

---

## Table of Contents

.....	1
Tools upgrade .....	1
Section 3 .....	5
WGN with LPF .....	5
WGN with HPF .....	7
Known 5 frequencies .....	8
Random 5 frequencies .....	9
Signal games .....	10
Signal Example 1 - Filtering in the Frequency Domain .....	10
Signal Example 2 - Filtering in time domain .....	12
Signal Example 3 - Filtering in frequency and time domain .....	14

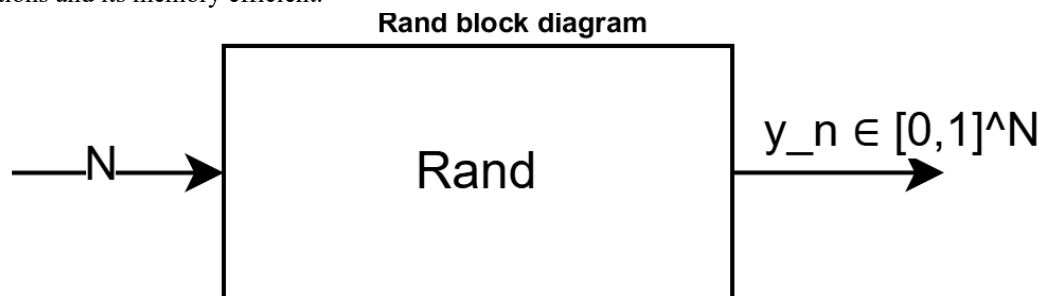
## Tools upgrade

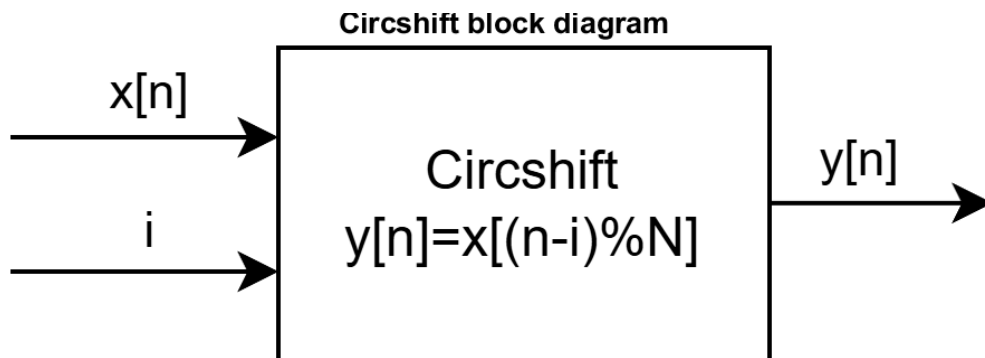
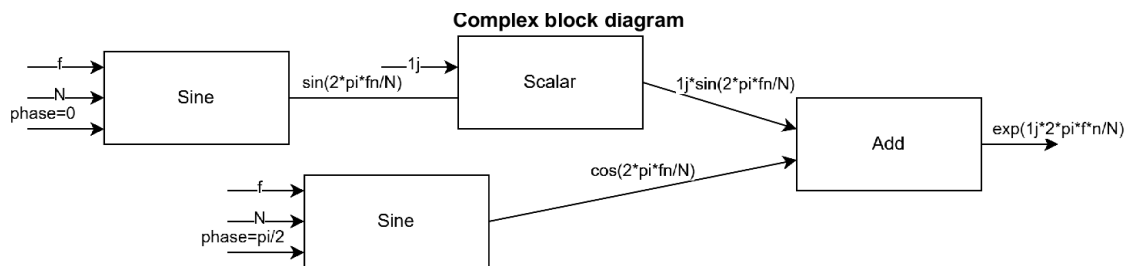
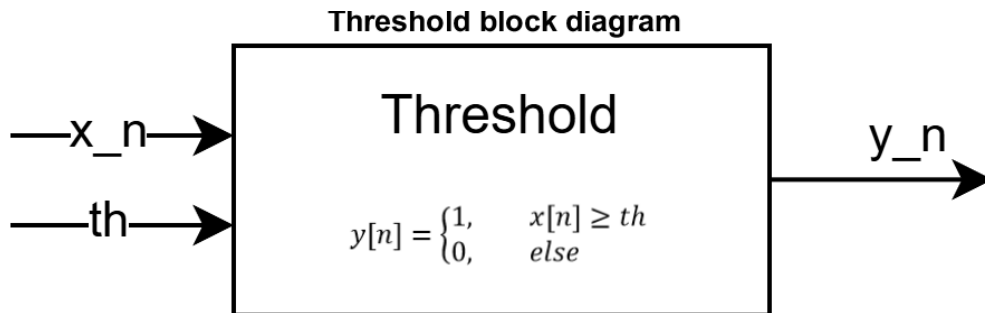
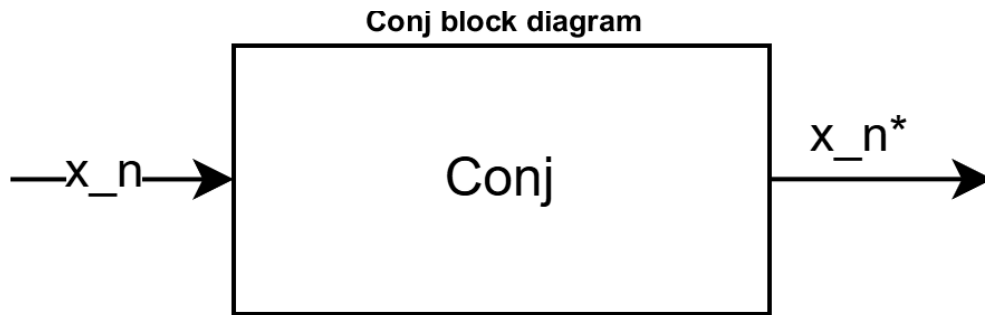
We have created the Rand, Conj, Threshold, Complex, STFT and ISTFT blocks. We implemented STFT\ISTFT as polyphase filters (decimation\interpolation and filtering with "DFT" filter, which is defined using Complex block). Since the polyphase filters in this case is scalars so we apply them using Scalar block.

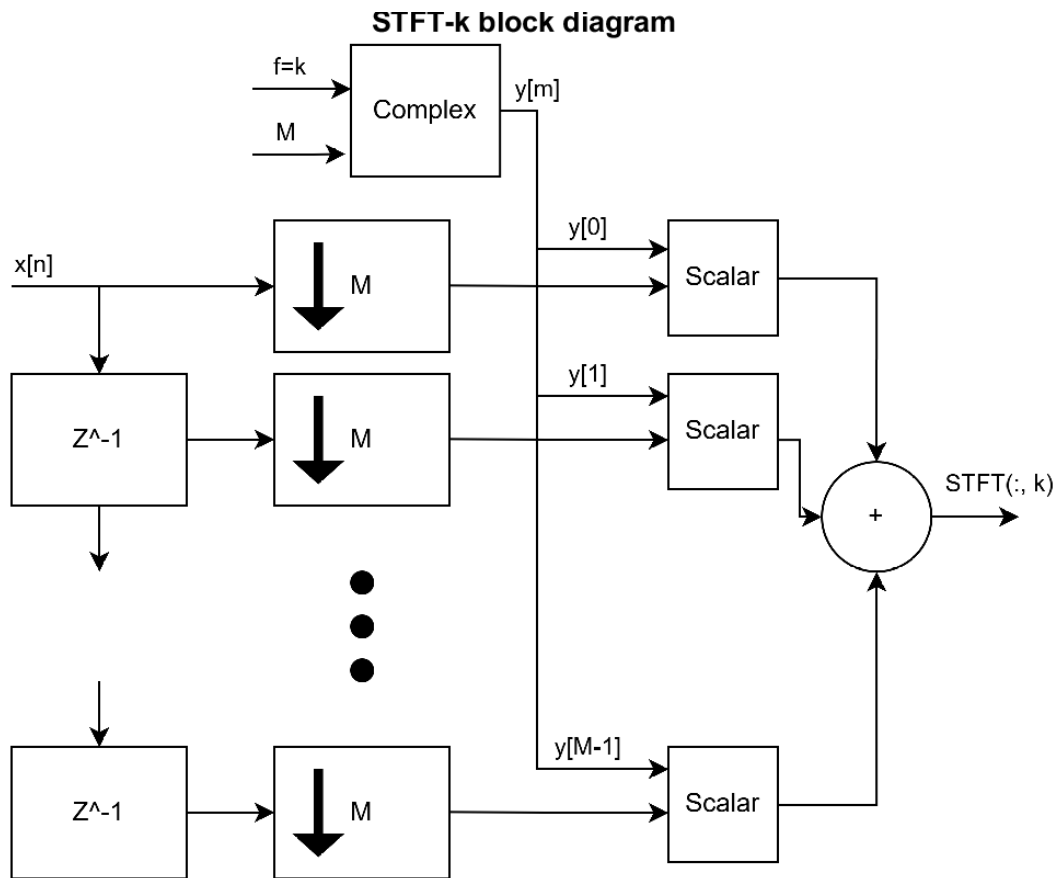
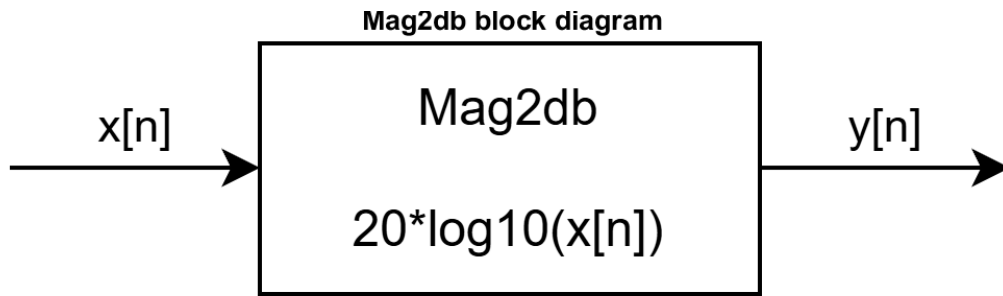
The STFT\ISTFT was divided into multiple steps (smaller blocks):

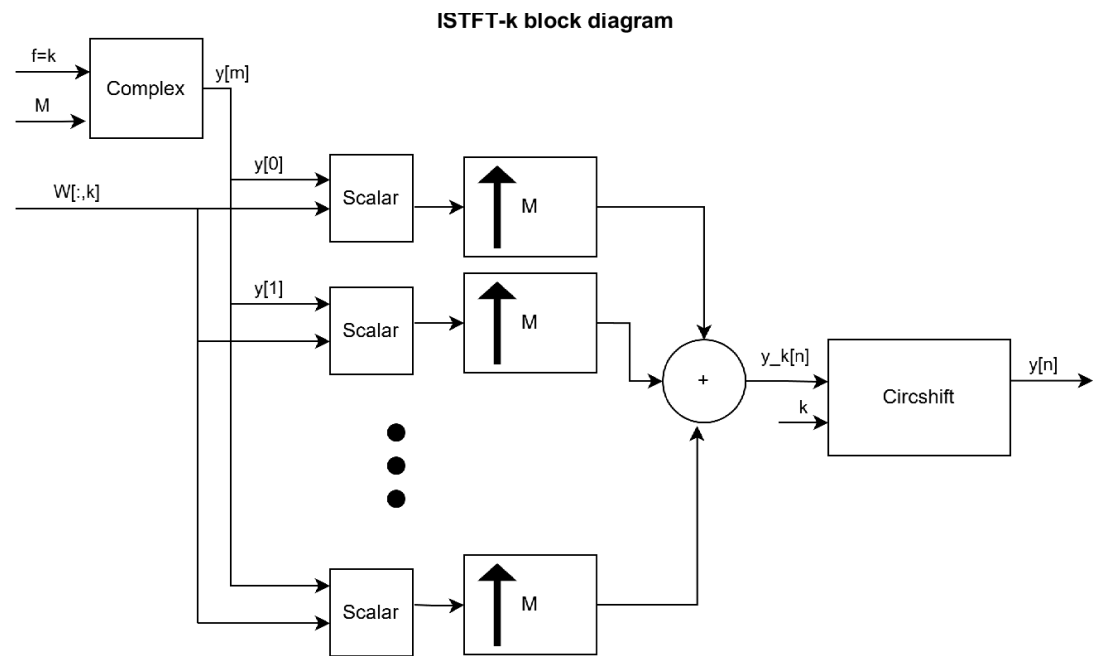
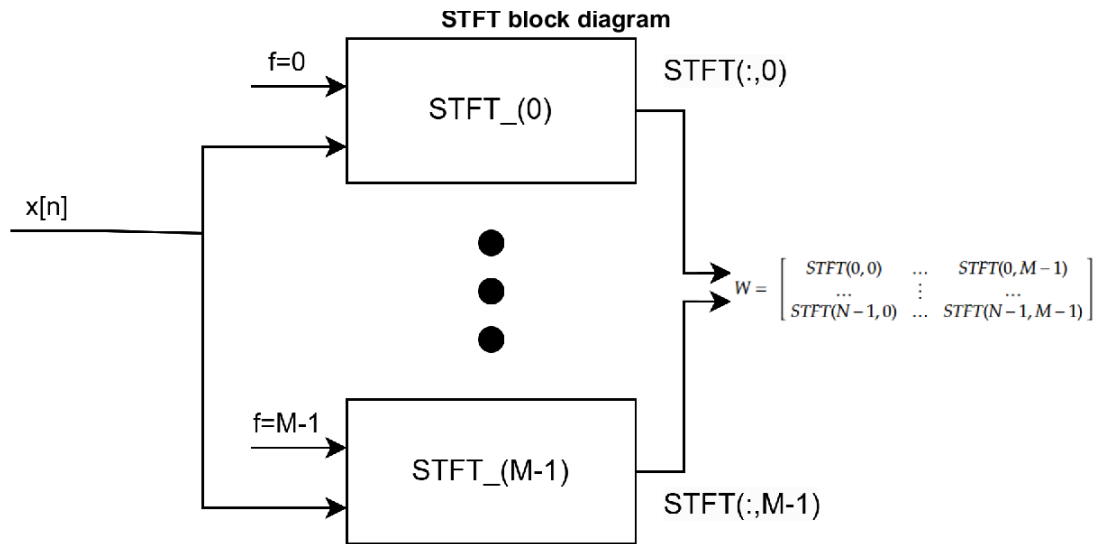
- Getting  $i$  element from DFT signal for  $k$  frequency - polyphase filter for delay  $i$  and frequency  $k$
- STFT<sub>k</sub> - computes STFT for frequency  $k$

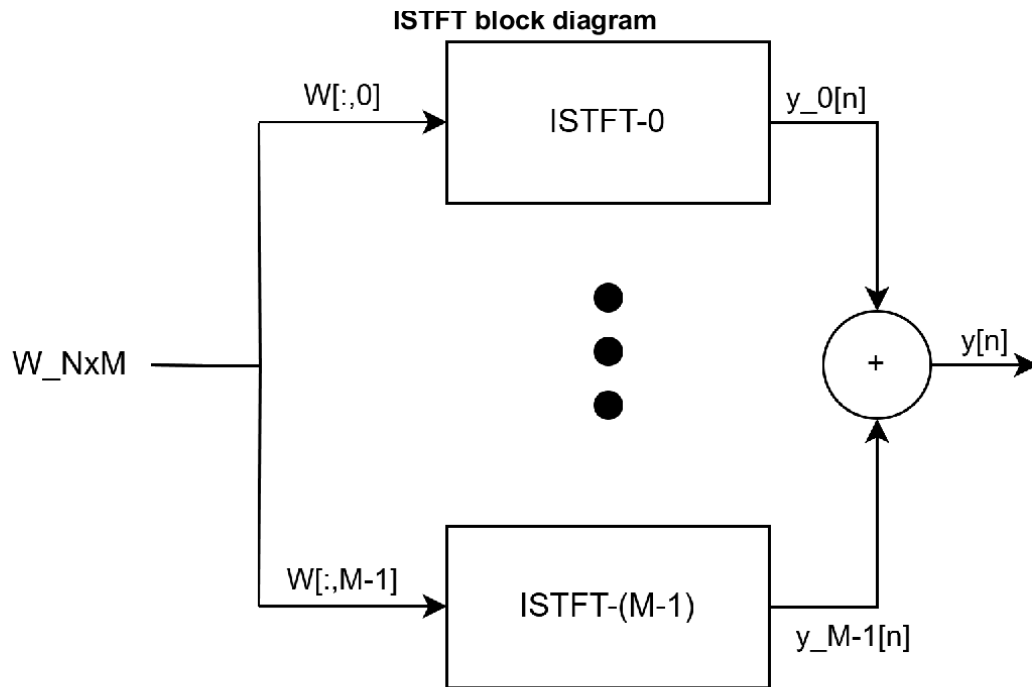
so then we call these blocks to fill the STFT\ISTFT matrices. The ISTFT was implemented using polyphase filters and Interpolation block from HW1. In addition we have defined Crcshift which was applied after the interpolation in order to align samples in the time domain for each frame. Using this implementation there is no redundant multiplications and its memory efficient.







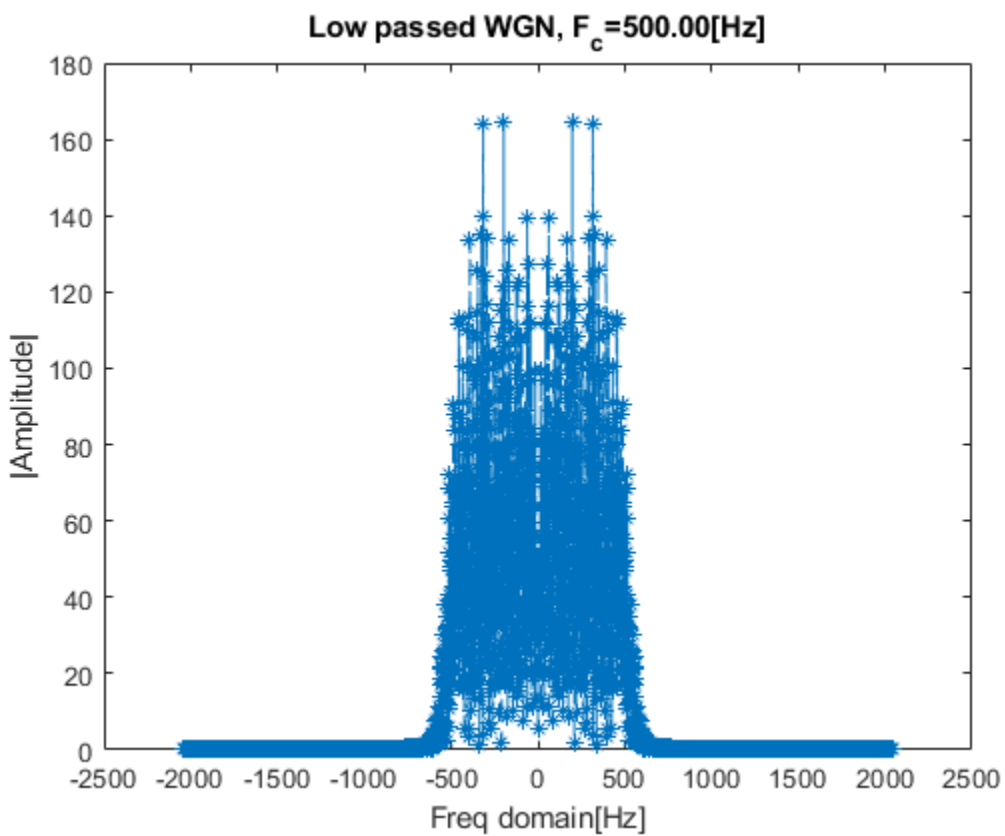
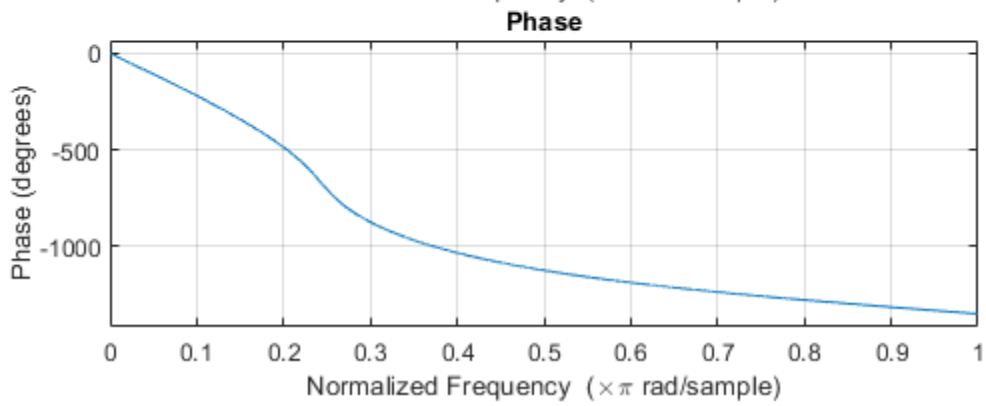
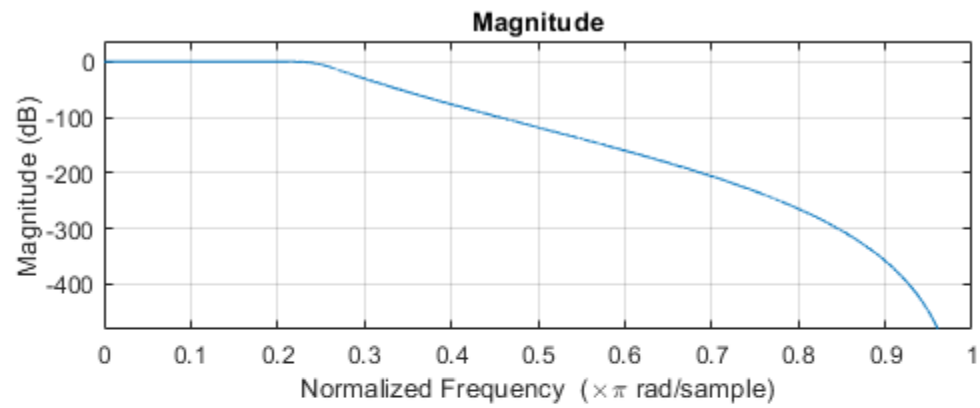




## Section 3

### WGN with LPF

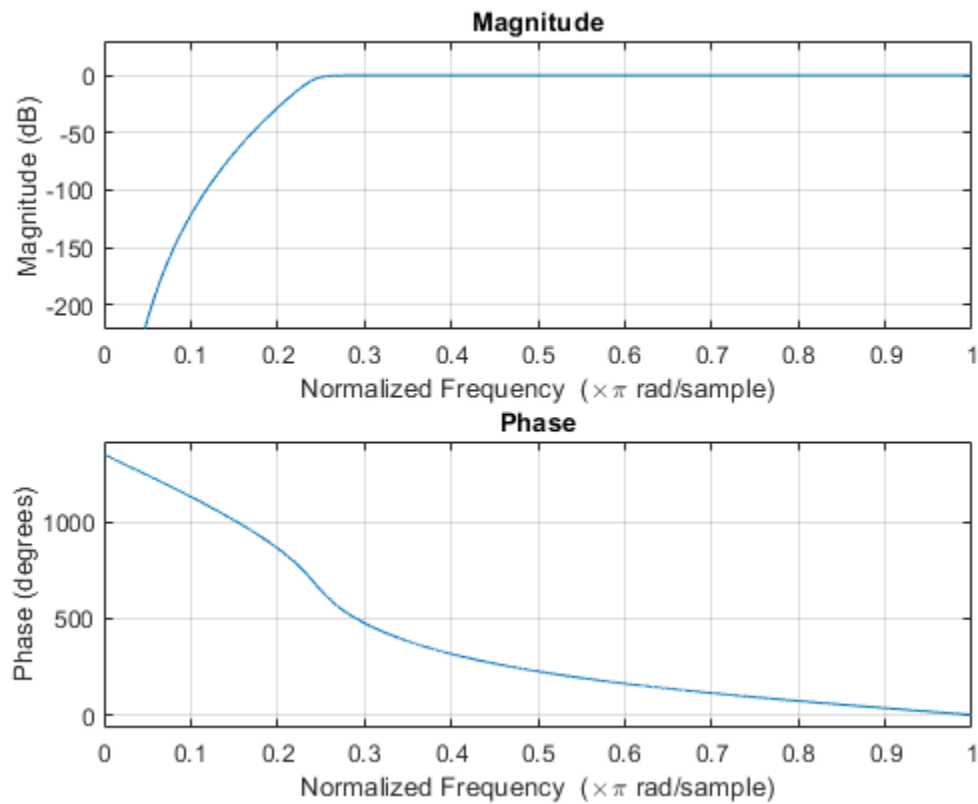
First, lets design IIR LPF, with  $F_c = 500[Hz]$  and apply it on WGN using Filter function from HW1.

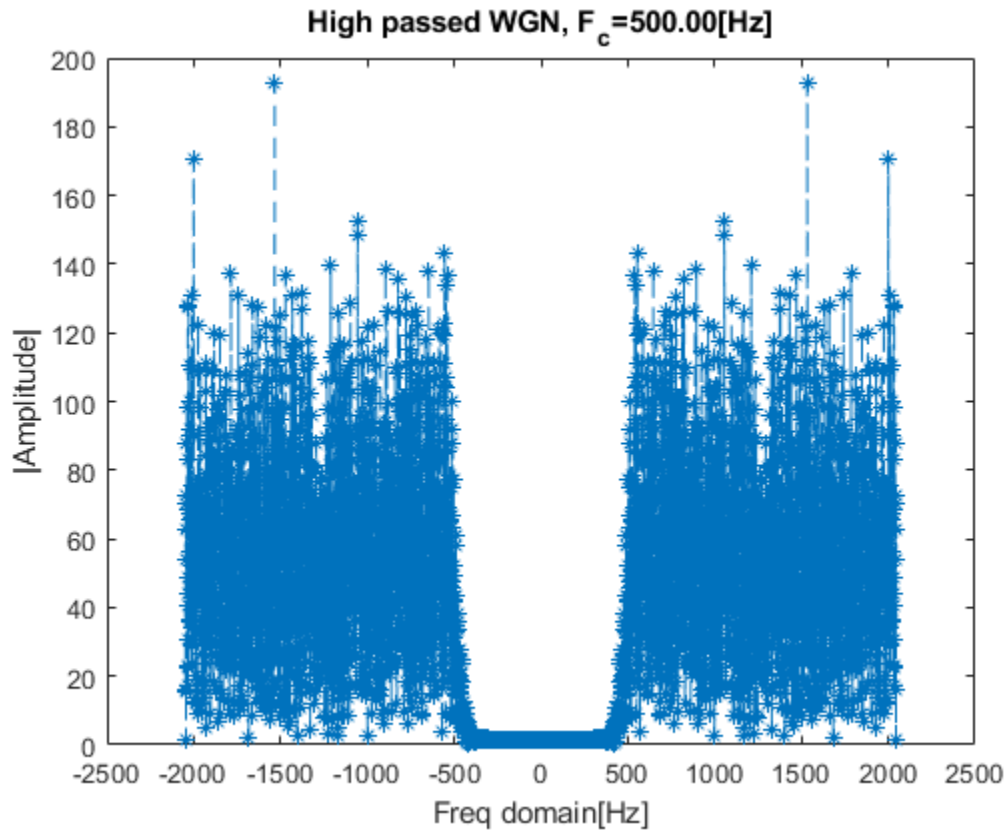


---

## WGN with HPF

Now, lets design IIR HPF, with the same  $F_c = 500[Hz]$  and procced the same flow as with LPF.

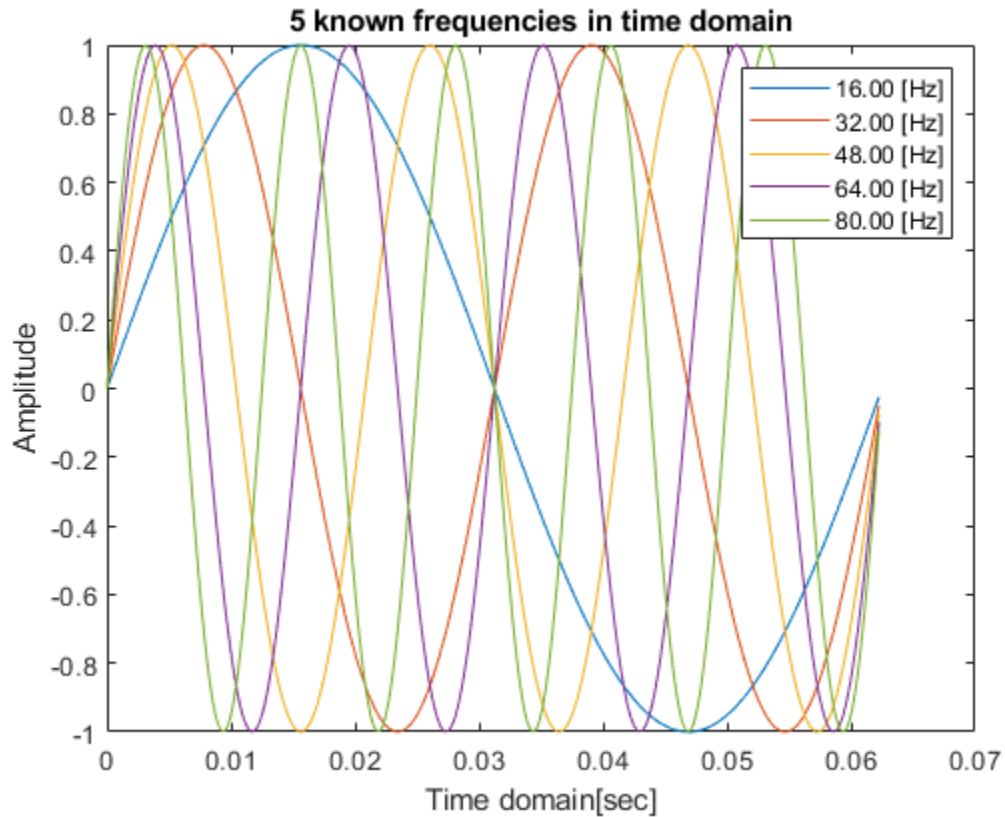




## Known 5 frequencies

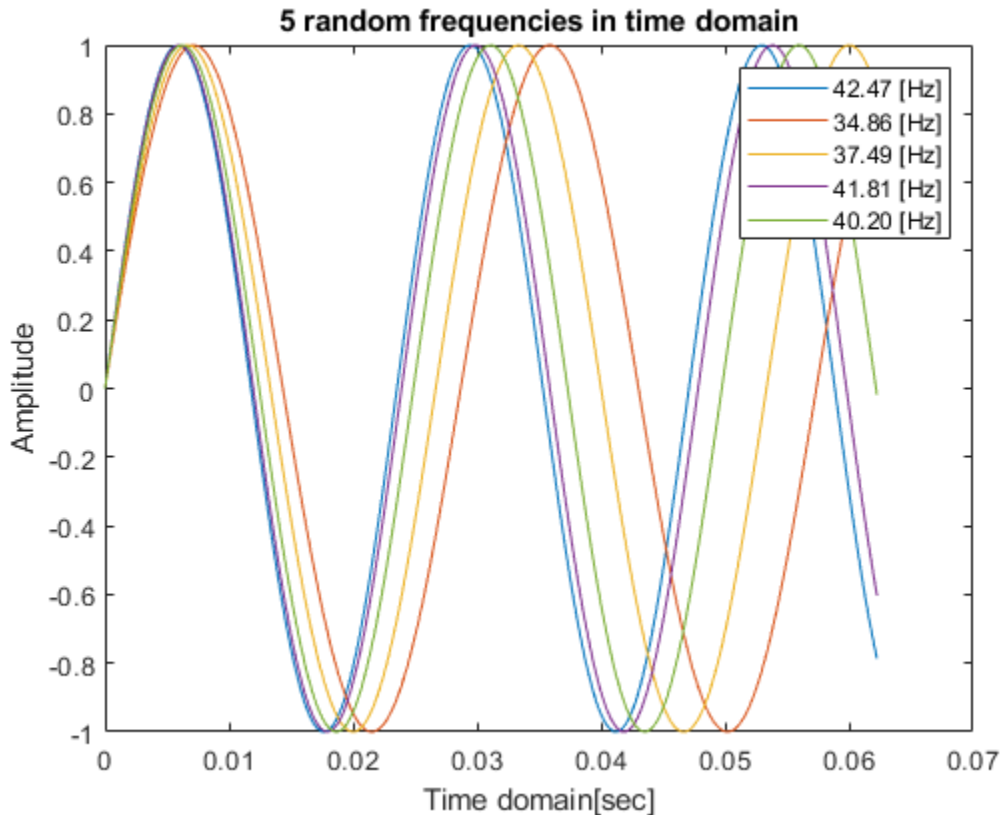
We define 5 sinusoidal signal using Sine func from HW1 at [16, 32, 48, 64, 80] [Hz].





## Random 5 frequencies

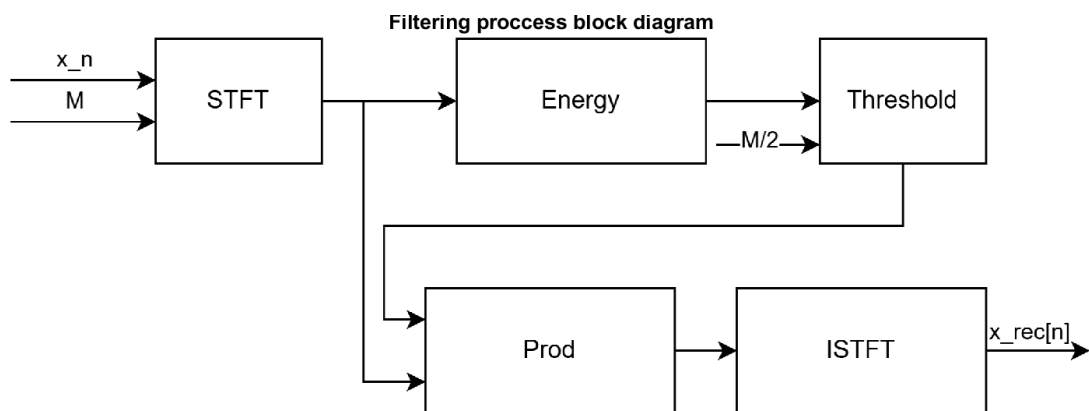
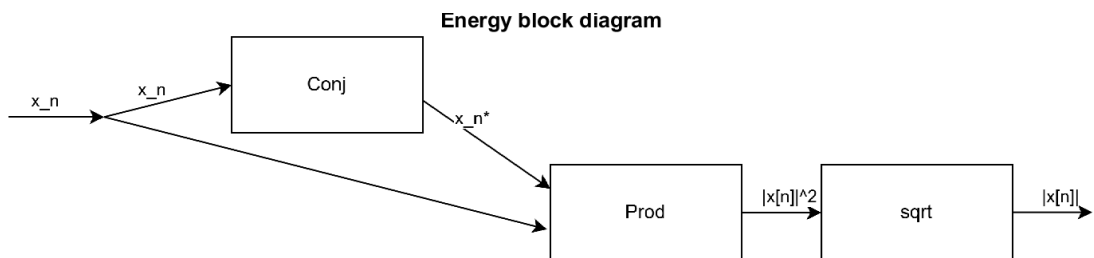
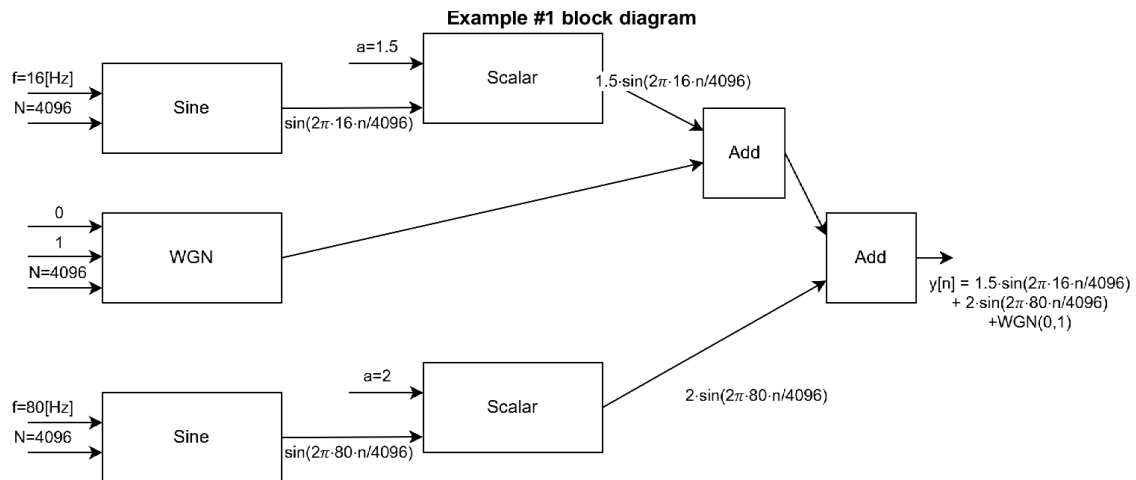
We sample 5 random frequencies using Rand block, scale them to [0, 30] Offset to [30, 60] using Add block from HW1 And create sinusoidal signal using Sine block.



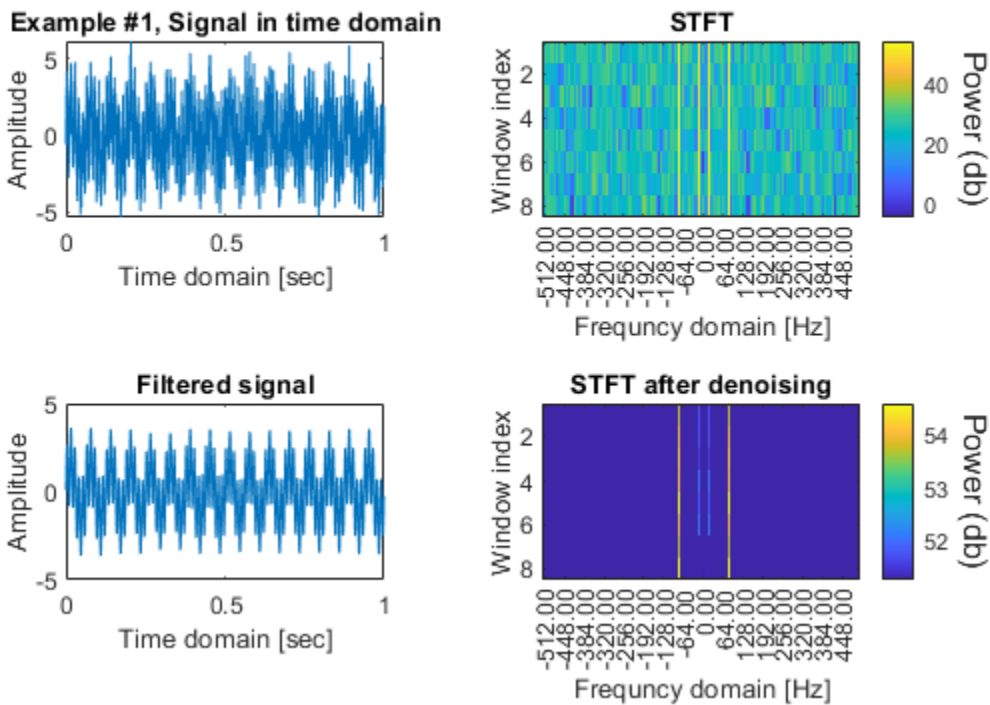
## Signal games

### Signal Example 1 - Filtering in the Frequency Domain

Filtering white Gaussian noise (WGN) using a threshold block in the frequency domain. Let's consider signals with known frequencies chosen such that they align directly with discrete frequency bins. This ensures no energy spreading in the frequency domain, allowing us to easily identify the expected energy at each frequency bin. As discussed in class, the energy in the frequency domain for a pure sine or cosine wave is equal to  $M/2$   $M/2$  at each delta. To filter out WGN, we set the threshold  $th = M/2$ . As demonstrated, the noise is effectively filtered out. The STFT reveals four deltas at  $\pm 16$  Hz and  $\pm 80$  Hz as expected, with no remaining noise, since it has been completely removed. This approach is effective because the frequencies align perfectly with the DFT bins, making it straightforward to determine the filtering threshold. However, if the frequencies did not align directly with the bins, identifying an appropriate threshold would be more challenging.

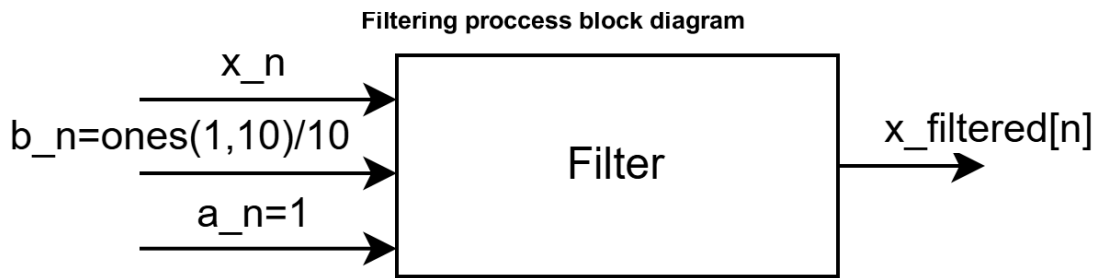
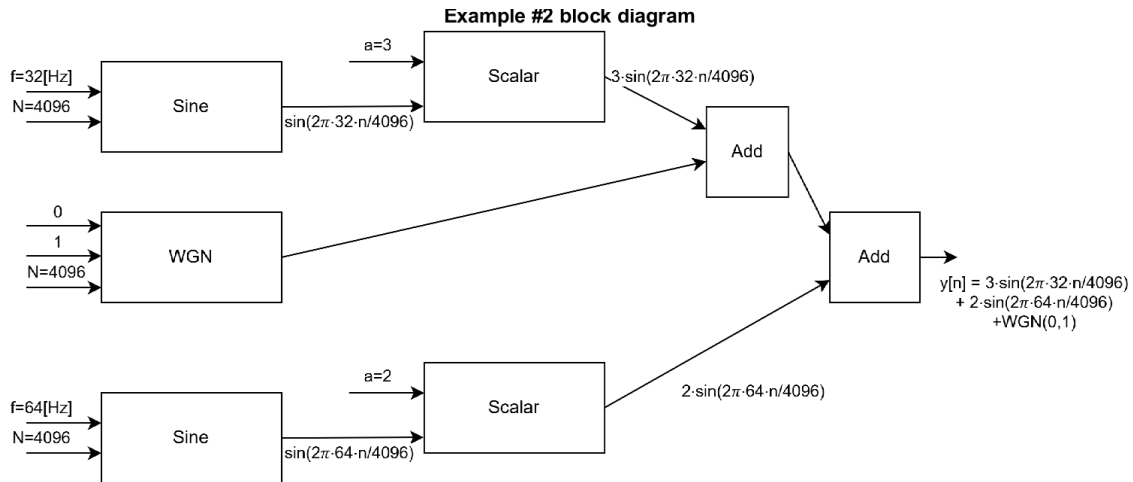


## Example #1 Filtering in frequency domain

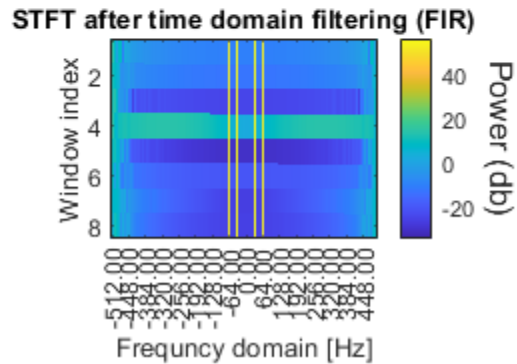
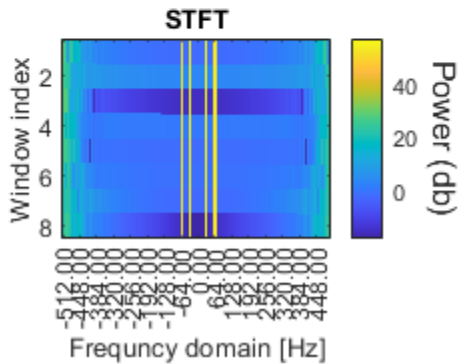
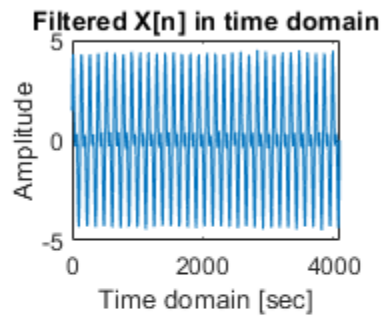
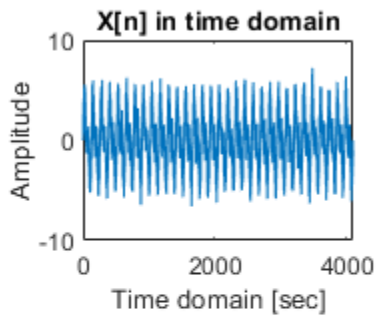


## Signal Example 2 - Filtering in time domain

The simplest low-pass filter is a moving average filter. Here, we define the LPF coefficients as  $\text{ones}(1,10)/10$  and apply it as an FIR filter using the `Filter` function from HW1. The intuition behind this approach is that averaging consecutive samples results in values that are closer to the mean, reducing the impact of extreme values. This smooths out rapid changes in the signal. As observed, the reconstructed signal becomes smoother, with significantly less distortion, although some noise remains. In the STFT, we notice reduced energy at higher frequencies, indicating that parts of the WGN have been effectively filtered out. As expected, both the original and filtered signals display four distinct delta components at  $\pm 32$  Hz and  $\pm 64$  Hz. This approach works well because the signal-to-noise ratio is relatively high, meaning there is meaningful information in closely spaced samples in the time domain.

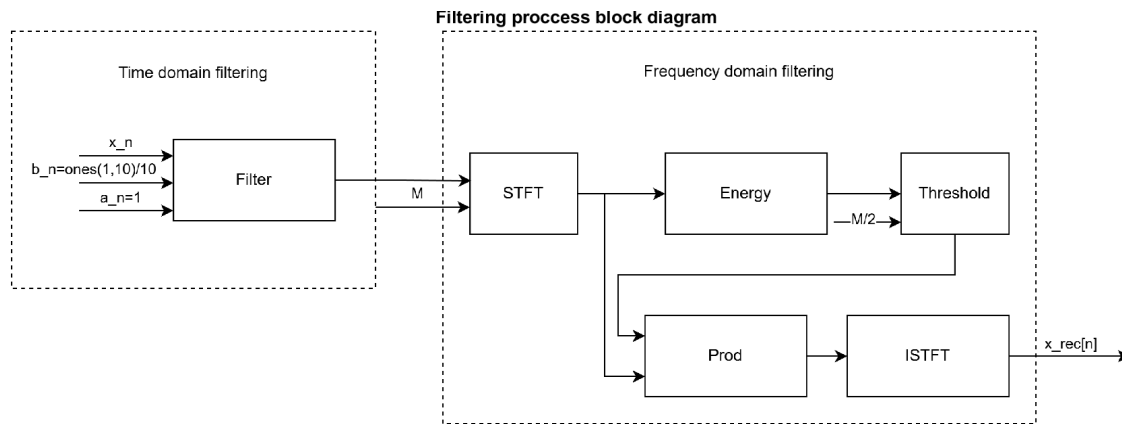
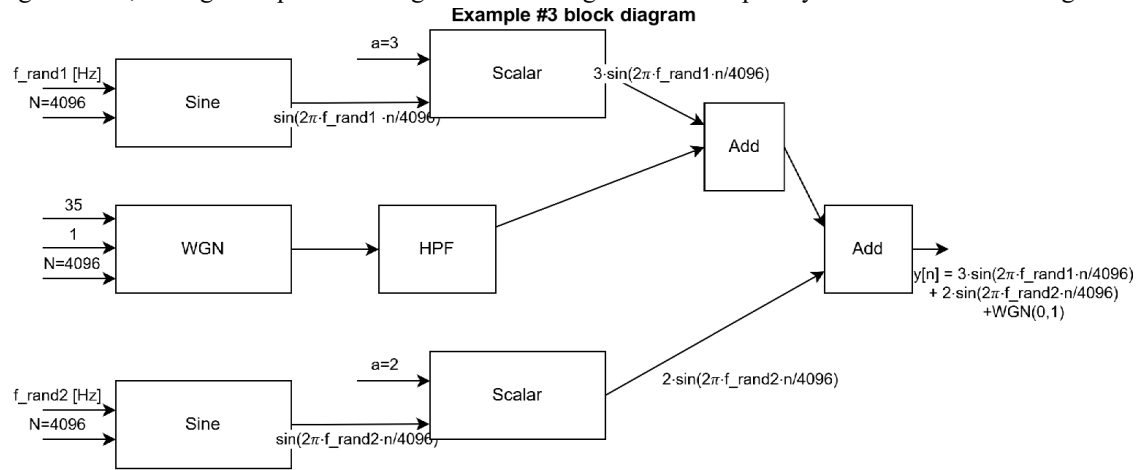


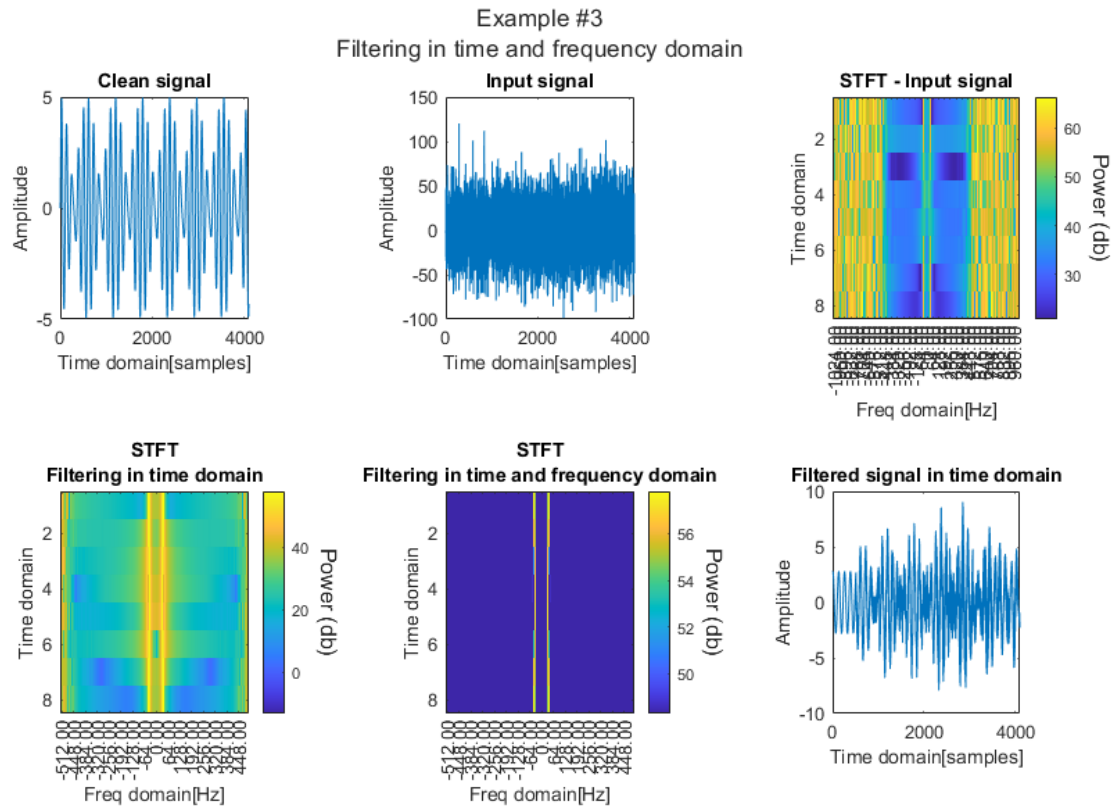
## Example #2 Filtering in time domain



# Signal Example 3 - Filtering in frequency and time domain

In this example, we will filter a signal composed of an amplified high-pass filtered white Gaussian noise combined with two random sinusoidal signals, amplified by factors of 3 and 2. The noise power is comparable to the power of the sinusoidal signals, making it challenging to filter out the noise as in the previous example. To extract the desired signal, we first apply a LPF in the time domain. This reduces the high-frequency noise while minimally affecting the data signal. Next, the signal is passed through a thresholding block to completely eliminate the remaining noise.





*Published with MATLAB® R2024a*