Αποθορυβοποίηση με φίλτρα Wiener

Μπούρχα Ιωάννα 58019

30 ΙΟΥΝΙΟΥ 2023

ΨΗΦΙΑΚΗ ΕΠΕΞΕΡΓΑΣΙΑ ΣΗΜΑΤΟΣ

Ερώτημα Α

Αρχικά, διαβάζω το αρχείο 'guit1.wav' που δόθηκε και αποθηκεύω τα δεδομένα και τον ρυθμό δειγματοληψίας αντίστοιχα στις μεταβλητές data και sample_rate. Σύμφωνα με την εκφώνηση, δημιουργώ Γκαουσιανό θόρυβο απόκλισης 0,01 και τον προσθέτω στα δεδομένα μου. Το αποτέλεσμα αυτό το αποθηκεύω στην μεταβλητή data_noise και δημιουργώ ένα αρχείο ήχου.

```
%% ---- QUESTION A ----
[data, sample_rate] = audioread('guit1.wav');


% Greating white gaussian noise with 0.01 deviation
deviation = 0.01;
noise = deviation * randn(size(data));
n = length(noise) ;

% Adding noise to my data
data_noise = data + noise;
audiowrite('guit1_noise.wav', data_noise, sample_rate);
```

Για την αποθορυβοποίηση του σήματός μου χρησιμοποιώ ένα FIR Wiener φίλτρο με τάξη 10, 20 και 30. Θεωρώντας ότι είναι γνωστή η αρχική μου κυματομορφή, εκείνη χωρίς θόρυβο, υλοποιώ το φίλτρο όπως ακριβώς και οι διαφάνειες 44 - 45 του αντίστοιχου μαθήματος. Εφόσον η μεθοδολογία αυτή θα εφαρμοστεί και σε επόμενα ερωτήματα καλώ μία συνάρτηση για την δημιουργία του φίλτρου. Το αποτέλεσμα της αποθορυβοποίησης σώζεται σε αρχείο ήχου. Ακόμη, προκειμένου να συγκρίνω ποιοτικά το φίλτρο μου, εκτυπώνω τις κυματομορφές μου σε ένα κοινό παράθυρο. (Εικόνα 1).

```
function z = wiener_filter(rank, data, data_noise, n)
    telos = n+rank-1;
    rxx = xcorr(data_noise);       % autocorrelation of the noised waveform
    rxd = xcorr(data_noise,data); % autocorrelation of the two waveforms
    rxx = rxx(n:telos);
    rxd = rxd(n:telos);
    Rxx = toeplitz(rxx);
    w = pinv(Rxx,0.0001)*rxd;
    z = filter(w,1,data_noise);
end
```

Το φίλτρο αυτό καλείται μία φορά για κάθε τάξη του φίλτρου ξεχωριστά.

```
p = [10, 20, 30];
for i = 1:length(p)
    z = wiener_filter(p(i), data, data_noise, n, sample_rate);
    fileName  = strcat('guit1_filtered_'  , num2str(p(i)), '.wav');
    fileName1 = strcat('guit1_filtered_A_', num2str(p(i)), '.wav');
    audiowrite(fileName , z, sample_rate);
    audiowrite(fileName1, z, sample_rate);
end
```

Ένας δείκτης της αποθορυβοποίησης είναι ο SNR (Signal to Noise Ration). Για τον υπολογισμό του δημιουργείται συνάρτηση η οποία καλείται κάθε φορά που ζητείται ο υπολογισμός του ενθόρυβου και του αποθορυβοποιημένου σήματος.

```
function snr = SNR_calculator(data, data_noise, noise, p)
    snr= zeros(1, length(p)+1);
    snr(1) = 10 * log10(sum(data_noise.^2)/ sum(noise.^2));
    disp(['SNR (noised signal): ' num2str(snr(1)) ' dB']);

    for i = 1:length(p)
        fileName = strcat('guit1_filtered_', num2str(p(i)), '.wav');
```

```
        [z, ~] = audioread(fileName);

        noise_temp = z - data;
        snr(1, i+1) = 10 * log10(sum(z.^2)/ sum(noise_temp.^2));
        disp(['SNR (' fileName '): ' num2str(snr(1, i+1)) ' dB']);
    end
end

% Calculating SNR of noised and denoised signal
snr_A = SNR_calculator(data, data_noise, noise, p);
```

Τα αποτελέσματα που εμφανίζονται στην κονσόλα είναι:

---- QUESTION A ----
SNR (noised signal): 19.4317 dB
SNR (guit1_filtered_10.wav): 19.7068 dB
SNR (guit1_filtered_20.wav): 19.7143 dB
SNR (guit1_filtered_30.wav): 19.7194 dB

Ήδη από τις τιμές του SNR γίνεται αντιληπτό ότι τα τρία αρχεία δεν έχουν κάποια αξιοσημείωτη διαφορά μεταξύ τους. Η παρατήρηση αυτή επιβεβαιώνεται τόσο με την ακρόαση των αρχείων όσο και από την δεύτερη στήλη της εικόνας 1.

Ερώτημα Β

Ακολουθώ την ίδια μεθοδολογία με το προηγούμενο ερώτημα με την διαφορά ότι τώρα την εφαρμόζω ξεχωριστά σε κάθε τμήμα (παράθυρο) του σήματός μου. Προκειμένου να δημιουργήσω τα παράθυρα καλώ τις συναρτήσεις frame_wind και να επανασυνθέσω το σήμα μου την frame_recon. Στην συνέχεια, καλώ την συνάρτηση για την εισαγωγή του φίλτρου Wiener. Προκειμένου να συγκρίνω ποιοτικά το φίλτρο μου, εκτυπώνω τις κυματομορφές μου σε ένα κοινό παράθυρο. (Εικόνα 1).

```
%% ---- QUESTION B ----
frameSize = 256;
overlap = 0.5;
windows        = frame_wind(data, frameSize, overlap);
windows_noised = frame_wind(data_noise, frameSize, overlap);

% Appling Wiener filter to each window
filteredWindows = zeros(size(windows_noised));
for i = 1:length(p)
    fileName  = strcat('guit1_filtered_', num2str(p(i)), '.wav');
    fileName1 = strcat('guit1_filtered_B_', num2str(p(i)), '.wav');

    for j = 1:size(windows_noised, 2)
        window       = windows(:, j);
        window_noise = windows_noised(:, j);

        filteredWindow = wiener_filter(p(i), window, window_noise, frameSize);
        filteredWindows(:, j) = filteredWindow;
    end

    % Convert the filtered windows back to a single denoised signal
    z = frame_recon(filteredWindows, overlap);

    audiowrite(fileName , z, sample_rate);
    audiowrite(fileName1, z, sample_rate);
end
```

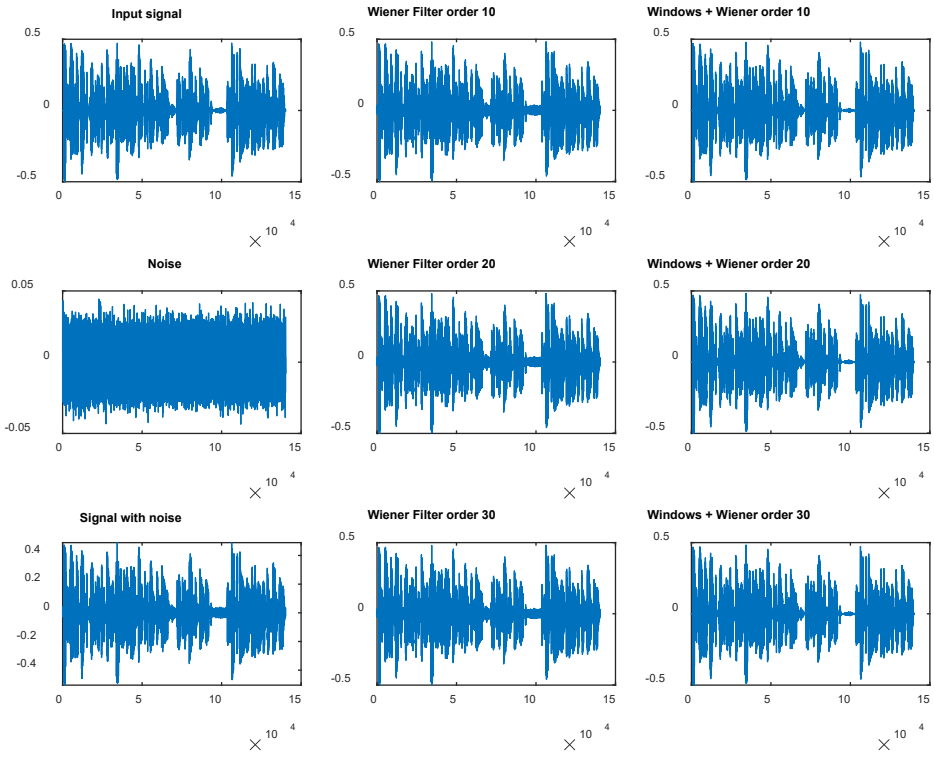
Για τον υπολογισμό του SNR απαιτείται να σπάσω σε παράθυρα και να επανασυνθέσω και τα αρχικά μου δεδομένα.

```
% Calculating SNR of noised and denoised signal
temp = frame_wind(data, frameSize, overlap);
temp = frame_recon(temp, overlap);
snr_B = SNR_calculator(temp', data_noise, noise, p);
```

Τα αποτελέσματα που εμφανίζονται στην κονσόλα είναι:

---- QUESTION B ----
SNR (noised signal): 19.4317 dB
SNR (guit1_filtered_10.wav): 21.3979 dB
SNR (guit1_filtered_20.wav): 21.44 dB
SNR (guit1_filtered_30.wav): 21.4546 dB

Όπως και προηγουμένως, ήδη από τις τιμές του SNR γίνεται αντιληπτό ότι τα τρία αρχεία δεν έχουν κάποια αξιοσημείωτη διαφορά μεταξύ τους. Η παρατήρηση αυτή επιβεβαιώνεται τόσο με την ακρόαση των αρχείων όσο και από την τρίτη στήλη της εικόνας 1.



Εικόνα 1: Αριστερά: Αρχικά Σήματα Κέντρο: Αποθορυβοποίηση με εφαρμογή φίλτρου, Δεξιά: Αποτελέσματα με μέθοδο παραθύρου και φίλτρο

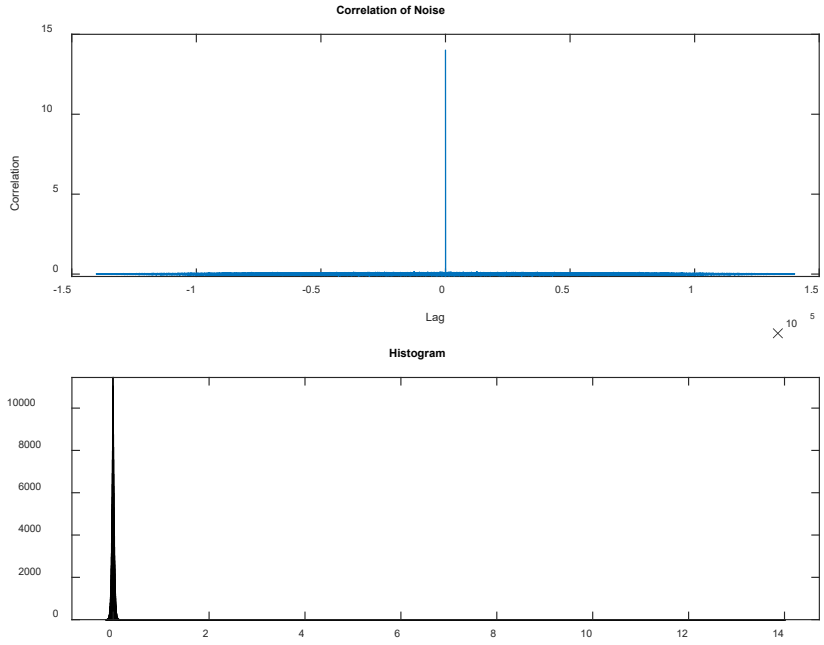
Συγκρίνοντας τα αποτελέσματα του δείκτη SNR των δύο ερωτημάτων όσο και τις αντίστοιχες κυματομορφές τους, συμπεραίνω ότι η μεθοδολογία των παραθύρων δίνει καλύτερα αποτελέσματα αποθορυβοποίησης.

Ερώτημα C:

Η αυτοσυσχέτιση του θορύβου είναι:

```
correlation = xcorr(noise);
```

Παρατηρώ ότι οι τιμές του πίνακα είναι πολύ κοντά στο μηδέν εκτός από μία συγκεκριμένη τιμή στην οποία εμφανίζει μέγιστο. Επίσης, στο ιστόγραμμα η κατανομή των δειγμάτων σχηματίζει μία καμπάνα με μικρή τυπική απόκλιση, η οποία μπορεί να προσεγγιστεί με γκαουσιανή κατανομή. Επομένως, ο θόρυβος είναι Γκαουσιανός, ή αλλιώς λευκός.



Εικόνα 2:
Πάνω: Διάγραμμα αυτοσυσχέτισης θορύβου. Εφόσον οι τιμές είναι κοντά στο μηδέν εκτός από μία τιμή στην οποία εμφανίζει μέγιστο, επιβεβαιώνω ότι ο θόρυβός μου είναι λευκός.
Κάτω: Ιστόγραμμα. Σχηματίζεται καμπάνα μικρής τυπικής απόκλισης η οποία μπορεί να προσεγγιστεί με γκαουσιανή κατανομή.

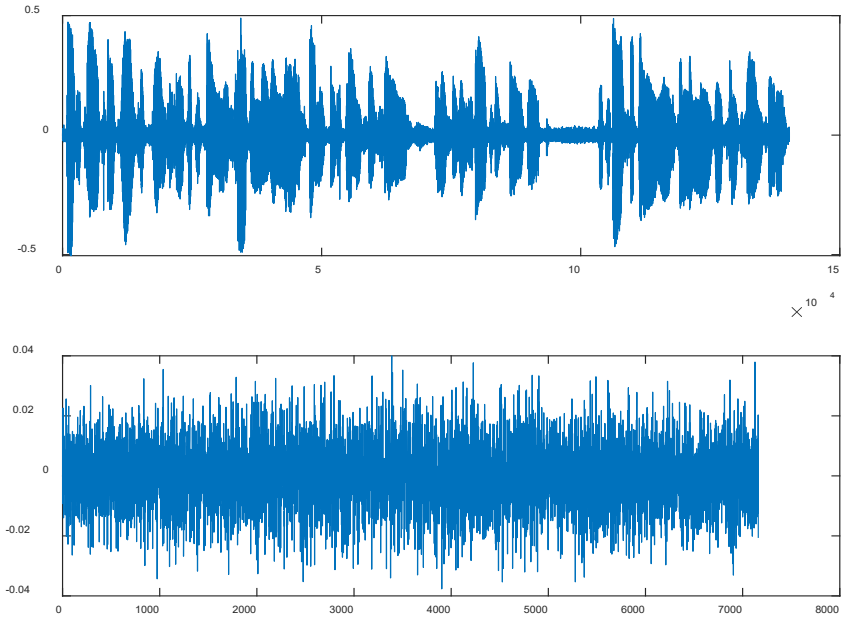
Ερώτημα D

Πλέον θεωρώ ότι δεν γνωρίζω καθόλου τα αρχικά μου δεδομένα. Με άλλα λόγια, έχω μονάχα το ενθόρυβο σήμα. Προκειμένου να προσδιορίσω τον θόρυβο που θα πρέπει να αφαιρέσω, εκτυπώνω την κυματομορφή του και προσπαθώ να βρω ένα κομμάτι που έχει μόνο θόρυβο.

```
figure('Name', 'Data with noise --> Finding an array with jusy noise');
subplot(2,1,1);     plot(data_noise);    % noise in [95440, 102600]
```

Από το επάνω διάγραμμα της εικόνας 4 συμπεραίνω ότι ο θόρυβος εντοπίζεται καθαρά στο διάστημα [95440, 102600], το οποίο απομονώνω και θεωρώ πλέον ως θόρυβο

```
disp('Noised is located in [95440 102600]');
noise_d = data_noise(95440 : 102600 , :);
subplot(2,1,2); plot(noise_d);
```



Εικόνα 3: Πάνω: Κυμματομορφή ενθόρυβου σήματος Κάτω: Απομόνωση τμήματος που θεωρώ ως θόρυβο

Προκειμένου να προσδιορίσω την αυτοσυσχέτιση του αρχικού πλήρως καθαρού σήματος:

1. Υπολογίζω την αυτοσυσχέτιση του ενθόρυβου σήματος και του θορύβου

```
corr_noised = xcorr(data_noise);
corr_noise  = xcorr(noise_d);
```

2. Προσδιορίζω τους συντελεστές ανάλογα με την τάξη του φίλτρου

```
mean_noised = round(length(corr_noised)/2, 0);
mean_noise  = round(length(corr_noise )/2,

indexes_noised = corr_noised(mean_noised : mean_noised+p(i)-1 );     % rxx
indexes_noise  = corr_noise (mean_noise  : mean_noise +p(i)-1 );     % rnn
```

3. Αφαιρώ τους συντελεστές μεταξύ τους

```
temp = (indexes_noised - indexes_noise)';                            % rxd
```

4. Χρησιμοποιώ αυτές τις τιμές για το φίλτρο Wiener

```
% Using the above colloration in Wiener Filter
Rxx = toeplitz(indexes_noised);
w = pinv(Rxx, 0.0001)*temp(1:p(i))';
z = filter(w,1,data_noise);
```

Η διαδικασία αυτή επαναλαμβάνεται ξεχωριστά για κάθε τάξη φίλτρου. Τα αποτελέσματα της αποθορυβοποίησης αποθηκεύονται ξεχωριστά σε αρχεία ήχου.

Για τον υπολογισμό του SNR καλώ την συνάρτηση που δημιούργησα στο πρώτο ερώτημα.

```
% Calculating SNR
snr_D = SNR_calculator(data, data_noise, noise, p);
```

Σημειώνω ότι σαφώς τα αποτελέσματα είναι χειρότερα από εκείνα του ερωτήματος α, καθώς δεν διαθέτω την αυτοσυσχέτιση του καθαρού σήματος, αλλά μία προσέγγισης αυτής.

Συγκεκριμένα, προκειμένου να προσδιορίσω το σφάλμα της αυτοσυσχέτισης που βρήκα και της πραγματικής αυτοσυσχέτισης του θορύβου, υπολόγισα το τετραγωνικό σφάλμα.

```
% Repeat for raw input in order to evaluate my results
corr_signal = xcorr(data);
mean_signal = round(length(corr_signal)/2, 0);


% zero padding
if length(temp) < max(p)
    temp = [temp, zeros(1,max(p)-p(i))];
end

indexes(i,:) = temp;

% Evaluating my results with Mean Square Error
indexes_signal = (corr_signal(mean_signal : mean_signal+p(i)-1 ))';
mse_d(i) = immse( indexes(i,1:p(i)), indexes_signal  );
```

Η τεχνική του zero padding χρησιμοποιήθηκε προκειμένου να αποθηκεύω τους συντελεστές που βρήκα για κάθε τάξης φίλτρου σε έναν πίνακα.

Τα αποτελέσματα που εκτυπώνονται στην κονσόλα είναι:

---- QUESTION D ----

Filter's order	mse	max_value	min_value
10	18.0357	1215.4664	181.2838
20	9.034	1215.4664	-437.2047
30	6.0523	1215.4664	-489.728

SNR (noised signal): 19.4329 dB
SNR (guit1_filtered_10.wav): 19.4681 dB
SNR (guit1_filtered_20.wav): 19.469 dB
SNR (guit1_filtered_30.wav): 19.4695 dB

Ερώτημα Ε

Συνδυάζοντας τις μεθοδολογίες του ερωτήματος Β και D προκύπτει ο κώδικας του τρέχοντος. Σημειώνεται ότι σε κάθε παράθυρο θεωρώ ως θόρυβο το τμήμα του οποίου η μέση τιμή των απολύτων τιμών του είναι μικρότερη από το threshold = 0.0001. Η τιμή αυτή προκύπτει τυχαία. Προσωπικά σκέφτηκα να βάλω το τετράγωνο της απόκλισης του θορύβου.

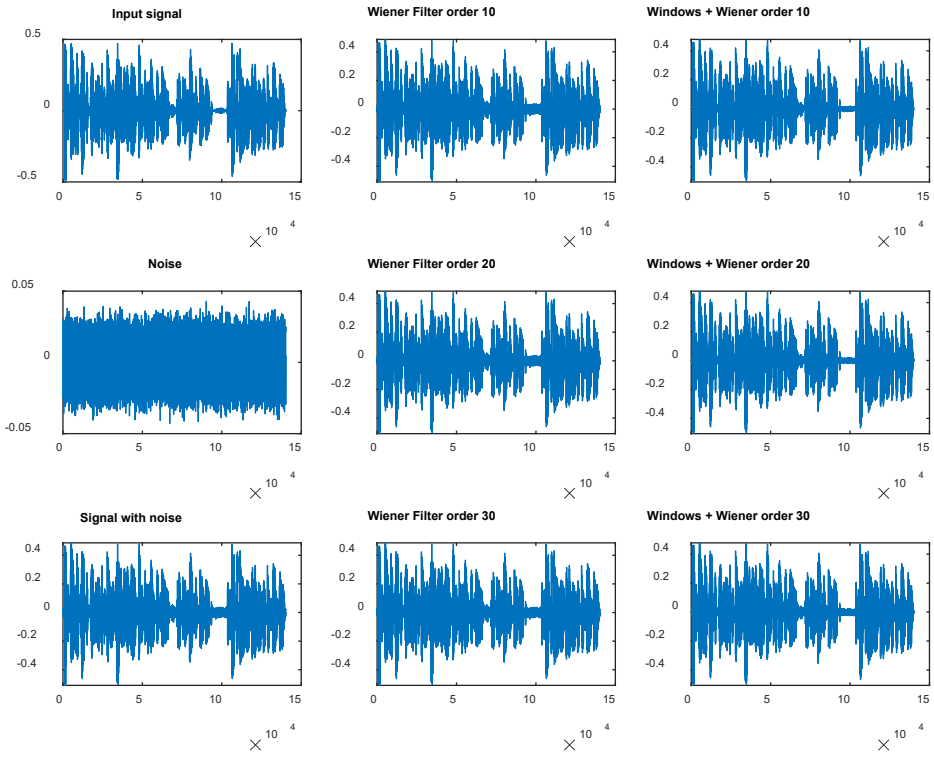
```
%% ---- QUESTION E ----
threshold = 0.0001;

windows        = frame_wind(data, frameSize, overlap);
windows_noised = frame_wind(data_noise, frameSize, overlap);

% Appling Wiener filter to each window
filteredWindows = zeros(size(windows_noised));
for i = 1:length(p)
    fileName = strcat('guit1_filtered_', num2str(p(i)), '.wav');
    fileName1 = strcat('guit1_filtered_E_', num2str(p(i)), '.wav');

    window_noise  = windows_noised(:, 1);

    for j = 1:size(windows_noised, 2)

        if mean(abs(windows_noised(:,j))) < threshold
            window_noise = windows_noised(:,j);
        end

        window_noised = windows_noised(:,j);

        corralation_noised = xcorr(window_noised);      % rxx
        corralation_noise  = xcorr(window_noise );      % rnn

        temp = corralation_noised - corralation_noise;  % rxd

        % Appling Winer Filter
        telos = frameSize + p(i) -1 ;
        rxd = temp(frameSize : telos);
        rxx = corralation_noised(frameSize : telos);
        Rxx = toeplitz(rxx);
        w = pinv(Rxx, 0.0001)*rxd;
        filteredWindow = filter(w,1,window_noised);

        filteredWindows(:, j) = filteredWindow;
    end
```

```
    % Convert the filtered windows back to a single denoised signal
    z = frame_recon(filteredWindows, overlap);

    audiowrite(fileName , z, sample_rate);
    audiowrite(fileName1, z, sample_rate);
end

% Calculating SNR of noised and denoised signal
temp = frame_wind(data, frameSize, overlap);
temp = frame_recon(temp, overlap);
snr_E = SNR_calculator(temp', data_noise, noise, p);
```

Τα αποτελέσματα που εκτυπώνονται στην κονσόλα είναι:

---- QUESTION E ----
SNR (noised signal): 19.4329 dB
SNR (guit1_filtered_10.wav): 21.0242 dB
SNR (guit1_filtered_20.wav): 20.8503 dB
SNR (guit1_filtered_30.wav): 20.3167 dB

Προκειμένου να συγκρίνω ποιοτικά τα αποτελέσματα των φίλτρων στην απλή εφαρμογή του φίλτρου και στην μέθοδο των παραθύρων, τα εκτυπώνω μαζί με τα αρχικά σε ένα παράθυρο (Εικόνα 4).



Εικόνα 4: Αριστερά: Αρχικά Σήματα Κέντρο: Αποθορυβοποίηση με εφαρμογή φίλτρου, Δεξιά: Αποτελέσματα με μέθοδο παραθύρου και φίλτρο

Ερώτημα F

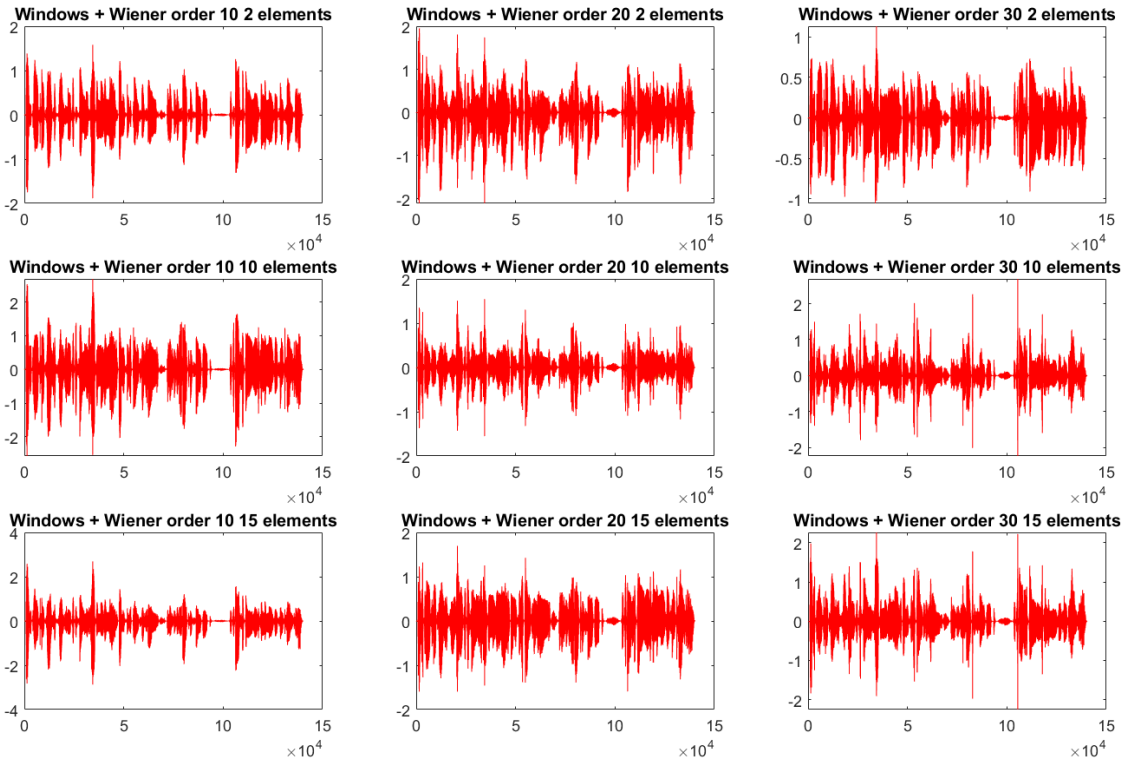
Στο συγκεκριμένο ερώτημα ζητείται η πρόβλεψη ορισμένων τιμών του σήματος. Η μεθοδολογία που υιοθετήθηκε είναι παρόμοια με εκείνη των αντίστοιχων διαφανειών του μαθήματος. Συγκεκριμένα, το μόνο που αλλάζει είναι το κομμάτι που κώδικα που αναφέρεται στο φίλτρο καθώς και η επανάληψή του για τις τρεις τιμές που ζητούνται κάθε φορά (2 10 15).

```
%% ---- QUESTION F ----
values = [2 10 15];
snr_F = zeros(1, length(p));

frameSize = 256;
overlap = 0.5;
windows = frame_wind(data, frameSize, overlap);
data2   = frame_recon(windows, overlap);

% Appling Wiener filter to each window
filteredWindows = zeros(size(windows));
for i = 1:length(p)
    for v = 1:length(values)
    % Wiener Filter
    for j = 1:size(windows, 2)
        window = windows(:, j);

        telos = frameSize+p(i)-1;
        rss = xcorr(data_noise);
        rxx = rss(frameSize   : telos  );
        rss = rss(frameSize+values(v) : telos+values(v));
        Rxx = toeplitz(rxx);
        w = pinv(Rxx,0.0001)*rss;
        filteredWindow = filter(w,1,window);

        filteredWindows(:, j) = filteredWindow;
    end

    % Convert the filtered windows back to a single denoised signal
    z = frame_recon(filteredWindows, overlap);

    % Calculating SNR of noised and denoised signal
    noise_temp = data2 - z;
    snr_F(1, i) = 10 * log10(sum(z.^2)/ sum(noise_temp.^2));
    disp(['SNR (Wiener ' num2str(p(i)) ' rank) with ' num2str(values(v)) '
elements: ' num2str(snr_F(1, i)) ' dB']);

    end
end
```

Οι τιμές των αποτελεσμάτων που εμφανίζονται στην κονσόλα είναι:
---- QUESTION F ----
SNR (Wiener 10 order) with 2 elements: 2.5493 dB
SNR (Wiener 10 order) with 10 elements: -0.51461 dB
SNR (Wiener 10 order) with 15 elements: 2.1365 dB
SNR (Wiener 20 order) with 2 elements: 1.6886 dB
SNR (Wiener 20 order) with 10 elements: -1.44 dB
SNR (Wiener 20 order) with 15 elements: -0.32772 dB
SNR (Wiener 30 order) with 2 elements: 0.85115 dB
SNR (Wiener 30 order) with 10 elements: -0.92594 dB
SNR (Wiener 30 order) with 15 elements: -1.5878 dB

Οι πολύ μικρές τιμές του SNR είναι αναμενόμενες, καθώς κόβουμε το αρχικό μας σήμα. Όσο πιο λίγα στοιχεία επιδιώκουμε να προβλέψουμε τόσο καλύτερα αποτελέσματα έχουμε. Όσο μεγαλώνει η τάξη του φίλτρου τόσο πιο πολύ φιλτράρεται το σήμα άρα τόσο περισσότερη πληροφορία χάνεται.

Προκειμένου να συγκρίνω ποιοτικά τα αποτελέσματα του κώδικά μου εκτυπώνω σε κοινό παράθυρο την έξοδο του φίλτρου (κόκκινο) και το αρχικό πλήρως καθαρό σήμα του οποίου προβλέπεται η τιμή (μπλε).



Εικόνα 5: Πρόβλεψη τιμών