دانشکده برق، کامپیوتر و فناوری‌های پیشرفته

عنوان:

دستورکار آزمایشگاه مخابرات دیجیتال

تهیه و تنظیم کننده:

سید صدرا کاشف

میرهادی کمالی

تاریخ تنظیم:

تابستان ۱۴۰۳

این صفحه تعمداً خالی گذاشته شده است.

پیش گفتار تنظیم کنندگان:

این دستورکار برای کمک به تدریس مباحث درس آزمایشگاه مخابرات دیجیتال تهیه شده است. سرفصل‌های پیشنهادی در بخش بعدی ارائه گشته است. کدها و شبیه‌سازی‌ها در اختیار مدرس محترم قرار خواهد گرفت. مدرس می‌تواند باتوجه به تشخیص خود از نرم افزار **متلب**، **پایتون** یا ترکیبی از آن‌ها برای انجام شبیه‌سازی‌ها استفاده کند. توصیه می‌گردد در جلسات اولیه جهت آشنایی دانشجویان با مبانی برنامه‌نویسی چند جلسه برای تدریس مقدمات زبان‌های برنامه‌نویسی اختصاص داده شود. در نهایت هفت آزمایش تهیه شده است که مباحث مطرح شده در مخابرات دیجیتال را پوشش می‌دهند و نیز برای دانشجویان علاقه‌مند می‌توانند به عنوان نقطه شروعی برای ادامه دادن مباحث مطرح شده باشند.

ملزومات انجام آزمایش‌ها و دستگاه‌های مورد نیاز به شرح زیر می‌باشد:

- به تعداد مورد نیاز سیستم کامپیوتر که حداقل متلب نسخه ۲۰۱۶ و آناکوندا با پایتون نسخه ۳ و ادیتور اسپایدر بر روی آن‌ها نصب شده باشد.
- کتابخانه‌های numpy و matplotlib بر روی پکیج آناکوندا نصب شده باشند.

در آزمایش‌های آماده شده، ابتدا مبانی مرتبط با آزمایش به صورت خلاصه ارائه شده است. در این بخش، سعی شده کلیات مبحث به صورت خلاصه گفته شود و در صورتی که دانشجویان مطالب را فراموش کرده باشند، می‌توانند به کتب مرجع یا سایر جزوات مراجعه کنند. برای هر کدام از فصل‌های درس مخابرات دیجیتال یک آزمایش تنظیم شده است که مباحث آن فصل را پوشش می‌دهد. پیشنهاد می‌شود که این آزمایش‌ها به صورت کامل توسط دانشجویان شبیه سازی شده و نتایج تولید گردند. در آخر سه آزمایش تکمیلی نیز آماده شده است که می‌توانند هم در کلاس و هم برای دانشجویان علاقه‌مند مورد استفاده قرار گیرند.

روال کار پیشنهادی برای کلاس به این صورت است که دانشجویان به صورت گروهی آزمایش‌ها را انجام دهند. دانشجویان برای هر جلسه یک گزارش کار از مباحث آن جلسه و نتایج بدست آمده از شبیه‌سازی به صورت انفرادی باید ارائه دهند. برای انجام آزمایش‌ها خلاصه‌ای از مبحث می‌تواند در نصف زمان کلاس در هرجلسه تدریس شده و بقیه زمان جلسه به کدنویسی و انجام آزمایش توسط دانشجویان اختصاص یابد. در صورتی که آزمایشی نیمه تمام بماند، دانشجویان می‌توانند ادامه آن را در زمان دیگری انجام دهند و نتایج را در گزارش کار خود بیاورند. بخشی از نمره می‌تواند برای فعالیت کلاسی (مثلا ۲۰ درصد)، بخشی برای گزارش کارهای تحویل داده شده (مثلا ۴۰ درصد) و درنهایت بخشی نیز برای امتحان (مثلا ۴۰ درصد) درنظر گرفته شود. توجه شود که امتحان برای این درس الزامی می‌باشد.

فهرست مطالب:

۱. مقدمه‌ای بر زبان برنامه نویسی پایتون

جهت انجام آزمایش‌ها نیاز است که دانشجویان با یکی از دو زبان برنامه نویسی متلب یا پایتون آشنایی داشته باشند. با توجه به شرایط کلاس و به تشخیص مدرس، یکی از این دو زبان و یا هردو می‌توانند برای انجام آزمایش‌ها مورد استفاده قرار گیرند. سرفصل مباحث پایتون که برای انجام آزمایش‌ها نیاز است در این بخش ارائه شده است.

- متغیرها
  - ✓ تعریف متغیر، حافظه و ذخیره شدن متغیرها، قوانین نام‌گذاری متغیرها
- عملگرها
  - ✓ تعریف عملگرها و عملوندها
  - ✓ عملگرهای حسابی (+. -, *, /, //, %, **)
  - ✓ عملگرهای انتساب (شامل ... =, +=, -=,)
  - ✓ عملگرهای مقایسه‌ای (... <, >, <=, >=, ==, !=,)
  - ✓ عملگرهای شناسایی (is, is not)
  - ✓ عملگرهای عضویت (in, not in)
  - ✓ عملگرهای منطقی (and, or, not)
  - ✓ عملگرهای باینری (&, |, ^, <<, >>)
- داده ساختارها: اعداد صحیح، اعداد اعشاری، و اعداد مختلط
  - ✓ داده ساختارهای اعداد صحیح (Integer) و برخی توابع مقدماتی کار با آن‌ها
  - ✓ اعداد اعشاری (Float) و برخی توابع مقدماتی کار با آن‌ها
  - ✓ بولین (Boolean) و برخی توابع مقدماتی کار با آن‌ها
  - ✓ اعداد مختلط (Complex) و برخی توابع مقدماتی کار با آن‌ها
- داده ساختارها: رشته‌ها، لیست‌ها، تاپل‌ها، دیکشنری‌ها، و مجموعه‌ها
  - ✓ تعریف رشته، جمع کردن رشته‌ها، اندیس در رشته‌ها (indexing)، تغییرناپذیری مقادیر رشته بعد از تعریف (immutable)
  - ✓ تعریف لیست، اندیس المان‌های لیست، ترتیب‌دار بودن المان‌های لیست (ordered)، تغییرپذیری لیست‌ها (mutable)
  - ✓ تعریف تاپل، اندیس در تاپل‌ها، ترتیب‌دار بودن المان‌های تاپل (ordered)، تغییرناپذیری تاپل‌ها (immutable)

- ✓ تعریف دیکشنری، مفاهیم کلید و مقدار (key, value) در دیکشنری بجای اندیس، تغییرپذیری مقادیر دیکشنری پس از تعریف (mutable)
- ✓ تعیریف مجموعه‌ها، نبود المان تکراری در مجموعه‌ها، نبود اندیس در مجموعه‌ها و نحوه دسترسی به المان‌های آن‌ها، مهم نبودن ترتیب عناصر در مجموعه‌ها (not ordered)

- عبارات شرطی
  - ✓ ساختار کلی دستور if
  - ✓ استفاده از ساختار if های تو در تو در برخی کاربردها
- حلقه‌های تکرار
  - ✓ ساختار کلی حلقه تکرار for
  - ✓ ساختار کلی حلقه تکرار while
  - ✓ استفاده از حلقه‌های تو در تو
  - ✓ کاربرد کلیدواژه‌های pass, break, continue
- توابع و متدها
  - ✓ فلسفه ایجاد توابع
  - ✓ برخی توابع مهم در پایتون (input(), print(), bin(), abs(), max(), min(), sum())
  - ✓ متدها و تفاوت آن با توابع
  - ✓ برخی متدهای مهم در پایتون (find(), Append(), index(), sort())
- آشنایی با کتابخانه‌ها و نحوه استفاده از آن‌ها
  - ✓ نصب کردن کتابخانه‌ها
  - ✓ روش‌های import کردن کتابخانه‌ها در برنامه
  - ✓ برخی کتابخانه‌های ابتدایی مهم مثل math, time, random
- کتابخانه numpy
  - ✓ تعریف آرایه در نامپای
  - ✓ اندیس‌ها در آرایه‌های نامپای
  - ✓ متدهای مرتبط با شکل و اندازه آرایه‌ها مثل size, shape, ndim
  - ✓ ساخت آرایه‌های صفر و یک با numpy.zeros, numpy.ones
  - ✓ ساخت آرایه‌های خطی با توابع np.arrange, np.linspace
  - ✓ متدهای np.reshape و np.concatenate
  - ✓ عملگرها در آرایه‌های نامپای (عملگرهای عادی)
  - ✓ عملگرهای ماتریسی در آرایه‌های نامپای (ضرب ماتریسی، دترمینان ماتریس، معکوس ماتریس)

- ✓ برخی توابع مهم نامپای (np.sin(), np.cos(), np.log(), np.sinc(), np.sqrt(), np.exp(), np.real(), np.imag(), np.abs(), np.floor(), np.ceil(), np.mean(), np.std(), np.var(), np.sum(), np.min(), np.max(), np.all(), np.any(), np.where())

- کتابخانه matplotlib
  - ✓ مباحث مقدماتی در مورد ترسیم‌ها در پایتون
  - ✓ ترسیم نمودار خطی (plot - با درونیابی خطی)
  - ✓ ترسیم نمودارهای میله‌ای (bar chart)
  - ✓ ترسیم هیستوگرام (histogram)
  - ✓ ترسیم نمودارهای پراکندگی (scatter plot)
  - ✓ ترسیم نمودارهای پای (pie chart)

۲. مقدمه‌ای بر زبان برنامه نویسی متلب

سرفصل مباحث متلب که برای انجام آزمایش‌ها نیاز است در این بخش ارائه شده است.

- معرفی نحوه استفاده از help متلب
- معرفی Community های متلب و متلب exchang
- متغیرها
  - ✓ تعریف متغیر، حافظه و ذخیره شدن متغیرها، قوانین نام‌گذاری متغیرها
- عملگرها
  - ✓ تعریف عملگرها و عملوندها
  - ✓ عملگرهای حسابی (+. -, *, /, //, %, **)
  - ✓ عملگرهای انتساب (شامل ... =, +=, -=, =+)
  - ✓ عملگرهای مقایسه‌ای (... =, ~=, ==, >=, <=, >, <)
  - ✓ عملگرهای شناسایی (is, is not)
  - ✓ عملگرهای عضویت (in, not in)
  - ✓ عملگرهای منطقی (and, or, not)
  - ✓ عملگرهای باینری (&, |, ^, <<, >>)
- داده ساختارها: اعداد صحیح، اعداد اعشاری، و اعداد مختلط
  - ✓ داده ساختارهای اعداد صحیح (Integer) و برخی توابع مقدماتی کار با آن‌ها
  - ✓ اعداد اعشاری (Double) و برخی توابع مقدماتی کار با آن‌ها
  - ✓ منطقی (Logical) و برخی توابع مقدماتی کار با آن‌ها
  - ✓ اعداد مختلط (Complex) و برخی توابع مقدماتی کار با آن‌ها
- داده ساختارها: رشته‌ها، لیست‌ها، تاپل‌ها، دیکشنری‌ها، و مجموعه‌ها
  - ✓ تعریف رشته، (String)
  - ✓ تعریف بردار و ماتریس
  - ✓ تعریف سلول (Cell)
  - ✓ تعریف ساختار (Struct)
- عبارات شرطی
  - ✓ ساختار کلی دستور if
  - ✓ استفاده از ساختار if های تو در تو در برخی کاربردها
- حلقه‌های تکرار

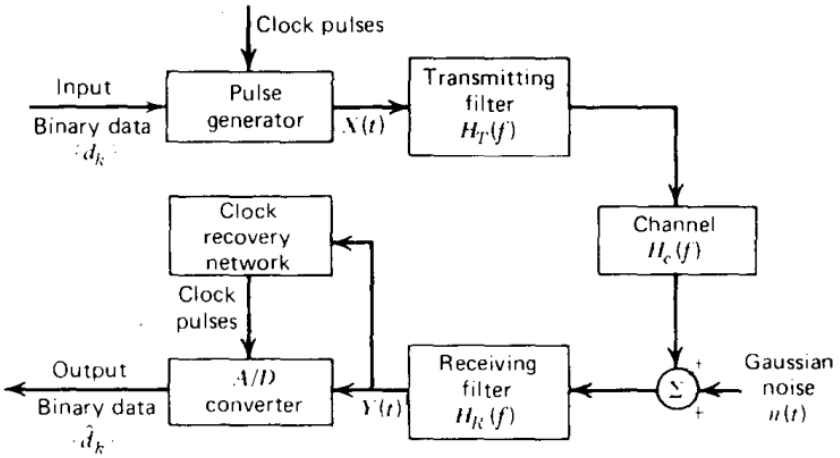
- ✓ ساختار کلی حلقه تکرار for
- ✓ ساختار کلی حلقه تکرار while
- ✓ استفاده از حلقه‌های تو در تو
- ✓ کاربرد کلیدواژه‌های pass, break, continue

- توابع و متدها
  - ✓ فلسفه ایجاد توابع
  - ✓ برخی توابع مهم (input(), print(), abs(), max(), min(), sum())
  - ✓ متدها و تفاوت آن با توابع
  - ✓ برخی متدهای مهم (find(), index(), sort())
- آشنایی با برخی توابع مهم
  - ✓ تابع load
  - ✓ تابع randn
  - ✓ تعریف توابع اختیاری $f(x)$
  - ✓ تابع cat
  - ✓ تابع reshape
  - ✓ تابع cell2mat
  - ✓ تابع plot و زیر توابع مربوطه
  - ✓ تابع zeros و ones
  - ✓ تابع size و length
  - ✓ تابع scatter
  - ✓ معرفی انواع توابع ریاضی موجود و مورد نیاز مثل توابع مثلثاتی و لگاریتمی
  - ✓ عملگرهای مهم (جذر، توان ...)
  - ✓ عملگرهای ماتریسی (ضرب ماتریسی، دترمینان ماتریس، معکوس ماتریس)
  - ✓ عملگرهای رسم و قابلیت‌های مختلف آن مثل هیستوگرام، legend، اضافه نمودن توضیحات، محدود نمودن مختصات و ...
  - ✓ تبدیل فوریه

۳. آزمایش اول: مفهوم انتقال پالس باند پایه

مقدمه

هدف سیستم‌های مخابراتی، انتقال کپی پیام از مبدا به مقصد می‌باشد که می‌تواند به صورت الکتریکی، رادیویی، صوتی، تصویری یا ... باشد. پیام‌ها را می‌توان به دو دسته پیام آنالوگ (با تغییرات پیوسته) و پیام دیجیتال (با تغییرات گسسته) طبقه‌بندی کرد. پیام آنالوگ را می‌توان به صورت آنالوگ روی کانال ارسال نمود که در درس اصول سیستم‌های مخابرتی مورد بحث قرار گرفته است. اگر بخواهیم پیام آنالوگ را به صورت دیجیتال بفرستیم، می‌بایست مراحل نمونه برداری، کوانتیزه کردن و کدبندی انجام گیرد. ارسال دیجیتال می‌تواند در باند پایه یا باند میانی صورت بگیرد. شمای کلی یک سیستم مخابراتی دیجیتال باینری باند پایه را می‌توان به صورت شکل ۱ در نظر گرفت.



شکل ۱. سیستم مخابراتی باند پایه باینری

محدودیت‌های موجود برای کانال‌های مخابراتی در چهار دسته طبقه‌بندی می‌شوند.

- تلفات که باعث تضعیف و کاهش دامنه سیگنال می‌شود.
- اعوجاج یا همان تغییر شکل سیگنال در طول کانال تاثیر دوم کانال بر روی سیگنال می‌باشد که به دسته‌های خطی، غیرخطی و متغیر بازمان تقسیم می‌شود که برای جبران اعوجاج خطی می‌توان از همسان‌سازها استفاده نمود.
- تداخل عارضه بعدی کانال می‌باشد که سیگنال‌هایی که در باند فرکانسی یکسان قرار داشته باشند، باهمدیگر تداخل خواهند داشت.
- نویز به عنوان آخرین اثر کانال در نظر گرفته می‌شود. نویز می‌تواند ریشه‌های متفاوتی داشته باشد که مهم‌ترین آن‌ها نویز حرارتی می‌باشد. نویز قابل حذف نبوده و می‌بایست اثر آن‌را تعدیل و طراحی

سیستم‌ها را در حضور آن انجام دهیم. مهم‌ترین نویزی که در سیستم‌های مخابراتی با آن سر و کار داریم، نویز سفید گوسی جمع شونده در گیرنده می‌باشد.

انتقال باندپایه می‌تواند به صورت‌های گوناگونی انجام شود (PAM, PDM, PPM) که هر کدام دارای ویژگی-های خود می‌باشند. در این بخش انتقال PAM مورد بحث قرار می‌گیرد. کانال انتقال اثرات متعددی بر روی پالس انتقالی خواهد داشت. مثلا تضعیف، تاخیر، نویز و تداخل بین سمبلی (ISI) می‌توانند باعث ایجاد خطا در سیستم شوند. ISI در واقع اثر دریافتی از پالس‌های ارسال شده در لحظات قبل‌تر می‌باشد. می‌توان شکل پالس‌های ارسالی را برای کاهش یا حذف اثرات آن طراحی نمود. یکی از پرکاربردترین پالس‌های استفاده شده، پالس کسینوس برجسته می‌باشد. رابطه این پالس در ادامه مباحث ارائه شده است. برای شکل‌دهی پالس باید طراحی فیلترهای فرستنده و گیرنده انجام شوند که در درس مخابرات دیجیتال مفصلا مورد بحث قرار گرفته است. پس از آن میزان احتمال خطای بیت برای سیستم طراحی شده محاسبه می‌گردد تا معیاری برای ارزیابی کارایی سیستم در مقایسه با سایر سیستم‌ها باشد. برای آشنایی با طراحی سیستم مذکور، آزمایش زیر ارائه شده است.

انجام آزمایش

۱. سیگنال نویز گوسی با میانگین صفر و واریانس یک تولید کرده و رسم نمایید.

راهنمایی پایتون: دستور تولید سیگنال تصادفی با توزیع گوسی با میانگین صفر و واریانس یک با استفاده از کتابخانه numpy در پایتون np.random.randn می‌باشد. برای ترسیم‌ها می‌توان از ماژول matplotlib.pyplot و تابع plot استفاده نمود.

راهنمایی متلب: دستور تولید سیگنال تصادفی با توزیع گوسی با میانگین صفر و واریانس یک با استفاده از randn در متلب می‌باشد. برای ترسیم‌ها می‌توان تابع plot استفاده نمود.

۲. پنج پالس کسینوسی صعودی پشت سر هم با فاصله‌ی $T_b = 1$ از هم و برای دو مقدار $\beta = 0{,}1$ تولید کرده و به صورت یکجا و در یک شکل هم در حوزه زمان و هم در حوزه فرکانس رسم نمایید. مقدار A_k برابر است با:

$$A_1 = 1, A_2 = -1, A_3 = 1, A_4 = -1, A_5 = -1$$

راهنمایی: رابطه پالس کسینوس صعودی برابر $f(t) = \frac{\cos(2\pi Bt)}{1-(4Bt)^2} sinc(t)$ می‌باشد. برای ترسیم سایر پالس‌ها می‌بایست t را به اندازه مدنظر شیفت داد (برای یک ثانیه شیفت به جای t در رابطه، t-1 قرار داد). برای ترسیم مجددا می‌توان از تابع plot استفاده نمود.

۳. سیگنال مرحله ب را با نویز گوسی به ازای سه مقدار $SNR = 10, 30, 100$ جمع کرده و هم در حوزه زمان و هم فرکانس رسم نمایید.

راهنمایی: با در دست داشتن سیگنال اصلی می‌توان توان سیگنال را بدست آورد. به این صورت که توان سیگنال برابر با $\frac{1}{N}\sum_{n=0}^{N-1}|x(n)|^2$ است. با در دست داشتن مقدار *SNR* و توان سیگنال، می‌توان توان نویزی که باید اضافه شود را بدست آورد. سپس از این توان برای تولید

نویز گوسی استفاده می‌شود. در نهایت با جمع کردن سیگنال اصلی و نویز تولید شده، سیگنال نویزی بدست می‌آید. تبدیل ***dB*** به ***decimal*** برای توان با استفاده از رابطه $dB = 10\log_{10}\left(\frac{P_2}{P_1}\right)$ انجام می‌شود.

توضیحات: ترسیم سیگنال نویزی در حوزه زمان مثل مراحل قبل انجام می‌شود. اما برای ترسیم حوزه فرکانس آن، ابتدا می‌بایست تبدیل فوریه (***DTFT***) سیگنال را محاسبه کرد. بجای محاسبه ***DTFT***، می‌توان از ***FFT*** که نسخه سریعتر محاسبه ***DFT*** می‌باشد استفاده نمود. ***DFT*** در واقع به تعداد نمونه‌های سیگنال از ***DTFT*** نمونه برمی‌دارد یا به عبارتی ضرایب سری فوریه‌ی نسخه متناوب شده سیگنال اصلی را تولید می‌کند که برابر با قبلی است. سپس با ترسیم این نمونه‌ها و وصل کردن آن‌ها به هم با استفاده از دستور ***plot***، پوش تقریبی تبدیل فوریه سیگنال بدست می‌آید. برای نشان دادن صحیح فرکانس‌ها (فرکانس‌های پایین حوالی صفر) پس از گرفتن ***fft***، می‌توان از ***fftshift*** برای جابجایی آن در روی محور استفاده نمود.

۴. بهترین نقاط نمونه برداری برای آشکارسازی پنج پالس مرحله ۲ را تعیین کرده و مقدار پالس را در آن نقاط بدست آورید.

راهنمایی: بهترین نقاط نمونه برداری با استفاده از تئوری این مبحث بدست می‌آید. برای محاسبه مقدار سیگنال در یک شماره نمونه خاص، در پایتون کافی است شماره نمونه را در داخل کروشه مقابل متغیر تعریف شده برای سیگنال قرار داد. برای مثال $signal[10]$ نمونه شماره ۱۰ام متغیری به اسم ***signal*** را نمایش می‌دهد. اگر مقدار سیگنال در یک زمان خاص مد نظر باشد، ابتدا می‌بایست با استفاده از نرخ نمونه برداری، شماره نمونه در آن زمان را بدست آورد و سپس مقدار سیگنال در آن را محاسبه نمود.

۵. مرحله ۴ را برای سیگنال مرحله ۳ در سه SNR مختلف انجام دهید.

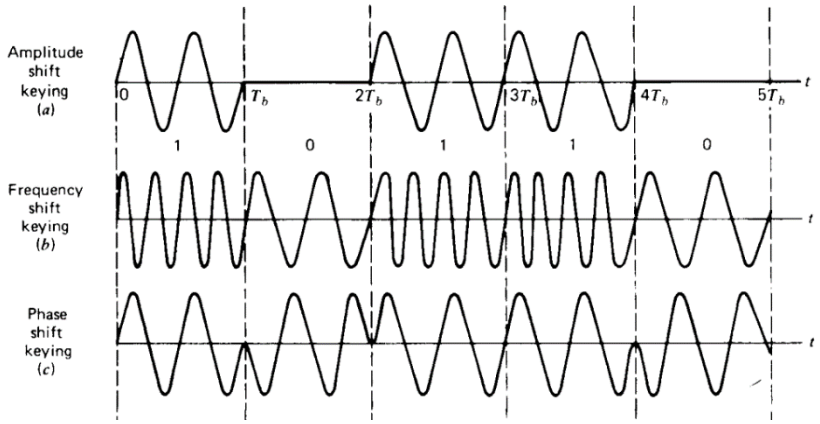
راهنمایی: باتوجه به اینکه نویز سفید گوسی جمع شونده می‌باشد، نقاط نمونه برداری تغییر نمی‌کنند.

۶. نتایج مرحله ۴ و ۵ را باهم مقایسه کنید.

۴. آزمایش دوم: مدولاسیون‌های دیجیتال

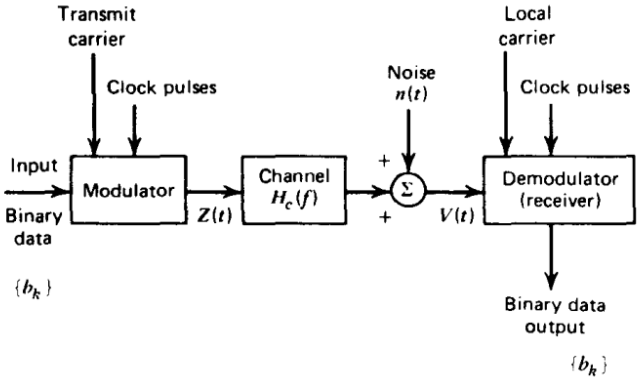
مقدمه

مدولاسیون به معنی تحت اللفظی به معنی تنظیم کردن می‌باشد و به مفهوم تبدیل یکسری علائم به سیگنال-های قابل ارسال به کار می‌رود. در آزمایش قبلی چندین روش برای انتقال اطلاعات دیجیتالی روی کانال های باند پایه با استفاده از تکنیک‌های ***PAM*** باند پایه گسسته مورد بحث قرار گرفت. اما در عمل اکثر کانال‌های ارتباطی در اطراف فرکانس صفر پاسخ بسیار ضعیفی دارند و به عنوان کانال‌های میان‌گذر شناخته می‌شوند. در نتیجه برای انتقال اطلاعات دیجیتال روی کانال‌های میان‌گذر، مجبور هستیم که اطلاعات را به یک موج حامل با فرکانس متناسب انتقال دهیم. اطلاعات دیجیتال می‌تواند توسط موج‌های حامل متعددی نمایندگی شود که مبنای مدولاسیون‌های دیجیتال را تشکیل می‌دهند. در این آزمایش، تعدادی از مدولاسیون‌های دیجیتال مشهور مورد بررسی قرار می‌گیرند که شامل مدولاسیون دامنه (***Amplitude Shift Keying***)، مدولاسیون فاز (***Phase Shift Keying***) و مدولاسیون فرکانس (***Frequency Shift Keying***) می‌باشند. در شکل ۲ نمونه شکل‌موج‌های هر کدام از این مدولاسیون‌های دیجیتال ترسیم شده است.



شکل ۲. نمونه شکل موج‌های مدوله شده با روش ***ASK, PSK, FSK***

شمای کلی یک سیستم انتقال داده باند میانی باینری در شکل ۳ نشان داده شده است. همانطور که در شکل نشان داده شده است، مدولاتور بخشی از فرستنده بوده و کار تنظیم پیام ارسالی روی سیگنال متناسب با کانال در دسترس را بر عهده دارد و دمدولاتور بخشی از گیرنده می‌باشند.



شکل ۳. سیستم انتقال داده باند میانی باینری

انجام آزمایش

۱. سیگنال خروجی فرستنده برای مدولاسیون‌های *ASK*، *PSK* و *FSK* با سیگنال حامل $\cos(2\pi f_c t)$ را برای رشته بیت زیر بسازید:

$$Input = [1\ 1\ 0\ 0\ 0\ 1\ 1\ 0], f_c = 20\ Hz, T_b = 0.5\ s$$

راهنمایی: در شکل ۲ هرکدام از این مدولاسیون‌ها مشخص شده‌اند.

۲. برای سیستم *FSK* یک f_d به کار گیرید که سیستم قابل ساخت باشد.

۳. سیگنال خروجی فرستنده را در هرکدام از مدولاسیون‌ها بر حسب زمان رسم نمایید (به صورت سه شکل جدا).

راهنمایی: برای هرکدام از مدولاسیون‌ها ابتدا باید شکل موج هر کدام از سمبل‌ها را تعریف نماییم. سپس از روی رشته بیت ورودی، به ازای هر کدام از بیت‌ها، شکل موج متناظر با آن را می‌بایست تولید کرد. در پایتون برای این منظور می‌توان یک لیست خالی ساخت، سپس با استفاده از یک حلقه *for*، به ازای هر کدام از بیت‌های ورودی، یکی از سیگنال‌ها را به آن لیست افزود.

۴. چگالی طیف توان سیگنال ورودی فرستنده را در هر کدام از مدولاسیون‌ها رسم نمایید (به صورت سه شکل جدا).

راهنمایی: هر کدام از این سیگنال‌های تولید شده یک فرآیند تصادفی بوده و نمی‌توان برای آن‌ها تبدیل فوریه محاسبه نمود. ولی روشی وجود دارد که می‌توان اندازه تبدیل فوریه به توان دو فرآیندهای تصادفی را محاسبه کرد. برای این منظور ابتدا خودهمبستگی سیگنال بدست می‌آید و سپس تبدیل فوریه آن محاسبه می‌گردد که به سیگنال حاصل چگالی طیف توان گفته می‌شود. برای بدست آوردن چگالی طیف توان در متلب می‌توان از دستور *pwelch* و در پایتون از دستور *welch* در ماژول *scipy.signal* استفاده نمود.

۵. سیگنال در ورودی گیرنده، بعد از جمع شدن نویز گوسی جمع شونده سفید با ***SNR=10*** دسیبل را در هر کدام از مدولاسیون‌ها بر حسب زمان رسم نمایید (به صورت سه شکل جدا).

راهنمایی: مشابه آزمایش اول، نویز گوسی سفید جمع شونده تولید شده و سپس با استفاده از ***SNR*** مقدار توان آن تعیین می‌گردد. این نویز با سیگنال اصلی جمع شده و سیگنال نهایی بدست می‌آید.

۶. در صورت استفاده از گیرنده بهینه، احتمال خطای مدولاسیون ***PSK*** را برای سیستم همزمان و غیر همزمان آن از ***SNR = 0*** تا ***SNR = 20*** دسیبل رسم نمایید.

راهنمایی: رابطه احتمال خطای مدولاسیون ***PSK*** در درس مخابرات دیجیتال محاسبه شده است و در اینجا نتیجه نهایی ارائه می‌شود:

$$P_{e,PSK}^{coherent} = \mathrm{Q}\left(\sqrt{SNR}\right) \quad (1)$$

$$P_{e,PSK}^{non-coherent} = \frac{1}{2}e^{-\left(\frac{SNR}{2}\right)} \quad (2)$$

۷. در صورت استفاده از گیرنده بهینه، احتمال خطاری مدولاسیون ***ASK*** را برای سیستم همزمان و غیر همزمان آن از ***SNR = 0*** تا ***SNR = 20*** دسیبل رسم نمایید.

راهنمایی: رابطه احتمال خطای مدولاسیون ***ASK*** در درس مخابرات دیجیتال محاسبه شده است و در اینجا نتیجه نهایی ارائه می‌شود:

$$P_{e,ASK}^{coherent} = \mathrm{Q}\left(\sqrt{\frac{SNR}{4}}\right) \quad (3)$$

$$P_{e,ASK}^{non-coherent} = \frac{1}{2}e^{-\left(\frac{SNR}{16}\right)} \quad (4)$$

۸. در صورت استفاده از گیرنده بهینه، احتمال خطای مدولاسیون ***FSK*** را برای سیستم همزمان و غیر همزمان آن از ***SNR = 0*** تا ***SNR = 20*** دسیبل رسم نمایید.

راهنمایی: رابطه احتمال خطای مدولاسیون ***FSK*** در درس مخابرات دیجیتال محاسبه شده است و در اینجا نتیجه نهایی ارائه می‌شود:

$$P_{e,FSK}^{coherent} = \mathrm{Q}\left(\sqrt{0.6 \times SNR}\right) \quad (5)$$

$$P_{e,FSK}^{non-coherent} = \frac{1}{2}e^{-\left(\frac{SNR}{4}\right)} \quad (6)$$

برای ترسیم‌ها در نمودارهای ***loglog*** می‌توان از دستور ***plt.loglog*** از کتابخانه ***matplotlib*** استفاده نمود. در متلب نیز از دستور ***semilog*** استفاده نمایید.

۹. سیگنال ***QPSK*** را در یک توان دلخواه برای رشته بیت سوال بالا بسازید. منظومه نقاط آن را در ***SNR=5,100*** دسیبل رسم کرده و تمام موارد خواسته شده در قسمت‌های ***2***، ***3***، ***4***، و ***5*** سوال ***A*** را برای مدولاسیون ***QPSK*** در دو ***SNR*** تکرار نمایید.

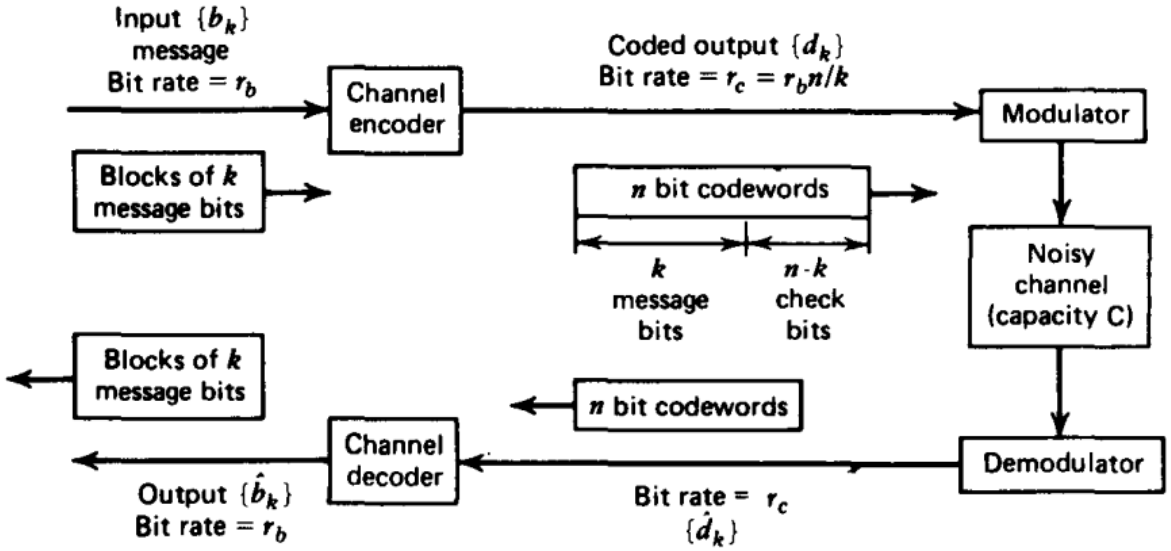
راهنمایی: در مدولاسیون ***QPSK*** چهار شکل موج داریم. بنابراین هر کدام از شکل موج‌ها می‌توانند مقدار دو بیت را نمایندگی کنند. به عبارتی، هر دو بیت را می‌توان با یکی از چهار شکل موج نشان داد. بنابراین ابتدا باید رشته بیت ارائه شده در بخش قبلی، باید به صورت دو بیت به دو بیت جدا شده و سپس به ازای هر دو بیت، یکی از شکل موج‌ها به لیست اضافه شود. برای رسم منظومه ابتدا می‌بایست نقاط

متناظر با هر کدام از شکل موج‌ها را مشخص کرده سپس آن‌ها روی نمودار رسم نمود. برای این منظور می‌توان از دستور ***plot*** استفاده نمود. برای حالت نویزی نیز می‌توان این رویکرد را در پیش گرفت و تعدادی نمونه نویزی تولید کرده و سپس مکان آن‌ها را در منظومه مشخص کرد. در نهایت می‌توان همه آن‌ها را در یک شکل رسم کرده و در کنار یکدیگر مشاهده نمود.

۵. آزمایش سوم: کدگذاری کانال با استفاده از روش کدگذاری قالبی خطی

مقدمه

در آزمایش‌های قبلی دیدیم که احتمال خطا برای یک سیستم مخابراتی مشخص تابعی از *SNR* در ورودی گیرنده می‌باشد. در سیستم‌های عملی بالا بردن توان سیگنال از یک حدی بیشتر شدنی نیست؛ زیرا برخی قوانین برای حداکثر میزان توان وجود دارد که بالاتر از آن می‌تواند صدمات جبران ناپذیری برای سلامتی انسان ایجاد کند یا باعث تداخل در کار سایر سیستم‌ها شود. همچنین توان نویز وابسته به محیط بوده و یک مقدار ثابت برای آن فرض می‌شود. بعلاوه در طراحی سیستم، پارامترهای مختلف برای المان‌های سیستم باید طوری در نظر گرفته شوند که سیستم ساده‌تر و با هزینه کمتر ساخته شود. با در نظر گرفتن تمامی این پارامترها و رعایت نکات آن در طراحی، همچنان ممکن است سیستم‌ها به میزان احتمال خطای قابل قبولی برای یک کاربرد خاص دست پیدا نکنند. در نتیجه برای کاهش احتمال خطا در سیستم‌ها، یک راهکار دیگر ارائه شده است که عبارت است از کدگذاری کانال (یا کدگذاری برای کنترل خطا). شمای کلی کدگذاری کانال در کنار یک سیستم مخابرات دیجیتال در شکل ۴ رسم شده است.



شکل ۴. کد گذاری کانال و دیکدر آن

برای کدگذاری کانال روش‌های گوناگونی توسعه داده شده‌اند که از جمله آن‌ها می‌توان به کدهای خطی بلوکی (قالبی)، کدهای گردشی و کدهای کانولوشنی اشاره کرد. در این آزمایش کد قالبی خطی معرفی شده و مورد بررسی قرار می‌گیرد. برای این منظور، چند بیت تحت عنوان بیت‌های توازن یا چک بیت به بیت‌های اطلاعات اضافه می‌شود و از روی آن‌ها سیستم می‌تواند تشخیص خطا یا تصحیح خطا را انجام دهد.

کد قالبی خطی یک کد سیستماتیک می‌باشد به این معنی که جای بیت‌های اطلاعات و بیت‌های توازن همگی معلوم است. در این روش از یک ماتریس مولد استفاده می‌شود که برای تولید کلمه کد از روی بیت‌های داده به کار می‌رود. در سمت فرستنده این کد ساخته می‌شود و متقابلا در سمت گیرنده، می‌بایست داده از روی کد دریافتی بدست بیاید. همچنین از روی بیت‌های توازن خطاهای احتمالی در داده ارسالی نیز وابسته به میزان قابلیت روش استفاده شده می‌تواند مشخص شود.

یک سیستم مخابراتی ساده را فرض کنید که از یک بخش فرستنده شامل کدگذار کانال، یک کانال مخابراتی که ممکن است مقدار برخی بیت‌ها را تغییر دهد و یک گیرنده شامل دیکدکننده کانال تشکیل شده است. برای هر کدام از بخش‌های این سیستم مخابراتی، موارد زیر را انجام دهید.

انجام آزمایش

۱. در بخش فرستنده، با استفاده از روش کدگذاری قالبی خطی، سه کلمه کد ۱۵ بیتی با قدرت تصحیح یک بیت بسازید.

راهنمایی: از ماتریس بررسی توازن برای ساخت استفاده کنید.

۲. در بخش کانال، در یکی از کدها یک بیت، در دیگری دو بیت و در دیگری سه بیت را تغییر دهید (بیت ۱ را به ۰ و ۰ را به ۱ تغییر دهید). در هر کدام از حالات رشته کد دریافت شده توسط گیرنده را چاپ نمایید.

راهنمایی: در عمل کانال نویز گوسی سفید جمع شونده یک مقدار نویز به سیگنال اضافه می‌کند که مقدار آن می‌تواند مثبت یا منفی باشد. در بخش گیرنده، از سیگنال نمونه برداری شده و بررسی می‌شود که نمونه بدست آمده به کدام یک از سطوح موجود برای شکل‌های مختلف نزدیک‌تر است (عمل آشکارسازی). اگر نویز باعث شود که سیگنال ارسالی به اشتباه آشکار سازی شود، یک خطا رخ می‌دهد، مثلا وقتی بیت ۱ ارسال شده باشد و ۰ آشکار سازی شود، گوییم خطا رخ داده است. در این سوال برای شبیه سازی کانال، جهت ساده‌تر کردن مساله، از این مباحث صرفنظر شده است و صرفا با تغییر یکی از بیت‌های کد دریافتی، می‌توان این کانال را شبیه‌سازی نمود. در نتیجه، هنگام شبیه سازی، فقط کافی است صفر، یک یا دو تا از بیت‌های کد ارسالی را تغییر داده و آن را به عنوان کد دریافتی تلقی نمایید.

۳. در بخش گیرنده، از کلمه کد دریافت شده، رشته بیت ارسالی و خطای احتمالی رخ داده در آن را پیدا کنید.

راهنمایی: در بخش گیرنده، کلمه کد دریافتی و ماتریس P استفاده شده در فرستنده در اختیار است. ابتدا مقدار ترانهاده ماتریس P را بدست آورده و سپس یک ماتریس ***identity*** با ابعاد بیت‌های توازن اضافه شده (در اینجا سه در سه) تولید نمایید. سپس با کنار هم قرار دادن این دو ماتریس (***concatenate***) ماتریس H را بدست بیاورید. پس از آن ماتریس ترانهاده H را تولید نمایید. با ضرب ماتریسی باینری کد دریافت شده (یک کد هشت بیتی) و ماتریس ترانهاده H، بردار S بدست می‌آید. تعداد بیت‌های بردار S برابر با سه خواهد بود. در نهایت با مقایسه S بدست آمده با سطرهای ماتریس H ترانهاده، می‌توان شماره بیتی که در آن خطا رخ داده است را پیدا نمود.

۴. در مرحله اول ۲۰ بردار داده به صورت تصادفی تولید نموده، کلمه کد مربوطه را طبق مرحله اول ساخته و پشت سر هم با استفاده از تکنیک ***interleaving*** با $\lambda = 5$ ارسال نمایید.

۵. در بخش کانال سیگنال ارسالی را دچار خطای قطاری با نه بیت نمایید.

۶. در سمت گیرنده، کلمه کد ارسالی را دی‌کد نموده و بردارهای داده دریافتی را با ارسالی مقایسه نمایید.

۶. آزمایش چهارم: کد گذاری منبع: روش شانون–فانو

مقدمه

سه جمله خبری مشهور زیر را در نظر بگیرید:

- جمله اول: تهران پایتخت ایران است.
- جمله دوم: فردا برف خواهد بارید (در یک روز تابستانی).
- جمله سوم: فردا شهاب سنگ بزرگی به کره زمین برخورد کرده و آن را نابود خواهد کرد.

در هر کدام از این جملات لازم است تا میزان اطلاعات اندازه گیری شود. به نظر می‌رسد که میزان اطلاعات موجود در هر کدام از این جملات رابطه معکوسی با احتمال رخداد آن‌ها دارد. بنابراین ابتدا تابعی بر حسب احتمال رخداد وقایع تعریف می‌شود که میزان اطلاعات موجود را نشان دهد. این تابع به صورت $\log_\alpha \frac{1}{P_i}$ تعریف می‌شود. با توجه به مقدار α، واحد اطلاعات مشخص می‌گردد که برای $\alpha = 2$ واحد آن ***bit***، برای $\alpha = 10$ واحد آن ***decit*** و برای $\alpha = e$ واحد آن برابر ***nat*** می‌باشد.

در گام بعدی نیاز است تا حداکثر میزان ارسال اطلاعات از یک کانال آغشته به نویز در احتمال خطای مورد نظر اندازه گیری شود. در حالت کلی برای یک منبع تولید اطلاعات، میزان متوسط اطلاعات (آنتروپی یا عدم قطعیت) از رابطه زیر محاسبه می‌گردد:

$$H(M) = \sum_{i=1}^{M} P_i \log_2 \left(\frac{1}{P_i}\right) \qquad (7)$$

قانون اول شانون بیان می‌دارد که حداقل به اندازه آنتروپی علائم مخابراتی به طور متوسط بیت برای کد کردن علائم مخابراتی لازم است که در نتیجه آن می‌توان استنباط نمود که بهتر است طول کدهای تخصیصی متغیر باشد. به عبارتی بیان می‌دارد که به سمبلی که احتمال ارسال آن بیشتر است، تعداد بیت‌های کمتری اختصاص داده شود. بر این مبنا، تعدادی از روش‌های کدگذاری منبع توسعه یافته‌اند که یکی از آن‌ها روش شانون-فانو می‌باشد.

سمبل‌های *'a'*، *'b'*، *'c'*، *'d'*، *'e'*و *'f'* را در نظر بگیرید. برای هر کدام از آنها به ترتیب احتمال‌های رخداد $\frac{1}{3}$، $\frac{1}{4}$، $\frac{1}{5}$، $\frac{1}{6}$، $\frac{1}{30}$ و $\frac{1}{60}$ را فرض کنید. اگر بخواهیم از کلمات کد به طول یکسان برای کدگذاری هر کدام از این سمبل‌ها استفاده کنیم، حداقل سه بیت نیاز خواهیم داشت تا بتوانیم همه این سمبل‌ها را بسازیم. اما قابل اثبات است که با استفاده از روش‌های دیگر کدگذاری مثل شانون-فانو می‌توان از کدهایی با طول متفاوت

برای سمبل‌ها به صورت بهینه‌تر کدها را بین سمبل‌ها توزیع کرد و به طور متوسط بیت‌های کمتری استفاده نمود.

در روش شانون-فانو، جدولی تهیه می‌شود که در ستون اول آن، سمبل‌های مورد استفاده نوشته می‌شوند. باید توجه کرد که این سمبل‌ها به ترتیب از احتمال رخداد زیاد به احتمال رخداد کمتر نوشته می‌شوند. در ستون دوم این احتمالات نوشته می‌شود. در ستون سوم مقدار n_i یا تعداد بیت‌های لازم برای کد کردن سمبل مد نظر نوشته می‌شود. در ستون بعدی مجموع احتمالات سمبل‌های قبلی نوشته می‌شود. ستون بعدی محل نوشتن بسط باینری مجموع احتمالات بدست آمده می‌باشد. در نهایت به میزان n_i بیت از بسط باینری هر کدام از سمبل‌ها برداشته شده و عنوان کد مدنظر به کار می‌رود که در ستون آخر نوشته می‌شود. در جدول ۱، این مساله حل شده است.

جدول ۱. کدگذاری به روش شانون-فانو

پیام‌ها	احتمالات	$n_i = \left\lceil \log_2 \frac{1}{p_i} \right\rceil$	$F_i = \sum_{n=1}^{i-1} p_i$	بسط باینری F_i	کد مربوطه
$M_1 = 'a'$	$\frac{1}{3}$	$n_1 = 2$	$F_1 = 0$	*0.00...*	*00*
$M_2 = 'b'$	$\frac{1}{4}$	$n_2 = 2$	$F_2 = \frac{1}{3}$	*0.01...*	*01*
$M_3 = 'c'$	$\frac{1}{5}$	$n_3 = 3$	$F_3 = \frac{7}{12}$	*0.100...*	*100*
$M_4 = 'd'$	$\frac{1}{6}$	$n_4 = 3$	$F_4 = \frac{47}{60}$	*0.110...*	*110*
$M_5 = 'e'$	$\frac{1}{30}$	$n_5 = 5$	$F_5 = \frac{57}{60}$	*0.11110...*	*11110*
$M_6 = 'f'$	$\frac{1}{60}$	$n_6 = 6$	$F_6 = \frac{59}{60}$	*0.111110...*	*111110*

بسط باینری یک عدد به صورت زیر نوشته می‌شود:

$$\frac{1}{3} \div \frac{1}{2} = 0\frac{2}{3} \div \frac{1}{2} = 1\frac{1}{3} \div \frac{1}{2} = 0\frac{2}{3}, \ldots \Rightarrow \frac{1}{3} = 0.\overline{01}$$

انجام آزمایش

۱. برنامه‌ای بنویسید که تعدادی سمبل با احتمالات رخداد متناظرشان را دریافت کرده و کد مربوطه آن‌ها را با استفاده از روش شانون-فانو تولید کند.

راهنمایی: می‌توان از یک حلقه *for* برای انتخاب هرکدام از سمبل‌ها جهت تولید کد متناظر با آن استفاده کرد. ابتدا سمبل‌ها را به ترتیب از احتمال بیشتر به احتمال کمتر مرتب نمایید. سپس برای هرکدام از سمبل‌ها، مقدار عبارت n_i را بدست آورید. علامت $\lceil \cdot \rceil$ به معنای جزصحیح رو به بالا می‌باشد که مقدار آن برابر است با $\lceil \cdot \rceil + 1$. در این رابطه مقدار P_i همان احتمال متناظر با سمبل i ام می‌باشد. همچنین مقدار F_i را بدست آورید. سپس بسط باینری F_i را نوشته و به تعداد n_i بیت از آن را بردارید تا کد مربوط به آن سمبل بدست بیاید.

۲. یک رشته شامل ۱۰۰۰۰ سمبل تصادفی (از بین سمبل‌های داده شده) و با احتمالات داده شده برای هر کدام تولید نمایید. با استفاده از کدگذاری با طول یکسان برای سمبل‌ها، کلمه کد کل این رشته را بدست آورده و طول آن را مشخص کنید.

راهنمایی: از یک حلقه *for* استفاده کنید.

۳. همان رشته بخش دو را این بار با استفاده از کلمات کد بدست آمده در بخش یک کدگذاری نمایید. طول کلمه کد کل این رشته را بدست آورده و با بخش دو مقایسه نمایید.

راهنمایی: انتظار می‌رود تعداد بیت لازم برای کدگذاری با استفاده از روش شانون-فانو کمتر از تعداد بیت لازم برای کدگذاری با طول یکسان باشد.

۷. آزمایش پنجم: مفهوم روش مونت کارلو و نتایج آن در شبیه سازی

مقدمه

شبیه سازی مونت کارلو در واقع روشی برای بدست آوردن نتایج به صورت simulation در برابر analytical می‌باشد. در این نوع شبیه سازی، بجای محاسبه احتمال خطا با استفاده از روابط و فرمول‌ها، یک سیستم طراحی می‌شود و در آن تمامی موارد لازم که در عمل وجود دارند، لحاظ می‌گردد. سپس روش مورد بحث در این سیستم مورد آزمایش قرار می‌گیرد. طبیعی است اگر روابط و فرمول‌ها درست نوشته شده باشند و شبیه سازی نیز به درستی انجام شود، این دو روش به نتایج یکسانی ختم می‌شوند. برای استفاده از روش مونت کارلو در شبیه سازی یک سیستم مخابراتی، آزمایش زیر طراحی شده است.

یک سیستم مخابرات دیجیتالی طراحی کنید که در آن از مدولاسیون ***PSK*** استفاده شده است. در این سیستم، در هر بار ارسال، یکی از بیت‌های صفر یا یک با احتمال برابر برای ارسال انتخاب می‌شوند. سپس شکل موج سینوسی برای بیت صفر و کسینوس برای بیت یک مورد استفاده قرار می‌گیرند. سیگنال تولید شده از کانال ***AWGN*** عبور داده می‌شود. این کانال می‌بایست در شبیه‌سازی در نظر گرفته شود. به عبارت دیگر، سیگنال ارسالی با یک مقدار نویز جمع بسته می‌شود. سپس در گیرنده سیگنال دریافتی (سیگنال ارسالی بعلاوه نویز) توسط آشکار ساز بهینه آشکارسازی شده و مقدار بیت صفر یا یک بدست می‌آید. در صورتی که مقدار بدست آمده برابر با مقدار ارسالی باشد (بیت صفر ارسال شود و بیت صفر دریافت گردد یا اینکه بیت یک ارسال شود و بیت یک دریافت گردد)، یک واحد به تشخیص‌های درست اضافه گردد و در غیر این صورت، یک واحد به تشخیص‌های غلط افزوده شود. این کار را به تعداد دفعات زیاد (مثلا پنجاه هزار بار) انجام داده و در نهایت میزان تشخیص‌های غلط به نسبت کل دفعات ارسال اندازه گیری کنید. عدد بدست آمده را با مقدار احتمال خطای محاسبه شده در درس مخابرات دیجیتال که به صورت ***analytical*** بدست آمده بود، مقایسه کنید.

انجام آزمایش

۱. دو شکل موج s_0 و s_1 را با مدت زمان ۱ ثانیه به صورت زیر تعریف کرده و نمایش دهید (مدولاسیون ***PSK***). فاصله نمونه‌های سیگنال را ۰٫۰۱ ثانیه در نظر بگیرید (نرخ نمونه برداری برابر ۱۰۰ نمونه در ثانیه).

$$s_0 = \sin(4\pi t)$$
$$s_1 = \cos(4\pi t)$$

راهنمایی: t را یک بردار خطی از زمان ۰ تا ۱ با فاصله نقاط ۰٫۰۱ با استفاده از دستور ***np.arange*** (یا ***linespace*** در متلب) تعریف کرده و مقدار سیگنال را در تمامی نقاط t بدست بیاورید.

۲. دو متغیر با مقدار اولیه صفر برای شمارش تعداد تشخیص‌های درست و تعداد تشخیص‌های نادرست ایجاد کنید.

۳. در یک حلقه تکرار به ازای تعداد تکرارهای مونت کارلو (در اینجا ۵۰۰۰۰ تکرار کافی است) ایجاد کرده و به ازای اجرای هر بار حلقه، کارهای زیر را انجام دهید.

- یکی از نمادهای ۰ یا ۱ را با احتمال برابر (۰,۵) انتخاب کنید. با توجه به اینکه کدام یک از نمادها انتخاب شده باشد، سیگنال متناظر با آن را (s_0 یا s_1) تولید نمایید.

راهنمایی: برای تولید سیگنال تصادفی با توزیع یکنواخت می‌توانید از ماژول ***random.uniform*** (یا ***randi*** در متلب) استفاده کنید.

- سیگنال تولید شده را ضربدر میزان تضعیف کانال (α) نمایید. مقدار تضعیف کانال را به ترتیب برابر با حالت‌های $\alpha = 0.01,0.1,1$ در نظر بگیرید.
- در این مرحله به سیگنال تضعیف شده نویز سفید گوسی اضافه نمایید. این مرحله را با $SNR = -10$ انجام دهید.
- در این مرحله سیگنال نویزی تضعیف شده در سمت گیرنده در اختیار می‌باشد. می‌بایست با استفاده از گیرنده همبستگی عمل آشکار سازی انجام شود. برای این منظور، ابتدا سیگنال دریافتی (که ۱۰۰ نمونه از آن برداشته شده است) را به هر کدام از شکل موج‌های s_0 و s_1 (که هر کدام دارای ۱۰۰ نمونه می‌باشند) ضرب نموده و سپس میانگین نمونه‌های آن‌ها را محاسبه نمایید. برای هر کدام از شکل موج‌ها عدد بزرگتری بدست بیاید، آن شکل موج آشکارسازی می‌شود.
- بررسی کنید که آیا سیگنال آشکارسازی شده برابر با سیگنال ارسالی می‌باشد یا خیر. در صورت آشکارسازی صحیح، تعداد تشخیص‌های درست را یک واحد افزایش بدهید و در غیر این صورت، تعداد تشخیص‌های نادرست را یک واحد افزایش دهید.

۴. نسبت تعداد تشخیص‌های غلط را به تعداد کل آزمایش‌های مونت کارلو بدست بیاورید. این عدد در واقع احتمال خطای سیستم طراحی شده می‌باشد. این عدد را با احتمال خطایی که به صورت analytical و با استفاده از روابط ریاضی بدست آورده‌اید مقایسه کنید.

۸. آزمایش ششم: استفاده از یادگیری ماشین برای تشخیص سیگنال ارسالی

مقدمه

امروزه دامنه هوش مصنوعی و یادگیری ماشین بسیار گسترده شده و تقریبا زمینه‌ای نمانده است که تلفیقی از هوش مصنوعی را تجربه نکرده باشد. مخابرات نیز از این قاعده مستثنی نبوده و دستخوش تغییرات زیادی با ورود یادگیری ماشین به این عرصه شده است. یادگیری ماشین در بخش‌های زیادی از یک سیستم مخابراتی برای اهداف مختلفی می‌تواند مورد استفاده قرار گیرد. مثلا از روش‌های یادگیری با نظارت می‌توان برای حل مسائل طبقه‌بندی استفاده نمود یا از روش‌های یادگیری تقویتی برای مسائل کنترلی بهره جست.

یکی از مسائل طبقه‌بندی در مخابرات، آشکارسازی سیگنال ارسالی می‌باشد. در سمت فرستنده یک سیگنال از بین شکل موج‌های موجود (تعیین شده توسط مدولاسیون) انتخاب گشته و ارسال می‌گردد. این سیگنال در کانال دستخوش تغییراتی شده و در نهایت یک سیگنال در گیرنده دریافت می‌گردد. مساله آشکارسازی عبارت است از اینکه از روی سیگنال دریافتی تعیین گردد که کدام یک از شکل موج‌ها ارسال شده است. روش‌های مختلفی برای حل این مساله ارائه شده‌اند که آشکار سازی فیلتر منطبق یکی از آن‌ها است. این روش که بهینه‌ترین روش در بین همه الگوریتم‌ها می‌باشد زمانی قابل استفاده می‌باشد که مدل کانال مشخص باشد. در برخی موارد این فرض نامعقول بوده یا به صورت دقیق قابل مدلسازی نمی‌باشد. در چنین مواقعی فیلتر منطبق بهینه‌ترین روش نخواهد بود. در چنین مواقعی ممکن است روش‌های یادگیری ماشین نتایج بهتری بدهند.

یادگیری ماشین دارای دو فاز می‌باشد، فاز آموزش (*train*) و فاز آزمایش (*test*). در فاز آموزش، تعدادی داده وجود دارد که برای آن‌ها لیبل مشخص شده است. یعنی تعدادی سیگنال تولید گشته و ارسال شده‌اند و در سمت گیرنده دریافت شده‌اند. برای این سیگنال‌های دریافتی مشخص است که کدام سیگنال ارسال شده بود. از این داده استفاده می‌شود تا سیستم آموزش داده شود. به عبارتی پارامترهای سیستم طوری تغییر می‌یابند و تنظیم می‌شوند تا سیستم بتواند نگاشت بین ورودی‌های مدنظرش را به لیبل داده‌ها را یادبگیرد. اما در فاز آزمایش، داده‌های بدون لیبل وجود دارند و سیستم به هرکدام از آن‌ها یک لیبل می‌زند و مشخص می‌کند متعلق به کدام کلاس هستند. از روی تعداد تشخیص‌های درست می‌توان عملکرد سیستم را آنالیز کرد.

یکی از زیرمجموعه‌های یادگیری ماشین و ساختارهایی که برای آن وجود دارد، شبکه‌های عصبی مصنوعی می‌باشند. مبنای عملکرد شبکه‌های عصبی مصنوعی از شبکه‌های عصبی طبیعی (مغز انسان) الهام گرفته شده است. کوچکترین عنصر وجودی شبکه‌های عصبی نرون‌ها می‌باشند. هر نرون تعدادی ورودی دارد که با اعمال برخی عملیات بر روی آن‌ها یک خروجی تولید می‌کند. ساختارهای متنوعی می‌توان با این نرون‌ها ایجاد کرد. یکی از ساختارها ساختار *MLP* می‌باشد که در آن تعدادی لایه نرون درنظر گرفته شده است که خروجی‌های

هر لایه به عنوان ورودی لایه بعدی مورد استفاده قرار می‌گیرد. یک تابع هدف برای شبکه تعیین شده و با تنظیم وزن‌های شبکه میزان این تابع هدف کمینه می‌گردد. پیاده سازی یک شبکه عصبی برای مساله آشکارسازی با دیتاست در اختیار در این آزمایش مورد بررسی قرار گرفته است.

انجام آزمایش

۱. در این آزمایش دیتاست مدولاسیون *32-QAM* در اختیارتان قرار می‌گیرد. این دیتاست حاوی ۳۶۰۰۰ تصویر با ابعاد ۴۰*x*۴۰ پیکسل در ۳۲ کلاس مختلف می‌باشد. تعداد ۶۴۰۰ داده (۲۰۰ داده از هر کلاس) را به صورت تصادفی انتخاب کرده و به عنوان داده‌ی آزمایش در نظر بگیرید. بقیه داده‌ها را برای آموزش سیستم استفاده کنید.

۲. یک تبدیل برای تبدیل تصاویر رنگی فراخوانی شده به تصاویر خاکستری و همچنین تبدیل آن‌ها به فرمت تنسور کتابخانه *pytorch* تعریف کنید. سپس آدرس پوشه داده‌های آموزش و آزمایش را مشخص نموده و تبدیل مد نظر را روی آن‌ها اعمال نمایید.

راهنمایی: برای تعریف تبدیل می‌توانید از ماژول *transforms* در کتابخانه *torchvision* استفاده نمایید. همچنین برای مشخص کردن فولدر دیتاست در برنامه، می‌توانید از ماژول *ImageFolder* در *datasets* از کتابخانه *torchvision* استفاده نمایید.

۳. برای اینکه داده‌ها قابل اعمال به شبکه باشند، ابتدا باید به صورت بچ آماده شوند (یعنی با اعمال ۳۲ داده، یکبار آپدیت وزن‌ها انجام شود). داده‌ها را به فرمت بچ‌های ۳۲ تایی آماده کنید.

راهنمایی: از ماژول *DataLoader* از کتابخانه *torch.utils.data* می‌توانید استفاده کنید.

۴. یک مدل *sequential* با مشخصات ارائه شده در شکل ۵ ایجاد نمایید.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1               [-1, 1, 256]         409,856
              ReLU-2               [-1, 1, 256]               0
            Linear-3                [-1, 1, 32]           8,224
================================================================
Total params: 418,080
Trainable params: 418,080
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 0.00
Params size (MB): 1.59
Estimated Total Size (MB): 1.61
----------------------------------------------------------------
```

شکل ۵. مشخصات مدل پیشنهادی

توضیحات: در این مدل، یک لایه پنهان با ۲۵۶ نرون در نظر گرفته شده است. ابعاد ورودی برابر با ۱۶۰۰ بعلاوه ۱ (بایاس) می‌باشد. بنابراین تعداد پارامترهای (وزن‌ها) این لایه برابر با $1601 * 256 = 409856$ می‌باشد. همچنین از تابع *ReLU* به عنوان تابع فعالساز نرون‌های این لایه استفاده شده است. برای لایه خروجی نیز ۳۲ نرون در نظر گرفته شده است (هر نرون برای یکی از کلاس‌ها). بنابراین تعداد پارامترهای این لایه برابر با $257 * 32 = 8224$ می‌باشد.

راهنمایی: برای ساخت این مدل از ماژول ***Sequential*** در کتابخانه ***torch.nn*** می‌توانید استفاده کنید.

۵. تابع ***loss*** را به صورت ***Cross Entropy Loss*** تعریف کنید.

راهنمایی: می‌توانید از تابع ماژول ***nn.CrossEntropyLoss*** استفاده نمایید.

۶. برای بهینه‌سازی از الگوریتم ***Adam*** استفاده نمایید.

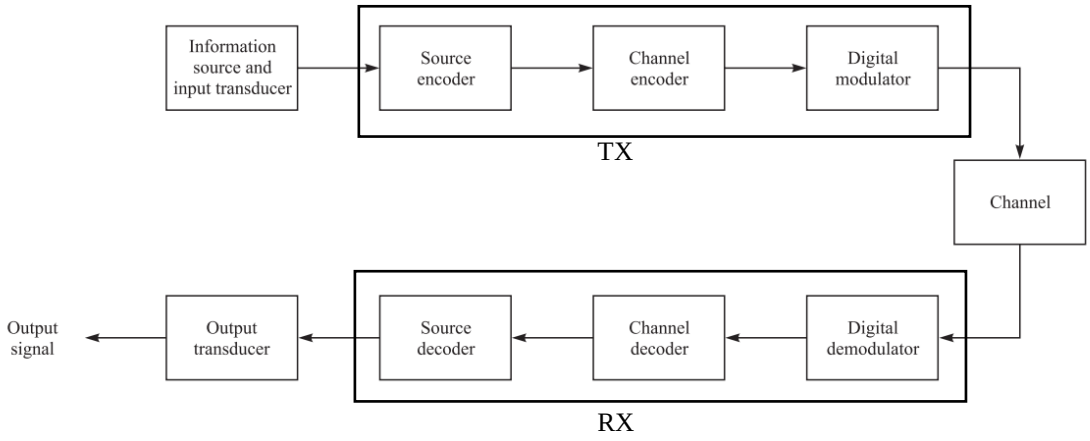
راهنمایی: می‌توانید از تابع ماژول ***torch.optim.Adam*** استفاده نمایید.

۷. برای آموزش شبکه، تعداد ایپوک‌های کل را برابر ۴۰ در نظر بگیرید. در یک حلقه ***for*** ابتدا داده‌ها را فرا بخوانید، سپس خروجی مدل را به ازای داده‌های ورودی بدست آورده و میزان تابع هزینه را محاسبه نمایید. در نهایت وزن‌ها را طوری تغییر دهید تا میزان تابع هزینه تعریف شده به مقدار کمینه همگرا شود. یک حلقه ***for*** دیگر برای داده‌های آزمایش در نظر بگیرید ولی در آن مقدار وزن‌ها را تغییر ندهید و فقط خروجی بدست آمده را با لیبل صحیح داده‌ها مقایسه کرده و دقت طبقه‌بندی را به ازای هر ایپوک نمایش دهید. در نهایت پس از اتمام آموزش شبکه، از داده‌های آزمایش برای بدست آوردن دقت طبقه‌بندی نهایی استفاده کنید.

۹. آزمایش هفتم: طراحی یک سیستم کامل

مقدمه و انجام آزمایش

همانطوری که در شکل زیر مشاهده می‌شود، یک سیستم مخابراتی شامل سه بخش اصلی فرستنده، گیرنده و کانال است. بخش‌های فرستنده و گیرنده دوگان هم بوده و هر کدام شامل سه بخش اصلی کدینگ منبع، کدینگ کانال و مدولاتور هستند. در این آزمایش هدف ترکیب پنج آزمایش اول و محاسبه‌ی احتمال خطا سیستم به صورت مونت-کارلو است.



۱. مشخصات سیستم مورد شبیه سازی به شرح زیر است:

- مدولاسیون : *QAM*
- کدینگ کانال: قالبی خطی ۴و۷
- کدینگ منبع: شانون-فائو
- کانال: *AWGN*
- نسبت سیگنال به نویز: ۲۰، صفر و ۲۰- دسیبل
- تعداد بیت: ۵۰ هزار
- نوع مدولاتور: همبستگی

۲. خروجی‌های لازم:

- رسم منظومه ورودی و خروجی
- محاسبه احتمال خطا در سه *SNR* مختلف، با و بدون کدینگ کانال
- مقایسه احتمال خطا با روابط تحلیلی

۳. ورودی‌های لازم:

- حروف الفبای انگلیسی به تعداد مورد نیاز