National Institute of Technology
Calicut

Department of Electronics and Communication
Engineering

Digital Signal Processing LAB

Week 7 Report

Date: 02-10–2025
Instructor: Dr. Sakthivel V

Submitted by:
Mummana Jagadeesh B231113EC
Muhammed Nihal MP B230437EC

Aim

- To design and analyze low-pass **Butterworth** and **Chebyshev** filters based on the given passband and stopband specifications.
- To realize the corresponding digital filters using the following transformation techniques: **Bilinear Transformation (BLT)** and **Impulse Invariant Transformation (IIT)**
- To plot and compare the frequency responses of both analog and digital implementations.

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:

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

Analog-to-Digital Transformations

After designing the analog prototype filter, it is converted into a digital filter using one of the following methods:

1. Impulse Invariant Transformation (IIT)

The Impulse Invariant Transformation method designs a digital filter by sampling the analog filter's impulse response.

- **Basic Idea:** The discrete-time impulse response is obtained by sampling the analog impulse response:
$$h_d[n] = T \cdot h_a(nT), \quad n \geq 0$$
- **Mapping Relation:** The relation between the s-plane and the z-plane is:
$$z = e^{sT} \quad \text{or equivalently} \quad s = \frac{1}{T}\ln(z)$$
Hence, a pole at $s = s_k$ in the analog domain maps to a pole at:
$$z = e^{s_k T}$$
- **Properties:**
  - Preserves the time-domain impulse response at sampling instants.
  - Simple to implement when $H_a(s)$ is expressed in partial fractions.
  - Suffers from aliasing in the frequency domain.
  - Works best for low-pass filters where most energy lies below the Nyquist frequency.
  - Does not introduce frequency warping (unlike BLT).

- **Digital Transfer Function:** For an analog transfer function

$$H_a(s) = \sum_k \frac{A_k}{s - s_k},$$

the corresponding digital transfer function is:

$$H_d(z) = \sum_k \frac{A_k T}{1 - e^{s_k T} z^{-1}}$$

Summary: IIT provides a straightforward way to obtain digital filters that preserve time-domain characteristics. However, aliasing limits its use, making it more suitable for low-pass filter designs.

2. 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

BUTTERWORTH BLT

```
clc; clear;

p1 = input('passband freq (rad/s): ');
p2 = input('stopband freq (rad/s): ');
r1 = input('passband ripple (dB): ');
r2 = input('stopband attenuation (dB): ');
fsamp = input('sampling freq (Hz): ');

ts = 1/fsamp;

p1 = (2/ts)*tan((p1*ts)/2);
p2 = (2/ts)*tan((p2*ts)/2);

dp = 10^(-r1/20);
ds = 10^(-r2/20);

A = (1/ds^2) - 1;
B = (1/dp^2) - 1;
C = log10(p2/p1);
val = log10(sqrt(A/B)) / C;

n = 1;
while n < val
    n = n + 1;
end

wc = p1 / ((1/dp^2 - 1)^(1/(2*n)));

disp(['N = ', num2str(n)]);
disp(['wc = ', num2str(wc)]);

wset = linspace(0, 2*p2, 500);
Hval = zeros(size(wset));

odd = false;
temp = n;
while temp > 1
    temp = temp - 2;
end
if temp == 1
    odd = true;
end
```

```
i = 1;
while i <= length(wset)
    s = 1j*wset(i);
    P = 1;
    if odd
        k = 1;
        while k <= (n - 1)/2
            b = 2*sin((2*k - 1)*pi/(2*n));
            P = P * (s^2 + b*wc*s + wc^2);
            k = k + 1;
        end
        Hval(i) = abs((wc^n)/((s + wc)*P));
    else
        k = 1;
        while k <= n/2
            b = 2*sin((2*k - 1)*pi/(2*n));
            P = P * (s^2 + b*wc*s + wc^2);
            k = k + 1;
        end
        Hval(i) = abs((wc^n)/P);
    end
    i = i + 1;
end

pts = 1024;
wd = linspace(0, pi, pts);
Hdb = zeros(size(wd));

j = 1;
while j <= length(wd)
    z = exp(1j*wd(j));
    s = (2/ts)*((1 - z^-1)/(1 + z^-1));
    P = 1;
    k = 1;
    while k <= n
        t = ((2*k) + n - 1)*pi/(2*n);
        ck = wc*exp(1j*t);
        P = P * (wc/(s - ck));
        k = k + 1;
    end
    Hdb(j) = P;
    j = j + 1;
end

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

```
plot(wset, 20*log10(Hval), 'LineWidth', 1.5);
xlabel('\omega (rad/s)');
ylabel('Magnitude (dB)');
title('Analog Butterworth LPF');
grid on; ylim([-60 5]);

subplot(2,1,2);
faxis = wd * fsamp / (2*pi);
plot(faxis, 20*log10(abs(Hdb)), 'LineWidth', 1.5);
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
title('Digital Butterworth LPF (BLT)');
grid on; ylim([-60 5]);
```

BUTTERWORTH IIT

```
clc; clear;

p1 = input('passband freq (rad/s): ');
p2 = input('stopband freq (rad/s): ');
r1 = input('passband ripple (dB): ');
r2 = input('stopband attenuation (dB): ');
fsamp = input('sampling freq (Hz): ');

ts = 1/fsamp;

dp = 10^(-r1/20);
ds = 10^(-r2/20);

A = (1/ds^2) - 1;
B = (1/dp^2) - 1;
C = log10(p2/p1);
val = log10(sqrt(A/B)) / C;

n = 1;
while n < val
    n = n + 1;
end

wc = p1 / ((1/dp^2 - 1)^(1/(2*n)));

disp(['N = ', num2str(n)]);
disp(['wc = ', num2str(wc)]);

wset = linspace(0, 2*p2, 500);
Ha = zeros(size(wset));

odd = false;
tmp = n;
while tmp > 1
    tmp = tmp - 2;
end
if tmp == 1
    odd = true;
end

i = 1;
while i <= length(wset)
    s = 1j*wset(i);
    P = 1;
```

```
    if odd
        k = 1;
        while k <= (n - 1)/2
            b = 2*sin((2*k - 1)*pi/(2*n));
            P = P * (s^2 + b*wc*s + wc^2);
            k = k + 1;
        end
        Ha(i) = abs((wc^n)/((s + wc)*P));
    else
        k = 1;
        while k <= n/2
            b = 2*sin((2*k - 1)*pi/(2*n));
            P = P * (s^2 + b*wc*s + wc^2);
            k = k + 1;
        end
        Ha(i) = abs((wc^n)/P);
    end
    i = i + 1;
end

pts = 1024;
wd = linspace(0, pi, pts);

S = zeros(1, n);
k = 0;
while k < n
    t = pi * (2*k + n + 1) / (2*n);
    S(k+1) = wc * exp(1j*t);
    k = k + 1;
end

R = zeros(1, n);
k = 1;
while k <= n
    sk = S(k);
    other = [S(1:k-1) S(k+1:n)];
    den = 1;
    m = 1;
    while m <= length(other)
        den = den * (sk - other(m));
        m = m + 1;
    end
    R(k) = ts * wc^n / den;
    k = k + 1;
end
```

```
H_iit = zeros(size(wd));
i = 1;
while i <= length(wd)
    z = exp(1j*wd(i));
    summ = 0;
    k = 1;
    while k <= n
        summ = summ + R(k) / (1 - exp(S(k)*ts) * z^-1);
        k = k + 1;
    end
    H_iit(i) = abs(summ);
    i = i + 1;
end

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

subplot(2,1,2);
faxis = wd * fsamp / (2*pi);
plot(faxis, 20*log10(H_iit), 'LineWidth', 1.5);
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
title('Digital Butterworth LPF (IIT)');
grid on; ylim([-60 5]);
```

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);

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; % even
else
    N_odd = true;  % odd
end

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

CHEBYSHEV IIT

```
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;

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; % even
else
    N_odd = true;  % odd
end

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);
        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);
        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);

s_poles = zeros(1,N);
for k = 1:N
    theta = (2*k-1)*pi/(2*N);
    sigma = -sinh(asinh(1/eps)/N) * sin(theta);
    omega = cosh(asinh(1/eps)/N) * cos(theta);
    s_poles(k) = omega_p*(sigma + 1j*omega);
end

R = zeros(1,N);
for k = 1:N
    s_k = s_poles(k);
    other_poles = s_poles([1:k-1, k+1:N]);
    den = 1;
    for m = 1:length(other_poles)
        den = den * (s_k - other_poles(m));
    end
    R(k) = T * omega_p^N / den;
end
```

```
Hd_iit = zeros(size(w_d));
for idx = 1:length(w_d)
    z = exp(1j*w_d(idx));
    Hsum = 0;
    for k = 1:N
        Hsum = Hsum + R(k) / (1 - exp(s_poles(k)*T) * z^-1)
            ;
    end
    Hd_iit(idx) = abs(Hsum);
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 LPF');
grid on; ylim([-60 5]);

subplot(2,1,2);
f_Hz = w_d*fs/(2*pi);
plot(f_Hz, 20*log10(abs(Hd_iit)),'LineWidth',1.5);
xlabel('Frequency (Hz)'); ylabel('Magnitude (dB)');
title('Digital Chebyshev Type-I LPF (Impulse Invariant)');
grid on; ylim([-60 10]);
```

Code Implementation

The implementation of the Butterworth and Chebyshev filters (both using Bilinear Transform (BLT) and Impulse Invariant Transform (IIT)) is carried out in MATLAB. Each program follows a structured sequence of steps as summarized below:

1. **Input Parameters:** The user provides essential specifications such as passband frequency, stopband frequency, passband ripple, stopband attenuation, and sampling frequency.

2. **Pre-Warping:** To compensate for frequency distortion due to digitization, the analog frequencies are pre-warped using the tangent relation:

$$\omega = \frac{2}{T} \tan\left(\frac{\omega_d T}{2}\right)$$

   where $T = 1/f_s$ is the sampling period.

3. **Filter Order and Cutoff Frequency:** The required filter order N and cutoff frequency ω_c (or ω_p for Chebyshev) are computed based on passband and stopband specifications using standard filter equations.

4. **Analog Prototype Design:** The analog low-pass prototype filter is designed using the Butterworth or Chebyshev equations. Its transfer function is determined based on whether the order N is odd or even.

5. **Transformation to Digital Domain:**

   - **Bilinear Transform (BLT):** The analog frequency variable s is replaced using
$$s = \frac{2}{T} \cdot \frac{1 - z^{-1}}{1 + z^{-1}}$$
     to obtain the corresponding digital transfer function.
   - **Impulse Invariant Transform (IIT):** The analog poles are mapped to the digital domain using
$$z = e^{sT}$$
     preserving the impulse response characteristics.

6. **Frequency Response Calculation:** The magnitude response is computed for a range of frequencies and plotted both for the analog and digital filters. The responses are displayed in decibels (dB).

7. **Plotting Results:** Two subplots are generated in each case:

   - The first subplot shows the analog low-pass filter response.
   - The second subplot shows the corresponding digital filter response using either BLT or IIT.

In summary, the code automates the process of designing and comparing analog and digital low-pass filters using both Butterworth and Chebyshev methods with two different transformation techniques (BLT and IIT).

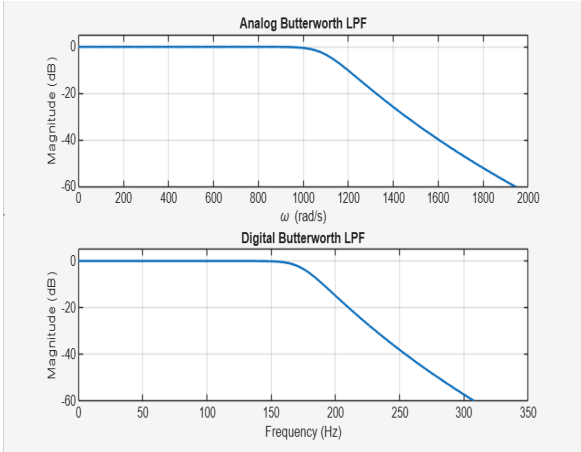
Output

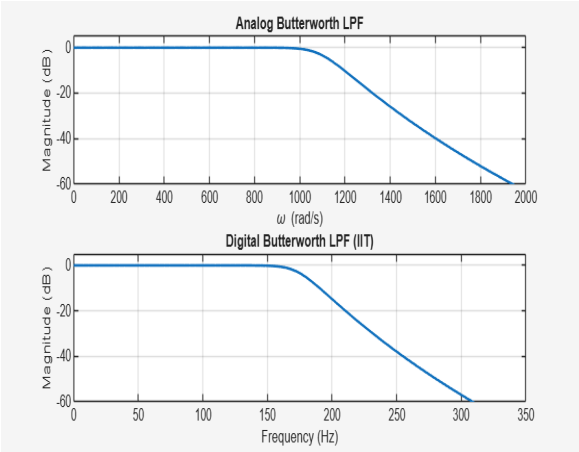
```
Enter passband edge frequency (rad/s): 1000
Enter stopband edge frequency (rad/s): 1500
Enter passband ripple (in dB): 0.5
Enter stopband attenuation (in dB): 30
Enter sampling frequency (in Hz): 8000
Analog Butterworth order N = 12
Analog cutoff frequency omega_c = 1093.0288
```

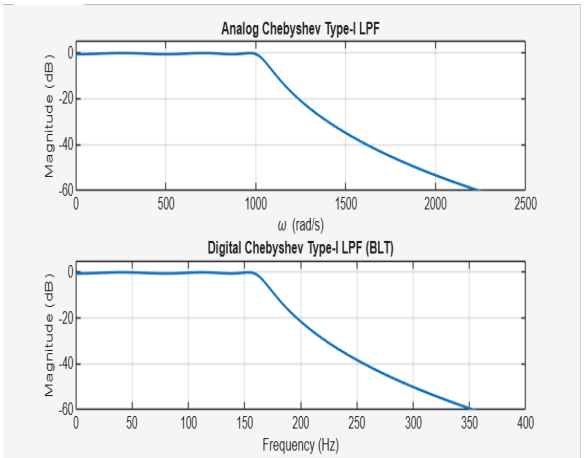
```
Enter passband edge frequency (rad/s): 1000
Enter stopband edge frequency (rad/s): 1500
Enter passband ripple (in dB): 0.5
Enter stopband attenuation (in dB): 30
Enter sampling frequency (in Hz): 8000
Analog Butterworth order N = 12
Analog cutoff frequency omega_c = 1091.6052
```

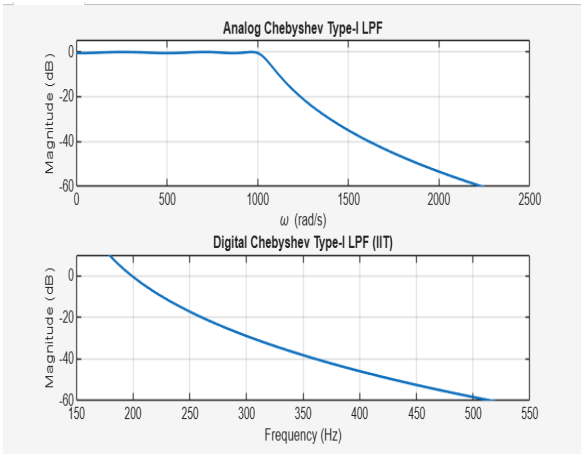
```
Enter passband edge frequency (rad/s): 1000
Enter stopband edge frequency (rad/s): 1500
Enter passband ripple (dB): 0.5
Enter stopband attenuation (dB): 30
Enter sampling frequency (Hz): 8000
Analog Chebyshev Filter order N = 6
>>
```

```
Enter passband edge frequency (rad/s): 1000
Enter stopband edge frequency (rad/s): 1500
Enter passband ripple (dB): 0.5
Enter stopband attenuation (dB): 30
Enter sampling frequency (Hz): 8000
Analog Chebyshev Filter order N = 6
>>
```









Observations

- The required filter order is influenced by the desired passband ripple and the attenuation in the stopband.
- Analog Butterworth filters are characterized by a smooth, flat passband response, whereas Chebyshev filters exhibit small oscillations (ripples) within the passband.
- When converting to digital filters, the Bilinear Transformation (BLT) compresses the entire analog frequency spectrum into the digital range, causing a nonlinear frequency mapping known as frequency warping.
- Impulse Invariant Transformation (IIT) maintains the time-domain shape of the analog impulse response, but may introduce aliasing due to overlapping frequency components.
- Overall, the digital filters designed closely replicate their analog counterparts. Butterworth filters produce a gentle attenuation slope, while Chebyshev filters offer sharper transitions at the cost of passband ripples.

Conclusion

- Both Butterworth and Chebyshev filters were effectively designed to meet the specified requirements and successfully converted to digital filters.
- Butterworth filters are preferable when a flat passband is essential, while Chebyshev filters are chosen for applications requiring rapid transition from passband to stopband, even if some passband ripple is acceptable.
- Regarding digital transformation techniques, BLT avoids aliasing but introduces frequency distortion, whereas IIT preserves the impulse response shape but is prone to aliasing.
- The optimal combination of filter type (Butterworth vs. Chebyshev) and digital transformation method (BLT vs. IIT) depends on the application priorities such as flatness of the passband, sharpness of frequency transition, and tolerance to aliasing effects.