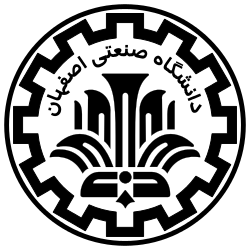
بنام خدا



دانشگاه صنعتی اصفهان  
دانشکده برق و کامپیوتر

**تمرین سری هفتم – مباحث ویژه در پردازش سیگنال های دیچیتال**

استاد: دکتر سعید صدری

پروانه رشوند 9410124

رضا سعادتی فرد 9411394

آرزو فرزانفر 9414724

تاریخ تحویل 9/03/95

**سوال 1 :**

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');

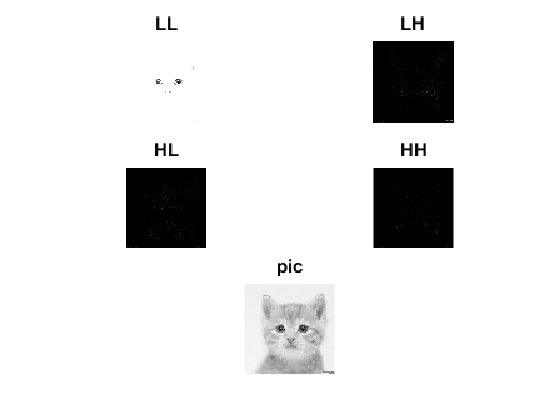
subplot(324)

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:

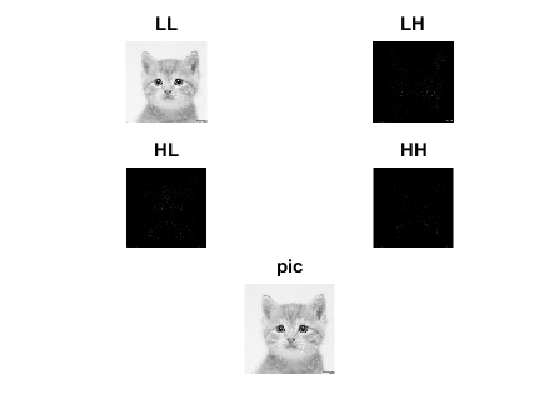


برای اینکه، نتیجه ی برنامه، واقعی تر شود، باید CA( در کد بالا) نیز بین 0 و 1 قرار گیرد چرا که تصویر اصلی را در ابتدای برنامه بین صفر و یک قرار داده ایم. برای این کار از همان متد اولیه در برنامه استفاده می نماییم یعنی کد زیر را به برنامه

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 LH، HH، HL 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 :

یک آستانه تعیین کنیم همچنین ضریب k را باید طوری تعیین کنیم که در حین آستانه گداری، 80 درصد ضرایب صفر شوند. به عبارت بهتر باید ضریب تراکم طبق تعریف زیر ، 0.8 شود:

ضریب تراکم=

%%%% calculate mean and std and determin threshold

k=.6;

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 0.8 خواسته شده در صورت سوال را پیدا میکند. در حالت k=0.6 و با اجرای دستور بالا، 0.81 خواهد شد. برای حالت k=0.7 ، این مقدار برابر با 0.83 و برای K=0.8 این مقدار برابر 0.85 خواهد بود. مشاهده می شود که با افزایش k ضریب تراکم افزایش می یابد. بنابراین با در نظر گرفتن مقدار 0.6 برای k و به دست آمدن مقدار 0.81 برای ضریب تراکم، آستانه گذاری نرم و سخت را انجام داده البته انتظار می رود که تغییر چندانی مشاهده نشود چرا که همانظور که در تصاویر بالا مشاهده نمودیم، بیس تصاویر سیاه است یعنی اکثر ضرایب صفر می باشند و لذا با آستانه گذاری و صفر نمودن 80 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 را حساب نماییم). سپس، بعد از محاسبه مقدار نرمالیزه ی ضرایب تصویر اصلی و ضرایب آستانه گداری شده، آنها را از هم کم نموده و مجددا پس از محاسبه ی نرم آن مقدار خطا را به صورت محاسبه می نماییم که a ماتریس ضرایب تصویر اصلی و مربوط به ضرایب آستانه گداری 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

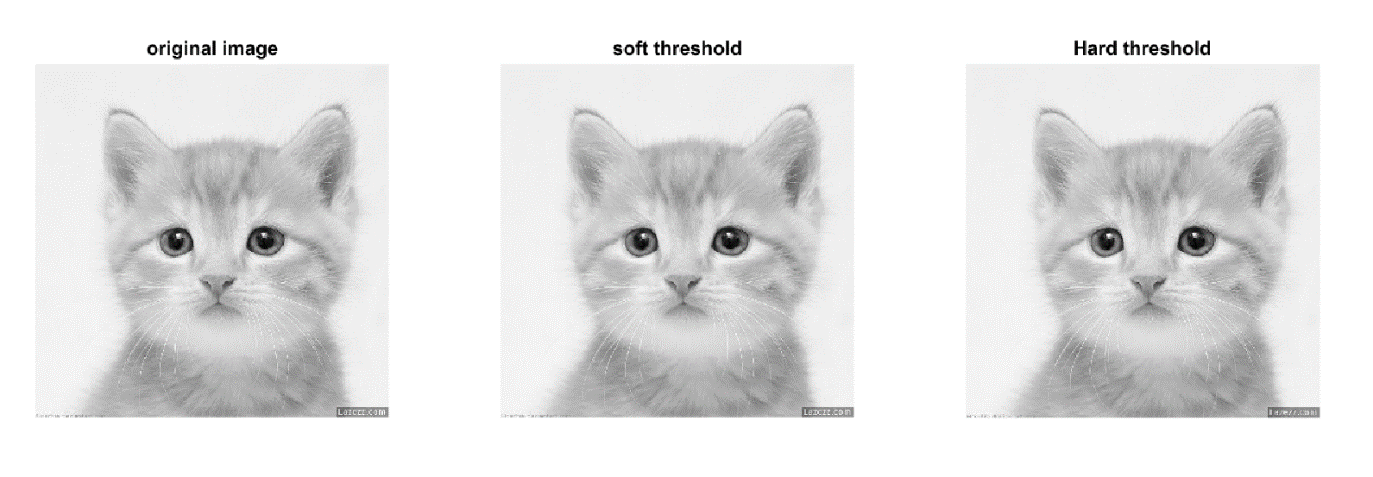
پس از ران کردن برنامه، مقدار خطا برای آستانه گذاری نرم، 0.0196 و برای آستانه گداری سخت، 0.01084 می باشد. مشاهده می شود که در این مورد، آستانه گذاری سخت، خطای کم تری دارد و در نتیجه بهتر عمل می کند. زیرا در این سوال، تصویر ما فاقد نویز است و می دانیم که آستانه گذاری سخت در واقع سیگنال اصلی را به ما می دهد در صورتی که آستانه گداری 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، 0.1897و مقدار خطای آستانه گذاری سخت، 0.01079 می باشد.

و تصاویر بدین شکل خواهد بود:



برای db6 :

مقدار خطای آستانه گذاری نرم، 0.1896و مقدار خطای آستانه گذاری سخت، 0.01067 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 2:**

**الف)**

1>>0

*= 0.5 =*

*f(t)=1- (\*)*

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

*f()=1*

*پس:*

*< k*

فقط به ازای 1>>0نامساوی 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 0.2 ، 0.4 و 0.8 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() = log(A) + (α+) 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 *α+* در نتیجه

α = m -

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 1 تا 64 یک واحد یک واحد افزایش دهیم و به ازای این 64 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=1) تقریبا از 3- تا 3+ است حدودا 64 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 64=s) و همچنین اینکه میزان step زمانی ما برابر مقدار بسیار کوچک 0.0005 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)*

(u,s) = f(u) \* WT(u,s)=

حال ما در این قسمت یک تغییر کوچک اعمال کردیم که باعث شد زمان ران برنامه که حدود 30 دقیقه طول می کشید به چند ثانیه کاهش یابد ، این تغییر به این صورت در رابطه فوق اعمال شد:

(u,s) = f(u) \* ⟹ WT(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 () 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;

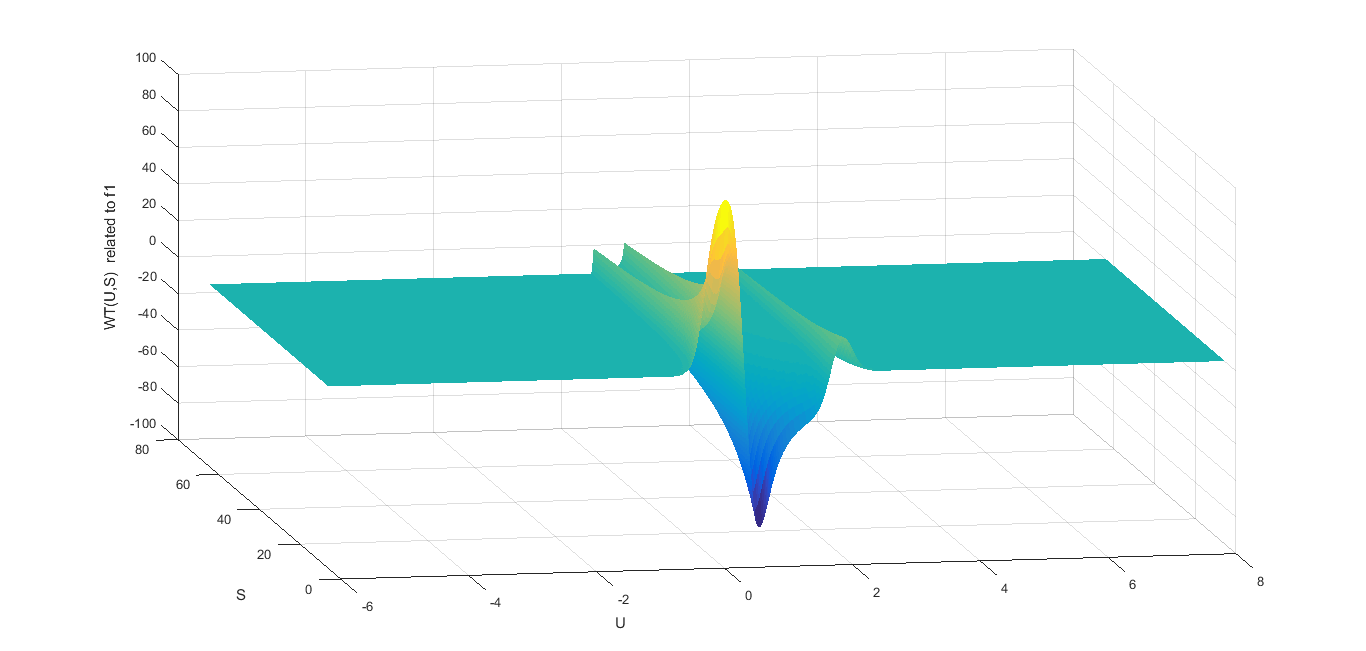
singular\_index=find(f3\_conv(s,:)==max(f3\_conv(s,:)));

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:

WTf 1, ψ(u,s)



%%% Mesh plot for wavelet coeeficients based on u , s

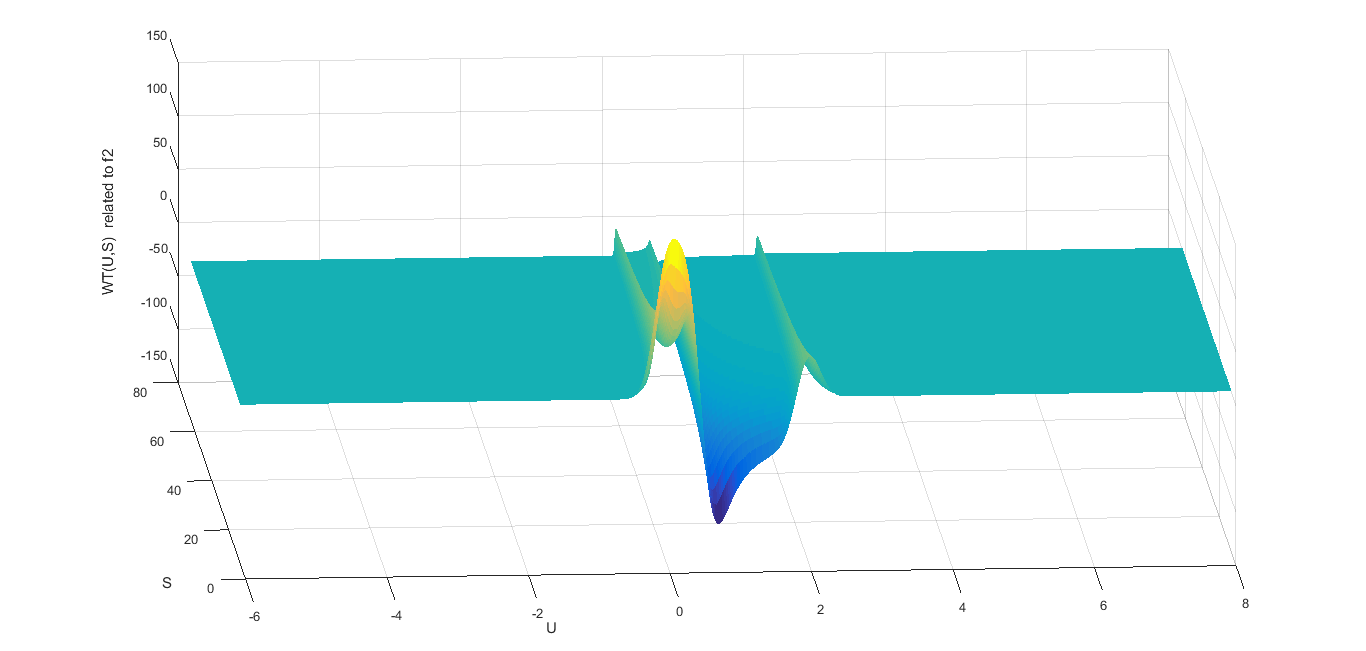
figure;

s=1:64;

[U,S]=meshgrid(t,s);

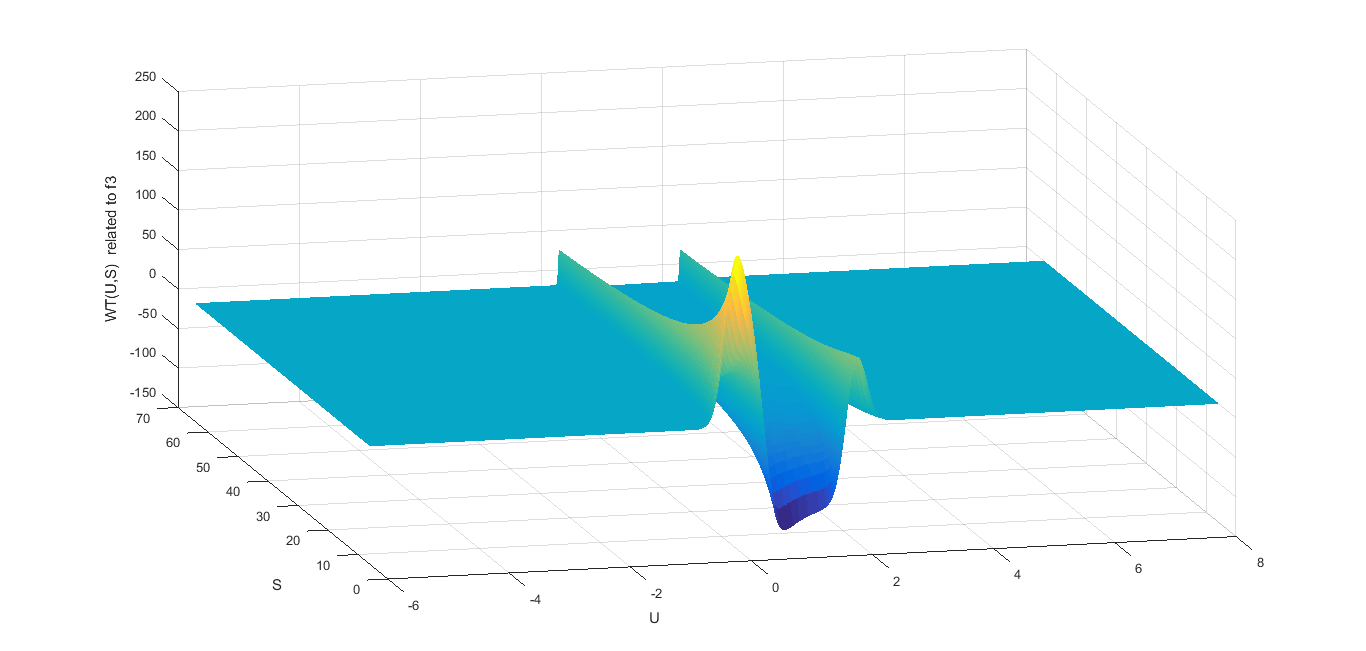
mesh(U,S,f1\_conv\_d1);

xlabel('U');ylabel('S');zlabel('WT(U,S) related to f1')



WTf 2, ψ(u,s)

WTf 3, ψ(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 قزمز 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 λ از 60 نمونه از 64 نمونه موجود استفاده کردیم و از چند 4-5 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 1 یا 2 است چون پهنای زمانی زیادی دارد(برای S=1 حدودا 6 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 λ 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 0.5 کم می کنیم عدد بدست آمده برابر است با میزان λ .

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);

|  |  |  |  |
| --- | --- | --- | --- |
| F3 | F2 | F1 |  |
| 0.8 | 0.4 | 0.2 | λ |
| 0.8050 | 0.3990 | 0.1922 | مقدار تخمین زده شده |

مشاهده می شود روش پیشنهادی با صرف زمان بسیار کمتر توانسته به خوبی میزان سینگولاریته را تخمین بزند (برای سایر λ های بین 0-1 ، از این روش برای تخمین λ استفاده کردیم که با دقت کافی توانست میزان سینگولاریته را تخمین بزند.)

**سوال 3:**

**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 sبه 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، ، از رابطه زیر به دست خواهد آمد:



\nu^'=\nu (\frac{v \pm v_o}{v \mp v_s})



**ب)**

**ارایه روشی برای تخمین مقادیر 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:

*=s ×*

*⟹ s =*

*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 =*

حالرابا توجه به مقدار sتخمین زده شده ، تخمین میزنیم.

برای

*اگر برای* این رابطه برقرار نشد رابه ترتیب افزایش میدهیم تا مقدار یک حاصل شود:

*همین روند را تا ادامه میدهیم تا نتیجه مورد نظر حاصل شود.*

*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 τ 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=1 فرض شده استو این فرض به هیچ وجه عمومیت حل مسئله را از بین نمی برد)

= = 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 1و 3 مشاهده می کنیم مشاهده می شود پهنای زمانی برای اسکیل 3 ، سه برابر ضریب اسکیل 1 است



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

= 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:

S= = = 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 2 است حال به محاسبه انرژی سیگنال می پردازیم

= =

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 را برای تاخیر های بزرگتر مساوی 0 (برای مثال از τ=0 شروع میکنیم و و با استپ 0.01 τ را افزایش میدهیم) تا وقتی که اختلاف مقدار انتگرال از از یک مقدار آستانه (فرضا ) کوچکتر شود ، آن گاه میتوان نتیجه گرفت که سیگنال دقیقا روی f(t) قرار گرفته است و در نتیجه f(t)= میشود

در زیر نمایش همزمان سیگنال دریافتی و سیگنال را به ازای τ های 0، 0.5 ، 1 و 1.5 مشاهده می شود.



τ = 0



τ = 0.5



τ = 1



τ = 1.5

همانطور که در شکل های فوق مشهود است به ازای τ=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) \* ((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 0.00001 کمتر شود از حلقه خارج می

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 0.00001 کمتر شود از حلقه خارج میشود که مشاهده میشود هنگامی که از حلقه خارج میشود τ=1 است که می توان گفت روش بیان شده توانسته بخوبی مقدار تاخیر زمانی را تشخیص دهد.

**سوال 4 :**

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 0.4 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 λ ی بهینه را حساب نماییم، بهتر است برای نمونه با 10 بار تکرار این نویز گوسی را تولید نماییم و λ ی بهینه ی متناظر با هربار از این تکرار ها را به دست آورده که چون نویز ماهیت تصادفی دارد این λ ها در هر تکرار تغییر میکنند و بنابراین در انتها، λ ی بهینه را می توان به عنوان میانگین این λ ها در نظر گرفت.

همچنین برای محاسبه λ، 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

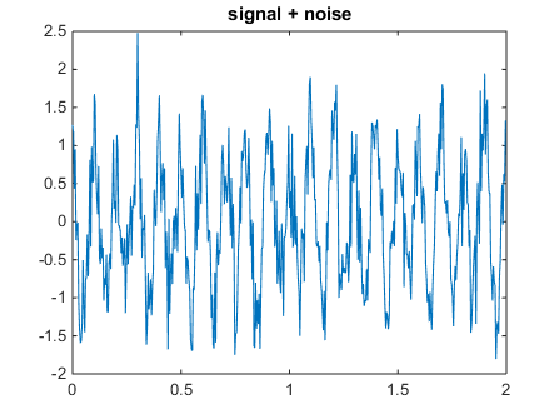
توجه شود که ما رنج λ را از 0.001 تا سه برابر σ در نظر گرفته ایم که همین رنج کفایت می نماید و در ادامه مقدار تابع SURE را حساب نموده ایم که λ ی بهینه در واقع مینیمم این تابع می باشد. در زیر برای نمونه نمودار تابع SURE را بر حسب λ مشاهده میکنیم(برای برخی از سطوح تجزیه - موجک db10):



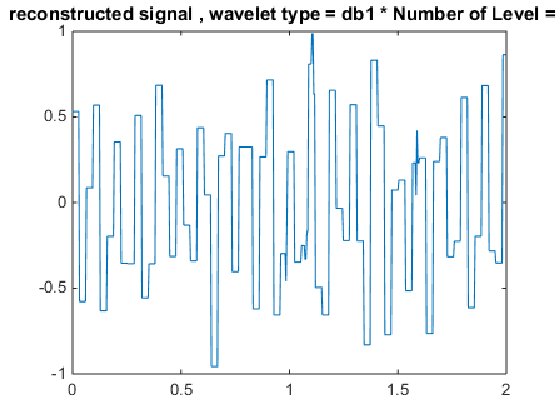




ابتدا تصویر سیگنال که با نویز جمع شده است را پس از ران کردن برنامه مشاهده می نماییم که به صورت زیر می باشد:

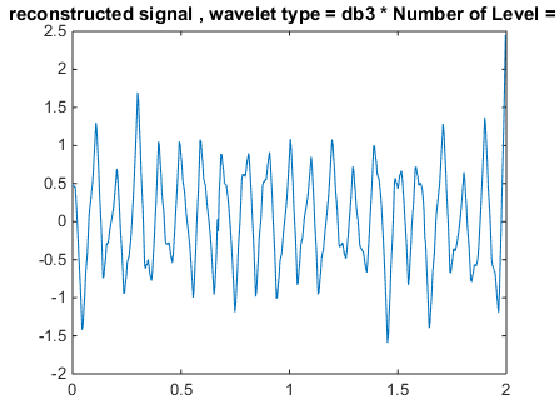


اگر از آستانه گذاری نرم و موجک db1 استفاده نماییم، پس از نویز زدایی نتیجه زیر حاصل می شود:

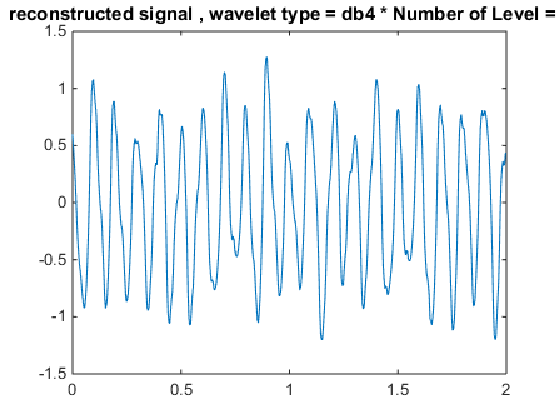


مشاهده نمودیم که واقعا نتیجه ی بدی حاصل شد.

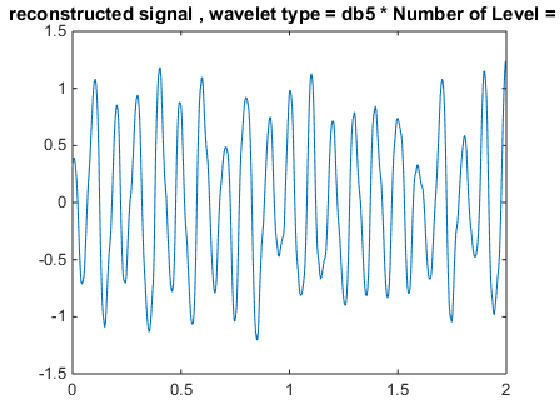
برای db2 نتیجه زیر مشاهده می شود:



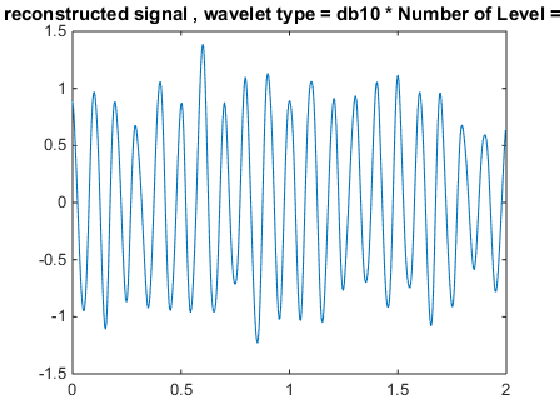
برای db4 نتیجه زیر را مشاهده می نماییم:



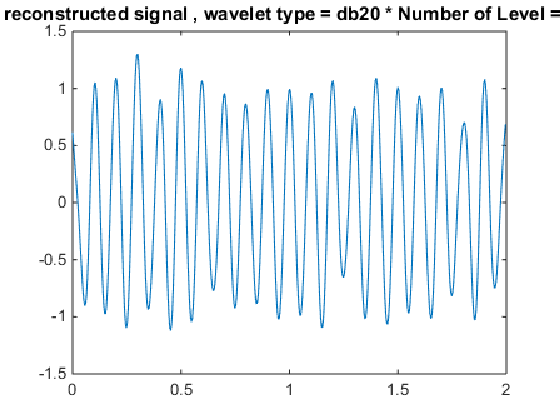
برای db5 نتیجه زیر را داریم:



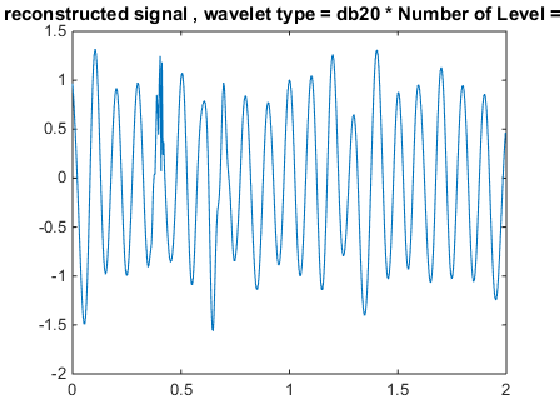
برای db10 داریم:



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 ایتدای این سوال نیز ذکر نمودیم، از روش SURE برای نویز زدایی استفاده کردیم که برای هر سطح تجزیه، یک λ 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 λ 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:

