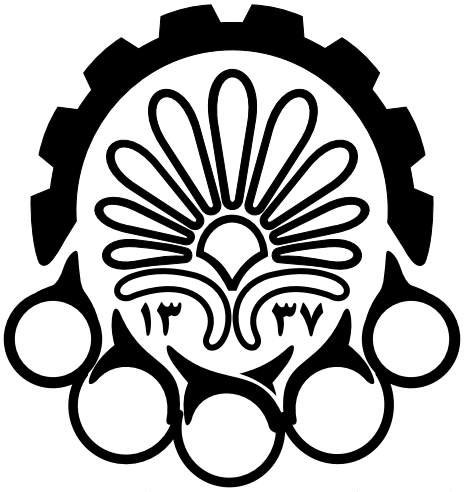
**به نام خدا**




# دانشگاه صنعتی امیرکبیر

## ( پلی تکنیک تهران )

**دانشکده مهندسی کامپیوتر**

## سیگنال‌ها و سیستم‌ها

## تمرین عملی چهارم: سیستم شناسایی موسیقی

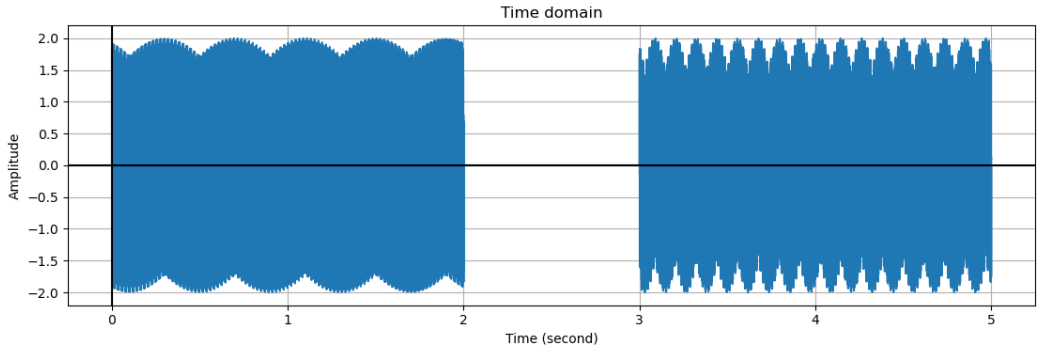
توضیحات:

- مهلت تحویل تا پنجشنبه ۲۷ خرداد در نظر گرفته شده است. لذا با توجه به سنگین‌تر بودن این تمرین عملی نسبت به تمرین‌های گذشته، برنامه‌ریزی مناسبی انجام دهید.
- پاسخ به تمرین باید به صورت انفرادی داده شود و در صورت مشاهده هرگونه تقلب نمره برای همه افراد صفر در نظر گرفته خواهد شد.
- کد و گزارش (شامل توضیح مختصر مراحل انجام کار و نمودار قسمت‌های مختلف) تمرین را در قالب یک فایل ZIP با نام «**PA4_StudentNumber.zip**» در سایت درس بارگذاری کنید.
- در صورت داشتن اشکال می‌توانید از طریق ایمیل ss.spring.2021@gmail.com با تدریس‌یاران درس در ارتباط باشید.

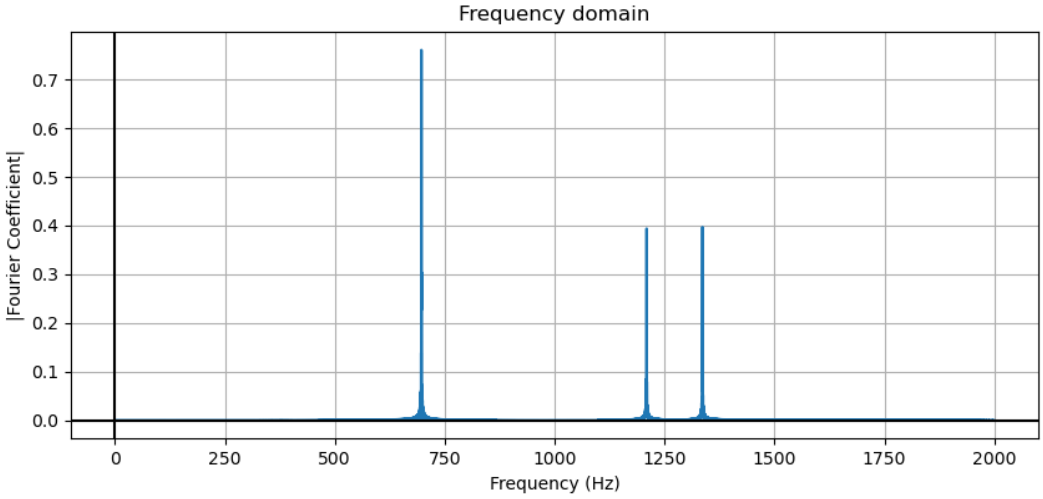
**نیمسال دوم ۰۰-۹۹**

به احتمال زیاد با سیستم‌های شناسایی موسیقی (مانند Shazam) آشنایی دارید. جالب است بدانید عمل تشخیص موسیقی در این سیستم‌ها به کمک تبدیل فوریه انجام می‌شود. در این تمرین می‌خواهیم چنین سیستمی را خودمان پیاده‌سازی کنیم!

در درس دیدیم که با استفاده از تبدیل فوریه می‌توانیم سیگنال‌ها را از حوزه زمان به حوزه فرکانس ببریم. با اینکار فرکانس‌های تشکیل‌دهنده یک سیگنال (که در حوزه زمان قابل شناسایی نبود) مشخص می‌شوند. به عنوان مثال سیگنال صوتی زیر را در حوزه زمان در نظر بگیرید. فرض کنید می‌خواهیم اجزای تشکیل‌دهنده این سیگنال را به دست آوریم.

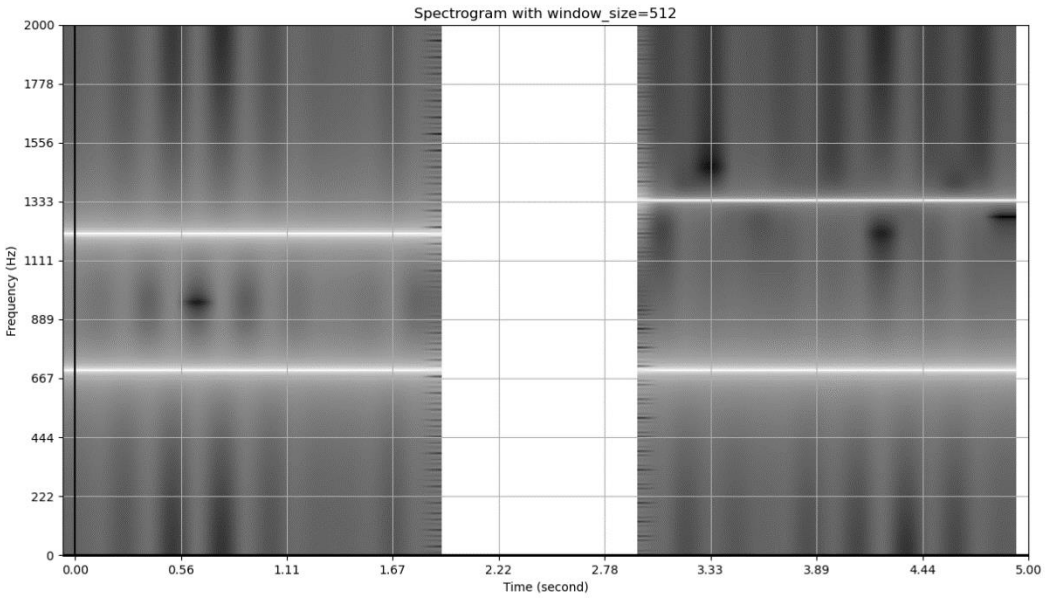



همانطور که می‌بینید نمودار بالا اطلاعات کاملی در خصوص تغییرات سیگنال در حوزه زمان می‌دهد. به عنوان مثال متوجه می‌شویم که این صوت بین ثانیه ۲ تا ۳ ساکت است. اما این نمودار در مورد فرکانس‌های موجود در سایر زمان‌ها هیچ‌گونه اطلاعاتی به ما نمی‌دهد. با محاسبه و رسم تبدیل فوریه به کمک تابع fft در numpy نمودار زیر به دست می‌آید.



این نمودار اطلاعات کاملی از فرکانس‌های موجود به ما می‌دهد. به عنوان مثال متوجه می‌شویم که فرکانس‌هایی حول ۷۵۰ و ۱۲۵۰ هرتز سیگنال ما را تشکیل می‌دهند. اما نمودار بالا هیچگونه اطلاعاتی در مورد این که هر یک از این فرکانس‌ها در چه زمان‌هایی حضور داشته‌اند به ما نمی‌دهد.

برای به دست آوردن این اطلاعات، نیاز به نموداری داریم که حالتی بینابین حوزه زمان و فرکانس را به ما بدهد. فرض کنید سیگنال داده شده را به پنجره‌های زمانی کوتاه‌تر (مثلا ۱۰۰ میلی‌ثانیه) و بدون اشتراک می‌شکنیم و برای هر پنجره زمانی به طور مستقل تبدیل فوریه آن را حساب می‌کنیم. با کنار هم گذاشتن این تبدیل فوریه‌ها، به نمودار فرکانس بر حسب زمان می‌رسیم که به آن طیف‌نگاره یا spectrogram سیگنال می‌گویند. به عنوان مثال برای سیگنال ذکر شده، spectrogram آن به شکل زیر می‌شود (نقاط روشن‌تر به معنی بزرگ‌تر بودن مقدار تبدیل فوریه در آن نقطه است).



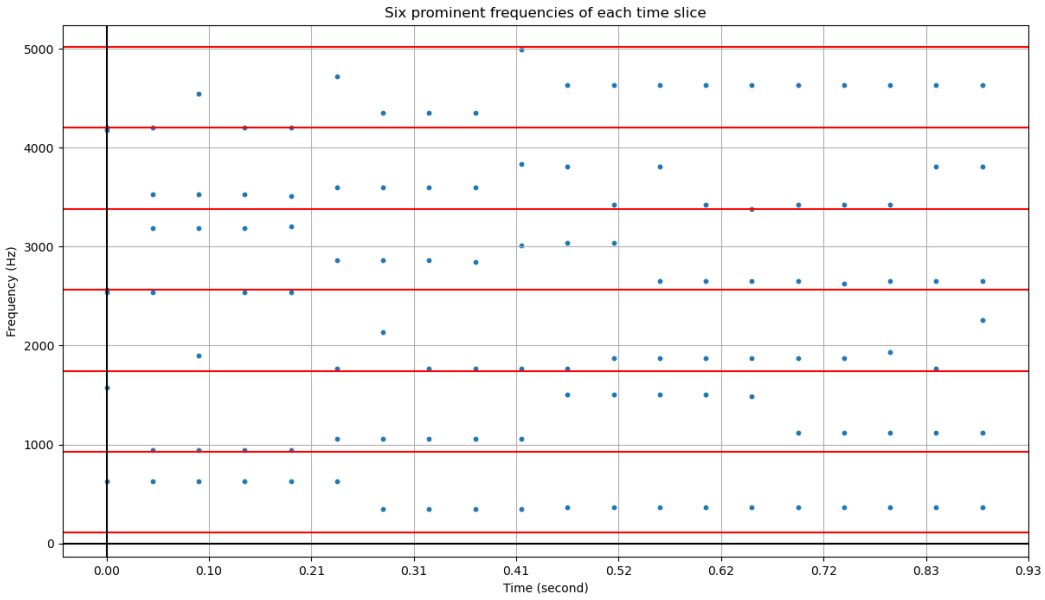

حال از روی این نمودار به خوبی مشخص است که در ۲ ثانیه اول فرکانس‌های حدود ۶۶۶ و ۱۲۰۰ هرتز و در ۲ ثانیه پایانی فرکانس‌های حدود ۶۶۶ و ۱۳۳۳ هرتز تشکیل‌دهنده سیگنال ما بودند. نتیجه‌ای که از هیچ یک از نمودارهای حوزه زمان و فرکانس قابل دست‌یابی نبود.

برای ساخت سیستم شناسایی موسیقی نیز، نیاز به مقایسه spectrogram سیگنال‌ها داریم. در این تمرین دو مجموعه فایل صوتی در اختیارتان قرار دارد. مجموعه data شامل نسخه اصلی چند موسیقی است و صوتی که کاربر ورودی می‌دهد باید با تمام اعضای این مجموعه مقایسه شود. مجموعه clip شامل چند نمونه از ورودی‌های کاربران است که می‌توانید برای تست کد خود از آن‌ها استفاده کنید. برای مقایسه سیگنال‌ها مراحل زیر را برای هر یک از سیگنال‌ها طی کنید:

۱. سیگنال را به پنجره‌های ۲۰۴۸ سمپلی (حدود ۵۰ میلی‌ثانیه) بشکنید و از هر پنجره زمانی به طور جداگانه با استفاده از numpy تبدیل فوریه بگیرید. می‌توانید این تبدیل فوریه‌ها را در قالب یک ماتریس ذخیره کنید که هر ستون آن خروجی تبدیل فوریه برای یک پنجره زمانی است.

۲. اطلاعات فرکانس‌های حدود ۱۰۰ تا ۵۰۰۰ هرتز را نگه دارید و بقیه را حذف کنید (برای به دست آوردن فرکانس متناظر با هر عنصر خروجی fft می‌توانید از fftfreq استفاده کنید).

۳. فرکانس‌های هر پنجره زمانی را به ۶ سبد با اندازه‌های مساوی تقسیم کنید و فرکانسی که در هر سبد مقدار بیشینه دارد را به عنوان فرکانس شاخص آن سبد در نظر بگیرید. برای درک بهتر این فرایند شکل زیر را به عنوان نمونه در نظر بگیرید. در این نمودار فرکانس شاخص هر سبد در هر پنجره زمانی با نقطه مشخص شده است.



پس از طی کردن مراحل بالا برای هر سیگنال یک ماتریس با ۶ سطر داریم که هر سطر آن نشان‌دهنده فرکانس شاخص در یکی از سبدها و هر ستون آن نشان‌دهنده ۶ فرکانس شاخص هر پنجره زمانی است. در اینجا به ماتریس حاصل noiseprint یک سیگنال می‌گوییم.

حال برای شناسایی یک موسیقی ورودی، کافیست noiseprint آن را با noiseprint موسیقی‌هایی که پیشتر در دیتابیس خود (مجموعه data) محاسبه کردیم مقایسه کنیم. برای اینکار می‌توانید از تابع زیر استفاده کنید.

```
def similarity(song_spec, clip_spec, points_per_slice=6):
    song_flat = song_spec.flatten()
    clip_flat = clip_spec.flatten()

    sim_window_size = points_per_slice - 1
    score = 0
    for anchor in range(clip_flat.shape[0] - points_per_slice):
        anchor_y = anchor % points_per_slice
        sim_window = clip_flat[anchor: anchor+sim_window_size]
        for song_anchor in range(anchor_y, song_flat.shape[0] -
points_per_slice - 1, points_per_slice):
            if clip_flat[anchor] == song_flat[song_anchor]:
                if
np.count_nonzero((song_flat[song_anchor:song_anchor+sim_window_size] -
sim_window) == 0) >= 4:
                    score += 1

    score /= song_flat.shape[0]
    return score
```

این تابع با دریافت noiseprint سیگنال موجود در دیتابیس در ورودی اول و noiseprint سیگنال داده شده توسط کاربر در ورودی دوم، میزان شباهت آن‌ها را در قالب یک عدد برمی‌گرداند. در نهایت هر سیگنالی از دیتابیس که شباهت بیشتری با ورودی کاربر داشت، به عنوان موسیقی شناسایی شده انتخاب می‌شود (ارائه الگوریتمی به جز الگوریتم داده شده برای محاسبه شباهت که دقت مناسبی داشته باشد نمره **امتیازی** دارد).

- ✓ سعی کنید پس از انجام هر یک از مراحل ذکر شده، با چاپ کردن ماتریس حاصل و کشیدن نمودار، درستی محاسبات خود را بررسی کنید. همچنین استفاده از **ژوپیتر** برای نوشتن و بررسی مرحله به مرحله کد بسیار توصیه می‌شود.

موفق باشید