DSP Worksheet 2

Sangheon Park, Donghoon Seu, Jeet Ajmani

March 3, 2025

1 Filters with Scipy

1.1 Input Signal Analysis

Table 1 shows the properties of the input signal $x[n]$

Table 1: Properties of file input_1.wav

Property	**Value**
Duration	4.000s
# of channels	1 (mono)
Sample rate	48000 Hz
Signal length	192000 samples
Bit depth	64 bits per sample

The waveform is illustrated in Figure 1. The input signal sounds like a sine wave obscured with noise. We chose to visualize the spectrum of $x[n]$ with its magnitude spectrum in Figure 2. The y-axis is scaled by log_{10} and is represented in decibels for a more detailed view.

In the magnitude spectrum, a peak is clearly visible between 0 Hz and 5000 Hz. To calculate the exact frequency value of the peak we used the Numpy function `np.argmax()`, but it resulted in the value of the frequency bin where the peak occurs. To convert the frequency bin to frequency in Hz, we used the following formula:

$$f = \frac{\text{bin index} \times \text{ sample rate}}{\text{FFT size}} \tag{1}$$

Solving for f resulted in 234.0087890625 Hz for the principal frequency component. Zooming into the magnitude spectrum in Figure 3 clearly corroborates this value as the peak.

We decided to take the extra step to plot the magnitude spectrum of a sine wave with the same frequency and overlay it on the plot of $x[n]$. Figure 4 shows the alignment of the peaks and again corroborates that the input signal $x[n]$ is a pure sine tone obscured by noise.

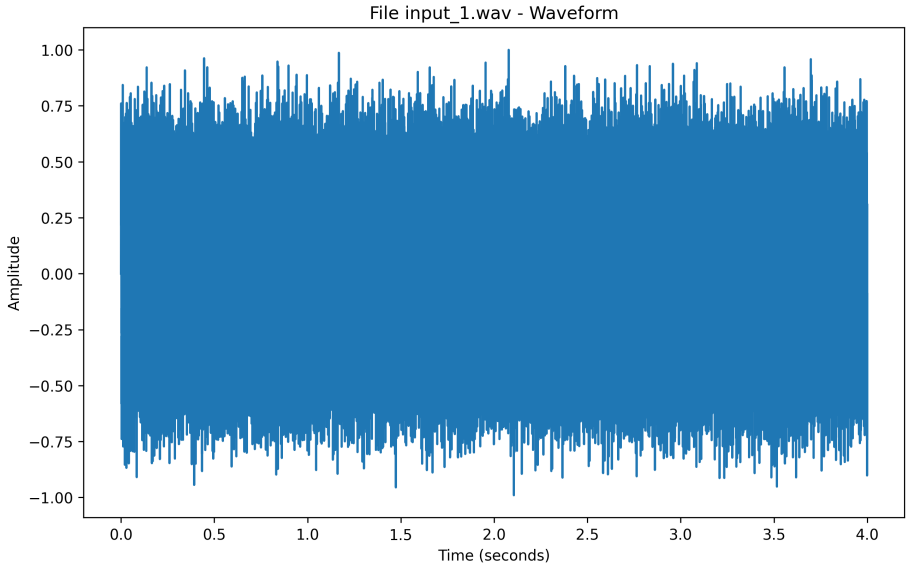

Figure 1: A waveform plot of the input signal.

1.2 Filter Design

We chose to use a Butterworth IIR Low-pass filter with a cutoff frequency of 300 Hz and order of 4 to suppress the noise in $x[n]$. The filter coefficients are as follows:

Numerator coefficients: [1.41272654e-07, 5.65090614e-07, 8.47635922e-07, 5.65090614e-07, 1.41272654e-07]

Denominator coefficients: [1, -3.89738598, 5.69739, -3.70246715, 0.90246539]

1.3 Filter Analysis

The magnitude response of the low-pass filter is shown in Figure 5 and the phase response is shown in Figure 6. The filter has a non-linear phase typical of IIR filters and the magnitude response has a roll-off consistent to the order and the attenuated frequencies. Since the order was only 4, there is no visible phase or magnitude distortion in the responses.

1.4 Filter Application

The spectrum of the filtered signal $y[n]$ in Figure 7 shows the reduced noise as a result of applying the low-pass filter with a cutoff frequency of 300 Hz. The original peak is still visible and the rest of the low frequency components remain unchanged. When listening to the audio result, attached as `filtered_signal.wav`, it is clear that the noise is isolated and only the pure sine wave from the original file remains.

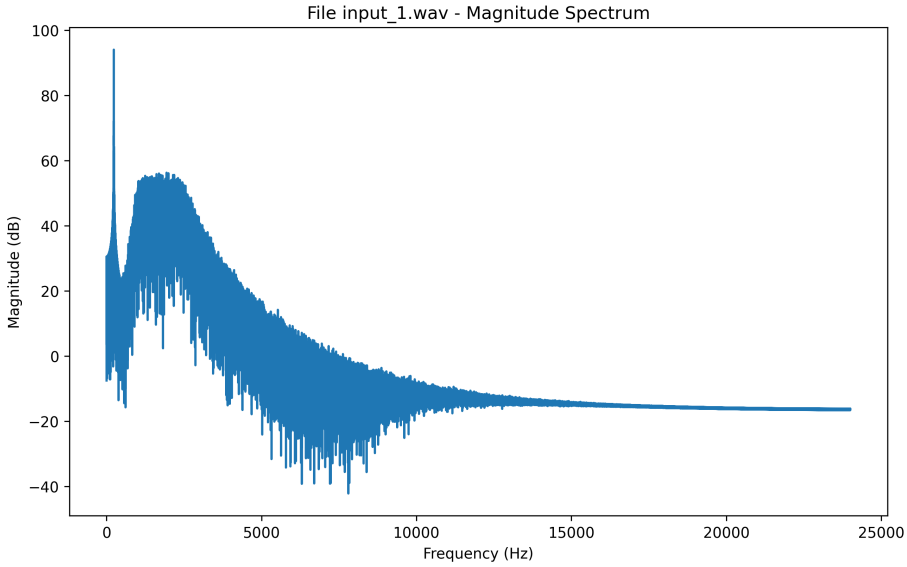

Figure 2: The magnitude spectrum of $x[n]$

2 IIR: Bilinear Transform

2.1 Transfer Function

The impedance of the capacitor is purely imaginary and depends on its capacity C:

$$Z_C = \frac{1}{sC}$$

The impedance of the resistor R is

$$Z_R = R$$

Transfer Function of the circuit is

$$H(s) = \frac{V_{out}}{V_{in}}$$

From the image, since the resistor is located between out and ground, overall functionis

$$H(s) = \frac{Z_R}{Z_R + Z_C}$$

, which can be converted to

$$H(s) = \frac{R}{R + \frac{1}{sC}}$$

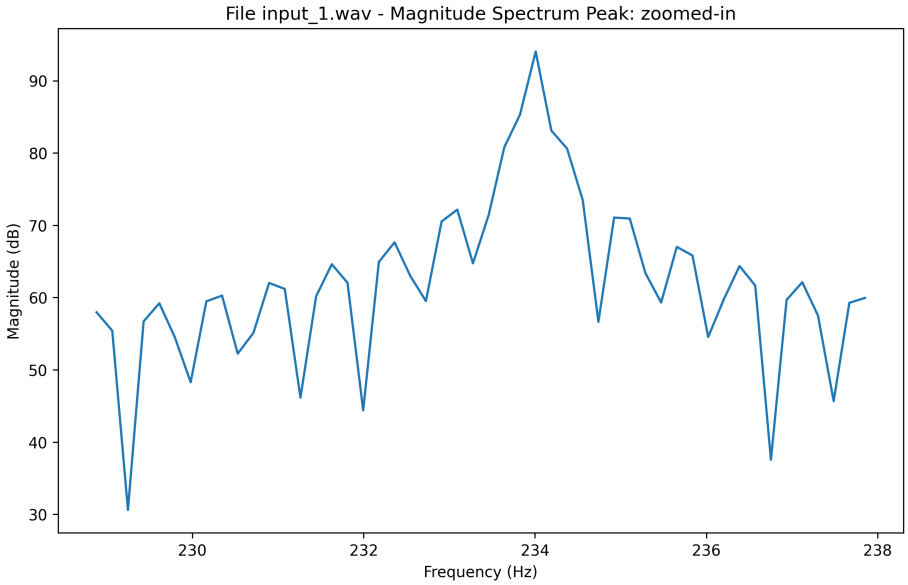

Figure 3: The magnitude spectrum of $x[n]$: zoomed-in to the peak.

$$H(s) = \frac{R}{R + \frac{1}{sC}}$$

$$H(s) = \frac{sRC}{sRC + 1}$$

2.2 Filter Coefficients

To get bilinear transform, we need to subsitute s with:

$$s = \frac{2}{T} \cdot \frac{1 - z^{-1}}{1 + z^{-1}}$$

where

- s is the Laplace transform variable,
- $T = \frac{1}{f_s}$ is the sampling period,
- z is the Z-transform variable.

In the Laplace domain, its transfer function is:

$$H(s) = \frac{\omega_c}{s + \omega_c}$$

ω_c is cutoff frequency,

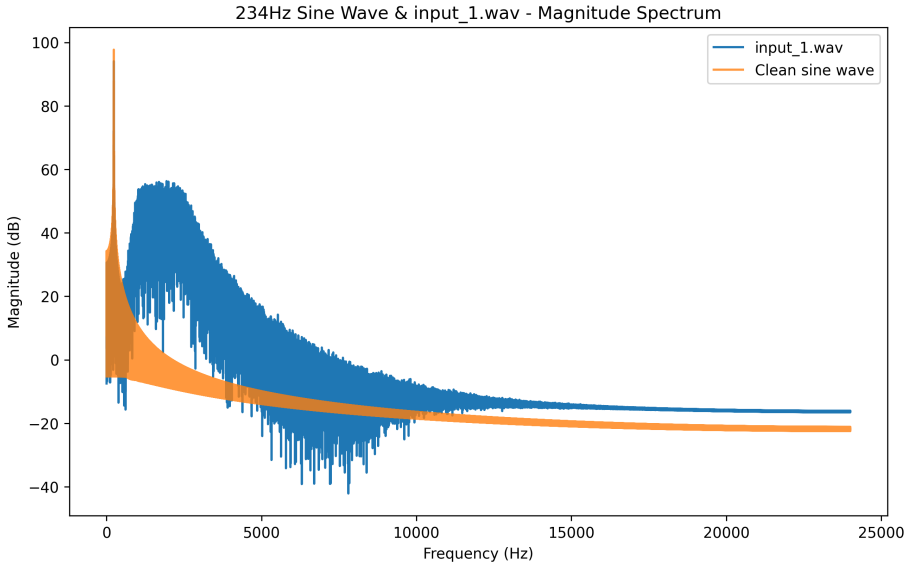

Figure 4: The magnitude spectrum of a pure sine wave with $f = 234.0087890625$ Hz overlaid on the magnitude spectrum of $x[n]$.

Applying the substitution of s,

$$H(z) = \frac{\omega_c}{\frac{2}{T} \cdot \frac{1-z^{-1}}{1+z^{-1}} + \omega_c}$$

Multiplying by $1 + z^{-1}$ in both numerator and denominator,

$$H(z) = \frac{\omega_c(1+z^{-1})}{\frac{2}{T}(1-z^{-1}) + \omega_c(1+z^{-1})}$$

$$H(z) = \frac{\omega_c(1+z^{-1})}{\frac{2}{T} - \frac{2}{T}z^{-1} + \omega_c + \omega_c z^{-1}}$$

$$H(z) = \frac{\omega_c(1+z^{-1})}{\frac{2}{T} + \omega_c + (\omega_c - \frac{2}{T})z^{-1}}$$

To make transfer function to standard form which is

$$H(z) = \frac{b_0 + b_1 z^{-1}}{1 + a_1 z^{-1}}$$

we divide the numerator and denominator by $\frac{2}{T} + \omega_c$

$$H(z) = \frac{\frac{\omega_c}{\frac{2}{T}+\omega_c} + \frac{\omega_c}{\frac{2}{T}+\omega_c} z^{-1}}{1 + \frac{\omega_c - \frac{2}{T}}{\frac{2}{T}+\omega_c} z^{-1}}$$

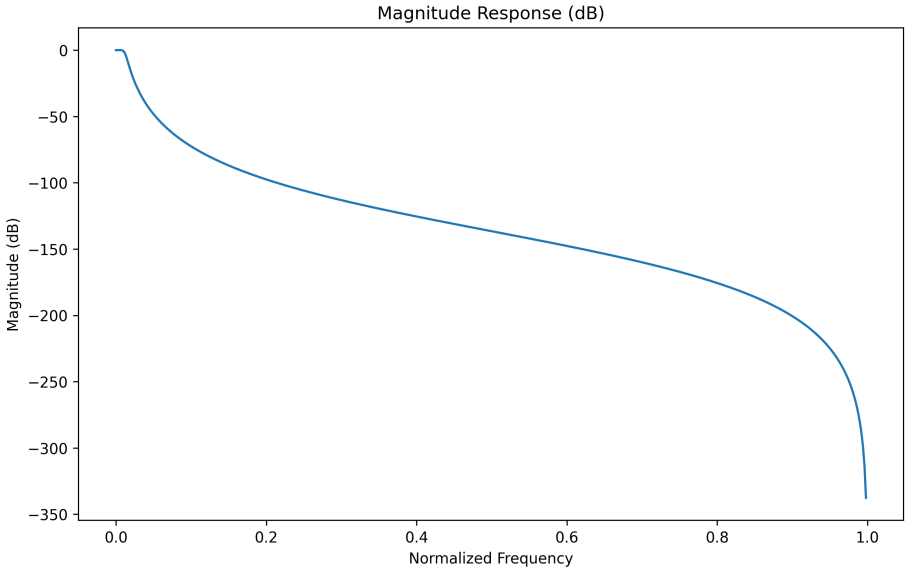

Figure 5: The magnitude response of the low-pass filter.

From this, we identify the filter coefficients,

$$b_0 = \frac{\omega_c}{\frac{2}{T} + \omega_c}, \quad b_1 = b_0$$

$$a_1 = \frac{\omega_c - \frac{2}{T}}{\frac{2}{T} + \omega_c}$$

Since $T = \frac{1}{f_s}$, substituting this into the coefficients

$$b_0 = \frac{\omega_c T}{2 + \omega_c T}, \quad b_1 = b_0$$

$$a_1 = \frac{\omega_c T - 2}{\omega_c T + 2}$$

The discrete-time difference equation is derived from $H(z)$

$$y[n] = b_0 x[n] + b_1 x[n-1] - a_1 y[n-1]$$

2.3 Coefficient Function

$$b_0 = 1, \quad b_1 = 1$$

$$a_0 = 1 + \frac{2RC}{T}, \quad a_1 = 1 - \frac{2RC}{T}$$

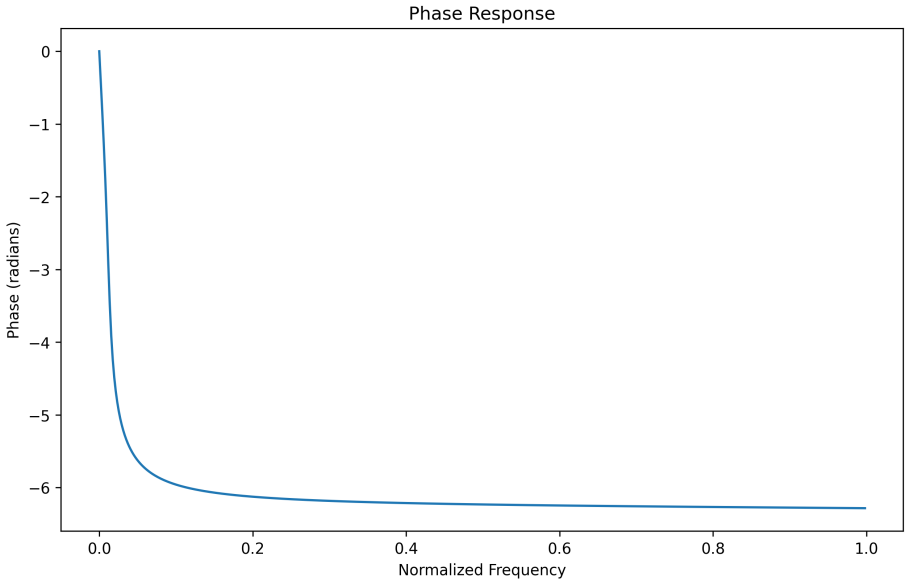

Figure 6: The phase response of the low-pass filter.

- T is the sampling period
$$T = \frac{1}{f_s}$$
- The sampling rate:
$$f_s = 48,000 \text{ Hz}$$
- Assumed values:
$$R = 1000\Omega, \quad C = 1\mu\text{F}$$

The frequency response of the filter is given by

$$H(e^{j\omega}) = \frac{b_0 + b_1 z^{-1}}{a_0 + a_1 z^{-1}}$$

This code computes and plots the magnitude and phase response of a simple recursive filter using given resistance and capacitance values.

```
from scipy.signal import freqz
import numpy as np
import matplotlib.pyplot as plt

# Filter coefficients
b = [1, 1]  # Numerator coefficients
T = 1 / 48000  # Sampling period (48,000 Hz sampling rate)
R = 1000       # Assumed resistance (1k Ohm)
```

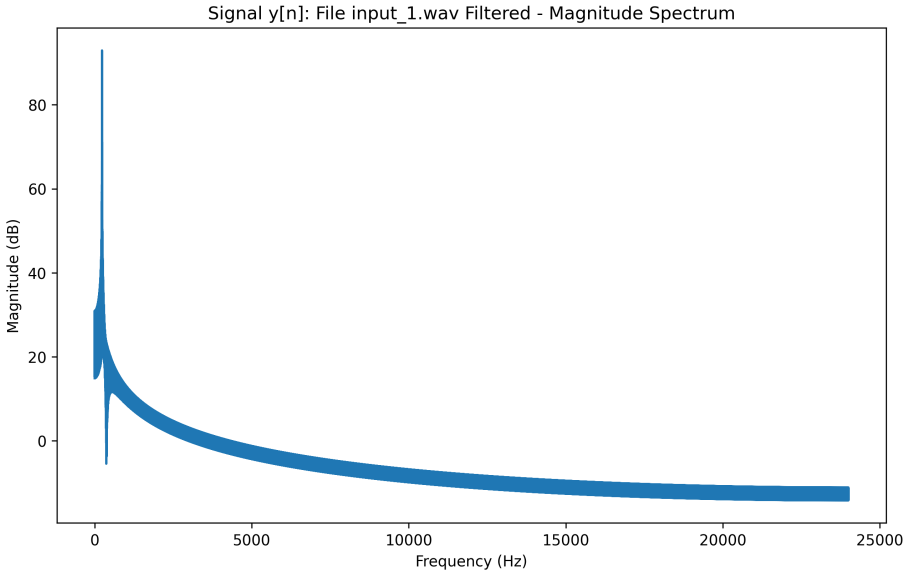

Figure 7: The magnitude spectrum of y[n], the filtered signal.

```
C = 1e-6       # Assumed capacitance (1 microfarad)

# Denominator coefficients
a = [1 + (2 * R * C) / T, 1 - (2 * R * C) / T]

# Calculate frequency response
w, h = freqz(b, a, worN=1024)
frequencies = w * 48000 / (2 * np.pi)

# Plot frequency response
plt.figure(figsize=(12, 8))

# Magnitude Response
plt.subplot(2, 1, 1)
plt.plot(frequencies, 20 * np.log10(abs(h) + 1e-10))
plt.title('Frequency Response - Magnitude')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Magnitude (dB)')
plt.grid()

# Phase Response
plt.subplot(2, 1, 2)
plt.plot(frequencies, np.angle(h))
```

```
plt.title('Frequency Response - Phase')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Phase (radians)')
plt.grid()

plt.tight_layout()
plt.show()
```

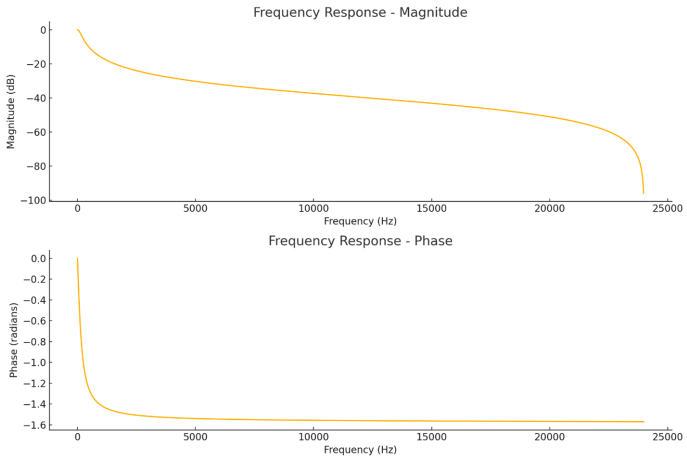

Figure 8: 2.3 Magnitude and Phase Response

The magnitude response has flat passband at lower frequencies. A sharp roll-off at higher frequencies, consistent with a low-pass filter.

The phase response is smooth without abrupt changes, which indicates minimal phase distortion in the filter's performance.

2.4 Filter Function

$$b_0 = 1, \quad b_1 = 1$$
$$a_0 = 1 + \frac{2RC}{T}, \quad a_1 = 1 - \frac{2RC}{T}$$

- Assumed values:
$$R = 1000\Omega, \quad C = 1\mu\text{F}$$

The discrete-time impulse response is,

$$h[n] = b_0\delta[n] + b_1\delta[n-1] - a_1 h[n-1]$$

The step response is

$$s[n] = \sum_{k=0}^{n} h[k]$$

This code computes and plots the impulse and step responses of a discrete-time filter based on given resistance and capacitance values.

```
from scipy.signal import lfilter, dimpulse, dstep
import matplotlib.pyplot as plt

# Define filter coefficients
b = [1, 1]  # Numerator coefficients
T = 1 / 48000  # Sampling period (48,000 Hz sampling rate)
R = 1000       # Assumed resistance (1k Ohm)
C = 1e-6       # Assumed capacitance (1 microfarad)

# Denominator coefficients
a = [1 + (2 * R * C) / T, 1 - (2 * R * C) / T]

# Calculate impulse response
t_imp, h_imp = dimpulse((b, a, 1), n=50)

# Calculate step response
t_step, s_step = dstep((b, a, 1), n=50)

# Plot impulse response
plt.figure(figsize=(12, 6))
plt.stem(t_imp, h_imp[0].flatten(), basefmt=" ", use_line_collection=True)
plt.title('Impulse Response')
plt.xlabel('n (samples)')
plt.ylabel('Amplitude')
plt.grid()

# Plot step response
plt.figure(figsize=(12, 6))
plt.stem(t_step, s_step[0].flatten(), basefmt=" ", use_line_collection=True)
plt.title('Step Response')
plt.xlabel('n (samples)')
plt.ylabel('Amplitude')
plt.grid()

plt.show()
```

The impulse response decays smoothly, indicating that the filter is stable. The impulse response also suggests that the filter has a finite duration characteristic, which is expected for the designed filter.

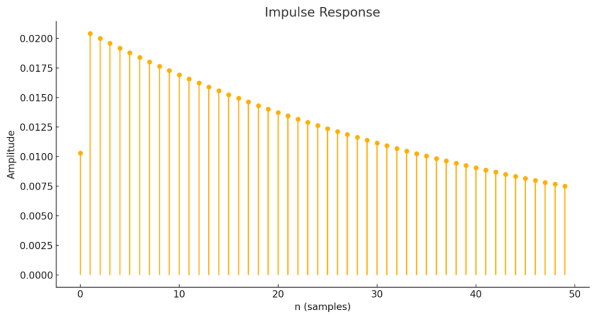

Figure 9: 2.4 Impulse Response

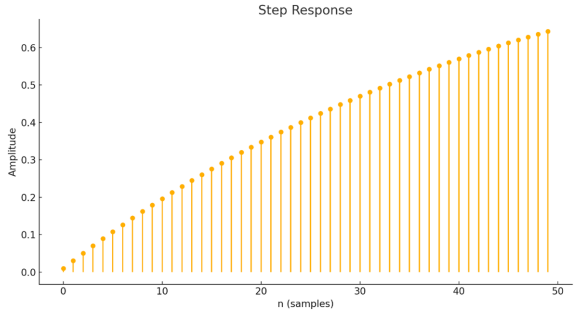

Figure 10: 2.4 Step Response

The step response shows a gradual increase that stabilizes while no oscillatory behavior, confirming filter stability.

2.5 Filter Sweep

A linear cutoff trajectory was created:

$$f_c[n] = \text{linspace}(10, 20000, N)$$

where the total number of samples is:

$$N = 4 \times 48,000$$

This trajectory transitions linearly from 10 Hz to 20 kHz.

A white noise signal was generated with a length of $N = 4 \times 48,000$ samples using:

$$x = \text{np.random.normal}(0, 1, N)$$

$$b_0 = 1, \quad b_1 = 1$$
$$a_0 = 1 + \frac{2RC}{T}, \quad a_1 = 1 - \frac{2RC}{T}$$

Python code that applies a time-varying low-pass filter to white noise using computed coefficients and saves the output as a WAV file.

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.io.wavfile as wav
from scipy.signal import spectrogram, lfilter

# Parameters
fs = 48000  # Sampling rate (Hz)
N = 4 * fs  # Total length: 4 * 48,000
R = 1000    # Assumed resistance (1k Ohm)
C = 1e-6    # Assumed capacitance (1 microfarad)
T = 1 / fs  # Sampling period

# Create a linear cutoff trajectory
fc = np.linspace(10, 20000, N)

# Generate white noise signal
np.random.seed(0)
x = np.random.normal(0, 1, N)

# Define filter coefficients
b = [1, 1]
y = np.zeros(N)

# Apply filtering
for i in range(1, N):
    a = [1 + (2 * R * C) / T, 1 - (2 * R * C) / T]
    y[i] = (b[0] * x[i] + b[1] * x[i - 1] - a[1] * y[i - 1]) / a[0]

# Save filtered signal
wav.write('filtered_sweep.wav', fs, y.astype(np.float32))
```

The input white noise exhibits a broad frequency spectrum, characteristic of white noise. The filtered signal shows attenuation of high frequencies due to the linear sweep of the cutoff frequency.

The STFT plots confirm that the filter sweep progressively attenuates high frequencies as the cutoff frequency increases. The filter maintains low-frequency components as expected. The filtered signal exhibits a smoother spectrum compared to the white noise input.

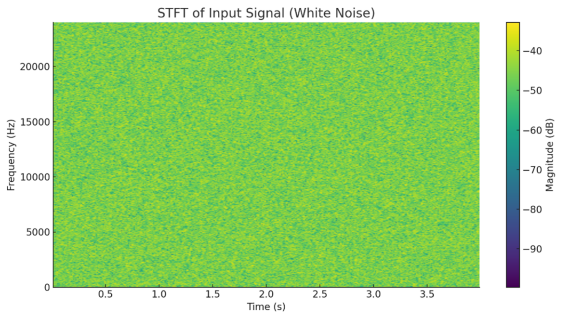

Figure 11: 2.5 White Noise Input Signal

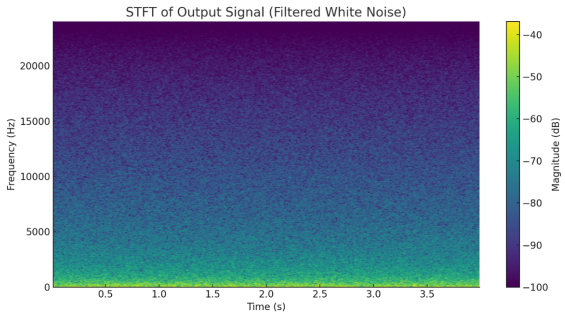

Figure 12: 2.5 White Noise Output Signal