به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

مخابرات بیسیم

استاد صباغیان

گزارشکار پروژه سوم

فاطمه جلیلی

۸۱۰۱۹۹۳۹۸

تاریخ تحویل : ۶ /۰۲/۱۴۰۳

سؤال ۱

(۱

کل پهنای باند کانال L (تعداد تپ های کانال) برابر پهنای باندی که هر تپ اشغال می کند یعنی $\frac{1}{T_d}$ است، بنابراین تعداد تپ های کانال مطابق رابطه ی زیر بدست می آید:

$$W = \frac{L}{T_d} \rightarrow 20 \times 10^6 = \frac{L}{10 \times 10^{-6}} \rightarrow L = 200$$

از طرفی طول *cyclic prefix* بزرگ تر مساوی یکی کم تر از تعداد تپ ها کانال است بنابراین آن را ۱۹۹ در نظر می گیریم:

$$cpLength \geq L - 1 \rightarrow cpLength = 199$$

(۲

تعداد زیر حامل ها باید به حدی تعیین شود که در طول یک بلاک که طول می کشد همه زیر حامل ها را ارسال کنیم وضعیت کانال ثابت باشد و بتوانیم حدودا آن را ثابت فرض کنیم، این زمان با زمان همدوسی کانال معین می شود بنابراین مشابه استدلال سوال ۱ داریم:

$$W \geq \frac{n_c}{T_c} \rightarrow n_c \leq W \times T_c \rightarrow n_c \leq 20 \times 10^6 \times 5 \times 10^{-3} \rightarrow n_c \leq 10^5$$

از طرفی مطابق مطالب سر کلاس تعداد زیر حامل ها باید حدودا بزرگ تر از ۷، ۸ برابر پیشوند گردشی باشد تا ریت پس از اضافه کردن گارد تایم بیش از اندازه کم نشود یعنی نسبت $\frac{n_c}{n_c+L}$ هر چه قدر بزرگ تر باشد مناسب تر است، بنابراین:

$$n_c \geq 8 \times L \rightarrow n_c \geq 1600$$

همچنین با توجه به الگوریتم انجام FFT که در ادامه کاربرد آن در سیستم OFDM توضیح داده می شود، برای انجام سریع تر این عملیات نیاز است تعداد نقاط FFT و IFFT توان دو باشد لذا ماکسیمم توان ۲ کم تر از 10^5 را به عنوان n_c انتخاب می کنیم:

$$n_c = 2^{16} = 65536$$

(۳

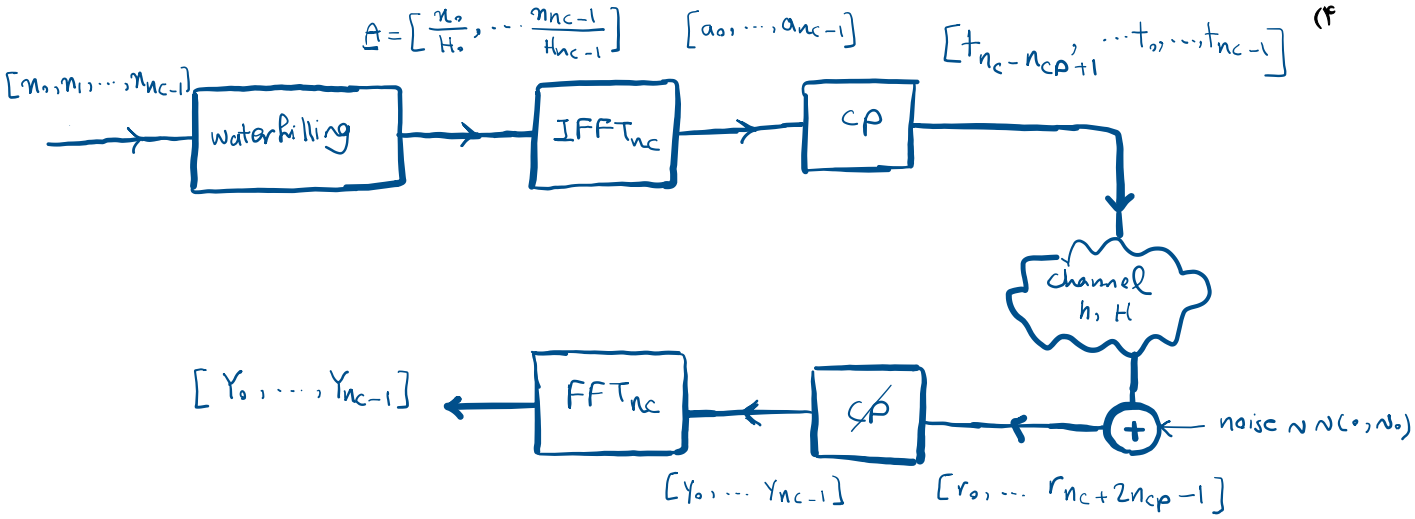
به وضوح برای اینکه تمامی پیام ارسال شود تعداد بلاک ها باید حد بالای تقسیم طول پیام به طول هر بلاک (تعداد زیرحامل ها) باشد، بنابراین:

$$blockNum = \left\lceil \frac{N}{n_c} \right\rceil$$

با توجه به محدودیت پردازشی لپ تاپ برای جلوگیری از زمان ران تایم خیلی طولانی طول پیام به جای 10^8، 10^6 در نظر گرفته شد بنابراین تعداد کل بلاک ها برابر است با:

$$\rightarrow blockNum = \left\lceil \frac{10^6}{65536} \right\rceil = 16$$

(۴



دیاگرام فرستنده و گیرنده سیستم OFDM همراه با Waterfilling و ورودی و خروجی های هر بلاک که مطابق نام گذاری کد متلب است رسم شده است.

بلاک Waterfilling :

در سیستم OFDM خالی بدون وجود این بلاک خروجی نهایی $x_k H_k$ است که H_k به ازای هر سمبل متفاوت است، در واقع OFDM راه حلی بود که یک کانال فرکانس گزین را به چندین کانال فلت تبدیل کنیم ولی برای بهبود عملکرد هر یک از این کانال های فلت نیاز است تا اثر کانال (هم فاز و هم اندازه) جبران شود تا سمبل ارسالی به خوبی بازیابی شود، سه روش در درس شامل Waterfilling، Diversity و Equalizer معرفی شدند که Waterfilling ساده ترین است.

بلاک Waterfilling که در درس معرفی شد با داشتن کانال در فرستنده هر سمبل ارسالی x_k را تقسیم بر کانال مربوطه H_k می کند تا سمبل بازیابی شده در گیرنده با ضرب در کانال همان سمبل اصلی را نتیجه بدهد، با این کار هم جبران فاز و هم اندازه صورت می گیرد.

روش بیان شده در صورت پروژه روش بهینه سازی را بیان می کند که در صورت داشتن توان کل محدود Pmax بتوانیم تخصیص توان به هر زیرحامل را به صورت بهینه انجام دهیم. فرمول بیان شده در صورت پروژه حاصل حل مسئله ماکسیمم کردن ظرفیت نسبت به P_i ها است که توسط روش لاگرانژ حل شده و فرمول بیان شده برای P_i حاصل برقراری شرایط KKT است، جزییات این روش در بهینه سازی محدب بیان می شود.

بلاک IFFT :

همانطور که در درس بیان شد برای ارسال تعدادی سیگنال سینوسی $\sin(2\pi k \Delta f T)$ به صورت همزمان به نحوی که بازیابی آن ها قابل انجام باشد نیازی است که این سیگنال ها در حوزه زمان بر هم عمود باشند یا به عبارتی $\Delta f = \frac{1}{T}$ در باند پایه و $\Delta f = \frac{1}{2T}$ در باند میانی باشد (مطابق آنچه در تمرین دوم ثابت شد) ، بنابراین مطابق نرخ نایکویست $f_s = 2 \times \frac{N\Delta f}{2} = \frac{N}{T}$ پس با نمونه برداری از سیگنال ارسالی $x(t) = \sum_{n=0}^{N-1} a_n e^{j2\pi n \Delta f t}$ داریم:

$$x_k = x\left(\frac{kT}{N}\right) = \sum_{n=0}^{N-1} a_n e^{\frac{j2\pi nk}{N}}$$

که همان رابطه $IFFT$ است.

بلاک CP :

با توضیحات پیشین مشکلات تداخل داخل بلاک را با ارسال زیر حامل های متعدد حل کردیم، ولی همچنان در اثر چند مسیره بودن کانال بین بلاک ها تداخل داریم، بنابراین بین هر دو بلاک $OFDM$ حداقل به اندازه طول پاسخ ضربه ی کانال زمان محافظ می گذاریم. از طرفی چون در هنگام عبور از کانال سیگنال ها با پاسخ ضربه کانال کانوالو می شوند با ایجاد فرم شبیه تناوبی در سمبل های ارسالی می توانیم این کانولوشن را به کانولوشن گردشی تبدیل کنیم تا از خواص ضرب تبدیل فوریه کانال و سمبل ها در حوزه فرکانس استفاده کنیم.

کانال :

با گذر از کانال مطابق فرض سوال نویز با انحراف معیار $N0$ به حاصل کانولوشن سمبل ها و پاسخ ضربه کانال اضافه می شود.

بلاک حذف CP و FFT :

در ادامه برای بازیابی سمبل های دریافتی، $cyclic\ prefix$ که در هنگام ارسال اضافه کرده بودیم را حذف می کنیم و n_c FFT نقطه ای می گیریم.

۵)

پیاده سازی سیستم فوق از دو for loop تو در تو روی SNR های مختلف و بلاک ها استفاده می کنیم.

از آنجایی که رابطه SNR بر حسب N0 مطابق $SNR = \frac{P_{max}}{n_c N0}$ داده شده است، یک بار آرایه ی N0 بر حسب SNR ها را بدست می آوریم و روی این آرایه for loop بیرونی را اجرا می کنیم.

```
SNR_dB = -20 : 30; %dB
SNR = 10 .^ (SNR_dB ./ 10);
N0Array = Pmax ./ (nc .* SNR);
```

قبل از وارد شدن به حلقه ها پس از تولید رندوم پیام ارسالی و ماژوله کردن آن بر حسب BPSK آن را reshape می کنیم و در ماتریس X قرار می دهیم که سطر های آن سمبل های ارسالی در هر بلاک را نشان می دهند، شایان ذکر است برای اینکه تمامی بیت های های پیام ارسال شوند و در ماتریس X قرار بگیرند تعدادی ۰ به انتهای پیام اضافه می کنیم که تعداد بیت ها بخش پذیر بر طول هر بلاک شود.

```
xm = 2 * randi([1, 2], 1, N) - 3;
X = reshape([xm, zeros(1, nc - mod(N, nc))], nc, blockNum).';
```

برای پیاده سازی بلاک Waterfilling تابعی به این نام تعریف می کنیم:

```
function A = waterfilling(x, N0, H, Pmax)
    func = @(lambda) Pmax - sum(max((1 ./ lambda) - (N0./ abs(H).^2), 0));
    lambdaHat = fzero(func, [1e-4, 1 / min(N0 ./ abs(H).^2)]);
    P = max((1./ lambdaHat) - (N0./ abs(H).^2), 0);
```

```
    W = sqrt(P) .* exp(-1i * angle(H));
    A = x .* W;
end
```

مطابق صورت پروژه ما سعی داریم P_i ها را به صورت بهینه بیابیم که در این راستا رابطه ی آن ها به صورت زیر بدست آمده است:

$$P_i^* = \max\left(\frac{1}{\lambda} - \frac{N_0}{|H_i|^2}, 0\right) \quad (*)$$

به طوری که :

$$P_{max} = \sum_{i=0}^{n_c-1} P_i^*$$

بنابراین هدف ما این است که اختلاف P_{max} و $\sum_{i=0}^{n_c-1} P_i^*$ به حداقل برسد، بنابرین تابعی به نام func بر حسب لامبدا برابر $P_{max} - \sum_{i=0}^{n_c-1} P_i^*$ مطابق کد تعریف می کنیم و λ هایی که این تابع را صفر می کنند یعنی ریشه های تابع را با استفاده از fzero می یابیم. تابع fzero نیاز به ورودی بازه جواب دارد که در آن تابع تغییر علامت می دهد، حد پایین این بازه به صورت سعی و خطا پیدا شد ولی برای حد بالا هر کدام از P_i ها نمی توانند بیش تر از $\frac{1}{\lambda} - \frac{N_0}{|H_{min}|^2}$ شوند.

با یافتن lambdaHat با کمک تابع fzero از ریشه های تابع func را مطابق رابطه ی (*) می یابیم و ضریبی که در بلوک waterfilling باید در هر زیرحامل ضرب شود را بر حسب P_i و فاز کانال برای جبران فاز می یابیم و با ضرب این ضرایب در x ، خروجی بلاک waterfilling یعنی A را می یابیم.

به ازای هر بلاک کانال به صورت رندوم تعریف می شود و fft آن به تابع waterfilling به عنوان ورودی داده می شود:

```
h = sqrt(hPower / 2) * (randn(1, L) + 1i * randn(1, L));
H = fft(h ,nc);
AWF = waterfilling(x, N0, H, Pmax);
```

در کنار محاسبات مربوط به سیستم OFDM با وجود بلاک waterfilling برای مقایسه با سیستم OFDM بدون این بلاک ضریبی که در زیر حامل ها در سیستم دوم ضرب می شود را بدون حل مسئله بهینه سازی ای برابر $\frac{P_{max}}{n_c}$ در نظر می گیریم:

```
A = sqrt(Pmax / nc) * x;
```

سپس بلاک های IFFT و CP را هم مطابق توضیحات سوال قبلی پیاده سازی می کنیم:

```
aWF = ifft(AWF, nc);
a = ifft(A, nc);
tWF = [aWF(nc - cpLength + 1 : end), aWF];
t = [a(nc - cpLength + 1 : end), a];
```

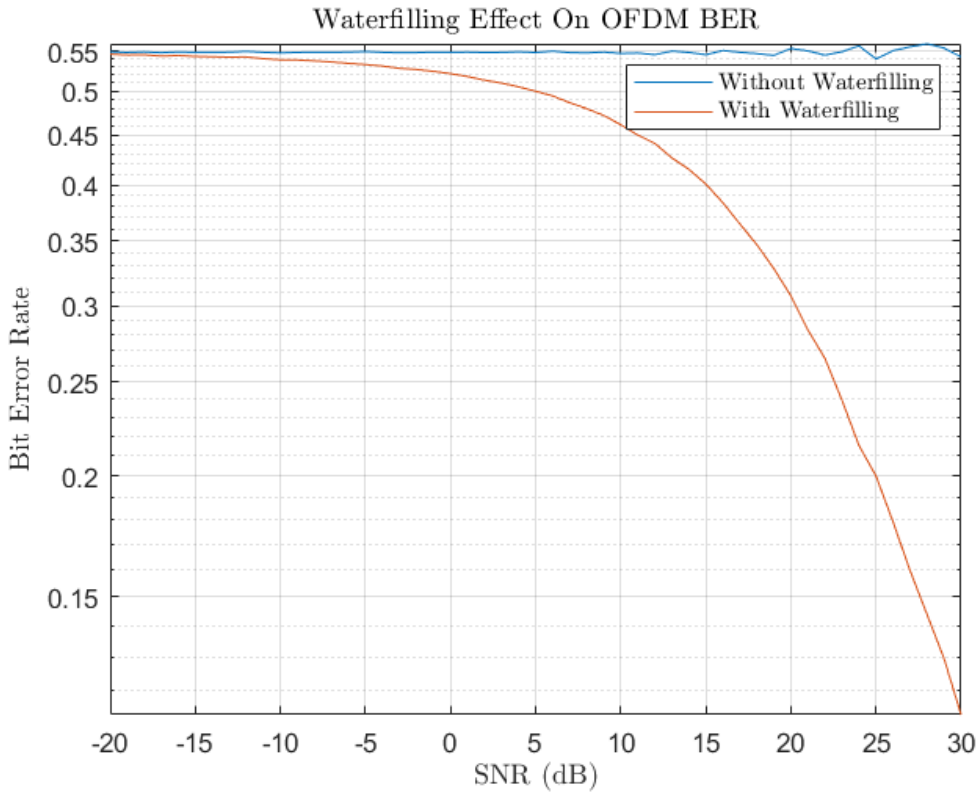
نویز را به صورت رندوم به طول $n_c + 2n_{cp}$ که طول سیگنال کانوالو شده به طول اولیه $n_c + n_{cp}$ با کانال است تولید می کنیم و بلاک های حذفCP و FFT را پیاده سازی می کنیم:

```
noise =  sqrt(N0 / 2) * (randn(1, nc + 2 * cpLength) + 1i * randn(1, nc + 2 *
cpLength));
rWF = conv(tWF, h) + noise;
r = conv(t, h) + noise;
yWF = rWF(cpLength + 1 : cpLength + nc);
y = r(cpLength + 1 : cpLength + nc);
YWF = fft(yWF, nc);
Y = fft(y, nc);
```

نحوه تصمیم گیری در مدولاسیون BPSK که در پروژه های قبلی اثبات شد بر اساس real سمبل دریافتی است، تصمیم گیری را انجام داده و با مقایسه با سمبل های اولیه ارسالی تعداد بیت های خطا را می یابیم :

```
xHatWF = real(YWF) > 0;
xHat = real(Y) > 0;
numErrorBitsWF = numErrorBitsWF + sum((2 * xHatWF - 1) ~= x);
numErrorBits = numErrorBits + sum((2 * xHat - 1) ~= x);
```

با تقسیم تعداد بیت خطا بر طول کل پیام ber را می یابیم و رسم می کنیم:



همانطور که انتظار داشتیم سیستم OFDM خالی که اثر کانال در سمبل های دریافتی آن جبران نشده است اصلا عملکرد درستی ندارد چرا که هر کدام از کانال های فلت سیستم به تنهایی کارکرد درستی ندارند و برای عملکرد درست نیاز به یکی از روش های Waterfilling، Diversity و Equalizer است.

۶

تمامی مراحل پیاده سازی مانند سوال قبل است، فقط در فرستنده خط مربوط به waterfilling را حذف می کنیم و از Equalizer در انتهای ساختار پس از بلوک FFT استفاده می کنیم.

برای این منظور نیاز است تا تنها ضرایبی که در سمبل های دریافتی ضرب می شوند را بسته به نوع Equalizer تعریف کنیم. با توجه به مطالب کلاسی برای دو Zero Forcing, MMSE Equalizer این ضرایب مطابق زیر تعریف می شوند:

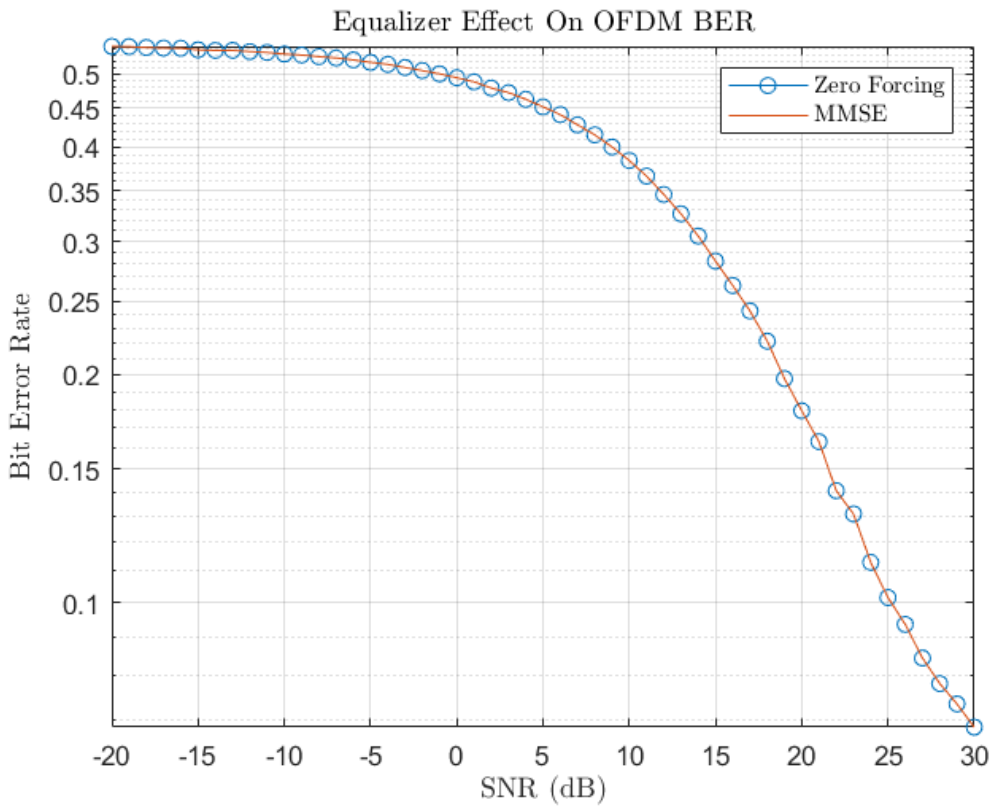
$$ZF:\ W_k = \frac{1}{H_k}$$

$$MMSE:\ W_k = \frac{H_k^*}{|H_k|^2 + N_0}$$

مطابق روابط فوق در کد ضرایب را تعریف می کنیم و پیش از تصمیم گیری در خروجی بلاک FFT ضرب می کنیم:

```
Wk_ZF = 1 ./ H;
Wk_MMSE = conj(H) ./ (abs(H) .^ 2 + N0);
xHatZF = real(Y .* Wk_ZF) > 0;
xHatMMSE = real(Y .* Wk_MMSE) > 0;
```

بقیه مراحل عینا مشابه است، نمودار bit error rate بر حسب snr مطابق زیر برای این دو روش بدست می آید:



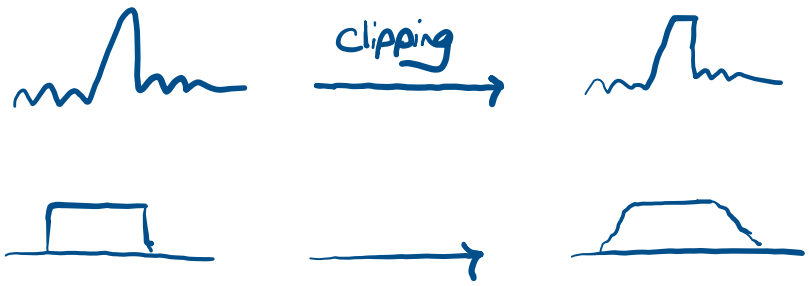
دیده می شود که این دو روش تقریبا عملکرد یکسانی دارند. انتظار می رود با افزایش اثر نویز در کانال های deep fade که $\frac{1}{|H_i|}$ ممکن است مقدار زیادی داشته باشد یا کاهش توان سمبل های ارسالی یعنی کاهش Pmax در ZF اثر noise enhancement باعث شود تا عملکرد MMSE بهتر باشد.

تمامی مرحل مطابق قبل است تنها در فرستنده پس از بلوک CP به جای ارسال مستقیم سمبل ها آن ها را به تابعی به نام Clipping که تعریف کرده ایم می دهیم و خروجی این تابع را ارسال می کنیم، این تابع مطابق زیر تعریف شده است:

```
function t = clipping(a, ratio)
    thr = ratio * max(abs(a));
    a(abs(a) > thr) = thr * exp(1j * angle(a(abs(a) > thr)));
    t = a;
end
```

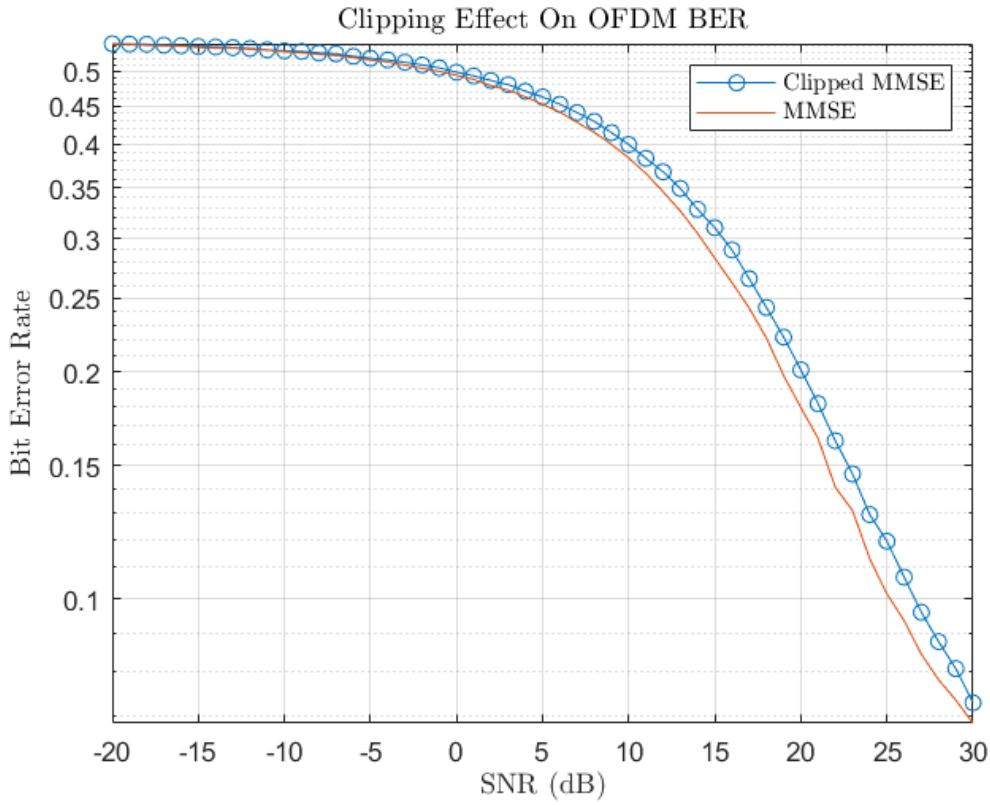
عملکرد این تابع به این نحو است که ابتدا با توجه به ورودی ضریب ratio ماکسیمم اندازه خروجی به نام thr را می یابد، سپس تمامی سمبل های ارسالی که بیش از این مقدار هستند را با مقدار thr جایگزین می کند ولی فاز آن ها را حفظ می کند و تغییر نمی دهد.

این کار باعث می شود به صورت کنترل شده قبل از ورود به high power amplifier سیگنال را فیلتر کنیم تا اثر مخرب HPA را کمی کاهش دهیم، البته همچنان in-band distortion و out of band radiation داریم که باعث می شود تا BER کمی افزایش یابد.

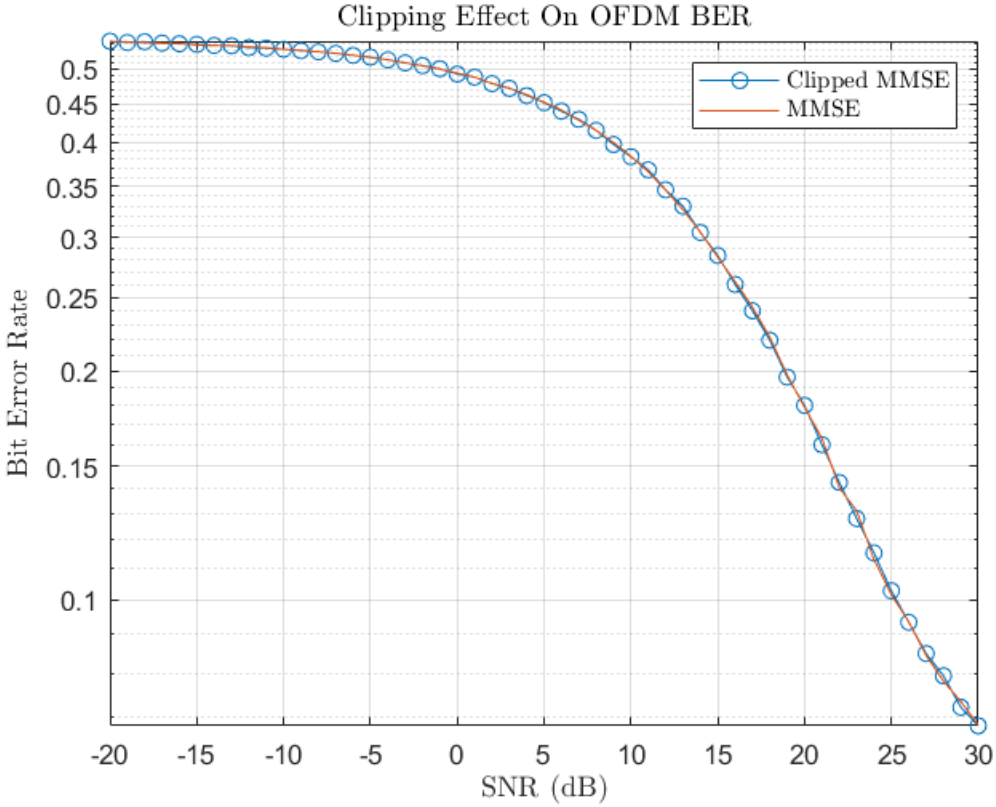


نمودار *BER* برای سیستم *OFDM* با استفاده از *MMSE Equalizer* با و بدون *clipping* :

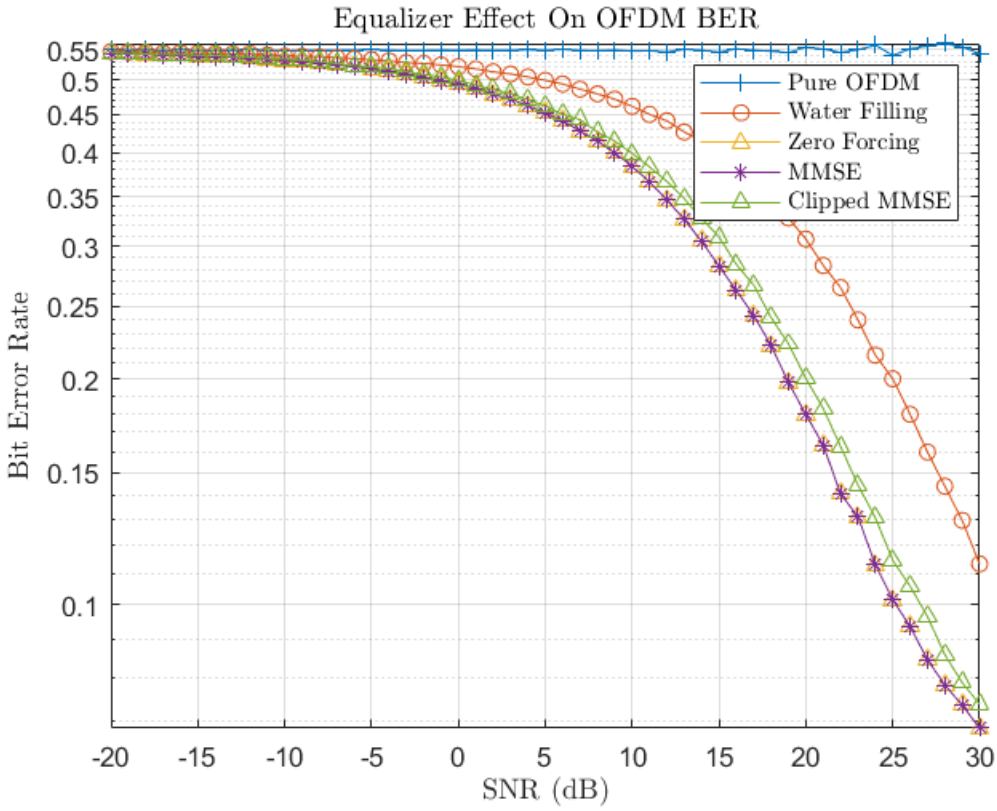
شایان ذکر است برای مشهود بودن اثر *clipping* مقدار *ratio* در کد *0.4* در نظر گرفته شده است.



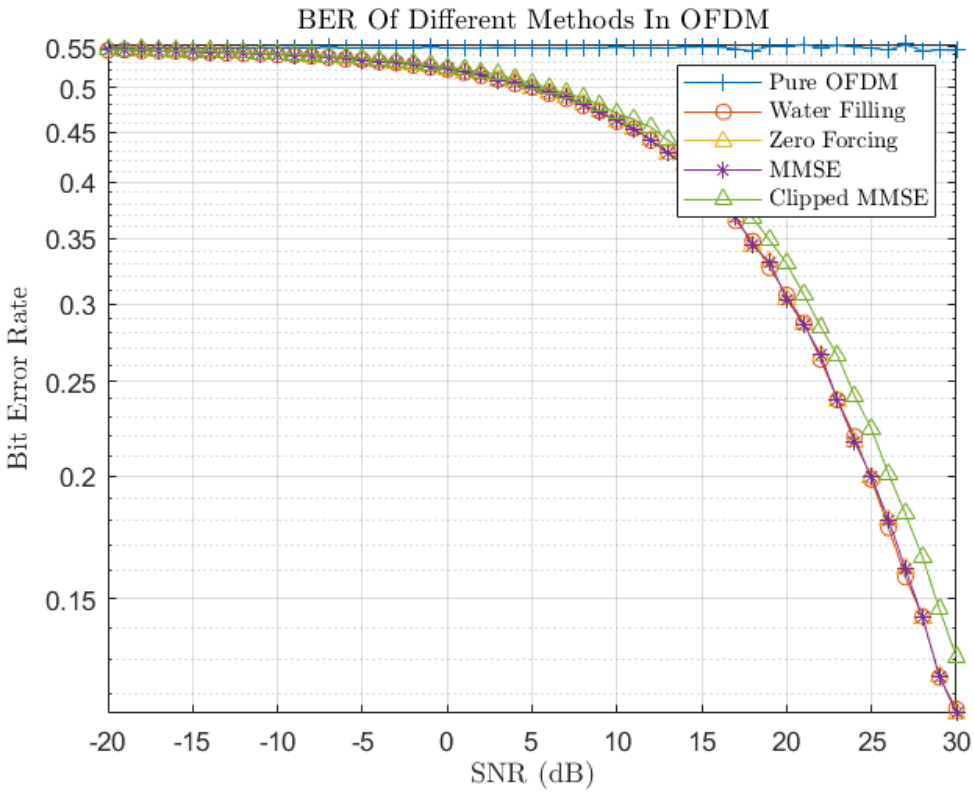
در حالتی که *ratio* را ۰.۸ در نظر بگیریم تفاوت دو نمودار چندان مشهود نیست:



در ادامه تمامی حالت های بررسی شده در یک نمودار رسم شده اند:



مشاهده می شود *waterfilling* نسبت به استفاده از *Equalizer* چه با استفاده از *clipping* و چه بدون آن عملکرد ضعیف تری دارد چرا که همان ابتدای ارسال با تقسیم سمبل ها بر *Hi* اندازه آن ها را کم می کند و باعث می شود اثر نویز روی سمبل ها بیش تر شود و خطای تشخیص بیت های دریافتی بالا رود، البته اگر اندازه کانال کم تر شود مثلا در کانال های *deep fade* و یا توان ارسالی یعنی *Pmax* را بیش تر کنیم عملکرد این روش ها نزدیک تر می شود:



در تصویر فوق *Pmax* برابر *nc* در نظر گرفته شده است در حالی که در تصویر قبلی *Pmax* برابر *nc/4* در نظر گرفته شده بود.