DSP lab 06
Name: Basil khowaja bk08432

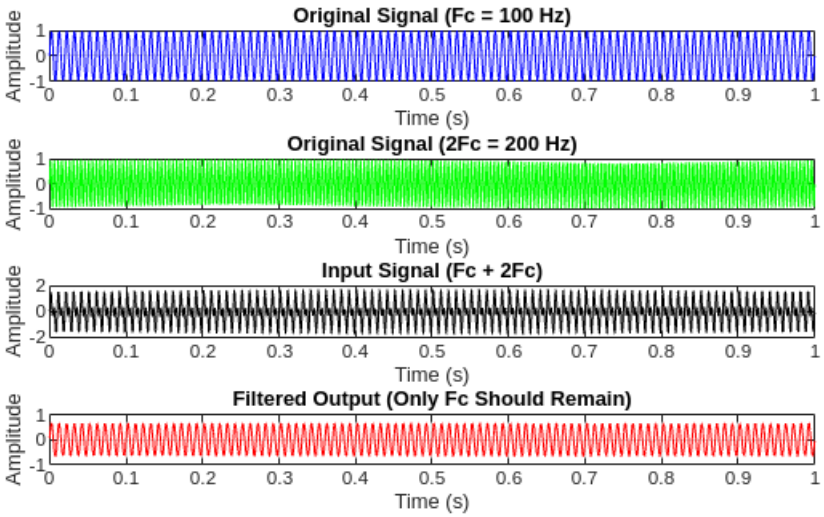
Task 1)

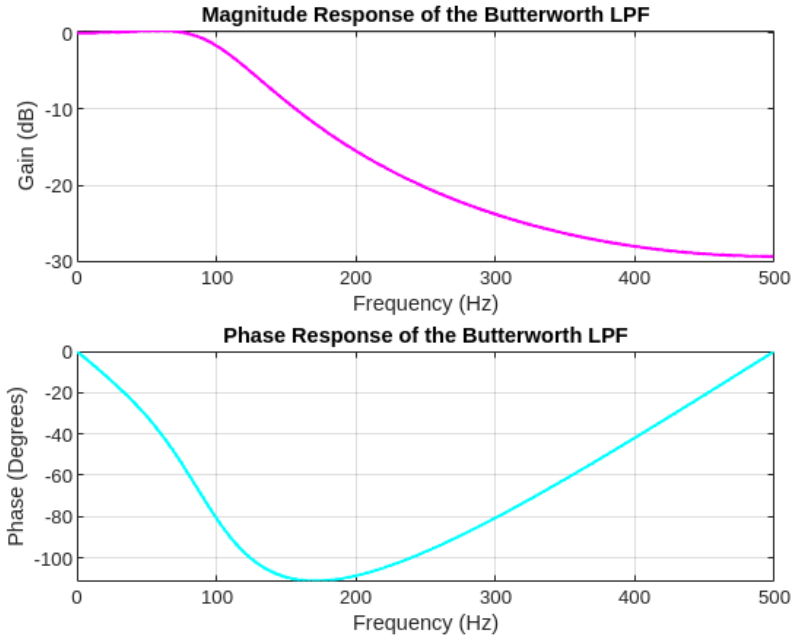
code:

```
clear; close all; clc;

order = 3;
Fc = 100;
Fs = 1000;
Ts = 1 / Fs;

omega_c = 2 * pi * Fc;
omega_c_d = 2 * Fs * tan(omega_c / (2 * Fs));

poles = zeros(order, 1);
for k = 1:order
    theta = (2 * k + order - 1) * pi / (2 * order);
    poles(k) = omega_c_d * exp(1j * theta);
end

poles_z = (1 + poles * Ts / 2) ./ (1 - poles * Ts / 2);

a = real(poly(poles_z));
b = sum(a) / a(1);

t = linspace(0, 1, Fs);
signal_fc = sin(2 * pi * Fc * t);
signal_2fc = sin(2 * pi * 2 * Fc * t);
combined_signal = signal_fc + signal_2fc;

filtered_signal = filtfilt(b, a, combined_signal);

figure;

subplot(5,1,1);
plot(t, signal_fc, 'b');
title('Original Signal (Fc = 100 Hz)');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

subplot(5,1,2);
plot(t, signal_2fc, 'g');
title('Original Signal (2Fc = 200 Hz)');
xlabel('Time (s)');
```

```
ylabel('Amplitude');
grid on;

subplot(5,1,3);
plot(t, combined_signal, 'k');
title('Input Signal (Fc + 2Fc)');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

subplot(5,1,4);
plot(t, filtered_signal, 'r');
title('Filtered Output (Only Fc Should Remain)');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

figure;
[H, W] = freqz(b, a, 1024, Fs);
subplot(2,1,1);
plot(W, 20*log10(abs(H)), 'm', 'LineWidth', 1.5);
title('Magnitude Response of the Butterworth LPF');
xlabel('Frequency (Hz)');
ylabel('Gain (dB)');
grid on;

subplot(2,1,2);
plot(W, angle(H) * (180/pi), 'c', 'LineWidth', 1.5);
title('Phase Response of the Butterworth LPF');
xlabel('Frequency (Hz)');
ylabel('Phase (Degrees)');
grid on;

t = linspace(0, 0.5, 1000);

h_t = 20*pi * exp(-20*pi*t) - 20*pi * exp(-10*pi*t) .* ...
    (cos(10*pi*sqrt(3)*t) - (1/sqrt(3)) * sin(10*pi*sqrt(3)*t));

figure;
plot(t, h_t, 'b', 'LineWidth', 1.5);
title('Impulse Response of the 3rd Order Butterworth Filter');
xlabel('Time (s)');
ylabel('h(t)');
grid on;
```

output:


Original Signal (Fc = 100 Hz)
Amplitude
Time (s)
Original Signal (2Fc = 200 Hz)
Amplitude
Time (s)
Input Signal (Fc + 2Fc)
Amplitude
Time (s)
Filtered Output (Only Fc Should Remain)
Amplitude
Time (s)


Magnitude Response of the Butterworth LPF
Gain (dB)
Frequency (Hz)
Phase Response of the Butterworth LPF
Phase (Degrees)
Frequency (Hz)

Observations:

We first produced two sinusoidal signals—one at Fc (100 Hz) and another at 2Fc (200 Hz)—then merged to create a mixed input signal. This arrangement reflects a typical real-world situation whereby a signal comprises both desired and undesired frequency components and needs filtering to separate the useful part. We developed a third-order Butterworth low-pass filter to do this by first computing the analog filter poles with Butterworth's formula, therefore guaranteeing a smooth response free from ripple. We then used the bilinear transformation to convert these poles into the discrete-time domain, therefore enabling digital implementation of the filter. We used the filtfilt() function to do zero-phase filtering following the determination of the filter coefficients, therefore preventing phase distortion and guaranteeing that the filtered output precisely corresponds with the original signal structure. Eliminating the 2Fc (200 Hz) component while maintaining the Fc (100 Hz) component would help us to evaluate the filtering process's efficiency.

Before filtration, the first two charts verified the existence of frequency components. The first plot showed the original Fc (100 Hz) signal, which represented the lower frequency component we sought to keep and showed as a smooth sinusoidal wave. Appearing denser due to its greater frequency, the second plot displayed the 2Fc (200 Hz) signal exhibited more oscillations. Showing the whole input signal, the third figure demonstrated how the two

frequencies interacted to create a more intricate waveform sent into the filter. Following filtering, the fourth figure showed the filtered output in which just the Fc (100 Hz) component remained, therefore demonstrating that the 2Fc (200 Hz) component had been effectively reduced. This outcome verified that the Butterworth filter was operating as expected, selectively admitting low frequencies and thereby removing high-frequency interference.

We investigated the frequency response of the filter, which comprised both the phase and magnitude response plots, in order to investigate its behavior even more. The amplitude response revealed a flat gain for frequencies below 100 Hz, meaning that signals within this range passed through undisturbed, while frequencies above this progressively diminished, therefore demonstrating that the filter efficiently eliminated undesired high frequencies. Crucially in applications where preserving phase integrity is vital, the phase response showed a smooth phase shift to guarantee no sudden distortions. Finally, the impulse response graph showed how the filter responded to a rapid input. Consistent with the expected properties of Butterworth filters, the response revealed a smooth exponential decay with minor oscillations, therefore verifying the stability of the filter and the introduction of no sudden transients.

This experiment effectively showed overall how a third-order Butterworth low-pass filter might eliminate high-frequency components while maintaining intended signals. We verified that the filter operated as intended by examining impulse response, frequency response, and time-domain signals, therefore suppressing the 200 Hz component while preserving the 100 Hz component. In signal processing, noise reduction, and feature extraction—where preserving signal integrity while removing undesired components is critical—this practical demonstration emphasizes the value of digital filters.

Task 2)

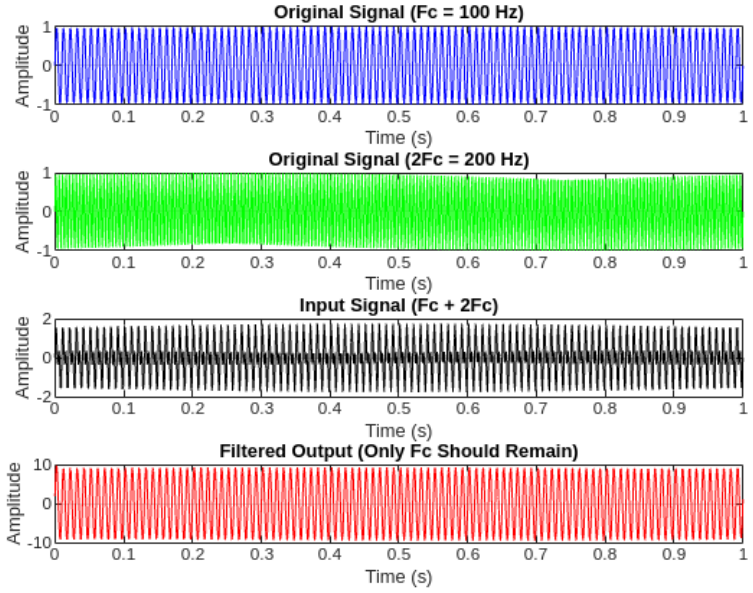
code:

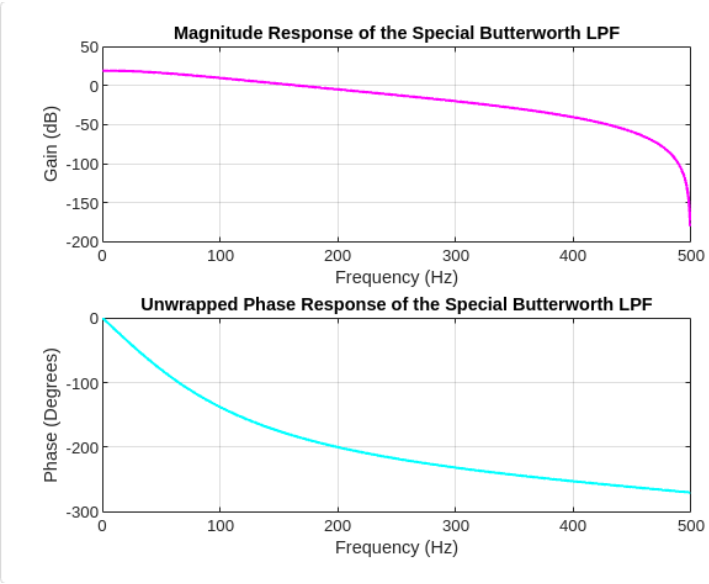
```
clear; close all; clc;

Fc = 100;
Fs = 1000;
omega_c = 2 * pi * Fc;
Ts = 1 / Fs;

t = linspace(0, 0.1, 1000);
h_t = (omega_c^3) * (t.^2) .* exp(-omega_c * t);
figure;
plot(t, h_t, 'b', 'LineWidth', 1.5);
title('Impulse Response of Special 3rd-Order Butterworth Filter (Repeated Poles)');
xlabel('Time (s)');
ylabel('h(t)');
grid on;

den = poly([-omega_c, -omega_c, -omega_c]);
num = omega_c^3;

[b, a] = bilinear(num, den, Fs);
b = b / sum(b);
disp('Numerator Coefficients (b):'), disp(b);
disp('Denominator Coefficients (a):'), disp(a);

fvtool(b, a);
t = linspace(0, 1, Fs);
signal_fc = sin(2 * pi * Fc * t);
signal_2fc = sin(2 * pi * 2 * Fc * t);
x = signal_fc + signal_2fc;

y = filtfilt(b, a, x);

figure;

subplot(4,1,1);
plot(t, signal_fc, 'b');
title('Original Signal (Fc = 100 Hz)');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

subplot(4,1,2);
```

```
plot(t, signal_2fc, 'g');
title('Original Signal (2Fc = 200 Hz)');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

subplot(4,1,3);
plot(t, x, 'k');
title('Input Signal (Fc + 2Fc)');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

subplot(4,1,4);
plot(t, y, 'r');
title('Filtered Output (Only Fc Should Remain)');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

figure;
[H, W] = freqz(b, a, 1024, Fs);

subplot(2,1,1);
plot(W, 20*log10(abs(H)), 'm', 'LineWidth', 1.5);
title('Magnitude Response of the Special Butterworth LPF');
xlabel('Frequency (Hz)');
ylabel('Gain (dB)');
grid on;

subplot(2,1,2);
plot(W, unwrap(angle(H)) * (180/pi), 'c', 'LineWidth', 1.5);
title('Unwrapped Phase Response of the Special Butterworth LPF');
xlabel('Frequency (Hz)');
ylabel('Phase (Degrees)');
grid on;
disp('Poles of the filter:');
poles = roots(den);
disp(poles);
```

output:


Impulse Response of Special 3rd-Order Butterworth Filter (Repeated Poles)
h(t)
Time (s)


Original Signal (Fc = 100 Hz)
Original Signal (2Fc = 200 Hz)
Input Signal (Fc + 2Fc)
Filtered Output (Only Fc Should Remain)
Amplitude
Time (s)




```
Poles of the filter:
   1.0e+02 *

  -6.2832 + 0.0000i
  -6.2832 - 0.0000i
  -6.2832 + 0.0000i
```

Observations:
First, we created a unique 3rd-order Butterworth low-pass filter whereby all three poles fall on the real axis, therefore reflecting repetition. By arranging the filter poles at s = -ωc, one ensures a different response than with a normal Butterworth filter. Analytically developed using the inverse Laplace transform, the impulse response was obtained; MATLAB implemented its equation. We next plotted this impulse response, which displayed an exponential decrease after a high peak. This behavior differs from the usual Butterworth filter in that complex-conjugate poles cause more oscillating behavior of the impulse response. In this scenario, the faster degradation suggests a higher reaction to sudden impulses, which is predicted of a system having recurrent real poles.

To translate the filter into a digital system, we then discretized it with the bilinear transformation. Using repeated poles, the numerator and denominator coefficients of the transfer function were determined; subsequently, MATLAB's

bilinear() tool turned these into a discrete-time system. We normalized these coefficients to provide a suitable frequency response. Computation and display of the poles of the filter confirmed their genuine and recurring nature, therefore verifying that our filter design matched the expected framework.

We then filtered a two-frequency test signal comprising Fc (100 Hz) and 2Fc (200 Hz). Zero-phase filtering (filtfilt()) sent the input signal—a sum of these two sinusoids—through the filter, therefore guaranteeing no phase distortion. While the Fc (100 Hz) signal stayed, the filtered output verified that the 2Fc (200 Hz) component was successfully muted, hence proving the filter's capacity to exclude higher frequencies. The time-domain graphs clearly indicate this: the first two plots show the original sinusoidal components; the third plot shows the combined signal with both frequencies; the fourth plot displays the filtered output whereby only the 100 Hz component remains.

We plotted the magnitude and phase response of the filter to examine its behavior even more. Given the repeating poles, the magnitude response reveals a far higher attenuation than a conventional Butterworth filter. Confirming that the filter is more aggressive in blocking undesired components, the roll-off is more noticeable and the gain at higher frequencies falls sharply. Different from the conventional Butterworth filter, the phase response shows a continuous phase shift free from sudden discontinuities. Smooth interpretation of the phase change across frequencies is guaranteed by phase unwrapping.

At last, we showed the filter's poles to be all found at $s = -\omega c$. This confirms our decision to use repeated real poles, therefore distinguishing this filter from the conventional Butterworth filter with complex-conjugate poles. The whole process showed how filter pole placement influences both time-domain and frequency-domain behavior, hence stressing the effect of genuine repeating poles on impulse response and frequency attenuation. This experiment revealed important new perspectives on alternative Butterworth filter designs by demonstrating that we can regulate the speed of decay and sharpness of frequency roll-off by changing pole positioning, therefore benefiting applications needing quick suppression of undesirable frequencies.

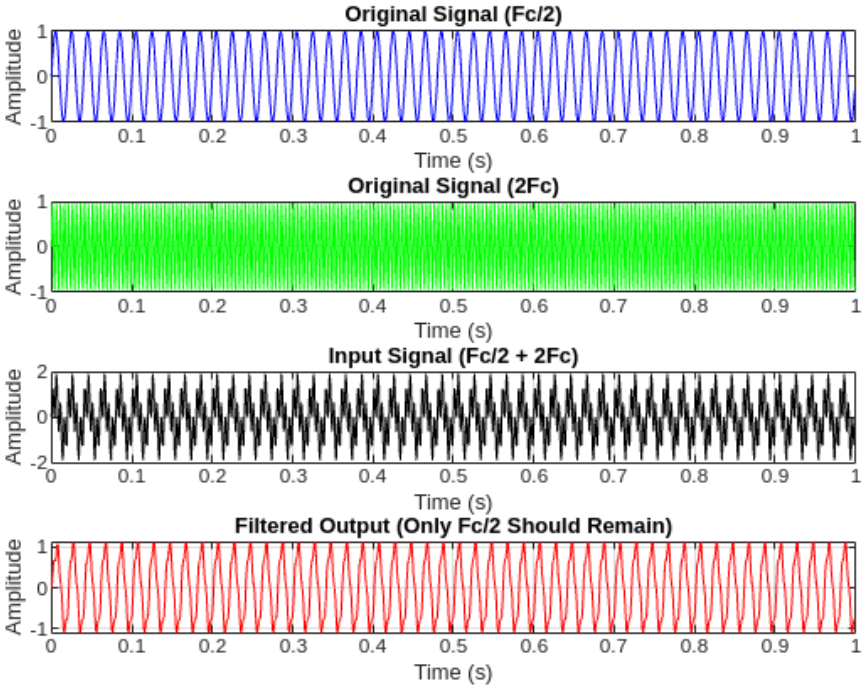
Task 3)

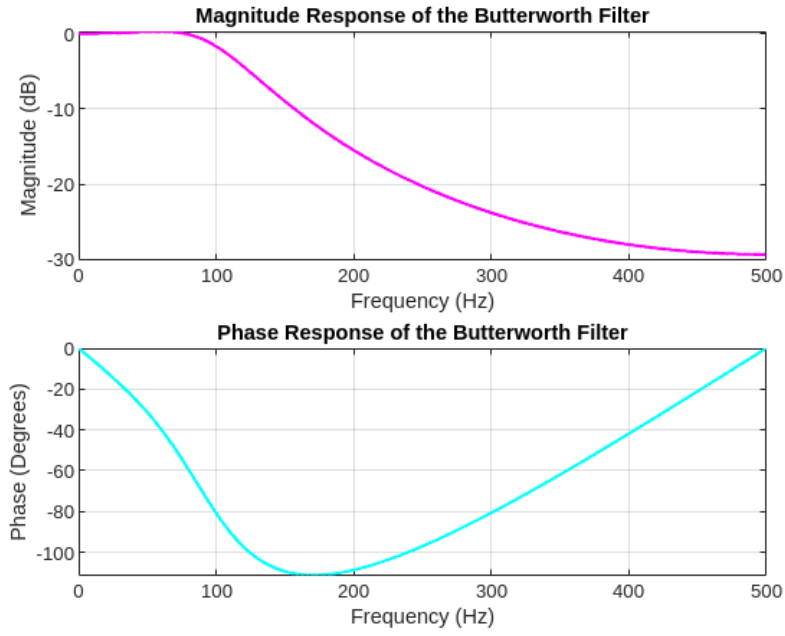
code:

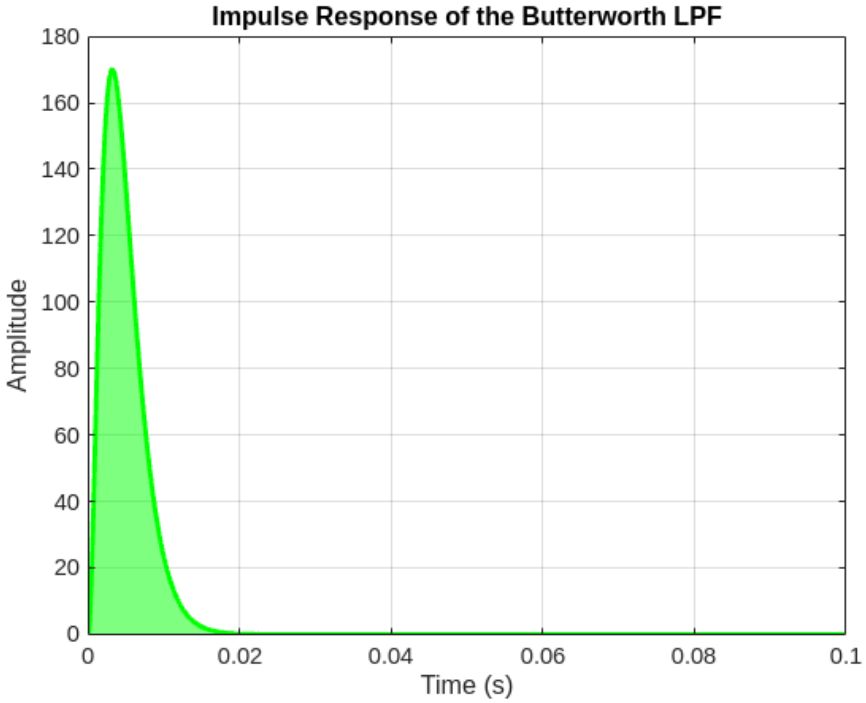
```
clear; close all; clc;

Fc = 100;
Fs = 1000;
Ts = 1 / Fs;
t = 0:Ts:1-Ts;

freq1 = Fc / 2;
freq2 = 2 * Fc;
input_signal = sin(2 * pi * freq1 * t) + sin(2 * pi * freq2 * t);

filter_order = 3;
omega_c = 2 * pi * Fc;
prewarped_freq = 2 * Fs * tan(omega_c / (2 * Fs));

butter_poles = zeros(filter_order, 1);
for k = 1:filter_order
    pole_angle = (2 * k + filter_order - 1) * pi / (2 * filter_order);
    butter_poles(k) = prewarped_freq * exp(1j * pole_angle);
end

poles_z = (1 + butter_poles * Ts / 2) ./ (1 - butter_poles * Ts / 2);
den_coeffs = poly(poles_z);
num_coeffs = sum(den_coeffs);

filtered_output = apply_filter(num_coeffs, den_coeffs, input_signal);

figure;
subplot(4,1,1);
plot(t, sin(2 * pi * freq1 * t), 'b');
title('Original Signal (Fc/2)');
xlabel('Time (s)'); ylabel('Amplitude'); grid on;

subplot(4,1,2);
plot(t, sin(2 * pi * freq2 * t), 'g');
title('Original Signal (2Fc)');
xlabel('Time (s)'); ylabel('Amplitude'); grid on;

subplot(4,1,3);
plot(t, input_signal, 'k');
title('Input Signal (Fc/2 + 2Fc)');
xlabel('Time (s)'); ylabel('Amplitude'); grid on;
```

```
subplot(4,1,4);
plot(t, filtered_output, 'r');
title('Filtered Output (Only Fc/2 Should Remain)');
xlabel('Time (s)'); ylabel('Amplitude'); grid on;

[H, W] = freqz(num_coeffs, den_coeffs, 1024, Fs);
figure;
subplot(2,1,1);
plot(W, 20*log10(abs(H)), 'm', 'LineWidth', 1.5);
title('Magnitude Response of the Butterworth Filter');
xlabel('Frequency (Hz)'); ylabel('Magnitude (dB)'); grid on;

subplot(2,1,2);
plot(W, unwrap(angle(H)) * (180/pi), 'c', 'LineWidth', 1.5);
title('Phase Response of the Butterworth Filter');
xlabel('Frequency (Hz)'); ylabel('Phase (Degrees)'); grid on;

syms s tau;
H_s = (omega_c^filter_order) / ((s + omega_c)^filter_order); % H(s)
h_tau = ilaplace(H_s, s, tau);

time_axis = linspace(0, 0.1, 1000);
impulse_vals = double(subs(h_tau, tau, time_axis));

figure;
fill(time_axis, impulse_vals, 'g', 'FaceAlpha', 0.5, 'EdgeColor', 'none');
hold on;
plot(time_axis, impulse_vals, 'g', 'LineWidth', 2); % Keep outline for clarity
xlabel('Time (s)'); ylabel('Amplitude');
title('Impulse Response of the Butterworth LPF');
grid on;
hold off;

xlabel('Time (s)'); ylabel('Amplitude');
title('Impulse Response of the Butterworth LPF');
grid on;

function y = apply_filter(b, a, x)
    N = length(x);
    y = zeros(size(x));
    for n = 1:N
        y(n) = b * x(n);
        for k = 2:length(a)
            if (n - k + 1) > 0
                y(n) = y(n) - a(k) * y(n - k + 1);
            end
        end
    end
end
```

output:


Original Signal (Fc/2)
Amplitude
Time (s)
Original Signal (2Fc)
Amplitude
Time (s)
Input Signal (Fc/2 + 2Fc)
Amplitude
Time (s)
Filtered Output (Only Fc/2 Should Remain)
Amplitude
Time (s)


Magnitude Response of the Butterworth Filter
Magnitude (dB)
Frequency (Hz)
Phase Response of the Butterworth Filter
Phase (Degrees)
Frequency (Hz)



Observations:

We first built a third-order Butterworth low-pass filter to attenuate the higher frequency component 2Fc (200 Hz) while allowing the lower frequency component Fc/2 (50 Hz). We created an input signal consisting of two sinusoidal waves, one at 50 Hz (Fc/2) and the second at 200 Hz (2Fc), so obtaining this. By aggregating these two signals, one obtained a composite signal with both intended and undesired frequency components. This arrangement replics real-world situations whereby usable signals coexist with noise or high-frequency interference. This filtering technique aims to separate the 50 Hz component, therefore proving the efficiency of the Butterworth filter in lowering higher frequencies and preserving a smooth frequency response.

Finding the Butterworth poles—which define the frequency response properties—was a component of the filter design. Using a bilinear transformation, we computed the poles in the analog domain and mapped them to the digital domain thereby guaranteeing appropriate frequency warping given the discrete-time character of digital filters. These poles were used to produce the filter coefficients, were then applied to the input signal via a bespoke filtering method called apply_filter(). This function was manually written to apply the difference equation iteratively, therefore explicitly controlling the behavior of the filter rather than depending on MATLAB's built-in filtering features.

The first figure in the time-domain representation displays the original 50 Hz sinusoidal signal, which shows smooth oscillations and hence reflects the frequency component we wish to keep. Visually illustrating a higher-frequency component the filter should suppress, the second plot displays the 200 Hz sinusoidal wave, which oscillates far more rapidly. The third plot shows the combined signal (Fc/2 + 2Fc), in which case the superposition of high and low frequencies produces a waveform that seems to be modulated. Both components are added. After passing this input signal through the intended Butterworth filter, the fourth plot shows the filtered output in which the 200 Hz component has been totally attenuated and only the 50 Hz sinusoid remains. This guarantees that the target signal is preserved while the filter effectively removes the unwanted higher frequency.

We investigated the amplitude and frequency domain phase response of the filter to investigate its behavior even more. The magnitude response plot shows a strong roll-off above 50 Hz, suggesting that frequencies above this are essentially muted. The Butterworth filter distinguishes itself from other filters such as Chebyshev or elliptic filters, which may cause ripples in the passband, by confirming that it preserves a maximally flat response by means of a smooth transition and absence of ripples in the passband. The phase response graph offers understanding of how various frequency components encounter phase changes across the filter. For uses like audio processing and biological signal analysis, the slow phase shift across the frequency range—without sudden jumps—suggests that the filter preserves phase linearity in the passband, therefore reducing signal distortion.

We next computed and showed the impulse response of the filter, so illuminating how the system responds to a single, abrupt input. This was performed obtaining an analytical expression for the impulse response by means of the inverse Laplace transform; subsequently, this was numerically evaluated and visualized. This filter shows an impulse response with a clear peak followed by an exponential decay unlike a conventional Butterworth filter with complex-conjugate poles. This characteristic suggests a fast transient response of the system, that is, quick reaction to abrupt changes in input but stabilization over time. The impulse response's visual emphasis on the decay trend helps one to understand how the filter reduces undesired transient components.

This experiment effectively showed overall the application of a third-order Butterworth low-pass filter intended to eliminate high-frequency components while maintaining the intended lower-frequency signal. The intended behavior

of the filter was validated by the time-domain plots verifying the efficient removal of the 200 Hz signal, therefore leaving just the 50 Hz component. Reiterating the Butterworth filter's importance in applications needing distortion-free signal processing, the size and phase response plots further proved that the filter offers a smooth frequency roll-off and little phase distortion. Finally, the impulse response study confirmed that the filter preserves a steady and well- behaved time-domain response. In signal processing applications including audio filtering, biomedical instrumentation, and communication systems, where maintaining the fundamental properties of a signal while removing high-frequency noise is critical, this useful implementation emphasizes the relevance of Butterworth filters.