Table of Contents

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). The it is applied 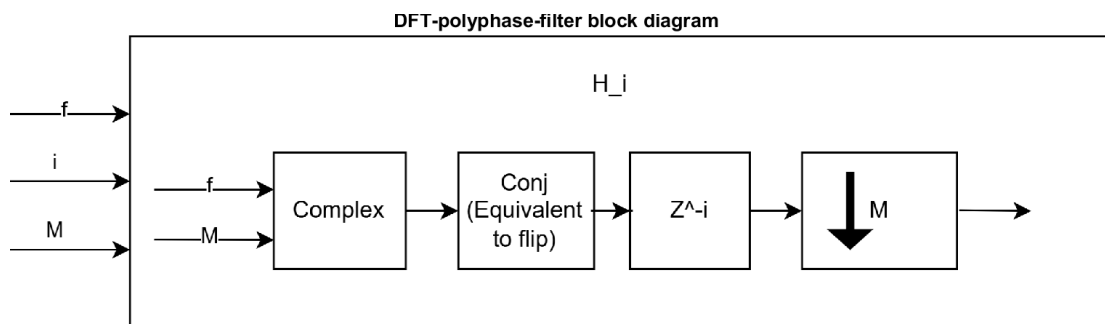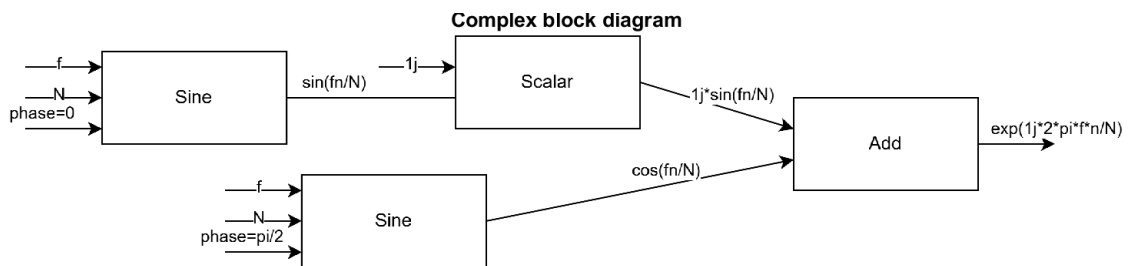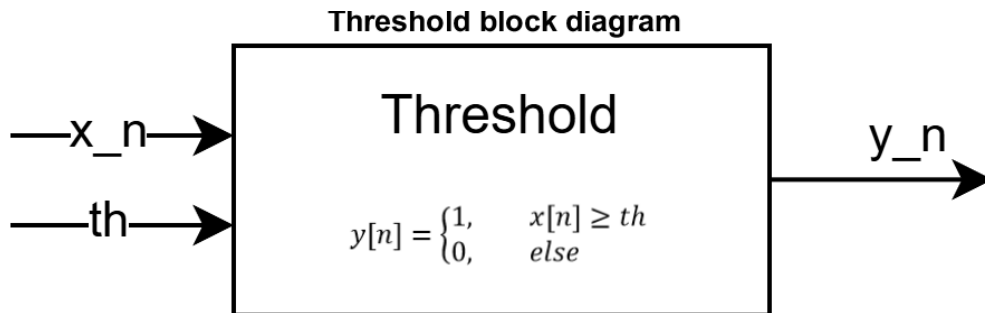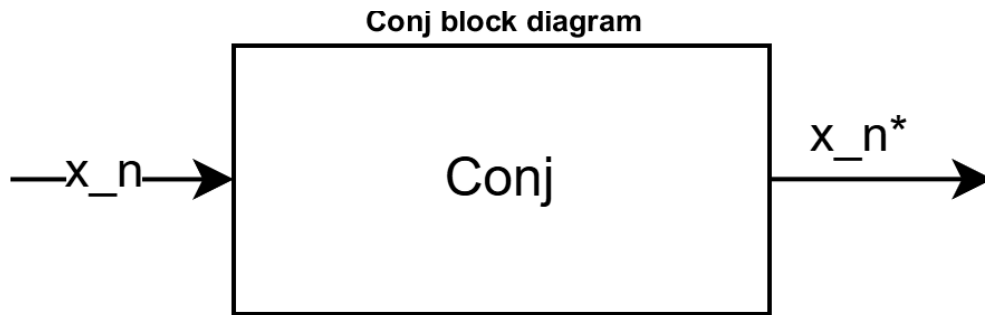using Filter block (the filter block also flips the filter coefficients, so before that we flip the filter using conj block).

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

• H_i_k - defines DFT polyphase filter for delay i and frequency k

• $STFT\_k$ - 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 Circshift which was applied after the interpolation in order align samples in time domain to between the frames. Using this implementation there is no redundant multiplications and its memory efficient.

Rand block diagram

$$N \rightarrow \boxed{\text{Rand}} \rightarrow y\_n \in [0,1]^{\wedge}N$$

Conj block diagram

x_n → **Conj** → x_n*

Threshold block diagram

x_n →
th →
Threshold

$$y[n] = \begin{cases} 1, & x[n] \geq th \\ 0, & else \end{cases}$$

→ y_n

Complex block diagram

f →
N →
phase=0 →
Sine → sin(fn/N) → 1j → **Scalar** → 1j*sin(fn/N) →

f →
N →
phase=pi/2 →
Sine → cos(fn/N) →

Add → exp(1j*2*pi*f*n/N)

DFT-polyphase-filter block diagram

H_i

f →
i →
M →

f →
M →
Complex → **Conj (Equivalent to flip)** → **Z^-i** → **↓ M** →

Circshift block diagram

x[n] →

i →

Circshift
y[n]=x[(n-i)%N]

→ y[n]

STFT-k block diagram

x[n] →

Z^-1

Z^-1

↓ M   f=k →   H_0(z)

↓ M   f=k →   H_1(z)

↓ M   f=k →   H_M-1(z)

+   → STFT(:, k)

STFT block diagram

f=0

STFT_(0)

STFT(:,0)

x[n]

$W = \begin{bmatrix} STFT(0,0) & \dots & STFT(0,M-1) \\ \dots & \vdots & \dots \\ STFT(N-1,0) & \dots & STFT(N-1,M-1) \end{bmatrix}$

f=M-1

STFT_(M-1)

STFT(:,M-1)

ISTFT-k block diagram

f=k

W[:,k]

H_0*(z)

↑ M

f=k

H_1*(z)

↑ M

+

y_k[n]

Circshift
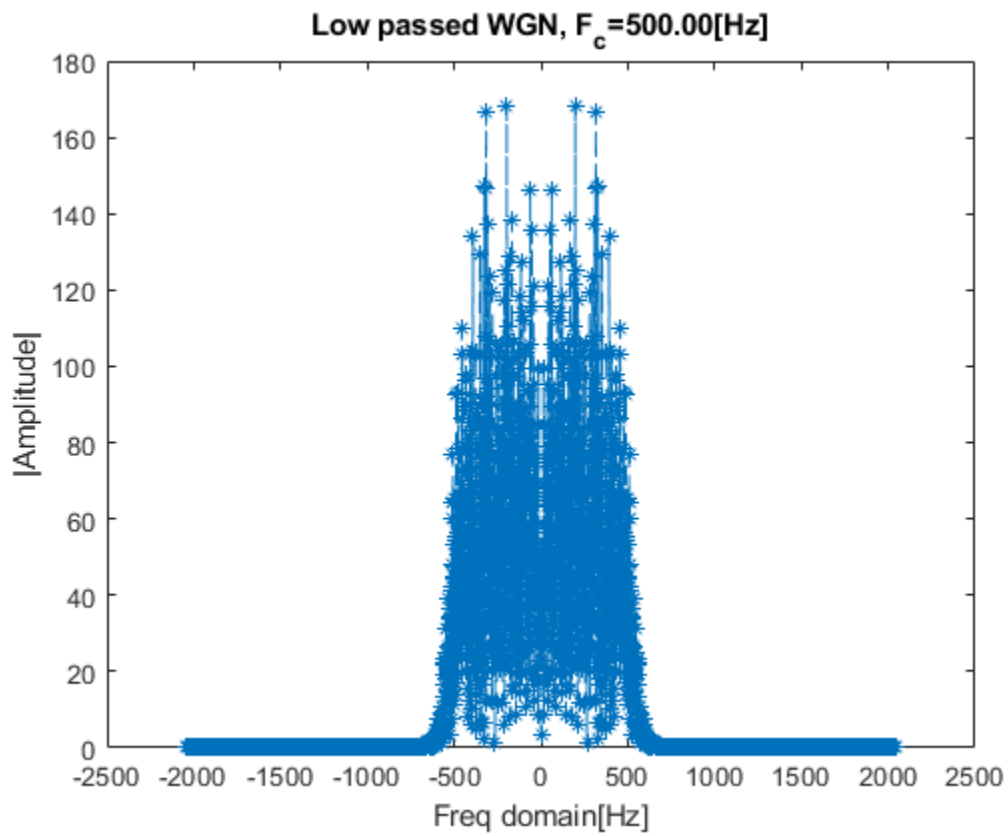
y[n]

k

f=k

H_M-1*(z)

↑ M

4

ISTFT block diagram
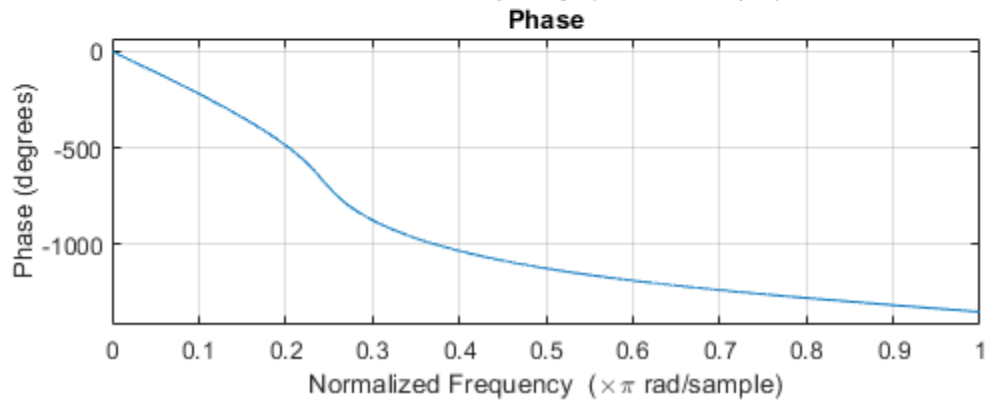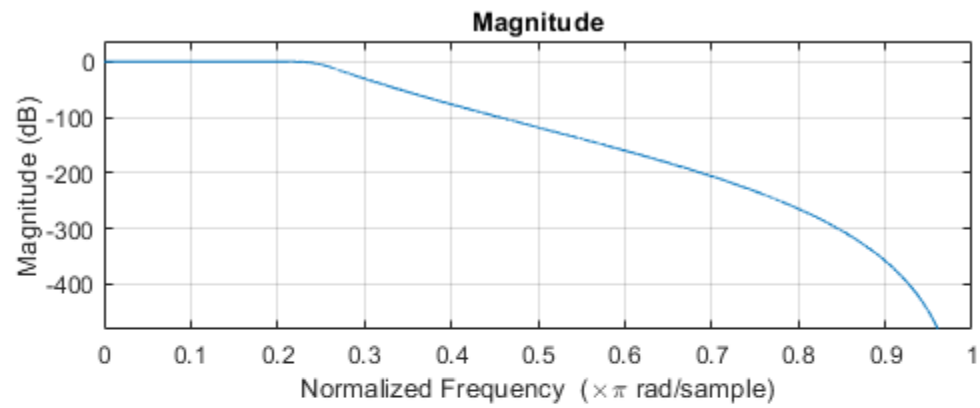

Section 3

WGN with LPF

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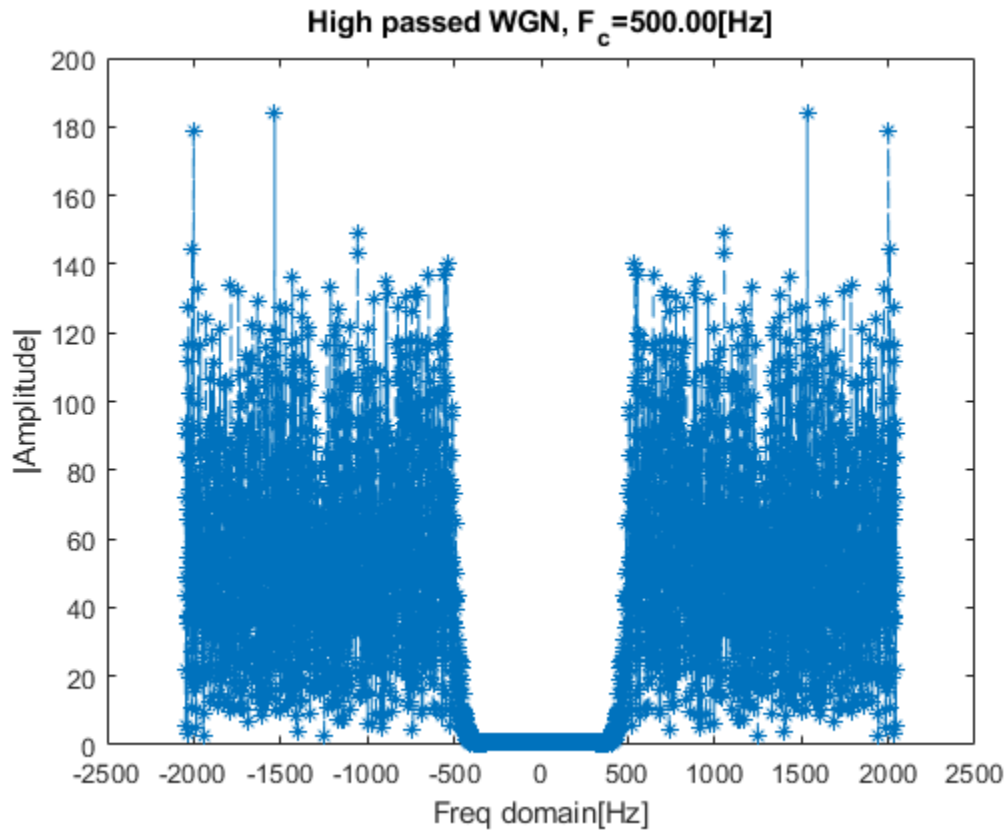
Magnitude

Phase

Low passed WGN, F$_c$=500.00[Hz]

WGN with HPF

Now, lets design IIR HPF, with the same $Fc = 500[Hz]$ and procced the same flow as with LPF.
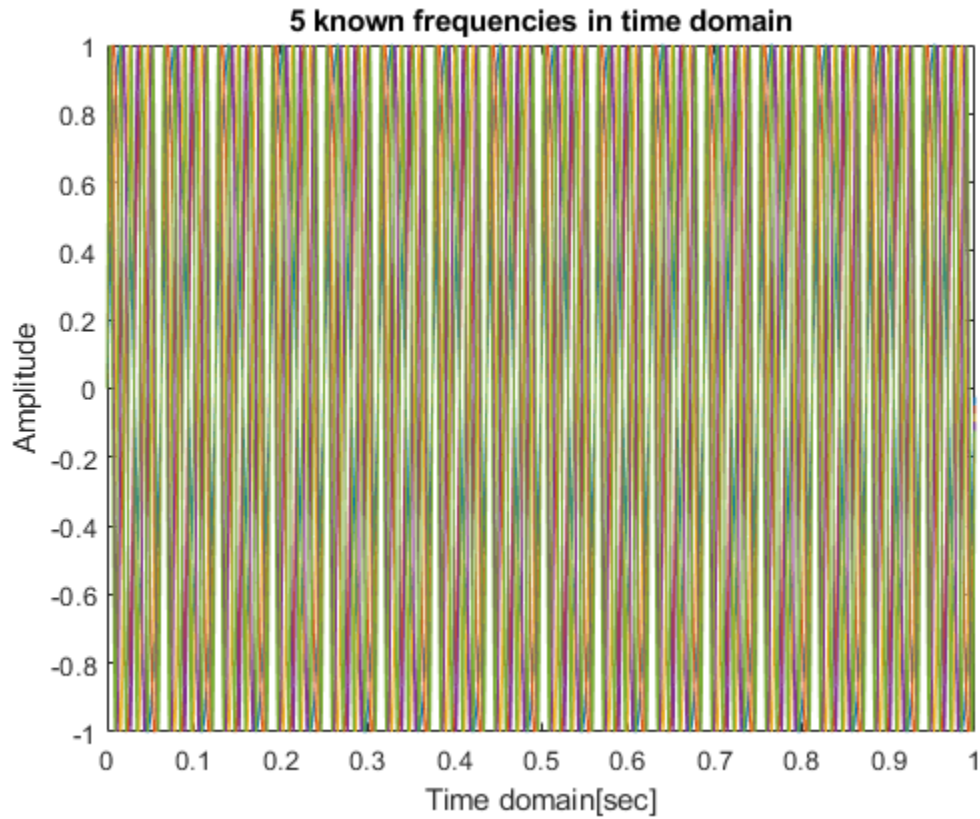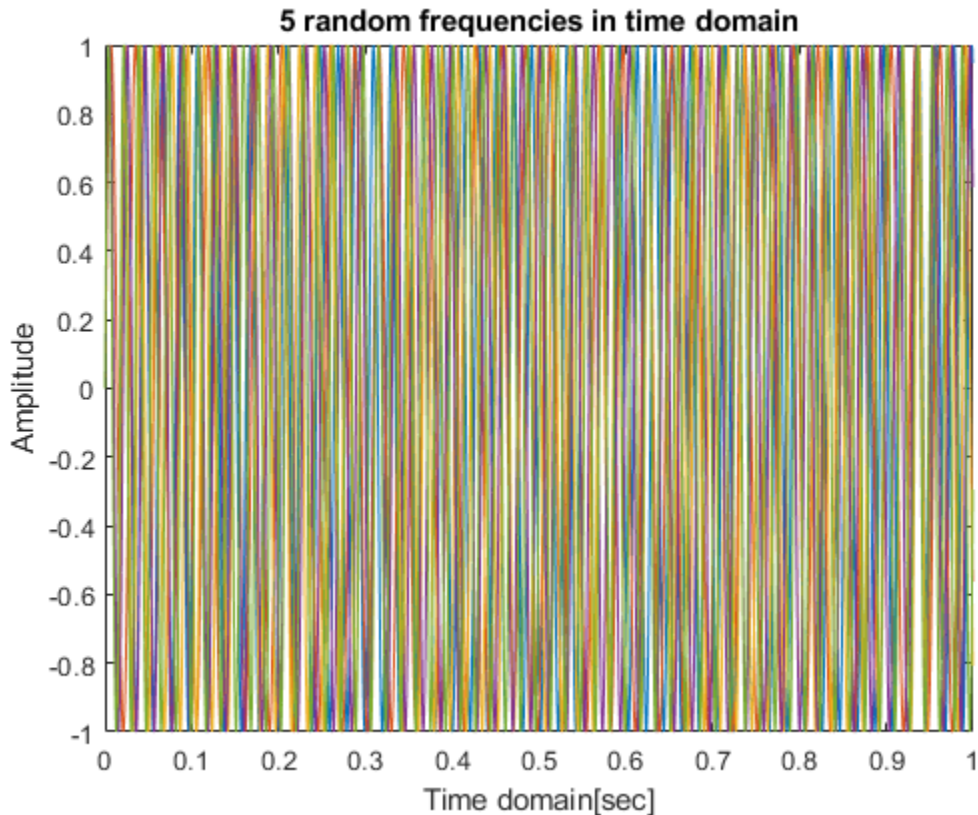
High passed WGN, F_c=500.00[Hz]
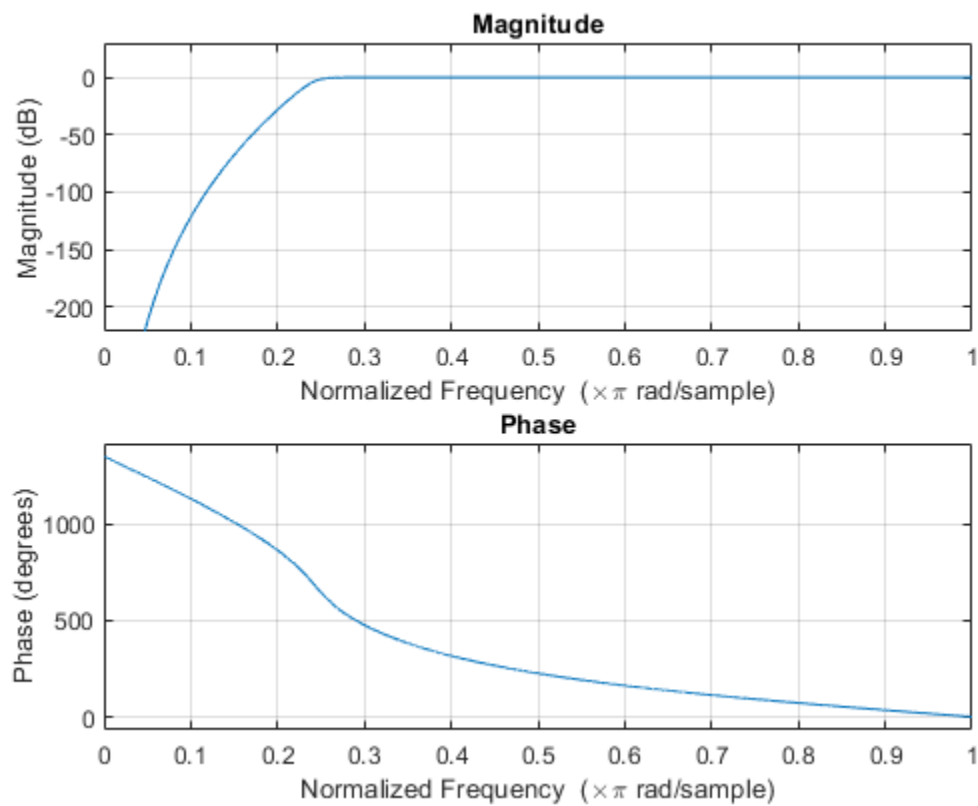
Known 5 frequencies

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

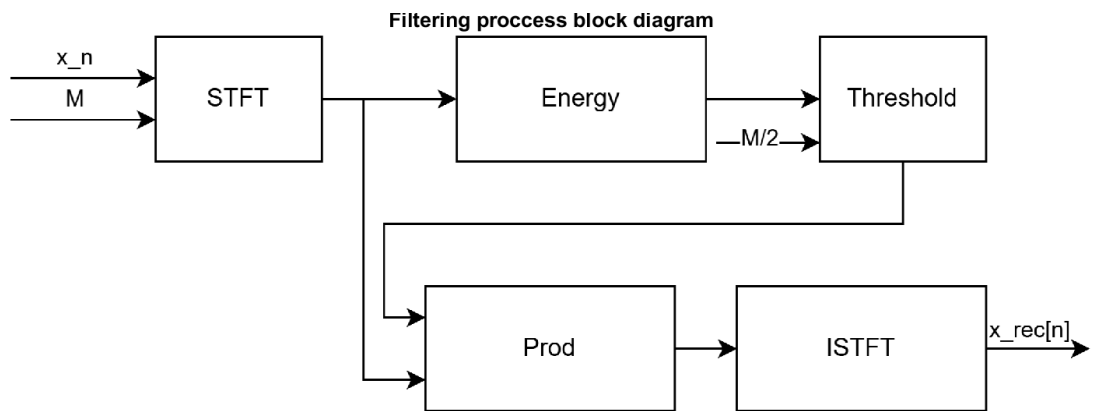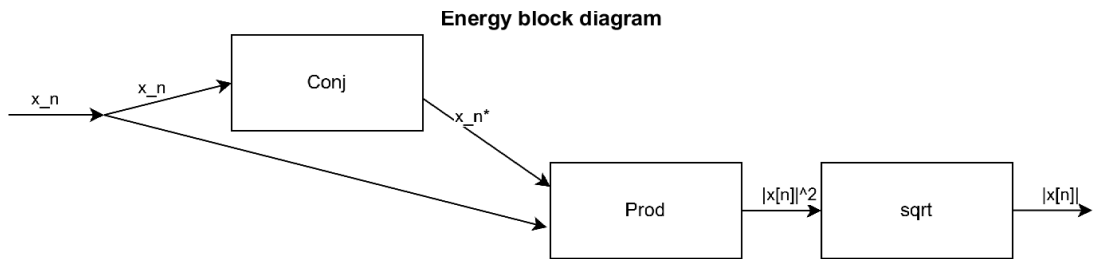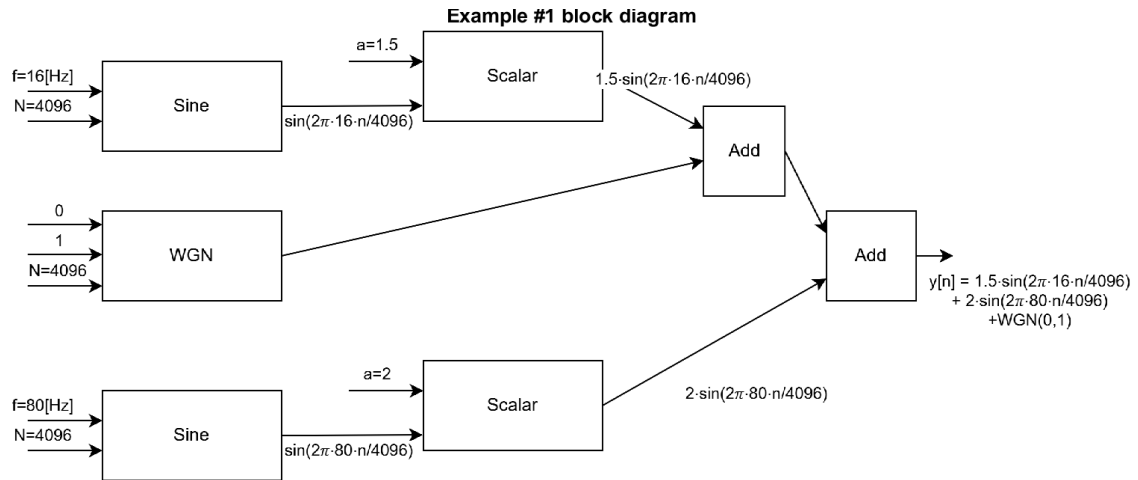**5 known frequencies in time domain**

Random 5 frequencies

We sample 5 random frequencies using Rand block, scale them to [0, 50] Offset to [30, 60] using Add block from HW1 And create sinusoidal signal using Sine block.

5 random frequencies in time domain

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 ± 16 Hz and ± 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 block diagram

f=16[Hz]
N=4096 → Sine → sin(2π·16·n/4096)
a=1.5 → Scalar → 1.5·sin(2π·16·n/4096) → Add

0
1
N=4096 → WGN → Add

f=80[Hz]
N=4096 → Sine → sin(2π·80·n/4096)
a=2 → Scalar → 2·sin(2π·80·n/4096) → Add

y[n] = 1.5·sin(2π·16·n/4096)
+ 2·sin(2π·80·n/4096)
+WGN(0,1)

Energy block diagram

x_n → x_n → Conj → x_n*
→ Prod → |x[n]|^2 → sqrt → |x[n]|

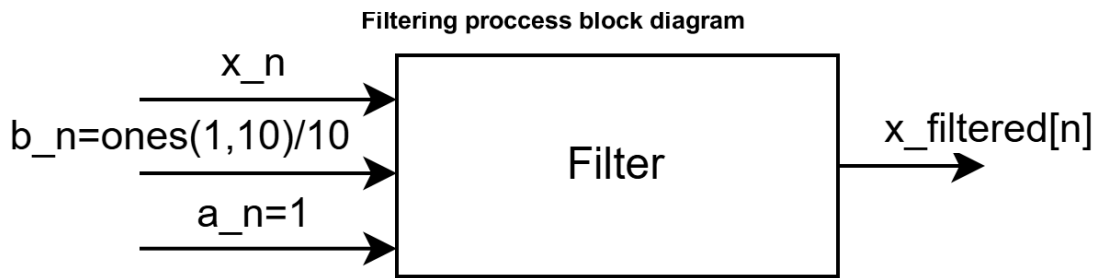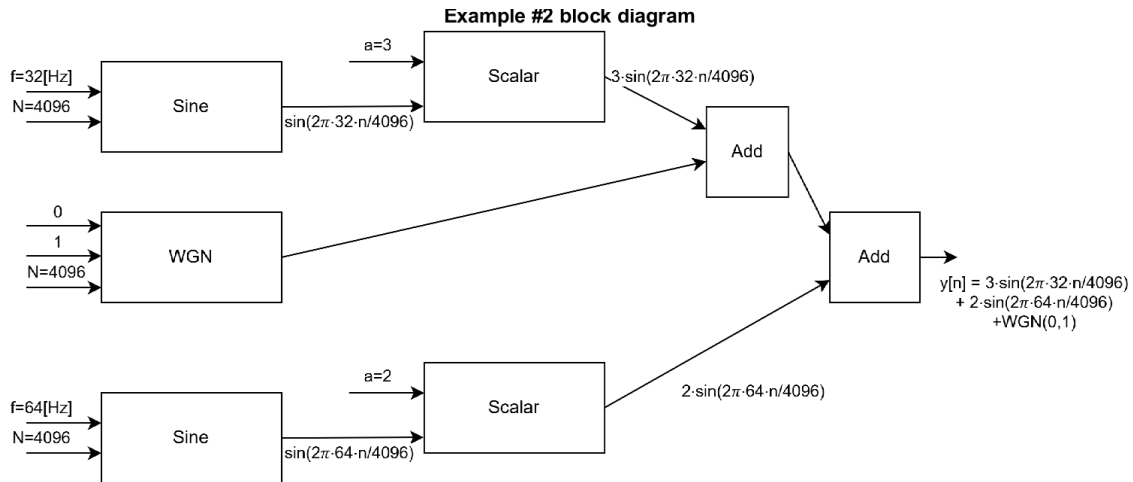**Filtering proccess block diagram**

x_n
M → STFT → Energy → Threshold
—M/2→

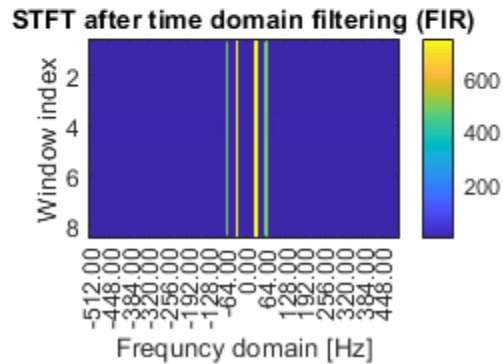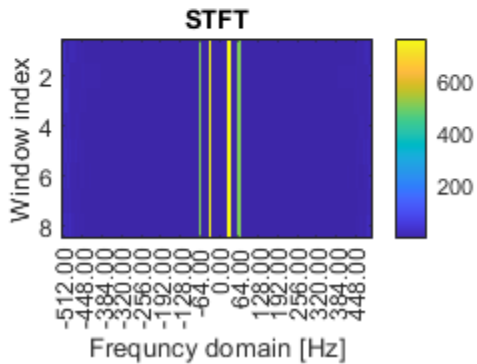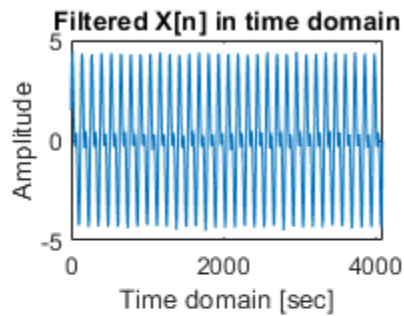STFT → Prod → ISTFT → x_rec[n]

Example #1
Filtering in frequency domain
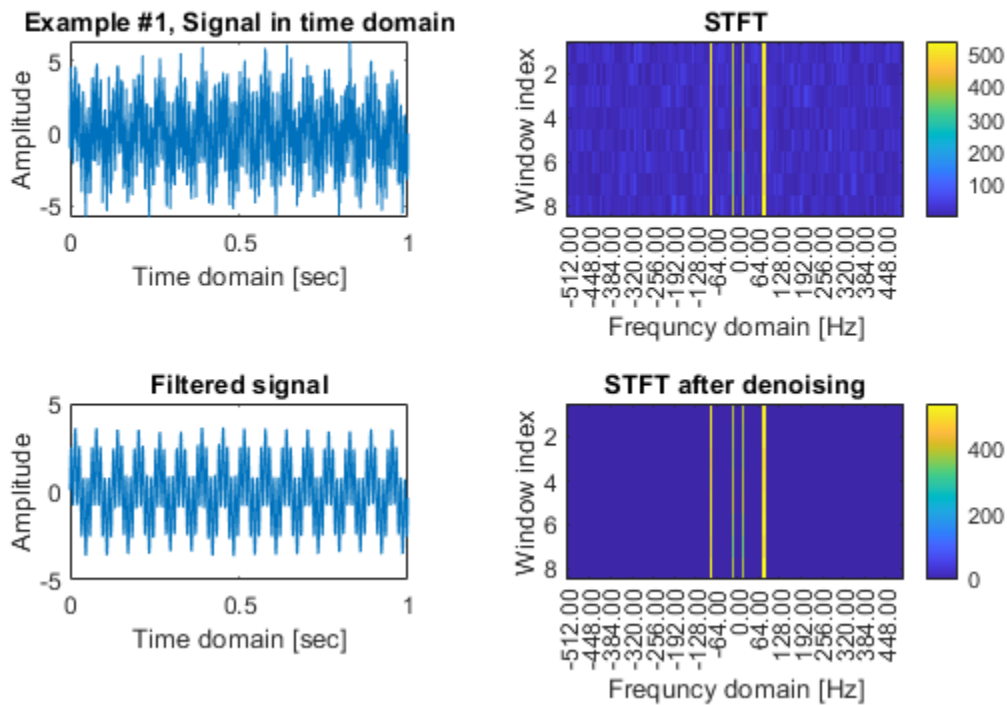
Signal Example 2 - Filtering in time domain

The simplest low-pass filter is a moving average filter. Here, we define the LPF coefficients as 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 ±32 Hz and ±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 block diagram

f=32[Hz]
N=4096 → **Sine** → $\sin(2\pi \cdot 32 \cdot n/4096)$

a=3 → **Scalar** → $3 \cdot \sin(2\pi \cdot 32 \cdot n/4096)$ → **Add**

0
1
N=4096 → **WGN** → **Add**

Add → $y[n] = 3 \cdot \sin(2\pi \cdot 32 \cdot n/4096)$
$+ 2 \cdot \sin(2\pi \cdot 64 \cdot n/4096)$
$+ WGN(0,1)$

f=64[Hz]
N=4096 → **Sine** → $\sin(2\pi \cdot 64 \cdot n/4096)$

a=2 → **Scalar** → $2 \cdot \sin(2\pi \cdot 64 \cdot n/4096)$

Filtering proccess block diagram

x_n →
b_n=ones(1,10)/10 → **Filter** → x_filtered[n]
a_n=1 →
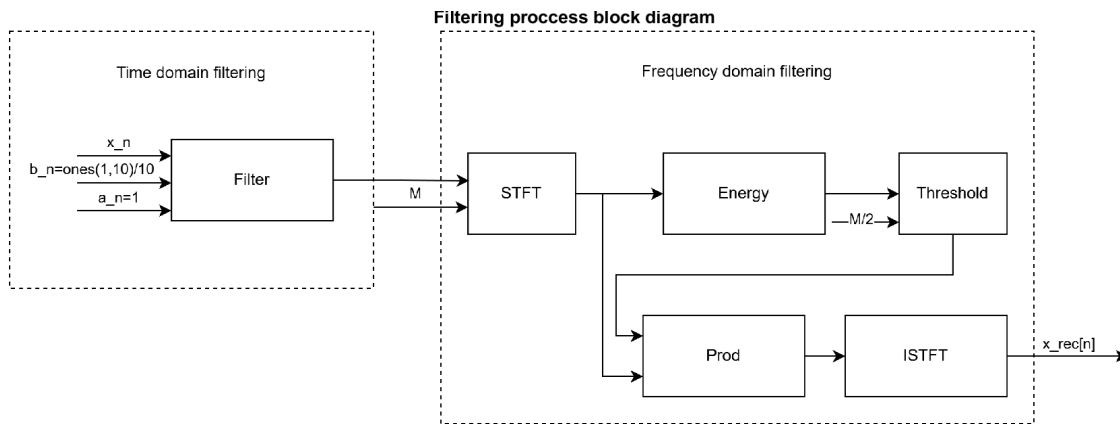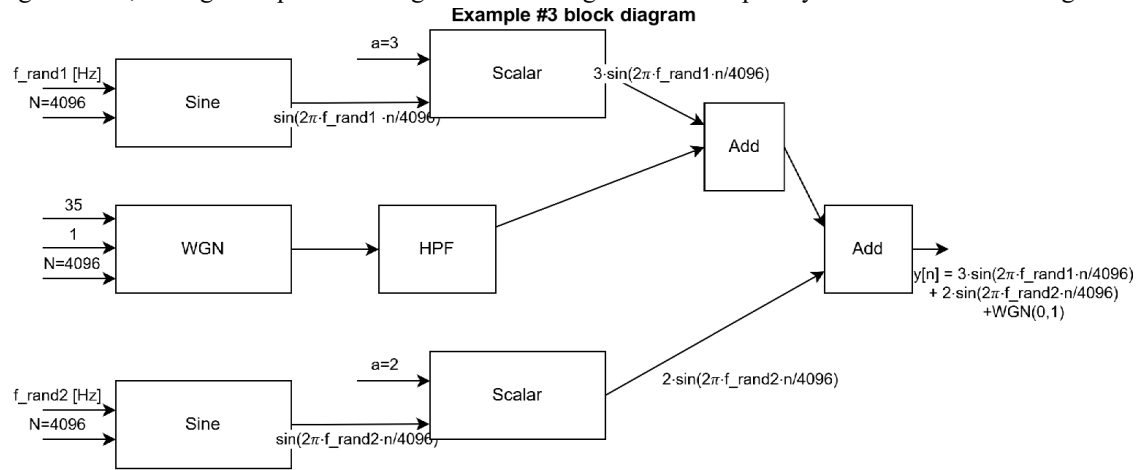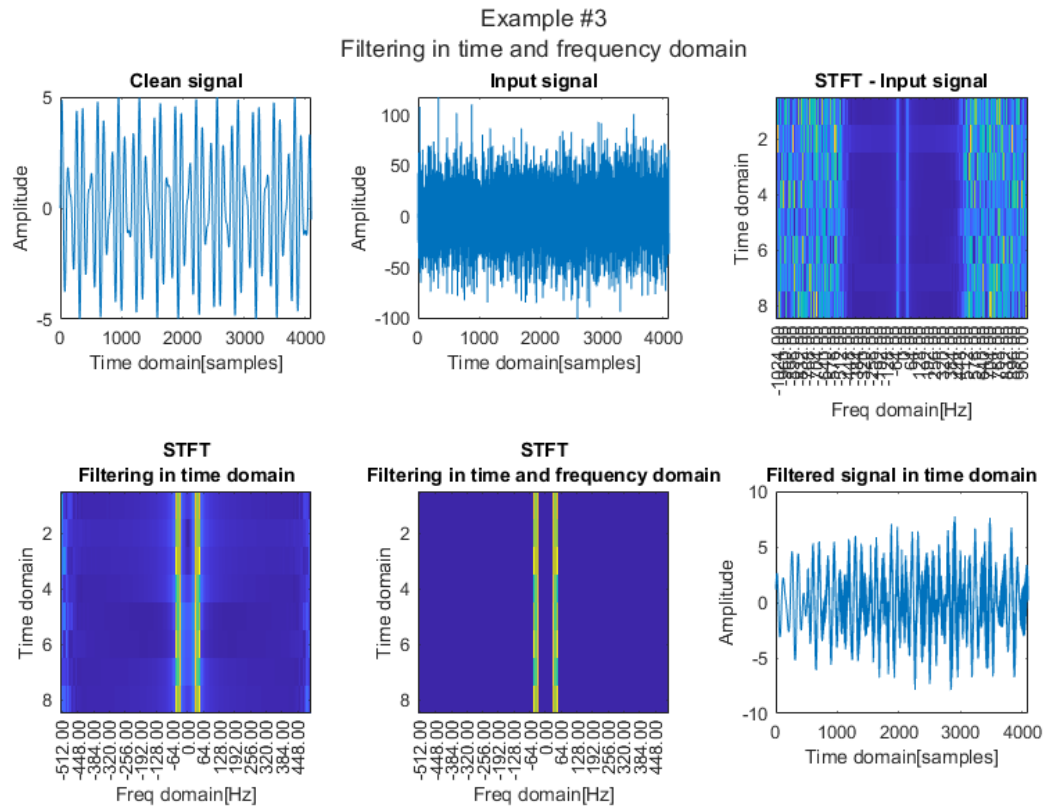

Example #2
Filtering in time domain
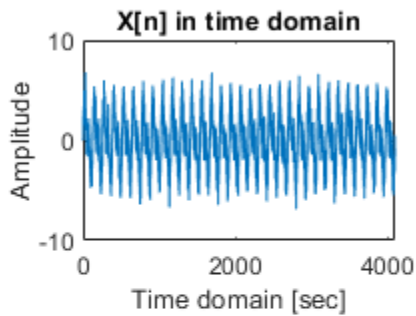
Signal Example 3 - Filtering in frequency and time domain

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.

Example #3 block diagram

Filtering proccess block diagram

Example #3
Filtering in time and frequency domain

Clean signal

Amplitude vs Time domain[samples]

Input signal

Amplitude vs Time domain[samples]

STFT - Input signal

Time domain vs Freq domain[Hz]

STFT
Filtering in time domain

Time domain vs Freq domain[Hz]

STFT
Filtering in time and frequency domain

Time domain vs Freq domain[Hz]

Filtered signal in time domain

Amplitude vs Time domain[samples]

Published with MATLAB® R2024a