بسم الله الرحمن الرحیم

گزارش تمرین سوم عملی

«کدینگ کانال و آنالیز عملکرد»

دانشجو: سیدسعید جزائری شوشتری

شماره دانشجویی: ۹۸۱۰۴۸۸۵

مخابرات دیجیتال

استاد درس: دکتر کرباسی

دانشکده مهندسی برق

زمستان ۱۴۰۲

۱. تولید سیگنال ارسال

در این بخش مطابق خواسته انجام شده، ۲۰ بردار اطلاعات هرکدام به طول ۵۰۰۰۰ بیت تولید کرده و عملیات مدولاسیون خواسته شده را روی آن اعمال کردیم. یعنی به ازای هر دو بیت متوالی در هریک از این بردار های اطلاعات، مپینگ زیر را انجام دادیم:

$(0, 0) \rightarrow +I+Q \qquad (0, 1) \rightarrow +I-Q \qquad (1, 0) \rightarrow -I+Q \qquad (1, 1) \rightarrow +I+Q$

نکته: بنظر میرسد که ترتیب خواسته ها صحیح نیست، یعنی ابتدا باید روی رشته داده ها عملیات کدینگ کانال انجام شود و سپس مدولاسیون QPSK آن ها را تبدیل به سیگنال های ارسالی کنیم که در فایل های ارسالی اینکار انجام شده است.

۲. کدینگ کانال

در این قسمت با استفاده از کلاس comm.TurboEncoder متلب، یک آبجکت از کدینگ توربو تولید کرده ایم و رشته بیت هایی که میخواهیم ارسال کنیم را با آن کد کرده ایم. در رابطه با کدینگ توربو و پارامتر هایی که دریافت میکند میتوان گفت که:

الف) برای مقابله با خطای burst، یک لیست به اندازه رشته بیت ارسالی میگیرد که آن را interleaverIndices مینامد. درباره بلوک اینترلیور میدانیم با تغییر چیدمان بیت های ارسالی، مپینگ کد هارا جابجا میکند تا درصورت وقوع خطای برست، از هر کلمه کد تعداد محدودی بیت دچار خطا شود. بنابراین این آرایه برای جابجا کردن ترتیب بیت های ارسالی است.

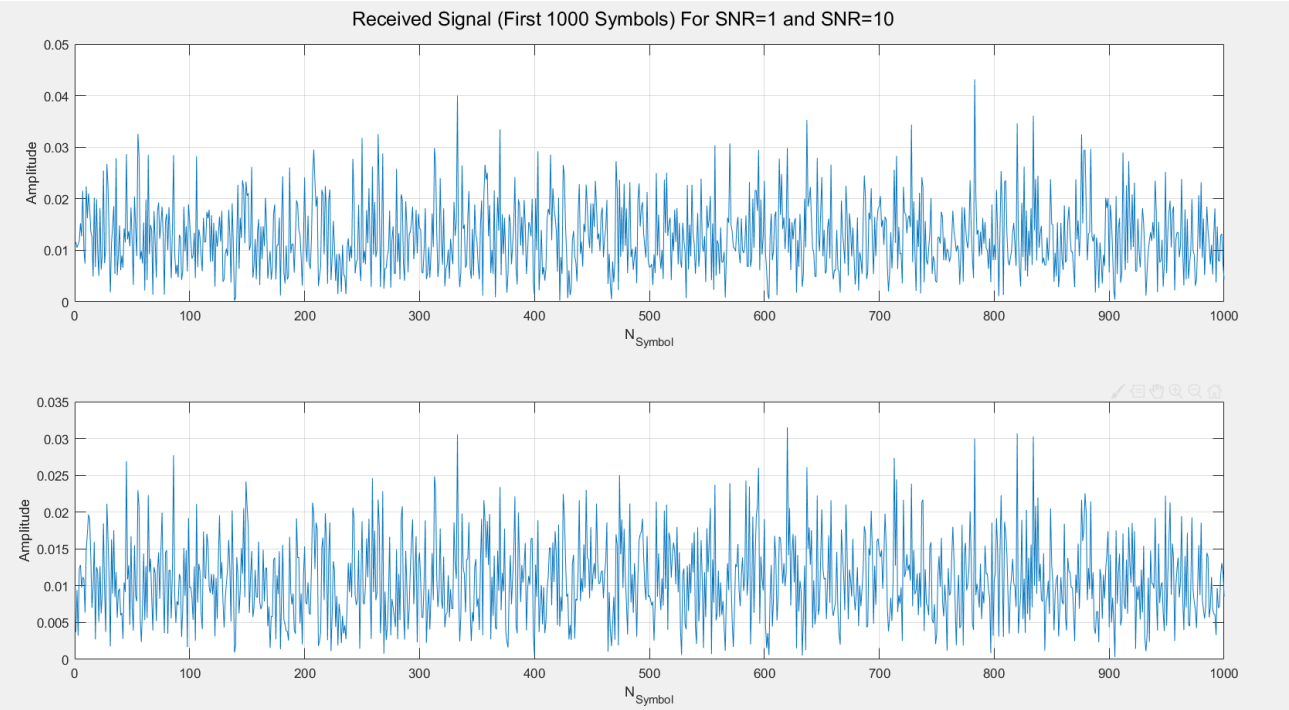
ب) با دریافت Trellis Diagram، ساختار ترلیس مورد استفاده در کدینگ کانولوشنال کانال را ایجاد میکند. این ساختار را با تابع poly2trellis متلب تولید کرده ایم.

۳. مدولاسیون OFDM

در این قسمت، با استفاده از دستور ofdmmod متلب، مدولاسیون کانال ofdm مبتنی بر fft را بر روی سیگنال ارسالی انجام داده ایم تا برروی 20 زیر حامل، اطلاعات را ارسال کنیم. دقت کنید با توجه به اینکه از cyclic prefix به طول ۱۶ استفاده کرده ایم کمی اورهد در دیتا خواهیم داشت.

۴. مدلسازی کانال AWGN

بدون توضیحات اضافه، و مشابه آنچه در تمرین سری قبل انجام دادیم، کانال نویز سفید خواسته شده را در توان های گفته شده ایجاد کردیم و سیگنال ارسالی را از آن عبور دادیم. برای دو نویز خاص سیگنال دریافتی به شکل زیر میباشد:



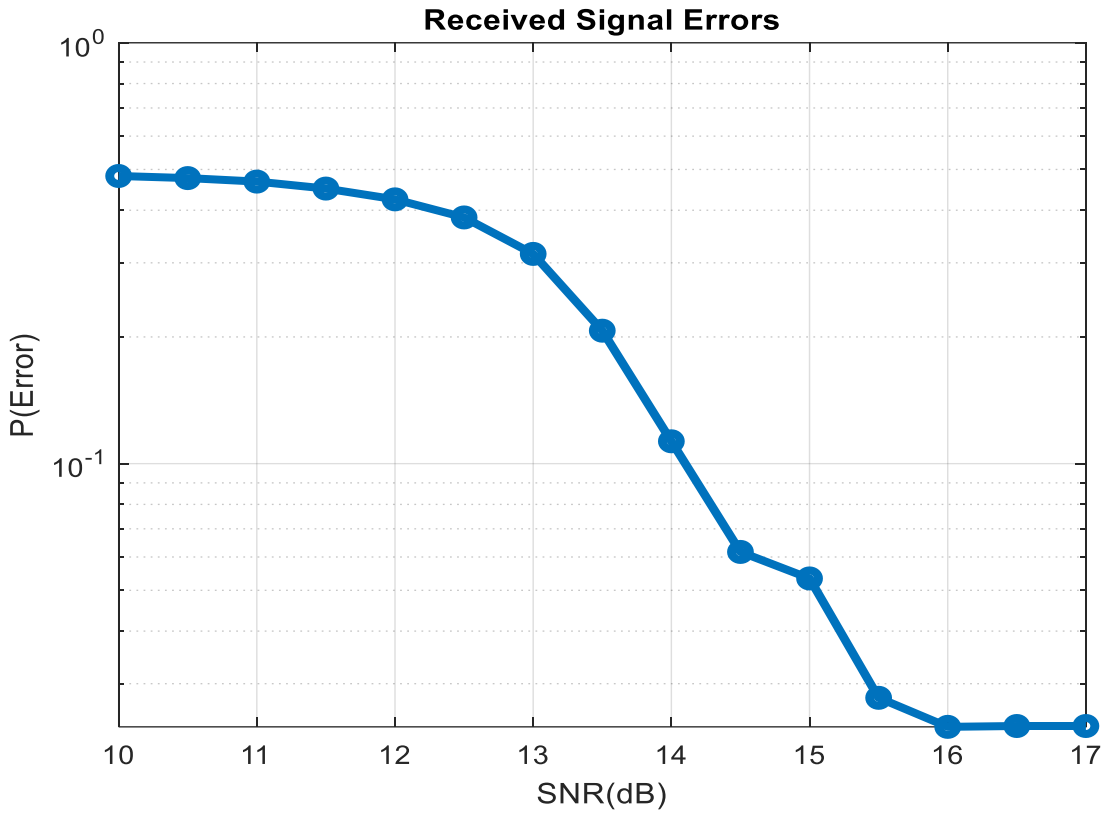
۵. آشکارسازی سمبل ها

معکوس فرایند های کدینگ و مدولاسیون های انجام شده را در سمت گیرنده (که سیگنال نویزی را دریافته کرده است) انجام میدهیم. یعنی ابتدا زیرحامل های ofdm را تشخیص میدهیم، سپس بر اساس constellation ای که در qpsk تولید کردیم یکی از چهار حالت یاد شده را تشخیص میدهیم و سپس دیکودینگ توربو را انجام میدهیم.

۶. محاسبه احتمال خطا

خطا را به صورت نسبت تعداد بیت هایی که به اشتباه دریافت شده اند به تعداد کل بیت ها تعریف میکنیم. دقت کنید اطلاعات دریافتی توسط ۲۰ زیرحامل را با هم به عنوان پیام دریافتی نهایی فرض

کرده و خطای آن را محاسبه کرده ایم. و اینکار را به ازای توان های نویز مختلف یاد شده در دستور کار انجام داده ایم که نتیجه به شرح زیر است:



توجه: دقت کنید برای معنادار تر شدن روند نزولی، بازه SNR را میان ۱۰ تا ۱۷ با رزولوشن ۰.۵ گرفته ایم تا این کاهشی بودن آن بر اثر تاثیر توان نویز دیده شود. شما میتوانید با تغییر بازه تغییرات SNR در خط ۳۶ کد متلب، برای هر بازه دلخواه این تصویر را پلات کنید.

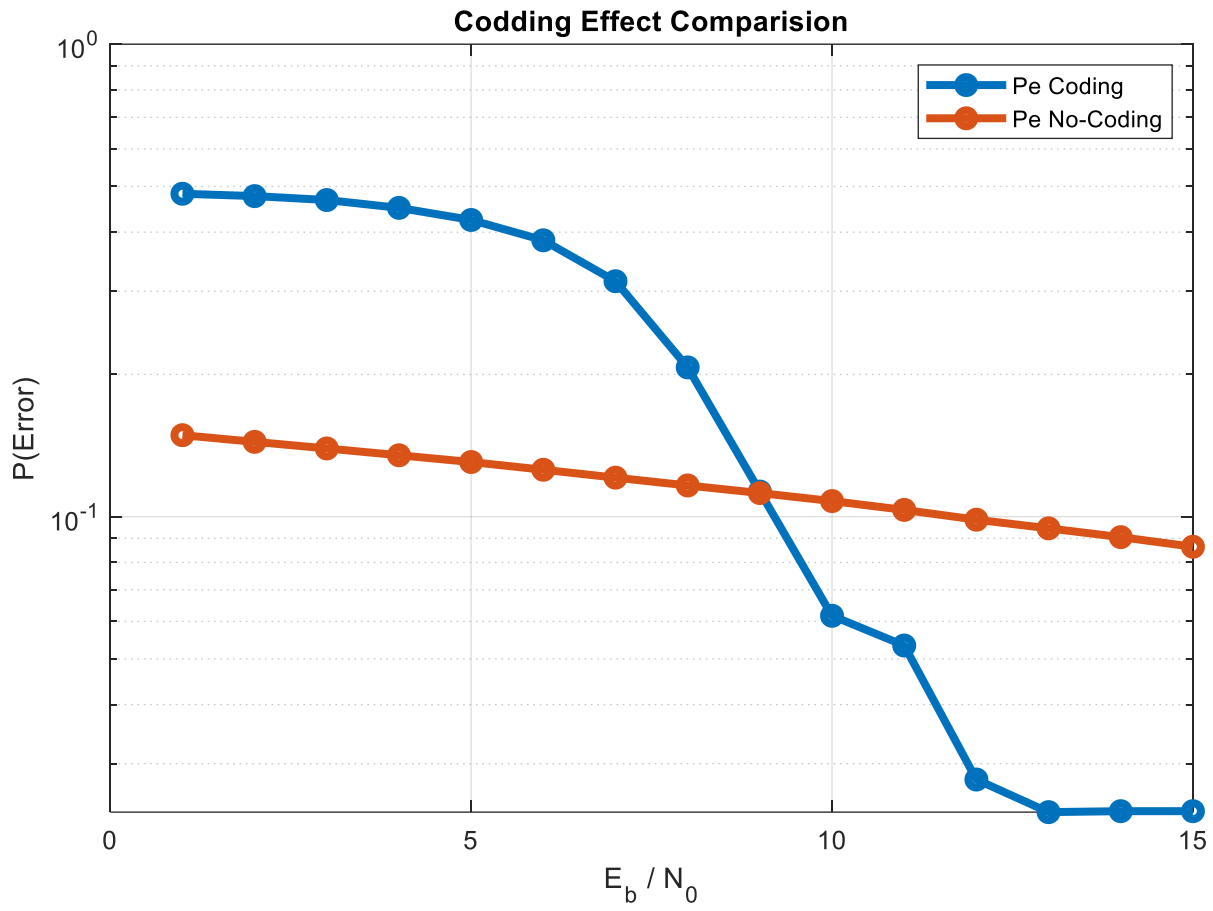
واضح است که میبینیم با افزایش SNR، احتمال خطا بسیار کاهش میابد و برای برخی SNR ها احتمال خطا تقریبا صفر میشود.

۷. بررسی تاثیر کدینگ کانال

باید تمامی مراحل فوق را بدون اعمال کدینگ کانال انجام دهیم، دقت کنید چون فرض کرده ایم توان فرستنده ثابت است اما در حالت کدینگ و غیرکدینگ تعداد کد نابرابری ارسال میکنیم، باید از رابطه زیر کمک بگیریم تا بتوانیم مقایسه درست انجام دهیم:

$$\frac{(SNR)_{nc}}{(SNR)_c} = \frac{E_{b_{nc}}/N_0}{E_{b_c}/N_0} = \frac{E_{b_{nc}}}{E_{b_c}} = \frac{P/R_{nc}}{P/R_c} = \frac{R_c}{R_{nc}} = \widehat{R_c}$$

که در آن $\widehat{R_c}$ برابر با نرخ کدینگ است، که یعنی به ازای هر رشته k بیتی ارسالی، چه مقدار بیت اضافی ارسال میکنیم و مجموع آن را n مینامیم. بنابراین برای مقایسه صحیح تصویر زیر ترسیم میشود:



که واضح است به ازای حداقلی از SNR، اعمال کدینگ بر احتمال خطا بسیار بسیار اثرگذار است و آن را به شدت (و حتی غیر خطی) کاهش میدهد. بنابراین خیلی معقول است در نمونه های واقعی مخابرات از کدینگ کانال حتما استفاده شود.