Project Phase02

Ali Ghavampour 97102293

Farhad Fallah 97102214

Sharif University of Technology

Signal and Systems

Dr. Hamid K. Aghajan

طراحی فیلتر

سوال 1) یافتن خروجی برای دو فیلتر

$$"\boldsymbol{\phi(\omega) = -\frac{\pi}{3} sgn(\omega)}"$$

$$\boldsymbol{x(t) = cos(\omega_0 t) + cos(5\omega_0 t) \Rightarrow y(t) = cos\left(\omega_0\left(t \pm \frac{\pi}{3\omega_0}\right)\right) + cos\left(5\omega_0\left(t \pm \frac{\pi}{15\omega_0}\right)\right)}$$

$$"\phi(\omega) = -\frac{\pi}{3}\omega"$$

$$y(t) = cos\left(\omega_0 t - \frac{\pi}{3}\omega_0\right) + cos\left(5\omega_0 t - \frac{\pi}{3}(5\omega_0)\right) = cos\left(\omega_0\left(t - \frac{\pi}{3}\right)\right) + cos\left(5\omega_0\left(t - \frac{\pi}{3}\right)\right)$$

همانطور که از خروجی های بالا می‌بینیم، اگر فاز $H(w)$ خطی باشد، کلی هارمونیک های سیگنال با هم شیفت پیدا می‌کنند. در واقع مانند این است که همان اصل سیگنال با کمی تاخیر به دست ما برسد. اما اگر فاز غیر خطی باشد، ممکن است هر یک از هارمونیک ها مقدار شیفت متفاوتی پیدا کنند که باعث می‌شود در هر لحظه مقدار سیگنال عوض شود. از این تغییر فاز نسبی هارمونیک ها می‌توان به عنوان اعوجاج سیگنال یاد کرد.

سوال 2) بررسی **group delay** برای دو فیلتر سوال قبل

برای فیلتر دوم:

$$gd(\omega_0) = \frac{\pi}{3}$$

همانطور که در بخش قبل نیز دیدیم، سیگنال اولیه با pi/3 تاخیر به خروجی می‌رسد.

برای فیلتر اول:

با مشتق گرفتن از فاز این فیلتر، به یک تابع ضربه در نقطه صفر می‌رسیم.

سوال 3) اثبات رابطه

$$\boldsymbol{H(\omega) = A(\omega)e^{j\phi(\omega)} \Rightarrow \frac{j\frac{dH(\omega)}{d\omega}}{H(\omega)} = \frac{j\left(\frac{d}{d\omega}A(\omega)\right)e^{j\phi(\omega)} - \left(\frac{d}{d\omega}\phi(\omega)\right)A(\omega)e^{j\phi(\omega)}}{A(\omega)e^{j\phi(\omega)}}}$$

$$\boldsymbol{\Rightarrow \frac{j\frac{dH(\omega)}{d\omega}}{H(\omega)} = -\frac{d}{d\omega}\phi(\omega) + j\frac{\frac{dA(\omega)}{d\omega}}{A(\omega)}}$$

$$\boldsymbol{\Rightarrow Re\{\frac{j\frac{dH(\omega)}{d\omega}}{H(\omega)}\} = -\frac{d}{d\omega}\phi(\omega)}$$

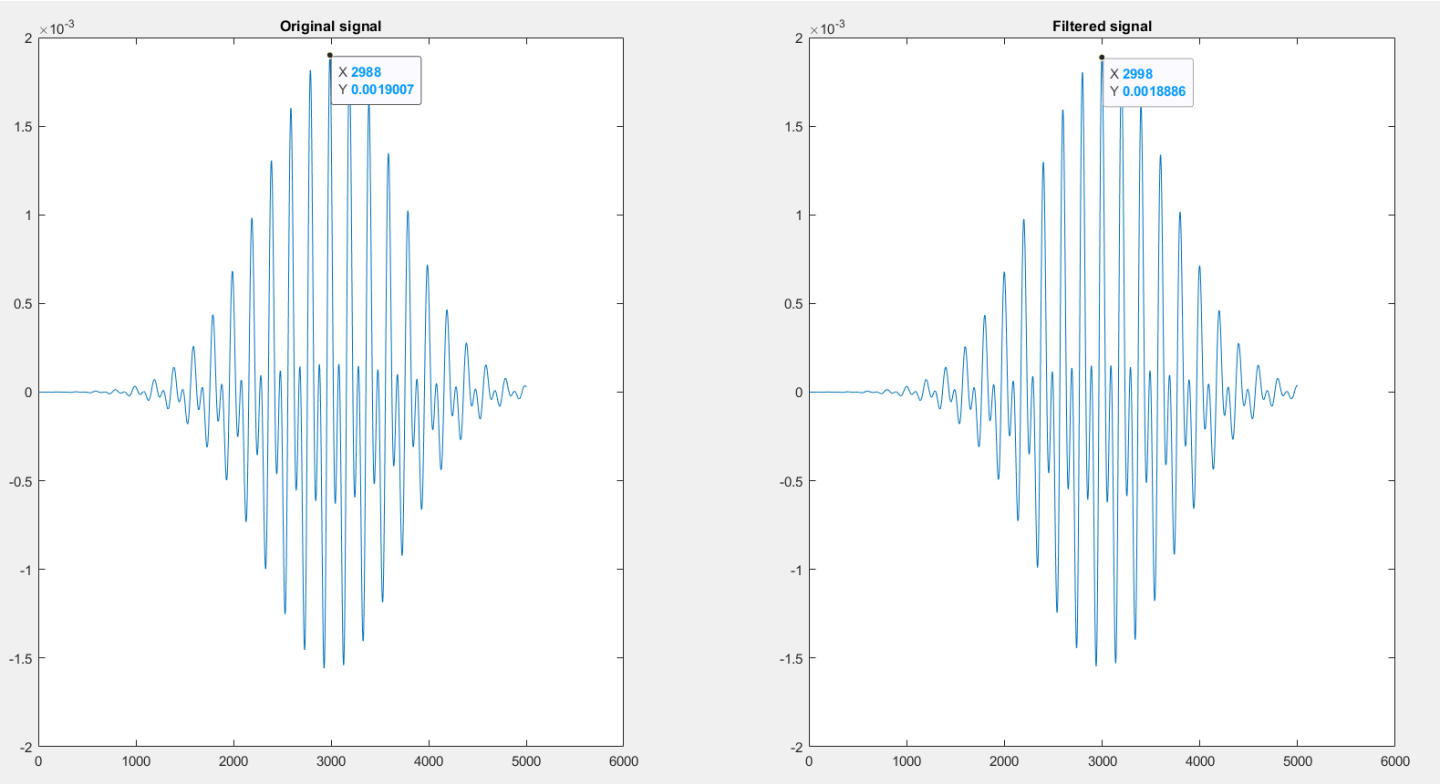
با توجه به اینکه تابع fft در واقع DFT سیگنال را محاسبه می‌کند، اگر N از طول فیلتر بزرگتر باشد، تابع fft درست عمل می‌کند و مقادیر به دست آمده برای gd دقیق خواهند بود.

در ابتدا برای بررسی عملکرد تابع groupdelay زده شده، یک سیگنال تست می‌سازیم که به صورت زیر می‌باشد:

```
7 -    x = (exp(-(t-3).^2) .* cos(2*pi*5*t) + exp(-(t-3).^2) .* cos(2*pi*10*t + 1))/1000;
```

همچنین یک فیلتر ساده طراحی می‌کنیم که فاز آن خطی باشد. خروجی gd تابع ما برای این فیلتر عدد 10 را نشان می‌دهد. همچنین خروجی تابع خود متلب نیز همین مقدار را تعیید می‌کند. در نتیجه سیگنال ما باید به میزان 10 داده تاخیر پیدا کند.

نمودار خروجی سیگنال، قبل و بعد از فیلتر:

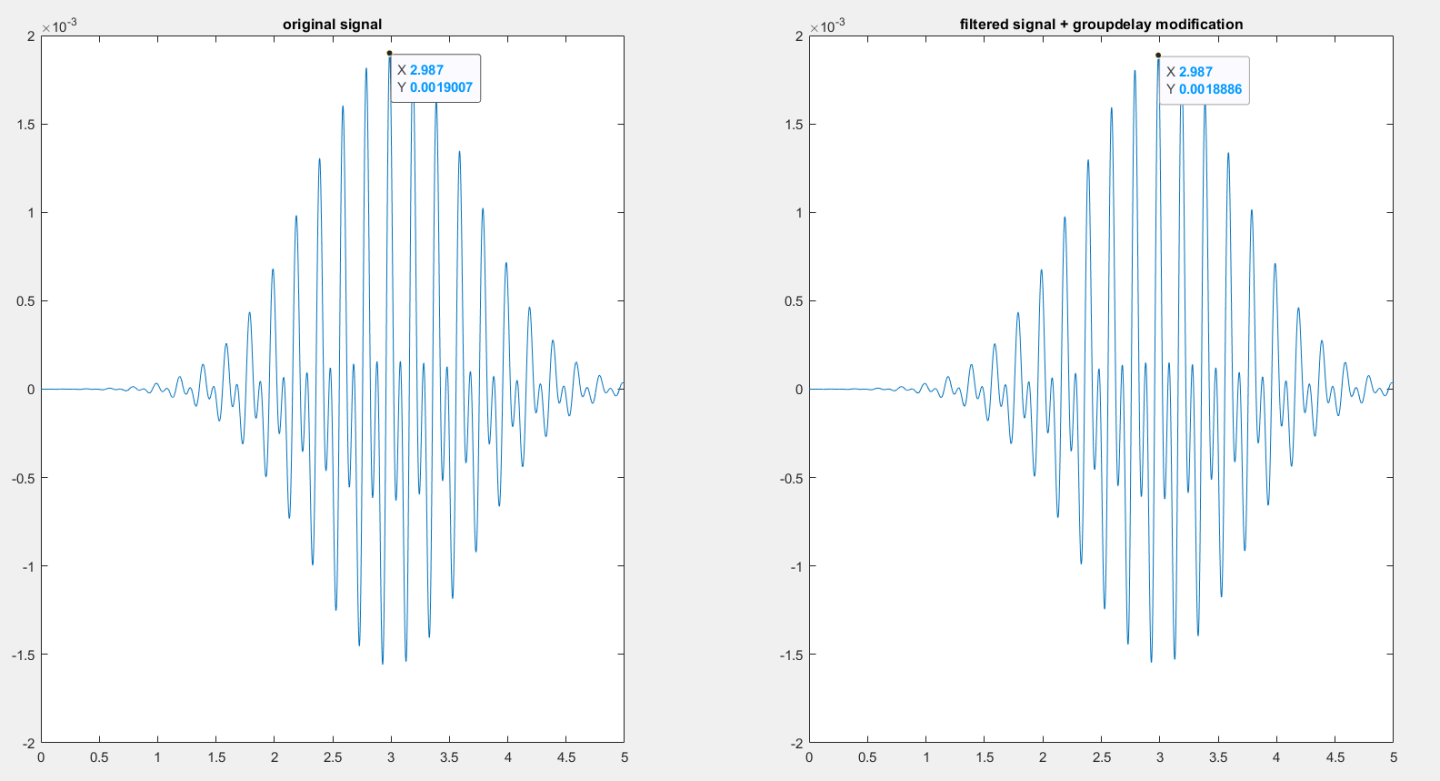


همانطور که می‌بینید، نمونه ای که در آن سیگنال به اوج خود می‌رسد را نشان داده ایم و اختلاف سمپل ها 10 می‌باشد.

برای نوشتن تابع zphasefilter باید تنها سیگنال را به اندازه gd به چپ یا راست شیفت دهیم. اگر gd مثبت بود باید سیگنال فیلتر شده را به چپ شیفت دهیم و اگر gd منفی بود باید سیگنال را به سمت راست شیفت دهیم.

در صفحه بعد، خروجی این فیلتر را برای ورودی و فیلتر بالا بررسی می‌کنیم.

سیگنال x در کنار خروجی متناظر آن برای تابع zphasefilter:



همانطور که مشاهده می‌کنید، تابع به درستی عمل کرده، و اثر gd را از بین برده است و در جفت سیگنال ها، ماکسیمم سیگنال در یک index قرار دارد.

```
>> find(y == max(y))

ans =

        2988

>> find(x == max(x))

ans =

        2988
```

فیلتری که علی و حقیقی است و فاز آن در تمام فرکانس ها صفر است به فرم زیر می‌باشد:

$$h[n] = k\delta[n]$$

حقیقی و علی بودن فیلتر که مشخص است. همچنین این فیلتر، سیگنال ورودی را بدون شیفت عینا به خروجی می‌دهد پس فاز آن نیز صفر است.

شناسایی کلمات

سوال اول)

ابتدا تمامی داده ها را خواندیم و در یک struct قرار دادیم ، برای هرکدام از این struct ها یک فیلد جدید با نام testindexvalue ایجاد کردیم که شامل 2 سطر است. سطر اول ، شماره خانه هایی از سطر دهم فیلد test که مقدار غیر صفر دارند ذخیره میکند و در سطر دوم مقدار این خانه ها ذخیره میشود.

با مشاهده کلی داده مشاهده میشود که هر زمان که هر حرف یا ستونی روشن میشود به مدت 4 خانه مقدار خود را حفظ میکند.

حال با توجه به اینکه کلمه ما 5 حرف است ، هر حرف یا ستون 15 بار روشن میشود ، و هر بار روشن شدن آن در 4 خانه ثابت است اندازه testindexvalue را برای هر یک از حالات RC و SC پیش بینی میکنیم.

RC = 12(تعداد سطر و ستون ها) × 15(تعداد تکرار) × 4(هر بار روشن شدن) × 5(تعداد حروف) = 3600

SC = 36(تعدادحروف) × 15(تعداد تکرار) × 4(هر بار روشن شدن) × 5(تعداد حروف) = 10800

با مشاهده 9 داده و با توجه به رابطه بالا جدول زیر برای نوع آزمایش به دست می آید:

Subject Data	1	2	3	4	5	6	7	8	9
RC/SC	SC	SC	RC	RC	RC	RC	RC	RC	RC

سوال دوم)

با بررسی سطر های 10و11 فیلد train داده ها و پیدا کردن خانه هایی که target شده اند و تطبیق حروف با حروف مورد انتظار و با توجه به نحوه قرار گیری حروف در تصویر برای حالت RC ستون های از چپ به راست اعداد 1 تا 6 و سطر ها از بالا به پایین اعداد 7 تا 12 شماره گذاره شده اند و در SC حروف از A تا Z با شماره های 1 تا 26 و اعداد 0 تا 9 با شماره های 27 تا 36 مشخص شده اند. این شرایط برای هر 9 سری داده بررسی شد اما برای کم تر شدن حجم کد ها صرفا یک نمونه از RC و SC قرار داده شد، که در فیلد target قابل مشاهده است.

سوال سوم)

در این بخش تابع IndexExtraction را تعریف کردیم که خروجی ان یک فیلد Time را به استراکت ما اضافه میکند که این فیلد دارای سه سطر هست.

سطر اول index هایی که در آن ها در بخش test کاراکتری دیده شده نشان می‌دهد.

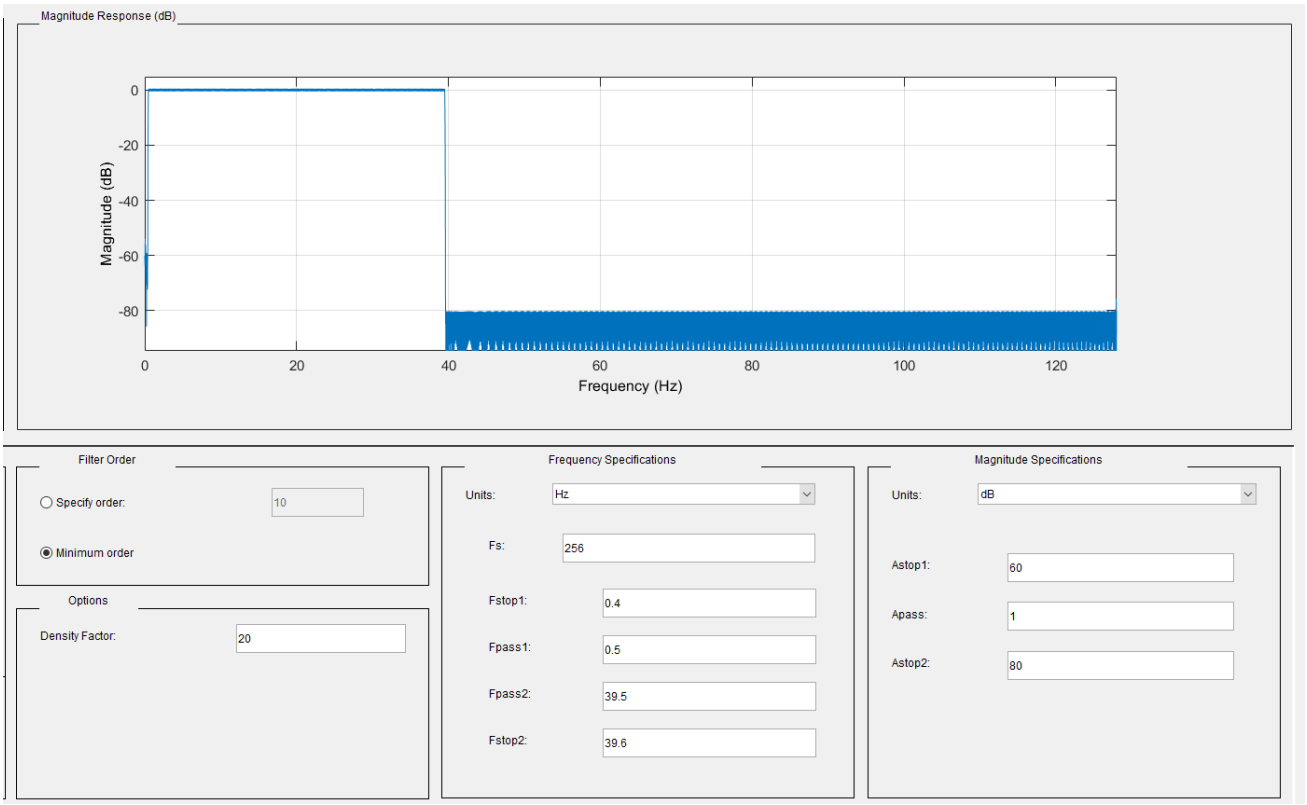
سطر دوم index هایی که در آن ها در بخش train کاراکتری دیده شده نشان می‌دهد.

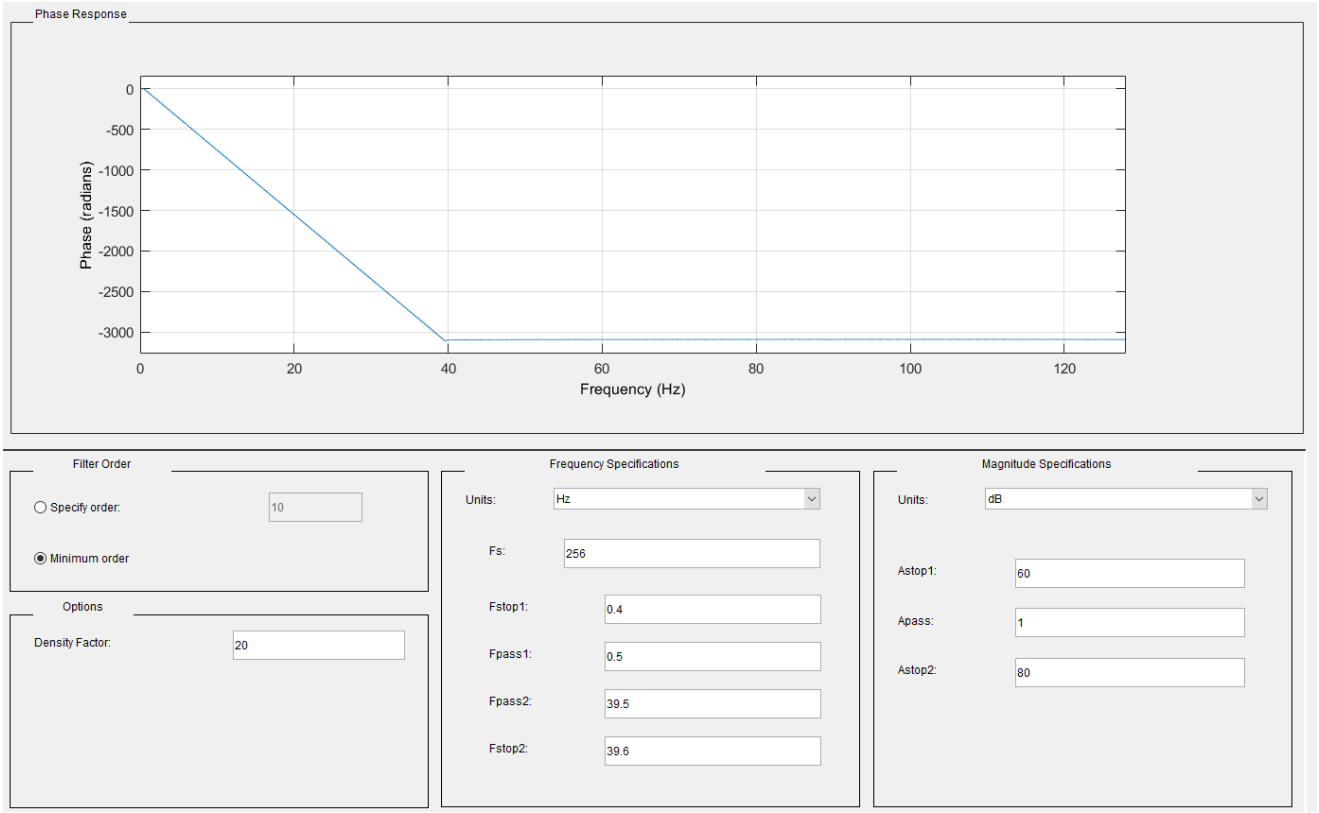
سطر سوم هم target و non-target بودن کاراکتر های دیده شده در سطر دوم در بخش train را زیر هر index نشان می‌دهد.

مرحله فیلترینگ و Epoching:

برای فیلتر کردن سیگنال ها، با مشاهده خود سیگنال ها و همچنین مطابق فاز 1، می‌بینیم که فیلتر مناسب، یک فیلتر BandPass از 0.5 تا 39.5 هرتز می‌باشد. همچنین مطابق فاز 1 باید میانگین داده ها را از خود داده کم کنیم.

فیلتر طراحی شده مشخصات زیر را دارد:





همانطور که مشاهده می‌کنید فاز فیلتر در بازه مورد نظر ما خطی است. فیلتر از نوع FIR می‌باشد و با نام BandPass ذخیره شده است. قبل از epoch کردن، ابتدا همه دیتا های train را فیلتر می‌کنیم. مقادیر فیلتر شده را در متغیر های filtered_signal می‌ریزیم. همچنین به دیتا های هر فرد، دو فیلد جدید با نام های StimuliTarget و StimuliNonTarget اضافه می‌کنیم. با استفاده از هر کدام از این Stimuli ها برای هر دیتا، دو epoching مختلف، مربوط به Target و NonTarget انجام می‌دهیم.

در نهایت 18 تا Epoch خواهیم داشت که در فولدر epochs در فولدر data ذخیره شده اند.(مدت زمان ران شدن این بخش طولانی است)

بخش Feature Extraction

ویژگی های حوزه فرکانس:

1) انرژی سیگنال

انرژی هر trial ها برای همه epoch ها محاسبه می‌کنیم.

2) میانگین فرکانسی

با استفاده از رابطه زیر، میانگین فرکانسی به دست می‌آید:

$$f_{mean} = \frac{\sum_{i=0}^{n} I_i \cdot f_i}{\sum_{i=0}^{n} I_i}$$

که n تعداد کل نقاط فرکانسی است که داریم و fi مقدار هر یک از این فرکانس ها است و Ii برابر با دامنه فرکانس (به dB) می‌باشد. این مقدار با استفاده از تابع meanfreq متلب محاسبه می‌شود. میانگین به هر حال یکی از بدیهی ترین شاخص های مقایسه ما می‌تواند باشد.

3) میانه فرکانسی

میانه هم مشخصا، میانه فرکانس ها می‌باشد که با استفاده از تابع medfreq متلب محاسبه می‌شود. میانه نیز مانند میانگین از شاخص های بدیهی می‌باشد.

4) انرژی باند بتا

این باند که آن را از 13 تا 30 هرتز می‌توان در نظر گرفت، مربوط به فعالیت های پردازشی و هوشیاری مغز می‌باشد در نتیجه می‌تواند معیار خوبی برای ما باشد. انرژی این باند را با استفاده از تابع bandpower متلب محاسبه می‌کنیم.

5) انرژی باند آلفا

معادل با ویژگی 4 برای باند آلفا هم که مربوط به استراحت مغز است، این ویژگی را نیز انتخاب می‌کنیم. همچنین مانند ویژگی قبلی، با استفاده از bandpower این انرژی را محاسبه می‌کنیم. فرکانس مربوط به این بند از 8 تا 13 هرتز می‌باشد.

6) تبدیل کسینوسی گسسته

در واقع اگر مطابق رابطه اویلر برای complex exponential ها، تبدیل فوریه را باز کنیم، به تبدیل های کسینوسی و سینوسی می‌رسیم که در واقع تشکیل دهنده تبدیل فوریه هستند. این رابطه به صورت زیر می‌باشد:

$$f(v) = \int_{-\infty}^{\infty} f(t) e^{-2\pi i v t}\, dt = f^{c}(v) + i f^{s}(v)$$

که در رابطه بالا، v فرکانس هرتز می‌باشد. قسمت تبدیل کسینوسی فرم زیر را پیدا می‌کند:

$$\hat{f}^{c}(\nu) = 2\int_{0}^{\infty} f(t)\cos(2\pi\nu t)\,dt.$$

حالا تبدیل کسینوسی گسسته ضرایب متناظر با همین تبدیل می‌باشند. در واقع اگر سیگنال را گسترش زوج دهیم و ضرایب سری فوریه را حساب کنیم، مشخصا ضرایب قسمت سینوسی صفر است و ضرایب قسمت کسینوسی باقی می‌مانند.

علت استفاده از این تبدیل این است که اولا محتوا های پایین فرکانسی را در چند جمله ابتدایی خود نمایندگی می‌کند و اگر مثلا 50 ضریب اول تبدیل را برداریم، می‌تواند برای پردازش ما کافی باشد. که به نسبت مثلا 300 دیتا پوینت، خیلی پردازش ما را سریع تر می‌کند. همچنین پیچیدگی کار با این سیگنال نسبت به خود سیگنال EEG بسیار کمتر است.

استخراج ویژگی از تبدیل wavelet گسسته

این تبدیل در واقع سیگنال اصلی را decompose می‌کند و یک نوع تبدیل زمان فرکانسی می‌باشد. نوعی از این تبدیل که ما استفاده می‌کنیم، Daubechies wavelet مرتبه 2 می‌باشد که با db2 نشان می‌دهند. در فرایند decomposition، سیگنال بار ها از تعدادی فیلتر عبور می‌کند و down sample می‌شود که به تعداد این مراحل فیلتر شدن، level می‌گویند. بلوک دیاگرام این کار به فرم زیر می‌باشد:

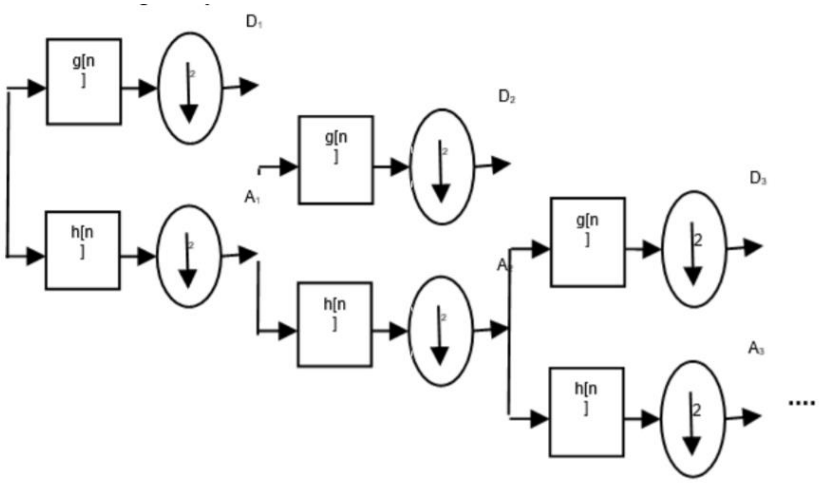
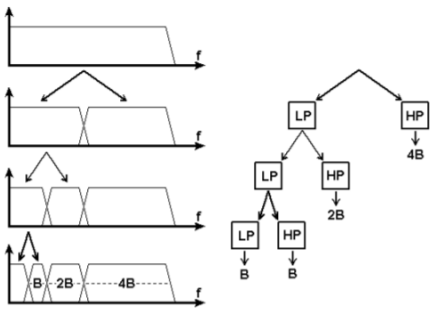


Figure 2. Implementation of decomposition of DWT.

فیلتر هایی که با g نشان داده شده اند، بالاگذر هستند و فیلتر های h پایین گذرند. تصویر زیر اتفاقی که برای فرکانس سیگنال می‌افتد را نمایش می‌دهد:



در متلب از تابع wavedec برای محاسبه تبدیل wavelet چند مرحله ای استفاده می‌کنیم. در واقع اگر ما 5 مرحله از سیگنال ورودی، تبدیل بگیریم، به ضرایب D1 D2 D3 D4 D5 A5 می‌رسیم که این ضرایب نماینده ای از انرژی سیگنال اصلی هستند. همچنین همانطور که در صفحه قبل گفته شد، هر کدام از D1 تا A5 محتوی یکسری فرکانس می‌باشند. در اینجا با توجه به بازه فرکانسی 0.5 تا 39 هرتز که مورد توجه ما می‌باشد، از ضرایب A5 D5 D4 D3 استفاده می‌کنیم که به ترتیب از راست به چپ مربوط به فرکانس های 16-32 هرتز، 8-16 هرتز، 4-8 هرتز و 0-4 هرتز می‌باشند. سپس از هر کدام از این ضرایب، ویژگی های **میانگین، ماکسیمم، مینیموم و واریانس** را استخراج می‌کنیم.

استخراج ویژگی های زمانی

ویژگی ها زمانی استخراج شده شامل خود سیگنال، میانگین زمانی و واریانس زمانی می‌باشد که در واقع هیچ یک از آنها جز سیگنال زمانی آنچنان به کار پردازش نمی‌آید.

تست ANOVA

تابع نوشته شده برای این بخش Anova1() می‌باشد. با استفاده از این برای همه ویژگی ها مقدار j محاسبه می‌شود و در ماتریس های مختلف باز گردانده می‌شود.

پدیده Overfitting:

این پدیده زمانی رخ می‌دهد که مدل لرن شده، زیاده از حد دقیق و جزیی باشد. در واقع معیار هایی که با آن مدل را می‌سازیم ممکن است یکسری نویز مخصوص به خود آن معیار ها داشته باشند و بقیه دیتا ها همچین نویز و رفتاری هایی نداشته باشند. در نتیجه اگر مدل ما نویز ها و رفتار های مختص به ورودی های یادگیری را هم جز ویژگی ها ورودی بداند و آن را یاد بگیرد، ممکن است بعد از دادن test data که آن ویژگی ها را ندارد، اشتباه متوجه بشود و آن دیتا را جزو دیتا های اصلی نداند.

معیار برای آستانه Anova و ویژگی های انتخاب شده:

با مشاهده خروجی های Anova، می‌توان دید که تعداد زیادی از ویژگی ها مقداری در اردر 10 به توان منفی 6 تا منفی 2 دارند. همچنین برخی دیگر از ویژگی ها مقادیری از 1 تا 10 و برخی دیگر مقادیر خیلی زیاد (حتی 1000 به بالا) دارند. با توجه به این موارد و به صورت شهودی، یک threshold برابر با 3 برای سیگنال زمانی و برابر با 2 برای تبدیل کسینوسی و برابر با 100 برای تبدیل ویولت می‌گذاریم. می‌گذاریم و با این کار عملا تنها ویژگی هایی که باقی می‌مانند، Cosine Transform, Wavelet Transform و خود سیگنال زمانی هستند. همچنین از هر یک از این ویژگی ها که چند خروجی دارند، لزوما همه ویژگی ها انتخاب نمی‌شود. برای مثال از 16 ویژگی مربوط به wavelet مقدار 14 یا 15 ویژگی (بسته به کانال) باقی می‌ماند و برای بقیه ویژگی ها نیز به همین ترتیب. در نهایت ماتریس ویژگی برای هر فرد تعداد ستون حدود 15 خواهد داشت.

بخش cross validation:

در این بخش با توجه به اینکه همه ویژگی های مورد نیاز را استخراج کرده ایم، از بخشی از دیتای هر فرد برای یادگیری استفاده می‌کنیم و از بخشی دیگر برای validation. به طور میانگین برای دیتای هر فرد، دقت 78 درصد را می‌گیریم.