به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

مخابرات بیسیم

استاد صباغیان

گزارشکار پروژه دوم

فاطمه جلیلی
۸۱۰۱۹۹۳۹۸
تاریخ تحویل : ۱۴۰۳/۰۳/۱۹

سؤال ۱

برای پیاده سازی این قسمت از فرمول انتگرالی که در درس خواندیم برای بدست آوردن میانگین احتمال خطای بیت استفاده می کنیم:

$$\overline{P_e} = \int_0^{\infty} P_e(\gamma)\, f_{\gamma_\Sigma}(\gamma) d\gamma$$

که در این سوال چون از مدولاسیون QAM استفاده می کنیم داریم:

$$P_{e_{symbol}}(\gamma) = 2Q\left(\sqrt{\frac{3\log_2 M}{M-1}\gamma}\right)$$

که فرمول فوق مربوط به احتمال خطای هر سمبل است و برای تقریب حدودی احتمال خطای هر بیت آن را تقسیم بر $\log_2 M$ می کنیم.

در رابطه با تابع توزیع γ_Σ زمانی که از MRC برای ترکیب شاخه ها استفاده می کنیم داریم:

$$\gamma_\Sigma = \frac{r^2}{N_{total}} = \frac{\left(\sum_{i=1}^M a_i r_i\right)^2}{N_0 \sum_{i=1}^M a_i^2}$$

برای ماکسیمم کردن SNR کل از فرمول فوق نسبت به ضرایب a_i مشتق می گیریم و برابر صفر قرار می دهیم تا ضریب هر شاخه را بدست آوریم:

$$\frac{\partial \gamma_\Sigma}{\partial a_i} = 0 \rightarrow a_i^2 = \frac{r_i^2}{N_0} = \gamma_i$$

با جایگذاری این ضرایب در فرمول γ_Σ داریم:

$$\gamma_\Sigma = \frac{\left(\sum_{i=1}^M a_i r_i\right)^2}{N_0 \sum_{i=1}^M a_i^2} = \frac{\left(\frac{1}{\sqrt{N_0}}\sum_{i=1}^M r_i^2\right)^2}{\sum_{i=1}^M r_i^2} = \sum_{i=1}^M \frac{r_i^2}{N_0} = \sum_{i=1}^M \gamma_{\mathrm{i}}$$

کانال ما از نوع رایلی است بنابراین:

$$r_i \sim Rayliegh \rightarrow r_i = \sqrt{X_i^2 + Y_i^2}, \quad X_i, Y_i \sim N(0, \sigma^2)\ \&\ IID$$

$$\rightarrow \gamma_i = \frac{r_i^2}{N_0} = \frac{X_i^2 + Y_i^2}{N_0} \rightarrow \gamma_\Sigma = \sum_{i=1}^M \gamma_{\mathrm{i}} = \frac{1}{N_0}\sum_{i=1}^M X_i^2 + Y_i^2$$

که مجموع $2M$ تا متغیر با توزیع گوسی است، طبق آمار و احتمال می دانیم مجموع k توزیع گوسی با میانگین صفر و واریانس یکسان $\sigma'^2 = \frac{\sigma^2}{N_0}$ دارای توزیه *chi-square* با k درجه آزادی است.

تابع توزیع *chi-square* :

$$f_Z(\zeta) = \frac{1}{2^{\frac{k}{2}}\,\Gamma\left(\frac{k}{2}\right)\sigma'^k}\zeta^{\frac{k}{2}-1} e^{-\frac{\zeta}{2\sigma'^2}}$$

قرار دهیم $k = 2M$:

$$f_{\gamma_\Sigma}(\gamma) = \frac{\gamma^{M-1} e^{-\frac{\gamma}{2\sigma'^2}}}{2^M\ \Gamma(M)\sigma'^{2M}}$$

داریم:

$$\rightarrow \bar{\gamma} = E\{\gamma_i\} = \frac{1}{N0}(E\{X_i^2\} + E\{Y_i\}^2) = \frac{2\sigma^2}{N_0} = 2\sigma'^2$$

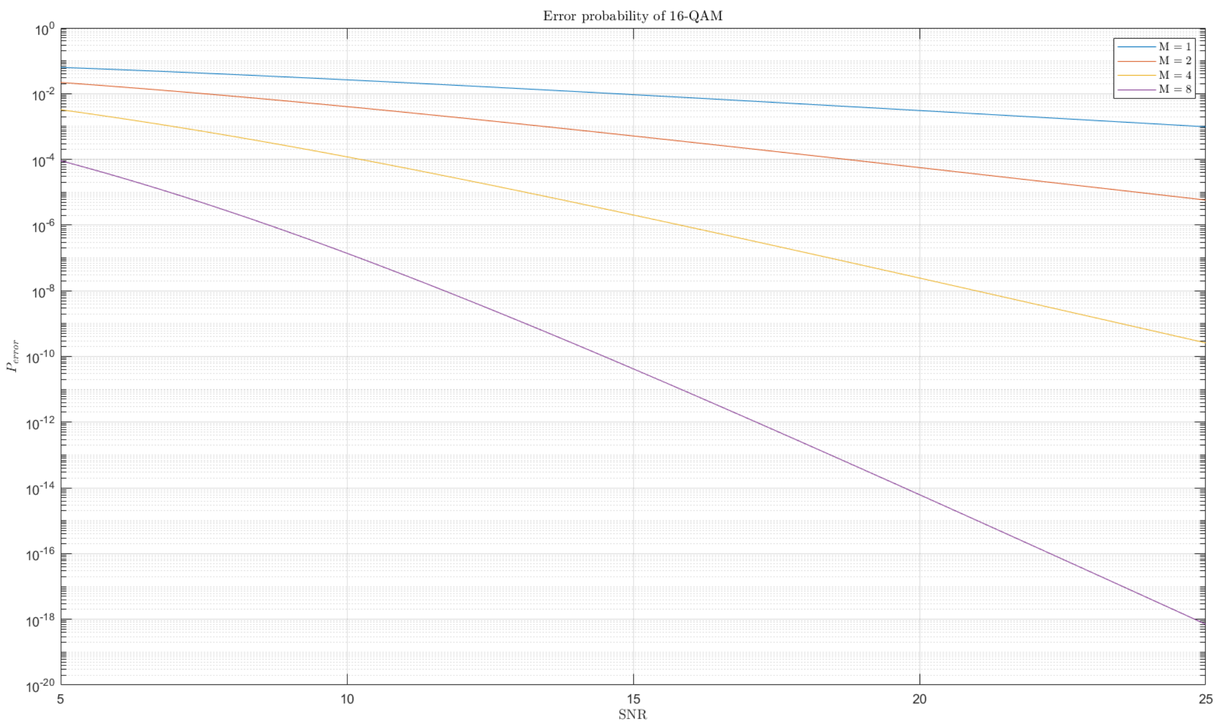
با جایگذاری داریم:

$$f_{\gamma_\Sigma}(\gamma) = \frac{\gamma^{M-1} e^{-\frac{\gamma}{\bar{\gamma}}}}{\bar{\gamma}^M\ (M-1)!}$$

حال که رابطه ی ورودی انتگرال به طور کامل مشخص شد ، در یک حلقه *for loop* به ازای $\bar{\gamma}$ های مختلف ورودی انتگرال را مطابق زیر بدست می آوریم و به کمک متلب از آن انتگرال می گیریم:

```
for snrBar = 10 .^ (snrBarList ./ 10)
        intInput = @(snr) (2 * qfunc(sqrt(snr * 3 * log2(M) / (M - 1) )) / log2(M)) ...
.* (snr .^ (m - 1)) .* exp(-snr ./ snrBar) / (snrBar ^ m * factorial(m - 1));
        meanPeBitList(snrIterNum) = integral(intInput, 0, inf);
        snrIterNum = snrIterNum + 1;
    end
```

و این کار را در یک *for loop* بیرونی برای تعداد شاخه های مختلف که سوال خواسته است تکرار می کنیم و نهایتا نمودار میانگین احتمال خطای بیت برای مدولاسیون *16-QAM* با تعداد شاخه های مختلف در گیرنده که از *MRC* استفاده می کند را در یک نمودار رسم می کنیم:



همانطور که انتظار داشتیم به در همه نمودار ها با بالاتر رفتن $\bar{\gamma}$ مقدار میانگین احتمال خطای بیت کم تر می شود و این روند کاهشی زمانی که از تعداد شاخه های *diversity* بیش تری استفاده می کنیم مشهود تر است .

سؤال ۲

الف)

برای اینکه در ارسال های متفاوت رفتار متفاوتی از کانال ببینیم یا به عبارتی کانال در ارسال های متفاوت ناهمبسته باشد تا استفاده از دایورسیتی سودمند باشد، برای مثال اگر در یک ارسال کانال وضعیت خوبی نداشت در دیگری این چنین نباشد و مزایای استفاده از دایورسیتی آشکار شود باید به اندازه ی T_c یا همان زمان همدوستی که محدوده ی زمانی است که $R_C(\Delta t)$ در آن غیر صفر است صبر کنیم.

$R_C(\Delta t)$ به صورت زیر تعریف می شود:

$$R_c(\tau_1,\tau_2;t,\Delta t)=E\{c(\tau_1,t)c^*(\tau_2,t+\Delta t)\}\xrightarrow{WSS\ \&\ US}R_c(\tau,\Delta t)\overset{\mathcal{F}_\tau}{\to}R_C(\Delta f,\Delta t)\xrightarrow{\Delta f=0}R_C(\Delta t)$$

ب)

در رابطه با نحوه تصمیم گیری بهینه در مدولاسیون BPSK داریم:

$$\begin{cases}H_1\colon r_l=\sqrt{\varepsilon_g}+n_l\\ H_2\colon r_l=-\sqrt{\varepsilon_g}+n_l\end{cases}\to f(r_l|H1)\gtrless f(r_l|H_2)$$

با توجه به اینکه n_l نویز سفید مختلط گوسی است داریم:

$$\frac{1}{2\pi N_0}e^{-\frac{\left\|r_l-\sqrt{\varepsilon_g}\right\|^2}{2N_0}}\gtrless\frac{1}{2\pi N_0}e^{-\frac{\left\|r_l+\sqrt{\varepsilon_g}\right\|^2}{2N_0}}$$

$$\to\left\|r_l\right\|^2+\varepsilon_g-r_l^*\sqrt{\varepsilon_g}-r_l\sqrt{\varepsilon_g}\lessgtr\left\|r_l\right\|^2+\varepsilon_g+r_l^*\sqrt{\varepsilon_g}+r_l\sqrt{\varepsilon_g}$$

$$\to 2\sqrt{\varepsilon_g}(r_l+r_l^*)\gtrless 0\to Re\{r_l\}\gtrless 0$$

لازم به ذکر است دایورسیتی زمانی رو می توان مشابه دایورسیتی مکانی در نظر گرفت چون بین ارسال هر سمبل به اندازه زمان همدوستی صبر می کنیم گویا در نهایت سمبل ها در چند کانال ناهمبسته به صورت موازی ارسال شده اند و فقط زمان بیش تری برای ارسال تمامی سمبل ها نسبت به حالت دایورسیتی مکانی صبر کرده ایم. بنابراین برای تصمیم گیری بهینه از همان روش MRC استفاده می کنیم.

نحوه محاسبه میانگین احتمال خطا زمانی که از MRC استفاده می کنیم در سوال قبل به طور کامل بیان شد، در این سوال هم تابع توزیع γ_Σ مانند سوال قبل و تنها چیزی که تغییر می کند $P_e(\gamma)$ است، در مدولاسیون $BPSK$ داریم:

$$P_e(\gamma)=P\{Re\{r_l\}<0\}=P\left\{-Re\{n_l\}>\sqrt{\varepsilon_g}\right\}=Q\left(\sqrt{\frac{\varepsilon_g}{N_0}}\right)=Q(\sqrt{\gamma})$$

با توجه به اینکه در مدولاسیون $M{=}2$ $BPSK$ است، احتمال خطای سمبل و بیت یکسان است

بنابراین حل تئوری دقیقا مشابه سوال اول است و تنها فرمول $P_e(\gamma)$ که در بالا بدست آوردیم در محاسبه ورودی انتگرال عوض می شود:

```
for snrBar = 10 .^ (snrBarList ./ 10)
        intInput = @(snr) qfunc(sqrt(snr))...
            .* (snr .^ (l - 1)) .* exp(-snr ./ snrBar) / (snrBar ^ l * factorial(l - 1));
        meanPeBitList(snrIterNum) = integral(intInput, 0, inf);
        snrIterNum = snrIterNum + 1;
    end
```

این کار را در یک *for loop* بیرونی برای تعداد دفعات ارسال متفاوت متناظر با تعداد شاخه های مختلف که سوال خواسته است تکرار می کنیم و نهایتا نمودار میانگین احتمال خطای بیت را در یک نمودار رسم می کنیم.

برای رسم نمودار ها با استفاده از شبیه سازی داریم:

```
Llist = [1 2 3 4 5];
snrBarList = -10 : 0.5 : 25;
sampleNum = 1e6 ;
N0 = 1;
x = 2 * randi([0 1], 1, sampleNum) - 1;
for l = Llist
    snrIterNum = 1;
    meanPeBitList = zeros(1, length(snrBarList));
    hNormal = randn(l, sampleNum) + 1j * randn(l, sampleNum);
    for snrBar = 10 .^ (snrBarList ./ 10)
        h = sqrt(snrBar / 2) * hNormal;
        w = sqrt(N0) .* (randn(l, sampleNum) + 1j * randn(l, sampleNum));
        y = x .* h + w;
        alpha = sqrt((abs(h) .^ 2) ./ N0) .* exp(-1j * angle(h));
        z = sum(y .* alpha, 1) ./ sum(abs(h) .^ 2, 1);
        errorBits = real(z) .* x < 0;
        meanPeBitList(snrIterNum) = sum(errorBits) / sampleNum;
        snrIterNum = snrIterNum + 1;
    end
    semilogy(snrBarList, meanPeBitList, '-o')
    hold on
end
```

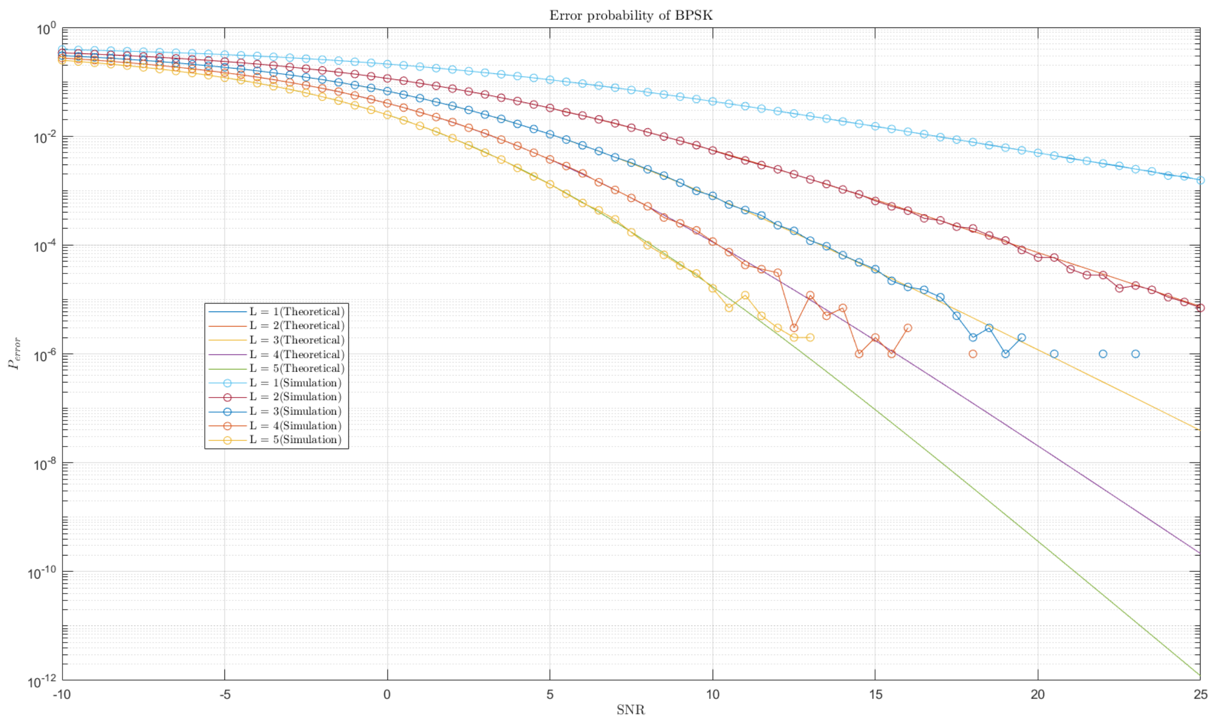
مشابه حالت تئوری دو *for loop* تو در تو برای تعداد دفعات ارسال مختلف و $\bar{\gamma}$ های مختلف داریم، ولی اینبار یه تعداد *sampleNum* بیت ارسالی ۱- یا ۱ مطابق مدولاسیون *BPSK* تولید می کنیم. کانال را به صورت جمع یک توزیع گوسی و j برابر شده توزیع گوسی دیگر که یک کانال با اندازه ای با توزیع رایلی و زاویه ای با توزیع یکنواخت بدست می دهد تعریف می کنیم.

در حلقه *for loop* درونی به ازای هر *snrBar* پارامتر توزیع های گوسی که کانال را تشکیل می دهند عوض می شود و w هم با توزیع مختلط گوسی تعریف می شود و بر حسب h و w بیت های دریافتی در y ذخیره می شوند.

ضریبی به نام *alpha* برای هر شاخه تعریف می شود که وظیفه دارد ضرایب مخصوص *MRC* که طبق محاسبات تئوری در سوال اول $a_i^2 = \frac{r_i^2}{N_0}$ بدست آمد و همچنین *co-phasing* را اعمال کند. برای اعمال *co-phasing* هر شاخه را در ضرب $e^{-j\measuredangle h_i}$ می کنیم تا اثر فاز کانال جبران شود.

در نهایت چون از *diversity* استفاده می کنیم خروجی همه شاخه ها با هم جمع می شود و مطابق تصمیم گیرنده بهینه برای مدولاسیون *BPSK* که در بالا بدست آوردیم یعنی $Re\{r_l\} \gtrless 0$ دمودلاسیون را انجام می دهیم و با مقایسه با بیت های ارسالی

تعداد اختلاف ها بدست می آید و میانگین احتمال خطا محاسبه می شود. لازم به ذکر است پس از جمع کردن خروجی همه شاخه ها، خروجی کلی را با تقسیم بر $\sum |h_i|^2$ نرمالایز می کنیم تا دمودالاسیون به درستی انجام گیرد، انجام اینکار در $BPSK$ تاثیری در خروجی ندارد چرا که علامت $Re\{r_l\}$ را عوض نمی کند اما در سوالات بعدی برای مدولاسیون QAM لازم هست که حتما اینکار انجام بگیرد تا دمولاتور به درستی تشخیص دهد.



این کار را برای تعداد دفعات ارسال متفاوت تکرار می کنیم و در نهایت نمودار های مربوط به پیاده سازی عملی و تئوری را همگی در یک $plot$ رسم می کنیم:

همانطور که انتظار داشتیم به در همه نمودار ها با بالاتر رفتن $\bar{\gamma}$ مقدار میانگین احتمال خطای بیت کم تر می شود و این روند کاهشی زمانی که از تعداد دفعات ارسال بیش تری استفاده می کنیم مشهود تر است، نتایج پیاده سازی عملی (خطوط دایروی) و پیاده سازی تئوری کاملا منطبق هستند، دلیل خطای پیاده سازی عملی در $snrBar$ های بالاتر به دلیل کم بودن تعداد سمپل ها $sampleNum$ است، چرا که مثلا ۱۰۰۰۰۰۰ بیت ارسال می شود و snr آنقدر زیاد است که هیچ کدام از بیت ها اشتباه نمی شود ولی در $loop$ بعدی به طور شانسی تنها یک بیت اشتباه می شود و لذا نمودار های عملی در $snrBar$ بالا دقیق نیستند چون به علت کم بود $sampleNum$ دیگر نتایج در آن میزان از $snrBar$ کاملا رندوم شده و نمی تواند توزیع اصلی خود را دنبال کند. علت قطع شدن نمودار های عملی از یه جایی به بعد هم به همین دلیل است که از تعداد محدود ۱۰۰۰۰۰۰ بیت ارسالی دیگری هیچ بیتی اشتباه نمی شود، احتمال خطا صفر می شود و متلب در پلات خود نمی تواند این را نمایش دهد.

سؤال ۳

الف)

همانطور که در سوال پیش توضیح داده شد این ساختار همان ساختار مشابه دایورسیتی در زمان است ولی زمانی که از دایورسیتی زمانی استفاده می کنیم اگر M بار بخواهیم یک سمبل را ارسال کنیم باید M برابر بیش تر از حالتی که از دایورسیتی مکانی استفاده می کنیم صبر کنیم. اگر بخواهیم این ساختار با کد الموتی دقیقا مشابه حالت دایورسیتی زمانی عمل کند باید تنها از یک آنتن فرستنده استفاده کنیم و در هر دو تایم اسلات مربوط به کد الموتی سمبل یکسان را ارسال کنیم، گویا از دایورسیتی زمانی استفاده می کنیم.

ب)

روندی که برای پیاده سازی دنبال می کنیم مشابه سوال قبل است با این تفاوت که یک بعد سوم به ماتریس های بیت های ارسالی یعنی X و کانال یعنی H اضافه می کنیم که این بعد تایم اسلات های مربوط به کد الموتی را در خود جای می دهد.

دیگر لازم نیست به ازای تعداد دفعات ارسال یا شاخه های متفاوت پیاده سازی کنیم لذا تنها یک *for loop* روی *snrBar* های مختلف در بازه ی خواسته شده داریم.

```
hNormal = randn(TxNum, sampleNum) + 1j * randn(TxNum, sampleNum);
HNormal = cat(3, hNormal, hNormal);

Xts1 = 2 * randi([0 1], TxNum, sampleNum) - 1;
Xts2 = [-conj(Xts1(2,:)); conj(Xts1(1,:))];
X = cat(3, Xts1, Xts2);
```

با توجه به اینکه کانال در تایم اسلات های مختلف یکسان فرض می شود مشابه روند قبل یکبار آن را تولید می کنیم و در بعد سوم دو بار تکرار می کنیم.

بیت های ارسالی *x1* , *x2* را هم رندوم ۱- و ۱ تولید می کنیم و در ردیف های یک ماتریس ۲ در *sampleNum* به نام *Xts1* ذخیره می کنیم و با توجه به اینکه بیت های ارسالی در تایم اسلات دوم $-x_2^*, x_1^*$ هستند *Xts2* هم تولید کرده و در بعد سوم به *Xts1* concat می کنیم.

```
for snrBar = 10 .^ (snrBarList ./ 10)
    H = sqrt(snrBar / 2) * HNormal;
    W = sqrt(N0) .* (randn(1, sampleNum, tsNum) + 1j * randn(1, sampleNum, tsNum));
    Y = sum(X .* H, 1) + W;
    m1 = conj(H(1, :, 1)) .* Y(1, :, 1) + H(2, :, 1) .* conj(Y(1, :, 2));
    m2 = conj(H(2, :, 1)) .* Y(1, :, 1) - H(1, :, 1) .* conj(Y(1, :, 2));
    errorBits1 = real(m1) .* Xts1(1, :) < 0;
    errorBits2 = real(m2) .* Xts1(2, :) < 0;
    meanPeBitList(snrIterNum) = (sum(errorBits1) + sum(errorBits2)) / (2 * sampleNum);
    snrIterNum = snrIterNum + 1;
end
```

در *for loop* کانال، W و خروجی Y مشابه سوال قبل تنها با اضافه کردن بعد سوم تعریف می شوند و سپس با توجه به رابطه ی زیر برای $m1, m2$ که در کلاس درس بیان شد تا هر بار $s1$ یا $s2$ از معادلات حذف شوند آن ها را محاسبه می کنیم:

$$m_1 = h_1^* y_1 + h_2 y_2^*, \quad m_2 = h_2^* y_1 - h_1 y_2^*$$

$$\rightarrow m_1 = (|h_1|^2 + |h_2|^2)\, s_1 + n_1 h_1^* + n_2^* h_2, \quad m_2 = (|h_1|^2 + |h_2|^2)\, s_2 + n_1 h_2^* - n_2^* h_1 \ (*)$$

و مشابه سوال قبل با گیرنده بهینه $Re\{r_l\} \gtrless 0$ تصمیم گیری را انجام می دهیم و تعداد بیت های خطا را می شماریم و میانگین احتمال خطا را بدست می آوریم و در نهایت رسم می کنیم. نتیجه رسم همه مدولاسیون ها با هم در قسمت د آمده است.

ج)

میانگین احتمال خطا با دایورسیتی در زمان با ۲ بار تکرار و استفاده از کد الموتی برای دو آنتن فرستنده کاملا یکسان است ولی آنچه مزیت دارد مطابق آنچه پیش تر هم توضیح داده شد نرخ ارسال دوبرابر هنگام استفاده از دایورسیتی در مکان است. در واقع هنگامی که از دایورسیتی مکانی استفاده می کنیم و آنتن های بیش تر استفاده می کنیم یک سمبل را چند بار ولی به صورت موازی می فرستیم و کد الموتی هم کمک می کند حتی با نداشتن کانال در فرستنده دمودلاسیون را انجام دهیم ولی زمانی که از دایورسیتی مکانی استفاده می کنیم ارسال تکراری یک سمبل گویی به صورت سری انجام می شود و لذا نرخ ارسال سمبل ها $\frac{1}{L}$ می شود که L تعداد دفعات ارسال هر سمبل است و گیرنده باید صبر کند تا تکرار ارسال های یک سمبل تمام شود و سپس دمودلاسیون را انجام دهد.

د)

روند دنبال شده دقیق مشابه قسمت الف است منتها چون مدولاسیون و دمودلاسیون QPSK و 16-QAM پیچیده تر از BPSK هستند برای پیاده سازی آن ها از توابع آماده متلب یعنی pskmod, pskdemod و qammod, qamdemod استفاده می کنیم.

QPSK:

```
x1 = randi([0 M-1], 1, sampleNum);
x2 = randi([0 M-1], 1, sampleNum);
Xts1 = [pskmod(x1, M, pi/M); pskmod(x2, M, pi/M)];
Xts2 = [-conj(Xts1(2, :)); conj(Xts1(1, :))];
X = cat(3, Xts1, Xts2);


for snrBar = 10 .^ (snrBarList ./ 10)
    H = sqrt(snrBar / 2) * HNormal;
    W = sqrt(N0) .* (randn(1, sampleNum, tsNum) + 1j * randn(1, sampleNum, tsNum));
    Y = sum(X .* H, 1) + W;
    m1 = conj(H(1, :, 1)) .* Y(1, :, 1) + H(2, :, 1) .* conj(Y(1, :, 2));
    m2 = conj(H(2, :, 1)) .* Y(1, :, 1) - H(1, :, 1) .* conj(Y(1, :, 2));
    errorBits1 = symerr(pskdemod(m1, M, pi/M), x1);
    errorBits2 = symerr(pskdemod(m2, M, pi/M), x2);
    meanPeBitList(snrIterNum) = (errorBits1 + errorBits2) / (2 * sampleNum);
    snrIterNum = snrIterNum + 1;
end
```

همانطور که مشاهده می شود تنها تفاوت در نحوه تولید بیت های ارسالی و دمودلاسیون است که با استفاده از توابع آماده متلب انجام می شود.

:16-QAM

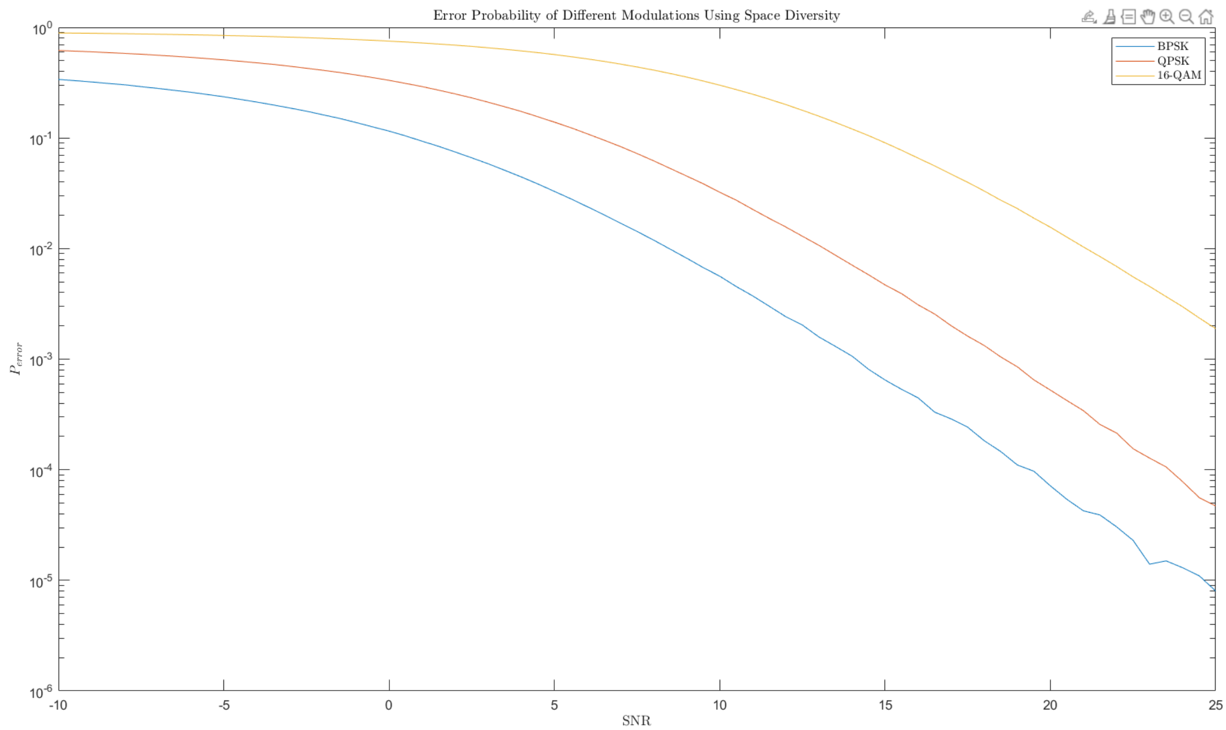
```
x1 = randi([0 M-1], 1, sampleNum);
x2 = randi([0 M-1], 1, sampleNum);
Xts1 = [qammod(x1, M, UnitAveragePower=true); qammod(x2, M, UnitAveragePower=true)];
Xts2 = [-conj(Xts1(2, :)); conj(Xts1(1, :))];
X = cat(3, Xts1, Xts2);


for snrBar = 10 .^ (snrBarList ./ 10)
    H = sqrt(snrBar / 2) * HNormal;
    W = sqrt(N0) .* (randn(1, sampleNum, tsNum) + 1j * randn(1, sampleNum, tsNum));
    Y = sum(X .* H, 1) + W;
    m1 = (conj(H(1, :, 1)) .* Y(1, :, 1) + H(2, :, 1) .* conj(Y(1, :, 2)))...
        ./ (abs(H(1, :, 1)) .^2 + abs(H(2, :, 1)) .^ 2);
    m2 = (conj(H(2, :, 1)) .* Y(1, :, 1) - H(1, :, 1) .* conj(Y(1, :, 2)))...
        ./ (abs(H(1, :, 1)) .^2 + abs(H(2, :, 1)) .^ 2);
    errorBits1 = symerr(qamdemod(m1, M, UnitAveragePower=true), x1);
    errorBits2 = symerr(qamdemod(m2, M, UnitAveragePower=true), x2);
    meanPeBitList(snrIterNum) = (errorBits1 + errorBits2) / (2 * sampleNum);
    snrIterNum = snrIterNum + 1;
end
```

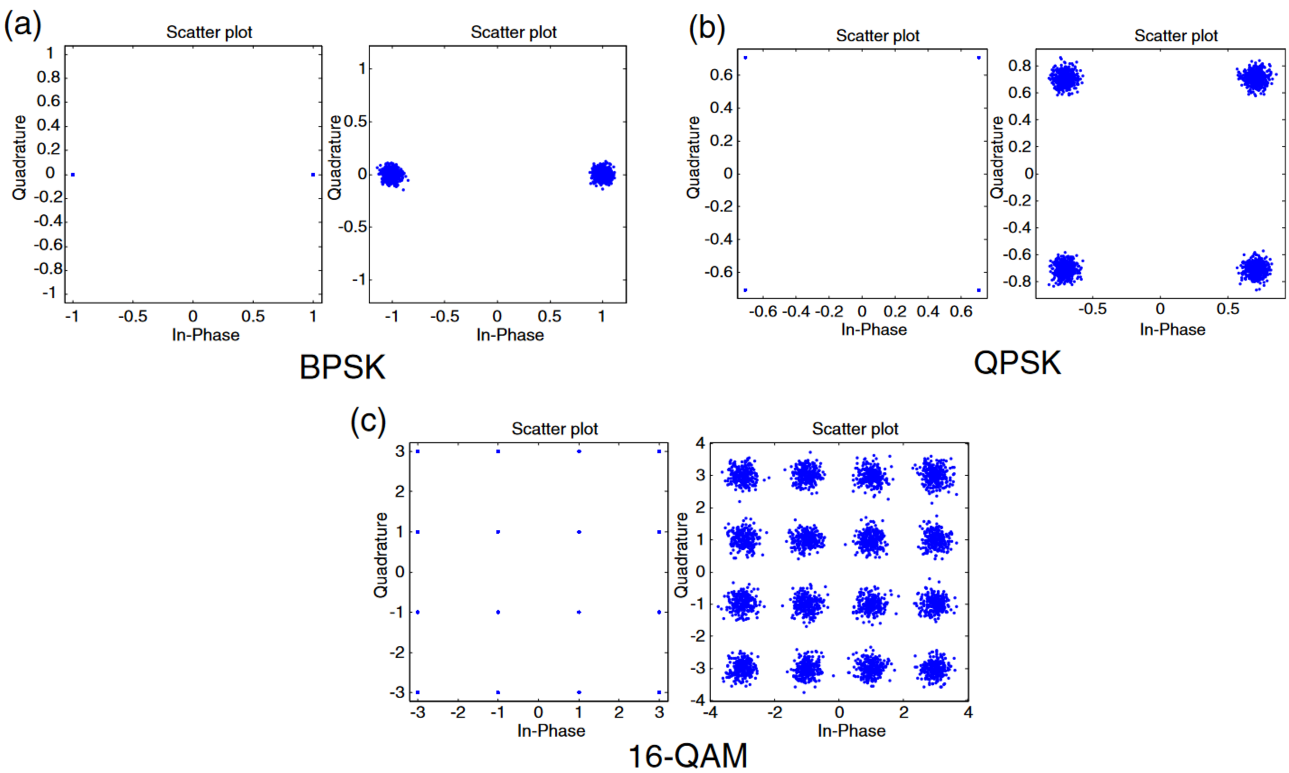
همانطور که در ابتدای صفحه ۶ توضیح داده شد در مدولاسیون QAM لازم است تا ورودی دمودلاتور نرمالایز شود به این معنی که ضریب s_1, s_2 در ورودی دمودلاتور یک باشد، بنابراین مطابق $(*)$ ، m1 , m2 را بر $(|h_1|^2 + |h_2|^2)$ تقسیم می کنیم و سپس دمولاسیون را انجام می دهیم انجام اینکار در مودلاسیون های قبلی تاثیری در عملکرد گیرنده بهینه مخصوص آن دمولاسیون نداشت و برای مثال علامت سمبل ارسالی دریافتی را تغییر نمی داد و لذا گیرنده حتی با عدم انجام اینکار به درستی عمل می کرد.

همچنین برای اینکه مطمین شویم هم در فرستنده و هم در گیرنده توان ارسالی سمبل ها ماکسیمم یک است تا فرستنده و گیرنده با هم سینک باشند و دمودلاسیون درست انجام شود از عبارت `UnitAveragePower=true` هم در تابع qammod و هم در تابع qamdemod استفاده می کنیم.

در نهایت میانگین احتمال خطا برای همه مدولاسیون های خواسته شده با استفاده از داورسیتی مکانی و کد الموتی را در یک plot رسم می کنیم:



همانطور که انتظار داشتیم به در همه نمودار ها با بالاتر رفتن $\bar{\gamma}$ مقدار میانگین احتمال خطای بیت کم تر می شود و این روند کاهشی زمانی که از تعداد نقاط *constellation* کم تری استفاده می کنیم مشهود تر است، به عبارتی زمانی که انرژی هر بیت ثابت است با بیش تر شدن نقاط *constellation* احتمال خطا در تشخیص سمبل های دریافتی بالاتر می رود این مورد در تصاویر *scatterplot* زیر کاملا مشهود است:



بنابراین به ترتیب *BPSK* ، *QPSK* و سپس *16-QAM* روند کاهش میانگین احتمال خطای بهتری با افزایش *snrBar* را دنبال می کند.