



دانشگاه اصفهان

درس سیگنال‌ها و سیستم‌ها

عنوان:

گزارش پروژه نهایی

استاد: دکتر باطنی

اعضای گروه به ترتیب الفبا:

مهدی رهبر

میعاد کیمیاگری

امیر طاهانجف

زمستان ۱۴۰۳

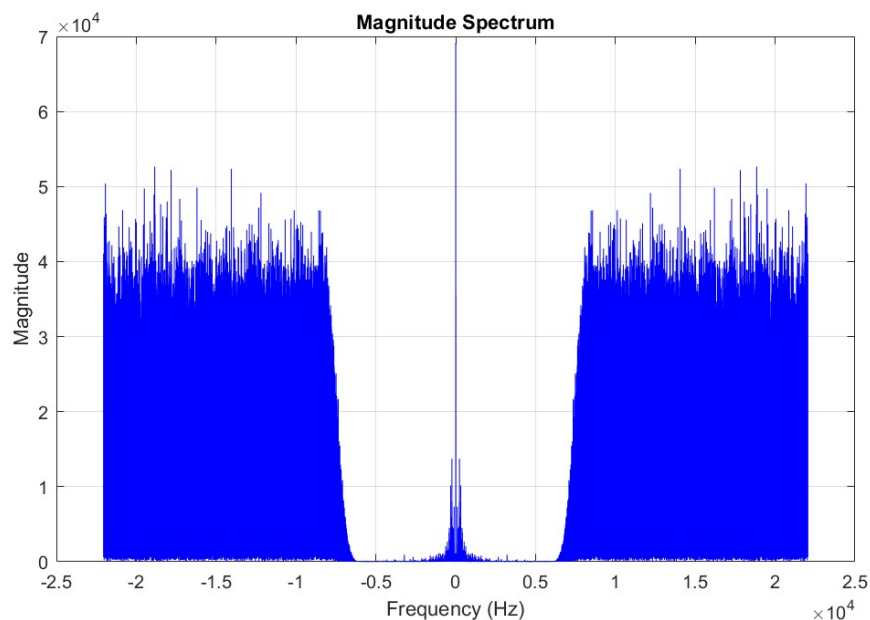
پروژه اول

الف) فایل Sound.mat را به کمک دستور load در MATLAB بارگذاری کنید. سیگنال صوتی ذخیره شده در این فایل در چه فرکانسی نمونه برداری شده؟ تعداد نمونه های این سیگنال چقدر است؟ به کمک دستور sound به این فایل گوش دهید. چه می شنوید؟

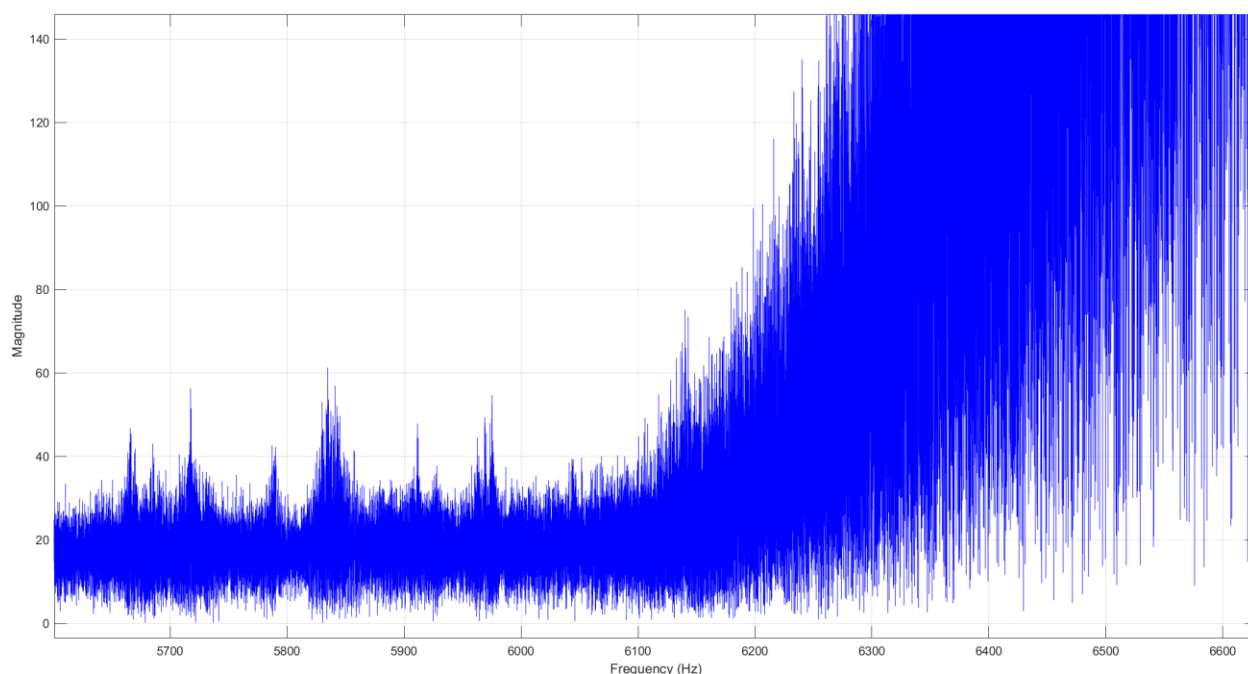
پس از بارگذاری کردن صدا، مقدار فرکانس نمونه برداری را 44100 هرتز به دست آوردیم و تعداد نمونه های این سیگنال 2000000 بود. با دستور sound که به آن گوش دادیم صدا کامل دارای نویز بود و اصلاً هیچ چیز قابل تشخیصی نداشت.

ب) فایل صوتی داده شده، حاوی یک سیگنال پایین گذر موزیک و یک نویز اضافه شده بالاگذر است. به کمک دستورات fft، fftshift و abs و plot نمودار شکل طیف این فایل را رسم کنید. از روی شکل تعیین کنید هر یک از سیگنال صوتی و نویز در چه محدوده فرکانسی (بر حسب هرتز) گسترده اند.

با استفاده از دستور fft تبدیل فوریه سیگنال حساب کردیم و با دستور fftshift نمودار را شیفت دادیم تا صفر فرکانس در وسط نمودار قرار بگیرد و از abs هم برای محاسبه بزرگی دامنه استفاده کردیم. با توجه به نمودار FFT مشخص شده در تصویر ۱ و ۲، سیگنال موزیک در فرکانس های حدود 6100 هرتز و کمتر قرار دارد و سیگنال نویز هم در فرکانس های حدود 6300 هرتز به بالا قابل مشاهده هست.



تصویر ۱- طیف سیگنال صوتی



تصویر ۲- طیف سیگنال صوتی (بزرگ شده)

چ می‌خواهیم به کمک یک فیلتر پایین گذر مناسب سیگنال صوتی اصلی را استخراج و آن را از وجود نویز پاک کنیم. برای این کار، فرکانس قطع فیلتر لازم بر حسب هرتز (معادل آنالوگ) و فرکانس دیجیتال معادل آن را (در DTFT و FFT) چقدر است؟

چون سیگنال صوتی در بازه ۰ تا ۶ کیلوهرتز قرار دارد، مقدار فرکانس قطع آنالوگ باید بیشتر از ۶ کیلوهرتز انتخاب شود. مقدار ۶۱۰۰ هرتز را طبق تصویر ۲ (به طور تقریبی) به عنوان فرکانس قطع فیلتر در نظر گرفتیم. مقدار فرکانس دیجیتال متناظر در DTFT به شکل زیر محاسبه می‌شود.

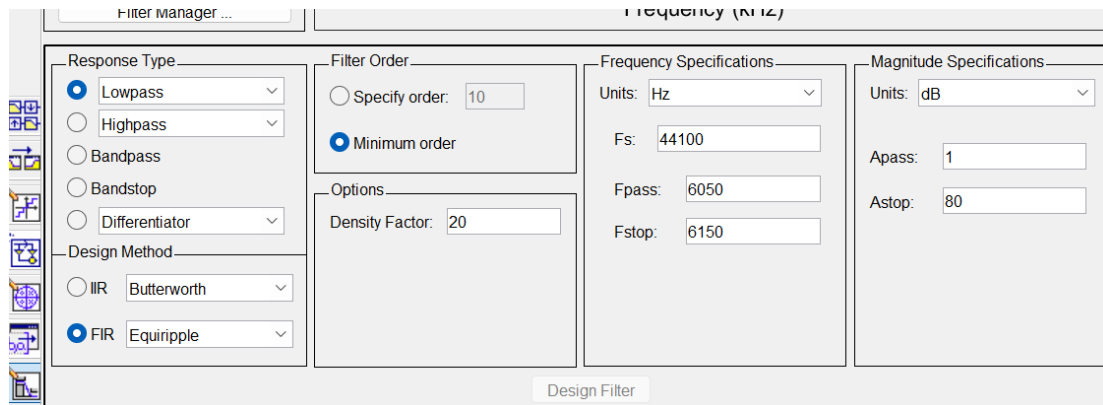
$$\omega_c = 2\pi \frac{f_c}{F_s} = 2\pi \frac{6100}{44100} \approx 0.869 \text{ rad/sample}$$

و برای محاسبه فرکانس دیجیتال متناظر در FFT داریم:

$$f_{\text{norm}} = \frac{f_c}{F_s} = \frac{6100}{44100} \approx 0.138$$

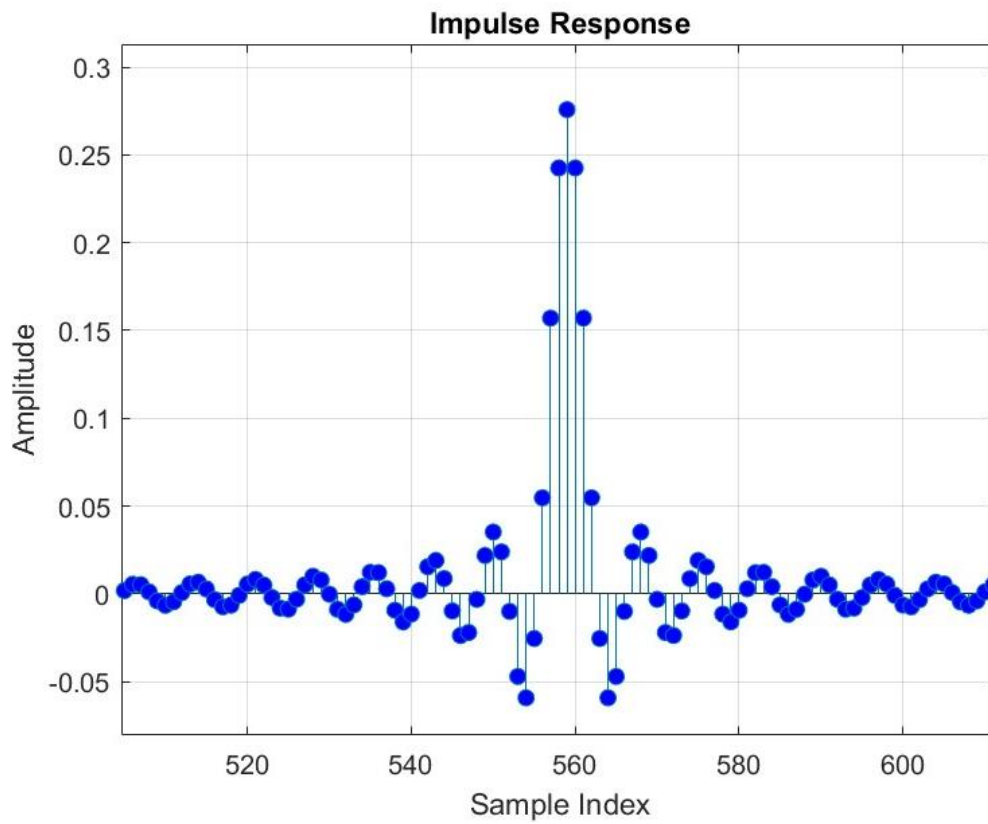
(د) در این بخش، می‌خواهیم فیلتر انتخاب‌شده بند قبل را پیاده‌سازی کنیم. ابزار fdatool در MATLAB باز کنید. به کمک این ابزار یک فیلتر Lowpass با فرکانس نمونه‌برداری برابر با فرکانس نمونه‌برداری سیگنال طراحی کنید. بر اساس پاسخ‌تان در بند قبل، پارامترهای Fpass و Fstop را جوری تعیین کنید که فیلتر فرکانس‌های سیگنال صوتی را به خوبی عبور داده و نویز را به خوبی حذف کند (بقیه پارامترها را در مقدار default قرار دهید). ضرایب فیلتر طراحی‌شده را به کمک گزینه Export از ابزار File در MATLAB منتقل و در پارامتر Num ذخیره کنید.

طبق قسمت الف می‌دانیم که فرکانس نمونه‌برداری سیگنال برابر با ۴۴۱۰۰ هرتز است. همچنین می‌دانیم که فرکانس قطع فیلتر که در قسمت ج، ۶۱۰۰ هرتز به دست آوردیم عموماً معادل میانگین Fstop و Fpass در نظر گرفته می‌شود. با توجه به این نکته و همچنین با کمی آزمایش و خطا، مقدار فرکانس عبور (Fpass) را ۶۰۵۰ هرتز و مقدار فرکانس توقف (Fstop) را ۶۱۵۰ هرتز در نظر گرفتیم. میانگین این دو پارامتر، برابر با ۶۱۰۰ هرتز است که همان فرکانس قطع می‌باشد. این تنظیمات در تصویر ۳ قابل مشاهده است.



تصویر ۳- تنظیمات fdatool

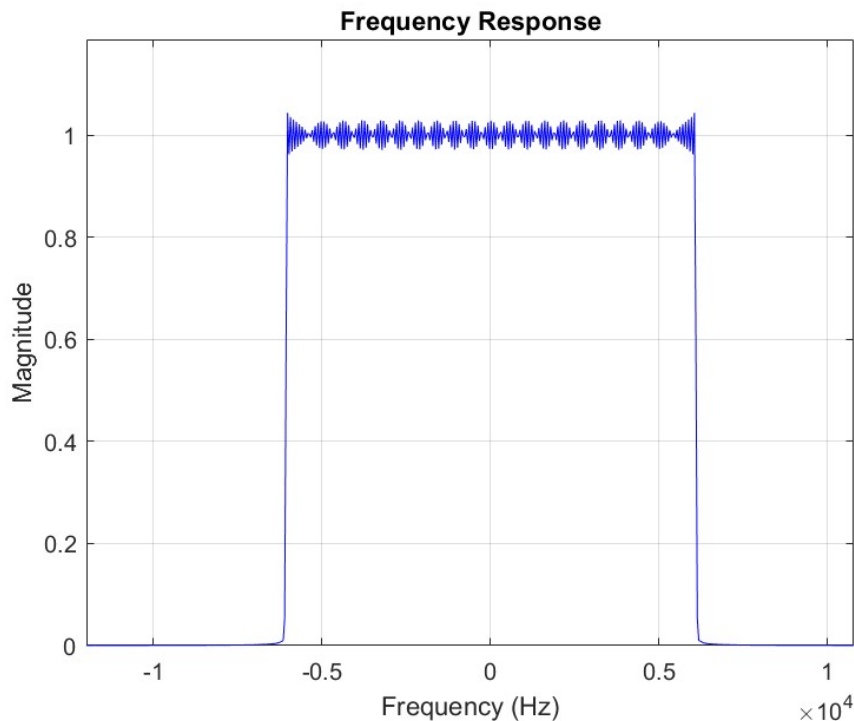
ه) شکل پاسخ ضربه این فیلتر را تعیین کنید و به کمک نمودار stem رسم و ضمیمه گزارش کنید.
پاسخ ضربه در تصویر ۴ قابل مشاهده است.



تصویر ۴- پاسخ ضربه فیلتر

و) به کمک دستورات `fft`، `fftshift`، `abs` و `plot` نمودار شکل پاسخ فرکانسی این فیلتر را رسم کنید. تعیین کنید فرکانس قطع فیلتر در حوزه دیجیتال (Ω) و در حوزه آنالوگ (f) چه مقداری دارد. نتایج را با مقادیر طراحی خود مقایسه و ضمیمه گزارش کنید.

پاسخ فرکانسی فیلتر در تصویر ۵ قابل مشاهده است.



تصویر ۵- نمودار پاسخ فرکانسی فیلتر

معمولا در عمل مقدار فرکانس قطع عملی فیلتر را جایی در نظر می‌گیریم که اندازه پاسخ فرکانسی (قابل مشاهده در تصویر ۵) نصف مقدار ماکزیمم آن شود. برای محاسبه دقیق این نقطه از تکه کد زیر در متلب استفاده می‌کنیم.

```
% findin cutoff freq
cutoff_thresh = max(H_magnitude) / 2;
idx_positive = find(f_filter >= 0);
positive_H = H_magnitude(idx_positive);
positive_f = f_filter(idx_positive);
cutoff_idx = find(positive_H <= cutoff_thresh, 1, 'first');
cutoff_freq = positive_f(cutoff_idx);
disp(['cutoff frequency: ', num2str(cutoff_freq), ' Hz']);
```

خروجی کد مقدار فرکانس قطع فیلتر را 6142.9619 هرتز نشان می‌دهد. طبق فرمول زیر فرکانس قطع عملی فیلتر را در حوزه دیجیتال به دست می‌آوریم.

$$\omega_c = 2\pi \frac{f_c}{F_s} = 2\pi \frac{6142.9619}{44100} \approx 0.875 \text{ rad/sample}$$

فرکانس قطع فیلتر را در مرحله طراحی (قسمت ج) ۶۱۰۰ هرتز در نظر گرفته بودیم ولی مقدار واقعی آن تقریباً ۶۱۴۲ هرتز شد. یعنی حدود ۴۲ هرتز اختلاف بین مقدار واقعی و طراحی فرکانس قطع وجود دارد. مقایسه نتایج در جدول زیر قابل مشاهده است.

مقدار طراحی	مقدار عملی	
6100.000 Hz	6142.961 Hz	فرکانس قطع آنالوگ
0.869 rad/sample	0.875 rad/sample	فرکانس قطع دیجیتال

جدول 1- مقایسه مقدارهای عملی و طراحی فرکانس قطع برای فیلتر پایین‌گذر

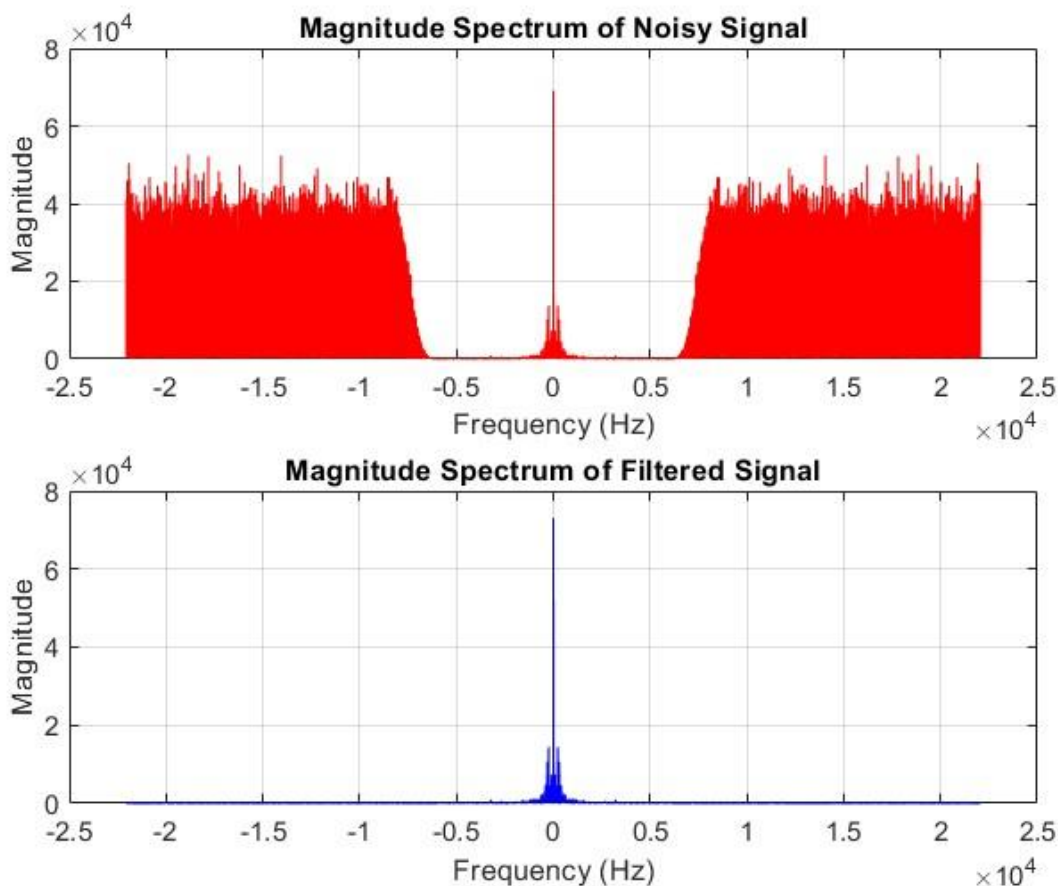
ز) به کمک دستور conv سیگنال صوتی ورودی را از فیلتر طراحی شده عبور داده (کانولوشن سیگنال و پاسخ فیلتر طراحی شده را به دست آورید). سیگنال حاصل را به نام Filtered_Sound ذخیره نمایید. به سیگنال حاصل به کمک دستور sound گوش کنید. چه می‌شنوید؟
موزیک به صورت واضح قابل شنیدن است.

```
% ز
Filtered_Sound = conv(y, Num, 'same');
sound(Filtered_Sound, Fs);
```

ح) به کمک دستور audiowrite سیگنال صوتی به‌دست‌آمده را در فایل به‌نام Filtered_Sound.wav ذخیره و ضمیمه گزارش کنید.

```
% ح
audiowrite("Resources\Filtered_Sound.wav", Filtered_Sound, Fs);
```

ط) به کمک دستورهایی `abs`، `fft`، `fftshift` و `plot` و نیز با استفاده از فرمان `Subplot`، طیف سیگنال‌های متناظر با سیگنال‌های `noisy_sound` و `filtered_sound` (یعنی سیگنال صوتی پیش و پس از اعمال فیلتر پایین گذر) را در زیر هم رسم و مقایسه کنید. ضمن مقایسه این نتایج، نتیجه شنیداری مشاهده شده در بند (د) را توجیه کنید.



تصویر 7- طیف سیگنال، پیش و پس از اعمال فیلتر

در این بخش، ابتدا سیگنال‌های `Noisy_Sound` و `Filtered_Sound` را با استفاده از `FFT` و `fftshift` به حوزه فرکانس منتقل کردیم و سپس طیف قدرمطلق هر دو سیگنال را محاسبه نمودیم. با استفاده از دستور `subplot`، طیف فرکانسی سیگنال نویزی و سیگنال فیلتر شده را که در تصویر ۲ قابل مشاهده است، رسم کردیم.

در نمودار اول، مشاهده کردیم که سیگنال نویزی دارای باند فرکانسی وسیع با نویزهای فرکانس بالا است. پس از اعمال فیلتر پایین‌گذر، در نمودار دوم، انرژی فرکانس‌های بالا به‌طور قابل‌توجهی کاهش یافت. همچنین، نتیجه شنیداری هم نشان می‌دهد که نویزهای فرکانس بالا حذف شده و صدای فیلتر شده شفاف‌تر و واضح‌تر به گوش می‌رسد.

ی) فایل صوتی نهایی (پاک‌سازی‌شده) شامل چه محتوایی است؟ به کمک جست‌وجوی اینترنتی یا دیگر ابزارهای آن عنوان آن را بیابید.

برای شناسایی موزیک از ابزار جست‌وجوی موسیقی Shazam استفاده نمودیم. بر اساس نتایج جست‌وجو، مشخص شد که این موزیک *Romeo & Juliet* از *Richard Clayderman* است.

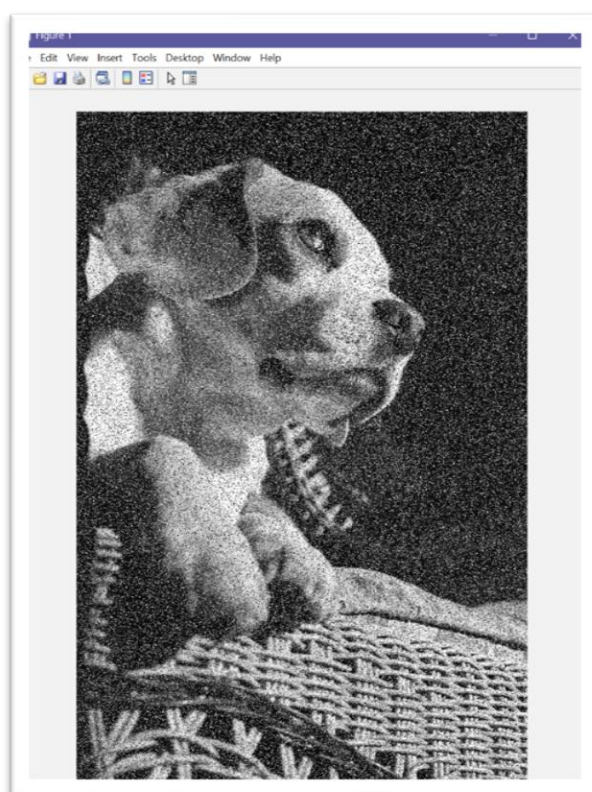
پروژه دوم

الف) فایل تصویری Noisy_Pic.png را به کمک دستور `imread` در محیط MATLAB خوانده و سیگنال متناظر با آن را ماتریس `I` بریزید. (اگر تصویر به صورت رنگی ذخیره شده است (شامل سه ماتریس است)، با استفاده از دستور `rgb2gray` آن را به صورت سیاه و سفید درآورید تا در یک ماتریس دوبعدی نشان داده شود).

پس از خواندن عکس، با چاپ کردن تعداد کانال آن (با دستور `size`) از این که عکس سیاه و سفید (تک کاناله) است اطمینان یافتیم.

```
I = imread("resources/Noisy_Pic.png");  
[~, ~, channels] = size(I);  
disp(['N_Channels:', num2str(channels)]);
```

ب) به کمک دستور `imshow` تصاویر متناظر با فایل داده شده را مشاهده و تصویر آن را در گزارش ضمیمه کنید.



تصویر ۷- عکس خوانده شده با `imread`

ج) تصویر بند قبل نمونه‌ای از تصویر آمیخته شده با نویز فلفل‌نمکی است که در پردازش تصویر شناخته شده است. در مورد این نویز، که ماهیت آن و این در چه کاربردهایی متداول و مشکل‌ساز است، تحقیق کرده و نتیجه را شرح دهید.

نویز فلفل‌نمکی به صورت نقاط سیاه و سفید تصادفی در تصویر ظاهر می‌شود. به صورتی که انگار نمک و فلفل داخل تصویر قرار داده شده است. این نوع نویز معمولا به دلیل خطاهای سنسور دوربین یا تداخل در حین انتقال داده‌ها یا نقص‌های سنسورهای تصویربرداری ایجاد می‌شود. این نویز، پیکسل‌های تصویر را به دو مقدار (۰ یا ۲۵۵) در تصاویر 8 بیتی تبدیل میکند. این پیکسل‌ها به شکل نقاط سفید (نمک) یا سیاه (فلفل) ظاهر می‌شوند و با مقادیر واقعی تصویر اصلی هیچ ارتباطی ندارند (به اصطلاح Outlier یا نقاط پرت هستند). این نقطه‌ها ممکن است به صورت تک پیکسل باشند یا به صورت چند پیکسل در کنار هم ظاهر شوند.

نویز نمک و فلفل در بسیاری از کاربردهای پردازش تصویر، از جمله موارد زیر مشکل‌ساز است:

- تصاویر پزشکی مثل MRI یا سیتی‌اسکن که وجود این نویز باعث می‌شود در تشخیص بیماری‌ها مشکل به وجود بیاید.
- دوربین‌های امنیتی، به این دلیل که در تصاویر تار و یا بی کیفیت، صورت افراد یا پلاک ماشین‌ها واضح نیست.

د) برای حذف نویز فلفل‌نمکی معمولا از فیلتر میانه (Median) بر روی تصاویر استفاده می‌شود. با جست‌وجو در اینترنت تحقیق کنید چرا چنین است. انتظار می‌رود دلیل موثر بودن این فیلتر برای حذف این نوع نویز را درک کرده و توضیح دهید (مثلا اشاره کنید که چرا این فیلتر برای این کاربرد بر فیلتر میانگین (Mean) برتری دارد. در این مرحله استفاده از منابع اینترنتی و ابزارهای هوش مصنوعی مجاز و سودمند است، اما در نهایت به توضیح مناسب موضوع توسط شما نمره تعلق می‌گیرد).

در فیلتر میانه، مقدار هر پیکسل با میانه مقادیر همسایه‌هایش جایگزین می‌شود، در صورتی که در فیلتر میانگین، مقدار هر پیکسل با میانگین مقادیر همسایه‌هایش جایگزین می‌شود.

یکی از علت‌های برتری فیلتر میانه نسبت به میانگین این است که داده‌های پرت در فیلتر میانه به حساب نمی‌آیند؛ چون این فیلتر ابتدا داده‌ها را مرتب می‌کند و سپس مقدار وسط را انتخاب می‌کند که همین باعث می‌شود برای نویز فلفل‌نمکی مقادیر غیر عادی مثل (۰ یا ۲۵۵) را نادیده بگیرد، ولی در فیلتر میانگین همه داده‌ها از جمله داده‌های پرت در میانگین تاثیر می‌گذارند، پس مقدار خطا افزایش پیدا می‌کند. فیلتر میانگین ممکن است لبه‌های تصویر را محو کند و جزئیات را از بین ببرد، در صورتی که فیلتر میانه

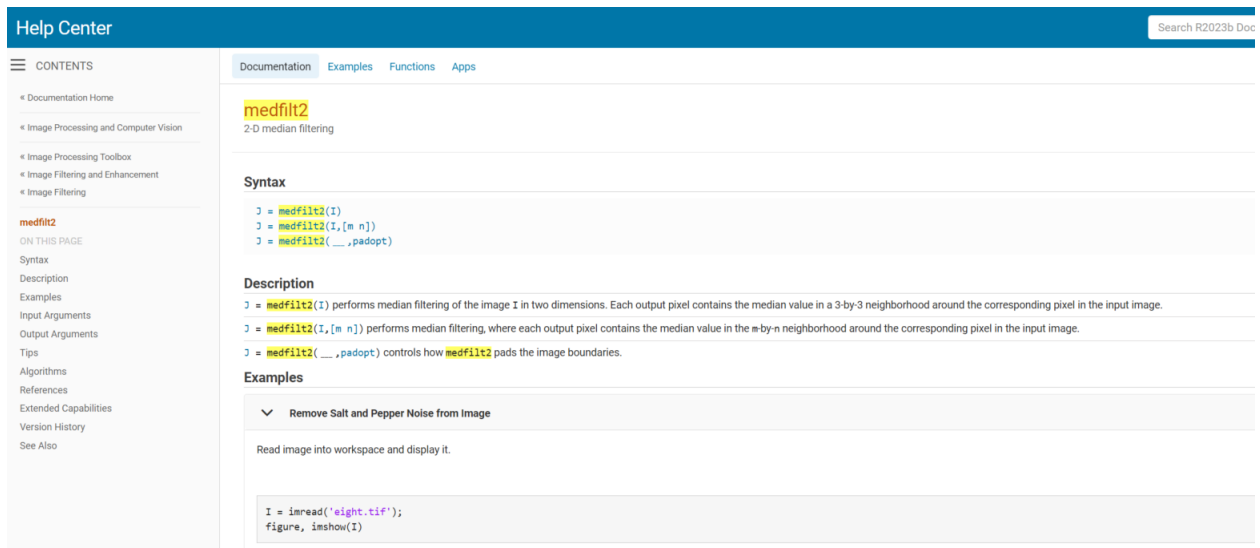
به دلیل ماهیت غیرخطی، لبه‌های تصویر را بهتر از فیلترهای خطی (مانند فیلتر میانگین) حفظ می‌کند و در نتیجه جزئیات بیشتری در تصویر حفظ می‌شود.

ه) با جست‌وجو در قسمت راهنمای MATLAB، تابع مورد استفاده در این نرم افزار برای پیاده‌سازی فیلتر میانه بر روی تصاویر را پیدا کنید. در مورد پارامترهای این تابع و نحوه و دستوره‌های لازم برای اعمال آن در MATLAB بر روی تصاویر توضیح دهید.

برای حذف نویز از تصویر، از فیلتر میانه (Median Filter) که در MATLAB با تابع `medfilt2` پیاده‌سازی شده است، استفاده می‌کنیم. این تابع مقدار هر پیکسل را با مقدار میانه‌ی همسایگی آن جایگزین کرده و به‌ویژه در حذف نویز Salt & Pepper مؤثر است. فرمت کلی دستور به شکل زیر است:

```
medfilt2(I,[m, n])
```

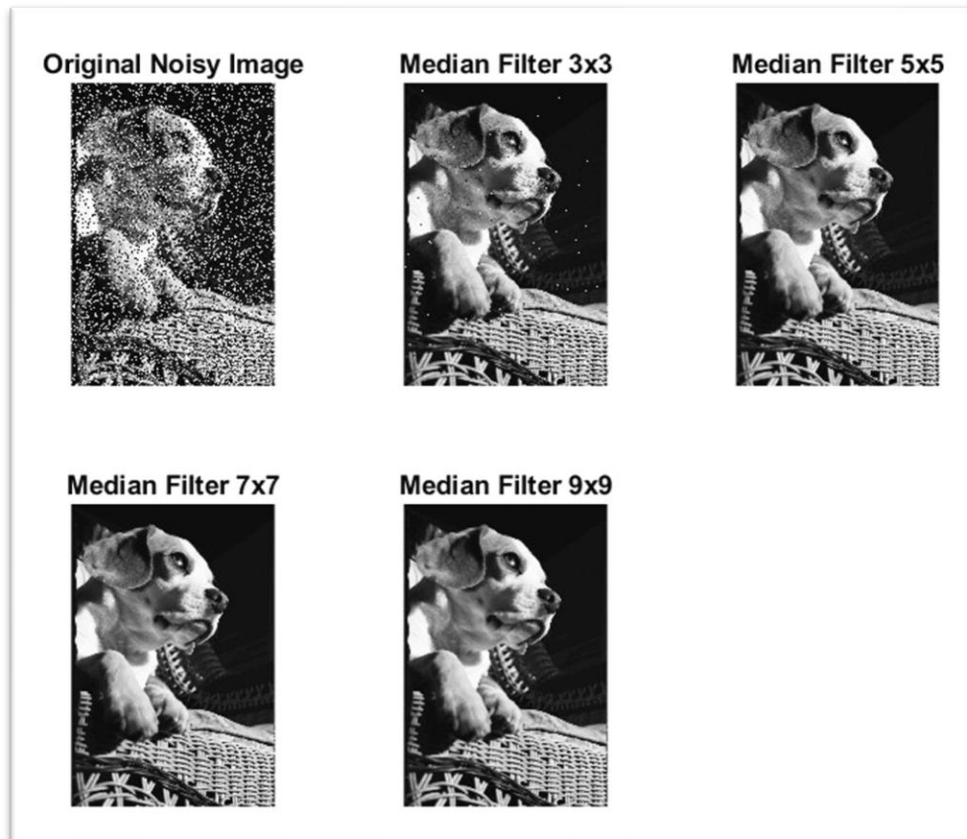
که در آن `[m n]` اندازه پنجره فیلتر را مشخص می‌کند. معمولاً از فیلتر `[3 3]` یا `[5 5]` استفاده می‌کنیم تا بدون از بین بردن جزئیات، نویز را کاهش دهیم.



The screenshot shows the MATLAB Help Center interface. On the left is a 'CONTENTS' sidebar with a tree view showing the path: Documentation Home > Image Processing and Computer Vision > Image Processing Toolbox > Image Filtering and Enhancement > Image Filtering > medfilt2. The main content area is titled 'medfilt2' and '2-D median filtering'. It includes sections for 'Syntax' with three function signatures, 'Description' explaining the function's purpose and parameters, and 'Examples' with a section titled 'Remove Salt and Pepper Noise from Image' that includes the code: `I = imread('eight.tif'); figure, imshow(I)`.

تصویر ۸- راهنمای MATLAB برای دستور `medfilt2`

و) به کمک توابع یافته‌شده در بندهای قبل، فایل تصویری داده شده را نویزدایی کنید. اندازه ماسک فیلتر را تغییر داده و اثر آن بر کیفیت خروجی را مشاهده کنید. مقدار بهینه‌اندازه ماسک را یافته و به بهترین شکل که می‌تواند تصویر را از نویزدایی و نتیجه را ضمیمه کنید. کیفیت تصویر خروجی و کیفیت شرح مبتنی بر درک شما از مطلوب تعیین‌کننده ارزیابی شما در این قسمت است.



تصویر ۹- تصاویر نویزدایی شده با ماسک‌های متفاوت

همان طور که در تصویر ۸ قابل مشاهده است؛ با استفاده از فیلتر 3×3 ، نویز کاهش یافته اما هنوز مقدار قابل توجهی از آن باقی مانده است. فیلتر 5×5 عملکرد بهتری داشته و نویز را حذف کرده در حالی که کیفیت تصویر اصلی حفظ شده است. در مقابل، فیلترهای 7×7 و 9×9 اگرچه نویز را کاملاً حذف کرده‌اند، اما باعث کاهش وضوح و محوشدگی تصویر شده‌اند. بنابراین، اندازه ماسک بهینه، 5×5 است زیرا هم نویز را به خوبی حذف می‌کند و هم کیفیت تصویر را حفظ می‌کند. خروجی نویزدایی شده با فیلتر 5×5 در تصویر ۹ قابل مشاهده است.

Median Filter 5x5



تصویر ۱۰- تصویر نویززدایی شده با فیلتر ۵x۵

پروژه سوم

الف) فایل تصویری Flower.png را به کمک دستور imread در محیط MATLAB خوانده و سیگنال متناظر با آن را ماتریس I بریزید. (اگر تصویر به صورت رنگی ذخیره شده است (شامل سه ماتریس است)، با استفاده از دستور rgb2gray آن را به صورت سیاه و سفید درآورید تا در یک ماتریس دوبعدی نشان داده شود).

مسیر فایل تصویر Flower.png را که در پوشه‌ی Resources قرار دارد، مشخص کردیم و آن را با استفاده از imread در متغیر I خواندیم. سپس بررسی کردیم که آیا تصویر رنگی است یا نه (متوجه شدیم که تصویر از ابتدا به صورت سیاه و سفید ذخیره شده است، زیرا دارای دو بعد بوده و بعد سوم که مربوط به کانال‌های رنگی در آن وجود نداشته است). بنابراین، نیازی به تبدیل آن با rgb2gray نبود و تصویر مستقیماً در قالب یک ماتریس دو بعدی باقی ماند.

ب) به کمک دستور imshow تصاویر متناظر با فایل داده شده را مشاهده و تصویر آن را در گزارش ضمیمه کنید. ابعاد این تصویر چند در چند است؟ هر پیکسل از این تصویر در چند بیت ذخیره شده است؟

ابعاد تصویر را با استفاده از size به دست آوردیم که مقدار آن برابر 532×769 پیکسل شد. همچنین، با استفاده از class نوع داده‌ی تصویر را بررسی کردیم که uint8 بود. در نهایت، تعداد بیت‌های هر پیکسل را محاسبه کردیم که مقدار آن 8 بیت به دست آمد، زیرا نوع داده uint8 از 8 بیت برای نمایش شدت روشنایی هر پیکسل استفاده می‌کنند.



تصویر ۱۱- تصویر بارگذاری شده با imread

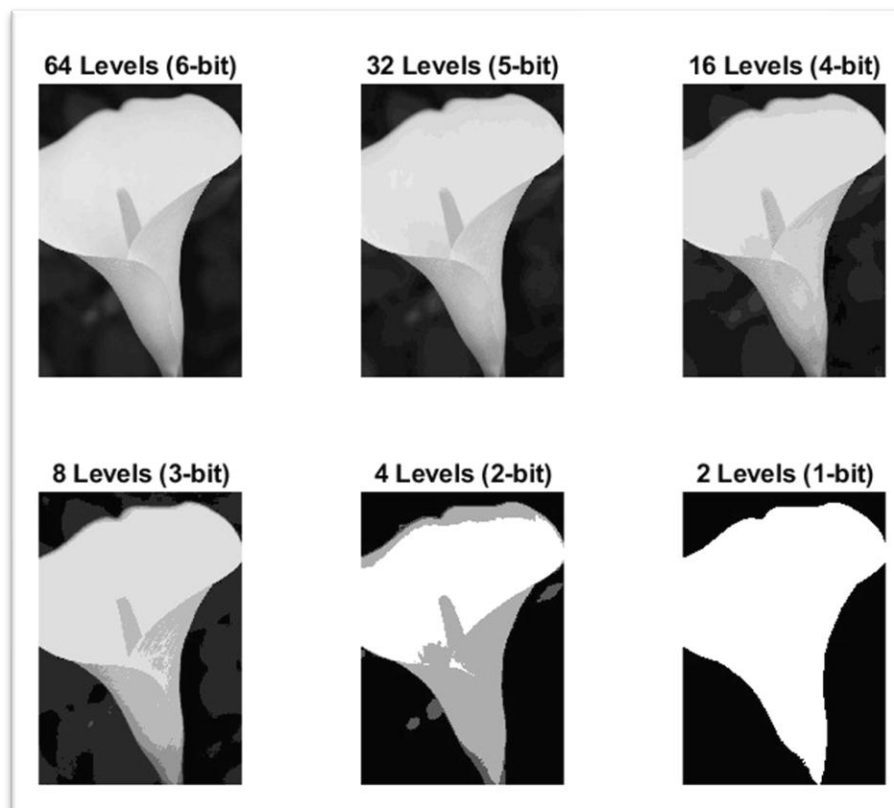
ج) در MATLAB کدی بنویسید که اصطلاحاً این تصویر را چندی‌سازی کند، یعنی مقادیر شدت روشنایی موجود در تصویر را در ۶۴ سطح، ۳۲ سطح، ۱۶ سطح، ۸ سطح، ۴ سطح و نهایتاً ۲ سطح گرد کند (دقت کنید که این کار به معنی ذخیره‌سازی مقدار شدت نور هر پیکسل به ترتیب در ۶، ۵، ۴، ۳، ۲ و نهایتاً ۱ بیت به جای مقدار اصلی است). کد شما باید پس از اعمال این شش سطح از چندسطحی‌سازی بر روی تصویر اولیه، نتیجه را در قالب شش تصویر نمایش دهد. کد و تصاویر حاصل را ضمیمه کنید.

بخشی از کد نوشته شده به صورت زیر است.

```
L = quantLevels(idx);

% normalize
I_normalized = double(I) / 255;
% quantize
I_quantized = round(I_normalized * (L-1));
% Rescale
Iq = uint8(round(I_quantized * (255/(L-1))));
```

خروجی در تصویر ۱۱ قابل مشاهده است.



تصویر 12- خروجی چندی‌سازی در سطوح‌های مختلف

د) با استفاده از جست‌وجو اینترنتی تحقیق کنید چگونه چندی‌سازی موجب پیدایش کانتورها (Contours) ناخوشایند در تصاویر می‌شود. دلیل و شکل رخداد این پدیده را شرح دهید.

وقتی یک تصویر با تغییرات نرم و پیوسته (مثلا تصویر آسمان) به تعداد کمی از سطوح گسسته تقسیم‌بندی می‌شود، به دلیل گرد شدن مقادیر و جمع‌شدن خطاهای کمی‌سازی، نواحی با مقادیر ثابت رنگ به وجود آمده و مرزهای بین این نواحی به صورت خطوط یا کانتورهای یکسان (با یک سطح از روشنایی) دیده می‌شوند. به این پدیده banding می‌گویند.

در واقع چون تعداد بیت و در نتیجه تعداد مقادیر روشنایی که یک پیکسل در یک عکس سیاه و سفید می‌تواند داشته باشد کاهش یافته، یعنی تعداد سطوح شدت روشنایی کم شده و دیگر امکان نمایش تغییرات تدریجی روشنایی وجود ندارد. در نتیجه، نوارهایی قابل مشاهده بین بخش‌های مختلف عکس به وجود می‌آیند.

ه) به کمک کد خود و تصاویر حاصل از آن، رخداد پدیده فوق در مورد تصویر داده شده را نشان دهید و بیان کنید به ازای چه تعداد سطوح چندی‌سازی این پدیده ظاهر می‌شود.

در تصاویر بدست آمده در 64 سطح (6 بیت) و 32 سطح (5 بیت) تصویر تقریبا بدون تغییر است و همان طور که در تصویر ۱۲ و ۱۳ قابل مشاهده است، اثری از کانتورها مشاهده نشد.

32 Levels (5-bit) 64 Levels (6-bit)



تصویر 13- چندی‌سازی در ۶۴ و ۳۲ سطح

در چندی‌سازی ۱۶ سطح (۴ بیت) همان‌طور که در تصویر ۱۳ قابل مشاهده است، اولین نشانه‌ها از کانتورها مشخص شده است. مخصوصاً در قسمت‌هایی که دارای تغییرات نرم است. (مثل سایه‌ها و پس‌زمینه)

16 Levels (4-bit)



تصویر 14 - چندی‌سازی در ۱۶ سطح

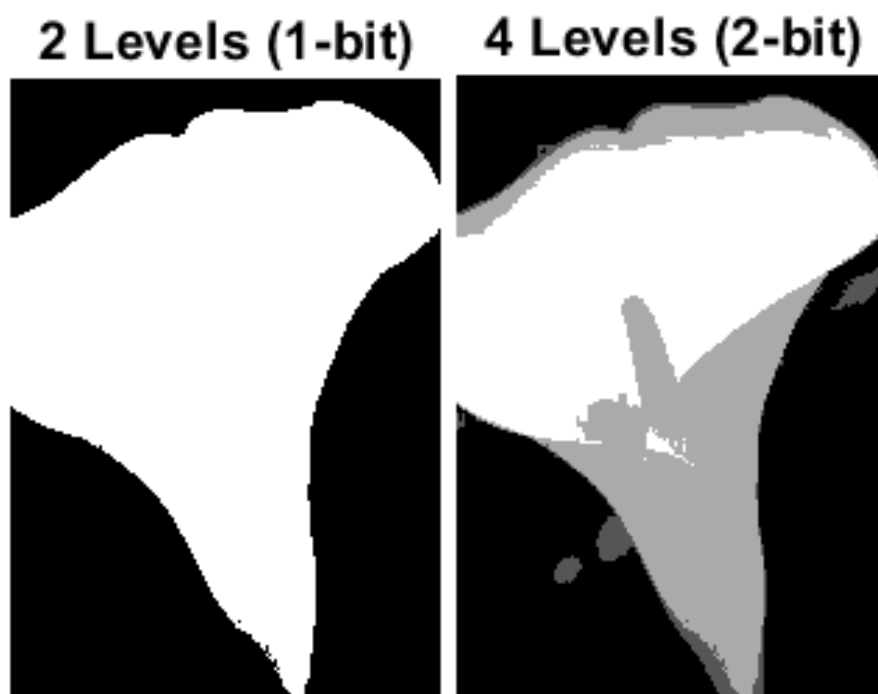
در چندی‌سازی ۸ سطح (۳ بیت) نوارهای مشخصی در تصویر به وجود آمده‌اند و همان‌طور که در تصویر ۱۴ قابل مشاهده است، تصویر کیفیت اصلی خود را از دست داده است.

8 Levels (3-bit)



تصویر 15 - چندی‌سازی در ۸ سطح

در چندی‌سازی ۲ و ۴ سطح (۱ و ۲ بیت)، تصویر بسیار ناپیوسته شده و تفاوت بین بخش‌های مختلف تصویر با کاهش تعداد سطح‌ها بسیار بزرگتر شده و می‌توان آن را به راحتی دید.



تصویر 16- چندی‌سازی در ۲ و ۴ سطح

در نتیجه از چندی‌سازی در 16 سطح (4 بیت) به پایین کانتورها به طور واضح قابل مشاهده بودند و کیفیت تصاویر به مرور کاهش پیدا کرد. پس برای جلوگیری از پدیده باندینگ (banding)، حداقل باید از چندی‌سازی در ۳۲ سطح (۵ بیت) یا بیشتر استفاده کنیم.