EXPERIMENT NO: 2

1. Aim: Implementation of the Denoising audio signals using filters.

2. Software used: MATLAB R2024b

3. Theory:

Audio signals are often corrupted by noise due to various factors such as environmental disturbances, recording equipment limitations, and transmission interference. Denoising techniques aim to remove noise while preserving the original signal characteristics. Filtering techniques play a crucial role in this process.

Types of Filters Used in Audio Denoising

Mean Filter: The mean filter is a simple smoothing technique that replaces each sample in the signal with the average of its neighbouring samples within a defined window size.

$$y_{\text{mean}}(n) = \frac{1}{N} \sum_{k=-h}^{h} x(n+k)$$

- x(n) is the original noisy signal,
- N is the filter size,
- h is the half window size.

Median Filter: The median filter replaces each sample with the median value of the samples within the window. It is particularly effective at removing impulse noise (salt-and-pepper noise).

$$y_{\text{median}}(n) = \text{median}\{x(n-h), \ldots, x(n), \ldots, x(n+h)\}$$

Median is the middle value of the sorted window elements.

Weighted Average Filter

- The weighted average filter assigns different weights to each sample within the window, prioritizing some values over others based on their importance. Larger weights are assigned to central values to preserve the local structure of the signal.

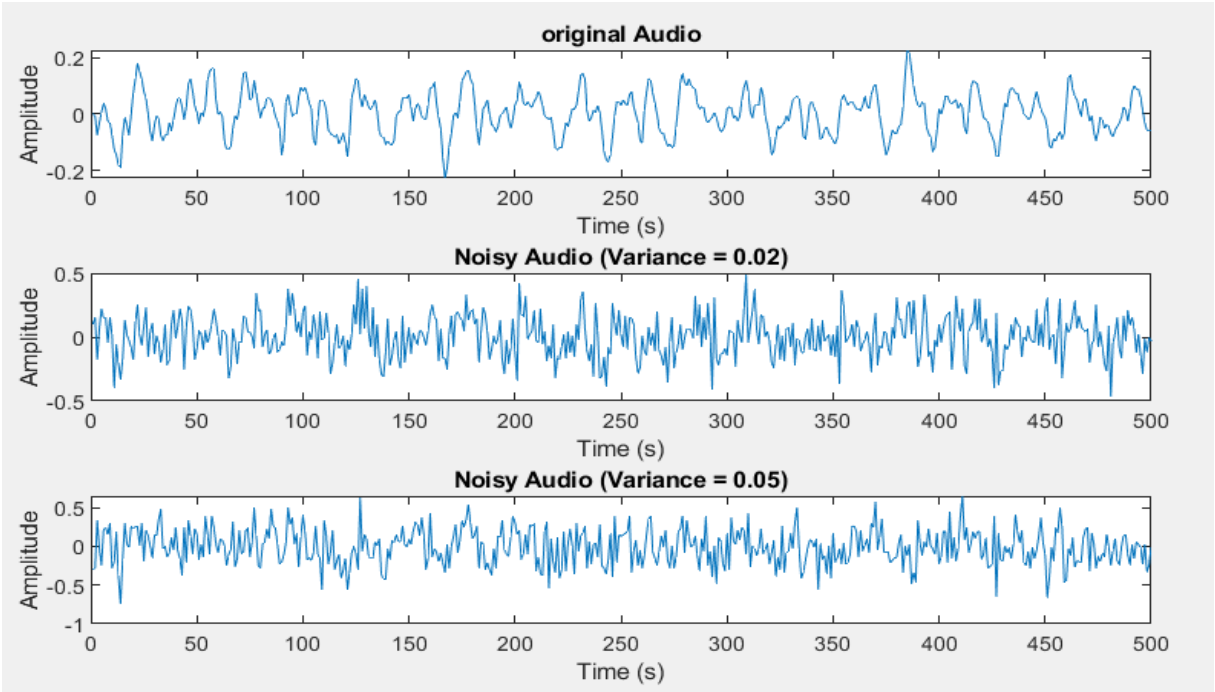
$$y_{\text{weighted}}(n) = \frac{\sum_{k=-h}^{h} w_k \cdot x(n+k)}{\sum_{k=-h}^{h} w_k}$$

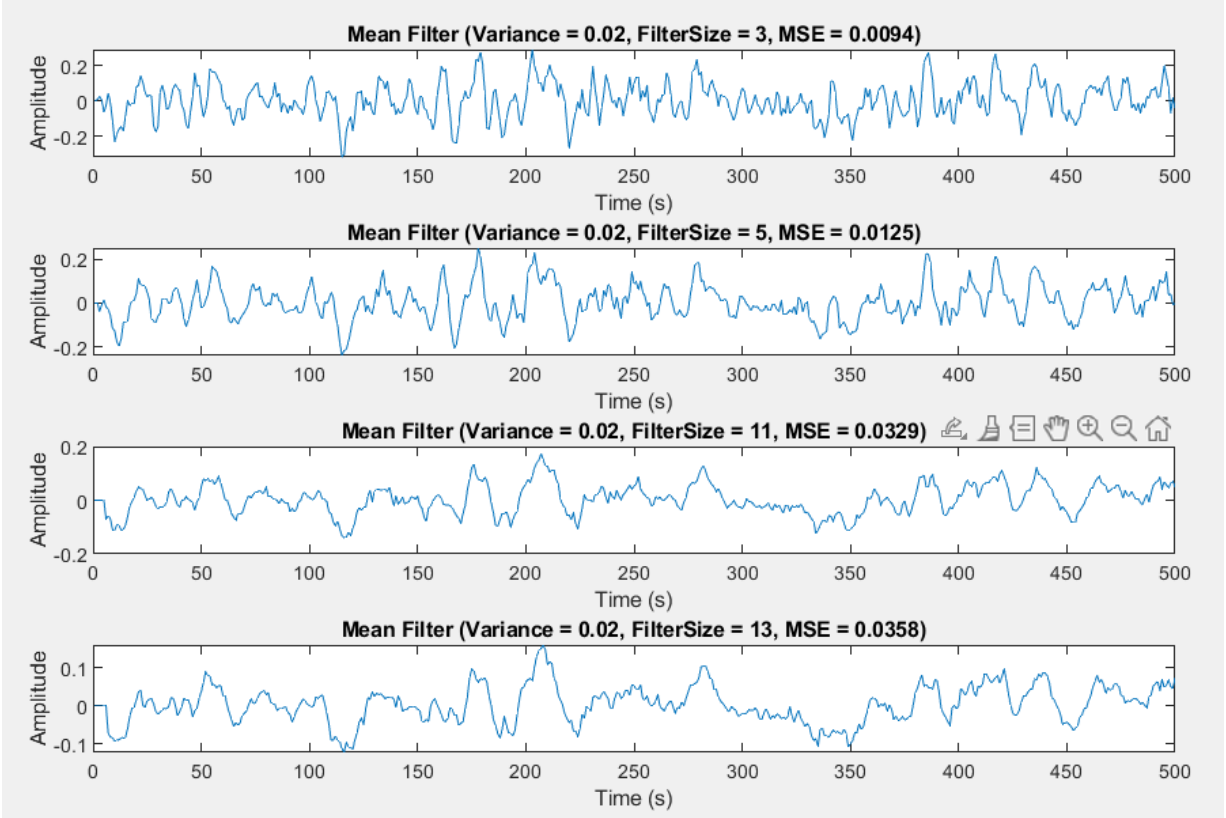
W_k represents the weights.

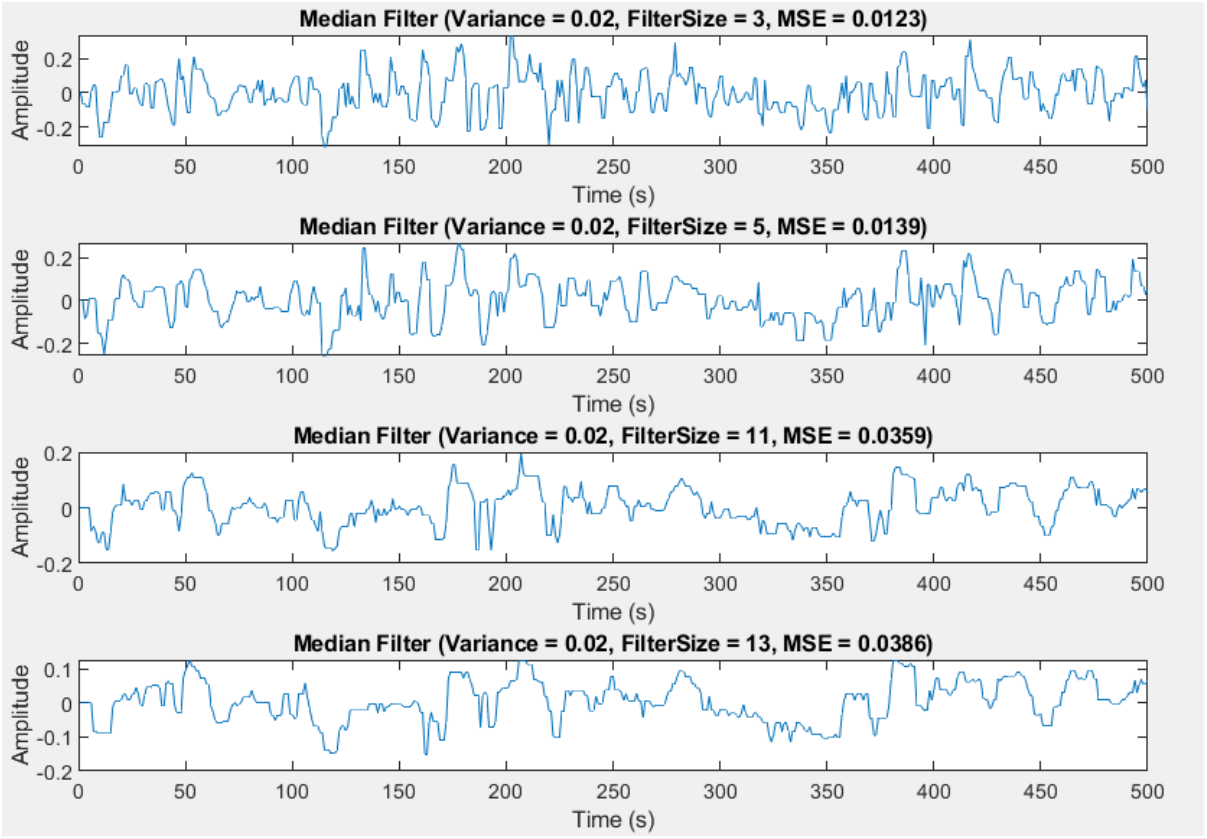
Weighted filter kernels:

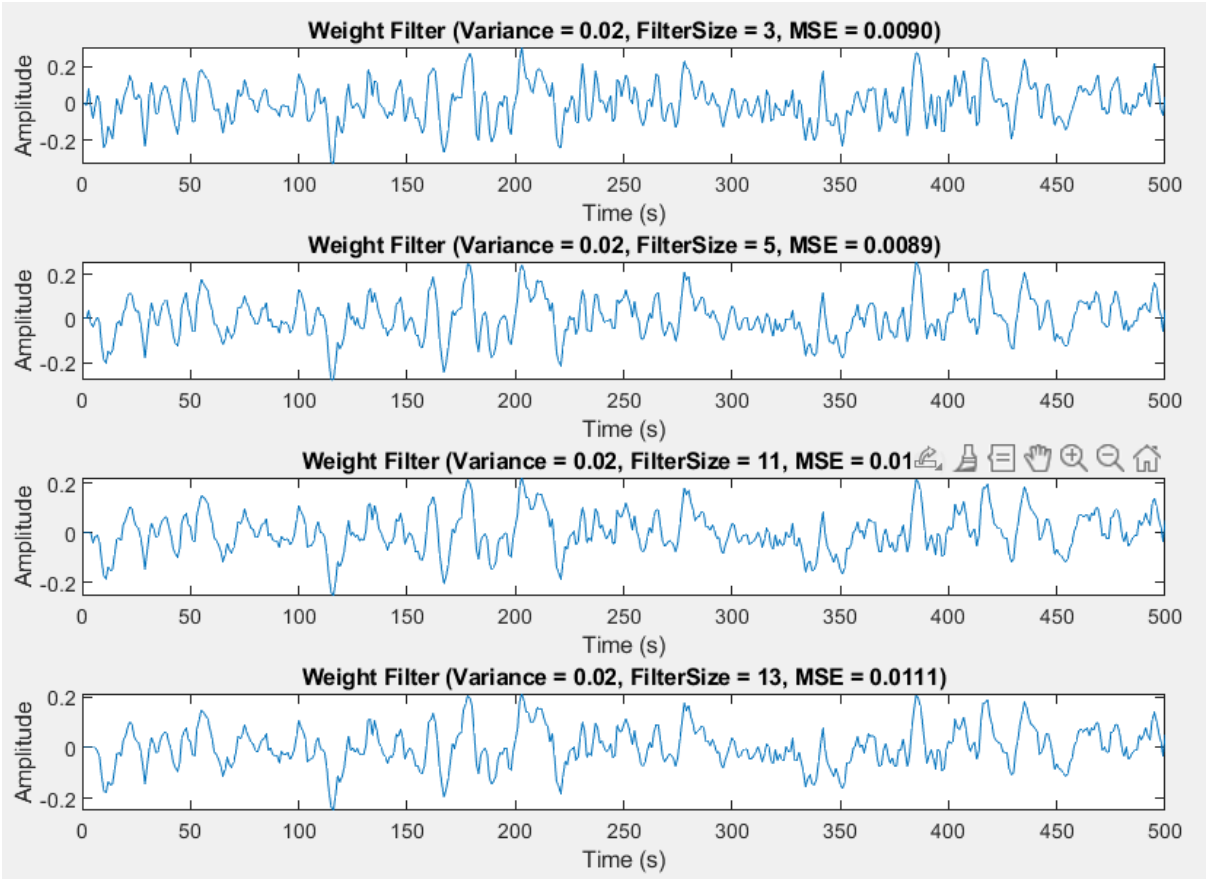
k = 3 [1, 2, 1]
k = 5 [1, 2, 4, 2, 1]
k = 11 [1, 2, 4, 8, 16, 32, 16, 8, 4, 2, 1]
k = 13 [1, 2, 4, 8, 16, 32, 64, 32, 16, 8, 4, 2, 1]

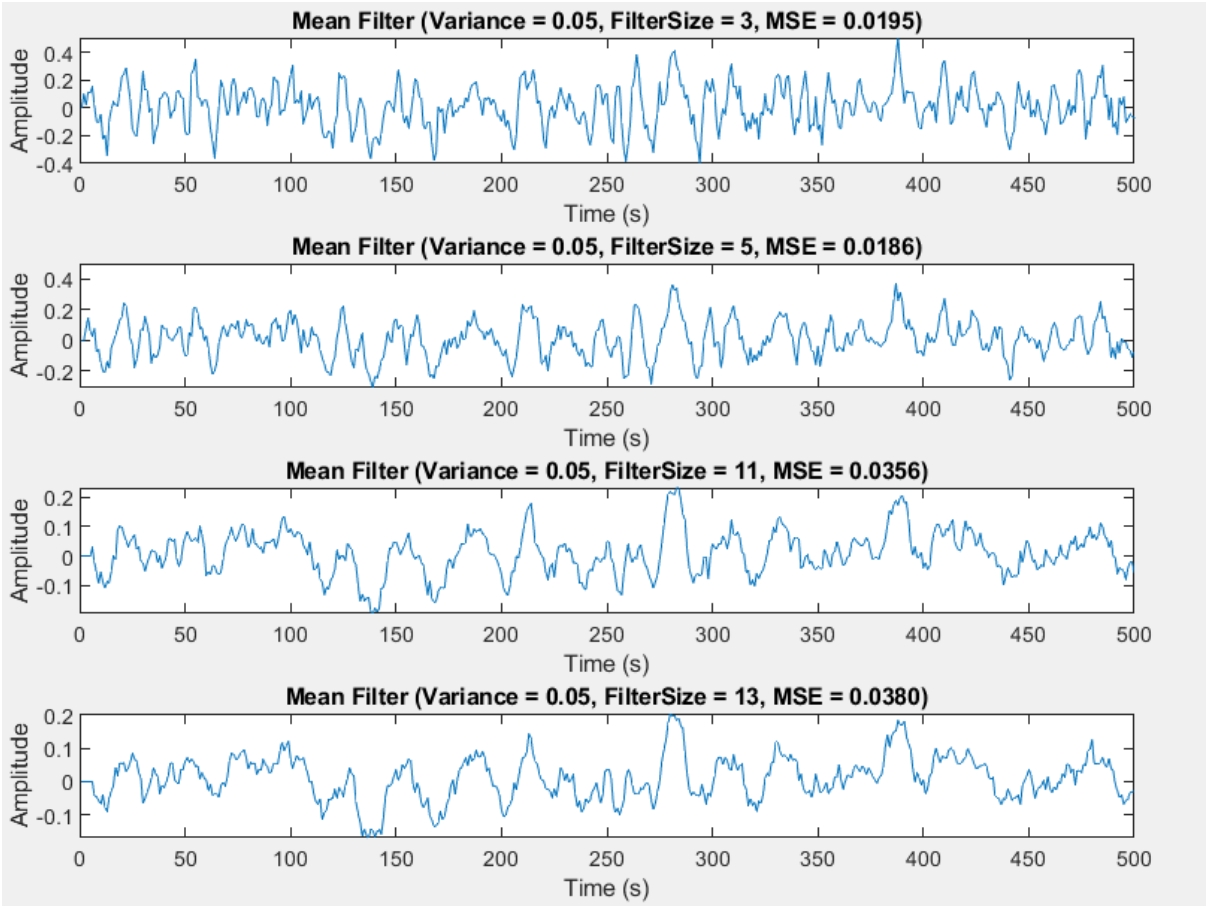
4. Result:

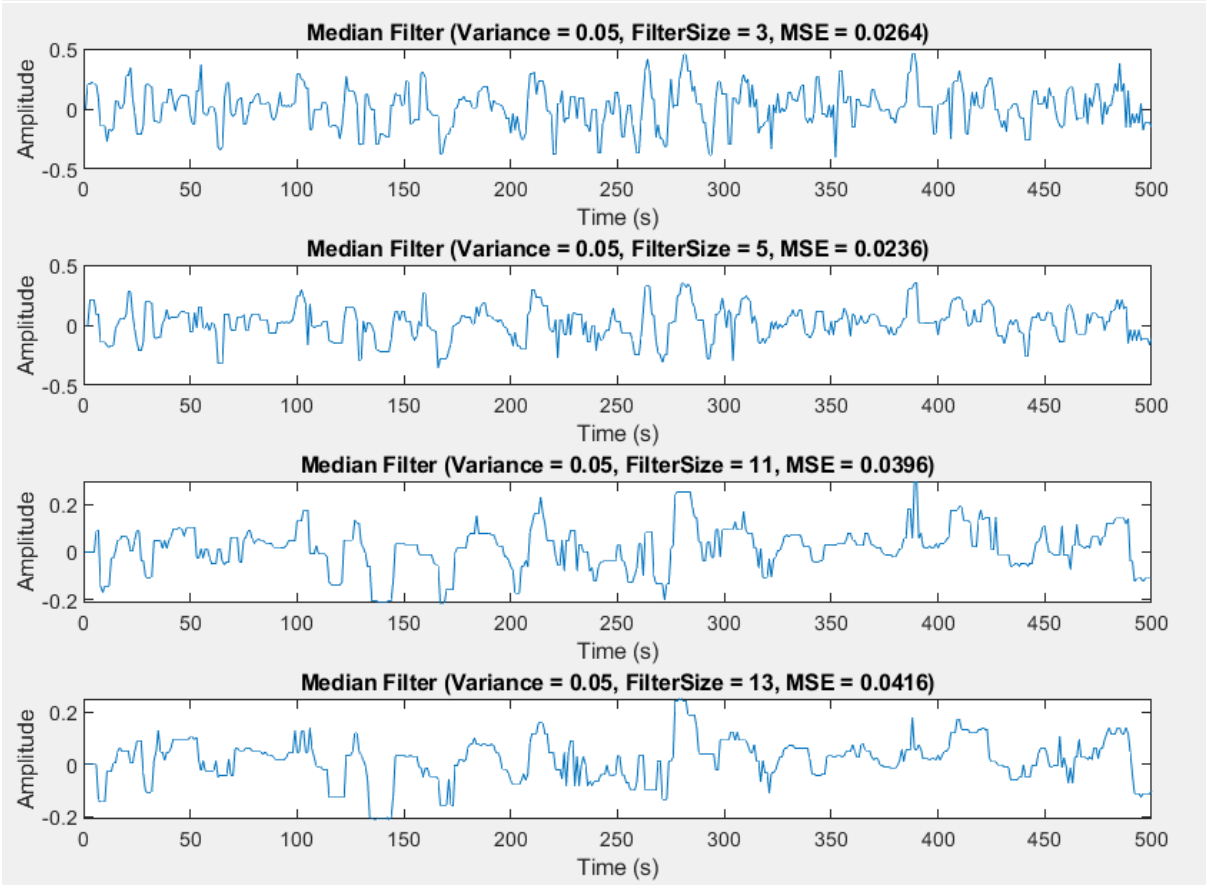

original Audio
Amplitude
Time (s)
Noisy Audio (Variance = 0.02)
Amplitude
Time (s)
Noisy Audio (Variance = 0.05)
Amplitude
Time (s)

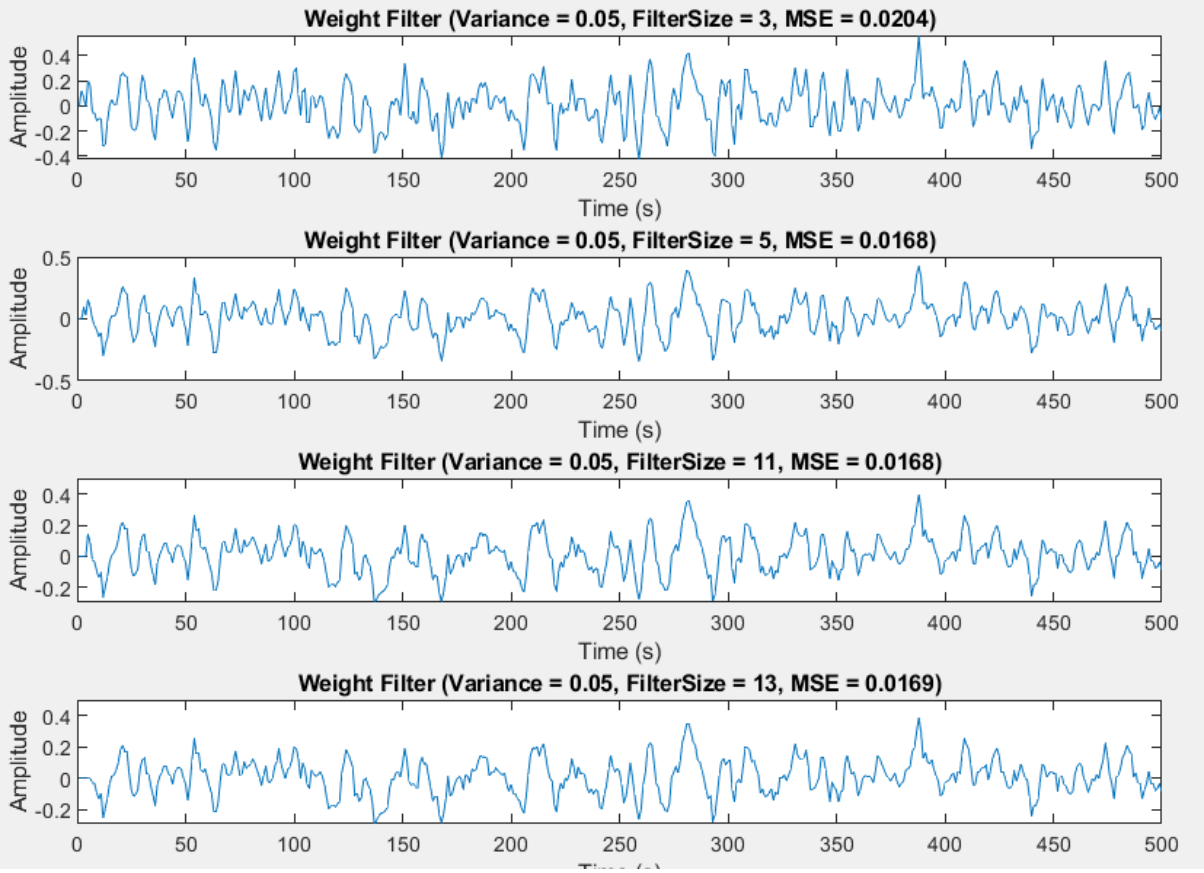

Mean Filter (Variance = 0.02, FilterSize = 3, MSE = 0.0094)
Amplitude
Time (s)
Mean Filter (Variance = 0.02, FilterSize = 5, MSE = 0.0125)
Amplitude
Time (s)
Mean Filter (Variance = 0.02, FilterSize = 11, MSE = 0.0329)
Amplitude
Time (s)
Mean Filter (Variance = 0.02, FilterSize = 13, MSE = 0.0358)
Amplitude
Time (s)


Median Filter (Variance = 0.02, FilterSize = 3, MSE = 0.0123)
Median Filter (Variance = 0.02, FilterSize = 5, MSE = 0.0139)
Median Filter (Variance = 0.02, FilterSize = 11, MSE = 0.0359)
Median Filter (Variance = 0.02, FilterSize = 13, MSE = 0.0386)
Amplitude
Time (s)


Weight Filter (Variance = 0.02, FilterSize = 3, MSE = 0.0090)
Weight Filter (Variance = 0.02, FilterSize = 5, MSE = 0.0089)
Weight Filter (Variance = 0.02, FilterSize = 11, MSE = 0.01
Weight Filter (Variance = 0.02, FilterSize = 13, MSE = 0.0111)
Amplitude
Time (s)


Mean Filter (Variance = 0.05, FilterSize = 3, MSE = 0.0195)
Amplitude
Time (s)
Mean Filter (Variance = 0.05, FilterSize = 5, MSE = 0.0186)
Amplitude
Time (s)
Mean Filter (Variance = 0.05, FilterSize = 11, MSE = 0.0356)
Amplitude
Time (s)
Mean Filter (Variance = 0.05, FilterSize = 13, MSE = 0.0380)
Amplitude
Time (s)


Median Filter (Variance = 0.05, FilterSize = 3, MSE = 0.0264)
Amplitude
Time (s)
Median Filter (Variance = 0.05, FilterSize = 5, MSE = 0.0236)
Amplitude
Time (s)
Median Filter (Variance = 0.05, FilterSize = 11, MSE = 0.0396)
Amplitude
Time (s)
Median Filter (Variance = 0.05, FilterSize = 13, MSE = 0.0416)
Amplitude
Time (s)



5. Discussion:

Two levels of noise variance were introduced: **0.02** (low noise) and **0.05** (high noise). The results indicate the following trends:

1. **Low Variance (0.02):**
    - All three filters (mean, median, weighted) were relatively effective in reducing noise while preserving the audio signal.
    - The **mean filter** performed well for smaller filter sizes but introduced minor smoothing artifacts.
    - The **median filter** was able to effectively remove impulsive components without much distortion.
    - The **weighted filter** performed the best by balancing noise reduction and preserving signal details.
    - MSE values were generally lower, indicating that the original signal was well preserved.
2. **High Variance (0.05):**
    - With an increase in noise variance, the effectiveness of all filters was challenged.
    - The **mean filter** struggled, as larger amounts of noise caused excessive smoothing, leading to a significant loss of detail in the audio signal.

- The **median filter** still performed reasonably well, though it required larger kernel sizes to handle the increased noise.
- The **weighted filter** outperformed the other two by providing the best trade-off between noise reduction and detail preservation, but it also introduced some delays due to larger kernel sizes.
- MSE values increased across all filters, highlighting the difficulty in recovering the original signal under high noise conditions.

As noise variance increases, the filters need larger kernel sizes to effectively suppress noise, but this comes at the cost of signal distortion.

Effect of Filter Kernel Size

The filter kernel sizes used were: **3, 5, 11, and 13**. The following observations were made:

1. **Small Kernel Sizes (3, 5):**
   - **Mean Filter:** Smaller kernels provided less smoothing, retaining more signal detail but failing to remove noise effectively.
   - **Median Filter:** For small kernels, it preserved sharp transients well but struggled with significant noise levels.
   - **Weighted Filter:** Showed decent performance with small kernels but had limited impact on high noise levels.
   - MSE values were relatively higher as noise persisted with small kernel sizes.
2. **Large Kernel Sizes (11, 13):**
   - **Mean Filter:** Larger kernels smoothed the noise effectively but led to significant signal distortion, resulting in muffled audio.
   - **Median Filter:** Larger kernel sizes improved noise removal but could introduce artifacts in clean parts of the signal.
   - **Weighted Filter:** Larger kernels resulted in significant noise suppression while maintaining better detail retention compared to other filters.
   - MSE values were lower for larger kernels, indicating improved noise reduction but also higher distortion potential

- **Small Kernels (3, 5):**
  - Better detail preservation but less effective at noise reduction.
  - Suitable for low-noise scenarios.
- **Large Kernels (11, 13):**
  - Strong noise suppression but caused loss of detail and smoothing effects.
  - Effective for high noise levels but may distort the signal.

Comparison of Results Across Different Conditions

Condition	Mean Filter	Median Filter	Weighted Filter
Low Variance (0.02) - Small Kernel	Moderate noise removal, minor distortion	Effective impulse noise removal	Best balance of noise and detail
Low Variance (0.02) - Large Kernel	Good noise removal, smooth but blurry signal	Slightly better noise suppression	Best performance with minimal distortion
High Variance (0.05) - Small Kernel	Ineffective, high noise remains	Moderate performance	Better noise suppression but not sufficient
High Variance (0.05) - Large Kernel	Strong smoothing but significant detail loss	Effective but introduces artifacts	Effective with some loss of sharpness

6. Conclusion:

The experiment showed that the mean filter effectively reduces random noise but blurs details, making it suitable for low noise levels. The median filter excels in removing impulsive noise while preserving signal edges. The weighted filter provides the best balance between noise reduction and detail retention, making it the most effective overall. Larger filter sizes improve noise suppression but can distort the signal, while smaller sizes preserve details but may leave some noise. For low noise, smaller kernels work well, whereas higher noise requires larger kernels, with the weighted filter performing best.

7. Code:

```
load handel.mat
audiowrite('handel.wav', y, Fs);
sound(y, Fs);

var = [0.02, 0.05];
filtersize = [3, 5, 11, 13];
wightfilterkernal{1} = [1, 2, 1];
wightfilterkernal{2} = [1, 2, 4,2, 1];
wightfilterkernal{3} = [1, 2, 4, 8, 16, 32, 16, 8, 4, 2, 1];
wightfilterkernal{4} = [1, 2, 4, 8, 16, 32, 64, 32, 16, 8, 4, 2, 1];
n = length(y);
a = audioread("handel.wav");
figure();
subplot(3,1,1);
plot(a(1:500));
title('original Audio');
xlabel('Time (s)');     ylabel('Amplitude');
```

```
noise = sqrt(var(1)) * randn(size(y));
subplot(3, 1, 2);
audio = y+ noise ; plot(audio(1:500));
title(['Noisy Audio (Variance = ', num2str(var(1)), ')']);
xlabel('Time (s)');     ylabel('Amplitude');
noise = sqrt(var(2)) * randn(size(y));
subplot(3, 1, 3);
audio = y+ noise ; plot(audio(1:500));
title(['Noisy Audio (Variance = ', num2str(var(2)), ')']);
xlabel('Time (s)');     ylabel('Amplitude');
for i = 1:2
   noise = sqrt(var(i)) * randn(size(y));
   audio = y + noise;
   mse_values = zeros(1, 4);
   figure();
   for j = 1:4
      start = filtersize(j);
      hole = floor(start / 2);
      y_mean = zeros(size(y));
      for k = hole + 1 : n - hole
         window = audio(k - hole : k + hole);
         y_mean(k) = mean(window);
      end
      mse_values(j) = mean((y - y_mean).^2);
      subplot(4, 1, j );
      plot(y_mean(1:500));
      title(['Mean Filter (Variance = ', num2str(var(i)), ', FilterSize = ', num2str(filtersize(j)), ',
MSE = ', num2str(mse_values(j), '%.4f'), ')']);
      xlabel('Time (s)');
      ylabel('Amplitude');
   end
   disp(['Variance = ', num2str(var(i)), ' (Mean Filter Results)']);
   disp('MSE values:');
   disp(mse_values);

   figure();
   for j = 1:4
      start = filtersize(j);
      hole = floor(start / 2);
      y_median = zeros(size(y));
      for k = hole + 1 : n - hole
         window = audio(k - hole : k + hole);
         y_median(k) = median(window);
      end
      mse_values(j) = mean((y - y_median).^2);
      subplot(4, 1, j );
```

```
        plot(y_median(1:500));
        title(['Median Filter (Variance = ', num2str(var(i)), ', FilterSize = ', num2str(filtersize(j)),
', MSE = ', num2str(mse_values(j), '%.4f'), ')']);
        xlabel('Time (s)');
        ylabel('Amplitude');
    end
    disp(['Variance = ', num2str(var(i)), ' (Median Filter Results)']);
    disp('MSE values:');
    disp(mse_values);

    figure();
    for j = 1:4
        kernel = wightfilterkernal{j};          % Select kernel for current filter
        kernel_len = length(kernel);            % Length of the kernel
        half_len = floor(kernel_len / 2);
        y_weight = zeros(size(y));
        for k = half_len + 1 : n - half_len
            window = audio(k - half_len : k + half_len);
            weighted_sum = 0;
            for m = 1:kernel_len
                weighted_sum = weighted_sum + window(m) * kernel(m);
            end
            y_weight(k) = weighted_sum / sum(kernel);
        end
        mse_values(j) = mean((y - y_weight).^2);
        subplot(4, 1, j);
        plot(y_weight(1:500));
        title(['Weight Filter (Variance = ', num2str(var(i)), ', FilterSize = ', num2str(filtersize(j)), ',
MSE = ', num2str(mse_values(j), '%.4f'), ')']);
        xlabel('Time (s)');
        ylabel('Amplitude');
    end
    disp(['Variance = ', num2str(var(i)), ' (Weighted Filter Results)']);
    disp('MSE values:');
    disp(mse_values);
end
```