Σήματα-Συστήματα

Μυρτώ Ελευθερία Γκόγκου (ΑΕΜ:3914)

Ιανουάριος 2023

Θέμα 1

1. Εισαγωγή

Σκοπός αυτής της εργασίας είναι η υλοποίηση της μεθόδου επικάλυψης και άθροισης για το φιλτράρισμα καταγραφής μουσικής με ένα χαμηλοπερατό φίλτρο με συχνότητα αποκοπής ίση με 0.15 και μήκους 300 δειγμάτων. Για την σχεδίαση του αλγορίθμου χρησιμοποιήθηκε η γλώσσα προγραμματισμού Matlab.

2. Υλοποίηση αλγορίθμου

```
1    [x,Fs] = audioread('grimelabinc_TurnDown.wav');
2    x = x(:,1);
3    x = x';
```

Αρχικά, φορτώνεται η μουσική (δηλαδή το input signal) μαζί με την συχνότητα της, μέσω της εντολής audioread και αποθηκεύονται στις μεταβλητές x και Fs αντίστοιχα. Επειδή όμως η μουσική αποτελείται από δύο κανάλια, το x θα είναι ένας δισδιάστατος πίνακας. Οπότε πρέπει να επιλεχθεί ένα από τα δύο κανάλια. Στην συγκεκριμένη περίπτωση επιλέγεται το πρώτο κανάλι (εντολή 2). Επίσης, επειδή η audioread επιστρέφει διανύσματα-στήλες χρειάζεται να μετατραπεί σε διάνυσμα-γραμμή ώστε να λειτουργήσει ο αλγόριθμος (εντολή 3).

```
1    h = fir1(299,0.15,'low');
```

Στην συνέχεια χρειάζεται να οριστεί το φίλτρο το οποίο πρέπει να είναι χαμηλοπερατό, πεπερασμένης διάρκειας 300 δειγμάτων και με συχνότητα αποκοπής ίση με 0.15. Για την επίτευξη αυτού, χρησιμοποιείται η εντολή fir1 η οποία δημιουργεί ένα window-based φίλτρο. Οι παράμετροι που χρειάζονται είναι ο αριθμός των δειγμάτων (εδώ έχει χρησιμοποιηθεί το 299, διότι η συνάρτηση αυτή πάντα επιστρέφει n + 1 δείγματα, άρα θα έχουμε 300), η συχνότητα αποκοπής (δηλαδή 0.15) και το είδος του φίλτρου ('low').

```
1    x_length = length(x);
2    h_length = length(h);
3    L = 300000;
4    N = L + h_length - 1;
```

Έπειτα αρχικοποιείται η μεταβλητή L με το μέγεθος που θα χρησιμοποιηθεί για να πραγματοποιηθεί ο τεμαχισμός του x. Το τελικό μήκος του κάθε τμήματος θα είναι ίσο με L + μήκος h +1, το οποίο αποθηκεύεται στην μεταβλητή N.

```
1    extra_zeros = mod(-x_length,L);
2    x = [x, zeros(1,extra_zeros)];
3    x_new_length = length(x);
4    h = [h, zeros(1,L-h_length)];
```

Η συγκεκριμένη εντολή $mod(-x_{length}, L)$ εκτελείται σε περίπτωση που το μέγεθος του σήματος εισόδου δεν διαιρείται ακριβώς με το L . Το αποτέλεσμα που επιστρέφει είναι ο αριθμός των μηδενικών που πρέπει να προστεθούν στο τέλος του σήματος

x έτσι ώστε να διαιρείται ακριβώς με το L, δηλαδή ο αριθμός των τμημάτων που θα σχηματιστούν να είναι ένας ακέραιος αριθμός. Επιπλέον, προστίθενται μηδενικά και στο τέλος του φίλτρου έτσι ώστε το μήκος του να είναι ίσο με N.

```
stages = x_new_length / L;
selection = 1:L;
H = fft(h);
y = [];
```

Η μεταβλητή stages είναι ίση με τον αριθμό των τμημάτων στα οποία θα σπάσει το x , ενώ η selection είναι το εύρος των δειγμάτων που θα επιλέγονται κάθε φορά από το σήμα για τον υπολογισμό των x_i και έχει πάντα μήκος ίσο με L . Επίσης υπολογίζεται ο μετασχηματισμός Fourier του h επειδή χρειάζεται στον υπολογισμό όλων των y_i . Το y που αρχικοποιείται θα είναι ένας δισδιάστατος πίνακας με σκοπό να αποθηκεύσει όλα τα y_i που πρόκειται να υπολογιστούν.

```
for stage=1:stages
    x_i = [x(selection), zeros(1,N- L)];
    X_i = fft(x_i);
    y_i = ifft(X_i.*H);
    y = [y; y_i];
    selection=stage*L+1:(stage+1)*L;
end
```

Στη συνέχεια, εκτελείται ένα for loop όσες φορές όσα είναι και τα τμήματα που θα σχηματιστούν. Για τον υπολογισμό των x_i πρώτα επιλέγονται L στοιχεία από τον πίνακα x και τοποθετούνται επιπλέον μηδενικά για να συμπληρωθούν οι θέσεις έτσι ώστε να έχει μήκος N . Στη συνέχεια υπολογίζεται ο μεταχηματισμός Fourier του x_i και πολλαπλασιάζεται με τον μετασχηματισμό Fourier του h (το οποίο είχε υπολογιστεί νωρίτερα). Στο αποτέλεσμα που προκύπτει υπολογίζεται ο αντίστροφος μετασχηματισμός Fourier οπότε έχει βρεθεί το y_i (το οποίο αποθηκεύεται στον πίνακα y σε μια ξεχωριστή γραμμή). Φυσικά, υπολογίζεται το νέο εύρος τιμών που θα επιλεχθεί στην επόμενη επανάληψη, δηλάδή ανανεώνεται το selection έτσι ώστε να αναφέρεται στα επόμενα L στοιχεία του πίνακα x.

```
y_new = [y(1,:), zeros(1,total_zeros)];
for i=2:stages
    y_temp = [y(i,:), zeros(1,total_zeros)];
    y_temp = circshift(y_temp,shift);
    y_new = [y_new; y_temp];
    shift = shift + L;
end
```

Αφού έχουν προσδιοριστεί όλα τα y_i , πρέπει να υπολογιστούν τα μετατοπισμένα στο χρόνο y_i και να αθροιστούν για να βρεθεί το σήμα εξόδου. Στον πίνακα y_{new} αποθηκεύονται τα y_i αφού όμως γίνει πρώτα μια ολίσθηση των στοιχείων τους. Το y_0 τοποθετείται χωρίς κάποια αλλαγή στην πρώτη γραμμή του πίνακα. Από εκεί και πέρα για κάθε y_i τα στοιχεία του ολισθαίνουν κατα ένα πόσο το οποίο είναι πάντα πολλαπλάσιο του L . Για την κυκλική ολίσθηση χρησιμοποιείται η εντολή circshift, ενώ το ποσό της ολίσθησης είναι αποθηκευμένο στην μεταβλητή shift και σε κάθε επανάληψη η τιμή της αυξάνεται κατά L .

Φυσικά το τελικό αποτέλεσμα δίνεται από την εντολή:

```
y_new = sum(y_new);
```

όπου αθροίζονται οι τιμές των time shifted y_i.

3. Επαλήθευση Αλγορίθμου

Είναι σημαντικό να αποδειχθεί ότι ο αλγόριθμος λειτουργεί ορθά. Οπότε, για οποιοδήποτε σήμα εισόδου και για οποιαδήποτε κρουστική απόκριση, η έξοδος του αλγορίθμου θα πρέπει να είναι ίση με το αποτέλεσμα της γραμμικής συνέλιξης των σημάτων. Έστω λοιπόν ότι το σήμα εισόδου είναι το x[n] = [1 1 1 1 1 1 1 1 1] , ενώ η κρουστική απόκριση είναι η h[n] = [1 1 1]. Για τον υπολογισμό της συνέλιξης προστίθενται οι εντολές:

```
y = conv(x,h);
disp(y);
```

Μετά από την εκτέλεση του αλγορίθμου επικάλυψης-άθροισης (για L = 3) αλλά και της συνέλιξης, προέκυψαν τα δύο παρακάτω αποτελέσματα:

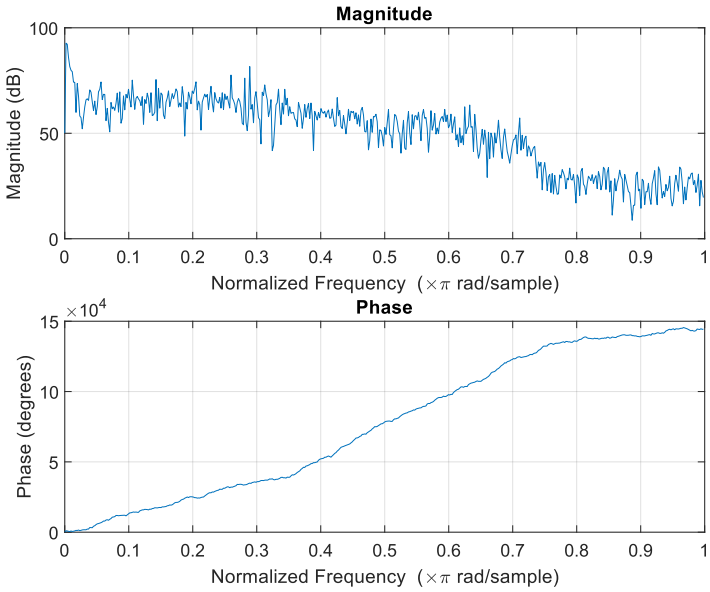
```
>> overlap_add_method
     1     2     3     3     3     3     3     3     3     2     1

  Columns 1 through 10

    1.0000    2.0000    3.0000    3.0000    3.0000    3.0000    3.0000    3.0000    3.0000    2.0000

  Column 11

    1.0000
```

Σχήμα 1: Αποτελέσματα

Στο σχήμα 1, το πρώτο αποτέλεσμα αντιστοιχεί στην συνέλιξη, ενώ το δεύτερο στην μέθοδο επικάλυψης και άθροισης. Τα διανύσματα ταυτίζονται, οπότε ο αλγόριθμος βγάζει τα επιθυμητά αποτελέσματα.

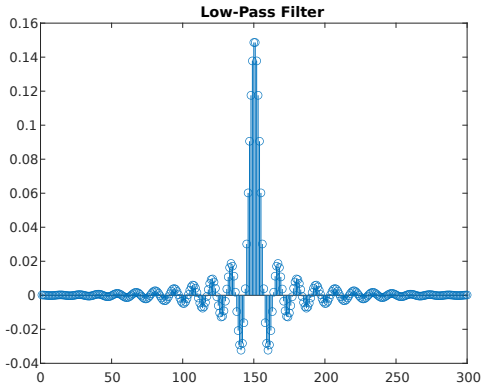
4. Επίδραση αλγορίθμου στη μουσική

Στο σχήμα 2 απεικονίζεται το μέτρο και η φάση της μουσικής που επιλέχθηκε για να φιλτραριστεί. Αποτελείται από ένα μεγάλο εύρος συχνοτήτων, το οποίο περιέχει και συχνότητες που είναι αρκετά μεγαλύτερες από 0.15.

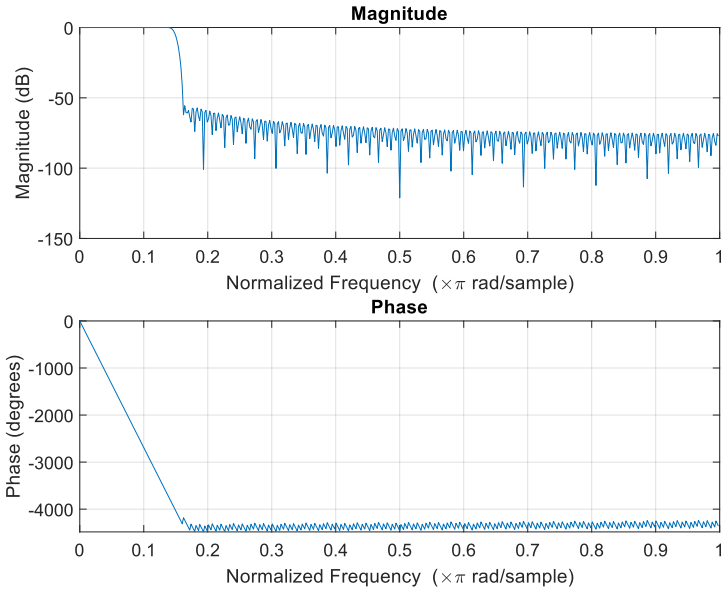


Σχήμα 2: Μέτρο και φάση καταγραφής μουσικής

Στο σχήμα 3 απεικονίζεται το φίλτρο που επέστρεψε η εντολή fir1 (time domain), ενώ στο σχήμα 4 το μέτρο και η φάση του. Παρατηρείται ότι στην μέση του διαστήματος [0.1,0.2] των κανονικοποιημένων συχνότητων (δηλαδή στο 0.15) υπάρχει μια ξαφνική πτώση από τα θετικά στα αρνητικά το οποίο μας εξασφαλίζει ότι θα κοπούν όλες οι συχνότητες που θα είναι μεγαλύτερες του 0.15 και άρα το φίλτρο έχει σχεδιαστεί σωστά.

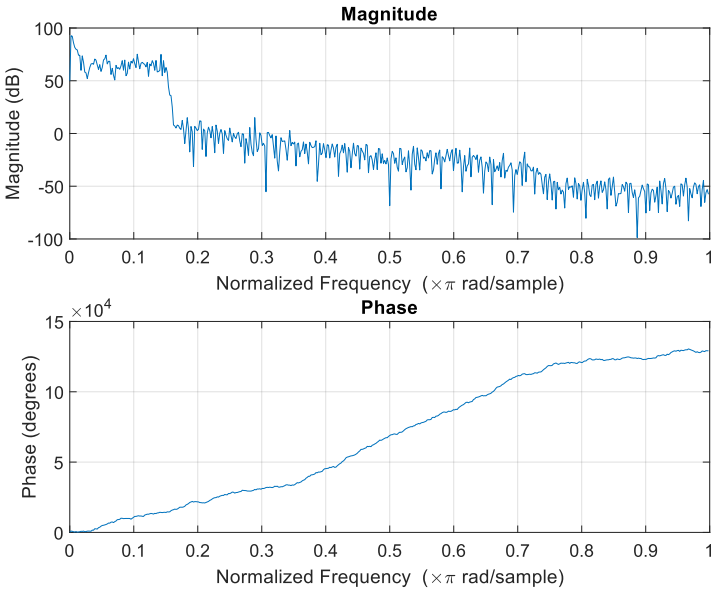


Σχήμα 3: Χαμηλοπερατό φίλτρο



Σχήμα 4: Μέτρο και φάση φίλτρου

Στο σχήμα 5 απεικονίζεται το μέτρο και η φάση του σήματος μετά από το φιλτράρισμα μέσω του αλγορίθμου επικάλυψης και άθροισης. Πολύ εύκολα παρατηρείται η επίδραση που είχε το χαμηλοπερατό φίλτρο στον ήχο, καθώς συγκρίνοντας με το πρώτο διάγραμμα, οι μόνες συχνότητες που πλέον έχουν θετικές τιμές είναι μεταξύ του διαστήματος [0,0.15]. Για συχνότητες μεγαλύτερες του 0.15 υπάρχει μια μεγάλη πτώση από τα θετικά στα αρνητικά. Οπότε σωστά δεν περιέχονται συχνότητες μεγαλύτερες του 0.15. Η μουσική που προέκυψε μετά από το φιλτράρισμα είναι αποθηκευμένη στο αρχείο με όνομα new_audio.wav



Σχήμα 5: Μέτρο και φάση καταγραφής μουσικής μετά από φιλτράρισμα