WIENER FILTER FOR DENOISING

Bourcha Ioanna 58019

30 JUNE 2023

DSP

Question_A

First of all, I read the file 'guit1.wav' provided and store the data and the sample rate in the variables 'data' and 'sample_rate', respectively. According to the prompt, I generate Gaussian noise with a standard deviation of 0.01 and add it to my data. I save this result in the variable 'data_noise' and create an audio file.

```
%% ---- 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);
```

For the denoising of my signal, I am using an FIR Wiener filter of order 10, 20, and 30. Assuming that my original waveform, without noise, is known, I implement the filter exactly as described in slides 44-45 of the corresponding lecture. Since this methodology will be applied in subsequent questions as well, I call a function to create the filter. The result of the denoising is saved in an audio file. Additionally, in order to qualitatively compare my filter, I plot my waveforms in a common window (Figure 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
```

This filter is called once for each filter class separately.

```
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
```

An indicator of denoising is SNR (Signal to Noise Ration). To calculate it, a function is created which is called every time the calculation of the noise and denoised signal is requested.

```
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);
```

The results displayed in the console are:

---- 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

Already from the SNR values it can be seen that the three files do not have any noticeable difference between them. This observation is confirmed both by listening to the files and by the second column of Figure 1.

Question B

I follow the same method as in the previous question except that now I apply it separately to each part (window) of my signal. In order to create the windows I call the given function frame_wind and to reconstruct my signal frame_recon. Next, I call the function to insert the Wiener filter. In order to qualitatively compare my filter, I print my waveforms in a common window. (Picture 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
```

Calculating SNR requires me to split and re-synthesize my original data as well.

```
% 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);
```

The results displayed in the console are:

---- QUESTION B ----
SNR (noise 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

As before, already from the SNR values it can be seen that the three files do not have any noticeable difference between them. This observation is confirmed both by listening to the records and by the third column of Figure 1.

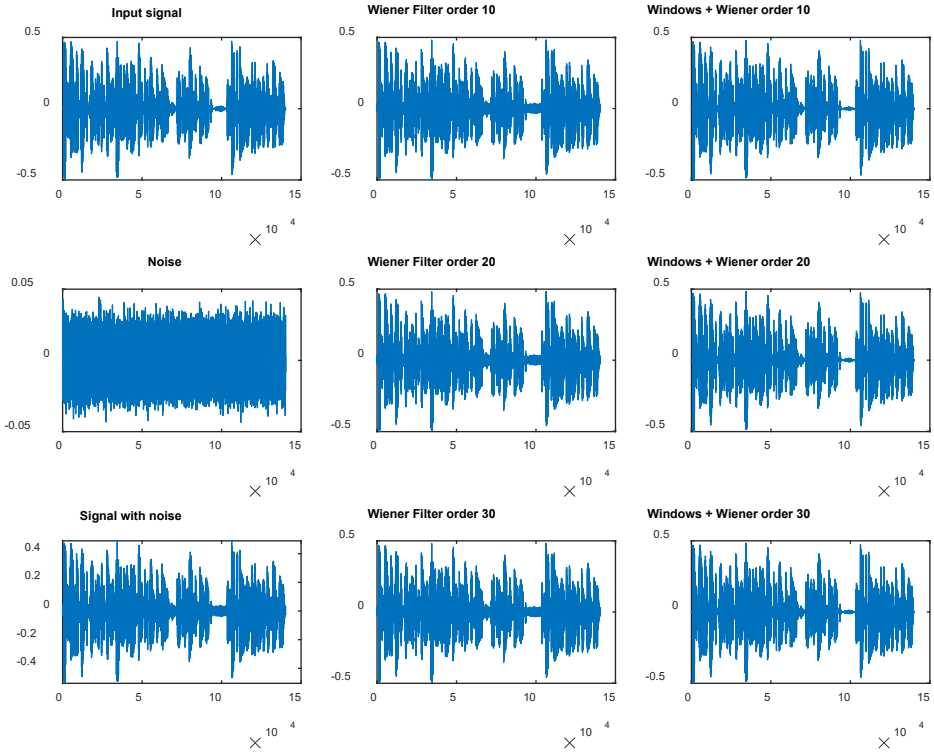

Figure 1: Left: Original Signals Center: Denoising with Filter Apply, Right: Results with Window Method and Filter

Comparing the SNR results of the two questions as well as their respective waveforms, I conclude that the windowing method gives better denoising results.

Question C:

The autocorrelation of the noise is:

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

I notice that the table values are very close to zero except for one particular value where it shows a maximum. Also, in the histogram the distribution of the samples forms a bell with a small standard deviation, which can be approximated by a Gaussian distribution. Therefore, the noise is Gaussian, or otherwise white.

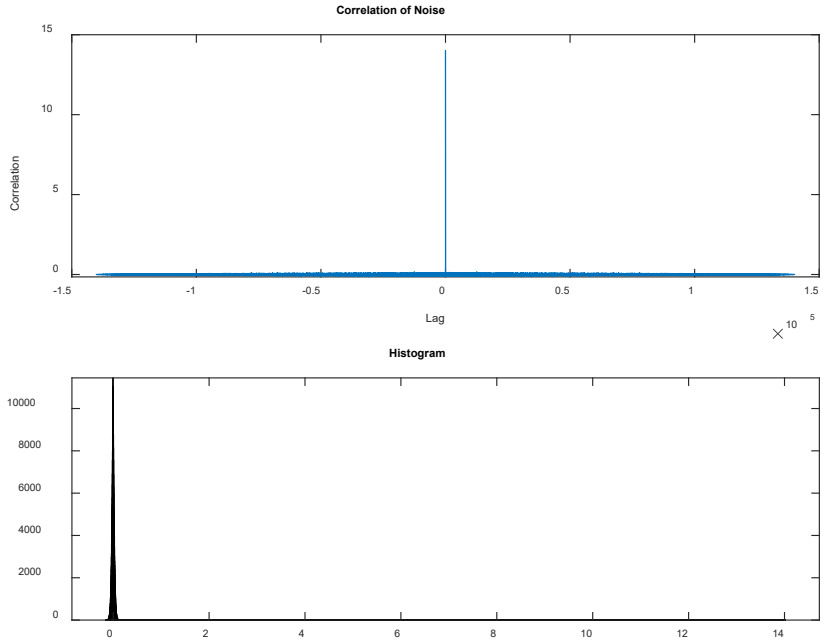

Figure 1:
Top: Noise autocorrelation plot. Since the values are close to zero except for one value at which it shows a maximum, I confirm my noise is white. Bottom: Histogram. A small standard deviation bell is formed which can be approximated by a Gaussian distribution.

Question D

Now I feel that I do not know my original data at all. In other words, I only have the noisy signal. In order to determine the noise I should remove, I print out its waveform and try to find a piece that has only noise.

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

From the top plot of image 4 I conclude that the noise is clearly located in the interval [95440, 102600], which I isolate and now consider as noise.

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

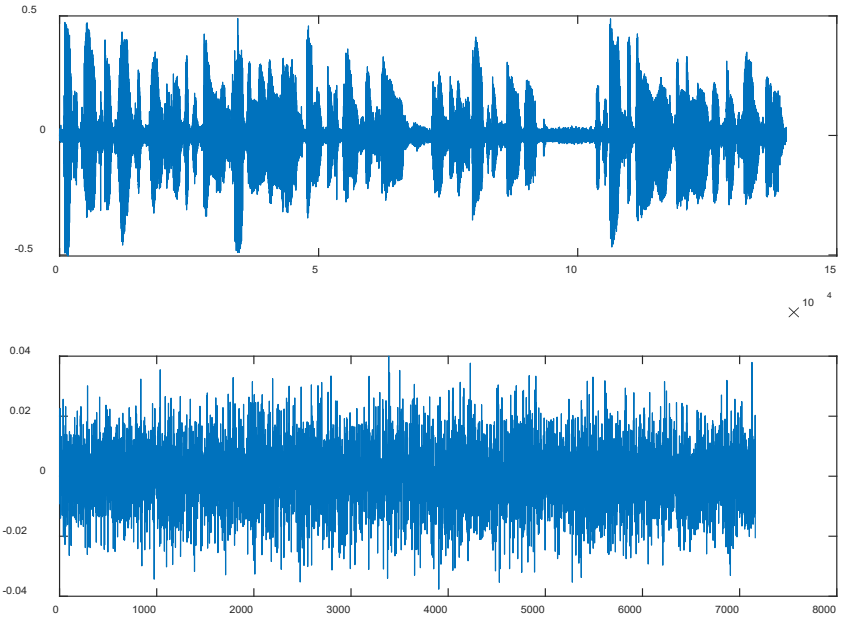

Figure 3: Top: Noisy signal waveform Bottom: Isolation of part I consider as noise

In order to determine the autocorrelation of the original fully clean signal:

1. I calculate the autocorrelation of the noisy signal and the noise

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

2. I determine the coefficients depending on the order of the filterφίλτρου

```
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. Subtract the coefficients from each other

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

4. I use these values for the Wiener filter

```
% 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);
```

This process is repeated separately for each filter class. Denoising results are saved separately in audio files.

To calculate the SNR I call the function I created in the first question.

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

Clearly the results are worse than those of question A, since I do not have the autocorrelation of the pure signal, but an approximation of it.

Specifically, in order to determine the error of the autocorrelation I found and the true autocorrelation of the noise, I calculated the squared error.

```
% 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  );
```

The zero padding technique was used to store the coefficients I found for each filter class in a table.

The results printed to the console are:

---- 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

Question E

Combining the methods of questions B and D yields the current code. It is noted that in each window I consider as noise the part whose average value of its absolute values is less than the threshold = 0.0001. This value is generated randomly. I personally thought of putting the square of the noise deviation.

```
%% ---- 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);
```

The results printed to the console are:

---- QUESTION E ----
SNR (noise 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

In order to qualitatively compare the results of the filters in the simple filter application and the windows method, I print them together with the originals in a window (Figure 4).

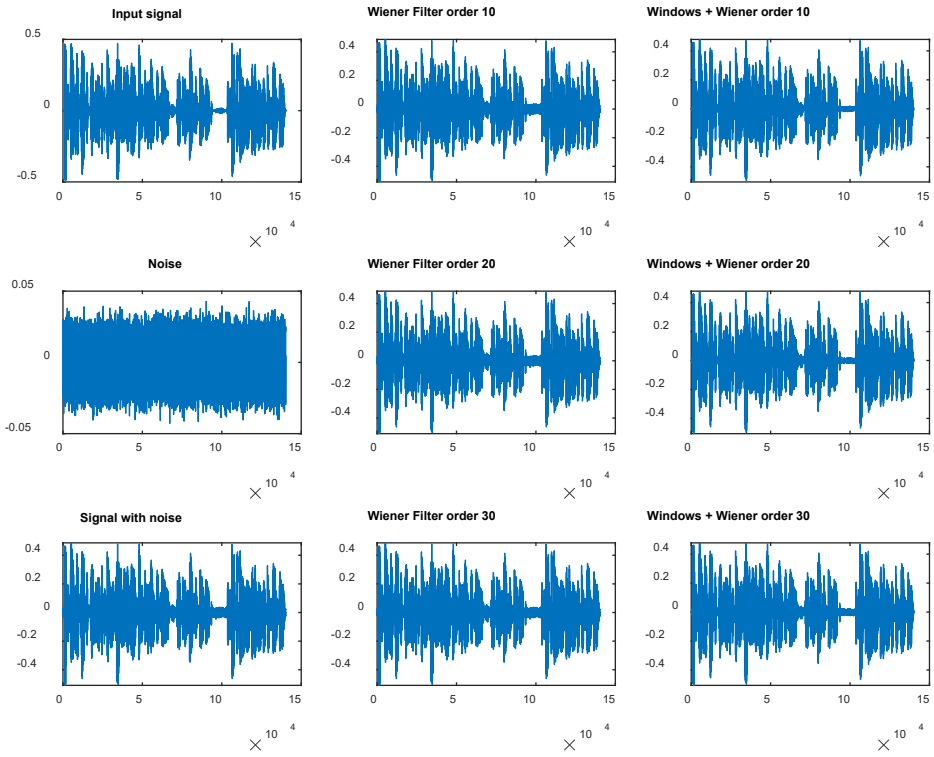

Figure 4: Left: Original Signals Center: Denoising with filter applied, Right: Results with window method and filter

QUESTION F

In this particular question, the prediction of certain signal values is requested. The method adopted is similar to that of the corresponding slides of the course. In particular, the only thing that changes is the piece of code that refers to the filter as well as its repetition for the three values requested each time (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
```

The result values displayed in the console are:

---- 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

The very small values of SNR are expected, as we are clipping our original signal. The fewer elements we try to predict, the better results we get. The higher the order of the filter, the more the signal is filtered, so the more information is lost.

In order to qualitatively compare the results of my code I print in a common window the output of the filter (red) and the original fully clean signal whose value is predicted (blue).

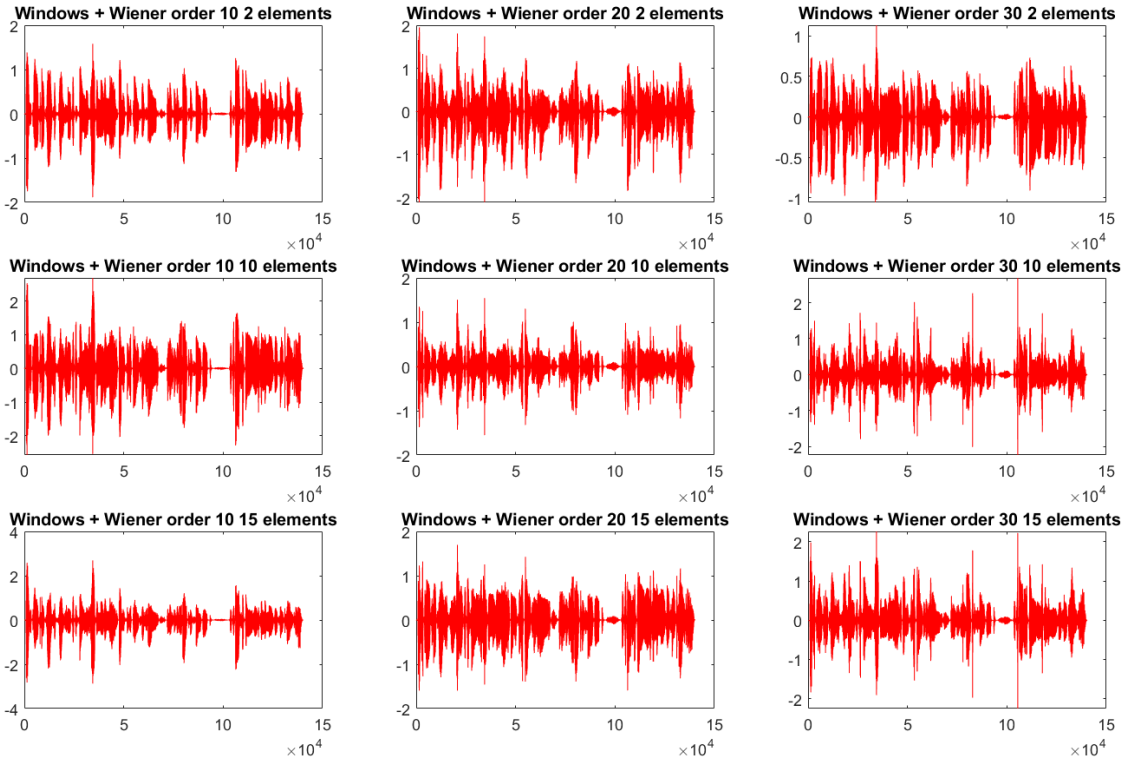

Figure 5: Price prediction