بسم الله الرحمن الرحیم

دانشگاه صنعتی اصفهان

دانشکده مهندسی برق و کامپیوتر

درس پردازش سیگنال‌های تصویری

تکلیف پنجم

استاد درس:

دکتر صدری

نام اعضای گروه (به ترتیب حروف الفبا):

ریحانه‌سادات ابطحی

فرزانه کوهستانی

محمدجواد نقدی

پاییز ۱۴۰۱

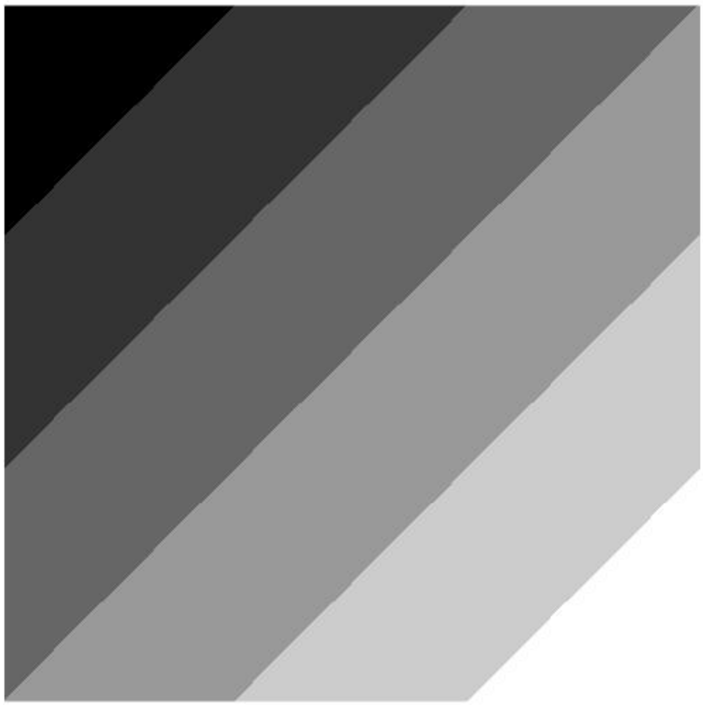
سوال اول

ابتدا تصویر مورد نظر با استفاده از کد زیر ایجاد شده است.

برنامه شماره ۱

```
f = zeros(500);

[M, N] = size(f);
for m=1:M
    for n=1:N
        if m+n < round(M/3)
            f(m, n) = 0*round(255/6);
        end

        if m+n >= round(M/3) && m+n < 2*round(M/3)
            f(m, n) = 1*round(255/6);
        end

        if m+n >= 2*round(M/3) && m+n < M
            f(m, n) = 2*round(255/6);
        end

        if m+n >= M && m+n < round(M/3)+M
            f(m, n) = 3*round(255/6);
        end

        if m+n >= M+166 && m+n <2*round(M/3)+M
            f(m, n) = 4*round(255/6);
        end

        if m+n >= 2*round(M/3)+M && m+n <= M+N
            f(m, n) = 5*round(255/6);
        end

    end
end

imshow(f, [])
```

در این کد مرزهای بین نوارها مشخص شده اند و نواحی بین هر دو مرز با استفاده از سطح روشنایی مناسب پر شده است.



شکل ۱-تصویر با سطوح مختلف موربی

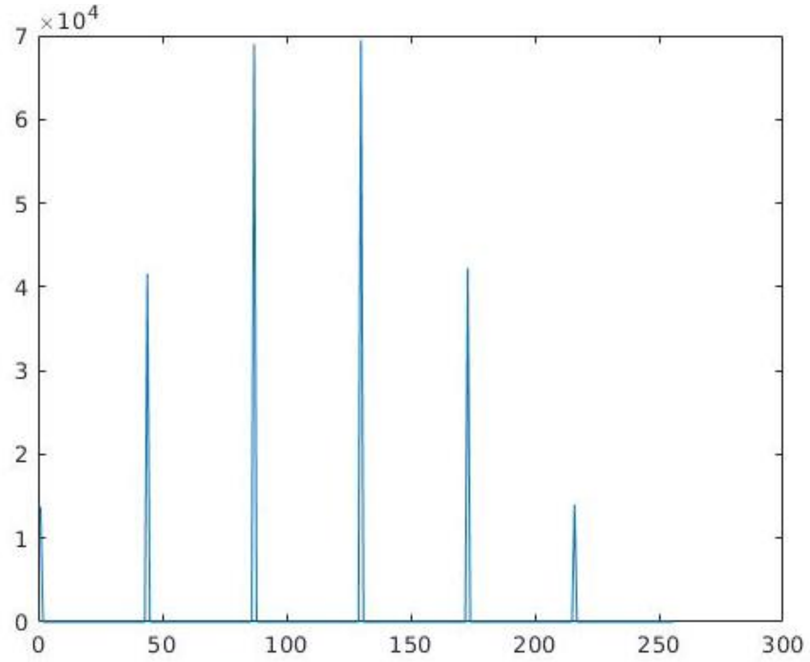
شکل بالا تصویر به دست آمده را نشان می‌دهد.

الف) در این تصویر تنها شش سطح روشنایی متفاوت وجود دارد. بنابراین در هیستوگرام آن شش پیک در نقاط زیر round(255/6 * (0:5)) = [0 43 86 129 172 215] قرار دارد. با توجه به اینکه تعداد پیکسل‌های نوارها دو به دو برابر است و هرچه به سمت وسط حرکت می‌کنیم مساحت نوارها بیشتر می شود، این پیک‌ها در نقاط 86 و 129 هم‌اندازه و با طول بیشتر، در نقاط 0 و 215 هم‌اندازه و با کمترین طول و در نقاط 43 و 172 هم اندازه و با طول متوسط است.

با استفاده از کد زیر هیستوگرام تصویر f به دست می‌آید.

برنامه شماره ۲

```
h = zeros(1, 256);
for m=1:M
    for n=1:N
        s=f(m, n);
        h(s+1)=h(s+1)+1;
    end
end
```

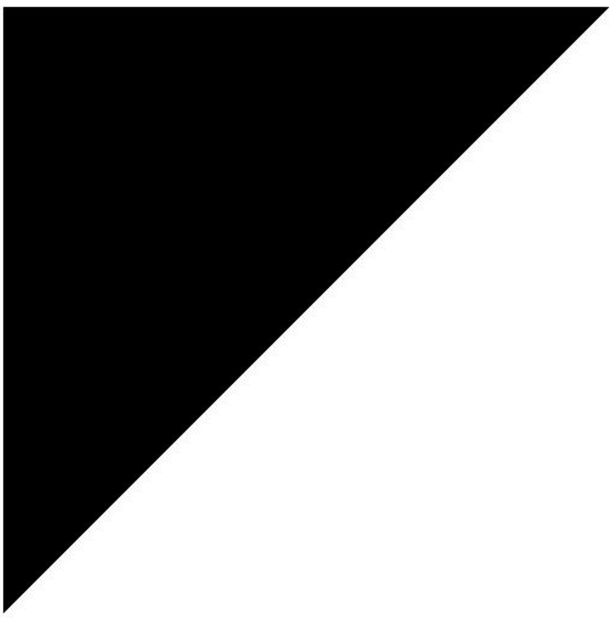


شکل ۲-هیستوگرام تصویر با سطوح مختلف موربی

شکل بالا هیستوگرام به دست آمده است که مطابق با پیش‌بینی انجام شده است.

ب) اگر از آستانه‌ی T1=127 استفاده کنیم، با توجه به هیستوگرام تصویر، نیمی از نوارها (سه نوار بالایی) سطح روشنایی 0 و نیمی دیگر (سه نوار پایینی) سطح روشنایی 1 پیدا میکنند.

شکل زیر حاصل باینری کردن تصویر با سطح آستانه‌ی T1 است.



شکل ۳- تصویر باینری‌شده با روش global

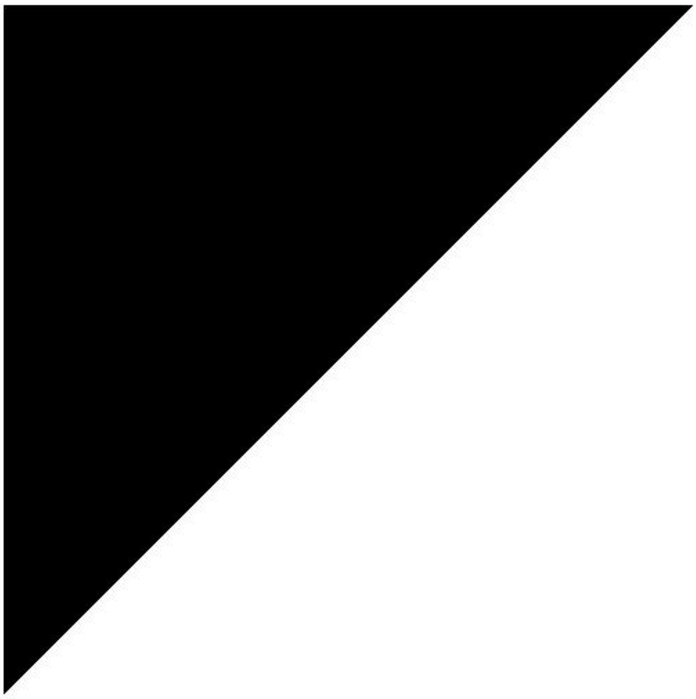
اگر از روش adaptive برای به دست آوردن سطح آستانه‌ی global استفاده کنیم، سطح آستانه‌ی به دست آمده برابر با $T2 = 107.5$ است. چون این روش سعی میکند هیستوگرام تصویر را به دو قسمت تقسیم کند؛ سطح آستانه‌ی به دست آمده بین دو پیک بزرگتر قرار میگیرد. با این سطح آستانه خروجی باینری سازی تصویر با حالت قبل تفاوتی ندارد.

با استفاده از تابع زیر سطح آستانه‌ی adaptive برای تصویر f به دست می‌آید. (مشابه تکلیف قبل)

برنامه شماره ۴

```
function T=global_adaptive(f, h)
    T = mean(mean(f));
    Max = max(max(f));
    Min = min(min(f));
    L = Max-Min+1;
    for iteration=1:100
       Threshold = round(T)-Min;

       h1 = h(1:Threshold);
       p1 = h1/sum(h1);
       u1 = Min:Min+Threshold-1;

       h2 = h(Threshold+1:L);
       p2 = h2/sum(h2);
       u2 = Min+Threshold:Max;

       U1 = p1*u1';
       U2 = p2*u2';

       T_new = (U1+U2)/2;
       if (Threshold - T_new)^2 < 10^-4
           break;
       end
       T = T_new;
    end
end
```

شکل زیر حاصل باینری کردن تصویر با سطح آستانه‌ی T2 است.



شکل ۴-تصویر باینری‌شده با روش Addaptive

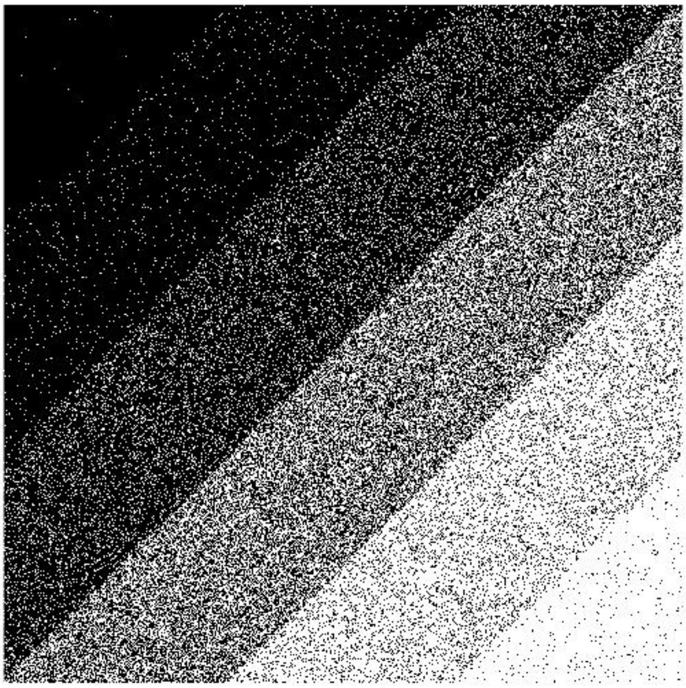
ج) در این قسمت ابتدا تصویر را به بزرگترین سطح روشنایی موجود در تصویر تقسیم میکنیم تا به بازه‌ی [0 ,1] مپ شود سپس با استفاده از روش random dither تصویر را باینری میکنیم. در تابع زیر این روش پیاده سازی شده است.

برنامه شماره ۵

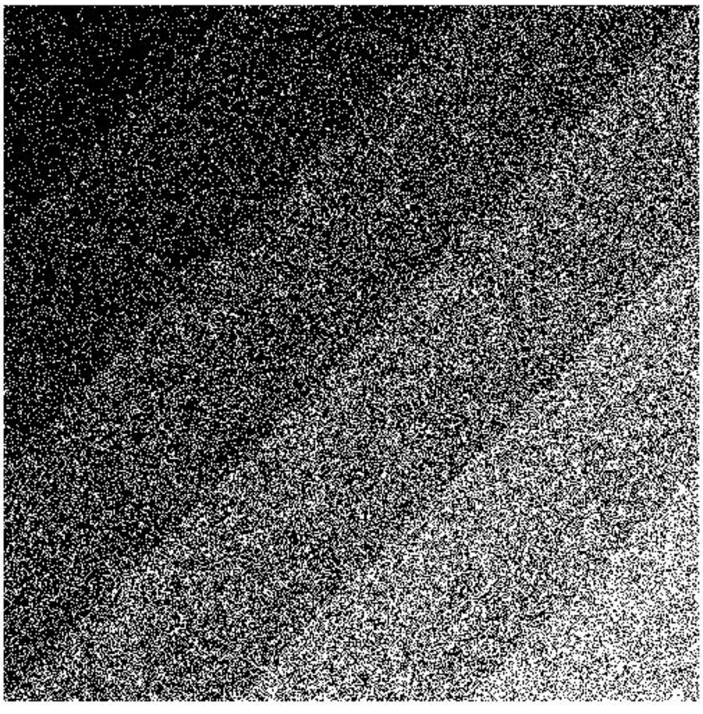
```
function out=random_dither(f, k)
    [M, N] = size(f);
    T = mean(mean(f));
    for m=1:M
        for n=1:N
            r=f(m, n) + k*(randn(1)-0.5);
            if r>T
                out(m, n) = 1;
            else
                out(m, n) = 0;
            end
        end
    end
end
```

در این تابع از میانگین سطح روشنایی پیکسل‌های تصویر به عنوان سطح آستانه استفاده شده است و هربار پیکسل رندوم شده با رابطه داده شده؛ با این سطح آستانه مقایسه شده است.

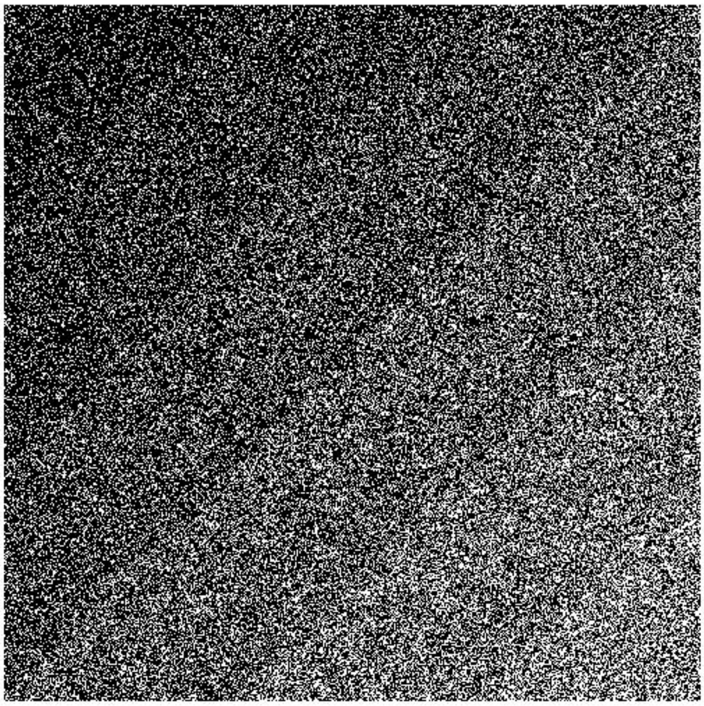
تصاویر زیر حاصل اعمال تابع بالا روی تصویر f را نشان میدهد. در این تصاویر مقدار k به ترتیب برابر با 0.2، 0.5، 1، 1.5 قرار داده شده است.



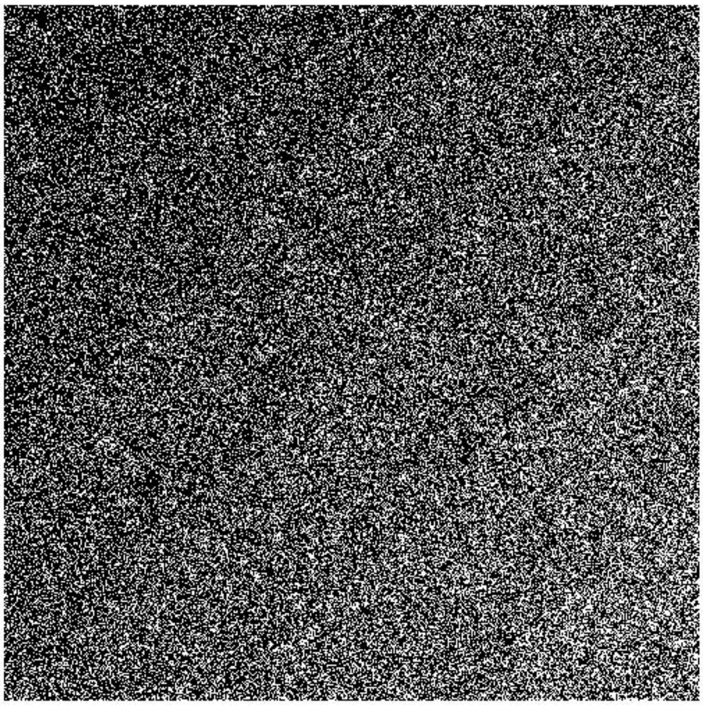
شکل ۵-باینری سازی با روش random dither با k=0.2



شکل ۶- باینری سازی با روش random dither با k=0.5



شکل ۷- باینری سازی با روش random dither با k=1



شکل ۸- باینری سازی با روش random dither با k=1.5

همان طور که مشخص است در تصویر اول (k=0.2) نوارها تا حدودی از یکدیگر جدا کرده است و شش نوار قابل تشخیص‌اند. اما هرچه مقدار k بزرگتر شده است نوارهای تصویر بیشتر از بین رفته است.

با افزایش مقدار k، مقدار رندومی که با پیکسل‌های تصویر جمع می‌شود بزرگتر میشود. این باعث میشود میزان تصادفی بودن تصویر افزایش یابد و ساختار منظمی که در تصویر وجود دارد (نوارها) از بین بروند و تصویر حالت برفک پیدا کند.

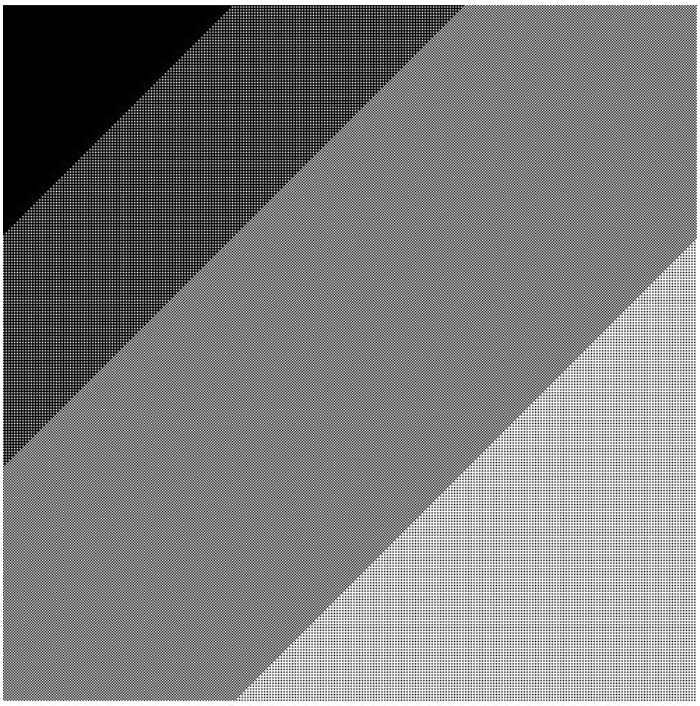
با توجه به اینکه تصویر ما از ساختار منظمی پیروی میکند؛ روش random dither که صرفا با تصادفی کردن تصویر سعی در باینری کردن آن دارد؛ نتوانسته است به خوبی این کار را انجام دهد. این روش ابتدا پیکسل‌های تصویر را با یک مقدار تصادفی جمع میکند. با افزایش این مقدار تصادفی تصویر ساختار منظم خود را از دست می‌دهد و نوارها قابل تفکیک نیستند.

د) در این قسمت با استفاده از کد زیر روش ordered dither پیاده سازی شده است.

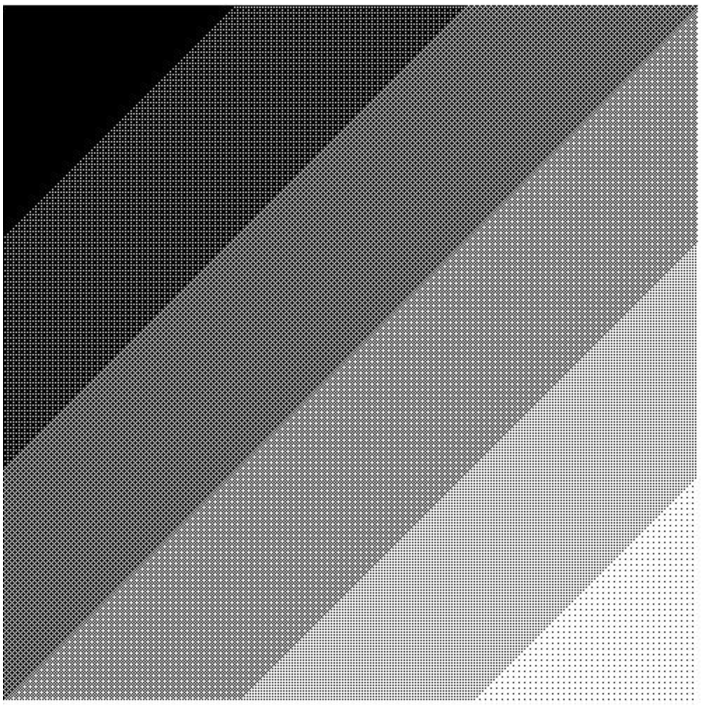
برنامه شماره ۶

```
function out=ordered_dither(f, I)
    [M, N] = size(f);
    [S, V] = size(I);
    for m=1:S:M-S
        for n=1:V:N-V
           out(m:m+S-1, n:n+V-1) = f(m:m+S-1, n:n+V-1) > I;
        end
    end
end
```

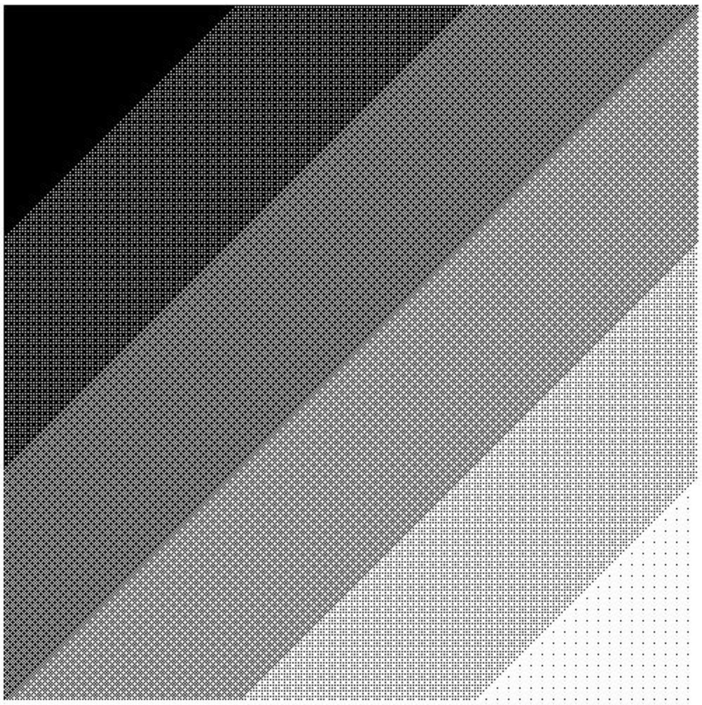
سپس تصویر f با ماتریس‌های I2، I4 و I8 و تابع بالا به ترتیب halftone شده است. تصاویر زیر نتایج به دست آمده را نشان می‌دهد.



شکل ۹-باینری سازی با روش halftoning و I2



شکل ۱۰- باینری سازی با روش halftoning و I4



شکل ۱۱- باینری سازی با روش halftoning و I8

همان طور که مشخص است I2 به‌خاطر نزدیک بودن روشنایی سطوح میانی و انتهایی در پایین-سمت راست، مقایسه این سطوح با ماتریس I2 نتایج یکسانی دارد که سبب می‌شود چند سطح به یک سطح نگاشت شود. بنابراین نتوانسته است نوارهای (3 و 4) و (5 و 6) را از یکدیگر جدا کند.

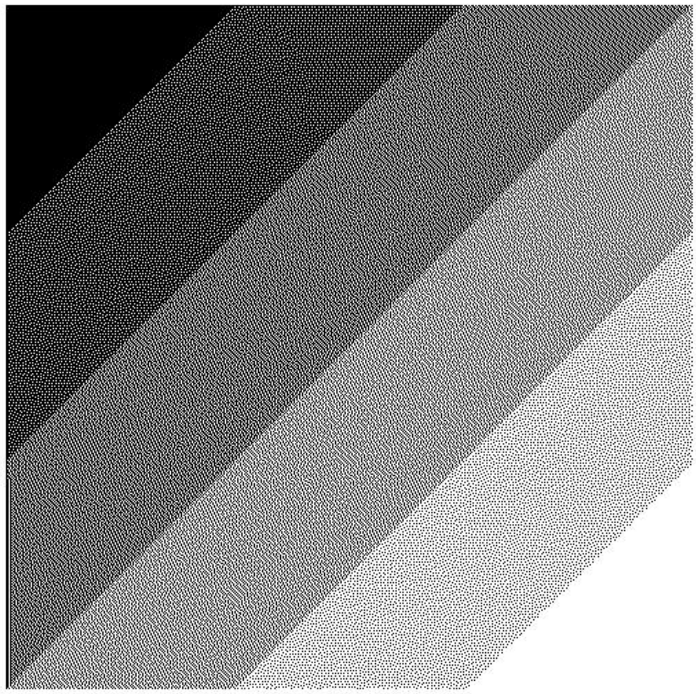
برای I4 و I8 نتایج یکسانی بدست می‌آید زیرا اختلاف سطوح تصویر در مقایسه با محدوده اعداد داخل ماتریس‌ها قادر به تولید نتایج مختلف برای ۰ و ۱ ها نخواهد بود. از طرفی از بین I4 و I8، ابعاد کوچک‌تر ماتریس I4 منجر به تولید ساختارهای منظم یا همان worm کمتری نسبت به I8 شده‌است که این مورد به وضوح در شکل ۱۰ و شکل ۱۱ دیده می‌شود.

ه) با استفاده از تابع زیر روش normal error diffusion پیاده سازی شده است.

برنامه شماره ۷

```
function out=normal_error_diffusion(f)
    [M, N] = size(f);
    if max(max(f)) > 1
       f = f/max(max(f));
    end
    f_hat = f;
    I = [0, 0, 7/16; 3/16, 5/16, 1/16];
    T = 0.5;
    for m=1:M-2
        for n=2:N-1
           if f_hat(m, n) >= T
               b(m, n)=1;
           else
               b(m, n)=0;
           end
           e = b(m, n) - f_hat(m, n);
           f_hat(m:m+1, n-1:n+1) = f_hat(m:m+1, n-1:n+1) - I*e;
        end
    end
    out = b;
end
```

تصویر زیر نتیجه‌ی اعمال تابع بالا بر روی تصویر است .



شکل ۱۲- باینری سازی با روش normal error diffusion

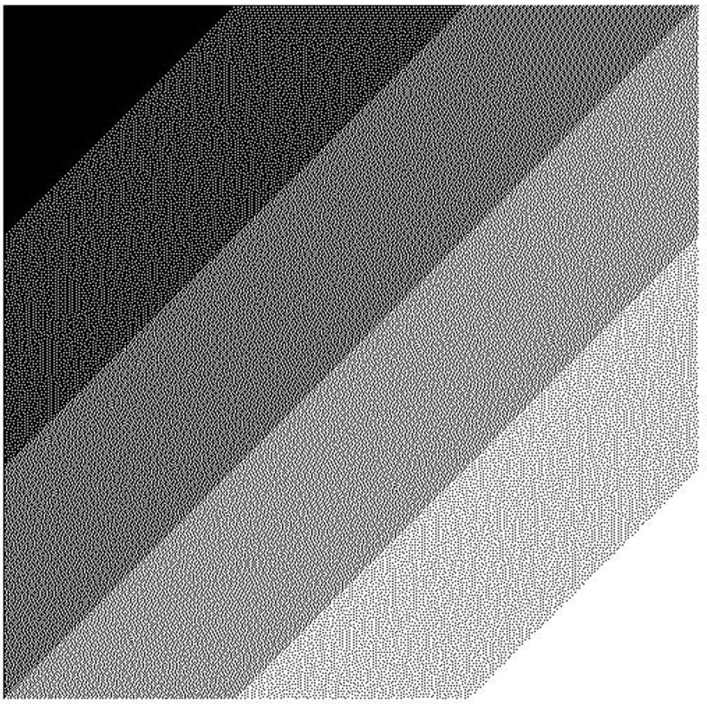
با استفاده از تابع زیر روش serpentine error diffusion پیاده سازی شده است.

برنامه شماره ۸

```
function out=serpentine_error_diffusion(f)
    [M, N] = size(f);
    if max(max(f)) > 1
       f = f/max(max(f));
    end
    f_hat = f;
    I = [0, 0, 7/16; 3/16, 5/16, 1/16];
    T = 0.5;
    for m=1:2:M-2
       for n=2:N-1
           if f_hat(m, n) >= T
               b(m, n)=1;
           else
               b(m, n)=0;
           end
           e = b(m, n) - f_hat(m, n);
           f_hat(m:m+1, n-1:n+1) = f_hat(m:m+1, n-1:n+1) - I*e;
       end

       m_hat = m+1;
       for n=N-1:-1:2
           if f_hat(m_hat, n) >= T
               b(m_hat, n)=1;
           else
               b(m_hat, n)=0;
           end
           e = b(m_hat, n) - f_hat(m_hat, n);
```

```
            f_hat(m_hat:m_hat+1, n-1:n+1) = f_hat(m_hat:m_hat+1, n-1:n+1) -
fliplr(I)*e;
        end
    end
    out = b;
end
```

تصویر زیر نتیجه‌ی اعمال تابع بالا بر روی تصویر است.



شکل ۱۳-باینری‌سازی با روش serpentine error diffusion

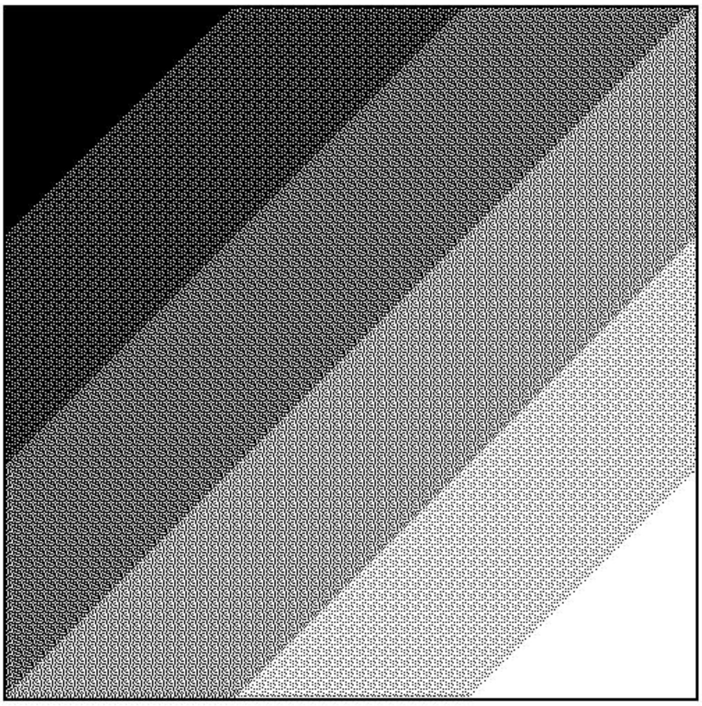
همانطور که مشخص است در این تصویر روش normal موفق بوده است و در روش serpentine تعداد worm های بیشتری تولید شده است. در واقع روش serpentine نتوانسته است ایجاد worm ها را کاهش دهد.

و) در این قسمت با استفاده از تابع زیر روش dot diffusion پیاده سازی شده است.

برنامه شماره ۹

```
function out=dot_diffusion(f)
    [M, N] = size(f);
    if max(max(f)) > 1
       f = f/max(max(f));
    end
    CM = [59 12 46 60 28 14 32 3;
        21 25 44 11 58 45 43 30;
        24 20 13 42 33 5 54 8;
        64 52 55 40 63 47 7 18;
        35 57 9 15 50 48 4 36;
        41 17 6 61 22 49 62 34;
        2 53 19 56 39 23 26 51;
        16 37 1 31 29 27 38 10];
    S=8;
    nn=64;
    DM=[1 2 1;
        2 0 2;
        1 2 1;];
   CM = repmat(CM,floor(M/S),floor(N/S));
   [P, Q] = size(CM);
   CM = padarray(CM, [round((M-P)/2) round((N-Q)/2)] ,'both');
   H=zeros(M, N);
  for ii=1:nn
      [p1,p2]=(find(CM==ii));
      i1=1;
      j1=1;
      for m1=1:1:size(p1,1)
            while(i1<=size(p1,1))
                  if(f(p1(i1),p2(j1))>0.5)
                      H(p1(i1),p2(j1))=1;
                  end
                  qerr=f(p1(i1),p2(j1))-H(p1(i1),p2(j1))     ;
                  cl=CM((p1(i1)-1):(p1(i1)+1),(p2(j1)-1):(p2(j1)+1));
                  k=cl>ii;
                  DM1=(DM.*k);
                  DM1=DM1/sum(DM1(:));
                  err=DM1*qerr;
                  f((p1(i1)-1):(p1(i1)+1),(p2(j1)-1):(p2(j1)+1))=
f((p1(i1)-1):(p1(i1)+1),(p2(j1)-1):(p2(j1)+1))+err;
                  i1=i1+1;
                  j1=j1+1;
            end
      end
  end
  out=H;
end
```

تصویر زیر نتیجه‌ی اعمال تابع بالا بر روی تصویر است.



شکل ۱۴-باینری‌سازی با روش dot diffusion

همانطور که مشخص است به خاطر تغییرات گام به گام سطوح روشنایی و وجود تعداد زیادی پیکسل با سطح روشنایی یکسان، تکرار ماتریس مربوط به error diffusion منجر به نتایج قابل قبول‌تر با الگوهای تکرارشونده کمتری نسبت به dot diffusion خواهد شد. در واقع در روش dot diffusion خطای باینری سازی بصورت نقطه به نقطه پخش خواهد شد که این امر در تصاویر با تعداد سطوح روشنایی محدود با چگالی بالا، worm های زیادی اضافه خواهد کرد در حالیکه روش error diffusion به دلیل عملکرد ماتریسی، worm کمتری خواهیم داشت.

سوال دوم

در این سوال از ما خواسته شده ثابت کنیم که اگر در حالت آنالوگ محورهای x-y به اندازه‌ی θ دوران کنند تبدیل فوریه‌ی نظیر آن‌ها نیز به همان اندازه و در همان جهت دوران می‌کند.

برای ثابت کردن فرض بالا کافی است ثابت کنیم تبدیل فوریه‌ی یک تابع دوران داده شده با نسخه‌ای که در آن تبدیل فوریه را برای تابع اصلی به‌دست آوریم و سپس دوران دهیم یکسان است.

فرض کنید تابع $g(x)$ تابعی از x است، در این صورت برای تبدیل فوریه‌ی آن داریم:

$$\mathcal{F}[g(x)] = G(X) = \int g(x)\exp\{-jx^T\}dx$$

می‌خواهیم تبدیل فوریه نسخه ی دوارن یافته‌ی g را بیابیم. تبدیل خطی دوران بصورت زیر تعریف می‌شود:

$$u = Ax$$

در رابطه‌ی بالا ماتریس A، Orthonormal است پس داریم:

$$* \quad x = A^{-1}u = A^T u$$

پس تبدیل فوریه‌ی نسخه‌ی دوران یافته g بصورت زیر تعریف می‌شود:

$$\mathcal{F}[g(Ax)] = \int g(u)\exp\{-j(A^T u)^T X\}\,|J|du$$

در رابطه‌ی بالا $|J|$، determinant ژاکوبین رابطه ی * است. همانطور که میدانیم $|J| = 1$ است، پس داریم:

$$\mathcal{F}[g(Ax)] = \int g(u)\exp\{-ju^T AX\}\,du = G[AX]$$

آخرین عبارت بیانگر نسخه‌س دوران یافته تبدیل فوریه‌ی $G(X)$ است و در اینجا فرض اثبات می‌شود.

سوال سوم

الف:

نویز گوسی $v(x,y)=\cos\left(\frac{2\pi}{T_1}x+\frac{2\pi}{T_2}y\right)$ در حالتی که تبدیل فوریه آن بدون شیفت دادن رسم شود، طبق تناسب‌های زیر و به‌خاطر متناوب بودن تبدیل فوریه سیگنال گسسته، در نقاط زیر رخ خواهد داد: $\left(\frac{M}{T_1}+1\,,\frac{N}{T_2}+1\right)$ و $\left(M-\left(\frac{M}{T_1}+1\right),N-\left(\frac{N}{T_2}+1\right)\right)$ پیک می‌زند. دلیل اضافه شدن یک به عبارات داخل تناسب، شروع اندیس برنامه متلب از شماره ۱ می‌باشد.

M	2π
$? = \dfrac{M * \frac{2\pi}{T_1}}{2\pi} = \dfrac{M}{T_1}$	$\dfrac{2\pi}{T_1}$

N	2π
$? = \dfrac{N * \frac{2\pi}{T_1}}{2\pi} = \dfrac{N}{T_2}$	$\dfrac{2\pi}{T_2}$

با انتقال تبدیل فوریه به مرکز چارت، نقاط بالا نیز حول $\left(\frac{M}{2},\frac{N}{2}\right)$ خواهند رفت و نقاط زیر را خواهیم داشت:

$$\left(\frac{M}{2}+\left(\frac{M}{T_1}+1\right),\frac{N}{2}+\left(\frac{N}{T_2}+1\right)\right) \text{ و } \left(\frac{M}{2}+\left(M-\left(\frac{M}{T_1}+1\right)\right),\frac{N}{2}+\left(N-\left(\frac{N}{T_2}+1\right)\right)\right)$$

ب:

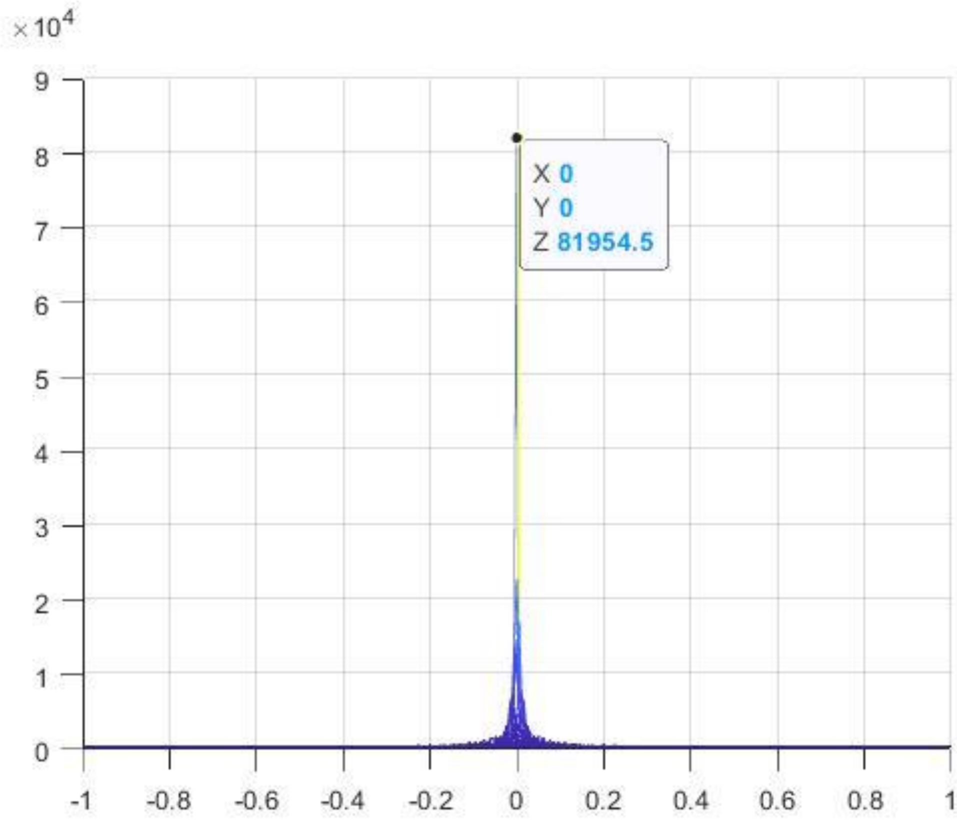
۱) مشاهده تصویر رنگی و سیاه‌وسفید $miner$ و محتوای فرکانسی آن به کمک دستور $mesh$

برنامه شماره ۱

```
h=imread('Sudbury INO  - Home page - Article page lead.jpg');
figure,imshow(h,[])
f=rgb2gray(h);
f=im2double(f); % f between 0 ,1
figure,imshow(f,[]),title('Original Miner')
%---- plot mesh noise
[M,N]=size(f);
F=fft2(f);
%---- plot mesh setup axises by meshgrid
x = -1:2/N:1-2/N;
y = -1:2/M:1-2/M;
[X,Y] = meshgrid(x,y);
figure,mesh(X,Y,abs(F));
```

شکل ۱۵ -تصویر رنگی Miner

شکل ۱۶-تصویر سیاه‌وسفید Miner



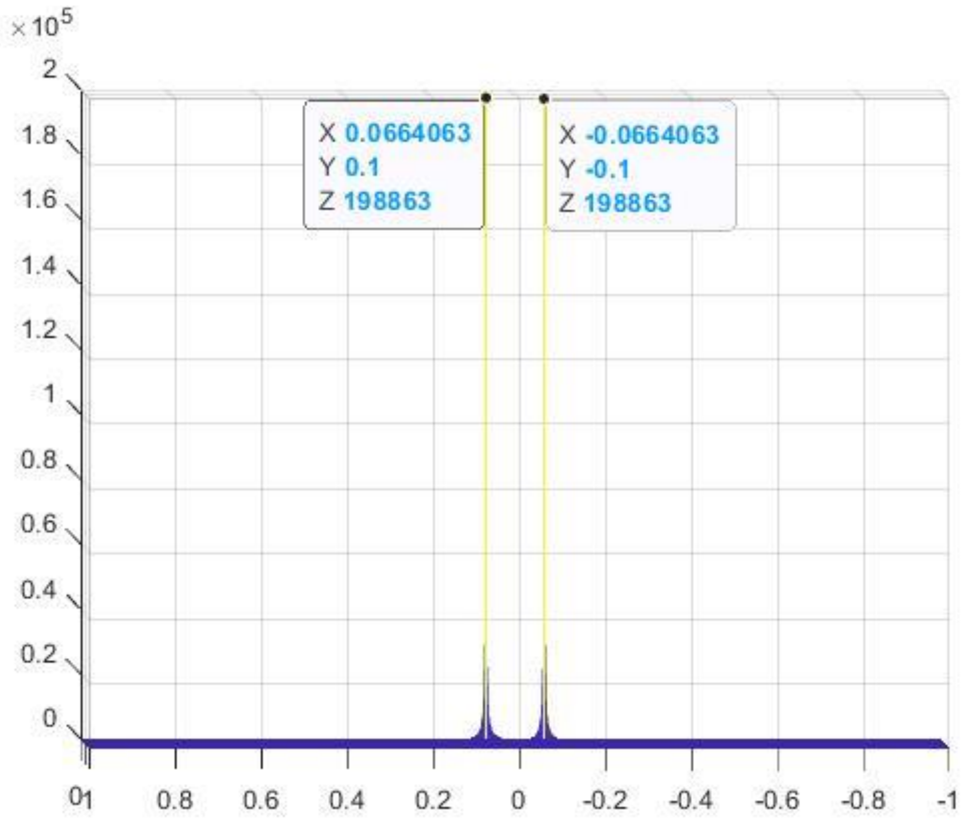
شکل ۱۷– محتوای فرکانسی تصویر Miner

تذکر: تمامی شکل‌ها با انتقال محتوای فرکانسی به مرکز چارت رسم شده‌اند اما در برنامه‌های ضمیمه شده این مورد تنها برای تصویر آغشته به نویز سینوسی مشاهده می‌گردد.

۲) مشاهده طیف فرکانسی نویز سینوسی جمع‌شونده $v(x,y)$

برنامه شماره ۲

```
T1=20;
T2=30;
v=zeros(M,N);
for m=1:M
    for n=1:N
        v(m,n)=cos(2*pi*(m/T1) + 2*pi*(n/T2));
    end
end

V=fft2(v);
%---- plot mesh setup axises by meshgrid
x = -1:2/N:1-2/N;
y = -1:2/M:1-2/M;
[X,Y] = meshgrid(x,y);
figure,mesh(X,Y,abs(V));
```



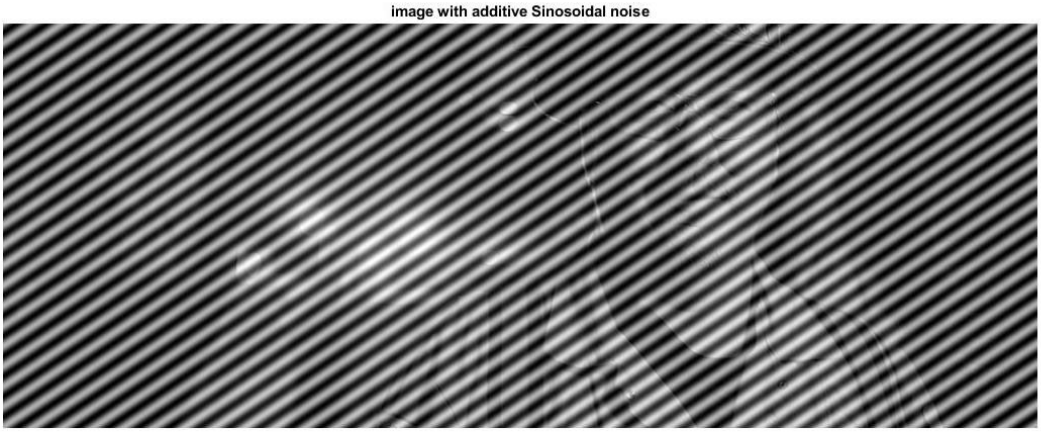
شکل ۱۸-محتوای فرکانسی نویز سینوسی

در شکل ۱۸، حدود محور افقی $-\pi$ تا π است که برای سادگی نمایش از 1- تا 1 استفاده شده‌است. لذا انتظار می‌رود شکل ۱۸ در نقطه (0.1,0.0664063) و نظیر آن در قسمت منفی پیک مشاهده شود. این اعداد معادل با $\frac{2}{T_1} = \frac{2}{20} = 0.1$ و $0.066 = \frac{2}{30} = \frac{2}{T_3}$ است.

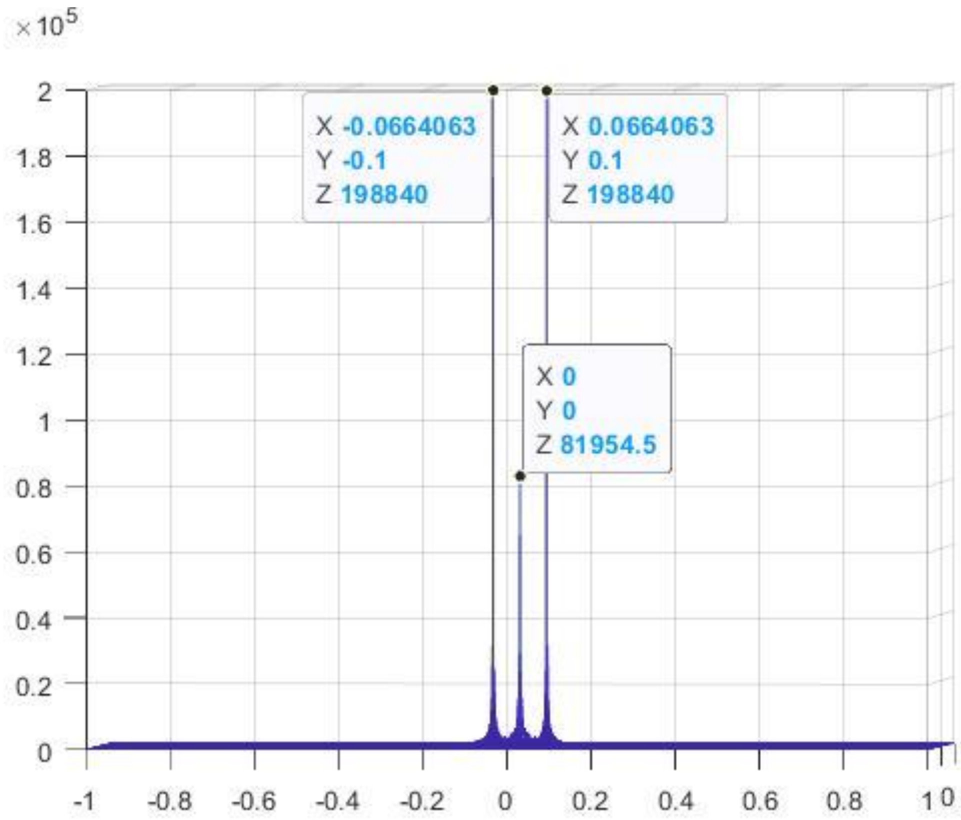
۳) مشاهده تصویر Miner آغشته به نویز سینوسی و انتقال تبدیل فوریه آن به مرکز چارت با حذف DC

برنامه شماره ۳

```
f1=f+v; %noisy image
figure,imshow(f1,[])
% f1=f1-mean(mean(f1));
g=zeros(M,N);
for m=1:M
    for n=1:N
        g(m,n)=(-1)^(m+n)*f1(m,n);
    end
end

g=g-mean(mean(g));
G=fft2(g); %shift middle of mesh grid
%---- plot mesh setup axises by meshgrid
x = -1:2/N:1-2/N;
y = -1:2/M:1-2/M;
[X,Y] = meshgrid(x,y);
figure,mesh(X,Y,abs(G))
```



شکل ۱۹- تصویر Miner آغشته به نویز سینوسی



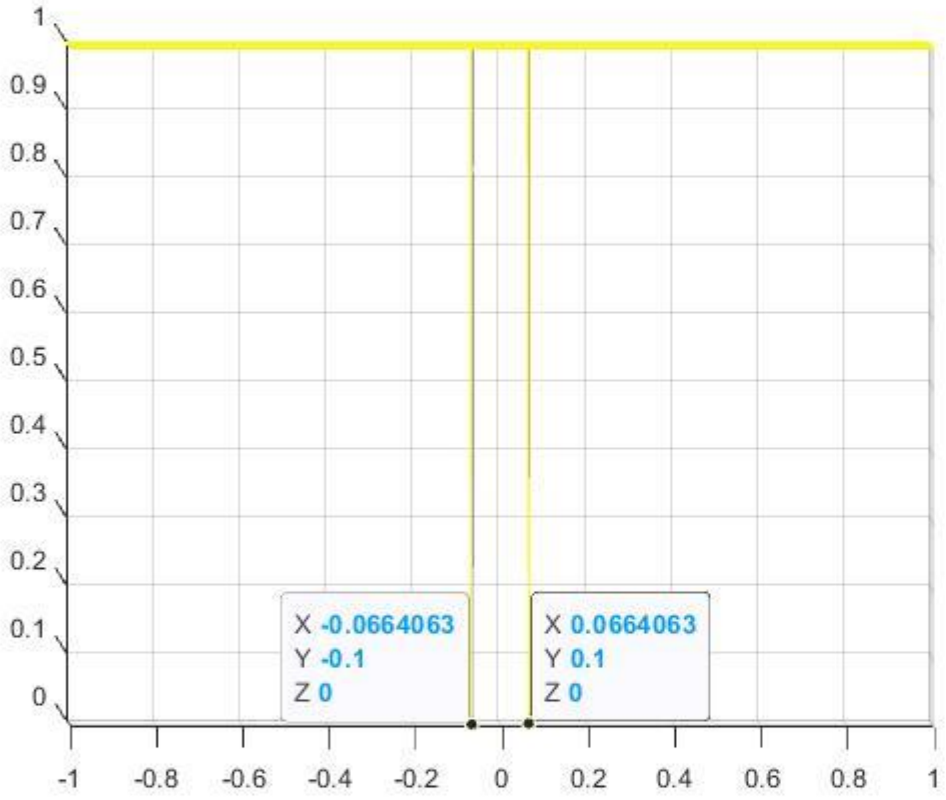
شکل ۲۰-محتوای فرکانسی تصویر آغشته به نویز سینوسی

۴) طراحی فیلتر میان نگذر (Notch Filter) به کمک معیار نصف حداکثر توان محتوای فرکانسی

برنامه شماره ۴

```
% design Notch fliter H
Gb=abs(G)>0.5*max(max(abs(G)));
H=1-Gb;
% plot mesh setup axises by meshgrid
x = -1:2/N:1-2/N;
y = -1:2/M:1-2/M;
[X,Y] = meshgrid(x,y);
```

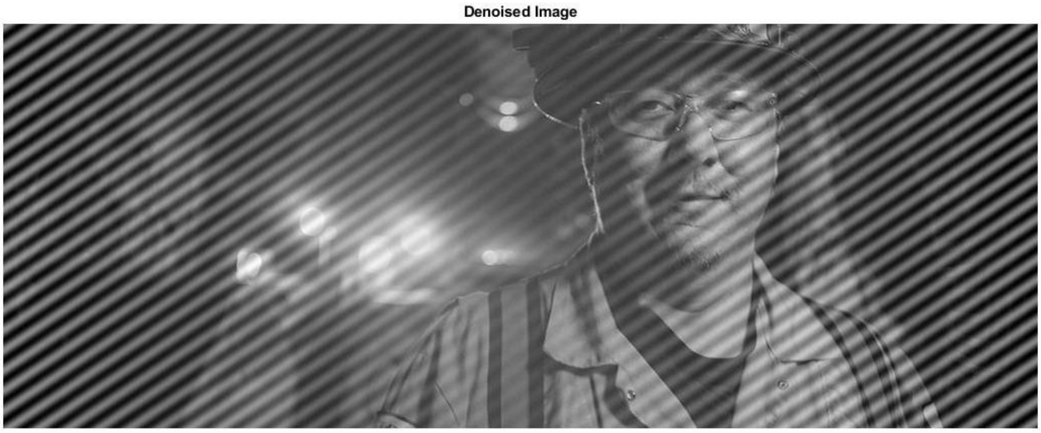
```
figure,mesh(X,Y,abs(H))
```



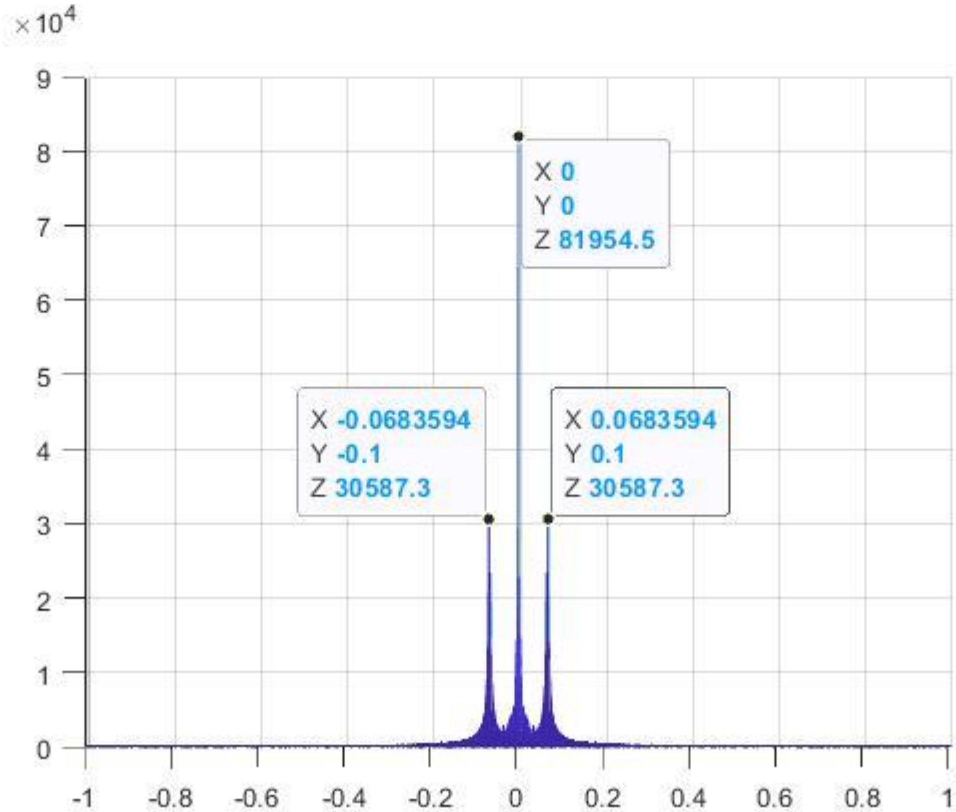
شکل ۲۱- محتوای فرکانسی فیلتر میان نگذر (Notch filter)

۵) مشاهده خروجی فیلتر میان گذر و تصویر بازسازی نویززدایی شده

```
FF=H.*G;
% plot mesh setup axises by meshgrid
x = -1:2/N:1-2/N;
y = -1:2/M:1-2/M;
[X,Y] = meshgrid(x,y);
figure,mesh(X,Y,abs(FF))
ff=ifft2(FF);
[M,N]=size(ff);
ff=real(ff);

f_hat=zeros(M,N);
for m=1:M
    for n=1:N
        f_hat(m,n)=(-1)^(m+n)*ff(m,n);
    end
end

figure,imshow(f_hat,[]),title('Denoised Image')
```



شکل ۲۲-تصویر خروجی نویززدایی‌شده



شکل ۲۳- محتوای فرکانسی تصویر نویززدایی‌شده

شکل ۲۲ نسبت به شکل ۱۹ بسیار بهتر شده اما هم‌چنان خطوط مورب در شکل واضح هستند. دلیل این امر را می‌توان در طیف فرکانسی نویز سینوسی در شکل ۱۸ یافت. از آن‌جا که نویز سینوسی در محدوده زمانی کوتاهی و نه در همه زمان‌ها مشاهده شده‌است لذا $rect(t) - rect(t - T)$ در کل سیگنال ضرب می‌شود. منظور از T بازه مشاهده سیگنال سینوسی است که در دو بعد تصویر همان ابعاد تصویر می‌باشد. در واقع روی سیگنال سینوسی پنجره‌گذاری اعمال می‌گردد و تبدیل فوریه آن حاصل کانولوشن سیگنال سینوسی با تبدیل فوریه پالس مستطیلی که سینک می‌باشد. پس در شکل ۱۸، سینک یک بار در جهت راست و بار دیگر در جهت چپ جابه‌جا می‌گردد و محتوای فرکانسی نویز را تولید می‌کند. فیلتر

میان‌نگذر اعمال شده تنها قله لوب اصلی این دو سینک را حذف خواهد کرد ولی هم‌چنان لوب فرعی که حدود ۲۰ دسی بل دامنه (1/13 دامنه لوب اصلی) را دارد باقی خواهد ماند لذا اثر نویز سینوسی باقی می‌ماند. برای حذف کامل نویز سینوسی، نیاز به حذف یک یا دو لوب فرعی از سینک‌ها می‌باشد تا توان سینک‌ها به‌خوبی تضعیف شود.

سوال چهارم

$$C_n=\sqrt{\frac{2}{n}}\begin{array}{cc}
\begin{array}{ccccc} k=0 & k=1 & k=2 & \longrightarrow & k=n-1 \end{array} & \\
\begin{bmatrix} \frac{1}{\sqrt{2}} & \cos\left(\frac{\pi}{n}\left(\frac{1}{2}\right)\right) & \cos\left(\frac{2\pi}{n}\left(\frac{1}{2}\right)\right) & \cdots & \cos\left(\frac{(n-1)\pi}{n}\left(\frac{1}{2}\right)\right) \\ \frac{1}{\sqrt{2}} & \cos\left(\frac{\pi}{n}\left(\frac{3}{2}\right)\right) & \cos\left(\frac{2\pi}{n}\left(\frac{3}{2}\right)\right) & & \cos\left(\frac{(n-1)\pi}{n}\left(\frac{3}{2}\right)\right) \\ \vdots & \vdots & & \ddots & \vdots \\ \frac{1}{\sqrt{2}} & \cos\left(\frac{\pi}{n}\left(\frac{2n-1}{2}\right)\right) & & \cdots & \cos\left(\frac{(n-1)\pi}{n}\left(\frac{2n-1}{2}\right)\right) \end{bmatrix} & \begin{array}{c} j=0 \\ j=1 \\ \downarrow \\ j=n-1 \end{array}
\end{array}.$$

در این سوال از ما خواسته شده ثابت کنیم که سطرهای ماتریس C_n برهم عمودند. این مطابق تعریف مفهوم $orthogonal\ matrix$ است. به ماتریسی orthogonal می‌گویند که سطر و ستونهای آن از orthonormal vectorها ساخته شده باشند. برای ماتریس orthogonal رابطه‌ی زیر برقرار است:

$$=Q^{-1}Q^{T}$$

پس برای اینکه ثابت کنیم ماتریس C_n orthogonal است کافی است ثابت کنیم:

$$C_n^{-1}=C_n^{T}$$

Proof: We can avoid a nightmare of trigonometry with an indirect proof. Let $A_n$be the real symmetric cirulant matrix

$$A_n=\begin{bmatrix} 1 & -1 & & & & 0 \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ 0 & & & -1 & 2 & -1 \\ & & & & -1 & 1 \end{bmatrix}$$

We will show that the columns $\vec{v}^{(k)}$ of C_n are the eigenvector of A_n. This mean that they are automatically orthogonal, since A_n is real symmetric. First, $\vec{v}^{(k)}=\frac{1}{\sqrt{2}}(1,1,\dots,1)^{T}$.

So,

$$A_n\vec{v}^{(0)}=\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & -1 & & & & 0 \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ 0 & & & -1 & 2 & -1 \\ & & & & -1 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}=\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Thus,

$$A_n\vec{v}^{(0)} = O\vec{v}^{(0)}$$

For every other column, the component take the form:

$$v_l^{(k)} = \sqrt{\frac{2}{n}}\cos\left(\frac{k\pi}{n}\left(l+\frac{1}{2}\right)\right)$$

Suppose: $\theta = \frac{k\pi}{n}$

For j=1, …, n-2

$$(A_n\vec{v}^{(k)})_j = \sum_{l=0}^{n-1}(A_n)_{jl}\, v_l^{(k)} = v_{j-1}^{(k)} + 2v_j^{(k)} - v_{j+1}^{(k)}$$

$$= \sqrt{\frac{2}{n}}\left(-\cos\left(\theta\left(j-\frac{1}{2}\right)\right) + 2\cos\left(\theta\left(j+\frac{1}{2}\right)\right) - \cos\left(\theta\left(j+\frac{3}{2}\right)\right)\right)$$

$$= \sqrt{\frac{2}{n}}\left(-\cos\left(\theta\left(j+\frac{1}{2}\right)-\theta\right) + 2\cos\left(\theta\left(j+\frac{1}{2}\right)\right) - \cos\left(\theta\left(j+\frac{3}{2}\right)+\theta\right)\right)$$

Suppose: $B = \theta(j+\frac{1}{2})$

$$= \sqrt{\frac{2}{n}}(-cosBcos\theta - sinBsin\theta + 2cosB - cosBcos\theta + sinBsin\theta)$$

$$= \sqrt{\frac{2}{n}}(2-2cos\theta)cosB$$

$$= \left(2-2\cos\left(\frac{k\pi}{n}\right)\right)v_j^{(k)}$$

In this equation $(2-2\cos(\frac{k\pi}{n}))$ are eigenvalues.

For j=0 we have:

$$(A_n\vec{v}^{(k)})_0 = v_0^{(k)} - v_1^{(k)} = \sqrt{\frac{2}{n}}\left(\cos\left(\frac{1}{2}\theta\right) - \cos\left(\frac{3}{2}\theta\right)\right)$$

$$= \sqrt{\frac{2}{n}}\left(\cos\left(\frac{1}{2}\theta\right) - \cos\left(\frac{1}{2}\theta+\theta\right)\right)$$

$$= \sqrt{\frac{2}{n}}\left(\cos\left(\frac{1}{2}\theta\right) - \cos\left(\frac{1}{2}\theta\right)cos\theta + \sin\left(\frac{1}{2}\theta\right)sin\theta\right)$$

$$= \sqrt{\frac{2}{n}} \cos\left(\frac{1}{2}\theta\right)(1 - cos\theta + 2sin^2(\frac{1}{2}\theta))$$

$$= \sqrt{\frac{2}{n}} \cos\left(\frac{1}{2}\theta\right)(2 - 2cos\theta)$$

$$= (2 - 2\cos(\frac{k\pi}{n}))v_0^{(k)}$$

Similarly,

$$(A_n\vec{v}^{(k)})_{n-1} = v_{n-1}^{(k)} - v_{n-2}^{(k)} = \sqrt{\frac{2}{n}}(\cos\left(\theta(n - \frac{1}{2})\right) - \cos\left(\theta(n - \frac{3}{2})\right))$$

$$= (2 - 2\cos(\frac{k\pi}{n}))v_{n-1}^{(k)}$$

To understand where this comes from, note that the columns $\vec{v}^{(k)}$ are discrete approximations of $\cos(kx)$ at the points x_j.

We know that $u(x) = \cos(kx)$ satisfies the differential equation $-\frac{d^2u}{dx^2} = k^2u_1$ ie. $\cos(kx)$ is an eigenvalue of the differential operator $-\frac{d^2u}{dx^2}$, with eigenvalue k^2.

The matrix A_n is a finite-difference approximation to $\frac{d^2u}{dx^2}$ at equally-spaced points,

$$(A_n\vec{u})_j = -u_{j-1} + 2u_j - u_{j+1} = -[(u_{j+1} - u_j) - (u_j - u_{j-1})]$$

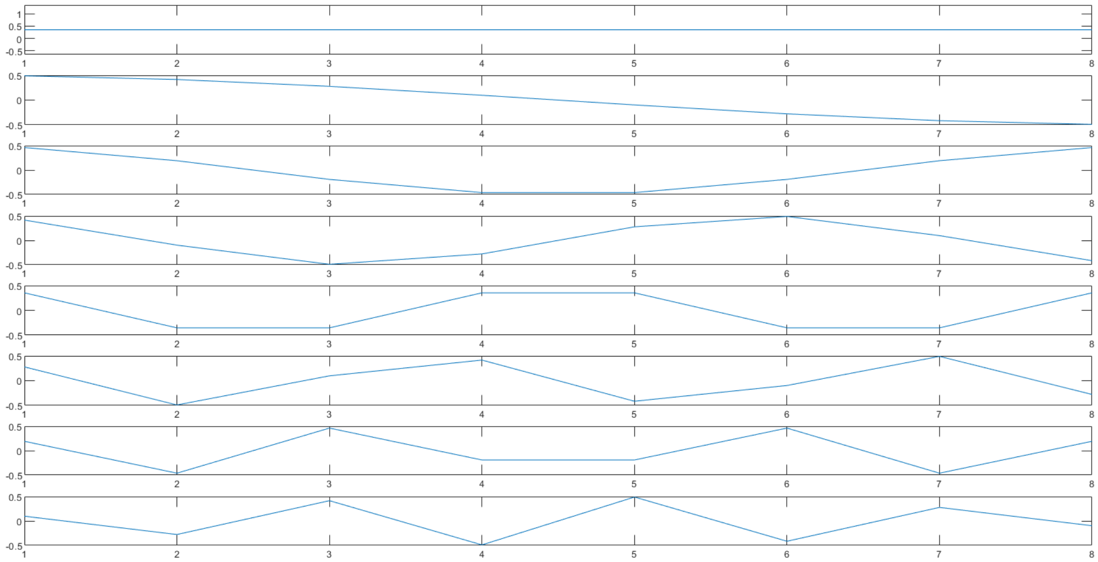
So its eigenvectors are discrete cosine.

از این نتیجه ثابت می‌شود که ماتریس C_n، Orthogonal است.

ماتریس C_n برای n=8:

$$\begin{bmatrix}
0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 \\
0.4904 & 0.4157 & 0.2778 & 0.0975 & -0.0975 & -0.2778 & -0.4157 & -0.4904 \\
0.4619 & 0.1913 & -0.1913 & -0.4619 & -0.4619 & -0.1913 & 0.1913 & 0.4619 \\
0.4157 & -0.0975 & -0.4904 & -0.2778 & 0.2778 & 0.4904 & 0.0975 & -0.4157 \\
0.3536 & -0.3536 & -0.3536 & 0.3536 & 0.3536 & -0.3536 & -0.3536 & 0.3536 \\
0.2778 & -0.4904 & 0.0975 & 0.4157 & -0.4157 & -0.0975 & 0.4904 & -0.2778 \\
0.1913 & -0.4619 & 0.4619 & -0.1913 & -0.1913 & 0.4619 & -0.4619 & 0.1913 \\
0.0975 & -0.2778 & 0.4157 & -0.4904 & 0.4904 & -0.4157 & 0.2778 & -0.0975
\end{bmatrix}$$

ب) اثبات برابری $inv(D) = D'$:



شکل ۲۴- شکل موج‌های سطرهای ماتریس D

```
>> D=dctmtx(8);
>> abs(inv(D)-D')<0.00000000000001

ans =

  8×8 logical array

   1   1   1   1   1   1   1   1
   1   1   1   1   1   1   1   1
   1   1   1   1   1   1   1   1
   1   1   1   1   1   1   1   1
   1   1   1   1   1   1   1   1
   1   1   1   1   1   1   1   1
   1   1   1   1   1   1   1   1
   1   1   1   1   1   1   1   1
```

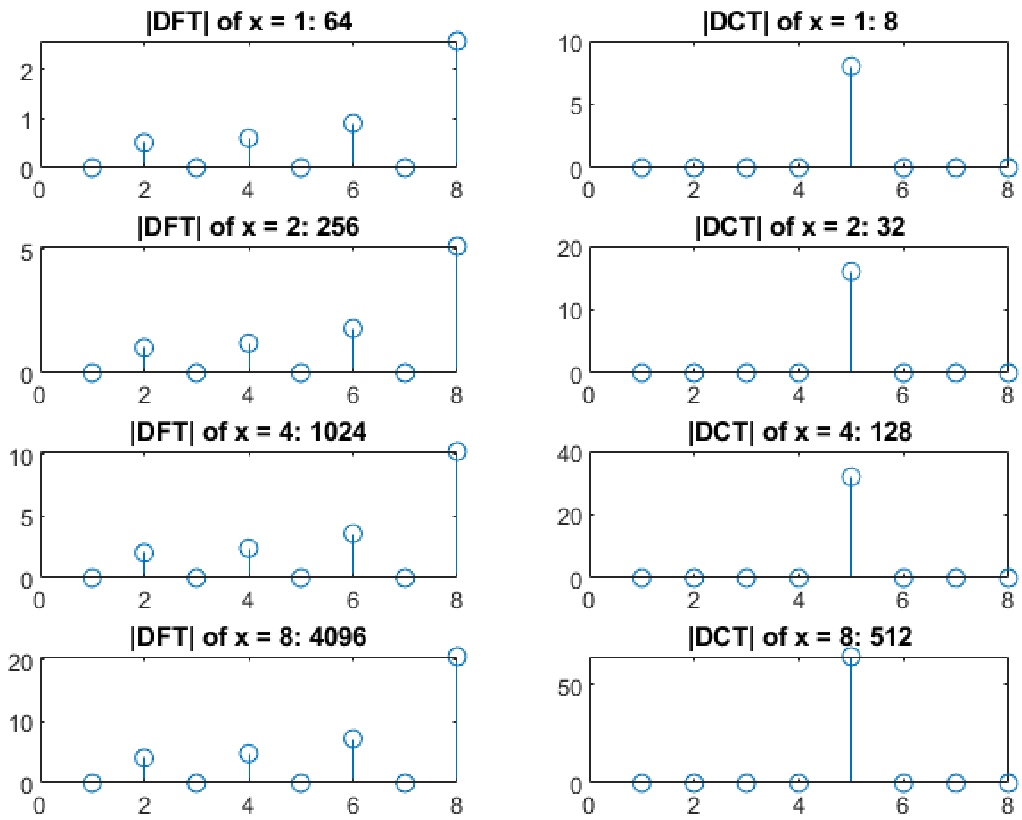
رسم شکل موجهای سطرهای ماتریس D:

```
for i=1:8
    subplot(8,1,i);
    plot(D(i,:));
end
```

بدست آوردن تبدیل DFT و DCT برای yهای گفته شده:

```
figure;
counter=1;
for x=0:3
    y = [2^x, -1*2^x, 2^x, -1*2^x, 2^x, -1*2^x, 2^x, -1*2^x];
    dct_ = D*y';
    dft_ = fft(y);
    subplot(4,2,counter);
    stem(dct_);
    title(['|DFT| of x = ',num2str(2^x),': ',num2str(sum(dft_.*dft_))]);
    counter = counter + 1;
    subplot(4,2,counter);
    stem(dft_);
    title(['|DCT| of x = ',num2str(2^x),': ',num2str(sum(dct_.*dct_))]);
    counter = counter + 1;
end
```

اندازه ی ماتریس‌های حاصل از تبدیل DFT و DCT در شکل مشخص شده است و مقادیر ضرایب به دست آمده را میتوان در تصویر زیر مشاهده نمود:



شکل ۲۵-شکل ضرایب تبدیل DFT و DCT

با بررسی حاصل دو عبارت |DCT| و |DFT| نتیجه میگیریم حاصل |DCT| برابر با مجموع مربعات عناصر آرایه‌ی y است و حاصل عبارت |DFT| برابر با مجموع دو برابر مربعات عناصر آرایه‌ی y می‌باشد.