فاز دوم

- فرض کنید فیلتری دارید که اندازه‌ی آن ثابت است. فاز این فیلتر را نیز به صورت زیر در نظر بگیرید:

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

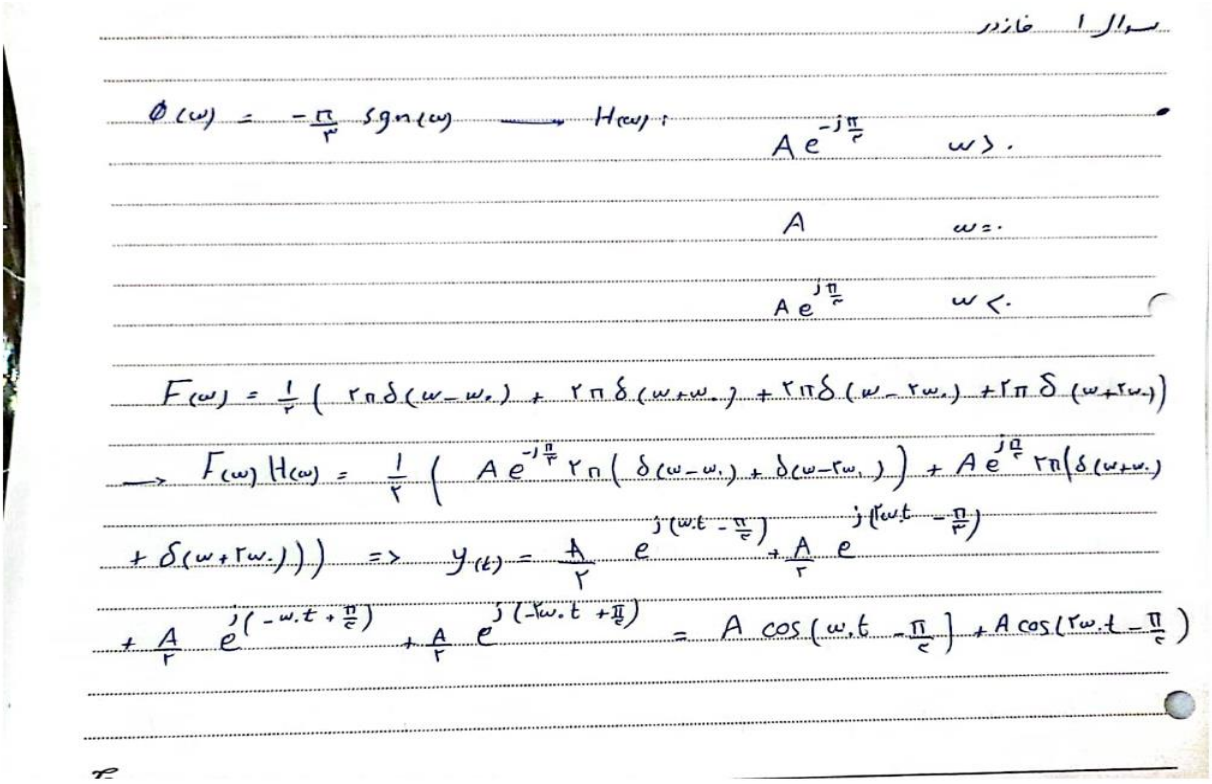
ورودی فیلتر، سیگنال زیر است:

$$x(t) = \cos(\omega_0 t) + \cos(2\omega_0 t)$$

خروجی سیستم را پیدا کنید. سیگنال ورودی از دو سیگنال تک‌فرکانس تشکیل شده بود. هر یک از این دو سیگنال، پس از عبور از این فیلتر چه مقدار در زمان تأخیر پیدا کرده‌اند؟
این بار فرض کنید فاز فیلتر خطّی باشد، یعنی

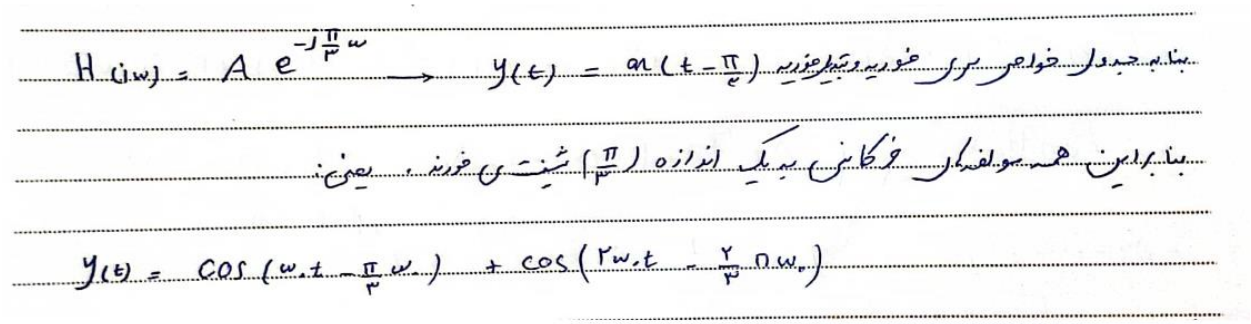
$$\phi(\omega) = -\frac{\pi}{3}\omega$$

پاسخ دو پرسش فوق را برای این حالت نیز بیان کنید. به نظر شما خطّی بودن فاز فیلتر چه اثر مثبتی دارد؟ خطّی نبودن فاز فیلتر چگونه می‌تواند باعث ایجاد اعوجاج در سیگنال خروجی شود؟



همانطور که مشاهده می شود دو سیگنال تک فرکانس دچار شیفت یکسان نشدند

و سیستم فرکانس کم تر را (ω_0)به اندازه بیشتری شیفت می دهد در حالی که فرکانس $2\omega_0$ به مقدار کم تری شیفت می خورد . به عبارتی سیگنال ما دچار اعوجاج می شود .



مشاهده می شود که تمامی مولفه های فرکانسی به یک اندازه دچار شیفت می شوند . این مسئله باعث می شود که بتوان به راحتی اثر شیفت ناشی از فاز سیستم را با اعمال شیفتی به اندازه gd مرتفع ساخت .همینطور سیگنال در خروجی دچار اعوجاج نمی شود . اما همانطور که مشاهده شد در صورت غیر خطی بودن فاز مولفه های فرکانسی غیر یکسان شیفت غیر یکسان دارند که این مسئله باعث ایجاد اعوجاج در سیگنال خروجی می شود و مسئله فاز سیستم را به یک مشکل تبدیل می کند .

گزاره‌ی مهم: تأخیر گروه برای محتوای هر فرکانس، نشان‌گر این است که پس از عبور از فیلتر، چه مقدار تأخیر (بر حسب سمپل یا ثانیه) خواهد یافت. (به عبارت دقیق‌تر، این کمیت در حالت گسسته بدون بُعد است.)

- درستی گزاره‌ی فوق را برای سیگنال ورودی و دو سیستم معرّفی‌شده در پرسش قبل تحقیق کنید.

1.$-\frac{d}{d\omega}\varphi(\omega) = -\frac{2}{3}\pi\delta(\omega)$

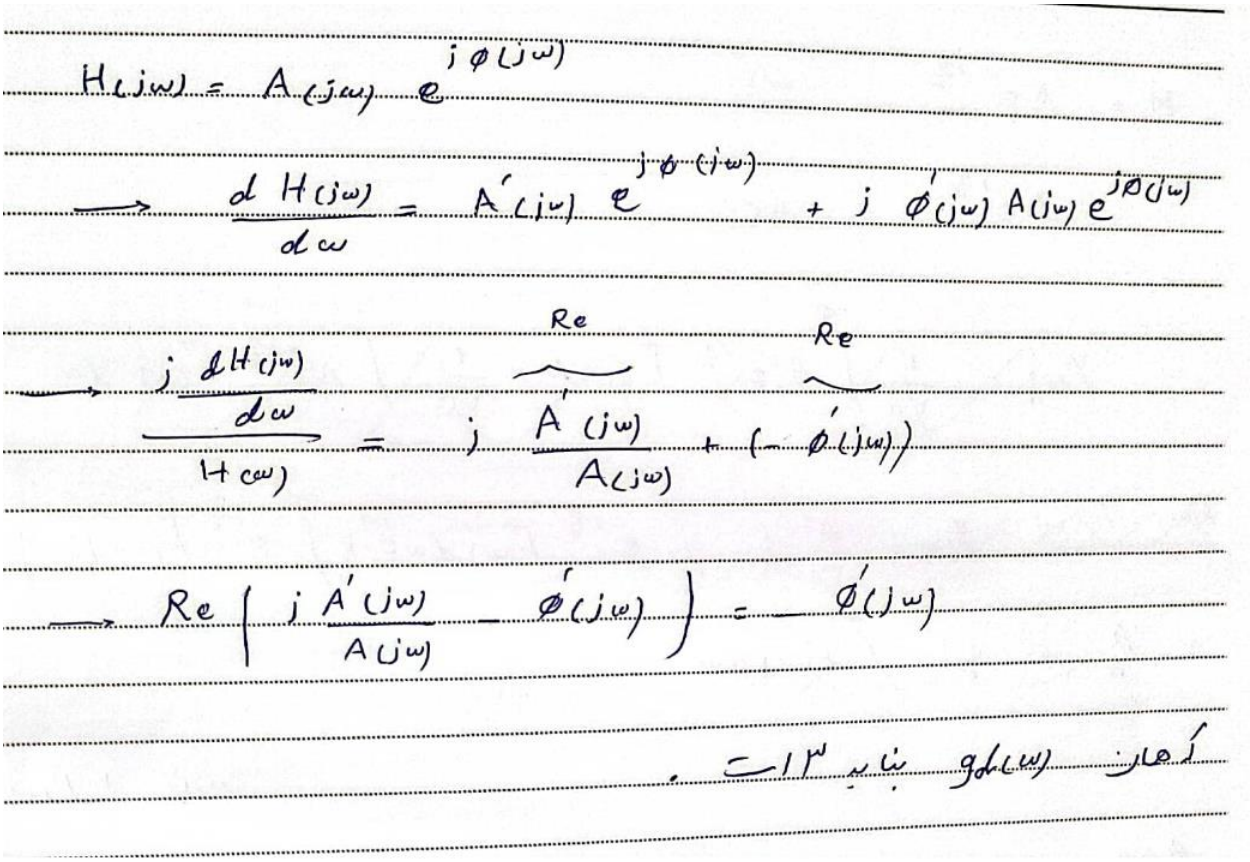
2.$-\frac{d}{d\omega}\varphi(\omega) = \frac{\pi}{3}$

مشاهده می شود که gd برای مورد اول اطلاعات و شهود خاصی درباره اعوجاج ایجاد شده به ما نمی دهد اما مورد دوم به شکل کامل راهنمایی مان می کند تا با استفاده از gd مقدار شیفت لازم برای سیگنال را به دست بیاوریم .این مسئله نشان می دهد که gd برای فاز خطی حاوی اطلاعات مربوط به مقدار شیفت است

و اصلا خود مقدار شیفت را نشان می دهد ولی برای فاز غیر خطی اطلاعات خاصی ندارد .

- با استفاده از روابط ۲ و ۳، ثابت کنید:

$$\mathrm{gd}(\omega) = \mathbf{Re}\left\{\frac{j\frac{d}{d\omega}H(\omega)}{H(\omega)}\right\} \qquad (۴)$$



$$\mathrm{gd}(\omega) = \mathbf{Re}\left\{\frac{j\frac{d}{d\omega}H(\omega)}{H(\omega)}\right\} \qquad (۴)$$

- تابعی بنویسید تا به کمک معادله‌ی ۴، تأخیر گروه را برای هر فیلتر دلخواه محاسبه کند. بدیهی است مجاز به استفاده از توابع آماده‌ی متلب که مستقیماً تأخیر گروه را محاسبه می‌کنند، نیستید. نام‌گذاری تابع شما باید به صورت زیر باشد:

```
groupdelay(h,N)
```

که در آن، متغیر `N` مشخّص می‌کند که DFT مورد استفاده در محاسبه‌ی تأخیر گروه (به کمک تابع `fft`) چند نقطه‌ای باشد. هر چه این عدد بزرگتر باشد دقت محاسبات بالاتر است. (چرا؟)

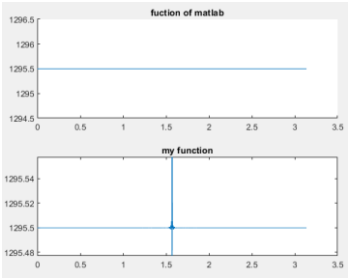
قرار است fft بیانگر سیگنال فرکانسی نمونه برداری شده ما باشد . هرچه شمار نمونه ها بیشتر باشد fft به تبدیل مدنظر نزدیک تر است و حاصل به دست آمده دقیق تر می باشد . در تابع مربوطه به جای مشتق گرفتن از تبدیل فوریه از معادل فرکانسی آن استفاده می کنیم . یعنی یک th(t) تشکیل می دهیم و فوریه می گیریم . در نهایت هم فرمول مربوطه را بازسازی می کنیم .

- با استفاده از تابع بالا، تابعی دیگر بنویسید که برای یک فیلتر و سیگنال داده‌شده، سیگنال را فیلتر کرده و تأخیر ناشی از فیلترینگ را نیز لحاظ کرده و خنثی کند. تابع را به صورت زیر نام‌گذاری کنید:


```
zphasefilter(h,x)
```

که در آن x سیگنال ورودی، و h پاسخ ضربه‌ی فیلتر است. آیا فیلتری علّی و حقیقی وجود دارد که فاز آن در تمام فرکانس‌ها

در ابتدا به بخش آخر می پردازیم . به فیلتری که فاز صفر داشته باشد فیلتر فاز صفر . فیلتر فاز صفر در واقع یک حالت خاص از فیلتر با فاز خطی است . باید توجه کرد که یک فیلتر فاز صفر به هیچ عنوان نمی تواند علی باشد (به جز در موارد بی اهمیت که فیلتر یک ضریب مقیاس ثابت است یعنی فرم دلتا دارد.)باید توجه داشت که در صورت حقیقی بودن فیلتر پاسخ ضربه آن زوج است .



مقایسه groupdelay با تابع متلب

سوال دو

همانطور که گفته شده است سطر دهم نوع تسکی که در زمان مربوطه انجام شده است را نشان می دهد . با توجه به اینکه داده های شماره دو در سطر دهم خود بازه ای از اعداد بین یک تا ۳۶ را دارند طبیعتا نمی توانند . از روش RC استفاده کرده باشند چرا که در این روش شش سطر و شش ستون داریم و نوع تسک روشن شدن یکی از این سطر و ستون ها است پس نمی توان انتظار داشت که در این روش بیش از ۱۲ نوع تسک متفاوت داشته باشیم پس روش آزمایش SC است . به همین روش می توان متوجه شد که داده های شماره سوم از طریق روش RC جمع آوری شده اند چرا که تنها ۱۲ نوع تسک مختلف داریم در حالی که روش SC 36 تسک متفاوت می خواهد.

برای شناسایی روش شماره گذاری خانه های صفحه بخش های تارگت شده متوالی را برداشت می کنیم و آن ها را با کلمه یادگیری انطباق می دهیم . برای نفر اول (داده های دوم) داریم :

10897	10898	10899	10900
42.5625	42.5664	42.5703	42.5742
5.0074	5.7062	4.8590	2.5956
9.2317	4.9116	-3.6681	-7.4006
-1.7883	-7.2233	-8.5707	-3.5205
-1.0588	-4.4325	-5.9914	-4.0923
-7.4731	-6.9116	-3.2172	0.1224
12.9052	5.9082	4.0073	8.2702
6.8985	4.2428	0.6568	1.0995
-20.0376	-10.1724	-0.6527	-2.0975
12	12	12	12
1	1	1	1

عکس اول : ۱۲ باید نشان گر L باشد

78.4844	78.4883	78.4922	78.4961
0.0843	4.2801	1.4023	-4.7005
-2.2297	3.7783	0.8924	-6.1730
-4.3045	-3.7511	-8.7104	-11.6364
-16.5945	-10.1918	-3.2095	-2.5296
-0.1897	-1.6457	-9.7884	-14.7497
0.1998	4.1492	3.0600	-1.1078
9.0076	-0.9354	-0.5526	9.8571
-14.1157	1.2683	7.9882	-0.0609
21	21	21	21
1	1	1	1

عکس دوم : ۲۱ نشان دهنده u است

115.3438	115.3477	115.3516	115.3555
-59.2437	-59.8802	-55.4829	-51.0500
-17.4501	-16.8645	-24.9919	-33.2677
-22.8173	-25.4570	-19.0500	-10.4759
-18.5171	-19.6465	-18.1446	-15.1967
-19.9855	-13.8394	-8.7232	-10.1755
-35.6722	-53.1432	-61.2699	-50.2593
-15.1249	-16.9885	-14.0197	-9.3992
-14.3647	-16.0281	-12.4203	-6.3793
11	11	11	11
1	1	1	1

عکس سوم نشان دهنده k است

160.9531	160.9570	160.9609	160.9648
6.1703	14.9482	14.0273	7.2647
2.2279	-6.0294	-7.6878	-0.8686
-5.3288	-8.1981	-5.7692	-0.4215
-12.2531	-12.0470	-4.5965	1.9373
-12.5115	-9.9685	-2.4433	0.9554
66.2555	68.8749	76.9872	80.1390
-1.4745	0.0694	-3.0312	-4.5446
-9.8713	-6.2903	-10.5418	-17.0982
1	1	1	1
1	1	1	1

عکس سوم نشان دهنده A است

205.8594	205.8633	205.8672	205.8711
15.6875	12.8109	6.4919	3.5221
5.8643	-0.3606	4.2179	13.7789
5.3744	2.9982	1.9758	4.6522
4.4749	-1.8189	-2.9572	1.6466
4.9751	8.5516	11.0904	8.0356
15.7377	16.0091	23.7580	30.7855
-15.8494	-9.2503	3.4037	6.9556
-1.6570	-6.2403	2.3472	12.8201
19	19	19	19
1	1	1	1

عکس ۵ ام : ۱۹ نشانگر s است

برای نفر دوم روش کار به این صورت است که سطر و ستون مربوط به حرف مورد نظر ب فصله زمانی نچندان زیاد روشن می شوند و شناسایی آن به صورت زیر انجام می شود . (به نظر می رسد کلمه یادگیری با Lukas تفاوت داشته و احتمالا lakus است .)

49.8242	49.8281	49.8320	49.8359	49.8398
0.3170	-4.1069	-0.6053	5.9687	7.2445
0.5341	11.0533	11.9137	2.3019	-6.5888
1.4759	7.2670	10.6235	7.5346	1.3971
1.6557	-4.3540	-2.6135	2.5702	4.8247
0.1963	-2.4889	-1.7361	2.0399	4.9322
4.9325	3.4944	5.4660	7.0604	5.1462
-0.6611	-8.4727	-1.9122	11.0495	15.2185
-2.9122	-5.6467	-1.2796	5.5009	7.0292
0	8	8	8	8
0	1	1	1	1

50.9531	50.9570	50.9609	50.9648
-10.8889	-1.9455	3.6310	0.0843
-2.1561	-4.4597	-11.7857	-15.3826
-19.3109	-11.9857	-4.3851	-3.3171
-15.9185	-15.8972	-7.5441	-0.3286
-1.6644	-8.7132	-13.5919	-10.1376
-20.0701	-22.1984	-16.5617	-8.1434
-2.3051	-10.0702	-16.5038	-13.2209
-11.9428	-12.8576	-6.4848	-0.0504
6	6	6	6
1	1	1	1

نمایش سطر L و ستون L

87.7656	87.7695	87.7734	87.7773
1.9788	2.0586	3.1113	3.9069
9.9095	10.5955	-1.1978	-11.1248
2.7896	7.1799	0.5681	-10.6159
-15.6773	-16.4767	-7.7196	1.6074
-14.4330	-14.5900	-3.9529	6.8253
5.8088	-0.3549	-9.0391	-10.8298
-7.3810	-16.8052	-10.7338	3.5858
-1.9473	6.9401	3.8857	-7.5278
10	10	10	10
1	1	1	1

88.1406	88.1445	88.1484	88.1523
5.3405	6.8001	14.1551	18.6278
14.3067	9.4277	7.5319	10.1454
17.2542	22.3002	12.2742	-1.8858
6.5025	2.9941	7.1693	12.4938
16.4021	6.7524	-0.2642	3.8974
15.4382	6.2858	1.3617	4.7339
17.7251	11.1656	-2.7187	-8.3063
4.3082	3.1231	2.1778	3.0818
3	3	3	3
1	1	1	1

نمایش سطر و ستون u

32085	32086	32087	32088
125.3281	125.3320	125.3359	125.3398
91.7246	97.1207	83.4019	67.4895
24.8298	25.4197	26.2173	25.3520
13.1503	11.2865	11.9265	13.2028
14.9213	23.4113	17.8855	4.4339
21.6431	10.4487	1.3397	4.6695
4.8330	7.9319	5.2046	-0.9800
8.1611	-3.3193	-8.6039	-1.8520
12.5101	10.0111	3.0472	0.6150
5	5	5	5
1	1	1	1

127.7656	127.7695	127.7734	127.7773
2.1435	-3.3193	-14.5498	-17.8351
0.3149	-0.5890	-13.5740	-22.5141
-16.8247	-10.0723	-3.4846	-2.9932
-17.2766	-19.3789	-11.9376	-3.0229
-12.2439	-18.9540	-18.6453	-10.7968
-2.8457	7.5979	-0.0048	-14.6309
-5.6378	-6.8145	-11.3423	-12.3769
-10.2425	-9.4616	-10.8245	-11.9944
8	8	8	8
1	1	1	1

سطر و ستون k

170.9531	170.9570	170.9609	170.9648
5.7705	0.4526	5.8608	17.2150
4.7765	6.8705	6.4448	4.0236
5.5948	5.2167	-2.4667	-7.8858
-0.0477	3.3308	6.2447	5.8750
-2.6066	0.3757	4.1451	5.5046
-13.9234	-7.5613	4.0215	9.5740
-12.1239	-10.7620	-1.3604	7.1568
0.6011	0.5829	-2.8573	-4.5743
7	7	7	7
1	1	1	1

43717	43718	43719	43720
170.7656	170.7695	170.7734	170.7773
1.5567	1.3153	-4.1949	-8.8092
-12.6814	-6.7454	-0.9263	-1.3138
-8.8774	0.9989	3.7940	-2.9544
-2.6403	2.9341	-2.1895	-11.5442
-5.8194	-4.4474	-2.2257	-1.9937
-5.0711	3.3564	8.0269	2.9036
3.1905	8.9530	5.7360	-5.9790
3.0783	0.7260	-1.1680	-1.2536
1	1	1	1
1	1	1	1

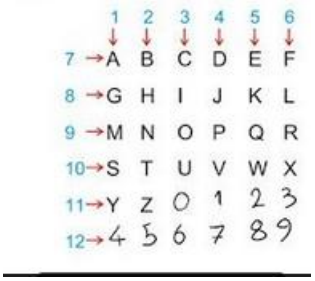
سطر و ستون A

51941	51942	51943	51944
202.8900	202.8943	202.8984	202.9023
9.3160	12.4248	10.1424	3.5057
12.2060	5.6834	4.1749	7.9925
14.5492	6.1525	-0.4815	1.4988
3.5263	10.0097	15.6536	13.2984
13.5513	14.7051	7.1830	-0.9487
-4.6250	-6.9766	0.9562	9.2498
2.9324	-6.0023	0.8878	14.4677
18.8631	10.4920	-0.0903	-1.5940
10	10	10	10
1	1	1	1

205.0781	205.0820	205.0859	205.0898
-1.6688	-0.4750	1.4152	1.7993
-9.2713	-16.0606	-8.3763	4.7310
-4.8191	-2.4639	-0.5618	-1.2961
-5.6218	-13.4699	-8.4789	4.3547
-8.3395	-10.9360	-7.8074	-2.4027
-2.0232	-2.1718	-8.0203	-12.4366
0.8257	-1.6361	-6.4283	-8.8915
-12.1945	-15.0536	-6.6424	3.0445
1	1	1	1
1	1	1	1

سطر و ستون S

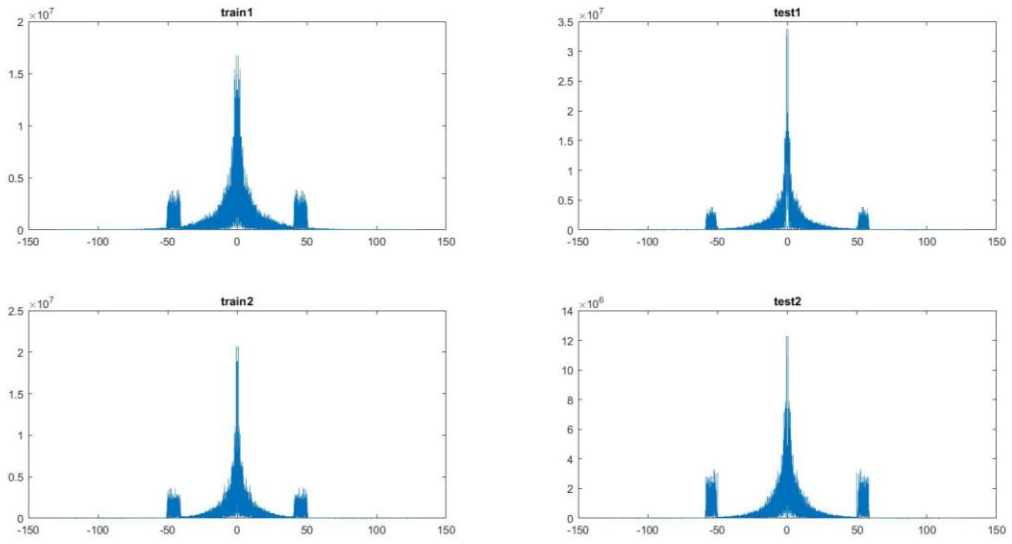
بنابراین شماره گذاری آزمایش های متد rc,sc به ترتیب به شکل زیر است :



در نهایت به بخش مربوط به زدن تابع می رسیم .در ابتدا دو عدد استراکت تعیین می کنیم که هر یک قرار است داده های یک آزمایش را ساپورت کند .هر یک از این استراکت ها یک فیلد trainو test و یک فیلد method دارند که نوع آزمایش را تعیین می کند . در بخش بعدی به استفاده از تابع می رسیم . تابع به این صورت طراحی شده که استراکت های اولیه را می گیرد و به آن ها یک فیلد

traintime و testtime اضافه می کند . ماتریس های تایم برای ترین ها دارای سه ستون هستند که هر یک منحصر به تایم تحریک ها نوع تحریک و تارگت یا نات تارگت بودن است .ماتریس تایم برای داده های تست بخش تارگت را ندارد (بنا به ماهیت داده های تست).

در بخش آخر این سوال باید ابتدا داده ها را فیلتر کنیم. برای اینکار ابتدا با استفاده از SubjectData2,3، داده های مربوط به struct2,3 را لود میکنیم. فرکانس نمونه برداری 256.41 Hz است. برای هریک از ۴ داده (ترین و تست آزمایش اول، و دوم)، نمودار طیف فرکانسی را با استفاده از plotfft که در فاز یک نوشته بودیم، رسم میکنیم (البته برای جلوگیری از کاهش سرعت اجرای کد، این بخش کامنت شده که با uncomment کردن آن، نمودارهای زیر رسم میشوند):



با دقت در نمودار ها میتوان فرکانس ماکزیممی که اطلاعات اصلی سیگنال تا آنجاست و سپس، بیشتر نویز داریم، را پیدا کرد. این فرکانس برای داده های ترین در حدود 50.5Hz و برای داده های تست در حدود 58.6Hz است. بدین ترتیب دو فیلتر برای

داده های ترین و تست، به ترتیب به نام های train1BPfilt و test1BPfilt طراحی شده اند که فرکانس قطع اولشان در حدود ۱ هرتز (برای حذف اثر DC در سیگنال ها) و فرکانس قطع دومشان مطابق توضیحات بالاست. این دو فیلتر در test1Filt,train1Filt ذخیره میشوند. حال train1,2 و test1,2 را تعریف میکنیم و توسط یک for، برای تمام کانالها، داده های struct2,3 را فیلتر میکنیم و مقادیر جدید را در این ۴ ماتریس میریزیم. حال اطلاعات struct2,3 را توسط این ۴ ماتریس، آپدیت و فیلتر میکنیم.

در ادامه میخواهیم شش پنجره بندی انجام دهیم. بدین صورت که در هر آزمایش، یک ماتریس ایپاک برای داده های تست، و دو ماتریس برای ترِین (target,non target) میسازیم. توضیحات هر متغیر تا حد خوبی در قالب کامنت نوشته شده است. در S1، ماتریس traintime شامل سه ستون است. ستون اول زمان تحریک ها، ستون دوم عدد متناظر با تحریک و ستون سوم، بیانگر target یا non-target بودن تحریک است (یک یا صفر). از ستون سوم میتوان تعداد تحریک های هدف را با nnz بدست آورد و مابقی نیز تحریک های غیر هدف اند. اندیس تحریک های هدف و غیرهدف را به ترتیب در targs1 و nontargs1 میریزیم. حال باید زمان این تحریک ها را پیدا کنیم. برای افزایش سرعت پردازش، تنها از یک فور استفاده میکنیم اما یک فرض کاملا منطقی میکنیم که تعداد تحریک های غیر هدف، بیشتر از تحریک های هدف باشد. این حلقه، به تعداد تحریک های غیر هدف تکرار میشود و زمان های متناظر با این تحریک ها از ستون ۱ پیدا میشوند. ضمنا تا زمانی که i از تعداد تحریک های هدف بیشتر نشده، همین کار را برای این تحریک ها نیز انجام میدهیم. بدین ترتیب زمان

تحریک ها در دو بردار targTimes1 و nontargTimes2 ریخته میشود. سپس با استفاده از ismember و find، اندیس زمان تحریک های target و non-target را پیدا کرده و در nontargTimeInd1 و targTimeInd1 میریزیم. تمام این فرایندها را برای آزمایش دوم و تحریک های train، تکرار میکنیم و بردار های nontargTimeInd2 و targTimeInd2 تشکیل میشود. حال به سراغ ساخت ماتریس epoch میرویم.

مشابه سوال ۳ فاز قبل، زمان قبل و بعد هر تحریک که ثبت میشود را ۲۰۰ و ۸۰۰ میلی ثانیه میگیریم. سپس تعداد خانه هایی که باید از قبل تحریک، و بعد از تحریک ثبت شوند را در backIndex و forIndex میریزیم. تعداد کل خانه ها نیز در trialLength ریخته میشود. چون اکنون در train، اندیس تمام تحریک ها را داریم، نیاز به اجرای کل فرایند تابع epoching نیست. بنابراین یک تابع epoching2 تعریف میکنیم که ماتریس کانال ها، اندیس تحریک ها، تعداد تحریک ها و تعداد خانه قبل و بعد هر تحریک را میگیرد و همان فرایند تابع epoching را انجام میدهد، با این تفاوت که نیازی به پیدا کردن اندیس تحریک ها نیست. برای ماتریس train، ماتریس target و non-target را با همین تابع میسازیم. برای test اما نیازی به اینکار نیست چرا که داده ها دو بخش نیستند و کافیست با استفاده از ماتریس test1 و اطلاعات struct2، و تابع epoching، ماتریس را بسازیم. همینکار را برای آزمایش دوم نیز انجام میدهیم و در نهایت شش ماتریس زیر، ماتریس های ایپاک هستند:

z1_train_nontarg, z1_train_targ, z1_test, z2_train_nontarg, z2_train_targ, z2_test

حال ایپاک شده داده های ترین تارگت ترین نات تارگت و تست هر دو آزمایش را به شکل تفکیک شده داریم . همانطور که گفته شده از همه داده های ترین برای learn استفاده نمی کنیم برای همین لازم است تعدادی خانه (معادل با هشتاد درصد داده ها) را به شکل رندم جدا کنیم . این کار را با استفاده از randperm انجام می دهیم . حاصل رندپرم را sort می کنیم و حاصل را به عنوان خانه هایی در نظر می گیریم که در عملیات تعیین ویژگی نقش دارند. این کار را برای هر شش ماتریس انجام می دهیم و داده هارا در ساختار های ماتریسی به نام testMat ذخیره می کنیم . حال این ماتریس هارا به تابعی می دهیم که قرار است برای ما ماتریس ویژگی و لیست ویژگی ها را طراحی کند . این تابع کمی پیچیده است ☺ . اما عملکرد آن به شکل کلی توضیح داده می شود. این تابع یک بردار ویژگی و ماتریش ویژگی تولید می کند . بردار ویژگی شامل نام ویژگی هاست که قرار است در نهایت به کارمان بیاید و ماتریس ویژگی همان خواسته سوال است . ویژگی ها به شکل کلی شامل corr و داده های آماری تک تک چنل ها و داده های آماری مجموعه چنل ها و داده های چند بخش شده ترایال ها (یعنی برای مثال بازه زمانی به پنج بخش تقسیم شده است .)

به شکل کلی به دست آوردن ویژگی های تجمیعی ترایال ها مثل میانگین همه چنل ها یا واریانس همه چنل ها چندان کاری ندارد و با توجه به توابع مورد استفاده مشابه به دست آوردن اطلاعات آماری تک تک چنل هاست . همینطور برای بخش فرکانسی اطلاعات از آن ها در راستای چنل ها fft می گیریم . برای corr ما بین چنل ها هم به این صورت رفتار می کنیم که که بر روی تک تک ترایال ها فور میزنیم سپس با استفاده از corr همبستگی بین چنل ها را به دست می آوریم فقط باید توجه کنیم که چنل ها باید در سطر قرار بگیرند نه ستون برای همین transpose می کنیم . در نهایت ۲۸ داده مهم ماتریس خروجی را (که نوعی جدول ضرب corr ها است و ۲۸ داده غیر تکراری آن داده های بالای قطر اصلی آن است) استخراجی میکنیم و به شکل پشت سر هم قرار داده در سطر های c می گذاریم . حال اگر c را در کنار ماتریس ویژگی بگذاریم ماتریس ویژگی جدید ما به دست می آید . در نهایت باید توجه کرد که اسم ویژگی ها را در خارج از corr به درستی وارد کنیم .

در نهایت به بررسی ویژگی ها می پردازیم برای این کار لازم است anova ماتریس ویژگی را حساب کنیم و توضیحات تابع anova در زیر آمده است .

تابع anov دو ماتریس targ و nontarg را که جدول ویژگی هایشان است، میگیرد و مقدار J را مطابق رابطه مذکور در سوال، برای هر ویژگی برمیگرداند. تعداد ویژگی ها در characs و تعداد ترایال ماتریس های targ,nontarg در trials1,2 ریخته میشود. حال کمیت ها را مطابق فرمول سوال، به صورت برداری و برای تمام ویژگی ها، پیدا میکنیم. u0 میانگین ستون به ستون تمام داده های targ,nontarg با هم را حساب میکند. u1,u2 نیز میانگین ستون به ستونِ فقط targ و فقط nontarg اند. Var1,var2 نیز همان واریانس ستون به ستون ماتریس های targ,nontarg اند. در نهایت با استفاده از فرمول مذکور، مقدار J را به صورت برداری و برای تمام ویژگی ها بدست می آوریم.

فرض کنید با استفاده از نقاط روی صفحه می خواهیم یک تابع را تخمین بزنیم اگر این تابع را به نوعی انتخاب کنیم که از همه نقاط بگذرد احتمالا درجه بالایی دارد که این باعث می شود انعطاف تابع کم شود . به عبارت دیگر اگر نقاط دیگری در جاهای دیگر صفحه باشند احتمال اینکه تابع از آن ها بگذرد یا حتی به آنها نزدیک شود بسیار کم است در حالی که اگر تابع را از درجه کمتر تخمین میزنیم و راضی می شدیم که به بعضی از داده ها صرفا نزدیک شود این امکان وجود داشت که در صورت انتخاب صحیح تابع به نقاط دیگر صفحه که احتمالا از قاعده ای پیروی می کنند نزدیک تر باشد . overfit دقیقا چنین چیزی است . یعنی ما آنقدر در در نظر گرفتن ویژگی ها زیاده روی کردیم که بعضی از نویز ها را هم به عنوان ویژگی در نظر گرفته ایم . به عبارتی در این صورت ساختار نهایی ما به جای اینکه یاد بگیرد حفظ می کند .