حذف نویز از سیگنال صوتی

تعدادی فایل در کنار این فایل قرار دارد. به فایل صوتی `soundCA2.wav` گوش دهید. مشاهده خواهید کرد که فایل صوتی دارای نویز است. در این تمرین قصد داریم این نویز را از صوت حذف کنیم. ابتدا فایل صوتی را در برنامه خود بخوانید. برای این کار از `wavfile` موجود در `scipy.io` استفاده کنید.

برای حذف نویز ابتدا نیاز است تا فرکانس مربوط به آن را مشخص کنیم. برای این کار از `fft` استفاده کنید. تابع `fft` برای محاسبه n نقطه از DTFT سیگنال گسسته-زمان با طول محدود در فرکانس های $\omega_k = 2\pi k/N$ استفاده می‌شود. برای محاسبه `fft` یک سیگنال و رسم آن، می‌توانید از قطعه کد زیر کمک بگیرید. (سعی کنید کارکرد قسمت های مختلف آن را متوجه شوید تا بتوانید در تمرین های مختلف تغییرات لازم را انجام دهید.)

```
import numpy as np
from scipy.io import wavfile
from numpy import fft
from matplotlib import pyplot as plt

def nextpow2(x):
    return (x - 1).bit_length()

# x is a row vector that represents your finite-length signal
L = len(x)
print('L =', L, sep=' ')
NFFT = 2 ** nextpow2(L) # Next power of 2 from length of x
X = fft.fft(x, n=NFFT)
X_abs = 2 * np.absolute(X) / L
half = int(NFFT/2)
freq = fft.fftfreq(NFFT, d=1/fs)

fig, ax = plt.subplots(1, 1)
ax.plot(freq[:half], X_abs[:half])
ax.set_title('Single-Sided Amplitude Spectrum of x(t)')
ax.set_xlabel('Frequency [Hz]')
ax.set_ylabel('|X(f)|')
ax.grid()
plt.show()
```

الف) با رسم نمودار `fft` فایل صوتی `soundCA2.wav`، فرکانس نویز را مشخص کنید. این فرکانس و نمودار را در گزارش خود ثبت کنید.

در این مرحله باید از یک فیلتر میان نگذر برای حذف نویز استفاده کنیم. برای بدست آوردن ضرایب این فیلتر از قطعه کد زیر کمک بگیرید:

```
from scipy import signal

def butter_bandstop_filter(lowcut, highcut, fs, order=5):
    nyq = 0.5 * fs
    low = lowcut / nyq
    high = highcut / nyq
    b, a = signal.butter(N=order, Wn=[low, high], btype='bandstop')
    return b, a
```

برای استفاده از تابع بالا باید سه ورودی آن را تنظیم کنید. دو ورودی اول را بر اساس فرکانس نویز که در قسمت قبل بدست آوردید، انتخاب کنید و ورودی سوم یعنی فرکانس نمونه برداری را هنگام خواندن فایل صوتی از تابع `wavfile.read` خروجی می‌گیرید.

ب) اندازه فیلتر بالا را با استفاده از `freqz` رسم کنید و برای اعمال آن بر روی سیگنال دارای نویز از `lfilter` استفاده کنید. با جزئیات توابع لازم در تمرین کامپیوتری اول آشنا شده اید.

پ) نمایش حوزه فرکانس سیگنال صوتی را بعد حذف نویز آن با کمک `fft` نشان دهید و از حذف شدن نویز مطمئن شوید. حال باید سیگنال خود را در یک فایل صوتی به اسم `noiseless.wav` ذخیره کنید.

صدای کلید های تلفن

در این تمرین با چگونگی استفاده از فرکانس های مختلف برای تشخیص کلید فشار داده شده در تلفن آشنا می شوید. صدایی که هنگام فشار دادن یک کلید می شنوید، 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 ترتیب فرکانس های استفاده شده برای کلید های مختلف را مشاهده می‌کنید. همچنین در کل این تمرین فرض کنید فرکانس نمونه برداری $[Hz]$ 8192 است و از `fft` با 2048 نقطه استفاده می‌کنیم.

	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 5 با سیگنال زیر بیان می‌شود:

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

الف) بردار های `d0` تا `d9` را که هر کدام صدای یکی از ۱۰ کلید را نشان می‌دهند، برای $0 \le n \le 999$ بسازید. این سیگنال ها را با فرکانس نمونه برداری گفته شده در فایل های `.wav` ذخیره کنید و به صدای آن ها گوش کنید. (باید صدایی شبیه به صدای کلید های تلفن بشنوید!)

ب) با استفاده از تابع `fft` سیگنال `d1` و `d9` را در حوزه فرکانس نمایش دهید تا فرکانس های مورد استفاده برای 1 و 9 را مشاهده کنید.

پ) یک بردار به طول 100 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. (حال صوتی شبیه شماره گیری می‌شنوید!)

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

در دو سیگنال خوانده شده از فایل های `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 سکوت های بین هر شماره استفاده شده است (یعنی 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` آن ها را در حوزه فرکانس نمایش دهید. از روی نمودار های رسم شده، شماره گرفته شده در این **دو فایل** را بدست آورید. برای بررسی درست بودن جواب، بدانید که مجموع کلید های 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 کلید های تلفن را فشار نمی‌دهند. در این قسمت باید بتوانید شماره گرفته شده در فایل های hard_phone1.csv و hard_phone2.csv بدست آورید. در این فایل ها الگوی مشخصی برای مدت زمان فشار کلید و مدت زمان سکوت بین کلید ها وجود ندارد. برای راحتی فرض کنید مدت زمان فشار هر کلید و سکوت بین آن ها کمتر از ۱۰۰ نمونه نیست. شماره گرفته شده در این فایل ها همان شماره های دو فایل قبلی هستند.

نمونه برداری (فرکانس نمونه برداری – درهم‌روی)

سیگنال سینوسی زیر را در نظر بگیرید.

$$x(t) = sin(\Omega_0 t)$$

اگر $x(t)$ با فرکانس $\Omega_s = 2\pi/T$ نمونه‌برداری شود، سیگنال گسسته-زمان $x[n]$ برابر است با:

$$x[n] = sin(\Omega_0 nT)$$

الف) سیگنال $x(t)$ را به ازای $\Omega_0 = 2\pi(20)$ در بازه $0 \leq t < 1$ رسم کنید. از آن‌جایی که نمی‌توان سیگنال پیوسته-زمان داشت، `t` را به شکل `t=numpy.linspace(0, 1, num=1000, endpoint=False)` تعریف کنید. حال با فرکانس $\Omega_s = 2\pi(25)$ از سیگنال $x(t)$ نمونه‌بردرای کنید. سیگنال $x(t)$ را با `plot` و سیگنال نمونه‌برداری شده را با `stem` در یک نمودار رسم کنید.

ب) قسمت الف را برای $\Omega_s = 25(50)$ تکرار کنید. مشخص کنید در کدام حالت `aliasing` رخ می‌دهد.

پ) با استفاده از سیگنال نمونه‌برداری شده، می‌خواهیم سیگنال اصلی را بازسازی کنیم. در درس دیدیم که برای این کار لازم است، سیگنال پیوسته-زمان $x_p(t)$ را تشکیل داده و آن را از یک فیلتر پایین گذر عبور دهیم.

$$x_p(t) = \sum_{n=-\infty}^{+\infty} x[n]\delta(n - nT)$$

این کار معادله استفاده از رابطه زیر است. که در آن $x_r(t)$ همان سیگنال بازسازی شده است.

$$x_r(t) = \sum_{n=-\infty}^{+\infty} \frac{x[n] \sin\left(\frac{\Omega_s(t - nT)}{2}\right)}{\frac{\Omega_s(t - nT)}{2}}$$

برای هر کدام از سیگنال‌های نمونه‌بردرای شده در قسمت‌های قبل، سیگنال بازسازی شده از آن را به‌دست‌آورده و رسم کنید. برای استفاده از رابطه بالا `t` را همانگونه که در قسمت الف تعریف شد استفاده کنید.

کم نمونه برداری

در این سوال می‌خواهیم یک فایل صوتی با فرمت wav را بخوانیم و تاثیر downsampling را برروی کیفیت صوت و حجم فایل ببینیم.

الف) فایل sound.wav را به کمک دستور wavfile.read بخوانید. fs فرکانس نمونه‌برداری است که نشان‌دهنده تعداد نمونه‌ها در ثانیه است. ابعاد x را مشخص کرده و بیان کنید به کمک آن چگونه می‌توان حجم فایل را به دست آورد.

```
from scipy.io import wavfile
fs, x = wavfile.read('./sound.wav')
```

ب) همانطور که در بخش قبل دیدیم، نمونه‌های صوت دارای دو ستون هستند که هر کدام مربوط به صدای سمت راست و چپ است. برای سادگی کار با صوت معمولا میانگین این دو را در نظر می‌گیرند. توجه کنید که بعد از میانگین گرفتن data type سیگنال حاصل را به int16 تبدیل کنید. به کمک تابع fft تبدیل فوریه سیگنال صوتی را رسم کنید.

پ) یکی از روش‌های کم کردن حجم فایل downsample کردن نمونه‌های سیگنال است. سیگنال صوتی به دست آمده در قسمت قبل را با N=3، downsample کنید. برای این منظور تنها کافی است از هر سه نمونه متوالی، نمونه اول را انتخاب کنید.

ت) سیگنال downsample شده را در فایل result.wav ذخیره کنید. برای ذخیره این فایل چه fs ای باید انتخاب کرد؟ حجم فایل چقدر کاهش می‌یابد؟ همچنین به فایل ذخیره شده گوش دهید تا ببنید چه میزان کیفیت آن کاهش پیدا کرده است.

```
wavfile.write('./result.wav', fs, downsampled_x)
```

ث) همانند قسمت ب، تبدیل فوریه سیگنال downsample شده را رسم کنید. با مقایسه با قسمت ب، مشخص کنید چه اتفاقی افتاده است.

نکات تحویل

گزارش خود را تا حد امکان کامل بنویسید. در فایل گزارش باید تمامی نمودار ها و توضیحات خواسته شده موجود باشند. (تمام نمودار ها باید برچسب های مناسب داشته باشند.)

از ارجاع به کد جدا خودداری فرمایید.

کل نمره این تمرین از چند قسمت تشکیل می‌شود که بخشی از آن به کد های نوشته شده و بخش دیگری به گزارش نوشته شده مربوط می‌شود. **این تمرین تحویل حضوری ندارد.** پس در صورت ناقص بودن هر بخش (کد یا گزارش) نمره آن کسر خواهد شد.

نوشتن کد ها در `jupyter notebook` و گزارش در کنار آن، در صورت کامل بودن نمره امتیازی دارد و در صورت ناقص بودن توضیحات نمره امتیازی تعلق نمی‌گیرد و نمره بخش های ناقص کسر خواهد شد.