Ψηφιακή Επεξεργασία Σημάτων

1η Εργαστηριακή Άσκηση

Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών ΕΜΠ

Εξάμηνο 6ο

Σερλής Εμμανουήλ Αναστάσιος, Α.Μ.:03118125, email: manosanastassis@hotmail.com

Δημήτρης Μητρόπουλος, Α.Μ.: 03118608, email: dimitris.7.mitropoulos@gmail.com

Μέρος 1

Άσκηση 1.1

	Ωcolumn		
Ωrow	0.9273	1.0247	1.1328
0.5346	*1*	*2*	*3*
0.5906	*4*	*5*	*6*
0.6535	*7*	*8*	*9*
0.7217		*0*	

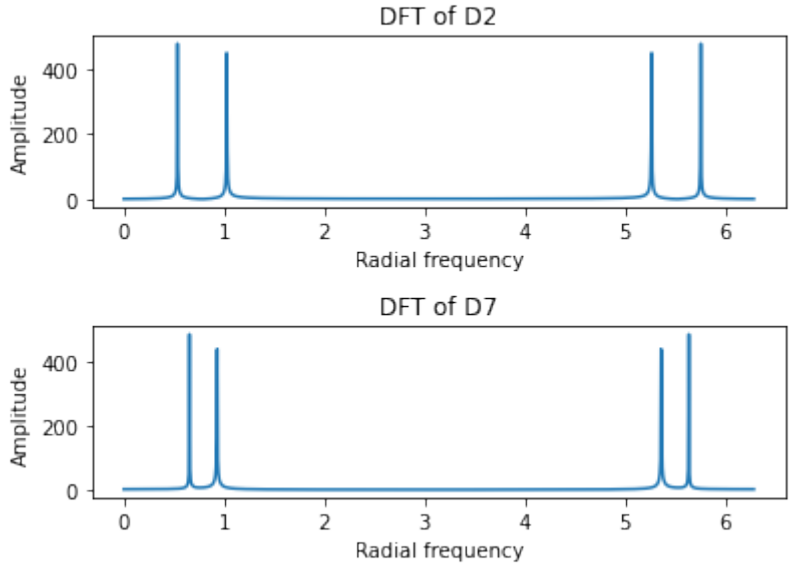
Από τον παραπάνω πίνακα παίρνουμε τους εξής συνδυασμούς για τους 10 τόνους:

1) $d_1[n] = \sin(0.5346n) + \sin(0.9273n)$
2) $d_2[n] = \sin(0.5346n) + \sin(1.0247n)$
3) $d_3[n] = \sin(0.5346n) + \sin(1.1328n)$
4) $d_4[n] = \sin(0.5906n) + \sin(0.9273n)$
5) $d_5[n] = \sin(0.5906n) + \sin(1.0247n)$
6) $d_6[n] = \sin(0.5906n) + \sin(1.1328n)$
7) $d_7[n] = \sin(0.6535n) + \sin(0.9273n)$
8) $d_8[n] = \sin(0.6535n) + \sin(1.0247n)$
9) $d_9[n] = \sin(0.6535n) + \sin(1.1328n)$
10) $d_0[n] = \sin(0.7217n) + \sin(1.0247n)$

Άσκηση 1.2

Χρησιμοποιώντας την fft() της numpy παίρνουμε για το D2 και το D7 τα εξής γραφήματα για το πλάτος:

Πλάτος των DFT_tone2 και DFT_tone7



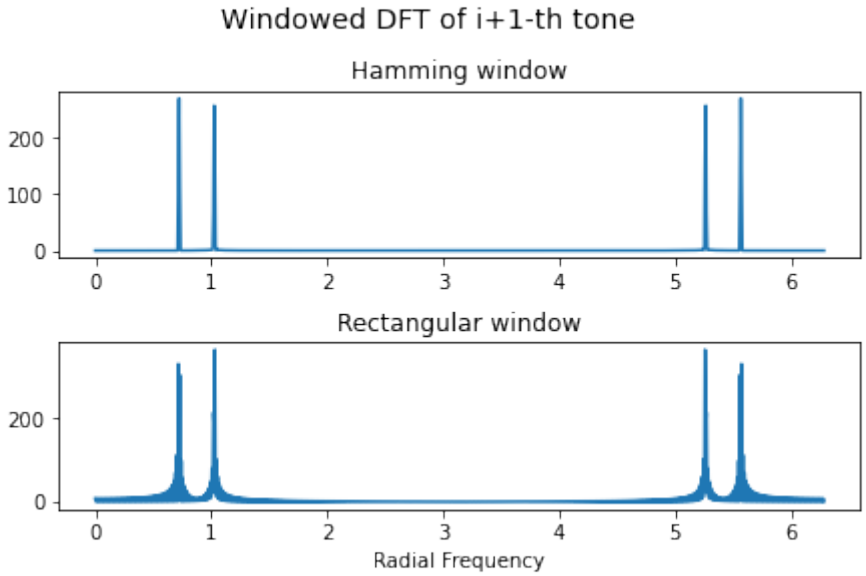
Άσκηση 1.3

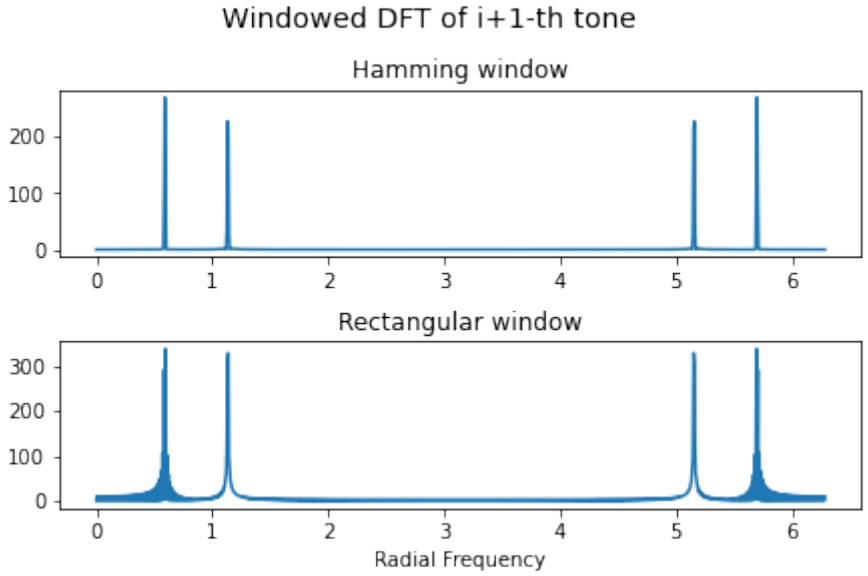
Στο συγκεκριμένο υποερώτημα, βάλαμε σε μια λίστα τις τιμές του κάθε tone όπως τα υπολογίσαμε στην άσκηση 1.2, με 100 μηδενικά ανάμεσα σε κάθε τόνο. Αυτό το σήμα μετά το αποθηκεύσαμε στο αρχείο tone_sequence.wav με fs=8192Hz.

Άσκηση 1.4

Χρησιμοποιώντας την hamming() της numpy για το Hamming window και έναν τετραγωνικό παλμό για το rectangular window υπολογίζουμε τον DFT των παραθυροποιημένων σημάτων, από το 1.3. Με αυτή τη διαδικασία παίρνουμε τα εξής γραφήματα (τα γραφήματα είναι 5 γιατί στο tone_sequence.wav έχουμε κάποιους αριθμούς παραπάνω από μια φορά):

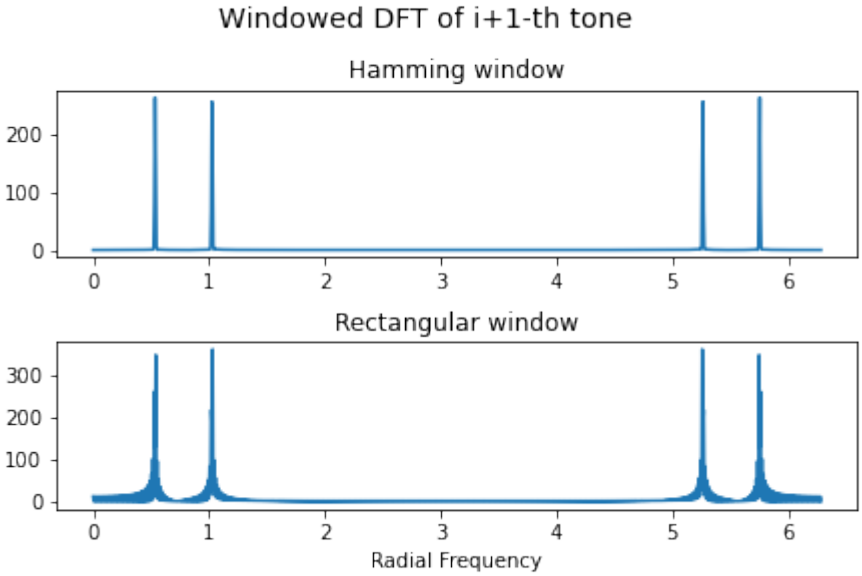
Windowed DFT of tone 0



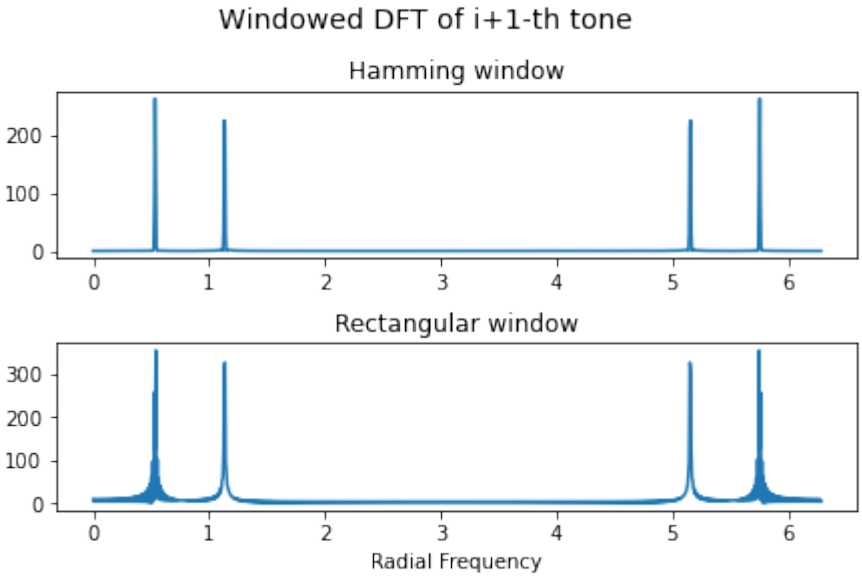


Windowed DFT of tone 6

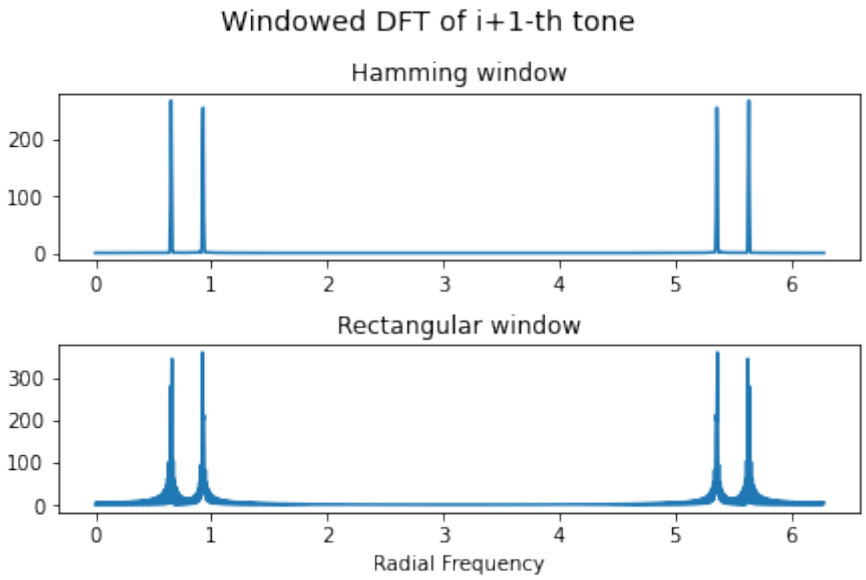
Windowed DFT of tone 2



Windowed DFT of tone 3



Windowed DFT of tone 7



Άσκηση 1.5

Οι συχνότητες που υπολογίσαμε πως είναι πιο κοντά στις touch-tone συχνότητες είναι οι εξής (τα νούμερα που υπολογίστηκαν για την συχνότητα είναι η θέση στην οποία βρίσκεται το peak στον DFT, και όχι η τιμή του peak καθε αυτή):

	Fcolumn	Frow	Ωcolumn	Ωrow
0	163	115	1.024159205070	0.722566310326
1	148	85	0.9299114254626	0.5340707511103
2	163	85	1.024159205070	0.5340707511103
3	180	85	1.130973355292	0.5340707511103
4	148	94	0.9299114254624	0.590619418875
5	163	94	1.024159205070	0.590619418875
6	180	94	1.130973355292	0.590619418875
7	148	104	0.9299114254626	0.653451271947
8	163	104	1.024159205070	0.653451271947
9	180	104	1.130973355292	0.653451271947

Άσκηση 1.6-1.7

Στην συγκεκριμένη άσκηση θέλαμε να υλοποιήσουμε μια συνάρτηση, η οποια θα λαμβάνει ως όρισμα ένα αρχείο και θα πρέπει να υπολογίζει τα touch-tones που ακούγονται. Για να γίνει αυτό, υπολογίζουμε αρχικά πόση χρήσιμη πληροφορία υπάρχει στο σήμα. Δηλαδή στο 1.3 ανάμεσα σε κάθε τόνο έχουμε 100 μηδενικά, τα οποια δεν προσθέτουν χρήσιμη πληροφορία στο συνολικό αποτέλεσμα. Έπειτα, υπολογίζουμε τα fft των touch-tone σημάτων που διαβάζουμε, και συγκρίνουμε τις τιμές των peaks με τις αντίστοιχες του ερωτήματος 1.5. Από το touch-tone σήμα της εισόδου, οι τιμές των peaks, οι οποίες είναι πιο κοντά στις “στάνταρ” τιμες του ερωτήματος 1.5, μας δείχνουν και ποιο touch-tone διαβάσαμε. Το κάθε touch-tone το αποθηκεύουμε σε μια λίστα και έπειτα εκτυπώνουμε αυτήν την λίστα.

Εξήχθησαν τα κάτωθι αποτελέσματα ανά αρχείο:

tone από 1.3: [0, 6, 2, 3, 6, 7, 3, 3]

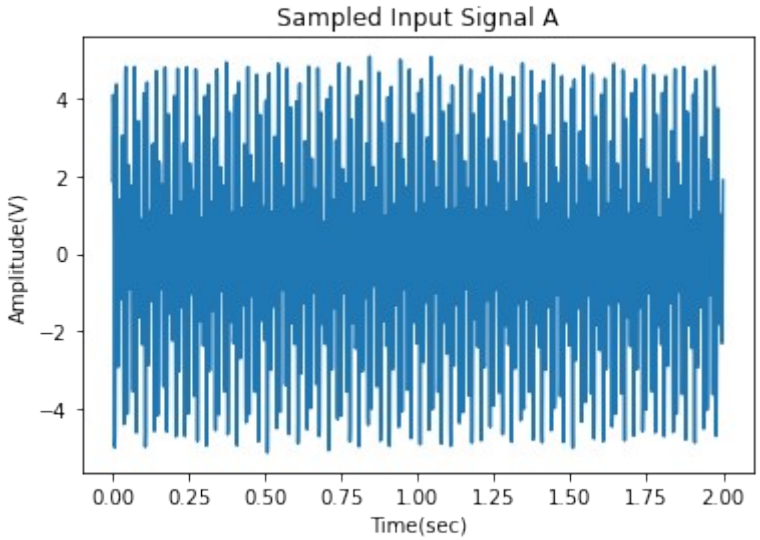
easySig: [8, 1, 0, 3, 9, 6, 3, 8]

hardSig: [4, 8, 1, 9, 2, 1, 5, 3, 6, 3]

Μέρος 2

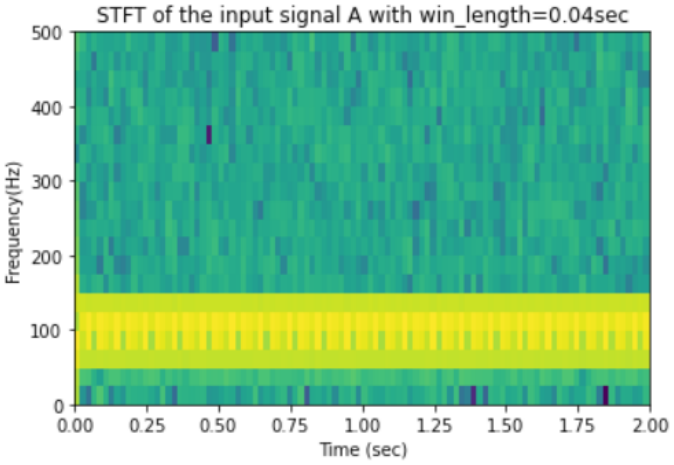
Άσκηση 2.1

A. Το δειγματοληπτημένο σήμα εισόδου:

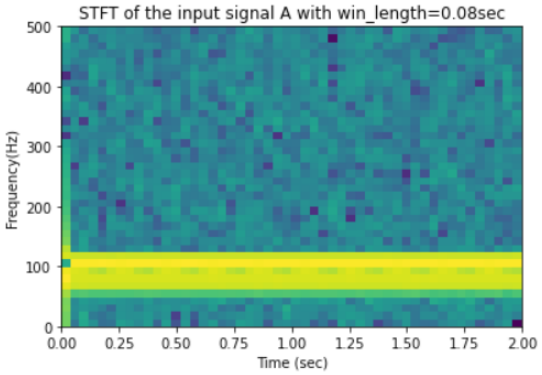


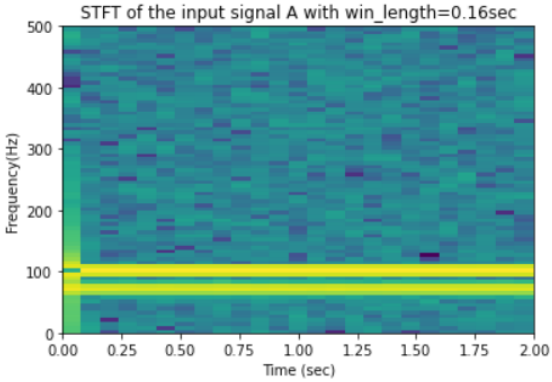
Σημειώνεται ότι:
#δειγμάτων = N =Τολ/Ts=Τολ*Fs = 2sec * 1000Hz => N=2000 δείγματα

B. STFT για μήκος παραθύρου 0.04sec



C. STFT για μήκη παραθύρου 0.08sec και 0.16sec[1]:



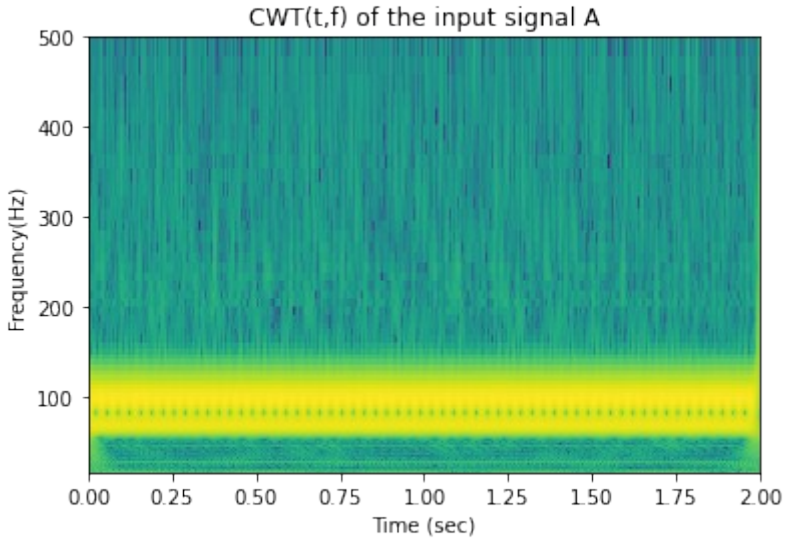


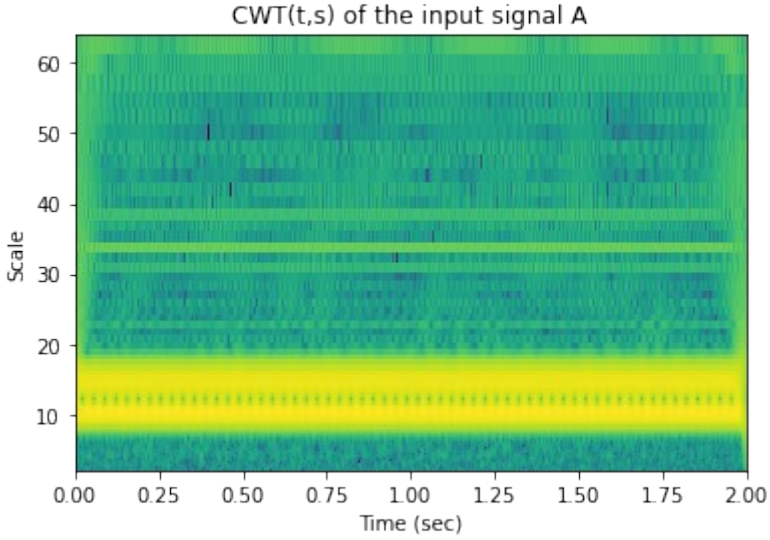
Όσον αφορά τα αποτελέσματα των ερωτημάτων (β) και (γ), παρατηρούμε ότι αύξηση του χρονικού μήκους του παραθύρου οδηγεί σε καλύτερο συχνοτικό προσδιορισμό του σήματος εισόδου.

Μάλιστα, στην περίπτωση όπου win_length = 0.16sec, δημιουργούνται δύο χωριστές συχνοτικές “λωρίδες”, που αντιστοιχούν στις συχνότητες των ημιτόνων στην είσοδο (δηλαδή σε 70 και 100 Hz)

1 Note: Σε όλα τα ερωτήματα που αφορούν STFT και CWT γίνεται χρήση λογαριθμικής κλίμακας για βελτιωμένο scaling

D. Οι ζητούμενες γραφικές παραστάσεις:



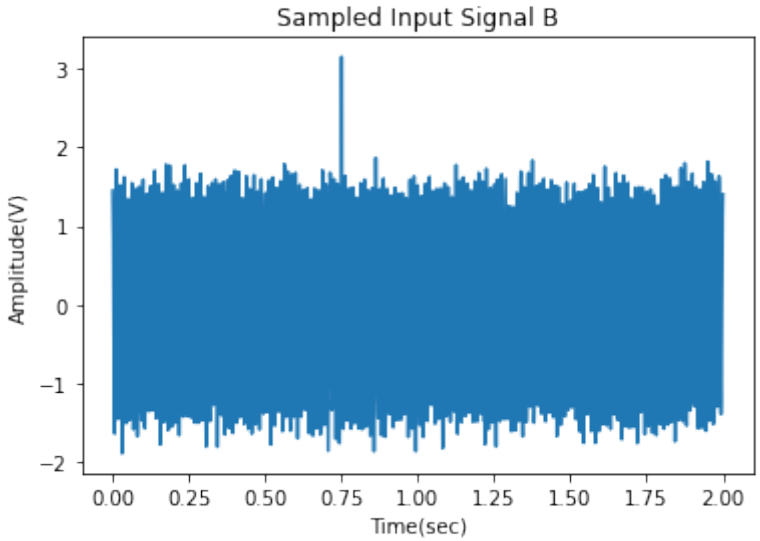


Ε. Παρατηρούμε ότι ο CWT πραγματοποιεί καλύτερη συχνοτική ανάλυση του σήματος εισόδου, χωρίς να χρειαστεί τροποποίηση κάποιας παραμέτρου του, όπως συμβαίνει στον STFT. Το γεγονός αυτό έγκειται στην χρήση χρονικά μεταβαλλόμενων παραθύρων και οδηγεί σε πιο “smooth” λωρίδα γύρω από τα 100Hz.

Ωστόσο, ο CWT αδυνατεί να διαχωρίσει (σε οπτικό επίπεδο) τις 2 συχνότητες εισόδου, πράγμα που επιτυγχάνει ο STFT για win_length=0.16.

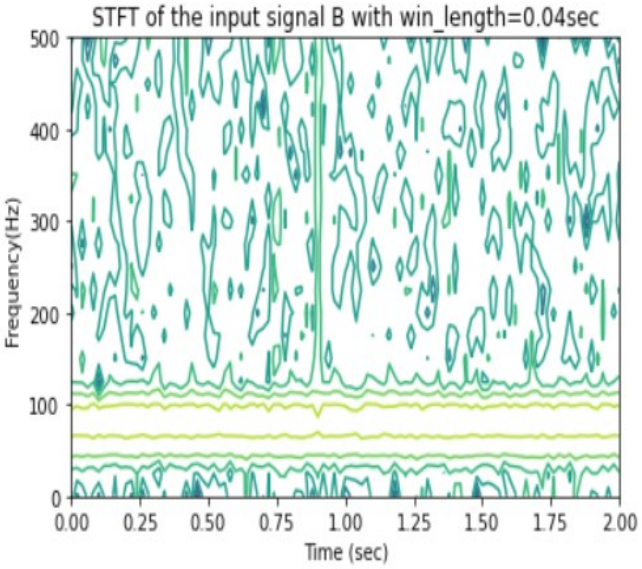
Άσκηση 2.2

A.

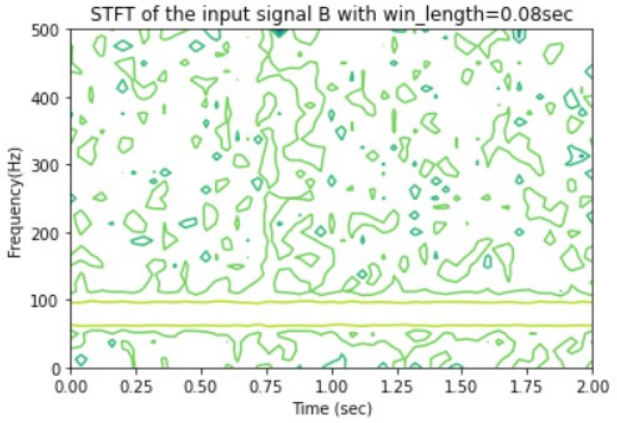


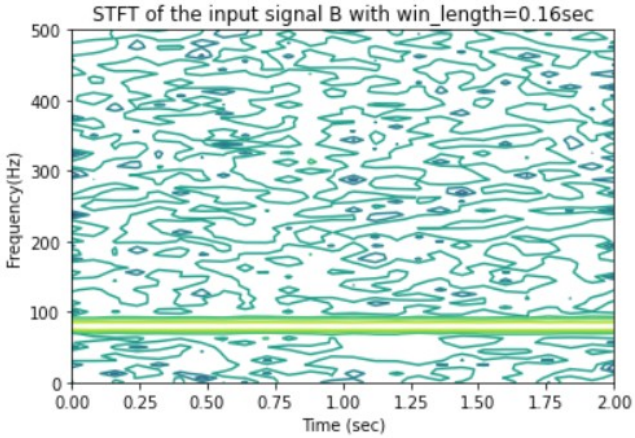
Η αυξημένη "πυκνότητα" των δειγματοληπτημένων σημείων (σε σχέση με το ερώτημα 2.2(α)) έγκειται στην προσθήκη θορύβου (ο όρος 1.5υ(t))

B.



C.

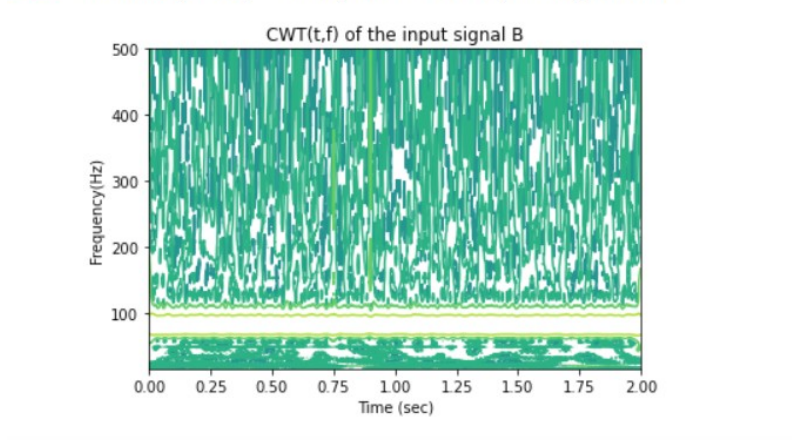


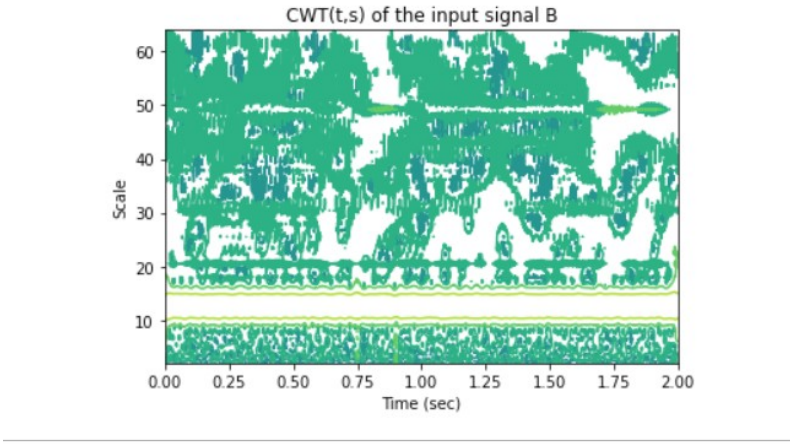


Παρατηρούμε ότι-όσον αφορά τις δυναμικές γραμμές του STFT- αυτές συγκλίνουν στην περιοχή των 80 Hz (συχνότητα cos) καθώς αυξάνεται το win_length.

Ωστόσο, αύξηση του παραθύρου σημαίνει και καλύτερη συχνοτική ανάλυση, γεγονός που προκύπτει από την σταδιακή μετατροπή των γραμμών από κυματικά μεταβαλλόμενες (για 0.04sec) σε ευθείες(στα 0.16 sec).

D.





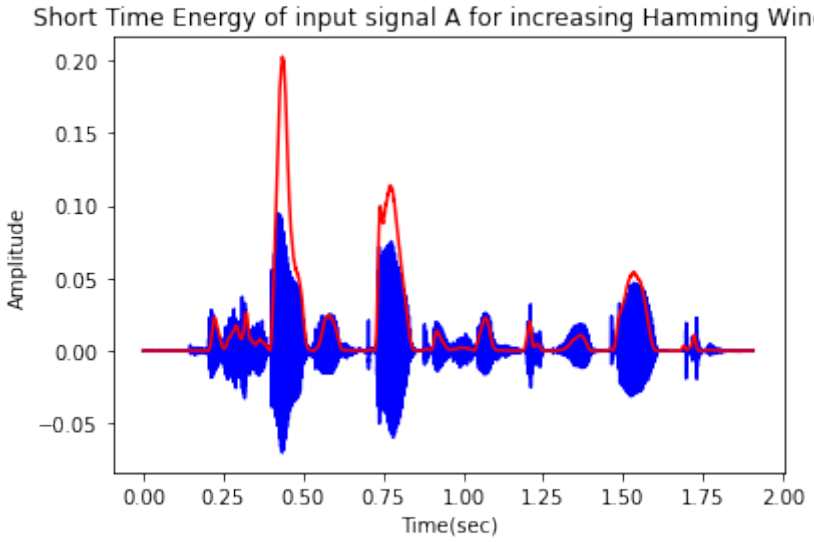
E. Ομοίως με το ερώτημα 2.1(ε), παρατηρείται η προσαρμοστική ικανότητα του CWT έναντι του STFT, μιας και οι δυναμικές γραμμές προκύπτουν εξ’αρχής “smooth”, λόγω των διαρκώς μεταβαλλόμενων παραθύρων που χρησιμοποιούνται.

Μέρος 3

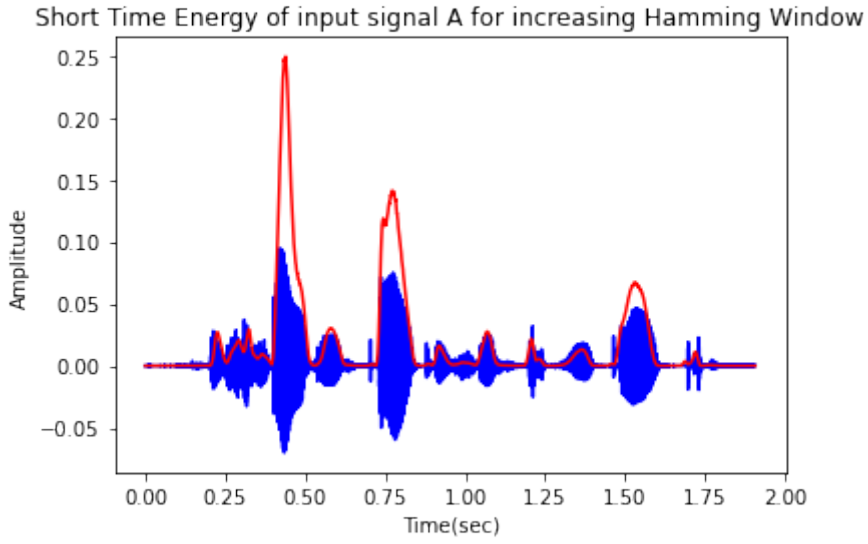
Άσκηση 3.1

Αφού πραγματοποιήθηκε εισαγωγή του ζητούμενου αρχείου στην Python και δειγματοληψία του στα 16kHz, υπολογίστηκαν η ενέργεια βραχέος χρόνου(STE) και ο ρυθμός εναλλαγής προσήμου (ZC) για διάφορες τιμές του win_length του Hamming Window:

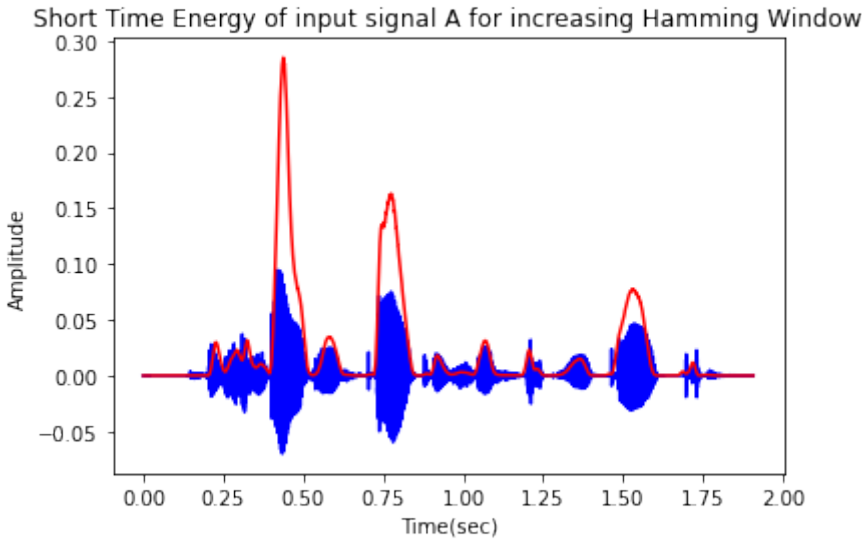
Short time energy of input signal A for 20ms



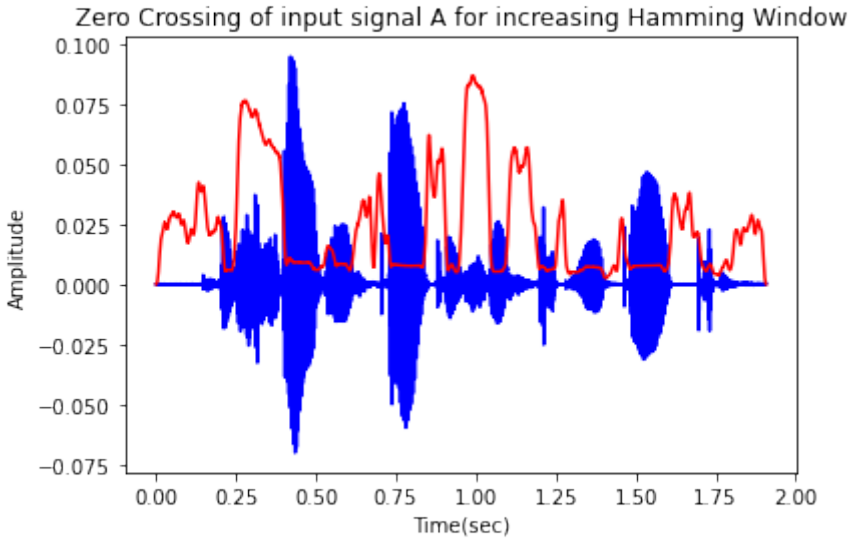
Short time energy of input signal A for 25ms



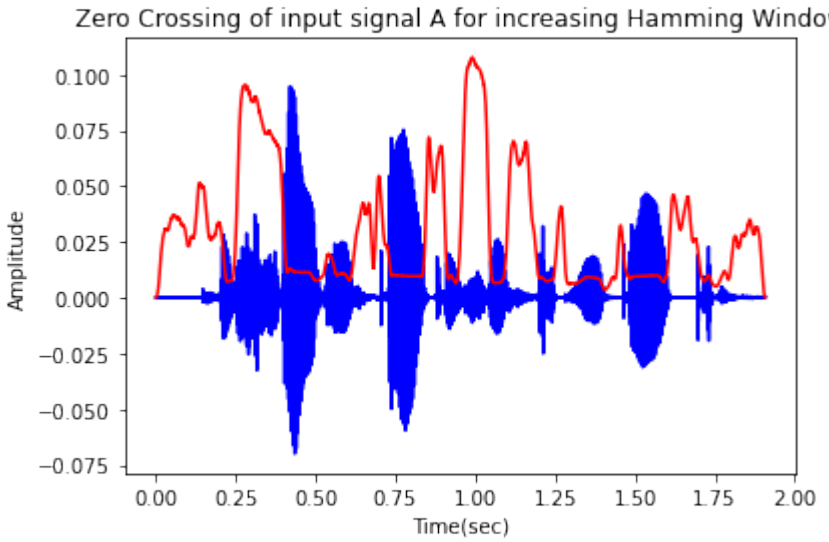
Short time energy of input signal A for 29ms



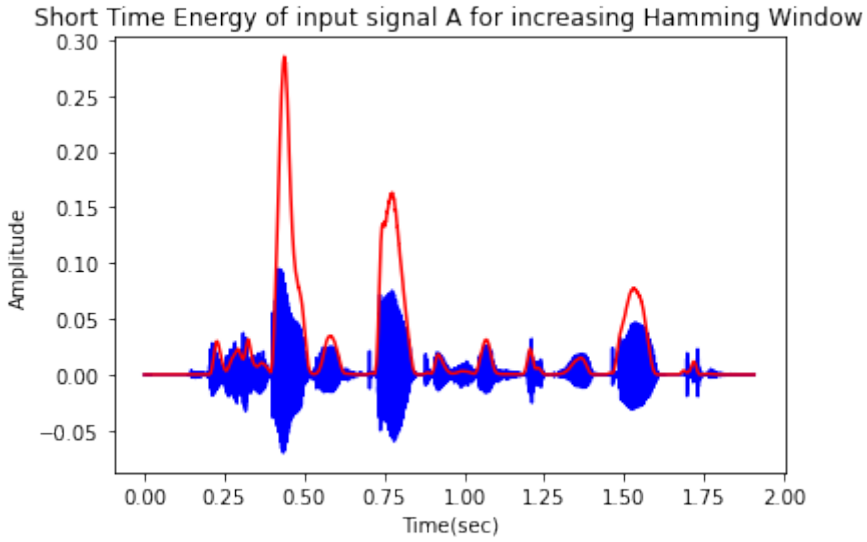
Zero crossing of input signal A for 20ms



Zero crossing of input signal A for 25ms



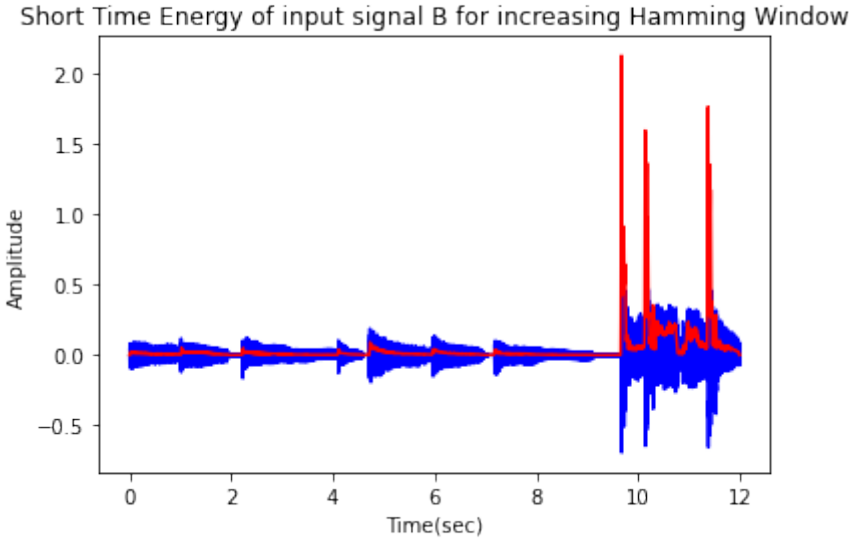
Zero crossing of input signal A for 29ms



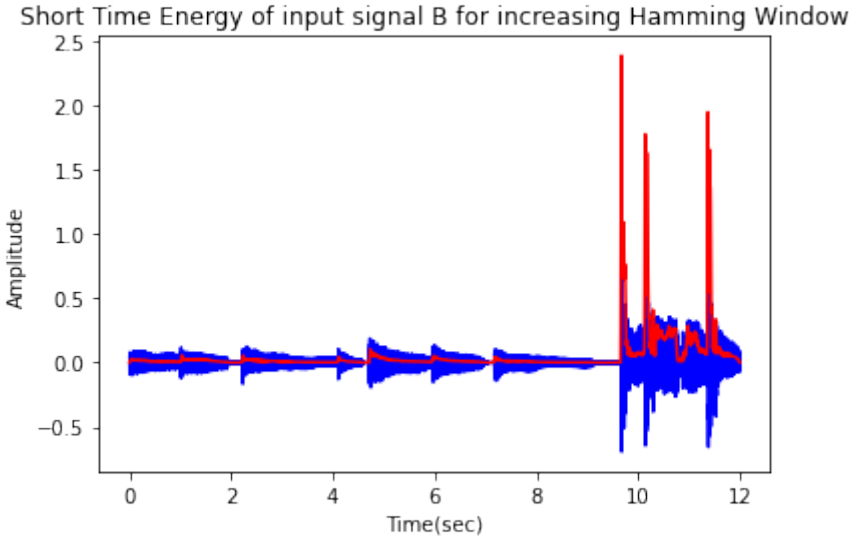
*<u>Σημείωση:</u> στο αρχείο python υπάρχουν τα γραφήματα για 20-29ms αλλά για την αναφορά διαλέξαμε 3 γραφήματα ενδεικτικά.

Άσκηση 3.2

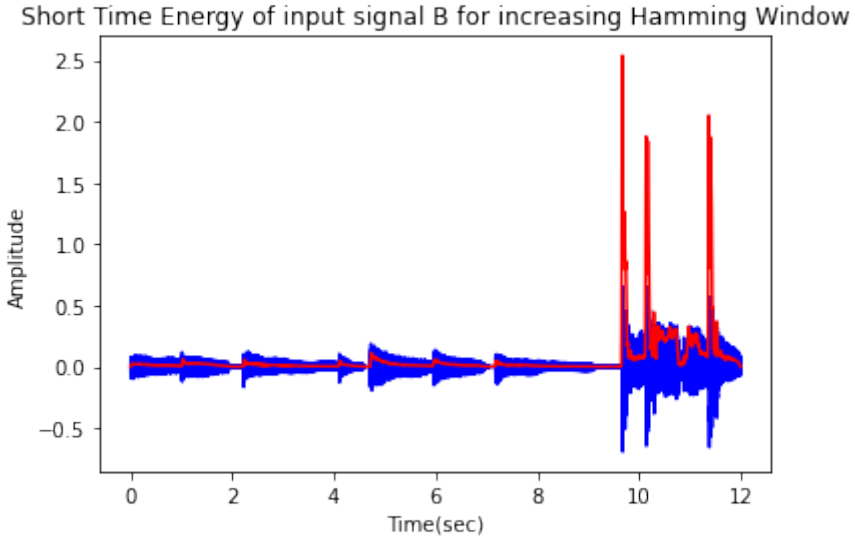
Short time energy of input signal A for 20ms



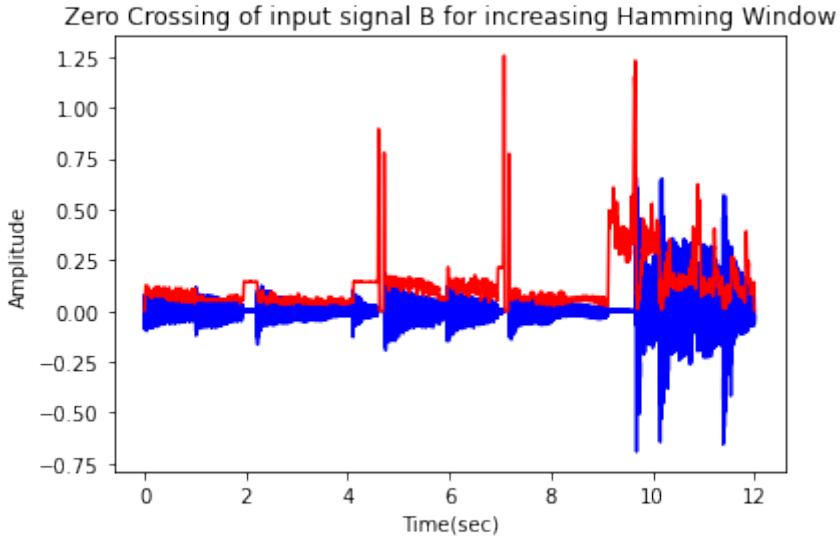
Short time energy of input signal A for 25ms



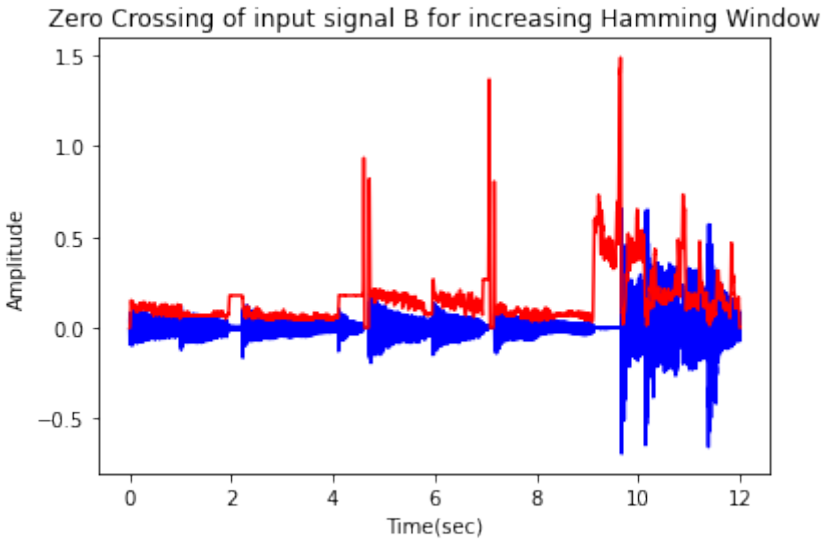
Short time energy of input signal A for 29ms



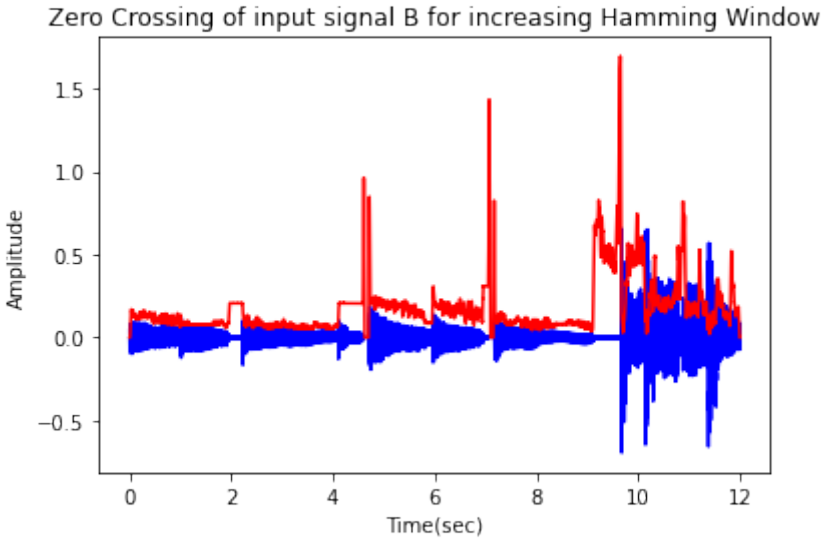
Zero crossing of input signal A for 20ms



Zero crossing of input signal A for 25ms



Zero crossing of input signal A for 29ms



*Σημείωση: στο αρχείο python υπάρχουν τα γραφήματα για 20-29ms αλλά για την αναφορά διαλέξαμε 3 γραφήματα ενδεικτικά.

Παρατηρήσεις-Συμπεράσματα:

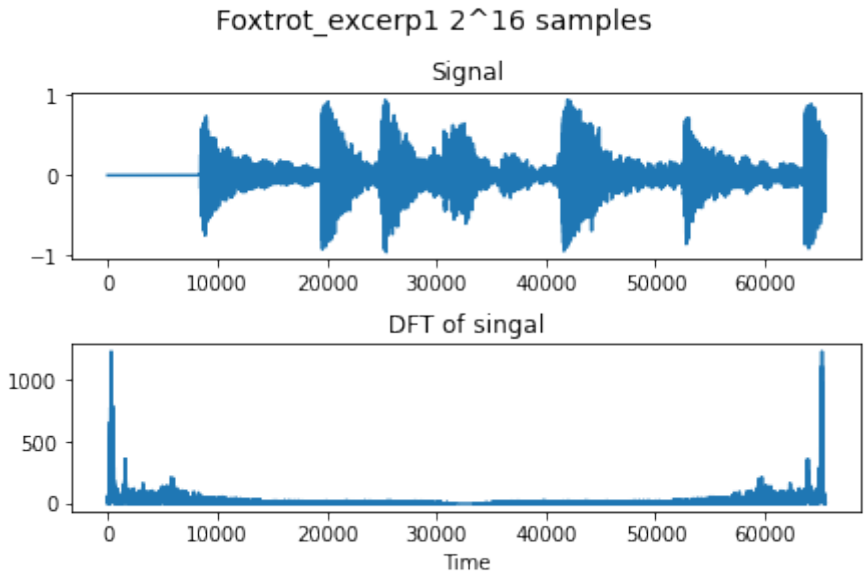
1) Αρχικά, παρατηρούμε την ορθή λειτουργία τόσο του STE όσο και του ZC στα 2 παραπάνω σήματα εισόδου.
    a) Από την μία πλευρά, ο STE λαμβάνει υψηλές τιμές για τα φωνήεντα και χαμηλές για τα σύμφωνα, μιας και Εφων >Εσυμφ
    b) Από την άλλη πλευρά, ο ZC λαμβάνει χαμηλές τιμές για τα φωνήεντα και υψηλές τιμές για τα σύμφωνα. Αυτό ισχύει, αφού ο ZC είναι ανάλογος του πλήθους/συχνότητας των μεταβολών του σήματος. Προφανώς, ένα σύμφωνο-έχοντας χαμηλή ενέργεια-βρίσκεται "πιο κοντά" στον άξονα-t και συνεπώς αλλάζει πρόσημο πιο εύκολα (εξού και έχει μεγάλο ZC Value), ενώ το ακριβώς αντίστροφο συμβαίνει με ένα φωνήεν.

Συνεπώς, καθεμία από τις παραπάνω μεθόδους ενδείκνυται για τον διαχωρισμό φωνηέντων και συμφώνων σε ένα ηχητικό σήμα.

2) Όσον αφορά τις τιμές του win_length, η αύξηση του μήκους του παραθύρου οδηγεί σε σταδιακό smoothing των γραφικών STE και ZC, αν και η διαφορά αυτή δεν είναι εμφανής μιας και τα παράθυρα κυμαίνονται σε μικρό εύρος τιμών (20-30 ms)

Μέρος 4

Άσκηση 4.1



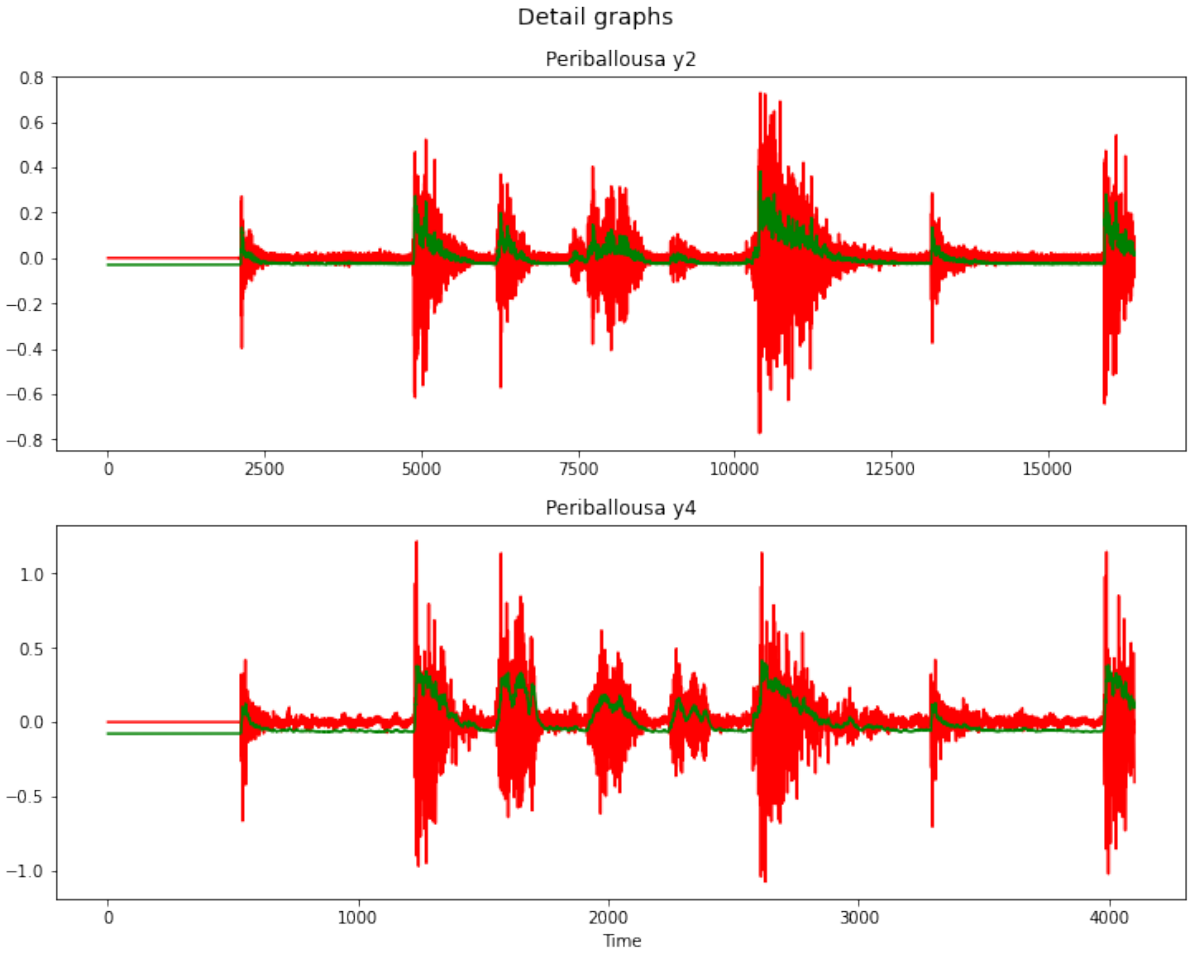
Παρατηρούμε στο αρχικό σήμα μήκους 2^16 samples (περίπου 3 δευτερολέπτων) πως είναι περιοδικό. Αυτό ισχύει, μιας και στο διάστημα των 3sec έχουμε-περίπου-6 επαναλήψεις του σήματος, γεγονός που αντιστοιχεί σε περίοδο 0.5sec

Άσκηση 4.2

Χρησιμοποιώντας την συνάρτηση wavedec() της pywavelet, ορίζουμε 8 μεταβλητές στις οποίες αποθηκεύουμε διαδοχικά το περιεχόμενο της κάθε detail του σήματος (στην 8η αποθηκεύουμε το approximation σήμα).

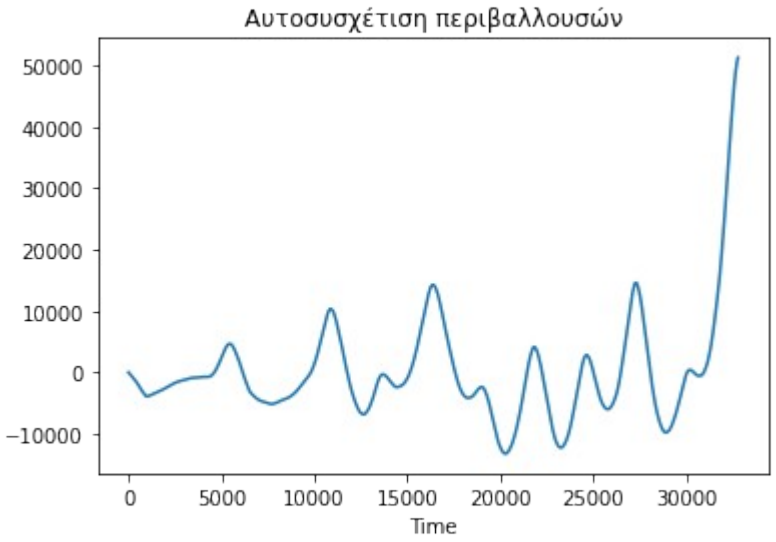
Άσκηση 4.3

Ακολουθώντας την διαδικασία της εκφώνησης παίρνουμε τα εξής 2 γραφήματα:



Άσκηση 4.4

Αθροίζοντας όλες τις περιβάλλουσες και το approximation signal υπολογίζοντας την αυτοσυσχέτιση της στη συνέχεια(για τον μισό αριθμό δειγμάτων), παίρνουμε το εξής γράφημα:

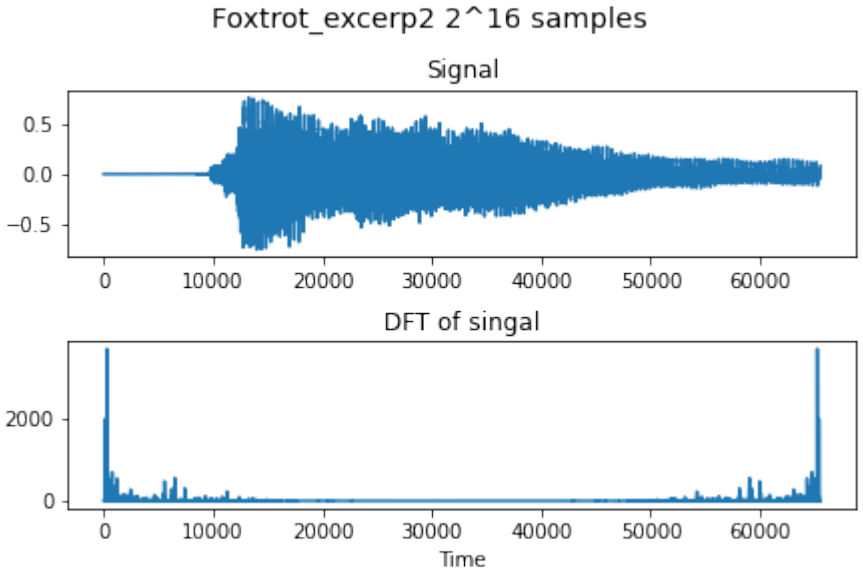


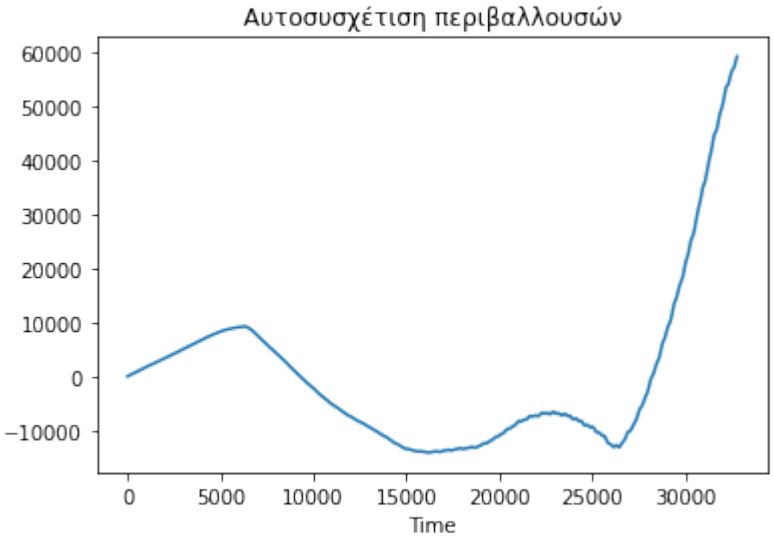
Άσκηση 4.5

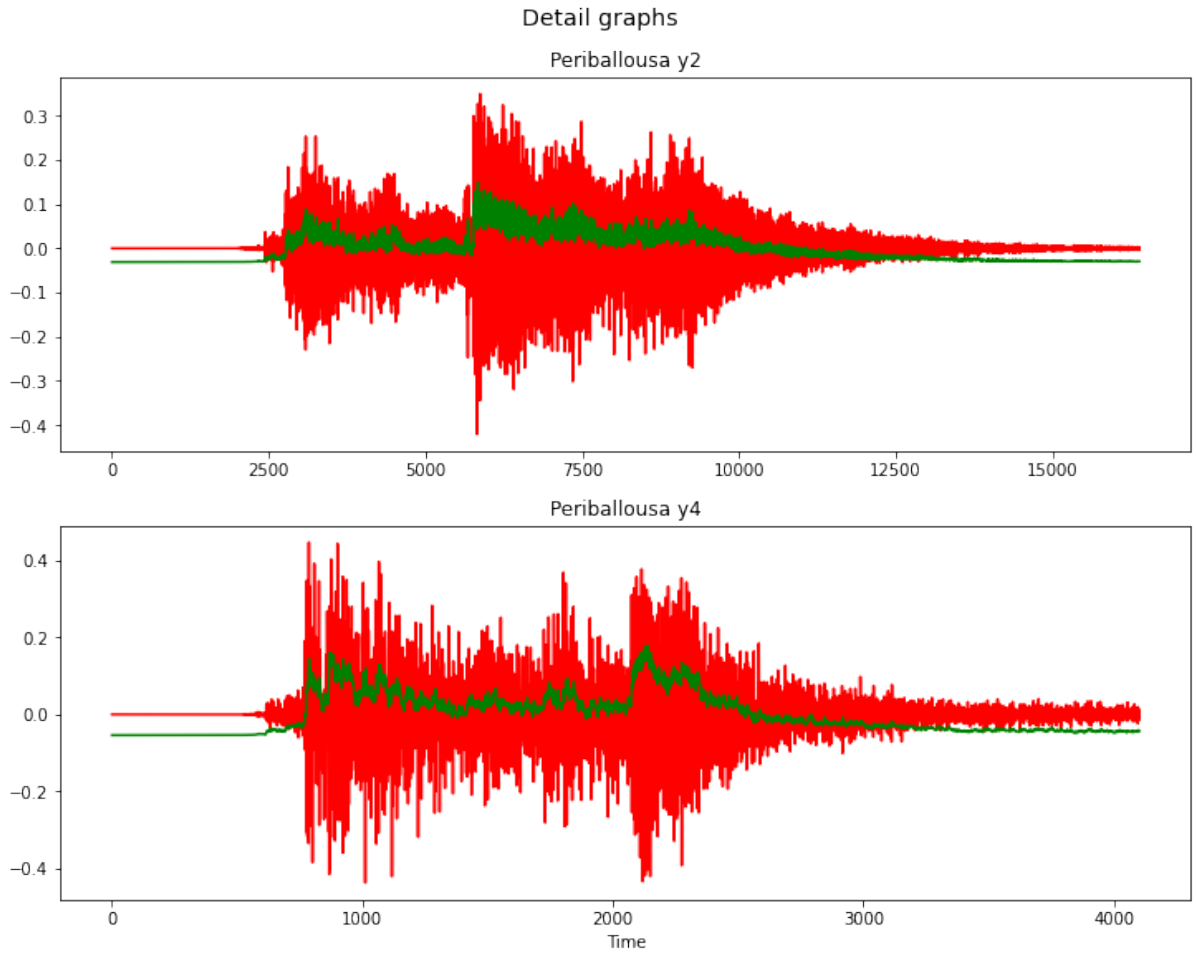
Για το BPM παίρνουμε το peak (το οποίο υπολογίζουμε με την argrelextrema()), το οποίο μας δίνει peak στο sample 10878. Για το BPM κάνουμε, BPM=60*22050/peak=121.

Άσκηση 4.6

Λειτουργώντας αντίστοιχα με την 4.5 υπολογίζουμε για το foxtrot_excerpt2 τα εξής γραφήματα:

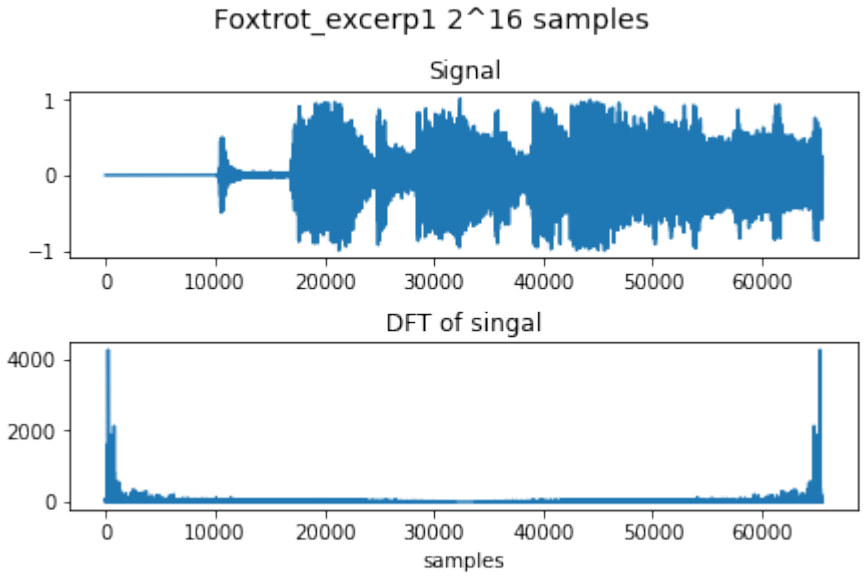


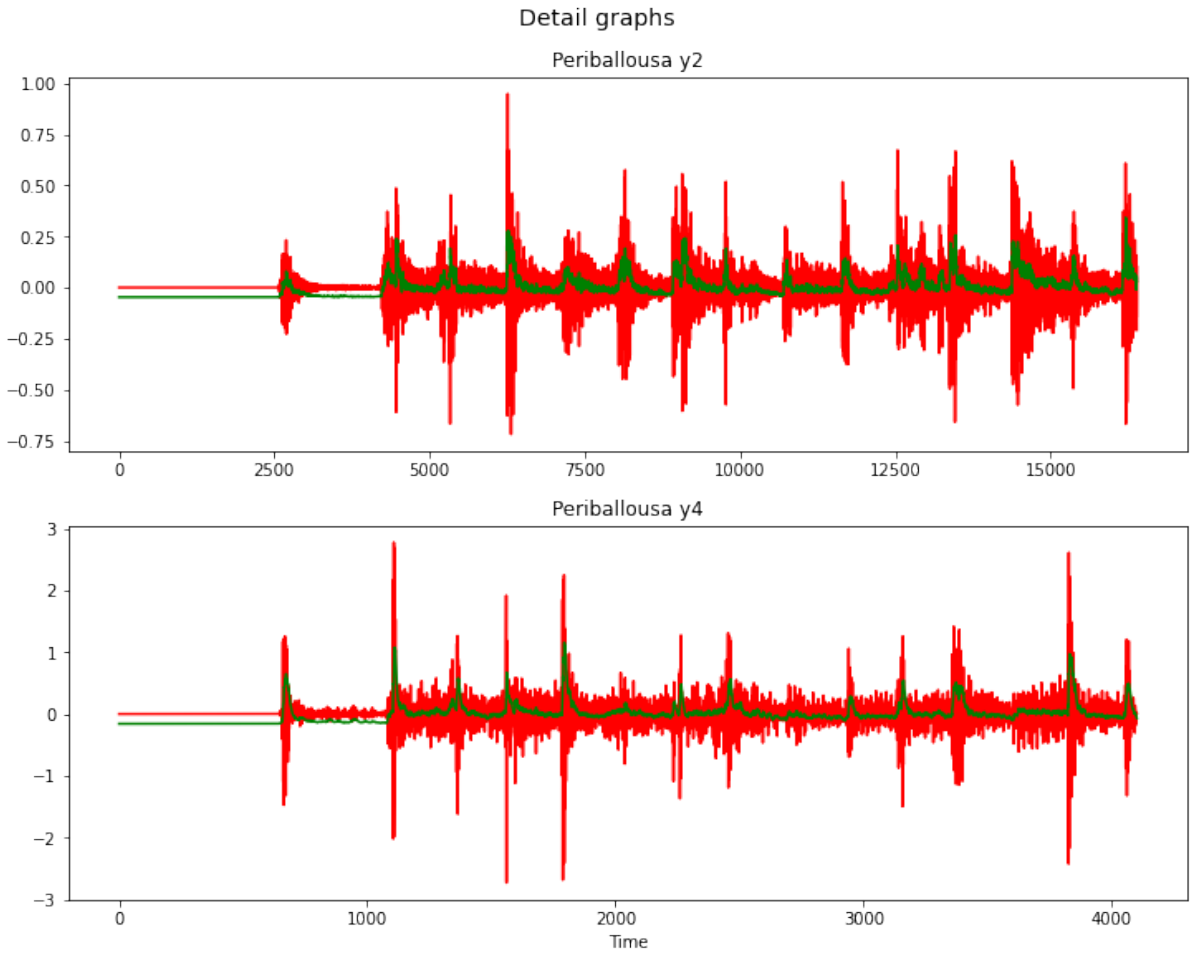


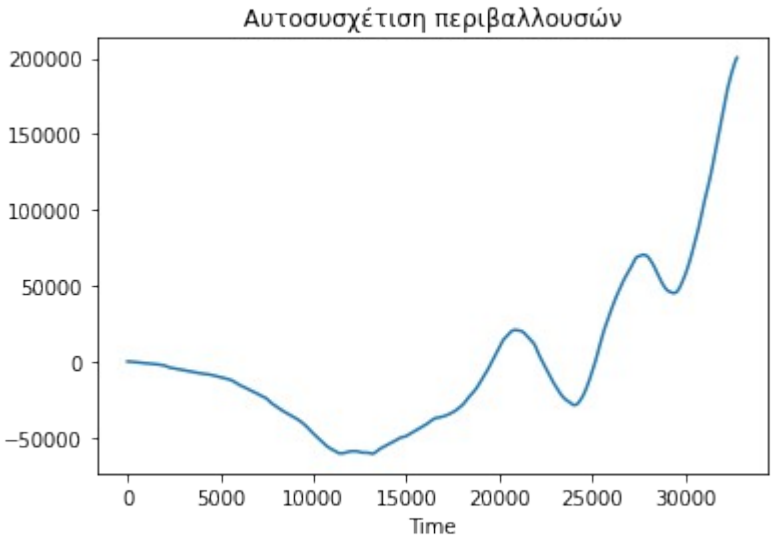


Το BPM υπολογίζεται περίπου στα 85.

Για το salsa_excerpt παίρνουμε:







Ακολουθώντας την διαδικασία παίρνουμε για το salsa_excerpt 108 για το BPM.