به نام خدا

دانشگاه صنعتی شریف

دانشکده مهندسی برق

گزارش پروژه نهایی

درس مخابرات دیجیتال

نگارنده

سید محمد اشتهاردیان (۹۸۱۰۹۶۹۱)

نیمسال اول ۰۲-۱۴۰۱

مقدمه

در پروژه درس مخابرات دیجیتال، هدف شبیه‌سازی یک سیستم مخابرات دیجیتال با مدولاسیون‌های QPSK و DQPSK است. میخواهیم اثر نویز موجود در کانال و همچنین ابهام فاز ایجاد شده توسط کانال را توسط این پروژه بررسی نماییم. برای اینکار از نویز سفید گوسی استفاده میکنیم و مقدار ابهام فاز کانال را مضرب صحیحی از $\frac{\pi}{4}$ در نظر میگیریم. همچنین برای کدینگ منبع تعدادی از بیت های LSB هر نمونه را حذف میکنیم. همچنین با کدهای BCH(15,11) و BCH(15,7) به بررسی اثر کدینگ کانال روی احتمال خطا میپردازیم. در نهایت خروجی های این پروژه احتمال های خطا و همچنین خروجی های صوتی بازسازی شده در گیرنده می‌باشند.

توضیح کدها

در این بخش از گزارش میخواهم به توضیح کدهای نوشته شده بپردازم. توابع به کار رفته در این پروژه به شکل های زیر هستند:

- `ZeroRemover` : کار این تابع این است که با دریافت یک آرایه از نمونه‌های یک فایل صوتی مانند x بخش ابتدایی صوت که شامل نویز است و هنوز صدا در آن وجود ندارد را از بقیه صوت جدا کند و این دو بخش را در غالب دو آرایه y_0 و y برگرداند. برای اینکار یک حد $0.1\max(|x|)$ در نظر گرفته شده است که هرگاه شدت صوت از آن بیشتر شد از آن نمونه به بعد نگه داشته می‌شود و نمونه‌های قبلی در آرایه y_0 قرار داده می‌شوند.

- `BitReducer` : این تابع برای کاهش تعداد بیت‌های هر نمونه در یک فایل wav. استفاده می‌شود. این تابع با گرفتن نمونه‌های یک فایل wav. در غالب آرایه مانند x و گرفتن یک عدد صحیح بین 0 تا 16 مانند n از هر نمونه در x تنها n بیت MSB آن را نگه میدارد و بقیه بیت ها را دور میریزد. در خروجی نیز یک آرایه y و یک رشته y_{bit} برمیگرداند. آرایه y صرفا برای تست عملکرد صحیح تابع استفاده شده است و فایده دیگری ندارد اما رشته y_{bit} شامل بیت‌های به هم پیوسته شده نمونه هاست که برای ارسال روی کانال استفاده میشوند.

- `BCH11Encoder` : این تابع کار انکود کردن رشته بیت‌ها را با استفاده از کدینگ BCH(15,11) انجام میدهد.

- `BCH11Decoder` : این تابع کار دیکود کردن رشته بیت‌ها را با استفاده از کدینگ BCH(15,11) انجام میدهد.

- `BCH7Encoder` : این تابع کار انکود کردن رشته بیت‌ها را با استفاده از کدینگ BCH(15,7) انجام میدهد.

- `BCH7Decoder` : این تابع کار دیکود کردن رشته بیت‌ها را با استفاده از کدینگ BCH(15,7) انجام میدهد.
- `QPSKMod` : وظیفه این تابع پیاده‌سازی مدولاسیون QPSK است. این تابع دو ورودی دریافت میکند. یکی انرژی هر بیت و دیگری رشته بیتهای ورودی هستند. اگر تعداد بیتهای ورودی عددی فرد باشد یک صفر نیز به انتهای آنها اضافه میکند تا مدولاسیون به درستی انجام پذیرد. سپس اگر دو بیت ورودی 00 باشند مقدار سیگنال خروجی را برابر $A(1+j)$ ، اگر ورودی 01 باشد خروجی را برابر $A(-1+j)$ ، اگر ورودی 11 باشد خروجی را برابر $-A(1+j)$ و اگر ورودی 10 باشد خروجی را برابر $A(1-j)$ قرار میدهد. دقت کنید در اینجا از gray code استفاده کرده‌ام و همچنین $A=\sqrt{\varepsilon_b}$ است.
- `QPSKDemod` : این تابع برای demodulation یک سیگنال QPSK استفاده می‌شود. برای demodulation نیز کافیست به QPSK مانند دو PAM باینری در دو راستا نگاه کنیم.
- `DQPSKMod` : این تابع مدولاسیون DQPSK را انجام میدهد. روند مدولاسیون درست مانند QPSK است با این تفاوت که در اینجا اختلاف فاز بین دو سیگنال متوالی نشان دهنده بیت های ورودی است. مثلا اگر بیت های ورودی 00 باشند فاز سیگنال تغییری نمیکند اما اگر بیت های ورودی 01 باشند فاز سیگنال به میزان $\frac{\pi}{2}$ افزایش میابد. همچنین بیت های ورودی 11 و 10 نیز به ترتیب معادل افزایش π و $\frac{3\pi}{2}$ فاز سیگنال هستند.
- `DQPSKDemod` : وظیفه این تابع demodulation سیگنال DQPSK است. برای demodulation فاز سیگنال $r_k r_{k-1}^*$ اندازه گیری می‌شود و با توجه به اینکه این فاز برابر با اختلاف فاز دو سیگنال است پس از روی آن میتوان بیتها را بازسازی نمود.
- `Channel` : این تابع همانطور که از نامش مشخص است عملکرد کانال را شبیه سازی میکند. با دریافت عدد N_0 این تابع یک نویز گوسی با $\sigma_n^2=\frac{N_0}{2}$ و میانگین صفر با سیگنال اصلی جمع میکند. همچنین این تابع یک پارامتر به نام phase نیز دریافت میکند که اگر مقدار آن یک باشد یک ابهام فاز تصادفی به میزان $k\frac{\pi}{4}$ به سیگنال اضافه میکند که k عدد صحیح رندومی بین صفر تا هفت میباشد.
- `Bits2Dec` : وظیفه این تابع این است که بیت های رسیده در گیرنده را پس از انجام demodulation و decoding به صورت نمونه‌های یک سیگنال صوتی با فرمت wav. در آورد تا قابل ذخیره سازی باشند.
- `QPSKSimulator` : این تابع وظیفه شبیه سازی یک سیستم مخابرات دیجیتال با مدولاسیون QPSK را دارد که در آن از توابعی که از قبل گفته شد استفاده می‌شود. ترتیب استفاده این توابع به این شکل است که ابتدا کدینگ صورت میگیرد (کدینگ میتواند یکی از سه حالت None ، BCH(15,11) و BCH(15,7) را دارا باشد). سپس مدولاسیون انجام میشود و سپس سیگنال از کانال عبور میکند. پس از آن فرآیند های demodulation و دیکودینگ اتفاق می‌افتند. در نهایت فایل صوتی خروجی ذخیره می‌شود و احتمال خطاهای تئوری و عملی بدست آمده نیز در یک فایل نوشتاری ذخیره می‌شوند.

- `DQPSKSimulator` : این تابع مشابه تابع `QPSKSimulator` برای مدولاسیون DQPSK عمل میکند.

در نهایت برای شبیه سازی نهایی از فایل main.m استفاده کرده‌ام.

نتایج

نتایج احتمال‌های خطا در پوشه error-probability آورده شده‌اند. در این پوشه سه عدد زیرپوشه قرار دارد. یکی از آنها qpsk نام دارد که در آن با فرض عدم وجود ابهام فاز نتایج مدولاسیون QPSK آورده شده است. یکی دیگر از این پوشه‌ها qpsk-phase-ambiguity نام دارد که با فرض وجود ابهام فاز در qpsk نتایج آن آورده شده است که همانطور انتظار داشتیم در حد فاجعه هستند. در نهایت در پوشه dqpsk نتایج حاصل از dqpsk آورده شده‌اند که همانطور که تئوری کلاس بدست آمد $3dB$ ضعیفتر از حالت qpsk هستند اما در عوض میتوانند در برابر ابهام فاز ناشی از کانال مقاوم باشند.
فایل های صوتی خروجی نیز در پوشه sounds قابل پخش هستند. زیر پوشه‌های این پوشه نیز مانند پوشه قبلی نام گذاری شده‌اند.