VOICED/UNVOICED CLASSIFICATION LPC AND GAUSSIAN MIXTURE MODEL

זיהוי מקטעים קוליים ואל קוליים בקול דיבור

שם קורס:עקרונות זיהוי זיבור

שם המגיש: יזיד בישאראת

שם המרצה: דר יצחק לפידות

Table of Contents

מטרת הפרויקטון

המטרה של הפרויקט להחליט האם מקטע מסוים בקול אמור להיות מסווג כי קולי או אל קולי(Voiced-Unvoiced Classifications), יש כמה דרכים בספרות שעושים החלטה טובה, בפרויקטון הזה אנחנו הולכים לחקור משערך אקוסטי LPC וגם השארית של ה LPC עם הורדת המימד שלה, עם GMM לסיווג.

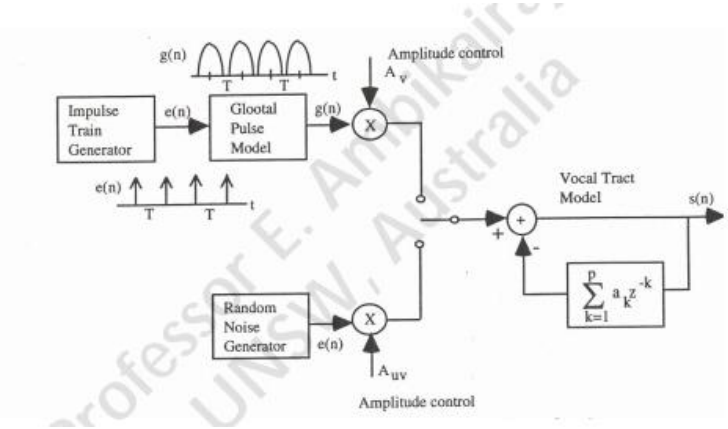
סיווג הקול למקטעים קוליים ואל קוליים מאוד חשוב, באנליזה של אות דיבור וגם שיפור איכות אות הדיבור.

הסבר כללי

שני חלקי הקול:

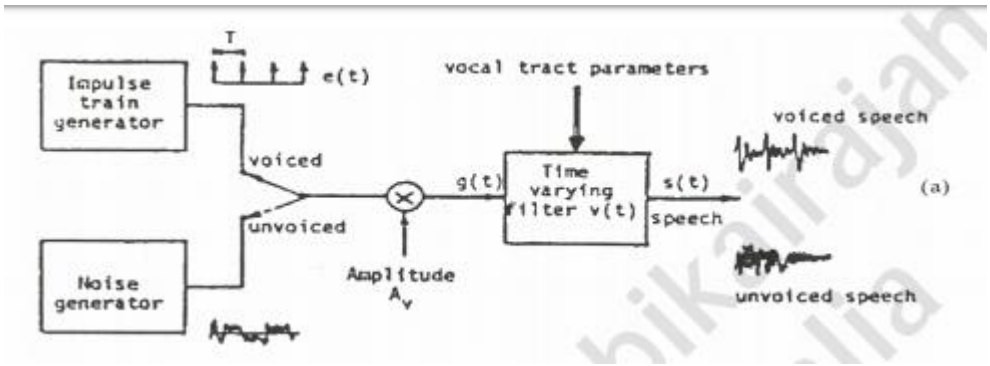
החלק הקולי נוצר כשמיתרי הקול (vocal cords) אקטיביים כלומר הם המשתתפים ביצירת הקול וזה קורה כשה-glottal pulse model עובד, במצב הזה מיתרי הקול מנתהגים כי רוטט.

החלק האל קולי נוצר כשה-glottal pulse mode לא עובד אז מה שמניע מיתרי הקול זה רעש עם אמפליטודה קטנה (קטנה יחסית לאות קולי), שקשה לזהות אותה.



רואים בתמונה למעלה את המערכת של האות הקולי והאל קולי

השפעת ה GLOTTAL PULSE:



רואים בציור הזה את ההבדל בין הקטע הקולי והאלקולי כפי שהסברנו למעלה.

בציור למטה נוכל לראות את ההבדל בין הקטעים הקוליים והקטעים האלקליים ה G(w) זה ה GLOTTAL PULSE אם נתייחס ל-GLOTTAL במקרה הקולי כי רכבת הלמים אז נוכל להגיע לאותו מבנה בתדר.

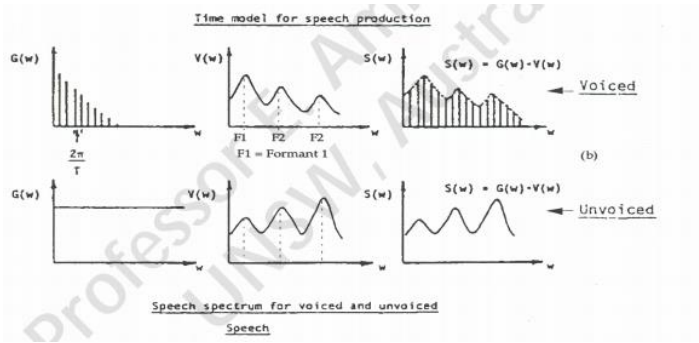
גם רואים מהגרף למטה שבקטעים הקוליים המוצא של מיתרי הקול יורדים בתדר זה נובע משתי סיבות מבנה מיתרי הקול וגם מה GLOTTAL PULSE (רכבת פולסים בזמן הופכת לרכבת הלמים יורדת בתדר) זה יעזור לנו מאוד לזהות אירועים קוליים ואלקוליים.

$$E(z) = Z\{e(n)\}$$

$$\sum_{n=-\infty}^{n=+\infty} e(n)z^{-n} = \sum_{n=0}^{n=+\infty} e(n)z^{-n}$$

$$E(z) = 1 + z^{-P} + z^{-2P} + \ldots$$

$$E(z) = \frac{1}{1 - z^{-P}}$$



תהליך עירור של ה GLOOTAL הוא:

$$G(z) = \frac{1}{\left(1 - e^{-cT} z^{-1}\right)^2} \quad \text{where c : speed of sound}$$

נעשה הזנחה

$$\text{But cT} << 1, \therefore e^{-cT} \approx 1$$

$$\therefore G(z) \cong \frac{1}{\left(1 - z^{-1}\right)^2} \quad \text{for voiced speech, G(z)= 1 for unvoiced speech}$$

אז תהליך העירור במקרה הקולי מאוד בולט ובמקרה האל קולי זה אחד

מודל מיתרי הקול:

אפשר לתאר מיתרי הקול במצב הקולי על ידי קטבים כלומר אפשר בפילטר IIR:

$$V(z)=\frac{U_l(z)}{U_g(z)}=\frac{1}{\prod_{k=1}^{K}1+b_k z^{-1}+c_k z^{-2}}=\frac{1}{1+\sum_{k=1}^{p}a_k z^{-k}}$$

במצב האל קולי כדי לתאר מיתרי הקול נצטרך פילטר עם אפסים וקטבים, אבל אפשר לקרב כל אפס לשני קטבים:

$$V(z)=\frac{1+\sum_{k=1}^{L}b_k z^{-k}}{1+\sum_{k=1}^{P}a_k z^{-k}}\approx\frac{1}{1+\sum_{k=1}^{P+2L}a_k z^{-k}}$$

***LIP RADIATION MODEL*:**

אחרי מיתרי הקל יש גם את הלשון והפה שמשפיעים על הקול כוראים לזה LIP RADIATION MODEL

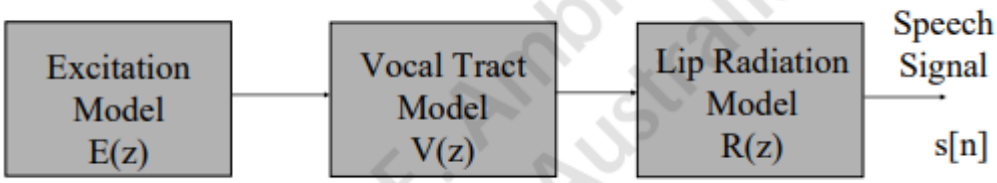
בדרך כלל המודל הזה מתנהג כי HPF:

בקירוב

$$R(z)=1-0.98z^{-1}$$

תהליך יצור הקול :

אז תהליך יצור הקול מחולק לשלוש חלקים:



ופונקצית התמסורת הכללית לקול:

$$\frac{S(z)}{E(z)} = AG(z)V(z)R(z)$$

פונקצית התמסורת לקטע קולי:

$$\frac{S(z)}{E(z)} = A_v G(z)V(z)R(z)$$

$$\frac{S(z)}{E(z)} = A_v \frac{1}{\left(1-z^{-1}\right)^2} \frac{1}{1+\sum_{k=1}^{P} a_k z^{-k}} 1-z^{-1}$$

$$\frac{S(z)}{E(z)} = A_v \frac{1}{\left(1-z^{-1}\right)} \frac{1}{1+\sum_{k=1}^{P} a_k z^{-k}} = \frac{A_v}{1+\sum_{k=1}^{P+1} a'_k z^{-k}}$$

פונקצית התמסורת לקטע אל קולי:

$$\frac{S(z)}{E(z)} = A_{uv} G(z)V(z)R(z)$$

$$\frac{S(z)}{E(z)} = A_{uv} 1 \frac{1}{1+\sum_{k=1}^{P+2L} a_k z^{-k}} (1-z^{-1})$$

$$\frac{S(z)}{E(z)} = A_{uv} \frac{1-z^{-1}}{1+\sum_{k=1}^{P+2L} a_k z^{-k}} = \frac{A_{uv}}{1+\sum_{k=1}^{P+2L+2} a'_k z^{-k}}$$

מציאת מאפיינים

Linear prdictive coding

כללי

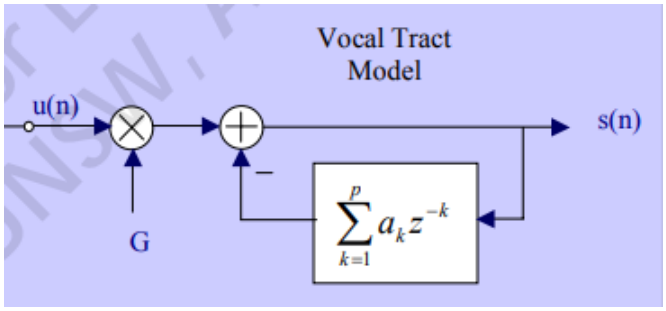
אחד המאפיינים הכי שימושיים בעבר ועד עכשיו יש שימושים בהם, משתמשים בשיטה הזאת להעברת אות דיבור בכמות ביתים נמוכה(דחיסה מאבדת מידע) וגם בזיהוי דובר וזיהוי פונמות.

- הרעיון המרכזי משיטת LPC זה שדגימה של אות דיבור אפשר לשערך על ידי קומבינציה לינארית של הדגימות שעברו.
- על ידי הקטנת השגיאה ביו הדגימה האמיתית והאות המשוערך(בדרך כלל משתמשים בMMSE), מגיעים למקדמים ייחודיים של פילטרי FIR.

מציאת מקדמים מתאימים:

כמו שדיברנו בהתחלה אפשר לקרב מיתרי הקול על ידי IIR פילטר

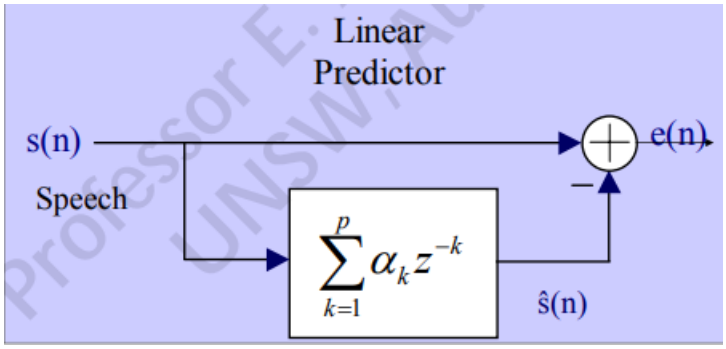
$$\frac{S(z)}{U(z)} = \frac{G}{1+\sum_{k=1}^{p} a_k z^{-k}}$$



וגם יודעים שהדגימות קשורות לעירור על ידי משוואת הפרשים פשוטה:

$$s[n] = \sum_{k=1}^{p} a_k s[n-k] + Gu[n]$$

אז על ידי סינון הפוך (כלומר העברת האות בפילטר ALLZERO FIR פילטר) נוכל למצוא את השגיאה של מקדמי הפילטר שלנו



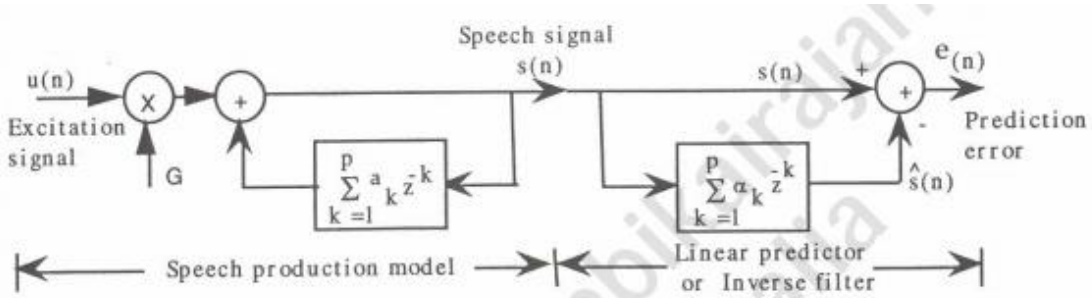
אם נעבירים אות הדיבור דרך משערך לינארי עם מקדמים ak אז המשערך מוציא את

$$\hat{s}(n) = \sum_{k=1}^{p} \alpha_k s(n-k)$$

והשגיאה בין האות המשוערך לאות האמיתי תהיה:

$$e(n) = s(n) - \hat{s}(n) = s(n) - \sum_{k=1}^{p} \alpha_k s(n-k)$$

אז המערכת הכללית תהיה מהצורה:



The prediction error signal e(n) is shown above.

שפילטר ה IIR זה מערכת הקול, ופילטר ה FIR מסנן הפוך למסנן הIIR

$$\frac{S(z)}{U(z)} = \frac{G}{1 - \sum_{k=1}^{P} a_k z^{-k}}$$ <- Vocal tract(IIR filter)

$$\frac{E(z)}{S(z)} = 1 - \sum_{k=1}^{P} \alpha_k z^{-k}$$ <- Linear Predictor(FIR filter)

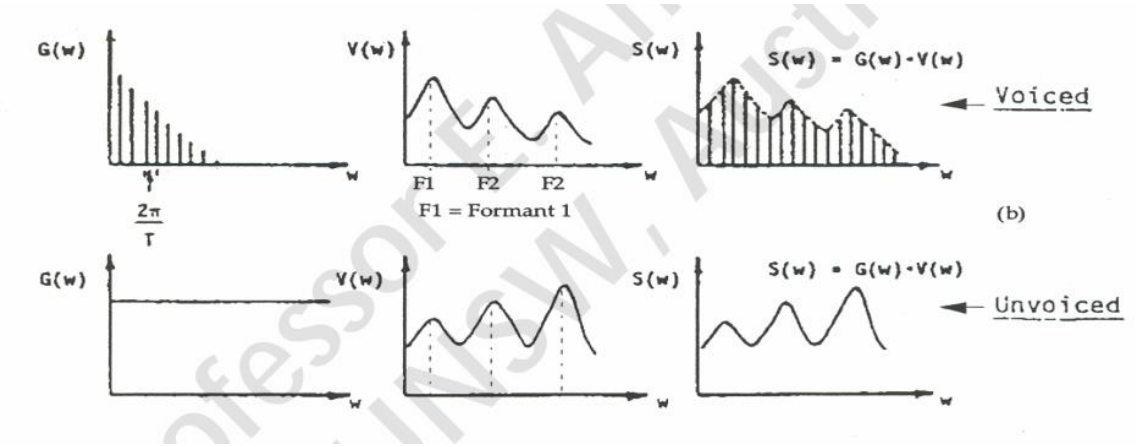
שאירית של LPC:

על ידי קירובים שלמדנו בכיתה) כמו (MMSE מקרבים את האות כמה שאפשר עד שהמקדמים של מיתרי הקול הופכים לקמט אותם מקדמים של המשערך שבנינו, מה שמשאיר בשגיאה רק את פונקצית עירור

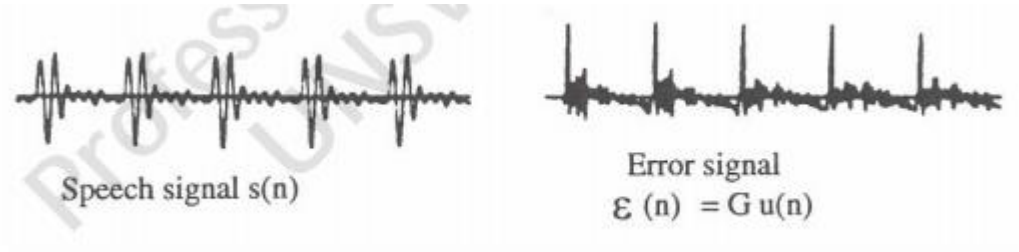
$$\frac{E(z)}{U(z)} = G \frac{1 - \sum_{k=1}^{P} \alpha_k z^{-k}}{1 - \sum_{k=1}^{P} a_k z^{-k}} \qquad \text{If } \alpha_k = a_k \Rightarrow \frac{E(z)}{U(z)} = G \Rightarrow e(n) = Gu(n)$$

כמו שהסברנו לפני במקטעים קוליים פונקצית העירור תהיה מאוד דומה לרכבת הלמים במקרה הקולי, ובמקרה האל קולי תהיה פשוט רעש,וקל לראות את זה בתדר, במערכת קולית רואים שהאמפליטודה יורדת כשעולים בתדר במערכת אל קולית רואים שהאמפליטודה לא משתנה אפילו עולה לפונמות מסוימים.

כלומר לפי הציור למטה מה שאמור לצת לנו מהשאירית זה הציור משמאל למעלה במקרה הקולי ומשמאל למטה במקרה האל קולי, זה המצב האידיאלי אבל אנחנו לא במצב האידיאלי לכן יהיה גם שארית במיתרי הקול שהשפיעו על האות ויקרבו אותו לצורה הימנית.



במצב הקולי בזמן:



בcepstrum אפשר

$$e(n) = G(n) * u(n)$$

$$E(w) = F[G(n) * u(n)] = G(w)U(w)$$

$$\log(E(w)) = \log(G(w)) + \log(U(w))$$

$$C(n) = F[\log(G(w))] + F[\log(U(w))] = \hat{G}(n) + \hat{U}(n)$$

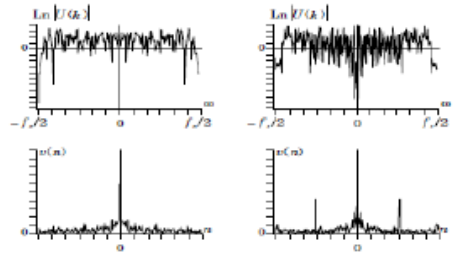


Figure 5: *Amplitude spectrum (top) and real cepstrum (bottom) of a residue. The left part is unvoiced, the right part is voiced.*

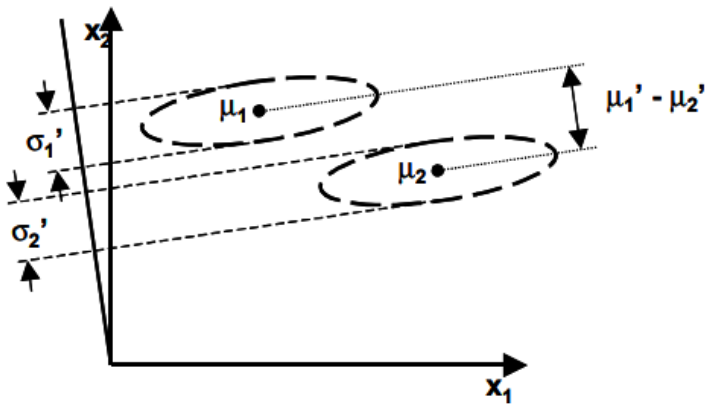
אם מוצאים את הכפסתרום של השארית רואים השפעת רכבת ההלמים לבד מה GLOTTAL

תהליך אימון

Linear discriminant analyses

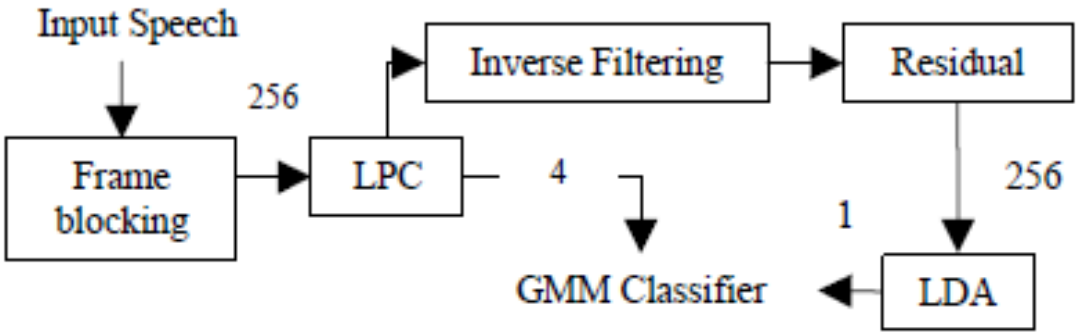
השתמשנו בשיטה הזו כדי להוריד את מימדי השארית ממימד גדול(גודל המימד כגודל הדגימות בכל פריים) לחד מימד, על ידי מציאת ההיטל שעושה דיסקרימינציה הכי גבוהה כלומר במצב שלנו יש שני מצבים אז הוא מנסה למצוא את ההיטל שמוציא לשני המצבים התוחלות הכי רחוקות והשונות הכי קטנה

$$\text{maximize} \quad \frac{(\mu_1' - \mu_2')^2}{\sigma_1'^2 - \sigma_2'^2}$$



מאפיינים לאימון

בוחרים ארבע מקדמי LPC והשארית של ה LPCאחרי הורדת המימדים שלה, הפריימים שלנו נלקחו באורך של 25 מילי שניות ובקפיצות של 110 מילי שניות ו תדר הדגימה של ההקלטות היה 16 קילו הרטס, כלומר יש 400 דגימה בכל פריים וקופצים כל פעם 160 דגימות.



GMM

באימון השתמשנו בשני GMMS אחד למקטעים קוליים ואחד לאל קוליים, לכל אחד מה GMMS השתמשנו ב M=50 מודלים בפנים(MEXTUER MODELS), ועשינו אתרציות עד 200 פעמים,

$$p(\vec{x} \mid \lambda) = \sum_{i=1}^{M} p_i b_i(\vec{x})$$

כדי לדעת לאיזה GMMS שייך הקטע שאנחנו רוצים לבדוק עשינו MLעל כל הגאוסיאנים בכל GMM וסכמנו אותם,

$$\hat{S} = \arg\max_{1 \le k \le S} \Pr(\lambda_k \mid X) = \arg\max_{1 \le k \le S} \frac{p(X \mid \lambda_k)\Pr(\lambda_k)}{p(X)}$$

בנוסחה למעלה הX הוא מאפייני הקטע שרוצים לדעת למי שייך

K: זה מספר הGMMS אצלנו דה שני מספרים

λ_k זה פרמטרי הגאוסיאנים

אבל עדיין הML תלוי בקמות הועת כל אחד מהם לכן נניח הסתברות שליפת קטע קולי שווה להסתברות שליפת קטע אל קולי לכן ה PRIOR לא ישפיע והמוואה שלנו הופכת ל

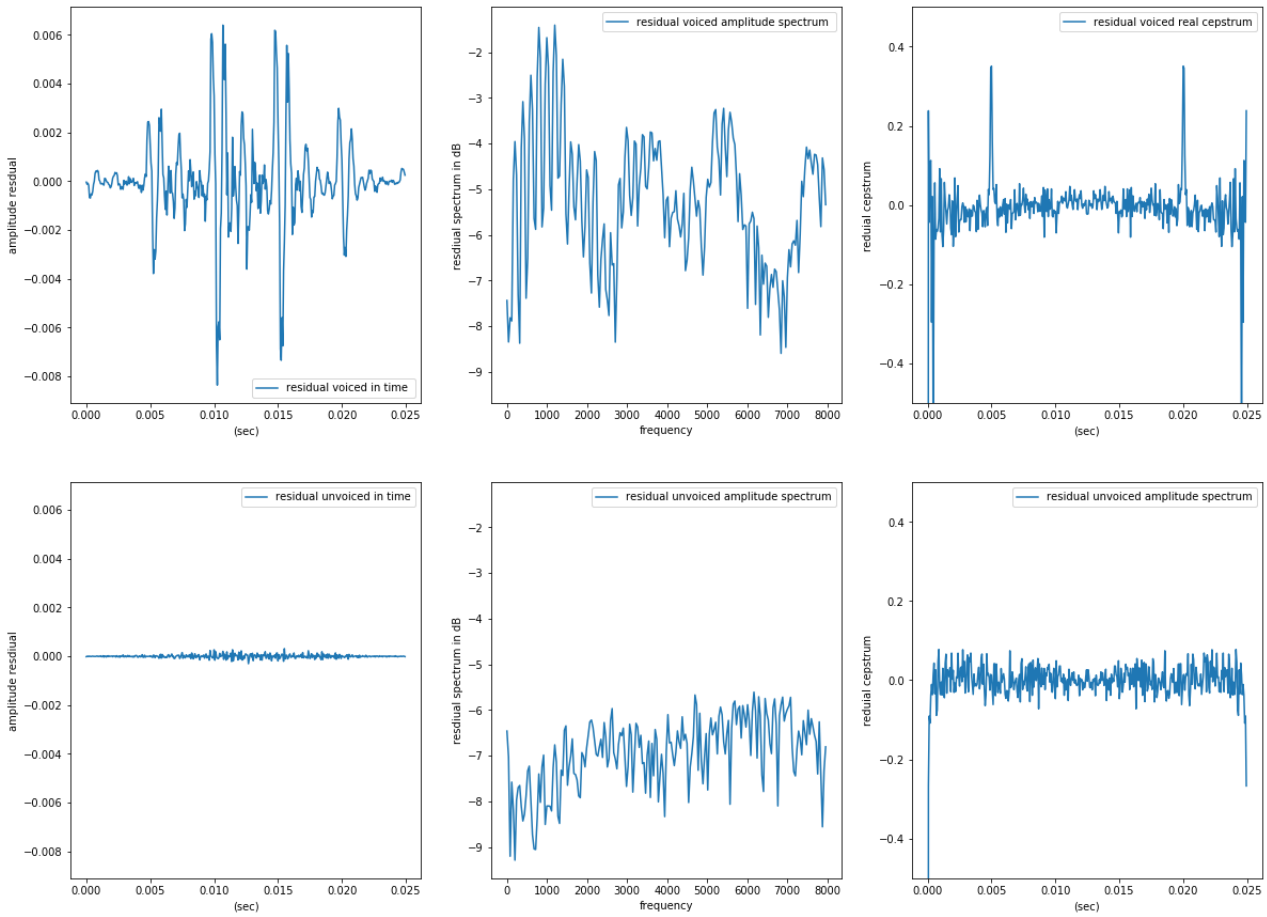
$$\hat{S} = \arg\max_{1 \le k \le S} p(X \mid \lambda_k).$$

תוצאות

דוגמא לתוצאות

לגבי השארית בדקנו אותה בשלושה מצבים המצב הראשון זה מציר הזמן השני בציר התדר השלישי בכבסתרום(cepstrum).

השורה הראשונה זה קולי השורה השנייה זה אל קולי העמודה הימנית זה אמפליטודה בזמן, עמודה אמצעית זה אמפליטודה בתדר, העמודה השמאלית זה הכבסתרום



הערכת התוצאות

הניסוי נעשה תשעה פעמים, פעם השאירית במישור הזמן פעם במישור התדר ופעם בכפסתרום אחרי כל שלושה פעמים שיניו את הרעש כדי לראות את השפעת הרעש (שיניו את הרעש שלושה פעמים.

אלה השיטות שהתמשתי בהם להערכה:

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

$$\textbf{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

Support : זה כמות הדגימות שניסינו עליהם

תוצאות:

SNR=30dB

```
---residual amplitude in time --
accuarcy =0.9497098646034816
              precision    recall  f1-score   support

           u       1.00      0.89      0.94       475
           v       0.91      1.00      0.96       559
---residual in spectrum --
accuarcy =0.9729206963249516
              precision    recall  f1-score   support

           u       1.00      0.94      0.97       475
           v       0.95      1.00      0.98       559
 ---residual cepstrum --
accuarcy =0.9700193423597679
              precision    recall  f1-score   support

           u       1.00      0.93      0.97       475
           v       0.95      1.00      0.97       559
```

SNR=10dB

```
---residual amplitude in time --
accuarcy =0.746615087040619
              precision    recall  f1-score   support

           u       0.99      0.45      0.62       475
           v       0.68      0.99      0.81       559


---residual spectrum --
accuarcy =0.9777562862669246
              precision    recall  f1-score   support

           u       1.00      0.95      0.98       475
           v       0.96      1.00      0.98       559
---residual cepstrum --
accuarcy =0.9468085106382979
              precision    recall  f1-score   support

           u       0.99      0.89      0.94       475
           v       0.91      0.99      0.95       559
```

SNR=4dB

```
---residual in time --
accuarcy =0.706963249516441
              precision    recall  f1-score   support

           u       0.87      0.43      0.57       475
           v       0.66      0.94      0.78       559
---residual in spectrum --
accuarcy =0.9671179883945842
              precision    recall  f1-score   support

           u       0.99      0.94      0.96       475
           v       0.95      0.99      0.97       559
---residual in cepstrum --
accuarcy =0.8878143133462283
              precision    recall  f1-score   support

           u       0.93      0.81      0.87       475
           v       0.86      0.95      0.90       559
```

מסכנות:

חילקנו את הניסוי לשלושה שלבים אחד עם יחס אות לרעש DB30 ואחד עם DB10 ואחד עם DB4 , אפשר לראות שיחס אות לרעש באמת משפיע על התוצאות, כשהיחס קטן התוצאות מתקלקלות.

בכל שלב בודקים את השארית של ה LPC בשלושה דרכים,דרך במישור הזמן דרך שנייה ספקטרום התדר, דרך שלישית קפסטרום.

אפשר לראות השיטה הכי טובה לשחזור התוצאות היא השארית בתדר שנותנת הדיוק הכי טוב גם עם הקטנת ה SNR , זה מהסיבה שהוסברה למעלה שקל לראות ההבדל בין קטע קולי ואל-קולי בתדר, בגלל קטע קולי יורד עם עליה בתדר וקטע אל-קולי לא.

השתמשתי ב 50 גאוסינים לכל GMM אבל זה היה מיותר, יותר מ 30 גאוסיאנים ההבדל לא כל כך מורגש, אבל השארתי את ה 50 גאוסיאנים בגלל שהשתמשו בכמות הזו במאמר.

לגבי מסנן pre-emphasis השפיע מאוד על התוצאות, הוא קילקל את התוצאת, וזה מהסיבה שהוא מתנהג כמו HPF , מה הוא עושה זה מקטין את התדר הנמוך של המקטעים הקוליים והופך אותם דומים יותר למקטעים אל קוליים או לשקט.

אחרי הסתכלות על מאפייני המקטעים הקוליים והאל קוליים, היינו יכולים להוסיף עוד מקטע של שקט, ונוכל לראות במקטעים הקוליים אמפליטודה גבוהה בתדר נמוך, במקטעים אל קוליים אמפליטודה גבוהה בתדר גבוהה, ובקטעי שקת האמפליטודה אחידה בתדר עם סטיית הרעש, אבל זו היתה הסתכלות על מקטעים בלי רעש בכלל, עם הוספת קצת רעש אי אפשר להבחין בין מקטעי השקת והאל קוליים.

References

[1] "Robust voiced/unvoiced classification using novel features and gaussian mixture model".

[2] Fundamental of Speech Recognition, 1993.

[3] D. A. Reynolds, "Identification, Robust Text-Independent Speaker," 1995.

[4] "Usefulness of the LPC-Residue in Text-Independent Speaker Veri".

In [250]:
```
from scipy.io import wavfile
import pysptk as sptk
from numpy import float64
import numpy as np
from scipy import signal
from sklearn import discriminant_analysis
import glob
import librosa
from scipy.signal import butter, lfilter

voiced = ['aa', 'ae', 'ah', 'ao', 'aw', 'ax', 'ax-h', 'axr', 'ay','ah','ao','oy','ow','uh','uw','ux','er','ax','ix','axr
unvoiced=['h#','epi','pau','bcl', 'dcl','gcl', 'pcl','tcl', 'kcl']

###################band pass filter##################
def butter_bandpass(lowcut, highcut, fs, order=5):
    nyq = 0.5 * fs
    low = lowcut / nyq
    high = highcut / nyq
    b, a = butter(order, low, btype='low')
    return b, a
def butter_bandpass_filter(data, lowcut, highcut, fs, order=8):
    b, a = butter_bandpass(lowcut, highcut, fs, order=order)
    y = lfilter(b, a, data)
    return y
###################pre-peocessing###############
def preprocessing(wavdata,rate,coe=0.95):
    wavdata = wavdata - np.mean(wavdata)
    #wavdata = preemphasis(wavdata, coe)
    #wavdata=butter_bandpass_filter(wavdata,75, 7500, rate, order=5)
    return wavdata
def add_noise_snr(sig,snr):
    x_watts = sig ** 2
    # Set a target SNR
    target_snr_db = 30
    # Calculate signal power and convert to dB
    sig_avg_watts = np.mean(x_watts)
    sig_avg_db = 10 * np.log10(sig_avg_watts)
    # Calculate noise according to [2] then convert to watts
    noise_avg_db = sig_avg_db - target_snr_db
    noise_avg_watts = 10 ** (noise_avg_db / 10)
    # Generate an sample of white noise
```

```
    mean_noise = 0
    noise_volts = np.random.normal(mean_noise, np.sqrt(noise_avg_watts), len(x_watts))
    # Noise up the original signal
    y_volts = sig + noise_volts
    return y_volts
############### pre emphasis#######################
def preemphasis(wav, coeff=0.95):
    preem_wav = signal.lfilter([1, -coeff], [1], wav)
    return preem_wav
############## lda dimensionality reduction##################
def lda(err,label, path):
    lda =  discriminant_analysis.LinearDiscriminantAnalysis(n_components=1)
    err_lda=lda.fit_transform(err, label)
    name="lda"
    savedata(lda,name, path)
    return err_lda
################ save data###########
def savedata(data,name, path):
    import pickle
    with open(path+name, 'wb') as f:
        pickle.dump(data, f, pickle.HIGHEST_PROTOCOL)
    return path+name
################open data###############
def opendata(name, path):
    import pickle
    with open(path+name, 'rb') as f:
        data = pickle.load(f)
    return data

################### find residual####################
def residual(sig,c):
    sig_hat = signal.lfilter([0] + -1*c[1:], [1], sig)
    ris= sig-sig_hat
    return ris

################## find ste#########################
def shortTermEnergy(sample):
    return float64(sum( [ abs(x)**2 for x in sample ] ) / len(sample))

################feature extraction###############
def real_spectrum(x, n=None):
    spec = abs(np.fft.rfft(x))
```

```
    return spec
def real_cepstrum(x, n=None):
    spectrum = np.fft.fft(x, n=n)
    ceps = np.fft.ifft(np.log(np.abs(spectrum))).real
    return ceps
################### add noise ##############
def noise(sig,SNR=15):
    noise = np.random.normal(0, 1, sig.shape)
    signal = sig + noise
    return signal

def feature(sig,lpc_o):

    WINDOW = np.hamming(len(sig))
    fr = sig*WINDOW
    data = librosa.lpc(fr, lpc_o)
    res=residual(fr,data)
    #res= real_cepstrum(res)
    res=real_spectrum(res)
    return data,res
```

In [251]:
```
import librosa as load

###############get data and save it###########
def getData2(filename, lpc_o):

    v = [np.asarray(()),np.asarray(()),np.asarray(())]
    u = [np.asarray(()),np.asarray(()),np.asarray(())]
    wavdata, rate = librosa.load(filename + ".wav",sr=16000)
    #rate, wavdata = wavfile.read(filename + ".wav")
    wavdata=add_noise_snr(wavdata,20)
    wavdata = preprocessing(wavdata,rate)
    labdata = open(filename[:-2] + ".PHN")
    step=int(rate*0.01)
    l=int(rate*0.025)
    for ln in labdata.readlines():
        sta, end, cls = ln.split()
        sta=int(sta)
        end=int(end)
        nFrames=int(((end-sta-l)/step))
        startIdx=sta
        for k in range(nFrames):# cutting signal to frames
            startIdx=sta+k*step
            stopIdx=sta+k*step+l
            data,ris=feature(wavdata[startIdx:stopIdx],lpc_o)
            if cls in unvoiced:
                if u[0].size == 0:
                    u[0] = data
                    u[1]= ris
                else:
                    u[0]=np.vstack((u[0],data))
                    u[1]=np.vstack((u[1],ris))
            elif cls in  voiced:
                if v[0].size == 0:
                    v[0] = data
                    v[1]= ris
                else:
                    v[0]=np.vstack((v[0],data))
                    v[1]=np.vstack((v[1],ris))
    return (v,u)
def getData(filename, lpc_o,path):
    c = np.asarray(())
```

```
    d = np.asarray(())
    wavdata, rate = librosa.load(filename + ".wav",sr=16000)
    #rate, wavdata = wavfile.read(filename + ".wav")
    wavdata=add_noise_snr(wavdata,20)
    wavdata = preprocessing(wavdata,rate)
    labdata = open(filename[:-2] + ".PHN")
    lda=opendata('lda',path)#### lda
    step=int(rate*0.01)
    l=int(rate*0.025)
    for ln in labdata.readlines():
        sta, end, cls = ln.split()
        sta=int(sta)
        end=int(end)
        nFrames=int(((end-sta-l)/step))
        startIdx=sta
        for k in range(nFrames):# cutting signal to frames
            startIdx=sta+k*step
            stopIdx=sta+k*step+l
            data,ris = feature(wavdata[startIdx:stopIdx],lpc_o)
            a=np.reshape(ris, (1,-1))
            ris_lda = lda.transform(a)
            data=np.hstack((data,ris_lda[0]))
            if cls in unvoiced:
                if d.size == 0:
                    c='u'
                    cl=cls
                    d=data
                else:
                    cl=np.vstack((cl,cls))
                    c=np.vstack((c,'u'))
                    d=np.vstack((d,data))
            elif cls in voiced:
                if d.size == 0:
                    cl=cls
                    c='v'
                    d=data
                else:
                    cl=np.vstack((cl,cls))
                    c=np.vstack((c,'v'))
                    d=np.vstack((d,data))
    return (d, c,cl)
```

In [252]:
```
def getfiles(name, directory,lpc_o,path):
    voiced = [np.asarray(()),np.asarray(())]
    unvoiced = [np.asarray(()),np.asarray(())]
    ris=np.asarray(())
    files =glob.glob(directory+'/*wv.wav')
    for f in files:
        filename =  f[:-4]

        v,u = getData2(filename, lpc_o)
        if (len(v[0]) !=0) and (len(u[0]) !=0):
            if voiced[0].size == 0:
                voiced[0] = v[0]
                voiced[1] = v[1]
            else:
                voiced[0]=np.vstack((voiced[0],v[0]))
                voiced[1]=np.vstack((voiced[1],v[1]))
            if unvoiced[0].size == 0:
                unvoiced[0] = u[0]
                unvoiced[1] = u[1]
            else:
                unvoiced[0]=np.vstack((unvoiced[0],u[0]))
                unvoiced[1]=np.vstack((unvoiced[1],u[1]))
    ##########connecting risedual ########
    print(voiced[1].shape)
    print(unvoiced[1].shape)
    ris=np.vstack((voiced[1],unvoiced[1]))
    #####making labels for the data########
    lin_v=(len(voiced[1]))
    lin_u=(len(unvoiced[1]))
    label_v=np.zeros(lin_v)
    label_u=np.ones(lin_u)
    label=np.hstack((label_v,label_u))
    ris_lda=lda(ris,label, path)
    v_f=np.hstack((voiced[0],ris_lda[0:len(voiced[1])]))
    u_f=np.hstack((unvoiced[0],ris_lda[(len(voiced[1])):len(unvoiced[1])+len(voiced[1])]))
    data = {'v': v_f,'u': u_f}
    savedata(data,"dat", path)
    return 1
```

In [253]:
```
from sklearn import mixture
def teach_gmm(path):
    data=opendata('dat',path)

    for t, d in data.items():
        gmm = mixture.GaussianMixture(n_components=50,max_iter=200, covariance_type='full', tol=0.001)
        gmm.fit(d)
        # dumping the trained gaussian model
        name = t+".gmm"
        savedata(gmm,name, path)
    return 1
```

In [254]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
def predict(directory, lpc_o,path):
    tr=['v','u']
    pa=[]
    ca=[]
    miss_sum=0
    files =glob.glob(directory+'\*wv.wav')
    ##open gmms
    gmm_v=opendata('v.gmm',path)
    gmm_u=opendata('u.gmm',path)
    for f in files:
        filename = f[:-4]
        d, c, cl= getData(filename, lpc_o,path)
        p=[]
        for i in range(len(c)):
            a=np.reshape(d[i],(1,-1))
            #a=d[i:i+5]
            log_likelihood=[0,0]
            log_likelihood=[gmm_v.score(a),gmm_u.score(a)]
            winner = np.argmax(log_likelihood)
            p.append(tr[winner])

        ca.extend(c)
        pa.extend(p)

    results = confusion_matrix(ca, pa)
    acc=accuracy_score(ca, pa)
    report= classification_report(ca, pa)
    print("---residual in spectrum --")
    print ("accuarcy ="+str(acc))
    print (report)
```

In [255]:
```
train_set="TIMIT\\TRAIN\\DR5\\FBJL0"
path="test\\lpc4_20"
lpc_o=5
getfiles("first",train_set,lpc_o,path)
```

```
(602, 201)
(378, 201)
```

Out[255]: 1

In [256]:
```
teach_gmm(path)
```

Out[256]: 1

In [257]:
```
test_set="TIMIT\\TEST\\DR5\\FMAH0"
predict(test_set, lpc_o,path)
```

```
---residual in spectrum --
accuarcy =0.995164410058027
              precision    recall  f1-score   support

           u       1.00      0.99      0.99       475
           v       0.99      1.00      1.00       559

   micro avg       1.00      1.00      1.00      1034
   macro avg       1.00      0.99      1.00      1034
weighted avg       1.00      1.00      1.00      1034
```

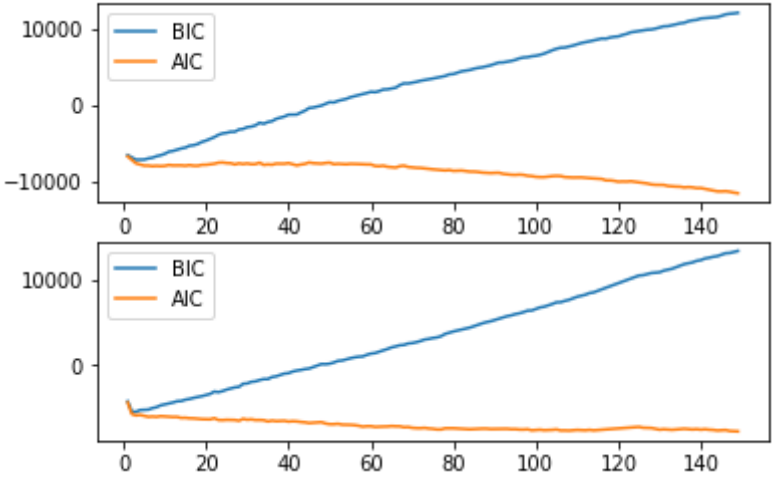
In [114]:
```
import matplotlib.pyplot as plt
import numpy as np
def testp(path):
    voiced = np.asarray(())
    unvoiced =np.asarray(())
    data=opendata('dat',path)
    n_components = np.arange(1,150)
    models_voiced = [mixture.GaussianMixture(n, covariance_type='full', random_state=0).fit(data['v'])
          for n in n_components]
    print("done")
    models_unvoiced = [mixture.GaussianMixture(n, covariance_type='full', random_state=0).fit(data['u'])
          for n in n_components]
    print("done")
    fig, axs = plt.subplots(2)
    axs[0].plot(n_components, [m.bic(data['v']) for m in models_voiced], label='BIC')
    axs[0].plot(n_components, [m.aic(data['v']) for m in models_voiced], label='AIC')
    axs[0].legend(loc='best')
    axs[1].plot(n_components, [m.bic(data['u']) for m in models_unvoiced], label='BIC')
    axs[1].plot(n_components, [m.aic(data['u']) for m in models_unvoiced], label='AIC')
    axs[1].legend(loc='best')
    return 1
```

```
  File "<ipython-input-114-4c5db9b23439>", line 2
    u       0.99      0.88      0.93       858
    ^
IndentationError: unexpected indent
```

In [55]:
```
testp(path)
```

```
done
done
```

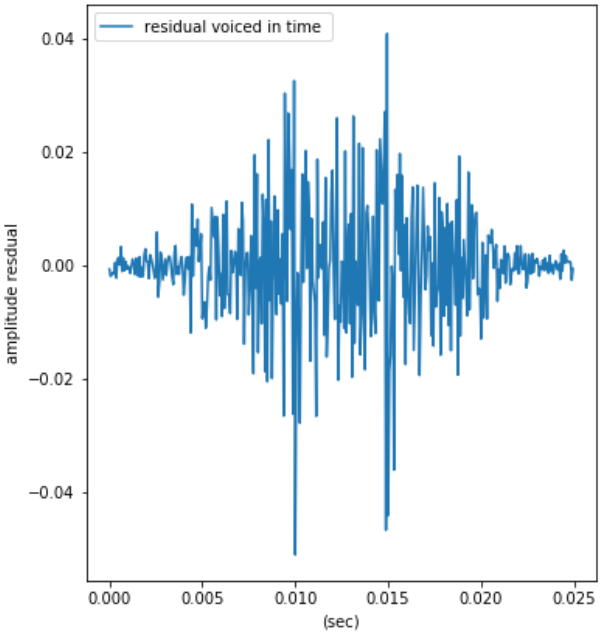
Out[55]: 1

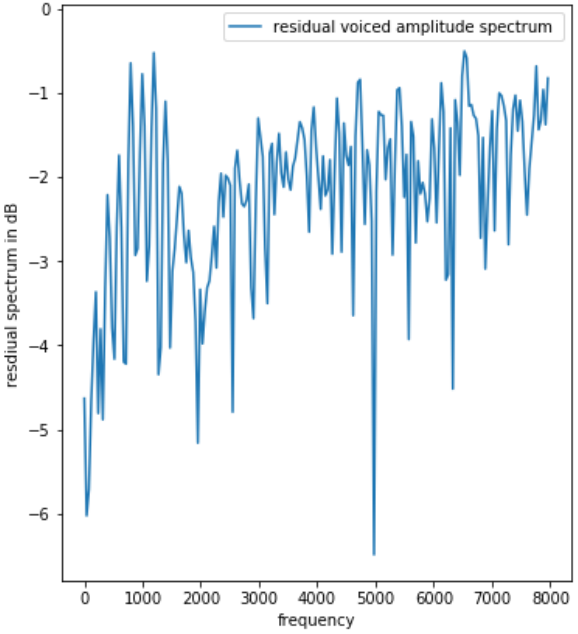


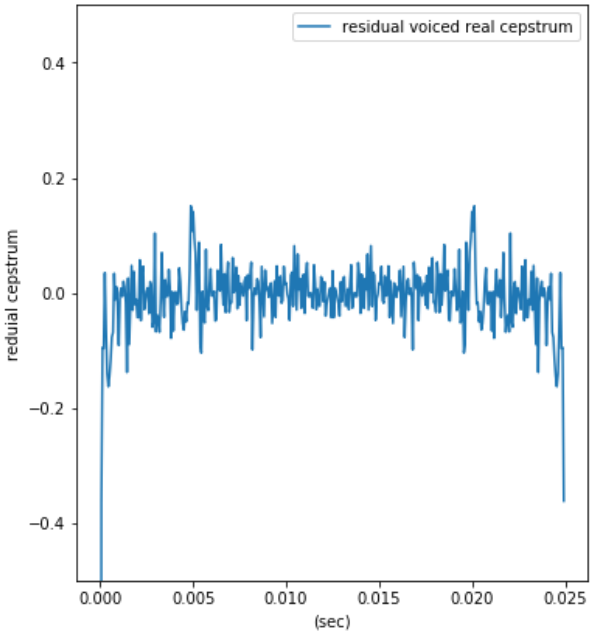
In [206]:
```
v,u = getData2("TIMIT\\TRAIN\\DR5\\FBJL0\\SX22wv",4)
```

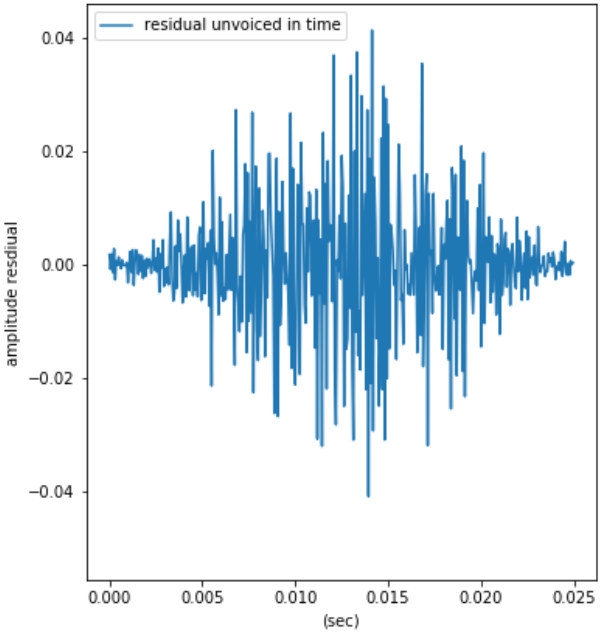
In [207]:
```
from numpy.fft import fft, ifft
import matplotlib.pyplot as plt
tpCount     = 400+2
values      = np.arange(int(tpCount/2))
timePeriod  = tpCount/16000
frequencies = values/timePeriod
time        = np.arange(0, 0.025,1/16000)
fig, axs = plt.subplots(2,3,figsize=[20,15],sharey='col')
axs[0,0].plot(time,v[1][5], label='residual voiced in time ')
axs[0,0].set_xlabel("(sec)")
axs[0,0].set_ylabel("amplitude resdual")
axs[0,0].legend(loc='best')
fv= np.log(real_spectrum(v[1][5]))
axs[0,1].plot(frequencies,fv, label='residual voiced amplitude spectrum ')
axs[0,1].set_xlabel("frequency")
axs[0,1].set_ylabel("resdiual spectrum in dB")
axs[0,1].legend(loc='best')
cepsv=real_cepstrum(v[1][5])
axs[0,2].set_ylim([-0.5,0.5])
axs[0,2].plot(time,cepsv, label='residual voiced real cepstrum')
axs[0,2].set_xlabel("(sec)")
axs[0,2].set_ylabel("reduial cepstrum")
axs[0,2].legend(loc='best')
axs[1,0].plot(time,u[1][3], label='residual unvoiced in time')
axs[1,0].set_xlabel("(sec)")
axs[1,0].set_ylabel("amplitude resdiual")
axs[1,0].legend(loc='best')
fu= np.log(real_spectrum(u[1][3]))

axs[1,1].plot(frequencies,fu, label='residual unvoiced amplitude spectrum')
axs[1,1].set_xlabel("frequency")
axs[1,1].set_ylabel("resdiual spectrum in dB")
axs[1,1].legend(loc='best')
cepsu=real_cepstrum(u[1][5])
axs[1,2].set_ylim([-0.5,0.5])
axs[1,2].plot(time,cepsu, label='residual unvoiced amplitude spectrum')
axs[1,2].set_xlabel("(sec)")
axs[1,2].set_ylabel("reduial cepstrum")
axs[1,2].legend(loc='best')
```

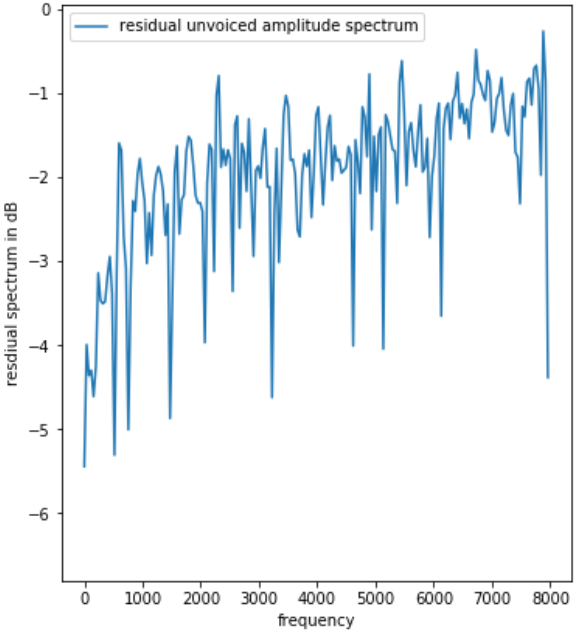
Out[207]:
```
<matplotlib.legend.Legend at 0x2d01db459e8>
```

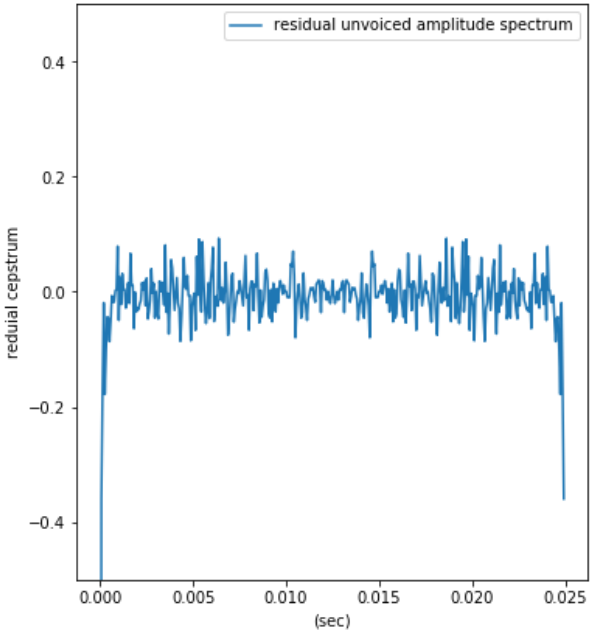

residual voiced in time
amplitude resdual
(sec)


residual voiced amplitude spectrum
resdiual spectrum in dB
frequency


residual voiced real cepstrum
reduial cepstrum
(sec)


residual unvoiced in time
amplitude resdiual
(sec)


residual unvoiced amplitude spectrum
resdiual spectrum in dB
frequency


residual unvoiced amplitude spectrum
reduial cepstrum
(sec)