بسم الله الرحمن الرحیم

دانشگاه صنعتی اصفهان

دانشکده مهندسی برق و کامپیوتر

درس سیگنال ها و سیستم ها

تمرین کامپیوتری

فهرست مطالب

توضیحاتی درباره پروژه و تحویل فایل

درباره پروژه نکات زیر را مورد توجه قرار دهید:

1. انجام پروژه با نرم افزار متلب می باشد. آموزش های نرم افزار در سامانه قرار داده شده است. برای تحویل فایل فایل با پسوند **m.** یا **mlx.** را تحویل دهید.
2. پیشنهاد میشود برای زیبایی و راحت بودن شما از بخش **live editor** متلب استفاده نمایید.
3. کد های هر سوال را در فایل های جداگانه ای قرار دهید میتوانید با استفاده از **section** بندی متلب قسمت های مختلف یک سوال را جدا کنید. اگر نیاز به تابع در حل سوال می باشد آن را در یک فایل جداگانه قرار دهید.
4. انجام پروژه با استفاده از زبان های دیگر مانند **python** نیز بلامانع است، اما پیشنهاد میشود با **matlab** انجام دهید.
5. برای تحویل فایل یک گزارش از کار خود تهیه کنید و در آن الگوریتم های خود را شرح داده و نتایج را نشان دهید. بدیهی است که تمیز و دقیق بودن گزارش نمره جداگانه ای خواهد داشت.
6. در نهایت فایل های متلب و گزارش با فرمت **pdf** خود را به صورت یک فایل فشرده **zip** یا **rar** داخل سامانه قرار دهید.
7. زمان تقریبی برای انجام تکلیف **2** روز در نظر گرفته شده است. لطفا برنامه ریزی لازم برای انجام آن را داشته باشید. موعد تحویل قابل تمدید نمی باشد.

توضیحاتی درباره FFT

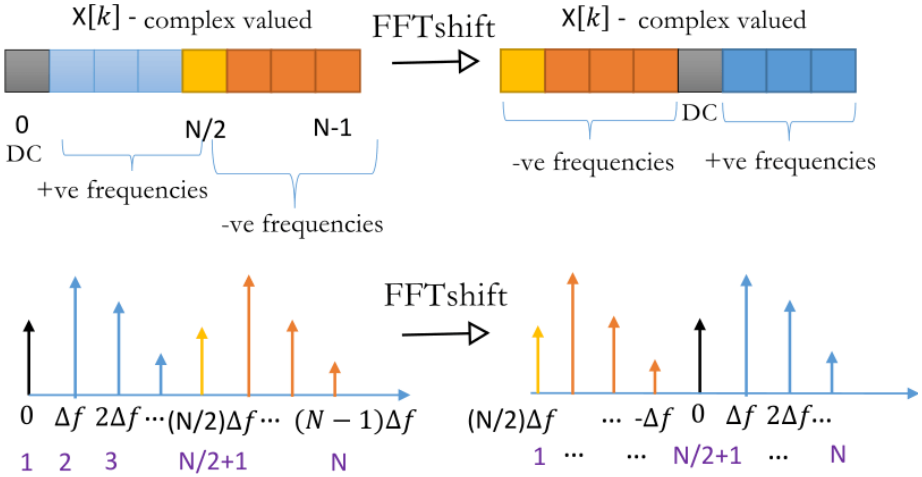
حتما می دانید که هنگام کار با کامپیوتر داده ها هم به صورت زمان گسسته هستند و هم کوانتیزه شده اند. بنابراین در عمل محاسبه تبدیل فوریه گسسته زمان که طیفی پیوسته از محور فرکانس را در خروجی به ما می دهد، در کامپیوتر غیر ممکن است. بنابراین اینجا نیاز به تبدیلی داریم که سیگنال زمان گسسته ما را به تعداد محدودی نمونه **map** کند. این تبدیل که آن را **DFT** میخوانند به صورت زیر تعریف میشود:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j \frac{2\pi}{N} kn}$$

همچنین رابطه محاسبه عکس **DFT** به صورت زیر می باشد:

$$x[n] = \sum_{k=0}^{N-1} X[k] e^{j \frac{2\pi}{N} kn}$$

حال این تبدیل قابل استفاده در کامپیوتر می باشد. تا کنون تفاوت بین **DTFT** و **DFT** را متوجه شدیم. برای اطلاعات بیشتر میتوانید به راهنمای متلب مراجعه کنید. در واقع میتوان گفت خروجی **DFT** نمونه هایی از خروجی **DTFT** می باشند. مانند آن است که سیگنال $X(e^{j\omega})$ را به یک A/D داده باشیم و از آن نمونه برداری فرکانسی انجام داده باشیم! حال با کمی دقت متوجه میشویم محاسبه **DFT** برای یک سیگنال کاری بسیار زمان بر است. با افزایش نمونه ها زمان محاسبه **DFT** با شیب زیادی افزایش میابد. در اینجاست که **FFT** خود را به نمایش میگذارد. **FFT** یا **fast Fourier transform** در واقع یک پیاده سازی و روش محاسبه برای تبدیل **DFT** می باشد که سرعت محاسبات را افزایش می دهد. در نهایت میتوان برای نمایش بهتر (مطابق آنچه سر کلاس کشیده میشود) از دستور **fftshift** استفاده کنید تا یک بازه متقارن را رسم کنید.



با این مقدمه میتوانید متوجه شوید که دستور **FFT** چه عملیاتی انجام می دهد. شما کافی است با استفاده از دو دستور **fft** و **ifft** متلب استفاده خود را انجام دهید.

نکته: در هنگام استفاده از دستور **fft** اگر طول سیگنال توانی از **2** باشد بهتر است، چرا ؟!

تمرین 1: شروع کار با متلب

1. سیگنال زمان پیوسته $x(t) = u(t+1) - u(t-2) + u(t-4)$ را در متلب با کمک دستور Heaviside رسم نمایید.
2. سیگنال زمان پیوسته $x(t) = u(t+1) - u(t-2) + u(t-4)$ را در متلب بدون کمک دستور Heaviside رسم نمایید.
3. سیگنال زمان گسسته ramp (شیب واحد) را در بازه $-3 \leq n \leq 6$ رسم نمایید.
4. در متلب تابعی بنویسید که دو عدد r, ω را از کاربر دریافت کرده و قسمت حقیقی، قسمت موهومی، اندازه و فاز $x[n] = r^n e^{j\omega n}$ را رسم نماید.

تمرین 2: دست گرمی با کانولوشن

1. میخواهیم حاصل کانولوشن دو پالس مربعی را مشاهده کنیم. برای این کار ابتدا سیگنال n که محور زمان می باشد را بسازید سپس تابع $x[n]$ را به صورت یک پالس مربعی تعریف کنید و آن را رسم کنید. در نهایت با استفاده از کانولوشن حاصل $y[n]$ را به دست آورده و رسم کنید.
2. حال میخواهیم حاصل کانولوشن دو پالس مربعی با زمان های متفاوت را مشاهده کنیم. ابتدا در گزارش کار خود به صورت تئوری بررسی کنید که حاصل به چه صورت میشود؟

(راهنمایی: حاصل به صورت یک ذوزنقه است، طول ساق ها و ارتفاع آن را محاسبه کنید.)

3. سیستمی با پاسخ ضربه $h(t)$ تعریف شده است. پاسخ این سیستم به ورودی $x(t)$ را محاسبه و رسم کنید.

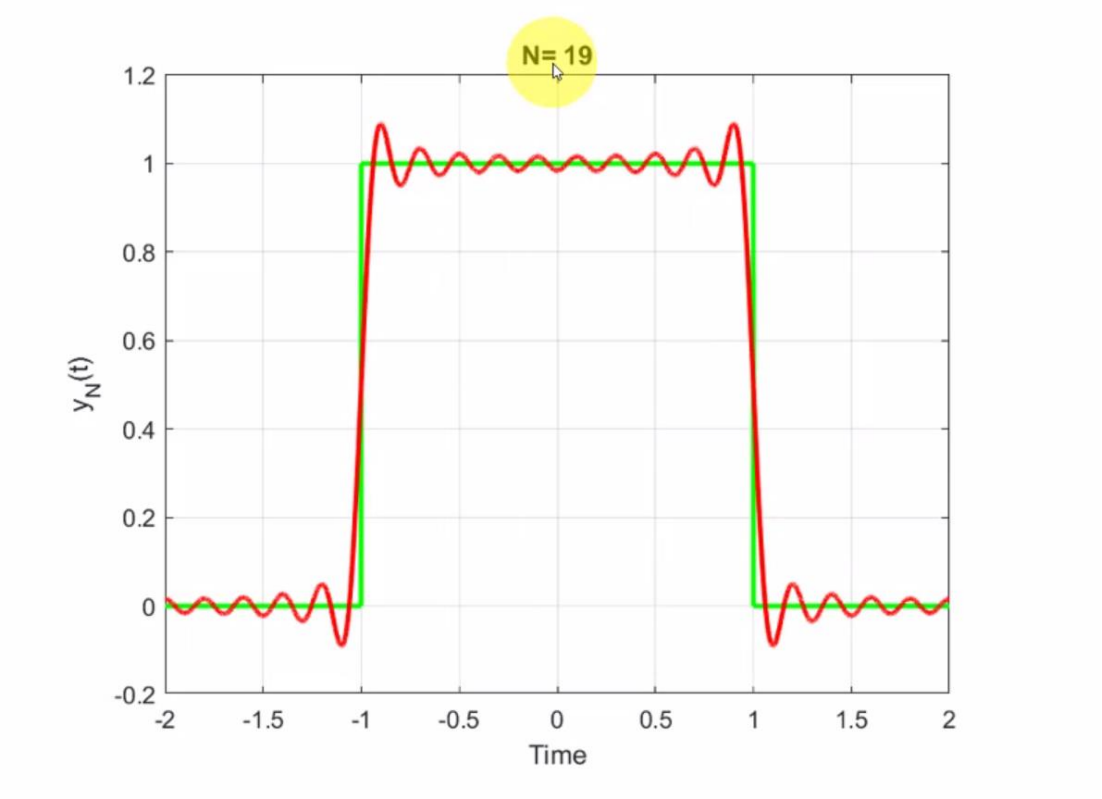
$$x(t) = \begin{cases} e^{-t} & 1 \leq t \leq 3 \\ 0 & O.W \end{cases}$$

$$x(t) = \begin{cases} t & 1 \leq t \leq 3 \\ 1 & 3 < t \leq 5 \\ 0 & O.W \end{cases}$$

تمرین 3: سری فوریه و مشاهده پدیده گیبس

1. اسکریپت متلبی بنویسید که سری فوریه یک موج مربعی را با استفاده از **N** جمله اول سری فوریه تشکیل داده و به همراه موج مربعی رسم نماید.

2. با افزایش تعداد **N**، نزدیک شدن سری فوریه به موج مربعی و پدیده گیبس را مشاهده کنید. مقدار بالا زدگی را در گزارش خود یادداشت نمایید.

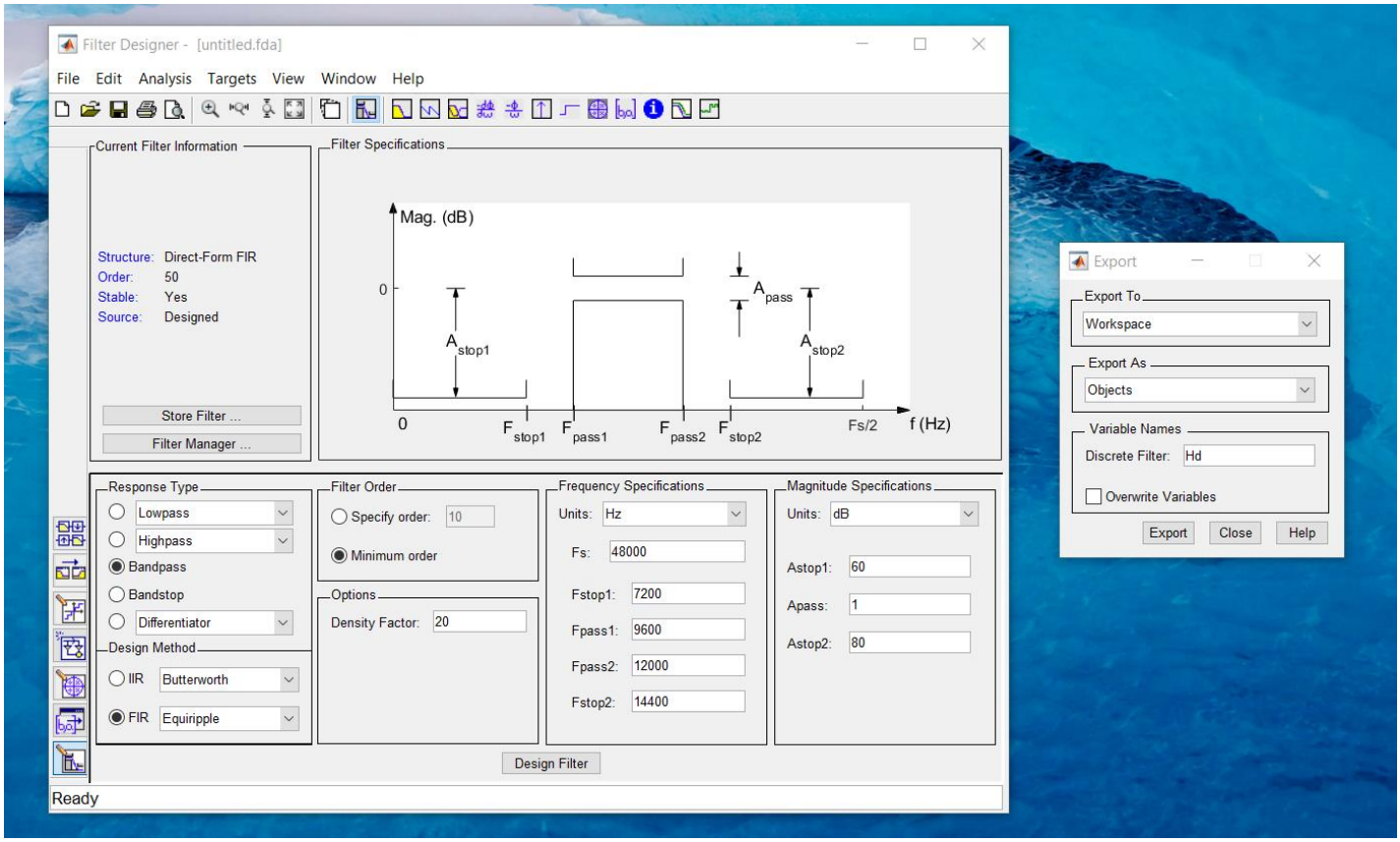


تمرین 4: تبدیل فوریه و کاربرد های بی شمار

بخش اول: شبیه سازی نویز و حذف آن به کمک فیلتر

1. فرض کنید نمونه های سیگنال $x(t) = cos(\omega_0 t)$ از یک کانال با نویز **AWGN** عبور می کند. با انتخاب ω_0 مناسب و تشکیل $x[n]$ و اضافه کردن نویز سفید گوسی جمع شونده (**AWGN**) تاثیر آن را بر روی سیگنال خروجی زمانی $y[n]$ مشاهده کرده و رسم کنید.

2. در قدم بعدی میخواهیم با استفاده از یک فیلتر میان گذر نویز را تا حد خوبی از سیگنال $y[n]$ حذف کنیم و سیگنال $\hat{x}[n]$ را به دست آوریم. برای این کار با وارد کردن دستور **filterDesigner** در قسمت **command window** متلب میتوانید فیلتر مناسب خود را طراحی کنید.



بعد از طراحی از منوی **File** گزینه **Export** را کلیک کنید و فیلتر خود را به صورت یک **Object** به **Workspace** بدهید. در نهایت میتوانید سیگنال فیلتر شده خروجی $\hat{x}[n]$ را رسم کنید.

3. (اختیاری): تابعی بنویسید که کانولوشن چرخشی دو دنباله که لزوما طول برابری نیز ندارند را با استفاده از **fft** و **ifft** پیاده سازی کند.

بخش دوم: کاهش خطای MSE به کمک تبدیل فوریه

گاهی اوقات یک سیگنال نسخه خراب شده سیگنال دیگری است، برای مثال زمانی که یک سیگنال از یک کانال ارتباطی عبور می کند یا زمانی که یک سیگنال فشرده می شود. برای بازیابی سیگنال اصلی از سیگنال خراب شده، می توانیم از یک سیستم **LTI** استفاده کنیم. این سیستم سیگنال تحریف شده را دریافت کرده و سیگنال بازیابی شده را برمی گرداند. این سیستم **LTI** بسته به نوع اعوجاج می تواند در حوزه فرکانس یا حوزه زمانی طراحی شود. برای اندازه گیری تفاوت بین دو سیگنال، از میانگین مربعات خطا (**MSE**) به عنوان متریک استفاده می کنیم. **MSE** برای سیگنال های معلوم $x[k]$ و $y[k]$ با طول L به صورت زیر تعریف می شود:

$$MSE = \frac{1}{L}\sum_{n=1}^{L}(y[n] - x[n])^2$$

دو فایل **Original.wav** و **Distorted.wav** به ترتیب سیگنال اصلی را با طول **2** ثانیه و نسخه تخریب شده آن را ارائه می دهند. فرکانس نمونه برداری برای این سیگنال های صوتی **Fs = 22050** هرتز و تعداد بیت ها در هر نمونه **16** است. بنابراین، تعداد نمونه ها در هر سیگنال **L = 44100** است.

1. دو فایل **Original.wav** و **Distorted.wav** را در برنامه خود بارگذاری کنید و آنها را در حوزه زمان رسم کنید.
2. از **MSE** برای مقایسه سیگنال اصلی و مخدوش استفاده کنید.
3. سیستمی طراحی کنید که سیگنال اصلی را از سیگنال تحریف شده بازیابی کند و آن را در فایلی به نام **Recovered.wav** ذخیره کنید. کدام دامنه را برای طراحی انتخاب کردید: دامنه فرکانس یا دامنه زمان؟ چرا؟
4. **MSE** را بین سیگنال بازیابی شده و سیگنال اصلی محاسبه کنید. آیا سیستم شما **MSE** را بهبود می بخشد؟
5. با پخش و گوش دادن به سیگنال بازیابی شده، آیا سیستم شما کیفیت صدا را بهبود می بخشد؟

لیست توابع مفید در متلب

- heaviside
- exp
- Plot
- stem
- abs
- angle
- real
- imag
- sqrt
- sum
- mean
- var
- std
- conv
- fft
- ifft
- fftshift
- filter
- Audioread
- awgn
- randn
- audiowrite
- sound

نکته: از صفحات بالا که به صورت **HiperLink** قرار داده شده می توانید به صورت رفرنس برای یادگیری استفاده کنید.

موفق باشید 😊