Tecnológico de Monterrey **Campus Ciudad de México** **School of Engineering and Sciences** **Department of Mechatronics**	**TE2019 Digital Signal Processing Laboratory** Student's 1 name and ID: Alfredo Zhu Chen - A01651980 Student's 2 name and ID: Luis Arturo Dan Fong - A016506720 Student's 3 name and ID: Miguel Alejandro Cruz Banda - A01656527 Lab # 5 Linear and non-linear filtering Date: 26/03/2020

1 INTRODUCTION

A linear filter is an electronic filter that applies a linear operator to a time-varying signal. They are widely used in signal processing. One of its most frequent applications is the elimination of unwanted frequencies from a certain input signal or, conversely, discriminating a certain frequency from the others[1].
Linear filters can be divided into two classes: infinite impulse response (IIR) filters and finite impulse response (FIR) filters[1]:

- FIR filters can be described as a weighted sum of inputs with a certain delay. For these filters, if the input at a given instant is zero, the output will be zero from an instant after the delays induced by the filter. In this way, there will only be a response for a finite time.
- IIR filters, on the other hand, can present output even when the input is zero, if the initial conditions are different from zero. The filter energy will decay over time, but it will not become zero. Therefore, the impulse response extends infinitely.

In signal processing a non-linear filter is a filter whose output is not a linear function of its input, this filter has many applications, especially in the removal of noise that is not additive. Non-linear filters are considerably more difficult to use and design than the linear filters, because the signal analysis and mathematical tools cannot be used on them, an example of this is the impulse response and the frequency response[2].
Even though the bases for both linear and non-linear filters were given by scientists in the start of the 20th century the knowledge and research they did remains in modern day technology[3]. In today's world the usage of filtering for cleaning images, audio or other types of input signals is fundamental for most day to day moden activities; such as face recognition in your phone and AI assistant in most IoT or domotics applications. Both linear and non-linear filters bring different capabilities for analyzing and obtaining specific information from an input signal [4].

OBJECTIVES

1. To understand the difference between linear and non-linear filters
2. To implement linear and non-linear filtering routines in MATLAB

2 MATERIALS & METHODS

The materials used in this laboratory were:

- 1 PC or laptop with at least 4GB RAM and and 10MB of space
- MATLAB R2019a or newer versions plus the Audio with the Signal Processing Toolboxes
- Scripts average.m and prob2.m provided by the instructor

2.1 Linear filtering

The function *averager(M,x)*, provided by the instructor of the course, takes *M* and *x* as arguments, the first one is the length of the moving average filter and the second one is the signal which would be filtered. A

variable *b* was defined as a vector of ones divided by length of the filter *M*, these were also the coefficients for present and previous values of the input. Then, by using MATLAB function *filter(b,a,x),* the signal x should be filtered using rational transfer function defined by the numerator and denominator coefficients b and a, which *b* was already defined and *a* was 1 because the filter used was a non-recursive system and it does not depend on the previous values of the output.
The input signal (1) was created within the interval *n* of 100 values in the code(see appendix). By using the MATLAB function *randn(sz1,...,szN),* it was possible to create a normally distributed random vector of numbers for a specific size(1xN in our case). This vector of noise multiplied by 0.3 was added as noise to the input signal and the sum was stored into variable *y.* After this, the *averager(M,x)* function was applied to the moving average filter to the noisy signal, 3 different lengths of the filter(5,10,15) were used for later analysis and comparison purpose in section 3.1

$$x[n] = cos(n\pi/16) \quad (1)$$

Finally, the signals were plotted in the same figure with the MATLAB *plot(Y)* function and *hold on* command. A legend was also created with MATLAB function *legend(label1,...,labelN)* in order to distinguish the plots and the labels were defined as the order of the plots were created.

2.2 Non-linear filtering

The input signal *x* was created with an interval of 200 values from 0 to 199, the first half first half of the signal is defined as (2) and the last half values of the signal are all 0. A vector with the same size of the input signal containing impulse noise was created. For each step in a *for* loop, a random number was created and the value of the noise in that index would be -1.5 if it was greater or equal than 0.95, otherwise it would be 0. The impulsive noise vector was added to the input signal and stored in variable *y1.* .

$$x[n] = cos(n\pi/256) \quad (2)$$

After creating the noisy signal, it was filtered with length 15 of a linear moving average filter as it has been done in section 2.1. For comparison purposes, a non-linear median filter was used to filter the noisy signal. The value of each index was taking the median considering two values from left and right. Since there are no defined values before the two first and two last values of the noisy signal, the median for these cases were declared outside the *for* loop and ceros were taken into account for those non defined values. To obtain the median, MATLAB function *median(A)* is useful to accomplish this. After obtaining the median for the beginning and final values of the noisy signal, the *for* loop is used to obtain the other median values remaining.
The only part of the code added to prob2.m for this section is to plot the resulting signals. Since it is required to compare the differences between the linear and non-linear filter, a figure containing all separated plots is created with *subplot(m,n,p),* which p is the position of the image in the *mxn* grid*,* and *plot(y),* which *y* is the signal to plot . These will be useful to discuss and analyse later in section 3.2.

3 RESULTS AND DISCUSSION

3.1 Linear filtering

The following figure 3.1.1 shows the plot of the noisy signal and the three averaged signals with 5th, 10th and 15th order moving-average filters. As it can be seen, all three averaged signals are less noisy, this is because the moving-average filter is taking the average value for a certain amount of previous values determined by the number the coefficients used and it has a smoothing effect to remove high frequencies. The higher the order of the filter is, the less noisy is the signal in this case. It is also important to notice that all the averaged signals are delayed with respect to the noisy signal, this is because the averager filter needs to accumulate previous values. Since getting higher order of the filter means that more previous values are accumulated, the output signal for this case would be more delayed, such as the 15th order averaged signal in purple color for this example.

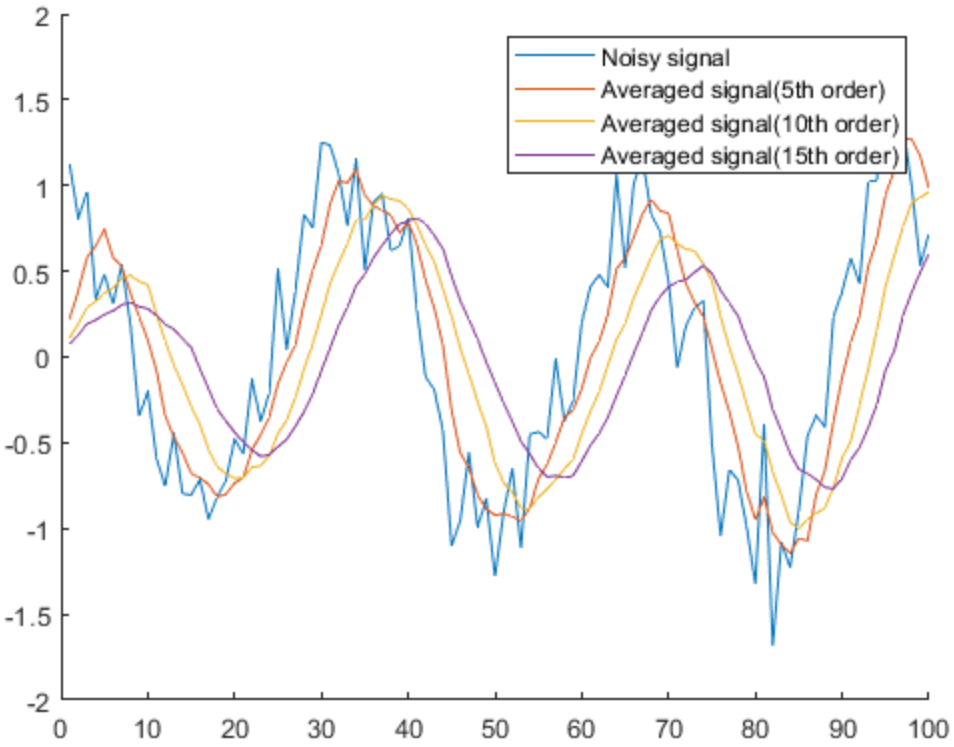

Figure 3.1.1 Noisy signal with the averaged signals

3.3 Non-linear filtering

As shown in figure 3.1.2, it is possible to compare the differences between using a linear filter and a non-linear filter for the noisy signal.Some random impulsive noises are added to the input signal *x* and it generates the noisy signal *y.* On one hand, by applying a non-linear median filter to the noisy signal(third graph of figure 3.1.2), it is notable that the result *z1* is very similar to the original input signal, so this filter has very good performance to remove impulsive noise. Since each sample of the signal is taking the median with values of its both sides, the impulsive noise will be removed since they are never chosen to be the median. On the other hand, from the last graph of the same figure, applying a linear filter to counter the impulsive noise gets poor results in this case. The output signal *z2* has no longer impulsive noise, but the parts which were the noise have some small deformation due to the averaging process with the impulsive noise, so it seems that a moving-average linear filter is not the best solution for this kind of noise problem.

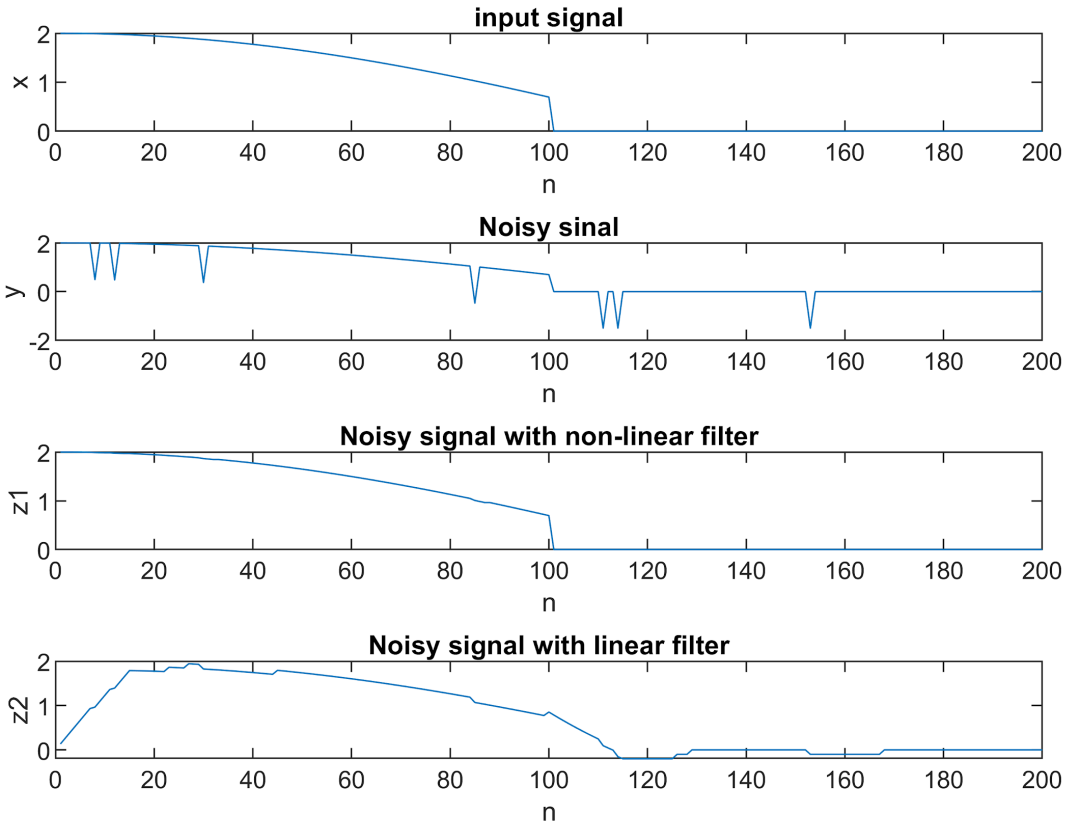

Figure 3.2.1 Comparison of linear filter and non-linear filter

4 CONCLUSION

In conclusion, the objectives planned were fulfilled, being that after testing both linear and non-linear filter routines, the resulting graphs show us the main differences of using each type of filter. The linear filter as the averaging filter is commonly used to smooth a signal by attenuating the high-frequency components of the signal due to the noise, while a non-linear filter as the median filter is more effective to remove impulse noise. Likewise, we investigated further and comprehend how both filters worked and used the information to implement our knowledge in the matlab script.

The importance of these filters nowadays is great, since you have different applications in areas in which we as ITSE students attract us, and that in the future we could apply or design. The elimination of noise is a process that always has to be dealt with when working with different signals, it also has an application in image processing, which ranges from a simple photograph to facial recognition on cell phones. It can be seen that digital signal processing is an activity that today has many applications, which although not everyone sees them, allows to improve human quality of life and even make signal analysis much easier.

5 REFERENCES

[1] Stanley, W. D., Dougherty, G. R., Dougherty, R., & Saunders, H. (1988). Digital signal processing.

[2] Miyara, F. (2004). Filtros activos. *Cátedra de Electrónica III FCEIA-UNR. Rosario*.

[3] Broomhead, D. S., Huke, J. P., & Muldoon, M. R. (1992). Linear filters and non-linear systems. Journal of the Royal Statistical Society: Series B (Methodological), 54(2), 373-382.

[4] MOHD PADZIL, F. (2016). LINEAR AND NONLINEAR FILTER FOR IMAGE PROCESSING USING MATLAB'S IMAGE PROCESSING TOOLBOX. Recuperado el 24 Marzo 2021, de https://researchrepository.murdoch.edu.au/id/eprint/30815/1/whole.pdf

6 APPENDIX

1. Linear filtering

```
%Clearing variables, command window and close all figures
clear all
close all
clc

N= 100; %number of values
n=0:N-1; %interval

x=cos(pi*n/16); %input function
noise = 0.3*randn(1,N); %random noise
y= x+noise; %add noise to the input signal

%Apply averager filter with length M to input signal y
z1=averager(5,y);
z2=averager(10,y);
z3=averager(15,y);

%Plot all the signals in the same figure
hold on
plot(y)
plot(z1)    %noisy signal filtered with averager of length 5
plot(z2)    %noisy signal filtered with averager of length 10
plot(z3)    %noisy signal filtered with averager of length 15
legend('Noisy signal', 'Averaged signal(5th order)','Averaged signal(10th order)','Averaged signal(15th
order)')
```

2. Non-linear filtering

```
%%
% Linear and non-linear filtering
%%
clear all
close all
clc
N =200;n=0:N-1; %Number of samples and interval
x=[2*cos(pi*n(1:100)/256) zeros(1,100)]; %input signal, half cosine and half zeros

% impulsive noise
for m = 1:N,    %Interating over each sample
   d = rand(1,1); %Generating random number from 0 to 1
   if d >= 0.95,
      noise(m) = -1.5;%Create impulsive noise of -1.5 if d greater or equal than 0.95
   else
      noise(m) = 0 ;  %otherwise no noise in this sample
```

```
   end
end
y1=x+noise; %Add noise to the input signal


% linear filtering
z2=averager(15,y1);

% non-linear filtering -- median filtering
%Applying median to the first two and last two samples of the signal
z1(1)=median([0 0 y1(1) y1(2) y1(3)]);
z1(2)=median([0 y1(1) y1(2) y1(3) y1(4)]);
z1(N-1)=median([y1(N-3) y1(N-2) y1(N-1) y1(N) 0]);
z1(N)=median([y1(N-2) y1(N-1) y1(N) 0 0]);
%Applying median to the rest of the samples
for k=3:N-2,
   z1(k)=median([y1(k-2) y1(k-1) y1(k) y1(k+1) y1(k+2)]);
end

%% Added section
%Plot of input signal, noisy signal, noisy signal with linear filter and
%noisy signal with non-linear filter
subplot(4,1,1)
plot(x)
title("input signal")
xlabel("n")
ylabel("x")
subplot(4,1,2)
plot(y1)
title("Noisy sinal")
xlabel("n")
ylabel("y")
subplot(4,1,3)
plot(z1)
title("Noisy signal with non-linear filter")
xlabel("n")
ylabel("z1")
subplot(4,1,4)
plot(z2)
title("Noisy signal with linear filter")
xlabel("n")
ylabel("z2")
```