ΣΥΣΤΗΜΑΤΑ ΠΟΛΥΜΕΣΩΝ

Νικόλαος Γιακουμόγλου 9043

Ιανουάριος 5, 2021

Abstract

Η εργασία στοχεύει στην υλοποίηση ενός κωδικοποιητή/αποκωδικοποιητή ήχου κατά το πρότυπο Advanced Audio Coding (AAC). Παραλλαγές του AAC χρησιμοποιούνται από πολλά διεθνή πρότυπα όπως τα MPEG-2, MPEG-4, H.264 κλπ. Η εκδοχή που παρουσιάζεται στην εργασία μοιάζει περισσότερο με τις προδιαγραφές 3GPP TS 26.403 όπου απουσιάζουν κάποια στάδια επεξεργασίας. Εξαίρεση αποτελεί το ψυχοακουστικό μοντέλο, που είναι μία λίγο απλουστευμένη εκδοχή του MPEG AAC. Παρόλες τις απλουστεύσεις, η συγκεκριμένη εκδοχή οδηγεί σε αρκετά καλά αποτελέσματα. Η κωδικοποίηση και αποκωδικοποίηση AAC ανήκει στην κατηγορία waveform compression και επιχειρεί να αναπαραστήσει το αρχικό σήμα με τέτοια μορφή ώστε η αποκωδικοποιημένη εκδοχή του να ακούγεται όσο γίνεται πιο όμοια με το αρχικό σήμα. Σαν κριτήριο πιστότητας χρησιμοποιείται το ψυχοακουστικό μοντέλο που επιτρέπει την εισαγωγή παραμορφώσεων του σήματος (θορύβου λόγω κβαντισμού) ο οποίος είναι κάτω από το κατώφλι ακουστότητας. Για το λόγο αυτό κυρίαρχο ρόλο παίζει ο μηχανισμός Psychoacoustic Model που καθοδηγεί το μηχανισμό Quantizer. Για τη μείωση της περίσσειας πληροφορίας ο AAC χρησιμοποιεί κατά βάση την προσέγγιση κωδικοποίησης μετασχηματισμού που υλοποιείται με τη χρήση του λεγόμενου Modifed Discrete Cosine Transform (MDCT) στη βαθμίδα Filterbank ενώ για κωδικοποίηση εντροπίας χρησιμοποιεί κωδικοποίηση Huffman που υλοποιείται στην ομώνυμη βαθμίδα. Αναλυτικότερα, κατά την κωδικοποίηση το αρχικό σήμα ήχου (για μας stereo με δειγματοληψία 48000 samples/sec) χωρίζεται σε επικαλυπτόμενα κατά 50% τμήματα (frames) μήκους 2048 δειγμάτων. Στη συνέχεια κάθε frame κωδικοποιείται αυτόνομα και συνεπώς το τελικά κωδικοποιημένο bitstream αποτελείται από την παράθεση των ακολουθιών bits που αντιστοιχούν στα διαδοχικά frames. Για την επικύρωση της λειτουργικότητας, εφαρμόζεται ο AAC στο τραγούδι *Licor De Calandraca*.

Contents

Part I
AAC Level 1

1 Sequence Segmentation Control ($SSC.m$)

Μετά το χωρισμό των δειγμάτων σε frames επιλέγεται από το μηχανισμό Sequence Segmentation Control ο τύπος του frame. Υπάρχουν δύο βασικοί και δύο συμπληρωματικοί τύποι:

- ONLY_LONG_SEQUENCE (OLS)
- EIGHT_SHORT_SEQUENCE (ESH)
- LONG_START_SEQUENCE (LSS)
- LONG_STOP_SEQUENCE (LPS)

Η επιλογή του τύπου βασίζεται στον τύπο του προηγούμενου και αν το επόμενο frame είναι ESH ή όχι. Αν το προηγούμενο frame είναι LPS ή LSS η απόφαση είναι τετριμμένη και είναι OLS και ESH αντίστοιχα. Αλλοιώς πρέπει να ελέγξουμε το επόμενο frame αν είναι ESH. Αυτό γίνεται ακολουθώντας την παρακάτω μεθοδολογία για κάθε ένα από τα 2 κανάλια:

1. Εφαρμογή φίλτρου $H(z) = \frac{0.7548 - 0.7548z^{-1}}{1 - 0.5095z^{-1}}$ στα δείγματα

```
filter([0.7548,-0.7548],[1,-0.5095],nextFrameT(:,channel))
```

2. Χωρίζουμε το frame σε 8 περιοχές 128 δειγμάτων με50% επικάλυψη και υπολογίζουμε την ενέργεια s_l^2 ως το άθροισμα των τετραγώνων των δειγμάτων της περιοχής και τα attack values ως $ds_l^2 = \frac{s_l^2}{\frac{1}{l-1} \cdot \sum_{m=0}^{l-1} s_m^2}$

```
energy = zeros(8,1);
for l=1:8
    energy(l) = sum(filteredNextFrameT(((l-1)*256+1):(l*256)).^2);
end
attackvalues = zeros(8, 1);
for l = 2:8
    attackvalues(l) = l*energy(l)/sum(energy(1:(l-1)));
end
```

3. Κατάταξη του frame ως ESH αν $\exists l \in \{0, 1, 2, ..., 7\} : s_l^2 > 10^{-3}$ και $ds_l^2 > 10$

```
if sum(energy>10^(-3))>0 && sum(attackvalues>10))
```

Η παραπάνω διαδικασία επαναλαμβάνεται για κάθε κανάλι. Κατόπιν έχουμε τα εξής πιθανά σενάρια:

- Αν το προηγούμενο είναι OLS και το επόμενο ESH τότε το κανάλι είναι τύπου LSS.
- Αν το προηγούμενο είναι OLS και το επόμενο δεν είναι ESH τότε το κανάλι είναι τύπου OLS.
- Αν το προηγούμενο είναι ESH και το επόμενο ESH τότε το κανάλι είναι τύπου ESH.
- Αν το προηγούμενο είναι ESH και το επόμενο δεν είναι ESH τότε το κανάλι είναι τύπου LPS.

Λογικά θα ισχύει κάτι από τα προηγούμενα εκτός της 1ης επανάληψης όπου ο προηγούμενος τύπος είναι κενός. Τότε ικανοποιείται μια τελευταία συνθήκη που κατατάσσει το κανάλι ως τύπο LSS αν το επόμενο είναι ESH, αλλοιώς ως το κατατάσσει ως OLS. Έτσι λαμβάνουμε 2 πιθανές τιμές για κάθε κανάλι και η τελική απόφαση λαμβάνεται ακολουθώντας το Table 1. Σημειώνεται ότι το τωρινό frame δεν χρησιμοποιείται άμεσα στην συνάρτηση αλλά μέσω της αναδρομής στι AACoder χρησιμοποιείται ως το επόμενο frame.

Τύπος καναλιού 0	Τύπος καναλιού 1	Κοινός τελικός τύπος
ONLY_LONG_SEQUENCE	ONLY_LONG_SEQUENCE	ONLY_LONG_SEQUENCE
ONLY_LONG_SEQUENCE	LONG_START_SEQUENCE	LONG_START_SEQUENCE
ONLY_LONG_SEQUENCE	EIGHT_SHORT_SEQUENCE	EIGHT_SHORT_SEQUENCE
ONLY_LONG_SEQUENCE	LONG_STOP_SEQUENCE	LONG_STOP_SEQUENCE
LONG_START_SEQUENCE	ONLY_LONG_SEQUENCE	LONG_START_SEQUENCE
LONG_START_SEQUENCE	LONG_START_SEQUENCE	LONG_START_SEQUENCE
LONG_START_SEQUENCE	EIGHT_SHORT_SEQUENCE	EIGHT_SHORT_SEQUENCE
LONG_START_SEQUENCE	LONG_STOP_SEQUENCE	EIGHT_SHORT_SEQUENCE
EIGHT_SHORT_SEQUENCE	ONLY_LONG_SEQUENCE	EIGHT_SHORT_SEQUENCE
EIGHT_SHORT_SEQUENCE	LONG_START_SEQUENCE	EIGHT_SHORT_SEQUENCE
EIGHT_SHORT_SEQUENCE	EIGHT_SHORT_SEQUENCE	EIGHT_SHORT_SEQUENCE
EIGHT_SHORT_SEQUENCE	LONG_STOP_SEQUENCE	EIGHT_SHORT_SEQUENCE
LONG_STOP_SEQUENCE	ONLY_LONG_SEQUENCE	LONG_STOP_SEQUENCE
LONG_STOP_SEQUENCE	LONG_START_SEQUENCE	EIGHT_SHORT_SEQUENCE
LONG_STOP_SEQUENCE	EIGHT_SHORT_SEQUENCE	EIGHT_SHORT_SEQUENCE
LONG_STOP_SEQUENCE	LONG_STOP_SEQUENCE	LONG_STOP_SEQUENCE

Table 1: Απόφαση βάσει τύπο κάθε καναλιού

2 Filterbanks ($filterbanks.m$, $iFilterbanks.m$)

Η πληροφορία του τύπου κάθε frame διαβιβάζεται στο μηχανισμό Filterbanks που χρησιμοποιεί το μετασχηματισμό Modified Discrete Cosine Transform ($MDCT$) για να μειώσει τη συσχέτιση των δειγμάτων μεταβαίνοντας ταυτόχρονα από το πεδίο του χρόνου στο πεδίο της συχνότητας. Πριν την εφαρμογή του $MDCT$, καθώς και μετά την εφαρμογή του αντίστροφου, $IMDCT$, τα δείγματα πολλαπλασιάζονται στο πεδίο του χρόνου με παράθυρο οι συντελεστές του οποίου ικανοποιούν συγκεκριμένες συνθήκες που επιτρέπουν την ανακατασκευή του σήματος στο αποκωδικοποιητή. Δύο είδη συναρτήσεων παραθύρων προδιαγράφονται στο πρότυπο: Τα παράθυρα Kaiser-Bessel-Derived (KBD) και sinusoid (SIN). Ακολουθώντας το αρχείο $w2203tfa.pdf$ σελ. $127-132$ κατασκευάζουμε το παράθυρο ανάλογα με τον τύπου του frame. Το $MATLAB$ έχει προσθέσει το παράθυρα KBD στην τελευταία ενημέρωση ($2020b$) οπότε τα χρησιμοποιούμε έτοιμα. Τα παράθυρα SIN παράγονται πολύ εύκολα. Ετσι για παράδειγμα αν θέλουμε να φτιάξουμε ένα παράθυρο μήκους 2048 με παράμετρο $a = 6$ για τα KBD εκτελούμε τις παρακάτω εντολές για κάθε ένα από τα δύο είδη:

```
W_KBD = kbdwin(2048,6)
WL_KBD = W_KBD(1:1024);
WR_KBD = W_KBD(1025:end);

WL_SIN = sin(pi/2048*([0:1023]+0.5))';
WR_SIN = sin(pi/2048*([1024:2047]+0.5))';
W_SIN = [WL;WR];
```

Ο μετασχηματισμός του frame από τον χρόνο διαστάσεων 2048×2 στην συχνότητα διαστάσεων 1024×2 προκύπτει ανάλογα με τον τύπου του frame. Η διαφορά έγγειται σε διαφορετική επιλογή παραθύρου:

- Για OLS το παράθυρο είναι συμμετρικό με 2048 δείγματα και $a = 6$. Εφαρμόζουμε $MDCT$ στο εσωτερικό γινόμενο του παραθύρου μας και του frame για κάθε channel με την συνάρτηση $mdct$ του MATLAB
- Για LSS το παράθυρο περιλαμβάνει ένα αριστερό παράθυρο με 2048 δείγματα και $a = 6$, 448 άσσους, ένα δεξί παράθυρο με 256 δείγματα και $a = 4$ και τέλος 448 μηδενικά. Εφαρμόζουμε $MDCT$ στο εσωτερικό γινόμενο του παραθύρου μας και του frame για κάθε channel με την συνάρτηση $mdct$ του MATLAB

- Για ESH το παράθυρο είναι συμμετρικό με 256 δείγματα και $a = 4$. Εφαρμόζουμε $MDCT$ στο εσωτερικό γινόμενο του παραθύρου μας και 256 δειγμάτων του frame, μετακινώντας κάθε φορά τα δείγματα κατά 128 ώστε να υπάρχει επικάλυψη 50%. Να σημειωθεί ότι αγνοούμε τα πρώτα και τελευταία 448 δείγματα, δηλαδή παίρνουμε μόνο τα 1152 κεντρικά. Το μεταχηματισμένο frame αποθηκεύεται σε πίνακα $128 \times 2 \times 8$ (κατόπιν θα διαπιστώσουμε ότι ηταν προτιμότερη μια επιλογή $128 \times 8 \times 2$). Προσοχή ότι όταν πάρουμε τον αντίστροφο μετασχηματισμό MDCT, πρέπει να λάβουμε το επικαλυπτόμενο τμήμα 2 φορές υπ'όψη μας.

- Για LPS το παράθυρο περιλαμβάνει 448 μηδενικά, ένα αριστερό παράθυρο με 256 δείγματα και $a = 4$, 448 άσσους και ένα δεξί παράθυρο με $N = 2048$ δείγματα και $a = 6$.Εφαρμόζουμε $MDCT$ στο εσωτερικό γινόμενο του παραθύρου μας και του frame για κάθε channel με την συνάρτηση *mdct* του MATLAB

Τονίζουμε ότι οι $MDCT$ και $IMDCT$ υπάρχουν υλοποιημένοι σε $MATLAB2020b$.
Οταν επιχειρήθηκε υλοποίηση του, ήταν πολύ αργή γι'αυτό προτιμήθηκε η καινούρια έκδοση του $MATLAB$. Η παράμετρος $'PadInput'$ τίθεται $false$ οπότε δεν πετυχαίνεται τέλεια ανακατασκευή του αρχικού σήματος. Αυτό οφείλεται στο γεγονός ότι η παράμετρος αυτή δεν κάνει zero padding στα άκρα με αποτέλεσμα να εμφανίζει "πρόβλημα" ο μετασχηματισμός. Για την αντίστροφη διαδικασία δημιουργούνται με τον ίδιο ακριβώς τρόπο τα παράθυρα και καλούμε τον αντίστροφο μετασχηματισμό *imdct*.
Για παράδειγμα, αφού δημιουργήσουμε ένα παράθυρο W, για να βρούμε τον $MDCT$ και $IMDCT$ εκτελούμε τις παρακάτω εντολές:

```
frameF = reshape(mdct(frameT,W,'PadInput',false),[1024  2]);
frameT(:,1) = imdct(frameF(:,1),W,'PadInput',false);
frameT(:,2) = imdct(frameF(:,2),W,'PadInput',false);
```

Η εντολή reshape προστίθεται γιατί ο mdct επιστρέφει $1024 \times 1 \times 2$.

3 Part I Final ($AACoder1.m$, $iAACoder1.m$, $demoAAC1.m$)

Εχοντας στην διάθεση μας τα παραπάνω, η κατασκευή το coder - $AACoder1$ δεν είναι τιποτα άλλο παρά εφαρμογή SSC και $Filterbank$:

1. Εφαρμόζουμε zero padding στο αρχικό σήμα προσθέτοντας 2048 δείγματα στην αρχή και 2048 στο τέλος συν μερικά ακόμα για να συμπληρώσουμε ακέραιο αριθμό από frames (1024 ηταν αρκετά, αλλά με 2048 είχαμε καλύτερα αποτελέσματα). Τα επιπλέον δείγματα υπολογίζονται ως $1024 - mod(1024, N_0)$ όπου N_0 το αρχικό μήκος του σήματος. Υπολογίζουμε τον αριθμό των frames ως $\frac{N}{1024} - 1$ όπου N το μήκος του αρχείου ήχου αφού εφαρμόσουμε zero padding και προσθέσουμε τα επιπλέον δείγματα στο τέλος. Ετσι ο αριθμός των frames είναι ακέραιος.

2. Βρίσκουμε το εκάστοτε frame στον χρόνο και το επόμενό του

3. Υπολογίζουμε για κάθε frame τον τύπο του με την συνάρτηση SSC

4. Υπολογίζουμε τον $MDCT$ με την συνάρτηση $filterbanks$ με ιδιαίτερη μνεία αν έχουμε τύπο ESH όπου απαιτείται μία αλλαγή του τύπου του frame στην συχνότητα από $128 \times 2 \times 8$ που είναι η έξοδος του $filterbank$ σε 128×8 για κάθε κανάλι που είναι η επιθυμητή διάσταση για τον την έξοδο του coder.

Σημειώνεται πως ο κυρίος βρόγχος τρέχει $N - 1$ φορές καθώς για το N-οστό frame δεν μπορούμε να πάρουμε το επόμενο για τον υπολογισμό του τύπου του στην συνάρτηση SSC.
Στον decoder - $iAACoder1$ ακολουθούμε την αντίστροφη διαδικασία:

1. Βρίσκουμε το frame στον χρόνο μέσω $iFilterbank$ με ιδιαίτερη μνεία και πάλι αν έχουμε τύπο ESH που μετατρέπουμε τα δείγματα στον χρόνο από 128×8 για κάθε κανάλι σε $128 \times 2 \times 8$. Σημειώνεται ότι στην ανακατασκευή του αρχικού σήματος, λαμβάνουμε την επικαλυπτόμενη περιοχή 2 φορές.

Πριν επιστρέψουμε το σήμα, μηδενίζουμε τιμές που είναι NaN ή άπειρο και θέτουμε ως 1 και -1 τις τιμές που είναι μεγαλύτερες του 1 ή μικρότερες του -1 αντίστοιχα.
Η επίδειξη $demoAAC1$ περιλαμβάνει την εφαρμογή $AACoder1$ και $iAACoder1$ διαδοχικά. Το αρχείο $demoAAC1$ εκτελείται σε < 1 δευτερόλεπτα (i7-6700HQ). Ο σηματοθορυβικός λόγος είναι ~ 285dB για το κανάλι 1 και

$\sim$ 284db για το κανάλι 2. Μάλιστα αν χρησιμοποιήσουμε τον $MDCT - IMDCT$ του Marios Athineos (https://www.ee.columbia.edu/~marios/mdct/mdct_giraffe.html) που υπάρχει στο διαδίκτυο, τα αποτελέσματα είναι ακόμα καλύτερα. ($\sim$ 300dB). Το plot των κυματομορφών για κάθε κανάλι πριν την εφαρμογή του AAC και μετά φαίνεται να ταυτίζονται (Figure 1). Στο Figure 2 βλέπουμε το plot του σφάλματος. Το μικρό σφάλμα της τάξης του 10^{-14} οφείλεται στο γεγονός ότι ο $MDCT$ δεν είναι $1-1$ με τον $IMDCT$ (το οποίο φαίνεται αν το δοκιμάσουμε για ένα τυχαίο frame). Ωστόσο το σφάλμα κανονικά έπρεπε να είναι μηδενικό καθώς το στάδιο αυτό δεν εισάγει παραμόρφωση αλλά κάνει αποσυσχέτιση.

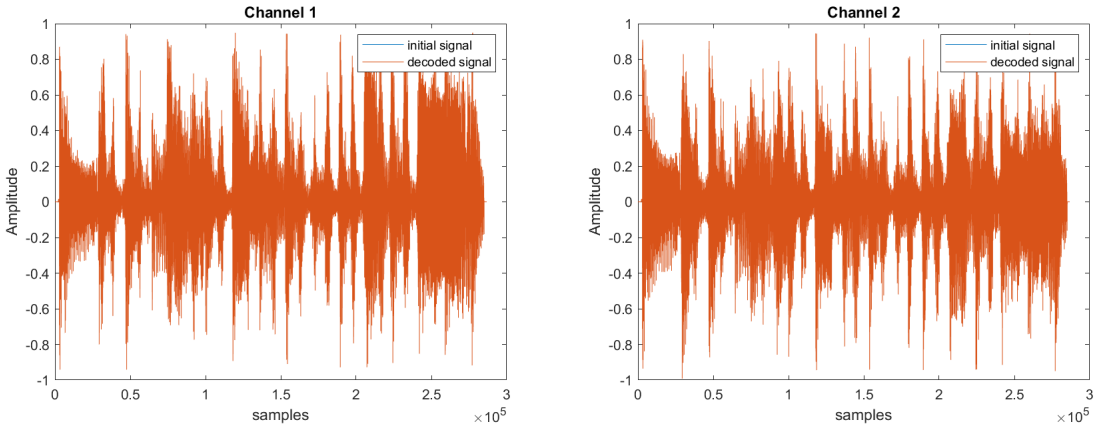

Figure 1: Intitial (blue) and decoded signal (red)

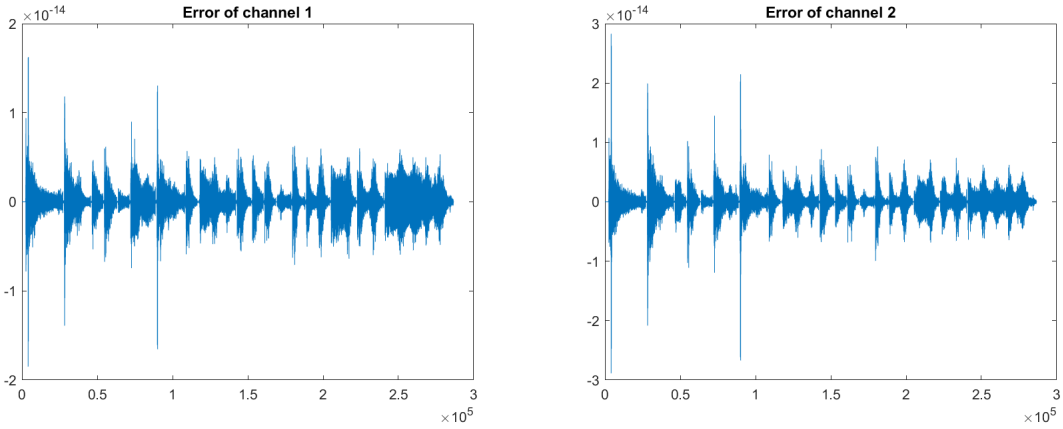

Figure 2: Error of channel 1 and 2

Part II
AAC Level 2

4 Temporal Noise Shaping ($TNS.m$, $iTNS.m$)

Ο μηχανισμός Temporal Noise Shaping (TNS) μετασχηματίζει τους συντελεστές $MDCT$ σε ένα νέο σύνολο ισάριθμων συντελεστών στο οποίο έχουν απαλοιφθεί οι περιοδικότητες. Θεωρούμε τις μπάντες του ψυχοακουστικού μοντέλου που ορίζονται στους πίνακες Table $B.2.1.9.a$ και $TableB.2.1.9.b$ των σελίδων $117-119$ του προτύπου (αρχείο $w2203tfa$) τους οποίους φορτώνουμε από το αρχείο $TableB219.mat$. Επιλέγουμε την στήλη 2 που αναφέρεται στο w_low από τον πίνακα B.2.1.9.b αν έχουμε ESH, αλλοιώς από τον πίνακα $B.2.1.9.a$ (που ονομάζουμε b για να τηρούμε τους συμβολισμούς).. Η βαθμίδα TNS εφαρμόζει την ακόλουθη διαδικασία:

1. Κανονικοποιεί τους συντελεστές $MDCT$CT X ως προς την ενέργεια της μπάντας στην οποία ανήκουν $X_w(k) = \frac{X(k)}{S_w(k)}$

```
Xw = X./Sw;
```

   Ο συντελεστής κανονικοποίησης $S_w(k)$ υπολογίζεται από την ενέργεια $P(j)$ της κάθε μπάντας j, όπου Nb ο αριθμός από μπάντες $P(j) = \sum_{k=b_j}^{b_{j+1}-1} X^2(k)$ με προσοχή στο γεγονός ότι το $MATLAB$ ξεκινάει αρίθμηση από το 1

```
P = zeros(Nb-1,1);
for j = 1:Nb-1
    P(j) = sum(X((b(j)+1):b(j+1)).^2);
end
```

   Έτσι $S_w(k) = \sqrt{P(j)}, b_j \leq k \leq b_{j+1}$

```
Sw = zeros(1024, 1);
for j = 1:(Nb-1
    Sw((b(j)+1):b(j+1)) = sqrt(P(j));
end
```

   Οι συντελεστές $S_w(k)$ που προκύπτουν με αυτό τον τρόπο έχουν σταθερή τιμή στο εσωτερικό κάθε μπάντας και αλλάζουν απότομα όταν μετακινούμαστε ανάμεσα σε μπάντες. Πριν λοιπόν εφαρμοστούν, εξομαλύνονται ως ακολούθως

```
for k = (length(Sw)-1):-1:1
    Sw(k) = (Sw(k) + Sw(k+1))/2;
end
for k = 2:1:(length(Sw))
    Sw(k) = (Sw(k) + Sw(k-1))/2;
end
```

2. Υπολογίζει τους συντελεστές γραμμικής πρόβλεψης $a = [a_1, a_2, a_3, a_4]^T$ για κάθε frame ή subframe αφού υπολογίσουμε τους πίνακες R, r με την βοήθεια της εντολής $corrmtx$ που υπολογίζει τον autocorrelation matrix

```
[~, H] = corrmtx(Xw, 4);
r = H(2 : 5, 1);
R = H(1 : 4, 1 : 4);
a = (R\r)';
```

3. Κβαντίζει τους παραπάνω συντελεστές με 4 bits (άρα $2^4 = 16$ στάθμες) χρησιμοποιώντας ομοιόμορφο συμμετρικό κβαντιστή βήματος 0.1. Για τον κβαντιστή χρησιμοποιούμε το παρακάτω trick:

```
step = 0.1;
bits = 4;
range = step*(2^bits - 1);
a_min = -range/2;
a_max = range/2;
a(a > a_max) = a_max;
a(a < a_min) = a_min;
a = round(a/(step))*step;
```

4. Εφαρμόζει το FIR φίλτρο $H_{TNS}(z) = 1 - a_1 z^{-1} - a_2 z^{-2} - a_3 z^{-3} - a_4 z^{-4}$. Σε περίπτωση που το φίλτρο δεν είναι ευσταθές (ελέγχεται μέσω της συνάρτησης *isstable* του *MATLAB*) εμφανίζεται warning.

```
filter([1 -a'], 1, X)
```

Οι συντελεστές TNS είναι το a που υπολογίσαμε και μετασχηματισμένοι συντελεστές $MDCT$ είναι η έξοδος του φίλτρου. Για τον αντίστροφο TNS - $iTNS$ εφαρμόζουμε το φίλτρο $\frac{1}{H_{TNS}(z)}$ και παίρνουμε τους αρχικούς συντελεστές $MDCT$. Η διαφορά στα ESH παράθυρα είναι ότι επαναλαμβάνουμε την διαδικασία 8 φορές.

5 Part II Final ($AACoder2.m$, $iAACoder2.m$, $demoAAC2.m$)

Στον $AACoder2$ ακολουθούμε την ίδια διαδικασία με τον $AACoder1$ με την διαφορά ότι μετασχηματίζουμε το frame στην συχνότητα με το TNS για να απαλοιφθούν οι περιοδικότητες. Συγκεκριμένα

1. Εφαρμόζουμε zero padding στο αρχικό σήμα προσθέτοντας 2048 δείγματα στην αρχή και 2048 στο τέλος συν μερικά ακόμα για να συμπληρώσουμε ακέραιο αριθμό από frames όπως στο Part I Final

2. Βρίσκουμε το εκάστοτε frame στον χρόνο, το επόμενό του καθώς και το προηγούμενο και το προ-προηγούμενο frame του εκάστοτε frame (τωρινό). Σημειώνεται πως αν το τωρινό frame είναι το 1ο, το προηγούμενο και το προ-προηγούμενο θεωρούνται μηδέν, όπως και αν το τωρινό frame είναι το 2ο, το προ-προηγούμενο θεωρείται μηδέν.

3. Υπολογίζουμε για το τωρινό frame τον τύπο του με την συνάρτηση SSC

4. Υπολογίζουμε τον $MDCT$ με την συνάρτηση $filterbanks$

5. Εφαρμόζουμε TNS με όρισμα την έξοδο του $filterbanks$ για να πάρουμε το μετασχηματισμένο frame στην συχνότητα μαζί με τους συντελεστές TNS

Στον $iAACoder2$, εκτελούμε την αντίστροφη διαδικασία:

1. Περνάμε το μετασχηματισμένο frame στην συχνότητα του $AACSeq2$ από το $iTNS$ για να πάρουμε το αρχικό frame στην συχνότητα

2. Βρίσκουμε το frame στον χρόνο μέσω $iFilterbank$

Πριν επιστρέψουμε το σήμα, μηδενίζουμε τιμές που είναι NaN ή άπειρο και θέτουμε ως 1 και -1 τις τιμές που είναι μεγαλύτερες του 1 ή μικρότερες του -1 αντίστοιχα.
Η επίδειξη $demoAAC2$ περιλαμβάνει την εφαρμογή $AACoder2$ και $iAACoder2$ διαδοχικά. Το αρχείο $demoAAC2$ εκτελείται μόλις σε 2 δευτερόλεπτα (i7-6700HQ). Η έξοδος αποθηκεύεται σε νέο αρχείο ήχου. Πράγματι αν ακούσουμε το νέο αρχείο ήχου δεν μπορούμε να καταλάβουμε διαφορά. Το plot των κυματομορφών για κάθε κανάλι φαίνεται να ταυτίζονται (Figure 3). Στο Figure 4 βλέπουμε το plot του σφάλματος. Το μικρό σφάλμα της τάξης του 10^{-14} οφείλεται στο γεγονός ότι ο $MDCT$ δεν είναι $1 - 1$ με τον $IMDCT$ (βλ. Κεφάλαιο 3). Ο σηματοθορυβικός λόγος είναι 285dB για το κανάλι 1 και 284dB για το κανάλι 2. Ο σηματοθορυβικός λόγος

ειναι ίδιος με το Part I καθώς η διαφορά στο Part II είναι η εφαρμογή ενός φίλτρου στο frame στην συχνότητα που στην πλειονότητα των περιπτώσεων είναι αντιστρέψιμο οπότε αναμένουμε ίδιο SNR.

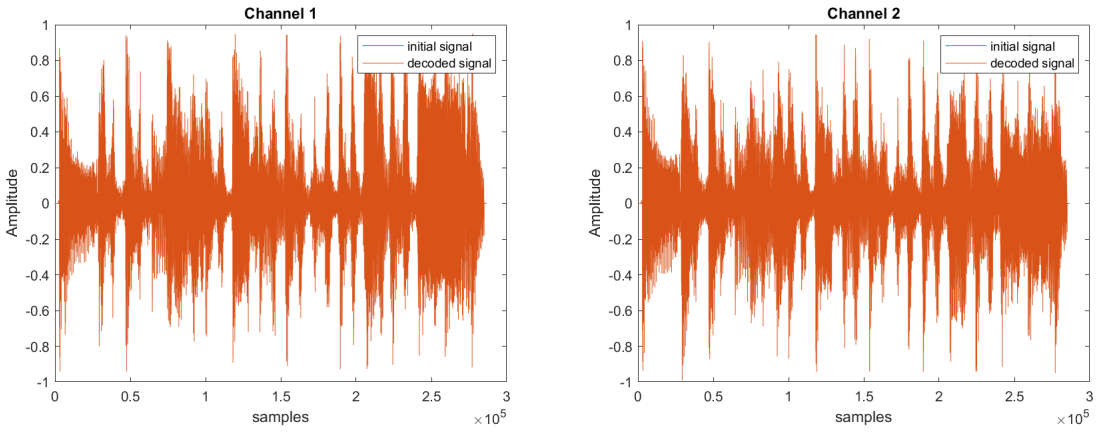

Figure 3: Intitial (blue) and decoded signal (red)

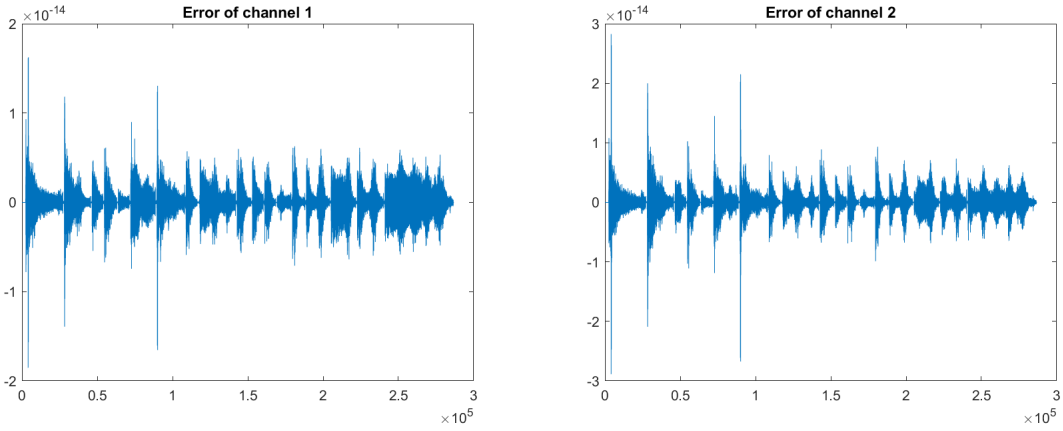

Figure 4: Error of channel 1 and 2

Part III
AAC Level 3

6 Psychoacoustic Model (*psycho.m*)

Για τη μείωση των απαιτούμενων bit κατά την κβάντιση, χρησιμοποιείται το ανθρώπινο ψυχοακουστικό μοντέλο. Αρχικά φορτώνουμε τους πίνακες και $B.2.1.9.b$ για ESH παράθυρο ή τους πίνακες $B.2.1.9.a$ για τα υπόλοιπα είδη. Σε κάθε επανάληψη υπολογίζουμε την spreading function ως ακολούθως (σημειώνεται πως μόνο 2 φορές χρειάζεται ο υπολογισμός της κανονικά, 1 για ESH και 1 για τα υπόλοιπα παράθυρα)

```
for i = 1:Nb
    for j = 1:Nb
        if i >= j
            tmpx = 3*(bval(j)-bval(i));
        else
            tmpx = 1.5*(bval(j)-bval(i));
        end
        tmpz = 8*min(0,(tmpx-0.5)^2-2*(tmpx-0.5));
        tmpy = 15.811389+7.5*(tmpx+0.474)-17.5*...
               (1+(tmpx+0.474)^2)^(0.5);
        if tmpy < -100
            spreadingfunction(i,j) = 0;
        else
            spreadingfunction(i,j) = 10^((tmpz+tmpy)/10);
        end
    end
end
```

Επειτά η μέθοδος βασίζεται στο εκάστοτε frame και τα 2 προηγούμενά του. Οι υπολογισμοί είναι οι ίδιοι για ESH, OLS, LSS, LPS αλλά στα ESH παράθυρα οι ακόλουθοι υπολογισμοί γίνονται 8 φορές. Μάλιστα στην περίπτωση του ESH, χωρίζουμε το κάθε frame σε 8 subframes και παίρνουμε για το εκάστοτε subframe τα 2 προηγούμενα subframes ώστε τελικά το προ-προηγούμενο frame να μην χρειάζεται καν. Θα παραθέσουμε κώδικα για λογική OLS/LSS/LPS. Για τα ESH κάνουμε ίδιο 8 φορές. Από εδώ και πέρα όταν κάνουμε υπολογισμούς για κάποιο frame/subframe, θα το αναθέτουμε στην μεταβλήτή s.

1. Πολλαπλασιαμός του frame/subframe $s(n)$ με παράθυρο Hann στο πεδίο του χρόνου $w_{Hann}(t) = 0.5 - 0.5 \cdot cos\left(\frac{\pi(n+0.5)}{N}\right)$: $s_w(n) = s(n) \cdot w_{Hann}(t)$

   ```
   sw = s.*(0.5 - 0.5*cos(pi/2048*([1:2048]-0.5)))';
   ```

2. Μετασχηματισμός Fourier του $S_w(\omega) = \mathcal{F}\{s_w(n)\}$ με αποκοπή των μισών δειγμάτων (duplicates), και χωρισμός σε μέτρο $r(\omega)$ και φάση $f(\omega)$. Ο πίνακας r και f έχουν διαστάσεις 1024×3 όπου στην 1η στήλη αποθηκεύονται τα r, f των τωρινού frame, στην 2η του προηγούμενου και στην 3η του προ-προηγούμενου. Επαναλαμβάνουμε την ακόλουθη διεργασία 3 φορές

   ```
   Sw = fft(sw);
   r(:,i) = abs(Sw(1:1024));
   f(:,i) = angle(Sw(1:1024));
   ```

3. Υπολογισμός πρόβλεψης για το μέτρο r_pred και την φάση f_pred ως εξής

   ```
   r_pred = 2*r(:,2) - r(:,3);
   f_pred = 2*f(:,2) - f(:,3);
   ```

4. Υπολογισμός μέτρου της προβλεψιμότητας c όπου $r = r(:,1), f = f(:,1)$

```
C = sqrt((r.*cos(f)-r_pred.*cos(f_pred)).^2+...
    (r.*sin(f)-r_pred.*sin(f_pred)).^2)./(r+abs(r_pred));
```

5. Υπολογισμός ενέργειας $e(b) = \sum_{\omega=w_{low}(b)}^{\omega_{high}(b)} r^2(\omega)$
   και επιβαρυμένης προβλεψιμότητας $c(b) = \sum_{\omega=\omega_{low}(b)}^{\omega_{high}(b)} c(\omega) \cdot r^2(\omega)$

```
e = zeros(Nb,1);
c = zeros(Nb,1);
for b = 1:Nb
    e(b) = sum(r((w_low(b)+1):(w_high(b)+1),:).^2);
    c(b) = sum(C((w_low(b)+1):(w_high(b)+1),:).*...
               r((w_low(b)+1):(w_high(b)+1),:).^2);
end
```

6. Συνδυασμός ενέργειας και προβλεψιμότητας με spreading function
   $ecb(b) = \sum_{bb=0}^{N_b-1} e(bb) \cdot spreading\_function(bb,b)$ και
   $ct(b) = \sum_{bb=0}^{N_b-1} c(bb) \cdot spreading\_function(bb,b)$
   και κανονικοποίηση $cb(b) = \frac{ct(b)}{ecb(b)}$ και $en(b) = \frac{ecb(b)}{\sum_{bb=0}^{N_b-1} spreading\_function(bb,b)}$

```
ecb = (e'*spreadingfunction)';
ct = (c'*spreadingfunction)';
cb = ct ./ ecb;
en = zeros(Nb,1);
for b = 1:Nb
    en(b) = ecb(b) / (sum(spreadingfunction(:,b)));
end
```

7. Υπολογισμός δείκτη τονικότητας $tb(b) = -0.299 - 0.43ln(cb(b))$ που επειδή πρέπει να ανήκει στο διάστημα $[0,1]$, όσες τιμές είναι κάτω του 0, τις μηδενίζουμε ενώ όσες είναι πάνω του 1, τις θέτουμε ίσες με 1

```
tb = -0.299 - 0.43 * log(cb);
tb(tb>1) = 1;
tb(tb<0) = 0;
```

8. Υπολογίζουμε $SNR(b) = 18 \cdot tb(b) + 6 \cdot (1 - tb(b))$

```
SNR = tb*18 + 6*(1-tb);
```

9. Μετατροπή από dB σε λόγο ενέργειας $bc(b) = 10^{-\frac{SNR(b)}{10}}$

```
bc = 10.^(-SNR/10);
```

10. Υπολογισμός κατωφλίου ενέργειας $nb(b) = en(b) \cdot bc(b)$

```
nb = en .* bc;
```

11. Υπολογισμός του $\hat{q}_{thr} = \varepsilon \cdot \frac{N}{2} \cdot 10^{q_{thr}/10}$ και υπολογισμός μέσω αυτού του επιπέδου θορύβου $npart(b) = max\{nb(b), \hat{q}_{thr}\}$

```
qthr_est = eps()*(2048/2)*10.^(qsthr/10);
npart = max([nb,qthr_est]')';
```

12. Υπολογισμός Signal to Mask Ratio $SMR(b) = \frac{e(b)}{npart(b)}$

```
SMR = e ./ npart;
```

Για τα ονόματα κρατήσαμε $1-1$ όλους τους συμβολισμούς.

7 Quantizer (*AACquantizer.m*, *iAACquantizer.m*)

Ο κβαντιστής παραλαμβάνει τους συντελεστές της βαθμίδας TNS και τα κατώφλια $T(b)$ της κάθε scalefactor band b και παράγει σύμβολα που στη συνέχεια θα κωδικοποιηθούν από τη βαθμίδα Huffman. Αρχικά φορτώνουμε τους πίνακες και $B.2.1.9.b$ για ESH παράθυρο ή τους πίνακες $B.2.1.9.a$ για τα υπόλοιπα είδη. Οι υπολογισμοί είναι οι ίδιοι για ESH, OLS, LSS, LPS αλλά στα ESH παράθυρα οι ακόλουθοι υπολογισμοί γίνονται 8 φορές. Θα παραθέσουμε κώδικα για λογική OLS/LSS/LPS. Από εδώ και πέρα όταν κάνουμε υπολογισμούς για κάποιο frame/subframe, θα το αναθέτουμε στην μεταβλήτή X.

1. Υπολογισμός $P(b) = \sum_{k=\omega_{low}(b)}^{\omega_{high}(b)} X^2(k)$

```
P = zeros(Nb, 1);
for b = 1:Nb
    P(b) = sum(X((w_low(b)+1):(w_high(b)+1), :).^2, 1);
end
```

2. Υπολογισμός των κατωφλίων ακουστότητας $T(b) = \frac{P(b)}{SMR(b)}$

```
T = P ./ SMR;
```

3. Υπολογισμός Scalefactor gain $a(b)$

   (a) Αρχικοποίηση ως $a(b) = 16/3 log_2\left(\frac{max(X(k))^{3/4}}{8191}\right)$

```
a = zeros(Nb,1);
for b = 1:Nb
    a(b) = round((16/3)*log2((max(X)^(3/4))/8191));
end
```

   (b) Επαναληπτικά αυξάνουμε κατά 1 μονάδα και ελέγχουμε αν της ισχύ του σφάλματος κβαντισμού $P_e(b) = \sum_{k=\omega_{low}(b)}^{\omega_{high}(b)} \left(X(k) - \widetilde{X}(k)\right)^2$. Αν η ισχύς είναι κάτω από το κατώφλι ακουστότητας, τότε αυξάνουμε τον αντίστοιχο scale factor $a(b)$ κατά 1. Η διαδικασια επαναλαμβάνεται έως ώτου φτάσουμε το κατώφλι $T(b)$. Επιπλέον, σταματάμε τη διαδικασία αύξησης του $a(b)$ αν $max\{a(b+1) - a(b)\} > 60$ (επομένως η μέγιστη διαφορά διαδοχικών scalefactors πρέπει να είναι μέχρι 60). Πιο αναλυτικά ξεκινάμε από όλες τις μπάντες N_b και υπολογίζουμε για κάθε μπάντα τις τιμές του $S(k), \widetilde{X}(k)$ για $k \in [\omega_{low}(b), \omega_{high}(b)]$ ώστε τελικά να βρούμε την ισχύ του σφάλματος κβαντισμού $P_e(b)$ της μπάντας b . Αν η ισχύς αυτή δεν ξεπέρασε το κατώφλι $T(b)$ τότε αυξάνουμε το αντίστοιχο $a(b)$ της μπάντας κατά 1. Επαναλαμβάνουμε μέχρι να μην μπορούμε να αυξήσουμε άλλο το a.

```
Pe = zeros(Nb, 1);
idx = 1:Nb;
X_hat = zeros(size(X));
while(true)
if max(abs(a(2:end) - a(1:end-1))) > 60
        break;
end
for b = idx
        Pe(b) = 0;
    for k=(w_low(b)+1):(w_high(b)+1)
        S(k) = sign(X(k)).*fix((abs(X(k)).*2.^(-1/4*a(b))).^(3/4)+0.4054);
        X_hat(k) = sign(S(k)).*(abs(S(k)).^(4/3)).*2.^((1/4)*a(b));
    end
    Pe(b) = sum((X([(w_low(b)+1):(w_high(b)+1)]) -...
        X_hat([(w_low(b)+1):(w_high(b)+1)])).^2);
end
idx = (find(Pe < T))';
a(idx) = a(idx) + 1;
end
```

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

και $\widetilde{X}(k) = sgn(S(k))|S(k)|^{4/3} \cdot 2^{\frac{1}{4}a}$

(c) Θέτουμε $G = sfc(0) = a(0)$ και $sfc(b) = \Delta a(b) = a(b) - a(b-1)$ (DPCM)

```
sfc = a;
G = a(1);
for b = 2:Nb
    sfc(b) = a(b)-a(b-1);
end
```

Για τα ονόματα κρατήσαμε $1-1$ όλους τους συμβολισμούς. Στον αποκβαντιστή *iAACquantizer* πρέπει να βρούμε το $a(b)$ για να ανακατασκευάσουμε το σήμα μέσω IDPCM, το οποίο προκύπτει ως εξής

```
a = zeros(Nb,1);
for b = 2:Nb
    a(b) = sfc(b) + a(b-1);
end
a = a + G;
```

8 Part III Final (*AACoder3.m*, *iAACoder3.m*, *demoAAC3.m*)

Στον *AACoder3* ακολουθούμε την ίδια διαδικασία με τον *AACoder2* με την διαφορά ότι προσθέτουμε το ψυχοακουστικό μοντέλο, τον κβαντιστή και τον κωδικοποιητή εντροπίας. Συγκεκριμένα:

1. Εφαρμόζουμε zero padding στο αρχικό σήμα προσθέτοντας 2048 δείγματα στην αρχή και 2048 στο τέλος συν μερικά ακόμα για να συμπληρώσουμε ακέραιο αριθμό από frames όπως στο Part I Final
2. Βρίσκουμε το εκάστοτε frame στον χρόνο, το επόμενό του καθώς και το προηγούμενο και το προ-προηγούμενο frame του εκάστοτε frame (τωρινό) όπως στο Part II Final
3. Υπολογίζουμε για το τωρινό frame τον τύπο του με την συνάρτηση *SSC*
4. Υπολογίζουμε τον *MDCT* με την συνάρτηση *filterbanks*

5. Εφαρμόζουμε TNS με όρισμα την έξοδο του $filterbanks$ για να πάρουμε το μετασχηματισμένο frame στην συχνότητα μαζί με τους συντελεστές.

6. Εφαρμόζουμε το ψυχοακουστικό μοντέλο για να πάρουμε το Signal to Mask Ratio SMR

   (a) Υπολογίζουμε το acoystic threshold T για να το σώσουμε στο $AASeq3$ (το οποίο υπολογίζεται και στον quantizer αλλά δεν επιστρέφεται)

7. Κβαντίζουμε το μετασχηματισμένο frame στην συχνότητα βάσει του SMR με τον AAC quantizer παράγοντας τα σύμβολα κβάντισης

8. Χρησιμοποιούμε τον έτοιμα υλοποιημένο κωδικοποιητή εντροπίας $Huffman$ στο sfc και στα σύμβολα κβάντισης για να πάρουμε μια ακολουθία από bits

Στον $iAACoder3$, εφαρμόζουμε την αντίστροφη διαδικασία:

1. Εφαρμόζουμε τον αποκωδικοποιητή $Huffman$ για να πάρουμε τα sfc και το μετασχηματισμένο frame στην συχνότητα και κόβουμε τυχόν επιπλέον τιμές μαζί με τον αποκβαντιστή. Σημειώνεται πως επειδή το 1ο frame είναι κενό από τον κωδικοποιητή Huffman ($MATLAB$: $''$), αρχικοποιούμε sfc και S με μηδενικά (βλ. συνθήκη $if\ k > 1$). Αυτό οφείλεται στο γεγονός ότι προσθέσαμε 2048 μηδενικά στο zero padding. Για το frame στην συχνότητα έχουμε το codebook, ενώ για το sfc κατόπιν trial and error, το codebook είναι το 11.

2. Περνάμε το μετασχηματισμένο frame στην συχνότητα από το $iTNS$ για να πάρουμε το αρχικό frame στην συχνότητα

3. Βρίσκουμε το frame στον χρόνο μέσω $iFilterbank$

Πριν επιστρέψουμε το σήμα, μηδενίζουμε τιμές που είναι NaN ή άπειρο και θέτουμε ως 1 και -1 τις τιμές που είναι μεγαλύτερες του 1 ή μικρότερες του -1 αντίστοιχα.
Η επίδειξη $demoAAC3$ περιλαμβάνει την εφαρμογή $AACoder3$ και $iAACoder3$ διαδοχικά. Το αρχείο $demoAAC3$ εκτελείται σε $\sim$ 80 δευτερόλεπτα (i7-6700HQ). Η έξοδος αποθηκεύεται σε νέο αρχείο ήχου. Αν ακούσουμε το νέο αρχείο ήχου καταλαβαίνουμε μικρή διαφορά. Το plot των κυματομορφών για κάθε κανάλι φαίνεται να μην ταυτίζονται λόγω θορύβου (Figure 5). Το σφάλμα φαίνεται στο Figure 6. Ο σηματοθορυβικός λόγος είναι 4.3742dB για το κανάλι 1 και 4.2029dB για το κανάλι 2. Το bitrate είναι $3.8726 \cdot 10^5$ ενώ η συμπίεση 3.9664 (αφαιρούμε τον χώρο που πιάνουν οι συντελεστές του acoystic threshold T καθώς δεν χρησιμοποιούνται στον $iAACoder3$ αλλα πιάνουν πολύ χώρο).

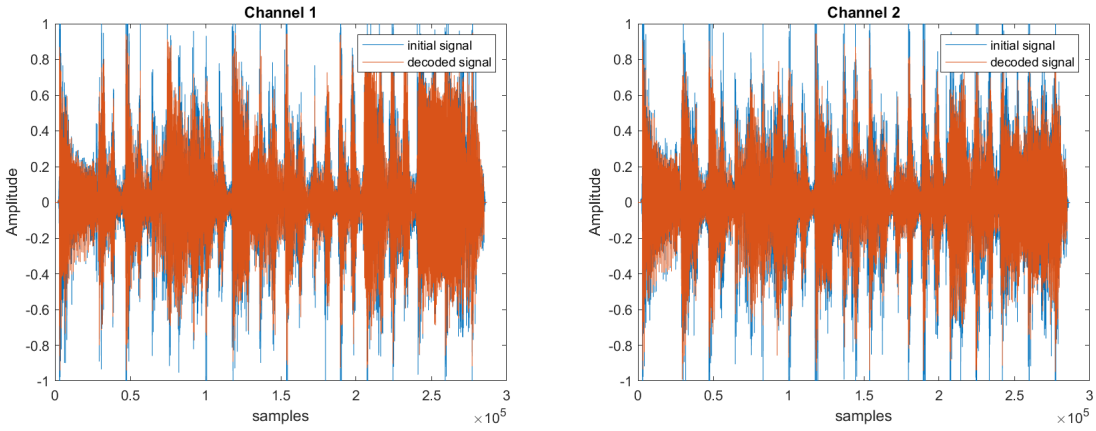

Figure 5: Intitial (red) and decoded signal (blue)

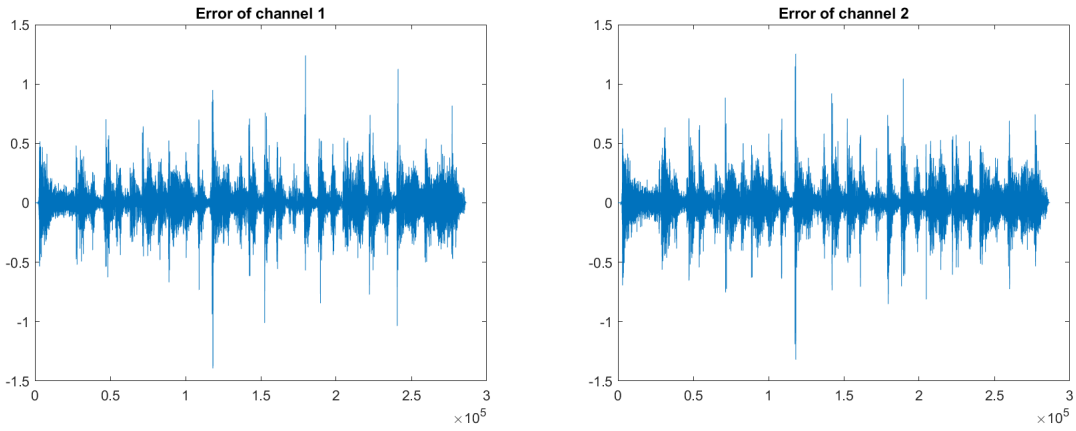

Figure 6: Error of channel 1 and 2

Part IV
Σύνοψη

9 Σύνοψη αποτελεσμάτων

Τα αποτελέσματα από κάθε επίπεδο συνοψίζονται στο Table 2. Τα επίπεδα 1 και 2 εμφανίζουν καλό SNR, ιδανικά θα θέλαμε άπειρο καθώς αυτά το επίπεδα δεν εισάγουν παραμόρφωση ακόμα. Στο επίπεδο 3 υπάρχει έντονος ο θόρυβος. Ωστόσο το άκουσμα του τραγουδιού προς δοκιμή δεν είναι τόσο κακό. Η συμπίεση είναι καλή σε λόγο $\sim 1:5$.

	SNR-ch.1	SNR-ch.2	bitrate	compression
demoAAC1	$\sim$ 285dB	$\sim$ 284dB	-	-
demoAAC2	$\sim$ 285dB	$\sim$ 284dB	-	-
demoAAC3	4.3742dB	4.2029dB	$3.8726 \cdot 10^5$	3.9664

Table 2: Σύνοψη αποτελεσμάτων

10 Σύντομη επισκόπηση συνολικού AAC: *AACoder*3 και *iAACoder*3

Για να γίνει απόλυτα κατανοητή η λειτουργία, παραθέτουμε 2 σχήματα (Figure 7) για τις εισόδους και εξόδους κάθε επιπέδου του συνολικού AACoder και iAACoder.

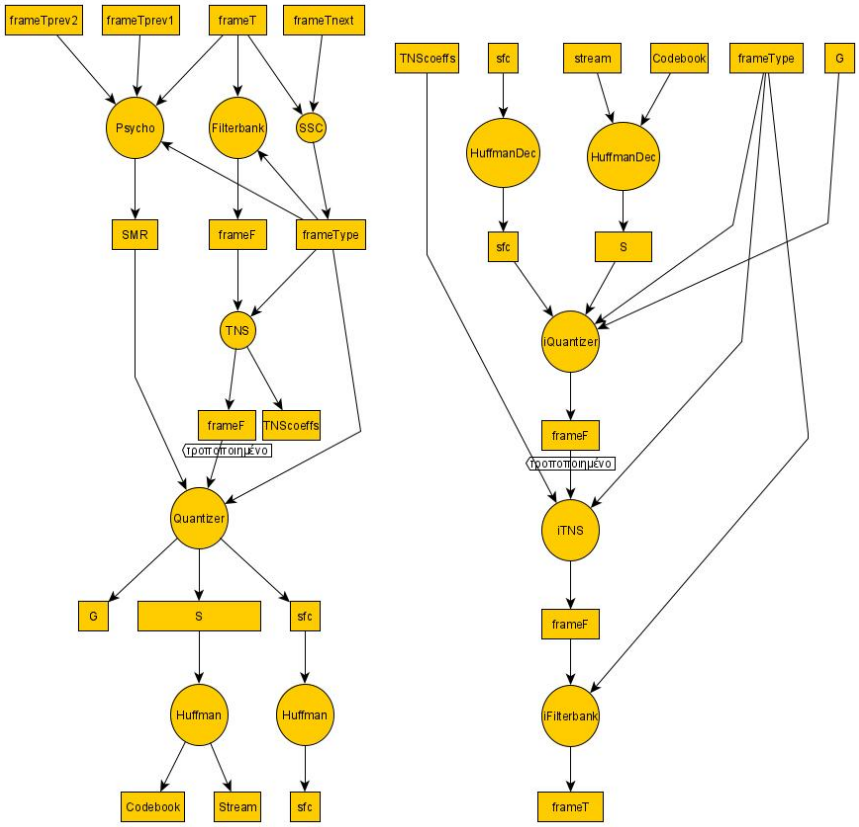

Figure 7: AACoder3-iAACoder