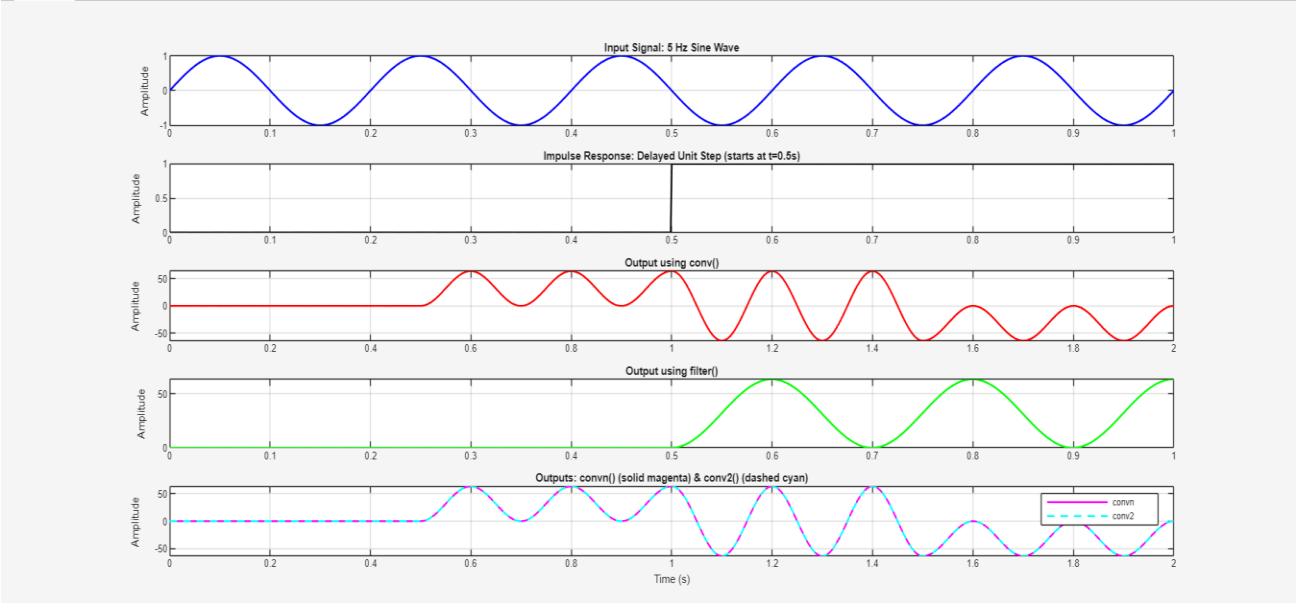
SignalScope

TASKS ON CONVOLUTION

Q1

PLOTS:



1. **Differences in Syntax and Usage:**
   - conv ()

   The syntax is y=conv(x,h,shape). The function returns a vector that is the convolution of vectors x and h based on the standard convolution definition. The size of the output vector depends on the parameter entered in the 'shape' part of the syntax. By default, convolution is done over the entire vectors.

   - filter()

   The syntax is y=filter(b,a,x). The function implements convolution using FIR or IIR filters by employing a rational transfer function based on the coefficients entered. The size of output vector always matches the input vector.

- convn() and conv2()

  The syntax for convn() and conv2() is the same as conv. They are designed to perform convolution on higher dimensional arrays . When used on a 1D array, they provide the same output as conv().

2. Observable Differences in Output and Edge Behavior:

- conv():
  - Full convolution is done in this function. It includes both past and future overlaps in non-causal signals.
  - Output is longer than the input vectors length.
- filter():
  - Filtering is done in causal signal. That is, the output only depends on past and present input.
  - Output starts after impulse response is introduced in the system(delayed step response).
  - Output length is same as input length.
- convn() & conv2():
  - They behave the same as conv() in this case as 1D arrays were used as input. But these functions are optimised for higher order arrays.
  - Overlap and edge effects are similar to conv().

3. Speed and Flexibility:

filter() is the fastest for 1D vectors as it does not include any edge overlap and output length is same as input. Conv() is slower as it performs full mathematical convolution on the input vectors. Convn() and conv2() are the slowest because they are optimised for higher order arrays.

Differences and Similarities

Aspect	conv()	filter()	convn()	conv2()
Purpose	1D Mathematical	FIR/IIR Filtering (System	N-D Convolution	2D Convolution

	Convolution	Realization)		
Causality	Non-causal (uses past and future data)	Causal (real-time, uses only past and present data)	Non-causal for N-D	Non-causal for 2D
Dimensionality	1D only	1D only	Multi-Dimensional (1D, 2D, 3D, etc.)	2D Data (Images, Matrices)
Real-Time Suitability	Not suitable	Highly suitable for real-time DSP	Not suitable for real-time	Not suitable for real-time
Similarity	All perform convolution operations in some form	Equivalent to convolution in FIR case	Like conv but N-D	Like conv but 2D

When should each function be used

- conv()
  Useful when performing mathematical convolutions to analyse signal interaction or system response in theory.

- filter()
  Use when implementing a real-world FIR/IIR filter in digital signal processing systems.

- convn()
  Use for multi-dimensional data such as volumetric signals, 3D images, or multi-dimensional datasets. Not efficient for simple 1D signals.

- conv2()
  Use for 2D data such as images and matrices, where 2D convolution is required. Not recommended for pure 1D signal operations.

Observations from Plots

- Input Signal and Impulse Response:
  The input signal is a 5 Hz sine wave sampled at 1000 Hz. The impulse response is a unit step function that starts at 0.5 seconds, introducing a delay in the system response.

- conv() Output:
  The convolution result shows the complete mathematical convolution of the input with the impulse response. The output starts rising after 0.5 seconds, indicating the delayed unit step.

- filter() Output:
  The output of filter() reflects the causal filtering behaviour, with the result starting after the 0.5-second delay introduced by the impulse response. The output length matches the input length.

- convn() and conv2() Outputs:
  These functions behave similarly to conv() when applied to 1D signals. Their outputs also start after the 0.5-second delay.

Q2: AUDIO FILTERING IN MATLAB

Objective:

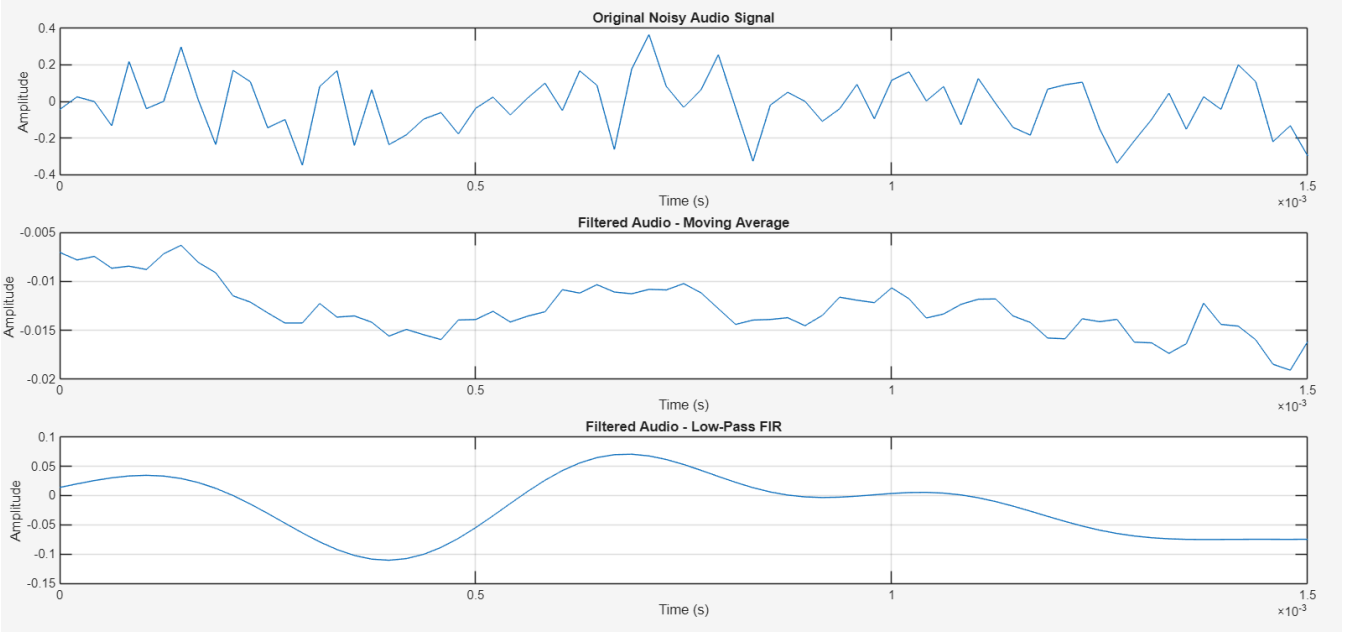
To apply 1D convolution-based filtering techniques on a real-world noisy audio signal in MATLAB using two different impulse response filters and analyse their effects on noise reduction.

- **Audio File:**
  A real-world noisy audio file was used as input in the code named "NoisyAudio.wav". The audio file has been uploaded in the repository so that the viewers can run the program and check themselves. Mono-channel audio was used to simplify the task.

- **Filters used:**
  1) Moving Average Filter:
  ->Order(N): 125
  ->These filters smooth rapid changes (high-frequency noise), but may blur sharp signal transitions.
  2) Low-Pass FIR Filter:
  ->Order: 100(101 coefficients)
  ->Cutoff frequency :0.1
  -> These filters attenuate frequencies above cutoff, preserving low-frequency signal components.

- **1D convolution was applied(conv()). 'same' option was used to maintain original signal length for simpler analysis.**

- **Plotting of the results**



Plot 1: Original Noisy Audio Signal

- ❖ High-frequency, random fluctuations (noise)

Plot 2: After Moving Average Filter

- ❖ Smoother waveform with reduced amplitude.
- ❖ Noise is significantly reduced but some signal flattening is observed.

Plot 3: After Low-Pass FIR Filter

- ❖ Smooth plot having sinusoidal-like shape.
- ❖ High-frequency components are removed.
- ❖ Less distortion plot and it showcases periodic properties.

- **<u>Audio Playback:</u>**

- ❖ MATLAB's sound() function was used to listen to:

1. Original noisy audio,
2. Moving Average filtered audio,
3. Low-Pass FIR filtered audio,

- ❖ 2-second pause inserted between each for clear comparison.

Observations & Analysis:

Filter	Effectiveness	Comment
Moving Average Filter	Smoothens signal, suppresses high-frequency noise, but causes signal flattening.	Suitable for general smoothing; but it is not frequency-selective.
Low-Pass FIR Filter	Removes high-frequency noise while preserving low-frequency features; better overall output.	Retains original signal's important features; preferrable choice for mono channel audio.

Conclusion:

Both filters effectively reduce noise but **Low-Pass FIR filtering** provides superior frequency-based noise removal without distorting the original signal shape.

Q3: Image Filtering In MATLAB

Objective:

To reduce noise in a digital image using common 2D convolution kernels and evaluate the performance using the metrics MSE (Mean Squared Error) and PSNR (Peak Signal-to-Noise Ratio).

1. **Input Image:**
    - A clean image (a.png) was loaded and converted to double precision to allow accurate mathematical processing.

2. **Noise Addition:**
    - Gaussian noise was synthetically added to the image using MATLAB's imnoise() function.
    - Noise characteristics:
        - Mean = 0
        - Variance = 0.01
        -

3. **Gaussian Filter Design:**
    - A 2D Gaussian filter kernel of size 5×5 was manually created using the Gaussian function:

$$G(x,y) = \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

    - Parameters used:
        - Sigma (σ) = 1

        - Kernel Size = 5×5
    - The filter kernel was then normalized so that the sum of all its elements is 1.

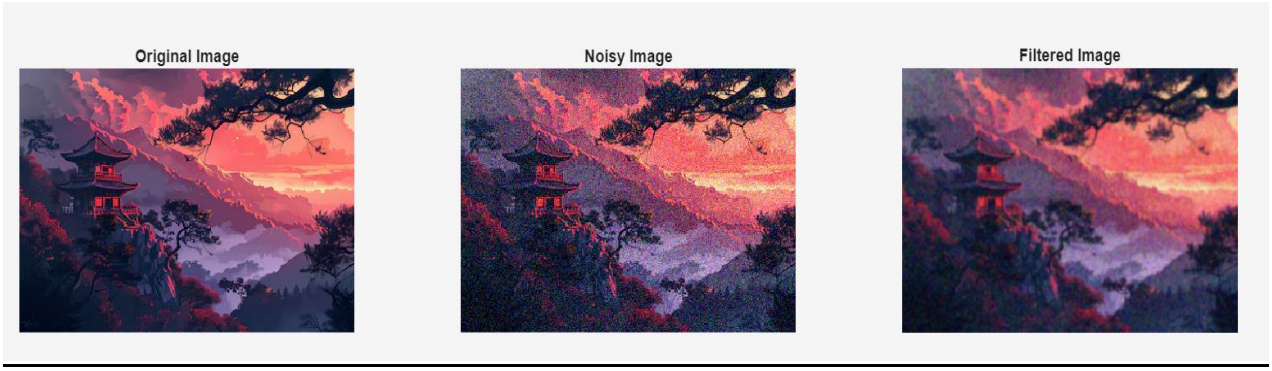
4. **Filtering Operation:**

    - The designed Gaussian filter was applied to the noisy image using MATLAB's imfilter() function with 'symmetric' padding.
    - 'symmetric' padding ensures that the boundary pixels are handled by mirroring the edges, preventing artifacts and preserving image dimensions.

5. **Performance Evaluation:**

    - After filtering, the denoised image was compared to the original (clean) image using:
        - **MSE (Mean Squared Error):** Measures the average squared difference between original and filtered image pixels.
        - **PSNR (Peak Signal-to-Noise Ratio):** Indicates the quality of the filtered image in decibels (dB); higher PSNR means better quality and less distortion.

Observations & Results



Stage	Observation
Original Image	Sharp, noise-free reference image.
Noisy Image	Visible grainy noise added uniformly (Gaussian noise).
Filtered Image	Noticeable reduction in noise; edges remain mostly preserved. Image is slightly blur.

The Gaussian filter effectively reduced noise by smoothing high-frequency noise components while maintaining image structure and details.

Performance Metrics (from MATLAB output):

- **MSE:** 0.0030
- **PSNR:** 25.1380 dB

Lower MSE and higher PSNR indicate successful noise suppression and minimal information loss.

Conclusion:

- The manually created Gaussian filter kernel proved effective for reducing Gaussian noise.
- The filtering process preserved important image features while suppressing unwanted noise.
- Evaluation metrics confirmed the filter's ability to balance smoothing and detail preservation.