Advanced Audio Coding

ΣΥΣΤΗΜΑΤΑ ΠΟΛΥΜΕΣΩΝ

Αναστάσιος Μουρατίδης 9040
Φεβρουάριος 2021 | atmourat@ece.auth.gr

Περιεχόμενα

Εισαγωγή

Η εργασία αφορά την απλοποιημένη υλοποίηση του κωδικοποιητή και αποκωδικοποιητή ήχου κατά το πρότυπο Advanced Audio Coding (AAC), που ανήκει στην κατηγορία waveform compression.

Το αρχικό σήμα που κωδικοποιούμε στο πλαίσιο αυτής της εργασίας είναι stereo, έχει, δηλαδή, δύο κανάλια ήχου και δειγματοληψία $48000\ {}^{samples}/_{sec}$. Το σήμα χωρίζεται σε επικαλυπτόμενα κατά 50% frames μήκους 2048 δειγμάτων.

Η υλοποίηση της εργασίας γίνεται σε 3 επίπεδα, όπως ζητείται στην εκφώνηση, οπότε στους τρεις φακέλους περιέχονται οι συναρτήσεις για την υλοποίηση του κάθε επιπέδου. Σημειώνεται κάθε επίπεδο περιέχει και αντίγραφα των συναρτήσεων του προηγούμενου επιπέδου που απαιτούνται για την λειτουργικότητά του, ώστε να μπορούμε να τρέξουμε κάθε επίπεδο αυτόνομα.

Σύμφωνα με την εκφώνηση χρησιμοποιούνται οι εξής συντομογραφίες για το τύπο του frame:

- “OLS”: ONLY_LONG_SEQUENCE
- “LSS”: LONG_START_SEQUENCE
- “ESH”: EIGHT_SHORT_SEQUENCE
- “LPS”: LONG_STOP_SEQUENCE

1. Level 1

1.1 Συναρτήσεις

1.1.1 SSC

Στη συνάρτηση γίνεται η επιλογή του τύπου του frame, δηλαδή "**OLS**", "**ESH**", "**LSS**", "**LPS**", σύμφωνα με παραπάνω. Η επιλογή στηρίζεται στον τύπο του προηγούμενου frame αλλά και του επόμενου. Ο τύπος του προηγούμενου frame δίνεται ως όρισμα στη συνάρτηση, ενώ για το επόμενο frame, χρησιμοποιούμε τα δείγματά του (επίσης όρισμα της συνάρτησης) για να αποφανθούμε αν είναι **ESH** ή όχι.

Εφόσον, υπολογίσουμε τον τύπο frame κάθε καναλιού, ο κοινός τελικός τύπος προκύπτει από τον πίνακα 1 της εκφώνησης, που υλοποιείται στη συνάρτηση

common = commonFinalType(channel0, channel1)

που παίρνει ως ορίσματα τον τύπο των δύο καναλιών κι επιστρέφει την τελική απόφαση.

Σημειώνουμε πως όρισμα της συνάρτησης *SSC* είναι και τα δείγματα του ίδιου frame που εξετάζουμε τον τύπο του (frameT), αλλά δεν χρησιμοποιήθηκαν τελικά στην πρόβλεψη του. Εφόσον, λοιπόν, η εργασία προϋποθέτει ότι χωρίζουμε το σήμα εισόδου σε επικαλυπτόμενα κατά 50% frames μήκους 2048 δειγμάτων, μπορούμε να ελέγχουμε σε κάθε κλήση της συνάρτησης αν το frame έχει μήκος 2048 δείγματα.

1.1.2 filterbank και iFilterbank

Ο επόμενος μηχανισμός που χρειάζεται στην κωδικοποίηση και αντίστοιχα στην αποκωδικοποίηση είναι το ζεύγος συναρτήσεων filterbank και iFilterbank.

Στη filterbank, εισάγεται το κάθε frame (ή subframe), και αφού πολλαπλασιάσουμε με το κατάλληλο παράθυρο, που είναι κριτήριο για την τέλεια ανακατασκευή, εφαρμόζουμε τον μετασχηματισμό MDCT. Έτσι, πετυχαίνουμε τόσο την μείωση της συσχέτισης των δειγμάτων όσο και την μετάβαση στο πεδίο της συχνότητας. Για N δείγματα προκύπτουν $N/2$ συντελεστές. Η υλοποίηση του μετασχηματισμού MDCT έγινε σύμφωνα με την https://www.ee.columbia.edu/~marios/mdct/mdct4.m, με κάποιες αλλαγές.

Από την άλλη, στην iFilterbank εκτελείται η αντίστροφη διαδικασία. Αφού εφαρμοστεί ο αντίστροφος μετασχηματισμός Modified DCT, δηλαδή, ο IMDCT. Αντιστοίχως με πριν, έγινε η χρήση https://www.ee.columbia.edu/~marios/mdct/imdct4.m, με κάποιες διορθώσεις. Στη συνέχεια, πολλαπλασιάζουμε με το ίδιο παράθυρο που είχαμε πολλαπλασιάσει και πριν. Σημειώνεται, πως για λόγους απλούστευσης

χρησιμοποιούμε έναν τύπο παραθύρου για όλα τα frames του σήματος, όπως ζητήθηκε και στην εκφώνηση.

Όσον αφορά τα παράθυρα κατασκευάστηκαν τα δύο είδη που περιγράφονται στην εκφώνηση, δηλαδή τα Kaiser-Bessel-Derived (KBD) και τα sinusoid (SIN). Για τα KBD έγινε χρήση της συνάρτησης *kbdwin(N,a)* η οποία περιέχεται στη MATLAB από την έκδοση 2019a και εξής και συμβαδίζει με τον τύπο της εκφώνησης. Για τα SIN χρησιμοποιούμε τον απλό τύπο που δίνεται.

Για κάθε τύπο frame χρησιμοποιείται διαφορετική διαδικασία παραθύρωσης και μετασχηματισμού.

1.1.3 AACoder1 και iAACoder1

Αρχικά η AACoder1 διαβάζει το αρχείο που παίρνει ως όρισμα fNameIn με τη χρήση της audioread. Θεωρούμε πως το πρώτο frame έχει τύπο ONLY_LONG_SEQUENCE και πως η επικάλυψη των frames είναι 0.5. Ο αριθμός των δειγμάτων της εισόδου θα πρέπει να διαιρείται ακριβώς με το μήκος του frame, οπότε προσθέτουμε μηδενικά στο τέλος της ακολουθίας εισόδου, έτσι ώστε να έχουμε τέλεια διαίρεση.

Ο αριθμός αυτών των μηδενικών υπολογίζεται ως εξής:

$$2048 - mod(\mu\acute{\eta}\kappa o\varsigma\ \tau\eta\varsigma\ \epsilon\iota\sigma\acute{o}\delta o\upsilon, 2048)$$

Επίσης, προσθέτουμε 1024 μηδενικά στην αρχή και 1024 στο τέλος, για να μην έχουμε πρόβλημα με τις οριακές συνθήκες του προηγούμενου και του επόμενου frame. Αυτό τελικά βελτιώνει και το SNR.

Υπολογίζουμε τον αριθμό των επικαλυπτόμενων frames κι επαναληπτικά υπολογίζουμε τον τύπο των frames με την SSC, τους συντελεστές MDCT με την filterbank και τα αποθηκεύουμε στο struct εξόδου.

Στην iAACoder1 εκτελείται η αποκωδικοποίηση, δηλαδή υπολογίζεται η ανακατασκευασμένη ακολουθία ήχου με την iFilterbank και αποθηκεύεται στο αρχείο fNameOut με τη χρήση της audiowrite.

1.1.4 demoAAC1

Η συνάρτηση αυτή επιδεικνύει την κωδικοποίηση του 1ου επιπέδου. Καλεί την AACoder1, κωδικοποιεί την είσοδο, και προκύπτει μια δομή των κωδικοποιημένων frames όπως έχουμε εξηγήσει και παραπάνω. Αυτή η δομή μπαίνει ως όρισμα στην iAACoder1 και προκύπτει η ανακατασκευασμένη ακολουθία εξόδου. Υπολογίζουμε τον θόρυβο που έχει προστεθεί σε κάθε κανάλι αφαιρώντας τα δείγματα της εισόδου εξόδου. Το Signal-to-Noise ratio (SNR) υπολογίζεται με τη χρήση της συνάρτησης snr της MATLAB για κάθε κανάλι ξεχωριστά και το συνολικό SNR προκύπτει ως μέσος όρος αυτών.

1.2 Αποτελέσματα

Στη γραμμή εντολών της MATLAB γράφουμε,

$$SNR = demoAAC2(\text{LicorDeCalandraca.wav, out.wav});$$

και προκύπτουν τα παρακάτω αποτελέσματα:

- Χρησιμοποιώντας το **SIN** παράθυρο προέκυψαν τα εξής αποτελέσματα:

$$SNR\ για\ το\ αριστερό\ κανάλι: 307.8288$$

$$SNR\ για\ το\ δεξί\ κανάλι: 307.8950$$

$$Συνολικό\ SNR: 307.8619$$

- Χρησιμοποιώντας το **KBD** παράθυρο προέκυψαν τα εξής αποτελέσματα:

$$SNR\ για\ το\ αριστερό\ κανάλι: 308.6251$$

$$SNR\ για\ το\ δεξί\ κανάλι: 308.6612$$

$$Συνολικό\ SNR: 308.6432$$

Η διαδικασία της κωδικοποίησης και αποκωδικοποίησης διαρκεί αμελητέο χρόνο και στις δύο περιπτώσεις.

2. Level 2

2.1 Συναρτήσεις

2.1.1 TNS και iTNS

Η βαθμίδα Temporal Noise Shaping εφαρμόζει ένα γραμμικό μοντέλο γραμμικής πρόβλεψης στους συντελεστές MDCT, κι έτσι τους μετασχηματίζει σε ένα σύνολο ισάριθμων συντελεστών, όπου έχουν απαλειφθεί οι περιοδικότητες.

Σημειώνεται ότι στις μπάντες του ψυχοακουστικού μοντέλου που δίνονται στους πίνακες Table B.2.1.9.b (για ESH) και Table B.2.1.9.a (για όλους τους υπόλοιπους τύπους frame) στις στήλες index, w_low και w_high προσθέτουμε + 1 για να υπάρχει αντιστοιχία με τα index της MATLAB που ξεκινούν από το 1 και όχι το 0. $(0 \dots 1023 \rightarrow 1 \dots 1024)$

Στην συνάρτηση **TNS** ακολουθείται η διαδικασία που περιγράφεται αναλυτικά στην εκφώνηση. Αναφέρεται συνοπτικά με κάποιες περαιτέρω διευκρινίσεις, πάνω στον κώδικα που την υλοποίησε:

1. *Κανονικοποίηση συντελεστών MDCT*

   Η ενέργεια της κάθε μπάντας υπολογίζεται από τον τύπο

   $$P(j) = \sum_{k=b_j}^{b_{j+1}-1} X(k)^2,\ j = 0, \dots, N_B - 1$$

   Δίνεται ότι b_j είναι ο αύξων αριθμός του συντελεστή w_low της υπ' αριθμόν j μπάντας. Οπότε, το άνω όριο της σειράς, $b_{j+1} - 1$ πρόκειται για το τον συντελεστή w_high, και χρησιμοποιείται στον κώδικα για τον υπολογισμό της ενέργειας της j μπάντας.

2. *Γραμμική πρόβλεψη*

Η επίλυση των κανονικών εξισώσεων $Ra = r$ για να βρούμε τους βέλτιστους συντελεστές a γίνεται με την συνάρτηση *lpc* της MATLAB, η οποία σύμφωνα με το documentation επιλύει αυτό το πρόβλημα.

3. *Οι συντελεστές κβαντίζονται με 4 bits χρησιμοποιώντας ομοιόμορφο συμμετρικό κβαντιστή βήματος 0.1.*
4. *Εφαρμόζεται το FIR φίλτρο:*

   $$H_{TNS}(z) = 1 - a_1 z^{-1} - a_2 z^{-2} - \dots - a_p z^{-p}$$

   Η εφαρμογή αυτού του φίλτρου γίνεται με την συνάρτηση *filter*.

Όσον αφορά την συνάρτηση **iTNS**, εφαρμόζεται το αντίστροφο φίλτρο που χρησιμοποιήθηκε στην TNS. Σημειώνεται πως το frameType δεν χρησιμοποιήθηκε στην υλοποίηση της **iTNS**, παρόλο που στην εκφώνηση δίνεται ως όρισμα**.**

2.1.2 AACoder2 και iAACoder2

Η υλοποίηση των AACoder2 και iAACoder2 είναι αντίστοιχη με αυτή των AACoder1 και iAACoder1 του πρώτου επιπέδου. Εκτελούν, δηλαδή, την κωδικοποίηση και αποκωδικοποίηση του δεύτερου επιπέδου, αντίστοιχα. Η διαφορά τους έγκειται στο ότι πλέον στην AACSeq2 αποθηκεύονται και οι συντελεστές TNS, οι οποίοι υπολογίζονται με κλήση της TNS. Αντίστοιχα, το frameF στην iAACoder2 προκύπτει από την κλήση της iTNS και στο εξής ακολουθείται η ίδια διαδικασία με την iAACoder1.

2.1.3 demoAAC2

Η λειτουργία της είναι αντίστοιχη με αυτή της demoAAC1, επιδεικνύει, δηλαδή, την κωδικοποίηση και αποκωδικοποίηση του δεύτερου επιπέδου.

2.2 Αποτελέσματα

Στη γραμμή εντολών της MATLAB γράφουμε,

$$SNR = demoAAC2(\text{LicorDeCalandraca.wav, out.wav});$$

και προκύπτουν τα παρακάτω αποτελέσματα:

- Χρησιμοποιώντας το **SIN** παράθυρο προέκυψαν τα εξής αποτελέσματα:

$$SNR\ για\ το\ αριστερό\ κανάλι: 307.8248$$

$$SNR\ για\ το\ δεξί\ κανάλι: 307.8630$$

$$Συνολικό\ SNR: 307.8439$$

- Χρησιμοποιώντας το **KBD** παράθυρο προέκυψαν τα εξής αποτελέσματα:

$$SNR\ για\ το\ αριστερό\ κανάλι: 308.5930$$

$$SNR\ για\ το\ δεξί\ κανάλι: 308.6149$$

$$Συνολικό\ SNR: 308.6039$$

Η διαδικασία της κωδικοποίησης και αποκωδικοποίησης διαρκεί αμελητέο χρόνο και στις δύο περιπτώσεις.

3. Level 3

3.1 Συναρτήσεις

3.1.1 Psycho

Πρόκειται για τη συνάρτηση που υλοποιεί την βαθμίδα του ψυχοακουστικού μοντέλου, που έχει ως στόχο τη μείωση των bit που απαιτούνται στην κβάντιση.

Σημειώνεται πως και σε αυτή τη συνάρτηση χρησιμοποιούνται οι πίνακες B.2.1.9.a και B.2.1.9.b.

Για τον υπολογισμό του ψυχοακουστικού μοντέλου ακολουθούνται τα παρακάτω που περιγράφονται στην εκφώνηση και αναφέρονται με κάποια σχόλια για τον κώδικα που χρησιμοποιήθηκε για την υλοποίηση τους:

1. *Υπολογίζεται η spreading function*

   Πρόκειται για μια συνάρτηση κατανομής της μάσκας, η οποία υπολογίζεται για όλους τους δυνατούς συνδυασμούς από μπάντες. Εφόσον χρησιμοποιείται η τιμή των bval για τον υπολογισμό του spreading table, ο πίνακας έχει διαφορετικές τιμές για long και short frames.

2. *Κάθε frame ή sub-frame πολλαπλασιάζεται με ένα παράθυρο Hann στο πεδίο του χρόνου*

   Το παράθυρο Hann υπολογίζεται ως εξής:
   hannWin = (0.5 - 0.5 * cos((pi*(0:N - 1))/(N)))'

3. *Υπολογίζονται οι προβλέψεις r(w) και f(w) για κάθε παράθυρο*

$$r_{pred}(w) = 2r_{-1}(w) - r_{-2}(w)$$
$$f_{pred}(w) = 2f_{-1}(w) - f_{-2}(w)$$

   - r, f: μέτρο και φάση FFT

4. *Υπολογίζεται ένα μέτρο προβλεψιμότητας*

$$c(w) = \frac{\sqrt{\left(r(w)\cos\left(f(w)\right) - r_{pred}(w)\cos\left(f_{pred}(w)\right)\right)^2 + \left(r(w)\sin\left(f(w)\right) - r_{pred}(w)\sin\left(f_{pred}(w)\right)\right)^2}}{r(w) + \left|r_{pred}(w)\right|}$$

Η υλοποίηση των δύο παραπάνω βημάτων είναι απλή και δεν παρατίθεται εδώ.

5. *Υπολογίζεται η ενέργεια και η βεβαρυμένη προβλεψιμότητα για όλες τις συχνότητες σε κάθε μπάντα με δείκτη b των πινάκων.*

$$e(b) = \sum_{w=w_{low}(b)}^{w_{high}(b)} r(w)^2$$

$$c(b) = \sum_{w=w_{low}(b)}^{w_{high}(b)} c(w)r(w)^2$$

Κώδικας υλοποίησης:

```
e  = zeros(size(bb));
c = zeros(size(bb));
for b = bb
    index = w_low(b):w_high(b);
    e(b) = sum( r_current(index) .^ 2 );
    c(b) = sum( c_w(index) .* ( r_current(index) .^ 2) );
end
```

6. *Συνδυάζεται η ενέργεια και η προβλεψιμότητα με τη spreading function ως εξής*

$$ecb(b) = \sum_{bb=0}^{N_B-1} e(bb)spreading\_function(bb,b)$$

$$ct(b) = \sum_{bb=0}^{N_B-1} c(bb)spreading\_function(bb,b)$$

Όπου N_B ο αριθμός από μπάντες. Ακολουθεί ο κώδικας υλοποίησης καθώς και η κανονικοποίηση των δύο τιμών σύμφωνα με την εκφώνηση:

```
ecb =  zeros(size(bb));
ct  =  zeros(size(bb));
for b = bb
    ct(b) = sum( c(bb) * spreadingTable(bb, b) );
    ecb(b) = sum( e(bb) * spreadingTable(bb, b) );
end
cb = ct ./ ecb;
en = ecb ./ sum(spreadingTable(bb,:));
```

7. *Υπολογίζεται το tonality index tb(b) κάθε μπάντας*

$$tb(b) = -0.299 - 0.43\ln\big(cb(b)\big)$$
$$tb\ \epsilon\ (0,1)$$

8. *Υπολογίζεται το SNR κάθε μπάντας*

$$SNR(b) = tb(b)\cdot 18 + \big(1 - tb(b)\big)\cdot 6\ \ (dB)$$

9. *Μετατρέπεται από dB σε λόγο ενέργειας*

$$bc(b) = 10^{-\frac{SNR(b)}{10}}$$

10. *Υπολογίζεται το κατώφλι ενέργειας*

$$nb(b) = en(b)bc(b)$$

11. *Υπολογίζεται το επίπεδο θορύβου ως:*

$$npart(b) = \max\{nb(b), \widehat{q_{thr}}(b)\}$$

$$\widehat{q_{thr}} = \varepsilon\frac{N}{2}10^{\frac{qsthr}{10}}$$

12. *Υπολογίζεται το Signal to Mask Ratio (SMR) της μπάντας με δείκτη b*

$$SMR(b) = \frac{e(b)}{npart(b)}$$

13. *Υπολογίζονται τα κατώφλια ακουστότητας του κωδικοποιητή T(b)*

$$P(b) = \sum_{k=w_{low}(b)}^{w_{high}(b)} X(k)^2$$

$$T(b) = \frac{P(b)}{SMR(b)}$$

Το συγκεκριμένο βήμα υλοποιείται στη συνάρτηση AACquantizer.

Οπότε τελικά επιστρέφεται σε κάθε κλήση της psycho οι συντελεστές SMR, δηλαδή ένας πίνακας διάστασης 42×8 για frames τύπου EIGHT_SHORT_SEQUENCE και 69×1 για όλους τους άλλους τύπους.

3.1.2 AACquantizer και iAACquantizer

Οι δύο συναρτήσεις υλοποιούν τις βαθμίδες κβαντιστή και αποκβαντιστή στον κωδικοποιητή και αποκωδικοποιητή του τρίτου επιπέδου αντίστοιχα.

Ο κβαντιστής προσαρμόζει τα επίπεδα κβαντισμού του ανάλογα με τα Scale Factor Bands (που υπολογίζονται στο 13ο βήμα του ψυχοακουστικού μοντέλου κι υλοποιούνται σε αυτή τη συνάρτηση) και εφαρμόζει τις ίδιες παραμέτρους κβάντισης για όλους τους συντελεστές της μπάντας.

Ο κβαντιστής υλοποιείται από την σχέση:

$$S(k) = sgn\big(X(k)\big) int\left[\left(|X(k)| \times 2^{-\frac{1}{4}a}\right)^{\frac{3}{4}} + 0.4054\right]$$

Ενώ ο αποκβαντιστής υλοποιείται από την σχέση:

$$\hat{X}(k) = sgn\big(S(k)\big)|S(k)|^{\frac{4}{3}} \times 2^{\frac{1}{4}a}$$

Κώδικας υλοποίησης:

```
%% Quantization Function
function S = quant(Xk, a)
MagicNumber = 0.4054;
S = sign(Xk) .* round( (abs(Xk) .* 2.^(- 1 / 4 * a)).^(3/4) + MagicNumber );
end
%% Dequantization Function
function Xk = dequant(S, a)
Xk = sign(S) .* (abs(S).^(4/3)).*2.^(1 / 4 * a);
end
```

Ο συντελεστής a ονομάζεται scale factor gain, ρυθμίζει την ποιότητα του κβαντιστή, ενώ η βέλτιστη τιμή του υπολογίζεται με βάση τα κατώφλια ακουστότητας που υπολογίστηκαν στο ψυχοακουστικό μοντέλο.

Ως πρώτη τιμή χρησιμοποιείται η τιμή

$$\hat{a}(b) = \frac{16}{3}\left(\frac{max_k\big(X(k)\big)^{\frac{3}{4}}}{8191}\right)$$

Στη συνέχεια, επαναληπτικά, αυξάνουμε μια μονάδα τον συντελεστή a(b) όταν η ισχύς του σφάλματος κβαντισμού είναι κάτω από το κατώφλι ακουστότητας. Η επαναληπτική διαδικασία τερματίζει σε δύο περιπτώσεις

i. Έχουμε φτάσει το κατώφλι T(b)

```
aHat = a_b + (Pe < T);
if (aHat == a_b)
    break;
end
```

ii. Αν $\max_b(|a(b+1) - a(b)|) > 60$

```
if (max(abs(diff(aHat))) > 60)
    break;
end
```

Τελικά, στην έξοδο κωδικοποιούνται, εκτός από τα S:

1. Global gain του frame $G = a(0)$
2. Τα Scale Factors του frame, με χρήση DPCM, δηλαδή:

```
sfc(:, colIndex) = [a_b(1); diff(a_b)]
```

Όσον αφορά τη συνάρτηση iAACQuantizer, οι τιμές του a ανακτώνται από την sfc με τη χρήση της cumsum της MATLAB, ενώ, τελικά οι συντελεστές MDCT του frameF προκύπτουν από τον αποκβαντιστή που δείξαμε πιο πάνω.

3.1.3 AACoder3 και iAACoder3

Η λογική των συναρτήσεων κωδικοποίησης και αποκωδικοποίησης είναι η ίδια με τα δύο προηγούμενα επίπεδα, και θα αναφερθούν τα κομμάτια κώδικα που προσθέτουν τη λειτουργικότητα που απαιτείται στο τρίτο επίπεδο.

Όσον αφορά την AACoder3, αρχικοποιούνται τα δύο προηγούμενα frames, frameTprev1 και frameTprev2 που χρειάζονται για την κλήση της psycho. Αρχικά, καλείται η TNS για κάθε κανάλι ξεχωριστά και προκύπτουν οι συντελεστές MDCT μετά τη διαδικασία Temporal Noise Shaping και οι συντελεστές TNS. Στη συνέχεια, για κάθε κανάλι καλούνται διαδοχικά:

- Η psycho που επιστρέφει τον πίνακα SMR.
- Η AACquantizer, που χρησιμοποιεί την έξοδο SMR και κβαντίζει με τις κατάλληλες παραμέτρους
- Τέλος, η encodeHuff που εκτελεί την κωδικοποίηση των διαφόρων συντελεστών κατά Huffman.

Από την άλλη στην iAACoder3, γίνεται η αποκωδικοποίηση για κάθε κανάλι ξεχωριστά, με διαδοχική κλήση της decodeHuff και της iAACquantizer, κι ανακτώνται οι συντελεστές του frameF. Έπειτα, ακολουθεί όπως και στα προηγούμενα επίπεδα, η διαδικασία iTNS , από όπου προκύπτει το ανακατασκευασμένο, ενώ το τελικό frameF υπολογίζεται από την κλήση της iFilterbank.

3.1.4 demoAAC3

Η demoAAC3, σε σύγκριση με τις αντίστοιχες demo των πρώτων δύο επιπέδων επιστρέφει εκτός από το SNR, το bitrate της ανακατασκευασμένης ακολουθίας ήχου καθώς και το ποσοστό συμπίεσης (compression) που επιτεύχθηκε.

Για των υπολογισμό των δύο νέων εξόδων χρησιμοποιούμε τη συνάρτηση dir της MATLAB, για την είσοδο και την έξοδο για να υπολογίσουμε το μέγεθος τους σε bits.

Το bitrate υπολογίζεται ως:

$$bitrate = outputBits/((length(audioOut)/48000))$$

Το ποσοστό συμπίεσης, ορίζεται ως το bitrate εξόδου προς το bitrate εισόδου, αλλά όπως βλέπουμε και από τον παραπάνω τύπο, αυτές οι δύο παράμετροι έχουν ίδιο παρονομαστή, οπότε το compression υπολογίζεται ως:

$$compression = outputBits \; / \; inputBits$$

3.2 Αποτελέσματα

Στη γραμμή εντολών της MATLAB γράφουμε,

$[SNR1, bitrate1, compression1] = demoAAC3("LicorDeCalandraca.wav", "out.wav", "out.mat");$

και προκύπτουν τα παρακάτω αποτελέσματα:

- Χρησιμοποιώντας το **SIN** παράθυρο, χωρίς κανονικοποίηση της αποκωδικοποιημένης ακολουθίας προέκυψαν τα εξής αποτελέσματα:

$$SNR \; για \; το \; αριστερό \; κανάλι = 307.8248$$
$$SNR \; για \; το \; δεξί \; κανάλι = 307.8630$$
$$Συνολικό \; SNR = 307.8439$$
$$Bitrate = 4.8371e + 05 \; bps$$
$$Compression = 0.3149$$

  Η διαδικασία της κωδικοποίησης διήρκησε 43.13 δευτερόλεπτα, ενώ της αποκωδικοποίησης 2.74 δευτερόλεπτα.

- Χρησιμοποιώντας το **SIN** παράθυρο, με κανονικοποίηση της αποκωδικοποιημένης ακολουθίας προέκυψαν τα εξής αποτελέσματα:

$$SNR \; για \; το \; αριστερό \; κανάλι = 10.3284$$
$$SNR \; για \; το \; δεξί \; κανάλι = 8.6215$$
$$Συνολικό \; SNR = 9.4749$$
$$Bitrate = 4.8371e + 05 \; bps$$

$$Compression = 0.3149$$

Η διαδικασία της κωδικοποίησης διήρκησε 42.85 δευτερόλεπτα, ενώ της αποκωδικοποίησης 2.75 δευτερόλεπτα.

- Χρησιμοποιώντας το **KBD** παράθυρο, χωρίς κανονικοποίηση της αποκωδικοποιημένης ακολουθίας προέκυψαν τα εξής αποτελέσματα:

$$SNR\ για\ το\ αριστερό\ κανάλι = 3.6090$$

$$SNR\ για\ το\ δεξί\ κανάλι = 3.1402$$

$$Συνολικό\ SNR = 3.3746$$

$$Bitrate = 4.8372e + 05\ bps$$

$$Compression = 0.3149$$

Η διαδικασία της κωδικοποίησης διήρκησε 54.35 δευτερόλεπτα, ενώ της αποκωδικοποίησης 3.06 δευτερόλεπτα.

- Χρησιμοποιώντας το **KBD** παράθυρο, με κανονικοποίηση της αποκωδικοποιημένης ακολουθίας προέκυψαν τα εξής αποτελέσματα:

$$SNR\ για\ το\ αριστερό\ κανάλι = 10.0685$$

$$SNR\ για\ το\ δεξί\ κανάλι = 9.8234$$

$$Συνολικό\ SNR = 9.9460$$

$$Bitrate = 4.8372e + 05\ bps$$

$$Compression = 0.3149$$

Η διαδικασία της κωδικοποίησης διήρκησε 52.94 δευτερόλεπτα, ενώ της αποκωδικοποίησης 4.27 δευτερόλεπτα.