گزارش قسمت اول پروژه درس پردازش زبان و گفتار

محمد مهدی امیرپور

شماره دانشجویی: ۴۰۰۰۳۰۳۳

40003033

محمد مهدی امیرپور

گزارش کامل قسمت اول

پروژه پایانی درس SNLP

---

① مقدمه: هدف این پروژه، تحلیل سیگنال گفتار با استفاده از روش کپستروم است، با هدف استخراج دو ویژگی اصلی:

- فرکانس گام (Pitch Frequency)
- فرمنت های گفتار (Formants)

روش کپستروم به ما این امکان را میدهد که منابع تحریک (tension of vocal cords) و پاسخ فیلتر مجرای گفتار (vocal tract) را به طور جداگانه تحلیل کنیم.

② جمله ی انتخاب شده:

Speech signals are composed of a sequence of sounds, which, for purposes of analysis and study, are assumed to be physical realizations of a discrete set of symbols.

– Chapter 3, part 3.2, Page 68

• ترجمه ی جمله ی انتخاب شده:

سیگنال های گفتاری از دنباله ای از صداها تشکیل شده اند که برای اهداف تحلیل و مطالعه، فرض می شود که تجسم های فیزیکی یک مجموعه گسسته از نماد ها هستند.

– فصل 3، قسمت 3.2، صفحه ی 68

③ پیش زمینه ی تئوری: تحلیل کپستروم

در مدل تولید گفتار: $h(n) * e(n) = s(n)$

که در آن:

$e(n)$: سیگنال منبع (منبع تحریک - تحریک تارهای صوتی)

$h(n)$: پاسخ سیستم مجرای گفتاری

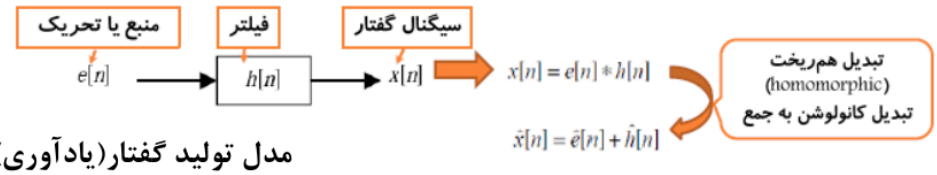
در حوزه ی فرکانس: $H(\omega) \cdot E(\omega) = S(\omega)$

با گرفتن لگاریتم: $\log|H(\omega)| + \log|E(\omega)| = \log|S(\omega)|$

و سپس: $IFFT(\log|S(\omega)|) = c(n)$

Lect14, page2

تحلیل کپستروم



مدل تولید گفتار(یادآوری)

- **در این روش هدف این است که فیلتر مجرا(h) و منبع تحریک(e) را با جدا کردن از یکدیگر به دست آورد.**
- **با داشتن یک تبدیل هم ریخت مناسب که کانولوشن را به جمع تبدیل می‌کند، می‌توان به این خواسته دست یافت.**
- **پس با داشتن رابطه درج شده در شکل بالا می‌توان فیلتر(h) و منبع(e) را به دست آورد.**
- **کپستروم(Cepstrum) موجب تبدیل کانولوشن به جمع می‌شود.**

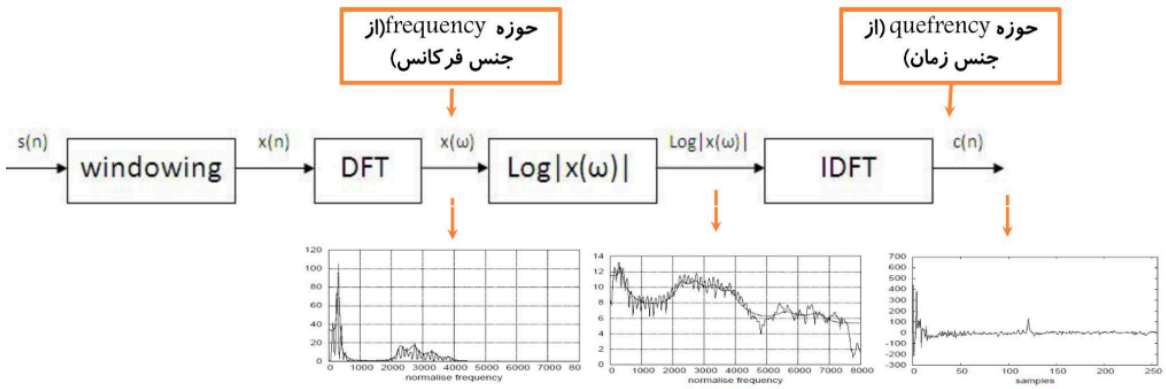
Lect14, page 3

ضرایب کپستروم

- **معکوس تبدیل فوریه لگاریتم دامنه تبدیل فوریه**

$$c[n] = \mathcal{F}^{-1}\{\log|\mathcal{F}\{x[n]\}|\}$$

$$c[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} \ln|X(e^{j\omega})|\, e^{j\omega n} d\omega$$



Lect14, page3

جدا کردن بخش فیلتر(مدل مجرای گفتار_h)

- **لیفتر کردن(liftering)**
- **جدا کردن فیلتر مجرای گفتار(h) با استفاده از فیلتر در حوزه کپستروم(لیفتر)**
- **لیفتر کردن زمان پایین (low-time liftering) برای جدا کردن h**
- **یعنی بخش پایین و ابتدایی کپستروم(h) را نگه می‌داریم.**

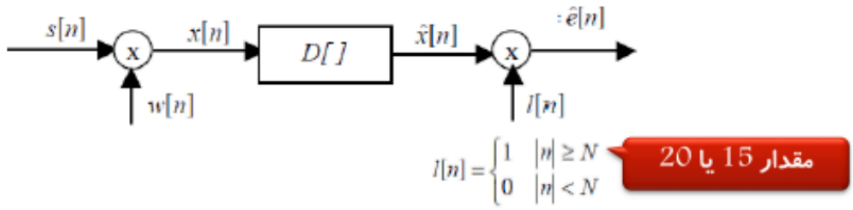
$$l[n] = \begin{cases} 1 & |n| < N \\ 0 & |n| \geq N \end{cases}$$

مقدار 15 یا 20

lect14, page 5

جدا کردن بخش سیگنال تحریک(e)

- **لیفتر کردن(liftering)**
- **جدا کردن منبع تحریک(e) با استفاده از فیلتر در حوزه کپستروم(لیفتر)**
- **لیفتر کردن زمان بالا (high-time liftering) برای جدا کردن e**
- **یعنی بخش بالایی و انتهایی کپستروم (e) را نگه می‌داریم.**



در اینجا، دامنه‌ی کپستروم $c(n)$ به ما اجازه می‌دهد تا با استفاده از لیفتر کردن (liftering)، بین بخش‌های مربوط به فیلتر و منبع تفکیک انجام دهیم:

- low-time frequency ← فرمنت ها
- high-time frequency ← پیچ

④ مراحل کد نویسی و تحلیل:

گام 1- لود کردن فایل صوتی ضبط شده با صدای خود:

- sf.read سیگنال و نرخ نمونه برداری را بارگذاری می‌کند.
- اگر سیگنال دو کانال باشد، فقط یک کانال استفاده می‌شود.

```python
import numpy as np
import matplotlib.pyplot as plt
import soundfile as sf
from scipy.signal import get_window
from scipy.fftpack import fft, ifft
from numpy.linalg import solve

# Load recorded sentence
signal, sr = sf.read("data/snlp-project-input.wav")
if len(signal.shape) > 1:
    signal = signal[:, 0]  # if stereo, keep one channel
```

گام 2- Framing و پنجره بندی :

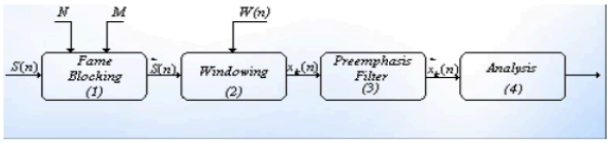
• چرا framing ؟

- گفتار، سیگنال غیرایستا (non_stationary) است.

- برای تحلیل دقیق، آن را به قطعات کوچکی که تقریباً ایستا هستند تقسیم می کنیم.

پیش‌پردازش

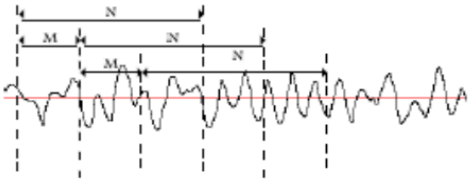
- **آنالیز و استخراج ویژگی، استخراج مشخصاتی از گفتار است که برای هدف مورد نظر در پردازش گفتار مفید هستند.**
- **قبل از آنالیز و استخراج ویژگی لازم است که پیش‌پردازش‌هایی بر روی سیگنال گفتار صورت گیرد:**
  - ❑ فریم‌بندی (**Frame Blocking**)
  - ❑ اعمال پنجره (**Windowing**)
  - ❑ پیش‌تاکید (**Preemphasis**)



2

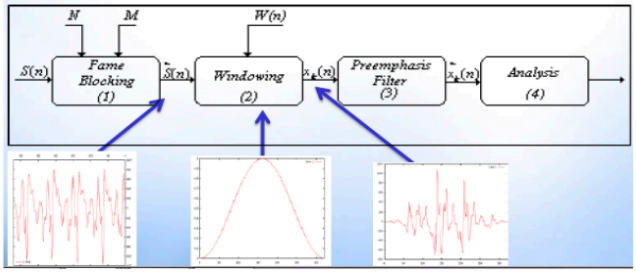
پیش‌پردازش- فریم‌بندی

- فریم‌بندی (**Frame Blocking**)
  - به دلیل غیرایستان بودن سیگنال گفتار، این سیگنال به صورت زمان کوتاه و در بخش‌های ۲۰ - ۳۰ میلی‌ثانیه از سیگنال به نام فریم مورد آنالیز قرار می‌گیرد.
  - سیگنال به فریم‌هایی تقسیم می‌شود که با هم همپوشانی دارند.
  - **N** بیانگر تعداد نمونه‌هایی است که می‌خواهیم در یک فریم وجود داشته باشد و **M** تعداد نمونه‌هایی است که فریم‌های مجاور را از هم جدا می‌کند.



پیش‌پردازش- پنجره‌گذاری(windowing)

- پنجره‌گذاری
  - ضرب کردن یک تابع پنجره در نمونه‌های سیگنال یک فریم
  - هر فریم را به طور مجزا در یک پنجره ضرب می‌کنیم.
  - شکل زیر یک فریم را قبل و بعد از پنجره‌گذاری نشان می‌دهد.
  - نتیجه اعمال پنجره، متناظر با کانولوشن طیفهای فریم و پنجره در حوزه فرکانس می‌باشد.



در این شکل **N** طول فریم یا پنجره است و **M** اندازه شیفت فریم است.

پنجره‌های معمول مورد استفاده

1. Rectangular $w[n]=\begin{cases}1 & 0\le n\le M\\ 0 & otherwise\end{cases}$

2. Bartlett $w[n]=1-\dfrac{2|n-M/2|}{M}$

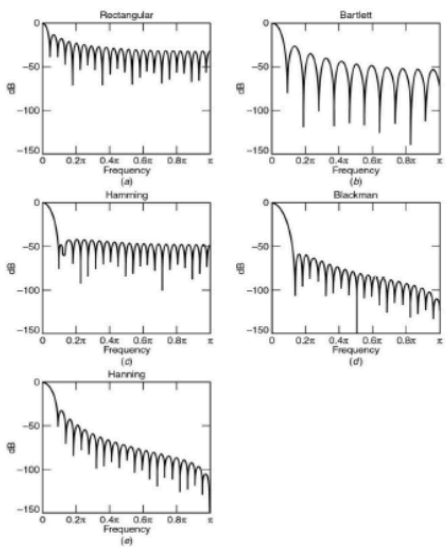
3. Blackman $w[n]=0.42-0.5\cos\left(\dfrac{2\pi n}{M}\right)+0.08\cos\left(\dfrac{4\pi n}{M}\right)$

4. Hamming $w[n]=0.54-0.46\cos\left(\dfrac{2\pi n}{M}\right)$

5. Hanning $w[n]=0.5-0.5\cos\left(\dfrac{2\pi n}{M}\right)$

6. Kaiser $w[n]=\dfrac{I_0\left\{\beta\sqrt{1-((n-M/2)/(M/2))^2}\right\}}{I_0\{\beta\}}$

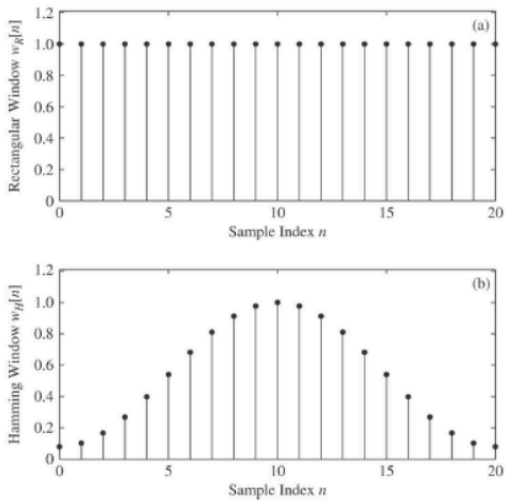
در این روابط ***M*** ***طول پنجره*** است.



Window	Mainlobe Width	Sidelobe Attenuation
Rectangular	$4\pi/M$	-13 dB
Bartlett	$8\pi/M$	-27 dB
Hanning	$8\pi/M$	-32 dB
Hamming	$8\pi/M$	-43 dB
Blackman	$12\pi/M$	-58 dB

- ***M*** ***طول پنجره*** است.
- پهنای باند لوب اصلی پنجره قدرت تفکیک را تعیین می‌کند. (پهنای باند کمتر، قدرت تفکیک بیشتر و رزولوشن فرکانسی بالاتر)
- نسبت (تضعیف) دامنه لوبهای فرعی(کناری) به لوب اصلی، میزان نشتی را تعیین می‌کند.(تضعیف بیشتر، نشتی کمتر)

پنجره‌های معمول مورد استفاده(ادامه)

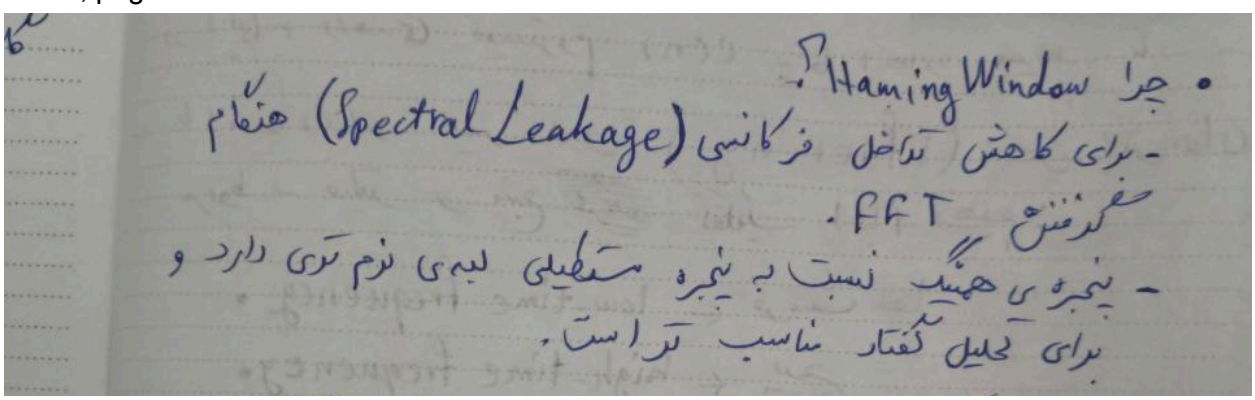


- پنجره های مربعی و همینگ (حوزه زمان)

$L = 21$ samples

L ***طول پنجره*** است.

Lect8, page 7

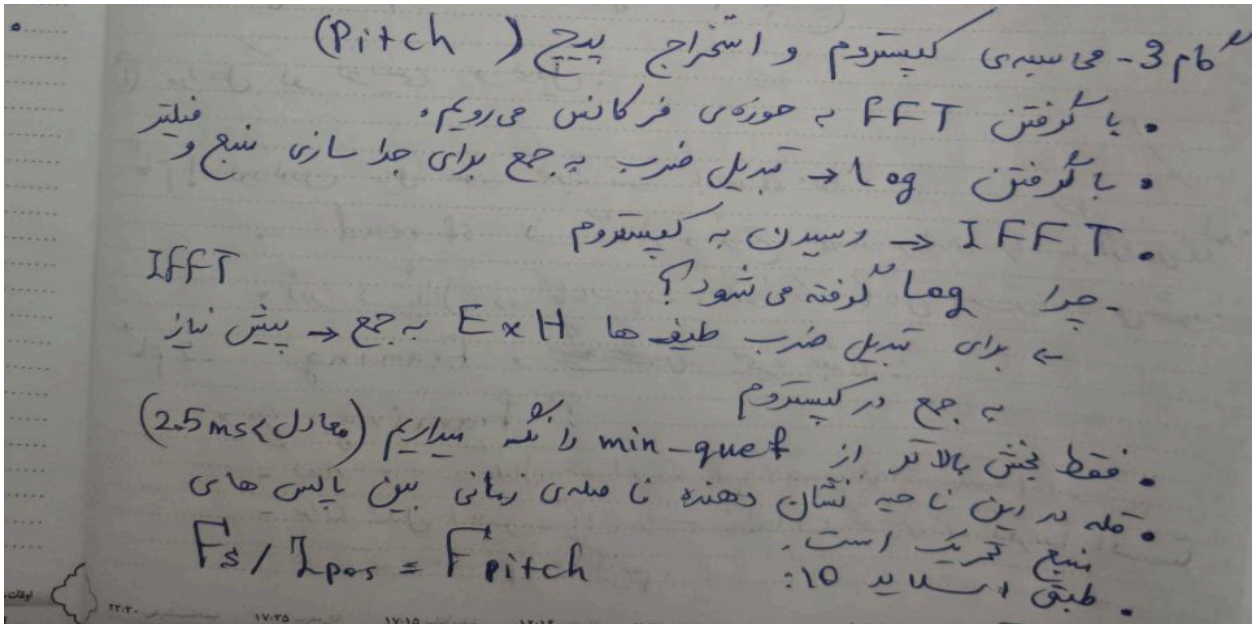


• چرا Haming Window؟

- برای کاهش تداخل فرکانسی (Spectral Leakage) هنگام گرفتن FFT.

- پنجره‌ی همینگ نسبت به پنجره مستطیلی لبه‌ی نرم‌تری دارد و برای تحلیل گفتار مناسب‌تر است.

```
# Frame parameters
frame_length = int(0.03 * sr)  # 30ms
hop_length = int(0.015 * sr)   # 15ms
window = get_window("hamming", frame_length)
```



برای به دست آوردن فرکانس گام:

- **رسم نمودار ضرایب کپسترال بر حسب n**
- **یافتن فاصله اولین قله I_{pos}**
- $F_{pitch} = \frac{F_s}{I_{pos}}$
- **عدم وجود یک قله کاملا مشخص، می‌تواند نشان از یک فریم بی‌واک باشد .**
- **در زمانی که تردید وجود داشته باشد می‌توان از اطلاعات دیگر مثل نرخ عبور از صفر استفاده کرد.**

```python
# Frame-by-frame analysis
for i in range(0, len(signal) - frame_length, hop_length):
    frame = signal[i:i + frame_length]
    frame = frame * window

    # --------------- PITCH (via high-time liftering in cepstrum)
    spectrum = fft(frame)
    log_magnitude = np.log(np.abs(spectrum) + 1e-10)
    cepstrum = np.real(ifft(log_magnitude))

    high_liftered = np.copy(cepstrum)
    high_liftered[:min_quef] = 0
    high_liftered[max_quef:] = 0

    peak_index = np.argmax(high_liftered)
    pitch = sr / peak_index if peak_index > 0 else 0
    pitch_values.append(pitch)
```

گام 4- استخراج فرمنت ها با روش LPC:

• تئوری LPC چیست؟

- فرض: هر نمونه از سیگنال گفتار $s(n)$ را می توان از ترکیب خطی P نمونه ی قبلی تقریب زد:

$$S(n) = -\sum_{k=1}^{P} a_k \, s(n-k) \quad \text{for some value of } P, a_k\text{'s}$$

• معادله را چند جمله ای:

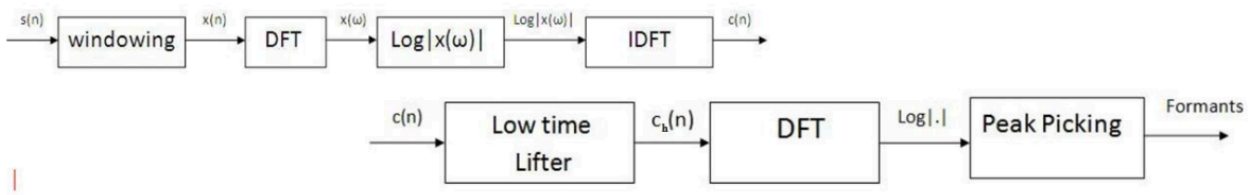
$$A(z) = 1 + a_1 z^{-1} + a_2 z^{-2} + \dots + a_P z^{-P}$$

• ریشه های $A(z)$ روی دایره مختلط، فرکانس های رزونانس سیستم گفتار (فرمنت ها) هستند.

و در آخر هم خروجی ها را نمایش میدهیم.

محاسبه فرمنت

- **گرفتن تبدیل فوریه از بخش پایین کپستروم(که توسط لیفتر پایین گذر جدا شد) و ترجیحا بردن به حوزه لگاریتم و یافتن نقاط بیشینه(پیک)**




```
# ---------------- FORMANTS (via LPC)
try:
    a = lpc_coefficients(frame, order=12)
    f1, f2 = extract_formants(a, sr)
    formant_f1.append(f1)
    formant_f2.append(f2)
except:
    formant_f1.append(0)
    formant_f2.append(0)
```

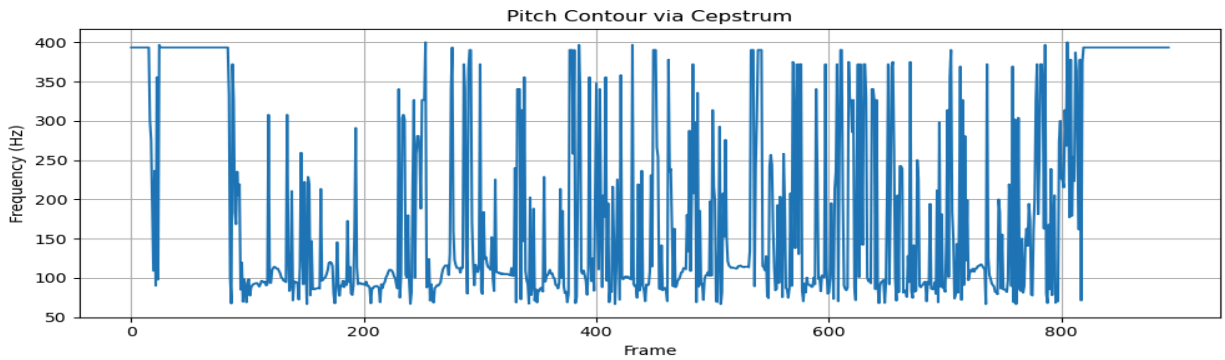
در نهایت، باید برای هر فریم این کد را اجرا کنیم:

```python
# Frame-by-frame analysis
for i in range(0, len(signal) - frame_length, hop_length):
    frame = signal[i:i + frame_length]
    frame = frame * window

    # --------------- PITCH (via high-time liftering in cepstrum)
    spectrum = fft(frame)
    log_magnitude = np.log(np.abs(spectrum) + 1e-10)
    cepstrum = np.real(ifft(log_magnitude))

    high_liftered = np.copy(cepstrum)
    high_liftered[:min_quef] = 0
    high_liftered[max_quef:] = 0

    peak_index = np.argmax(high_liftered)
    pitch = sr / peak_index if peak_index > 0 else 0
    pitch_values.append(pitch)

    # --------------- FORMANTS (via LPC)
    try:
        a = lpc_coefficients(frame, order=12)
        f1, f2 = extract_formants(a, sr)
        formant_f1.append(f1)
        formant_f2.append(f2)
    except:
        formant_f1.append(0)
        formant_f2.append(0)
```

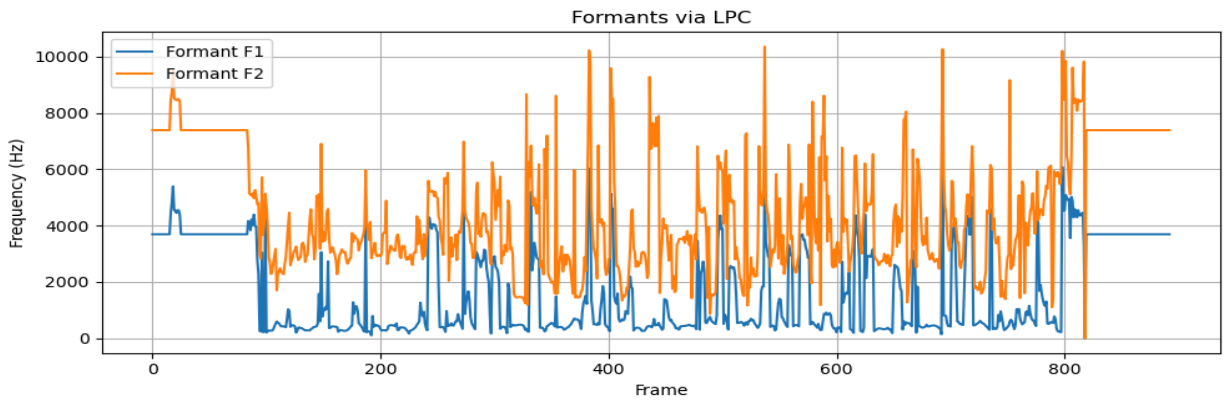
و سپس باید نتایج را نشان بدهیم:

```python
# Plot pitch contour
plt.figure(figsize=(10, 4))
plt.plot(pitch_values, label="Pitch (Hz)")
plt.title("Pitch Contour via Cepstrum")
plt.xlabel("Frame")
plt.ylabel("Frequency (Hz)")
plt.grid(True)
plt.tight_layout()
plt.savefig("pitch_contour.png")
plt.show()

# Plot formants
plt.figure(figsize=(10, 4))
plt.plot(formant_f1, label='Formant F1')
plt.plot(formant_f2, label='Formant F2')
plt.title("Formants via LPC")
plt.xlabel("Frame")
plt.ylabel("Frequency (Hz)")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.savefig("formants.png")
plt.show()
```

Pitch countor:


Pitch Contour via Cepstrum
Frequency (Hz)
400
350
300
250
200
150
100
50
0
200
400
600
800
Frame

Formants:


Formants via LPC
Formant F1
Formant F2
Frequency (Hz)
10000
8000
6000
4000
2000
0
0
200
400
600
800
Frame