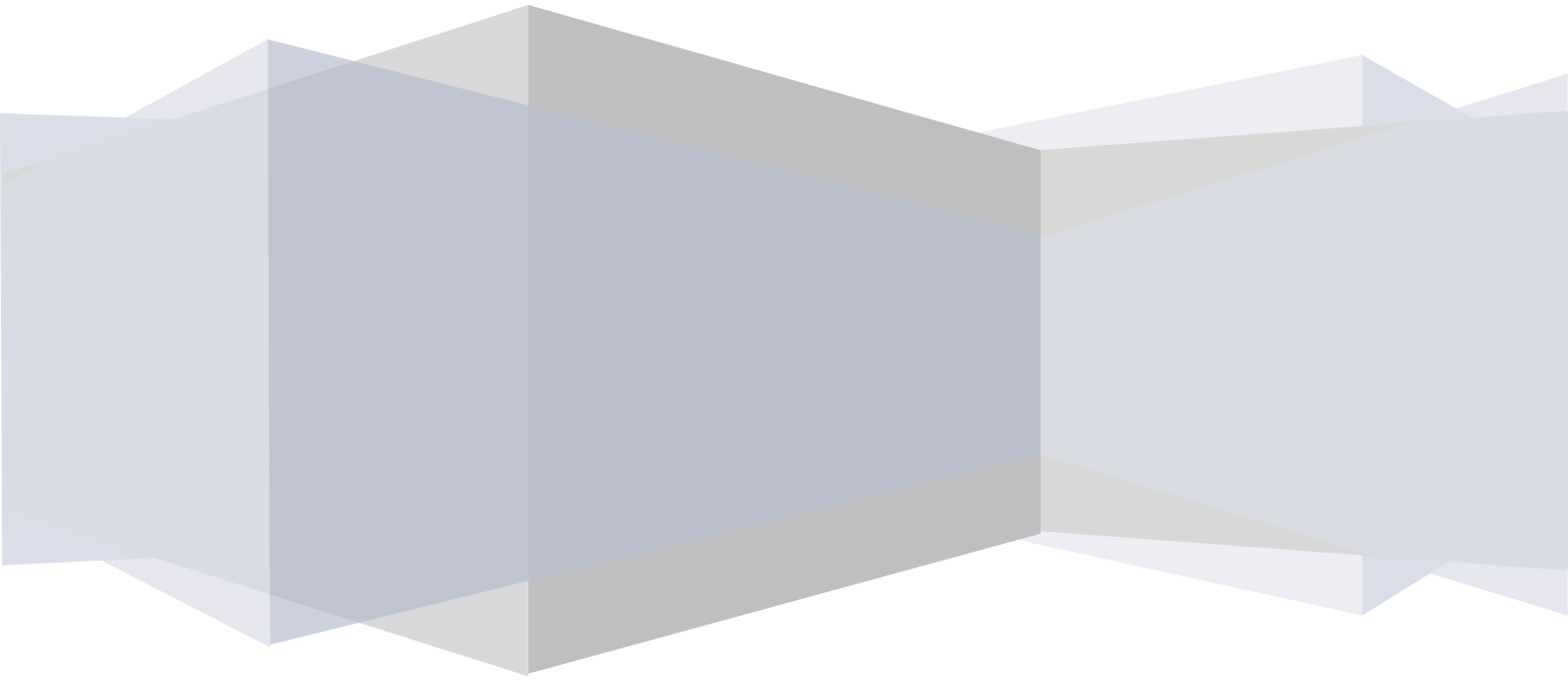
Digital Signal Processing

2η Εργαστηριακή Άσκηση: Κωδικοποίηση Μουσικής βάσει Ψυχοακουστικού Μοντέλου(Perceptual Audio Coding)

Διδυμιώτου Καουκάκη Κωνσταντίνα ΑΜ 03115726

Κυριάκου Αθηνά ΑΜ 03117405



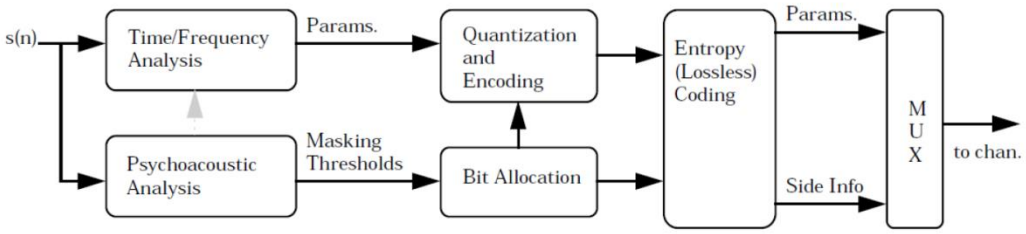
Ανάγκη για κωδικοποίηση ακουστικών σημάτων

Στις μέρες μας καθίστανται αναγκαία η αναπαράσταση, αποθήκευση και μεταφορά ψηφιακών ακουστικών σημάτων με τη χαμηλότερη δυνατή ανάγκη για πόρους την οποία συναντάμε κυρίως σε τεχνολογικές εφαρμογές όπως τα CD, αλλά και γενικότερα σε δικτυακά και τηλεπικοινωνιακά συστήματα. Σχεδόν όλοι οι σύγχρονοι αλγόριθμοι κωδικοποίησης μουσικής, οι οποίοι βασίζονται σε ιδιότητες του ανθρώπινου ακουστικού συστήματος (ψυχοακουστικό μοντέλο), έχουν στόχο να συμπιέσουν ένα αρχείο ήχου ώστε να πετύχουν όσον το δυνατόν μικρότερο χώρο αποθήκευσης αλλά και υψηλή ευκρίνεια στον ήχο.

Οι αλγόριθμοι αυτοί δίνουν έμφαση στις αντιλήψιμες συχνότητες των κρίσιμων συχνοτικών περιοχών οι οποίες ορίζονται με βάση το ψυχοακουστικό μοντέλο που θα δούμε παρακάτω. Ανάλογα με τον βαθμό συμπίεσης που θέλουμε να πετύχουμε, τα bits του κβαντιστή κατανέμονται ανά χρονικό τμήμα και κρίσιμη συχνοτική ζώνη με σκοπό να ελαχιστοποιούμε το σφάλμα κβαντισμού και να λαμβάνουμε ύστερα από την κωδικοποίηση τις χρονο-συχνοτικές συνιστώσες του σήματος που ακούγονται περισσότερο από αυτές που επικαλύπτονται και χάνονται (φαινόμενο masking effect).

Στο πρώτο μέρος της άσκησης θα υλοποιήσουμε μια συνάρτηση που θα προσομοιάζει το Ψυχοακουστικό Μοντέλο 1. Θα πραγματοποιήσουμε ψυχοακουστική ανάλυση και θα βρούμε τις τονικές μάσκες και τις μάσκες θορύβου (maskers) πριν και μετά την μείωση αλλά και το συνολικό κατώφλι κάλυψης Tg.

Στο δεύτερο μέρος θα χρησιμοποιήσουμε χρονο-συχνοτική ανάλυση με χρήση συστοιχιών ζωνοδιαβατών φίλτρων για την εξαγωγή παραμέτρων χρήσιμων για την κβαντοποίηση και κωδικοποίηση του ηχητικού σήματος.



Σχήμα 1: Generic Perceptual Audio Encoder

Χρήσιμες Έννοιες

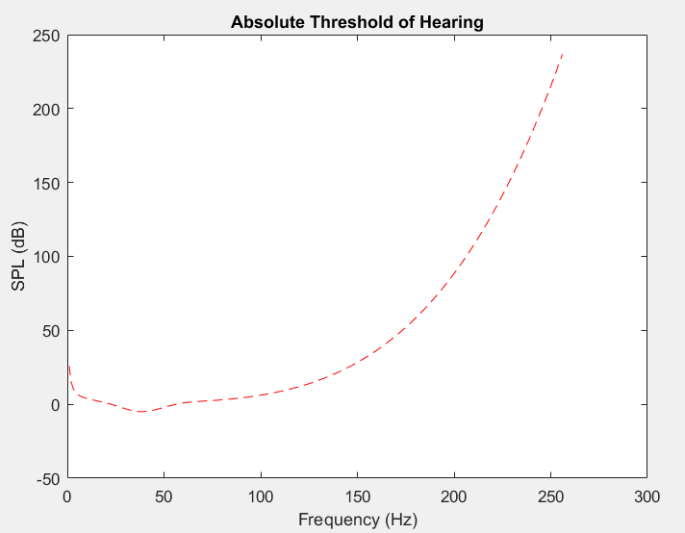
- ***Absolute Threshold of Hearing***

Το Απόλυτο Κατώφλι Ακουστότητας (ΑΚΑ) υποδηλώνει συναρτήσει της συχνότητας, τη στάθμη της ηχητικής πίεσης (SPL σε dB) ενός θορύβου στενής ζώνης ή ενός τονικού σήματος η οποία γίνεται ακουστή σε περιβάλλον πλήρους ησυχίας.

Το εύρος συχνοτήτων που μπορεί να ακούσει το ανθρώπινο αυτί κυμαίνεται από 20Hz μέχρι 20 kHZ, με ανάλυση 2Hz. Αυτό πρακτικά σημαίνει ότι συχνοτικές αλλαγές μικρότερες από 2Hz, σε ένα ακουστικό σήμα, δε θα γίνονται αντιληπτές από τον άνθρωπο. Σύμφωνα με πειράματα το αυτί μας εμφανίζει μέγιστη ευαισθησία σε τιμές 1kHz - 5kHz. Η μη γραμμική συνάρτηση του κατωφλίου είναι η εξής:

$$\mathrm{T}_q(\mathrm{f}) = 3.64(\frac{\mathrm{f}}{1000})^{-0.8} - 6.5e^{-0.6(\frac{f}{1000}-3.3)^2} + 10^{-3}(\frac{f}{1000})^4$$

Και η καμπύλη του ΑΚΑ είναι η εξής



Σχήμα 1α: Absolute Threshold of Hearing

Σε εφαρμογές συμπίεσης το κατώφλι αυτό θα μπορούσε να ερμηνευθεί ως το μέγιστο επιτρεπόμενο ποσό ενέργειας για την κωδικοποίηση των παραμορφώσεων που εισάγονται στο πεδίο της συχνότητας.

- ***Critical Bands***

Η βασική μεμβράνη του αυτιού λειτουργεί σαν αναλυτής φάσματος του σήματος, διαιρώντας το σε διαφορετικές ζώνες συχνότητας. Αυτό συμβαίνει γιατί σημεία της μεμβράνης ταλαντώνονται και συντονίζονται σε διαφορετικές συχνότητες ανάλογα

με το πάχος τους. Οι ζώνες συχνοτήτων λοιπόν ονομάζονται κρίσιμες ζώνες και παίζουν βασικό ρόλο στο masking effect που θα δούμε παρακάτω. Η κάθε κρίσιμη ζώνη αντιστοιχεί σε μια σταθερή περιοχή στη βασική μεμβράνη του αυτιού και σε μια αντίστοιχη περιοχή συχνοτήτων από την οποία διεγείρεται. Για τον άνθρωπο υπάρχουν 25 κρίσιμες ζώνες που καλύπτουν συχνότητες έως και 24kHz. Η κλίμακα που συσχετίζει τη θέση του σημείου που ταλαντώνεται πάνω στη βασική μεμβράνη με τη συχνότητα, λέγεται κλίμακα Bark. Για να μετατρέψουμε μια συχνότητα f από Hz στην κλίμακα Bark χρησιμοποιούμε την σχέση:

$$b(f) = 13\arctan(0.00076f) + 3.5\arctan[\left(\frac{f}{7500}\right)^2]$$

Η κλίμακα Bark είναι πιο κατάλληλη για την αναπαράσταση των διαφορετικών περιοχών του συστήματος ακοής, από την κλίμακα Hz, διότι αντιπροσωπεύει ακριβείς θέσεις στη βασική μεμβράνη.

- ***<u>Παραθυριοποίηση</u>***

Η παρούσα άσκηση υλοποιεί το Layer 1 του MPEG-1 ή αλλιώς MP1. Σύμφωνα με τα πρότυπα του MPEG-1 η ανάλυση πρέπει να γίνεται σε πλαίσια (frames) ανάλυσης του αρχικού σήματος. Το μήκος των παραθύρων ισούται με L=512 δείγματα. Η επεξεργασία με το Ψυχοακουστικό Μοντέλο και τη Συστοιχία Φίλτρων εκτελείται σε κάθε πλαίσιο ανάλυσης χωρίς επικάλυψη. Ως παράθυρο θα χρησιμοποιήσουμε παράθυρο Hanning όπως θα δούμε και παρακάτω.

Μέρος 1ο - Ψυχοακουστικό Μοντέλο 1

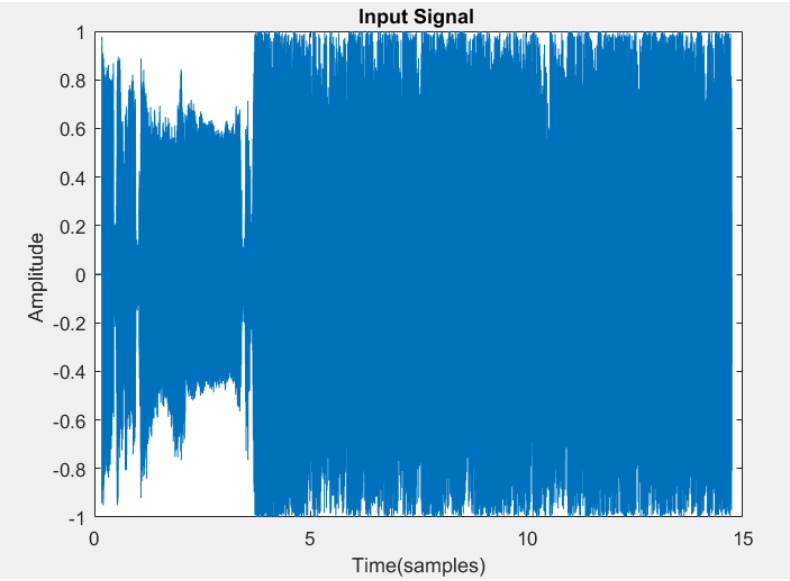
Σκοπός της άσκησης αυτής είναι η δημιουργία μιας συνάρτησης **T_G** η οποία θα παίρνει ως είσοδο το **συνολικό σήμα σε παραθυροποιημένα πλαίσια ανάλυσης** και επιστρέφει το συνολικό κατώφλι κάλυψης T_g . Ο κώδικας της T_G βρίσκεται στο αρχείο **T_G.m**. Η κλήση της συνάρτησης και η διαδικασία μέχρι την παραθυριοποίηση του αρχικού σήματος βρίσκεται στο αρχείο **ex1_main.m**.

Βήμα 1.0

Διαβάζουμε το αρχείο music_0.wav με την συνάρτηση **audioread()** της Matlab με συχνότητα δειγματοληψίας fs=44,1 kHz και το κανονικοποιούμε στις τιμές [-1,1].

```
%read the signal with audioread
fs=44100;
[s,fs]=audioread('music_0.wav');
s=s(:,1); %keeping only the one channel


%normalization of the signal s
s_normalized=s/max(abs(s));

%plotting input normalized signal
figure(2);
t=[0:length(s_normalized)-1]/fs; %time
plot(t,s_normalized);
title(['Input Signal ']);
xlabel('Time(samples)');
ylabel('Amplitude ');
```

Το ακουστικό σήμα προς ανάλυση είναι το εξής:



Σχήμα 1.0α: Ακουστικό Σήμα Εισόδου

Στην συνέχεια για την παραθυριοποίηση θα εφαρμόσουμε παράθυρο Hanning μήκους L=512 δειγμάτων το οποίο ορίζεται ως :

$$w(n) = \frac{1}{2}[1 - \cos\left(\frac{2\pi n}{N}\right)]$$

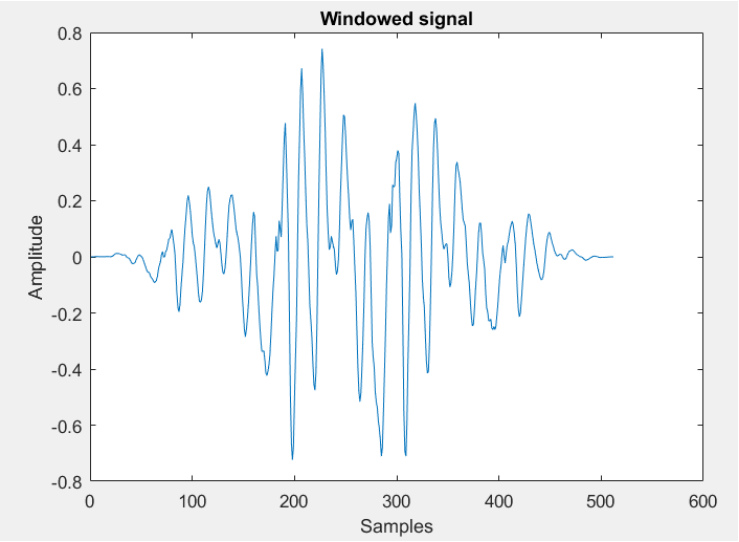
Χρησιμοποιείται η συνάρτηση **hanning()** της Matlab.

Η παραθυριοποίηση θα εκτελεστεί για όλα τα πλαίσια ανάλυσης 512 δειγμάτων που προέκυψαν από τη χρήση της συνάρτησης **buffer()** του Matlab, όπως φαίνεται στον παρακάτω κώδικα.

Στη συνέχεια, καλείται η συνάρτηση T_G λαμβάνοντας ως όρισμα το συνολικό παραθυρωμένο σήμα σε πλαίσια. Το αποτέλεσμα από την κλήση αποθηκεύεται στον πίνακα Tg.

```
%windowing of the signal
w=hanning(L); %hanning window used for windowing
[rown,coln]=size(s_framed);
for i=1:coln
    s_windowed(:,i)=s_framed(:,i).*w; %windowed signal
end

%plotting windowed
figure(2);
n=[1:L];
plot(n,s_windowed(:,frame));
title(['Windowed signal']);
xlabel('Samples');
ylabel('Amplitude');

%calculation of the total threshold Tg
Tg=T_G(s_windowed,L,frame,Tq,b,fs);
```

Το παραθυριοποιημένο σήμα που παίρνουμε είναι το εξής:



Σχήμα 1.06: Παραθυροποιημένο Σήμα

Παρακάτω πραγματοποιείται ανάλυση ανά παράθυρο για όλα τα παράθυρα και για την παρουσίαση των αποτελεσμάτων μας (image plots) χρησιμοποιούμε το 100ο πλαίσιο (δηλαδή **frame=100**).

Βήμα 1.1: Φασματική Ανάλυση

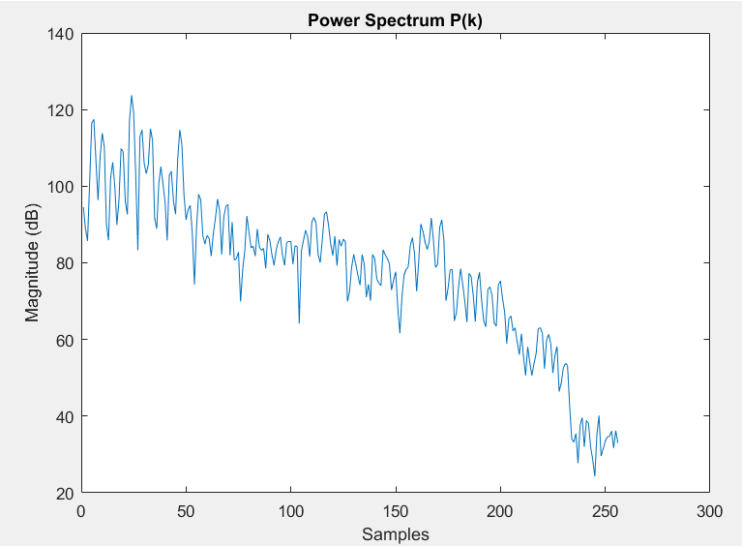
Σύμφωνα με το πρότυπο MPEG Layer-1 το φάσμα ισχύος εκφρασμένο σε SPL μονάδες δίνεται από την σχέση:

$$P(k) = PN + 10\log_{10}\left|\sum_{n=0}^{N-1} w(n)x(n)e^{-j\frac{2\pi kn}{N}}\right|^2, 0 \le k \le \frac{N}{2}.$$

Όπου το εσωτερικό στο άθροισμα είναι ο DFT του παραθυροποιημένου σήματος που υπολογίσαμε παραπάνω. Έτσι χρησιμοποιώντας την συνάρτηση **fft()** του Matlab δημιουργούμε το φάσμα ισχύος για $k\epsilon[0,\frac{L}{2}]$, όπου L το μήκος παραθύρου.

```
function[Tg]=T_G(s_windowed,L,frame,Tq,b,fs)

    %----------ex 1.1 estimation of spectrum (SPL)------------------
ssym_fft=fft(s_windowed);
s_fft=ssym_fft(1:L/2,:);
PN=90.302;
s_P=PN+10.*log10((abs(s_fft)).^2);
%s_P=s_P(1:L/2,:);

%plotting s_P
figure(3);
n=[1:L/2];
plot(n,s_P(:,4));
title(['Power Spectrum']);
xlabel('Samples');
ylabel('Amplitude');
```

Το φάσμα ισχύος που υπολογίσαμε για το 100ο πλαίσιο ανάλυσης :



Σχήμα 1.1: Φάσμα ισχύος του σήματος σε dB

Ορίζουμε στην main μας την κλίμακα Bark και το κατώφλι ακοής

```
f=[1:L/2]*(fs/L); %frequency
b=13.*atan(0.00076.*f)+3.5.*atan((f/7500).^2); %bark scale
Tq=3.64.*(f/1000).^(-0.8)-6.5.*exp(-0.6.*((f/1000)-3.3).^2)+10.^(-3).*(f/1000).^4; %Absolute Threshold of Hearing
```

Βήμα 1.2: Εντοπισμός μασκών τόνων και θορύβου (Maskers)

Το φαινόμενο Masking effect εμφανίζεται όταν δρουν παράλληλα δύο ήχοι κοντινών συχνοτήτων διαφορετικών ηχητικών πιέσεων, με αποτέλεσμα ο δυνατότερος ήχος (masker) να επικαλύπτει τον πιο αδύναμο (maskee). Ανάλογα με τη συχνότητα, μπορεί να οριστεί ένα ακουστικό κατώφλι επικάλυψης (Auditory Masking Threshold, ATM) και ήχοι που βρίσκονται κάτω από αυτό, όπως παραμορφώσεις και θόρυβος, δε θα γίνονται αντιληπτοί. Αν δεν υπάρχει masker τότε ο ήχος θα είναι αντιληπτός στο αυτί μόνο αν η ηχητική του πίεση είναι υψηλότερη από το απόλυτο κατώφλι ακουστότητας (AKA) T_q.

Το Ψυχοακουστικο Μοντέλο 1 βρίσκει τα τοπικά μέγιστα τα οποία είναι μεγαλύτερα από τις γειτονικές συχνότητες κατά 7dB.

Το εύρος γειτονιάς Δκ διαφέρει ανά διακριτική ικανότητα k όπως παρακάτω:

$$\Delta_k \in \begin{cases} 2, & 2 < k < 63 \quad (0.17 - 5.5\text{kHz}) \\ [2,3] & 63 \leq k < 127 \quad (5.5 - 11\text{kHz}) \\ [2,6] & 127 \leq k \leq 250 \quad (11 - 20\text{kHz}) \end{cases}$$

Έτσι ορίσαμε την συνάρτηση Dk:

```
function[DK]=dk(k)
  if (k>2) && (k<63)
        DK=2;
    elseif(k>=63) && (k<127)
        DK=23;
    elseif (k>=127) && (k<=250)
        DK=23456;
    end
    %l=length(DK);
end
```

Η συνάρτηση S_T θα επιστρέφει 1 αν στην θέση k υπάρχει μάσκα και 0 αν δεν υπάρχει σύμφωνα με αυτή την εξίσωση:

$$S_T = \begin{cases} 0, & \text{αν } k \notin [3,250] \\ P(k) > P(k \pm 1) \bigwedge P(k) > P(k \pm \Delta_k) + 7\text{dB}), & \text{αν } k \in [3,250] \end{cases}$$

την οποία και υλοποιήσαμε:

```
function[ST]=St(P_matrix)
    [rown,coln]=size(P_matrix);
    for i=1:coln %for each frame of the signal
        for k=1:256
            if (k>=3 && k<=250)
                DK=dk(k);
                if(DK==2)
                    ST(k,i)=((P_matrix(k,i)>P_matrix(k+1,i)) && (P_matrix(k,i)>P_matrix(k-1,i)) && (P_matrix(k,i)
                elseif(DK==23)
                    ST(k,i)=((P_matrix(k,i)>P_matrix(k+1,i)) && (P_matrix(k,i)>P_matrix(k 1,i)) && (P_matrix(k,i)
                elseif(DK==23456)
                    ST(k,i)=((P_matrix(k,i)>P_matrix(k+1,i)) && (P_matrix(k,i)>P_matrix(k-1,i)) && (P_matrix(k,i)
                end
            else
                ST(k,i)=0;
            end
        end
    end
end
```

Έτσι τώρα μπορούμε να υπολογίσουμε την ισχύ των τονικών μασκών με βάση την σχέση:

$$P_{TM}(k) = \begin{cases} 10\log_{10}(10^{0.1(P(k-1))} + 10^{0.1(P(k))} + 10^{0.1(P(k+1))})(\text{dB}), & \text{αν } S_T(k) = 1 \\ 0, & \text{αν } S_T(k) = 0 \end{cases}$$

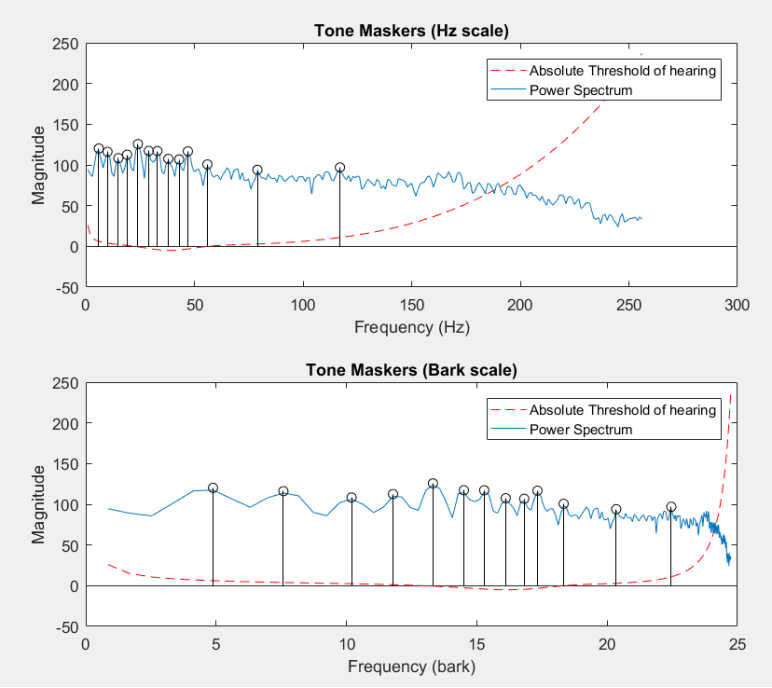
Δημιουργούμε έτσι το πίνακα P_TM:

```
%----------ex 1.2 masks------------------
s_ST=St(s_P); %binary matrix of masks
[rown,coln]=size(s_ST);
for i=1:coln
        for k=1:256
            if (s_ST(k,i)==1)
                P_TM(k,i)=10*log10(10.^(0.1.*s_P(k-1,i))+10.^(0.1.*s_P(k,i))+10.^(0.1.*s_P(k+1,i)));
            else
                P_TM(k,i)=0;
            end
            ks(k,i)=P_TM(k,i);
        end
end

for i=1:L/2
        ks_frame(i)=ks(i,frame);    %k potition of a tone mask for a specific frame
end

ks_frame=ks_frame';
ks_frame(ks_frame==0)=NaN;  %plot non zero samples
```

Και για το πλοτάρισμα των tone maskers έχουμε:

```
%plotting Tone Maskers Frequency(Hz)
figure(5);
subplot(2,1,1);
p1=plot(Tq,'r--');
hold on
p2=plot(s_P(:,frame));
hold on
stem(ks_frame,'o','k')
hold off
title(['Tone Maskers (Hz scale)']);
xlabel('Frequency (Hz)');
ylabel('Magnitude');
legend([p1,p2],{'Absolute Threshold of hearing','Power Spectrum'});
%plotting Tone Maskers Frequency(bark)
subplot(2,1,2);
p1=plot(b,Tq,'r--');
hold on
p2=plot(b,s_P(:,frame));
hold on
stem(b,ks_frame,'o','k')
hold off
title(['Tone Maskers (Bark scale)']);
xlabel('Frequency (bark)');
ylabel('Magnitude');
legend([p1,p2],{'Absolute Threshold of hearing','Power Spectrum'});
```

Όταν θα τρέξουμε την main βλέπουμε ότι έχουμε 13 τονικές μάσκες. Στο διάγραμμα εμφανίζεται τόσο το φάσμα ισχύος και το κατώφλι ακοής όσο και οι τονικές μάσκες, συναρτήσει της συχνότητας σε Hz και σε Bark.

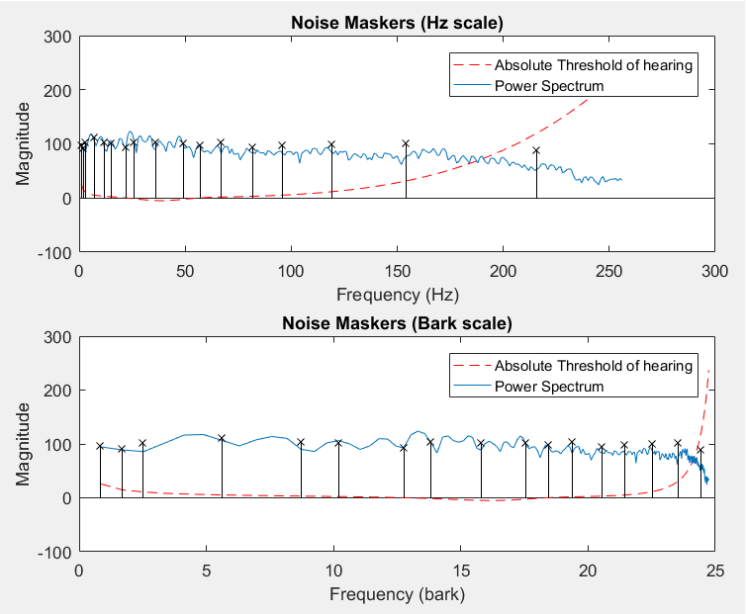


Σχήμα 1.2α: Tone Maskers (σε Hz και Bark) για το 100ο πλαίσιο

Για τις Noise Maskers χρησιμοποιούμε την έτοιμη συνάρτηση **findNoiseMaskers()** για κάθε frame και έχουμε τον πίνακα P_NM:

```
%Calculate Noise Maskers
for i=1:coln
    P_NM(:,i)=findNoiseMaskers(s_P(:,i),P_TM(:,i),b);
end
```

Παρατηρούμε ότι έχουμε 17 noise maskers. Ομοίως με πριν έχουμε το εξής γράφημα:



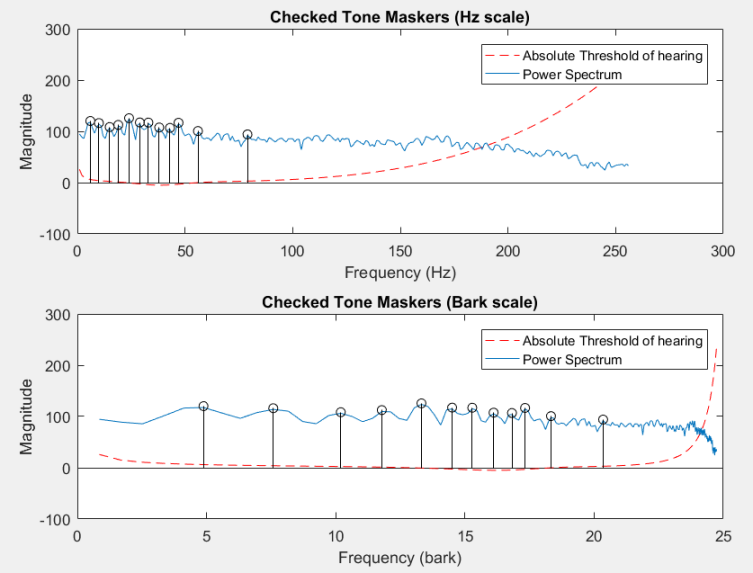
Σχήμα 1.2β: Noise Maskers (σε Hz και Bark) για το 100ο πλαίσιο

Βήμα 1.3: Μείωση και αναδιοργάνωση των μασκών

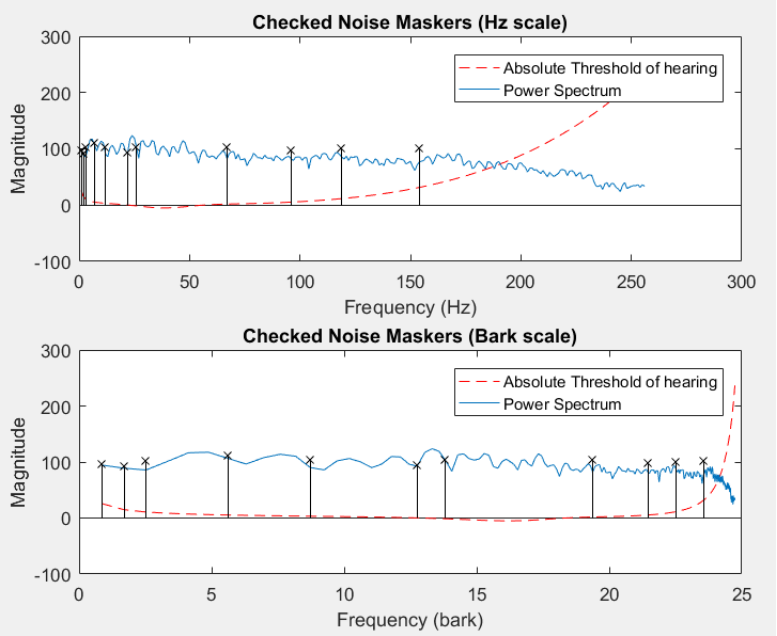
Σε αυτό το βήμα μειώνουμε τον αριθμό μασκών. Το Ψυχοακουστικό Μοντέλο 1 αρχικά προσδιορίζει αν τονικά και μη τονικά στοιχεία έχουν πλάτος μεγαλύτερο του κατωφλίου ακοής, και κρατάει αυτά που ακούγονται. Στην συνέχεια όταν δύο τονικά στοιχεία βρίσκονται σε μια κρίσιμη ζώνη σε απόσταση μεταξύ τους μικρότερη από 0.5 Bark τότε διατηρείται το στοιχείο με την μεγαλύτερη τιμή ισχύος και το άλλο διαγράφεται από τον πίνακα τονικών στοιχείων. Έτσι θα έχουμε δύο νέους πίνακες P_TMnew και P_NMnew οι οποίοι δημιουργήθηκαν με την έτοιμη συνάρτηση ***[P_TMnew , P_NMnew] = checkMaskers(P_TM,P_NM,Tq,b)*** για κάθε frame.

```
%----------ex 1.3 Reduction of masks-----------------
for i=1:coln
    [P_TMnew(:,i),P_NMnew(:,i)]=checkMaskers(P_TM(:,i)',P_NM(:,i)',Tq,b)
end

for i=1:coln
        for k=1:256
            ktm(k,i)=P_TMnew(k,i);
            knm(k,i)=P_NMnew(k,i);
        end
end

for i=1:L/2
        ktm_frame(i)=ktm(i,frame);
        knm_frame(i)=knm(i,frame);
end

ktm_frame(ktm_frame==0)=NaN;
knm_frame(knm_frame==0)=NaN;
```

Με παρόμοιο τρόπο εμφανίζουμε τα image plots και παρατηρούμε ότι πλέον έχουμε 12 tone maskers (δηλαδή οι τονικές μάσκες από 13 μειώθηκαν στις 12) και 11 noise maskers (οι μάσκες θορύβου μειώθηκαν από 17 μειώθηκαν στις 11).



Σχήμα 1.3α: Checked Tone Maskers (Hz και Bark) για το 100ο Πλαίσιο



Σχημα 1.3β: Checked Noise Maskers (Hz και Bark) για το 100ο πλαίσιο

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

Μετά την μείωση του αριθμού των μασκών (Βήμα 1.3), υπολογίζουμε τα δύο διαφορετικά κατώφλια κάλυψης T_TM και T_NM. Το κάθε κατώφλι αντιπροσωπεύει το ποσοστό κάλυψης στο σημείο i το οποίο προέρχεται από την μάσκα τόνου ή θορύβου στο σημείο j.

$$T_{TM}(i,j) = P_{TM}(j) - 0.275b(j) + SF(i,j) - 6.025(\text{dB SPL})$$

$$T_{NM}(i,j) = P_{NM}(j) - 0.175b(j) + SF(i,j) - 2.025(\text{dB SPL})$$

Αρχικά ορίζουμε την συνάρτηση SF(i,j), που υπολογίζεται για κάθε μάσκα κάθε πλαισίου, η οποία προσεγγίζει το ελάχιστο επίπεδο το οποίο πρέπει να έχουν οι γειτονικές συχνότητες ώστε να είναι αντιληπτές στο ανθρώπινο αυτί. Το i ορίζεται σε μια γειτονιά 12 bark με κέντρο το j δηλαδή στο διάστημα [b(j)-3,b(j)+8]. Αν το i δεν ανήκει στο περιθώριο των 12 bark σε σχέση με τη συχνότητα j όπου και βρίσκεται μία μάσκα, μηδενίζουμε κατευθείαν τα T_TM(i) και T_NM(i).

$$SF(i,j) = \begin{cases} 17\Delta_b - 0.4P_{TM}(j) + 11, & -3 \le \Delta_b < -1 \\ (0.4P_{TM}(j) + 6)\Delta_b, & -1 \le \Delta_b < 0 \\ -17\Delta_b, & 0 \le \Delta_b < 1 \\ (0.15P_{TM}(j) - 17)\Delta_b - 0.15P_{TM}(j), & 1 \le \Delta_b < 8 \end{cases}$$

Τα Δb στο SF(i,j) είναι η απόσταση σε barks του i από το j.

Αρχικά φτιάξαμε την συνάρτηση SF την οποία την καλούμε στις συναρτήσεις T_TM και T_NM αντίστοιχα όπως παρακάτω:

```
function[SF]=Sf(Db,Pmasker)
    if (Db>=-3 && Db<-1)
        SF=17*Db-0.4*Pmasker+11;
    elseif (Db>=-1 && Db<0)
        SF=(0.4*Pmasker + 6) * Db;
    elseif (Db>=0 && Db<1)
        SF=-17*Db;
    elseif (Db>=1 && Db<8)
        SF=(0.15*Pmasker-17)*Db-0.15*Pmasker;
    end
end
```

Σχήμα 1.4α: Συνάρτηση SF

```
function[T_TM]=T_tm(bi,bj,Ptm)
    if (bi<=bj-3 || bi>= bj+8)
        T_TM=0; %there is not a mask in 12 bark bandwidth
    else
        Db=bi-bj;
        SF=Sf(Db,Ptm);
            T_TM=Ptm-0.275*bj + SF - 6.025; % dB SPL
    end
end
```

Σχήμα 1.4β: Υλοποίηση Συνάρτησης T_TM

```
function[T_NM]=T_nm(bi,bj,Pnm)
    if (bi<=bj-3 || bi>= bj+8)
        T_NM=0;
    else
        Db=bi-bj;
        SF=Sf(Db,Pnm);
            T_NM=Pnm-0.175*bj + SF - 2.025; % dB SPL
end
```

Σχήμα 1.4γ: Υλοποίηση Συνάρτησης T_NM

Αυτές τις καλούμε στην συνάρτηση μας T_G. Αρχικά αρχικοποιούμε έναν 3διαστατο πίνακα με μηδενικά με διαστάσεις 256x(max_num_T)x1271.Υπολογίζονται τα κατώφλια T(i,j) για 1<= i <=256 και για j όπου υπάρχει masker (tone ή noise) για κάθε πλαίσιο(διάσταση 1271) . Χρησιμοποιούμε το μέγιστο αριθμό μασκών και αυτό γιατί υπολογίζουμε τα κατώφλια για κάθε πλαίσιο και το κάθε πλαίσιο έχει διαφορετικό αριθμό μασκών. Επομένως έχουμε τον κώδικα:

```
max_num_TM=0;

for num_frame=1:coln %check each frame to find the max_num of tone masks
    max_new=length(find(P_TMnew(:,num_frame)>0)); %how tone maskers we have
    if(max_new>max_num_TM)
        max_num_TM=max_new;
    end
end

T_TM=zeros(L/2,max_num_TM,coln); %make a 3d array of zeros size 256 x max_num_TM x 1271

for num_frame=1:coln %for each frame there is a threshold

    TM_places_array=find(P_TMnew(:,num_frame)>0);  %places of tones maskers in the frame

    for num_mask=1:length(TM_places_array)    %for each tone masker in the frame

        j=TM_places_array(num_mask);

        for i=1:L/2     %for each i frequency

            T_TM(i,num_mask,num_frame)=T_tm(b(i),b(j),P_TMnew(j,num_frame));

        end
    end
end
```

Σχήμα 1.4δ: Υλοποίηση πίνακα T_TM

```
max_num_NM=0;

for num_frame=1:coln %check each frame to find the max_num of noise masks
    max_new=length(find(P_NMnew(:,num_frame)>0));
    if(max_new>max_num_NM)
        max_num_NM=max_new;
    end
end

T_NM=zeros(L/2,max_num_NM,coln); %make a 3d array of zeros size 256 x max_num_NM x 1271

for num_frame=1:coln %for each frame there is a threshold

    NM_places_array=find(P_NMnew(:,num_frame)>0);  %places of masks in the frame

    for num_mask=1:length(NM_places_array)    %for each noise mask in the frame

        j=NM_places_array(num_mask);

        for i=1:L/2     %for each i frequency

            T_NM(i,num_mask,num_frame)=T_nm(b(i),b(j),P_NMnew(j,num_frame));

        end
    end
end
```

Σχήμα 1.4ε: Υλοποίηση πίνακα T_NM

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

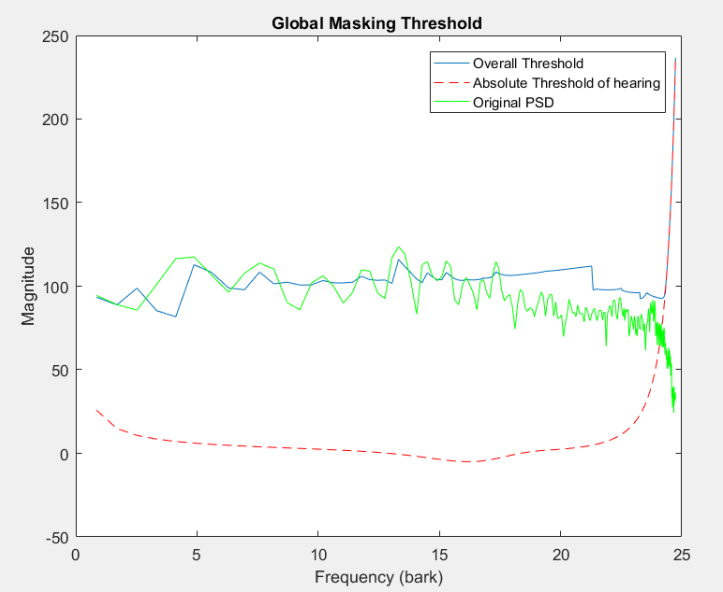
Τα ξεχωριστά αυτά κατώφλια που υπολογίστηκαν παραπάνω (βήμα 1.4) συνδυάζονται για την δημιουργία του Global Masking Threshold Tg σε κάθε διακριτική συχνότητα ξεχωριστά και τέλος το συνολικό κατώφλι για όλο το σήμα υπολογίζεται από τον τύπο:

$$T_g(i) = 10\log_{10}\left(10^{0.1T_q(i)} + \sum_{l=1}^{L} 10^{0.1T_{TM}(i,\ell)} + \sum_{m=1}^{M} 10^{0.1T_{NM}(i,m)}\right) \text{ dB SPL}$$

Έχουμε τον κώδικα:

```
for num_frame=1:coln
    for i=1:L/2
        Tg(i,num_frame)=10^(0.1.*Tq(i));
        for TTM_mask=1:max_num_TM %number of tone maskers
            Tg(i,num_frame)=Tg(i,num_frame)+10^(0.1*T_TM(i,TTM_mask,num_frame));
        end
        for TNM_mask=1:max_num_NM
            Tg(i,num_frame)=Tg(i,num_frame)+10^(0.1*T_NM(i,TNM_mask,num_frame));
        end
        Tg(i,num_frame)=10*log10(Tg(i,num_frame));
    end
end
```

Και εμφανίζουμε το συνολικό κατώφλι μαζί με το απόλυτο κατώφλι ακουστότητας και το φάσμα ισχύος για το 100ο πλαίσιο.



Σχήμα 1.5: Συνολικό Κατώφλι Κάλυψης

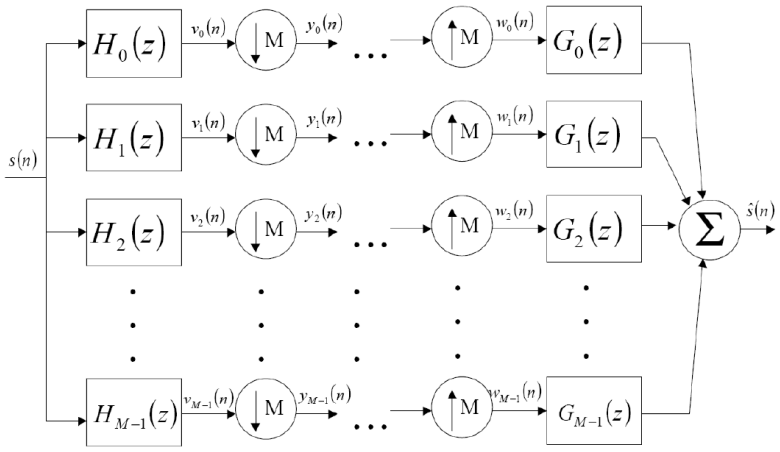
*Σε όλους τους άξονες το Magnitude είναι σε εκφρασμένο σε μονάδες SPL dB και η Frequency είτε σε Hz είτε σε Bark.

Μέρος 2ο - Χρονο-Συχνοτική Ανάλυση με Συστοιχία Ζωνοπερατών Φίλτρων

Όπως αναφέραμε και στην αρχή (Σχήμα 1) η συστοιχία ζωνοπερατών φίλτρων διευκολύνει την ανάλυση με το ψυχοακουστικό μοντέλο. Με την μέθοδο αυτή διαιρούμε το φάσμα σε υποζώνες συχνοτήτων, παρέχοντας έτσι πληροφορίες σχετικές με την συχνοτική κατανομή του σήματος, οι οποίες βοηθούν στην ταυτοποίηση των αντιληπτικά περιττών σημείων του σήματος (περιττή κβάντιση εφόσον ο ήχος είναι μη αντιληπτός). Η αποσύνθεση αυτή του σήματος στις διαφορετικές συχνοτικές περιοχές βοηθά στην μείωση των στατιστικών redundancies.

Στόχος της άσκησης είναι η μείωση των bit που απαιτούνται για την αποθήκευση του μουσικού κομματιού χωρίς σημαντική αλλοίωση του περιεχομένου του και της ποιότητάς του.

Η διαδικασία που θα ακολουθήσουμε είναι η εξής:



Σχήμα 2: Uniform M-Band Maximally Decimated Analysis-Synthesis Filterbank

Υλοποιούμε δύο συναρτήσεις για δύο είδη κβαντιστών που καλούνται από το αρχείο **ex2_main.m**.

Και οι δύο συναρτήσεις λαμβάνουν ως όρισμα το **συνολικό σήμα σε μορφή πλαισίων** όπως προκύπτει από την **audioread()** και την **buffer()** και επιστρέφουν το **συνολικό ακατασκευασμένο** και **τον αριθμό των bits που χρειάστηκαν για την κωδικοποίηση**.

Ο πρώτος είναι ένας προσαρμοζόμενος ομοιόμορφος κβαντιστής $2^{Bκ}$ επιπέδων, όπου Bκ ο αριθμός των bits κωδικοποίησης ανά δείγμα της ακολουθίας yk(n) για κάθε φίλτρο και για κάθε πλαίσιο ανάλυσης και με προσαρμοζόμενο στο πλαίσιο ανάλυσης βήμα Δ. Η συνάρτηση για την ανάλυση με αυτόν τον κβαντιστή είναι η **adaptable_quantizer.m**.

Ο δεύτερος είναι ένας μη προσαρμοζόμενος κβαντιστής με σταθερό ρυθμό bit κωδικοποίησης ανά δείγμα σε κάθε φίλτρο και πλαίσιο ανάλυσης Bk=8 και σταθερό βήμα

κβαντισμού Δ, το οποίο καθορίζεται από ένα υποτιθέμενο σταθερό πεδίο τιμών του σήματος [-1,1]. Η συνάρτηση για την ανάλυση με αυτόν τον κβαντιστή είναι η **non_adaptable_quantizer.m**.

Η ανάλυση είναι παρόμοια και για τους δύο για τα βήματα 2.0, 2.1 και 2.3.

Βήμα 2.0: Συστοιχία Ζωνοπερατών Φίλτρων (Filterbank)

Οι κρουστικές αποκρίσεις των M=32 φίλτρων που θα χρησιμοποιήσουμε δίνονται από τον τύπο:

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

$$g_k(n) = h_k(2M-1-n)$$

Ο πρώτος όρος είναι ένα βαθυπερατό φίλτρο γνωστό ως ημιτονικό παράθυρο. Επίσης στην παραπάνω σχέση έχουμε $0 \le n \le L-1$,όπου L=2M=64 μήκος φίλτρων και $0 \le k \le M-1$

Δημιουργούμε αυτές τις κρουστικές αποκρίσεις των φίλτρων h και g στο Matlab:

```
%----------ex 2.0 Defining the filters------------

%initialization of the filters (represented as matrices)
%total number of filters is M=32 and length of each filter is L=2*M=64

h_filter=zeros(2*M,M);   %analysis
g_filter=zeros(2*M,M);   %synthesis

for k=1:M %num_filter
    for n=1:2*M   %num_sample
        h_filter(n,k)=sin((n+1/2)*(pi/(2*M)))*sqrt(2/M)*cos((2*n+M+1)*(2*k+1)*pi/(4*M));
    end
end

for k=1:M %num_filter
    for n=0:2*M-1   %num_sample
        g_filter(n+1,k)=h_filter(2*M-n,k);
    end
end
```

Βήμα 2.1: Ανάλυση με Συστοιχία Φιλτρων

Η εφαρμογή των φίλτρων αυτών στο πλαισιωμένο (x_framed) σήμα μας γίνεται μέσω της εφαρμογής της συνέλιξης στο χρόνο με αυτά για κάθε frame. Από τις συνελίξεις που προκύπτουν εφαρμόζουμε decimation (αποδεκατισμό - downsamplig ανά M δείγματα), στο μέγιστο βαθμό ώστε να μην επιφέρεται επικάλυψη.

Αυτό γίνεται με τις συναρτήσεις της Matlab **conv()** και **downsample()**:

```
%------ex 2.1 Convolution and undersampling of the x with the filters------

[rown,frames]=size(x_framed);

 %convolution of each signal frame with the filters
 %for each frame, 32 filtering -> 3D matrix

for num_frame=1:frames %num_frame
    for k=1:M   %num_filter
         v_conv(:,k,num_frame)=conv(h_filter(:,k)',x_framed(:,num_frame));
    end
end

%downsampling
for num_frame=1:frames
    for k=1:M   %num_filter
        y_downsamp(:,k,num_frame)=downsample(v_conv(:,k,num_frame),M);
    end
end
```

Βήμα 2.2: Κβαντοποίηση

Αρχικά θα υλοποιήσουμε έναν **προσαρμοζόμενο (μη γραμμικό) κβαντιστή** $2^{Bκ}$ επιπέδων όπου το Bκ θα είναι ο αριθμός των bits κωδικοποίησης ανά δείγμα της ακολουθίας που προέκυψε ύστερα από τον αποδεκατισμό (Βήμα 2.1). Το βήμα Δ του κβαντιστή θα προσαρμόζεται σε κάθε πλαίσιο ανάλυσης.

$$B_k = \left\lceil \log_2 \left(\frac{R}{min(T_g(i))} - 1 \right) \right\rceil$$

Όπου το Tg το συνολικό κατώφλι κάλυψης του παραθυρωποιημένου σήματος (windowed)

Για την εύρεση των bits κωδικοποίησης Bk δεν έγινε η διαδικασία με τις κεντρικές συχνότητες που αναφέρθηκε στην εκφώνηση αλλά υλοποιήθηκε με το σπάσιμο (συνάρτηση **buffer()** του Tg σε 32 διαστήματα των 8 bits χωρίς επικάλυψη και στην συνέχεια υπολογίστηκε το min(Tg) για κάθε frame.

```
%----------ex 2.2 Quantization of the signal y_downsamp--------------------
R=2^16;
%partitioning the global masking threshold of each frame into 32 frames of
%length 8
for num_frame=1:frames
    Tg_framed(:,:,num_frame)=buffer(Tg(:,num_frame),8,0,'nodelay');
end
%find the coding bits for each filter of each frame
for num_frame=1:frames
    for k=1:M
        Tg_min=min(Tg_framed(:,k,num_frame)); %find the min Tg of each filter in each frame
        B(k,num_frame)=ceil(log2(R/Tg_min)-1); %bits/sample
    end
end
```

Στην συνέχεια βρίσκουμε το βήμα και υλοποιούμε την κβάντιση:

```
%find the quantization step D for each filter of each frame
for num_frame=1:frames
    for k=1:M
        range=max(x_framed(:,num_frame))-min(x_framed(:,num_frame)); %Dk=xmax-xmin/2^Bk
        D(k,num_frame)=range/(2^B(k,num_frame)); %step for each frame
    end
end

%quantization of y_downsamp with
for num_frame=1:frames
    for k=1:M
        Dk=D(k,num_frame);
        y_quantized(:,k,num_frame)=Dk*(ceil(y_downsamp(:,k,num_frame)/Dk)+1/2);
    end
end
```

Παρόμοια δημιουργούμε έναν μη προσαρμοζόμενο κβαντιστή με σταθερό αριθμό bit του κβαντιστή, όπου Bk=8 και σταθερό βήμα κβαντισμού Δ στο πεδίο [-1,1]. (Για την δημιουργία του δημιουργούμε νέα συνάρτηση που διαφέρει μόνο ως προς το βήμα 2.2)

```
%----------ex 2.2 Quantization of the signal y_downsamp-------------------

R=2^16;

%partitioning the global masking threshold of each frame into 32 frames of
%length 8
for num_frame=1:frames
    Tg_framed(:,:,num_frame)=buffer(Tg(:,num_frame),8,0,'nodelay');
end

Bk=8;   %constant number of coding bits for each sample, of each filter and frame

B_total=Bk*M*frames;

Dk=2/(2^Bk);   %constant quantization step Dk=1-(-1)/(2^Bk)

%quantization of y_downsamp with
for num_frame=1:frames
    for k=1:M
        y_quantized(:,k,num_frame)=Dk*(ceil(y_downsamp(:,k,num_frame)/Dk)+1/2);
    end
end
```

Βήμα 2.3: Σύνθεση

Στην συνέχεια οι κβαντισμενες τιμές (Βήμα 2.2) στέλνονται στον αποκωδικοποιητή όπου παρεμβάλονται με M μηδενικά και υπερδειγματοληπτούνται (overpolation) και στην συνέχεια πραγματοποιείται συνέλιξη για κάθε frame με τα φίλτρα σύνθεσης gk. Χρησιμοποιούμε τις συναρτήσεις **conv()** και **upsample()** .

```
%-------------ex 2.3 Synthesis of the signal y_downsamp---------------

 %upsampling

[rown_y,coln_y,frames]=size(y_quantized);

w=zeros(rown_y*M,M,frames);
for num_frame=1:frames
    for k=1:M   %num_filter
        w(:,k,num_frame)=upsample(y_quantized(:,k,num_frame),M);
    end
end

%convolution with synthesis filter
for num_frame=1:frames
    for k=1:M
        w_conv(:,k,num_frame)=conv(g_filter(:,k)',w(:,k,num_frame));
    end
end
```

Η τελική ανακατασκευή του σήματος μουσικής γίνεται εφαρμόζοντας την τεχνική της OverLap-Add:

```
%reconstructed framed signal: x_rec_windowed (639x1271)

[rown_w,coln_w,frames]=size(w_conv);

x_rec_framed=zeros(rown_w,frames);

for num_frame=1:frames
    for n=1:rown_w
        for k=1:M
            x_rec_framed(n,num_frame)=x_rec_framed(n,num_frame)+w_conv(n,k,num_frame);
        end
    end
end
```

```
%final reconstructed signal: x_rec

x_rec_shifted(1:rown_w)=x_rec_framed(1:rown_w,1);   %first 639 samples of 1st frame inserted

index=rown_w; %639 samples are added

overlap=rown_w-L;  %overlap=127

for num_frame=2:frames
    for n=1:overlap  %the first 127 samples of the frame are added to the previous one
        x_rec_shifted(index+n-overlap)=x_rec_framed(n,num_frame);
    end
    for n=overlap+1:rown_w
        index=index+1;
        x_rec_shifted(index)=x_rec_framed(n,num_frame);
    end
end

x_rec_shifted=x_rec_shifted';
```

Λόγω καθυστέρησης των φίλτρων παρατηρούμε ότι έχουμε χρονική μετατόπιση στην αρχή μήκους 2M=64 που θέλουμε να αφαιρέσουμε:

```
%reducing the delay, there is a shift of 2*M=64

x_rec=zeros(length_x,1);
for n=1:length_x
    x_rec(n,1)=x_rec_shifted(2*M+n-1,1);
end
```

Τρέχοντας την main μας βλέπουμε πόσα bits έχουμε συνολικά με την χρήση του προσαρμοζόμενου και του μη προσαρμοζόμενου κβαντιστή αλλά και του αρχικού μας σήματος (σήμα x):

b	*1x256 double*
coln	1271
f	*1x256 double*
fs	44100
i	1271
L	512
M	32
rown	512
Tg	*256x1271 double*
total_bits_ada...	361723
total_bits_non...	325376
Tq	*1x256 double*
w	*512x1 double*
x	*650415x1 double*
x_framed	*512x1271 double*
x_normalized	*650415x1 double*
x_rec_adapt	*650415x1 double*
x_rec_non_ad...	*650415x1 double*
x_windowed	*512x1271 double*

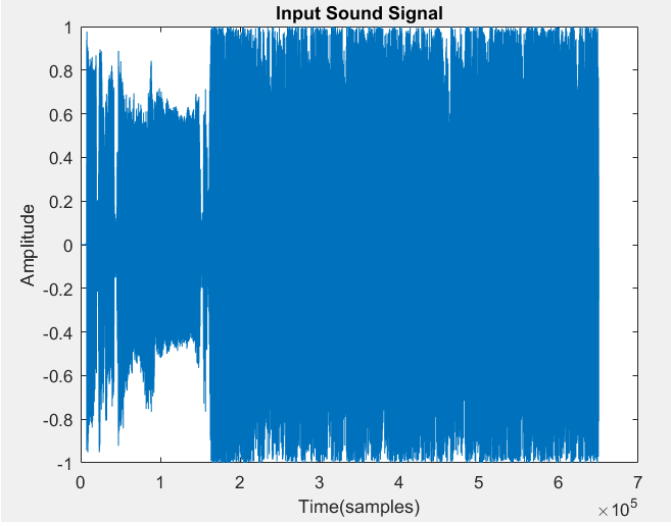
Για το σήμα x έχουμε 650415 δείγματα αλλά είναι τύπου double που είναι 16 bit άρα έχουμε $bits_{original} = 650415$ και $bits_{adapt} = 361723$ και $bits_{original} = 325376$.

Επομένως για τον προσαρμοζόμενο κβαντιστή έχουμε **ποσοστό κβάντισης** 361723/650415=0.56 ή **56%** και **άρα ποσοστό συμπίεσης 44%**

Αντίθετα για τον μη προσαρμοζόμενο κβαντιστή έχουμε **ποσοστό κβάντισης** 325376/650415=0.50 ή **50%** όπως και ήταν λογικό αφού έχουμε σταθερό αριθμό bit κβαντισμού (8/16). Άρα **ποσοστό συμπίεσης 50%**

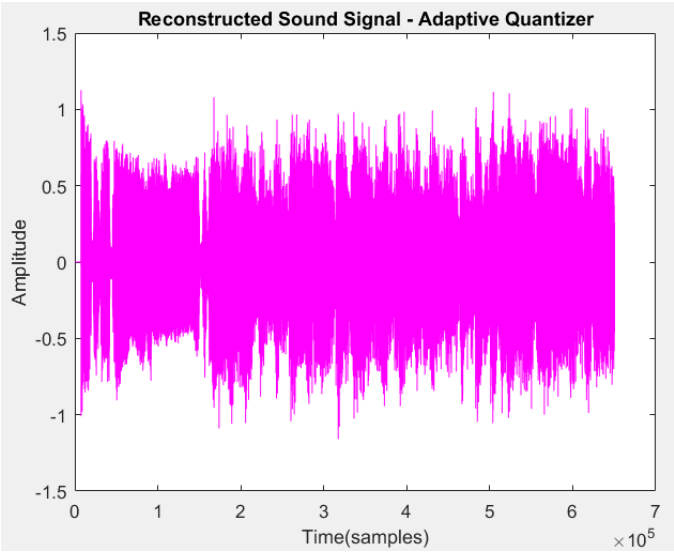
<u>Γραφικές Εισόδου - Εξόδου για τους δύο κβαντιστές:</u>

Σήμα εισόδου:



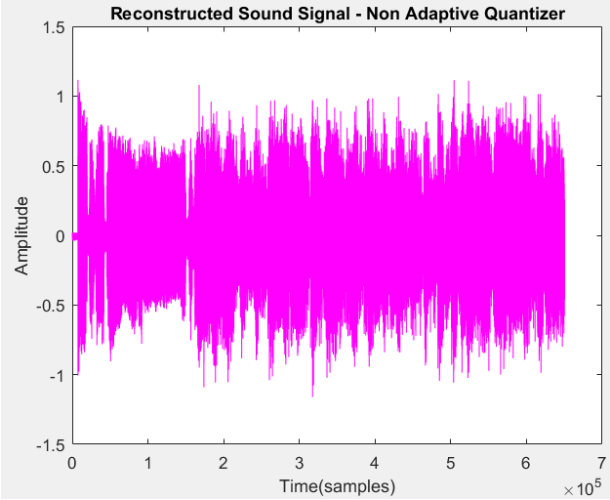
Σχήμα 2.3α

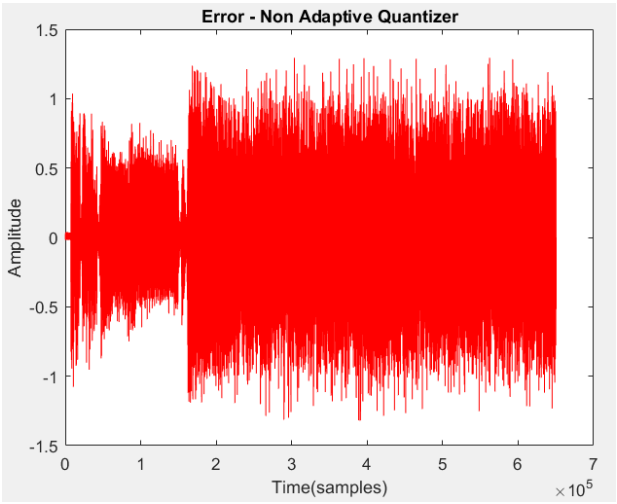
Γραφικές για τον προσαρμοζόμενο κβαντιστή:



Σχήμα 2.3β

Γραφικές για τον μη προσαρμοζόμενο κβαντιστή:





Σχήμα 2.3γ

<u>Υπολογισμός των Μέσων Τετραγωνικών Σφαλμάτων:</u>

Με τον παρακάτω κώδικα υπολογίσαμε ότι το Μέσο Τετραγωνικό Σφάλμα για τον προσαρμοζόμενο κβαντιστή είναι 0.1353 ή 13.53% , ενώ για τον μη προσαρμοζόμενο κβαντιστή είναι 0,1354 ή 13.54%.

```
%MSE
error1=error_adapt.^2;
error2=error_non_adapt.^2;
mse_adapt=0;
mse_non_adapt=0;
for i=1:length(x)
    mse_adapt=mse_adapt+error1(i);
    mse_non_adapt=mse_non_adapt+error2(i);
end
mse_adapt=mse_adapt/length(x);
mse_non_adapt=mse_non_adapt/length(x);
disp(mse_adapt);
disp(mse_non_adapt);
```

Παρατηρήσεις

• Έχουμε καλύτερο ποσοστό συμπίεσης χρησιμοποιώντας μη προσαρμοζόμενο κβαντιστή διότι από 16 bits/δείγμα έχουμε στο συμπιεσμένο 8 bits/δείγμα. Αντίθετα, στον προσαρμοσμένο κβαντιστή όπως είδαμε παραπάνω έχουμε λιγότερη συμπίεση. Ωστόσο επειδή έχουμε περισσότερα bits/δείγμα στον προσαρμοζόμενο κβαντιστή σημαίνει ότι στην έξοδο θα έχουμε καλύτερης ποιότητας ήχο.

• Τα μέσα τετραγωνικά σφάλματα που υπολογίσαμε παραπάνω παρατηρούμε ότι έχουν πολύ μικρή απόκλιση και στις δύο μεθόδους με μικρότερο το σφάλμα στον προσαρμοσμένο κβαντιστή και αυτό γιατί χρησιμοποιούνται περισσότερες στάθμες για την κωδικοποίηση.

Από τα παραπάνω συμπεραίνουμε ότι για κωδικοποίηση ηχητικών σημάτων προτιμάται προσαρμοζόμενος κβαντιστής Γιατι με μικρό αριθμό από επιπλέον bits/δειγμα, μας βελτιωνει ικανοποιητικα την ποιοτητα του σηματος και ελαττωνει το σφάλμα.