به نام خدا

دانشگاه تهران
دانشکده مهندسی برق و کامپیوتر

پردازش سیگنال‌های زمان-گسسته

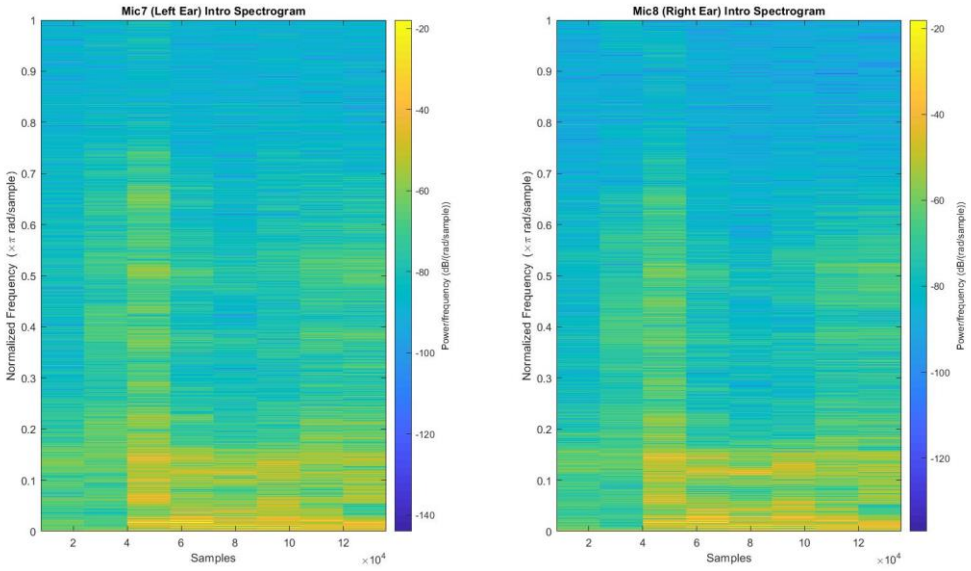
تمرین کامپیوتری سری سوم

نام و نام خانوادگی	یاسمن پرهیزکار
شماره دانشجویی	۸۱۰۱۹۵۵۵۹
تاریخ ارسال گزارش	۱۳۹۹/۵/۳

فهرست گزارش سوالات

سوال ۱ – رسم Spectrogram

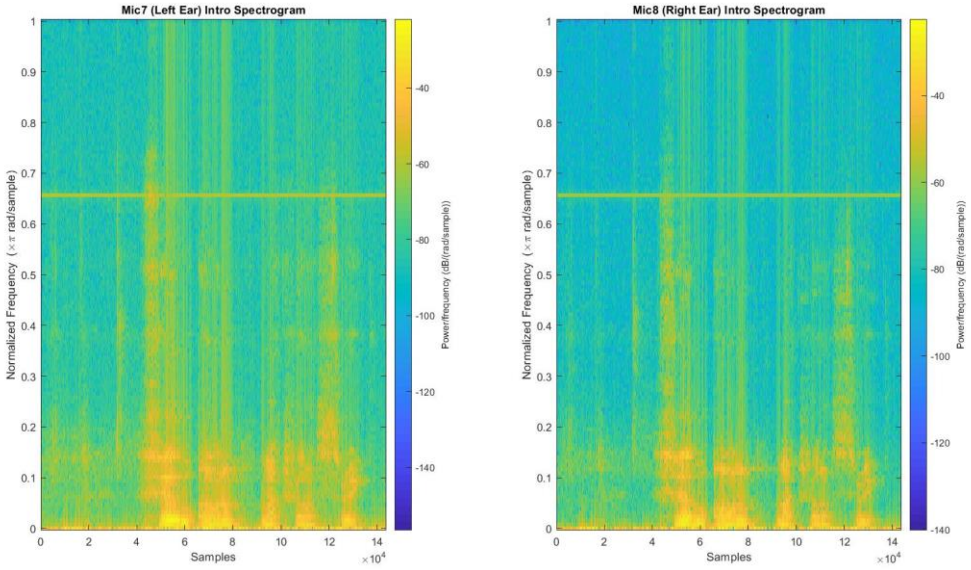
با دستور spectrogram() متلب، نمودار های خواسته شده را رسم می کنیم. اگر از تنظیمات دیفالت این دستور استفاده کنیم، کل سیگنال را به سگمنت هایی با سایز floor(length(intro)/4.5) و ۵۰ درصد overlap تقسیم می کند و هر سگمنت را قبل از گرفتن STFT از پنجره همینگ رد می کند.



شکل 1- نمودار spectrogram سیگنال معرفی، میکروفون ۷ (چپ) و ۸ (راست)، با تنظیمات دیفالت

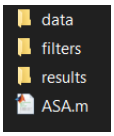
اما برای اینکه spectrogram کشیده شده در این بخش، با سلول های زمان-فرکانس بخش بعدی تطابق داشته باشد، یک نمودار هم با تنظیمات زیر می کشیم:

سیگنال به سگمنت هایی با طول ۲۵۶ نمونه تقسیم می شود، بدون overlap و پنجره هم مستطیلی با همان سایز ۲۵۶ است.



شکل 2- نمودار spectrogram سیگنال معرفی، میکروفون ۷ (چپ) و ۸ (راست)، با تنظیمات دلخواه

همانطور که مشاهده می کنید، رزولوشن در حوزه زمان بیشتر و درنتیجه، رزولوشن در حوزه فرکانس کمتر شده است.

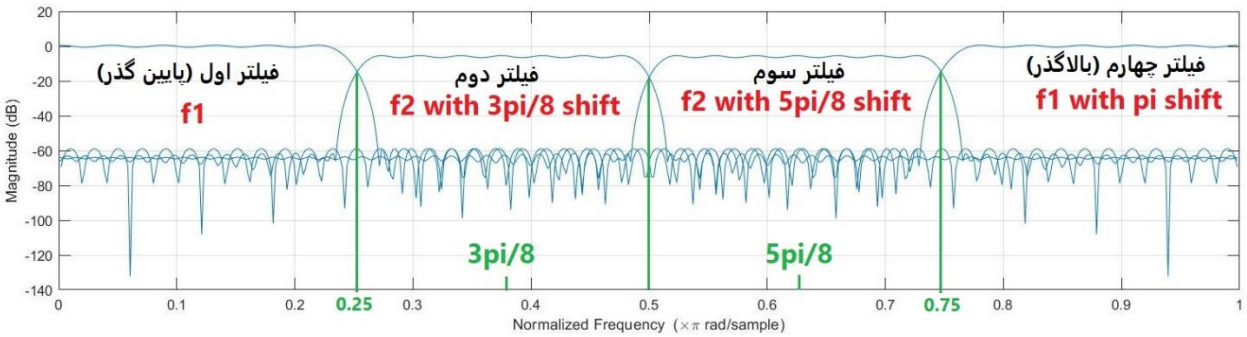


کد تمام این پروژه را می توانید در فایل ASA.m مشاهده کنید.

نکته: برای اجرا شدن کد، ابتدا پوشه ی data را در محل ASA.m کپی کرده و سپس کد را ران کنید:

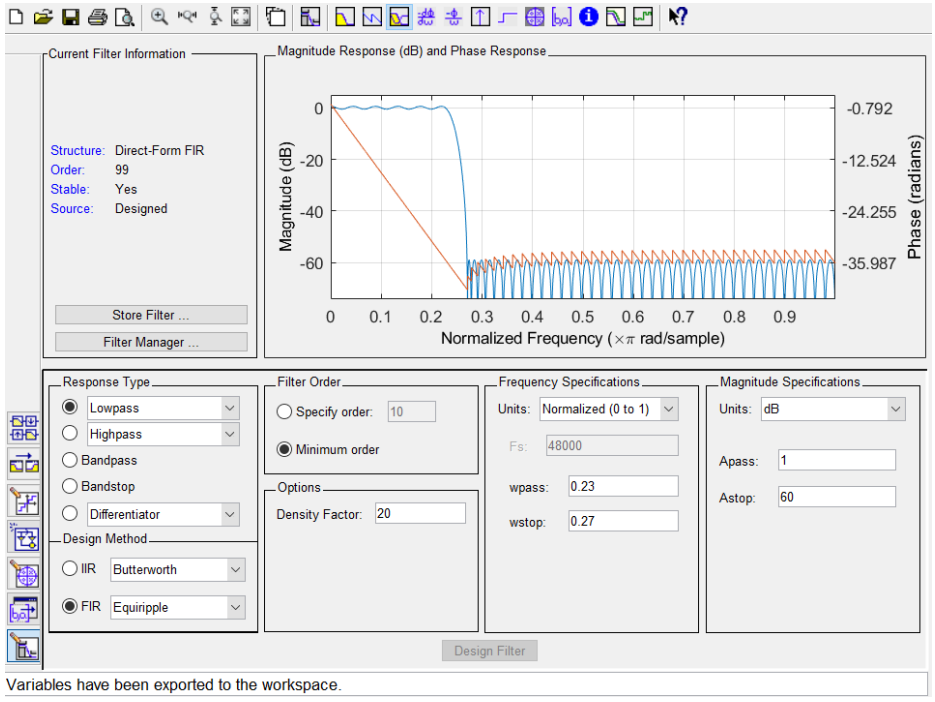
سوال ۲ – تقسیم سیگنال به چهار زیرکانال

در بانک فیلتر نیاز به چهار فیلتر داریم که پهنای باند هر کدام $\frac{\pi}{4}$ rad/symbol باشد، تا به این ترتیب، سیگنال ورودی را به چهار کانال با پهنای باند مساوی تقسیم کنیم.

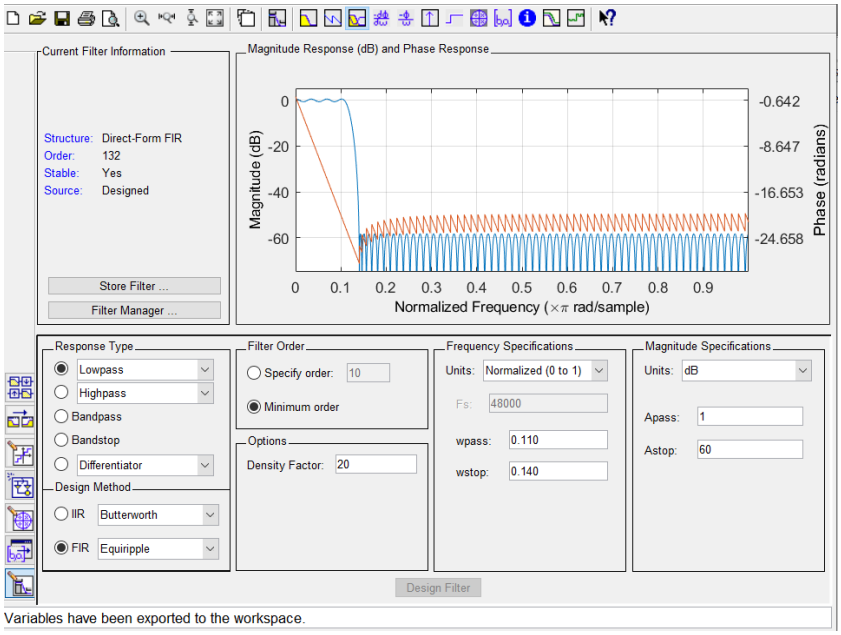


شکل 3- اندازه پاسخ فرکانسی چهار فیلتر موجود در بانک فیلتر، در یک نمودار

برای ساخت این چهار فیلتر، ابتدا در FilterDesigner دو فیلتر پایین گذر با پهنای باند $\frac{\pi}{4}$ و $\frac{\pi}{8}$ به نام های f1 و f2 طراحی می کنیم و سپس، با شیفت دادن این دو فیلتر، هر چهار فیلتر لازم را بدست می آوریم.



شکل 4- فیلتر پایین گذر f1 با پهنای باند pi/4 در محیط FilterDesigner متلب

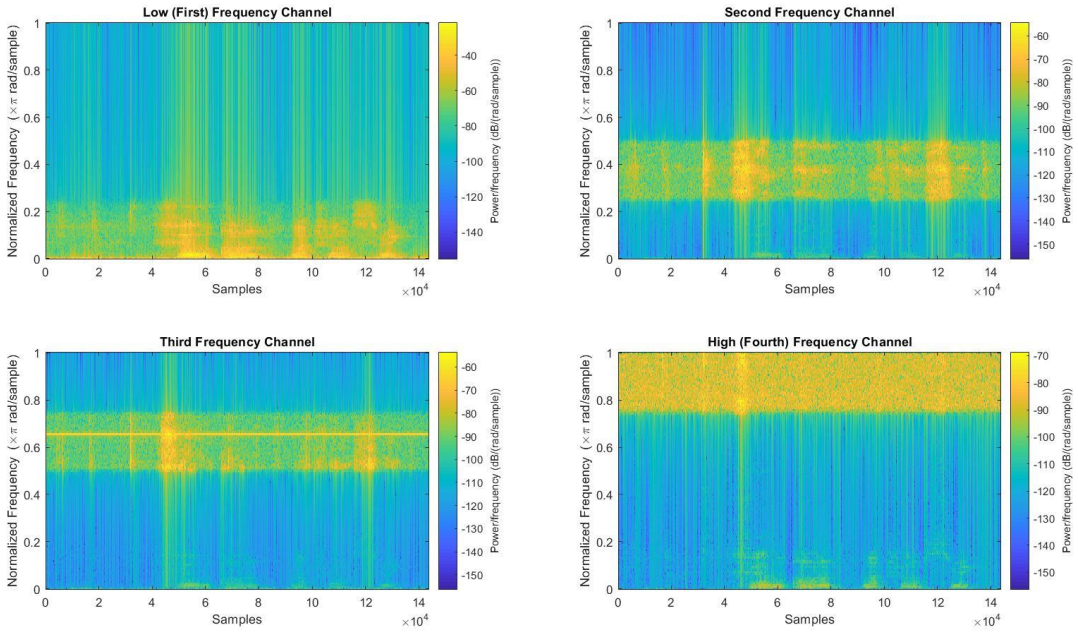


شکل 5- فیلتر پایین گذر f2 با پهنای باند pi/8 در محیط FilterDesigner متلب

فیلتر اول که همان f1 است. اما فیلتر های دوم و سوم با شیفت دادن f2 به اندازه $\frac{3\pi}{8}$ و $\frac{5\pi}{8}$ ، و فیلتر چهارم با شیفت دادن f1 به اندازه π به دست می آیند.

از آنجایی که طراحی فیلتر ایده آل عملا غیر ممکن است، این چهار فیلتر اندکی با هم overlap دارند.

* حال همانطور که در صورت سوال گفته شده، سیگنال های intro را از این بانک فیلتر رد کرده و هر کدام را به چهار زیرکانال تقسیم می کنیم. با کشیدن spectrogram هر کدام از این زیرکانال ها، به خوبی می توانیم تقسیم شدن فرکانس های سیگنال اصلی را مشاهده کنیم.



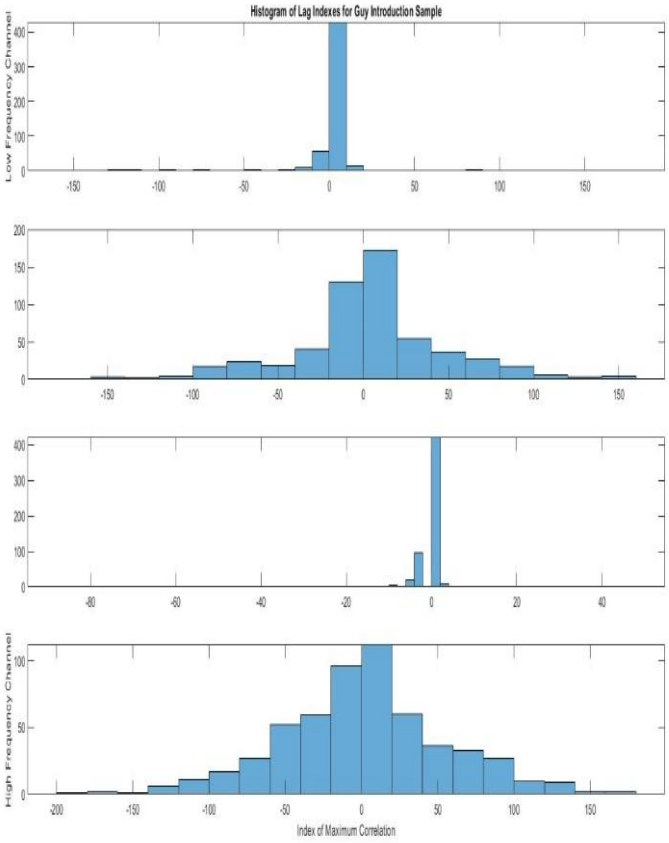
شکل 6- نمودار spectrogram چهار زیرکانال سیگنال معرفی - میکروفون V

سوال ۳ – Chunk کردن سیگنال هر زیرکانال

هر کدام از هشت سیگنال بدست آمده در قسمت قبل را به chunk های ۲۵۶ نمونه ای تقسیم می کنیم. سیگنال حاصل، یک ماتریس دو بعدی است که هر ستون آن شامل ۲۵۶ نمونه ی یک chunk می باشد (برای اطلاعات بیشتر، فایل ASA.m را مشاهده کنید).

سوال ۴ - lagindex

حال طبق صورت سوال، با استفاده از تابع xcorr متلب بین chunk های متناظر در سیگنال های میکروفون ۷ و ۸، کورلیشن گرفته و تاخیر زمانی بین این chunk ها را محاسبه می کنیم (که به این تاخیر زمانی lagindex می گویند). سپس، با استفاده از تابع histogram متلب، نمودار هیستوگرام مربوط به lagindex های هر زیرکانال را رسم می کنیم.



شکل 7- هیستوگرام LagIndex های سیگنال intro در هر چهار زیرکانال

حال، با توجه به مقادیر به دست آمده از هیستوگرام ها، میانگین (μ) و انحراف معیار (σ) lagindex ها را در هر زیرکانال محاسبه می کنیم.

فرمول محاسبه:

$\mu = \sum$ ارتفاع هر مستطیل / مقدار وسط بازه آن مستطیل روی محور افقی $*$ ارتفاع هر مستطیل $\sum$

$$= \sum_i hi * di \ / \ \sum_i hi$$

$$\sigma^2 = \sum_i (di - \mu)^2 * hi \ / \ \sum_i hi \quad \rightarrow \quad \sigma = \sqrt{\sigma^2}$$

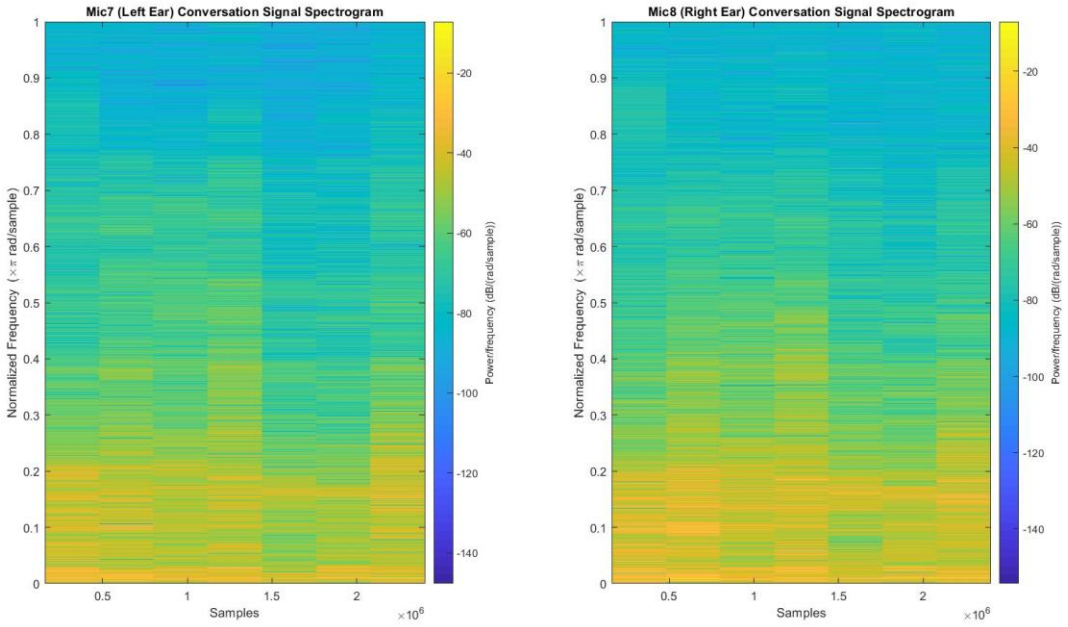
نتیجه محاسبه با متلب:

mu = [2.03, 4.03, -0.01, 1.69]

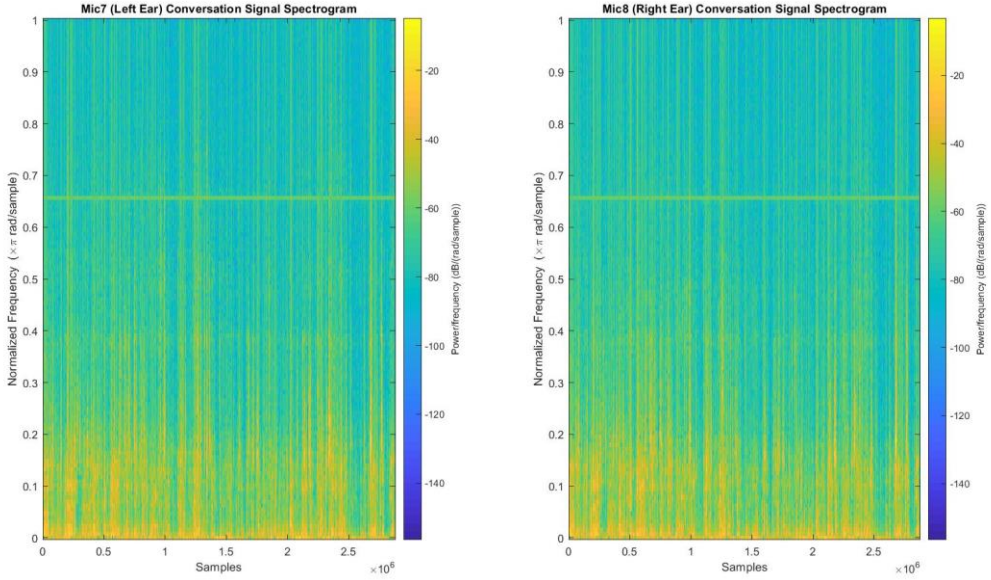
sigma = [30.17, 46.10, 5.59, 55.33]

سوال ۵ – انجام تمام مراحل برای سیگنال های اصلی

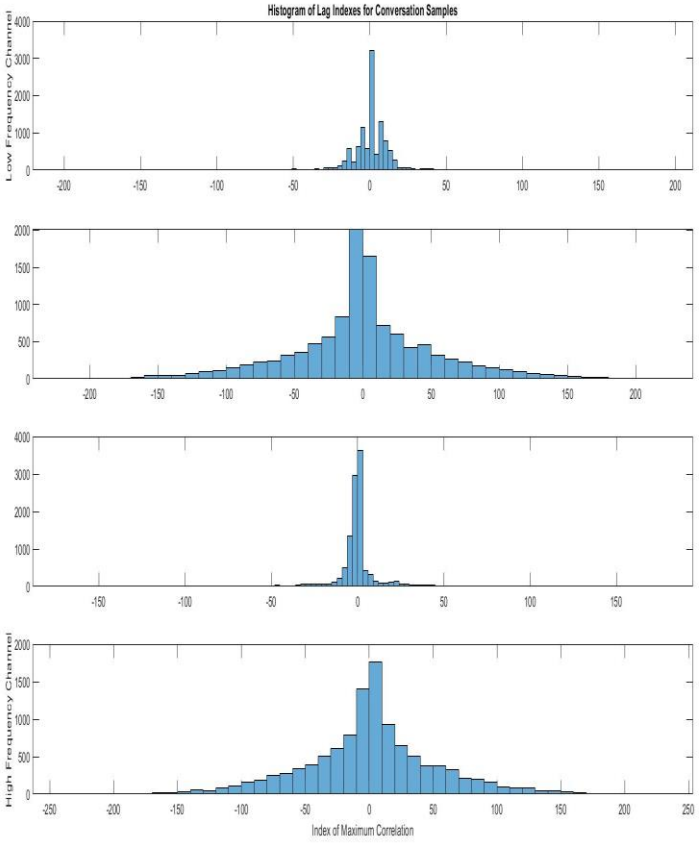
در اینجا فقط نمودارهای نتیجه گذاشته ام.



شکل 8- نمودار spectrogram سیگنال اصلی، میکروفون ۷ (چپ) و ۸ (راست)، با تنظیمات دیفالت



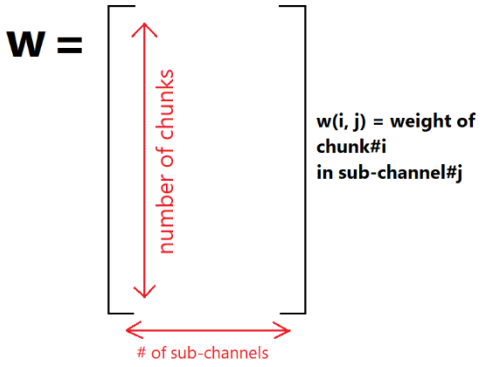
شکل 9- نمودار spectrogram سیگنال اصلی، میکروفون ۷ (چپ) و ۸ (راست)، با تنظیمات دلخواه



شکل 10- هیستوگرام LagIndex های سیگنال main در هر چهار زیرکانال

سوال ۶ – ساخت ماتریس وزن ها (w) و بازیابی سیگنال ها

طبق فرمول داده شده در صورت سوال، ماتریس w را حساب می کنیم. سپس، هر chunk سیگنال اصلی را در وزن متناظرش ضرب کرده و آنگاه، تمام چهار زیرکانال سیگنال را از SynthesisFilterBank عبور می دهیم تا دوباره سیگنال سرهم شود.

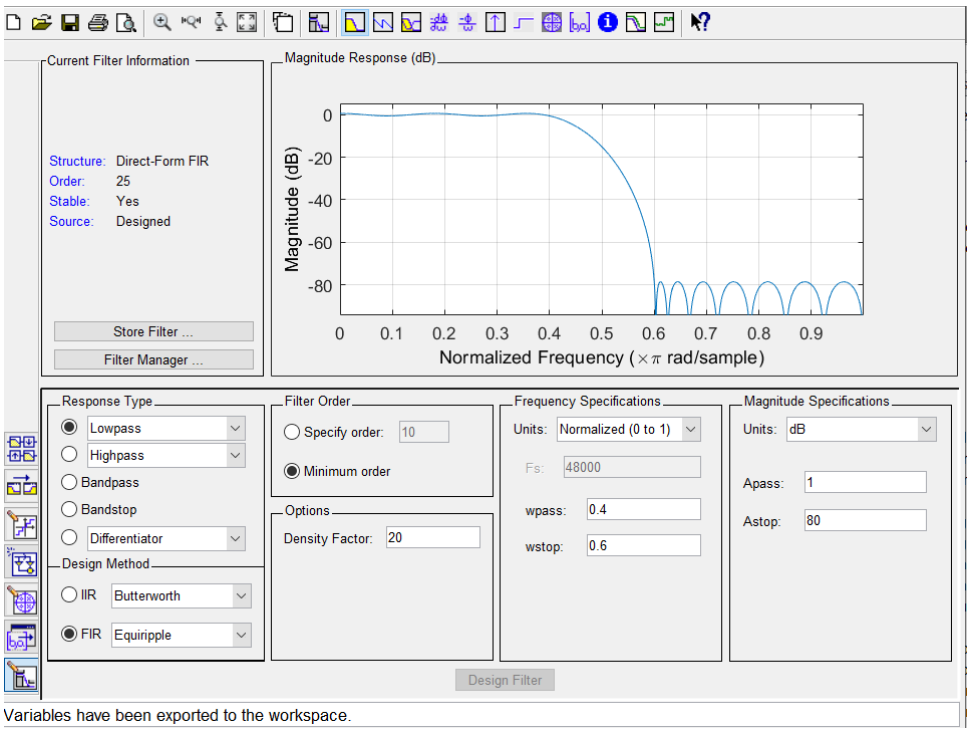


طبق فرمول داده شده، وزن مربوط به هر chunk طوری بدست می آید که اگر lagindex ـ آن chunk به lagindex ـ نفر سوم نزدیک باشد، آن chunk تقویت شده و در غیر این صورت، تضعیف می شود.

در SynthesisFilterBank اگر دوباره زیرکانال ها را از فیلتر مربوط به خودشان رد کنیم، تاثیر وزن دار کردن chunk ها تاحدی از بین می رود؛ به همین دلیل، تصمیم گرفتم که در SynthesisFilterBank صرفا سیگنال های همه زیرکانال ها را با هم جمع کنم، بدون فیلتر کردنِ دوباره.

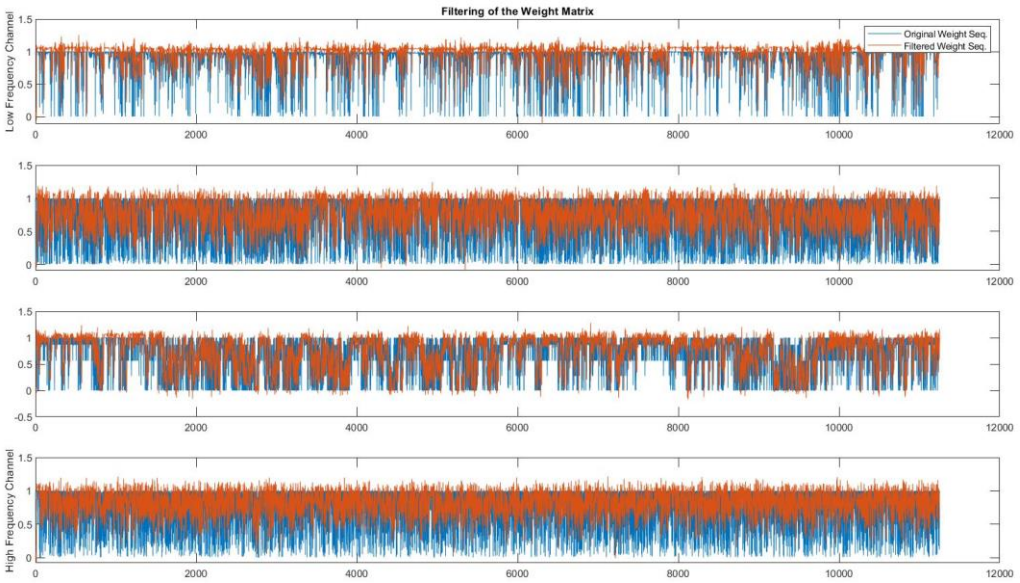
سیگنال بازیابی شده میکروفون های ۷ و ۸، به ترتیب با نام های processed_sig1.wav و processed_sig2.wav ذخیره شده اند.

* حال، وزن های بدست آمده برای هر زیر کانال را با فیلتر پایین گذر f0 فیلتر می کنیم.



شکل 11- فیلتر پایین گذر f0 در محیط FilterDesigner متلب

تغییر وزن های w را پیش و پس از فیلتر شدن، در شکل ۱۲ مشاهده می کنید.



شکل 12- وزن های هر زیرکانال، پیش از فیلتر شدن (آبی) و پس از فیلترشدن (قرمز)

سیگنال ها را با وزن های فیلترشده، دوباره بازیابی کرده و با نام های fil_processed_sig1.wav و fil_processed_sig2.wav ذخیره کردیم.

سوال ۷ – مقایسه و نتیجه گیری

با گوش دادن به صوت های بخش قبلی (مثلا fil_processed_sig1.wav) فرق محسوسی بین آن ها و صوت اولیه احساس نکردم. به همین دلیل، فرمول محاسبه ی وزن ها را کمی تغییر داده و واریانس زیر کانال ها (σ_{ch}^2) را کوچک تر گرفتم.

هرچه واریانس را کوچک تر درنظر بگیریم، وزن chunk های غیر مرتبط با گوینده سوم کم تر شده و درنتیجه، صدای افراد غیر از گوینده سوم بیشتر تضعیف می شود. اما درعوض، این خطر وجود دارد که برخی از chunk های مرتبط با گوینده سوم هم تضعیف شوند (چون بالاخره، تاخیر زمانی chunk های مربوط به گوینده سوم هم کمی انحراف معیار دارند). همچنین، به دلیل تغییرات شدیدتر w در chunk های مجاور، نویز سیگنال حاصل بیشتر می شود.

برای محسوس شدن این تاثیرات، من یک حالت شدید درنظر گرفتم: واریانس ها را ۱۰۰ برابر کوچک کردم! یعنی:

$$W_{ch}[n] = \exp\left(-100 * \frac{(lag - \mu_{ch})^2}{2\ \sigma_{ch}^2}\right)$$

سیگنال میکروفون ۷ را با این وزن ها بازسازی کرده و صوت حاصل را با نام extreme_sig1.wav ذخیره کرده ام و فرستاده ام.

همانطور که با شنیدن این صوت ها متوجه می شوید، صدای افراد نامطلوب تا حدود زیادی حذف شده (تقریبا در حد صوت solo حذف شده اند)، اما نویز صدا آنقدر زیاد است که تشخیص کلمات خود گوینده سوم هم کار راحتی نیست. از طرفی، برخی از chunk های مربوط به خود گوینده سوم در اواسط فایل (هنگام گفتن yeah و Hmm) هم تضعیف شده است.

سپس، این وزن های شدید را با همان فیلتر f0 فیلتر کرده و صوت ها را دوباره بازیابی کردم، اما متاسفانه به دلیل پرحجم شدن فایل ارسالی نتوانستم صوت حاصل را بفرستم.

اما به هر حال، در این یکی فایل چون فیلتر کردن w تغییرات آن را کند می کند، کمی همصدایی بیشتر شده (یعنی در برخی chunk ها صدای گوینده های نامطلوب حذف نشده، یا هم صدای گوینده های نامطلوب و هم گوینده ی سوم با هم تضعیف شده)، اما در عوض نویز کم تر شده است.

پایان