

به نام خدا

گزارش پروژه نهایی درس پردازش صوت

دانشگاه صنعتی شاهرود

۸۹۰۱۰۰۴

هادی تقی زاده

استاد: دکتر مروی

تیر ۱۳۸۹

	فهرست	
۲	مقدمه	
۳	بخش ۱ تشخیص VUS از روی شکل موج	
۱۱	بخش ۲ محاسبه AC، ZC و انرژی زمان-کوتاه	
۱۶	بخش ۳ محاسبه ضرایب کپسترال مل و ضرایب LPC	
۱۹	بخش ۴ تعیین VUS در هر فریم با استفاده از پارامترهای انرژی و عبور از صفر و روش فصل الگوریتم‌ها	
۳۰	بخش ۵ محاسبه FFT و LogFFT به صورت مستقیم و با استفاده از ضرایب کپسترال و LPC	
۳۲	بخش ۶ نمایش صفحه زمان-فرکانس سیگنال صحبت و تعیین pitch, formant و نویز روی آن	
۳۴	بخش ۷ تعیین formant از روی پیک‌های FFT	
۳۶	بخش ۸ الگوریتمی برای محاسبه pitch برای فریم‌های voiced	
۳۹	ضمیمه کدهای برنامه‌ها	

این گزارش مربوط به پروژه پایانی درس پردازش صوت رشته مهندسی برق الکترونیک گرایش سیستم است. هدف از این پروژه مرور مطالب درسی و پیاده سازی عملی آن‌ها به منظور درک بهتر الگوریتم‌ها و مفاهیم تئوری است.

در بخش ۱ نواحی **voiced, unvoiced, silence** مربوط به یک سیگنال صحبت کوتاه فقط با بررسی چشمی شکل موج آن تعیین شده‌اند، نتایج این بخش با نتیجه بخش‌های بعدی که این نواحی با الگوریتم‌های ریاضی به صورت کامپیوتری به دست می‌آیند مقایسه خواهد شد. در بخش ۲ پارامترهای زمان کوتاه شامل **Zero-crossing**، **Energy** و **Autocorrelation** محاسبه می‌شوند، از این پارامترها در تشخیص **VUS**^۱ در بخش ۴ استفاده می‌شود. در بخش ۳ ضرایب مربوط به روش **Mel Cepstrum** و **LPC** محاسبه می‌شوند. در بخش ۴ از روی پارامترهای محاسبه شده در بخش ۲ برای طبقه بندی فریم‌ها به **VUS** استفاده می‌شود، همچنین از روش طبقه بندی بیزین فصل الگوریتم‌ها هم برای همین منظور استفاده شده است. در بخش ۵ مقادیر **FFT** و **LogFFT** هم مستقیم و هم با استفاده از ضرایب محاسبه شده در بخش ۳ و خطای مربوط به آن محاسبه می‌شود. در بخش ۶ **Pitch**، **Formant** و نویز بر روی نمودار زمان-فرکانس (**spectrogram**) فریم‌های صوت نشان داده می‌شود. در بخش ۷ مقادیر فرمنت از روی پیک شکل موج **FFT** به دست می‌آید و در بخش ۸ فرکانس **Pitch** نواحی **voiced** با یکی از الگوریتم‌های کتاب محاسبه می‌شود. در پایان هر بخش نتایج و مقایسه‌های لازم آورده شده است. کدهای برنامه‌ها در ضمیمه آورده شده است و فایل‌های مربوطه هم در سی دی همراه گزارش موجود است.

¹ Voiced/Unvoiced/Silence

تشخیص VUS از روی شکل موج

در این قسمت می‌خواهیم فقط با نگاه به شکل موج یک تکه صوت مربوط به صحبت قسمت‌های unvoiced، voiced و silence را تشخیص بدهیم. این کار با استفاده از اطلاعات قبلی صورت می‌گیرد؛ یعنی ما می‌دانیم که محتوای متن گفته شده چیست پس می‌توانیم با تقسیم کردن این متن بدون توجه به شکل موج بگوییم که به ترتیب چه نوع اصواتی داریم، مثلاً در فایل صوتی برنامه پروژه^۱، متن گفته شده “The discrete” است بنابراین می‌توانیم آن را به صورت زیر تقسیم کنیم:

1- Silence	/S
2- th	/U
3- e	/V
4- silence	/S
5- d	/U
6- i	/V
7- s	/U
8- k	/U
9- r	/U
10- i:	/V
11- t	/U
12- silence	/S

همچنین با توجه به اطلاعات قبلی درباره شکل موج هر کدام از انواع اصوات voiced، silence و unvoiced می‌توان مشخص کرد هر قسمت از چه زمانی شروع می‌شود و تا کجا ادامه می‌یابد. **Voiced** دارای شکل موج تقریباً متناوب است، **unvoiced** دارای تغییرات خیلی سریع است که ممکن است انرژی (دامنه) آن کم یا زیاد باشد و **silence** هم قسمت‌های باقی‌مانده است. طول زمانی هر قسمت بستگی به شکل بیان آن دارد. چون تشخیص قسمت‌های **voiced** راحت‌تر است بهتر است اول قسمت‌های **voiced** را مشخص کنیم بعد از روی کلمه گفته شده قسمت‌های دیگر را تشخیص دهیم. البته وقتی کلمه گفته شده را نداریم کار سخت می‌شود که در این صورت دیگر جداسازی قسمت‌های **voiced**، **silence** و

^۱ فایل speech_dft.wav در پوشه \dspblks\dspblks\toolbox\برنامه Matlab

unvoiced از روی شکل موج سخت می‌شود و خطای آن زیاد می‌شود و باید سراغ روش‌های ریاضی رفت که هم دقت بهتری دارند و هم به صورت خودکار انجام می‌شوند. این روش‌ها در بخش‌های بعدی استفاده می‌شوند.

شکل قسمت‌های جداشده در ادامه آورده شده است، چند نکته در این شکل‌ها وجود دارد که با بررسی دقیق و جزئی از روی شکل‌ها به دست آمده است:

اول این که همان طور که در این شکل‌ها مشخص است نمی‌توان قسمت‌های مختلف را کاملاً با یک مرز دقیق از هم جدا کرد. هر صوت به تدریج به صوت بعدی تبدیل می‌شود، یعنی در مرز بین دو صوت مختلف مخلوطی از هر دو شکل موج وجود دارد. همچنین شکل متناوب اصوات **voiced** کاملاً منظم نیست، این اصوات در ابتدا دامنه کم دارند، سپس دامنه آن‌ها زیاد می‌شود و در انتها دوباره دامنه‌اش کم می‌شود. نکته دیگر این که برای اصوات **unvoiced** و **silence** مقدار متوسط غیر صفر داریم که به دلیل وجود ^۱Hum و مقدار DC است. در این صورت برای بخش‌های بعدی باید این مؤلفه‌های فرکانسی حذف شوند. آخرین نکته این که اصوات **unvoiced** که پشت سرهم می‌آیند قابل تفکیک نیستند، در شکل مربوط به قسمت‌های **unvoiced** متوالی **sc** کاملاً به هم متصل و غیر قابل تفکیک هستند.

بعد از تحلیل شکل موج، قسمت‌های صوت به شکل زیر به دست آمده است:

1- Silence	/S
2- th	/U
3- e	/V
4- d	/U
5- i	/V
6- sk	/UU
7- r	/U
8- i:	/V
9- t	/U

همان طور که دیده می‌شود تغییراتی نسبت به حدس اولیه دیده می‌شود. اول این که قسمت **silence** بین دو کلمه **The** و **discrete** وجود ندارد چون این کلمات پشت سرهم گفته می‌شود. قسمت **unvoiced**

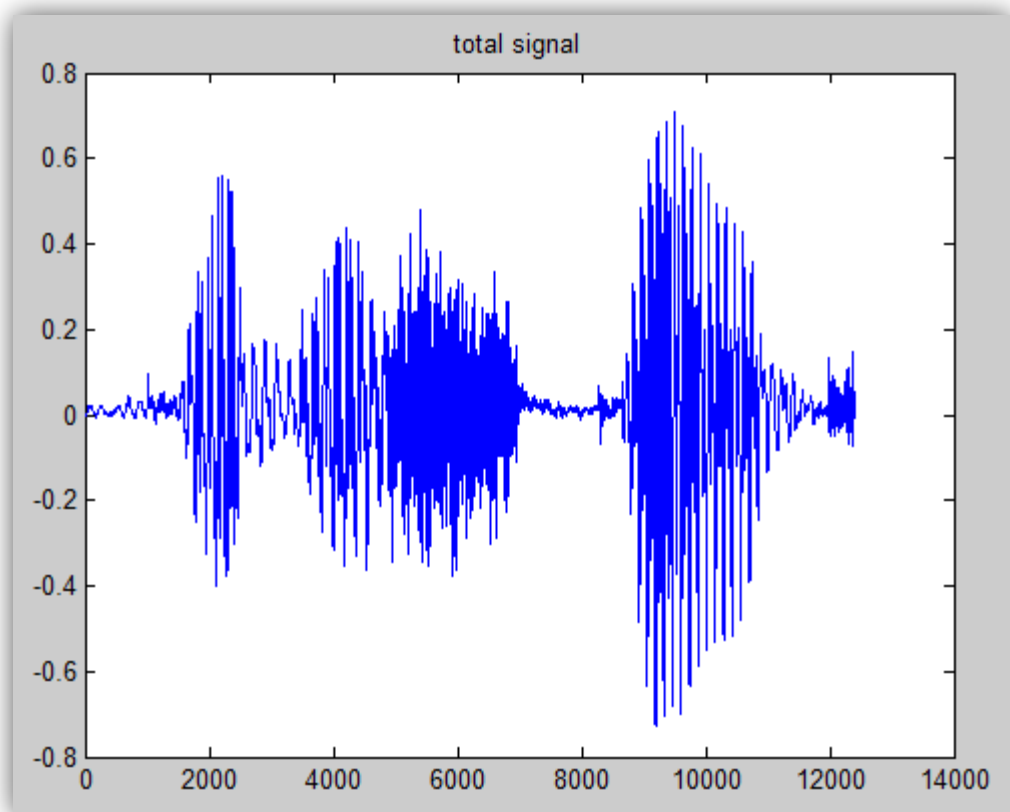
^۱ صدای ناشی از مؤلفه فرکانس ۵۰ هرتز برق شهر مربوط به وسایل برقی

مربوط به حروف S و C هم قابل تفکیک نیستند. قسمت silence آخر هم وجود ندارد چون کلمه بعدی که Fourier است بلافاصله گفته می شود طوری که حرف t آخر discrete و F اول Fourier کاملاً به هم وصل هستند.

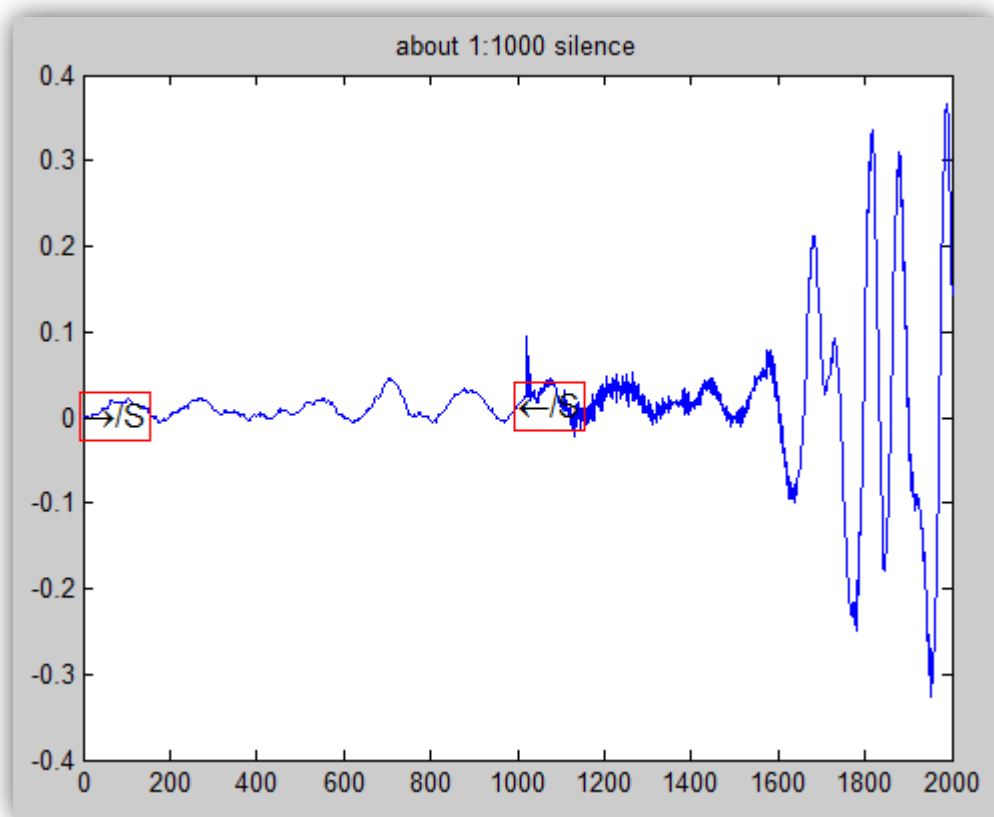
نتایج این بخش در جدول صفحه بعد خلاصه شده است. در این جدول برای بعضی از نواحی که به هیچ کدام از نواحی VUS شباهت نداشتند با علامت ؟ مشخص شده است. این نواحی در مرز بین دو فنوم از نوع مختلف قرار گرفته اند که در نتیجه به هر دو شباهت دارند.

Start sample	End sample	Type
1	1000	s
1001	1640	u
1641	3370	v
3371	3469	?
3470	3650	u
3651	4680	v
4681	8270	u
8271	8660	u
8661	8819	?
8820	11000	v
11001	11949	?
11950	12400	u

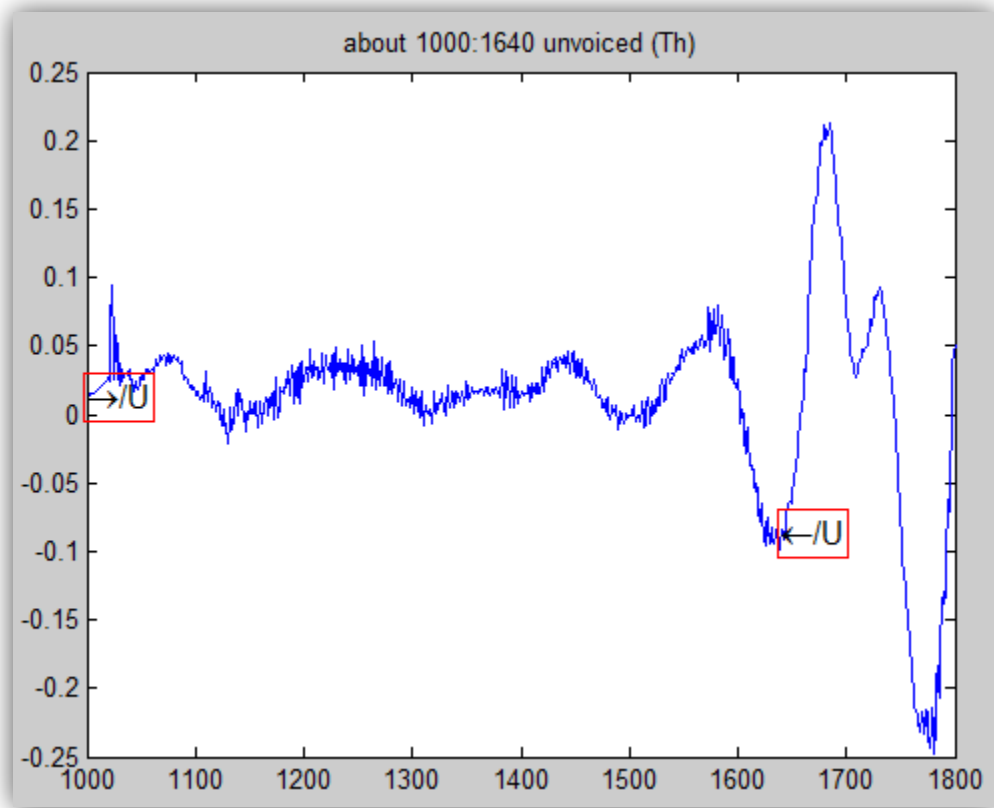
جدول ۱- تشخیص نوع نواحی صوت بر اساس مشاهده شکل موج



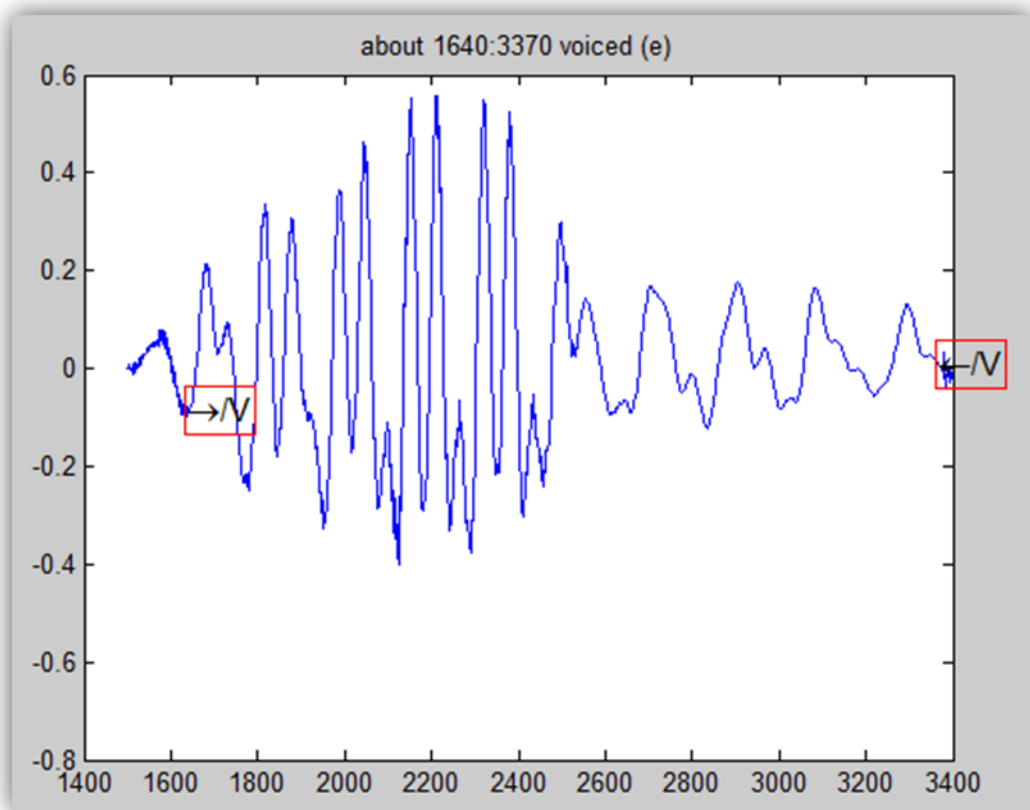
شکل ۱- شکل موج کل سیگنال صوت



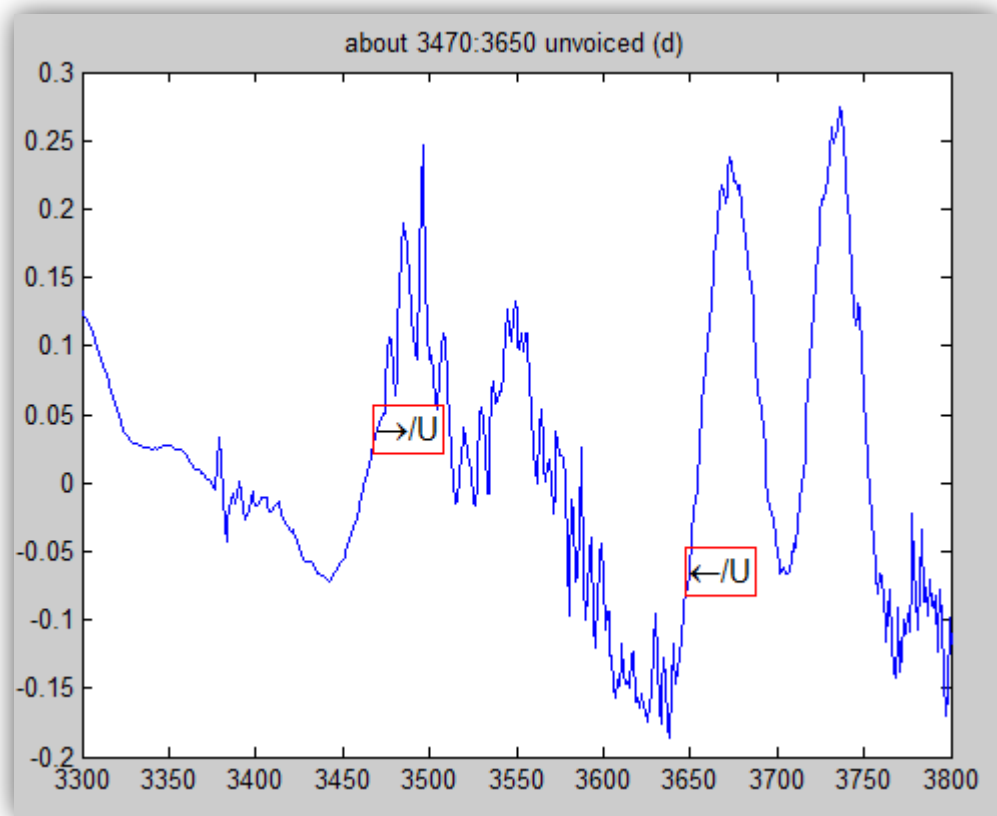
شکل ۲- قسمت سکوت اولیه



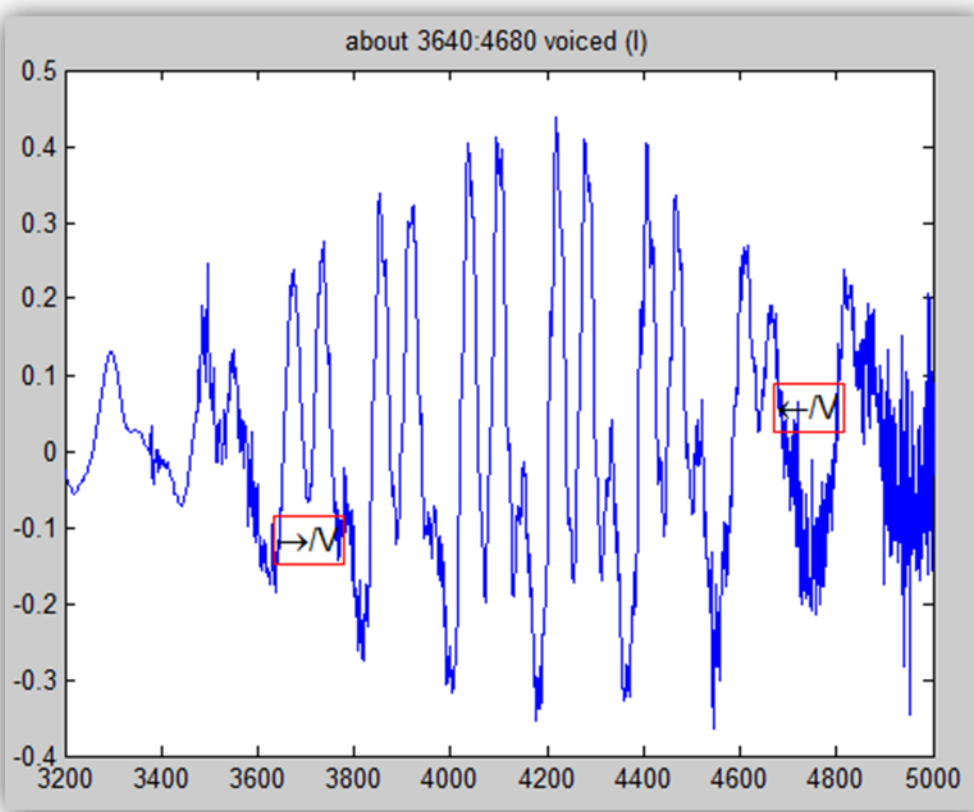
شکل ۳- حرف بی صدای th



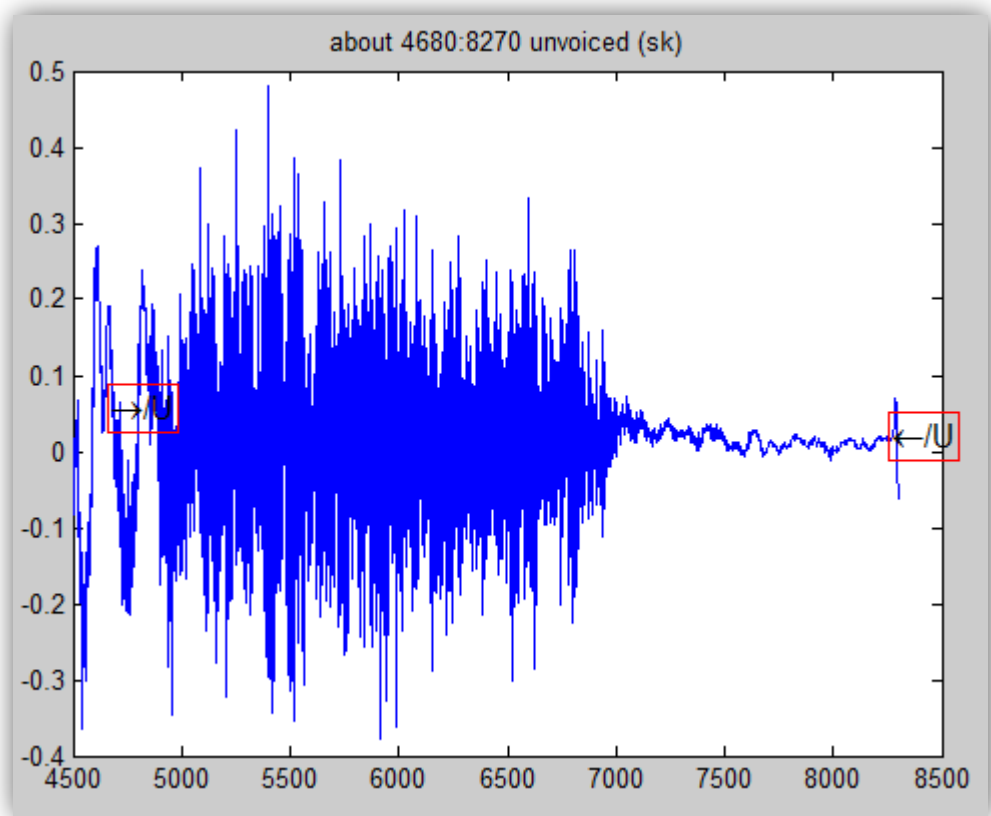
شکل ۴- حرف صدادار e



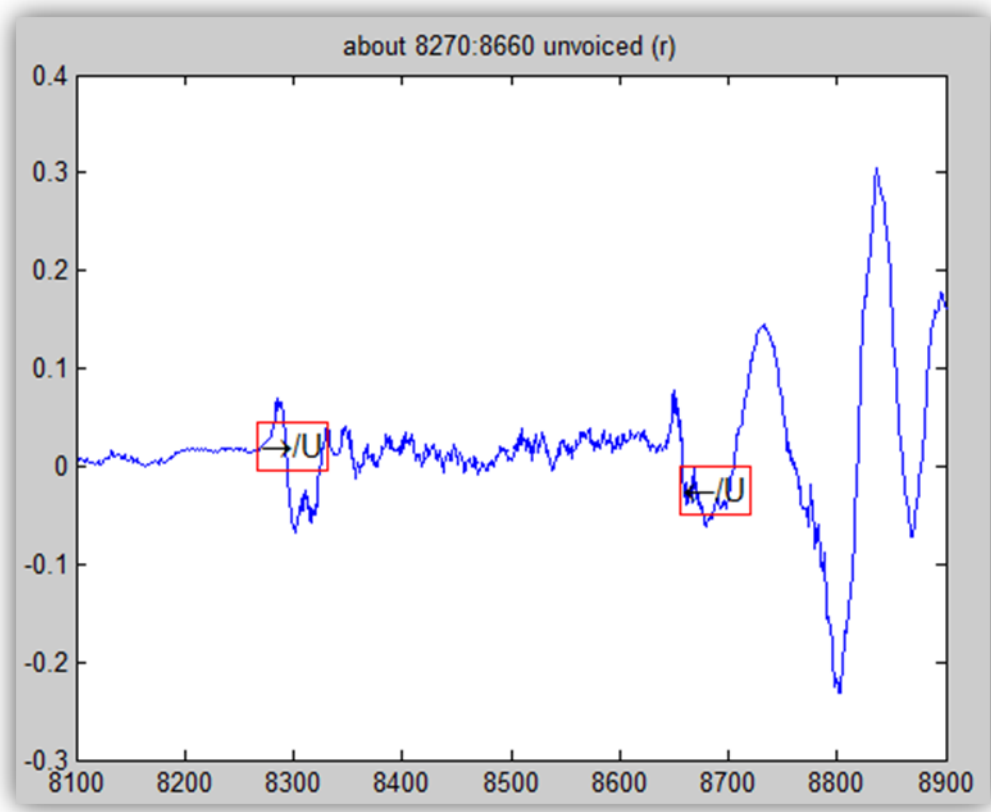
شکل ۵- حرف بی صدای d



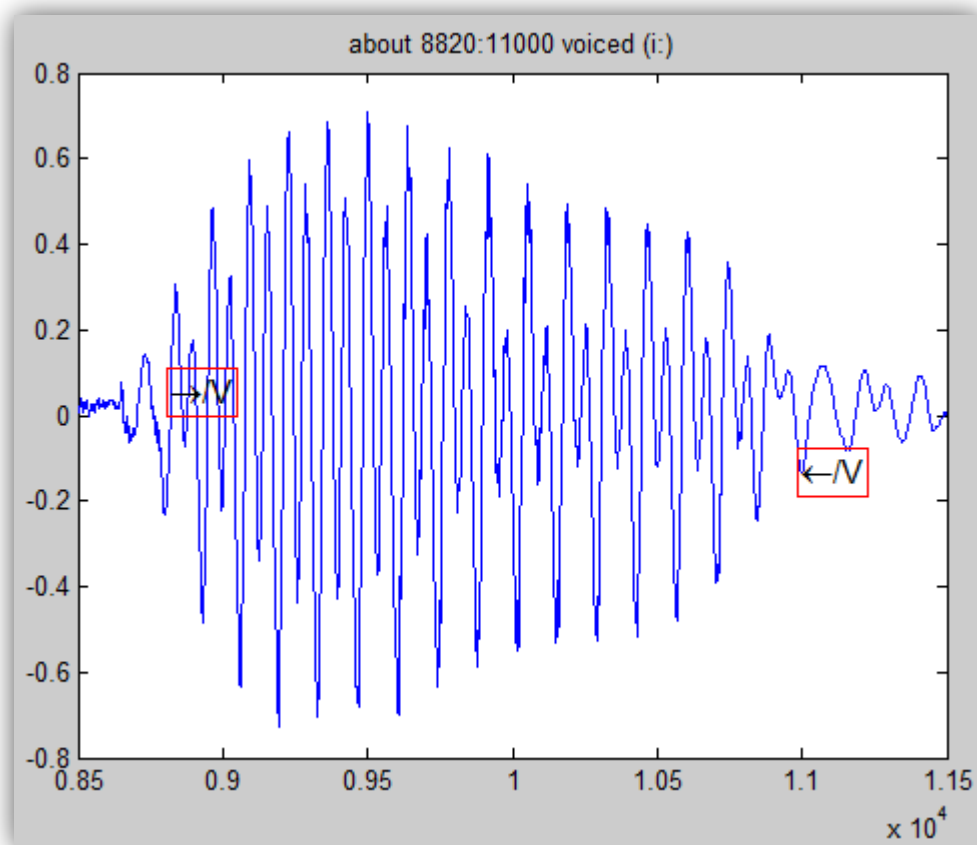
شکل ۶- حرف صدadar l



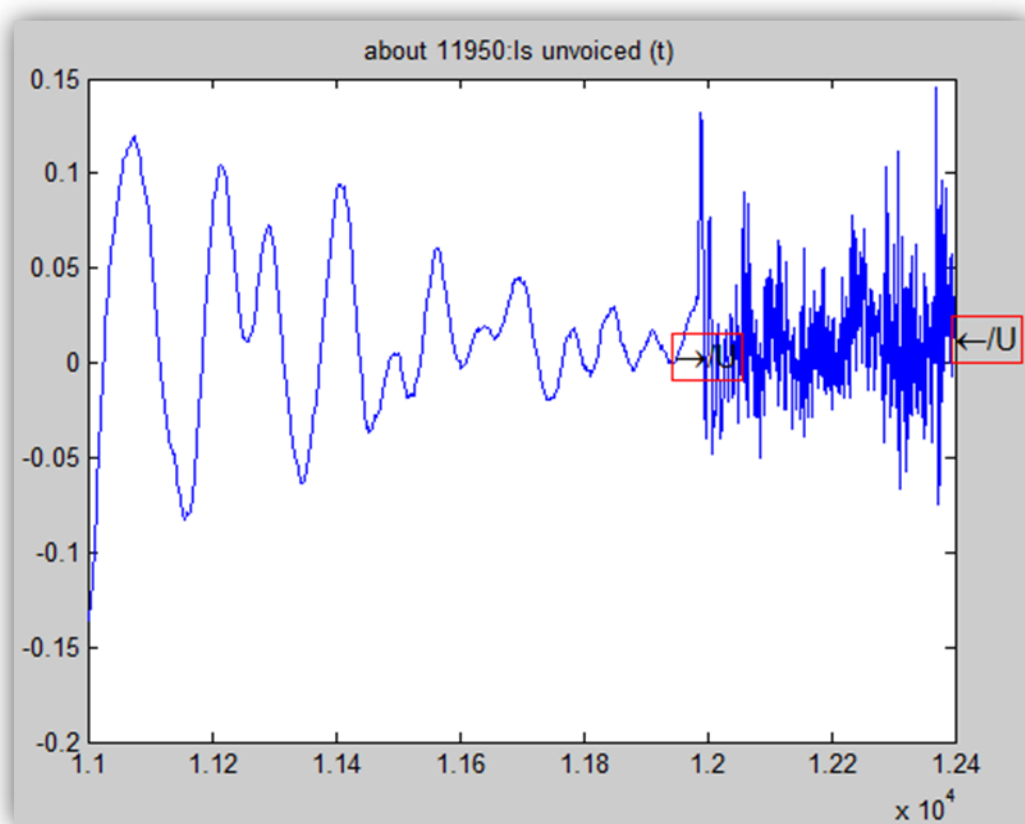
شکل ۷- حروف بی صدای sk



شکل ۸- حرف بی صدای r



شکل ۹- حرف صدادار i:



شکل ۱۰- حرف بی صدای t

محاسبه ZC^1 ، AC^2 و انرژی زمان-کوتاه

در تحلیل زمان کوتاه سیگنال صوت مهم‌ترین پارامترها انرژی زمان-کوتاه، نرخ عبور از صفر زمان-کوتاه و خود همبستگی زمان-کوتاه هستند. این پارامترها در تشخیص نوع صوت (بی صدا، صدادار یا سکوت) استفاده می‌شوند. ابتدا باید فرکانس‌های DC و hum باید حذف شوند.

سپس باید کل سیگنال صوت را به صورت فریم‌های زمان-کوتاه درآورد. به دست آوردن زمان ایده آلی برای تقسیم بندی سیگنال صوت به قطعات کوچک امکان پذیر نیست اگر این زمان خیلی کوچک باشد ممکن است نتواند کل زمان یک ناحیه صوت را شامل شود و اگر خیلی بزرگ باشد ممکن است بیش از یک فنوم در آن قرار بگیرد، اما با توجه به تجربه می‌توان یک زمان تقسیم بندی خوب به دست آورد که مصالحه خوبی بین دو مورد بالا برقرار کند. این مقدار بنا به پیشنهاد کتاب مقداری بین ۱۰ تا ۲۰ میلی ثانیه مناسب است که مقدار اولیه آن ۲۰ میلی ثانیه انتخاب شد. با آزمایش و خطا می‌توان بهترین مقدار را به دست آورد. البته اگر به تقسیم بندی نواحی در بخش ۱ برگردید می‌بینید که طول فنوم‌ها خیلی متغیر است به عنوان مثال فنوم حرف بی صدای d فقط حدود ۱۸۰ نمونه است در حالی که فنوم حرف صدادار i: حدود ۲۱۸۰ نمونه است.

مقدار این زمان در برنامه در متغیر delta ذخیره شده است. تعداد نمونه‌های سیگنال که معادل این زمان است برابر است با $fs * delta$ که fs فرکانس نمونه گیری است که برای فایل صوتی استفاده شده در برنامه ۲۲۰۵۰ هرتز است. برای هر فریم ۵۰٪ همپوشانی با فریم مجاور در نظر گرفته شده است. تعداد نمونه‌های بدون همپوشانی در هر فریم در متغیر nol محاسبه شده است. متغیر nw تعداد کل فریم‌ها است.

پنجره‌های Hamming و مستطیلی برای پنجره بندی سیگنال استفاده شده‌اند. متغیرهای مربوط به پنجره Hamming با پیشوند یا پسوند h و متغیرهای مربوط به پنجره مستطیلی با پیشوند یا پسوند r نام‌گذاری شده‌اند. مقادیر پنجره بندی شده در متغیرهای hv و rv ذخیره شده‌اند، که هر سطر شامل یک فریم است و ستون‌های این ماتریس‌ها نمونه‌های مربوط به هر فریم است.

¹ Zero-Crossing

² AutoCorrelation

پس از محاسبه فریم‌های صوت، باید انرژی **Energy**، نرخ عبور از صفر **Zero-crossing(ZC)** و خودهمبستگی **Autocorrelation(AC)** را برای هر فریم محاسبه کنیم. انرژی برای هر فریم مجموع مجذور نمونه‌های آن است. نرخ عبور از صفر تعداد تغییر علامت نمونه‌های فریم است که با بررسی شرط نامساوی بودن علامت هر دو نمونه متوالی به دست می‌آید. تابع خود همبستگی طبق تعریف زیر استفاده شده است:

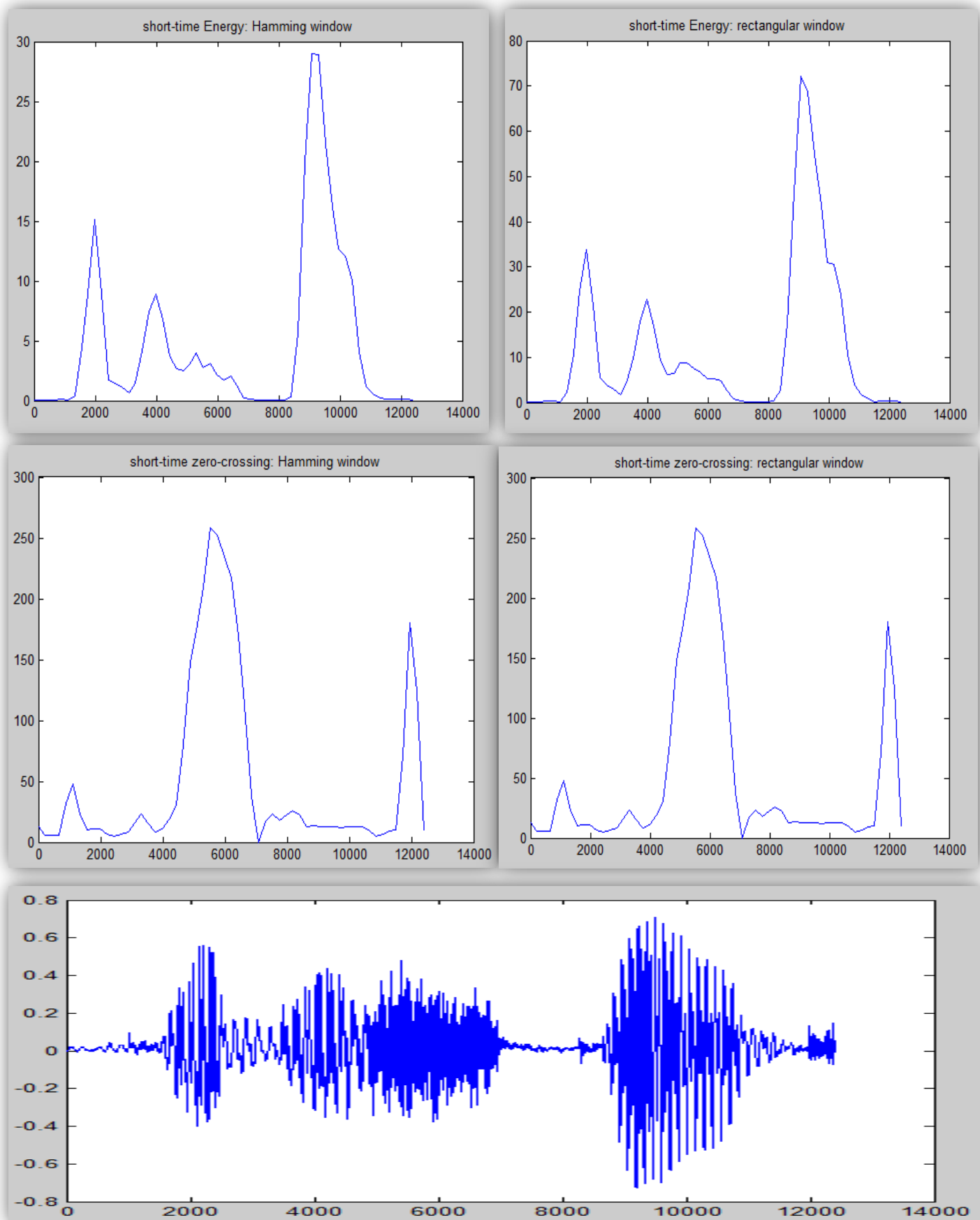
$$R_n(k) = \sum_{m=0}^{N-1-k} [x(n+m)w(m)][x(n+m+k)w(k+m)]$$

که n شماره فریم است، m اندیس نمونه در فریم است و k که از 0 تا N تغییر می‌کند اندیس تابع خود همبستگی است.

در انتها مقادیر به دست آمده شامل سیگنال فریم بندی شده، انرژی، نرخ عبور از صفر و خود همبستگی زمان-کوتاه برای هر دو نوع پنجره همینگ و مستطیلی در یک فایل **mat** برای استفاده در بخش‌های بعدی ذخیره می‌شوند.

نمودارهای زیر برای انرژی و نرخ عبور از صفر زمان-کوتاه به دست آمده‌اند، محور افقی با تعداد نمونه‌های کل سیگنال هم مقیاس شده است. شکل موج کل سیگنال هم آورده شده است تا مقایسه راحت‌تر شود.

در ادامه برای هر فریم مقادیر به دست آمده برای **Energy** و **Zero-crossing** در جدول ۲ آورده شده است. همچنین نوع آن فریم با استفاده از جدول ۱ مشخص شده است. در بعضی از فریم‌ها کل فریم در یک ناحیه قرار نگرفته است، در این موارد نوع فنوم‌های قرار گرفته در آن فریم به ترتیب از چپ به راست نوشته شده است. Z در فریم آخر هم به معنای صفرهای اضافه شده در عمل **zero-padding** است.



شکل ۱۱ تا ۱۴- پارامترهای زمان کوتاه و شکل ۱۵- کل شکل موج صوت

جدول ۲- مقادیر انرژی و عبور از صفر و نوع صوت برای هر فریم

Frame #	Start address	End address	Zero-crossing	energy	Type
1	1	441	13	0.03	s
2	222	662	6	0.01	s
3	443	883	6	0.08	s
4	664	1104	6	0.07	s-u
5	885	1325	32	0.11	u
6	1106	1546	48	0.09	u
7	1327	1767	23	0.44	u-v
8	1548	1988	10	4.46	v
9	1769	2209	11	8.87	v
10	1990	2430	11	15.10	v
11	2211	2651	7	8.93	v
12	2432	2872	5	1.67	v
13	2653	3093	7	1.48	v
14	2874	3314	8	1.11	v
15	3095	3535	16	0.63	v-?
16	3316	3756	23	1.47	u-v
17	3537	3977	15	4.16	v
18	3758	4198	8	7.40	v
19	3979	4419	11	8.86	v
20	4200	4640	19	6.76	v
21	4421	4861	30	3.69	v-u
22	4642	5082	80	2.71	u
23	4863	5303	148	2.53	u
24	5084	5524	174	3.06	u
25	5305	5745	209	3.96	u
26	5526	5966	258	2.73	u
27	5747	6187	252	3.10	u
28	5968	6408	235	2.18	u
29	6189	6629	217	1.72	u
30	6410	6850	170	2.08	u
31	6631	7071	113	1.28	u
32	6852	7292	38	0.25	u
33	7073	7513	0	0.11	u
34	7294	7734	17	0.06	u
35	7515	7955	23	0.02	u

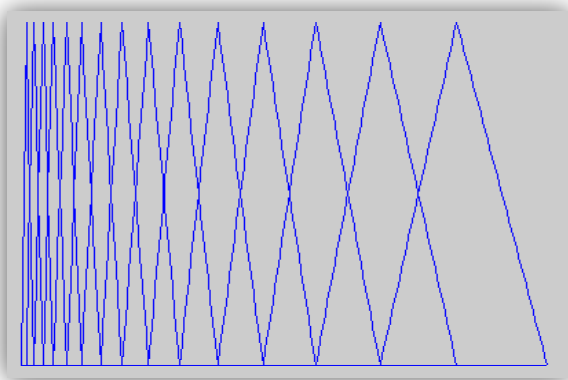
36	7736	8176	18	0.02	u
37	7957	8397	22	0.05	u
38	8178	8618	26	0.09	u
39	8399	8839	22	0.31	u-?-v
40	8620	9060	13	5.65	v
41	8841	9281	14	19.85	v
42	9062	9502	13	29.01	v
43	9283	9723	13	28.93	v
44	9504	9944	13	21.46	v
45	9725	10165	12	16.21	v
46	9946	10386	13	12.71	v
47	10167	10607	13	12.06	v
48	10388	10828	13	10.05	v
49	10609	11049	10	4.06	v
50	10830	11270	5	1.21	v-?
51	11051	11491	7	0.55	?
52	11272	11712	9	0.24	?
53	11493	11933	10	0.09	?
54	11714	12154	71	0.12	?-u
55	11935	12375	180	0.13	u
56	12156	12596	124	0.17	u
57	12377	12817	9	0.00	u-z

محاسبه ضرایب کپسترال مل و ضرایب LPC

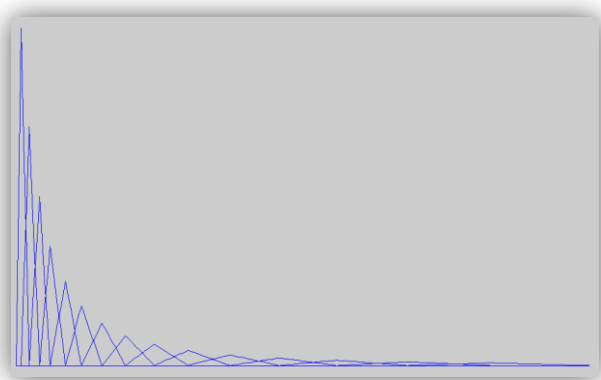
برنامه این بخش به دو قسمت تقسیم شده است. برنامه p3_1 مربوط به محاسبه ضرایب کپسترال مل و برنامه p3_2 مربوط به محاسبه ضرایب LPC است.

در برنامه محاسبه ضرایب مل از یک تابع خارجی به نام `melfilter` استفاده شده است. این برنامه ضرایب مربوط به فیلتربانک مل را می‌سازد. ورودی‌های این تابع `fs`, `L`, `N` هستند. `N` تعداد نمونه‌های هر فریم است، `L` تعداد ضرایب فیلتربانک است و `fs` فرکانس نمونه گیری است. خروجی تابع هم یک ماتریس است که در هر سطر آن یکی از فیلترهای فرکانسی مل قرار دارد. منبع این برنامه سایت `Mathworks` شرکت سازنده برنامه `Matlab` است.

در برنامه اصلی از تابع `melfilter` فقط برای محاسبه ضرایب فیلتربانک مل استفاده شده است. فیلتربانک مل شامل `L` فیلتر است که هر کدام یک شکل مثلثی دارد، این فیلترها در فرکانس‌های پایین دارای عرض باند کم هستند و به سمت فرکانس‌های بالا که می‌روند عرض باند آن‌ها بزرگ می‌شود. هر فیلتر از وسط فیلتر قبلی شروع می‌شود و عرض آن دو برابر فیلتر قبلی است. اندازه فیلترها می‌تواند با افزایش فرکانس کاهش پیدا کند اما در این تابع که توسط `Plannerer` نوشته شده همه فیلترها هم دامنه هستند البته با دستکاری برنامه می‌توان آن را طوری اصلاح کرد که مقدار انرژی تمام فیلترها برابر شود؛ در این صورت کافی است که در سمت راست تساوی در خطوط ۴۳ و ۴۸ برنامه `melfilter` ضریب $\sqrt{2}^{(1-c)}$ را اضافه کرد.



شکل ۱۷- فیلتر بانک مل با دامنه ثابت



شکل ۱۶- فیلتربانک مل با دامنه کاهشی

ضرایب برای سه مقدار ۱۰ و ۱۲ و ۱۵ برای L محاسبه شده‌اند. چون L ضریب نماینده N نمونه در هر فریم هستند که در این برنامه $N=441$ به دست آمده است در نتیجه این کار منجر به خطا می‌شود، طبیعی است که مقدار L هر چه بیشتر باشد خطا کمتر می‌شود اما تجربه نشان داده است که حدود ۱۳ برای تعداد ضرایب مقدار خوبی است چون با افزایش بیشتر L باعث کم شدن محسوس خطا نمی‌شود. این حقیقت برای پارامتر p در LPC هم تقریباً صدق می‌کند.

در این برنامه مقدار Emel از فرمول زیر به دست آمده است:

$$E_{mel}[n, l] = \frac{1}{A_l} \sum_{k=L_l}^{U_l} |V_l(\omega_k) X[n, \omega_k]|^2 \quad A_l = \sum_{k=L_l}^{U_l} |V_l(\omega_k)|^2$$

در فرمول‌های بالا n شماره فریم است. L_l و U_l حدود پایین و بالای l امین فیلتر هستند. ω_k هم k امین نمونه در فیلتربانک است. X هم تبدیل فوریه (FFT) فریم n ام است. چون در این جا نمونه‌ها از جنس فرکانس هستند از نماد ω برای آن‌ها استفاده شده است. البته در این جا به جای frequency باید از quefrequency استفاده می‌کنیم چون با کپسترال سروکار داریم ولی هنوز از لگاریتم استفاده نکرده ایم. برای محاسبه ضرایب کپسترال از لگاریتم Emel تبدیل DCT می‌گیریم که حاصل این عمل cmel است که همان ضرایب کپسترال مل هستند.

ضرایب نهایی cmel برای هر دو پنجره همینگ و مستطیلی برای استفاده ذخیره شده‌اند.

محاسبه ضرایب LPC برای فریم‌ها با دستور lpc برنامه Matlab انجام شده است. فرمت این دستور به شکل زیر است:

$[a,g]=lpc(x,p)$

که x سیگنالی است که می‌خواهیم ضرایب فیلتر پیش‌بینی کننده را برای آن به دست آوریم، p درجه فیلتر است. a بردار ضرایب است که برای فیلتر خطای روبه جلو به کار می‌رود و g خطای پیش‌بینی است. بردار ضرایب فیلتر خطای پیش‌بینی، منفی شده بردار فیلتر پیش‌بینی است که یک عنصر با مقدار ۱ در ابتدای آن اضافه شده است، به همین دلیل برای محاسبه ضرایب فیلتر پیش‌بینی باید طبق دستور زیر عمل شود:

$a=-a(2:end);$

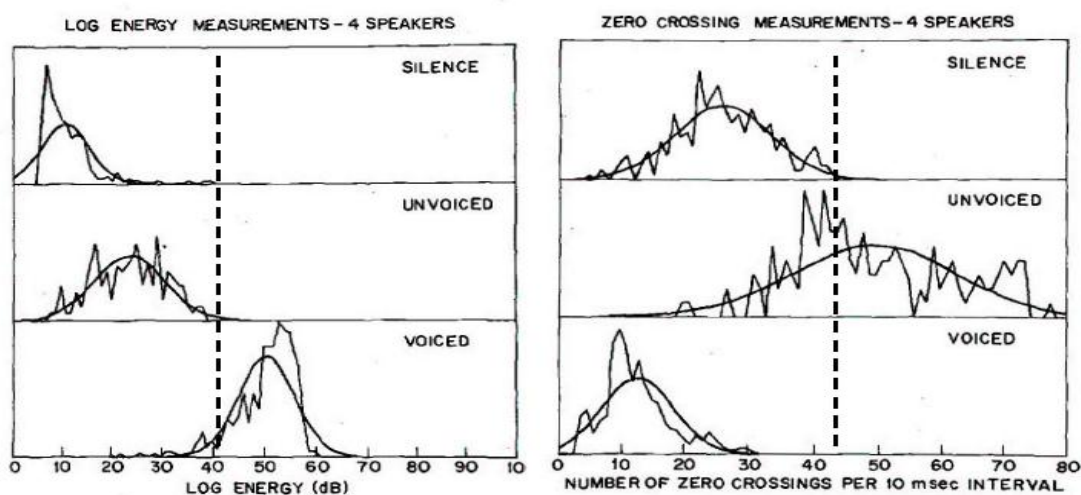
این دستور منجر به حذف مقدار ۱ در ابتدای بردار ضرایب و منفی کردن بقیه ضرایب می‌شود.

ضرایب LPC برای مقادیر $p=10,12,15$ برای فریم‌ها هم برای پنجره مستطیلی و هم پنجره همینگ محاسبه و ذخیره شده است.

تعیین VUS در هر فریم با استفاده از پارامترهای انرژی و عبور از صفر و روش فصل الگوریتم‌ها

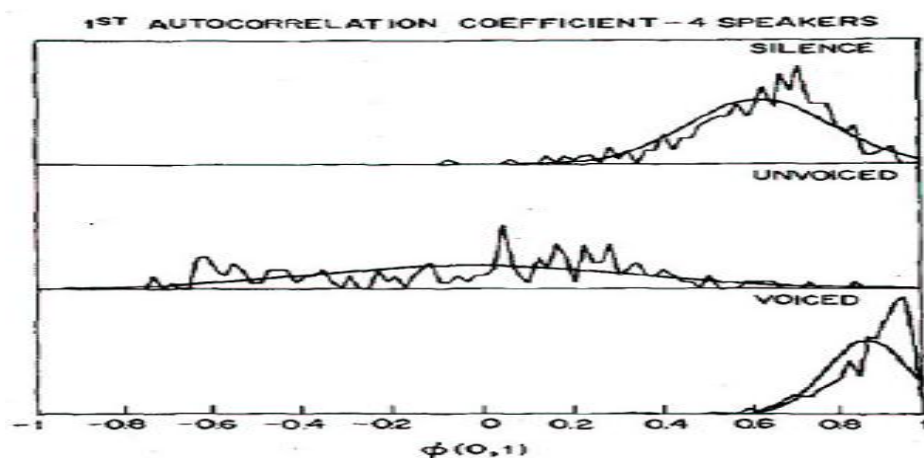
برنامه این بخش هم به دو قسمت تقسیم شده است، برنامه p4_1 مربوط به تعیین VUS با استفاده از پارامترهای زمان-کوتاه محاسبه شده در بخش ۲ است و برنامه p4_2 مربوط به روش گفته شده در فصل الگوریتم‌ها در بخش "A Bayesian approach ..." است.

هیستوگرام مربوط به انرژی و عبور از صفر طبق کتاب به شکل زیر است:



شکل ۱۸ و ۱۹- هیستوگرام Log Energy و Zero-crossing برای فنوم های silence, unvoiced, voiced

همچنین شکل تابع خود همبستگی (کواریانس) به شکل زیر است:



شکل ۲۰- هیستوگرام $\phi(0,1)$ برای فنوم های silence, unvoiced, voiced

چون در اینجا ما نمونه آموزشی نداریم به ناچار از همان سیگنال اصلی هم به عنوان نمونه آموزشی و هم به عنوان نمونه تست استفاده می‌کنیم؛ یعنی از نتیجه بخش ۱ استفاده می‌کنیم تا مقدار آستانه برای انرژی و عبور از صفر را پیدا کنیم و نتایج را روی فریم‌ها اعمال می‌کنیم. برای پیدا کردن محل هر فریم در سیگنال اصلی به صورت زیر عمل می‌کنیم:

تعداد نمونه‌های بدون همپوشانی در هر فریم ۲۲۱ نمونه است و تعداد کل فریم‌ها ۵۷ تا است که برای کامل شدن فریم‌ها به انتهای سیگنال صفر اضافه شده پس فریم آخر به دلیل عمل **zero-padding** خیلی معتبر نیست چون ممکن است تعداد صفرهای اضافه شده به آن نسبت به تعداد نمونه‌های فریم زیاد باشد.

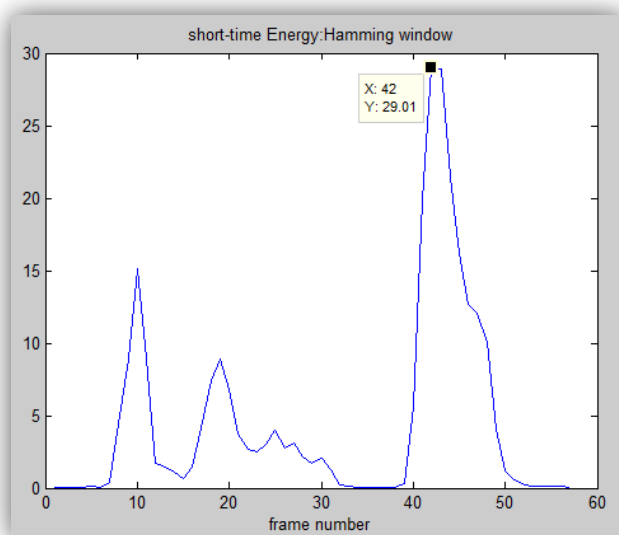
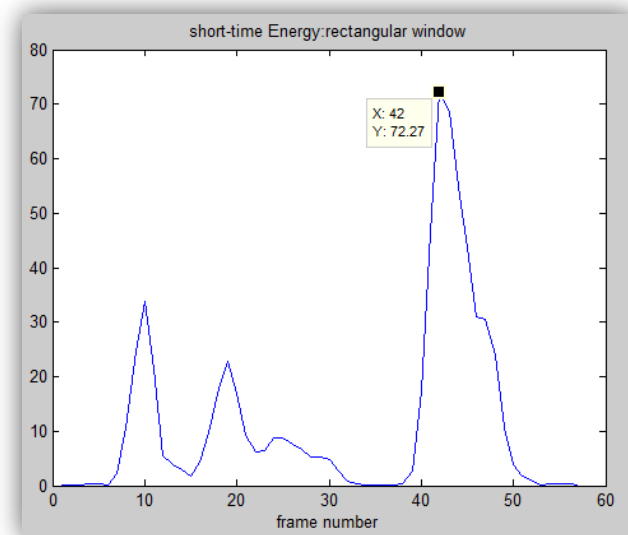
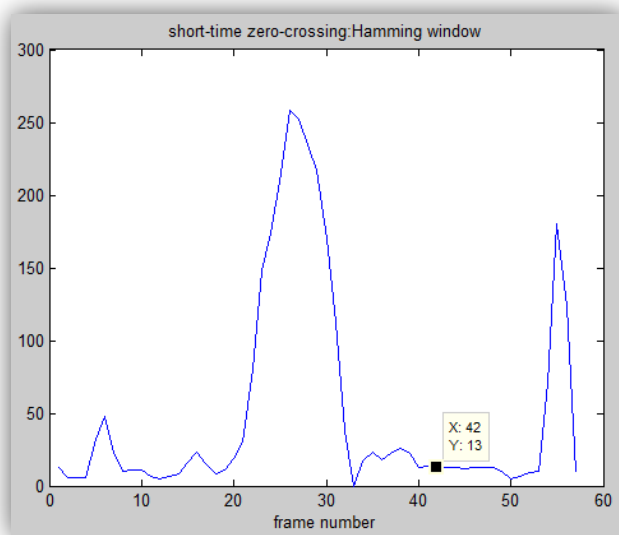
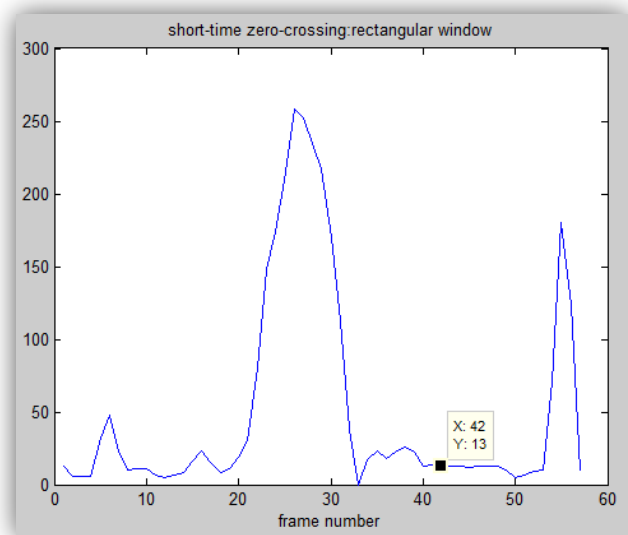
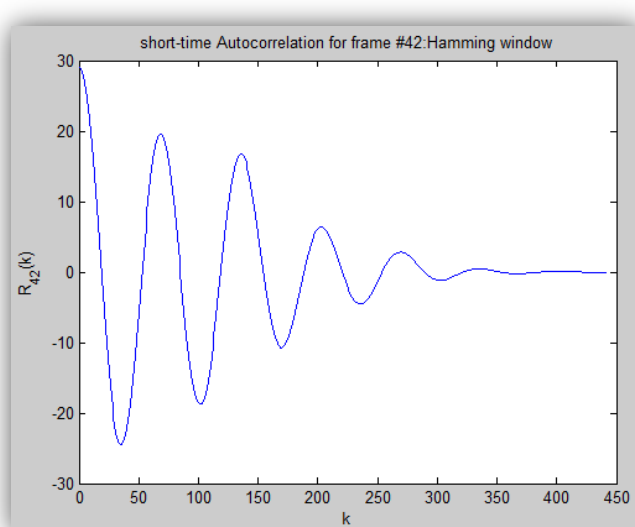
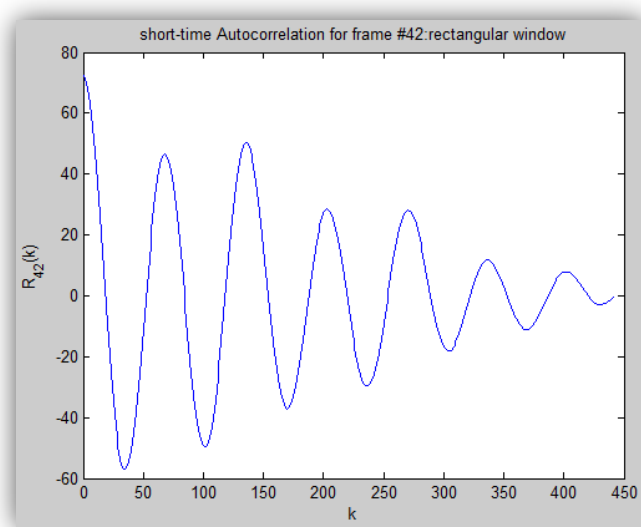
شماره فریم: i

اندیس اولین نمونه فریم در سیگنال اصلی: $(i-1)*221+1$

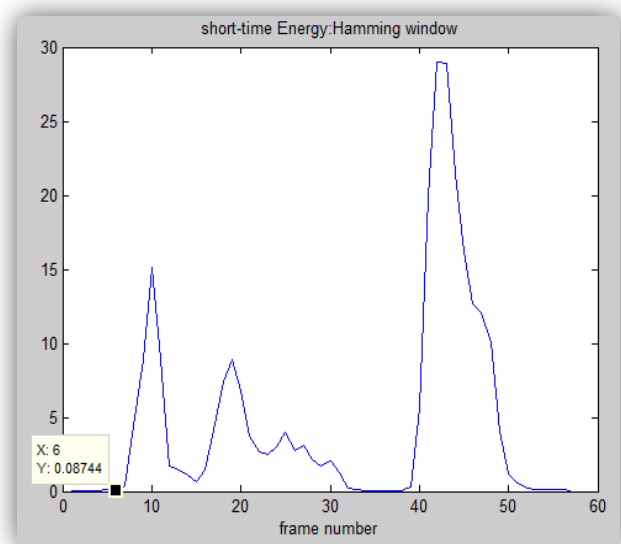
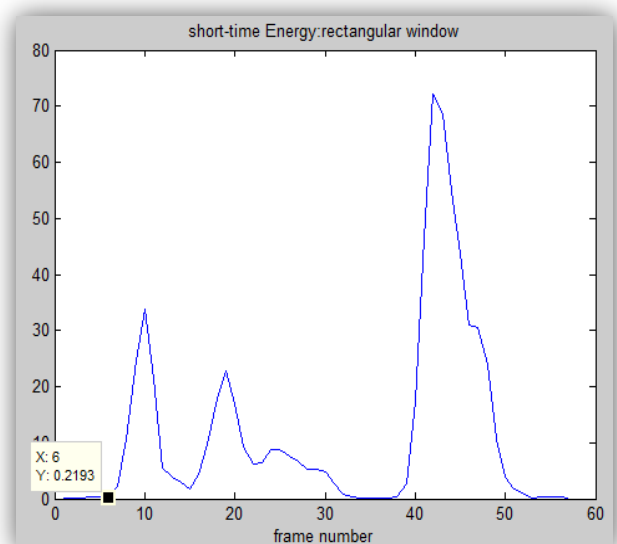
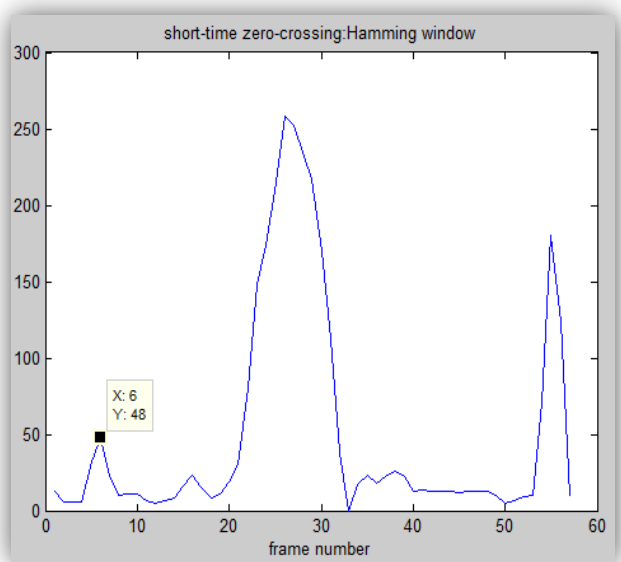
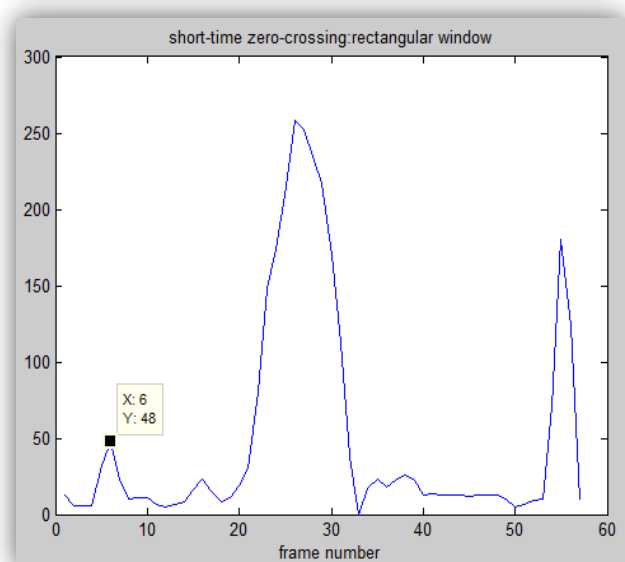
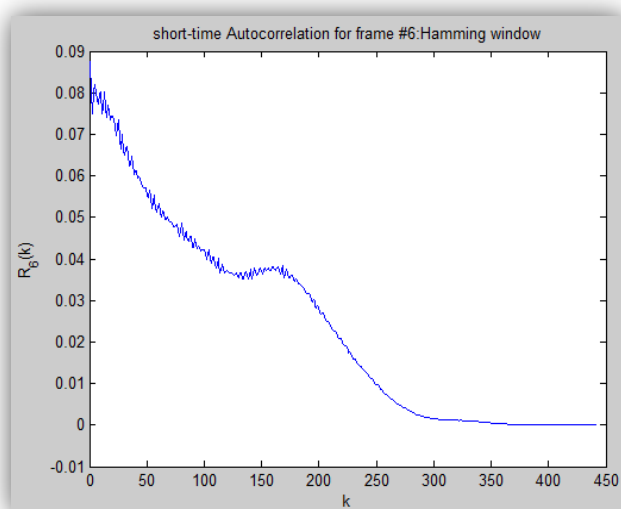
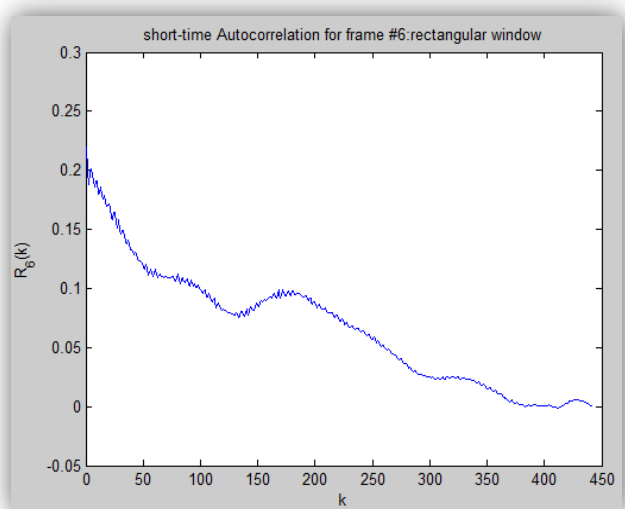
اندیس آخرین نمونه فریم در سیگنال اصلی: $(i+1)*221-1$

برای مثال فریم شماره ۴۲ معادل با نمونه‌های ۹۰۶۲ تا ۹۵۰۲ سیگنال اصلی است که اگر به بخش ۱ مراجعه کنید می‌بینید مربوط به فنوم صدادار i است. در شکل‌های زیر انرژی، عبور از صفر و خود همبستگی مربوط به فریم ۴۲ را که از بخش ۲ به دست آمده است می‌بینید:

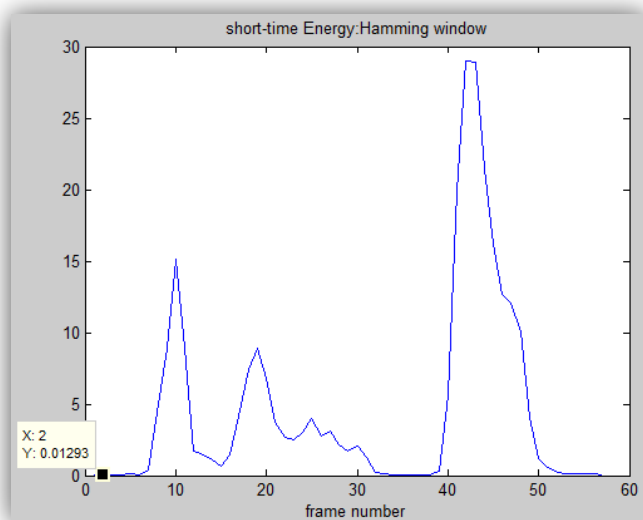
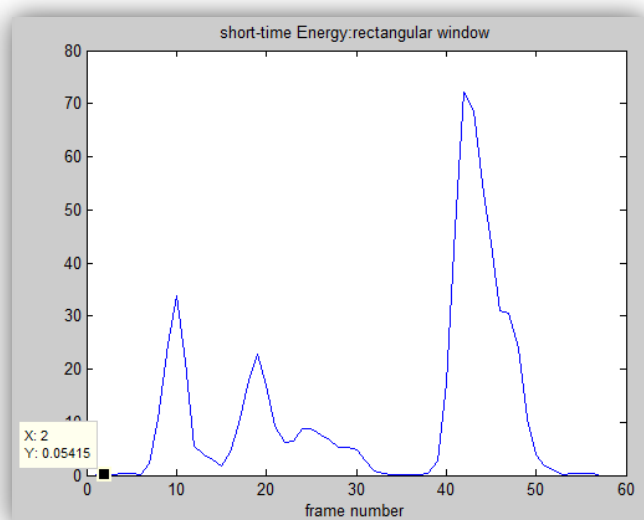
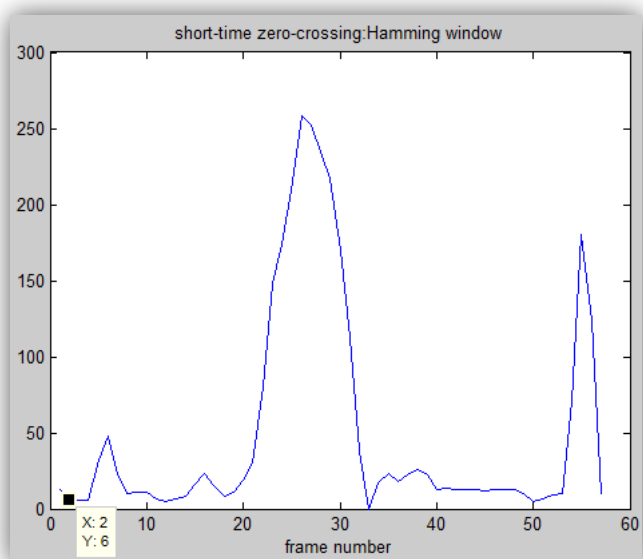
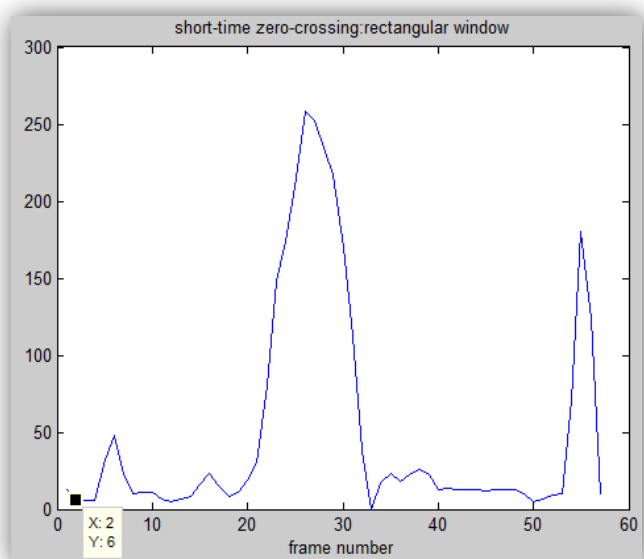
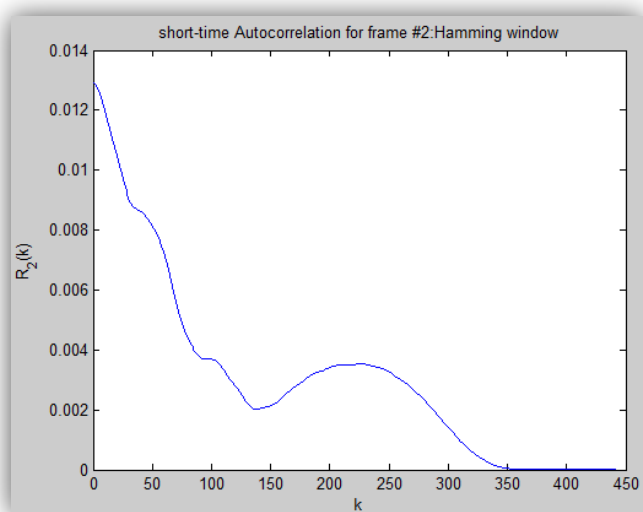
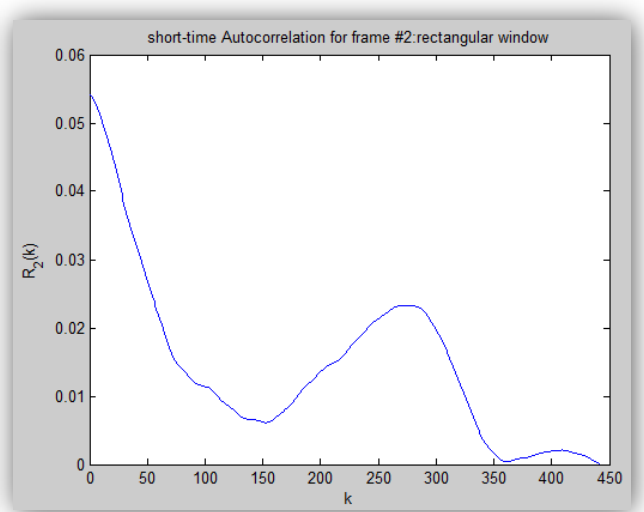
در این شکل‌ها چند نکته تأیید می‌شود، انرژی در فریم‌های مربوط به فنوم‌های صدادار بیشتر است، نرخ عبور از صفر در فنوم‌های صدادار کم و در فنوم‌های بی صدا زیاد است (با مقایسه با مجموعه شکل‌های بعدی که مربوط به یک فنوم صدادار است معلوم می‌شود)، خود همبستگی با پنجره مستطیلی شکل بهتری دارد چون بر خلاف پنجره همینگ دامنه را در ابتدا و انتهای فریم کاهش نمی‌دهد. نرخ عبور از صفر برای هر دو پنجره همینگ و مستطیلی یکسان است چون فقط علامت اهمیت دارد و نوع پنجره در علامت هیچ تأثیری ندارد. پنجره همینگ منجر به شکل انرژی زمان کوتاه تیزتری می‌شود. برای فنوم‌های صدادار شکل تابع خود همبستگی تقریباً متناوب است.



شکل ۲۱ تا ۲۶- انرژی، عبور از صفر و خود همبستگی برای فریم شماره ۴۲ مربوط به فنوم صدادار i:



شکل ۲۷ تا ۳۲- انرژی، عبور از صفر و خود همبستگی برای فریم شماره ۶ مربوط به فنوم بی صدای th

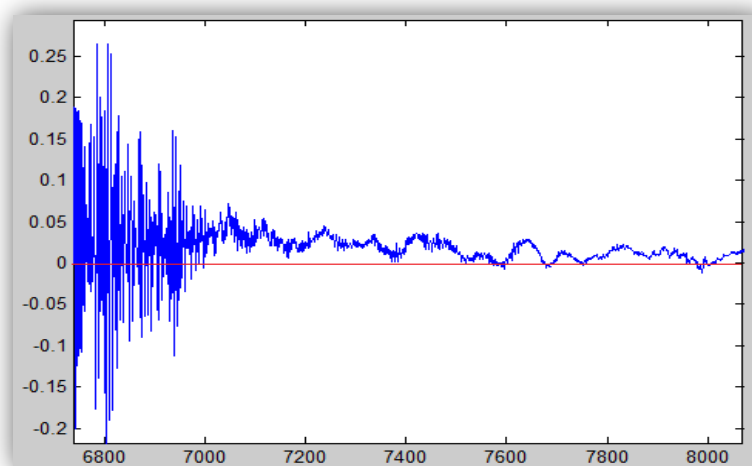


شکل ۳۳ تا ۳۸- انرژی، عبور از صفر و خود همبستگی برای فریم شماره ۲ مربوط به ناحیه سکوت

ابتدا می‌خواهیم با آستانه گذاری روی مقادیر انرژی و نرخ عبور از صفر نوع هر فریم را مشخص کنیم. با استفاده از جدول ۲ در بخش ۲ این مقادیر آستانه را به دست می‌آوریم؛ بدین منظور برای فریم‌هایی که فقط voiced هستند مقادیر Energy و Zero-crossing را با فریم‌هایی که فقط unvoiced و همچنین فریم‌هایی که فقط silence هستند مقایسه می‌کنیم.

برای فریم‌های voiced حداقل مقدار انرژی 1.11 و حداکثر آن 29.01 است. این مقادیر برای فریم‌های unvoiced برابر با 0.02 و 3.96 و برای فریم silence برابر با 0.01 و 0.08 است.

در مورد پارامتر تعداد عبور از صفر برای فریم‌های voiced دارای حداقل و حداکثر ۵ و ۱۹ است. برای فریم‌های unvoiced این مقادیر ۰ و ۲۵۸ هستند و برای فریم‌های silence هم ۶ و ۱۳. علت پایین بودن تعداد عبور از صفر فریم‌های unvoiced عدم حذف قسمت DC سیگنال است که در شکل زیر دیده می‌شود:



شکل ۳۹- اثر مؤلفه DC در پارامتر ZC

بعد از حذف مؤلفه DC سیگنال مجدداً مقادیر انرژی و عبور از صفر محاسبه شده‌اند و در جدول ۳ آمده است. بعد از این اصلاح مقادیر مربوط به Zero-crossing برای فریم‌های unvoiced که قبلاً خیلی کوچک شده بود، بیشتر شده است. مقادیر جدید حداقل و حداکثر مقادیر مربوط به پارامترهای انرژی و عبور از صفر برای فریم‌های voiced, unvoiced, silence در جدول ۴ آمده است.

جدول ۳- مقادیر انرژی و عبور از صفر پس از حذف DC

Frame #	Zero-crossing	Energy
1	6	0.02
2	8	0.02
3	7	0.04
4	10	0.03
5	47	0.05
6	85	0.03
7	48	0.38
8	10	4.43
9	13	8.85
10	11	15.10
11	7	8.92
12	5	1.62
13	7	1.39
14	6	0.98
15	12	0.53
16	23	1.44
17	17	4.18
18	10	7.44
19	13	8.85
20	15	6.67
21	28	3.56
22	82	2.57
23	148	2.42
24	178	3.00
25	217	3.89
26	262	2.65
27	254	3.06
28	232	2.20
29	209	1.71
30	166	2.04
31	131	1.18
32	78	0.11
33	47	0.02
34	33	0.02

35	22	0.02
36	26	0.02
37	41	0.03
38	61	0.07
39	37	0.26
40	13	5.61
41	14	19.80
42	13	28.94
43	13	28.86
44	13	21.40
45	12	16.20
46	13	12.68
47	13	12.04
48	13	10.00
49	10	3.97
50	7	1.13
51	7	0.46
52	10	0.23
53	14	0.06
54	69	0.09
55	173	0.12
56	127	0.15
57	16	0.00

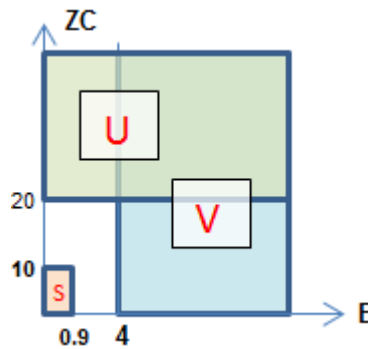
جدول ۴- مقادیر حداقل و حداکثر پارامترهای انرژی و عبور از صفر

	Energy		Zero-crossing	
	Min	Max	Min	Max
voiced	0.98	28.94	5	17
unvoiced	0.02	3.89	22	262
silence	0.02	0.04	6	8

با توجه به مقادیر جدول ۴ می‌توان مقادیر آستانه را به صورت زیر تعیین کرد:

- اگر در یک فریم مقدار انرژی از ۴ بیشتر باشد، آن فریم مربوط به یک فنوم voiced است.
- اگر مقدار انرژی از ۰.۹ کمتر باشد و مقدار Zero-crossing از ۱۰ کمتر باشد، فریم مربوط به silence است.
- اگر Zero-crossing از ۲۰ بیشتر باشد، فریم مربوط به فنوم unvoiced است.

اگر این قواعد را به صورت شکل زیر نشان دهیم، باید تکلیف نواحی مبهم را هم مشخص کنیم:



شکل ۴۰- آستانه‌های تصمیم‌گیری

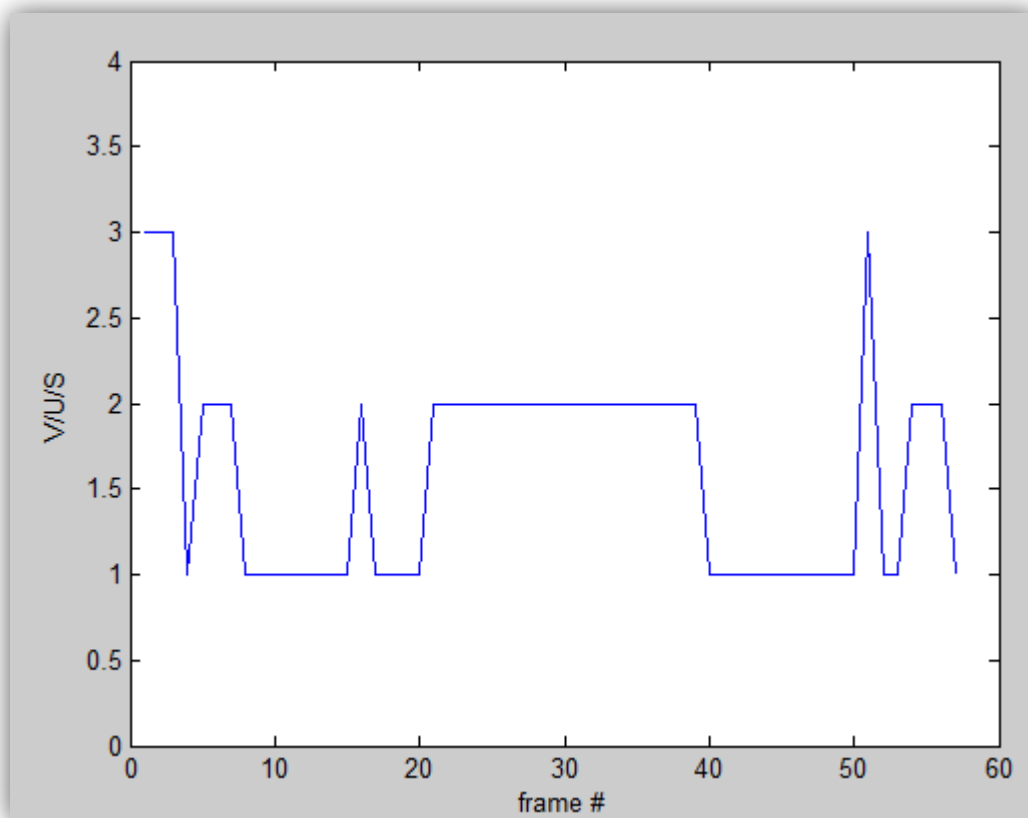
الگوریتم را به صورت زیر در نظر می گیریم:

- ۱- اگر ZC از ۲۰ بیشتر باشد: unvoiced در غیر این صورت
- ۲- اگر E بیشتر از ۴ باشد: voiced در غیر این صورت
- ۳- اگر انرژی از ۰.۹ کمتر باشد و ZC کمتر از ۱۰ باشد: silence در غیر این صورت: voiced

در این صورت ابهامی هم نمی ماند. بعد از اجرای برنامه، نتیجه برای فریم های ۱ تا ۵۷ به ترتیب از چپ به راست به صورت زیر است:

[illegible]

با مقایسه این نتیجه با طبقه بندی دستی که در جدول ۲ آمده است و در نظر گرفتن فریم‌های کاملاً مشخص (فریم‌هایی که فقط متعلق به یک نوع فنوم هستند که آن فنوم یا V یا U یا S است) مشخص می‌شود که این برنامه ۱۰۰٪ جواب داده است و هیچ خطایی ندارد. البته همان طور که قبلاً هم ذکر شد داده‌های آموزشی و آزمایشی یکسان هستند و اگر همین برنامه برای قسمت‌های بعدی همان فایل صوتی که مورد استفاده قرار نگرفته است (به دلیل طولانی بودن و وقت گیر شدن طبقه بندی دستی) استفاده شود احتمال بروز خطا وجود دارد.



شکل ۴۱- نمایش گرافیکی نوع فریم‌ها

در برنامه P4_2 از روش طبقه بندی بیزین بر اساس کلاسیفایر Nearest neighbor استفاده شده است. با داشتن نوع فریم‌ها که از بخش ۱ به دست آمده است از ویژگی‌های همین فریم‌ها (فقط فریم‌هایی که از نوع آن مطمئن هستیم) به عنوان داده آموزشی استفاده می‌کنیم. سه کلاس داریم: voiced, unvoiced, silence. برای هر کدام از این کلاس‌ها تعدادی فریم داریم که باید ویژگی از آن‌ها استخراج کنیم. ۵ ویژگی از هر فریم استخراج می‌شود. با در نظر گرفتن هر کلاس به عنوان یک خوشه در فضای ۵ بعدی می‌توانیم هر نقطه ۵ بعدی جدید که به عنوان داده تست وارد می‌شود را به نزدیک‌ترین خوشه (کلاس) نسبت دهیم.

این ۵ ویژگی عبارتند از:

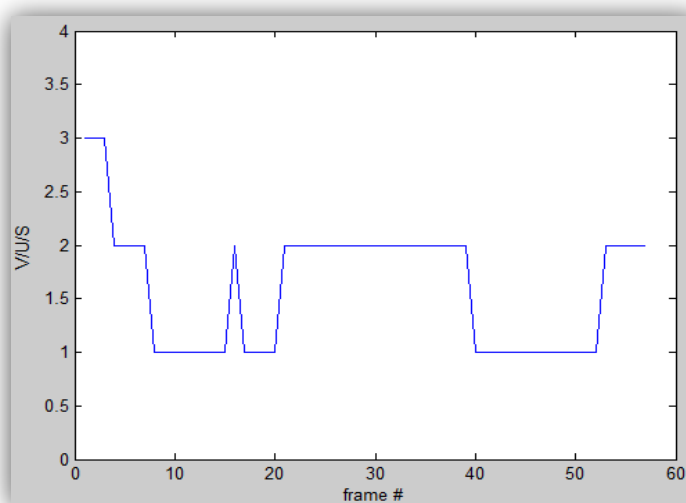
- ۱- لگاریتم انرژی
- ۲- عبور از صفر
- ۳- اولین ضریب تابع خود همبستگی

۴- اولین ضریب LPC برای $p=12$

۵- لگاریتم خطای LPC

هر کلاسی که فاصله مهالنویس آن از داده جدید کمتر باشد به داده جدید نسبت داده می‌شود.

نتیجه برنامه برای همان داده آموزشی به صورت زیر است که باز هم طبقه بندی بدون خطا صورت گرفته است.

[illegible]

شکل ۴۲- نمایش گرافیکی VUS برای روش طبقه بندی بیزین

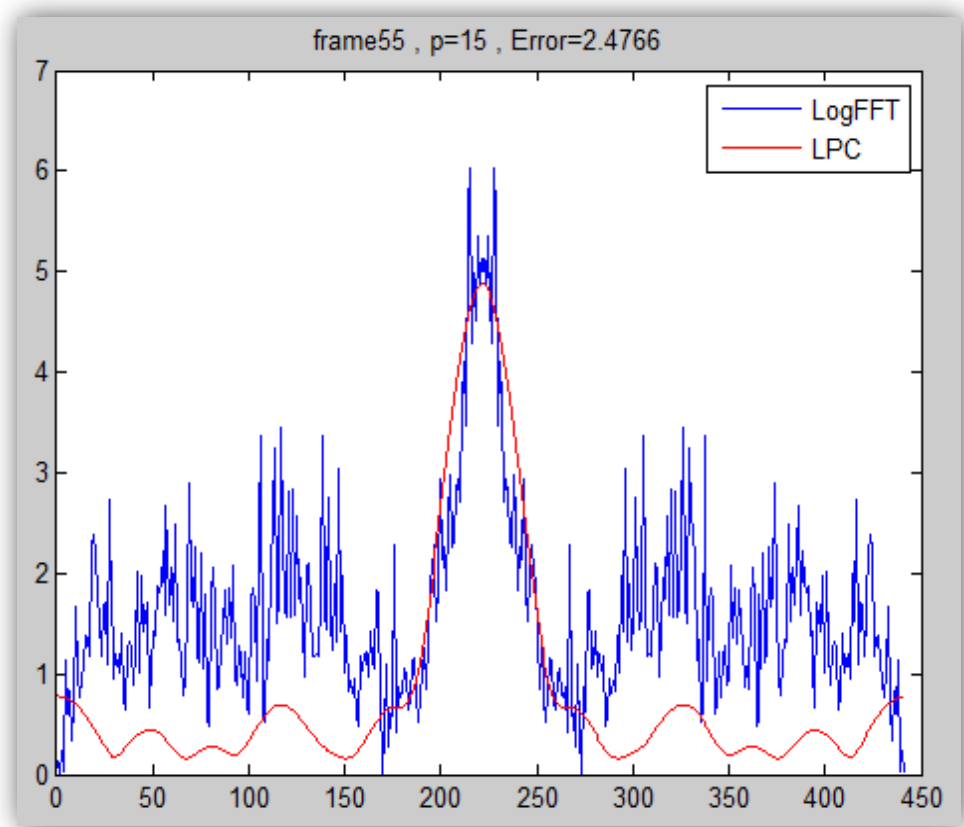
محاسبه FFT و LogFFT به صورت مستقیم و با استفاده از ضرایب کپسترال و LPC

استفاده از حوزه فرکانس برای سیگنال‌های صحبت اطلاعات مفیدی به ما می‌دهد، در بخش ۳ که ضرایب کپسترال مل را محاسبه کردیم از تبدیل فوریه در محاسبه کپستروم استفاده شد ولی هنوز از این اطلاعات استفاده نکرده‌ایم، یکی دیگر از اطلاعات مفید حوزه فرکانس FFT یا خود تبدیل فوریه سیگنال است. با گرفتن Log از FFT هم LogFFT به دست می‌آید. البته چون FFT دارای مقدار مختلط است یا باید از قسمت‌های حقیقی و موهومی آن جدا لگاریتم گرفت که نتیجه این عمل بعد از گرفتن معکوس تبدیل فوریه را **Complex Cepstrum** می‌گوییم ولی اگر فقط از اندازه تبدیل فوریه لگاریتم بگیریم و بعد تبدیل معکوس فوریه بگیریم به آن **Real Cepstrum** یا به اختصار **Cepstrum** می‌گوییم.

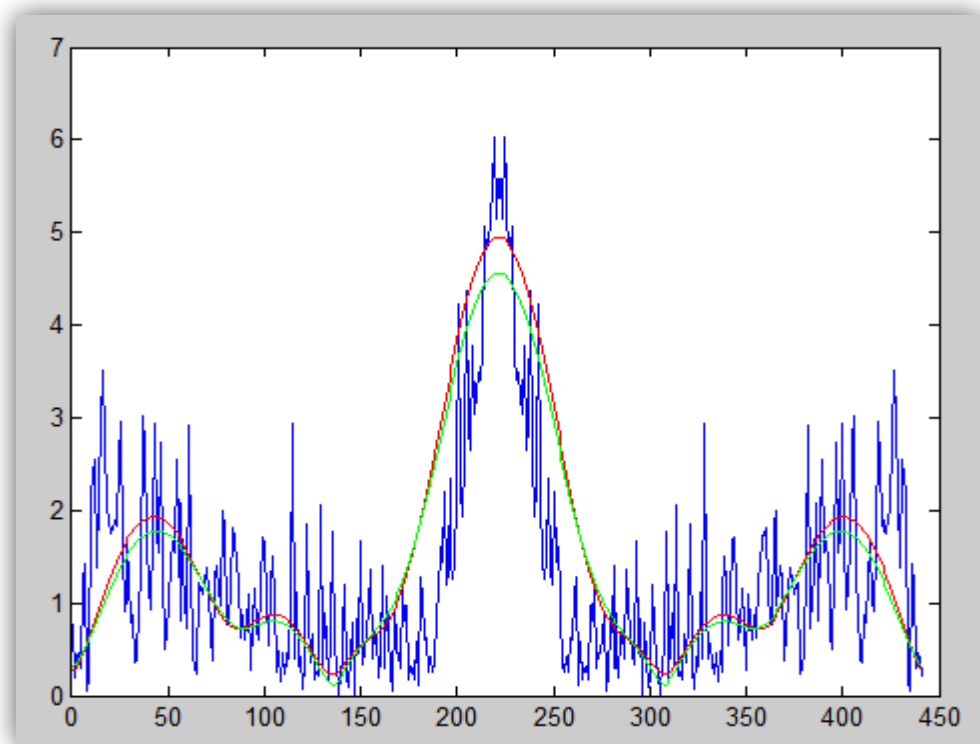
گرفتن FFT و LogFFT از یک سیگنال کار ساده ایست که با دستور `fft` در برنامه **Matlab** امکان پذیر است. مرحله اصلی کار محاسبه این مقادیر با استفاده از ضرایب کپستروم مل و LPC است که در بخش ۳ محاسبه شدند. چون در محاسبه مل و LPC از تقریب استفاده کرده‌ایم و یک سیگنال با ۴۴۱ نمونه (طول پنجره یا فریم) را به سیگنالی با طول بسیار کمتر (مقادیر ۱۰، ۱۲ و ۱۵) تبدیل کرده‌ایم پس در راه بازگشت خطا وجود دارد که در این برنامه مقدار این خطا را هم محاسبه می‌کنیم.

اندازه تبدیل فوریه ضرایب LPC تخمینی از LogFFT است، این تقریب برای همه فریم‌ها و برای تعداد ضرایب $p=10,12,15$ محاسبه شده است. مقدار خطای متوسط برای هر نمونه هم محاسبه شده است. برای یکی از فریم‌ها که مقدار این خطا کم بوده است، شکل این دو سیگنال رسم شده است.

(برای ضرایب مل، تقریب LogFFT از روی ضرایب کپستروم مل؟)



شکل ۴۳- LogFFT و تقریب آن از روی ضرایب LPC



شکل ۴۴- نمودار آبی: LogFFT، سبز: تقریب از روی LPC با پنجره مستطیلی، قرمز: با پنجره همینگ

نمایش صفحه زمان-فرکانس سیگنال صحبت و تعیین **pitch, formant** و نویز روی آن

با استفاده از دستور **spectrogram** برنامه **Matlab** به فرمت زیر:

spectrogram(x,window,noverlap,nfft,fs)

x سیگنالی که می‌خواهیم **spectrogram** آن را نمایش بدهیم.

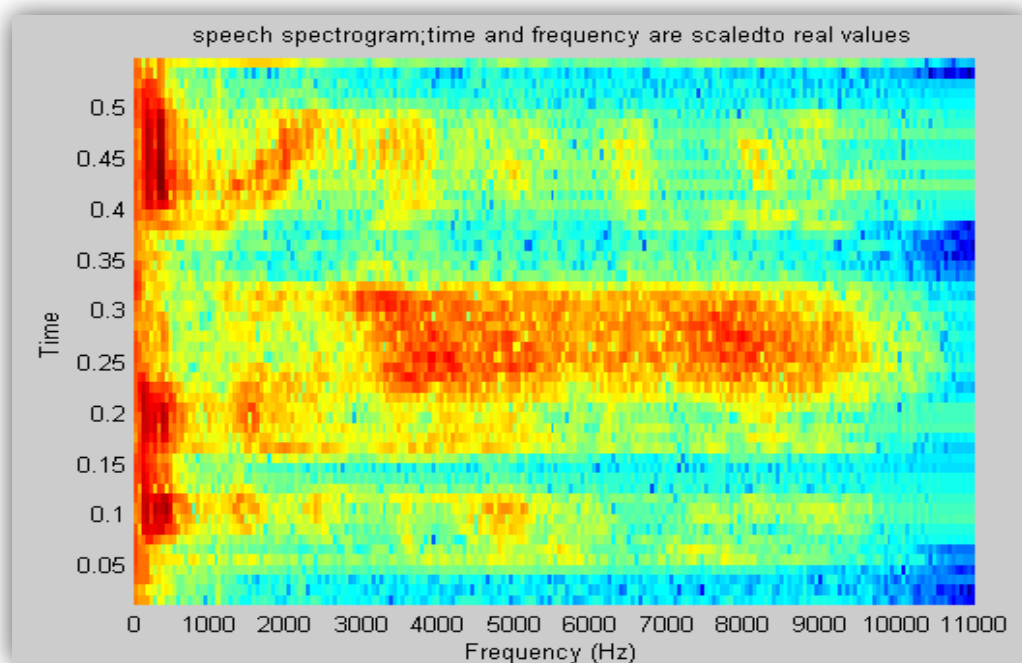
window اندازه پنجره در **STFT**

noverlap تعداد نمونه‌های با همپوشانی با پنجره مجاور

nfft تعداد نقاط برای گرفتن **FFT**

fs فرکانس نمونه گیری

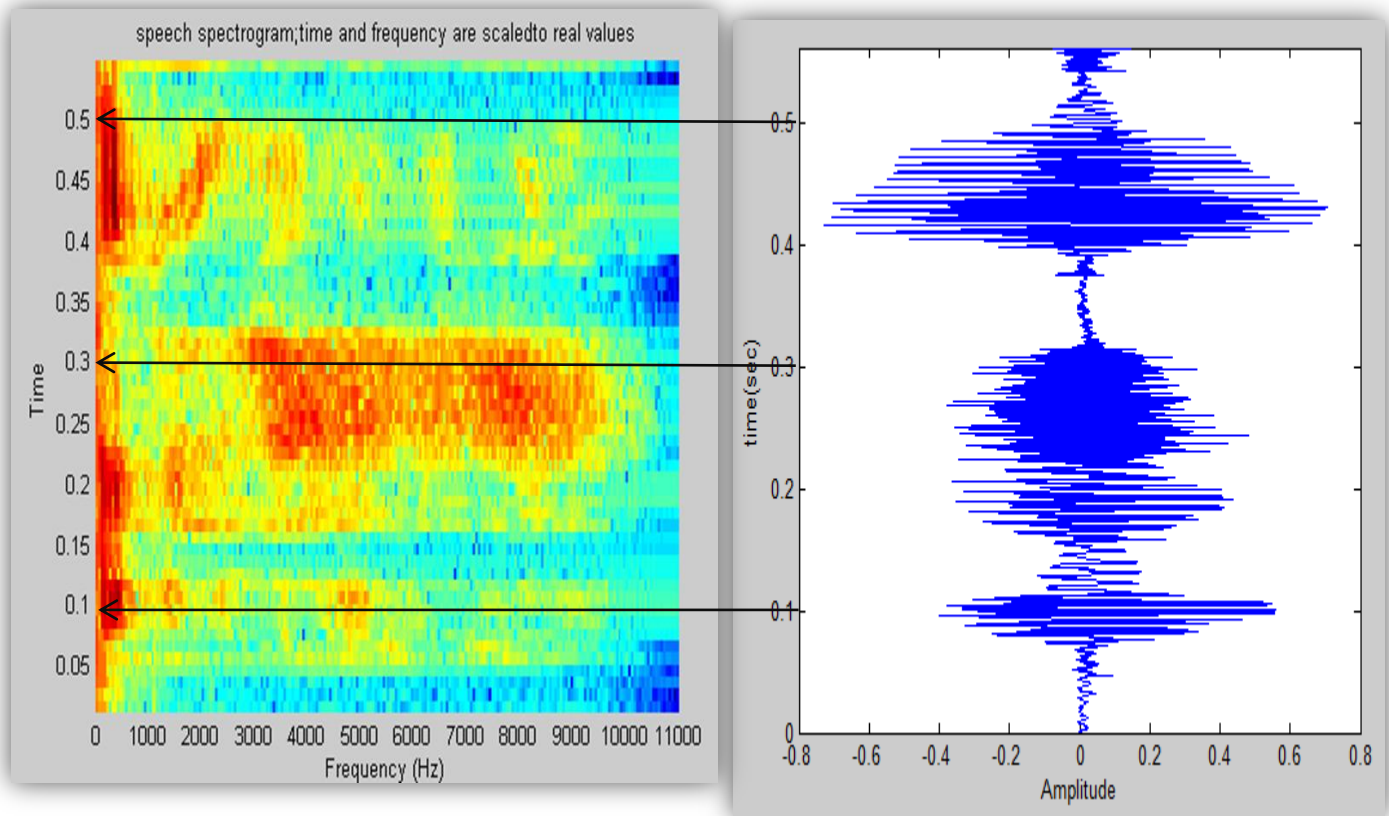
شکل زیر به دست آمده است.



شکل ۴۵- نمودار زمان-فرکانس سیگنال صحبت

چون فرکانس نمونه گیری به دستور داده شده است، زمان‌ها و فرکانس‌های روی نمودار واقعی‌اند.

در شکل‌های زیر هم محورهای زمان صفحه زمان-فرکانس و شکل موج سیگنال، موازی قرار داده شده‌اند تا مقایسه راحت‌تر باشد.



شکل ۴۶- مقایسه spectrogram و شکل موج زمانی سیگنال

چون فرمت‌ها معادل پیک‌های FFT هستند، در نمودار بالا در بازه‌های زمانی کوتاه جاهایی که دامنه بیشتری دارند (رنگ‌های گرم‌تر) همان پیک‌های FFT هستند که تقریباً فرمت‌ها را مشخص می‌کنند.

بازه‌های زمانی که در فرکانس‌های بالا مقادیر کوچک دارند (رنگ سرد) نواحی silence و voiced هستند،

برای نواحی voiced فرکانس pitch فرکانسی است که دارای بیشترین دامنه است که در نمودار به خوبی قابل تشخیص نیست.

این سیگنال تقریباً بدون نویز است ولی در صورت وجود در نواحی silence مقادیر بالا در فرکانس‌های بالا معادل نویز است.

تعیین formant از روی پیک‌های FFT

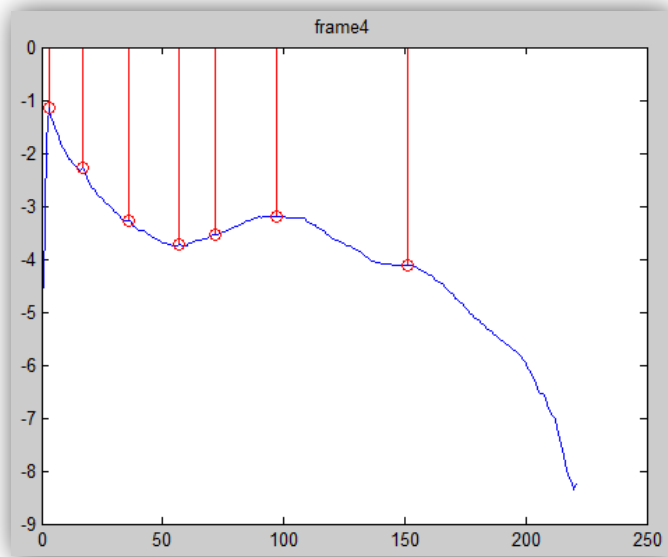
پیک‌های FFT فریم‌های سیگنال صوتی محل فرمنت‌ها را مشخص می‌کند. فرمنت‌ها ریشه‌های تابع تبدیل $H_v(z)$ یا $H_u(z)$ هستند که مدلی برای **Vocal Tract** هنگام تولید اصوات صدادار و بی صدا هستند. هر چه تعداد این قطب‌ها را بیشتر بگیریم مدل دقیق‌تری برای **Vocal Tract** به دست می‌آید. همچنین **فرمنت** ها برابر با ضرایب **LPC** هستند بنابراین می‌توان گفت تعداد مناسب فرمنت‌ها برابر با مقدار مناسب برای p در **LPC** است.

در برنامه این بخش برای فریم‌های صوت که در بخش ۲ به دست آمد، مقدار **LogFFT** را محاسبه شده است. چون این تابع دارای تغییرات کوچک خیلی زیادی است با استفاده از دستور **smooth** برنامه **Matlab** شکل این تابع نرم‌تر شده است. سپس پیک‌های **LogFFT** با استفاده از دستور **findpeaks** استخراج شده است و با تنظیم پارامتر **minpeaksdistance** حداقل فاصله بین پیک‌ها ۱۰ نمونه در نظر گرفته شده است. تعداد نمونه‌های **LogFFT** برای هر فریم **۲۲۱** نمونه است در حالی که تعداد نمونه‌های **هر فریم ۴۴۱** نمونه است که به دلیل تقارن تبدیل فوریه سیگنال حقیقی است. چون تعداد این پیک‌ها زیاد است و ما فقط حداکثر ۱۲ فرمنت نیاز داریم پس مقادیر پیک‌ها به ترتیب نزولی مرتب شده است و فقط ۱۲ مقدار اول آن برداشته شده است. سپس نمودار **LogFFT** و محل پیک‌ها که برابر با فرمنت‌هاست نمایش داده شده است. برای تبدیل مقادیر محور افقی به فرکانس واقعی، لازم است حداکثر مقدار اندیس محور افقی یعنی ۲۲۱ به فرکانس نمونه‌گیری که ۲۲۰۵۰ است تبدیل شود و بقیه مقادیر هم با همین نسبت تغییر کنند.

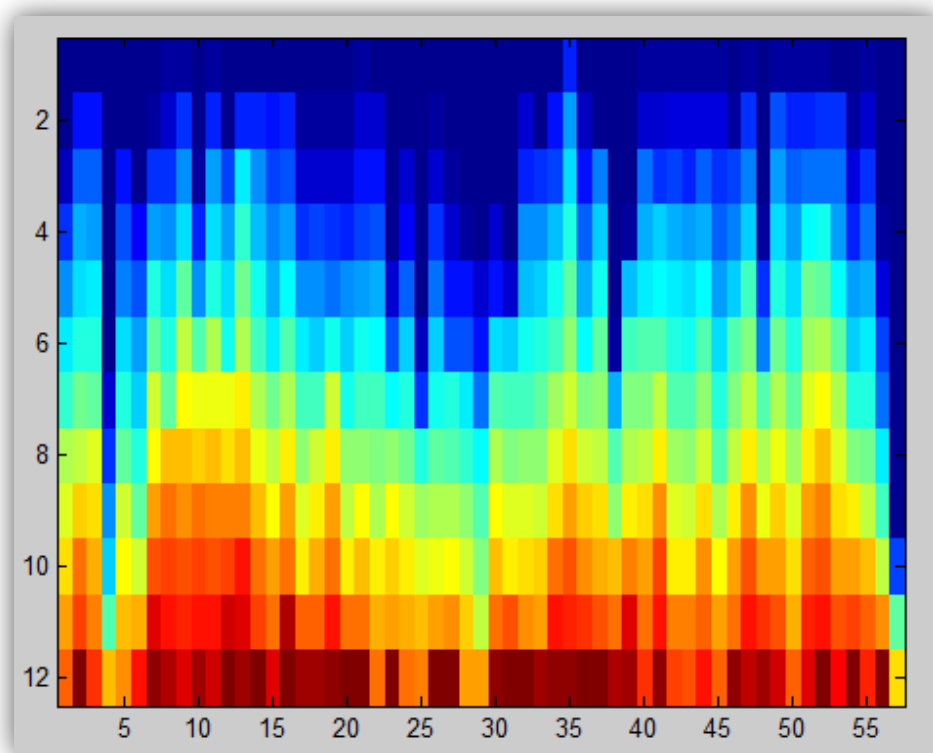
در نهایت مقادیر فرمنت بر حسب فرکانس برای هر فریم در ماتریس **Formants** ذخیره شده است.

شکل ۴۷ نمودار **LogFFT** و پیک‌های آن را برای یکی از فریم‌ها نشان می‌دهد.

شکل ۴۸ هم نمایشی از مقادیر فرمنت‌ها در کل فریم‌هاست، رنگ‌های گرم‌تر معادل مقادیر بیشتر است. محور افقی شماره فریم‌ها و محور عمودی ۱۲ فرمنت استخراج شده است.



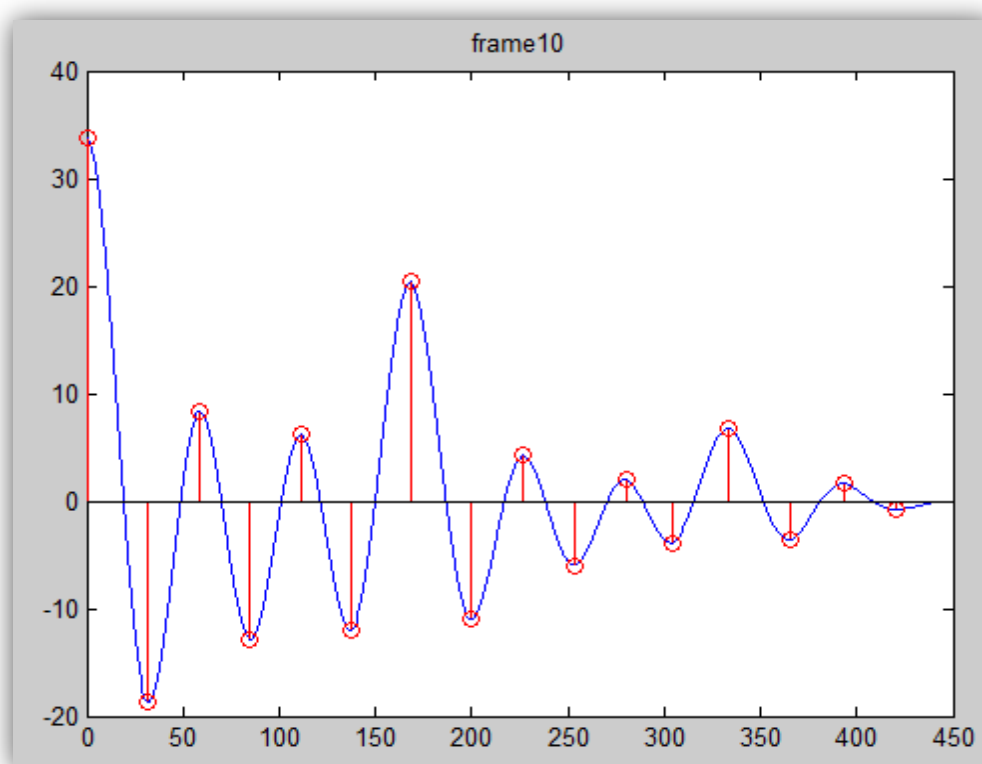
شکل ۴۷- نمایش LogFFT یک فریم و محل پیک‌ها معادل فرمنت های فریم



شکل ۴۸- نمایش مقدار کیفی فرمنت ها برای همه فریم‌ها

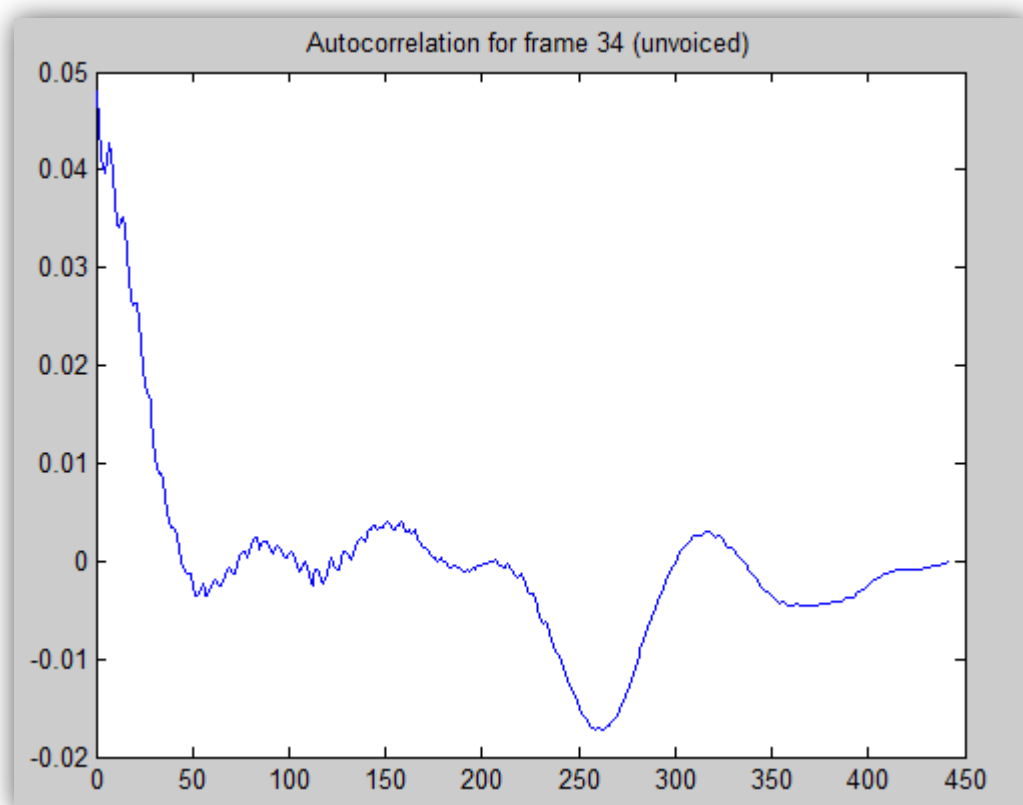
الگوریتمی برای محاسبه pitch برای فریم‌های voiced

در این بخش الگوریتمی برای محاسبه پریود pitch پیاده سازی می‌شود. در برنامه این بخش فقط پریود pitch برای فریم‌های شامل فنوم voiced محاسبه شده است. برای تعیین pitch روش‌های مختلفی وجود دارد که در اینجا از روش Autocorrelation استفاده شده است. تابع خود همبستگی برای فریم‌های voiced تقریباً شکل متناوب دارد. شکل زیر تابع خود همبستگی را برای یک فریم voiced نشان می‌دهد. در این برنامه از short-time Autocorrelation با پنجره مستطیلی استفاده شده است، چون همان طور که قبلاً ذکر شد پنجره همینگ برای Autocorrelation مناسب نیست چون روی اندازه پیک‌ها در ابتدا و انتهای فریم زمانی تأثیر زیادی دارد که منجر به غیر متناوب شدن تابع خود همبستگی برای مقادیر بالای k می‌شود.



شکل ۴۹- تابع خود همبستگی برای یک فریم شامل فنوم voiced

در حالی که برای یک فریم unvoiced شکل نامنظمی مانند زیر دارد.



شکل ۵۰- تابع خود همبستگی برای یک فریم unvoiced

پریود **pitch** فاصله بین دو پیک مثبت یا منفی تابع خود همبستگی است. مقادیر short-time Autocorrelation در بخش ۲ محاسبه شده‌اند و نوع فریم‌ها هم در بخش ۱ تعیین شده است.

برای محاسبه مکان پیک‌ها در Autocorrelation از دستور **findpeaks** برنامه Matlab استفاده شده است. این برنامه پیک واقع در نمونه اول را استخراج نمی‌کند به همین دلیل مقدار ۱ به ابتدای بردار مربوط به مکان پیک‌ها اضافه شده است. این دستور فقط پیک‌های مثبت را استخراج می‌کند، برای استخراج پیک‌های منفی کافیست پیک‌های مثبت را برای منفی بردار مورد نظر پیدا کنیم.

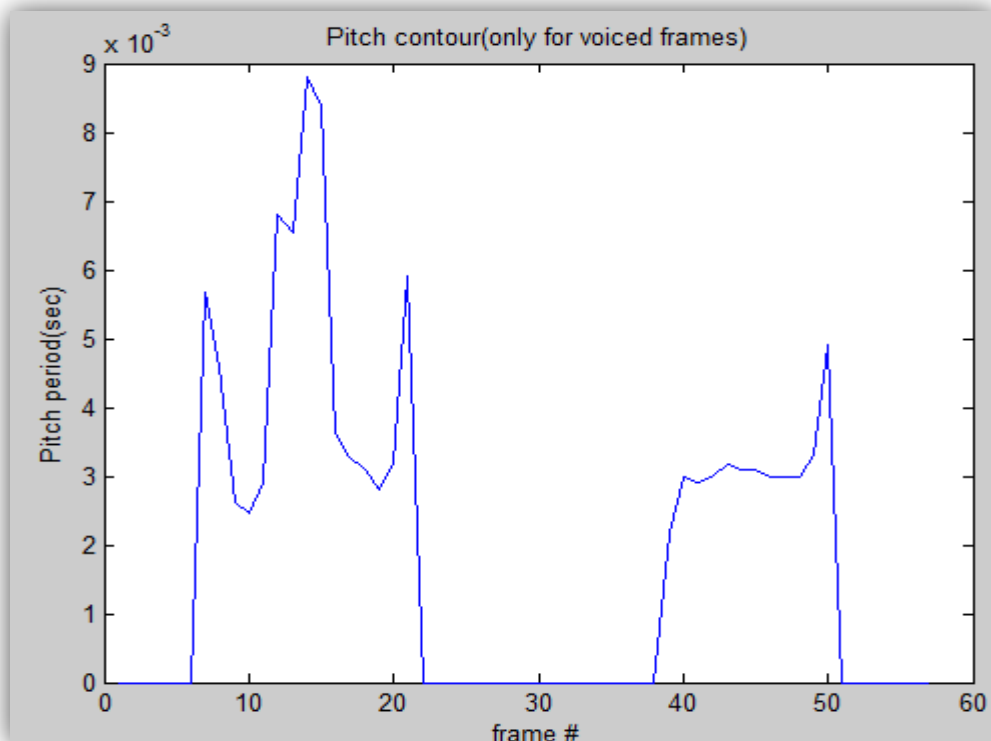
پارامتر **MINPEAKDISTANCE** در این دستور حداقل فاصله بین پیک‌ها تنظیم می‌کند که به طور تجربی مقدار ۲۰ (معادل با **0.9 msec**) مناسب تشخیص داده شده است، یعنی مقدار پریود **pitch** هیچ فنوم voiced از این مقدار کمتر نیست.

مکان پیک‌های مثبت در متغیر **locsp** و مکان پیک‌های منفی در متغیر **locsn** ذخیره شده‌اند. همه این مکان‌ها باید در کنار هم قرار بگیرند تا هم پیک‌های مثبت و هم پیک‌های منفی را داشته باشیم. خطوط ۱۷ تا ۲۵ برنامه این کار را انجام می‌دهند، همچنین در این قسمت کنترل شده است که فاصله بین پیک‌های مثبت و منفی مجاور از ۱۰ نمونه کمتر نشود.

سپس این مکان‌ها با دستور **sort** مرتب شده‌اند و مقدار ۱ که معادل اولین پیک است هم به بردار مکان پیک‌ها اضافه شده است.

پریود **pitch** برابر با فاصله بین پیک‌های هم علامت مجاور یا دو برابر فاصله بین پیک‌های مثبت و منفی مجاور است. چون پریود **pitch** یک مقدار کاملاً ثابت نیست و همچنین در ابتدا و انتهای فنوم **voiced** که مجاور با یک فنوم غیر **voiced** بوده است، تغییرات زیادی دارد از دستور **median** برای حذف این آثار استفاده شده است.

در نهایت کانتور **pitch** فریم‌های **voiced** رسم شده است.



شکل ۵۱- کانتور **pitch** برای فریم‌های **voiced**

کدهای برنامه‌ها

بخش ۱

```
% Part 1 - plot speech waveform, then determine each part as 'Voiced',
% 'Unvoiced' and 'Silence' ocular
clear;clc;close all

% 1- read speech .wav file

[s,fs]=wavread('speech_dft',[6101,18500]);
ls=length(s);
t=ls/fs*1000;

sound(s,fs)
disp('each figure shows one part of speech, times on figures are
approximate')
disp('total speech duration is')
disp(t);disp('msec');disp('spoken words: The discrete')
disp('      /S')
disp('Th   /U')
disp('e     /V')
disp('d     /U')
disp('I     /V')
disp('sk    /U')
disp('r     /U')
disp('i:    /V')
disp('t     /U')

% 2- plot waveform and 3- determine each part type

t1=1:2000;
s1=s(t1);
figure(1);plot(t1,s1)
title('about 1:1000 silence')
text(1,s(1),'\fontsize{12}\rightarrow/S','EdgeColor','red')
text(1000,s(1000),'\fontsize{12}\leftarrow/S','EdgeColor','red')
pause(1)

t2=1000:1800;
s2=s(t2);
figure(2);plot(t2,s2)
title('about 1000:1640 unvoiced (Th)')
text(1000,s(1000),'\fontsize{12}\rightarrow/U','EdgeColor','red')
text(1640,s(1640),'\fontsize{12}\leftarrow/U','EdgeColor','red')
pause(1)

t3=1500:3400;
s3=s(t3);
figure(3);plot(t3,s3)
title('about 1640:3370 voiced (e)')
text(1640,s(1640),'\fontsize{12}\rightarrow/V','EdgeColor','red')
text(3370,s(3370),'\fontsize{12}\leftarrow/V','EdgeColor','red')
```

```

pause(1)

t4=3300:3800;
s4=s(t4);
figure(4);plot(t4,s4)
title('about 3470:3650 unvoiced (d)')
text(3470,s(3470),'\fontsize{12}\rightarrow/U','EdgeColor','red')
text(3650,s(3650),'\fontsize{12}\leftarrow/U','EdgeColor','red')
pause(1)

t5=3200:5000;
s5=s(t5);
figure(5);plot(t5,s5)
title('about 3640:4680 voiced (I)')
text(3640,s(3640),'\fontsize{12}\rightarrow/V','EdgeColor','red')
text(4680,s(4680),'\fontsize{12}\leftarrow/V','EdgeColor','red')
pause(1)

t6=4500:8300;
s6=s(t6);
figure(6);plot(t6,s6)
title('about 4680:8270 unvoiced (sk)')
text(4680,s(4680),'\fontsize{12}\rightarrow/U','EdgeColor','red')
text(8270,s(8270),'\fontsize{12}\leftarrow/U','EdgeColor','red')
pause(1)

t7=8100:8900;
s7=s(t7);
figure(7);plot(t7,s7)
title('about 8270:8660 unvoiced (r)')
text(8270,s(8270),'\fontsize{12}\rightarrow/U','EdgeColor','red')
text(8660,s(8660),'\fontsize{12}\leftarrow/U','EdgeColor','red')
pause(1)

t8=8500:11500;
s8=s(t8);
figure(8);plot(t8,s8)
title('about 8820:11000 voiced (i:)')
text(8820,s(8820),'\fontsize{12}\rightarrow/V','EdgeColor','red')
text(11000,s(11000),'\fontsize{12}\leftarrow/V','EdgeColor','red')
pause(1)

t9=11000:1s;
s9=s(t9);
figure(9);plot(t9,s9)
title('about 11950:1s unvoiced (t)')
text(11950,s(11950),'\fontsize{12}\rightarrow/U','EdgeColor','red')
text(1s,s(1s),'\fontsize{12}\leftarrow/U','EdgeColor','red')

figure(10);plot(s);title('total signal')

```

```

% Part 2 - compute 'short-time energy', 'short-time Zero-crossing' and
% 'short-time autocorrelation' with different types of windows
clear;clc;close all

% 1- read speech .wav file

[s,fs]=wavread('speech_dft',[6101,18500]);
ls=length(s);
t=ls/fs*1000;
sound(s,fs)

% remove DC and hum before short-time processing

DC=sum(s)/ls;
s=s-DC;

% divide signal to 20 msec frames with 50% overlap

delta=.020; % window duration
overlap=50; % overlap percent between two neighbor frame
N=fs*delta; % number of window samples
nol=round(N*(100-overlap)/100); % number of not overlapped samples in each
frame
nw=floor(ls/nol)+1; % total number of frames

% define two types of windows:Hamming & rectangular

% Hamming window
hw=zeros(N,1);
for i=1:N
    hw(i)=.54-.46*cos(2*pi*i/N);
end
% rectangular window
rw=ones(N,1);

% compute frames

hv=zeros(nw,N); % output matrix, each row is one frame; for Hamming window
rv=zeros(nw,N); % output matrix; for rectangular window
sz=zeros(N+(nw-1)*nol,1);sz(1:ls)=s; % expand s with zeros(zero-padding)
for i=1:nw
    for j=1:N
        k=j+(i-1)*nol;
        hv(i,j)=sz(k)*hw(j);
        rv(i,j)=sz(k)*rw(j);
    end
end

% till now frames are obtained, now compute short-time ZC,AC,E

% short-time Energy

Eh=zeros(nw,1);
Er=zeros(nw,1);
for i=1:nw

```

```

        Eh(i)=sum(hv(i,:).^2);
        Er(i)=sum(rv(i,:).^2);
    end
    haxis=linspace(1,ls,nw);    % horizontal axis scaled to total number of
    samples
    figure(1);plot(haxis,Eh);title('short-time Energy: Hamming window')
    figure(2);plot(haxis,Er);title('short-time Energy: rectangular window')

    % short-time Zero-crossing

    ZCh=zeros(nw,1);
    ZCr=zeros(nw,1);
    for i=1:nw
        for j=1:N-1
            if sign(hv(i,j))~=sign(hv(i,j+1))
                ZCh(i)=ZCh(i)+1;
            end
            if sign(rv(i,j))~=sign(rv(i,j+1))
                ZCr(i)=ZCr(i)+1;
            end
        end
    end
    figure(3);plot(haxis,ZCh);title('short-time zero-crossing: Hamming window')
    figure(4);plot(haxis,ZCr);title('short-time zero-crossing: rectangular
    window')

    % short-time Autocorrelation

    ACh=zeros(nw,N+1);
    ACr=zeros(nw,N+1);
    for i=1:nw
        eh=[hv(i,:) zeros(1,N)];
        er=[rv(i,:) zeros(1,N)];
        for k=0:N
            for j=1:N
                ACh(i,k+1)=ACh(i,k+1)+eh(j)*eh(j+k);
                ACr(i,k+1)=ACr(i,k+1)+er(j)*er(j+k);
            end
        end
    end

    % save computed values for future use

    save('short_times.mat','hv','rv','Eh','Er','ZCh','ZCr','ACh','ACr');

```

```

% Part 3-1 - compute 'Mel cepstrum'
clear;clc;close all
load short_times % hv:frames by Hamming window & rv:frames by
rectangular window
clear('ACh','ACr','Eh','Er','ZCh','ZCr') % clear extra variables
[nw N]=size(hv);
fs=22050;
for L=[10,12,15]
    M=melfilter(N,L,fs); % obtain mel filter bank coefficient by calling
melfilter function
    Ml=size(M,2); % number of samples of each mel filter bank frequency
band
    Emelh=zeros(nw,L);cmelh=zeros(nw,L);
    Emelr=zeros(nw,L);cmelr=zeros(nw,L);
    for i=1:nw
        hw=hv(i,:); % i-th frame:Hamming window
        rw=rv(i,:); % i-th frame:rectangular window
        HW=abs(fft(hw)); % fft-abs:Hamming
        RW=abs(fft(rw)); % fft-abs:rectangular
        for l=1:L
            Al=sum(abs(M(l,:)).^2); % Mel window energy for normalizing
Mel coef
            Emelh(i,l)=sum((abs(M(l,:)).*HW(1:Ml)).^2)/Al; % mel scale
filter bank:Hamming
            Emelr(i,l)=sum((abs(M(l,:)).*RW(1:Ml)).^2)/Al; % mel scale
filter bank:rectangular
        end
        cmelh(i,:)=dct(log(Emelh(i,:))); % mel frequency filter bank
coefficients:Hamming
        cmelr(i,:)=dct(log(Emelr(i,:))); % mel frequency filter bank
coefficients:rectangular
    end
    save(strcat('Mel',num2str(L),'.mat'),'cmelh','cmelr') % save computed
values for future use
end

```

```

function M=melfilter(N,L,fs)
% compute mel filterbank

% N:number of window samples
% L:number of cepstrum coefficients
% fs:sampling frequency

%-----
nofChannels = L;    %number of mel channels
%-----

%compute resolution etc.
df = fs/N; %frequency increment on linear scale
Nmax = round(N/2); % maximum fft index
fmax = fs/2; % maximum frequency
melmax = freq2mel(fmax); % maximum mel frequency
melinc = melmax / (nofChannels + 1); % frequency increment on mel scale

%center frequencies on mel scale
melcenters = (1:nofChannels) .* melinc;

%center frequencies in linear scale [Hz]
fcenters = mel2freq(melcenters);

%quantize into FFT indices
indexcenter = round(fcenters ./df);

%compute start indices of windows
indexstart = [1 , indexcenter(1:nofChannels-1)];

%compute stop indices of windows
indexstop = [indexcenter(2:nofChannels),Nmax];

%compute triangle-shaped filter coefficients
M = zeros(nofChannels,Nmax);
for c = 1:nofChannels
    %left ramp
    increment = 1.0/(indexcenter(c) - indexstart(c));
    for i = indexstart(c):indexcenter(c)
        M(c,i) = (i - indexstart(c))*increment;%(sqrt(2)^( 1- c))
    end
    %right ramp
    decrement = 1.0/(indexstop(c) - indexcenter(c));
    for i = indexcenter(c):indexstop(c)
        M(c,i) = (1.0 - ((i - indexcenter(c))*decrement));%(sqrt(2)^( 1-
c))
    end
end

function m = freq2mel (f)
% compute mel value from frequency f
m = 2595 * log10(1 + f./700);

function b = mel2freq (m)
% compute frequency from mel value
b = 700*((10.^(m ./2595)) -1);

```

```

% Part 3-2 - compute 'LPC'
clear;clc;close all
load short_times % hv:frames by Hamming window & rv:frames by
rectangular window
clear('ACh','ACr','Eh','Er','ZCh','ZCr') % clear extra variables
[nw N]=size(hv);
for p=[10,12,15]
    LPCh=zeros(nw,p);errorLPCh=zeros(nw,1);
    LPCr=zeros(nw,p);errorLPCr=zeros(nw,1);
    for i=1:nw
        hw=hv(i,:); % i-th frame:Hamming window
        rw=rv(i,:); % i-th frame:rectangular window
        [ah,gh]=lpc(hw,p); % compute LPC coefficients:Hamming
        [ar,gr]=lpc(rw,p); % compute LPC coefficients:rectangular
        LPCh(i,:)=-ah(2:end);% first coefficient of a is 1 (for computation
of error) and other are negative
        errorLPCh(i)=gh;
        LPCr(i,:)=-ar(2:end);% so it is ignored; LPC coefficients saved in
appropriate form
        errorLPCr(i)=gr;
    end

save(strcat('LPC',num2str(p),'.mat'),'LPCh','LPCr','errorLPCh','errorLPCh')
% save computed values for future use
end

```

```
% Part 4-1 - determine 'VUS' parts of input signal with respect to part 2
% and part 3
% based on report chapter 4
clear;clc;close all
load short_times % Eh:Energy by Hamming window & ZCh:zero-crossing
clear('hv','rv','ACh','ACr','Er','ZCr') % clear extra variables

nw=length(Eh);
ZC_th_u=20;
E_th_v=4;
E_th_s=0.9;
ZC_th_s=10;
nVUS=zeros(nw,1);
for i=1:nw
    if ZCh(i)>ZC_th_u
        VUS(i)='U';
        nVUS(i)=2;
    else
        if Eh(i)>E_th_v
            VUS(i)='V';
            nVUS(i)=1;
        else
            if Eh(i)<E_th_s && ZCh(i)<ZC_th_s
                VUS(i)='S';
                nVUS(i)=3;
            else
                VUS(i)='V';
                nVUS(i)=1;
            end
        end
    end
end
end
VUS
plot(nVUS);xlabel('frame #');ylabel('V/U/S');ylim([0 4])
```

```

% Part 4-2 - determine 'VUS' parts of input signal using "A Bayesian
% Approach ..." in "Algorithms for Speech Processing" chapter

clear;clc;close all
load short_times % Eh:Energy by Hamming window & ZCh:zero-crossing
% & ACr:Autocorrelation for rectangular window
clear('hv','rv','ACh','Er','ZCr') % clear extra variables
load LPC12
clear('LPCr','errorLPCr') % clear extra variables
voiced=[8:14 17:20 40:49]; % training samples for voiced frames
unvoiced=[5:6 22:38 55:56]; % training samples for unvoiced frames
silence=1:3; % training samples for silence frames

% prepair features
Es=10*log10(1e-5+Eh);
C1=ACr(:,1);
Ep=10*log10(1e-6+errorLPCh);

% Training Data

% voiced
j=1;
lv=length(voiced);
v=zeros(lv,5);
for i=voiced
    x1=Es(i);
    x2=ZCh(i);
    x3=C1(i,1);
    x4=LPCh(i,1);
    x5=Ep(i);
    v(j,:)= [x1 x2 x3 x4 x5];
    j=j+1;
end
% unvoiced
j=1;
lu=length(unvoiced);
u=zeros(lu,5);
for i=unvoiced
    x1=Es(i);
    x2=ZCh(i);
    x3=C1(i,1);
    x4=LPCh(i,1);
    x5=Ep(i);
    u(j,:)= [x1 x2 x3 x4 x5];
    j=j+1;
end
% silence
j=1;
ls=length(silence);
s=zeros(ls,5);
for i=silence
    x1=Es(i);
    x2=ZCh(i);
    x3=C1(i,1);
    x4=LPCh(i,1);
    x5=Ep(i);
    s(j,:)= [x1 x2 x3 x4 x5];

```

```

        j=j+1;
end

% Prepair classification parameters

% voiced
mv=mean(v);
Wv=cov(v);
% unvoiced
mu=mean(u);
Wu=cov(u);
% silence
ms=mean(s);
Ws=cov(s);

% Produce feture vector for test samples

lt=length(Eh);
t=zeros(lt,5);
for i=1:lt
    x1=Es(i);
    x2=ZCh(i);
    x3=C1(i,1);
    x4=LPCh(i,1);
    x5=Ep(i);
    t(i,:)=[x1 x2 x3 x4 x5];
end

% Test
nVUS=zeros(lt,1);
for i=1:lt
    x=t(i,:);
    dv=(x-mv)*inv(Wv)*(x-mv)';
    du=(x-mu)*inv(Wu)*(x-mu)';
    ds=(x-ms)*inv(Ws)*(x-ms)';

    if dv<=du && dv<=ds
        VUS(i)='V';
        nVUS(i)=1;
    elseif du<=dv && du<=ds
        VUS(i)='U';
        nVUS(i)=2;
    else
        VUS(i)='S';
        nVUS(i)=3;
    end
end
VUS
plot(nVUS);xlabel('frame #');ylabel('V/U/S');ylim([0 4])

```

```

% Part 5 - plot 'FFT' and 'LogFFT' of speech frame then obtain these from
% 'Cepstrum' and 'LPC' and compare results
% use 10,12,15 coefficients for cepstrum, calculate error function
clear;clc;close all
load short_times % hv:frames by Hamming window & rv:frames by
rectangular window
clear('ACh','ACr','Eh','Er','ZCh','ZCr') % clear extra variables

[nw N]=size(hv);
LogFFT=zeros(1,N);
for i=1:nw
    F=fft(hv(i,:));
    LogFFT(i,:)=log(abs(F));
end

% compute LogFFT from Mel Cepstrum coefficients
for L=[10,12,15]
    load (strcat('Mel',num2str(L)))

end

% compute LogFFT from LPC coefficients
for p=[10,12,15]
    load (strcat('LPC',num2str(p)));clear('LPCr')
    for i=1:nw

        h=LPCr(i,:); % for Hamming window
        H=(abs(fft(h,N)));
        LF=LogFFT(i,:); % LogFFT for frame i
        Err=sum(abs(LF-H))/N; % Mean Error for frame

        % plots
        plot(abs(LF),'b');
        hold on
        plot(H,'r')
        legend('LogFFT','LPC');title(strcat('frame ',num2str(i),' ',
p=',num2str(p),' , Error=',num2str(Err)))
        hold off
        pause(.4)
        % end of plots

    end
end
end

```

بخش ۶

```
% Part 6 - plot 'spectrogram', determine 'Formants', 'Pitch' frequency and
% noise
clear;clc;close all

[s,fs]=wavread('speech_dft',[6101,18500]);
colormap 'default'
N=441;
nol=221;
spectrogram(s,N,nol,N,fs);
title('speech spectrogram;time and frequency are scaled to real values')
```

بخش ۷

```
% Part 7 - find 'Formants' from 'FFT' peaks
clear;clc;close all

fs=22050;
load short_times % hv:frames by Hamming window
clear('rv','ACh','ACr','Eh','Er','ZCh','ZCr') % clear extra variables

[nw N]=size(hv);
Formants=zeros(nw,12);
for i=1:nw
    x=fft(hv(i,:));
    F=smooth(log(abs(x(1:round(N/2))))),31);
    [peaks,locs]=findpeaks(F,'minpeakdistance',10);
    [peaks indx]=sort(peaks,'descend');
    locs=locs(indx(1:min(12,length(indx))));
    locs=sort(locs);
    plot(F,'b');title(strcat('frame ',num2str(i)));
    hold on
    stem(locs,F(locs),'r')
    hold off
    pause(.3)
    Formants(i,1:length(locs))=locs*fs/length(F);
end
```

```

% Part 8 - a program for finding 'Pitch' of 'Voiced' parts of speech
% frame then plot pitch 'Contour'
clear;clc;close all
fs=22050;
load short_times % ACr:Autocorrelation for rectangular window
nw=size(hv,1);
clear('hv','rv','ACh','Er','Eh','ZCh','ZCr') % clear extra variables
voiced=[7:21 39:50]; % voiced frames
MinPeakDistance=10;
Pitch=zeros(nw,1);
for i=voiced
    x=ACr(i,:);
    [~,locsp]=findpeaks(x,'MINPEAKDISTANCE',2*MinPeakDistance);
    [~,locsn]=findpeaks(-x,'MINPEAKDISTANCE',2*MinPeakDistance);
    lp=length(locsp);ln=length(locsn);
    locs=locsn;
    k=ln+1;
    for j=1:lp
        if abs(locs-locsp(j))>MinPeakDistance
            locs(k)=locsp(j);
            k=k+1;
        end
    end
    locs=sort(locs);
    locs=[1 locs];
    p=[];k=1;
    for j=1:length(locs)-1
        if (x(locs(j))*x(locs(j+1)))<0
            p(k)=locs(j+1)-locs(j);
            k=k+1;
        end
    end
    Pitch(i)=2*median(p)/fs;
    figure
    plot(x,'b');title(strcat('frame ',num2str(i)));
    hold on
    stem(locs,x(locs),'r')
    pause(.3)
end
figure;plot(Pitch);title('Pitch contour(only for voiced frames)')
xlabel('frame #');ylabel('Pitch period(sec)')

```