**به نام خدا**

# پروژه پردازش گفتار

**سید شایان دانشور**

**9726523**

**استاد درس: دکتر دوست**

**تیر 1400**

سوال 1)

در این پروژه می خواهیم فرکانس گام و فرمنت های یک واکدار را به کمک تبدیل کپستروم بدست اوریم. از فایل a.wav موجود در سایت، در این پروژه استفاده شده است.

مراحل انجام کار:

ابتدا باید فریم بندی انجام داده و پنجره مناسبی روی صوت اعمال کنیم، طول فریم 25 میلی ثانیه و شیفت فریم 10 میلی ثانیه در نظر گرفته شده است. از پنجره همینگ (hamming) برای پنجره بندی (windowing) استفاده کردیم، سپس در هر فریم، مقدار dc را حذف میکنیم تا نویز نداشته باشیم.

```
FL_sec = 0.025; % frame length in seconds
FSL_sec = 0.01; % frame shift length in seconds
lifter_cutoff = 20; % n in liftering of cepstrum

[data,fs]=audioread('a.wav'); % reading the audio

frame_length = FL_sec * fs;
frame_shift_length = FSL_sec * fs;

fn=(length(data)- frame_length)/frame_shift_length + 1; % number of frames
window=hamming(frame_length); % hamming window

pitch_freqs = zeros(1, fn); % pitch frequencies of frames
formants = zeros(4, fn); % 4 formants  of frames
energy_of_frames=zeros(1,fn);% energy of frames


for i=1:fn
    frame = data((i-1)* frame_shift_length + 1:(i-1)* frame_shift_length +
frame_length);  %  framing speech
    frame = frame.*window;  %  windowing frame using hamming window
    frame = frame - sum(frame)/length(frame); % removing DC component from frame
```

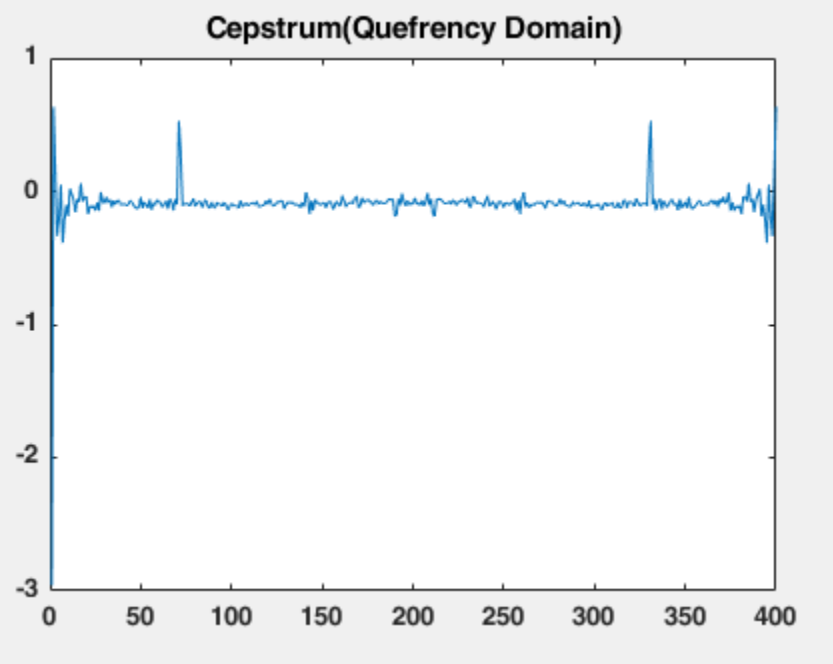
حال که پیش پردازش مورد نیاز روی فریم انجام شد، روی آن تبدیل کپستروم را پیاده میکنیم، که خود ناشی از سه مرحله ی تبدیل فوریه گسسته، و گرفتن لگاریتم از قدر مطلق آن و در نهایت گرفتن تبدیل فوریه گسسته معکوس از آن می باشد، به طور خلاصه برای هر فریم اینطور توصیف می توان کرد:

$$Cepstrum(frame) = ifft(\log(|fft(frame)|))$$

```
quefrency= ifft(log(abs(fft(frame)))); %% cepstrum transform
```

که همانطور که دیده می شود، روی فریم این تبدیل زده می شود.

تبدیل کپستروم (ضرایب کپسترال) را برای نمونه یک فریم (پر انرژی ترین فریم) را می توانیم ببینیم:




حال برای محاسبه فرکانس گام (Pitch Frequency) باید ابتدا یک لیفتر زمان بالا روی این مقادیر بزنیم و با الگوریتم peak picking ماکسیمم ها را پیدا کنیم، در اینجا در ابتدا برای همه فریم ها این کار را انجام میدهیم، درنهایت فریم های بی انرژی را حذف کرده و از بقیه میانگین میگیریم، دلیل این کار دقت بیشتر در محاسبه فرمنت است.

رابطه محاسبه فرکانس گام به کمک تبدیل کپستروم:

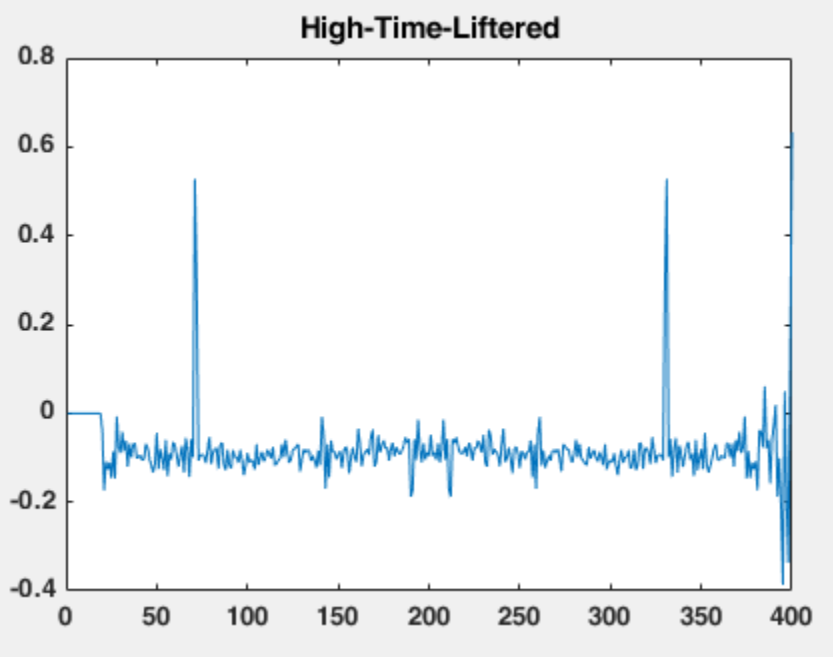
$$F_{pitch} = \frac{F_s}{I_{Pos}}$$

```
%%%%% calculating pitch frequncy
high_liftered = high_time_lifter(quefrency, lifter_cutoff); % high time liftering
[pks,locs] = findpeaks(high_liftered); % peak picking
[~,index] = max(pks);
peak_index = locs(index); %% finding indexes of peaks

if size(peak_index)> 0 % sometimes it is empty
    pitch_freqs(i)= fs/peak_index; %% calculating pitch frequency of frame
end
```

که همانطور که دیده می شود با ضریب کات آف (همان n در اسلاید های برابر 20) لیفتر زمان بالا زده ایم، سپس به کمک تابع findPeaks موجود در متلب این ماکسیمم های محلی را پیدا کرده و اندیس آن هارا در می آوریم، سپس در صورت وجود چنین ماکسیمم های محلی، با تقسیم کردن فرکانس نمونه برداری گفتار بر

مکان (اندیس) اولین ماکسیمم (peak)، فرکانس گام برای این فریم بدست می آید، در زیر تابع لیفتر زمان بالا را مشاهده می کنید: (که صرفا n عنصر آخر را نگه داشته باقی را صفر میکند)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% Functions %%%%%%%%%%%%%

function output = high_time_lifter(frame, n)
    high = zeros(1,length(frame));
    for i=1:length(frame)
        if i >= n
            high(i)=frame(i);
        end
    end
    output= high;
end
```

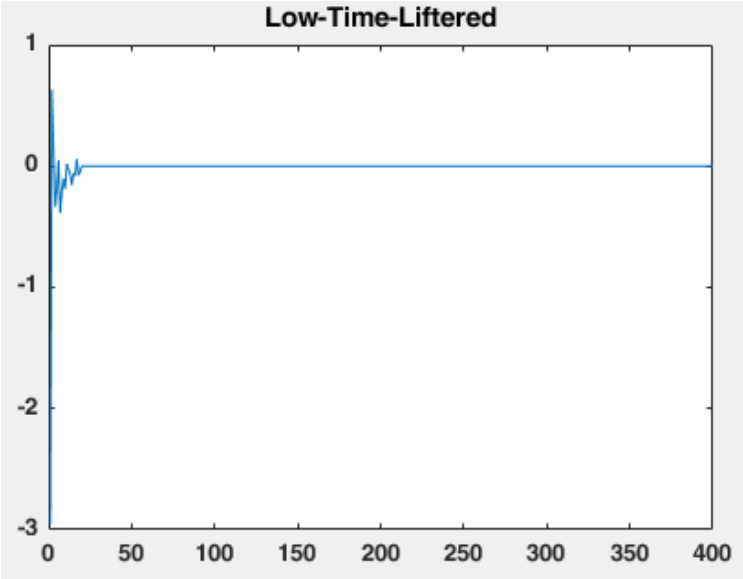


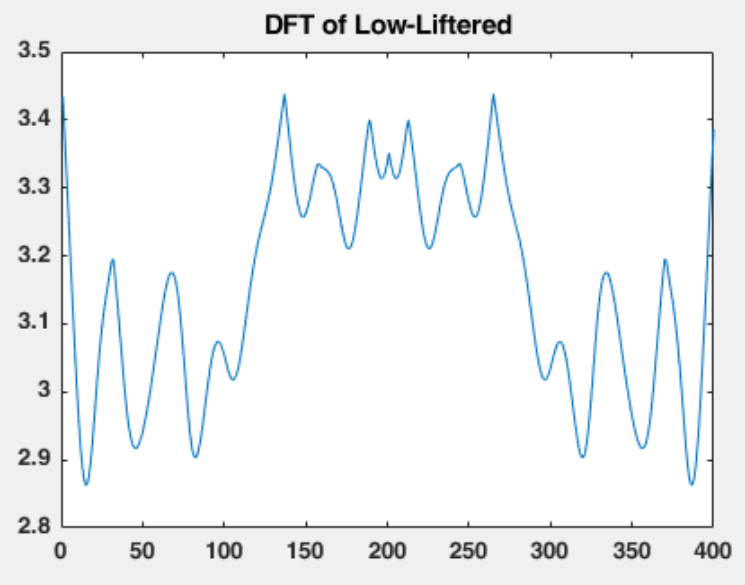

خب تا به اینجای کار فرکانس گام برای هر فریم بدست آمده، حال میرویم تا فرمنت ها را نیز برای هر فریم بدست آوریم، برای این کار باید روی تبدیل، لیفتر زمان پایین بزنیم ، دوباره روی آن تبدیل فوریه زده و سپس الگوریتم peak picking را اجرا کرده، 4 ماکسیمم محلی که زودتر دیده می شود را حساب کنیم.

```
    low_liftered = low_time_lifter(quefrency, lifter_cutoff); % low-time liftering

    low_liftered_dft = abs(log(fft(low_liftered))); % calculating dft for formants

    [~,locs] = findpeaks(low_liftered_dft); % peak picking

    for formant_no=1:4
        if length(locs)>= formant_no %% returned locs might be less than 4
            formants(formant_no,i) = locs(formant_no) * fs/fn;
        end
    end
```

در اینجا از تابع لیفتر زمان پایین استفاده شده و مابقی مشابه فرکانس گام است، با این تفاوت که در الگوریتم peak picking حال باید به دنبال 4 مقدار باشیم و هر کدام موجود بود برای این فریم در نظر بگیریم، در زیر تابع برای لیفتر زمان پایین آمده است:

```
function output = low_time_lifter(frame, n)
    low = zeros(1,length(frame));
    for i=1:length(frame)
        if i < n
            low(i)=frame(i);
        end
    end
    output= low;
end
```



حال هم فرکانس فرمت ها و هم فرکانس گام برای هر فریم را بدست آوردیم، اما همه فریم ها کاربردی نیستند و اصلا صوتی غیر از نویز (و یا غیر واکدار احتمالی) در آن ها وجود ندارد، به همین دلیل، انرژی هر فریم را نیز محاسبه میکنیم تا بر اساس آن تصمیم بگیریم. (البته از ZCR نیز می توان استفاده کرد)

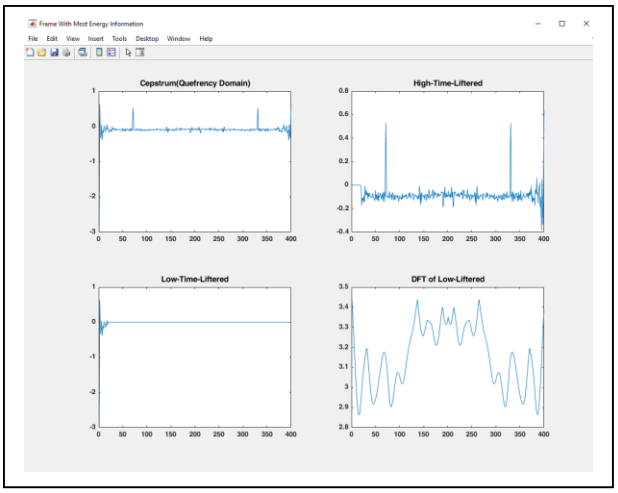
```
% calculating energy of each frame
energy_of_frames(i) = sum(frame.*frame)/ length(frame); % calculating frame's energy
```

حال باید حداکثر انرژی را مشخص کنیم و فریم هایی که تا حد قابل قبولی به حداکثر انرژی ماکزیمم نزدیک نیستند را در نظر نگیریم:

```
max_energy = max(energy_of_frames); % finding max energy of a frame
max_energy_frame_index = 0;
for i=1:fn %% finding index of frame with max energy
    if max_energy == energy_of_frames(i)
        max_energy_frame_index = i;
        break
    end
end
```

که در اینجا اندیس فریم با حداکثر انرژی را نیز گرفتیم تا اطلاعات این فریم را بتوانیم رسم کنیم، در ادامه اطلاعات مربوط به محاسبه دوباره تبدیل کپستروم، فیلتر های زمان بالا و پایین و فوریه از فیلتر پایین گذر و نیز رسم آن ها آمده است و استفاده دیگری ندارند:

```
% recalculating these values for ploting one frame
frame = data(max_energy_frame_index * frame_shift_length + 1:max_energy_frame_index *
frame_shift_length + frame_length);
frame = frame.*window;  %  windowing frame using hamming window
frame = frame - sum(frame)/length(frame); % removing DC component from frame
quefrency= ifft(log(abs(fft(frame))));
high_liftered = high_time_lifter(quefrency, lifter_cutoff);
low_liftered = low_time_lifter(quefrency, lifter_cutoff);
low_liftered_dft = abs(log(fft(low_liftered)));
% ploting cepstrum, high & low liftered , also dft of low time liftered
figure('NumberTitle', 'off', 'Name', 'Frame With Most Energy Information');
subplot(2,2,1);
plot(quefrency);
title('Cepstrum(Quefrency Domain)');
subplot(2,2,2);
plot(high_liftered);
title('High-Time-Liftered');
subplot(2,2,3);
plot(low_liftered);
title('Low-Time-Liftered');
subplot(2,2,4);
plot(low_liftered_dft);
title('DFT of Low-Liftered');
```



حال باید فریم های بدون انرژی کافی و همچنین آن هایی که به اندازه کافی برای محاسبه فرکانس گام و فرمنت ها، ماکزیمم نداشتند را حذف کنیم و روی این ها میانگین گرفته و عدد منطقی با خطای کم برای فرکانس گام و فرمنت ها را محاسبه کنیم:

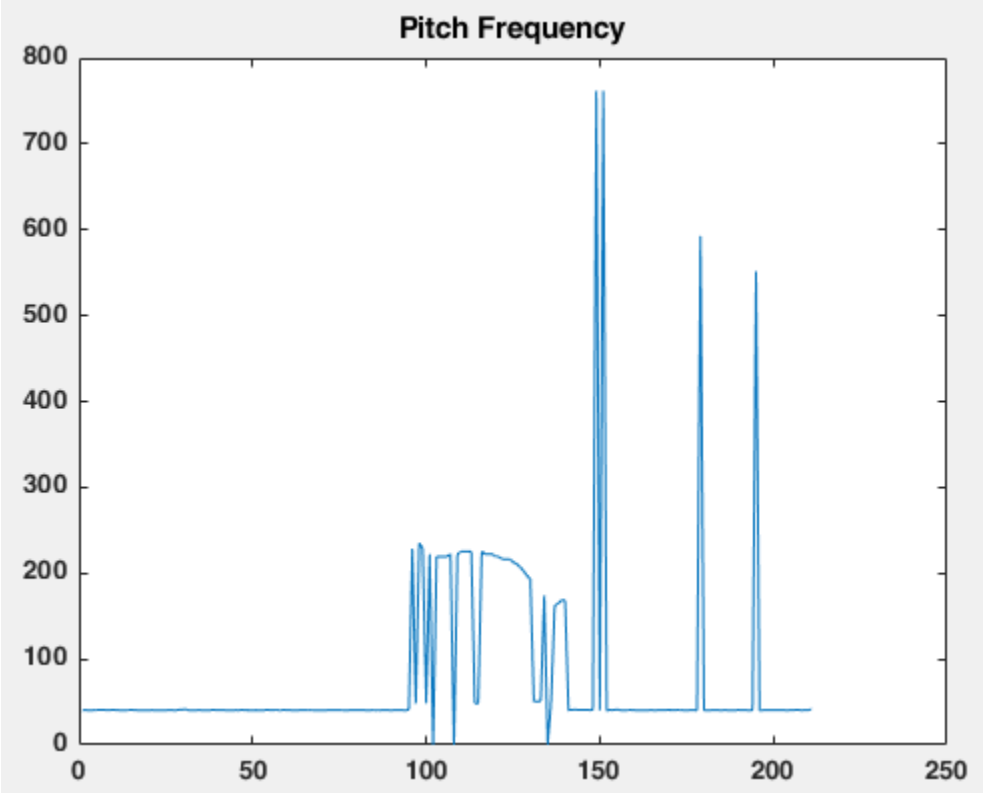
```
% Going to Calculate Average pitch and formant between frames with
% acceptable energy, so we get a more reliable result.
avg_pitch_freq = 0;
avg_formant1 = 0;
avg_formant2 = 0;
avg_formant3 = 0;
avg_formant4 = 0;
count_f1 = 0;
count_f2 = 0;
count_f3 = 0;
count_f4 = 0;
count_pitch = 0;

% neglecting frames with energy lower than half the maximum
for i=1:fn
    if energy_of_frames(i) > max_energy/2
        if pitch_freqs(i) > 0
            avg_pitch_freq = avg_pitch_freq + pitch_freqs(i); %% calculating average
of pitch frequencies and ignoring 0's due to size(peak_index)> 0
            count_pitch = count_pitch + 1;
        end
        if formants(1,i) > 0
           avg_formant1 = avg_formant1 + formants(1,i); %% calculating average of f1
and ignoring 0's due to length(locs)>= formant_no
           count_f1 = count_f1 + 1;
        end
        if formants(2,i) > 0
           avg_formant2 = avg_formant2 + formants(2,i); %% calculating average of f2
and ignoring 0's due to length(locs)>= formant_no
           count_f2 = count_f2 + 1;
        end
        if formants(3,i) > 0
           avg_formant3 = avg_formant3 + formants(3,i); %% calculating average of f3
and ignoring 0's due to length(locs)>= formant_no
           count_f3 = count_f3 + 1;
        end
        if formants(4,i) > 0
           avg_formant4 = avg_formant4 + formants(4,i); %% calculating average of f4
and ignoring 0's due to length(locs)>= formant_no
           count_f4 = count_f4 + 1;
        end
    end
end

avg_pitch_freq = avg_pitch_freq/count_pitch; %% final result of pitch frequency
avg_formant1 = avg_formant1 / count_f1; %% final result of formants 1,2,3,4
avg_formant2 = avg_formant2 / count_f2;
avg_formant3 = avg_formant3 / count_f3;
avg_formant4 = avg_formant4 / count_f4;
```

حال که میانگین فرمنت ها و نیز فرکانس گام را از فریم های با انرژی کافی (فرض کردیم فریم ها با بیش از نصف انرژی ماکیزیمم قابل قبول باشند)، بدست آوردیم، کافی است نتیجه را چاپ کرده و برای داشتن نیم نگاهی از فرکانس گام و فرمنت های فریم های مختلف آن هارا نیز رسم کنیم:

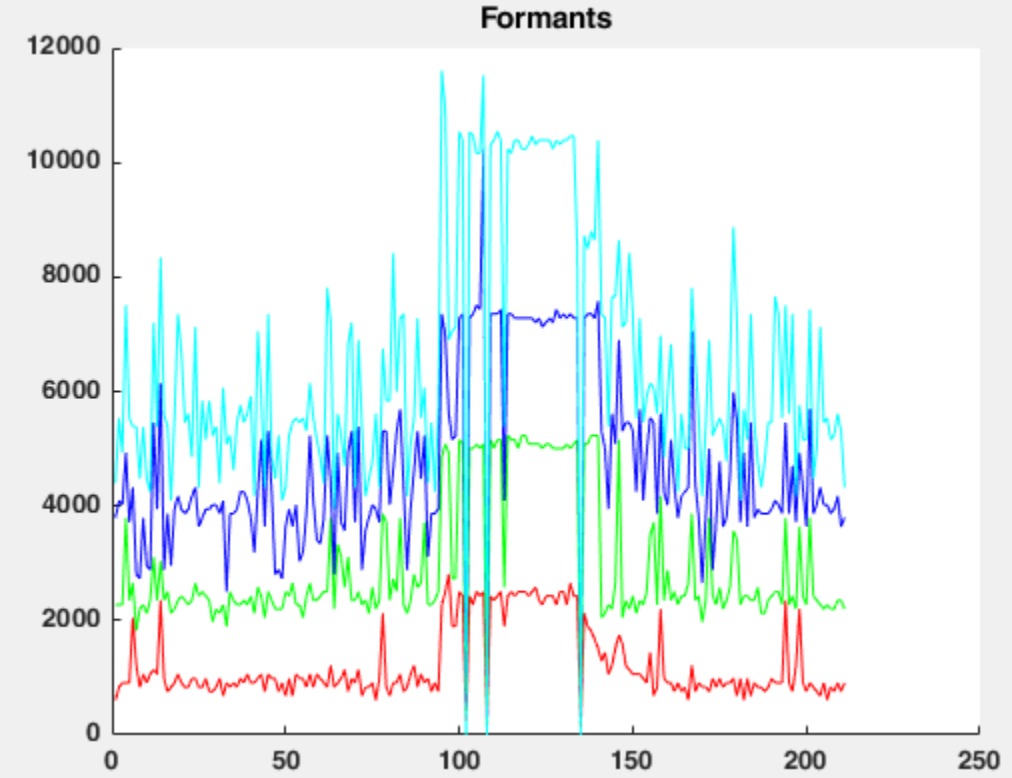
```
% ploting pitch frequencies
figure('NumberTitle', 'off', 'Name', 'Pitch Frequncies of Frames');
plot(pitch_freqs);
title('Pitch Frequency');
% ploting formants
figure('NumberTitle', 'off', 'Name', 'Formants of Frames');
hold on
plot(formants(1,:),'r');
plot(formants(2,:),'g');
plot(formants(3,:),'b');
plot(formants(4,:),'c');
title('Formants');
% printing final results (Pitch & Formant)
fprintf("Results: \n Pitch Frequncy %f \n", avg_pitch_freq);
fprintf(" Formants: F1= %f, F2= %f, F3= %f, F4= %f \n", avg_formant1, avg_formant2,
avg_formant3, avg_formant4)
```

که در زیر نمودار فرکانس های گام و فرمت ها را می بینید:

نمودار فرکانس های گام از همه فریم ها:



نمودار فرمنت ها برای فریم های مختلف: (از پایین به بالا فرمنت های 1 و 2 و 3 و 4):



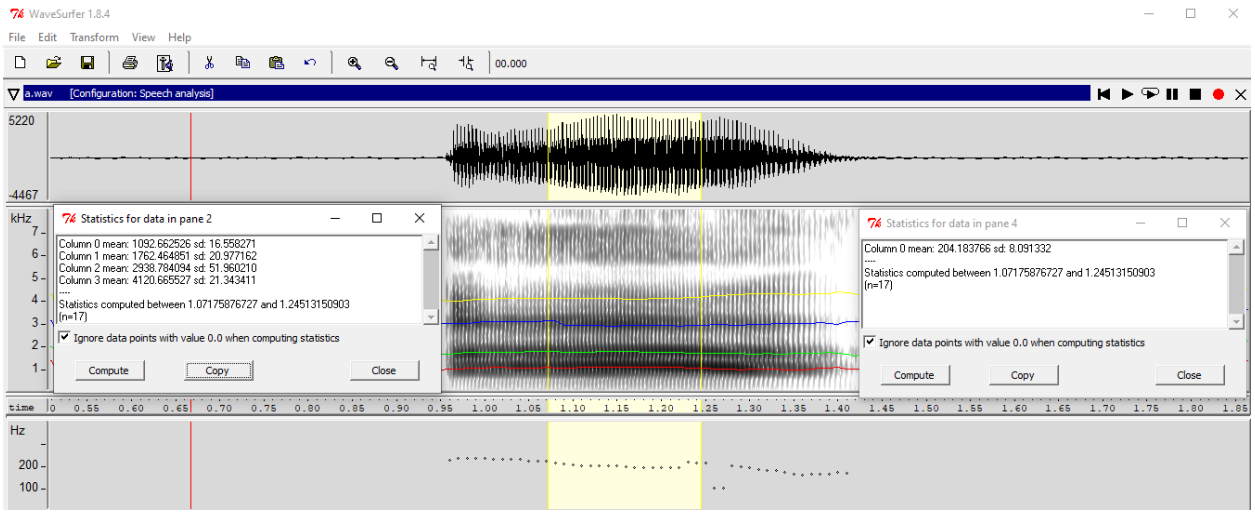

در نهایت نتایج چاپ می شوند:

```
Command Window
  Results:
   Pitch Frequncy 203.375794
   Formants: F1= 2324.462268, F2= 4986.540284, F3= 7294.786730, F4= 10191.469194
fx >>
```

همانطور که میبینیم مقدار فرکانس گام، برابر 203.375 بدست آمد، وقتی در wave surfer فرکانس گام را در میاوریم، حدود 203 الی 205 است، بنابرین در محاسبه فرکانس گام روش کپستروم کاملا موفق عمل می کند.

اما در محاسبه فرمنت ها چندان موفق عمل نکردیم، فرمنت اول در این روش حدود 2300 بدست آمده، در حالی که در wave surfer حدود 1100 است (حدود 1000 واحد اختلاف)، همچنین فرمنت دوم در اینجا حدود 4900 در آمده در حالیکه در نرم افزار حدود 1800 است (حدود 3000 واحد اختلاف)، این برای فرمنت های سوم وچهارم نیز بدتر می شود و اختلاف برای فرمنت سوم حدود 3100 و برای فرمنت چهارم حدود 5900 واخد اختلاف دارد، بنابرین نمی توان نظر داد که روش کپستروم در این زمینه تا چه حد خوب عمل میکند و به راحتی می توان به دقت بالایی در محاسبه فرمنت رسید. (به لحاظ تئوری به فرمنت ها میرسیم، اما خطای محاسبه و پیاده سازی نسبتا زیاد است)

این مقادیر در wave surfer نیز محاسبه و در زیر آمده است:




سوال دوم)

تبدیل های مختلف برای سهولت یا امکان پذیر کردن تجزیه و تحلیل به کارمی روند، تبدیل Z در بررسی خواص فرکانسی، پایداری و در کل تجزیه و تحلیل سیگنال ها و سیستم های دیجیتالی کاربرد دارد.

با استفاده از تبدیل فوریه می توانیم بگوییم طیف سیگنال یا محتوای فرکانسی آن چگونه است و برای بررسی و مطالعه اطلاعات در حوزه فرکانس مناسب است و ما را از حوزه زمان به حوزه فرکانس می برد، ویژگی های بسیاری را همانطور که در کلاس درس گفته شد، در این حوزه می توان استخراج کرد، اما از طرفی اطلاعاتی را نیز می توان از حوزه زمان بهتر استخراج کرد، که با استفاده از تبدیل فوریه دیگر به این اطلاعات دسترسی نداریم، از طرف دیگر در تبدیل کپستروم این اطلاعات از حوزه زمان وجود دارد و می توانیم از آن ها نیز استفاده کنیم. علاوه بر این ها، تبدیل فوریه کانولوشن که عمل پیچیده ای است را به ضرب تبدیل می کند و تبدیل کپستروم، آن را به جمع تبدیل می کند که بسیار ساده است.

به لحاظ شباهت، تبدیل فوریه پیوسته، حالت خاصی از تبدیل لاپلاس دو طرفه است، تبدیل فوریه گسسته حالت خاصی از تبدیل Z است، بنابرین از تبدیل Z می توان بسیار از خواص موجود در حوزه فرکانس را بدست آورد. تفاوت Z با فوریه گسسته این است که در Z با عدد مختلط سر و کار داریم ولی در فوریه با عدد موهومی.

ولی در عین حال هر دو در حوزه فرکانس اند، تبدیل کپستروم نیز شباهت اش به تبدیل فوریه در این است که در دل خود باید تبدیل فوریه گرفت و کانولوشن را به عمل ساده تری تبدیل می کند، تفاوت آن این است که تبدیل کپستروم در حوزه زمان است ولی فوریه در حوزه فرکانس فعالیت می کند.

پایان