

Assignment # 2



Fall 2023

CSE-402 Digital Signal Processing

Submitted by: **Ali Asghar**

Registration No.: **21PWCSE2059**

Class Section: **C**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Submitted to:

Engr. Ihsan Ul Haq

Date:

7th November 2023

Department of Computer Systems Engineering
University of Engineering and Technology, Peshawar

Code:

```
Editor - D:\Uni\DSP\Assignment 2\Smoothing.m
Task1.m x Smoothing.m x rand_randn.m x +
1 - fs = 1000;
2 - t = 0:1/fs:1;
3 - A = 5;
4 - f=5;
5 - x = A * sin(2*pi*f*t);
6
7 - meanNoise1 = 0;
8 - std_dev_noise1 = 1;
9
10 - meanNoise2 = 1;|
11 - std_dev_noise2 = 2;
12
13 - meanNoise3 = 2;
14 - std_dev_noise3 = -1;
15
16 - noise1 = meanNoise1 + std_dev_noise1 * randn(size(t));
17 - noise2 = meanNoise2 + std_dev_noise2 * randn(size(t));
18 - noise3 = meanNoise3 + std_dev_noise3 * randn(size(t));
19
20 - noisySig1 = x + noise1;
21 - noisySig2 = x + noise2;
22 - noisySig3 = x + noise3;
23
24 - % Create plots for the clean and noisy signals
25 - figure;

25 - figure;
26
27 - % Plot the clean sine wave
28 - subplot(2,2,1);
29 - plot(t, x);
30 - title(['Sine Wave with A = ' num2str(A) ' And f = ' num2str(f)]);
31 - xlabel('Time (s)');
32 - ylabel('Amplitude');
33
34 - % Plot the noisy signals with different standard deviations
35
36 - subplot(2,2,2);
37 - plot(t, noisySig1);|
38 - title(['Noisy Signal 1 (Std Dev = ' num2str(std_dev_noise1), ' Mean = ' num2str(meanNoise1) ' )']);
39 - xlabel('Time (s)');
40 - ylabel('Amplitude');
41
42
43 - subplot(2,2,3);
44 - plot(t, noisySig2);
45 - title(['Noisy Signal 2 (Std Dev = ' num2str(std_dev_noise2), ' Mean = ' num2str(meanNoise2) ' )']);
46 - xlabel('Time (s)');
47 - ylabel('Amplitude');
48
49 - subplot(2,2,4);
```

```

49- subplot(2,2,4);
50- plot(t, noisySig3);
51- title(['Noisy Signal 3 (Std Dev = ' num2str(std_dev_noise3), ' Mean = ' num2str(meanNoise3) ' )']);
52- xlabel('Time (s)');
53- ylabel('Amplitude');
54-
55-
56- %Filtering Process
57-
58- window_size1 = 3; % Size of the moving average window
59- window_size2 = 5; % Size of the moving average window
60- window_size3 = 7; % Size of the moving average window
61-
62- % Filter the noisy signals using movmean
63- filteredSig1_1 = movmean(noisySig1, window_size1);
64- filteredSig1_2 = movmean(noisySig1, window_size2);
65- filteredSig1_3 = movmean(noisySig1, window_size3);
66-
67-
68- % Filter the noisy signals using movmean
69- filteredSig2_1 = movmean(noisySig2, window_size1);
70- filteredSig2_2 = movmean(noisySig2, window_size2);
71- filteredSig2_3 = movmean(noisySig2, window_size3);
72-

```

```

74- % Filter the noisy signals using movmean
75- filteredSig3_1 = movmean(noisySig3, window_size1);
76- filteredSig3_2 = movmean(noisySig3, window_size2);
77- filteredSig3_3 = movmean(noisySig3, window_size3);
78-
79- % Create plots for the filtered signals
80- figure;
81-
82- subplot(3,1,1);
83- plot(t, filteredSig1_1);
84- title(['Filtered Sine Wave (Std Dev = ' num2str(std_dev_noise1), ' Mean = ' num2str(meanNoise1) ' ) & window size = ' num2str(window_size1)']);
85- xlabel('Time (s)');
86- ylabel('Amplitude');
87-
88- subplot(3,1,2);
89- plot(t, filteredSig1_2);
90- title(['Filtered Sine Wave (Std Dev = ' num2str(std_dev_noise1), ' Mean = ' num2str(meanNoise1) ' ) & window size = ' num2str(window_size2)']);
91- xlabel('Time (s)');
92- ylabel('Amplitude');
93-
94- subplot(3,1,3);
95- plot(t, filteredSig1_3);
96- title(['Filtered Sine Wave (Std Dev = ' num2str(std_dev_noise1), ' Mean = ' num2str(meanNoise1) ' ) & window size = ' num2str(window_size3)']);
97- xlabel('Time (s)');

```

```

97- xlabel('Time (s)');
98- ylabel('Amplitude');
99-
100-
101- % Create plots for the filtered signals
102- figure;
103-
104- subplot(3,1,1);
105- plot(t, filteredSig2_1);
106- title(['Filtered Sine Wave (Std Dev = ' num2str(std_dev_noise2), ' Mean = ' num2str(meanNoise2) ' ) & window size = ' num2str(window_size1)']);
107- xlabel('Time (s)');
108- ylabel('Amplitude');
109-
110- subplot(3,1,2);
111- plot(t, filteredSig2_2);
112- title(['Filtered Sine Wave (Std Dev = ' num2str(std_dev_noise2), ' Mean = ' num2str(meanNoise2) ' ) & window size = ' num2str(window_size2)']);
113- xlabel('Time (s)');
114- ylabel('Amplitude');
115-
116- subplot(3,1,3);
117- plot(t, filteredSig2_3);
118- title(['Filtered Sine Wave (Std Dev = ' num2str(std_dev_noise2), ' Mean = ' num2str(meanNoise2) ' ) & window size = ' num2str(window_size3)']);
119- xlabel('Time (s)');
120- ylabel('Amplitude');
121-

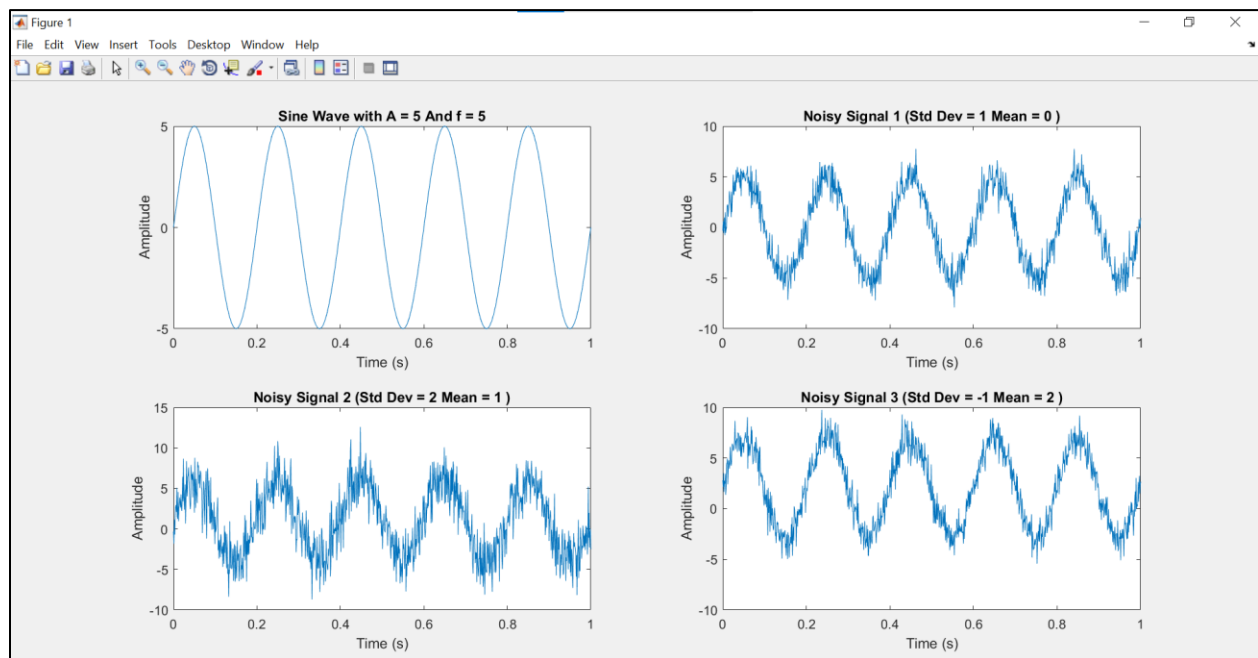
```

```

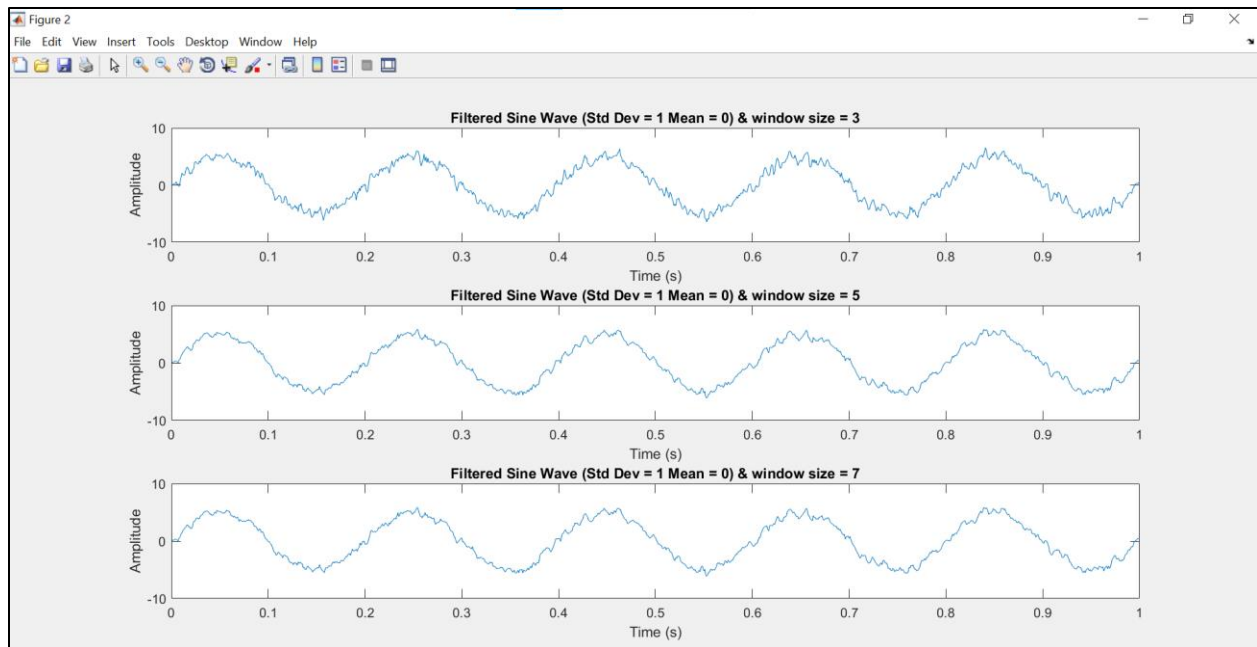
120- ylabel('Amplitude');
121-
122-
123- % Create plots for the filtered signals
124- figure;
125-
126- subplot(3,1,1);
127- plot(t, filteredSig3_1);
128- title(['Filtered Sine Wave (Std Dev = ' num2str(std_dev_noise3), ' Mean = ' num2str(meanNoise3) ') & window size = ' num2str(window_size1)']);
129- xlabel('Time (s)');
130- ylabel('Amplitude');
131-
132- subplot(3,1,2);
133- plot(t, filteredSig3_2);
134- title(['Filtered Sine Wave (Std Dev = ' num2str(std_dev_noise3), ' Mean = ' num2str(meanNoise3) ') & window size = ' num2str(window_size2)']);
135- xlabel('Time (s)');
136- ylabel('Amplitude');
137-
138- subplot(3,1,3);
139- plot(t, filteredSig3_3);
140- title(['Filtered Sine Wave (Std Dev = ' num2str(std_dev_noise3), ' Mean = ' num2str(meanNoise3) ') & window size = ' num2str(window_size3)']);
141- xlabel('Time (s)');
142- ylabel('Amplitude');
143-
144-

```

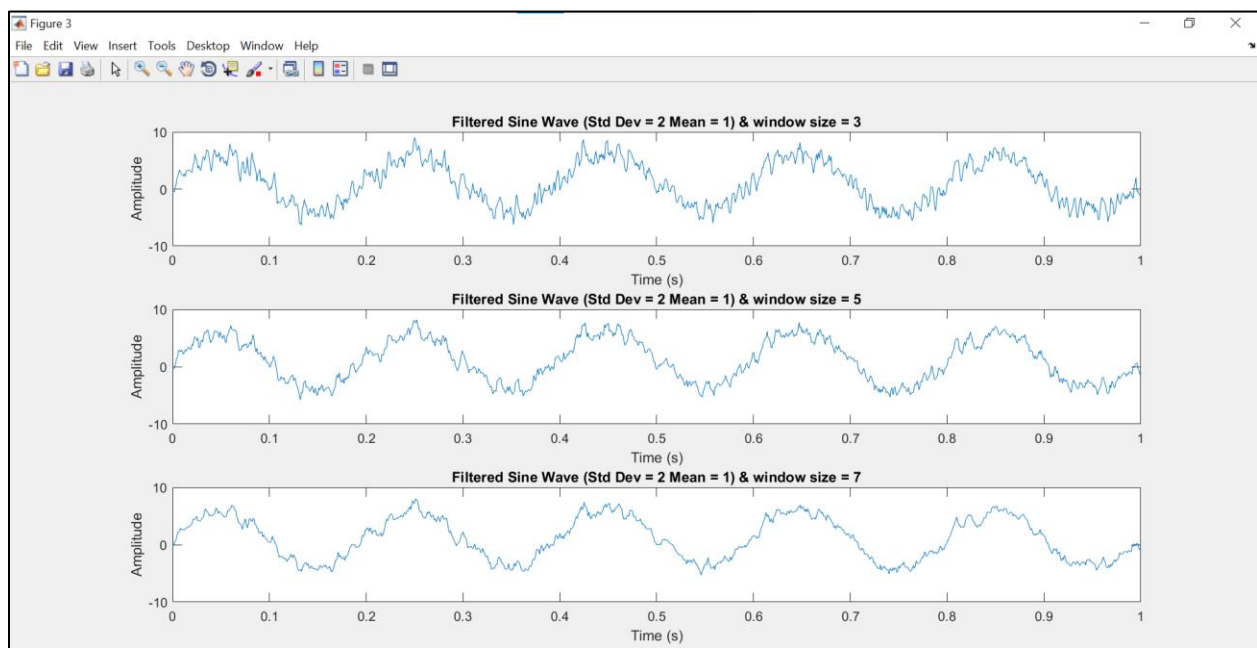
Output:



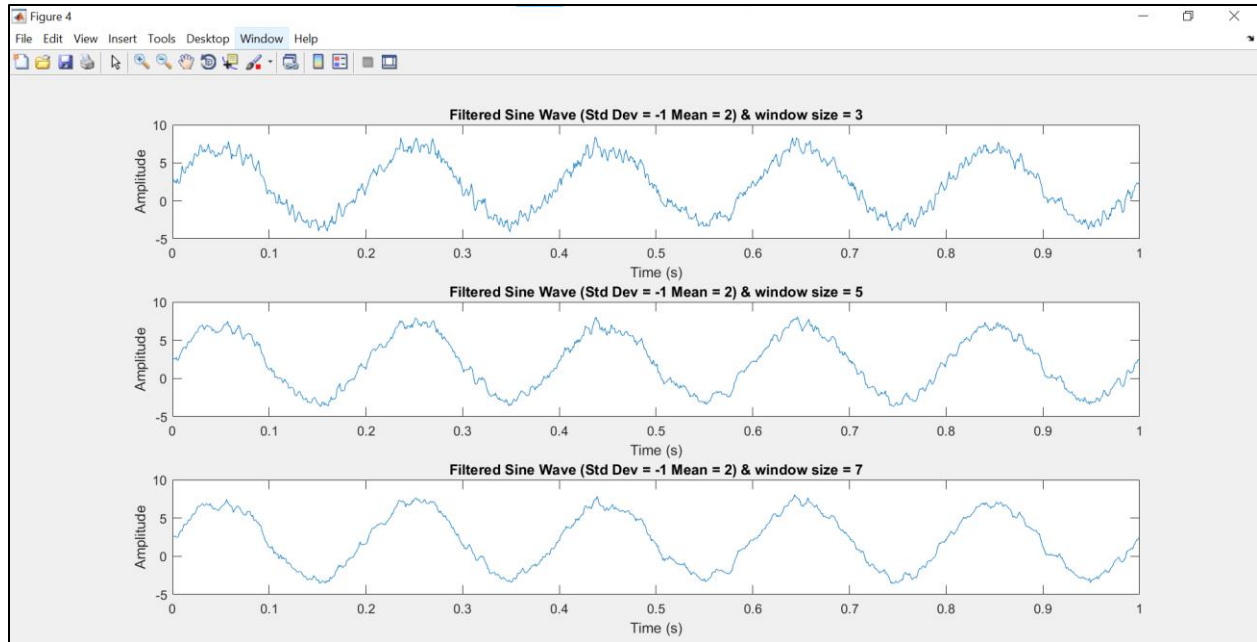
Generation of 3 Noisy Gaussian Signals for different mean and standard deviation



Filtering of Noisy Signal 1 for different window sizes



Filtering of Noisy Signal 2 for different window sizes



Filtering of Noisy Signal 3 for different window sizes

Remarks on Output:

With the help of `movmean()` function, I was able to filter the noisy signal. The choice of window size is of interest in here. For window size 3 and 5, the output isn't very smooth. For window size 7, the output waveform looks smoothen and denoised.

`movmean()` is a MATLAB function for moving average filtering, which smooths time series data by calculating the mean within a sliding window of a specified size (k). It's commonly used to reduce noise and emphasize underlying trends or patterns in data.

In summary, the code above showcases a practical approach to signal processing, including noise generation, smoothing, and analysis. It allows for a clear visualization and assessment of the effects of smoothing on noisy signals, which is a common task in various signal processing applications.