر میان‌گذر است. در نتیجه تابع Channel عملاً یک فیلتر میان‌گذر است که در ورودی سیگنال اصلی، فرکانس نمونه‌برداری، فرکانس مرکزی، و پهنای باند را دریافت کرده و در خروجی سیگنال گذریافته از کانال (فیلتر) را به دست می‌دهد. کد متلب این تابع در ادامه آمده است.

```
function out = Channel(x, fs, f0, BW)
h = BPF(1001, f0-BW/2, f0+BW/2, fs);
out = filter(h, 1, x);
```

همان طور که مشاهده می‌شود، در تعریف تابع Channel از تابع BPF استفاده شده است که دقیقاً یک فیلتر میان‌گذر است. کد این تابع نیز در ادامه قابل مشاهده است.

```
function h = BPF(L,lowF,higF,FS)

    %{
        Inputs:
            L: Lenght of filter
            lowF: Low frequency
            higF: High frequency
            FS: Sampling frequency
        Outut:
            h: Impulse response of the filter
    %}
    beta = 3;
    h = fir1(L-1,[2*lowF/FS,2*higF/FS], kaiser(L,beta));
    h = h(:);

end
```

ضمناً ابتدا برای ایده‌آل در نظر گرفتن کانال و حذف تأخیر، تابعی دیگری نوشتیم (که در نهایت مورد از آن استفاده نکردیم تا اثر تأخیر را نیز در شبیه‌سازی مشاهده کنیم.) آن تابع را IdealChannel نامیدیم و کد آن نیز در ادامه ضمیمه شده است. روش ساخت چنین فیلتری (که البته در واقعیت نمی‌تواند وجود داشته باشد و به همین دلیل هم در شبیه‌سازی از آن استفاده نشده است) آن است که عمل فیلتر کردن را در حوزه فرکانس و با صفر کردن تبدیل فوریه‌ی نواحی نامطلوب فرکانسی انجام می‌دهیم.

```
function out = IdealChannel(x, fs, f0, BW)
X = fft(x);
L = length(X);
f0_normalized = f0 / (fs/2);
BW_normalized = BW / (fs/2);
N =ceil((L-1)/2);
Nh = (f0_normalized + BW_normalized/2)*N+1;
Nl = (f0_normalized - BW_normalized/2)*N+1;
Y = zeros(1,L);
Y(Nl:Nh) = X(Nl:Nh);
Y(2*(L/2+1)-Nh : 2*(L/2+1)-Nl) = conj(fliplr(X(Nl : Nh)));
out = ifft(Y);
```

۵.۱ تابع AnalogDemod

این تابع سیگنال عبوری از کانال را همراه با فرکانس نمونه‌برداری، پهنای باند پیام، و فرکانس حامل به عنوان ورودی دریافت کرده و عمل demodulation را بر روی آن انجام می‌دهد. دقت داریم که از ابتدا دو سیگنال x_1 و x_2 با دو حامل سینوسی و کسینوسی به عنوان پیام ارسال شده بودند. برای بازیابی هر یک از این دو سیگنال، کافی است مجدداً سیگنال مدوله‌شده x_c را در سینوس یا کسینوس ضرب کرده، از یک فیلتر پایین‌گذر با پهنای باند پیام عبور دهیم. همچنین برای ساخت فیلتر پایین‌گذر نیز از همان تابع BPF که در قسمت ۴.۱ معرفی شد استفاده کرده‌ایم و برای تبدیل آن به یک فیلتر پایین‌گذر، کران پایین باند عبوری را eps قرار داده‌ایم تا به جز پیام DC، همه‌ی فرکانس‌های پایین را عبور دهد. نهایتاً کد تابع را نیز در ادامه مشاهده می‌کنید.

```
function [out1, out2] = AnalogDemod(x, fs, BW, fc)
T = (length(x)-1)/fs;
t = 0 : 1/fs : T;
out1 = x.*cos(2*pi*fc*t);
out2 = x.*sin(2*pi*fc*t);
h = BPF(1000, eps, BW, fs);
out1 = filter(h, 1, out1);
out2 = filter(h, 1, out2);
```

۶.۱ تابع MatchedFilt

این تابع وظیفه‌ی پیاده‌سازی یک Matched Filter را به منظور بازیابی پیام دیجیتال به عهده دارد. از تئوری درس به خاطر داریم که اگر شکل پالس متناظر با یک بیت را با $g(t)$ نشان دهیم، در این صورت فیلتر مورد استفاده به شکل $kg(T-t)$ است که باید بر روی سیگنال دریافتی اثر داده شود؛ سپس در لحظه T از خروجی نمونه‌برداری شود. در این صورت، مطابق با معادله‌ی ۱، کافی است همبستگی (بدون شیفت زمانی) سیگنال دریافتی و شکل موج مربوطه را محاسبه کنیم.

$$x(t) * g(T-t)\Big|_{t=T} = \int_{-\infty}^{\infty} x(\tau)g\big(T-(t-\tau)\big)d\tau\Big|_{t=T} = \int_{-\infty}^{\infty} x(\tau)g(\tau)d\tau \qquad (۱)$$

تابع MatchedFilt عملیات مذکور را با استفاده از دو شکل موج متناظر با صفر و یک بر روی سیگنال دریافتی انجام می‌دهد و در خروجی، میزان همبستگی با هر یک از دو شکل موج را همراه با تصمیم نهایی مبتنی بر صفر یا یک بودن بیت متناظر (با مقایسه‌ی مقدار همبستگی با دو شکل موج و انتخاب همبستگی بیشتر به عنوان بیت تخمینی) ارائه می‌دهد. دقت کنید که این فرایند باید به ازای هر بیت انجام شود تا کل دنباله‌ی ارسالی بازیابی شود؛ بنابراین در این تابع یک حلقه‌ی for با طول دنباله‌ی ارسالی وجود دارد که برای هر بیت، فرایند مذکور را انجام می‌دهد. نهایتاً خروجی‌های تابع مقدار همبستگی و بیت تخمینی را برای کل دنباله در بر دارد. ضمناً لازم به ذکر است مقدار همبستگی خروجی با استفاده از norm سیگنال نرمالیزه شده است تا اندازه‌ی خروجی آن بیشتر از یک نباشد. کد تابع نیز در ادامه ضمیمه شده است.

```
function [ZeroCorr, OneCorr, y_est] = MatchedFilt(x, ZeroWaveform,
    OneWaveform)
L = length(ZeroWaveform);
N = length(x)/L;
y_est = zeros(1,N);
ZeroCorr = zeros(1,N);
OneCorr = zeros(1,N);
for i = 1 : N
    ZeroCorr(i) = sum(x((i-1)*L+1:i*L).*ZeroWaveform)/norm(
       ZeroWaveform)/norm(x((i-1)*L+1:i*L));
    OneCorr(i) = sum(x((i-1)*L+1:i*L).*OneWaveform)/norm(
       OneWaveform)/norm(x((i-1)*L+1:i*L));
    if (ZeroCorr(i) > OneCorr(i))
        y_est(i) = 0;
    else
        y_est(i) = 1;
    end
end
```

۲ انتقال دنباله‌ی تصادفی صفر و یک

۱.۲ استفاده از پالس مربعی برای انتقال پیام

۱.۱.۲ شبیه‌سازی فرآیند ارسال پیام بدون نویز

- طول دنباله‌ی ارسالی را در این قسمت ۲۰ بیت در نظر گرفتیم. البته ۲۰ بیت برای یک پیام بسیار کم است، اما برای شبیه‌سازی ارسال و دریافت بدون نویز می‌تواند کافی باشد، چرا که تعداد بیت‌ها پیچیدگی خاصی به سیستم اضافه نمی‌کند. همچنین با توجه به این که نویزی در سیستم وجود ندارد، انتظار داریم خطای بازسازی صفر باشد که اگر سیستم به درستی کار نکند؛ احتمال آن که به صورت تصادفی خروجی صحیح مشاهده کنیم (با توجه به ۲۰ بیتی بودن پیام) آن قدر ناچیز است که این تعداد بیت می‌تواند کافی باشد. در قسمت ۳.۱.۲ و به منظور رسم constellation diagram، تعداد بیت‌ها را افزایش خواهیم داد.

- برای شبیه‌سازی کامل سیستم مخابرات دیجیتال، علاوه بر توابع قسمت قبل، یک تابع جدید نیز با نام Combine می‌نویسیم که در انتهای فرآیند انتقال و پس از تخمین بیت‌های دریافتی، دو دنباله‌ی حاصل را به صورت یکی‌درمیان در کنار هم قرار دهد و نهایتاً به یک دنباله (که همان دنباله‌ی ارسالی اولیه بوده است) برسد. کد این تابع در ادامه آمده است.

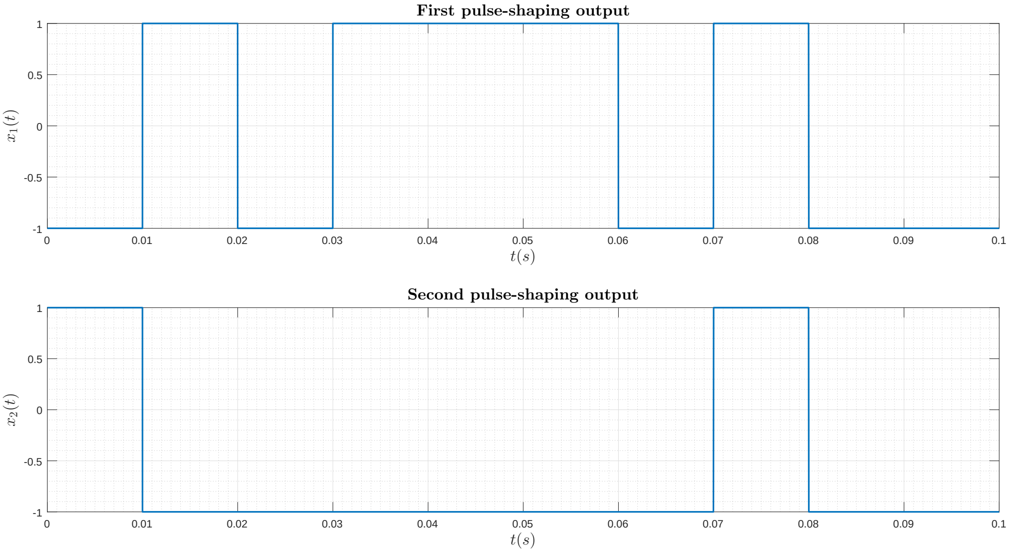
```
function y = Combine(x1, x2)
L = length(x1);
y = zeros(1,2*L);
y(1:2:end-1) = x1;
y(2:2:end) = x2;
```

- نتایج شبیه‌سازی مطابق با انتظاری که از سیستم بدون نویز داشتیم، خطای صفر و بازیابی کامل پیام را نشان می‌دهند. ضمناً یک نکته‌ی مهم آن است که در این بخش، اگر چه اثر تأخیر کانال و فیلتر مورد استفاده در demodulation را در نظر نگرفتیم، اما مشکلی پیش نیامد. علت این امر استفاده از شکل موج مربعی است و مشاهده می‌شود که اگر این شکل موج تغییر کند و به صورت سینوسی در بیاید (قسمت ۲.۲)، لازم است راه حلی برای از بین بردن اثر این دو تأخیر در نظر بگیریم.

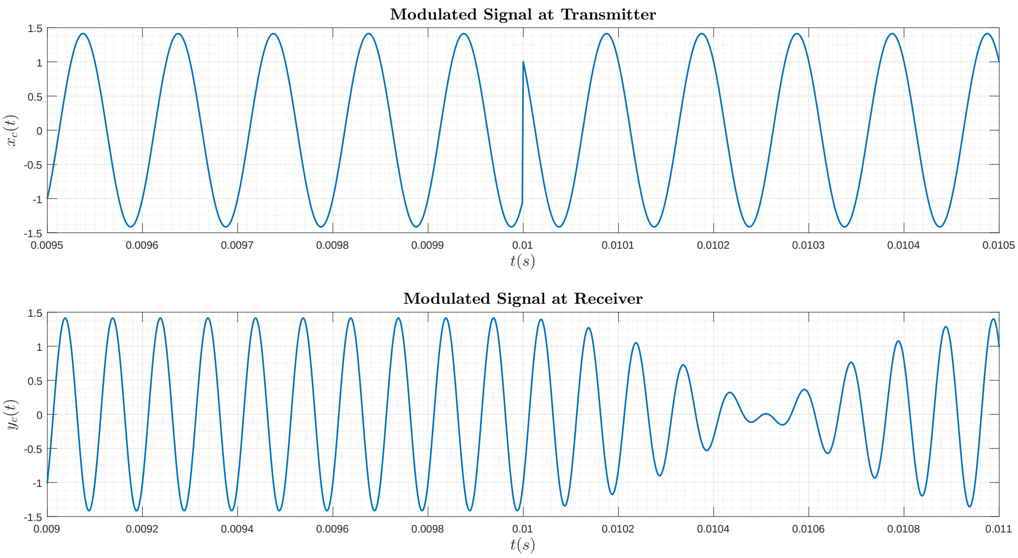
- شکل ۱ دنباله‌های ارسالی را پس از مدولاسیون دیجیتال (خروجی تابع PulseShaping) نمایش می‌دهد.

- شکل ۲ یک نمونه سیگنال ارسالی (شکل بالا) و دریافتی (شکل پایین) را نمایش می‌دهد. مشاهده می‌شود که سیگنال ارسالی (پس از مدولاسیون آنالوگ) در محل تغییر بیت، دارای ناپیوستگی است. این ناپیوستگی اثر خود را در سیگنال دریافتی نیز نشان می‌دهد، اما دیگر یک جهش ناگهانی در سیگنال مشاهده نمی‌شود. علت این امر آن است که کانال مخابراتی یک فیلتر میان‌گذر است و اجازه‌ی عبور فرکانس‌های بالا را نمی‌دهد؛ در حالی که می‌دانیم شکستگی و ناپیوستگی تنها در حضور فرکانس‌های بسیار بالا میسّر می‌شود. همچنین اثر تأخیر کانال را نیز در این شکل می‌بینیم، چرا که اثر ناپیوستگی در سیگنال دریافتی، دیرتر از زمان آن در سیگنال ارسالی مشاهده شده است. (اگر از تابع `IdealChannel` که در قسمت ۴.۱ معرفی شد استفاده می‌کردیم، این تأخیر در سیگنال دریافتی مشاهده نمی‌شد.)

- شکل ۳ سیگنال دریافتی پس از demodulation آنالوگ را نشان می‌دهد. به دلیل عدم وجود نویز، مشاهده می‌شود که پالس‌های دریافتی منطبق بر پالس‌های ارسالی هستند و تنها اثر تأخیر کانال و فیلتر پایین‌گذر مورد استفاده در demodulation در آن مشاهده می‌شود.



شکل ۱: یک نمونه سیگنال ارسالی و دریافتی پس از مدولاسیون دیجیتال



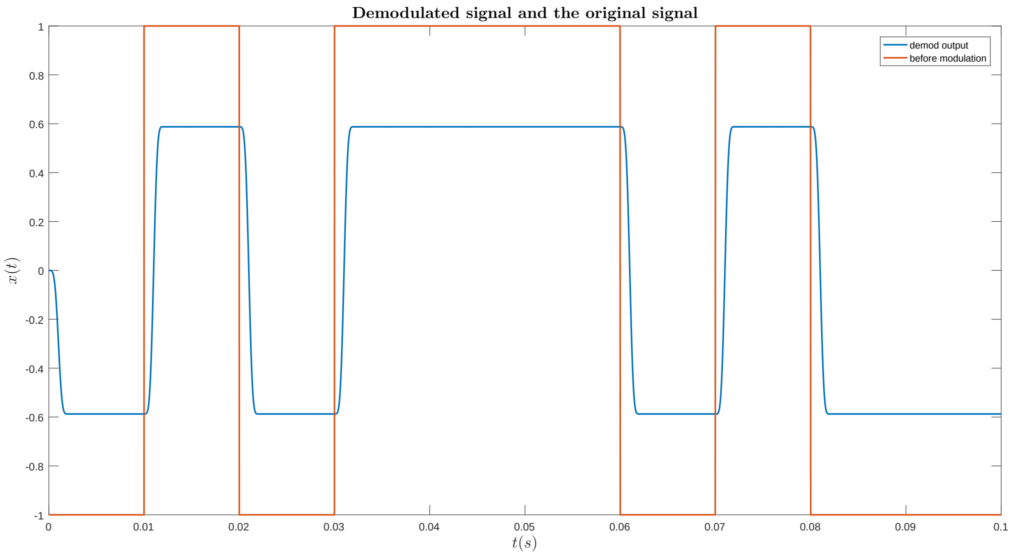
شکل ۲: سیگنال‌های ارسالی و دریافتی

۲.۱.۲ بررسی احتمال خطا در حضور نویز

- با افزودن نویز به سیستم و عبور نویز از یک مقدار حدّاقلی، دیگر خطای بازسازی صفر نخواهد بود. شکل‌های ۴ و ۵ احتمال خطا بر حسب واریانس نویز را در دو مقیاس عادی و لگاریتمی نشان می‌دهند.

- در رسم نمودارهای ۴ و ۵، ابتدا مشاهده می‌شود که رفتار نمودار به شدت نویزی است و حول مقدار نهایی خود نوسانات شدیدی دارد. علت این امر آن است که ما صرفاً به تحقّق‌هایی محدود از یک متغیر تصادفی نگاه می‌کنیم، و نمی‌توانیم انتظار داشته باشیم که رفتارهای تئوری آن را به تمامی در این تحقّق‌ها مشاهده کنیم. برای بهبود این مسأله، فرایند محاسبه‌ی احتمال خطا را چندین بار انجام دادیم و روی این تکرارها میانگین‌گیری کردیم تا از رفتار نویزی آن کمی کاسته شود و شکل تئوری مورد انتظار آن بیشتر آشکار شود.

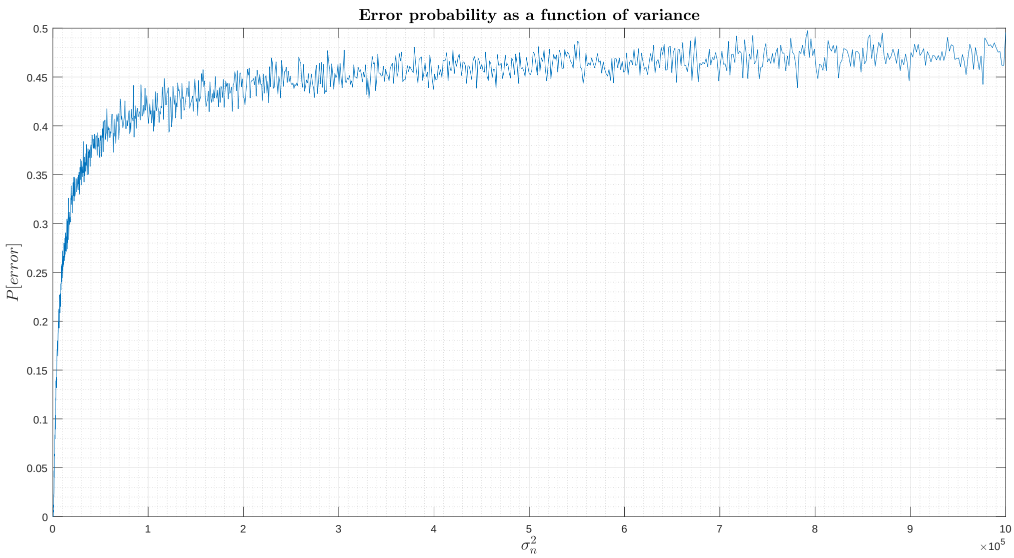
- **توجیه رفتار حدی احتمال خطا:** مشاهده می‌شود که با افزایش مقدار نویز، احتمال خطا به سمت 0.5 میل می‌کند. علت این امر آن است که زمانی که نویز بیش از اندازه زیاد می‌شود، عملاً سیگنال اولیه‌ی ارسالی در مقابل نویز چنان



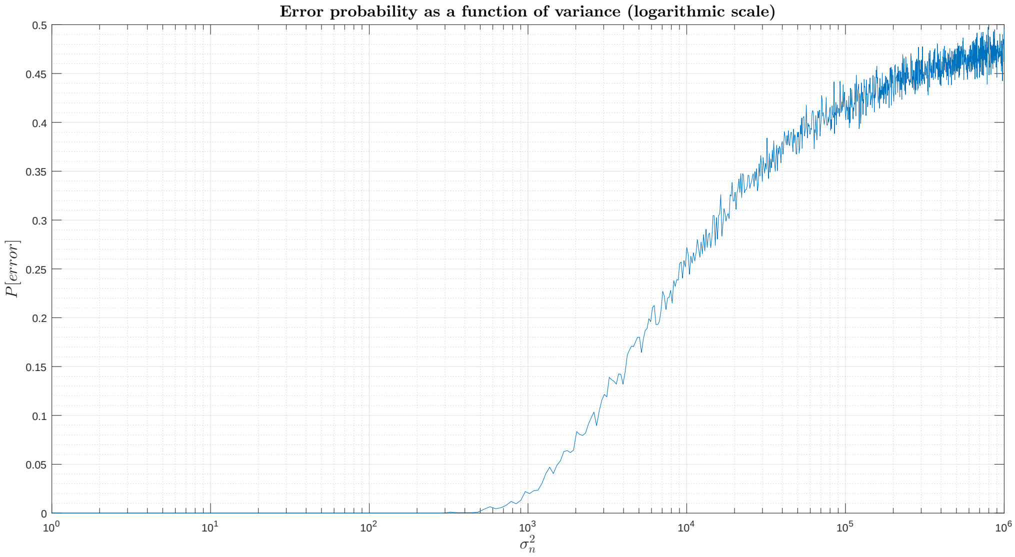
شکل ۳: سیگنال‌های ارسالی پس از demodulation آنالوگ

کوچک و قابل صرف‌نظر کردن است که می‌توان در نظر گرفت پیام دریافتی صرفاً نویز است. در این حالت مشخص است که هر بیت به صورت تصادفی و با احتمال $\frac{1}{2}$ ممکن است درست یا غلط باشد؛ بنابراین انتظار داریم که نمودار احتمال خطا نیز برای مقادیر بزرگ نویز، به سمت همان مقدار میل کند.

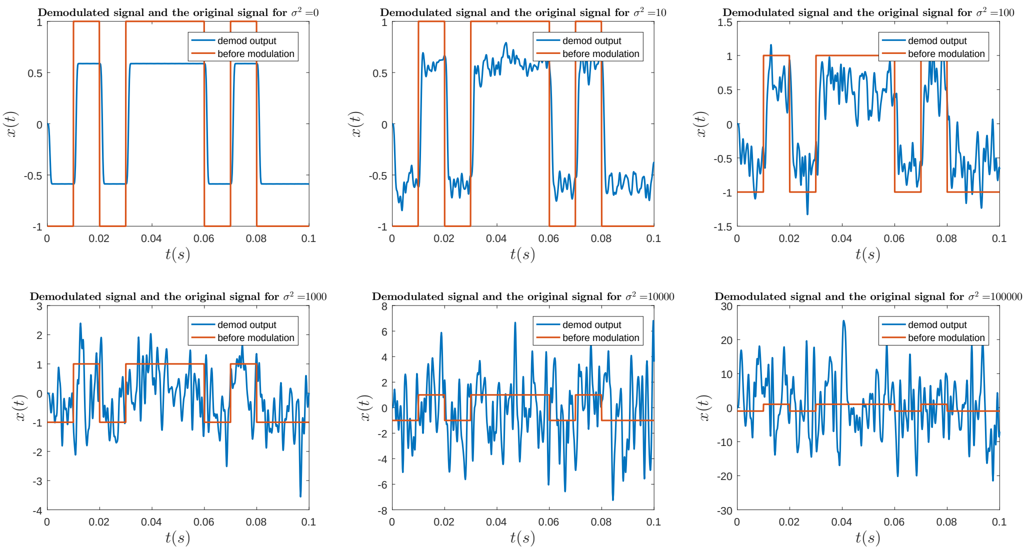
- برای دست‌یابی به شهودی بهتر از اثر نویز بر سیگنال دریافتی، نمودار سیگنال دریافتی پس از demodulation آنالوگ را در مقایسه با شکل مورد انتظار آن به ازای ۶ مقدار متفاوت از واریانس نویز در شکل ۶ مشاهده می‌کنید. مطابق این شکل، با بزرگ شدن نویز، عملاً نویز به سیگنال اصلی غالب می‌شود و دیگر سیگنال اصلی قابل بازیابی نیست.



شکل ۴: احتمال خطا بر حسب واریانس نویز در مقیاس عادی



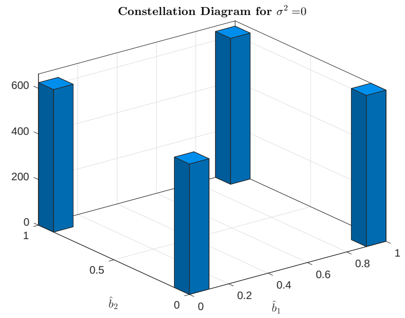
شکل ۵: احتمال خطا بر حسب واریانس نویز در مقیاس لگاریتمی



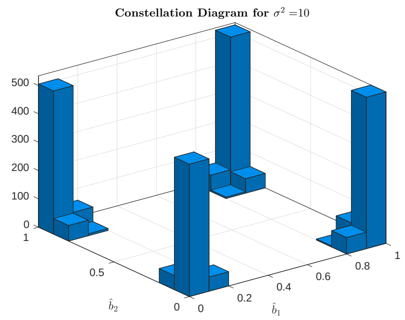
شکل ۶: اثر نویز بر سیگنال‌های بازیابی‌شده

۳.۱.۲ بررسی نمودارهای constellation

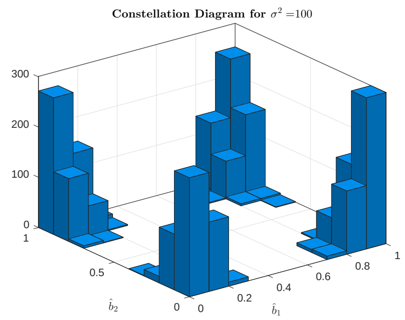
- constellation diagram ها به ازای ۶ مقدار مختلف از واریانس نویز در شکل‌های ۷ تا ۱۲ قابل مشاهده هستند. همان طور که انتظار داریم، ابتدا چهار ناحیه کاملاً مجزّا هستند، امّا به مرور و با افزایش نویز به هم نزدیک‌شده و نهایتاً در هم ادغام می‌شوند. ادغام شدن این نواحی به معنی عدم امکان تشخیص صحیح همه‌ی اطلاعات دریافتی، و به تبع آن ظهور خطا است.
- برای رسم constellation diagramها، ابتدا هیستوگرام خروجی Matched filter متناظر با شکل پالس بیت یک را رسم کردیم. نتیجه‌ی این کار، آن است که هیستوگرام حول چهار نقطه با مختصات $(\pm1, \pm1)$ به وجود می‌آید. در ادامه و با استفاده از نگاشت $y = \frac{1}{2}x + \frac{1}{2}$ ناحیه‌ی حاصل را به ناحیه‌ی مطلوب تصویر کردیم.
- در این بخش برای دستیابی به هیستوگرام‌های بهتر، تعداد بیت‌های انتقالی را برابر با ۵۰۰۰ بیت در نظر گرفتیم.



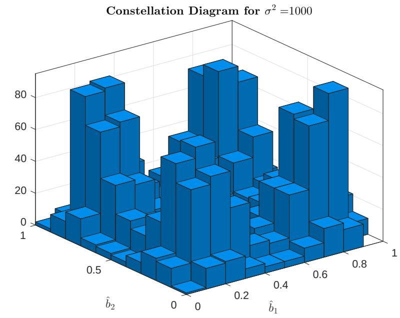
شکل ۷: نمودار constellation به ازای $\sigma_n^2 = 0$



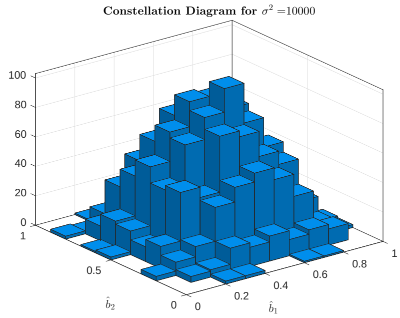
شکل ۸: نمودار constellation به ازای $\sigma_n^2 = 10$



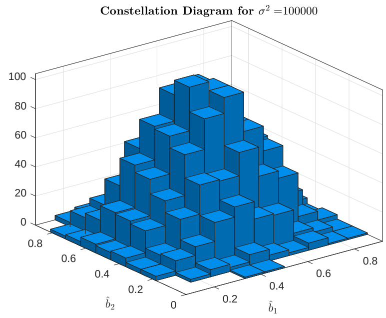
شکل ۹: نمودار constellation به ازای $\sigma_n^2 = 100$



شکل ۱۰: نمودار constellation به ازای $\sigma_n^2 = 1000$



شکل ۱۱: نمودار constellation به ازای $\sigma_n^2 = 10000$

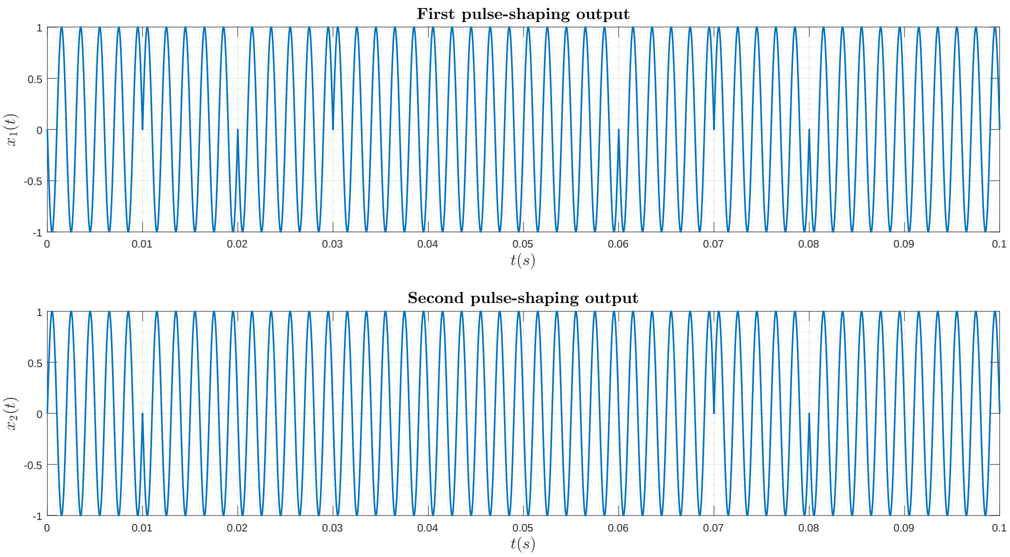


شکل ۱۲: نمودار constellation به ازای $\sigma_n^2 = 100000$

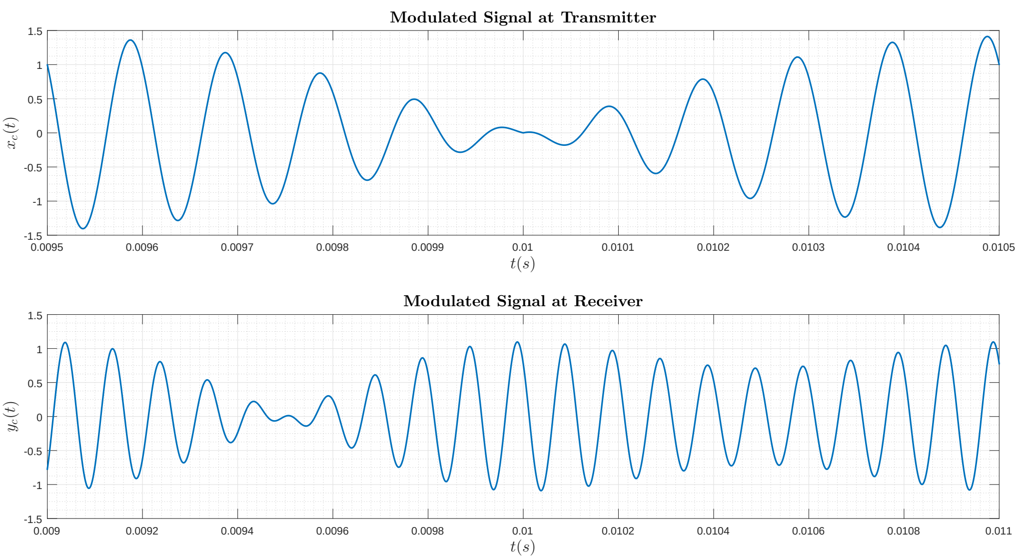
۲.۲ استفاده از شکل پالس سینوسی برای انتقال پیام

۱.۲.۲ شبیه‌سازی فرآیند ارسال پیام بدون نویز

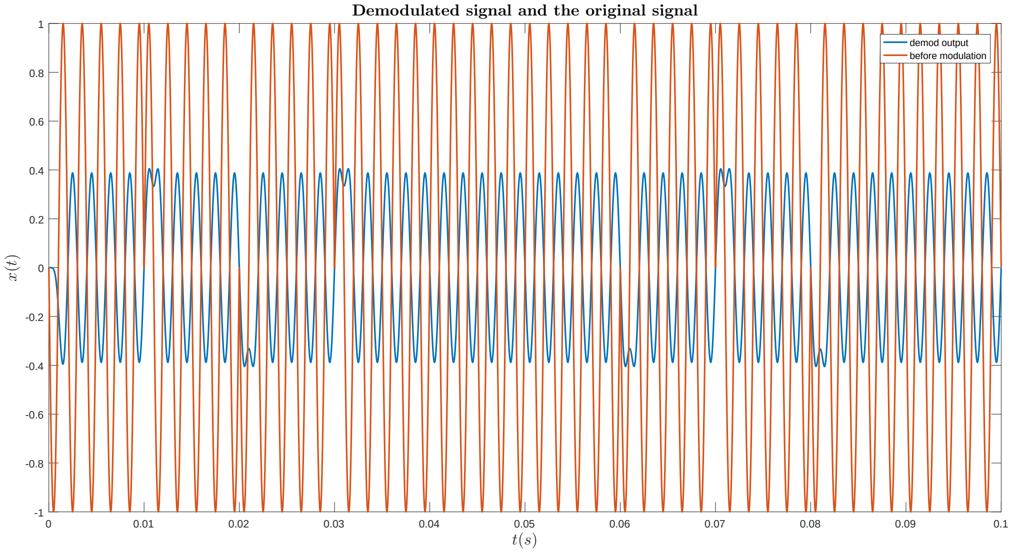
- توضیحات این بخش مشابه با قسمت ۱.۱.۲ است. تنها تفاوت مهم و عمده در بحث اثر تأخیر کانال و فیلتر مربوط به demodulation آنالوگ است. برای رفع مشکل تأخیر، شکل پالس مورد استفاده در Matched Filter را نیز از کانال عبور می‌دهیم. در نتیجه، همه‌ی تأخیرهای ممکن برای سیگنال به صورت مشابه برای شکل پالس متناظر با صفر و یک نیز رخ می‌دهند، لذا این دو نسبت به یک‌دیگر تأخیر نخواهند داشت و Matched Filter بهترین عملکرد ممکن خود را خواهد داشت.
- شکل‌های ۱۳ تا ۱۵ متناظر با شکل‌های بخش ۱.۱.۲ هستند و توضیحات مربوط به آن‌ها مشابه است.



شکل ۱۳: یک نمونه سیگنال ارسالی و دریافتی پس از مدولاسیون دیجیتال



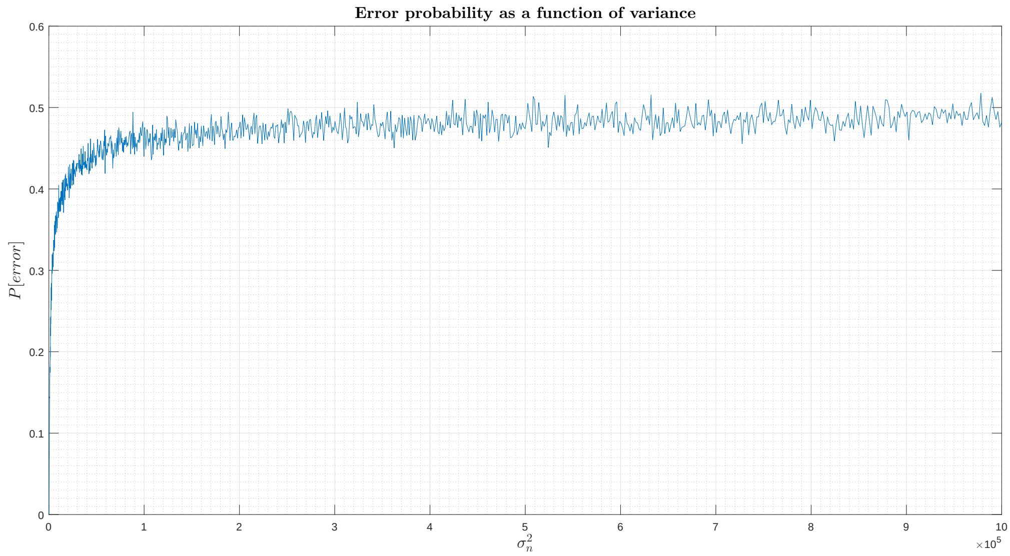
شکل ۱۴: سیگنال‌های ارسالی و دریافتی



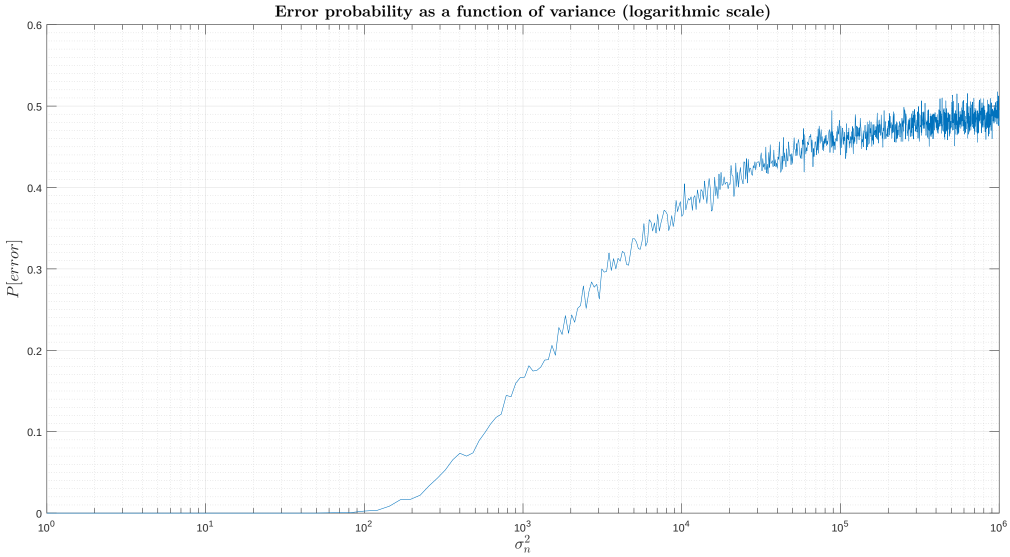
شکل ۱۵: سیگنال‌های ارسالی پس از demodulation آنالوگ

۲.۲.۲ بررسی احتمال خطا در حضور نویز

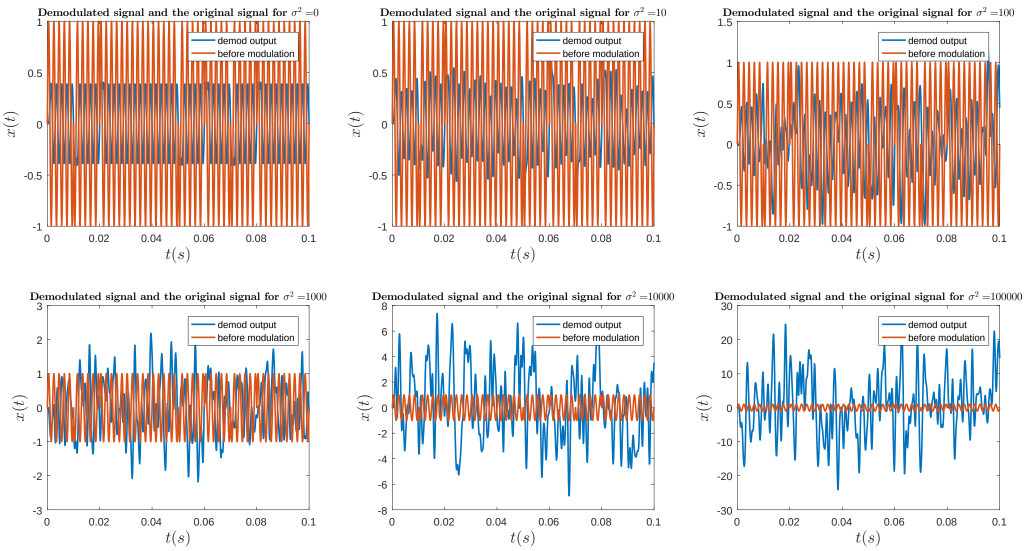
- توضیحات این بخش دقیقاً مشابه و متناظر با بخش ۲.۱.۲ می‌باشند. همچنین شکل‌های ۱۶ تا ۱۸ نیز دقیقاً متناظر با شکل‌های بخش ۲.۱.۲ می‌باشند.



شکل ۱۶: احتمال خطا بر حسب واریانس نویز در مقیاس عادی



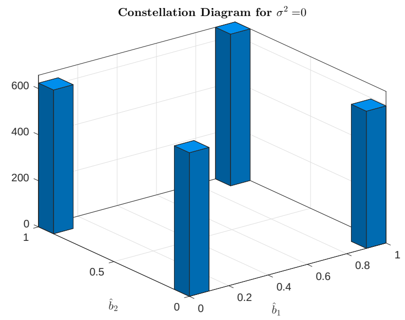
شکل ۱۷: احتمال خطا بر حسب واریانس نویز در مقیاس لگاریتمی



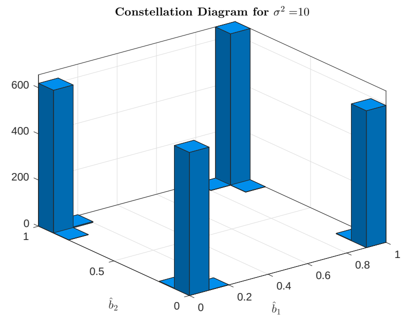
شکل ۱۸: اثر نویز بر سیگنال‌های بازیابی‌شده

۳.۲.۲ بررسی نمودارهای constellation

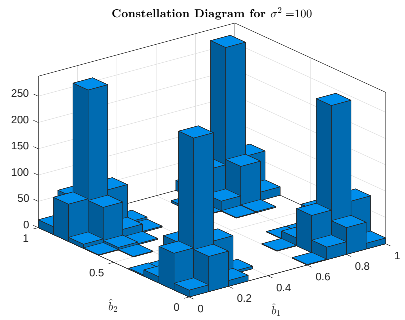
- توضیحات و شکل‌های این بخش (شکل‌های ۱۹ تا ۲۴) نیز دقیقاً متناظر با توضیحات و شکل‌های بخش ۳.۱.۲ می‌باشند.



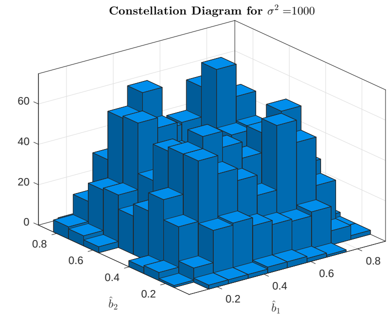
شکل ۱۹: نمودار constellation به ازای $\sigma_n^2 = 0$



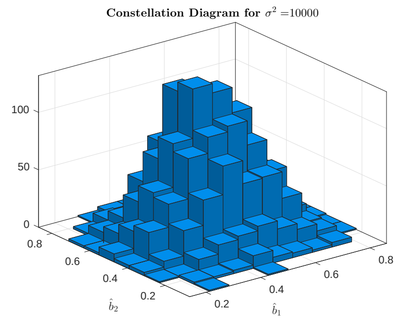
شکل ۲۰: نمودار constellation به ازای $\sigma_n^2 = 10$



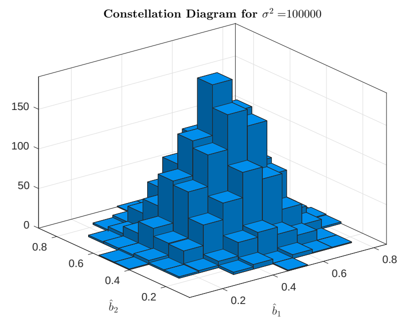
شکل ۲۱: نمودار constellation به ازای $\sigma_n^2 = 100$



شکل ۲۲: نمودار constellation به ازای $\sigma_n^2 = 1000$



شکل ۲۳: نمودار constellation به ازای $\sigma_n^2 = 10000$



شکل ۲۴: نمودار constellation به ازای $\sigma_n^2 = 100000$

۳.۲ مقایسه‌ی سیستم‌های بخش‌های ۱.۲ و ۲.۲

رفتار کلی دو سیستم از نظر عملکرد در حضور نویز و نیز نمودارهای constellation مشابه بود. مهم‌ترین تفاوتی که به نظر می‌رسد، اثر تأخیر در دو سیستم است، به گونه‌ای که در سیستم دوم برای رفع این مشکل مجبور شدیم نمونه‌ای از شکل موج‌های متناظر با بیت صفر و یک را نیز از کانال مخابراتی عبور دهیم و در مقصد از آن‌ها برای پیاده‌سازی Matched Filter استفاده کنیم، در حالی که در سیستم اول و با استفاده از پالس مربعی نیازی به چنین کاری احساس نشد. (البته این کار در حالت کلی درست‌ترین عمل است، چرا که عملکرد Matched Filter را در مقابل انواع اعواجاج‌های کانال تا حد امکان مقاوم می‌کند و همچنین می‌تواند مشکل سنکرونیزاسیون فرکانسی فرستنده و گیرنده را نیز حل کند.)

۳ انتقال دنباله‌ای از اعداد ۸ بیتی

۱.۳ توابع SourceGenerator و OutputDecoder

- تابع SourceGenerator:

این تابع در ورودی دنباله‌ای از اعداد در بازه صفر تا ۲۵۵ دریافت می‌کند و در خروجی رشته‌ای باینری متناظر با دنباله‌ی ورودی می‌دهد که هر یک از اعداد ورودی به صورت دودویی و با نمایش ۸بیتی در خروجی ظاهر شده است. کد این تابع نیز در ادامه قابل مشاهده است.

```
function out = SourceGenerator(x)
out = reshape(de2bi(x,8)',1,[]);
```

- تابع OutputDecoder:

این تابع دقیقاً عمل عکس تابع قبل را انجام می‌دهد. ورودی تابع رشته‌ای باینری است که هر ۸ بیت از آن، متناظر با یک عدد بین صفر تا ۲۵۵ است. تابع عدد ده‌دهی متناظر با هر ۸ بیت را شناسایی کرده و در خروجی، دنباله‌ی متناظر از اعداد صفر تا ۲۵۵ را به دست می‌دهد. کد تابع نیز در ادامه ضمیمه شده است.

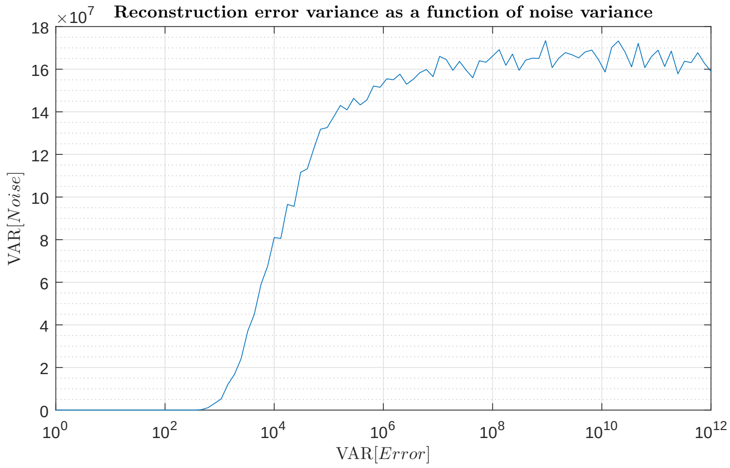
```
function out = OutputDecoder(x)
temp = zeros(length(x)/8,8);
for i = 1 : 8
    temp(:,i) = x(i:8:end);
end
out = bi2de(temp)';
```

۲.۳ مخابره‌ی دنباله‌ی اعداد صحیح

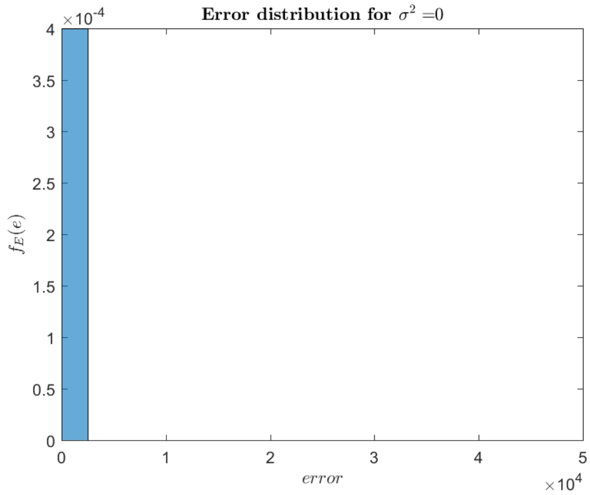
با کنار هم قرار دادن بلوک‌های ساخته‌شده در بخش ۱ و نیز توابع قسمت ۱.۳، سیستم انتقال داده را شبیه‌سازی می‌کنیم. طول دنباله‌ی اعداد صحیح را نیز در این بخش برابر با ۲۰۰ در نظر گرفته‌ایم. در این صورت، نمودار واریانس خطای بازسازی بر حسب واریانس نویز به صورت شکل ۲۵ درمی‌آید. برای آن که این نمودار را با نوسانات کم به دست آوریم، فرآیند محاسبه‌ی واریانس خطا را (به ازای یک وارانس مشخص برای نویز) ۳۰ بار تکرار کردیم و روی این مقادیر میانگین‌گیری نمودیم. اگر چنین نمی‌کردیم، نوسانات و نویز نمودار شکل ۲۵ بسیار بیشتر از چیزی که در حال حاضر قابل مشاهده است می‌شد.

۳.۳ توزیع خطای بازسازی

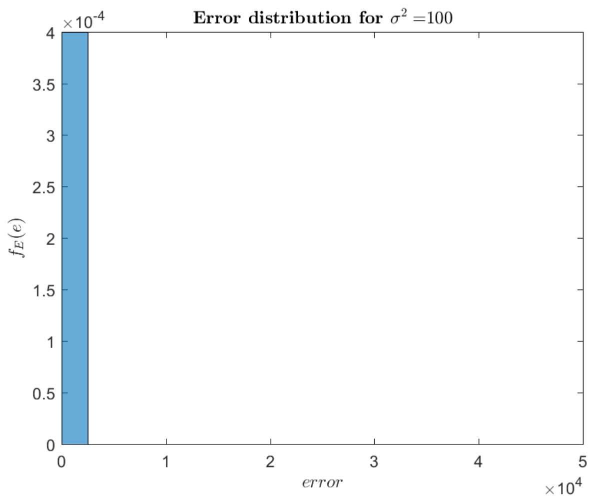
ابتدا هیستوگرام خطای بازسازی را به ازای ۷ مقدار متفاوت از واریانس نویز رسم می‌کنیم. برای آن که هیستوگرام‌ها به شکل واقعی توزیع‌های احتمالاتی نزدیک‌تر باشند، تعداد نمونه‌ها را افزایش داده و در این بخش از ۲۰۰۰ عدد صحیح جهت مخابره استفاده می‌کنیم. نتایج در شکل‌های ۲۶ تا ۳۲ قابل مشاهده است. دقت کنید که در این جا، تعریفی که از خطا داشته‌ایم، به تعبیری *خطای مربّع* است، به این معنی که مجذور اختلاف مقدار حاصل و مقدار صحیح را به عنوان خطا در نظر گرفته‌ایم.
برای بررسی رفتار حدّی خطا، خوب است ابتدا به خطای عادی (که صرفاً اختلاف مقدار حاصل و مقدار مورد انتظار است ــ بدون به توان ۲ رساندن ــ) توجه کنیم. ابتدا این خطا را در شکل‌های ۳۳ تا ۳۹ رسم کرده‌ایم.
مشاهده می‌شود که این خطا با بزرگ شدن واریانس نویز به یک توزیع مثلثی میل می‌کند. علّت این امر نسبتاً واضح است؛ با بزرگ شدن نویز، SNR سیگنال دریافتی به حدّی پایین می‌آید که می‌توان ادّعا کرد آن‌چه در گیرنده دریافت می‌شود نویز خالص است. در این حالت، خطا می‌تواند در بازه $[-255, 255]$ تغییر کند، امّا احتمال وقوع این مقادیر یکسان نیست. به



شکل ۲۵: واریانس خطای بازسازی بر حسب واریانس نویز

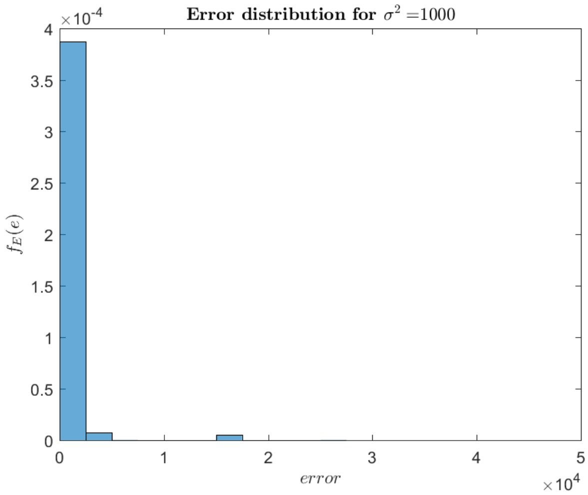


شکل ۲۶: توزیع خطای مربع به ازای $\sigma_n^2 = 0$

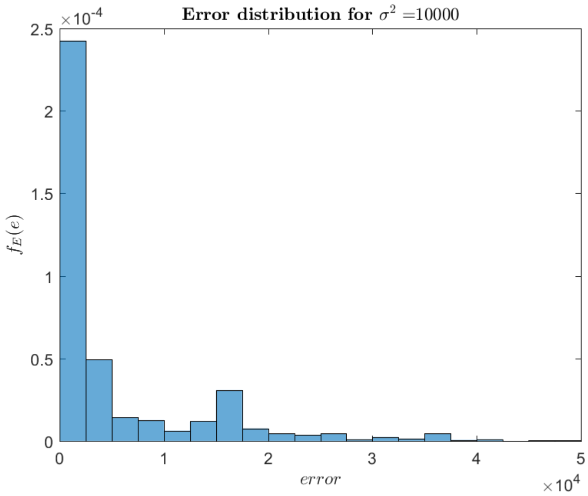


شکل ۲۷: توزیع خطای مربع به ازای $\sigma_n^2 = 100$

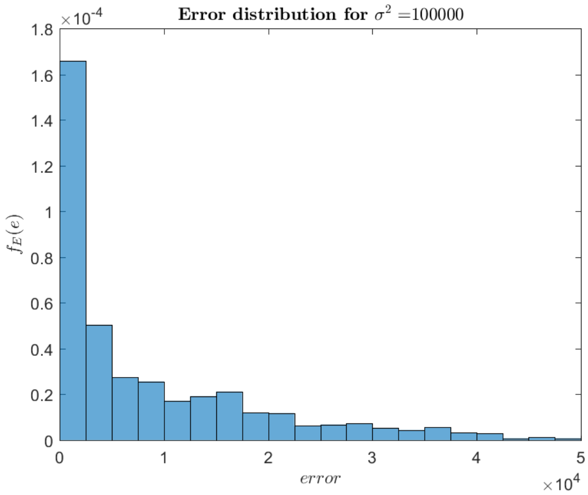
عنوان مثال خطای 0 در 256 حالت مختلف می‌تواند اتّفاق بیفتد امّا خطای 255 تنها در یک حالت می‌تواند رخ دهد (و آن حالتی است که عدد ارسالی 255 و عدد دریافتی 0 باشد.) لذا مشاهده می‌شود که احتمال خطا در خطای صفر بیشینه بوده و



شکل ۲۸: توزیع خطای مربع به ازای $\sigma_n^2 = 1000$

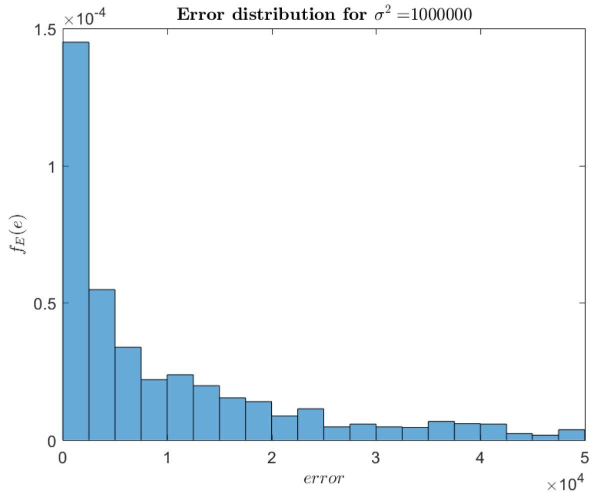


شکل ۲۹: توزیع خطای مربع به ازای $\sigma_n^2 = 10^4$

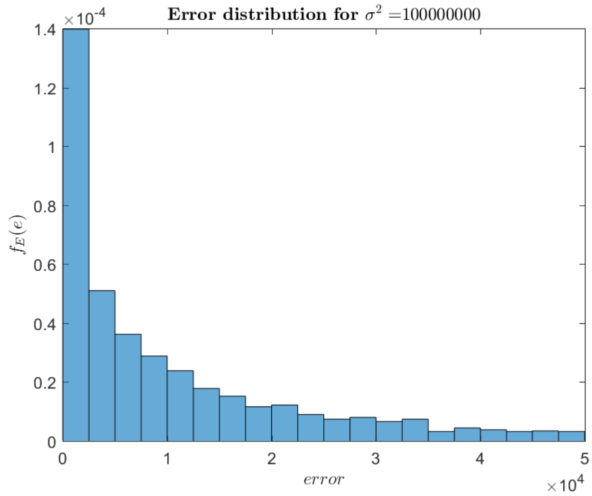


شکل ۳۰: توزیع خطای مربع به ازای $\sigma_n^2 = 10^5$

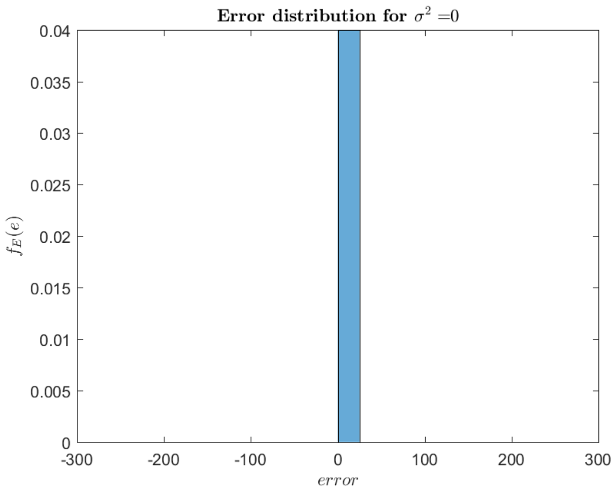
با افزایش مقدار خطا به صورت خطی کوچک و کوچک‌تر می‌شود و این توزیع، دقیقاً همان توزیع مثلثی مشاهده‌شده را به دست می‌دهد.



شکل ۳۱: توزیع خطای مربع به ازای $\sigma_n^2 = 10^6$

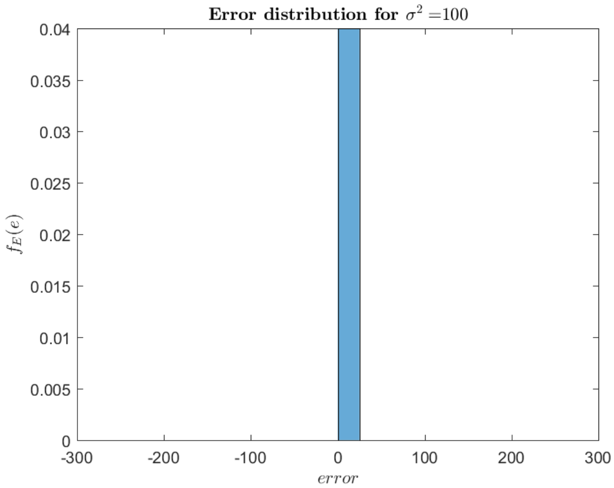


شکل ۳۲: توزیع خطای مربع به ازای $\sigma_n^2 = 10^8$

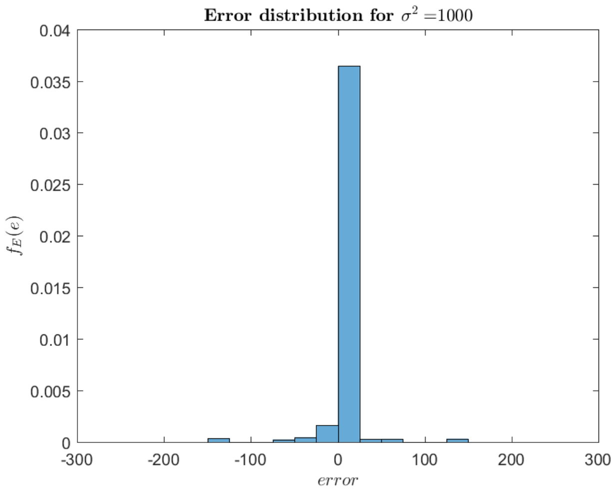


شکل ۳۳: توزیع خطا به ازای $\sigma_n^2 = 0$

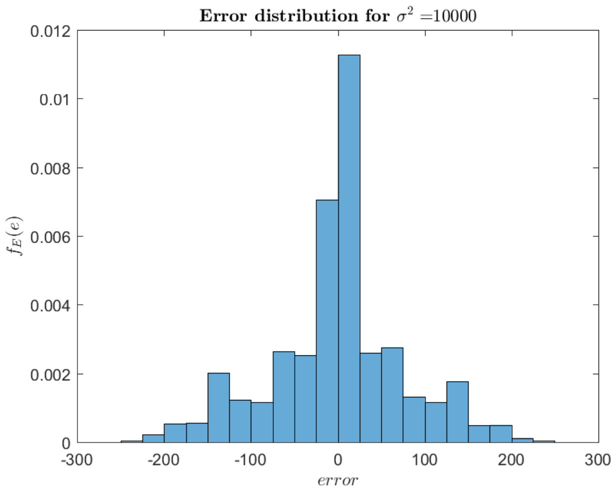
در نهایت اگر به سراغ خطای مربّع برویم، توزیع دیگر مثلثی نیست، چرا که مقادیر خطا دیگر اعداد صحیح در بازه $[-255, 255]$ نیستند، بلکه اعداد مجموعه $\{n^2 | 0 \leq n \leq 255\}$ هستند. در این حالت توزیع به جای شکل خطی،



شکل ۳۴: توزیع خطا به ازای $\sigma_n^2 = 100$

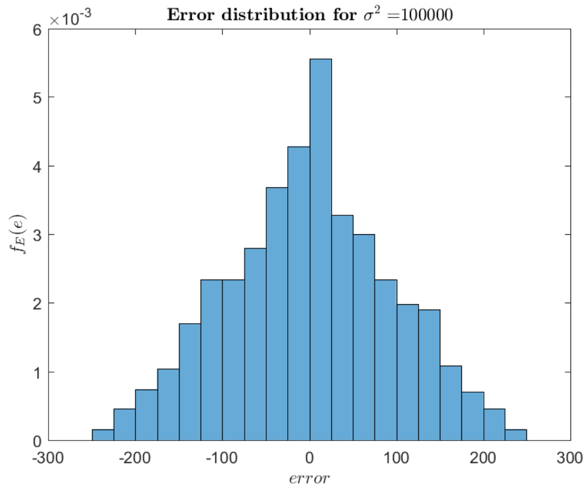


شکل ۳۵: توزیع خطا به ازای $\sigma_n^2 = 1000$

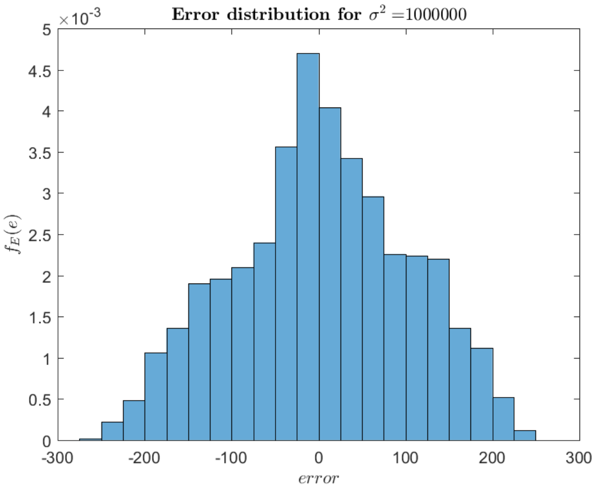


شکل ۳۶: توزیع خطا به ازای $\sigma_n^2 = 10^4$

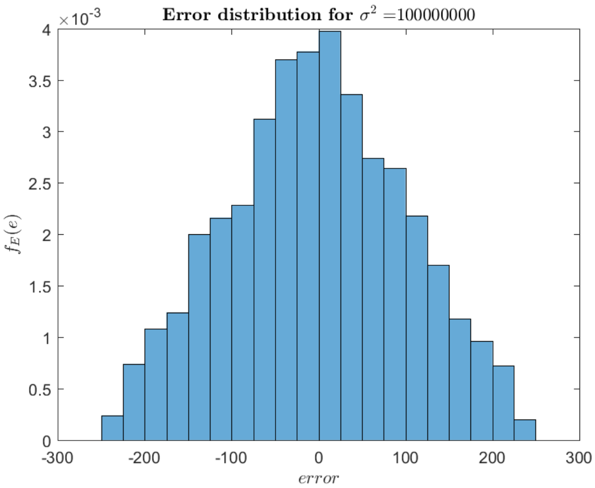
شکلی منحنی مشابه با نمودار ۳۲ به خود می‌گیرد.



شکل ۳۷: توزیع خطا به ازای $\sigma_n^2 = 10^5$



شکل ۳۸: توزیع خطا به ازای $\sigma_n^2 = 10^6$



شکل ۳۹: توزیع خطا به ازای $\sigma_n^2 = 10^8$

۴.۳ واریانس خطا در نویز بی‌نهایت

در حالتی که نویز به سمت بی‌نهایت برود، مطابق با توضیحات بخش قبل، می‌توان سیگنال دریافتی را مستقل از سیگنال ارسالی در نظر گرفت. در این حالت، واریانس خطا از معادله‌ی ۲ قابل محاسبه است.

$$\begin{aligned}\sigma_E^2 &= \mathbb{E}[E^2] - \mathbb{E}^2[E] \\ &= \sum_{i=0}^{255} x_i^2 p_i - \Big(\sum_{i=0}^{255} x_i p_i\Big)^2 \\ &= \sum_{i=0}^{255} i^4 p_i - \Big(\sum_{i=0}^{255} i^2 p_i\Big)^2\end{aligned} \tag{۲}$$

همچنین مقدار p_i در معادله‌ی ۲ نیز با توجّه به قسمت ۳.۳ از معادله‌ی ۳ محاسبه می‌شود.

$$p_i = \begin{cases} \dfrac{1}{256} & i = 0 \\ \\ \dfrac{2(256-i)}{256^2} & i \neq 0 \end{cases} \tag{۳}$$

با توجه به معادلات ۲ و ۳ خواهیم داشت:

$$\sigma_E^2 = 1.6702 \times 10^8$$

این مقدار با رفتار حدی نمودار شکل ۲۵ منطبق است و آن را توجیه می‌کند.

۴ ضمیمه: کد متلب شبیه‌سازی‌ها

```
%% 3.1 - Parameters
fs = 10^6;                % Sampling Frequency
PulseWidth = 10 * 10^-3;  % Pulse Width
N = fs * PulseWidth;      % Vector Length for each bit (corresponding
    to 10ms)
fc = 10^4;                % Carrier Frequency
f0 = 10^4;
BW = 10^3;
NumberOfBits = 20;
t = 0 : 1/fs : NumberOfBits/2*PulseWidth-1/fs;
OneWaveform = ones(1,N);
ZeroWaveform = -ones(1,N);
b = double(rand(1,NumberOfBits)>0.5)
%% 3.1 - a
[b1, b2] = Divide(b);
x1 = PulseShaping(b1, ZeroWaveform, OneWaveform);
x2 = PulseShaping(b2, ZeroWaveform, OneWaveform);
figure
subplot(2,1,1)
plot(t,x1,'LineWidth',1.5)
xlabel('$t(s)$','interpreter','latex','FontSize', 15)
ylabel('$x_1(t)$','interpreter','latex','FontSize', 15)
title('\textbf{First pulse-shaping output}','interpreter','latex','
    FontSize', 15)
grid on
grid minor
subplot(2,1,2)
plot(t,x2,'LineWidth',1.5)
xlabel('$t(s)$','interpreter','latex','FontSize', 15)
ylabel('$x_2(t)$','interpreter','latex','FontSize', 15)
title('\textbf{Second pulse-shaping output}','interpreter','latex',
    'FontSize', 15)
grid on
grid minor

Xc = AnalogMod(x1, x2, fs, fc);
figure
subplot(2,1,1)
plot(t, Xc,'LineWidth',1.5)
xlim([0.01-0.0005, 0.01+0.0005])
```

```
xlabel('$t(s)$','interpreter','latex','FontSize', 15)
ylabel('$x_c(t)$','interpreter','latex','FontSize', 15)
title('\textbf{Modulated Signal at Transmitter}','interpreter','latex','FontSize', 15)
grid on
grid minor

Yc = Channel(Xc, fs, f0, BW);
subplot(2,1,2)
plot(t, Yc,'LineWidth',1.5)
xlim([0.01-0.001, 0.01+0.001])
xlabel('$t(s)$','interpreter','latex','FontSize', 15)
ylabel('$y_c(t)$','interpreter','latex','FontSize', 15)
title('\textbf{Modulated Signal at Receiver}','interpreter','latex','FontSize', 15)
grid on
grid minor

[b1_demod, b2_demod] = AnalogDemod(Yc, fs, BW, fc);
figure
plot(t, b1_demod,'LineWidth',1.5);
hold on
plot(t, x1,'LineWidth',1.5);
xlabel('$t(s)$','interpreter','latex','FontSize', 15)
ylabel('$x(t)$','interpreter','latex','FontSize', 15)
title('\textbf{Demodulated signal and the original signal}','interpreter','latex','FontSize', 15)
legend('demod output', 'before modulation')
[~, ~, b1_est] = MatchedFilt(b1_demod, ZeroWaveform, OneWaveform);
[~, ~, b2_est] = MatchedFilt(b2_demod, ZeroWaveform, OneWaveform);

y = Combine(b1_est, b2_est)
sum((y-b).^2)

%% 3.1 - b
MaxSigma = 1000;
NumOfTrials = 100;
Error = zeros(NumOfTrials, MaxSigma);
for sigma = 1 : MaxSigma
    for trial = 1 : NumOfTrials
        [b1, b2] = Divide(b);
        x1 = PulseShaping(b1, ZeroWaveform, OneWaveform);
        x2 = PulseShaping(b2, ZeroWaveform, OneWaveform);
```

```
            Xc = AnalogMod(x1, x2, fs, fc);
            Xcn = Xc + sigma * randn(1,length(Xc));
            Yc = Channel(Xcn, fs, f0, BW);
            [b1_demod, b2_demod] = AnalogDemod(Yc, fs, BW, fc);
            [~, ~, b1_est] = MatchedFilt(b1_demod, ZeroWaveform,
                OneWaveform);
            [~, ~, b2_est] = MatchedFilt(b2_demod, ZeroWaveform,
                OneWaveform);
            y = Combine(b1_est, b2_est);
            Error(trial, sigma) = sum((y-b).^2)/length(b);
        end
end
error = mean(Error, 1);
close all
%%
figure
plot((1:MaxSigma).^2,error)
xlabel('$\sigma^2_n$','interpreter','latex','FontSize', 15)
ylabel('${P}[error]$','interpreter','latex','FontSize', 15)
title('\textbf{Error probability as a function of variance}','
    interpreter','latex','FontSize', 15)
grid on
grid minor

figure
semilogx((1:MaxSigma).^2,error)
xlabel('$\sigma^2_n$','interpreter','latex','FontSize', 15)
ylabel('${P}[error]$','interpreter','latex','FontSize', 15)
title('\textbf{Error probability as a function of variance (
    logarithmic scale)}','interpreter','latex','FontSize', 15)
grid on
grid minor
%%
Sigma = [0 10^(0.5) 10 10^(1.5) 100 10^2.5];
NumOfTrials = 1;
figure
for sigma = Sigma
    for trial = 1 : NumOfTrials
        [b1, b2] = Divide(b);
        x1 = PulseShaping(b1, ZeroWaveform, OneWaveform);
        x2 = PulseShaping(b2, ZeroWaveform, OneWaveform);

        Xc = AnalogMod(x1, x2, fs, fc);
```

```
        Xcn = Xc + sigma * randn(1,length(Xc));

        Yc = Channel(Xcn, fs, f0, BW);

        [b1_demod, b2_demod] = AnalogDemod(Yc, fs, BW, fc);
        subplot(2,3,find(sigma == Sigma))
        plot(t, b1_demod,'LineWidth',1.5);
        hold on
        plot(t, x1,'LineWidth',1.5);
        xlabel('$t(s)$','interpreter','latex','FontSize', 15)
        ylabel('$x(t)$','interpreter','latex','FontSize', 15)
        title(['\textbf{Demodulated signal and the original signal for $\sigma^2=$}' num2str(sigma^2)],'interpreter','latex','FontSize', 10)
        legend('demod output', 'before modulation')
        [~, ~, b1_est] = MatchedFilt(b1_demod, ZeroWaveform, OneWaveform);
        [~, ~, b2_est] = MatchedFilt(b2_demod, ZeroWaveform, OneWaveform);

        y = Combine(b1_est, b2_est);
    end
end
%%
Sigma = [0 10^(0.5) 10 10^(1.5) 100 10^2.5];
b = double(rand(1,5000)>0.5);
for sigma = Sigma
    [b1, b2] = Divide(b);
    x1 = PulseShaping(b1, ZeroWaveform, OneWaveform);
    x2 = PulseShaping(b2, ZeroWaveform, OneWaveform);

    Xc = AnalogMod(x1, x2, fs, fc);

    Xcn = Xc + sigma * randn(1,length(Xc));

    Yc = Channel(Xcn, fs, f0, BW);

    [b1_demod, b2_demod] = AnalogDemod(Yc, fs, BW, fc);

    [b01, b11, b1_est] = MatchedFilt(b1_demod, ZeroWaveform, OneWaveform);
```

```matlab
    [b02, b12, b2_est] = MatchedFilt(b2_demod, ZeroWaveform, OneWaveform);

    y = Combine(b1_est, b2_est);

    b1_hist = b11/2 + 0.5;
    b2_hist = b12/2 + 0.5;
    figure
    histogram2 (b1_hist, b2_hist,10)
    xlabel('$\hat{b}_1$','interpreter','latex')
    ylabel('$\hat{b}_2$','interpreter','latex')
    title(['\textbf{Constellation Diagram for $\sigma^2=$}',num2str(sigma^2)],'interpreter','latex')
end
%% 3.2 - Parameters
fs = 10^6;                 % Sampling Frequency
PulseWidth = 10 * 10^-3;   % Pulse Width
N = fs * PulseWidth;       % Vector Length for each bit (corresponding to 10ms)
fc = 10^4;                 % Carrier Frequency
f0 = 10^4;
BW = 10^3;
NumberOfBits = 20;
t = 0 : 1/fs : NumberOfBits/2*PulseWidth-1/fs;
OneWaveform = sin(2*pi*500*(0:1/fs:PulseWidth-1/fs));
ZeroWaveform = -sin(2*pi*500*(0:1/fs:PulseWidth-1/fs));
%% 3.2 - a
b = double(rand(1,NumberOfBits)>0.5)
[b1, b2] = Divide(b);
x1 = PulseShaping(b1, ZeroWaveform, OneWaveform);
x2 = PulseShaping(b2, ZeroWaveform, OneWaveform);
figure
subplot(2,1,1)
plot(t,x1,'LineWidth',1.5)
xlabel('$t(s)$','interpreter','latex','FontSize', 15)
ylabel('$x_1(t)$','interpreter','latex','FontSize', 15)
title('\textbf{First pulse-shaping output}','interpreter','latex','FontSize', 15)
grid on
grid minor
subplot(2,1,2)
plot(t,x2,'LineWidth',1.5)
xlabel('$t(s)$','interpreter','latex','FontSize', 15)
```

```
ylabel('$x_2(t)$','interpreter','latex','FontSize', 15)
title('\textbf{Second pulse-shaping output}','interpreter','latex','FontSize', 15)
grid on
grid minor

Xc = AnalogMod(x1, x2, fs, fc);
figure
subplot(2,1,1)
plot(t, Xc,'LineWidth',1.5)
xlim([0.01-0.0005, 0.01+0.0005])
xlabel('$t(s)$','interpreter','latex','FontSize', 15)
ylabel('$x_c(t)$','interpreter','latex','FontSize', 15)
title('\textbf{Modulated Signal at Transmitter}','interpreter','latex','FontSize', 15)
grid on
grid minor

Yc = Channel(Xc, fs, f0, BW);
subplot(2,1,2)
plot(t, Yc,'LineWidth',1.5)
xlim([0.01-0.001, 0.01+0.001])
xlabel('$t(s)$','interpreter','latex','FontSize', 15)
ylabel('$y_c(t)$','interpreter','latex','FontSize', 15)
title('\textbf{Modulated Signal at Receiver}','interpreter','latex','FontSize', 15)
grid on
grid minor

[b1_demod, b2_demod] = AnalogDemod(Yc, fs, BW, fc);
figure
plot(t, b1_demod,'LineWidth',1.5);
hold on
plot(t, x1,'LineWidth',1.5);
xlabel('$t(s)$','interpreter','latex','FontSize', 15)
ylabel('$x(t)$','interpreter','latex','FontSize', 15)
title('\textbf{Demodulated signal and the original signal}','interpreter','latex','FontSize', 15)
legend('demod output', 'before modulation')

temp1 = AnalogMod(ZeroWaveform, OneWaveform, fs, fc);
temp2 = Channel(temp1,fs,f0,BW);
```

```
[Final_ZeroWaveform ,Final_OneWaveform] = AnalogDemod(temp2,fs,BW,
    fc);

[~, ~, b1_est] = MatchedFilt(b1_demod, Final_ZeroWaveform,
    Final_OneWaveform);
[~, ~, b2_est] = MatchedFilt(b2_demod, Final_ZeroWaveform,
    Final_OneWaveform);


y = Combine(b1_est, b2_est)
sum((y-b).^2)
%% 3.2 - b
MaxSigma = 1000;
NumOfTrials = 100;
Error = zeros(NumOfTrials, MaxSigma);
for sigma = 1 : MaxSigma
    for trial = 1 : NumOfTrials
        [b1, b2] = Divide(b);
        x1 = PulseShaping(b1, ZeroWaveform, OneWaveform);
        x2 = PulseShaping(b2, ZeroWaveform, OneWaveform);
        Xc = AnalogMod(x1, x2, fs, fc);
        Xcn = Xc + sigma * randn(1,length(Xc));
        Yc = Channel(Xcn, fs, f0, BW);
        [b1_demod, b2_demod] = AnalogDemod(Yc, fs, BW, fc);
        temp1 = AnalogMod(ZeroWaveform, OneWaveform, fs, fc);
        temp2 = Channel(temp1,fs,f0,BW);
        [Final_ZeroWaveform ,Final_OneWaveform] = AnalogDemod(temp2
            ,fs,BW,fc);

        [~, ~, b1_est] = MatchedFilt(b1_demod, Final_ZeroWaveform,
            Final_OneWaveform);
        [~, ~, b2_est] = MatchedFilt(b2_demod, Final_ZeroWaveform,
            Final_OneWaveform);

        y = Combine(b1_est, b2_est);
        Error(trial, sigma) = sum((y-b).^2)/length(b);
    end
end
error = mean(Error, 1);
close all
%%
figure
plot((1:MaxSigma).^2,error)
```

```
xlabel('$\sigma^2_n$','interpreter','latex','FontSize', 15)
ylabel('${P}[error]$','interpreter','latex','FontSize', 15)
title('\textbf{Error probability as a function of variance}','
    interpreter','latex','FontSize', 15)
grid on
grid minor

figure
semilogx((1:MaxSigma).^2,error)
xlabel('$\sigma^2_n$','interpreter','latex','FontSize', 15)
ylabel('${P}[error]$','interpreter','latex','FontSize', 15)
title('\textbf{Error probability as a function of variance (
    logarithmic scale)}','interpreter','latex','FontSize', 15)
grid on
grid minor
%%
Sigma = [0 10^(0.5) 10 10^(1.5) 100 10^2.5];
NumOfTrials = 1;
figure
for sigma = Sigma
    for trial = 1 : NumOfTrials
        [b1, b2] = Divide(b);
        x1 = PulseShaping(b1, ZeroWaveform, OneWaveform);
        x2 = PulseShaping(b2, ZeroWaveform, OneWaveform);

        Xc = AnalogMod(x1, x2, fs, fc);

        Xcn = Xc + sigma * randn(1,length(Xc));

        Yc = Channel(Xcn, fs, f0, BW);

        [b1_demod, b2_demod] = AnalogDemod(Yc, fs, BW, fc);
        subplot(2,3,find(sigma == Sigma))
        plot(t, b1_demod,'LineWidth',1.5);
        hold on
        plot(t, x1,'LineWidth',1.5);
        xlabel('$t(s)$','interpreter','latex','FontSize', 15)
        ylabel('$x(t)$','interpreter','latex','FontSize', 15)
        title(['\textbf{Demodulated signal and the original signal
            for $\sigma^2=$}' num2str(sigma^2)],'interpreter','latex
            ','FontSize', 10)
        legend('demod output', 'before modulation')
        temp1 = AnalogMod(ZeroWaveform, OneWaveform, fs, fc);
```

```
        temp2 = Channel(temp1,fs,f0,BW);
        [Final_ZeroWaveform ,Final_OneWaveform] = AnalogDemod(temp2
           ,fs,BW,fc);

        [~, ~, b1_est] = MatchedFilt(b1_demod, Final_ZeroWaveform,
           Final_OneWaveform);
        [~, ~, b2_est] = MatchedFilt(b2_demod, Final_ZeroWaveform,
           Final_OneWaveform);


        y = Combine(b1_est, b2_est);
    end
end
%%
Sigma = [0 10^(0.5) 10 10^(1.5) 100 10^2.5];

b = double(rand(1,5000)>0.5);
for sigma = Sigma
    [b1, b2] = Divide(b);
    x1 = PulseShaping(b1, ZeroWaveform, OneWaveform);
    x2 = PulseShaping(b2, ZeroWaveform, OneWaveform);

    Xc = AnalogMod(x1, x2, fs, fc);

    Xcn = Xc + sigma * randn(1,length(Xc));

    Yc = Channel(Xcn, fs, f0, BW);

    [b1_demod, b2_demod] = AnalogDemod(Yc, fs, BW, fc);

    temp1 = AnalogMod(ZeroWaveform, OneWaveform, fs, fc);
    temp2 = Channel(temp1,fs,f0,BW);
    [Final_ZeroWaveform ,Final_OneWaveform] = AnalogDemod(temp2,fs,
       BW,fc);

    [b01, b11, b1_est] = MatchedFilt(b1_demod, Final_ZeroWaveform,
       Final_OneWaveform);
    [b02, b12, b2_est] = MatchedFilt(b2_demod, Final_ZeroWaveform,
       Final_OneWaveform);


    y = Combine(b1_est, b2_est);

```

```
    b1_hist = b11/2 + 0.5;
    b2_hist = b12/2 + 0.5;
    figure
    histogram2 (b1_hist, b2_hist,10)
    xlabel('$\hat{b}_1$','interpreter','latex')
    ylabel('$\hat{b}_2$','interpreter','latex')
    title(['\textbf{Constellation Diagram for $\sigma^2=$}',num2str
       (sigma^2)],'interpreter','latex')
end
%% 4 - Parameters
NumOfInts = 200;
b = randi(255,1,NumOfInts)
b_bin = SourceGenerator(b);

fs = 10^6;                % Sampling Frequency
PulseWidth = 10 * 10^-3;  % Pulse Width
N = fs * PulseWidth;      % Vector Length for each bit (corresponding
    to 10ms)
fc = 10^4;                % Carrier Frequency
f0 = 10^4;
BW = 10^3;
NumberOfBits = NumOfInts*8;
t = 0 : 1/fs : NumberOfBits/2*PulseWidth-1/fs;
OneWaveform = ones(1,N);
ZeroWaveform = -ones(1,N);
%% 4.2
Sigma = logspace(0,6,100 );
NumOfTrials = 5;
errorVar = zeros(NumOfTrials, length(Sigma));
errorVar2 = zeros(NumOfTrials, length(Sigma));
for i = 1 : length(Sigma)
    for trial = 1 : NumOfTrials
        b = randi(255,1,NumOfInts)
        b_bin = SourceGenerator(b);
        sigma = Sigma(i)
        [b1, b2] = Divide(b_bin);
        x1 = PulseShaping(b1, ZeroWaveform, OneWaveform);
        x2 = PulseShaping(b2, ZeroWaveform, OneWaveform);

        Xc = AnalogMod(x1, x2, fs, fc);

        Xcn = Xc + sigma*randn(1,length(Xc));

```

```
        Yc = Channel(Xcn, fs, f0, BW);

        [b1_demod, b2_demod] = AnalogDemod(Yc, fs, BW, fc);

        temp1 = AnalogMod(ZeroWaveform, OneWaveform, fs, fc);
        temp2 = Channel(temp1,fs,f0,BW);
        [Final_ZeroWaveform ,Final_OneWaveform] = AnalogDemod(temp2
           ,fs,BW,fc);

        [~, ~, b1_est] = MatchedFilt(b1_demod, Final_ZeroWaveform,
           Final_OneWaveform);
        [~, ~, b2_est] = MatchedFilt(b2_demod, Final_ZeroWaveform,
           Final_OneWaveform);

        y_bin = Combine(b1_est, b2_est);
        y = OutputDecoder(y_bin);
        errorVar(trial,i) = sum((y-b).^2);
        errorVar2(trial,i) = var((y-b).^2);
    end
end
%%
figure
semilogx(Sigma.^2,mean(errorVar2,1))
xlabel('$\mathrm{VAR}[Error]$','interpreter','latex')
ylabel('$\mathrm{VAR}[Noise]$','interpreter','latex')
title('\textbf{Reconstruction error variance as a function of noise
    variance}','interpreter','latex')
grid on
grid minor
%% 4.3 - squared error distribution
Sigma = [0 10 10^(1.5) 100 10^2.5 1000 10000];
b = randi(255,1,2000);
b_bin = SourceGenerator(b);
for i = 1 : 7
    sigma = Sigma(i)
    [b1, b2] = Divide(b_bin);
    x1 = PulseShaping(b1, ZeroWaveform, OneWaveform);
    x2 = PulseShaping(b2, ZeroWaveform, OneWaveform);

    Xc = AnalogMod(x1, x2, fs, fc);

    Xcn = Xc + sigma*randn(1,length(Xc));

```

```
    Yc = Channel(Xcn, fs, f0, BW);

    [b1_demod, b2_demod] = AnalogDemod(Yc, fs, BW, fc);

    temp1 = AnalogMod(ZeroWaveform, OneWaveform, fs, fc);
    temp2 = Channel(temp1,fs,f0,BW);
    [Final_ZeroWaveform ,Final_OneWaveform] = AnalogDemod(temp2,fs,
       BW,fc);

    [~, ~, b1_est] = MatchedFilt(b1_demod, Final_ZeroWaveform,
       Final_OneWaveform);
    [~, ~, b2_est] = MatchedFilt(b2_demod, Final_ZeroWaveform,
       Final_OneWaveform);

    y_bin = Combine(b1_est, b2_est);
    y = OutputDecoder(y_bin);
    Error = (y-b).^2;
    figure
    histogram(Error,'normalization','pdf','binWidth',2500)
    xlim([0 50000])
    xlabel('$error$','interpreter','latex')
    ylabel('$f_E(e)$','interpreter','latex')
    title(['\textbf{Error distribution for $\sigma^2=$}' num2str(
       sigma^2)],'interpreter','latex')
end
%% 4.3 - simple error distribution
Sigma = [0 10 10^(1.5) 100 10^2.5 1000 10000];
b = randi(255,1,2000);
b_bin = SourceGenerator(b);
for i = 1 : 7
    sigma = Sigma(i)
    [b1, b2] = Divide(b_bin);
    x1 = PulseShaping(b1, ZeroWaveform, OneWaveform);
    x2 = PulseShaping(b2, ZeroWaveform, OneWaveform);

    Xc = AnalogMod(x1, x2, fs, fc);

    Xcn = Xc + sigma*randn(1,length(Xc));

    Yc = Channel(Xcn, fs, f0, BW);

    [b1_demod, b2_demod] = AnalogDemod(Yc, fs, BW, fc);

```

```
    temp1 = AnalogMod(ZeroWaveform, OneWaveform, fs, fc);
    temp2 = Channel(temp1,fs,f0,BW);
    [Final_ZeroWaveform ,Final_OneWaveform] = AnalogDemod(temp2,fs,
       BW,fc);

    [~, ~, b1_est] = MatchedFilt(b1_demod, Final_ZeroWaveform,
       Final_OneWaveform);
    [~, ~, b2_est] = MatchedFilt(b2_demod, Final_ZeroWaveform,
       Final_OneWaveform);

    y_bin = Combine(b1_est, b2_est);
    y = OutputDecoder(y_bin);
    Error = (y-b);
    figure
    histogram(Error,'normalization','pdf','binWidth',25)
    xlim([-300 300])
    xlabel('$error$','interpreter','latex')
    ylabel('$f_E(e)$','interpreter','latex')
    title(['\textbf{Error distribution for $\sigma^2=$}' num2str(
       sigma^2)],'interpreter','latex')
end
```