GIE.M.Π. ΣΗΜΜΥ

ΨΗΦΙΑΚΉ ΕΠΕΞΕΡΓΑΣΊΑ ΣΉΜΑΤΟΣ

ΑΚΑΔ.- ΈΤΟΣ 2024-2025

Εισαγωγή στην Ψηφιακή Επεξεργασία Σημάτων με Python και Εφαρμογές σε Ακουστικά Σήματα

2η Εργαστηριακή Άσκηση

Ον/μα Συνεργατών:

Δημήτριος Βασιλειάδης ΑΜ: 03122111

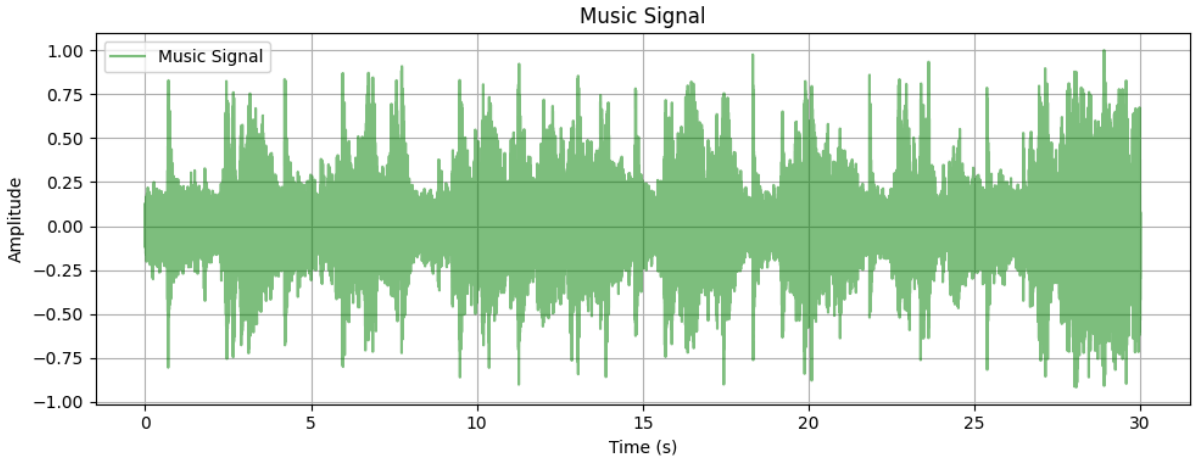
Γρηγόριος Σταματόπουλος ΑΜ: 03122039

Μέρος 1 – Ψυχοακουστικό Μοντέλο 1

Σκοπός του Μέρους 1 είναι η δημιουργία μιας συνάρτησης που υλοποιεί το Ψυχοακουστικό Μοντέλο 1, η οποία παίρνει σαν είσοδο το παραθυροποιημένο πλαίσιο ανάλυσης και επιστρέφει το συνολικό κατώφλι κάλυψης Tg.

Βήμα 1.0: Προεπεξεργασία του σήματος

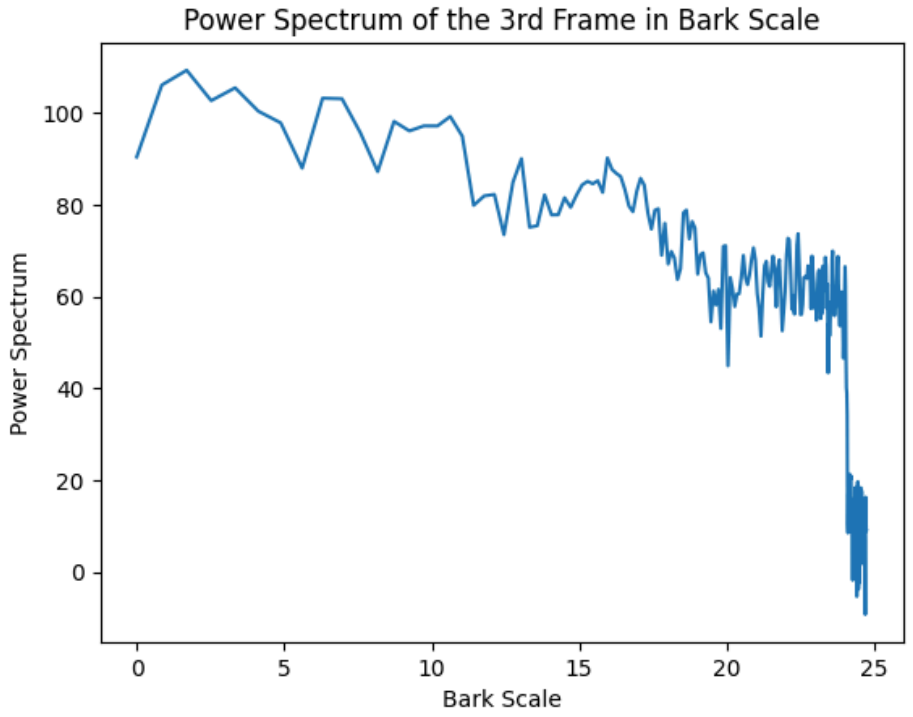
Καταρχήν, κάνουμε import τις κατάλληλες βιβλιοθήκες που θα χρειαστούν σε όλη την διάρκεια της άσκησης. Στο βήμα αυτό, διαβάσαμε το αρχικό ηχητικό σήμα μουσικής από το αρχείο music_dsp.wav και το μετατρέψαμε από στερεοφωνικό σε μονοφωνικό μέσω της συνάρτησης librosa.load(). Στη συνέχεια, κανονικοποιήσαμε το σήμα σε όλη τη διάρκειά του στο διάστημα [-1, 1] διαιρώντας όλα τα δείγματα με τη μέγιστη απόλυτη τιμή του. Στη συνέχεια, εφαρμόζουμε παράθυρα Hanning 512 δειγμάτων. Το σήμα χωρίζεται σε συνεχόμενα πλαίσια ανάλυσης μήκους 512 δειγμάτων (χωρίς επικάλυψη), πάνω στα οποία θα εφαρμόσουμε το ψυχοακουστικό μοντέλο. Συνολικά, για σήμα διάρκειας 30 sec με συχνότητα δειγματοληψίας = 44.1kHz λαμβάνουμε 2583 πλαίσια ανάλυσης. Παρακάτω φαίνεται η κυματομορφή του σήματος μουσικής , μέσω συναρτήσεων της βιβλιοθήκης της maplotlib.pyplot.

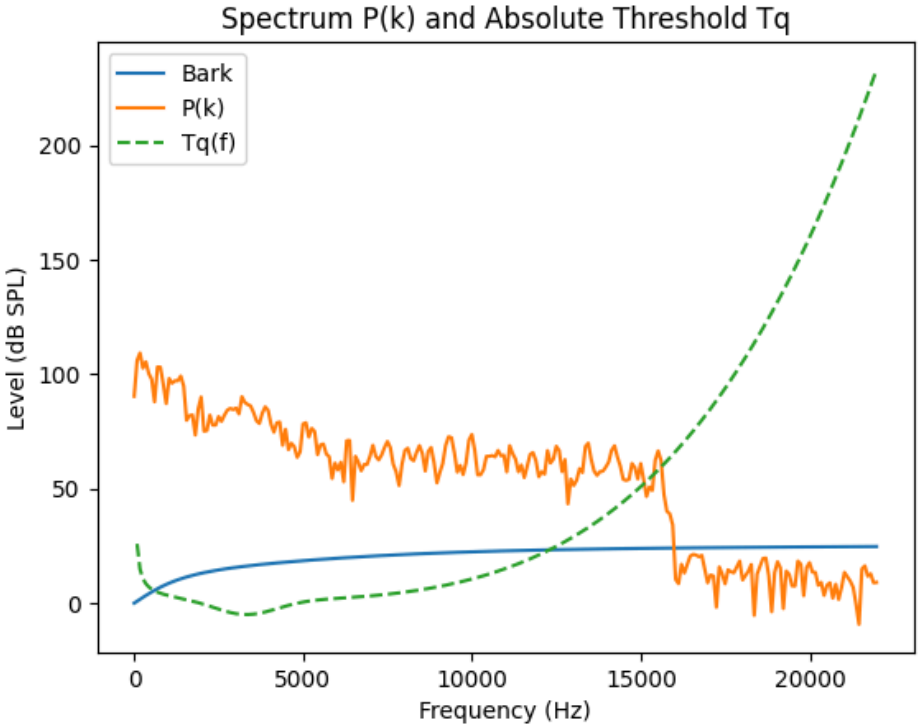


Βήμα 1.1: Φασματική Ανάλυση

Αρχικά θεωρούμε την κλίμακα bark(f) χρησιμοποιώντας τον δοσμένο τύπο. Έπειτα θεωρούμε συνάρτηση για να υπολογίζουμε το φάσμα ισχύος. Υπολογίζουμε μέσω της np.fft.rfft() τον μετασχηματισμό FFT για k απο 0 έως N/2 και στη συνέχεια υπολογίζουμε το φάσμα ισχύος P(k) από τον τύπο που μας δίνεται. Τώρα, απεικονίζουμε το φάσμα ισχύος P(k) ενός ενδεικτικού πλαισίου και συγκεκριμένα του 3ου πλαισίου σε συνάρτηση με την κλίμακα Bark. Επίσης, έχοντας υπολογίσει σωστά με βάση τον δοσμένο τύπο το απόλυτο

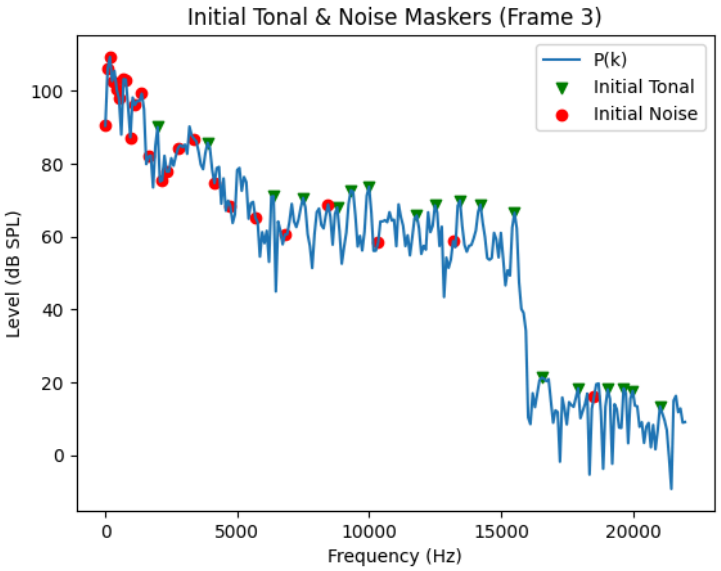
κατώφλι ακοής, σε κοινό διάγραμμα παρουσιάζουμε το φάσμα ισχύος, την κλίμακα Bark, και το κατώφλι ακοής συγκριτικά με την συχνότητα, ως προς το επίπεδο τους σε dB.





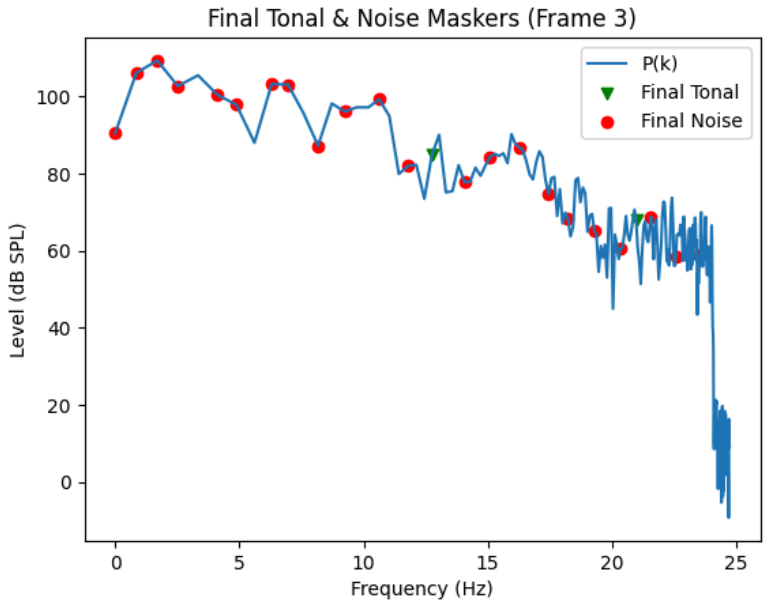
Βήμα 1.2: Εντοπισμός μασκών τόνων και θορύβου (Maskers)

Στη συνέχεια, έχοντας φορτώσει μέσω της **np.load()** τα αρχεία numpy, βάσει του μαθηματικού μοντέλου που μας παρέχεται, υλοποιούμε την συνάρτηση που εντοπίζει τις τονικές μάσκες ελέγχοντας για τοπικά μέγιστα στις διαφορετικές συχνοτικές περιοχές. Πιο συγκεκριμένα έχουμε θεωρήσει μία συνάρτηση για το Δκ, και έχουμε υλοποιήσει τον πίνακα ST(k). Είναι ένας **boolean πίνακας** που σηματοδοτεί τις θέσεις **k** όπου υπάρχουν τονικές μάσκες. Για κάθε συχνότητα k, ελέγχει αν είναι τοπικό μέγιστο (k > k±1) και επαληθεύει αν η ισχύς του k υπερβαίνει τις γειτονικές συχνότητες (σε εύρος Δk) κατά τουλάχιστον 7 dB, με το Δκ να εξαρτάται από τη ζώνη συχνοτήτων (χαμηλές/μεσαίες/υψηλές). Έπειτα ορίζουμε και την εξής συνάρτηση που **PTM(k)** , που υπολογίζει τη συνολική ισχύ κάθε μάσκας αθροίζοντας την ενέργεια της συχνότητας k και των άμεσων γειτόνων της (k-1, k+1), μετατρεπόμενη σε dB. Μόνο οι θέσεις με **ST[k] = True** λαμβάνουν μη μηδενική τιμή. Τέλος βρίσκουμε μέσω της **np.where()** τα αρχικά σημεία τόνων και μασκών,μία φορά για την συνάρτηση μας PTM(k) και μία φορά για τον **προϋπολογισμένο πίνακα P_NM** , και μέσω των γνωστών συναρτήσεων εμφανίζουμε το αντίστοιχο διάγραμμα, για ένα συγκεκριμένο frame.



Βήμα 1.3: Μείωση και αναδιοργάνωση των μασκών

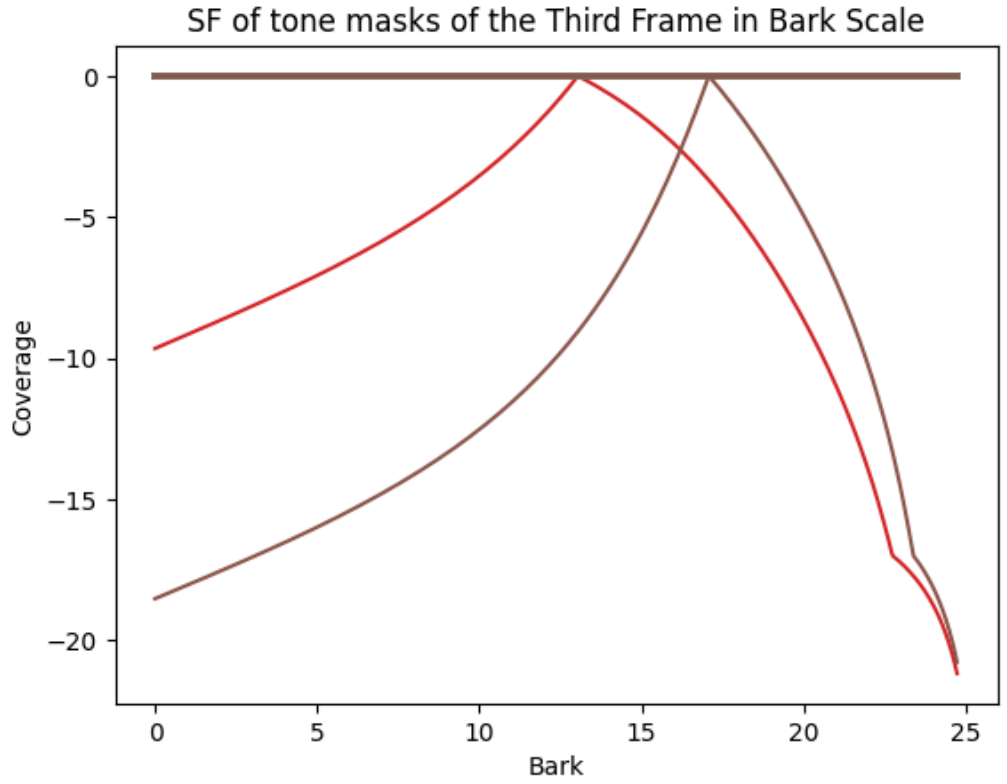
Επαναλαμβάνουμε το προηγούμενο βήμα και μέσω της **np.where()** , βρίσκουμε για τους καινούριους, μειωμένους πίνακες , τις θέσεις τόνων και μασκών. Το αποτέλεσμα είναι γραφικά το εξής:



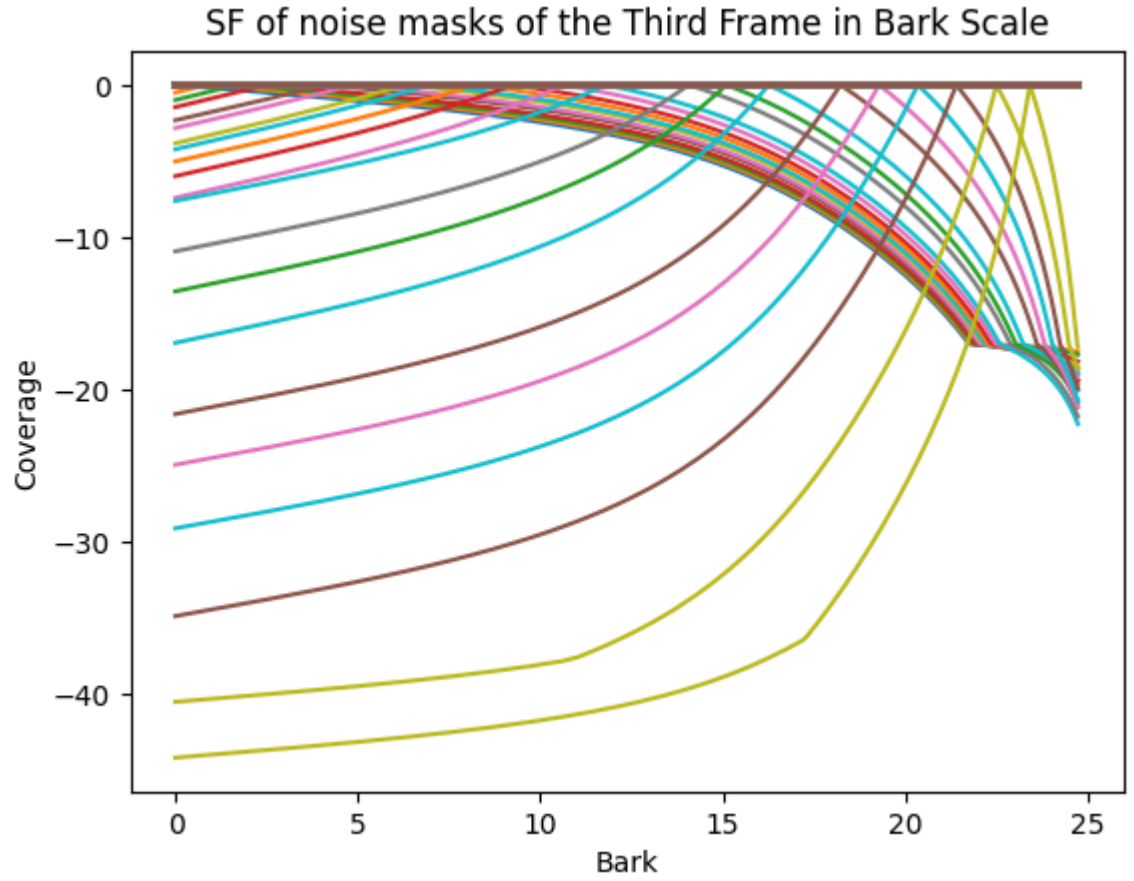
Βήμα 1.4: Υπολογισμός των δύο κατωφλίων κάλυψης (Individual Masking Thresholds)

Αρχικά αναθεωρούμε σε περίπτωση που χρειαστεί την συνάρτηση απόλυτου κατωφλίου ακοής, δίνοντας μία ευελιξία στο πλαίσιο των τιμών των συχνοτήτων. Έπειτα, με αυστηρή μαθηματική ακρίβεια, ορίζουμε την **spreading function** , η οποία προσεγγίζει το ελάχιστο επίπεδο ισχύος το οποίο πρέπει να έχουν οι γειτονικές συχνότητες έτσι ώστε να γίνουν αντιληπτές από τον άνθρωπο. Η SF(i, j) υπολογίζεται για κάθε μάσκα τόνου, ενώ οι θέσεις j μπορούν να βρεθούν ως j : PTM(j) > 0. Στο συγκεκριμένο μοντέλο θεωρούμε πως το κατώφλι TTM ορίζεται σε ένα διάστημα γειτονιάς των 12-Bark της μάσκας στο σημείο j, δηλαδή στις θέσεις i : b(i) ∈ [b(j) − 3, b(j) + 8]. Το Δb = b(i)−b(j) είναι η διαφορά των συχνοτήτων σε κλίμακα Bark μεταξύ της θέσης j της μάσκας και του κάθε σημείου i της γειτονιάς. Εδώ, για το **τρίτο πλαίσιο (frame) του ακουστικού σήματος**, παίρνουμε τις τιμές των τονικών και θορυβικών μασκών από τους πίνακες που έχουμε φορτώσει (**PTMc** και **PNMc**) και υπολογίζουμε τη συνάρτηση εξάπλωσης κάλυψης (spreading function - SF) και για τις δύο κατηγορίες μασκών. Στη συνέχεια, μετατρέπουμε τις συχνότητες σε ψυχοακουστική κλίμακα Bark, που αντιστοιχεί στον τρόπο που το ανθρώπινο αυτί αντιλαμβάνεται τις συχνότητες. Τέλος, σχεδιάζουμε δύο γραφήματα που δείχνουν πώς "εκτείνονται" (καλύπτουν) οι τονικές και οι θορυβικές μάσκες σε αυτή την κλίμακα, ώστε να οπτικοποιήσουμε την επίδραση τους στο ακουστικό φάσμα του συγκεκριμένου frame.

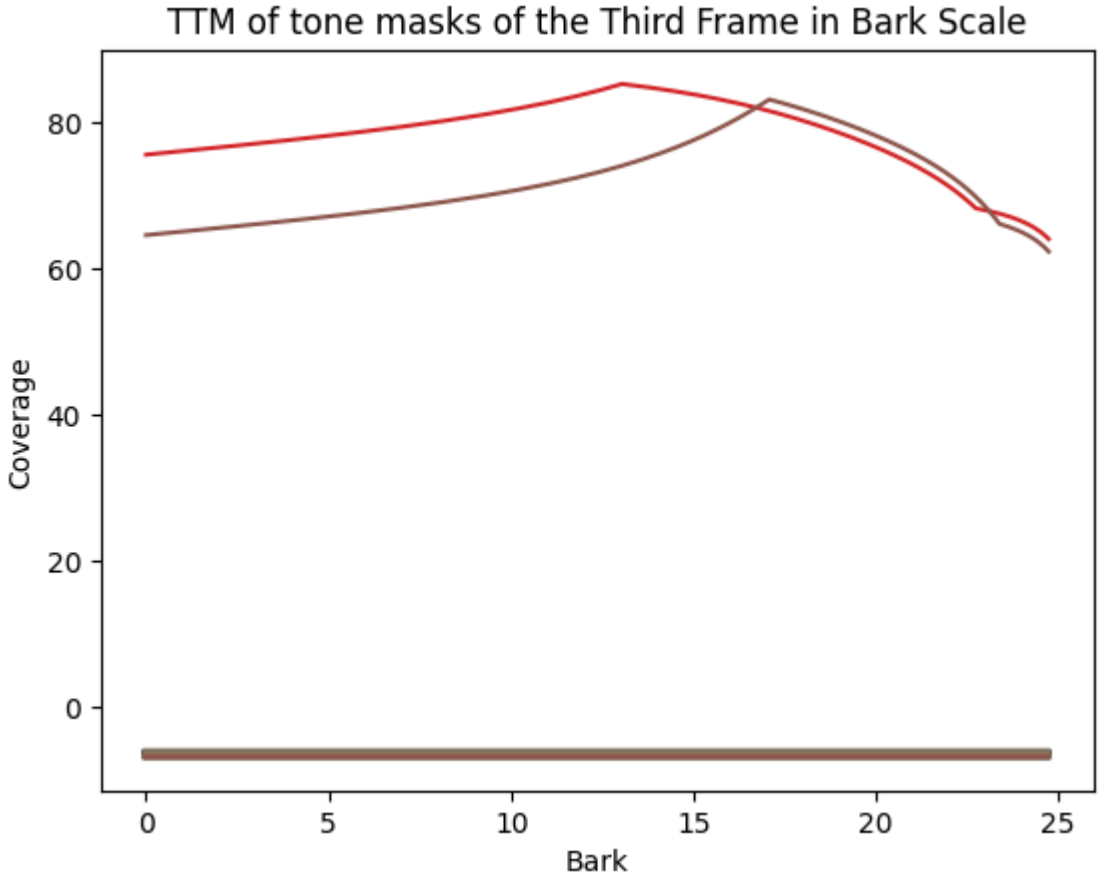
Οι τόνοι :

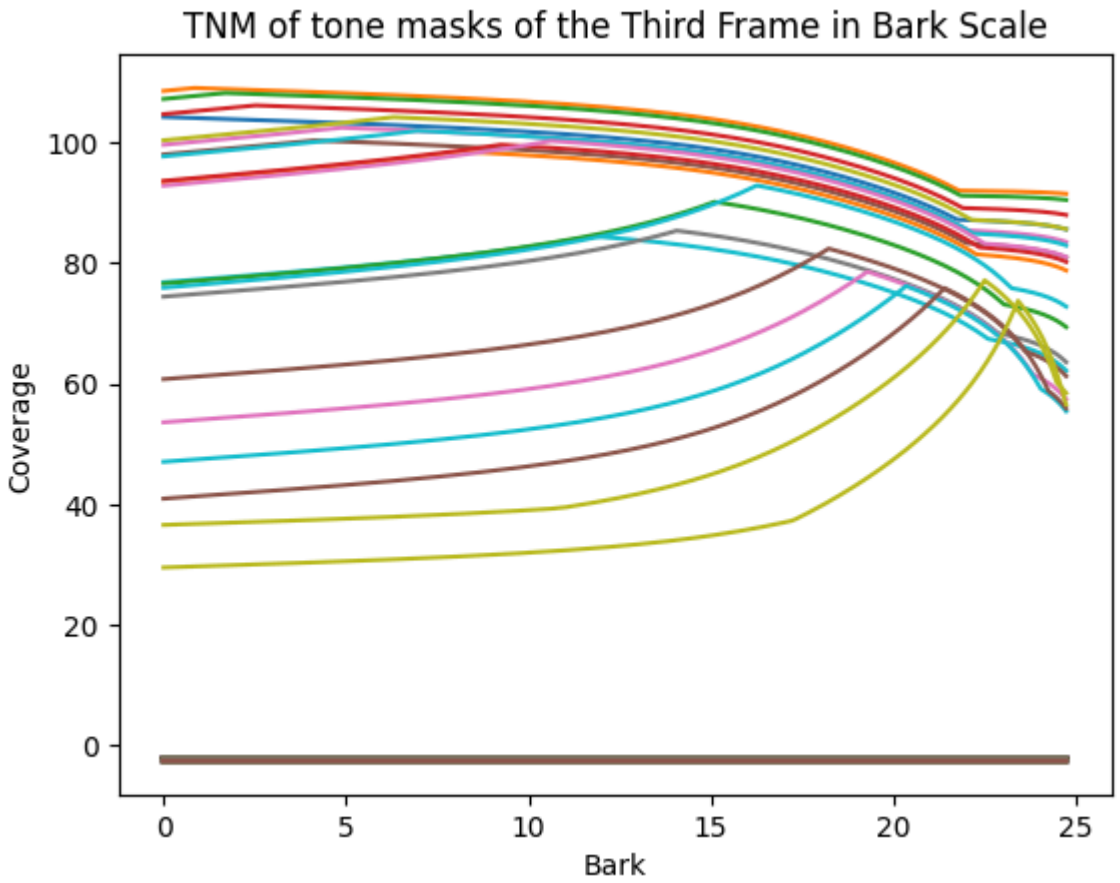


Οι θορυβικές μάσκες :



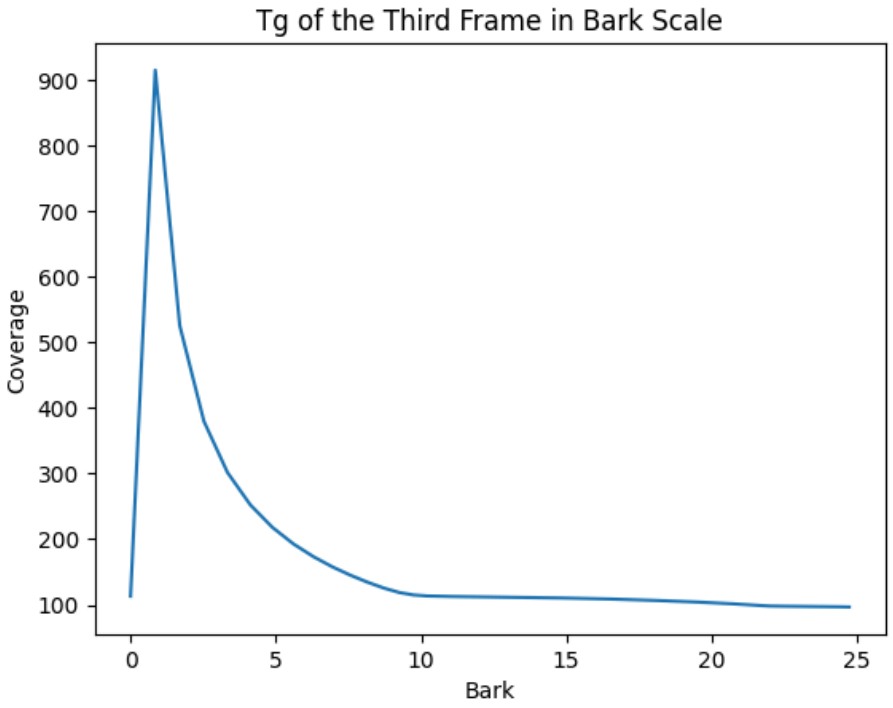
Στη συνέχεια υπολογίζουμε το κατώφλι κάλυψης που λέγεται TTM που συνδυάζει την τονική μάσκα με μια παράμετρο που εξαρτάται από την κλίμακα Bark και το spreading function, που περιγράφει κάλυψη στο χρόνο και συχνότητα. Για κάθε συνδυασμό συχνοτήτων, εφαρμόζουμε τον δοσμένο μαθηματικό τύπο που μειώνει ή αυξάνει την τιμή της μάσκας, λαμβάνοντας υπόψη τη σχετική θέση στη Bark κλίμακα και άλλους παράγοντες. Στη συνέχεια, μετατρέπουμε τις συχνότητες σε κλίμακα Bark και σχεδιάζουμε το αποτέλεσμα σε ένα γράφημα, για να δούμε πώς η μάσκα κατανέμεται στο τρίτο πλαίσιο.
Αντίστοιχα, εδώ θεωρούμε την συνάρτηση TNM για το κατώφλι κάλυψης που αναπαριστά ποσοστό κάλυψης σε σημείο που προέρχεται από μάσκα θορύβου, και τη σχετική κάλυψη .
Για κάθε ζεύγος συχνοτήτων, εφαρμόζουμε τον δοσμένο τύπο που μειώνει ή αυξάνει την τιμή της μάσκας, λαμβάνοντας υπόψη τη θέση τους στην κλίμακα Bark και άλλους παράγοντες.
Όμοια, μετατρέπουμε τις συχνότητες στην κλίμακα Bark και εμφανίζουμε το γράφημα.





Βήμα 1.5: Υπολογισμός του συνολικού κατωφλίου κάλυψης (Global Masking Threshold)

Σε αυτό το βήμα, υπολογίζουμε το “global” όριο μάσκας (Tg) για το τρίτο πλαίσιο, συνδυάζοντας τρεις διαφορετικές συνιστώσες: το φάσμα ισχύος του σήματος (Tq), τη μάσκα τονικότητας (TTM) και τη μάσκα θορύβου (TNM). Για κάθε συχνότητα, προσθέτουμε τις ενέργειες αυτών των τριών συνιστωσών σε γραμμική κλίμακα και μετά τις μετατρέπουμε πίσω σε ντεσιμπέλ, ώστε να πάρουμε ένα συνολικό όριο που δείχνει πόσο δυνατή πρέπει να είναι μια συχνότητα για να γίνει αντιληπτή μέσα στο πλαίσιο αυτό. Τέλος, μετατρέπουμε τις συχνότητες στην κλίμακα Bark και σχεδιάζουμε το αποτέλεσμα, ώστε να δούμε τη συνολική κατανομή του ορίου μάσκας. Παρατηρούμε επίσης ότι, για υψηλές συχνότητες το κατώφλι είναι μηδενικό, ενώ για χαμηλές , υπάρχει ένα peak. Αυτό σημαίνει ότι στις υψηλες συχνότητες δεν υπάρχει κάλυψη μεταξύ γειτονικών. Άρα, στο ανθρώπινο αυτί οι χαμηλότερες συχνότητες είναι αυτές που αλληλοκαλυπτονται πιο “εύκολα” με αποτέλεσμα να δυσκολευόμαστε να τις διακρίνουμε. Παρακάτω παρουσιάζεται το ολικό κατώφλι για ένα frame, όπως ολα τα διαγράμματα, ωστόσο παραθέτουμε και τον τροπο με τον οποίο μπορούμε να το κάνουμε για όλα τα πλαίσια ανάλυσης.

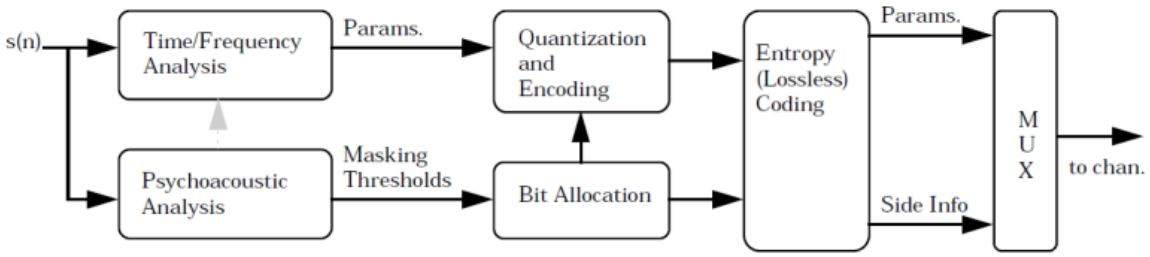



```python
# Βρίσκουμε Tg για όλα τα πλαίσια
for m in range(num_frames):
    # PTMc table
    PTMc_m = P_TMc[:, m]
    # PNMc table
    PNMc_m = P_NMc[:, m]
    # Υπολογισμός Tq
    m_frame_ps = compute_power_spectrum(frames_windowed[:,m])
    tq_m = Tq(m_frame_ps)
    # SF τονικής μάσκας
    sf_tm_m = sf(PTMc_m)
    # SF μάσκας θορύβου
    sf_nm_m = sf(PNMc_m)
    ttm_m = ttm(sf_tm_m, PTMc_m)
    tnm_m = tnm(sf_nm_m, PNMc_m)

    Tg_all[m, :] = Tg(tq_m, ttm_m, tnm_m)

    freqs_tg[m, :] = np.arange(len(Tg_all)) * (sr / N)
    # Μετατροπή συχνότητας σε Bark scale
    bark_scale_tg[m, :] = bark(freqs_tg[m,:])
```

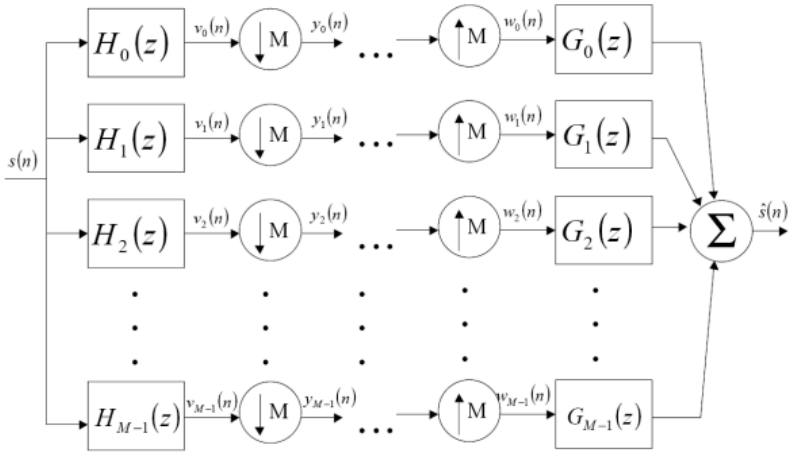
Μέρος 2. Χρονο-Συχνοτική Ανάλυση με Συστοιχία Ζωνοπερατών Φίλτρων

Σκοπός της άσκησης είναι η εξοικείωση με έννοιες όπως η **κβάντιση** και η **κωδικοποίηση** ενός ηχητικού σήματος. Για τον σκοπό αυτό θα χρησιμοποιηθούν κάποιες **συστοιχίες ζωνοπερατών φίλτρων** οι οποίες διαιρούν το φάσμα σε υποζώνες συχνοτήτων. Η χρησιμότητα αυτών φαίνεται στην ταυτοποίηση των αντιληπτικά περιττών σημείων του ηχητικού σήματος που αυτές μας παρέχουν.

Πιο συγκεκριμένα θα υλοποιηθεί η διαδικασία του παρακάτω σχήματος:



Σχήμα 2: Generic Perceptual Audio Encoder.



Σχήμα 3: Uniform M-Band Maximally Decimated Analysis-Synthesis Filterbank.

Βήμα 2.0: Συστοιχία Ζωνοπερατών Φίλτρων (Filterbank)

Ανά χρονικό πλαίσιο, χρησιμοποιούμε συστοιχίες ζωνοπερατών φίλτρων τα οποία σχεδιάζονται με βάση τον **διακριτό μετασχηματισμό συνημιτόνων (MDCT).** Χρησιμοποιήσαμε M=32 φίλτρα ανάλυσης και σύνθεσης, $h_k[n]$ και $g_k[n]$ αντίστοιχα, των οποίων οι κρουστικές αποκρίσεις φαίνονται παρακάτω:

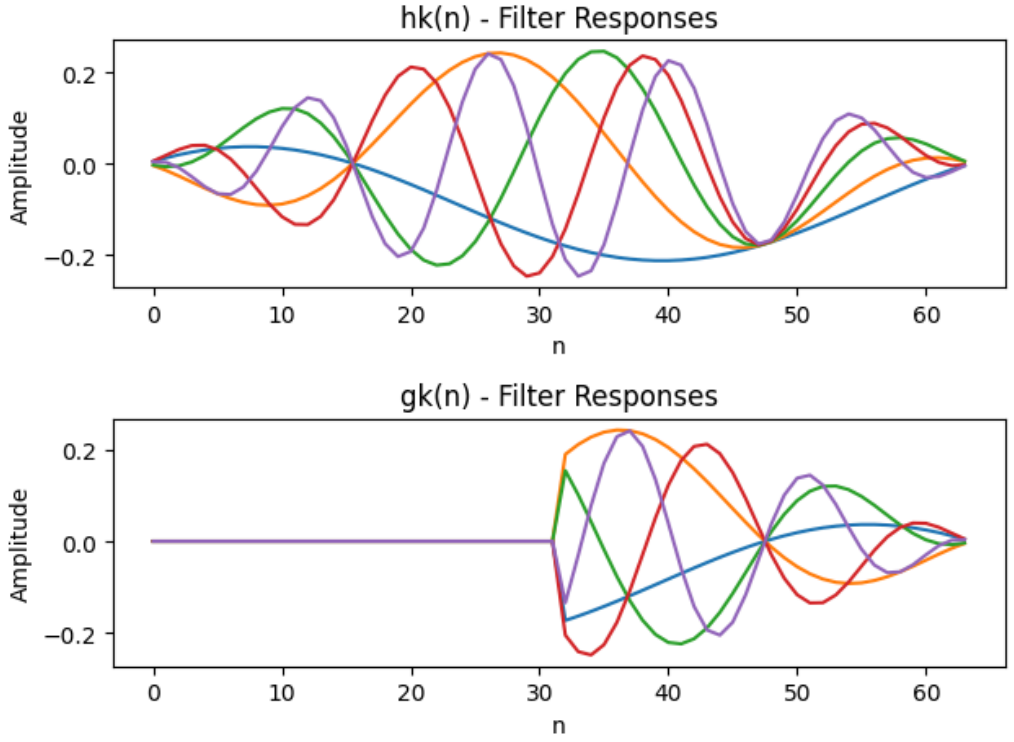
$$h_k(n) = \sin\left[\left(n + \frac{1}{2}\right)\frac{\pi}{2M}\right]\sqrt{\frac{2}{M}}\cos\left[\frac{(2n + M + 1)(2k + 1)\pi}{4M}\right] \quad (12)$$

και

$$g_k(n) = h_k(2M - 1 - n) \quad (13)$$

με μήκος L = 2M, ενώ sin[(n + 1/2) π / 2M] (για 0 ≤ n ≤ L − 1 και 0 ≤ k ≤ M − 1) ένα βαθυπερατό φίλτρο γνωστό ως "ημιτονικό παράθυρο".

Ο κώδικας κατασκευής των φίλτρων φαίνεται **στο αντίστοιχο αρχείο notebook**
Για να έχουμε μία εικόνα των κρουστικών αποκρίσεων των φίλτρων κάνουμε plot τα πρώτα πέντε φίλτρα ανάλυσης και σύνθεσης (για M=0,1,...4) τα οποία και παραθέτουμε αμέσως παρακάτω:



Βήμα 2.1: Ανάλυση με Συστοιχία Φίλτρων

Αρχικά φορτώνουμε το σήμα μουσικής μέσω της librosa.load() και κανονικοποιούμε στο [-1,1]. Έπειτα χωρίζουμε το σήμα μας σε N = 512 frames μήκους 512 μέσω της librosa.util.frame() και τέλος παραθυρώνουμε τα πλαίσια με παράθυρο hanning μήκους N = 512.

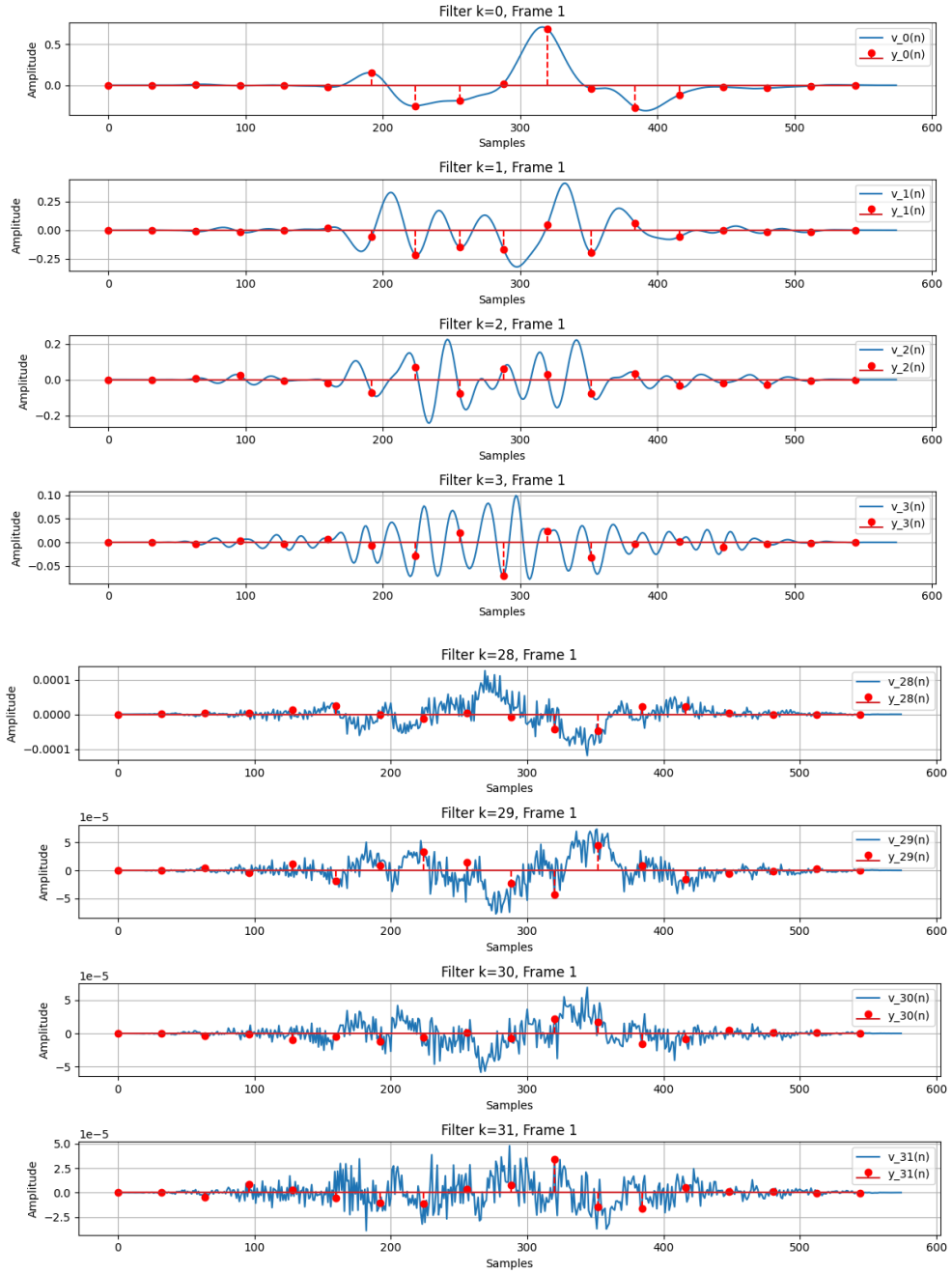
Στο αντίστοιχο αρχείο, έχει δημιουργηθεί **κώδικας** με τον οποίο **συνελίσσουμε** το **x[n]** με τα **φίλτρα ανάλυσης h_k[n]** μέσω της **np.convolve() της numpy με hanning παράθυρα**, περνάμε δηλαδή το σήμα μέσα από τα φίλτρα και παίρνουμε εξόδους ίσες με τις συνελίξεις των σήματος με την κρουστική απόκριση καθενός από τα φίλτρα. Αξίζει να σημειωθεί ότι

θεωρούμε αμελητέες τις επικαλύψεις που εισάγουν τα μη ιδανικά φίλτρα. Παρακάτω φαίνεται ο μαθηματικός υπολογισμός των συνελίξεων $u_k[n]$:

$$v_k(n) = h_k(n) * x(n) = \sum_{m=0}^{L-1} x(n-m)h_k(m), \;\; k = 0, 1, \ldots, M-1, \qquad (14)$$

Έπειτα, ακολουθεί μια **υποδειγματοληψία** κάθε ενός uk[n] **κατά παράγοντα M=32** έτσι ώστε το αρχικό σήμα να διαιρεθεί στις χρονικές συνιστώσες yk(n) = uk(Mn). Αυτό επιτυγχάνεται με τη μέθοδο **slicing** της python **(yk = vk[::M] # Decimation).**

Κάνουμε παρουσίαση τεσσάρων φίλτρων πρώτων και τελευταίων για το πρώτο frame, και έχουμε το εξής:



Βήμα 2.2: Προσαρμοζόμενος και μη Προσαρμοζόμενος Κβαντιστής

Το επόμενο στάδιο είναι η **κβαντοποίηση** των συνιστωσών yk(n). Η κβαντοποίηση αντιστοιχίζει τις αναλογικές τιμές (πραγματικούς αριθμούς) των δειγμάτων σε ένα πεπερασμένο σύνολο διακριτών επιπέδων. Στην άσκηση υλοποιούνται **δύο τύποι ομοιόμορφου κβαντιστή**:

- **Προσαρμοζόμενος κβαντιστής** με μεταβλητό αριθμό επιπέδων $\mathbf{2^{Bk}}$ ανά υποζώνη, όπου **Bk** είναι τα **bits κωδικοποίησης ανά δείγμα** της ακολουθίας yk στην υποζώνη *k* του τρέχοντος πλαισίου. Ο υπολογισμός του Bk γίνεται δυναμικά βάσει του ψυχοακουστικού μοντέλου: χρησιμοποιείται το **συνολικό κατώφλι κάλυψης** Tg(f) που υπολογίστηκε στο Μέρος 1. Συγκεκριμένα, βρίσκουμε τη χαμηλότερη στάθμη κατωφλίου **min(Tg(i))** εντός της συχνοτικής περιοχής του φίλτρου *k* (δηλ. στο Bark παράθυρο που αντιστοιχεί στη ζώνη αυτή). Συγκρίνοντας το κατώφλι με το δυναμικό εύρος του αρχικού σήματος, μπορούμε να επιλέξουμε bits έτσι ώστε ο λόγος σήματος προς θόρυβο κβαντισμού να παραμένει κάτω από το κατώφλι. Ο δοσμένος τύπος είναι:

$$B_k = \text{int}\left(\log_2 \left(\frac{R}{\min(T_g(i))} \right) - 1 \right),$$

όπου R = 2^B είναι το πλήθος διακριτών επιπέδων έντασης του αρχικού σήματος (για B=16 bits ανά δείγμα, $R = 65536$). Η τιμή **min(Tg(i))** αντιστοιχεί στο πλέον ευαίσθητο σημείο (χαμηλότερο κατώφλι σε dB SPL) της ζώνης *k*. Έτσι, αν το σήμα σε κάποια υποζώνη είναι κάτω από το κατώφλι ακρόασης, ο αλγόριθμος μπορεί να δώσει Bk=0 (δηλ. παραλείπεται πλήρως η πληροφορία σε αυτή τη ζώνη χωρίς αντιληπτή διαφορά). Αντίθετα σε ζώνες όπου το κατώφλι είναι χαμηλό (δηλ. το ανθρώπινο αυτί είναι πολύ ευαίσθητο), θα δοθούν περισσότερα bits ώστε το σφάλμα να μειωθεί. Με αυτόν τον τρόπο γίνεται **ψυχοακουστική κατανομή των bits**: τα διαθέσιμα bits κβαντισμού κατανέμονται ανά χρονικό πλαίσιο και υποζώνη με στόχο (α) το λάθος κβαντισμού να μην είναι αντιληπτό και (β) οι περισσότερο ακουστές συνιστώσες να λαμβάνουν περισσότερους πόρους κωδικοποίησης έναντι των επισκιασμένων (masked) συνιστωσών.

Επιπλέον, ο προσαρμοζόμενος κβαντιστής ρυθμίζει το **βήμα κβάντισης Δ** ανά υποζώνη, ανάλογα με το εύρος τιμών του σήματος σε κάθε πλαίσιο. Συγκεκριμένα, αν **[xmin, xmax]** είναι οι ελάχιστες/μέγιστες τιμές της ακολουθίας yk(n) στην υποζώνη *k*, τότε:

$$\Delta = \frac{x_{max} - x_{min}}{2^{B_k}}$$

δηλ. το βήμα κβάντισης υπολογίζεται έτσι ώστε οι 2^{Bk} διακριτές στάθμες να καλύπτουν πλήρως το τοπικό πεδίο τιμών. Με τη δυναμική αυτή προσαρμογή, ο κβαντιστής αξιοποιεί **με μεγαλύτερη ακρίβεια** τα επίπεδα κβάντων για κάθε υποζώνη, πετυχαίνοντας πιο αποδοτική κωδικοποίηση.

- **Μη προσαρμοζόμενος κβαντιστής** όπου ο αριθμός bits είναι *σταθερός* για όλες τις ζώνες. Εδώ ορίστηκε **Bk=8 bits** ανά δείγμα (ίδιος για κάθε k σε κάθε πλαίσιο) και σταθερό βήμα Δ που αντιστοιχεί **στο *σταθερό* πεδίο τιμών [−1, 1]** του ομαλοποιημένου σήματος. Συνεπώς, σε αυτή την περίπτωση κάθε δείγμα κωδικοποιείται ή **κβαντίζεται σε 2^8 = 256 επίπεδα** με ομοιόμορφη διακριτότητα Δ≈0.0078. Ο μη προσαρμοζόμενος κβαντιστής δεν εκμεταλλεύεται το ψυχοακουστικό κατώφλι – κατανέμει **ίσα bits σε όλες τις ζώνες**, ανεξαρτήτως του αν μια ζώνη περιέχει ακουστό ή καλυμμένο σήμα. Πρόκειται ουσιαστικά για **ομοιόμορφη κβαντοποίηση** του πλήρους φάσματος χωρίς διαφοροποίηση.

Στην πράξη, η υλοποίηση έγινε υπολογίζοντας τα Bk για κάθε υποζώνη (με βάση τα αποτελέσματα του Μέρους 1 για Tg) και στη συνέχεια κβαντίζοντας την κάθε ακολουθία yk(n) ανά δείγμα. Για την **αποκωδικοποίηση**, εκτός από τους δείκτες των επιπέδων, μεταφέρονται στον δέκτη (για κάθε υποζώνη κάθε πλαισίου) η πρώτη στάθμη και το Δ, κωδικοποιημένα σε 16 bits, ώστε να είναι εφικτή η ανακατασκευή των πραγματικών τιμών.

```python
# Βήμα 2.2: Κβαντοποίηση

# Number of coding bits per y sample
def Bk(Tg, B=16):
    R = 2**16
    Bk = int(np.log2(R / np.min(Tg)) - 1)

    return Bk


# Adaptive quantizer
def adaptive_quantizer(y, Tg):

    levels = 2**Bk(Tg,16)
    # Step of quantization
    ymin = np.min(y)
    ymax = np.max(y)
    D = (ymax - ymin) / levels

    # Quantization
    adaptive_quantizer_result = np.round(y / D)

    return adaptive_quantizer_result


# Non adaptive quantizer
def non_adaptive_quantizer(y, B=8):

    # Step of quantization
    D = 2 / (2**B) #xmax - xmin = 1 - (-1) = 2

    # Quantization
    non_adaptive_quantizer_result = np.round(y / D)

    return non_adaptive_quantizer_result
```

Βήμα 2.3: Σύνθεση

Στο βήμα αυτό, οι ακολουθίες που υπέστησαν κβαντισμό υπόκεινται **υπερδειγματοληψία** κατά **παράγοντα M**. Στην πραγματικότητα παίρνουμε το κβαντισμένο σήμα μας και κάθε 0,M,2M,... δείγματα παρεμβάλλουμε μηδενικά φτιάχνοντας έτσι το σήμα **wk[n]** (Δηλαδή το w είναι ίδιο με το σήμα στις θέσεις 0,M,2M,... και αλλού μηδέν). Έπειτα το σήμα αυτό περνά από τα φίλτρα σύνθεσης **gk[n]** που κατασκευάσαμε σε προηγούμενο ερώτημα, δηλαδή **συνελίσσεται**, και προκύπτει το **σήμα εξόδου x[n]**.Αυτό μαθηματικά περιγράφεται ως:

$$w_k(n) = \begin{cases} \hat{y}_k(n/M), & n = 0, M, 2M, 3M, \dots \\ 0, & \text{αλλιώς.} \end{cases}$$

Για την ανασύνθεση του τελικού σήματος 30s μουσικής χρησιμοποιείται η τεχνική **Overlap-Add**. Επειδή στο στάδιο ανάλυσης επιλέχθηκαν μη επικαλυπτόμενα χρονικά πλαίσια (512 δείγματα διαδοχικά), η επανένωση των πλαισίων γίνεται απλά **προσθέτοντας** τις **επικαλύψεις** από τις ουρές των φίλτρων. Στην υλοποίηση, το κάθε πλαίσιο εξόδου **x[n]** προστίθεται με overlap-add στο συνολικό σήμα **s[n]**. Με αυτή τη διαδικασία αποκαθίσταται η συνέχεια μεταξύ των πλαισίων χωρίς κενά ή διπλοϋπολογισμένες περιοχές. Το τελικό **αποσυντεθημένο σήμα s[n]** είναι πλέον στη χρονική περιοχή και μπορεί να **συγκριθεί** άμεσα με το **αρχικό σήμα μουσικής**.

Εφαρμογή στο σήμα Μουσικής

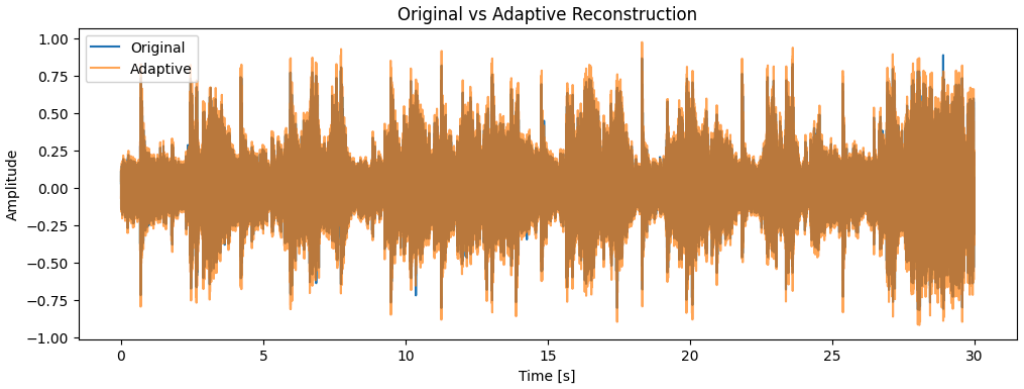
Σύγκριση Μεθόδων Κβαντοποίησης – Σφάλμα και Συμπίεση

ΔΙΕΥΚΡΙΝΗΣΗ: οι λύσεις για τα ερωτήματα 2.0 μέχρι 2.2 περιέχονται στο αρχείο "exercise2.ipynb", ενώ για το ερώτημα 2.3 στο αρχείο "exercise2_3.ipynb".

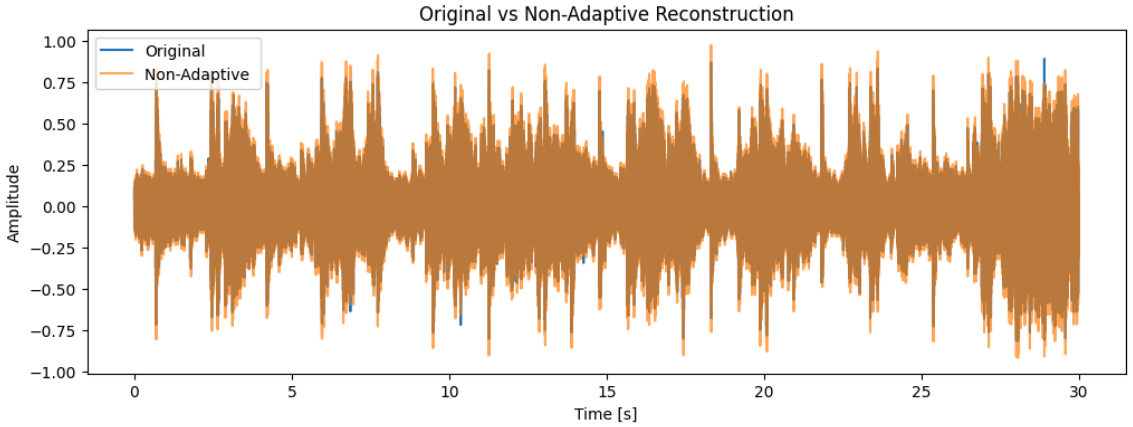
Για την εφαρμογή των παραπάνω, όπως φαίνεται και στο αρχείο κώδικα, παραθέσαμε και κάποιες χρήσιμες συναρτήσεις του μέρους ένα.

Μετά την πλήρη ανακατασκευή, το αρχικό σήμα και τα δύο ανακατασκευασμένα σήματα (από κάθε μέθοδο κβάντισης) συγκρίνονται ως προς το μέσο τετραγωνικό λάθος και την συμπίεση. Και στις δύο περιπτώσεις κβαντιστή, το ανακατασκευασμένο ηχοσήμα ακολουθεί πολύ πιστά το πρωτότυπο – οι διαφορές είναι δυσδιάκριτες στο μάτι ή το αυτί.

Σύγκριση αρχικού σήματος (μπλε) και ανακατασκευασμένου σήματος (πορτοκαλί) σε ολόκληρη τη διάρκεια:



Εδώ απεικονίζεται η περίπτωση προσαρμοζόμενης κβαντοποίησης



Εδώ απεικονίζεται η περίπτωση μη προσαρμοζόμενης κβαντοποίησης

Οι δύο κυματομορφές συμπίπτουν σχεδόν πλήρως και στις δύο περιπτώσεις, υποδεικνύοντας ότι η **παραμόρφωση** από την **κβαντοποίηση** είναι πολύ **μικρή**. Η πορτοκαλί καμπύλη καλύπτει τη μπλε στα περισσότερα σημεία, με ανεπαίσθητες αποκλίσεις μόνο τοπικά. Αποθηκεύουμε σε .wav αρχεία τα δύο σήματα και παρατηρούμε ότι ο θόρυβος στο **προσαρμοζόμενα** κβαντισμένο σήμα είναι αισθητά **λιγότερος** όπως αναμέναμε.

Για να **ποσοτικοποιήσουμε το σφάλμα**, ορίζουμε το μέσο τετραγωνικό σφάλμα (Mean Square Error, MSE) ως τη μέση τιμή του τετραγώνου της διαφοράς αρχικού και συμπιεσμένου σήματος. Ταυτόχρονα, υπολογίζουμε τα ποσοστά συμπίεσης των διαφορετικών μεθόδων. Από την υλοποίησή μας προκύπτει πως:

- Adaptive : rate=0.748, MSE=0.001995
- Fixed : rate=0.500, MSE=0.001913

Επιπλέον, τα δύο σήματα είναι αρκετά μικρότερου μεγέθους του αρχικού γεγονός που δηλώνει **επιτυχή συμπίεση του αρχείου μουσικής**.

Τέλος, απεικονίζουμε το σφάλμα των σημάτων μας σε σχέση με το αρχικό σήμα: