تمرین کامپیوتری اول

سیگنال‌ها و سیستم‌ها

علیرضا وثوقی‌راد
a.vosoughi99@gmail.com

دکتر اخایی
پاییز ۹۸

مقدمه

در این تمرین قصد داریم تا سیستم‌های LTI را در حوزه ی زمان و فرکانس بررسی کنیم. همچنین در ادامه با فیلتر ها و نمودار اندازه و فاز آن ها آشنا می‌شویم و به تحلیل آن ها می‌پردازیم.

سوال ۱) پاسخ ضربه

پاسخ ضربه یک سیستم نقش بسیار مهمی در پیدا کردن سیگنال خروجی یک سیستم به یک سیگنال ورودی دارد. در این بخش می‌خواهیم با استفاده از پاسخ ضربه‌ی یک سیستم، خروجی آن را تعیین کنیم.

فرض کنید $\mathrm{y(t)}$ سیگنال خروجی یک سیستم $\mathrm{H\ (LTI)}$ به یک ورودی دلخواه است. سیگنال $\mathrm{x(t)}$ را می‌توانیم به صورت یک کانولوشن با سیگنال دلتای دیراک بنویسیم:

$$x(t) = \int_{-\infty}^{\infty} x(\tau).\delta(t-\tau)\mathrm{d}\tau$$

اپراتور $\mathrm{H(.)}$ پاسخ سیستم به وروی دلخواه را مشخص می‌کند. در نتیجه برای پیدا کردن خروجی سیستم به ورودی داریم:

$$y(t) = H\{\int_{-\infty}^{\infty} x(\tau).\delta(t-\tau)\mathrm{d}\tau\}$$

از آن جایی که اپراتور H بر روی t عمل می‌کند، بنابراین عبارت $\mathrm{x}(\tau)$ مانند یک ضریب می‌شود:

$$y(t) = \int_{-\infty}^{\infty} x(\tau).H\{\delta(t-\tau)\}\mathrm{d}\tau$$

پاسخ یک سیستم را به سیگنال ضربه با $\mathrm{h(t)}$ نشان می‌دهیم و از آنجایی که سیستم مورد نظر ما LTI می‌باشد،پس هر شیفت در ورودی منجر به همان مقدار شیفت در خروجی می‌شود.

$$h(t) = H\{\delta(t)\}$$

$$H\{\delta(t-\tau)\} = h(t-\tau)$$

$$y(t) = \int_{-\infty}^{\infty} x(\tau).h(t-\tau)\mathrm{d}\tau$$

همانطور که مشاهده می‌کنید پاسخ این سیستم به ورودی دلخواه برابر کانولوشن پاسخ ضربه سیستم و سیگنال ورودی شد.

۱.۱ معادله دیفرانسیل

یکی از راه های نمایش سیستم، نوشتن معادله‌ی ode حاکم بر سیستم است.
فرض کنید سیستمی داریم که ورودی آن x(t) و خروجی آن y(t) می‌باشد. معادله دیفرانسیل حاکم بر این سیستم به صورت زیر می‌باشد. حال می‌خواهیم پاسخ این سیستم را به ورودی مورد نظرمان بیابیم.

$$y(t) + \frac{\mathrm{d}}{\mathrm{d}t}y(t) = x(t)$$

$$x(t) = e^{-2t}u(t)$$

برای حل معادله دیفرانسیل باید شرایط اولیه‌ی سیستم موجود باشد. شرایط اولیه‌ی سیستم فوق را به صورت زیر در نظر بگیرید.

$$y(t)|_{t=0-} = 0 \quad and \quad \frac{\mathrm{d}}{\mathrm{d}t}y(t)|_{t=0-} = 0$$

برای حل معادله ی فوق در پایتون و پیدا کردن پاسخ سیستم به ورودی ابتدا لازم است که معادله‌ی فوق را تعریف کنیم.
ابتدا کتابخانه ی sympy را فراخوانی کرده و با استفاده از توابع symbol و function متغیرهای خود را تعریف می‌کنیم. متغیرهای مستقل باید با symbol تعریف شوند و سیگنال‌های ورودی و خروجی که از یک ضابطه‌ی خاص تبعیت می‌کنند باید با function تعریف شوند. مثال:

```
ind_var = sym.symbols('ind_var', real=True)
signal = sym.Function('signal')(t)
```

بعد از تعریف متغیرها حال باید معادله را تشکیل دهیم. برای اینکار از تابع Eq از کتابخانه sympy استفاده می‌کنیم. قبل از استفاده از آن در مورد این تابع و آرگومان‌های آن مطالعه کنید.
هنگامی که می‌خواهید مشتق سیگنال y را به تابع فوق بدهید از y.diff(t) استفاده کنید.
برای حل معادله از تابع dsolve استفاده کنید و objectای که معادله در آن است را به آن پاس بدهید. دقت کنید که قبل از آن با استفاده از دستور subs ضابطه‌ی x را در معادله جایگذاری کنید. سپس با استفاده از solve شرایط اولیه را اثر دهید و ثابت ها را پیدا کنید و در نهایت با دستور subs ثابت ها را در پاسخ بدست آمده جایگذاری کرده و سیگنال نهایی را پلات (sympy.plot) کنید.

```
solution = sym.solve(ode.subs(x,"what is function of x?"))
integration_constants = sym.solve((solution.rhs.limit(t,0,'-'),
solution.rhs.diff(t).limit(t,0,'-')),'C1')
```

حال با استفاده از روش فوق پاسخ سیستم به ورودی ضربه یا همان دلتای دیراک را محاسبه کنید و سپس آن را رسم کنید. در پایتون برای سیگنال دلتای دیراک می‌توانید از عبارت sympy.DiracDelta(t) استفاده کنید.
بعد از اینکه پاسخ ضربه‌ی سیستم را محاسبه کردید با استفاده از دستور sympy.integrate حاصل انتگرال زیر که همان کانولوشن سیگنال ورودی و پاسخ ضربه آن است را بیابید.
توجه: از آن جایی که هر دو سیگنال ورودی و پاسخ ضربه "علی" هستند در نتیجه می‌توانید حد پایین انتگرال را صفر قرار دهید.

$$y(t) = \int_0^t x(\tau).h(t-\tau)\mathrm{d}\tau$$

بعد از انجام مراحل فوق، به پرسش‌های زیر پاسخ دهید:
۱) به صورت دستی بررسی کنید که آیا h(t) که در قسمت قبل آن را بدست آوردید جواب معادله ode می‌باشد؟
۲) آیا پاسخ سیستم به ورودی از هر دو روش بالا یکسان است؟ بحث کنید.

سوال ۲ تبدیل فوریه

با استفاده از تابع fourier_transform کتابخانه sympy می‌توان تبدیل فوریه‌ی یک سیگنال را بدست آورد. در این قسمت می‌خواهیم تبدیل فوریه ی سیگنال مستطیلی را محاسبه کنیم. حل دستی این تبدیل به صورت زیر می‌باشد:

$$F\{rect(t)\} = \int_{-\infty}^{\infty} rect(t)e^{-j\omega t}\mathrm{d}t = \int_{-0.5}^{0.5} e^{-j\omega t}\mathrm{d}t = \frac{\sin(\frac{\omega}{2})}{\frac{\omega}{2}}$$

$$sinc(x) = \begin{cases} \frac{\sin(x)}{x} & ; x \neq 0 \\ 1 & ; x = 0 \end{cases}$$

برای اینکه تبدیل فوریه ی سیگنال rect را محاسبه کنیم ابتدا آن را به صورت زیر تعریف می‌کنیم. کلاس سیگنال rect حتما باید از sympy.function ارثبری کند و classmethod آن eval نامگذاری شده و دو آرگمان cls و arg داشته باشد.

```
class rect(sym.function):
    @classmethod
    def eval(cls, arg):
        return sym.Heaviside(arg + sym.S.Half) - sym.Heaviside(arg -
sym.S.Half)
```

حال تبدیل فوریه ی تابع rect را با استفاده از تابع fourier_transform محاسبه و ترسیم کنید.

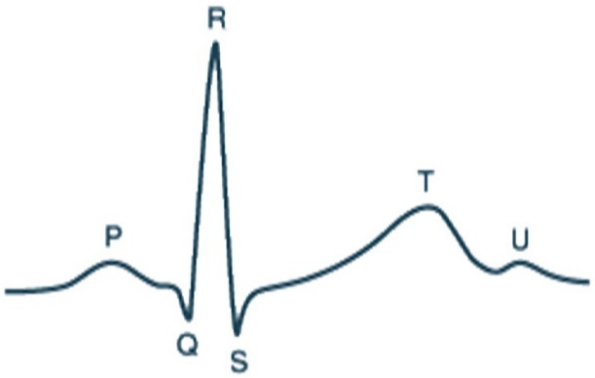
بعد از انجام مراحل فوق به پرسش‌های زیر پاسخ دهید:

۱) با تغییرکدی که برای محاسبه ی فوریه‌ی rect در همین قسمت زدید، اینبار فوریه‌ی rect(at) را حساب کنید و نتایج را تحلیل کنید.(یکبار a را عددی بزرگتر از یک و بار دیگر عددی بین صفر و یک قرار دهید.)

۲) اگر بازه ی سیگنال مستطیلی را تغییر دهیم فوریه ی آن چه تغییری می‌کند؟

سوال ۳) قلب ممد

یکی از ابزارهای اندازه‌گیری فشار خون کاف می‌باشد. این روش با اینکه دقت بالایی دارد اما معایبی نیز دارد. در این روش برای اندازه‌گیری فشار خون مجبور به بستن رگ می‌شویم که همین بسته شدن رگ می‌تواند مشکلاتی را برای بیماران قلبی ایجاد کند. همچنین به دلیل بسته شدن رگ حداقل تا پانزده دقیقه نمی‌توانیم فشار خون را اندازه‌گیری کنیم. برای حل این مشکل می‌توانیم با مقایسه‌ی سیگنال قلب و نبض سرعت خون در بدن را بدست آوریم و در نتیجه فشار خون را محاسبه کنیم. در این تمرین هدف فیلتر کردن سیگنال قلب و آماده‌سازی آن برای انجام پردازش است. همانطور که در شکل ۱ مشاهده می‌کنید هر سیگنال قلب از قله‌هایی تشکیل شده است که آن ها را $\mathrm{P, Q, R, S, T, U}$ می نامیم. برای اندازه گیری فشار سیستولیک $(\mathrm{systolic})$ ما فقط به قله‌ی R نیاز داریم. در بسیاری از موارد قله‌ی T بسیار نزدیک به قله‌ی R می‌باشد و کار ما برای تشخیص قله‌ی R دشوار می‌شود. هر کدام از این قله‌ها دارای یک فرکانس هستند که با یک فیلتر مناسب می‌توانیم قله‌ی مورد نظر خود نگه داشته و باقی قله‌ها را تضعیف کنیم. در اینجا برای حل مشکل نزدیک بودن قله‌ی T به قله‌ی R از یک فیلتر بالا گذر استفاده خواهیم کرد.



شکل ۱: سیگنال قلب

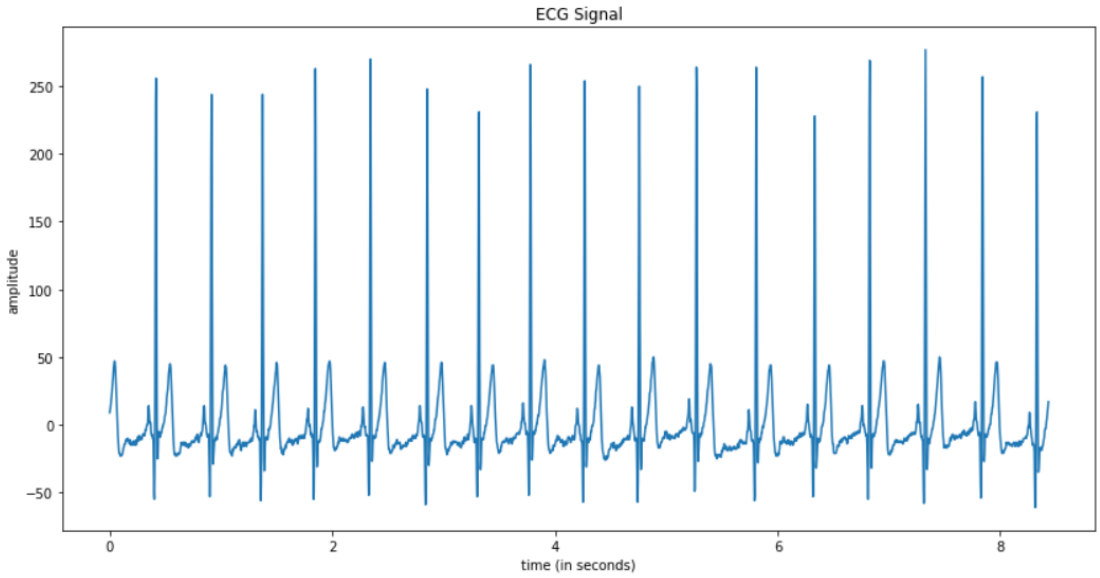
روش های مختلفی برای فیلتر کردن یک سیگنال وجود دارد. یک روش غیر کاربردی این است که تبدیل فوریه‌ی سیگنال را محاسبه کنیم و و سیگنال‌های ناخواسته را حدف کنیم و و از آن فوریه‌ی معکوس بگیریم. روش دیگر استفاده از فیلترهای باتروورث است. فیلترهای باتروورث را می‌توان به دو نوع پایین گذر و بالا گذر تقسیم کرد.

$$|H(j\omega)|^2 = \frac{1}{1 + (\frac{j\omega}{s})^{2N}}$$

فیلترهای باتروورث معایبی نیز دارند که می‌توان به غیر خطی بودن فاز آن‌ها اشاره کرد.
یکی دیگر از روش‌های فیلتر کردن استفاده از فیلت های FIR و IIR است. استفاده از این فیلترها در صنعت بسیار رایج است. این فیلترها همگی در حوزه ی زمان اعمال می‌شوند و نحوه‌ی عملکرد آن ها با تبدیل Z توجیح می‌شود که در ادامه‌ی درس با این تبدیل آشنا خواهید شد. در این تمرین فقط از این فیلتر استفاده خواهید کرد.
سوال:
۱) درباره‌ی فیلترهای باتروورث تحقیق کنید و شکل اندازه و فاز این فیلتر را در گزارش کار خود آورده و هر کدام را تحلیل کنید.
۲) درباره‌ی فاز خطی و تاثیر آن بر سیگنال در حوزه‌ی زمان تحقیق کنید و سپس استدلال کنید که چرا برای فیلتر کردن سیگنال‌ها فیلتر باتروورث نمی‌تواند گزینه‌ی خوبی باشد.

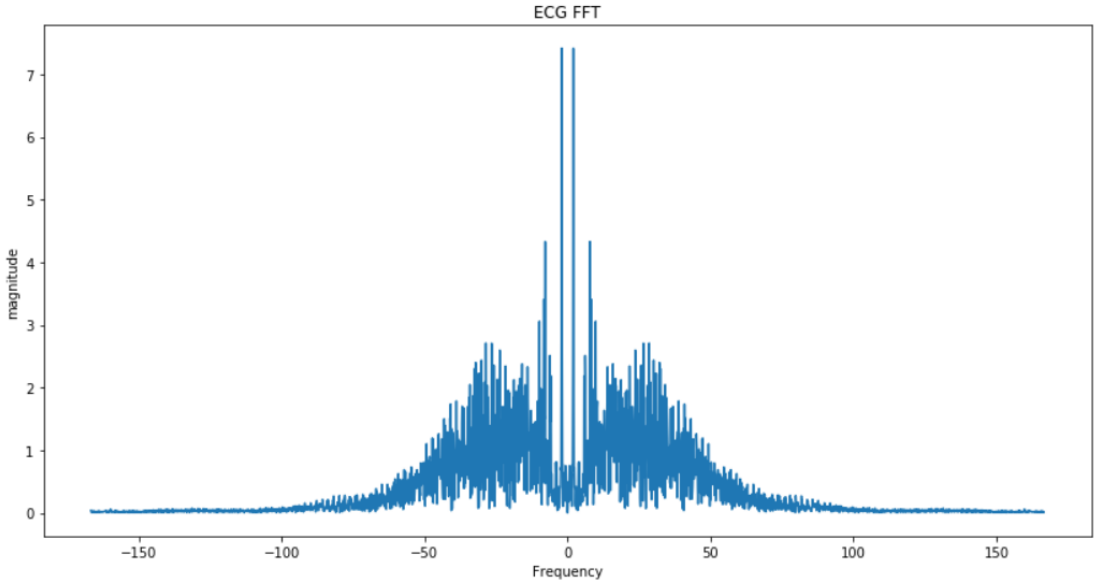
۳) درباره ی فیلترهای FIR و IIR تحقیق کرده و به اختصار در گزارش خود دربارهی آنها توضیح دهید و دربارهی اندازه و فاز این فیلترها بحث کنید.
در ادامه می‌خواهیم با استفاده از روش‌های فوق سیگنال قلب را فیلتر کنیم.

قسمت اول رسم سیگنال:
سیگنال قلب ممد به صورت یک فایل csv در اختیار شما قرار داده شده است. در مرحلهی اول لازم است تا این دادههارا فراخوانی کرده و آن را در یک متغیر ذخیره کنید. راههای مختلفی برای خواندن یک فایل csv وجود دارد که یکی از آنها استفاده از تابع genfromtxt از پکیج numpy است. (دقت کنید که لزومی برای استفاده از توابع ذکر شده در سوال نیست و شما می‌توانید از روش دلخواه خودتان استفاده کنید.) در ادامه قسمت dc سیگنال را حذف کنید و آن را در یک متغیر ذخیره کنید. این سیگنال با فرکانس نمونه برداری ۳۴.۳۳۳ هرتز نمونه برداری شده است. حال بردار زمانی را با توجه به فرکانس نمونه برداری و تعداد نمونهها محاسبه کنید و آن را در یک متغیر ذخیره کنید. حال سیگنال قلب را بر حسب بردار زمانی رسم کنید.



شکل ۲: سیگنال ECG

قسمت دوم سیگنال در حوزه ی فرکانس:
در این قسمت می‌خواهیم تبدیل فوریهی سیگتال قلب را محاسبه کرده و آن را نمایش دهیم.
با استفاده از دستور numpy.fft.fft از این سیگنال تبدیل فوریه می‌گیریم و با استفاده از numpy.fft.fftshift تبدیل فوریهی را انتقال می‌دهیم تا مبدا آن به مرکز منتقل شود به صورت متقارن و صحیح نمایش داده شود. چون سیگنال حاصل، مختلط است برای نمایش آن حتما باید اندازهی آن را نمایش دهیم .پس در ادامه اندازهی هر یک از المان‌ها را محاسبه کرده و در یک بردار جدید ذخیره می‌کنیم.
حال باید محور افقی را بدست آوریم. تحقیق کنید که بازه ی محور افقی تبدیل فوریهی یک سیگنال در چه محدودهای باید باشد. سپس دامنهی فرکانسی را در یک بردار ذخیره کنید و سپس آن را رسم کنید.



شکل ۳: تبدیل فوریه‌ی سیگنال ECG

قسمت سوم فیلتر کردن:

قسمت ۱.۳: حذف دستی سیگنال های ناخواسته

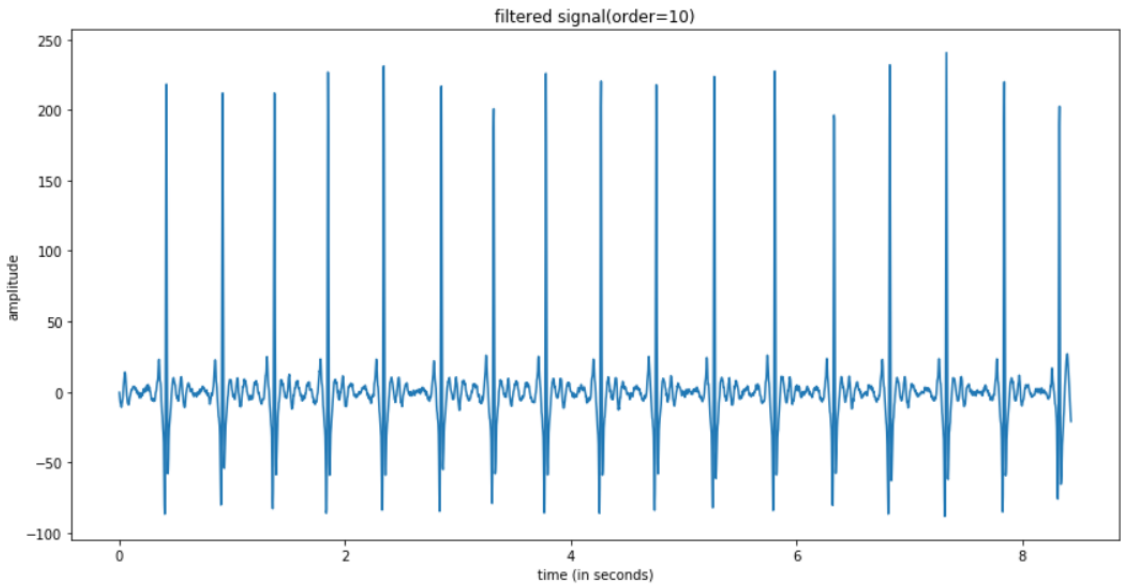
بعد از عدد گذاری صحیح محور فرکانسی، اندازه‌ی فرکانس‌های بین ۶ و -۶ را حذف کنید و سپس از آن با استفاده از تابع numpy.fft.ifft فوریه معکوس بگیرید و ((اندازه)) آن را در حوزه‌ی زمان رسم کنید.

قسمت۲.۳ : استفاده از فیلتر باتروورث:

برای استفاده از فیلتر باتروورث می‌توانیم از تابع scipy.signal.butter استفاده کنیم و دو ضریب a و b را به عنوان خروجی بگیریم. هر یک از ورودی‌های این تابع را توضیح دهید و مشخص کنید که هرکدام باید چه مقداری داشته باشند.

بعد از بدست آوردن ضرایب از تابع scipy.signal.filtfilt استفاده می‌کنیم و داده‌ها را فیلتر می‌کنیم.

مرتبه‌ی این فیلتر را یک بار ۶ بار دیگر ۱۰ و سپس ۵۰ قرار دهید و شکل‌ها را رسم کنید و تغییراتی که با افزایش مرتبه رخ می‌دهد را توجیه کنید. همچنین توضیح دهید که با افزایش مرتبه، اندازه و فاز فیلتر چه تغییری می‌کنند.



شکل ۴: سیگنال فیلتر شده

قسمت ۳.۳: فیلتر های FIR

یک فایل با نام filter.csv در اختیار شمما قرار گرفته است که شما باید آن را خوانده و در یک متغیر ذخیره کنید.
سپس با استفاده از دستور numpy.convolve این فیلتر را در سیگنال اصلی کانوالو کنید و نتیجه ی حاصل را رسم کنید.
همچنین این فیلتر و فیلتر باترورث در قسمت قبل را مقایسه کنید.

صدای کلید تلفن

در این تمرین با چگونگی استفاده از فرکانس‌های مختلف برای تشخیص کلید فشار داده شده در تلفن آشنا می‌شوید. صدایی که هنگام فشار دادن یک کلید می‌شنوید، جمع دو سیگنال سینوسی است. سیگنال با فرکانس بالا ستون کلید و سیگنال با فرکانس پایین ردیف کلید در صفحه کلید تلفن را مشخص می‌کند. در دو جدول پایین فرکانس‌های استفاده شده برای کلیدهای مختلف را مشاهده می‌کنید. همچنین در کل این تمرین فرض کنید فرکانس نمونه‌برداری برابر $8192\ \text{Hz}$ است و از fft با 2048 نقطه استفاده کنید.

	ω_{col}		
ω_{row}	0.9273	1.0247	1.1328
0.5346	1	2	3
0.5906	4	5	6
0.6535	7	8	9
0.7217		0	

Digit	ω_{row}	ω_{col}
0	0.7217	1.0247
1	0.5346	0.9273
2	0.5346	1.0247
3	0.5346	1.1328
4	0.5906	0.9273
5	0.5906	1.0247
6	0.5906	1.1328
7	0.6535	0.9273
8	0.6535	1.0247
9	0.6535	1.1328

برای مثال رقم شماره ۵ با سیگنال زیر بیان می‌شود:

$$d_5[n] = \sin(0.5906n) + \sin(1.0247n)$$

الف) بردارهای d0 تا d9 را که هر کدام صدای یکی از ۱۰ کلید را نشان می‌دهند، برای $0 \leq n \leq 999$ بسازید. این سیگنال‌ها را با فرکانس نمونه‌برداری گفته شده در فایل‌های wav. ذخیره کنید و به صدای آن‌ها گوش کنید. (باید صدایی شبیه به صدای کلیدهای تلفن بشنوید!)

ب) با استفاده از تابع fft سیگنال d0 و d9 را در حوزه فرکانس نمایش دهید تا فرکانس‌های مورد استفاده برای ۱ و ۹ را مشاهده کنید.

پ) یک بردار به طول ۱۰۰ با نام space و با استفاده از تابع zeros بسازید. سیگنال phone را به صورتی که هفت رقم آخر شماره دانشجویی شما را بیان کند، به صورت زیر بسازید.(برای مثال ۰۱۹۴۸۶۲):

```
phone = [d0 space d1 space d9 space d4 space d8 space d6 space d2 space]
```

برای این کار می‌توانید از تابع append یا concatenate استفاده کنید. این سیگنال را نیز در یک فایل wav. ذخیره کنید و به آن گوش دهید. (صدایی شبیه به شماره‌گیری می‌شنوید.)

ت) در این قسمت باید از روی سیگنال داده شده، شماره گرفته شده را بدست آورید. ابتدا فایل‌های phone1.csv، phone2.csv، hard_phone1.csv و hard_phone2.csv را در برنامه خود بخوانید.

در دو سیگنال خوانده شده از فایل‌های phone1.csv یعنی X1 و از فایل phone2.csv یعنی X2 ، فرض کنید طول شماره گرفته شده ۷ است و از فرمت قسمت قبل برای سکوت‌های بین هر شماره استفاده شده است. (یعنی هر ۱۰۰۰ نمونه از سیگنال کلید فشار داده شده و ۱۰۰ نمونه صفر به عنوان سکوت). پس سیگنال حوزه زمان مربوط به

هر رقم شماره را جدا کنید و با استفاده از fft آن‌ها را در حوزه فرکانس نمایش دهید. از روی نمودارهای رسم شده، شماره گرفته شده در این دو فایل را بدست آورید. برای بررسی درست بودن جواب، بدانید که مجموع کلیدهای فشرده شده ۴۱ می‌شود.

ث) در این قسمت باید تابعی بنویسید که این کار را به صورت اتوماتیک برای دو فایل phone1.csv و phone2.csv انجام دهد. برای مثال اگر شماره گرفته شده ۵۵۵۷۳۱۹ باشد، نحوه کارکرد تابع شما باید به صورت زیر باشد:

```
testout = ttdecode(phone)
print(testout)
5 5 5 7 3 1 9
```

(امتیازی) بسیاری از افراد همانند فرمت داده شده با دقت یکسان کلیدهای تلفن را فشار نمی‌دهند. در این قسمت باید بتوانید شماره گرفته شده در فایل‌های hard_phone1.csv و hard_phone2.csv را بدست آورید. در این فایل‌ها الگوی مشخصی برای مدت زمان فشار کلید و مدت زمان سکوت بین کلیدها وجود ندارد. برای راحتی فرض کنید مدت زمان فشار هر کلید و سکوت بین آن‌ها کمتر از ۱۰۰ نمونه نیست. شماره گرفته شده در این فایل‌ها همان شماره‌های دو فایل قبلی هستند.

به نکات زیر توجه کنید:
۱) تمامی کدها باید به زبان برنامه‌نویسی python باشند.
۲) گزارش کار شما در روند تصحیح از اهمیت ویژه‌ای برخوردار است. لطفا تمامی نکات و محاسبات خود را به صورت دقیق در گزارش خود ذکر کنید.
۳) فایل pdf مربوط به گزارش را به همراه کدهای پایتون مربوطه در یک فایل zip با عنوان "نام و شماره دانشجویی" خود بنویسید و upload کنید.
۴) در صورت هر گونه تقلب نمرات تمامی افراد شرکت کننده در آن صفر تلقی خواهد شد.
استفاده از کدهای آماده مجاز نمی‌باشد. مگر اینکه صورت سوال به صورت واضح استفاده از کد را بلامانع اعلام کرده باشد.
۵) در صورت وجود هرگونه ابهام و سوال با دستیار آموزشی مربوطه در تماس باشید:

a.vosoughi99@gmail.com