National Institute of Technology Calicut

Department of Electronics and Communication Engineering

Digital Signal Processing LAB

Week 8 Report

Date: 16-10–2025
Instructor: Dr. Sakthivel V

Submitted by:

Mummana Jagadeesh	B231113EC
Muhammed Nihal MP	B230437EC

Aim

To design and implement High-Pass, Band-Pass, and Band-Stop filters using normalized Low-Pass Filters (Butterworth and Chebyshev) in MATLAB, employing Bilinear Transformation (BLT) and Impulse Invariant Transformation (IIT) methods, and to compare their characteristics

Theory

IIR Filters

Infinite Impulse Response (IIR) filters are digital filters whose impulse response theoretically extends to infinity. They are widely employed because they can achieve a sharp frequency response using a lower filter order compared to FIR filters.

The general design procedure for IIR filters involves:

1. Starting with an analog low-pass prototype filter that satisfies the given specifications.
2. Applying a suitable frequency transformation to obtain the desired analog filter type (e.g., low-pass, high-pass, band-pass, or band-stop).
3. Converting the analog filter into a digital filter using transformations such as the *Impulse Invariant Transformation (IIT)* or the *Bilinear Transformation (BLT)*.

Analog Prototypes for IIR Filters

Two standard analog prototype filters commonly used in IIR filter design are:

- The **Butterworth filter**
- The **Chebyshev (Type-I) filter**

Butterworth Filter (in terms of δ_p and δ_s)

The squared magnitude response of a Butterworth filter is given by:

$$|H(j\Omega)|^2 = \frac{1}{1 + \left(\frac{\Omega}{\Omega_c}\right)^{2N}}$$

If the passband and stopband magnitude constraints are defined as:

$$|H(j\Omega_p)| \geq \delta_p, \quad |H(j\Omega_s)| \leq \delta_s,$$

then at the passband and stopband edges:

$$\left(\frac{\Omega_p}{\Omega_c}\right)^{2N} = \frac{1}{\delta_p^2} - 1, \quad \left(\frac{\Omega_s}{\Omega_c}\right)^{2N} = \frac{1}{\delta_s^2} - 1$$

Eliminating Ω_c gives the required filter order N:

$$N \geq \frac{\log_{10}\left(\frac{1/\delta_s^2-1}{1/\delta_p^2-1}\right)}{2\log_{10}\left(\frac{\Omega_s}{\Omega_p}\right)}$$

Once N is determined, the analog cutoff frequency Ω_c can be computed using either edge condition:

$$\Omega_c = \Omega_p\left(\frac{1}{\delta_p^2}-1\right)^{-1/(2N)} = \Omega_s\left(\frac{1}{\delta_s^2}-1\right)^{-1/(2N)}$$

(Using either expression yields the same Ω_c when N satisfies both specifications.)

Conversion to dB

If the specifications are given in decibels (dB) — for example, passband ripple A_p and stopband attenuation A_s — the corresponding linear values are:

$$\delta_p = 10^{-A_p/20}, \quad \delta_s = 10^{-A_s/20}$$

Thus,

$$\frac{1}{\delta_s^2} - 1 = 10^{0.1A_s} - 1$$

This allows the order formula to be expressed conveniently in terms of A_p and A_s.

Chebyshev Filters (Type-I)

The Type-I Chebyshev filter allows ripples in the passband, achieving a sharper transition between the passband and stopband compared to the Butterworth filter.

The squared magnitude response is:

$$|H(j\Omega)|^2 = \frac{1}{1+\epsilon^2 C_N^2\left(\frac{\Omega}{\Omega_c}\right)}$$

where $C_N(x)$ denotes the Chebyshev polynomial of order N, and ϵ controls the passband ripple.

At the band edges:

$$C_N\left(\frac{\Omega_p}{\Omega_c}\right) = \frac{1}{\epsilon}\sqrt{\frac{1}{\delta_p^2}-1}, \quad C_N\left(\frac{\Omega_s}{\Omega_c}\right) = \frac{1}{\epsilon}\sqrt{\frac{1}{\delta_s^2}-1}$$

The ripple factor ϵ is defined by:

$$\epsilon = \sqrt{\frac{1}{\delta_p^2}-1}$$

The required order N is:

$$N \geq \frac{\cosh^{-1}\left(\sqrt{\frac{1/\delta_s^2-1}{1/\delta_p^2-1}}\right)}{\cosh^{-1}\left(\frac{\Omega_s}{\Omega_p}\right)}$$

The cutoff frequency Ω_c is computed using:

$$\Omega_c = \frac{\Omega_p}{\cosh\left(\frac{1}{N}\cosh^{-1}\left(\frac{1}{\epsilon}\right)\right)}$$

Conversion to dB Specifications

For specifications in dB:

$$\delta_p = 10^{-A_p/20}, \quad \delta_s = 10^{-A_s/20}$$

Then:

$$\epsilon = \sqrt{10^{0.1A_p} - 1} \quad \text{(Type-I)}, \quad \epsilon = \sqrt{10^{0.1A_s} - 1} \quad \text{(Type-II)}$$

Bilinear Transformation (BLT)

The Bilinear Transformation is a commonly used technique for converting analog filters to digital form. It maps the entire $j\Omega$-axis of the analog domain onto the unit circle in the digital domain, maintaining stability and avoiding aliasing.

- **Mapping Relation:**

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

  This ensures:

  - The entire $j\Omega$-axis maps onto the unit circle ($|z| = 1$).
  - Stable analog filters map to stable digital filters.

- **Properties:**

  - Eliminates aliasing by providing a one-to-one frequency mapping.
  - Introduces frequency warping due to the nonlinear relation between analog and digital frequencies.
  - Frequency warping can be compensated by pre-warping.

- **Pre-warping:** To preserve the desired critical frequency (e.g., cutoff frequency), the pre-warped analog frequency is:

  $$\Omega_c = \frac{2}{T} \tan\left(\frac{\omega_c T}{2}\right)$$

- **Digital Transfer Function:** For an analog transfer function $H_a(s)$, the digital version is obtained by substituting the bilinear mapping:

  $$H_d(z) = H_a\left(\frac{2}{T} \cdot \frac{1 - z^{-1}}{1 + z^{-1}}\right)$$

Summary: The Bilinear Transformation avoids aliasing and guarantees stability but introduces frequency warping. By applying pre-warping, the analog prototype's critical frequencies can be accurately matched in the digital filter.

MATLAB Code

HIGH PASS FILTER BUTTERWORTH BLT

```
clc; clear;

omega_p = input('Enter passband edge frequency (rad/s): ');
omega_s = input('Enter stopband edge frequency (rad/s): ');
A_p = input('Enter passband ripple (in dB): ');
A_s = input('Enter stopband attenuation (in dB): ');
fs = input('Enter sampling frequency (in Hz): ');

T = 1 / fs;

omega_p = (2 / T) * tan(omega_p * T / 2);
omega_s = (2 / T) * tan(omega_s * T / 2);

temp = omega_p;
omega_p = omega_s;
omega_s = temp;

del_p = 10^(-A_p / 20);
del_s = 10^(-A_s / 20);

a = (1 / del_s^2) - 1;
b = (1 / del_p^2) - 1;
c = log10(omega_s / omega_p);
x = log10(sqrt(a / b)) / c;

N = 0;
while N < x
    N = N + 1;
end

omega_c = 1;

disp(['Analog Butterworth order N = ', num2str(N)]);
disp(['Analog cutoff frequency omega_c = ', num2str(omega_c
   )]);

w = linspace(0, 2 * omega_s, 500);
Ha = zeros(size(w));

temp = N;
while temp >= 2
    temp = temp - 2;
end
```

```
if temp == 0
    N_odd = false;
else
    N_odd = true;
end

if N_odd
    for idx = 1:length(w)
        s = 1j * w(idx);
        s = omega_s / s;
        Hprod = 1;
        for k = 1:(N - 1) / 2
            bk = 2 * sin((2 * k - 1) * pi / (2 * N));
            Hprod = Hprod * (s^2 + bk * omega_c * s +
               omega_c^2);
        end
        Ha(idx) = abs((omega_c^N) / ((s + omega_c) * Hprod)
           );
    end
else
    for idx = 1:length(w)
        s = 1j * w(idx);
        s = omega_s / s;
        Hprod = 1;
        for k = 1:(N) / 2
            bk = 2 * sin((2 * k - 1) * pi / (2 * N));
            Hprod = Hprod * (s^2 + bk * omega_c * s +
               omega_c^2);
        end
        Ha(idx) = abs((omega_c^N) / Hprod);
    end
end

Nfft = 1024;
w_d = linspace(0, pi, Nfft);
Hd_blt = zeros(size(w_d));

for idx = 1:length(w_d)
    z = exp(1j * w_d(idx));
    s = (2 / T) * (1 - z^-1) / (1 + z^-1);
    s = omega_s / s;
    Hprod = 1;
    for k = 1:N
        theta_k = (2 * k + N - 1) * pi / (2 * N);
        ck = omega_c * exp(1j * theta_k);
        Hprod = Hprod * (omega_c / (s - ck));
    end
    Hd_blt(idx) = Hprod;
end

figure;
subplot(2, 1, 1);
plot(w, 20 * log10(Ha), 'LineWidth', 1.5);
xlabel('\omega (rad/s)');
ylabel('Magnitude (dB)');
title('Analog Butterworth LPF (Product Form)');
grid on;
ylim([-60 5]);

subplot(2, 1, 2);
f_Hz = w_d * fs / (2 * pi);
plot(f_Hz, 20 * log10(abs(Hd_blt)), 'LineWidth', 1.5);
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
title('Digital Butterworth LPF (Bilinear Transform)');
grid on;
ylim([-60 5]);
```

HIGH PASS FILTER CHEBYSHEV BLT

```
clc; clear;

omega_p = input('Enter passband edge frequency (rad/s): ');
omega_s = input('Enter stopband edge frequency (rad/s): ');
A_p = input('Enter passband ripple (dB): ');
A_s = input('Enter stopband attenuation (dB): ');
fs = input('Enter sampling frequency (Hz): ');

T = 1 / fs;

omega_p = (2 / T) * tan(omega_p * T / 2);
omega_s = (2 / T) * tan(omega_s * T / 2);

temp = omega_p;
omega_p = omega_s;
omega_s = temp;

del_p = 10^(-A_p / 20);
del_s = 10^(-A_s / 20);

a = (1 / del_s^2) - 1;
b = (1 / del_p^2) - 1;
eps = sqrt(b);

c = acosh(omega_s / omega_p);
x = acosh(sqrt(a / b)) / c;

N = 0;
while N < x
    N = N + 1;
end

disp(['Analog Chebyshev Filter order N = ', num2str(N)]);

temp = N;
while temp >= 2
    temp = temp - 2;
end

if temp == 0
    N_odd = false;
else
    N_odd = true;
end
```

```
omega_p = 1;
yN = ((sqrt(1 + 1 / eps^2) + 1 / eps)^(1 / N) - (sqrt(1 + 1
    / eps^2) + 1 / eps)^(-1 / N)) / 2;

w = linspace(0, 2 * omega_s, 500);
Ha = zeros(size(w));

if N_odd
    for idx = 1:length(w)
        s = 1j * w(idx);
        s = omega_s / s;
        H = (omega_p^N * yN) / (s + omega_p * yN);
        num = 1; deno = 1;
        for k = 1:(N - 1) / 2
            ck = yN^2 + (cos((2 * k - 1) * pi / (2 * N)))
                ^2;
            bk = 2 * yN * sin((2 * k - 1) * pi / (2 * N));
            num = num * ck;
            deno = deno * (s^2 + bk * omega_p * s + ck *
                omega_p^2);
        end
        Ha(idx) = abs(H * num / deno);
    end
else
    for idx = 1:length(w)
        s = 1j * w(idx);
        s = omega_s / s;
        H = omega_p^N / sqrt(1 + eps^2);
        num = 1; deno = 1;
        for k = 1:(N / 2)
            ck = yN^2 + (cos((2 * k - 1) * pi / (2 * N)))
                ^2;
            bk = 2 * yN * sin((2 * k - 1) * pi / (2 * N));
            num = num * ck;
            deno = deno * (s^2 + bk * omega_p * s + ck *
                omega_p^2);
        end
        Ha(idx) = abs(H * num / deno);
    end
end

Nfft = 1024;
w_d = linspace(0, pi, Nfft);
Hd_blt = zeros(size(w_d));

for idx = 1:length(w_d)
```

```
        z = exp(1j * w_d(idx));
        s = (2 / T) * (1 - z^-1) / (1 + z^-1);
        s = omega_s / s;
        if N_odd
            H = (omega_p^N * yN) / (s + omega_p * yN);
            num = 1; deno = 1;
            for k = 1:(N - 1) / 2
                ck = yN^2 + (cos((2 * k - 1) * pi / (2 * N)))
                   ^2;
                bk = 2 * yN * sin((2 * k - 1) * pi / (2 * N));
                num = num * ck;
                deno = deno * (s^2 + bk * omega_p * s + ck *
                   omega_p^2);
            end
            Hd_blt(idx) = H * num / deno;
        else
            H = omega_p^N / sqrt(1 + eps^2);
            num = 1; deno = 1;
            for k = 1:(N / 2)
                ck = yN^2 + (cos((2 * k - 1) * pi / (2 * N)))
                   ^2;
                bk = 2 * yN * sin((2 * k - 1) * pi / (2 * N));
                num = num * ck;
                deno = deno * (s^2 + bk * omega_p * s + ck *
                   omega_p^2);
            end
            Hd_blt(idx) = H * num / deno;
        end
    end

    figure;

    subplot(2, 1, 1);
    plot(w, 20 * log10(Ha), 'LineWidth', 1.5);
    xlabel('\omega (rad/s)');
    ylabel('Magnitude (dB)');
    title('Analog Chebyshev Type-I HPF');
    grid on;
    ylim([-60 5]);

    subplot(2, 1, 2);
    f_Hz = w_d * fs / (2 * pi);
    plot(f_Hz, 20 * log10(abs(Hd_blt)), 'LineWidth', 1.5);
    xlabel('Frequency (Hz)');
    ylabel('Magnitude (dB)');
    title('Digital Chebyshev Type-I HPF (Bilinear Transform)');
```

```
grid on;
ylim([-60 5]);
```

BANDPASS BUTTERWORTH BLT

```
clc; clear;

omega_p1 = input('Enter first passband edge frequency (rad/
    s): ');
omega_p2 = input('Enter second passband edge frequency (rad
    /s): ');
omega_s1 = input('Enter first stopband edge frequency (rad/
    s): ');
omega_s2 = input('Enter second stopband edge frequency (rad
    /s): ');
A_p = input('Enter passband ripple (in dB): ');
A_s = input('Enter stopband attenuation (in dB): ');
fs = input('Enter sampling frequency (in Hz): ');

T = 1 / fs;

omega_p1 = (2 / T) * tan(omega_p1 * T / 2);
omega_p2 = (2 / T) * tan(omega_p2 * T / 2);
omega_s1 = (2 / T) * tan(omega_s1 * T / 2);
omega_s2 = (2 / T) * tan(omega_s2 * T / 2);

pepdt = omega_p1 * omega_p2;
sepdt = omega_s1 * omega_s2;

if pepdt > sepdt
    omega_s1 = pepdt / omega_s2;
elseif pepdt < sepdt
    omega_s2 = pepdt / omega_s1;
end

omega_p = 1;
omega_s = (omega_s2 - omega_s1) / (omega_p2 - omega_p1);
BW = omega_p2 - omega_p1;

del_p = 10^(-A_p / 20);
del_s = 10^(-A_s / 20);

a = (1 / del_s^2) - 1;
b = (1 / del_p^2) - 1;
c = log10(omega_s / omega_p);
x = log10(sqrt(a / b)) / c;

N = 0;
while N < x
    N = N + 1;
```

```
end

omega_c = 1;

disp(['Analog Butterworth order N = ', num2str(N)]);
disp(['Analog cutoff frequency omega_c = ', num2str(omega_c
   )]);

w = linspace(omega_p1 / 2, omega_s2 * 1.2, 1000);
Ha = zeros(size(w));

temp = N;
while temp >= 2
    temp = temp - 2;
end

if temp == 0
    N_odd = false;
else
    N_odd = true;
end

if N_odd
    for idx = 1:length(w)
        s1 = 1j * w(idx);
        s = (s1 * s1 + pepdt) / (BW * s1);
        Hprod = 1;
        for k = 1:(N - 1) / 2
            bk = 2 * sin((2 * k - 1) * pi / (2 * N));
            Hprod = Hprod * (s^2 + bk * omega_c * s +
               omega_c^2);
        end
        Ha(idx) = abs((omega_c^N) / ((s + omega_c) * Hprod)
           );
    end
else
    for idx = 1:length(w)
        s1 = 1j * w(idx);
        s = (s1 * s1 + pepdt) / (BW * s1);
        Hprod = 1;
        for k = 1:(N) / 2
            bk = 2 * sin((2 * k - 1) * pi / (2 * N));
            Hprod = Hprod * (s^2 + bk * omega_c * s +
               omega_c^2);
        end
        Ha(idx) = abs((omega_c^N) / Hprod);
    end
end

Nfft = 1024;
w_d = linspace(0, pi, Nfft);
Hd_blt = zeros(size(w_d));

for idx = 1:length(w_d)
    z = exp(1j * w_d(idx));
    s1 = (2 / T) * (1 - z^-1) / (1 + z^-1);
    s = (s1 * s1 + pepdt) / (BW * s1);
    Hprod = 1;
    for k = 1:N
        theta_k = (2 * k + N - 1) * pi / (2 * N);
        ck = omega_c * exp(1j * theta_k);
        Hprod = Hprod * (omega_c / (s - ck));
    end
    Hd_blt(idx) = Hprod;
end

figure;
subplot(2, 1, 1);
plot(w, 20 * log10(Ha), 'LineWidth', 1.5);
xlabel('\omega (rad/s)');
ylabel('Magnitude (dB)');
title('Analog Butterworth Bandpass Filter (Product Form)');
grid on;
ylim([-60 5]);

subplot(2, 1, 2);
f_Hz = w_d * fs / (2 * pi);
plot(f_Hz, 20 * log10(abs(Hd_blt)), 'LineWidth', 1.5);
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
title('Digital Butterworth Bandpass Filter (Bilinear
   Transform)');
grid on;
ylim([-60 5]);
```

BANDPASS CHEBYSHEV BLT

```
clc; clear;

omega_p1 = input('Enter first passband edge frequency (rad/
    s): ');
omega_p2 = input('Enter second passband edge frequency (rad
    /s): ');
omega_s1 = input('Enter first stopband edge frequency (rad/
    s): ');
omega_s2 = input('Enter second stopband edge frequency (rad
    /s): ');
A_p = input('Enter passband ripple (in dB): ');
A_s = input('Enter stopband attenuation (in dB): ');
fs = input('Enter sampling frequency (in Hz): ');

T = 1 / fs;

omega_p1 = (2 / T) * tan(omega_p1 * T / 2);
omega_p2 = (2 / T) * tan(omega_p2 * T / 2);
omega_s1 = (2 / T) * tan(omega_s1 * T / 2);
omega_s2 = (2 / T) * tan(omega_s2 * T / 2);

pepdt = omega_p1 * omega_p2;
sepdt = omega_s1 * omega_s2;

if pepdt > sepdt
    omega_s1 = pepdt / omega_s2;
elseif pepdt < sepdt
    omega_s2 = pepdt / omega_s1;
end

omega_p = 1;
omega_s = (omega_s2 - omega_s1) / (omega_p2 - omega_p1);
BW = omega_p2 - omega_p1;

del_p = 10^(-A_p / 20);
del_s = 10^(-A_s / 20);

a = (1 / del_s^2) - 1;
b = (1 / del_p^2) - 1;
eps = sqrt(b);

c = acosh(omega_s / omega_p);
x = acosh(sqrt(a / b)) / c;

N = 0;
```

```
while N < x
    N = N + 1;
end

disp(['Analog Chebyshev Filter order N = ', num2str(N)]);

temp = N;
while temp >= 2
    temp = temp - 2;
end

if temp == 0
    N_odd = false;
else
    N_odd = true;
end

yN = ((sqrt(1 + 1 / eps^2) + 1 / eps)^(1 / N) - (sqrt(1 + 1
    / eps^2) + 1 / eps)^(-1 / N)) / 2;

w = linspace(omega_p1 / 2, omega_s2 * 1.2, 1000);
Ha = zeros(size(w));

if N_odd
    for idx = 1:length(w)
        s1 = 1j * w(idx);
        s = (s1 * s1 + pepdt) / (BW * s1);
        H = (omega_p^N * yN) / (s + omega_p * yN);
        num = 1; deno = 1;
        for k = 1:(N - 1) / 2
            ck = yN^2 + (cos((2 * k - 1) * pi / (2 * N)))
               ^2;
            bk = 2 * yN * sin((2 * k - 1) * pi / (2 * N));
            num = num * ck;
            deno = deno * (s^2 + bk * omega_p * s + ck *
               omega_p^2);
        end
        Ha(idx) = abs(H * num / deno);
    end
else
    for idx = 1:length(w)
        s1 = 1j * w(idx);
        s = (s1 * s1 + pepdt) / (BW * s1);
        H = omega_p^N / sqrt(1 + eps^2);
        num = 1; deno = 1;
        for k = 1:(N / 2)
```

```
            ck = yN^2 + (cos((2 * k - 1) * pi / (2 * N)))
               ^2;
            bk = 2 * yN * sin((2 * k - 1) * pi / (2 * N));
            num = num * ck;
            deno = deno * (s^2 + bk * omega_p * s + ck *
               omega_p^2);
        end
        Ha(idx) = abs(H * num / deno);
    end
end

Nfft = 1024;
w_d = linspace(0, pi, Nfft);
Hd_blt = zeros(size(w_d));

for idx = 1:length(w_d)
    z = exp(1j * w_d(idx));
    s1 = (2 / T) * (1 - z^-1) / (1 + z^-1);
    s = (s1 * s1 + pepdt) / (BW * s1);
    if N_odd
        H = (omega_p^N * yN) / (s + omega_p * yN);
        num = 1; deno = 1;
        for k = 1:(N - 1) / 2
            ck = yN^2 + (cos((2 * k - 1) * pi / (2 * N)))
               ^2;
            bk = 2 * yN * sin((2 * k - 1) * pi / (2 * N));
            num = num * ck;
            deno = deno * (s^2 + bk * omega_p * s + ck *
               omega_p^2);
        end
        Hd_blt(idx) = H * num / deno;
    else
        H = omega_p^N / sqrt(1 + eps^2);
        num = 1; deno = 1;
        for k = 1:(N / 2)
            ck = yN^2 + (cos((2 * k - 1) * pi / (2 * N)))
               ^2;
            bk = 2 * yN * sin((2 * k - 1) * pi / (2 * N));
            num = num * ck;
            deno = deno * (s^2 + bk * omega_p * s + ck *
               omega_p^2);
        end
        Hd_blt(idx) = H * num / deno;
    end
end
```

```
figure;
subplot(2, 1, 1);
plot(w, 20 * log10(Ha), 'LineWidth', 1.5);
xlabel('\omega (rad/s)');
ylabel('Magnitude (dB)');
title('Analog Chebyshev Type-I Bandpass Filter');
grid on;
ylim([-60 5]);

subplot(2, 1, 2);
f_Hz = w_d * fs / (2 * pi);
plot(f_Hz, 20 * log10(abs(Hd_blt)), 'LineWidth', 1.5);
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
title('Digital Chebyshev Type-I Bandpass Filter (Bilinear
   Transform)');
grid on;
ylim([-60 5]);
```

BANDSTOP BUTTERWORTH BLT

```
clc; clear;

omega_p1 = input('Enter first passband edge frequency (rad/
    s): ');
omega_p2 = input('Enter second passband edge frequency (rad
    /s): ');
omega_s1 = input('Enter first stopband edge frequency (rad/
    s): ');
omega_s2 = input('Enter second stopband edge frequency (rad
    /s): ');
A_p = input('Enter passband ripple (in dB): ');
A_s = input('Enter stopband attenuation (in dB): ');
fs = input('Enter sampling frequency (in Hz): ');

T = 1 / fs;

omega_p1 = (2 / T) * tan(omega_p1 * T / 2);
omega_p2 = (2 / T) * tan(omega_p2 * T / 2);
omega_s1 = (2 / T) * tan(omega_s1 * T / 2);
omega_s2 = (2 / T) * tan(omega_s2 * T / 2);

pepdt = omega_p1 * omega_p2;
sepdt = omega_s1 * omega_s2;

if pepdt > sepdt
    omega_s1 = pepdt / omega_s2;
elseif pepdt < sepdt
    omega_s2 = pepdt / omega_s1;
end

omega_p = 1;
omega_s = (omega_p2 - omega_p1) / (omega_s2 - omega_s1);
BW = omega_p2 - omega_p1;

del_p = 10^(-A_p / 20);
del_s = 10^(-A_s / 20);

a = (1 / del_s^2) - 1;
b = (1 / del_p^2) - 1;
c = log10(omega_s / omega_p);
x = log10(sqrt(a / b)) / c;

N = 0;
while N < x
    N = N + 1;
```

```
end

omega_c = 1;

disp(['Analog Butterworth order N = ', num2str(N)]);
disp(['Analog cutoff frequency omega_c = ', num2str(omega_c
    )]);

w = linspace(1, omega_p2 * 1.5, 2000);
Ha = zeros(size(w));

temp = N;
while temp >= 2
    temp = temp - 2;
end

if temp == 0
    N_odd = false;
else
    N_odd = true;
end

if N_odd
    for idx = 1:length(w)
        s1 = 1j * w(idx);
        s = (BW * s1) / (s1^2 + pepdt);
        Hprod = 1;
        for k = 1:(N - 1) / 2
            bk = 2 * sin((2 * k - 1) * pi / (2 * N));
            Hprod = Hprod * (s^2 + bk * omega_c * s +
               omega_c^2);
        end
        Ha(idx) = abs((omega_c^N) / ((s + omega_c) * Hprod)
           );
    end
else
    for idx = 1:length(w)
        s1 = 1j * w(idx);
        s = (BW * s1) / (s1^2 + pepdt);
        Hprod = 1;
        for k = 1:(N / 2)
            bk = 2 * sin((2 * k - 1) * pi / (2 * N));
            Hprod = Hprod * (s^2 + bk * omega_c * s +
               omega_c^2);
        end
        Ha(idx) = abs((omega_c^N) / Hprod);
    end
end

% Bilinear Transform (BLT)
Nfft = 1024;
w_d = linspace(0, pi, Nfft);
Hd_blt = zeros(size(w_d));

for idx = 1:length(w_d)
    z = exp(1j * w_d(idx));
    s1 = (2 / T) * (1 - z^-1) / (1 + z^-1); % Bilinear
       mapping
    s = (BW * s1) / (s1^2 + pepdt);
    Hprod = 1;
    for k = 1:N
        theta_k = (2 * k + N - 1) * pi / (2 * N);
        ck = omega_c * exp(1j * theta_k);
        Hprod = Hprod * (omega_c / (s - ck));
    end
    Hd_blt(idx) = Hprod;
end

figure;

subplot(2, 1, 1);
plot(w, 20 * log10(Ha), 'LineWidth', 1.5);
xlabel('\omega (rad/s)');
ylabel('Magnitude (dB)');
title('Analog Butterworth Bandstop Filter (Product Form)');
grid on;
ylim([-60 5]);

subplot(2, 1, 2);
f_Hz = w_d * fs / (2 * pi);
plot(f_Hz, 20 * log10(abs(Hd_blt)), 'LineWidth', 1.5);
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
title('Digital Butterworth Bandstop Filter (Bilinear
   Transform)');
grid on;
ylim([-60 5]);
```

BANDSTOP CHEBYSHEV BLT

```
clc; clear;

omega_p1 = input('Enter first passband edge frequency (rad/
   s): ');
omega_p2 = input('Enter second passband edge frequency (rad
   /s): ');
omega_s1 = input('Enter first stopband edge frequency (rad/
   s): ');
omega_s2 = input('Enter second stopband edge frequency (rad
   /s): ');
A_p = input('Enter passband ripple (in dB): ');
A_s = input('Enter stopband attenuation (in dB): ');
fs = input('Enter sampling frequency (in Hz): ');

T = 1 / fs;
omega_p1 = (2 / T) * tan(omega_p1 * T / 2);
omega_p2 = (2 / T) * tan(omega_p2 * T / 2);
omega_s1 = (2 / T) * tan(omega_s1 * T / 2);
omega_s2 = (2 / T) * tan(omega_s2 * T / 2);

pepdt = omega_p1 * omega_p2;
sepdt = omega_s1 * omega_s2;

if pepdt > sepdt
    omega_s1 = pepdt / omega_s2;
elseif pepdt < sepdt
    omega_s2 = pepdt / omega_s1;
end

omega_p = 1;
omega_s = (omega_p2 - omega_p1) / (omega_s2 - omega_s1);
BW = omega_p2 - omega_p1;

del_p = 10^(-A_p / 20);
del_s = 10^(-A_s / 20);
a = (1 / del_s^2) - 1;
b = (1 / del_p^2) - 1;
eps = sqrt(b);

c = acosh(omega_s / omega_p);
x = acosh(sqrt(a / b)) / c;

N = 0;
while N < x
    N = N + 1;
```

```
end

disp(['Analog Chebyshev Filter order N = ', num2str(N)]);

temp = N;
while temp >= 2
    temp = temp - 2;
end

if temp == 0
    N_odd = false;
else
    N_odd = true;
end

yN = ((sqrt(1 + 1 / eps^2) + 1 / eps)^(1 / N) - (sqrt(1 + 1
    / eps^2) + 1 / eps)^(-1 / N)) / 2;

w = linspace(1, omega_p2 * 1.5, 2000);
Ha = zeros(size(w));

if N_odd
    for idx = 1:length(w)
        s1 = 1j * w(idx);
        s = (BW * s1) / (s1^2 + pepdt);
        H = (omega_p^N * yN) / (s + omega_p * yN);
        num = 1; deno = 1;
        for k = 1:(N - 1) / 2
            ck = yN^2 + (cos((2 * k - 1) * pi / (2 * N)))
               ^2;
            bk = 2 * yN * sin((2 * k - 1) * pi / (2 * N));
            num = num * ck;
            deno = deno * (s^2 + bk * omega_p * s + ck *
               omega_p^2);
        end
        Ha(idx) = abs(H * num / deno);
    end
else
    for idx = 1:length(w)
        s1 = 1j * w(idx);
        s = (BW * s1) / (s1^2 + pepdt);
        H = omega_p^N / sqrt(1 + eps^2);
        num = 1; deno = 1;
        for k = 1:(N / 2)
            ck = yN^2 + (cos((2 * k - 1) * pi / (2 * N)))
               ^2;
            bk = 2 * yN * sin((2 * k - 1) * pi / (2 * N));
            num = num * ck;
            deno = deno * (s^2 + bk * omega_p * s + ck *
               omega_p^2);
        end
        Ha(idx) = abs(H * num / deno);
    end
end

Nfft = 1024;
w_d = linspace(0, pi, Nfft);
Hd_blt = zeros(size(w_d));

for idx = 1:length(w_d)
    z = exp(1j * w_d(idx));
    s1 = (2 / T) * (1 - z^-1) / (1 + z^-1);
    s = (BW * s1) / (s1^2 + pepdt);
    if N_odd
        H = (omega_p^N * yN) / (s + omega_p * yN);
        num = 1; deno = 1;
        for k = 1:(N - 1) / 2
            ck = yN^2 + (cos((2 * k - 1) * pi / (2 * N)))
               ^2;
            bk = 2 * yN * sin((2 * k - 1) * pi / (2 * N));
            num = num * ck;
            deno = deno * (s^2 + bk * omega_p * s + ck *
               omega_p^2);
        end
        Hd_blt(idx) = H * num / deno;
    else
        H = omega_p^N / sqrt(1 + eps^2);
        num = 1; deno = 1;
        for k = 1:(N / 2)
            ck = yN^2 + (cos((2 * k - 1) * pi / (2 * N)))
               ^2;
            bk = 2 * yN * sin((2 * k - 1) * pi / (2 * N));
            num = num * ck;
            deno = deno * (s^2 + bk * omega_p * s + ck *
               omega_p^2);
        end
        Hd_blt(idx) = H * num / deno;
    end
end

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

```
plot(w, 20 * log10(Ha), 'LineWidth', 1.5);
xlabel('\omega (rad/s)');
ylabel('Magnitude (dB)');
title('Analog Chebyshev Type-I Bandstop Filter');
grid on;
ylim([-60 5]);

subplot(2, 1, 2);
f_Hz = w_d * fs / (2 * pi);
plot(f_Hz, 20 * log10(abs(Hd_blt)), 'LineWidth', 1.5);
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
title('Digital Chebyshev Type-I Bandstop Filter (Bilinear
   Transform)');
grid on;
ylim([-60 5]);
```

Code Implementation

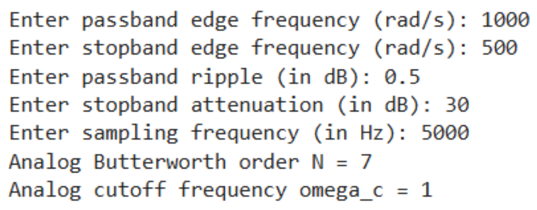
The filter design and implementation were performed in MATLAB for various analog and digital filters using Butterworth and Chebyshev characteristics. The programs accept user-defined specifications such as passband and stopband edge frequencies, passband ripple, stopband attenuation, and sampling frequency.

Each analog filter is first designed and analyzed in its continuous-time domain. The analog prototype is then converted into its corresponding digital form using the Bilinear Transformation (BLT) method, which effectively maps the s-plane to the z-plane while preserving the frequency response characteristics. Frequency pre-warping is applied to ensure accurate transformation of critical frequencies.

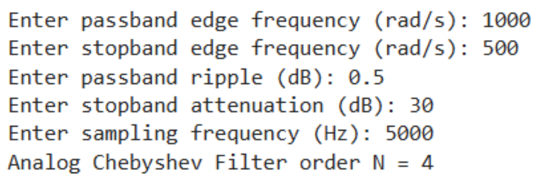
The MATLAB scripts compute the required filter order, determine cutoff frequencies, and generate frequency responses for both analog and digital versions. The magnitude responses are plotted in decibels (dB), allowing for a clear comparison between the analog and digital filter behaviors.

Overall, this implementation demonstrates the complete process of designing and realizing Butterworth and Chebyshev filters for different types (Low Pass, Band Pass, and Band Stop) using Bilinear Transformation in MATLAB.

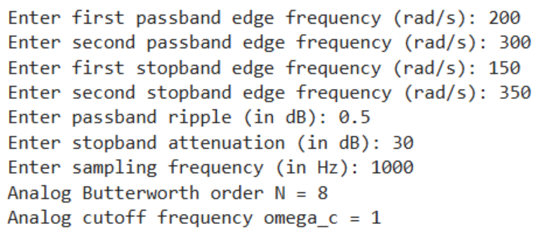
Output



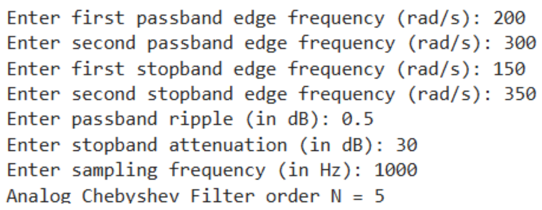
```
Enter passband edge frequency (rad/s): 1000
Enter stopband edge frequency (rad/s): 500
Enter passband ripple (in dB): 0.5
Enter stopband attenuation (in dB): 30
Enter sampling frequency (in Hz): 5000
Analog Butterworth order N = 7
Analog cutoff frequency omega_c = 1
```



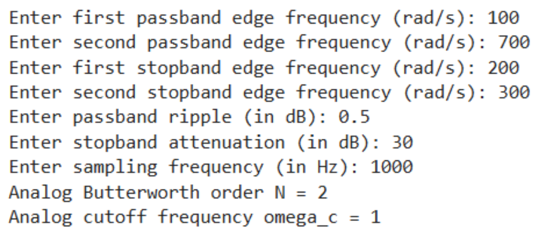
```
Enter passband edge frequency (rad/s): 1000
Enter stopband edge frequency (rad/s): 500
Enter passband ripple (dB): 0.5
Enter stopband attenuation (dB): 30
Enter sampling frequency (Hz): 5000
Analog Chebyshev Filter order N = 4
```



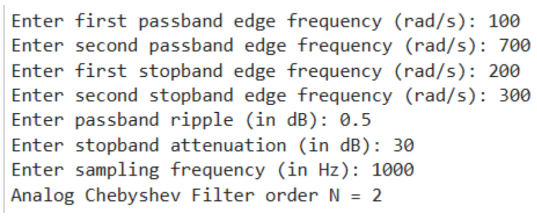
```
Enter first passband edge frequency (rad/s): 200
Enter second passband edge frequency (rad/s): 300
Enter first stopband edge frequency (rad/s): 150
Enter second stopband edge frequency (rad/s): 350
Enter passband ripple (in dB): 0.5
Enter stopband attenuation (in dB): 30
Enter sampling frequency (in Hz): 1000
Analog Butterworth order N = 8
Analog cutoff frequency omega_c = 1
```



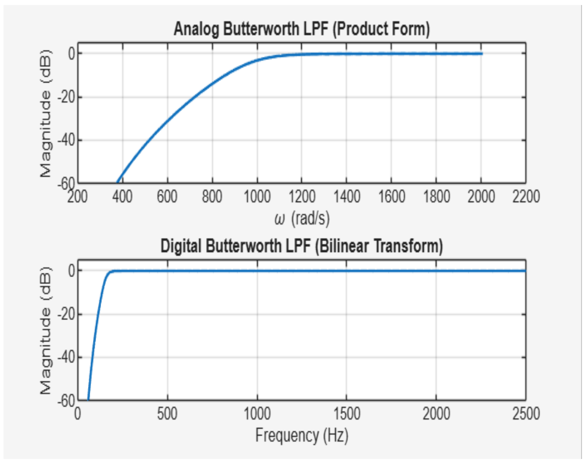
```
Enter first passband edge frequency (rad/s): 200
Enter second passband edge frequency (rad/s): 300
Enter first stopband edge frequency (rad/s): 150
Enter second stopband edge frequency (rad/s): 350
Enter passband ripple (in dB): 0.5
Enter stopband attenuation (in dB): 30
Enter sampling frequency (in Hz): 1000
Analog Chebyshev Filter order N = 5
```

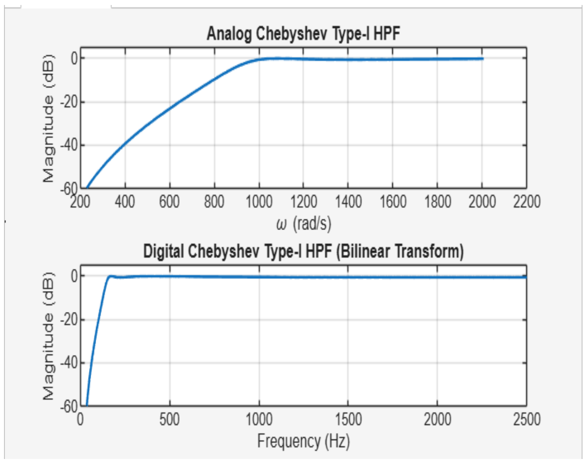


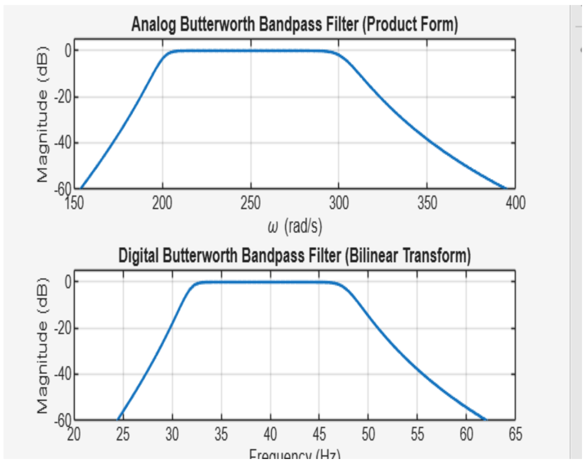
```
Enter first passband edge frequency (rad/s): 100
Enter second passband edge frequency (rad/s): 700
Enter first stopband edge frequency (rad/s): 200
Enter second stopband edge frequency (rad/s): 300
Enter passband ripple (in dB): 0.5
Enter stopband attenuation (in dB): 30
Enter sampling frequency (in Hz): 1000
Analog Butterworth order N = 2
Analog cutoff frequency omega_c = 1
```

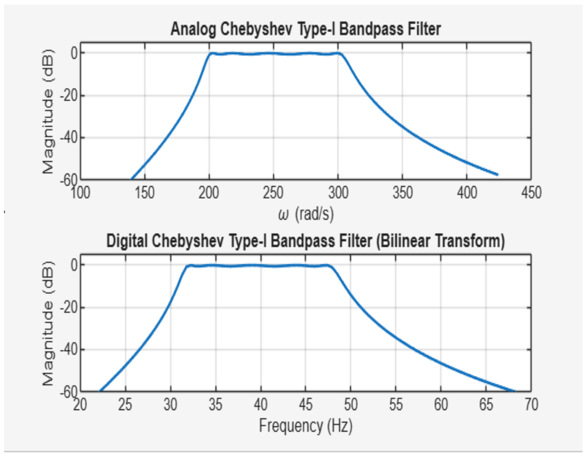


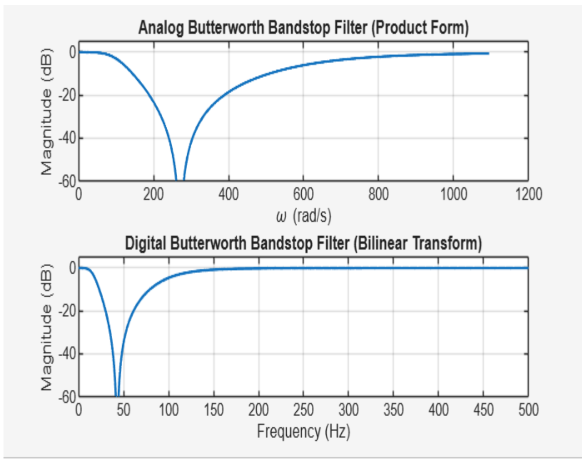
```
Enter first passband edge frequency (rad/s): 100
Enter second passband edge frequency (rad/s): 700
Enter first stopband edge frequency (rad/s): 200
Enter second stopband edge frequency (rad/s): 300
Enter passband ripple (in dB): 0.5
Enter stopband attenuation (in dB): 30
Enter sampling frequency (in Hz): 1000
Analog Chebyshev Filter order N = 2
```

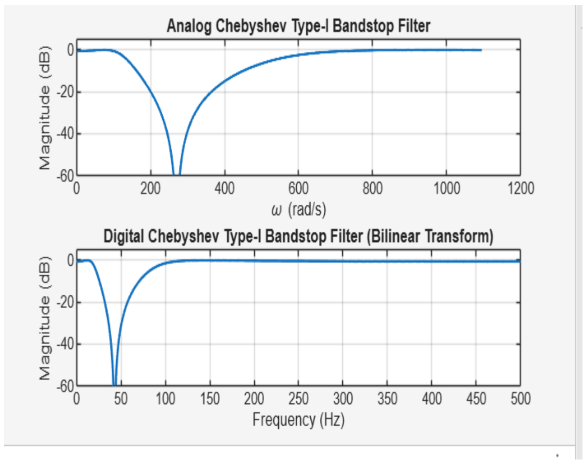












Observations

From the simulation results obtained through MATLAB:

- The analog and digital frequency responses show that the Bilinear Transformation (BLT) method preserves the general shape of the filter characteristics after transformation.
- The magnitude response plots confirm that the Butterworth filters provide a maximally flat passband, while the Chebyshev filters exhibit a ripple in the passband but achieve a steeper roll-off.

- For Band Pass and Band Stop filters, the bandwidth and center frequency were accurately maintained after applying BLT.
- The increase in filter order results in improved selectivity but also increases the computational complexity.
- Frequency pre-warping in BLT effectively compensates for the nonlinear frequency mapping between analog and digital domains.

Conclusion

The design and implementation of Butterworth and Chebyshev filters using the Bilinear Transformation method were successfully carried out in MATLAB. The results demonstrate that:

- The Bilinear Transformation effectively converts analog filter prototypes to digital filters with good accuracy.
- Butterworth filters are preferable for applications requiring a smooth passband, whereas Chebyshev filters are suitable when sharper transitions are desired.
- The theoretical and simulated responses align closely, validating the correctness of the design approach.

Overall, the practical implementation confirms the expected behavior of different filter types and highlights the trade-offs between passband flatness, transition sharpness, and filter order.