به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

مخابرات بیسیم

استاد صباغیان

گزارشکار پروژه اول

فاطمه جلیلی
۸۱۰۱۹۹۳۹۸
تاریخ تحویل : ۱۴۰۳/۰۲/۲۵

سؤال ۱

الف)

ابتدا پارامتر های سوال را مطابق زیر تعریف و مقدار دهی می کنیم:

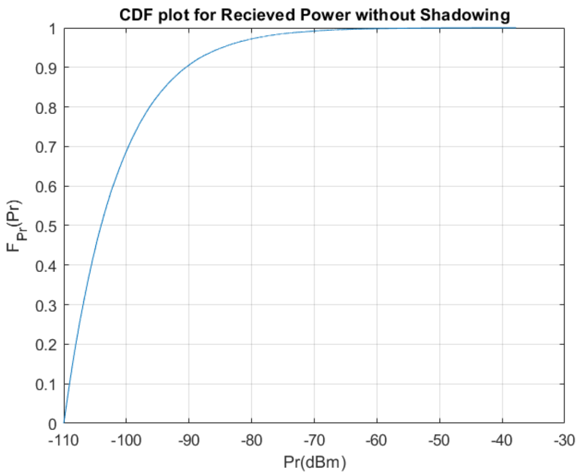
```
numUser = 1e5;
d0 = 10;
D = 1000;
n = 4;
Pr_d0 = 10*log10(1e-6) + 30; %dBm
N0 = 10 ^ (-175 / 10) * 1e-3;
BW = 1e6;
```

فاصله ی کاربران را به صورت رادیکال توزیع یونیفرم بین d0 تا D مطابق زیر تعریف کرده و آن ها sort می کنیم:

```
d = sort((D-d0) * sqrt(rand(numUser, 1)) + d0);
```

دقت شود از sqrt(rand(...)) برای این استفاده شده است که تراکم توزیع کاربران در سراسر دایره اطراف BS یکسان باشد، در صورتی که اینکار انجام نشود گویی تراکم کاربران نزدیک BS بیش تر در نظر گرفته شده است.

سپس با استفاده از فرمول داده شده میانگین توان سیگنال دریافتی برای هر کاربر را بدست می آوریم و با استفاده از تابع cdfplot تابع توزیع تجمعی میانگین توان سیگنال دریافتی را رسم می کنیم:

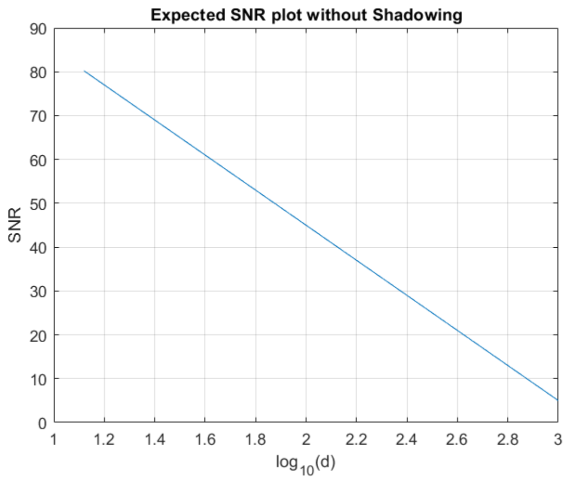


ب)

پس از محاسبه ی توان نویز در واحد لگاریتمی برحسب dBm مطابق رابطه زیر

```
Pn = 10 * log10(N0 * BW * 1e3);
```

مقدار SNR را بر اساس اختلاف بردار میانگین توان سیگنال دریافتی که در مرحله قبل بدست آوردیم با توان نویز محاسبه می کنیم و بر حسب فاصله از مرکز در مقیاس لگاریتمی رسم می کنیم. همانطور که انتظار داشتیم مقدار SNR با افزایش مقدار 10log10(d) که موجب افزایش تلف مسیر می شود به صورت خطی کاهش می یابد.

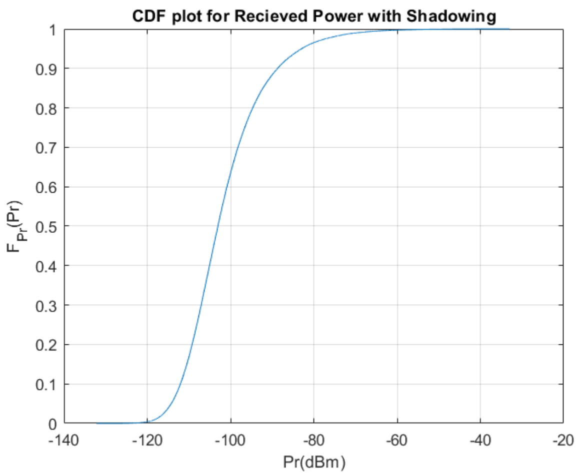


ج)

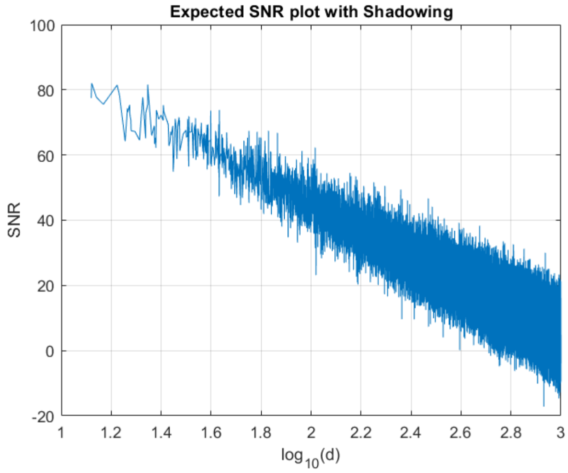
متغیر تصادفی گوسی X را با میانگین و انحراف معیار خواسته شده مطابق زیر تعریف می کنیم و میانگین توان سیگنال دریافتی با در نظر گیری اثر سایه از مجموع X و آنچه برای میانگین توان سیگنال دریافتی بدون در نظر گیری X بدست آمده بود محاسبه می شود:

```
sigma = 5;
mu = 0;
X_dB = mu + sigma * randn(numUser, 1);
Pr_shadow = Pr_noShadow + X_dB;
```

مطابق قبل با کمک تابع cdfplot تابع توزیع تجمعی میانگین توان سیگنال دریافتی را رسم می کنیم، مشاهده می شود نمودار مشابه حالت قبل است و فقط حالت آن کمی به تابع توزیع تجمعی متغیر گوسی نزدیک شده است.



مشابه قسمت قبل نمودار SNR بر حسب 10log10(d) را نیز رسم می کنیم، مشاهده می شود با در نظر گیری اثر سایه دیگر نمودار کاملا خطی نیست و با افزایش فاصله میزان نویز اضافه شده به نمودار هم اثر خود را بیش تر نشان می دهد:



د)

برای محاسبه ی احتمال خاموشی در مدل log normal داریم:

$$P_{out} = P_r\{SNR(d) < SNR_{min}\} = P_r\{P_r(d) - P_n < SNR_{min}\}$$

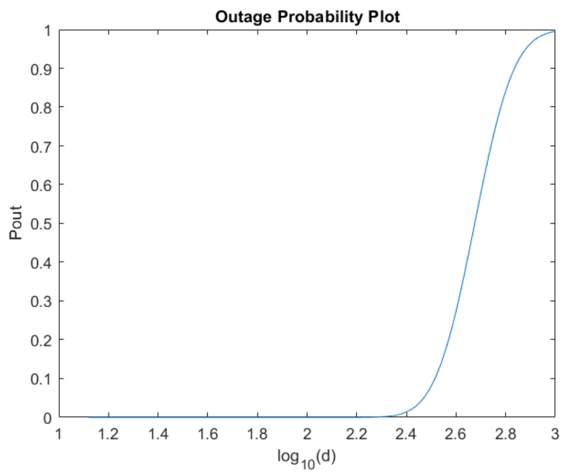
$$= P_r\left\{P_0 + 10\log_{10} k - 10\gamma\log_{10}\frac{d}{d0} + X - P_n < SNR_{min}\right\}$$

$$= P_r\left\{X < SNR_{min} + P_n - P_0 + 10\gamma\log_{10}\frac{d}{d0}\right\}$$

$$= P_r\{X < SNR_{min} - SNR_{no\ shadow}(d)\} = 1 - Q\left(\frac{SNR_{min} - SNR_{no\ shadow}(d)}{\sigma}\right)$$

با استفاده از رابطه فوق احتمال خاموشی به ازای $SNR_{min} = 18\ dB$ را مطابق کد زیر محاسبه و رسم می کنیم:

```
SNR_min = 18; %dB
Pout = 1 - qfunc((SNR_min - SNR) / sigma);
```



همانطور که انتظار داشتیم به دلیل وجود اثر سایه در فاصله معینی احتمال خاموشی از صفر شروع به افزایش شبیه توزیع گوسی می کند و زمانی که به حد کافی از BS دور شویم به ۱ می رسد.

۵)

محاسبه ی عملی در متلب:

تعداد کاربرانی که که در آن ها مقدار SNR آن ها از SNRmin کم تر است را پیدا می کنیم و نسبت به تعداد کل کابران نسبت را بدست می آوریم ، از آنجایی که کاربران به صورت یکنواخت رو مساحت پخش شده اند درصد کابرانی که SNR کافی دارند معادل درصد مساحتی است که تحت پوشش BS است:

```
Ncoverange = sum(SNR_shadow > SNR_min);
C3 = Ncoverange / numUser;
s_coverage_sim2 = pi * C3 * D ^ 2;
```

به این ترتیب بدست می آید:

```
Based on Simulation Results, Coverage area of BS is 823097.275241 where C = 0.262000
```

محاسبه تئوری:

مطابق آنچه در کتاب Goldsmith آمده است داریم:

$$P_r(D) = P_0 - 10n\, log_{10}\left(\frac{D}{d_0}\right), \quad a = \frac{P_{r,min} - P_r(D)}{\sigma} = , \quad b = \frac{10nlog_{10}(e)}{\sigma}$$

$$\rightarrow C = Q(a) + \exp\left(\frac{2 - 2ab}{b^2}\right) Q\left(\frac{2 - ab}{b}\right)$$

$$\rightarrow S = \pi D^2 C$$

روابط فوق را مطابق کد زیر در متلب می نویسیم :

```
Pr_min = SNR_min + (N0_dBm + 10 * log10(BW));
Pr_D = Pr_d0 - 10 * n * log10(D / d0);
a = (Pr_min - Pr_D) / sigma;
b = 10 * n * log10(exp(1)) / sigma;
C2 = qfunc(a) + exp((2 - 2 * a * b) / b^2) * qfunc((2 - a * b) / b);
s_coverage_theor = pi * C2 * D ^ 2;
```

به این ترتیب بدست می آید:

```
Based on Theoritical Results, Coverage area of BS is 826877.590298 where C = 0.263203
```

مشاهده می شود مساحت ناحیه تحت پوشش که از راه شبیه سازی و تئوری بدست آمده بسیار نزدیک هستند و اختلاف جزیی آن ها را می توان با توزیع رندوم کابران توجیه کرد ، در واقع تعداد کاربران محدود است و این دو مقدار برای هر چه بیش تر تعداد کابران نزدیک و نزدیک تر می شوند.

سؤال ۲

الف)

ابتدا پارامتر های سوال را مطابق کد زیر تعریف می کنیم:

```
N = 15;
fc = 3e9;
c = 3e8;
lambda = c / fc;
v = 30;
simulationTimes = 1e5;
```

سپس در یک حلقه for هر بار مطابق آنچه گفته شده θ و τ را به صورت یکنواخت در بازه های گفته شده تعریف می کنیم در ادامه ri و rq را با توزیع نرمال و انحراف معیار σ تعریف می کنیم ، مطابق درس می دانیم با این تعریف توزیع $\alpha = \sqrt{r_i^2 + r_q^2}$ رایلی خواهد بود.

شیفت فرکانس داپلر را بر اساس فرمول $f_D = \frac{v}{\lambda} cos\theta$ بدست می آوریم.

مطابق درس برای پاسخ ضربه کانال Rayleigh Fading داریم:

$$h(\tau, t) = \sum_{i=1}^{N} \alpha_i(t)\, e^{-j\phi_i(t)}\, \delta(\tau - \tau_i) \quad (*)$$

$$\phi_i(t) = 2\pi f_c \tau_i - \varphi_{Di} = 2\pi f_c \tau_i - \int 2\pi f_{Di}\, dt = 2\pi f_c \tau_i - 2\pi f_{Di} \tau_i = 2\pi (f_c - f_{Di})\, \tau_i$$

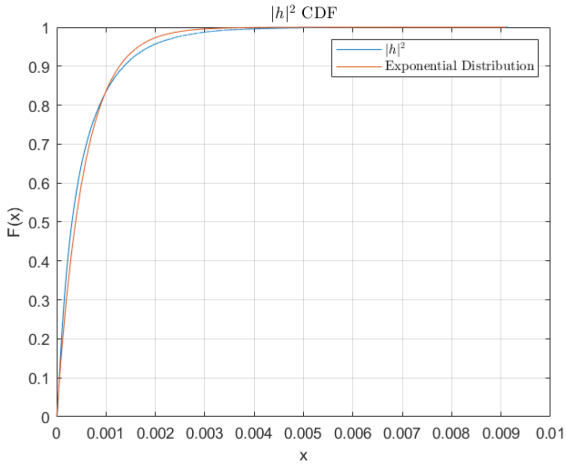
بنابراین مطابق روابط فوق و آنچه توضیح داده شد مطابق کد زیر کانال را ۱۰۰۰۰۰ باز شبیه سازی می کنیم و هربار مقدار توان آن را که از مجموع توان دو درایه های h بدست می آید را ذخیره می کنیم.

```
for iter = 1 : simulationTimes
    theta = 0.5 * pi * rand(N, 1);
    tau = 9 * rand(N, 1) + 1;
    sigma = sqrt(1e-3 * tau .^ (-4) / 2);
    ri = sigma .* randn(N, 1);
    rq = sigma .* randn(N, 1);
    alpha = abs(ri + 1i * rq);
    fD = v * cos(theta) / lambda;
    phi = 2 * pi * (fc - fD) .* tau * 1e-6;
    h = alpha .* exp(-1i * phi);
    power(iter) = sum(abs(h) .^ 2);
end
```

در نهایت مقدار $E\{|h^2|\}$ را از میانگین گیری از درایه های آرایه ی $power$ بدست می آوریم:

```
E{|h|^2} = 0.000552
```

برای رسم تابع توزیع تجمعی از تابع *cdfplot* متلب استفاده می کنیم، همزمان برای مقایسه یک توزیع نمایی نیز به نمودار فیت کرده و آن را نیز رسم می کنیم:



همانطور که بالا توضیح داده شد α دارای توزیع رایلی است بنابراین $|h|$ هم مطابق رابطه ای که بالاتر نوشتیم $(*)$ توزیع مشابه α دارد.

فرض کنیم $|h|^2 = Y, |h| = X$:

$$F_Y(y) = \Pr(Y \le y) = \Pr(X^2 \le y) = \Pr\left(X \le \sqrt{y}\right) = \int_0^{\sqrt{y}} \frac{x}{\sigma^2} e^{-x^2/(2\sigma^2)}\, dx = 1 - e^{-\frac{y}{2\sigma^2}}$$

بنابراین $|h|^2$ توزیع نمایی خواهد داشت که مطابق تصویر فوق هم تابع توزیع تجمعی آن به خوبی با تابع توزیع تجمعی نمایی مطابقت دارد.

ب)

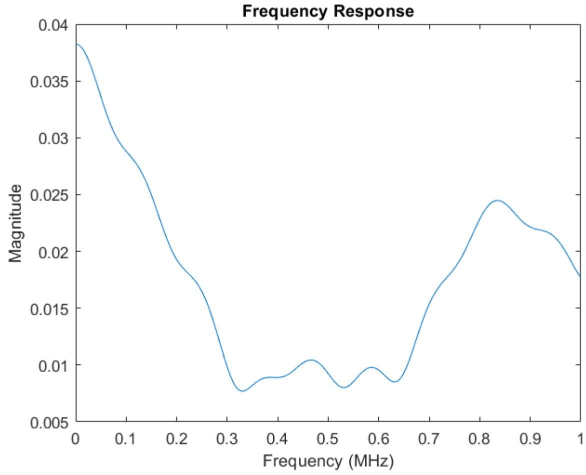
رابطه ی $(*)$ را در حوزه فرکانس می توان به صورت زیر نوشت:

$$h(\tau, t) \xrightarrow{\mathrm{F}} H(f) = \sum_{i=1}^{N} \alpha_i(t)\, e^{-j\phi_i(t)}$$

دقیقا مشابه آنچه درون حلقه for در قسمت پیش نوشته بودیم ri, rq و از روی آن ها α را محاسبه می کنیم و مطابق رابطه ی فوق در یک حلقه for سیگما را پیاده سازی می کنیم و $H(f)$ را رسم می کنیم:

```
theta = 0.5 * pi * rand(N, 1);
tau = 9 * rand(N, 1) + 1;
sigma = sqrt(1e-3 * tau .^ (-4) / 2);
ri = sigma .* randn(N, 1);
rq = sigma .* randn(N, 1);
alpha = abs(ri + 1i * rq);
fD = v * cos(theta) / lambda;
freq = (0 : 1e6-1).';
H = zeros(length(freq), 1);
```

```
for i = 1 : N
    phi = 2 * pi * (freq - fD(i)) .* tau(i) * 1e-6;
    H = H + alpha(i) * exp(-1i * phi);
end
```



مطابق شکل فوق اندازه پاسخ فرکانسی کانال در هر فرکانس متفاوت است بنابراین کانال فرکانس گزین است چرا که کانال از نوع چند مسیره و wideband است.

برای اینکه در هر بار شبیه سازی پاسخ تغییر نکند از rng استفاده شده است، در غیر اینصورت با توجه به تعریف تصادفی متغیر های مسئله هر بار شکل پاسخ متفاوت خواهد بود.

سؤال ۳

داریم:

$$\mu_m(averange\ delay\ spread) = \frac{\int_0^\infty \tau R_c(\tau)\, d\tau}{\int_0^\infty R_c(\tau)\, d\tau}$$

$$\sigma_m(rms\ dely\ spread) = \sqrt{\frac{\int_0^\infty (\tau - \mu_m)^2 R_c(\tau)\, d\tau}{\int_0^\infty R_c(\tau)\, d\tau}}$$

همچنین می دانیم:

$$PDP \triangleq R_c(\tau, \Delta t)|_{\Delta t = 0} = R_c(\tau)$$

روابط فوق را به *symbolic* تعریف می کنیم تا μ_m, σ_m بدست آوریم:

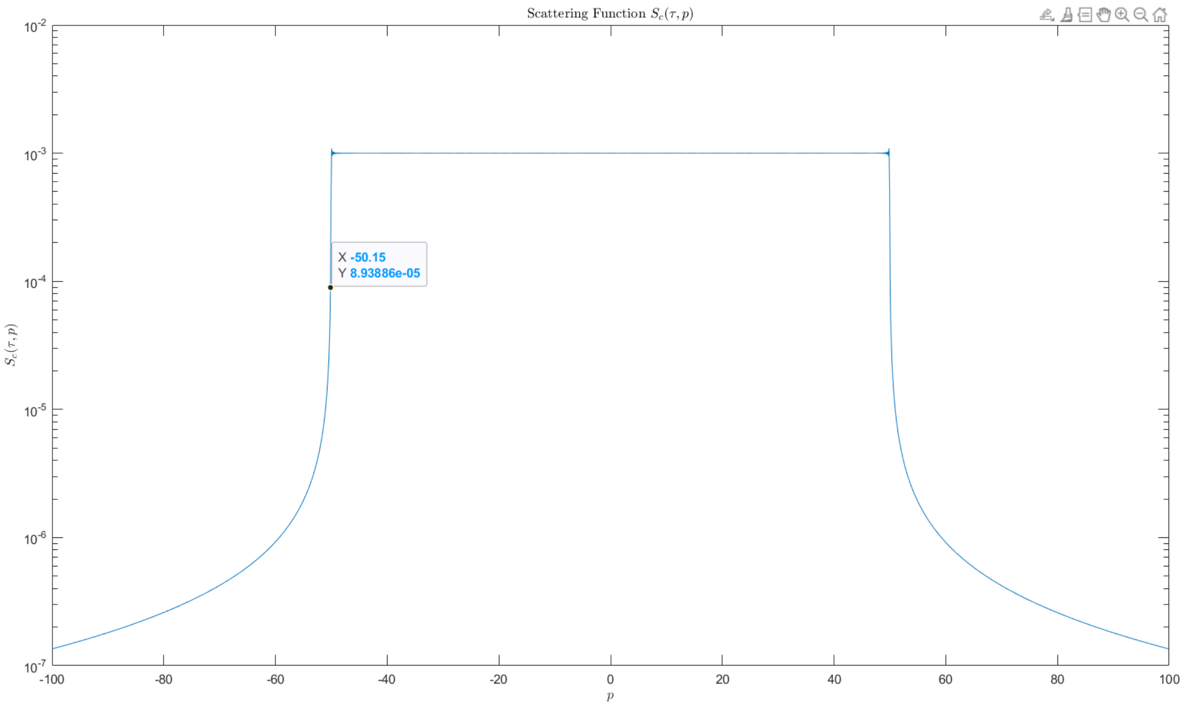
```
syms tau;
syms delta_t;


PDP = double(sinc(0 * delta_t));
mu_m = double(int(tau * PDP, tau, 0, 10) / int(PDP, tau, 0, 10));
sigma_m = double(sqrt(int((tau - mu_m) ^ 2 * PDP, tau, 0, 10) / int(PDP, tau, 0,
10)));
```

بدست می آید:

```
Averange Power Delay Spread = 5.000000
RMS Power Delay Spread = 2.886751
```

با توجه به فرکانس نمونه برداری داده شده Δt و محور فرکانس که قرار است به ازای آن ها تبدیل فوریه را بدست آوریم را تعریف می کنیم. Δt را بین -۵ تا ۵ در نظر می گیریم، هر چه این بازه بزرگ تر باشد تبدیل فوریه دقیق تر خواهد بود و به شکل پالس با اثر کمتر پدیده گیپس نزدیک تر می شویم. پس از شیفت تبدیل فوریه روی ۰ با *fftshift* و نرمالایز کردن آن ، تابع *scattering* را رسم می کنیم. دقت شود چون *Rc* تابعیت τ ندارد در یک بعد رسم شده است و به ازای بازه ی که در آن τ تعریف شده است این شکل تکرار می شود. برای محاسبه ی پهنای باند *10dB* محور عمودی لگاریتمی در نظر گرفته شده است:



مطابق تصویر فوق که نقطه ای که روی آن اندازه تابع *scattering* حدودا *10dB* از ماکسیمم خود کم تر می شود مشخص شده است ، پهنای باند *10dB* دو برابر این طول این نقطه و برابر ۱۰۰.۳ خواهد بود . دقت شود هر چه بازه Δt را بزرگتر بگیریم این تبدیل فوریه به پالس کامل نزدیک تر شده و پهنای باند *10dB* به طول پالس یعنی ۱۰۰ نزدیک تر می شود.

پهنای باند *10dB* می تواند نشانگر اثر داپلر باشد، در واقع هر چه این پهنای باند بزرگ تر باشد یعنی محیط با سرعت بالاتری تغییر می کند. این پهنای باند به نوعی معیاری از گستردگی طیفی ناشی از نرخ زمانی تغییرات کانال است. پهنای باند های مختلفی اعم از *10dB*، *5dB* و خود پهنای باند داپلر وجود دارد که همگی معیاری برای اندازه گیری این موضوع هستند و بسته به کاربرد می توان از اینها استفاده کرد.

سؤال ۴

الف)

مطابق خواسته سوال از روش *gradient Ascent* استفاده می کنیم، به این منظور نیاز داریم مشتق تابع هدف را به متغیر هایی که می خواهیم روی آن ها تابع هدف ماکسیمم شود یعنی P_i ها را بیابیم، داریم:

$$f = \sum_i \log(R_i)$$

طبق تعریف SINR داریم:

$$\gamma_j = \frac{P_j G_{jj}}{P_n + \sum_{k \neq j} P_k G_{jk}}$$

$$\frac{\partial f}{\partial P_i} = \frac{\partial}{\partial P_i} \sum_j \log(R_j) = \sum_j \frac{\partial}{\partial P_i} \log(R_j) = \sum_j \frac{1}{R_j} \frac{\partial R_j}{\partial P_i} \xrightarrow{R_j = \log(1+\gamma_j)}$$

$$= \sum_j \frac{1}{R_j} \times \frac{1}{1+\gamma_j} \times \frac{\partial \gamma_j}{\partial P_i} =$$

$$\longrightarrow \text{if } i \neq j : \frac{\partial \gamma_j}{\partial P_i} = \frac{-P_j G_{jj}}{\underbrace{\left(P_n + \sum_{k \neq j} P_k G_{jk}\right)^2}_{A}} \times \frac{\partial A}{\partial P_i} = \frac{-P_j G_{jj} G_{ji}}{\left(P_n + \sum_{k \neq j} P_k G_{jk}\right)^2} \quad \gamma_j$$

$$= \frac{-\gamma_j G_{ji}}{P_n + \sum_{k \neq j} P_k G_{jk}} = \frac{\gamma_j}{G_{jj} P_j} \quad \frac{-\gamma_j^2 G_{ji}}{G_{jj} P_j}$$

$$\longrightarrow \text{if } i = j : \frac{\partial \gamma_j}{\partial P_j} = \frac{G_{jj}}{P_n + \sum_{k \neq j} P_k G_{jk}} = \frac{\gamma_j}{P_j}$$

$$\xrightarrow{\text{جایگذاری}} \frac{\partial f}{\partial P_i} = \underbrace{\frac{1}{R_j} \times \frac{1}{1+\gamma_j} \times \frac{\gamma_j}{P_j}}_{\text{step } j} - \underbrace{\sum_{i \neq j} \frac{1}{R_j} \times \frac{1}{1+\gamma_j} \times \frac{\gamma_j^2 G_{ji}}{P_j G_{jj}}}_{\text{step } i}$$

- در کد جای اندیس i, j برعکس محاسبات تئوری فوق است که البته تفاوتی نمی کند، توضیحاتی که در ادامه آمده بر حسب اندیس های کد متلب است.

به ترتیب روابط فوق مقادیر $SNRi$، Ri، $stepi$ و $stepj$ را بدست می آوریم، برای پیاده سازی مخرج SNR که سیگما دارد ولی برای اندیس های غیر یکسان ، درایه متناظر P و G را حذف می کنیم و یکی را در ترنزپوز دیگری ضرب می کنیم تا سیگما مخرج درست شود . برای پیاده سازی سیگما استفاده شده در $stepj$ از یک حلقه *for* استفاده می کنیم و اگر اندیس ها مخالف باشند حاصل عبارت جلو سیگما را به ازای j خاص حساب کرده و با مقادیر قبلی جمع می کنیم:

از آنجا که مشتق تابع هدف باید نسبت به همه Pi ها حساب شود و همگی هر بار آپدیت شوند تمامی کار های فوق را در یک حلقه *for* انجام می دهیم:

```
for i = 1 : length(Pmax)
            Pj = P;
            Pj(i) = [];
            Gij = G(i, :);
            Gij(i) = [];
            SINR(i) = P(i) * G(i, i) / (N0(i) + Pj * Gij.');
            R(i) = log(1 + SINR(i));
            stepi = SINR(i) / (R(i) * P(i) * (SINR(i) + 1));
            stepj = 0;
            for j = 1 : length(Pmax)
                if j ~= i
                    stepj = stepj + ...
                        SINR(j) ^ 2 * G(j, i) / (G(j, j) * R(j) * P(j) * (SINR(j)
+ 1));
                end
            end
```

سپس مطابق GA مقادیر Pi را اپدیت می کنیم، مقدار ضریب آلفا که در آپدیت استفاده می شود ۰.۱ تنظیم شده است. طبق روابط تئوری فوق مقدار مشتق تابع هدف به Pi از اختلاف $stepj$, $stepi$ که در قطعه کد فوق بدست آوردیم بدست می آید، پس از آپدیت چک می کنیم اگر مقدار جدید Pi شرط $Pmax$ را اغنا نکند و در واقع از مقدار حد ماکسیمم خود بیش تر شود ، مقدار جدید Pi همان $Pmaxi$ در نظر گرفته می شود:

```
Pi_next = P(i) + alpha * (stepi - stepj);
            if Pi_next > Pmax(i)
                Pi_next = Pmax(i);
            end
```

در هر حلقه مقدار فاصله ی کل بردار P با مقدار قبلی آن ذخیره می شود و تمامی کار های فوق در حلقه *while* تا زمانی که این این فاصله نرمالایز شده به P از حدی کم تر شود ادامه پیدا می کند ، این مقدار حدی $1e\text{-}4$ در نظر گرفته شده است. نرمالایز کردن فاصله به این خاطر است که با یک *threshold* ثابت در دیتاست های مختلف بتوانیم همگرا شویم.

```
dist(i) = P(i) - Pi_next;
```

همچنین مقدار تابع هدف را هم در هر حلقه `while` برای رسم نتایج در ادامه ذخیره می کنیم:

```
fR = cat(2, fR, sum(log(R)));
```

برای دیتاست های مختلف فایل ها از روی *repository* خوانده می شود و ماتریس ها درون *cell* های همنام ذخیره می شوند و در ادامه هر چه بالاتر گفته شده است در یک حلقه *for* کلی برای همه دیتاست ها تکرار می شود.

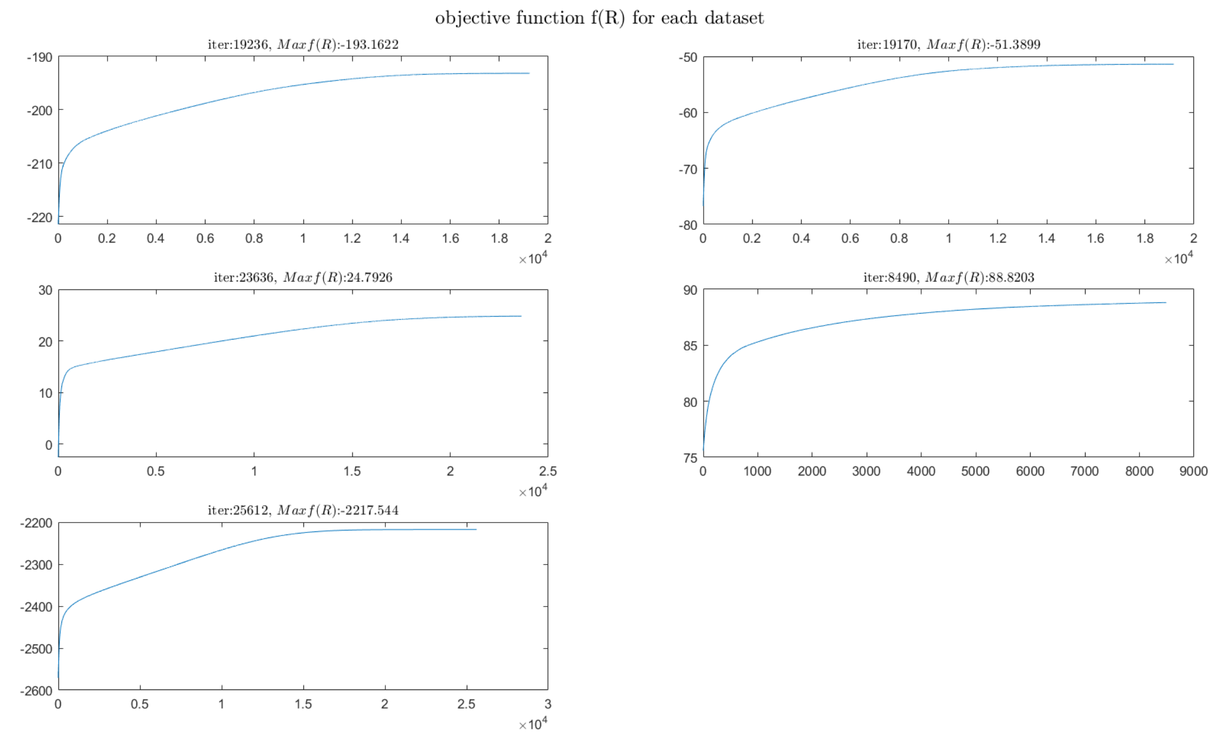
مقدار *Pi initialization* برابر نصف مقدار *Pimax* در نظر گرفته شده است.

تابع هدف تابعی لگاریتمی است، توابع لگاریتمی خوش رفتار و کانوکس هستند به این معنا که اگر *alpha* مطلوبی انتخاب کنیم حتما الگوریتم همگرا خواهد شد.

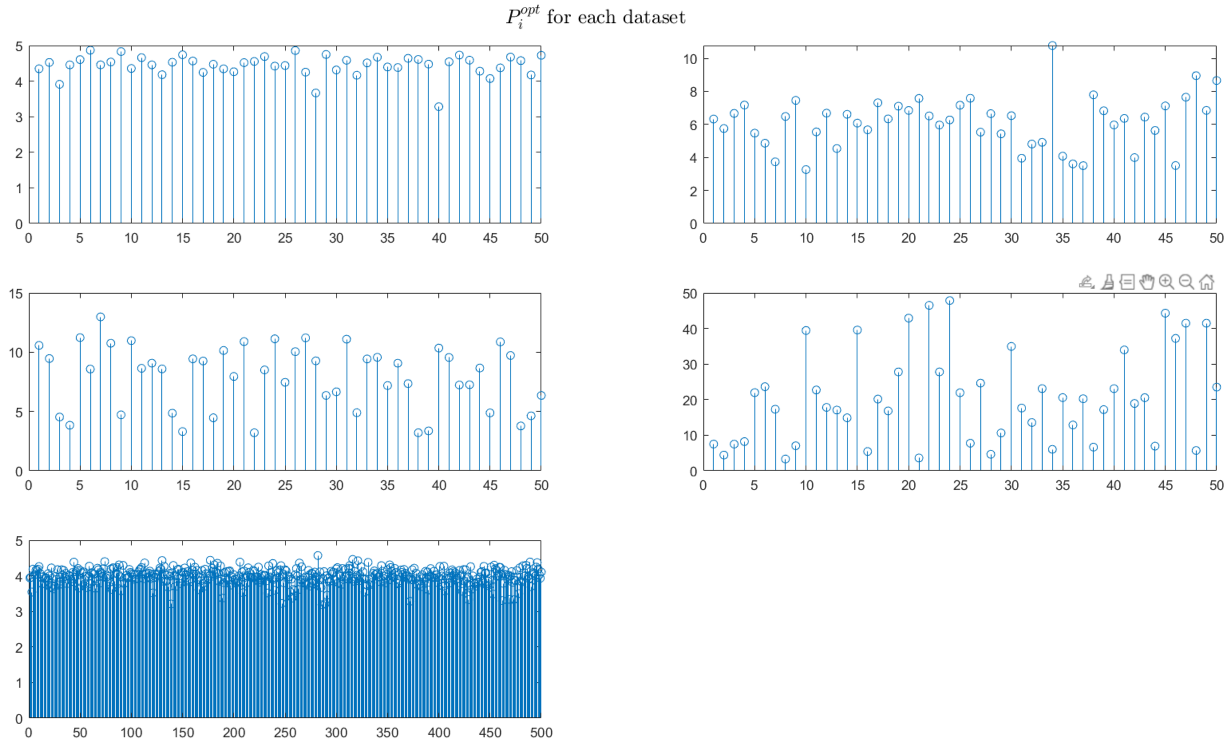
اگر مقدار *alpha* از حدی بزرگ تر باشد الگوریتم مطابق تصاویر فوق همگرا نمی شود ، همچنین اگر مقدار *threshold* بسیار کوچک در نظر گرفته شود تعداد تکرار حلقه های الگوریتم بسیار زیاد می شود و باید *alpha* بسیار کوچک انتخاب شود تا پس از زمان طولانی الگوریتم همگرا شود وگر نه مثل تصویر فوق سمت راست به مقدار اکسترمم نزدیک شده ولی مجدد از آن دور خواهیم شد چرا که *alpha* به قدر کافی کوچک نیست.

ب)

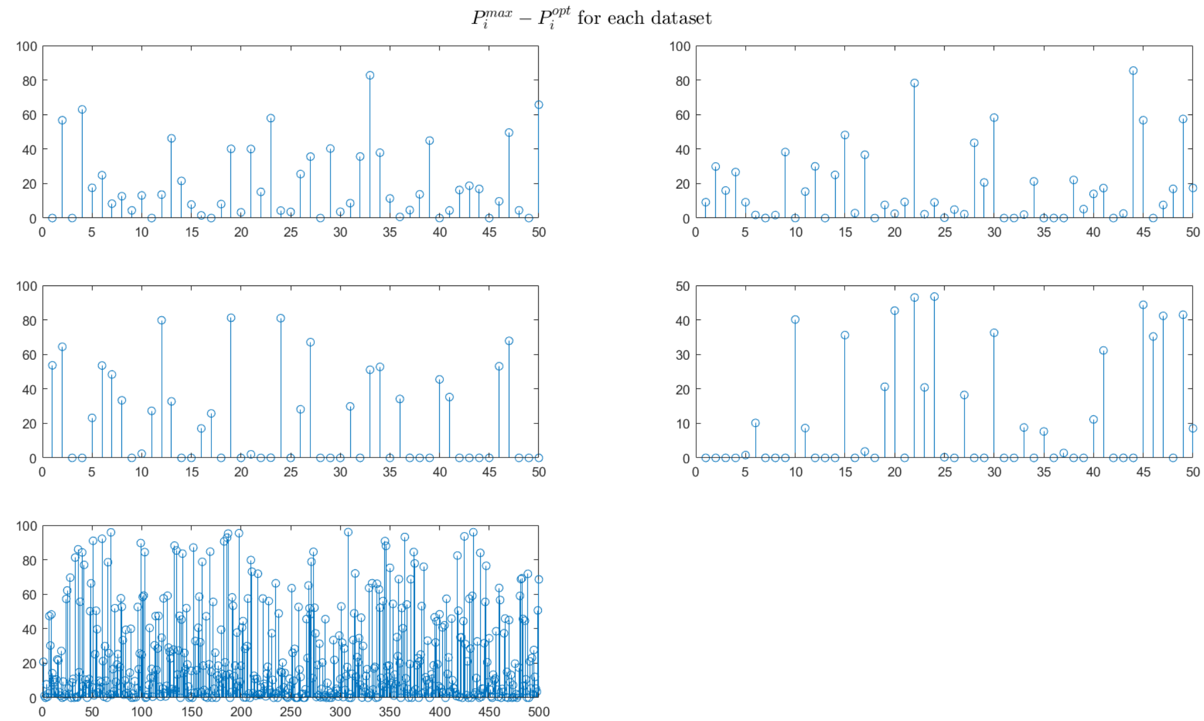
در تصویری که در ادامه آمده است مقادیر تابع هدف در هر تکرار ، تعداد تکرار ها و مقدار ماکسیمم تابع هدف به ازای هر دیتاست آمده است:



در تصویر که در ادامه آمده مقدار P_i^{opt} برای هر کاربر به ازای دیتاست های مختلف آمده است:



در تصویر که در ادامه آمده مقدار $P_i^{max} - P_i^{opt}$ برای هر کاربر به ازای دیتاست های مختلف آمده است:



یک بار محاسبات فوق را به ازای $Pi = Pimax$ تکرار می کنیم و مقادیر تابع هدف و توان ارسالی برای دیتاست های مختلف درون یک آرایه ذخیره می کنیم:

```
for i = 1 : length(Pmax)
            Pj = P;
            Pj(i) = [];
            Gij = G(i, :);
            Gij(i) = [];
            SINR_max(i) = P(i) * G(i, i) / (N0(i) + Pj * Gij.');
            R_max(i) = log2(1 + SINR_max(i));
    end
    fR_max(datasetNum) = sum(log2(R_max));
    Pt_max(datasetNum) = sum(Pmax);
```

سپس مقادیر بهبود تابع هدف و صرفه جویی در توان ارسالی در مقایسه استفاده از P_i^{opt} به جای P_i^{max} برای هر دیتاست مطابق زیر بدست می آید:

```
objective function improvement for dataset 1 is 27.783139
Savings in transmission power for dataset 1 is 994.511817
-----------------------------------
objective function improvement for dataset 2 is 25.362350
Savings in transmission power for dataset 2 is 855.218639
-----------------------------------
objective function improvement for dataset 3 is 27.327010
Savings in transmission power for dataset 3 is 1090.606966
-----------------------------------
objective function improvement for dataset 4 is 13.235351
Savings in transmission power for dataset 4 is 559.655023
-----------------------------------
objective function improvement for dataset 5 is 343.132141
Savings in transmission power for dataset 5 is 11832.288067
-----------------------------------
```

پ)

دیتاست اول و دوم در توزیع $P_i^{max} - P_i^{opt}$ تقریبا یکسان هستند لذا مقادیر بهبود تابع هدف در مقایسه استفاده از P_i^{opt} به جای P_i^{max} برای هر دو تقریبا یکسان است ولی به طور کلی $Popt$ بدست آمده برای دیتاست دوم سطح پایین تری نسب به دیتاست اول دارند، دلیل این امر درون ماتریس G نهفته است، مقادیر تداخل یا به عبارتی اندازه درایه های غیر قطری در مقایسه با درایه های قطری تعیین کننده هستند که تابع هدف چه قدر می تواند پیشروی کند.

همانطور که در تصویر زیر دیده می شود دیتاست چهارم به بهترین حد تابع هدف در مقایسه با دیگر دیتاست ها رسیده است ؛ همانطور که دیده می شود مقادیر درایه های قطری این ماتریس بزرگ تر از درایه های غیر قطری آن در مقایسه با دیگر دیتاست ها است:

	1	2	3	4	5	6	7	8	9	10	11	12
1	1.4675	5.4847e-04	4.9359e-05	1.7470e-04	2.7262e-04	2.7765e-04	0.0017	5.4215e-04	5.7324e-04	9.0979e-05	5.9021e-05	3.7276e-04
2	0.0011	1.4887	6.1147e-05	5.0588e-05	2.6767e-04	4.8818e-05	4.8779e-05	2.9007e-04	5.4800e-04	0.0019	0.0012	1.4884e-04
3	7.8576e-05	0.0015	1.5889	2.8849e-04	0.0018	2.4681e-04	6.5897e-05	2.2206e-04	0.0012	5.6421e-04	0.0018	2.5695e-04
4	2.9231e-04	6.0464e-05	9.5874e-04	1.3508	6.4263e-04	5.3810e-05	6.4384e-04	1.3944e-04	0.0030	0.0015	0.0023	0.0022
5	0.0015	3.7953e-05	7.9926e-04	0.0025	1.1932	0.0019	8.3209e-04	3.6934e-05	1.9084e-04	0.0013	2.5028e-04	2.3214e-04
6	4.4562e-05	5.6841e-04	0.0016	0.0010	3.4528e-05	1.9330	1.1316e-04	1.9196e-04	0.0030	5.4000e-05	0.0020	0.0020
7	2.0713e-04	0.0025	0.0017	9.6815e-05	8.3256e-05	4.2194e-05	1.8924	3.2674e-04	7.0765e-04	0.0011	3.2520e-04	4.2815e-05
8	1.5348e-04	5.7514e-04	2.2480e-04	2.1042e-04	5.6175e-05	3.4953e-05	0.0030	1.0440	6.2060e-05	2.5148e-04	4.4545e-05	0.0031
9	5.6540e-04	0.0018	0.0010	0.0015	0.0028	1.4074e-04	3.1701e-04	2.3636e-04	1.0193	5.7539e-05	2.3202e-04	8.1976e-05
10	5.7227e-05	2.0320e-04	0.0011	6.0573e-04	0.0014	1.0062e-04	3.8376e-04	9.0165e-05	4.8544e-05	1.5104	0.0012	8.9326e-05
11	3.4834e-04	6.5059e-04	2.7078e-04	6.3268e-05	4.1969e-05	5.2860e-05	6.2429e-05	6.6764e-04	6.2479e-05	7.4890e-04	1.8647	2.0205e-04
12	7.4102e-05	0.0016	2.7605e-04	2.1724e-04	1.2678e-04	0.0017	0.0031	4.7321e-05	6.1760e-05	4.8664e-04	5.1226e-05	1.2916

در مقابل دیتاست پنجم بدترین مقدار تابع هدف را دارد، دلیل این امر با توجه به ماتریس G آن که در ادامه آمده قابل توجیه است، تداخل آن قدر زیاد است که توان Pi ها بیش از یه حدی بالاتر نمی روند:

	1	2	3	4	5	6	7	8	9	10	11	12
1	1.2633	0.2184	0.2977	0.0140	0.0276	0.0346	0.0638	0.0229	0.2190	0.0458	0.0957	0.0248
2	0.0185	1.2610	0.0161	0.0144	0.0787	0.0668	0.0987	0.1788	0.1334	0.0319	0.1492	0.0347
3	0.2277	0.0306	1.6408	0.0644	0.1254	0.1592	0.0152	0.0195	0.0195	0.0221	0.1971	0.2061
4	0.0247	0.1323	0.1028	1.8651	0.0332	0.0263	0.2431	0.0573	0.2004	0.0491	0.2421	0.1454
5	0.0579	0.0112	0.0788	0.0731	1.2961	0.0375	0.0285	0.0136	0.0205	0.0828	0.0157	0.1208
6	0.1961	0.0478	0.0861	0.0125	0.0334	1.1469	0.0235	0.0291	0.0150	0.0128	0.0326	0.0107
7	0.0308	0.0137	0.0795	0.0133	0.0179	0.0443	1.2587	0.0158	0.0666	0.0329	0.0423	0.0809
8	0.0137	0.0377	0.0112	0.0780	0.2882	0.0913	0.0241	1.5307	0.2402	0.0595	0.2200	0.2368
9	0.1454	0.0162	0.0552	0.0110	0.1238	0.0380	0.0348	0.0935	1.8553	0.0529	0.0275	0.1758
10	0.2304	0.0603	0.0548	0.0870	0.0137	0.2754	0.1485	0.0571	0.0755	1.4034	0.0120	0.0693
11	0.2428	0.0729	0.0129	0.0771	0.0863	0.0130	0.0303	0.0511	0.1688	0.0195	1.6168	0.1919
12	0.0271	0.0177	0.1619	0.2016	0.0476	0.1304	0.0315	0.0216	0.0151	0.0152	0.1870	1.0158

بنابراین اگر درایه های قطری در مقایسه با درایه های غیر قطری خیلی بزرگ تر باشند ، تداخل کم خواهد بود و می توان با افزایش توان کاربر ها به مقدار $Pimax$ رسید و تابع هدف هم بیش ترین مقدار را پیدا کند.