بنام خدا


دانشگاه صنعتی اصفهان

دانشکده برق و کامپیوتر


# تمرین سری هفتم – مباحث ویژه در پردازش سیگنال های دیجیتال


استاد: دکتر سعید صدری

پروانه رشوند ۹۴۱۰۱۲۴

رضا سعادتی فرد ۹۴۱۱۳۹۴

آرزو فرزانفر ۹۴۱۴۷۲۴


تاریخ تحویل ۹۵/۰۳/۹

## سوال ۱:

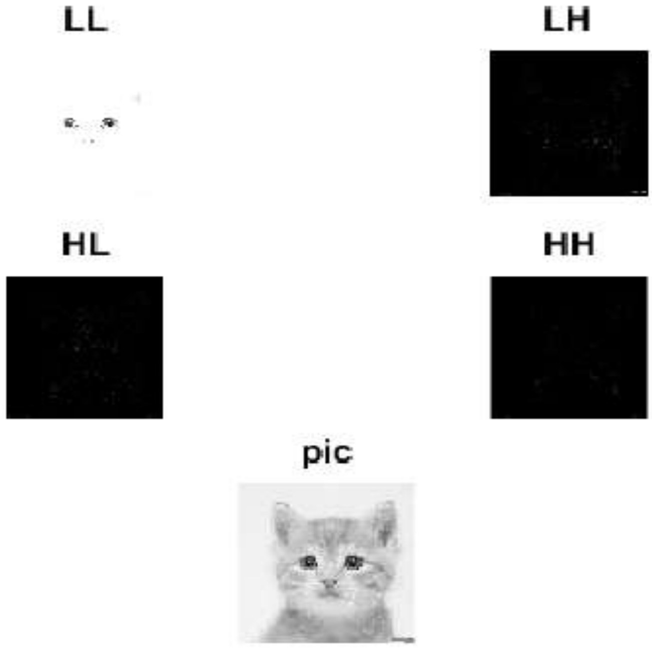
در این سوال قرار است روی یک تصویر گربه، پردازش انجام دهیم. تصویر مورد نظر را در متلب خوانده و توسط دستور زیر، تصویر خاکستری شده و سپس در بازه ی صفر تا یک قرار می گیرد:

```
clc;
clear;
close all;

%%%%%% Prob1 %%%%%
wname='db2';

a=imread('cat.jpg');
a=rgb2gray(a);
a=im2double(a);

a = (a-min(a(:))) ./ (max(a(:)-min(a(:)))); %%%%     SCALED A BETWEEN 0-1
```

سپس با استفاده از دستورات زیر، ضرایب LL، LH، HH و HL را به دست می آوریم.

```
%%% Decomposition

[C,S] = wavedec2(a,1,wname);

cA=appcoef2(C,S,wname,1);             %% get Approximation Coeffitents(LL)

[cH,cV,cD] = detcoef2('all',C,S,1);   %% get Horizental Vertical & Diogonal
Coefficents

% %%%% PLOTTING COMPONENTS

figure;
subplot(321)
imshow(cA); title('LL');
subplot(322)
imshow(cH); title('LH');
subplot(323)
imshow(cV); title('HL');
subplot(324)
imshow(cD); title('HH');
subplot(3,2,[5,6])
imshow(a); title('pic');
```

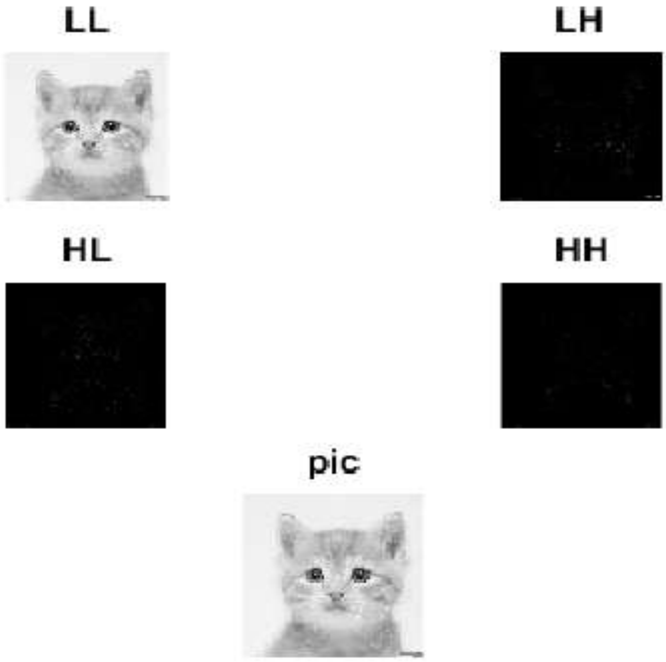
با ران کردن کد بالا و با رسم ضرایب، نتیجه زیر را مشاهده می نماییم:




برای اینکه، نتیجه ی برنامه، واقعی تر شود، باید CA( در کد بالا) نیز بین ۰ و ۱ قرار گیرد چرا که تصویر اصلی را در ابتدای برنامه بین صفر و یک قرار داده ایم. برای این کار از همان متد اولیه در برنامه استفاده می نماییم یعنی کد زیر را به برنامه اضافه می نماییم:

```
cA = (cA-min(cA(:))) ./ (max(cA(:)-min(cA(:)))); %%%%  SCALED A BETWEEN 0-1
```

اگر این کار را انجام دهیم شکل به صورت زیر در خواهد امد:



با مشاهده ی تصاویر HL ،HH ،LH نتیجه می شود که تقریبا سیاه میباشند این به این خاطر است که این ضرایب حداقل یک بار از یک فیلتر بالا گذر عبور کرده اند و از آنجا که ضرایب فیلتر بالاگذر، قرینه ی یکدیگر می باشند در حین جمع شدن، صفر می شوند که در واقع، سیاه می شوند و به همین دلیل، کلیت تصاویر سیاه می باشد.

در ادامه می خواهیم ابتدا میانگین و واریانس ضرایب را برای هر سه ماتریس به دست آوریم و سپس با توجه به فرمول :

$$\lambda = \text{میانگین} + k(\text{انحراف معیار})$$

یک آستانه تعیین کنیم همچنین ضریب k را باید طوری تعیین کنیم که در حین آستانه گداری، ۸۰ درصد ضرایب صفر شوند. به عبارت بهتر باید ضریب تراکم طبق تعریف زیر ، ۰٫۸ شود:

$$\text{ضریب تراکم} = \frac{\text{تعداد عناصر صفر سه ماتریس پس از آستانه گذاری}}{\text{تعداد کل عناصر سه ماتریس}}$$

```
%%%% calculate mean and std and determin threshold
k=.۶;
mean_cH=mean(cH(:)); std_cH=std(cH(:));
Th_cH= mean_cH + k * std_cH;
cH_S=wthresh(cH,'s',Th_cH);      %% soft Thresholding
cH_H=wthresh(cH,'h',Th_cH);      %% hard Thresholding


mean_cV=mean(cV(:)); std_cV=std(cV(:));
Th_cV= mean_cV + k * std_cV;
cV_S=wthresh(cV,'s',Th_cV);      %% soft Thresholding
cV_H=wthresh(cV,'h',Th_cV);      %% hard Thresholding

mean_cD=mean(cD(:)); std_cD=std(cD(:));
Th_cD= mean_cD + k * std_cD;
cD_S=wthresh(cD,'s',Th_cD);      %% soft Thresholding
cD_H=wthresh(cD,'h',Th_cD);      %% hard Thresholding

Density_Ratio = (length(find(cH_S(:)==0)) + length(find(cV_S(:)==0)) +
length(find(cD_S(:)==0)))/(3*length(cH(:)));
```

برای مقادیر مختلف k ، ضریب تراکم را حساب می کنیم تا ببینیم در چه حالتی، ضریب تراکم مقدار ۰٫۸ خواسته شده در صورت سوال را پیدا میکند. در حالت k=0.6 و با اجرای دستور بالا، ۰٫۸۱ خواهد شد. برای حالت k=0.7، این مقدار برابر با ۰٫۸۳ و برای K=0.8 این مقدار برابر ۰٫۸۵ خواهد بود. مشاهده می شود که با افزایش k ضریب تراکم افزایش می یابد. بنابراین با در نظر گرفتن مقدار ۰٫۶ برای k و به دست آمدن مقدار ۰٫۸۱ برای ضریب تراکم، آستانه گذاری نرم و سخت را انجام داده البته انتظار می رود که تغییر چندانی مشاهده نشود چرا که همانطور که در تصاویر بالا مشاهده نمودیم، بیس تصاویر سیاه است یعنی اکثر ضرایب صفر می باشند و لذا با آستانه گذاری و صفر نمودن ۸۰ درصد این ضرایب که عمدتا از قبل سیاه بوده اند، نباید تغییر چندانی مشاهده نماییم. نتیجه مشاهده شده در حین آستانه گذاری سخت و نرم، پس از ران کردن برنامه، بدین صورت خواهد بود:


همانطور که از قبل نیز انتظار داشتیم، مشاهده می شود با وجود اینکه هشتاد درصد ضرایب صفر شده اند، اما به دلیل اینکه ضرایب از قبل هرکدام حداقل یک بار از یک فیلتر بالا گذر عبور کرده بودند و خودشان صفر بودند، با صفر کردن هشتاد درصد ضرایب، تصویر تغییر چندانی نکرده است.

در ادامه، در ابتدا ضرایب تصویر اصلی و سپس ضرایبی که از آستانه عبور کرده اند، را نرمالیزه کرده.،(می دانیم که برای نرمالیره کردن مقدار مثلا **x**، در واقع باید مقدار $\frac{x}{\sqrt{\|x\|}}$ را حساب نماییم). سپس، بعد از محاسبه مقدار نرمالیزه ی ضرایب تصویر اصلی و ضرایب آستانه گذاری شده، آنها را از هم کم نموده و مجددا پس از محاسبه ی نرم آن مقدار خطا را به صورت $\epsilon = \|a-\hat{a}\|^2$ محاسبه می نماییم که a ماتریس ضرایب تصویر اصلی و $\hat{a}$ مربوط به ضرایب آستانه گذاری شده، می باشد.

با کمک کد زیر، خطا را محاسبه می نماییم:

```
%%% Normalize
norm_a=norm(a);
norm_ST_a=norm(ST_a);
norm_HT_a=norm(HT_a);

a_n=a/sqrt(norm_a);
ST_a_n=ST_a/sqrt(norm_ST_a);
HT_a_n=HT_a/sqrt(norm_HT_a);

e_ST=norm(a_n-ST_a_n);      %%% error of soft Thresholding
e_HT=norm(a_n-HT_a_n);      %%% error of hard Thresholding
```

پس از ران کردن برنامه، مقدار خطا برای آستانه گذاری نرم،0.0196 و برای آستانه گذاری سخت، 0.01084 می باشد. مشاهده می شود که در این مورد، آستانه گذاری سخت، خطای کم تری دارد و در نتیجه بهتر عمل می کند. زیرا در این سوال، تصویر ما فاقد نویز است و می دانیم که آستانه گذاری سخت در واقع سیگنال اصلی را به ما می دهد در صورتی که آستانه گذاری نرم، مقادیری کمتر از سیگنال اصلی را به ما می دهد. بنابراین، آستانه گذاری نرم تنها در حضور نویز بهتر عمل می کند در صورتی که در این سوال، نویز وجود ندارد و بنابراین همانطور که مشاهده شد، آستانه گذاری سخت بهتر عمل نمود و خطای کمتری به دست آمد.

همچنین، با اجرای مراحل بالا، برای db6 و sym6 مشاهده می نماییم که خطا و تصاویر به صورت زیر خواهد بود:

برای sym6 :

مقدار خطای آستانه گذاری نرم، ۰,۱۸۹۷و مقدار خطای آستانه گذاری سخت، ۰,۰۱۰۷۹ می باشد.

و تصاویر بدین شکل خواهد بود:


برای db6 :

مقدار خطای آستانه گذاری نرم، ۰,۱۸۹۶و مقدار خطای آستانه گذاری سخت، ۰,۰۱۰۶۷ می باشد.

و تصاویر بدین شکل خواهد بود:


با توجه به مقادیر به دست آمده برای خطا مشاهده می شود که db6 بهتر عمل می نماید. البته تفاوت ها آن چنان زیاد و قابل مشاهده نمی باشند.

## سوال ۲:

### الف)

۰<λ<۱

$\nu = 0.5 = t$

$f(t)=1-|0.5-t|^{\lambda}$ (*)

میدانیم که تابع فوق در t=0.5مشتق پذیر نمی باشد.پس:

$f(\nu)=1$

پس:

$$|f(t)-1| < k\,|t-0.5|^{\alpha} \overset{(*)}{\Rightarrow}$$
$$|0.5-t|^{\lambda} < k|t-0.5|^{\alpha}$$

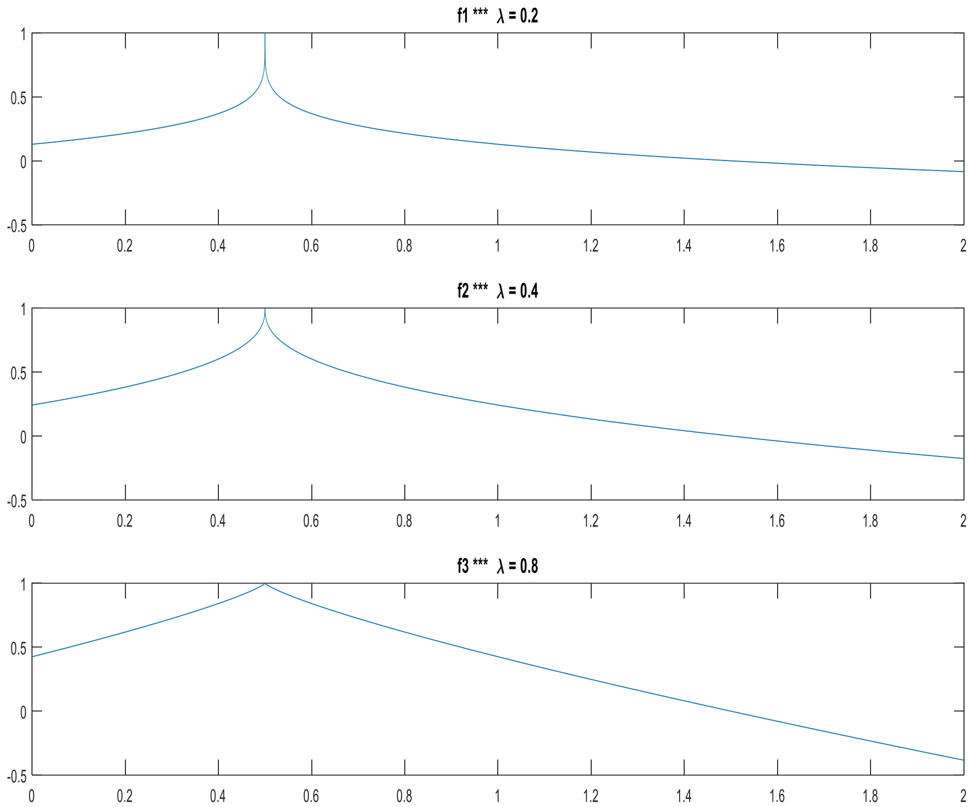
فقط به ازای ۰<λ<۱نامساوی فوق برقرار است. در نتیجه با توجه به مطالب عنوان شده در درس می توان عنوان داشت که تابع f(t) ، λ_Lip است.

### ب )

در این قسمت ، f(t) فوق را به ازای λ های ۰,۲ ، ۰,۴ و ۰,۸ بدست می آوریم و آن ها را در یه شکل کنار هم رسم میکنیم.

کد متلب و شکل های مربوطه به صورت زیر می باشند:

```
clc;
clear;
close all;

%%%%% Prob 2 %%%%%%%

step = 0.0005; f_d=0; f_u=2;
t_f = f_d : step : f_u ;
f=zeros(length(t_f),3);
for i=1:3
    landa=.1 * 2^i;
    f(:,i)= 1- abs(0.5 - t_f).^landa;
end
f1=f(:,1); f2=f(:,2); f3=f(:,3);

figure;
subplot(311), plot(t_f,f1), title('f1 ***  \lambda = 0.2');
subplot(312), plot(t_f,f2), title('f2 ***  \lambda = 0.4');
subplot(313), plot(t_f,f3), title('f3 ***  \lambda = 0.8');
```




## ج)

در این قسمت از سوال هدف تخمین میزان سینگولاریته بودن نقاط سینگولار تابع f1 , f2 , f3 می باشد .

برای این کار ما از قضیه jafard و مشخصه modulus maxima استفاده می کنیم.

با استفاده از مطالب عنوان شده در درس می دانیم که برای تخمین α (میزان سینگولاریته در نقطه مورد نظر) می توان از رابطه زیر استفاده کرد: (با استفاده از قضیه jafard)

$$Log\left(\left|\begin{matrix} WT(u \; and \; s) \\ f \; \psi \end{matrix}\right|\right) = log(A) + (\alpha+\tfrac{1}{2}) \, log(s)$$

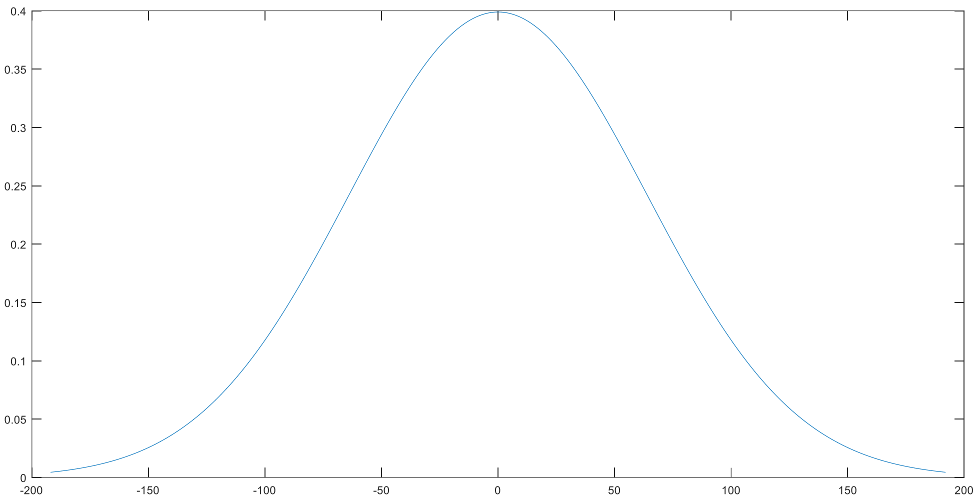
در نتیجه برای بدست آوردن مشخصه میزان α می توان اندازه لگاریتم ضرایب موجک را بر حسب لگاریتم S ها (اسکیل) های مختلف رسم کرد و شیب نمودار را به دست آورد این شیب برابر میشود با $\alpha+\frac{1}{2}$ در نتیجه $\alpha = m - \frac{1}{2}$

در حل این مساله ما از یک نوآوری استفاده کردیم که این امر باعث شد زمان ران کردن برنامه تا حد بسیار زیادی کاهش باید و جواب ما کماکان به خوبی صحت خود را حفظ کند. که این نوآوری در ادامه توضیح داده خواهد شد.

در این سوال از ما خواسته شده است که s را از مقدار ۱ تا ۶۴ یک واحد یک واحد افزایش دهیم و به ازای این ۶۴ اسکیل مختلف مشخصه modulus maxima را رسم کنیم.

هنگامی که برای مثال s=64 باشد با توجه با اینکه پهنای زمانی سیگنال گوسی ( برای σ=۱) تقریبا از ۳- تا ۳+ است حدودا ۶۴ برابر می شود.

که در شکل زیر شاهد آن هستیم:



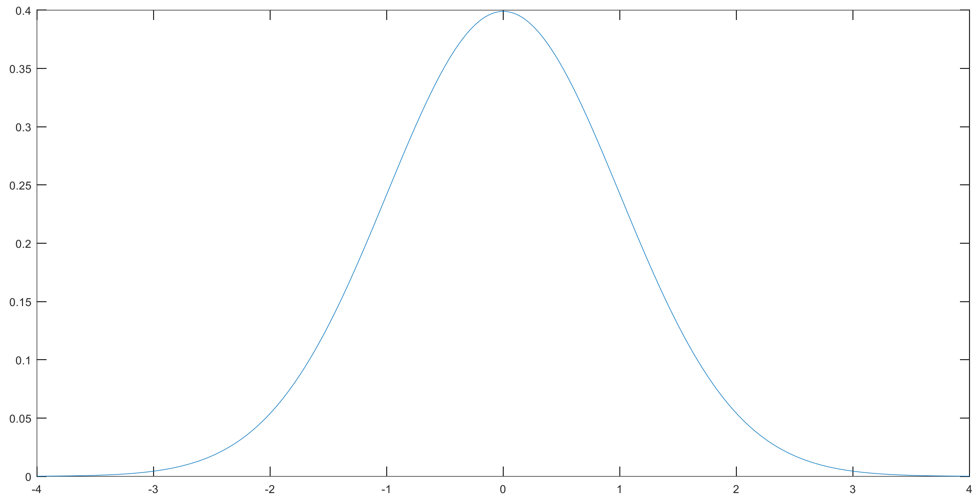

حال برای بدست آوردن ضرایب تبدیل موجک مشتق اول تابع گوسی ، باید اسکیل های مختلف تابع گوسی را با ***f(t)*** کانولوشن کنیم سپس یک مرتبه مشتق بگیرم .با توجه به اینکه برای اعداد اسکیل بزرگ (برای مثال ***s***=۶۴) و همچنین اینکه میزان ***step*** زمانی ما برابر مقدار بسیار کوچک ۰,۰۰۰۵ می باشد ، میزان زمان محاسبه کانولوشن بسیار طول می کشد .به همین علت برای محاسبه ضرایب موجک بجای استفاده از رابطه زیر (با توجه به مطالب بیان شده در درس می توان برای محاسبه ضرائب موجک مشتق اول گوسی ابتدا تابع گوسی را با تابع ***f(t)*** کانولوشن کرد سپس از حاصل آن مطابق رابطه زیر مشتق گرفت)

$$WT'(\mathrm{u,s}) = \mathrm{f(u)} \ * \frac{1}{\sqrt{s}}\ \psi(\frac{-u}{s}) \xrightarrow[\text{}]{\substack{\psi(\mathrm{t}) is\ Gussian \\ and\ we\ need \\ the\ firsrt\ Deivative\ of\ Gussian\ as\ wavelet}} \mathrm{WT(u,s)}= -s * \frac{d}{dt}(WT'(\mathrm{u\ s}))$$

حال ما در این قسمت یک تغییر کوچک اعمال کردیم که باعث شد زمان ران برنامه که حدود ۳۰ دقیقه طول می کشید به چند ثانیه کاهش یابد ، این تغییر به این صورت در رابطه فوق اعمال شد:

$$\xrightarrow{new\ Modify\ on\ scale\ factor} \quad WT'(\mathrm{u,s}) = \mathrm{f(u)} \ * \sqrt{s}\,\psi(-u*s) \ \Rightarrow\ \mathrm{WT(u,s)}= -\frac{1}{s} * \frac{d}{dt}(WT'(\mathrm{u\ s}))$$

این معکوس کردن اسکیل باعث می شود که تابع گوسی ما در بزرگ ترین حالت برای S=1 رخ دهد



مشاهده می شود در این حالت تا حد زیادی عرض سیگنال گوسی در حالت بیشینه، نسبت به حالت قبل کاهش یافته ($\frac{1}{64}$) که این امر میزان نقاطی که باید عمل کانولوشن روی آنها انجام شود را تا حد بسیار زیادی کاهش میدهد و سرعت برنامه به مراتب افزایش می یابد.

تنها تفاوتی که در تحلیل های ما ایجاد می شود این است که با افزایش S بجای اینکه منحنی modulus maxima افزایش یابد ، کاهش می یابد و در نتیجه شیب منحنی ما منفی می شود که برای مقابله با این تفاوت ایجاد شده ، کافی است مقادیر modulus maxima را در یک منفی ضرب کرد تا شیب مثبت شود در این شرایط دقیقا جواب ها مشابه حل معمولی مسئله می گردد.

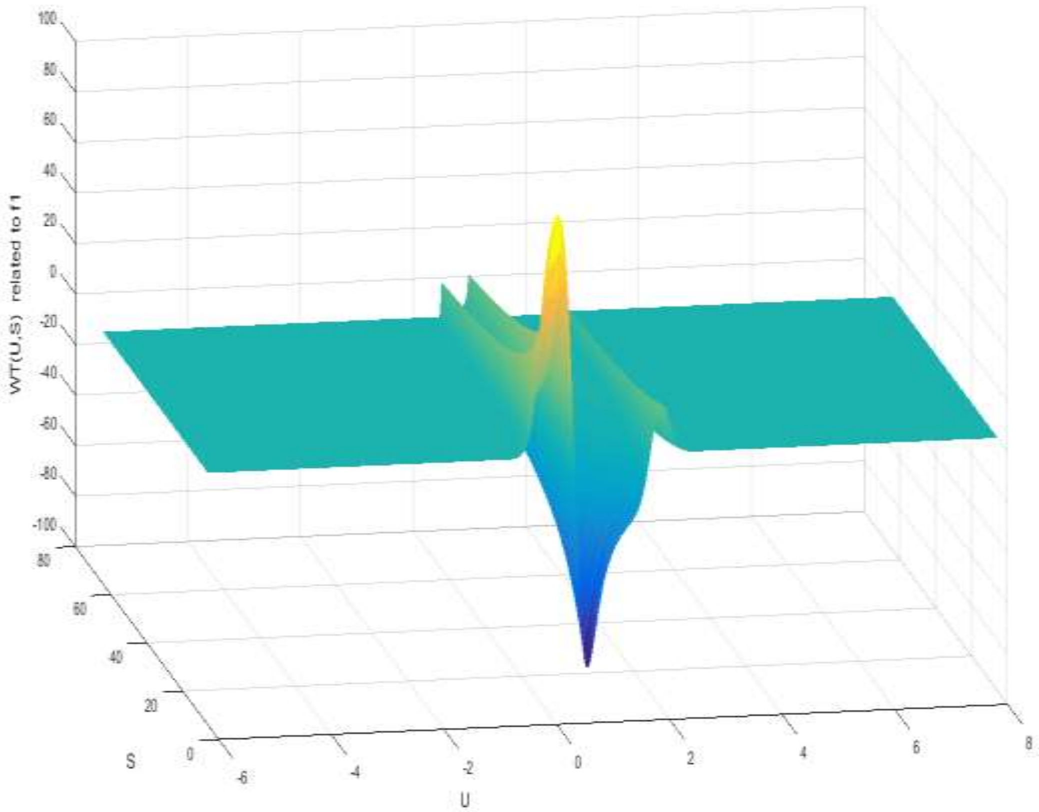
در زیر کد مربوط برای بدست آوردن ضرایب تبدیل موجک و همچنین مقدار ماکسیموم ضرائب تبدیل را برای رسم مشخصه modulud maxima مشاهده میکنیم(این کد بر اساس روش پیشنهادی و بهینه فوق نوشته شده است)

```
Gu_d=-6; Gu_u=6;                    %% time domain for Gussian Function
t_Gu=Gu_d:step:Gu_u;

t=Gu_d + f_d :step: Gu_u + f_u;       %% time domain for conv(f,gussian)

for s=1:64

Gu=(sqrt(s)/sqrt(2*pi)).*exp(-(t_Gu/(sqrt(2))*s).^2);

%%%% f1
f1_conv(s,:) =    conv(f1,Gu);  %%  calculate f(u) * (sqrt(s)×Psi((-u*s))

f1_conv_d1(s,:) = 1/s*(gradient(f1_conv(s,:))/step); %% first derivative of f1

singular_index=find(f1_conv(s,:)==max(f1_conv(s,:))); %% obtain the index's
number of t= 0.5

max_WT_f1_d1(s)= max(f1_conv_d1(s,(singular_index-500:singular_index+500)));
%% caclulate Local Max for t=.5
```

```
%%%% f2
f2_conv(s,:) = conv(f2,Gu);  %%  calculate f(u) * (sqrt(s)×Psi((-u*s))

f2_conv_d1(s,:) = 1/s*gradient(f2_conv(s,:))/step; %% first derivative of f2

singular_index=find(f2_conv(s,:)==max(f2_conv(s,:)));
max_WT_f2_d1(s)= max(f2_conv_d1(s,singular_index-500:singular_index+500));

%%%% f3
f3_conv(s,:) = conv(f3,Gu);
f3_conv_d1(s,:) = 1/s*gradient(f3_conv(s,:))/step;

singular_index=find(f3_conv(s,:)==max(f3_conv(s,:)));
max_WT_f3_d1(s)= max(f3_conv_d1(s,singular_index-500:singular_index+500));
end
```

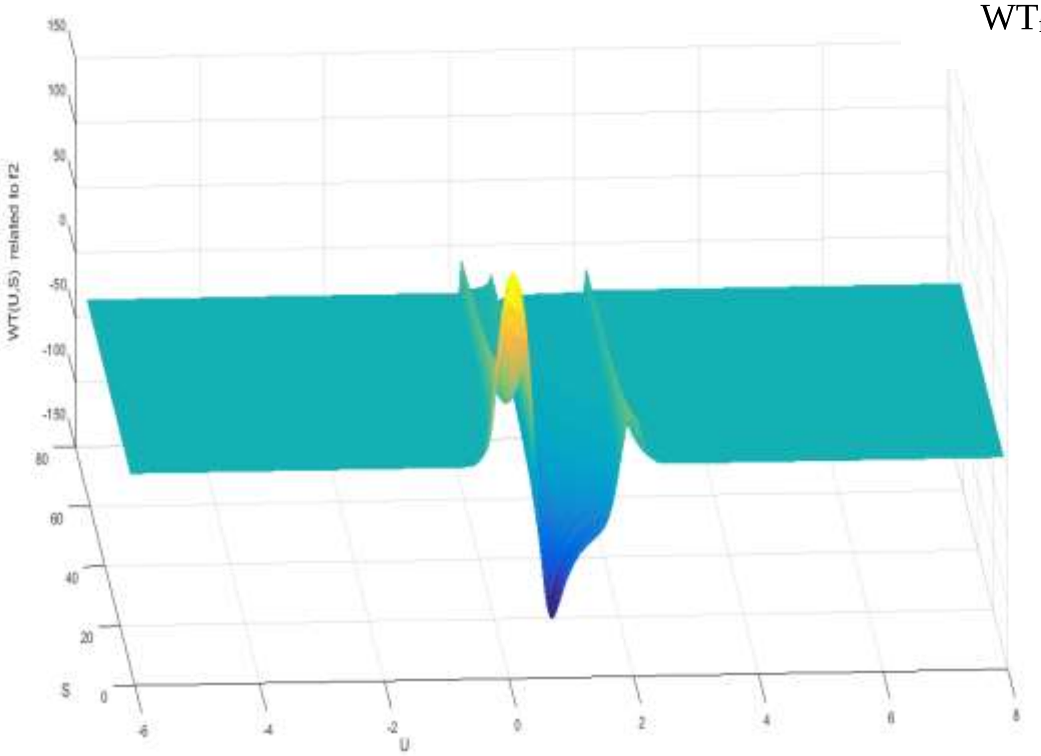
در زیر کد و نمایش سه بعدی ضرائب تبدیل موجک را برای مشتق اول گوسی(بر حسب u , s) را مشاهده میکنیم:

$WT_{f\,1,\psi}(u,s)$

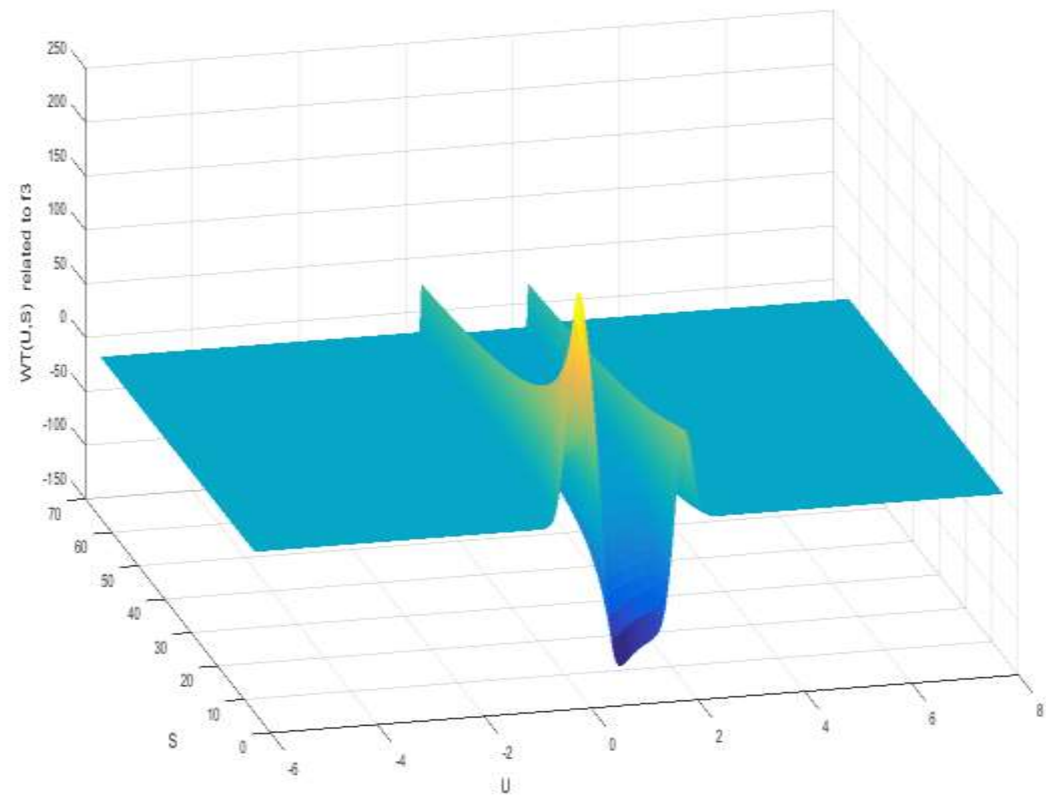



```
%%% Mesh plot for wavelet coeeficients based on u , s
figure;
s=1:64;
[U,S]=meshgrid(t,s);
mesh(U,S,f1_conv_d1);
xlabel('U');ylabel('S');zlabel('WT(U,S)  related to f1')
```

$WT_{f\,2,\,\psi}(u,s)$



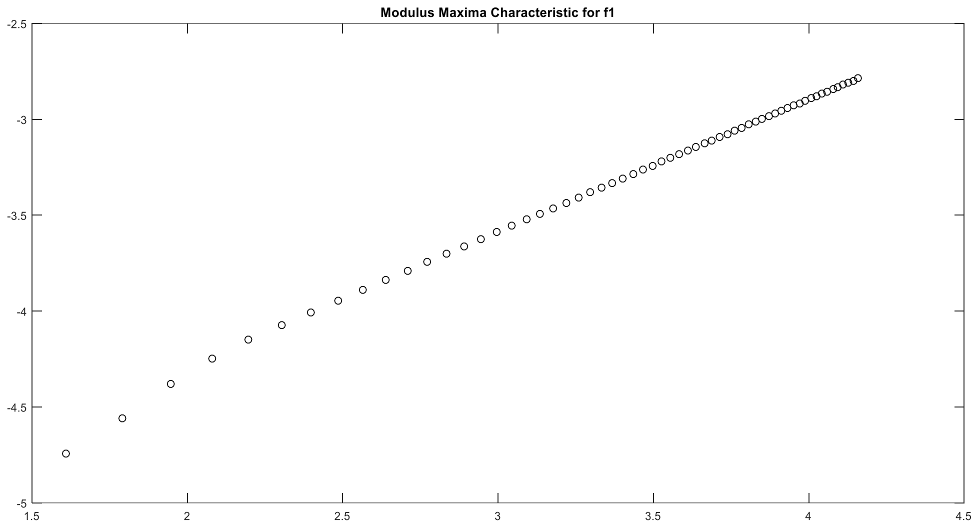

$WT_{f\,3,\,\psi}(u,s)$

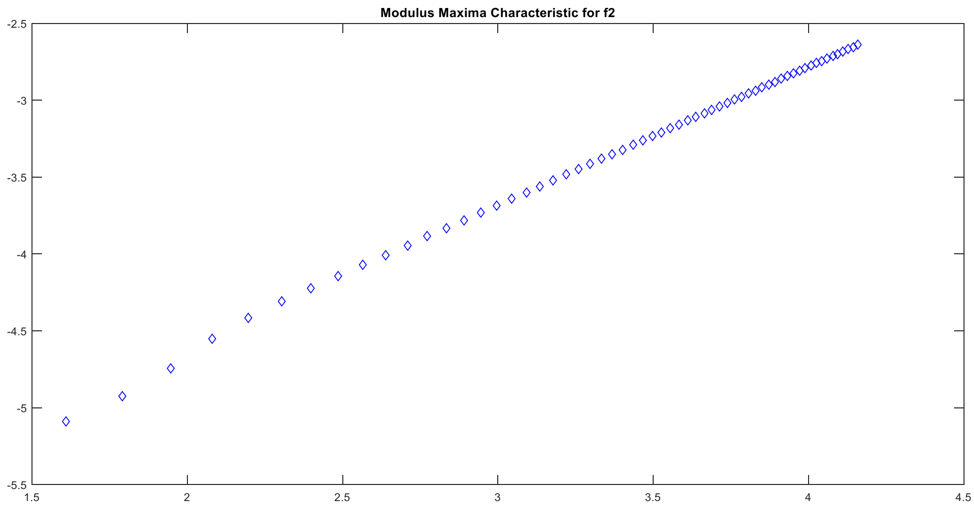


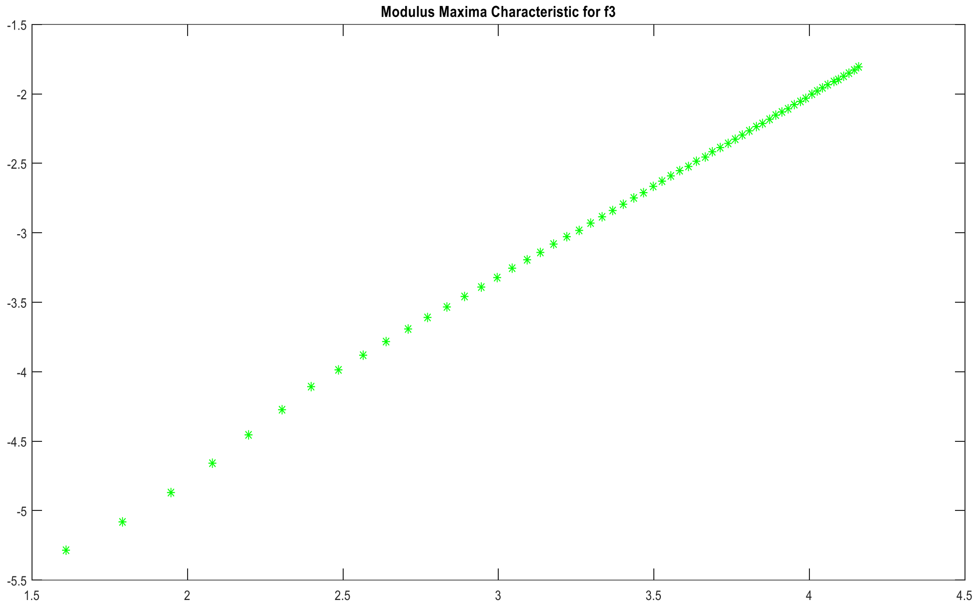
حال از ما خواسته شده است مشخصه  modulus maxima را رسم کنیم برای اینکه لگاریتم بیشینه مقدار ضرائب ویولت را(کافی است تنها در مخروط نفوذ) که در کد قسمت قبل محاسبه کردیم را بر حسب  log(s) را رسم میکنیم(همان طور که در سه شکل فوق مشهود است این مقدار ماکسیموم در cone of influence رخ می دهد)

در زیر کد و منحنی  Modulus maxima را برای f1 , f2 , f3 شاهد هستیم. (توجه شود منفی موجود در کد زیر به علت روش پیشنهادی و فوق الذکر می باشد)

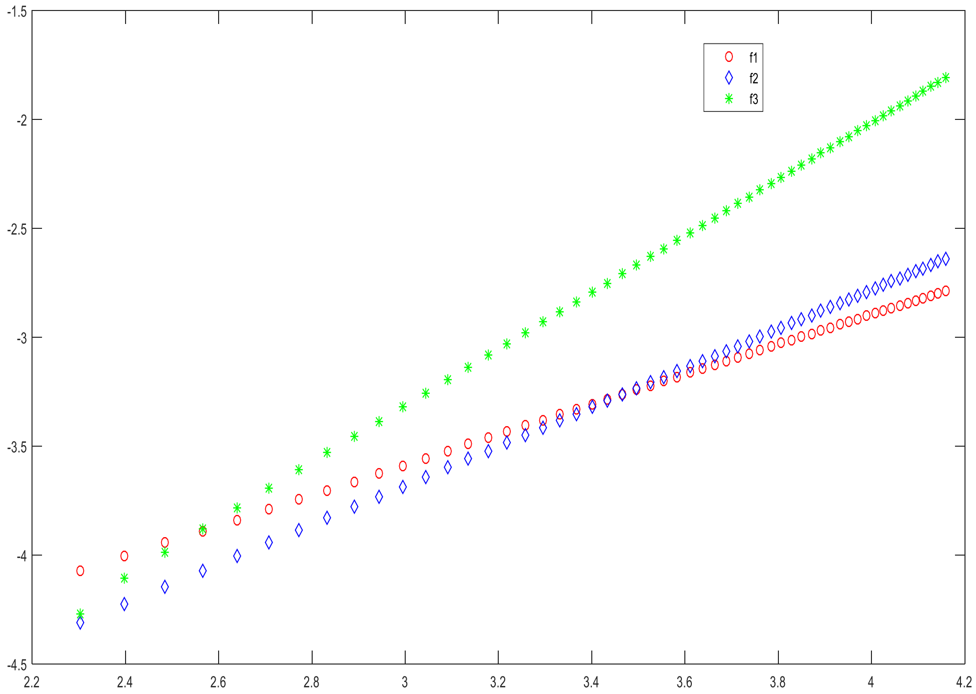
```
log_s = log(1:64);
log_WT_f1= -log(max_WT_f1_d1);
log_WT_f2= -log(max_WT_f2_d1);
log_WT_f3= -log(max_WT_f3_d1);

figure;
plot((log_s(4:64)),log_WT_f1(4:64),'ko')
title('Modulus Maxima Characteristic for f1')
```






در زیر منحنی modulus maxima سه تابع را با هم مشاهده میکنیم که رنگ قرمز f1 رنگ آبی f2 و رنگ سبز f3 می باشد.




بدون محاسبه دقیق شیب و تنها بر اساس نمودار فوق کاملا مشهود است که بیشترین شیب متعلق به f3 و کمترین شیب متعلق به f1 است(مشاهده شد در صورت استفاده از چند نمونه اولیه مقداری خطا وجود دارد و باعث می شود خط ما از حالت درجه اول اندکی فاصله بگیرد به همین علت ما در رسم و تخمین $\lambda$ از ۶۰ نمونه از ۶۴ نمونه موجود استفاده کردیم و از چند ۴-۵ نمونه اول استفاده نکردیم هر چند با وجود این نمونه ها باز هم تخمین مناسبی شاهد بودیم اما برای کاهش خطا چند

نمونه اول را در تخمین دخالت ندادیم. علت این امر احتمالا به این دلیل است که در روش پیشنهادی با افزایش s ، طول موجک ما کاهش می یابد و درنتیجه هنگامی که با تابع f(t) کانولوشن میشود رزولوشن زمانی بهتری دارد اما هنگامی که s برای مثال ۱ یا ۲ است چون پهنای زمانی زیادی دارد(برای S=1 حدودا ۶ است ) قدرت تفکیک زمانی مناسب ندارد و به خوبی نمیتواند سینگولاریته نقطه را مشخص کند)

حال باید با استفاده از منحنی Modulus Maxima و محاسبه شیب آن و طبق روابط بیان شده می توان میزان سینگولاریته و $\lambda$ را تخمین زد.

برای اینکار و بدست آوردن شیب منحنی های فوق ،کافی است نمودار های فوق را با یک منحنی درجه اول برازش کرد برای اینکار در matlab از دستور polyfit می توان استفاده کرد. سپس میزان شیب بدست آمده را از ۰,۵ کم می کنیم عدد بدست آمده برابر است با میزان $\lambda$ .

کد متلب و مقادیر تخمین زده را مشاهده میکنیم:

```
%%%% estimation lambda for f1
m_1=polyfit(log_s(4:64),log_WT_f1(4:64),1);
alpha_1=(m_1(1)-.5);
%%%% estimation lambda for f2
m_2=polyfit(log_s(4:64),log_WT_f2(4:64),1);
alpha_2=(m_2(1)-.5);
%%%% estimation lambda for f3
m_3=polyfit(log_s(4:64),log_WT_f3(4:64),1);
alpha_3=(m_3(1)-.5);
```

| | F1 | F2 | F3 |
|---|---|---|---|
| $\lambda$ | ۰,۲ | ۰,۴ | ۰,۸ |
| مقدار تخمین زده شده | ۰,۱۹۲۲ | ۰,۳۹۹۰ | ۰,۸۰۵۰ |

مشاهده می شود روش پیشنهادی با صرف زمان بسیار کمتر توانسته به خوبی میزان سینگولاریته را تخمین بزند (برای سایر $\lambda$ های بین ۰-۱ ، از این روش برای تخمین $\lambda$ استفاده کردیم که با دقت کافی توانست میزان سینگولاریته را تخمین بزند.)

## سوال ۳:

**الف)**

- **علت وجود τ**

به علت تاخیر زمانی ارسال و دریافت سیگنال بین رادار و فرستنده بوجود می آید( با فرض اینکه سرعت انتشار موج برابر سرعت نور باشد، و وجود فاصله مکانی بین فرستنده و گیرنده ، باعث ایجاد یک تاخیر می گردد . )

- **علت وجود S**

هم چنین s به خاطر پدیده داپلر (درصورتی که مسیر حرکت متحرک( سیگنال ارسالی ) به سمت رادار نزدیک شونده باشد اثر داپلر تاثیر مثبت و اگر متحرک نسبت به رادار دور شونده باشد ، داپلر اثر منفی دارد).

اگر بسامد موج تولید شده در منبع $\nu$ باشد و سرعت شنونده و منبع به ترتیب $v_o$ و $v_s$ باشد، بسامد موجی که شنونده می‌شنود، $\nu'$، از رابطه زیر به دست خواهد آمد:

$$\nu' = \nu\left(\frac{v \pm v_o}{v \mp v_s}\right)$$

**ب)**

**ارایه روشی برای تخمین مقادیر τ و s:**

در ابتدا s تخمین زده میشود:

اگر فرض شود که سیگنال دریافتی ما f(t) باشد برای تخمین s با استفاده از رابطه انژی سیگنال دریافتی و موجک ارسالی می توان نوشت:

$$\int_{-\infty}^{+\infty} f(t)f^*(t)\,dt \xrightarrow{\text{با فرض حقیقی بودن سیکنال دریافتی}} \int_{-\infty}^{+\infty} f(t)^2\,dt$$

$$\xrightarrow{\text{از طرفی}} \int_{-\infty}^{+\infty} \psi\left(\frac{t}{s}\right) * \psi\left(\frac{t}{s}\right) dt = s \times \int_{-\infty}^{+\infty} \psi(t) * \psi(t) dt$$

$$\Rightarrow \quad s = \frac{\int_{-\infty}^{+\infty} \psi\left(\frac{t}{s}\right)*\psi\left(\frac{t}{s}\right)dt}{\int_{-\infty}^{+\infty} \psi(t)*\psi(t)dt}$$

می دانیم سیگنال دریافتی دارای یک تاخیر و یک اسکیل است همچنین می دانیم که تاخیر تاثیری در میزان انرژی ندارد (با توجه به اینکه پهنای زمانی موجک محدود است و ما برای محاسبه انرژی سیگنال انتگرال را از − بی نهایت تا + بی نهایت می گیریم ) می توان با توجه به روابط فوق نوشت

$$s = \frac{\int_{-\infty}^{+\infty} f(t)^2\, dt}{\int_{-\infty}^{+\infty} \psi(t) * \psi(t) dt}$$

حال$\tau$ رابا توجه به مقدار s تخمین زده شده ، تخمین میزنیم.

برای $\tau = 0$

$$\int_{-\infty}^{+\infty} \psi\left(\frac{t-\tau}{s}\right) * f(t)dt = \int_{-\infty}^{+\infty} \psi\left(\frac{t}{s}\right) * \psi\left(\frac{t}{s}\right) dt$$

اگر برای $\tau = 0$این رابطه برقرار نشد $\tau$رابه ترتیب افزایش میدهیم تا مقدار یک حاصل شود:

$$\int_{-\infty}^{+\infty} \psi\left(\frac{t-1}{s}\right) * f(t)dt = 1$$

همین روند را تا $\tau = k$ادامه میدهیم  تا نتیجه مورد نظر حاصل شود.

$$\int_{-\infty}^{+\infty} \psi\left(\frac{t-k}{s}\right) * f(t)dt = 1$$

بدین ترتیب مقدار شیفت مورد نظر هم تخمین زده میشود.

**شبیه سازی با استفاده از نرم افزار متلب و مشاهده نتیجه:**

در ابتدا کار با استفاده از روش پیشنهاد شده در فوق S را تخمین می زنیم و سپس به سراغ تخمین $\tau$ می رویم.

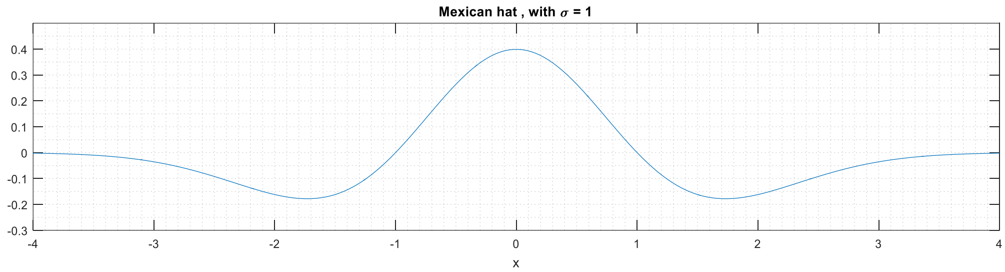
یک سیگنال به نام  f(t) در یافت کرده ایم و می دانیم سیگنال ارسالی ما یک موجک کلاه مکزیکی بوده است (ψ(t)) .

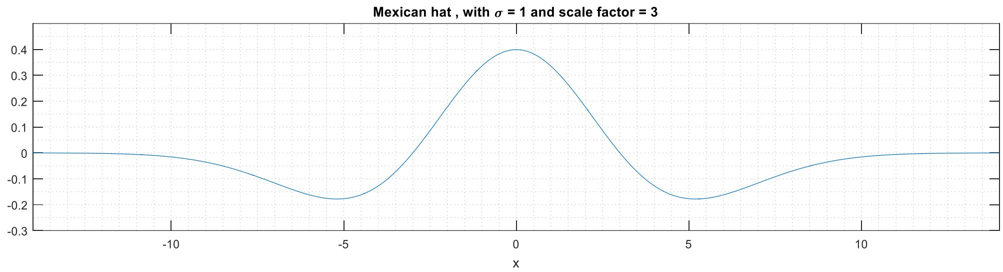
میدانیم انرژی موجک کلاه مکزیکی به صورت زیر است : (لازم به ذکر است در تمام محاسبات σ=۱  فرض شده استو این فرض به هیچ وجه عمومیت حل مسئله را از بین نمی برد)

$$\int_{-\infty}^{+\infty} \psi(t)\psi(t)^*\, dt \xrightarrow{\text{موجک حقیقی است}} \int_{-\infty}^{+\infty} |\psi(t)|^2\, dt = \int_{-\infty}^{+\infty} \left|\frac{1}{\sqrt{2\pi}} \times -1(1-\frac{t^2}{1})e^{-\frac{t^2}{2}}\right|^2 dt = 0.2116$$

حال هنگامی که این موجک با S اسکیل میشود پهنای زمانی آن S برابر می شود که با توجه با اینکه دامنه موجک بدون تغییر مانده این اسکیل زمانی باعث میشود مقدار انرژی آن S برابر میشود.

در زیر شکل موج کلاه مکزیکی را برای ضریب اسکیل ۱و ۳ مشاهده می کنیم مشاهده می شود پهنای زمانی برای اسکیل ۳ ، سه برابر ضریب اسکیل ۱ است



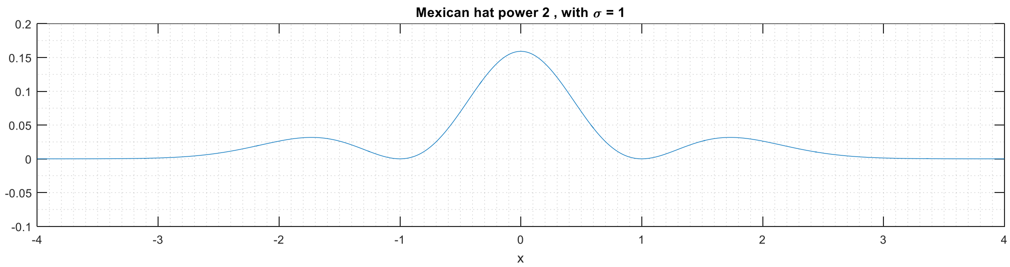



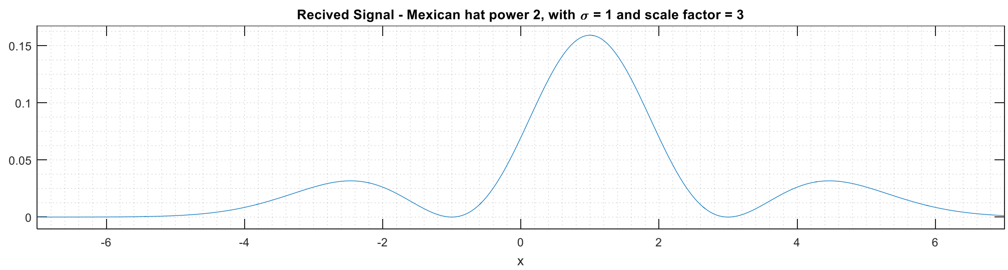

حال به محاسبه انرژی سیگنال f(t) می پردازیم:

$$\int_{-\infty}^{+\infty} f(t)\mathrm{f(t)}^{*}\ dt \xrightarrow{\text{موجک حقیقی است}} \int_{-\infty}^{+\infty} |\mathrm{f(t)}|^{2}\ dt \xRightarrow[\text{داریم}]{\text{با توجه به اینکه } f(t)=\psi(\frac{t-1}{2})}$$

$$\int_{-\infty}^{+\infty}\left|\frac{1}{\sqrt{2\pi}}\times -1(1-\frac{(\frac{t-1}{2})^{2}}{1})e^{-\frac{\frac{t-1}{2}^{2}}{2}}\right|^{2} dt\ = 0.4231$$

در زیر شکل سیگنال به توان دو مربوط به موجک کلاه مکزیکی و سیگنال دریافتی را مشاهده می کنیم:




حال با توجه به مطالب عنوان شده فوق مقدار اسکیل برابر است با:

$$\mathrm{S}=\frac{\int_{-\infty}^{+\infty}|\mathrm{f(t)}|^{2}\,dt}{\int_{-\infty}^{+\infty}|\psi(\mathrm{t})|^{2}\,dt}=\frac{0.4231}{0.2116}=2$$

در زیر کد متلب مربوطه را برای محاسبه و نمایش شکل های فوق مشاهده میکنیم:

```
clc;
clear;
close all;

%%%%%%%%%%%%%%%%%%%%%%%%%% Problem 3
syms x;
%%%%% Calculate Scale

%%%% Mexican hat
Mex_hat_p2 = @(x) (1/sqrt(2*pi)*(1-x.^2).*exp(-x.^2/2)).^2; %% mexican hat power 2
figure
subplot(211),ezplot(Mex_hat_p2,[-4 4 -.1 .2]), title('Mexican hat power 2 , with \sigma = 1') , grid
minor
int_psi_p2 = integral(Mex_hat_p2,-Inf,Inf);


%%%%% recived signal
s0=2; t0=1;
rec_ss = @(x) (1/sqrt(2*pi)*(1-(1/s0*(x-t0)).^2).*exp(-(1/s0*(x-t0)).^2/2)).^2; %% mexican hat shift 1;
scale 2;  power 2

subplot(212),ezplot(rec_ss,[-7  7 .2]), title('Recived Signal - Mexican hat power 2, with \sigma = 1 and
scale factor = 3'), grid minor

int_rec_sig = integral(rec_ss,-Inf,Inf);

S = int_rec_sig/int_psi_p2;
```

محاسبه تاخیر زمانی:

در این مرحله ، باید از مقدار اسکیل بدست آمده در مرحله قبل استفاده کنیم.

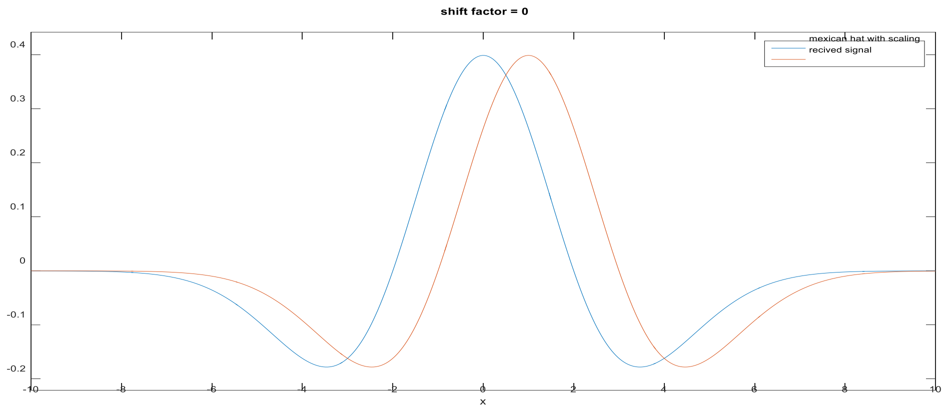
می دانیم مقدار اسکیل سیگنال دریافتی برابر ۲ است حال به محاسبه انرژی سیگنال $\psi(\frac{t}{2})$ می پردازیم

$$\int_{-\infty}^{+\infty} \psi(\tfrac{t}{2})\psi(\tfrac{t}{2})^* \; dt \quad \xrightarrow{\text{موجک حقیقی است}} \quad \int_{-\infty}^{+\infty} \left|\psi(\tfrac{t}{2})\right|^2 \; dt \; = \int_{-\infty}^{+\infty} \left|\frac{1}{\sqrt{2\pi}} \times -1(1-\frac{\frac{t^2}{2}}{1})e^{-\frac{\frac{t^2}{2}}{2}}\right|^2 \; dt \; =0.4231$$

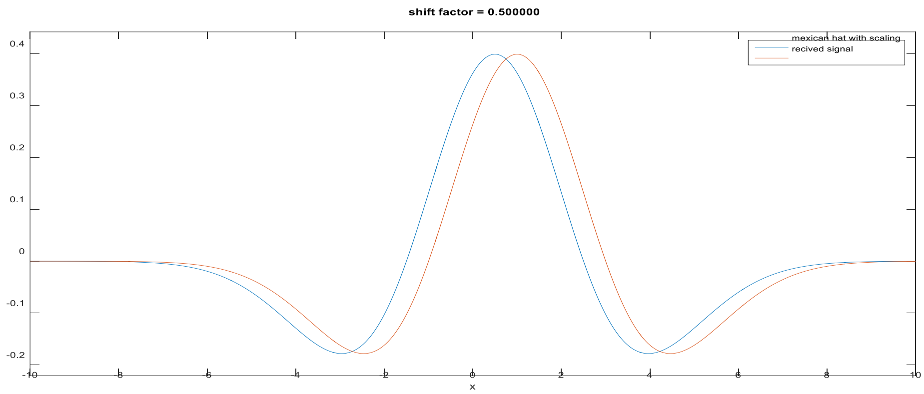
با توجه با اینکه شیفت زمانی در اندازه انرژی سیگنال بی تاثیر است میدانیم انرژی سیگنال دریافتی نیز برابر مقدار فوق است.

حال برای اینکه بتوانیم مقدار این تاخیز زمانی را بدست آوریم هر بار $\psi(\frac{t-\tau}{2})$ را برای تاخیر های بزرگتر مساوی ۰ (برای مثال از $\tau=0$ شروع میکنیم و و با استپ $\tau$۰،۰۱ را افزایش میدهیم) تا وقتی که اختلاف مقدار انتگرال $\int_{-\infty}^{+\infty} f(t)\psi(\frac{t-\tau}{2})^* \; dt$ از $\int_{-\infty}^{+\infty} \psi(\frac{t}{2})\psi(\frac{t}{2})^* \; dt$ از یک مقدار آستانه (فرضا $10^{-5}$) کوچکتر شود ، آن گاه میتوان نتیجه گرفت که سیگنال $\psi(\frac{t-\tau}{2})$ دقیقا روی f(t) قرار گرفته است و در نتیجه $f(t)= \psi(\frac{t-\tau}{2})$ میشود

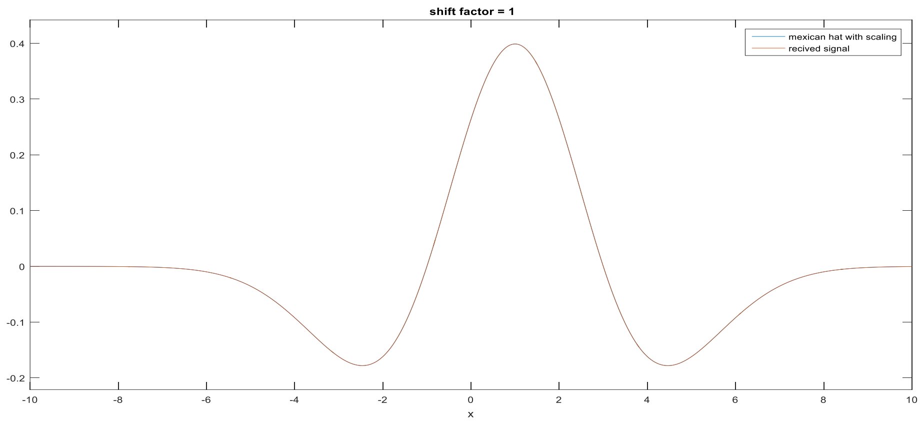
در زیر نمایش همزمان سیگنال دریافتی و سیگنال $\psi(\frac{t-\tau}{2})$ را به ازای $\tau$ های ۰، ۰،۵ ، ۱ و ۱،۵ مشاهده می شود.



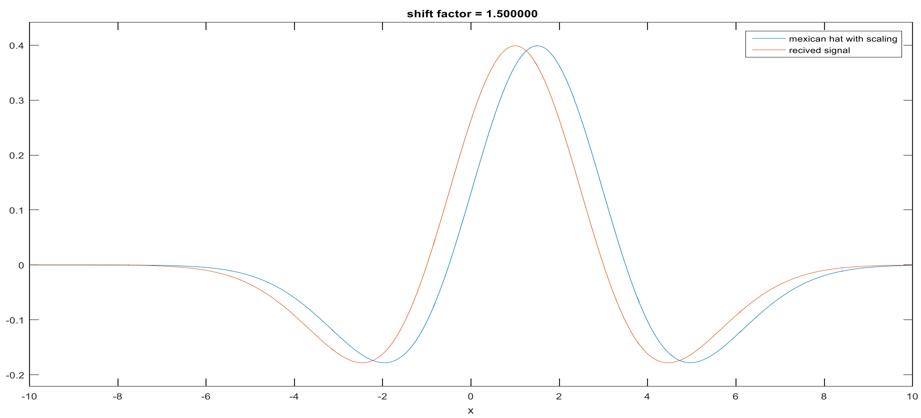

$\tau = 0$




$\tau = 0.5$




$\tau = 1$




$\tau = 1.5$

همانطور که در شکل های فوق مشهود است به ازای $\tau=1$ به طور کامل دو سیگنال بر روی یکدیگر قرار می گیرند.

```
%%%%% Calculate  Shift
Mex_hat_p2_s2 = @(x)  (1/sqrt(2*pi)*(1-(x/2).^2).*exp(-(x/2).^2/2)).^2; %% mexican hat power 2 , scale 2
int_psi_p2_s2 = integral(Mex_hat_p2_s2,-Inf,Inf);


int_rec_shiftPsi=0;
t0=-0.01;

while abs(int_rec_shiftPsi - int_psi_p2_s2) > 0.00001
 t0=t0+.01;
Mex_hat_s2 = @(x) 1/sqrt(2*pi)*(1-(1/S*(x-t0)).^2).*exp(-(1/S*(x-t0)).^2/2); %% mexican hat scale 2
% figure;
ezplot(Mex_hat_s2,[-10 10]);
%%%%% recived signal

rec_ss = @(x) 1/sqrt(2*pi)*(1-(1/2*(x-1)).^2).*exp(-(1/2*(x-1)).^2/2); %% mexican hat shift 1; scale 2;

hold on, ezplot(rec_ss,[-10 10]);
legend('mexican hat with scaling','recived signal');
title(sprintf(' shift factor = %f',t0));

%%% fun is f(t) * ψ ((t-?)/2)
fun = @(x)  (1/sqrt(2*pi)*(1-(1/S*(x-t0)).^2).*exp(-(1/S*(x-t0)).^2/2)).*(1/sqrt(2*pi)*(1-(1/2*(x-
1)).^2).*exp(-(1/2*(x-1)).^2/2));

int_rec_shiftPsi = integral(fun,-Inf,Inf);

end
t0
```

در کد متلب ما از آزمون تفاوت انرژی استفاده کردیم که در فوق توضیح داده شده  که کد آن را در ادامه مشاهده می کنید در اینحا از یک حلقه  while استفاده شده است که هنگامی اختلاف انرژی از مقدار ۰,۰۰۰۰۱  کمتر شود از حلقه خارج میشود که مشاهده میشود هنگامی که از حلقه خارج میشود τ=۱  است که می توان گفت روش بیان شده توانسته بخوبی مقدار تاخیر زمانی را تشخیص دهد.

## سوال ۴ :

در این سوال قرار است، یک سیگنال سینوسی را که با دو نویز متفاوت، یکی پریودیک و پالسی شکل و دیگری گوسی با میانگین صفر و پراش ۰,۴ جمع شده است را، نویز زدایی نماییم.

می دانیم که نویز گوسی در فضای تبدیل موجک، دارای پهنای باند بزرگی می باشد و انرژی آن در تمام ضرایب توزیع می شود و به همین علت، طبق اصل پارسوال، سطح انرژی در هر زیر باند پایین آمده و می توان با آستانه گذاری بر ضرایب موجک، مولفه های کم انرژی که عمدتا مربوط به نویز می باشند را کاهش داد. حال برای این آستانه گذاری باید یک λ که همان آستانه ی آستانه گذاری می باشد را تعیین نمود. ما در این تمرین مراحل تجزیه را برای چهار سطح انجام داده ایم. می توان مراحل تجزیه را برای سطوح بالاتر نیز انجام داد اما پس از تجزیه به مراحل بالاتر، تفاوت چندانی مشاهده ننمودیم لذا به همین چهار سطح بسنده نمودیم. اما می دانیم که ضرایب تجزیه مثلا در سطوح سوم و چهارم نسبت به سطوح اول و دوم دارای فرکانس های پایین تری بوده و لذا دارای رنج پایین تری نیز می باشند و به همین علت چون ضرایب تجزیه دارای رنج های متفاوتی می باشند، لذا بهتر است که برای هر سطح تجزیه، از یک λ ی مخصوص به همان سطح استفاده نمود و اگر از λ ی global استفاده نماییم نتیجه ی خوبی مشاهده نمی کنیم. پس در این جا مناسب است که از روش SURE که برای هر سطح از یک λ ی خاص استفاده می نماید، استفاده کنیم.

برای مشاهده ی نتایج، از کد زیر اتفاده نموده ایم:

```
clc
clear;
close all;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%      problem 4

wname='db20';

step=2/500;
t=0:step:2-step;

%%%% generate signal
f = cos(2*pi*10*t);

%%%%  generate rectangular noise
t0=1/20;
s=1;
p=zeros(1,500);
for i=0:step:2-step
    if mod(floor(s/20),2)==1
        p(s)=.2;
    else
        p(s)=-.2;
    end
    s=s+1;
end


Lambda_d1=zeros(1,10); Lambda_d2=zeros(1,10); Lambda_d3=zeros(1,10);
Lambda_d4=zeros(1,10); Lambda_d5=zeros(1,10); Lambda_d6=zeros(1,10);
Lambda_c_d=zeros(1,10);

%%% we use this FOR , for consider variation of Gussian Noise
```

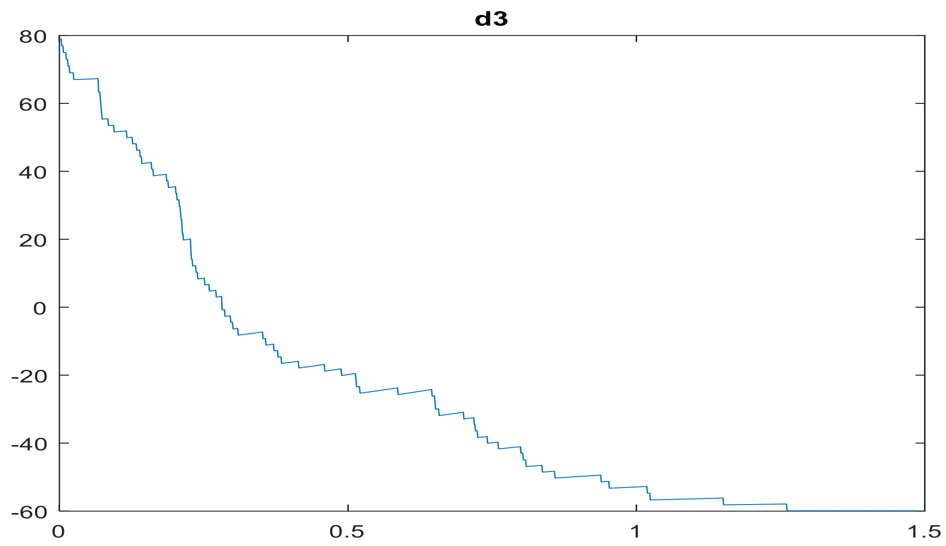
```
for i=1:10

    mu = 0; std=.4;
    v=random('norm',mu,std,1,500); %% generate Gussian noise

    x = f + p + v;     %%%  signal + noise

    %%%%%%%%%%% Decomposition signal
    lev=4;
    [c,l]=wavedec(x, lev ,wname);

    [d1 ,d2, d3, d4]=detcoef(c,l,[1 2 3 4 ]);
    [c_d]=appcoef(c,l,wname,lev);



    %%%% calculate Lambda for each level
    Lambda_d1(i) = calc_Lambda_Optimum(d1,'d1');
    Lambda_d2(i) = calc_Lambda_Optimum(d2,'d2');
    Lambda_d3(i) = calc_Lambda_Optimum(d3,'d3');
    Lambda_d4(i) = calc_Lambda_Optimum(d4,'d4');
    Lambda_c_d(i) = calc_Lambda_Optimum(c_d,'c_d');
end

th_type='s';
Lambda_d1=mean(Lambda_d1);                 %% average of Lambda
d1_N = wthresh(d1,th_type,Lambda_d1);      %%  Thresholding

Lambda_d2=mean(Lambda_d2);
d2_N = wthresh(d2,th_type,Lambda_d2);      %%  Thresholding

Lambda_d3=mean(Lambda_d3);
d3_N = wthresh(d3,th_type,Lambda_d3);      %%  Thresholding

Lambda_d4=mean(Lambda_d4);
d4_N = wthresh(d4,th_type,Lambda_d4);      %%  Thresholding


Lambda_c_d =mean(Lambda_c_d);
c_d_N = wthresh(c_d,th_type,Lambda_c_d);      %%  Thresholding


%%%%% Reconstruction Signal after thresholding components
% % %%% 4 level
C=[c_d_N d4_N d3_N d2_N d1_N];
L=[length(c_d_N) length(d4_N) length(d3_N) length(d2_N) length(d1_N)
length(x)];
x_New=waverec(C,L,wname);



%%% plottong result of recondtruction
figure;
plot(t,x), title('signal + noise');
figure;
plot(t,x_New), title(sprintf('reconstructed signal , wavelet type = %s *
Number of Level = %d',wname,lev));
```

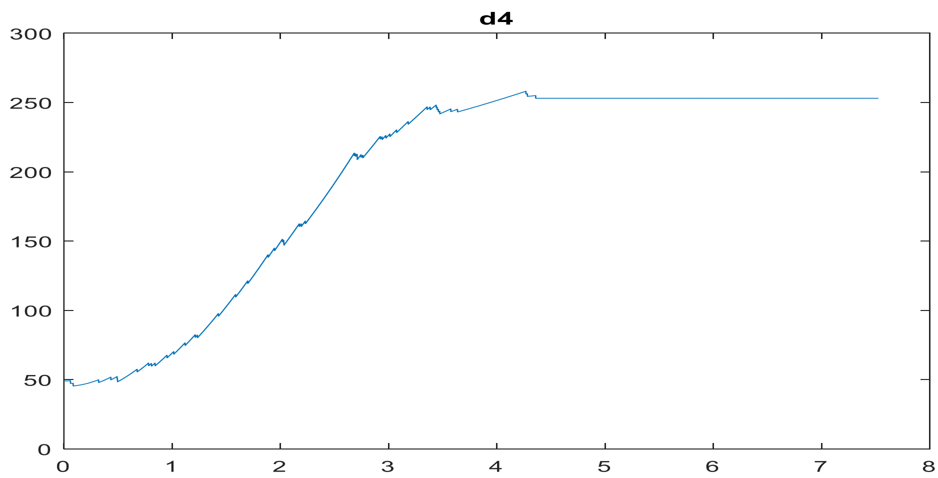
توجه شود که از آنجا که نویز گوسی وجود دارد و این نویز ماهیتی تصادفی دارد لذا برای اینکه اثر خطای نویز را کاهش دهیم و یک $\lambda$ ی بهینه را حساب نماییم، بهتر است برای نمونه با ۱۰ بار تکرار این نویز گوسی را تولید نماییم و $\lambda$ ی بهینه ی متناظر با هربار از این تکرار ها را به دست آورده که چون نویز ماهیت تصادفی دارد این $\lambda$ ها در هر تکرار تغییر میکنند و بنابراین در انتها، $\lambda$ ی بهینه را می توان به عنوان میانگین این $\lambda$ ها در نظر گرفت.

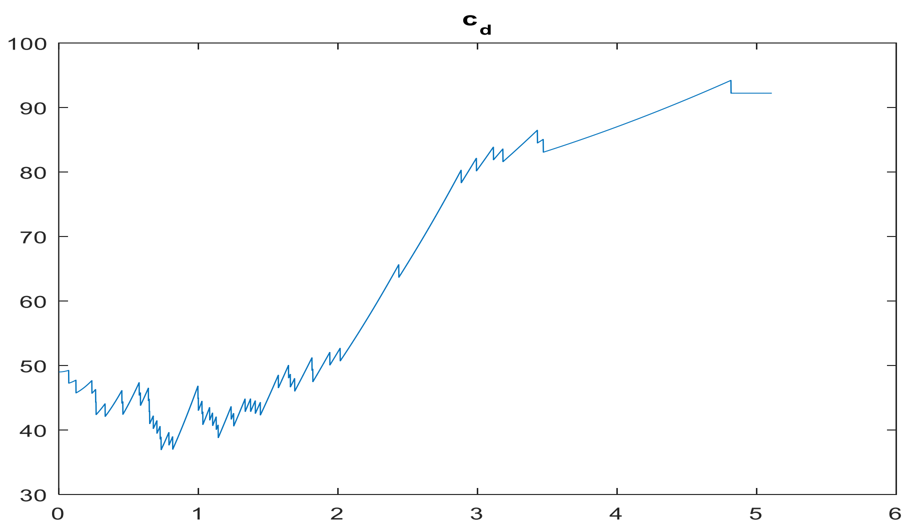
همچنین برای محاسبه $\lambda$، از تابع `calc_Lumbda_Optimum` که در برنامه نیز مشاهده می نمایید استفاده کرده ایم که بدین صورت عمل می نماید:

```
function Lambda_Optimum = calc_Lambda_Optimum(d,level)
L_d = length(d);  %% length d
std_d=std(d);


index=1;
lambda_d=0.001; lambda_u=3*std_d; step=.001;

SURE=zeros(1, length((lambda_u-lambda_d)/step));
for lambda=lambda_d:step:lambda_u

    temp1=0; temp2=0;

    for i=1:L_d

       if abs(d(i))<=  lambda
           temp1=temp1+1;
       end
       temp2=temp2+(min(abs(d(i)),lambda))^2;
    end

    SURE(index) = L_d - 2*temp1 + temp2;

    index=index+1;
end

t_lambda = lambda_d: step :lambda_u;
%
figure;
plot(t_lambda,SURE); title(level);

min_SURE=min(SURE);
aa=find(SURE==min_SURE);
Lambda_Optimum= t_lambda(aa(1));  %% the first element of min lambda

end
```

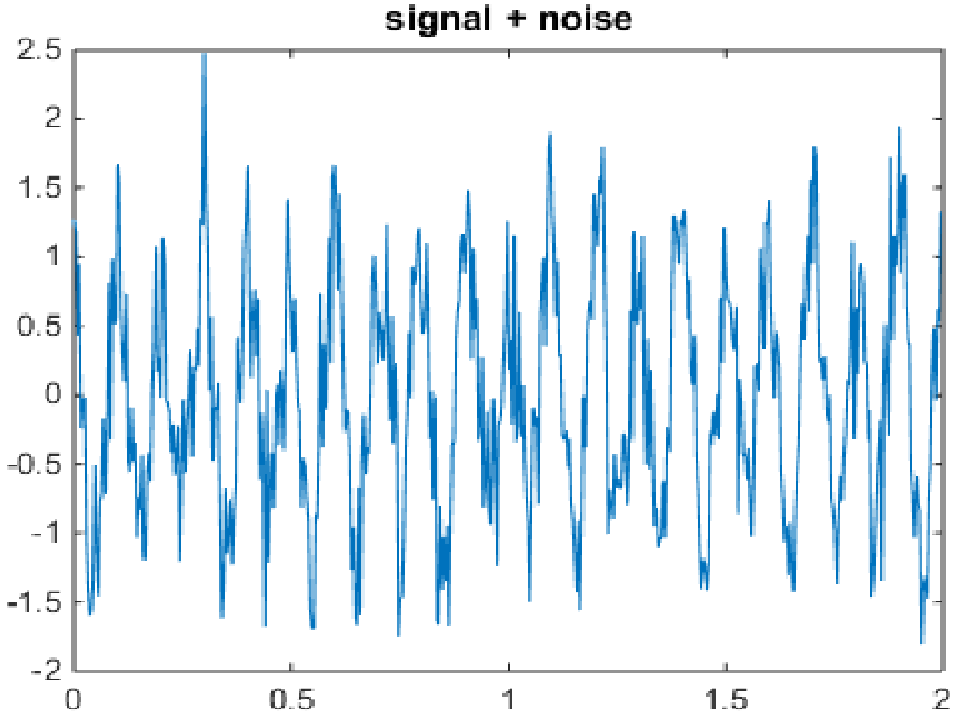
توجه شود که ما رنج $\lambda$ را از ۰,۰۰۱ تا سه برابر $\sigma$ در نظر گرفته ایم که همین رنج کفایت می نماید و در ادامه مقدار تابع SURE را حساب نموده ایم که $\lambda$ ی بهینه در واقع مینیمم این تابع می باشد. در زیر برای نمونه نمودار تابع SURE را بر حسب $\lambda$ مشاهده میکنیم(برای برخی از سطوح تجزیه - موجک db10):


d3
80
60
40
20
0
-20
-40
-60
0
0.5
1
1.5

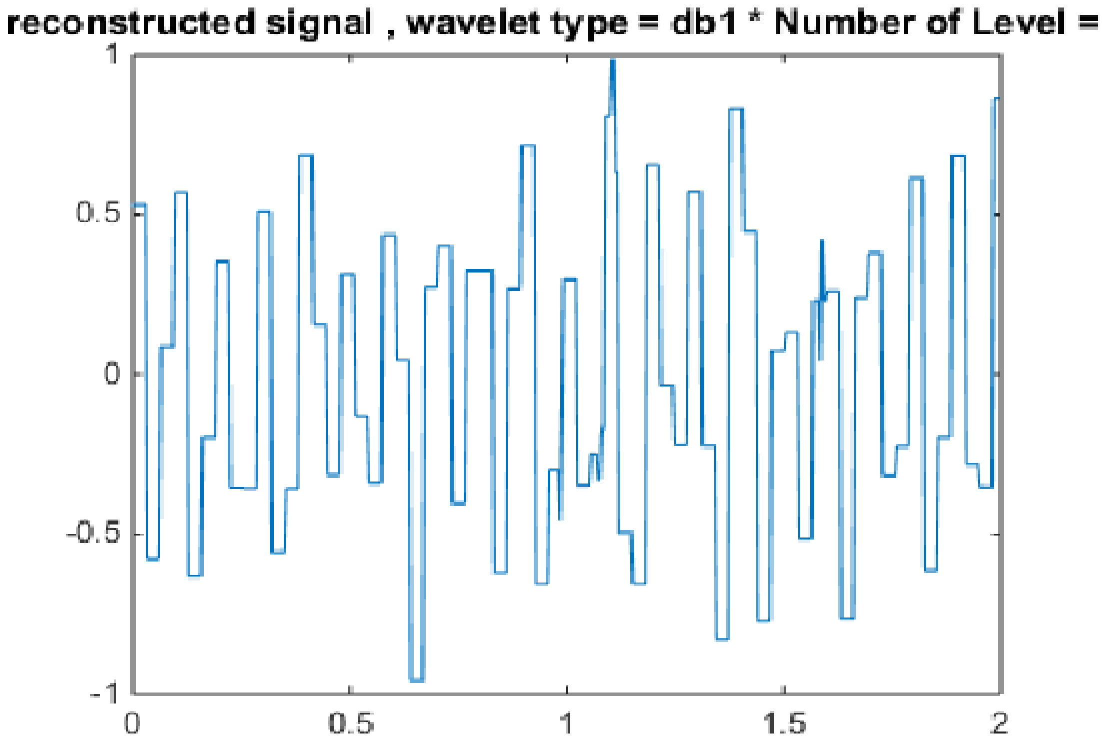

d4
300
250
200
150
100
50
0
0
1
2
3
4
5
6
7
8


$c_d$
100
90
80
70
60
50
40
30
0
1
2
3
4
5
6

ابتدا تصویر سیگنال که با نویز جمع شده است را پس از ران کردن برنامه مشاهده می نماییم که به صورت زیر می باشد:

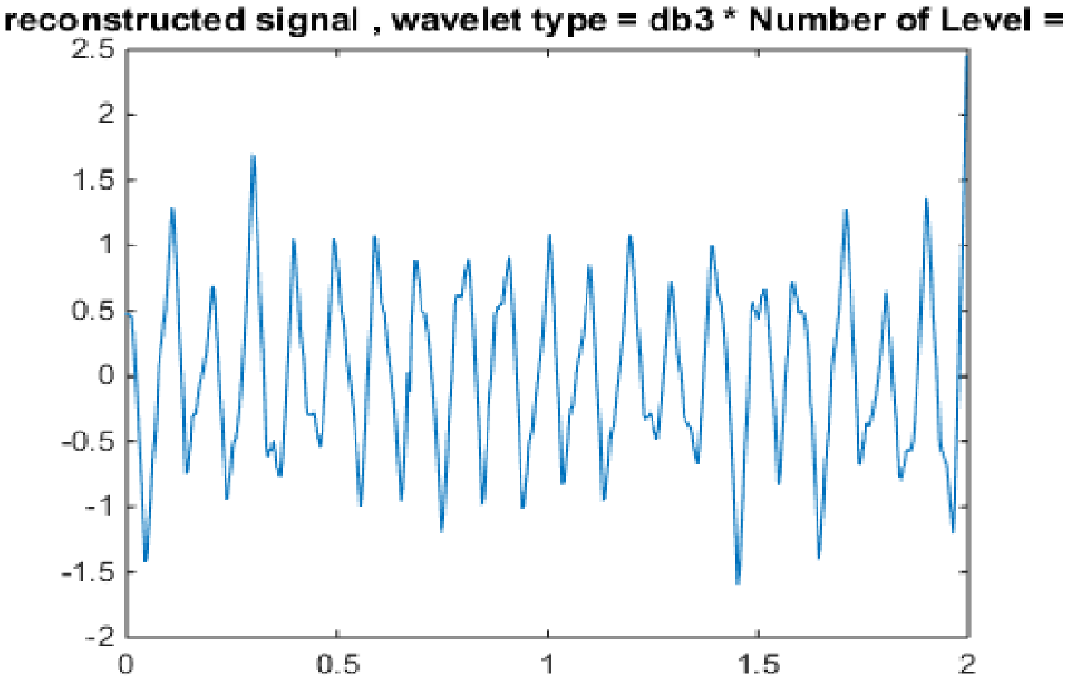



اگر از آستانه گذاری نرم و موجک db1 استفاده نماییم، پس از نویز زدایی نتیجه زیر حاصل می شود:

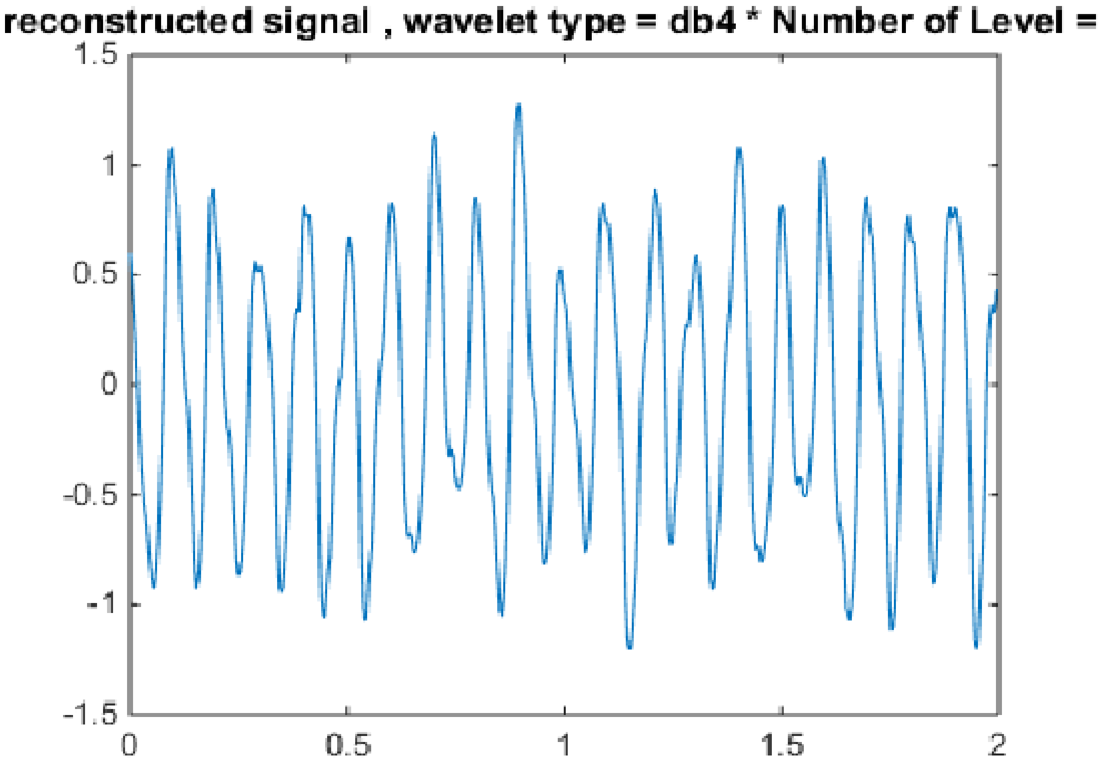



مشاهده نمودیم که واقعا نتیجه ی بدی حاصل شد.

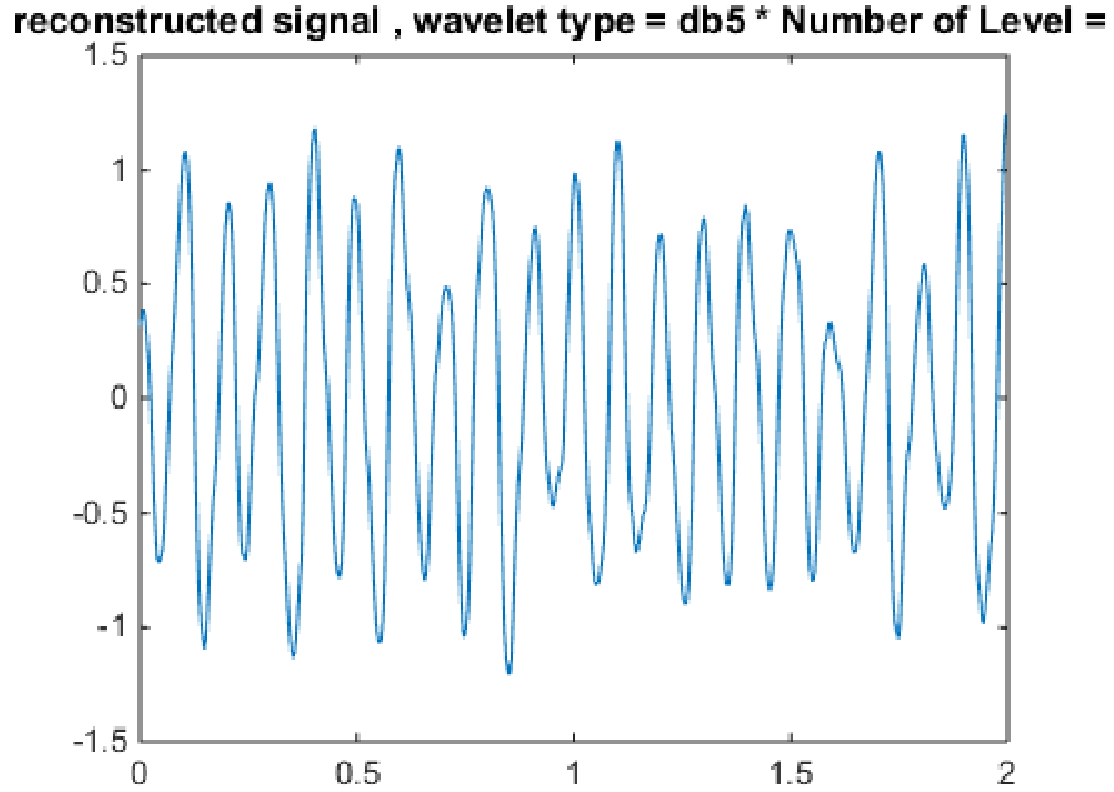
برای db2  نتیجه زیر مشاهده می شود:



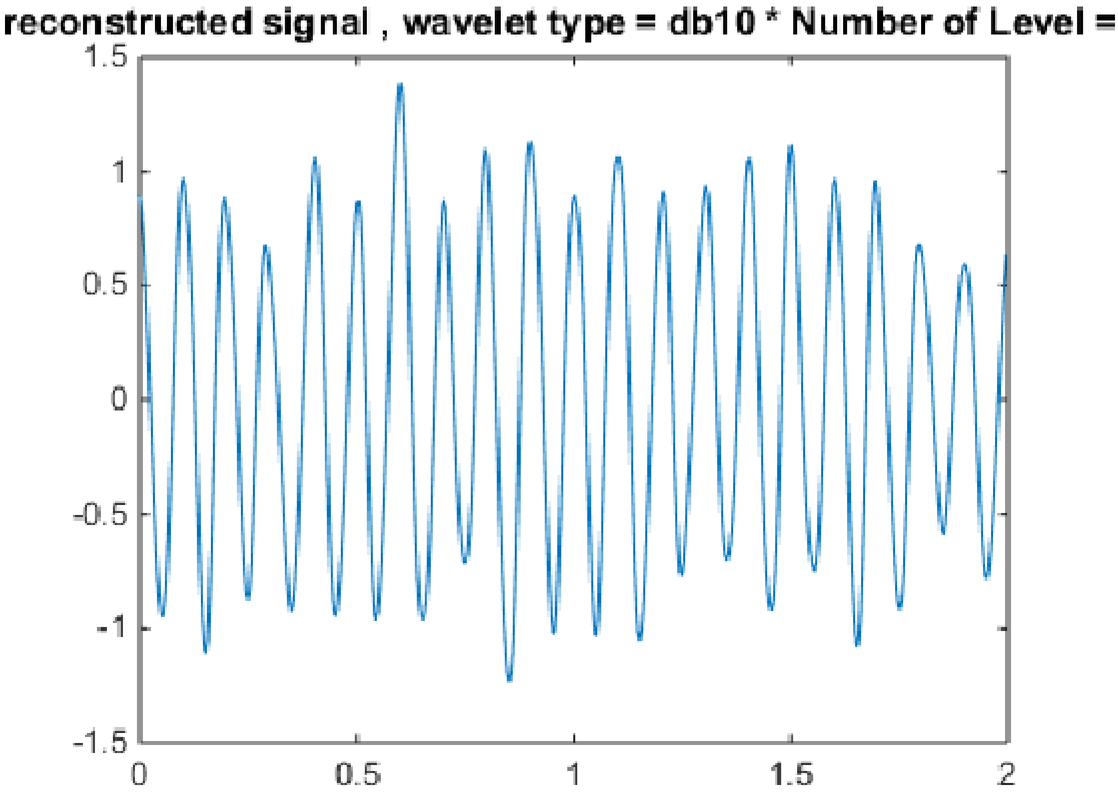

برای db4 نتیجه زیر را مشاهده می نماییم:



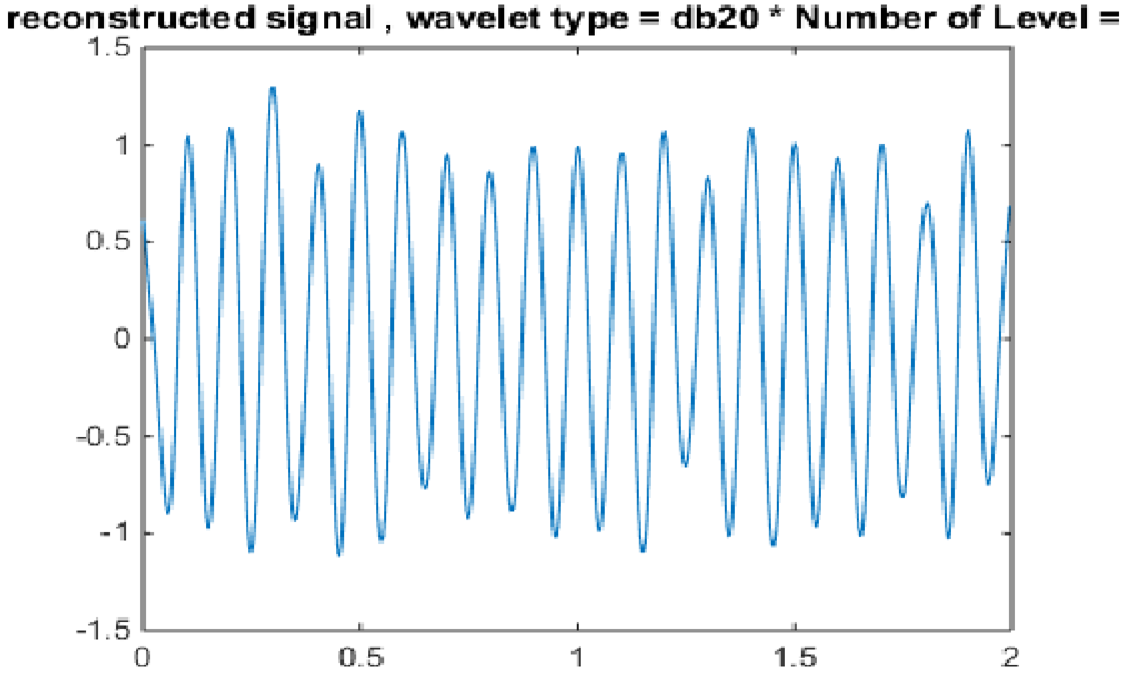
برای db5  نتیجه زیر را داریم:



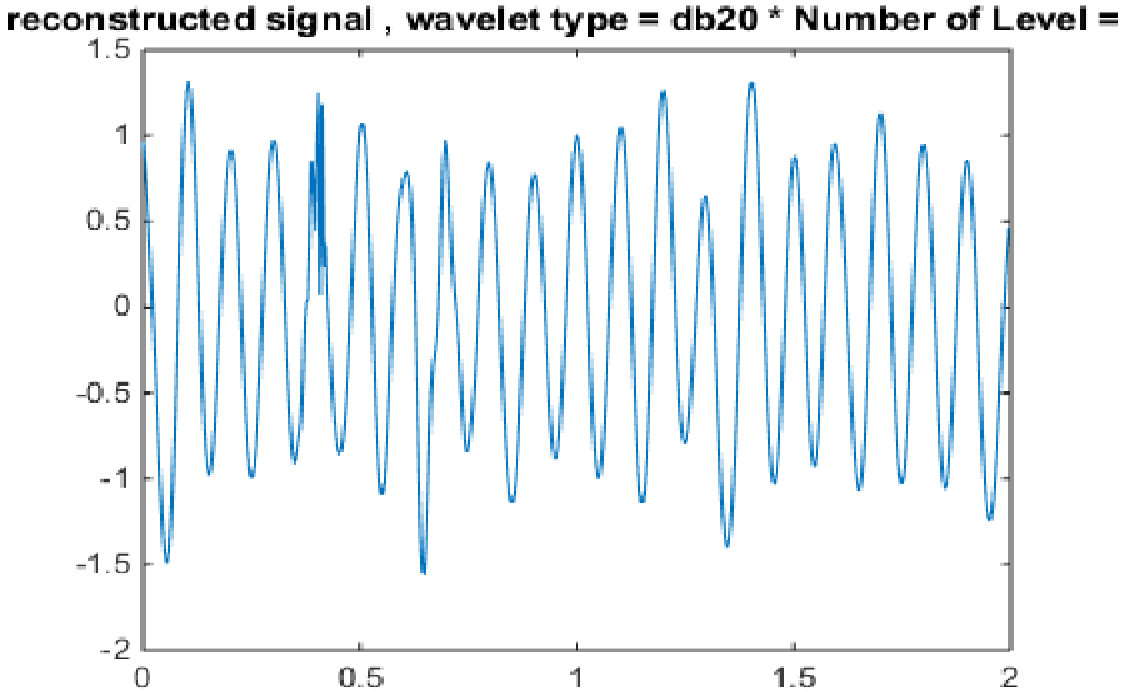

برای db10 داریم:



برای db20 نیز داریم:




مشاهده می شود هرچه مرتبه ی موجک، بالاتر می رود،دقت تفکیک فرکانسی بالاتر رفته و بهتر می تواند نویز را که عمدتا در فرکانس های بالا می باشد را حذف نماید و لذا نتیجه به سیگنال اصلی شبیه تر می شود. می توان گفت db20 نتیجه ی قابل قبولی را به ما ارایه می دهد. با افزایش مرتبه ی موجک، نتایج خیلی فرقی نکرد و لذا به db20 بسنده می نماییم.حال اگر همین مراحل نویز زدایی را توسط موجک db20 و همراه با آستانه گذاری سخت انجام دهیم نتیجه زیر را مشاهده می نماییم:



حتی با چند بار تکرار اجرای برنامه نتوانستیم در حالت آستانه گذاری سخت، نتیجه بهتری نسبت به آستانه گذاری نرم مشاهده نماییم.

بنابراین به عنوان نتیجه نهایی، موجک db20 همراه با آستانه گذاری نرم، نتیجه ی نسبتا قابل قبولی را ارایه می دهد. از آنجا که نویز در تمام سطح سیگنال پخش شده است لذا آستانه گذاری نرم عملکرد بهتری را ارایه داد. همچنین همانطور که در توضیحات ابتدای این سوال نیز ذکر نمودیم، از روش SURE برای نویز زدایی استفاده کردیم که برای هر سطح تجزیه، یک $\lambda$ مخصوص به همان سطح را به ما می دهد. علت اینکه از روش global نیز برای نویززدایی استفاده نشد همانطور که در ابتدای این سوال نیز مطرح نمودیم، این بود که ضرایب تجزیه مثلا در سطوح سوم و چهارم نسبت به سطوح اول و دوم دارای فرکانس های پایین تری بوده و لذا دارای رنج پایین تری نیز می باشند و به همین علت چون ضرایب تجزیه دارای رنج های متفاوتی می باشند، لذا بهتر است که برای هر سطح تجزیه، از یک $\lambda$ ی مخصوص به همان سطح استفاده نمود.

در ادامه نتیجه نهایی را در کنار سیگنال اصلی نمایش می دهیم که می توان گفت نسبتا قابل قبول می باشد: