به نام خدا

گزارش پروژه‌ی درس پردازش سیگنال‌های دیجیتال

دکتر امینی

علی فتحی

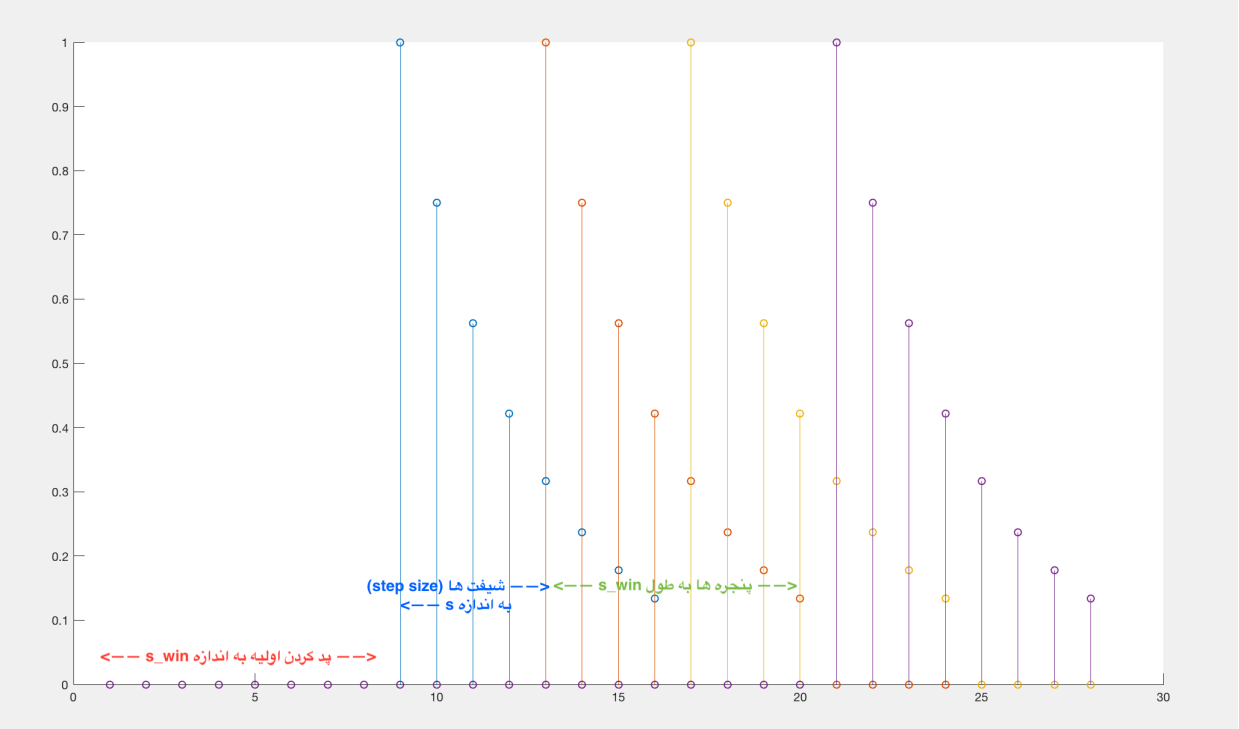
۹۴۱۰۹۲۰۵

بخش اول:

برای unwrap کردن فاز، از دستور unwrap متلب استفاده نشده است و کد آن بصورت کامل نوشته شده است.
در مورد تابع اضافه کردن افکت، چند مورد ناسازگاری و اشتباه مشاهده کردم که به شرح زیرند :

1. خروجی Fs تابع هیچگاه استفاده نشده است و این باعث warning متلب می‌شود. به همین دلیل از لیست خروجی‌های تابع برداشته شده است.

2. در قسمت Padding ، تعداد صفرهای پد شده به گونه‌ای بود که پارامتر n_seg عددی طبیعی در نمی‌آمد. خواسته‌ی ما از پد کردن شبیه شکل زیر است:



فاصله هر دو پنجره‌ای که برمیداریم، s است و تنها آخرین پنجره است که پس از آن تعداد بیشتری نقطه (s_win) مستقلاً ظاهر می‌شود. این بیانگر این است که جدا از یک تعداد نقاط به طول s_win که همین تعداد در ابتدا اضافه شده است، باید تعداد نقاط باقی‌مانده بر s بخش‌پذیر باشد. پس لازم است هرچه تعداد نقطه از این کمتر داریم بیافزاییم.
کد اولیه این بود:

```
input         = [zeros(s_win,1);input;zeros(s_win-mod(l,s),1)];
```

که به کد زیر آنرا تغییر دادم:

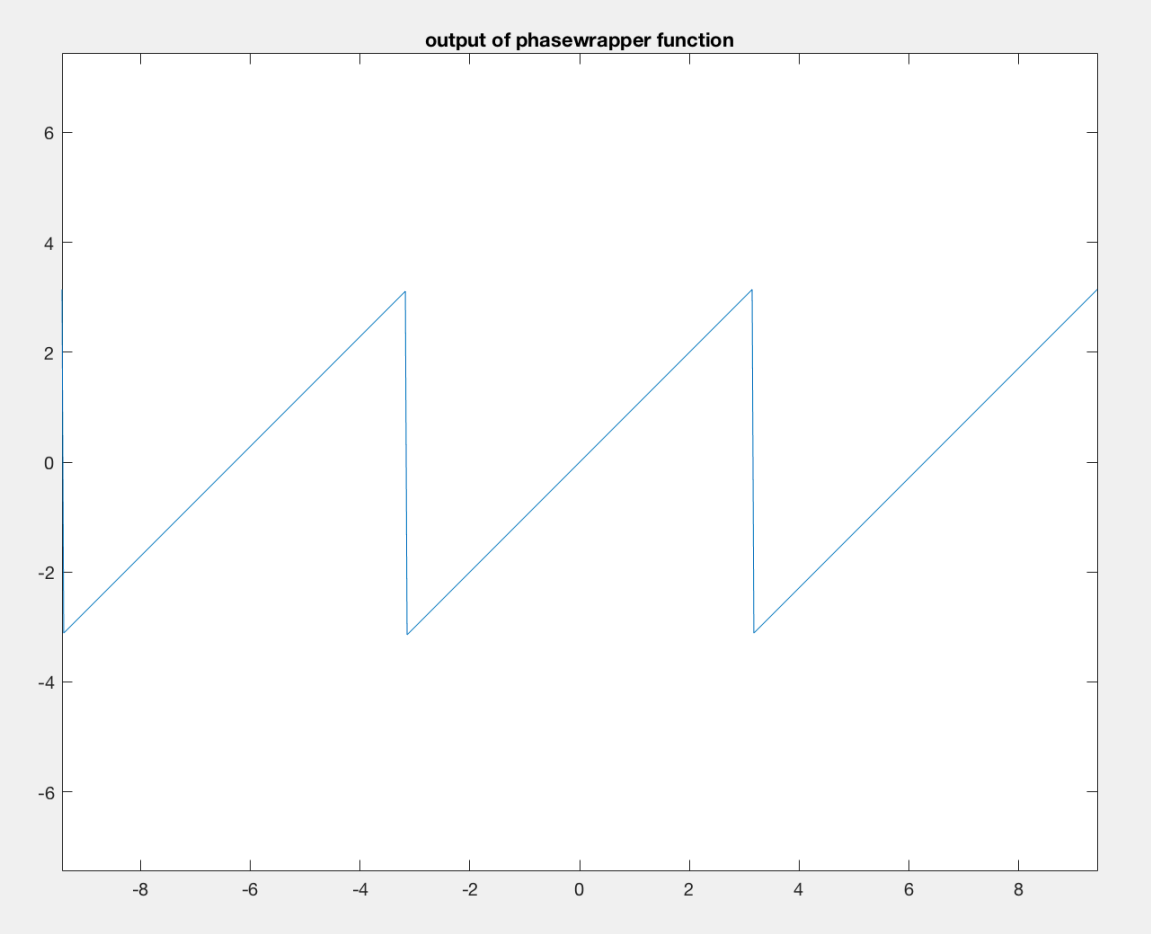
```
endPadding = mod( (s-mod(l,s)) , s );
input         = [zeros(s_win,1);input;zeros(endPadding,1)];
```

همچنین با این اوصاف، تعداد پنجره‌ها باید ۱ عدد بیشتر از آنچه آمده باشد، و به صورت زیر اصلاح شده است:

```
n_seg         = (length(input)-s_win)/s + 1;
```

قسمت اول
بخش ۴.۰)

نتیجه unwrap کردن بصورت زیر است:



بخش ۴.۱)

نکته: در بخش خش‌دار کردن و هم روبوتیزه کردن صدا، هنگام ساختن صدای خروجی سگمنت‌ها با overlapشان باهم جمع شده اند. در حالی که عملیات ما ضرب است و فیلتر نیست که همپوشانی در آن لازم باشد و اینجا یک سیگنال دوبار جمع می‌شود. برای همین نکته است که حتی وقتی ضریب خشدار کردن برابر است (no hoarsening) صدا با صدای اولیه تفاوت دارد. البته شایان ذکر است برای این overlap پنجره hamming به نظر مناسب‌ترین انتخاب می‌باشد. در کد، من با یک حالت دیگر هم پاسخ ها را تست کردم. استفاده از پنجره مستطیلی rectangular و کد زیر:

```
% Default Win, with overlap
output(pointer+1:pointer+s_win) = ...
            output(pointer+1:pointer+s_win) + result;

% My own one
%output(pointer+1:pointer+s_win) = result(:);

pointer    = pointer + s;
```

که در این عکس کامنت شده است. در اینجا به جای اورلب، هر سگمنت تنها قسمتی به طول s از خروجی را می‌سازد. با اینکار اگر ضریب خشدار کردن را صفر قرار دهیم صوت نهایی دقیقا با صوت ورودی مشابه است (و خودم از این راه راضی تر بوده ام!) . این کد کامنت شده است اما در فایل خروجی برای مقایسه، نمونه ای با ضریب خشدار کردن ۰.۹ را قرار داده ام (با پسوند MyOwnWin) . در خروجی قسمت خشدار، صداهایی با ضریب خشدار شدن ۰ (صرفا برای مقایسه)، ۰.۵، ۰.۹ و ۱ قرار داده شده است.

بررسی (باز چک کردن) مهم‌تر بودن اثر دامنه در سیگنال صوتی:

همانطور که مشخص است، سیگنال خروجی ساخته شده ما با ضریب خشدار کردن ۱ هم، که فاز به تمامی ممکن است به صورت رندوم بهم بریزد، برای ما به خوبی قابل شنیدن است و کل پیام گفته شده شنیده می‌شود، و این بیانگر این است که حتی اگر کل فاز را هم تغییر دهیم باز صوت قابل فهمیدن است.

بخش ۴.۲)

در خروجی قسمت روبوتیزه کردن، صداهایی با ضریب اسکیل پنجره ۱، ۲ و ۱۶ قرار داده شده است. قابل تشخیص بودن صدایی با فاز حذف شده (با پنجره‌ای مناسب، در حالت multiplier_win=1) ادعای بخش قبل را تصدیق میکند (که اندازه در صوت از فاز مهم‌تر است). هرچه طول پنجره را بزرگتر میکنیم، صدا غیر قابل تشخیص‌تر می‌شود بصورتی که در حالت اسکیل با ضریب ۳۲ صدا تقریبا دیگر قابل تشخیص نیست. علت آن هم بصورت خلاصه توضیح داده شد، و آن همان overlap است. هرچه طول پنجره نسبت به شیفت هر سگمنت بیشتر باشد قسمت های بیشتری از آنها روی هم افتاده و همدیگر را خراب میکنند.

بخش اول، قسمت دوم

بخش ۶)

کد این بخش به صورت زیر است:

```
% Default code:
%ft = zeros(s_win,1);
ft = (a(:).* exp(1i*phi(:)));
```

که البته کد آماده نوشته شده در قسمت خشدار کردن راهنمای نوشتن این بوده است. همچنین بابت استفاده از متغیر j در این قسمت و آزاد گذشتن i ممنونیم!

بخش ۷)

{ **یک تغییر دیگر در کد اولیه:** هنگام ذخیره صوت خروجی، بردار ratio تعریف نشده است و باید آن را اضافه می‌کردیم. }

```
221          % Creating the output name
222          % Added a Line:
223 -        ratio        = [rmin rmax];
224 -        rationame    = [num2str(min(ratio)) 'to' num2str(max(ratio))];
225 -        filename     = ['Section2-subsec' num2str(sample number)...
```

{ **یک مسئله دیگر در کد اولیه:** کد آماده به ازای sub_section=2 خروجی مناسبی نمی‌دهد و شباهت کارکرد آن با قسمت ۳ مشخص نیست. }

بخش ۷.۱)

حالت اول: sub_section=1 (حالت ساده‌انگارانه) و ضرایب کشش هردو برابر واحد rmin=1 , rmax=1 :

سیگنال صوتی شنیده شده به ترتیب یک سیگنال downsample شده و دو سیگنال upsample شده است.
سیگنال downsample شده شبیه صدای ضبط سریع شده است (صدای موش مانند!) که صدایی زیر است، و دو صدای upsample شده صدای بمی هستند که کند شده صوت اولیه هستند.
فرق دو مدل آپ‌سمپل شده (up و up2): در حالت ساده صدا شدت کمتری دارد، یعنی در واقع پر کردن صفر کردن میانی این اثر را دارد که شدت صدای نهایی را بیشتر می‌کند و در فرکانس و شکل صدا تاثیری ندارد. در واقع گویی صفر ها شدت صوت را میکاهند (سیگنال ورودی، سیگنال زیر شده و سیگنال بم شده با روش up2 شدت های یکسانی دارند).

حالت دوم: sub_section=3 (حالت پیشرفته) و ضرایب کشش هردو برابر واحد rmin=1 , rmax=1 :

سیگنال صوتی شنیده شده در این حالت تفاوتی با صوت اولیه ندارد.

حالت سوم: sub_section=3 (حالت پیشرفته) و ضرایب کشش هردو برابر نصف rmin=0.5 , rmax=0.5 :

سیگنال صوتی شنیده شده در این حالت سریع شده است اما صدا زیر نشده است! و انگار شخص گوینده خودش سریعتر حرف زده است.

حالت چهارم: sub_section=3 (حالت پیشرفته) و ضرایب کشش هردو برابر دو rmin=2 , rmax=2 :

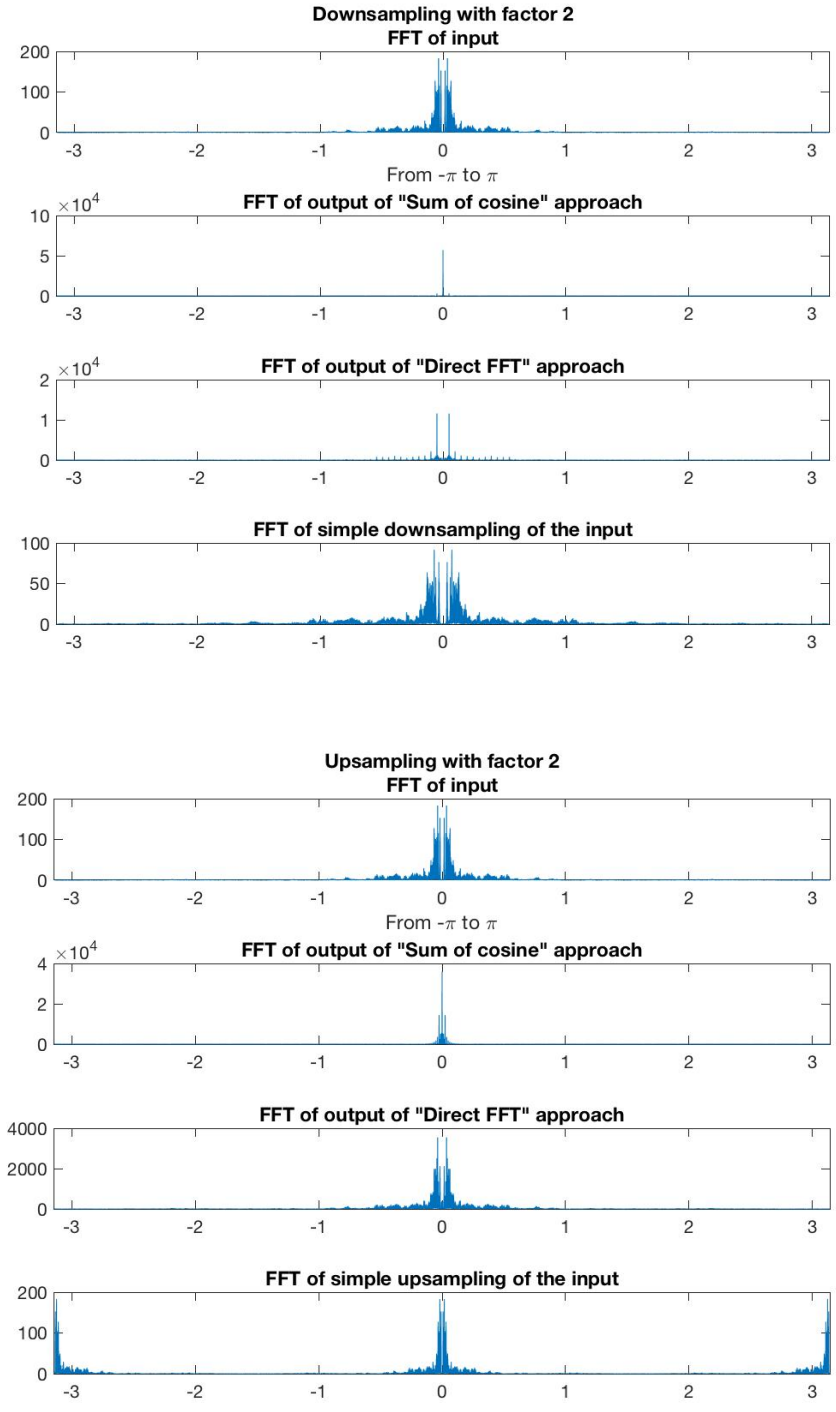
سیگنال صوتی شنیده شده در این حالت نیز با فرکانس گوینده کند شده است(بم نشده) و انگار شخص گوینده خودش آرامتر حرف زده است.

بخش ۷.۲)

sub_section=3 (حالت پیشرفته) و ضرایب کشش متفاوت rmin=0.33 , rmax=1.5:

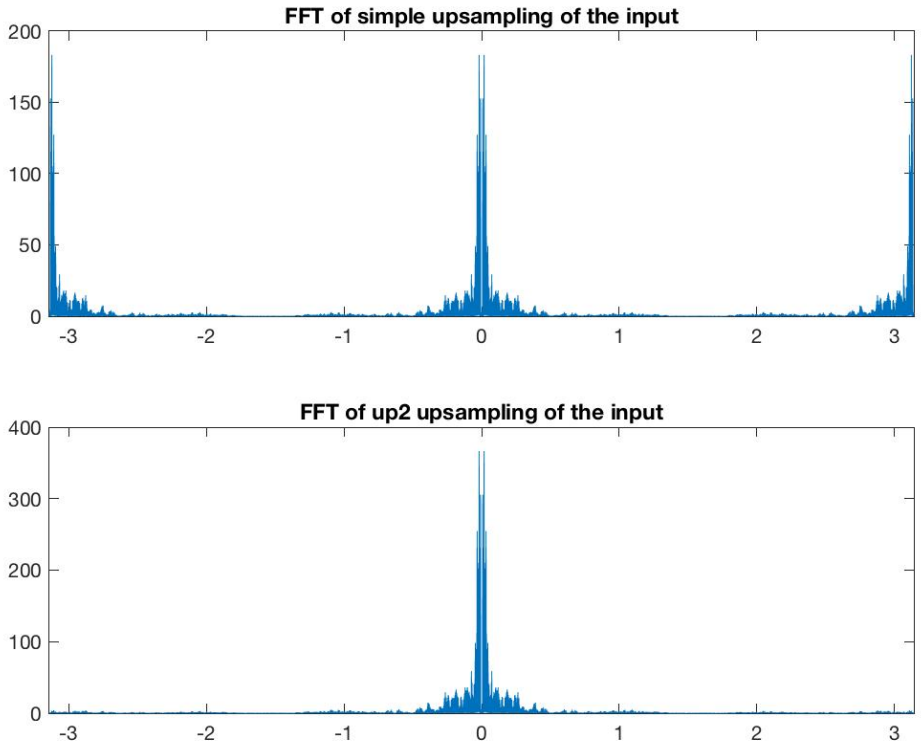
سیگنال صوتی شنیده شده در این حالت ابتدا کند شده و به تدریج سریع می‌شود. فرکانس صحبت همان فرکانس گوینده است.

بخش ۷.۳)



دو تصویر آورده شده، نمودارهای upsample و downsample شده‌ی ورودی با نرخ ۲ هستند که با ۳ روش این بخش تهیه شده اند.

بخش ۷.۴) امتیازی

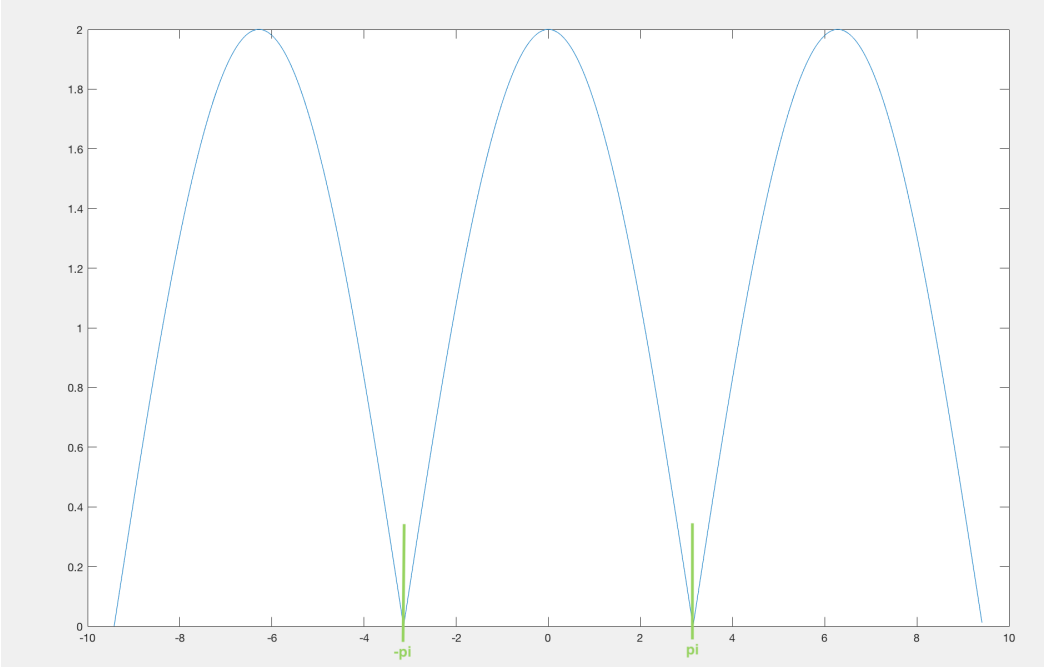


تفاوت این دو نمودار، در فرکانس های بالا (نزدیک پی) در آنهاست، و اینکه فرکانس های پایین در روش دوم تقویت شده اند.
دلیل آن این است که در روش دوم گویا سیگنال up با شیفت یافته خود به اندازه یک واحد جمع شده است، و این شبیه یک فیلتر پایین گذر است. روابط ریاضی در ادامه آمده است:

L: Length of input Signal => Length of up & up2 signals = 2L
up2(1:2L) = up(1:2L) + concatenation{ 0, up(1:2L-1) }

- ⇨ in System form : $up2[n] = up[n] + up[n-1]$ as a System: $y[n] = x[n] + x[n-1]$
- ⇨ $h[n] = \delta[n] + \delta[n-1]$
- ⇨ $H(e^{j\omega}) = 1 + e^{-j\omega}$

و پاسخ فرکانسی این تبدیل به شکل زیر است:



که بیان می‌دارد شدت سیگنال در فرکانس های پایین دوبرابر شده (همانطور که در بخش مربوطه ذکر شد) و فرکانس‌های بالا هم که در اثر upsample بوجود می‌آیند از آن حذف شده است (تشخیص آن با شنیدن دشوار است برای همین در بخش ۱ به ان به عنوان چیز قابل درکی اشاره نکردم).

بخش دوم:

بخش ۱)

برای ساخت پنجره‌ی Blackman-Harris فرض میکنیم طول پنجره همان طول FFT ، یعنی N است. همچنین bin های فرکانسی را فرکانس های ضریب $2\pi/N$ فرض میکنیم.

{ نکته: رابطه پنجره B-H را عبارت زیر یافتم اما مطابق با خواسته سوال، تناوب کسینوس‌ها را N و ضرایب a را مثبت گرفتم. }

$$w(n) = a_0 - a_1 \cos\left(\frac{2\pi n}{N-1}\right) + a_2 \cos\left(\frac{4\pi n}{N-1}\right) - a_3 \cos\left(\frac{6\pi n}{N-1}\right)$$

بخش ۲)

هر سینوس در حوزه زمان، با ضرب در یک پنجره Blackman-Harris محدود می‌شود. این به این معنی است که در حوزه فرکانس bin های فرکانسی پنجره Blackman-Harris به مرکز فرکانس سینوسی می‌روند. پس کافی است این bin ها را در اندازه سیگنال سینوسی (pmag متناظر) ضرب کنیم و مرکز آن را به محل loc ببریم.

بخش ۳)

این بخش جزییات زیادتری دارد که به شرح زیرند:

1) ابتدا بردار d با رابطه‌ی زیر محاسبه شده است:

$$d(\tau) = \sum_{j=1}^{j=W} (x_j - x_{j+\tau})^2$$

که درواقع در رابطه داده شده t=0 قرار داده شده است. در کد نوشته شده مقدار d(0) که برابر صفر است محاسبه نشده است و این بردار با d(1) آغاز می‌شود.

2) سپس با رابطه داده شده d’ محاسبه شده است (در کد با نام d2 آمده است):

$$d_t(\tau') = \frac{d_t(\tau)}{(1/\tau)\sum_{j=1}^{\tau} d_t(j)}$$

که در کد این بخش هم مقدار d’ (0) که برابر یک است در بردار قرار داده نشده است (و چون مطمئناً از مقدار threshold کمتر نخواهد بود نبودنش اهمیتی ندارد.)

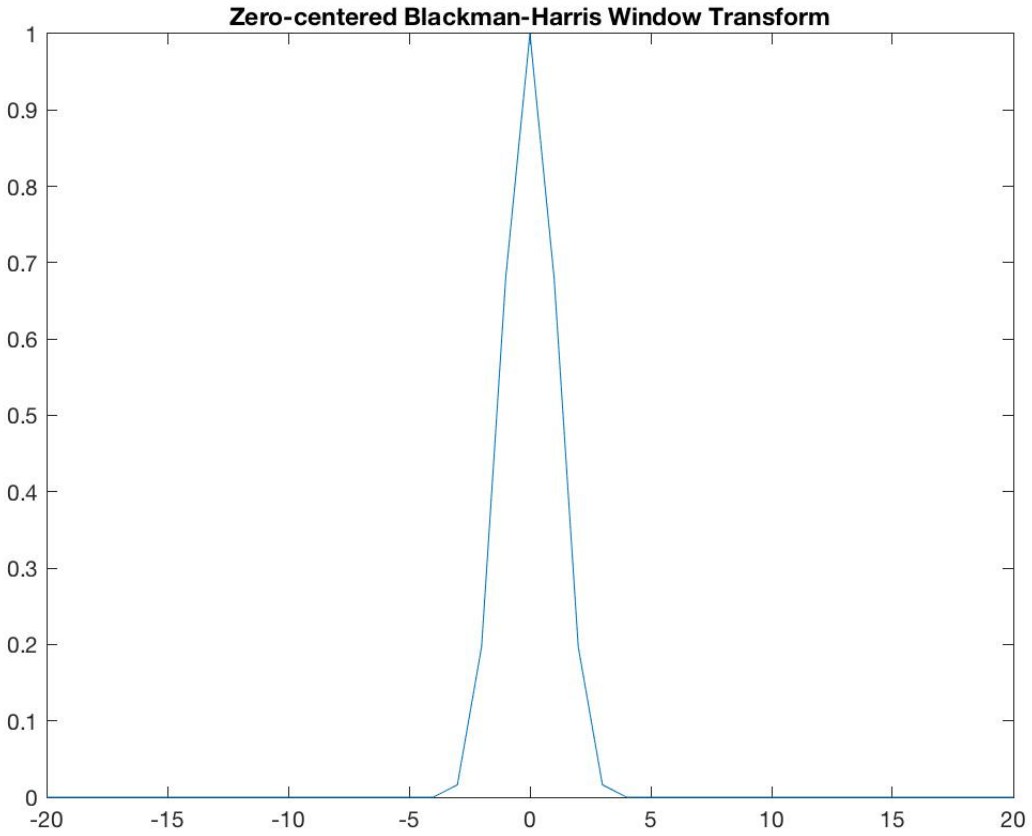
3) مقدار d’ برای شیفت هایی خارج از $\tau_{min} = \frac{f_s}{f_{max}}$ و $\tau_{max} = \frac{f_s}{f_{min}}$ برابر یک مقدار بزرگ (در اینجا ۱۰۰) قرار داده شده اند تا فرکانس‌های خارج از محدوده f_{min} و f_{max} به عنوان مینیمم این تابع انتخاب نشوند.

4) سپس کاندید Lag مطلوب را پیدا می‌کنیم. اگر هیچ کدام از مقادیر d’ کمتر از ۰.۲ نباشند، داده ها خوب نبوده و خروجی را $f_0 = 0$ به عنوان داده‌هایی با فرکانس اصلی نامشخص اعلام می‌کنیم. اگر هیچکدام کدام از مقادیر d’ کمتر از threshold برابر ۰.۱ نباشند، شیفتی که کمترین مقدار را دارد به عنوان کاندید Lag مطلوب اعلام می‌کنیم. در غیر این صورت اولین مقدار کمتر از threshold کاندید Lag مطلوب است.

5) در اینجا برای دقت بیشتر در تشخیص فرکانس اصلی سیگنال، حول مقادیر فرکانس پیدا شده تقریب درجه ۲ (سهمی) می‌زنیم. در اینجا به جای ۳ نقطه از ۵ نقطه استفاده کرده‌ام (یعنی ۲ نقطه قبل و بعد شیفت کاندید شده، زیرا ۵ نقطه نتیجه بهتری از ۳ نقطه بدست داد). در کد از دستور polyfit استفاده شده است که سه ضریب را در معادله $p_2x^2 + p_1x + p_0$ فیت میکند. سپس با دستور poleder ضرایب مشتق آن حساب شده است زیرا جایی که عبارت $p_2'x + p_1' = 2p_2x + p_1$ صفر شود، همان مینیمم مدنظر ماست. (در کد برای مواجهه نشدن با عرض از مبدا زیاد، تقریب حول نقطه کاندید زده شده است نه حول صفر).

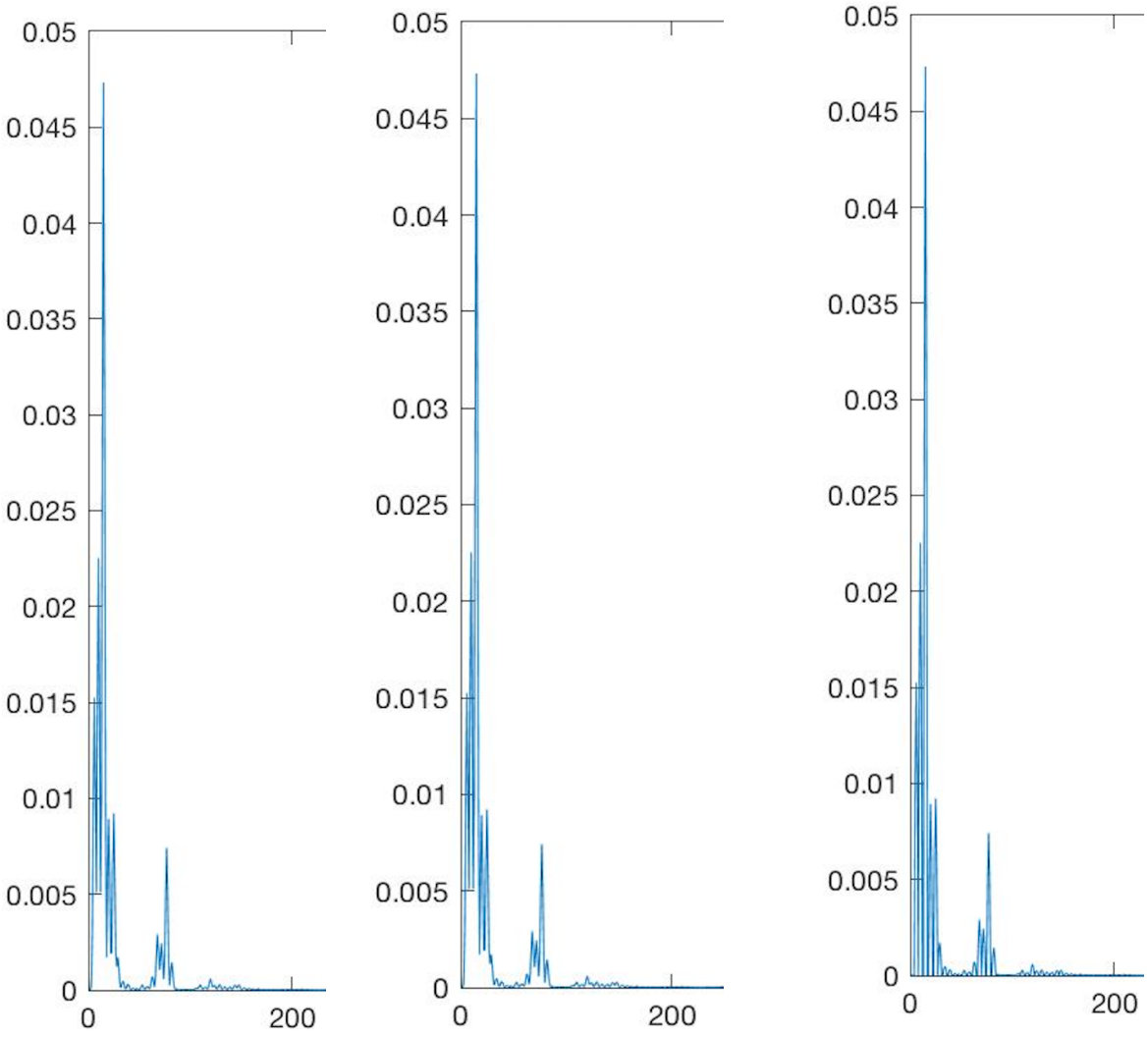
بخش تحلیلی ۴)

الف) نمودار فرکانسی bin های پنجره Blackman-Harris به شکل زیر است:



همانطور که مشخص است، بیشتر انرژی آن هول صفر است (تنها bin های ۱ و ۲ و ۳ و منفی هایشان نا صفرند).
همچنین تغییر بازه bin ها تأثیری در شکل نمودار ندارد و فقط صفر های کناری آن را می‌افزاید یا کم می‌کند.

ب) نتیجه یک تکه از فرکانس سیگنال تست برای تعداد bin به ترتیب از راست به چپ برابر ۱، ۴ و ۱۰ به شکل زیر است:



مشاهده می‌کنیم تعداد bin های ۱ هنوز تمام باند فرکانسی را پوشش نمی‌دهد، اما تعداد bin های ۴ و ۱۰ تفاوت قابل ملاحظه‌ای ندارند. زمان محاسبه نیز نسبت به تعداد bin ها خطی است (زیرا در هر فرکانس باید به تعداد bin ها جاینمایی کند، درواقع for داخلی به تعداد bin هاست.) پس استفاده از تعداد bin برابر ۴ کافی و بهینه است.

پ) نتیجه پیدا کردن فرکانس اصلی سیگنال برای طول‌های متفاوت پنجره به شرح زیر است:

Window Length (sec)	f_0 (Hz)
0.003	0.0000
0.005	240.798519
0.01	241.554671
0.0125	240.907751

همانطورکه ملاحظه می‌شود، فرکانس اصلی سیگنال حدود ۲۴۰ هرتز است، و این یعنی حدود هر $n_{sample} = f_s/f_0 \cong 184$ نمونه یکبار سیگنال به شکل مشابهی تکرار می‌شود، یا به عبارتی دیگر برای مشاهده دوباره‌ی سیگنال، باید حداقل زمان $T_0 = \frac{1}{f_0} \cong 0.0042\ sec$ از آن را مشاهده کنیم. برای همین است که پنجره اول به طول ۰.۰۰۳ ثانیه که طولی کمتر از ۰.۰۰۴۲ ثانیه دارد در تشخیص فرکانس اصلی ناکارآمد است. از طرفی هنگام محاسبه، maxLag نباید از طول سیگنال بیشتر شود، و از آنجا که در کد این پارامتر از قبل تعریف شده است، طول پنجره نباید از ۰.۰۲۵ ثانیه بیشتر شود. از طرفی هر چه طول پنجره بیشتر باشد نتیجه دقیق‌تر است (زیرا ایده‌آل ریاضی جمع از منفی بی‌نهایت تا مثبت بی‌نهایت است)، بیشترین مقدار تست شده یعنی ۰.۰۱۲۵ برای انتظارات ما کافیست.

ت) دو اثر fscale و timbremapping را مشاهده کرده ایم. fscale ، صدا را زیر و بم می‌کند (و پارامتر اصلی در مردانه یا زنانه کردن صداست) و timbremapping باعث ته دماغی شدن صدا می‌شود (مثلا اگر پارامتر خروجی بیشتر از ورودی باشد شبیه صدای موجودات کوتوله در کارتون‌ها می‌شود). در واقع پارامتر fscale ، فرکانس های اصلی صدا را به محدوده دیگری مقیاس میکند، و این کار را بصورت خطی و همگن انجام می‌دهد (کد تابع افکت دادن را هم نگاه کرده‌ام). اما پارامتر timbremapping ، مانند یک افکت غیر خطیست که در فرکانس های پایین (بین ۰ تا حدود ۴۰۰۰ یا ۵۰۰۰ هرتز) یک جور و در فرکانس های بالا (از حدود ۴۰۰۰ یا ۵۰۰۰ هرتز تا $f_s/2$ برابر ۲۲۰۵۰ هرتز) صوت را جور دیگری مقیاس می‌کند (بخاطر استفاده از interp1 خطی با ۳ نقطه). هرچه fscale را بیشتر کنیم، صدا زنانه‌تر(زیرتر) می‌شود و هرچه ناحیه مقصد timbremapping را بیشتر کنیم، صدا بچگانه‌تر می‌شود. در این قسمت، از صوت اول (basket) استفاده شده است.

ث) صوت خروجی این قسمت در فولدر این بخش آمده است (از صوت دوم، meeting استفاده شده است).

ج) صوت خروجی این قسمت نیز در فولدر این بخش آمده است که برای آن از صوت اول، basket، استفاده شده است. خروجی شبیه صدای یک پسربچه است. برای امتحان، صدای صوت دوم(meeting) را هم از این افکت با همان ضرایب عبور داده ام که خروجی به شکل خیلی خوبی شبیه صدای یک دختربچه است!

---------- پایان ----------