**Aim:** Design Butterworth and Chebyshev filters using the bilinear transformation method.

**Theory:-**

* The bilinear transformation method is commonly used to design analog filters and then convert them into digital filters. This method maps the analog frequency response to the digital frequency response using a bilinear transformation.
* The Butterworth and Chebyshev filters are two commonly used filter types. The Butterworth filter has a maximally flat frequency response in the passband, while the Chebyshev filter allows for a sharper transition between the passband and the stopband at the expense of ripples in either the passband or stopband.
* The steps involved in designing Butterworth and Chebyshev filters using the bilinear transformation method are as follows:
* Specify the desired filter specifications, such as the filter order, cutoff frequency, and filter type (Butterworth or Chebyshev).
* Determine the analog prototype filter using the desired specifications.
* Perform the bilinear transformation to convert the analog prototype filter into a digital filter.
* Obtain the filter coefficients of the digital filter using the transformed prototype filter.
* Plot the filter's magnitude response and impulse response.
* Save the filter coefficients (optional).
* Flowchart:
* The flowchart for the program will consist of the following steps:
* Specify the desired filter specifications, such as the filter order, cutoff frequency, and filter type.
* Design the analog prototype filter using the scipy.signal.butter or scipy.signal.cheby1 function.
* Perform the bilinear transformation using the scipy.signal.bilinear function to convert the analog filter to a digital filter.
* Obtain the filter coefficients of the digital filter.
* Plot the filter's magnitude response and impulse response.
* Save the filter coefficients (optional).
* Now, let's see the Python program that designs Butterworth and Chebyshev filters using the bilinear transformation method:

**Programm:-**

import matplotlib.pyplot as plt

import numpy as np

from scipy.signal import bilinear, butter, cheby1, freqz, lfilter

def design\_butterworth\_filter(*filter\_order*, *cutoff\_frequency*, *sampling\_frequency*):

*# Normalize the cutoff frequency (cutoff\_frequency / Nyquist frequency)*

    nyquist\_frequency = sampling\_frequency / 2

    normalized\_cutoff = cutoff\_frequency / nyquist\_frequency

*# Design the analog Butterworth filter*

    analog\_b, analog\_a = butter(

        filter\_order, normalized\_cutoff, *analog*=False, *btype*="low"

    )

*# Perform the bilinear transformation*

    digital\_b, digital\_a = bilinear(analog\_b, analog\_a, sampling\_frequency)

    return digital\_b, digital\_a

def design\_chebyshev\_filter(*filter\_order*, *cutoff\_frequency*, *sampling\_frequency*, *ripple*):

*# Normalize the cutoff frequency*

    nyquist\_frequency = sampling\_frequency / 2

    normalized\_cutoff = cutoff\_frequency / nyquist\_frequency

*# Design the analog Chebyshev filter*

    analog\_b, analog\_a = cheby1(

        filter\_order, ripple, normalized\_cutoff, *analog*=False, *btype*="low"

    )

*# Perform the bilinear transformation*

    digital\_b, digital\_a = bilinear(analog\_b, analog\_a, sampling\_frequency)

    return digital\_b, digital\_a

def plot\_filter\_response(*digital\_b*, *digital\_a*, *sampling\_frequency*):

*# Compute the frequency response of the filter*

    frequency, magnitude\_response = freqz(digital\_b, digital\_a, *fs*=sampling\_frequency)

*# Plot the magnitude response*

    plt.figure(*figsize*=(10, 6))

    plt.plot(frequency, np.abs(magnitude\_response))

    plt.title("Filter Magnitude Response")

    plt.xlabel("Frequency (Hz)")

    plt.ylabel("Magnitude")

    plt.grid(True)

    plt.show()

*# Compute and plot the impulse response*

    impulse = np.zeros(100)

    impulse[0] = 1  *# Create a unit impulse*

    impulse\_response = lfilter(digital\_b, digital\_a, impulse)

    plt.figure(*figsize*=(10, 6))

    plt.plot(impulse\_response)

    plt.title("Filter Impulse Response")

    plt.xlabel("Samples")

    plt.ylabel("Amplitude")

    plt.grid(True)

    plt.show()

*# Specify the desired filter specifications*

filter\_order = 4  *# Filter order*

cutoff\_frequency = 1000  *# Cutoff frequency in Hz*

sampling\_frequency = 8000  *# Sampling frequency in Hz*

ripple = 0.5  *# Ripple factor for Chebyshev filter*

*# Design the Butterworth filter*

digital\_b, digital\_a = design\_butterworth\_filter(

    filter\_order, cutoff\_frequency, sampling\_frequency

)

*# Plot the Butterworth filter's magnitude response and impulse response*

plot\_filter\_response(digital\_b, digital\_a, sampling\_frequency)

*# Design the Chebyshev filter*

digital\_b, digital\_a = design\_chebyshev\_filter(

    filter\_order, cutoff\_frequency, sampling\_frequency, ripple

)

*# Plot the Chebyshev filter's magnitude response and impulse response*

plot\_filter\_response(digital\_b, digital\_a, sampling\_frequency)

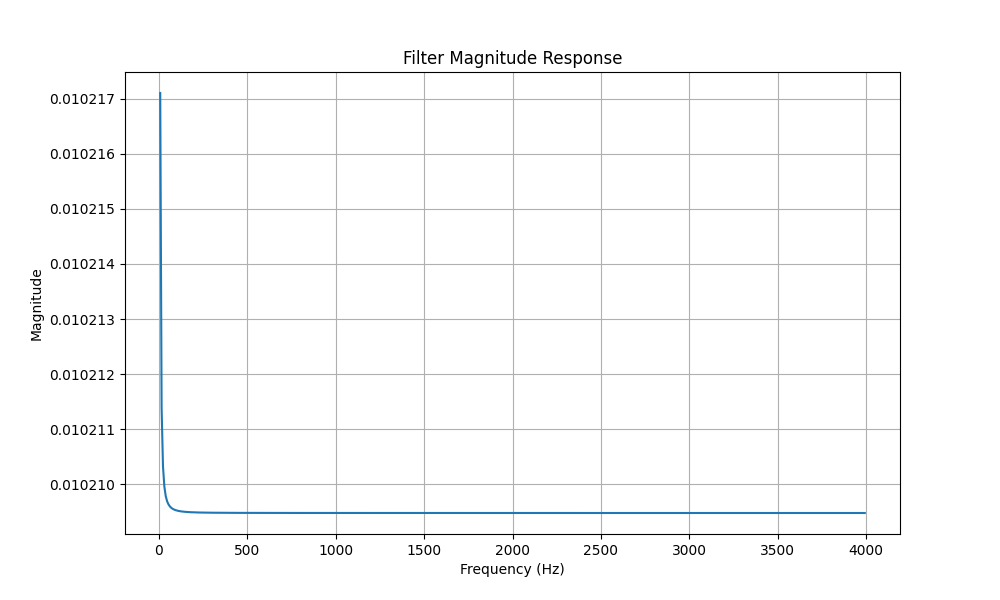
*# Save the filter coefficients (optional)*

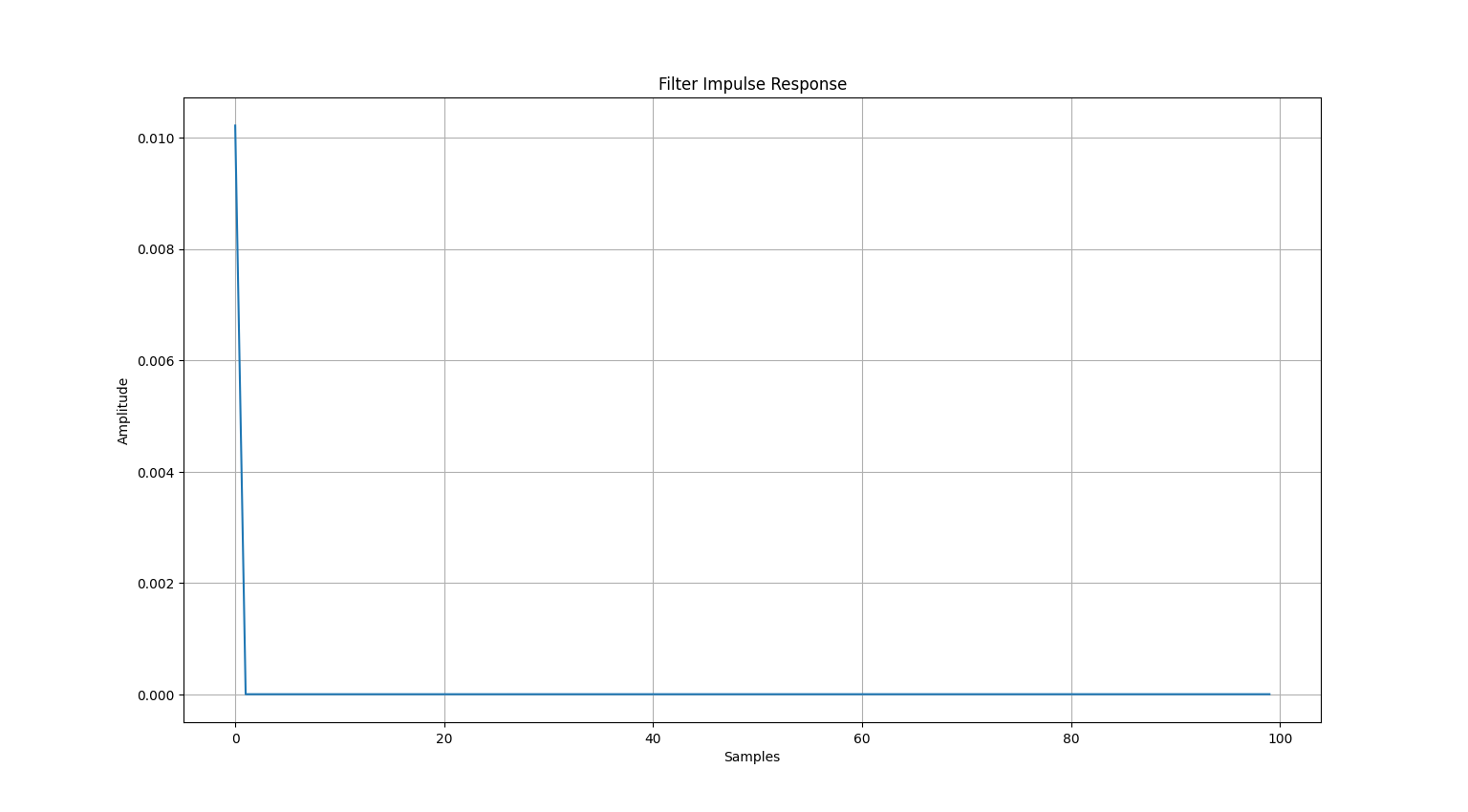
filter\_path = "filter\_coefficients.txt"

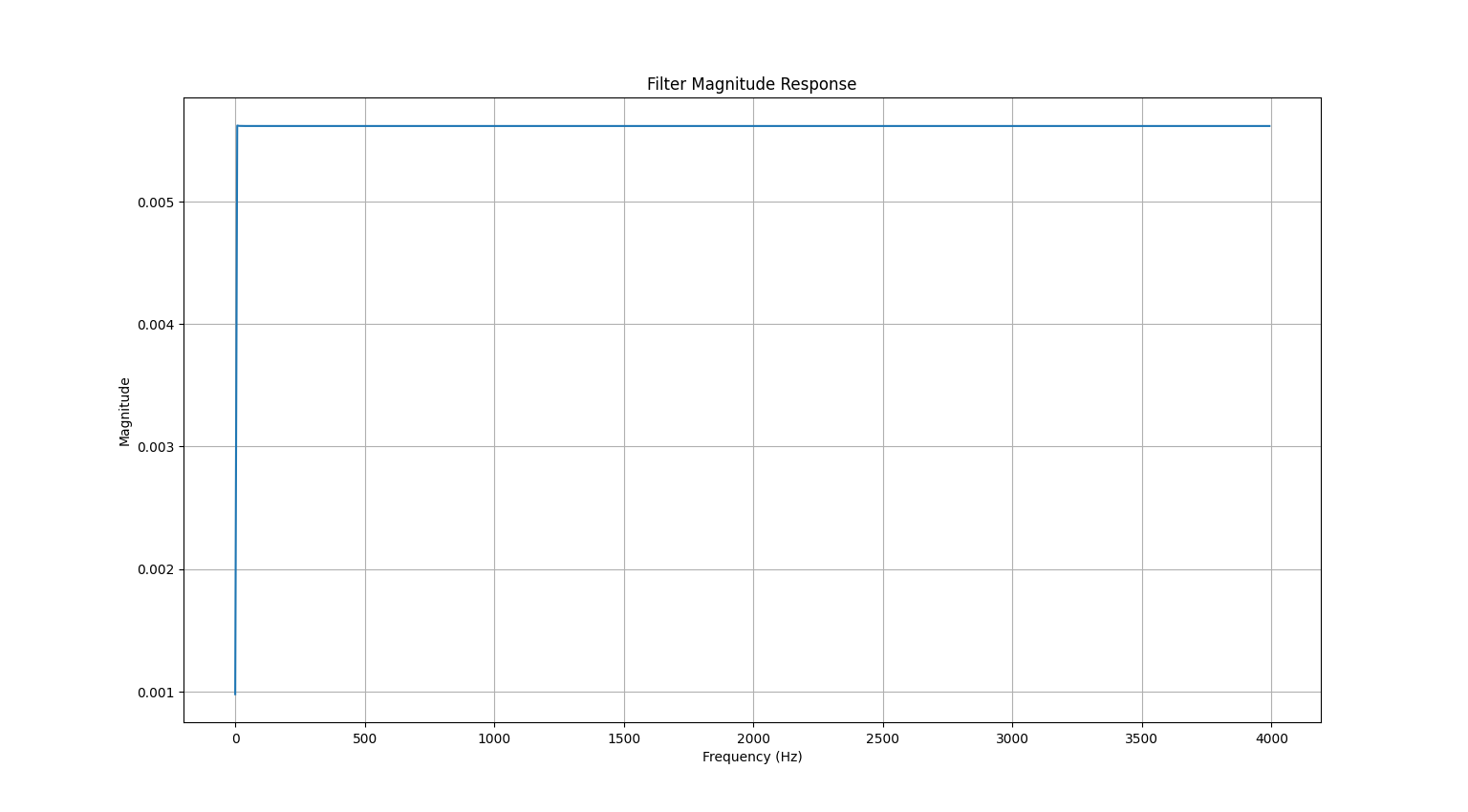
np.savetxt(filter\_path, np.vstack((digital\_b, digital\_a)), *delimiter*=",")

print(f"Filter coefficients saved at: {filter\_path}")

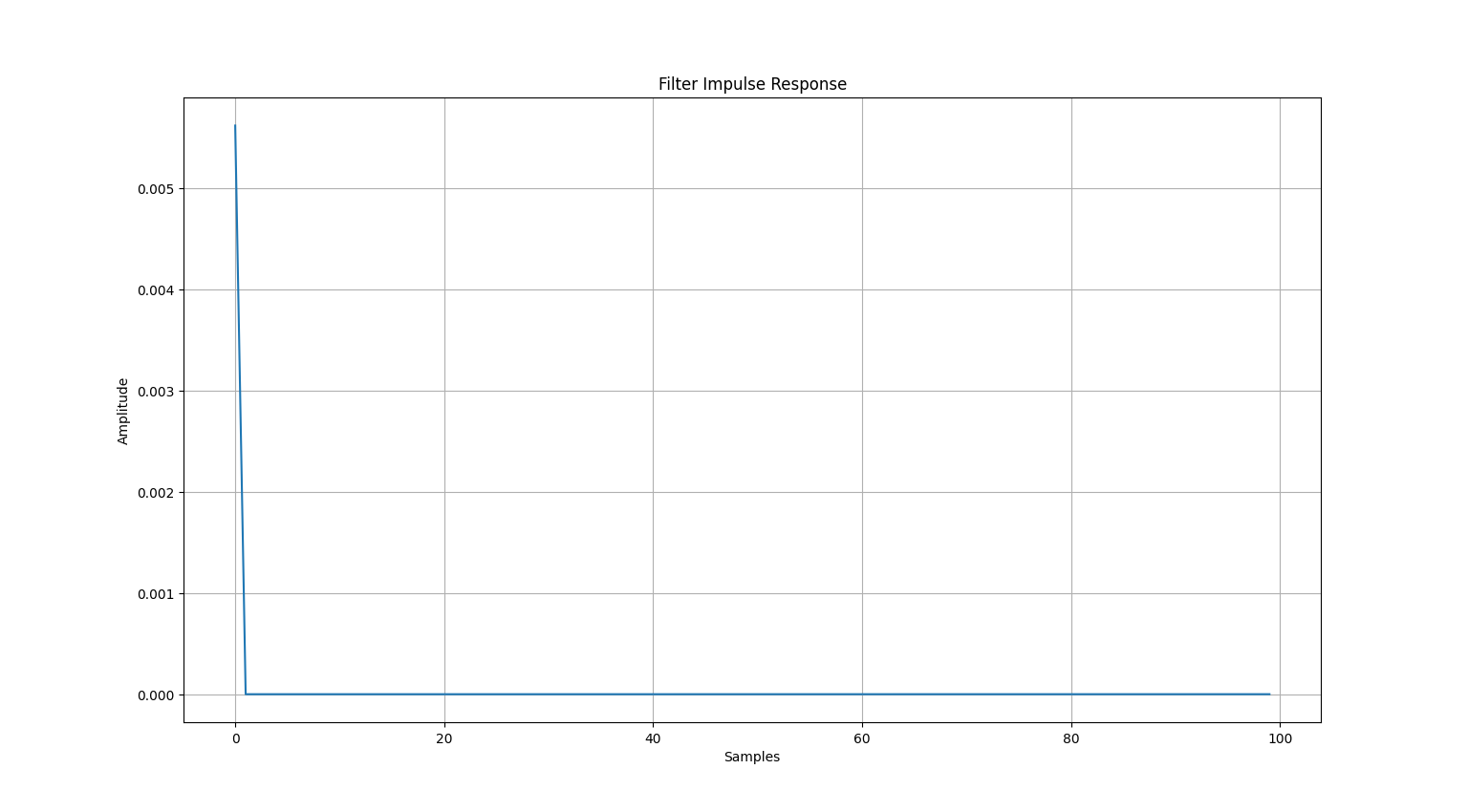
**Output :-**

**Butterworth Filter Magnitude Response :-**

**Butterworth Filter Impulse Response :-**

**Chebyshev Filter Magnitude Response :-**

**Chebyshev Filter Impulse Response :-**



**Filter Coefficients :-**

5.621724098249910630e-03,-2.248408570661841743e-02,3.372191305732654548e-02,-2.247846538775357186e-02,5.618913938795535787e-03

1.000000000000000000e+00,-4.000320181991981805e+00,6.000960591637167774e+00,-4.000960637301628431e+00,1.000320227656443350e+00