بسمه تعالی

گزارش پروژه پایانی درس سیستم های مخابراتی

استاد : دکتر حمید بهروزی

عنوان پروژه : مدولاسیون دیجیتال

نویسندگان گزارش :

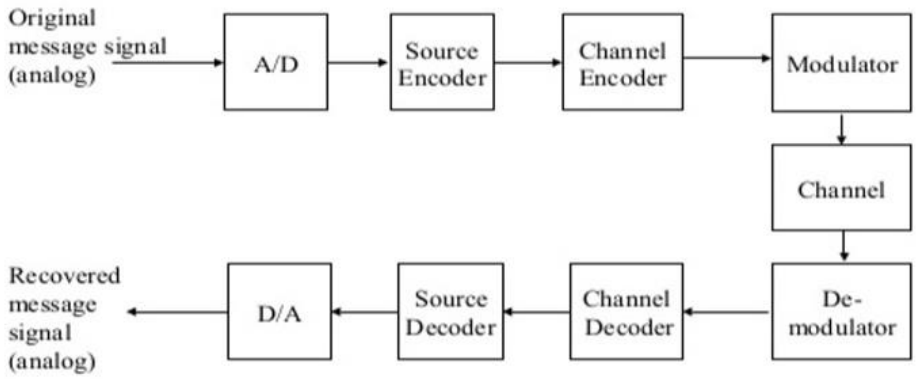
امرحسین رستمی (۹۶۱۰۱۶۳۵)

آروین قویدل (۹۶۱۱۰۵۸۳)

- **معرفی و توضیحات اولیه پروژه :**

هدف در این پروژه، انتقال تصویر، با استفاده از مدولاسیون دیجیتال است.

ساختار کلی پروژه در شکل زیر آورده شده است :



تمرکز اصلی پروژه روی بلوک های Source Decoder و Source Encoder ، و بلوک های Modulator و Demodulator هستند.

کانال در کل به عنوان یک فیلتر میانگذر با نویز AWGN است و همانطور که در دستور کار آورده شده است، بلوک های A/D و D/A و دیکودر و اینکودر های کانال نیازی به پیاده سازی ندارند.

ابتدا توضیحات نظری مربوط به هر بخش را ارائه میکنیم و سپس نحوه پیاده سازی و نتایج خواسته شده ارائه خواهد شد.

توجه : در بخش های مربوط به محاسبه SNR و نویز، با مراجعه به TA محترم درس (آقای طه عسگری) قرار بر آن شد که بتوانیم دامنه سیگنال ها را در صورت نیاز کم کنیم.

در صورتی که در یکی از بخش ها چنین کرده باشیم آن را اعلام کرده ایم.

- **توضیحات نظری پروژه :**

در فاز اول باید بلوک های دیکودر و اینکودر منبع پیاده سازی شوند.

هدف از این کار کاهش بیت بر سمبل است که سمبل ها در اینجا یک پیکسل grayscale هستند.

در این پروژه روش کدگذاری مطرح شده، کدگذاری شانون ـ فانو است که الگوریتم آن به طور کامل در صورت پروژه مطرح شده است و تنها پیاده سازی آن نیاز است که ما در بخش های بعدی به آن می‌پردازیم.

با کدگذاری به این روش، هر عکس (که مجموعه ۴۰۹۶ پیکسل است) به صورت قطاری از کد های شانون ـ فانو در می‌آید و وارد بلوک مودولاتور می‌شود.

بلوک مودولاتور وظیفه دارد که سیگنال را به باند مناسب ببرد تا در کانال ما (که همانطور که مطرح شد یک فیلتر میان گذر ایده آل است) قابل انتشار باشد (ممکن است در اینجا نویزی به سیگنال اضافه شود که مشکلاتی را به وجود خواهد آورد، در ادامه آنها را شرح می‌دهیم).

دنباله کد های تولید شده به صورت دو به دو (یعنی هر بخش معادل عدد ۱ و ۲ و ۳ و ۴ است) به بلوک مودولاتور داده می‌شوند، بلوک مودولاتور سیگنال QAM زیر را تولید می‌کند :

$$S_m = \begin{cases} A_{mc}\sqrt{\frac{۲}{T_s}}cos(۲\pi f_c t) + A_{ms}\sqrt{\frac{۲}{T_s}}sin(۲\pi f_c t) & ۰ \le t \le T_s \\ ۰ & otherwise \end{cases}$$

$$A_{mc} = \begin{cases} ۱ & m = ۱, ۴ \\ -۱ & m = ۲, ۳ \end{cases}$$

$$A_{ms} = \begin{cases} ۱ & m = ۱, ۲ \\ -۱ & m = ۳, ۴ \end{cases}$$

این سیگنال با مقدار مناسب فرکانس حامل به راحتی در کانال قابل انتقال است و هر آنچه نیاز است در بر دارد.

این سیگنال از کانال عبور می‌کند، در حالت ایده آل اگر فرکانس حامل درست انتخاب شده باشد، سیگنال بدون اعوجاج عبور می‌کند، اما ممکن است که در حین عبور به سیگنال نویز اضافه شود.

در صورتی که به سیگنال نویز اضافه شود باید آشکار سازی درستی روی آن انجام شود.

همانطور که در درس مطرح شده است، در صورتی که توزیع نویز گاوسی باشد، یک روش تصمیم گیری بهینه این است که همبستگی هر یک از سمبول ها، یعنی هر یک از S_m ها را با سیگنال ورودی، مثلا y مقایسه کنیم :

$$R_{yS_i}(0) \gtrless_j^i R_{yS_j}(0) \qquad for\ i,j = 1,2,3,4$$

که البته شایان ذکر است که چون فقط همبستگی در نقطه صفر نیاز است، کافی است ضرب داخلی بردار ها را مقایسه کنیم :

$$< y, S_i > \gtrless_j^i < y, S_j > \qquad for\ i,j = 1,2,3,4$$

البته یادآوری میشود که در حالت عادی باید ضریب همسبتگی مقایسه شود (در واقع باید همسبتگی ها به انرژی دو سیگنال نرمالیزه شوند)، در اینجا این کار موضوعیت ندارد، چرا که انرژی تمام سمبل ها یکی است.

اثبات این مورد در درس انجام شده است و ما آن را در اینجا نمی‌آوریم.

همچنین یک روش تصمیم گیری دیگر نیز ممکن است:

سیگنال نهایی کانال با در نظر گرفتن نمایش گابور نویز مانند زیر است :

$$y = \left(A_{mc} \sqrt{\frac{2}{T_s}} + n_c(t) \right) \cos(2\pi f_c t) + \left(A_{ms} \sqrt{\frac{2}{T_s}} + n_s(t) \right) \sin(2\pi f_c t)$$

اگر دامنه نویز به اندازه کافی کوچک باشد، میتوان تبدیل هیلبرت سیگنال را گرفت و نوشت :

$$A_{mc} \sqrt{\frac{2}{T_s}} + n_c(t) \approx y \cos(2\pi f_c t) + \hat{y} \sin(2\pi f_c t)$$

$$A_{ms} \sqrt{\frac{2}{T_s}} + n_c(t) \approx y \sin(2\pi f_c t) - \hat{y} \cos(2\pi f_c t)$$

در اینجا به طور خاص فقط علامت A_{mi} ها برای ما مهم است و لذا میتوان نوشت :

$$A_{mc} = sign\ (y \cos(2\pi f_c t) + \hat{y} \sin(2\pi f_c t))$$
$$A_{ms} = sign\ (y \sin(2\pi f_c t) - \hat{y} \cos(2\pi f_c t))$$

این روش به شدت به نویز مقاوم است ولی یک مشکل اساسی دارد...

همانطور که عرض شد در حالت ایده آل تاثیر کانال بر سیگنال ارسال شده هیچ و یا بسیار کم است، ولی از آنجا که سیگنال ها ارسال شده (قطار سمبل هایی که وارد کانال می‌شوند) سینوسی های ناپیوسته ای هستند، در حوزه فرکانس دیگر تابع ضربه های کاملی نیستند و لذا پس از عبور دچار اعوجاج خواهند شد، در این حالت ممکن است تبدیل هیلبرت به شدت متفاوت شود، این اثر با افزایش پهنای باند کانال قابل جبران است ولی ممکن است این روش برای پهنای باند داده شده جوابگو نباشد، در این صورت ناچار به استفاده از روش اول هستیم.

حال برای رسم Constellation های سیگنال، باید بتوانیم نزدیکترین ضرایبی که به سمبول قابل نسبت باشد را پیدا کنیم، در واقع فرض میکنیم که :

$$S = \widetilde{A_{cm}} \sqrt{\frac{2}{T_s}} \cos(2\pi f_c t) + \widetilde{A_{sm}} \sqrt{\frac{2}{T_s}} \sin(2\pi f_c t)$$

یک روش راحت برای گرفتن ضرایب این است که با سینوس و کسینوس همبستگی را محاسبه کنیم :

$$\widetilde{A_{cm}} = \sqrt{\frac{T_S}{2}}\ R(S, \cos(2\pi f_c t))$$

$$\widetilde{A_{sm}} = \sqrt{\frac{T_S}{2}}\, R(S, \sin(2\pi f_c t))$$

سپس این زوج ضرایب را میتوان به ازای هر پیکسل در صفحه رسم کرد تا Constellation حاصل شود.

لذا در این مرحله، سیگنال متناظر با هر یک از گروه های دو بیتی مدوله شده باید به روش بهینه آشکار شده باشد (حتی با حضور نویز).

اکنون ما می‌دانیم که هر سمبل چه بوده و میتوانیم دنباله های صفر و یک را بازسازی کنیم.

از آنجا که کد شانون ـ فانو پیشوندی است، پس از ساخت سیگنال، کافیست از ابتدا سیگنال به ته سیگنال حرکت کنیم، هر لحظه که تشخیص داده شد که یکی از کد های پیشفرض ساخته شده، بخش خوانده شده را جدا میکنیم و پیکسل معادل با کد را ارسال می‌کنیم، این روش درست است چرا که هرگز ممکن نیست کدی جز این مورد در پیشوند کد دیگری بوده باشد.

البته یک مورد می‌تواند رخ دهد...

به علت وجود نویز ممکن است که تصمیم‌گیری اشتباهی انجام شده باشد و لذا خروجی های تولید شده، عملا دیگر کد های شانون ـ فانو نباشند و یا خاصیت پیشوندی کد به هم بریزد.

در این حالت اگر با روش قبل به آشکار سازی ادامه دهیم ممکن است به نقطه ای برسیم که سیگنال دیده شده با هیچ یک از کد ها مطابقت نداشته باشد و نتوانیم تشخیص دهیم پیکسل مورد نظر چه بوده...

راه کار دچار اغتشاش شدن بیت ها و از بین رفتن احتمالی ساختار شانونی رشته کد ها:

همانطور که می دانید رشته کد های شانون خصلت پیشوندی دارند،بدین صورت که هیچ رشته کد عی داخل رشته کد دیگر وجود ندارد و در گیرنده کافیست ما شروع کنیم به خواندن رشته ها و به محض رسیدن به اولین حالت مطابق با یکی از اعضای دیکشنری شانون موجود،جای قطعه کد شانون دیده شده معادل سمبل آن را قرار بدهیم(demodulation) و از بیت بعدی مجدد همین سناریو رو تکرار کنیم.اما مشکل کار وقتی نویز رخ می دهد این است که این خطا ممکن است باعث شود رشته ی کدی که به گیرنده می رسد مطابق با عضوی از دیکشنری ما نباشد.در این حالت من با صحبت و مشورت تی ای(آقای عسگری) در حالت نویزی تدبیری که در ادامه مطرح خواهم کرد را درنظرگرفتم.

حال که سیگنال های مورد نیاز آشکار شده اند، میتوان پیکسل های مورد نیاز را از روی آنها استخراج کرد و تصویر را باز تولید کرد.

حال در صورتی که فرستنده و گیرنده همزمان نباشند، و یک اختلاف فاز مثل ϕ داشته باشند:

$$D = R(A_{ci}\cos(2\pi f_c t + \phi) + A_{mi}\sin(2\pi f_c t + \phi), S) =$$

$$(A_{ci}\cos(\phi) + A_{mi}\sin(\phi))\, R(\cos(2\pi f_c t), S) + (-A_{ci}\sin(\phi) + A_{mi}\cos(\phi))\, R(\sin(2\pi f_c t), S)$$

حال فرض کنید که سیگنال ورودی معادل سمبول j باشد :

$$S = A_{cj}\cos(2\pi f_c t) + A_{sj}\sin(2\pi f_c t)$$

در این صورت داریم :

$$D = (A_{ci}\cos(\phi) + A_{si}\sin(\phi))\frac{A_{cj}}{2} + (-A_{ci}\sin(\phi) + A_{mi}\cos(\phi))\frac{A_{mj}}{2}$$

حال باید مقداری از i را انتخاب کنیم که عبارت بالا را بیشینه کند، دو حالت در نظر می‌گیریم :

- $\phi = \frac{\pi}{2}$:

در این حالت داریم :

$$D = \frac{A_{mi}A_{cj}}{2} - \frac{A_{ci}A_{mj}}{2}$$

که حالت بهینه آن هست :

$$A_{mi} = A_{cj}\,, \qquad A_{mj} = -A_{ci}$$

- $\phi = \frac{\pi}{4}$:

در این حالت :

$$D = \frac{1}{2\sqrt{2}}\left(A_{ci}A_{cj} + A_{mi}A_{cj} - A_{mj}A_{ci} + A_{mi}A_{mj}\right)$$

در این حالت ممکن است تصمیم گیری بهینه واضح نباشد، مثلا اگر داشته باشیم :

$$A_{cj} = A_{mj} = 1 \Longrightarrow (A_{ci} = 1, A_{mi} = 1) \text{ یا } (A_{ci} = -1, A_{mi} = 1)$$

- حال در حالت کلی میتوان نوشت :

$$D(\phi) = \frac{\sqrt{2}}{2}\left(A_{cj}\sin\left(\phi + \tan^{-1}\left(\frac{A_{ci}}{A_{si}}\right)\right) + A_{sj}\sin\left(\phi + \tan^{-1}\left(\frac{A_{si}}{-A_{ci}}\right)\right)\right)$$

حال اگر تعریف کنیم :

$$\alpha = \tan^{-1}\left(\frac{A_{ci}}{A_{si}}\right)$$

$$D(\phi,\alpha) = \frac{\sqrt{2}}{2}\left(A_{cj}\sin(\phi+\alpha) + A_{sj}\sin\left(\phi + \pi - \tan^{-1}\left(\frac{A_{si}}{A_{ci}}\right)\right)\right) =$$

$$D(\phi,\alpha) = \frac{\sqrt{2}}{2}\left(A_{cj}\sin(\phi+\alpha) + A_{sj}\sin\left(\phi + \frac{\pi}{2} + \alpha\right)\right) = \frac{\sqrt{2}}{2}\left(A_{cj}\sin(\phi+\alpha) + A_{sj}\cos(\phi+\alpha)\right)$$

$$D(\phi,\alpha) = \sin\left(\phi + \alpha + \tan^{-1}\left(\frac{A_{sj}}{A_{cj}}\right)\right)$$

حال برای سمبول شماره j و i داریم :

$$\tan^{-1}\left(\frac{A_{sj}}{A_{cj}}\right) = (2j-1)\frac{\pi}{4}$$

$$\alpha = \frac{\pi}{2} - (2i-1)\frac{\pi}{4}$$

لذا:

$$D(\phi,i,j) = \sin\left(\phi + \frac{\pi}{2} + (2j-2i)\frac{\pi}{4}\right) = \cos\left(\phi + \frac{\pi}{2}(j-i)\right)$$

لذا با افزایش اختلاف فاز، به طور ناگهانی بعضی از نقطه های خروجی در Constellation ها جا به جا می‌شوند و لذا پیکسل ها خراب می‌شوند.

- **پیاده سازی پروژه :**

در این بخش به توضیح روش های استفاده شده در پیاده سازی بلوک و همچنین نتایج خواسته شده در هر بخش می‌پردازیم.

در پروژه خواسته شده که برای هر بلوک تابعی جداگانه پیاده سازی شود، ما در هر یک از این بخش ها توابع را می آوریم و در صورت نیاز مثال ارائه می‌دهیم.

در فاز اول پیاده سازی بلوک ها به طور جداگانه بررسی می‌شود و درفاز دوم اتصال بلوک ها و ساخت پروژه سیستم نهایی و تست آن را انجام می‌دهیم.

- **پیاده سازی فاز اول :**
- **بلوک های Source Encoder و Source Decoder :**

در این بخش هدف پیاده سازی بلوک های اینکودر و دیکودر های منبع است.

همانطورکه در بخش پیشین عرض شد، بلوک اینکودر وظیفه دارد که عکس را پیکسل به پیکسل دریافت کند و به صورت کد شانون ـ فانو برگرداند، در مقابل بلوک دیکودر ای فرآیند را برعکس میکند:

توضیح نحوه ی پیاده سازی الگوریتم شانون:

1 ـ درابتدا که ماتریس احتمالات گرفته می شود در نظر گرفته شده است که ممکن است ترتیب ورودی داده شده به صورت اکیدانزولی(از سمبل با احتمال بیشتر به سمبل با احتمال کمتر) مرتب نشده است و لذا ابتدا ماتریس احتمالات وارد شده را مرتب سازی می کنیم به ترتیب(اکیدانزولی) از بیشترین احتمال به کمترین احتمال و سپس به مرحله ی 2 ام می رویم.

2 ـ در این مرحله از سمبل اول(که طبیعتا بیشترین احتمال را بین سمبل ها دارد) شروع کرده و پله پله جلو رفته و مجموع احتمال های موجود رو از ابتدا تا نقطه ی فعلی حساب می کنیم و این مرحله تا آن جایی ادامه می دهیم که مجموع بخش اول در اولین وهله از مجموع باقی بیشتر گردد،از آنجا که هدف الگوریتم تقسیم کم اختلاف ترین دسته بندی موجود است،در این مرحله یک مرحله اندیس فعلی را عقب می بریم و اختلافِ مجموعِ احتمالات دو مجموعه ی حالت قبل را با اختلافِ مجموعِ احتمالات دو مجموعه ی حالت فعلی مقایسه می کنیم،و خب طبیعتا بهترین حالت ممکن انتخاب دسته بندی ایست که کمترین اختلاف را داشته باشد و لذا با بررسی دو حاصل به دست آمده،آن دسته بندی ای را انتخاب می کنیم که کم ترین اختلاف را داشته باشد و به دسته ی با احتمال بیشتر رشته ی "0" افزوده می شود(در ابتدا همه ی رشته ها "" اند)و به دست با احتمال کمتر رشته ی "1" افزوده می گردد.حال به صورت بازگشتی شانون را برای دو بخش باقی مانده فرامی خوانیم و بدین ترتیب برای کل سمبل ها کد شانون ساخته می شود و در خروجی دو ماتریس احتمالات مرتب شده و ماتریس رشته های شانون(که ترتیبا سینک با ماتریس احتمالات مرتب شده است) داده می شود.

پیش از بیان تدبیر درنظرگرفته شده ذکر این نکته مهم است که بنده 2 روش برای دیماژولاسیون در نظرگرفتم:

1ـ**حالت عدم وجود نویز**:در این حالت همانطور که گفته شد شروع کردم از ابتدای رشته ی ورودی به بررسی قطعه رشته ها و مقایسه با دیکشنری موجود تا اینکه نهایتا قطعه قطعه بازیابی بشوند...(توضیح مفصل در بالا)

2ـ**حالت وجود نویز**:در این روش که با تایید و مشورت تی ای محترم(اقای عسگری) انجام شد **بنده در فرستنده بین هر دو سمبل شانون شده flag** قرار دادم،توجه کنید که هرچه بدانیم که **اثر دامنه ی نویز زیاد است لازم است تا بیت های بیشتری** را به flag تخصیص بدهیم **چرا که خود flag نیز دچار نویز** نیز می گردد(چرا که کنار سیگنال اصلی از کانال عبور می کند و تحت تاثیر نویز و اثرات کانال قرار می گیرد)ـحال در گیرنده با جاروب روی قطعه قطعه ی سیگنال و استفاده از **کورلیشن با الگوی flag** به **استخراج flagها** می پردازیم و سپس چون قطعه شانون ها جدا شده اند(بحران قاطی شدن قطعه شانون هارا نداریم)حال به بررسی قطعه کد های موجود می کنیم،اکنون 2 سناریو در پیشه رو داریم:

1- قطعه کد جدا شده توسط flag ها عینا در دیکشنری موجود باشد.
2- قطعه کد در keyset شانون ما وجود ندارد.

اگر حالت اول رخ بدهد که مستقیما معادل سمبل کد شانون را قرار می دهیم اما اگر حالت دوم رخ داد به ترتیب مراحل زیر را انجام می دهیم.

1ـجاروب روی Keyset شانون های موجود.

2ـحساب کردن فاصله ی کد i ام موجود در دیکشنری شانون با رشته کد فعلیِ بینِ دو flag.

- اگر رشته ی فعلی و رشته ی iام شانون طول برابر داشته باشند بروید به قدم ۳ اگرنه بروید به قدم ۱:
  1- ابتدا رشته ی بلند تر را تعیین می کنیم
  2- رشته ی کوتاه را ZeroExtend می کنیم به رشته ای با طول رشته بلند تر
  3- فاصله ی دو رشته را برابر با تعداد بیت متفاوت آن ها درنظر میگیریم.
- رشته کدشانون ای که کمترین فاصله را(طبق گفته ی بالا) از رشته ی فعلی ما داشته باشد را به عنوان رشته ی شانون تخمین زده شده برای رشته ی فعلی درنظر می گیریم.
- معادل کد موجود در دیکشنری شانون برای رشته ی شانون تخمینی را به عنوان demodule شده ی رشته ی بین دو flag درنظر میگیریم و به همین ترتیب تا پایان رشته روند را ادامه می دهیم تا demodulation کامل انجام شود.

در زیر 3 شکل اول عکس در فرمت پیکسل به پیکسل هستند.

عکس اول مقادیر پیکسل های تصویر است، عکس دوم کدگذاری شانون آن است و عکس بعدی، پیکسل های بازیابی شده از کد شانون است، در پایین نیز عکس های بازسازی شده آورده شده اند.

beforeCoding = 64x64 uint8 matrix

193	184	120	103	89	82	154	79	31	129	166	170	167	158	147	142	144	129	107	99	152	161	160	158	158	159	160	165	109	88	99	102	...
192	179	88	105	91	56	126	117	95	131	134	121	108	92	76	61	50	44	46	72	155	161	158	158	158	161	162	165	111	82	94	102	
186	170	58	105	154	125	76	95	106	71	44	36	30	26	25	25	28	32	39	87	157	159	157	156	158	160	163	164	122	89	91	98	
178	145	37	114	105	45	38	32	28	28	28	27	26	30	32	35	34	43	67	138	171	154	155	156	158	163	165	165	127	98	98	99	
170	101	28	119	95	30	33	32	30	29	28	26	26	29	34	40	59	126	127	163	167	151	153	155	158	162	164	161	127	104	106	101	
170	100	98	152	98	31	34	33	33	34	32	26	27	30	31	36	78	136	114	158	145	152	147	149	153	159	160	159	124	105	103	98	
136	144	144	140	108	43	44	45	44	47	36	25	29	30	32	52	103	123	119	141	131	157	154	155	155	159	160	158	121	107	108	105	
61	55	46	86	124	60	51	48	48	48	31	27	28	26	44	78	117	109	124	98	131	161	154	154	158	161	162	162	116	100	107	107	
37	43	48	100	123	49	49	52	52	46	38	53	84	96	62	82	107	99	126	64	133	160	156	157	157	156	148	143	115	95	95	93	
39	46	54	112	124	47	46	55	53	48	79	128	171	169	130	100	103	100	123	48	139	149	138	135	134	143	146	161	140	122	128	141	
⋮																																

codedImage = 64x64 string array

"111101111"	"11001000"	"1000011"	"0001100"	"10111011"	"01111100"	"0010110"	"10111010"	"10101011"	"1001010"	"1011111"	"11001101"	"10100010"
"111011000"	"1010111"	"10000101"	"0011101"	"10001100"	"111000001"	"0000110"	"0011111"	"01110001"	"1001011"	"1010000"	"0111111"	"10001101"
"11011101"	"11001101"	"11001100"	"0011101"	"0010110"	"10010001"	"01001000"	"01110001"	"0010000"	"11100010"	"0100011"	"10000100"	"0011100"
"111101001"	"10000001"	"10011000"	"0100110"	"0011101"	"01101111"	"11000010"	"0111101"	"01101110"	"01101110"	"01101110"	"11011011"	"11001001"
"11001101"	"0100101"	"01101110"	"0001011"	"01110001"	"0011100"	"0110001"	"0111101"	"0011100"	"10100011"	"01101110"	"11001001"	"11001001"
"11001101"	"0001111"	"01001001"	"0111100"	"01001001"	"10101011"	"11000001"	"0110001"	"0110001"	"11000001"	"0111101"	"11001001"	"11011011"
"0111011"	"0101010"	"0101010"	"0110101"	"10001101"	"10110000"	"0100011"	"01101111"	"0100011"	"11010010"	"10000100"	"111100110"	"10100011"
"10101100"	"0110010"	"0001110"	"10011101"	"0110100"	"10110110"	"11001011"	"0101101"	"0101101"	"0101101"	"10101011"	"11011011"	"01101110"
"10011000"	"10110000"	"0101101"	"0001111"	"0000111"	"01110000"	"01110000"	"10100100"	"10100100"	"0001110"	"11000010"	"11000100"	"10001011"
"11001010"	"0001110"	"111001101"	"1000111"	"0110100"	"11010010"	"0001110"	"0110010"	"11000100"	"0101101"	"10111010"	"1001001"	"1010100"
"11001010"	"01110000"	"11010011"	"1000011"	"10010001"	"0100011"	"111001101"	"111011100"	"11100010"	"10001101"	"0110110"	"1010111"	"0000010"
"11000011"	"11001011"	"10111000"	"1001001"	"1001001"	"11010011"	"10001010"	"0010100"	"0000111"	"0000010"	"11110101"	"0010110"	"1010100"
"11000011"	"11001011"	"01001000"	"1001011"	"1010000"	"10000101"	"0000111"	"10101101"	"110111001"	"11000000"	"0101010"	"0011011"	"01110001"
"111000000"	"01110000"	"10011101"	"0111010"	"1000001"	"0000110"	"0010001"	"11000000"	"10100010"	"0101001"	"0100111"	"11000101"	"0011100"
"11001010"	"01110000"	"10110100"	"0100111"	"0010110"	"0110000"	"11010000"	"111100000"	"0110101"	"0111010"	"111001101"	"01101110"	"10011000"
"10011000"	"0001110"	"0011101"	"1000000"	"0100000"	"11000000"	"0001000"	"0010110"	"0010001"	"1011001"	"0010100"	"1000001"	"0101100"

afterDecoding = 64x64 uint8 matrix

193	184	120	103	89	82	154	79	31	129	166	170	167	158	147	142	144	129	107	99	152	161	160	158	158	159	160	165	109	88	99	102	...
192	179	88	105	91	56	126	117	95	131	134	121	108	92	76	61	50	44	46	72	155	161	158	158	158	161	162	165	111	82	94	102	
186	170	58	105	154	125	76	95	106	71	44	36	30	26	25	25	28	32	39	87	157	159	157	156	158	160	163	164	122	89	91	98	
178	145	37	114	105	45	38	32	28	28	28	27	26	30	32	35	34	43	67	138	171	154	155	156	158	163	165	165	127	98	98	99	
170	101	28	119	95	30	33	32	30	29	28	26	26	29	34	40	59	126	127	163	167	151	153	155	158	162	164	161	127	104	106	101	
170	100	98	152	98	31	34	33	33	34	32	26	27	30	31	36	78	136	114	158	145	152	147	149	153	159	160	159	124	105	103	98	
136	144	144	140	108	43	44	45	44	47	36	25	29	30	32	52	103	123	119	141	131	157	154	155	155	159	160	158	121	107	108	105	
61	55	46	86	124	60	51	48	48	48	31	27	28	26	44	78	117	109	124	98	131	161	154	154	158	161	162	162	116	100	107	107	
37	43	48	100	123	49	49	52	52	46	38	53	84	96	62	82	107	99	126	64	133	160	156	157	157	156	148	143	115	95	95	93	
39	46	54	112	124	47	46	55	53	48	79	128	171	169	130	100	103	100	123	48	139	149	138	135	134	143	146	161	140	122	128	141	
⋮																																

Image:beforeCoding(64&64 rescaled to 512&512)

Image:AfterDecoding(64&64 rescaled to 512&512)

عکس ها تار به نظر میرسند، چرا که ابتدا به ۶۴ در ۶۴ پیکسل مقیاس شده بودند ولی برای نمایش از همان جا به ۵۱۲ در ۵۱۲ بازگردانده شده اند.

- **بلوک های Modulator و Demodulator :**

این بلوک ها وظیفه دارند که سیگنال را به کانال فرستاده و از کانال دریافت کنند.

بلوک مودولاتور، همانطور که در صورت آمده، یک بلوک QAM است که به صورت دوتایی بیت های کد شانون تولید شده را گرفته به صورت 4 سمبل به بیرون می‌دهد.

فرمت این سمبل ها در بخش قبل گفته شده و لذا آوردن دوباره آن نیازی نیست.پیاده سازی این بلوک نیز به راحتی انجام می‌شود، به این شکل که کد شانون دریافتی به رشته تبدیل می‌شود و به ازای هر دو بیت، سمبل مورد نظر به طول $f_s T_S$ نمونه ارسال می‌شود.

کد این ماژول در زیر آورده شده است :

```
function modulated = minModule(m,fc,Ts,fs)

% detecting Amc
if (m == 1 || m == 4)
   Amc = 1;
else
   Amc = -1;
end

% detecting Ams
if(m == 1 || m == 2)
   Ams = 1;
else
   Ams = -1;
end
% sampling rate?!
t = 1/fs:1/fs:Ts;
modulated = Amc*sqrt(2/Ts).*cos(2*pi*fc*t) +
Ams*sqrt(2/Ts).*sin(2*pi*fc*t);
end
```

سیگنال وارد کانال می‌شود و پس از عبور به بلوک دمودولاتور می‌رسد.

پیاده سازی بلوک در زیر آورده شده است :

```
function [out1,out2] = minDemodulateCorr(signal,fc,Ts,fs,phi)
sample_times = 1/fs:1/fs:Ts;
a = sqrt(2 / Ts);
n = ceil(length(signal) / 2);

s11 = a.*(cos(2*pi*fc*sample_times +
phi)+sin(2*pi*fc*sample_times + phi));
s12 = a.*(cos(2*pi*fc*sample_times + phi)-sin(2*pi*fc*sample_times
+ phi));
s21 = a.*(-cos(2*pi*fc*sample_times +
phi)+sin(2*pi*fc*sample_times + phi));
s22 = a.*(-cos(2*pi*fc*sample_times + phi)-
sin(2*pi*fc*sample_times + phi));

c11 = sum(s11 .* signal);
c12 = sum(s12 .* signal);
c21 = sum(s21 .* signal);
c22 = sum(s22 .* signal);
```

```
c = [c11,c12,c21,c22];
[~, i] = max(c);

a_ap = xcorr(cos(2*pi*fc*sample_times), signal) ./ fs;
b_ap = xcorr(sin(2*pi*fc*sample_times), signal) ./ fs;
a_ap = a_ap(1, n) * 2 * a;
b_ap = b_ap(1, n) * 2 * a;

if i == 1
   Ac = 1; As = 1;
end
if i == 2
   Ac = 1; As = -1;
end
if i == 3
   Ac = -1; As = 1;
end
if i == 4
   Ac = -1; As = -1;
end
out1 = [Ac,As];
out2 = [a_ap,b_ap]; % -> Ac As aprox.
end
```

خروجی این بلوک زوج ضرایب $[A_{cm}, A_{sm}]$ است که از روی این ضرایب، زوج بیت های متناظر به شکل زیر حاصل می‌شوند :

$$A_{mc} = \begin{cases} ۱ & m = ۱, ۴ \\ -۱ & m = ۲, ۳ \end{cases}$$

$$A_{ms} = \begin{cases} ۱ & m = ۱, ۲ \\ -۱ & m = ۳, ۴ \end{cases}$$

$$m = 1 \Longrightarrow 00, \qquad m = 2 \Longrightarrow 01, \qquad m = 3 \Longrightarrow 10, \qquad m = 4 \Longrightarrow 11$$

و لذا دنباله بیت های ارسالی بازیابی می‌شوند.

این کار وظیفه بلوک Detector است که در زیر آورده شده است :

```
function m = detector(constellation)
Ac = constellation(1,1);
As = constellation(1,2);
m = 0;
if(Ac == 1 && As == 1)
   m = 1;
elseif (Ac == 1 && As == -1)
   m = 4;
elseif (Ac == -1 && As == 1)
   m = 2;
elseif(Ac == -1 && As == -1)
   m = 3;
end

end
```

همچنین در کنار ضرایب $[A_{cm}, A_{sm}]$ ضرایب تخمین زده شده برای رسم Constellation نیز داده می‌شوند.

صحت عملکرد این بخش ها در فاز بعدی مشخص خواهد شد.

- **پیاده سازی فاز دوم :**

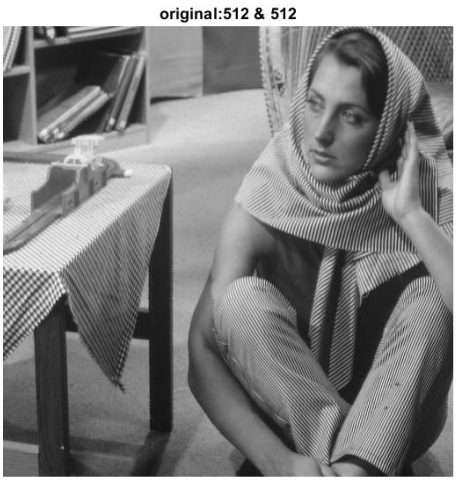
اکنون تمام بلوک ها حاضر هستند، این بلوک ها را به هم متصل میکنیم و موارد خواسته شده را گزارش می‌کنیم :

- **بخش الف) :**

توضیحات مربوط به این بخش و مثال آن در بخش قبل داده شده است.

در فاز قبلی ما عکس را کد و دیکود کردیم و نشان دادیم، در این بخش و بخش های بعدی برای نشان دادن صحت عملکرد این بخش ها، خود عکس را ارسال و سپس بازیابی میکنیم.

ما برای نشان دادن صحت عملکرد از یک عکس معیار استفاده میکنیم، عکس اصلی عکس زیر خواهد بود :



در زیر همین عکس، قبل و پس از مودولاسیون آورده شده است.

Image(resized 64&64 to 512&512):before module/demodulation

Image(resized 64&64 to 512&512):after module/demodulation

فعلا اثرات نویز در نظر گرفته نشده است.

- **بخش ب) و پ) :**

در این بخش باید اثر کانال را در نظر بگیریم، ابتدا سیگنال های داده شده را در حوزه فرکانس محدود میکنیم، به این شکل که کانال یک فیلتر میان گذر ایده آل است که سیگنال ورودی را (که قطاری از سمبول های S_m است) گرفته و فیلتر می‌کند.

در زیر تابع نوشته برای کانال آورده شده است :

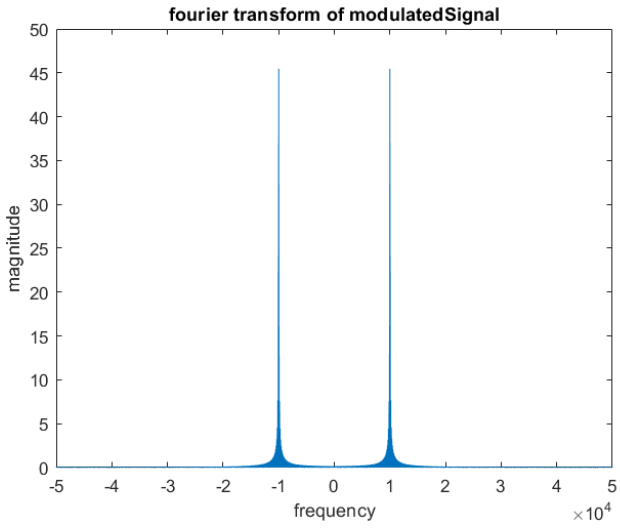
```
function out = Channel(signal,fc,fs,w)
fpassDown = fc - w/2;
fpassUp = fc + w/2;
out = bandpass(signal,[fpassDown fpassUp],fs);
end
```

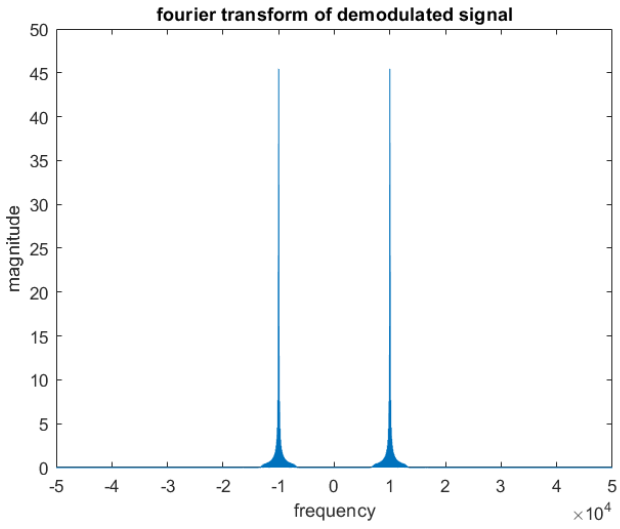
در محاسبه پهنای باند، ما برای عکس اول پهنای باند ۹۹ درصد انرژی را گزارش کردیم.

در مرحله بعد هم طبق خواسته سوال، پهنای باند ۹۹ درصد انرژی تمام عکس ها را محاسبه کرده و میانگین گرفتیم و اعلام کردیم (این مورد را از این جهت انجام دادیم که مطمئن شویم عکس ها را میتوان از کانال عبور داد.)

در زیر مثالی از طیف سیگنال قبل و پس از عبور از کانال آورده شده است :

تبدیل فوریه پیش و پس از ورود به کانال :

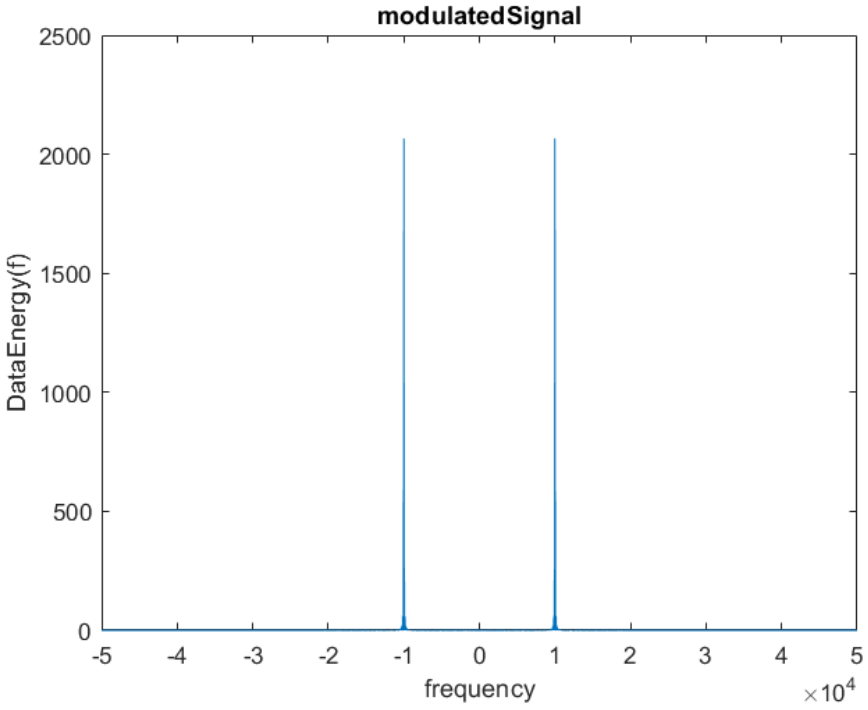




همانطورکه دیده می‌شود، پس از عبور از کانال تبدیل کات شده است و لذا سیگنال دچار اعوجاج شده است.

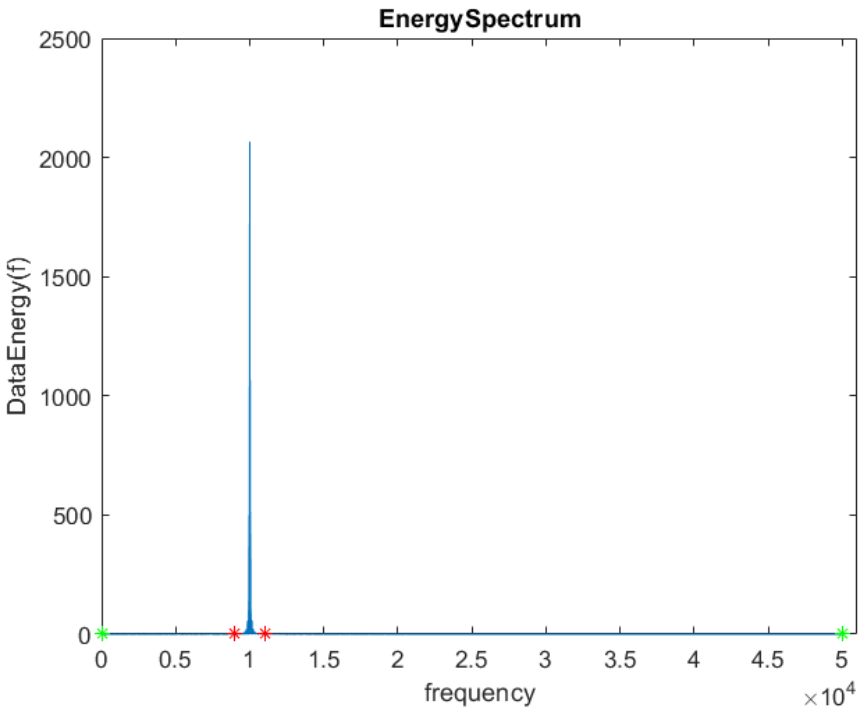
حال چگالی طیف ها را داریم :

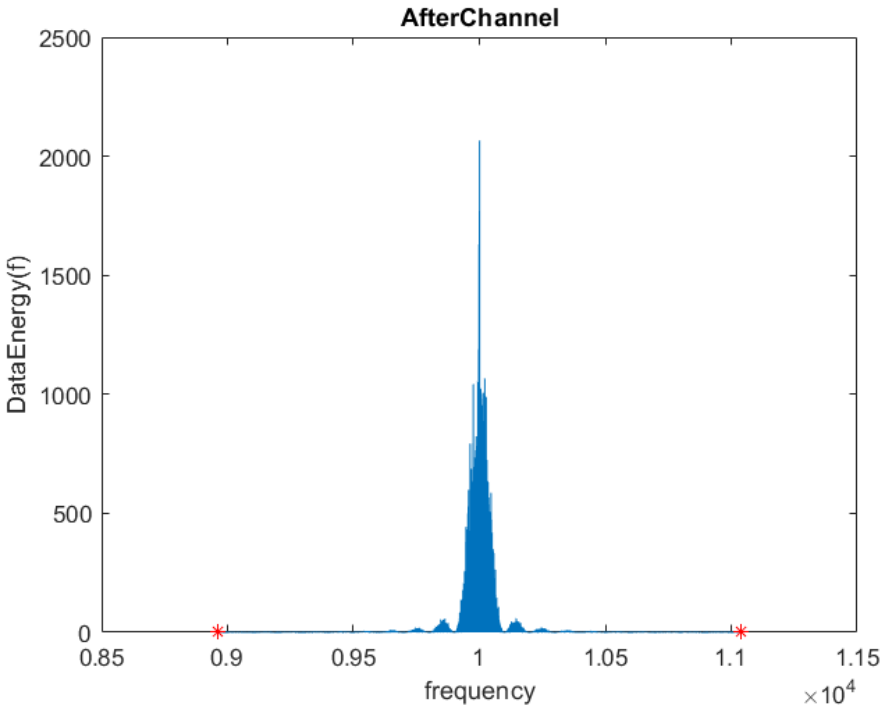
در زیر چگالی طیف کل سیگنال پیش از ورود به کانال آورده شده است.



همانطورکه دیده می‌شود، چون سمبل های ما سینوسی های کامل نیستند، طیف آنها حول فرکانس مرکزی (۱۰ کیلوهرتز) پخش شده است و لذا پس از عبور از کانال دچار اندکی اعوجاج می‌شوند.

در زیر شمایی نزدیکتر از طیف **یکطرفه** و همچنین محاسبه پهنای باند کل و پهنای باند 99 درصد انرژی را می‌آوریم.





فاصله بین نقاط سبز مبین پهنای باند کل سیگنال است و فاصله بین دو نقطه قرمز مبین پهنای باند 99 درصد انرژی است.

در شکل پایین نیز نمایی بزگ شده از طیف نمایش داده شده است.

مقدار پهنای باند عکس اول برابر ۲۰۸۰ هرتز شده است.

```
BW99percent = 2081.5299490619527205126360058784
```

پهنای باند سایر عکس ها نیز در زیر آورده شده اند :

```
name = '1.gif'
bw = 2081.5299490619527205126360058784
name = '2.gif'
bw = 2079.2801323062685696640983223915
name = '3.gif'
bw = 2069.8450145105216506635770201683
name = '4.gif'
bw = 1954.7743937175637256586924195 29
name = '5.gif'
bw = 2053.4888159333313524257391691208
name = '6.gif'
bw = 2053.1103712703006749507039785385
name = '7.gif'
bw = 1882.1030072850207943702116608 62
name = '8.gif'
bw = 1943.5917132161175686633214354515
name = '9.gif'
bw = 2083.7251163129949418362230062485
name = '10.gif'
bw = 2067.2414018844265228835865855217
```

```
name = '11.gif'
bw = 2077.3407108100836921948939561844
name = '12.gif'
bw = 2049.1747613829393230844289064407
name = '13.gif'
bw = 2040.2011514860914758173748850822
name = '14.gif'
bw = 2086.9910954386505181901156902313
name = '15.gif'
bw = 2036.2641557699007535120472311974
name = '16.gif'
bw = 2068.2623108690768276574090123177
name = '17.gif'
bw = 2067.6616967831505462527275085449
name = '18.gif'
bw = 2064.4553711202352133113890886307
name = '19.gif'
bw = 2066.9924680385502142598852515221
name = '20.gif'
bw = 2059.5035463055319269187748432159
```

```
name = '21.gif'
bw = 1985.5849016186093649594113230705
name = '22.gif'
bw = 2083.7607132865887251682579517365
name = '23.gif'
bw = 2000.7406146420380537165328860283
name = '24.gif'
bw = 2056.5889034364972758339717 9842
name = '25.gif'
bw = 2071.8406219579410389997065067291
name = '26.gif'
bw = 2077.0994795187707495642825961113
name = '27.gif'
bw = 2063.8854927935062733013182878494
name = '28.gif'
bw = 2038.8482775637294253101572394371
name = '29.gif'
bw = 1918.9028509822801424888893961906
name = '30.gif'
bw = 2067.6239143716575199505314230919
```

```
name = '31.gif'
bw = 2059.8978723619056836469098925591
name = '32.gif'
bw = 2059.1796946321355790132656693459
name = '33.gif'
bw = 2053.7523248692741617560386657715
name = '34.gif'
bw = 2070.2215960242974688299000263214
name = '35.gif'
bw = 2063.5478756784668803447857499123
name = '36.gif'
bw = 2060.9396290689637680770829319954
name = '37.gif'
bw = 2059.0830437303902726853266358376
name = '38.gif'
bw = 2056.7327413762832293286919593811
name = '39.gif'
bw = 2073.2045630666143551934510469437
name = '40.gif'
bw = 2074.5861308244948304491117596626
```

و پهنای میانگین کل برابر ۲۰۴۷ هرتز شده است :

```
vpa(mean(BWs))
```

```
ans = 2047.0389606326793909829575568438
```

- **بخش ت) :**

در این حالت یک نویز سفید گاوسی در کانال وجود دارد، اصولا در حالت کلی اثر نویز نسبتا کم است، مگر در SNR های بسیار بسیار پایین، چرا که توان سیگنال ارسال شده بسیار زیاد است. (توجه کنید که دامنه با معکوس طول سیگنال، T_s متناسب است و چون این مقدار کم است دامنه و لذا توان خیلی زیاد است).

لذا فقط در SNR های بسیار بسیار پایین اثر نویز واقعا خود را نشان می‌دهد.

برای آن که بدست آوردن نتایج راحت تر باشد، پس از صحبت با TA محترم درس (آقای طه عسگری) قرار بر آن شد که دامنه را در صورت نیاز کم کنیم، به این منظور دامنه را به فاکتور ۱۰۰ تا ۳۰۰ کاهش دادیم.

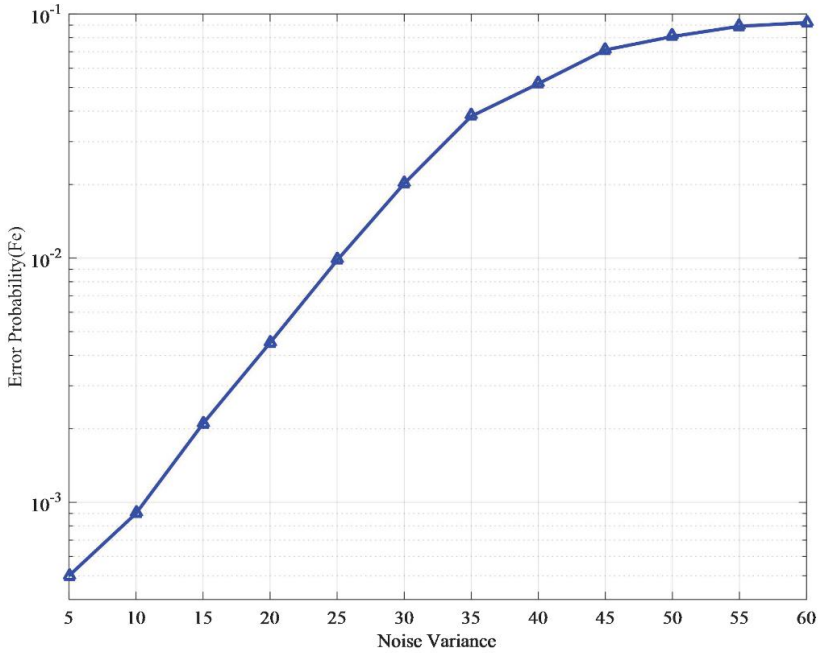
در کل سیستم به شدت نسبت به نویز مقاوم است و باید SNR را **خیلی خیلی** کم کرد تا اثر نویز نمایان شود.

در هر بخش از اکنون به بعد، در صورت کاهش دامنه ضریب آن را ذکر می‌کنیم.

در حالت کلی با افزایش اثر نویز ممکن است که در بخش Demodulator تصمیم گیری های اشتباهی رخ دهد و لذا بعضی از پیکسل ها اشتباه تشخیص داده شوند.

در زیر نمودار احتمال خطا بر حسب واریانس آورده شده است (در اینجا مراد از احتمال خطا، تعداد پیکسل های معیوب به پیکسل های کل است).

نمودار با تست های مختلف روی مقدار واریانس و تغییر دادن مقدار ضریب کاهش دامنه حاصل شده است.



در زیر تعدادی مثال از این حالت در SNR های مختلف آورده شده است:

به ترتیب از بالا چپ به پایین راست بر حسب افزایش SNR .

با کاهش بیشتر SNR (معادلا افزایش بیشتر واریانس نویز) این خطا ها بیشتر و بیشتر می‌شوند و در نهایت نویز گاوسی به طور کامل تصاویر را در بر میگیرد و لذا نمودار احتمالا خطا بر حسب واریانس در نهایت باید روی مقدار یک اشباع شود.

در شروع شیب افزایش نمودار باید کم باشد چرا که در شروع دمودولاتور به شدت در مقابل نویز مقاومت می‌کند ولی پس از رسیدن به مقدار به اندازه کافی زیاد دمودولاتور دیگر نمی‌تواند در مقابل نویز مقاومت کند و شیب ناگهان زیاد می‌شود.

برای تصمیم گیری بهینه در حضور نویز، نیز توابعی پیاده سازی شد که در بخش های قبل به طور کامل توضیح داده شدند.

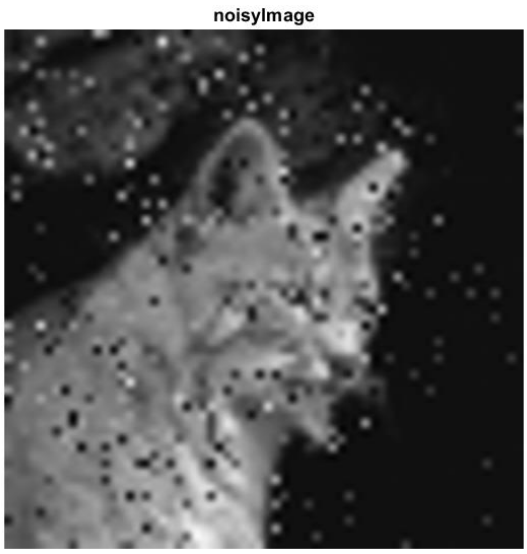
- **بخش ث) :**

همانطور که در دستور کار گفته شده است، تمامی عکس ها ارسال شده و SNR های آنها محاسبه شد و در نهایت میانگین شد.

چند مورد قابل ذکر است :

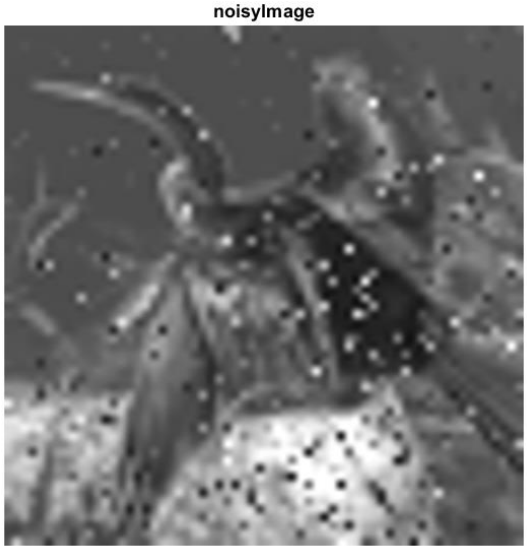
- در این بخش دو SNR مطرح است، یکی در ورودی دمودولاتور و یکی در خروجی مربوط به عکس، برای محاسبه SNR عکس باید اختلاف بین پیکسل های عکس ارسالی و عکس دریافتی را بدست آورد، این مقدار معادل نویز نهایی است و سپس باید توان آن و توان عکس را با هم مقایسه کرد.
  برای محاسبه SNR در ورودی دمودولاتور نیز کافیست که سیگنال های ارسال شده و اصلی بدون نویز (سیگنال میانگذر که با عبور از کانال حاصل شده) و نویز را بدست آورده و توان ها را مقایسه کنیم.
- محاسبه SNR سیگنال آنالوگ ورودی بدون مشکل است، برای محاسبه SNR عکس باید ماتریس را reshape کرده و تبدیل به یک بردار ساده بکنیم، مشابه همین کار را با عکس اصلی میکنیم و سپس بین این دو SNR میگیریم.
- همانطور که عرض شد، این دو سیگنال در دو جای کاملا متفاوت و با دو خصلت کاملا متفاوت هستند و لذا مقایسه SNR این دو سیگنال نسبتا بی معنی است.

در زیر نمونه هایی از تصاویر ارسالی و دریافتی آورده شده اند، تمام این تصاویر با ضریب افت دامنه ۱۵۰ و SNR پیش از دمودولاسیون (پس از کانال) برابر با منفی ۵۰ دسی بل پردازش شده اند :



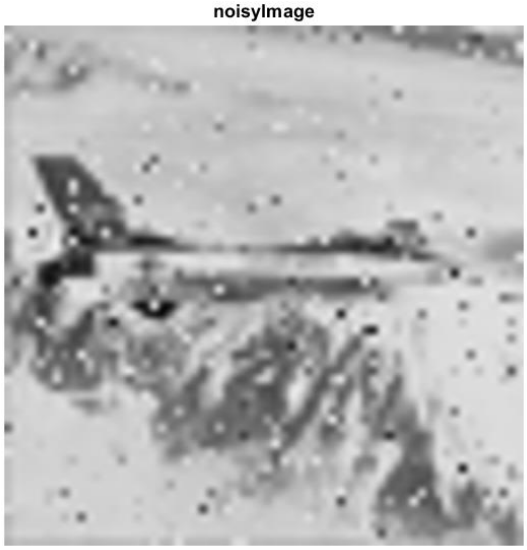
mainImage

noisyImage



mainImage

noisyImage



mainImage

noisyImage

مقدار SNR میانگین :

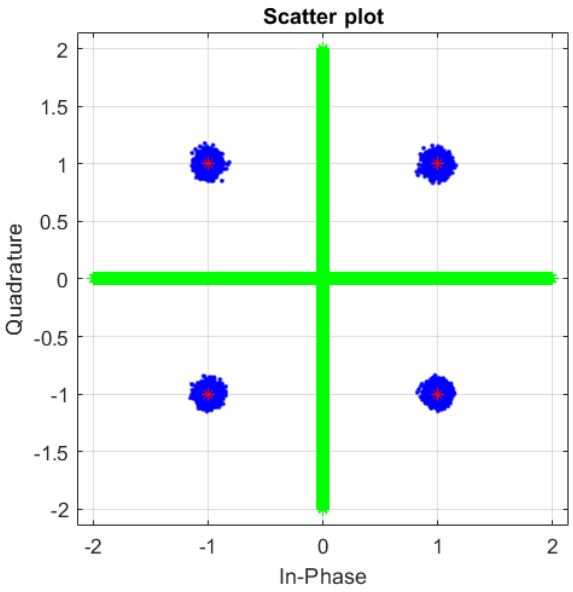
```
meanSnrBefore = 0.3134      meanSnrAfter = 15.7922
```

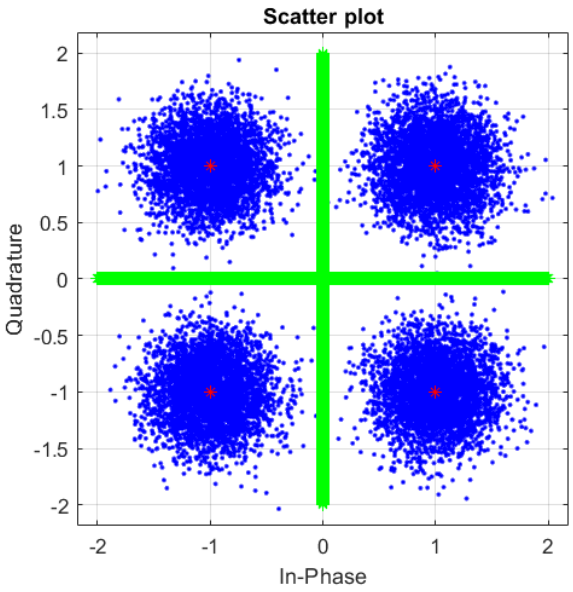
- **بخش ج) :**

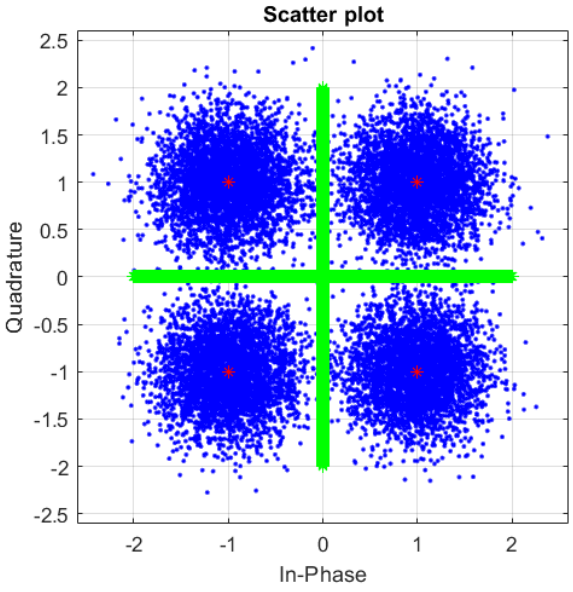
منحنی Constellation وظیفه دارد که معیار اساسی ما برای تعیین مرز نواحی تصمیم گیری باشد، هر یک از سیگنال های دریافت شده از کانال باید به یکی از 4 سمبل نگاشته شوند، یک راه تصمیم گیری برای آن که هر سمبل به کدام گروه اختصاص داشته در بخش قبل آورده شده است. (در این بخش برای وضوح بیشتر با اجازه TA درس (آقای طه عسگری) از ضریب افت دامنه ۵۰ تا ۲۵۰ استفاده کردیم).

برای رسم نمودار های Constellation نیز از ضرایب تقریبی تولید شده توسط تابع Demodulator استفاده میکنیم و با آن نمودار های Scatter را رسم می‌کنیم، از آنجا که دامنه خیلی زیاد است، نویز به سختی اثر گذار است و لذا مجبوریم دامنه را کم کنیم (مثل بخش قبل).

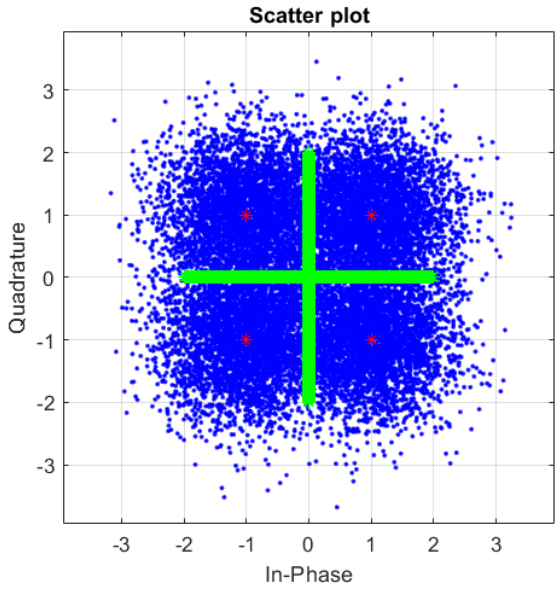
یک بار نویز کم، یک بار نویز بالا ولی بدون عبور از حد و یک بار نویز در حالت عبور از حد و نویز بالا و نویز بسیار بالا.



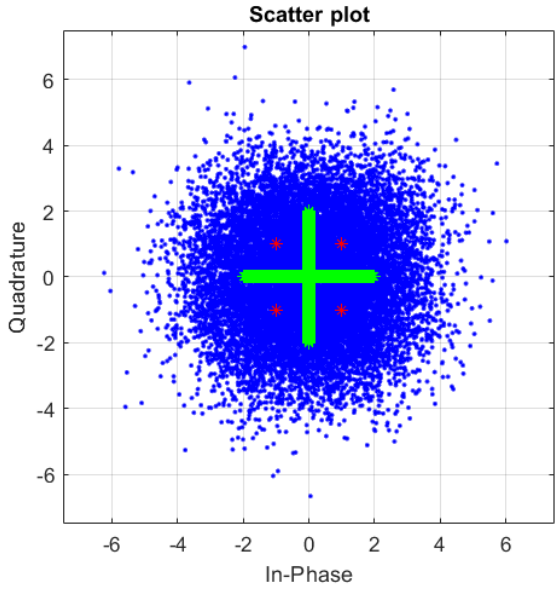


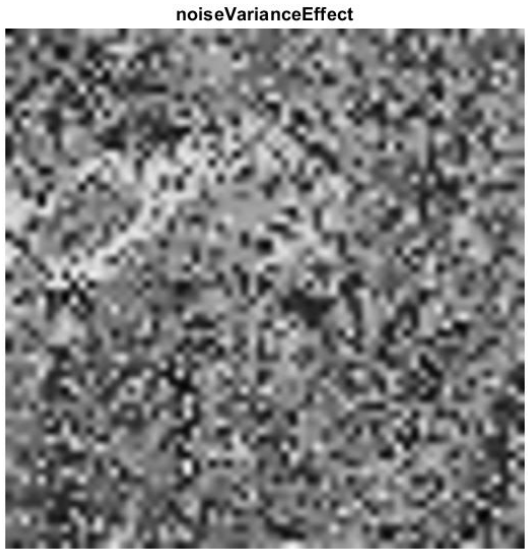

Scatter plot
Quadrature
In-Phase
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-2
-1
0
1
2


noiseVarianceEffect


Scatter plot
Quadrature
In-Phase
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3


noiseVarianceEffect


Scatter plot
Quadrature
In-Phase
6
4
2
0
-2
-4
-6
-6
-4
-2
0
2
4
6


noiseVarianceEffect

با افزیش نویز، خوشه ها پخش‌تر و پخش تر می‌شوند، در نهایت به نقطه ای میرسیم که ممکن است خوشه ها با قدری پخش شوند که بعضی از نقاط به طور قطعی برای یک خوشه نباشند، یک معیار برای این که کی چنین حالتی رخ می‌دهد، این است که چک کنیم آیا نقاط روی خوشه ها به مرز بین خوشه ها رسیده اند یا نه، با عبور از حد، نویز به سرعت شدید خواهد شد.

- **بخش چ) :**

همانطور که در توضیحات نظری آورده شده، افزایش اختلاف فاز باعث می‌شود که برخی سمبل ها جابه‌جا شوند، با اشاره به نمودار های Constellation میتوان این موضوع را بهتر شرح داده.

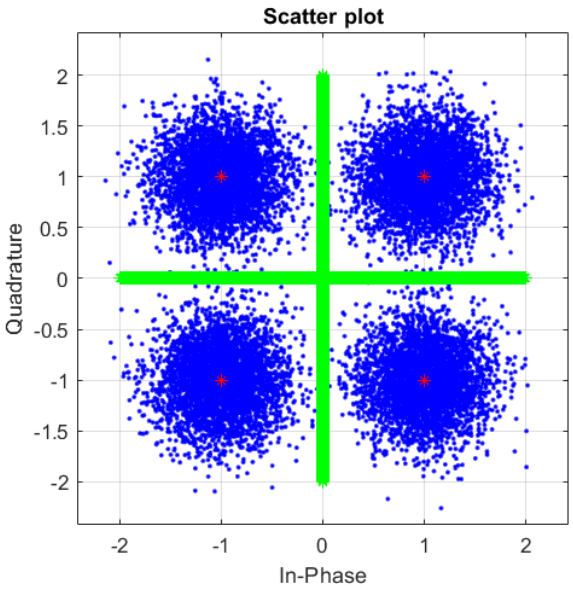
احتمالا لازم باشد نتایج بخش تئوری را بیاوریم :

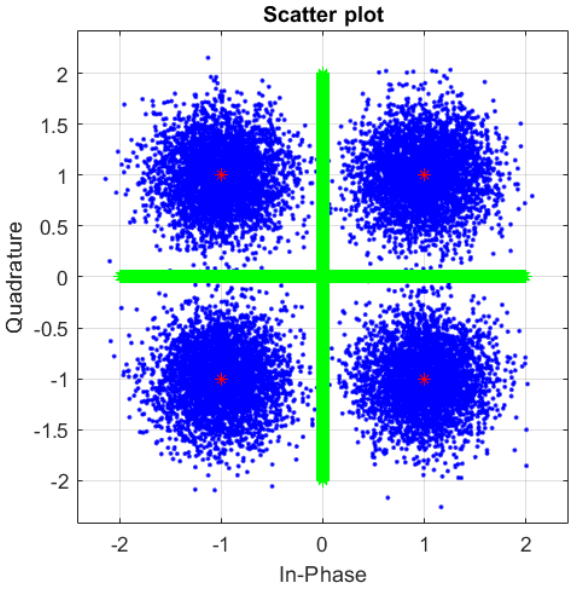
$$D(\phi, i, j) = \cos\left(\phi + \frac{\pi}{2}(j - i)\right)$$

با افزایش اختلاف فاز برخی از سمبل ها که به یک خوشه اولیه تعلق داشتند ممکن است که به یک خوشه دیگر منتقل شوند و لذا ممکن است سمبلی که معادل عدد 00 به 01 یا دیگری تبدیل شود، این مورد میتواند پیکسل ها را به طور کامل عوض کند، با افزایش اختلاف فاز از صفر، بسته به عکس، ابتدا سمبل ها عوض نمی‌شوند، سپس به طور ناگهانی سمبل های 1 و سپس 2 و ... شروع به تغییر می‌کنند، لذا کیفیت عکس ممکن است با افزایش کوچکی در فاز به طور ناگهانی کاهش پیدا کند.

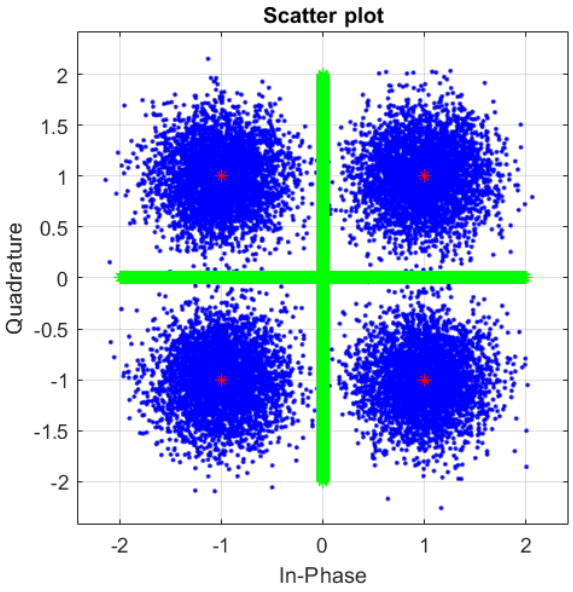
این مورد از رابطه بالا نیز واضح است، سمبل فرستاده شده اگر j باشد، باید i به نحوی انتخاب شود که مقدار D به احد امکان به ۱ نزدیک شود، لذا باید آرگومان کسینوس به اندازه امکان به 0 یا 2π نزدیک شود، لذا ممکن است به ازای یک زاویه خاص، اندکی نغییر در زاویه به طور ناگهانی پاسخ را عوض کند.

در زیر نمودار های خواسته شده به ترتیب در حالت های مشخص شده آورده شده اند (ضریب افت دامنه در این بخش ۷۰ است):

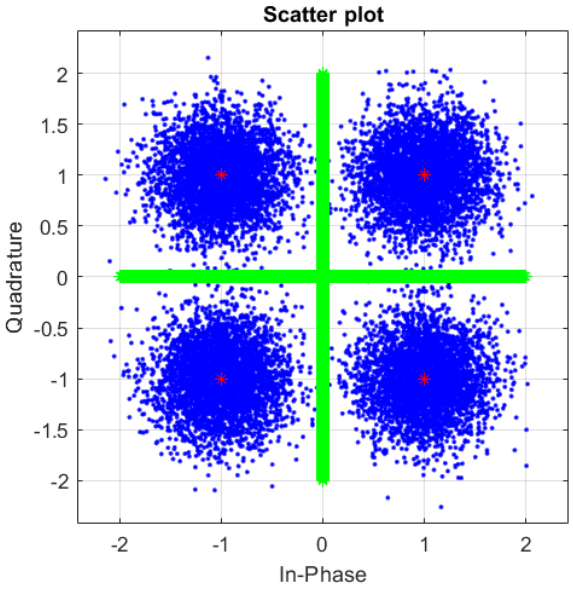


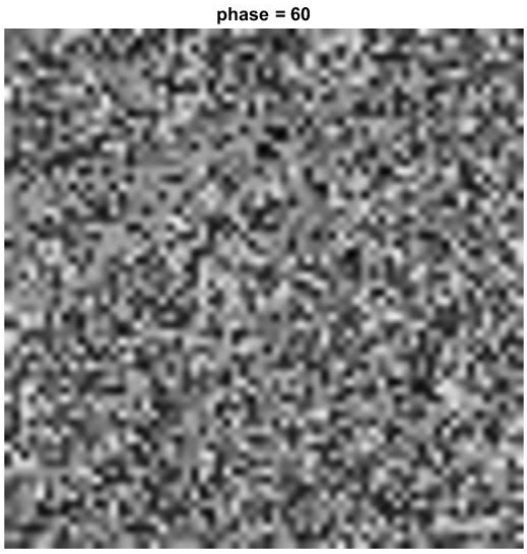

Scatter plot
Quadrature
In-Phase
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2


phase = 15


Scatter plot
Quadrature
In-Phase
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2


phase = 30


Scatter plot
Quadrature
In-Phase
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2


phase = 60

همانطورکه دیده می‌شود، با پخش شدن خوشه ها تا جایی‌که نقاط از مرز ها رد شوند، به طور ناگهانی نویز عکس بالا خواهد رفت.