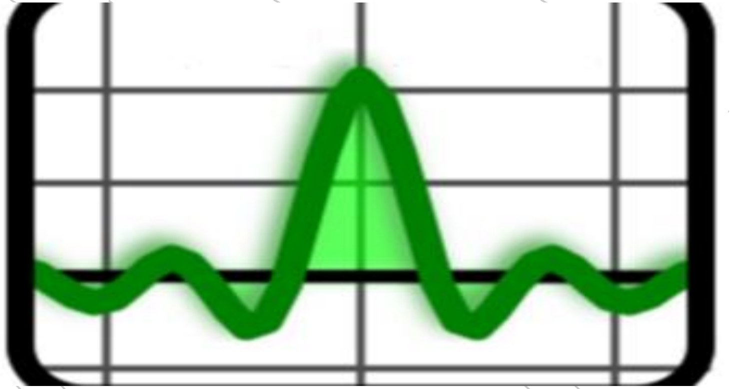
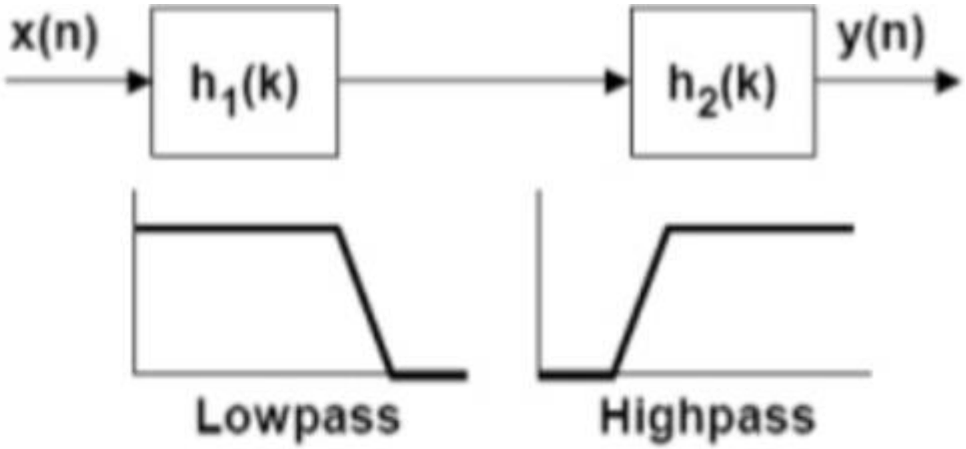


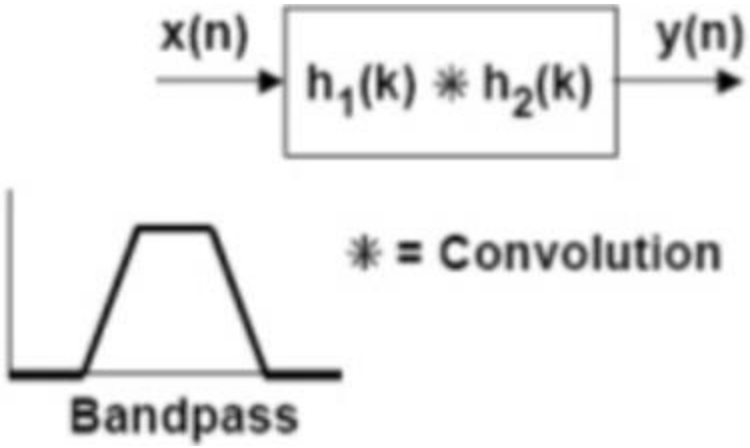
Signals & Systems For Computer Engineering

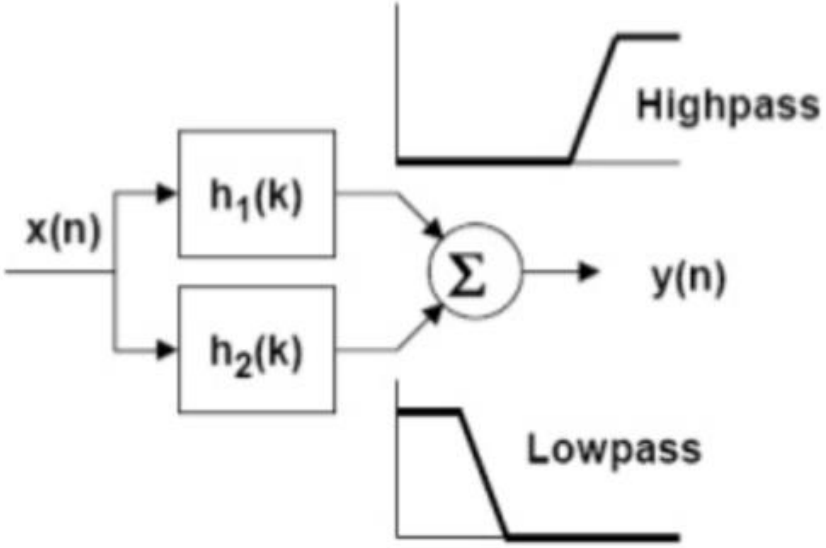
Prof.Dr. B.Berk ÜSTÜNDAĞ
Istanbul Technical University
Faculty of Computer Engineering and Informatics
bustundag@itu.edu.tr

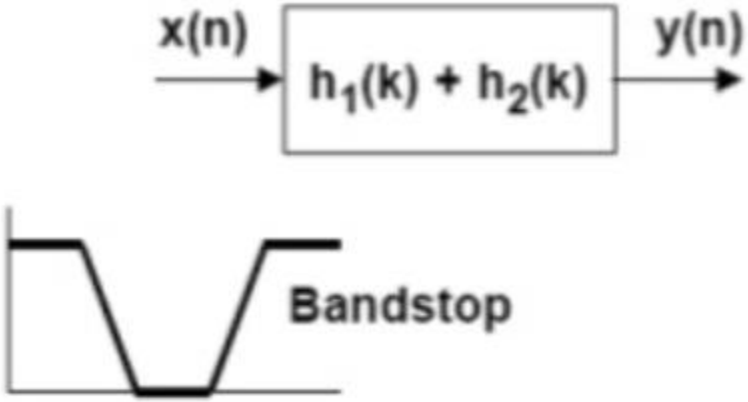
BLG354E / CRN: 21560
14th Week Lecture

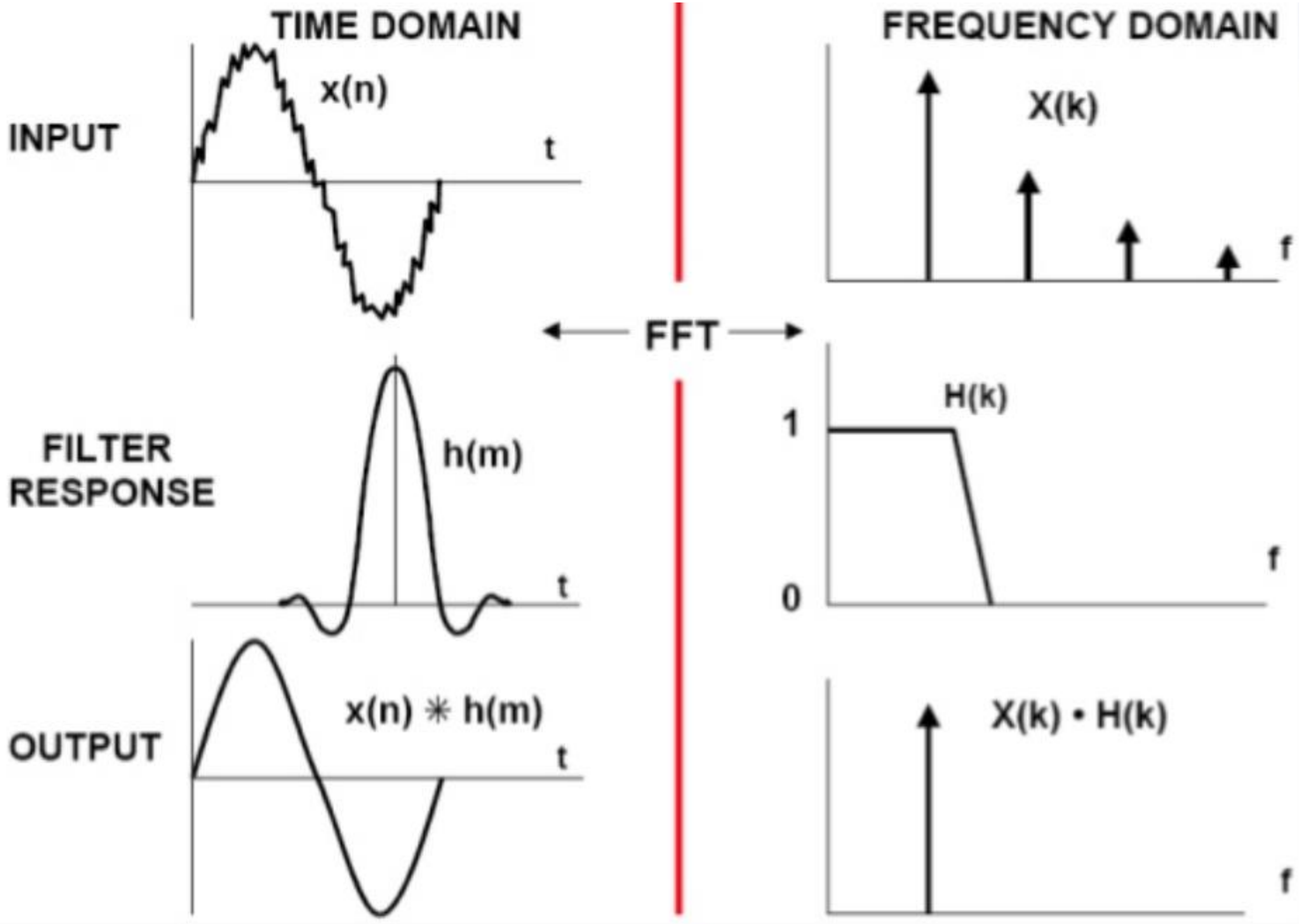
Common Filter Characteristics depending on LPF and HPF



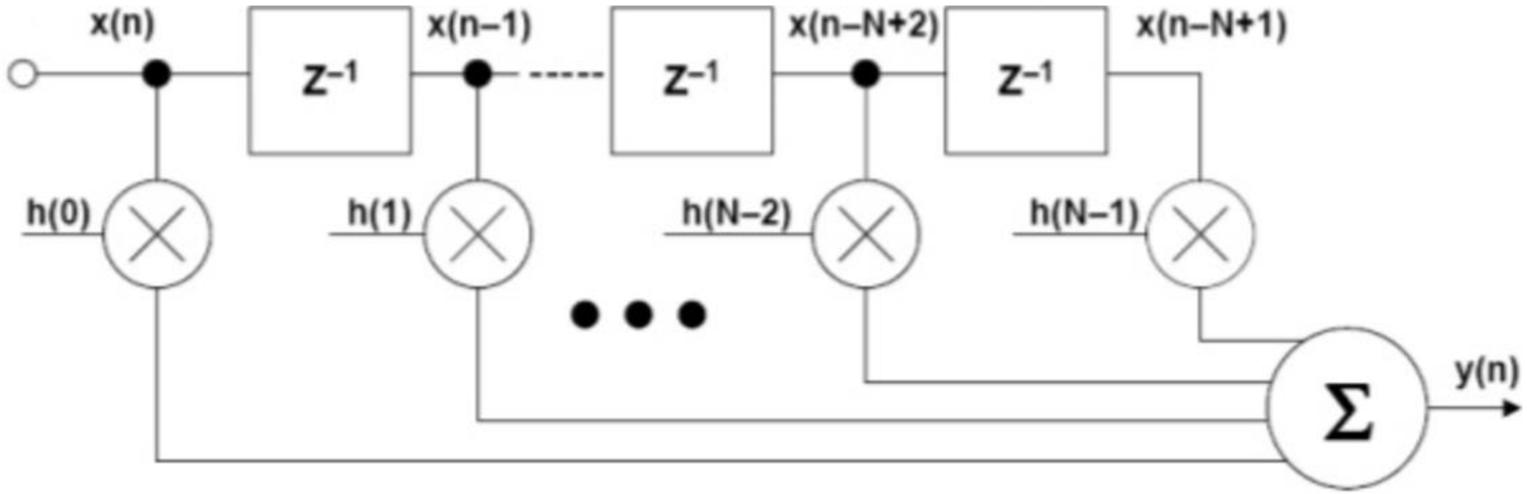






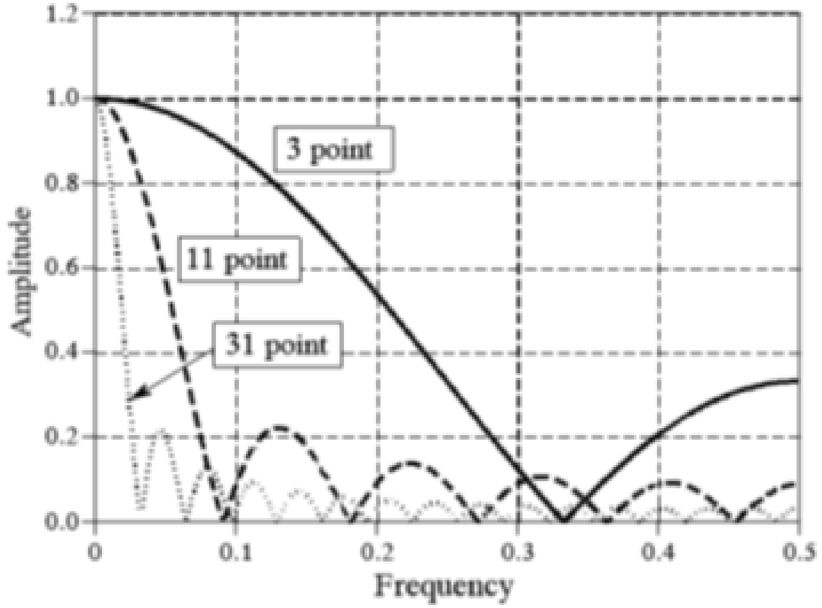

TIME DOMAIN
FREQUENCY DOMAIN
INPUT
x(n)
t
X(k)
f
FFT
FILTER
RESPONSE
h(m)
t
H(k)
1
0
f
OUTPUT
x(n) ∗ h(m)
t
X(k) • H(k)
f

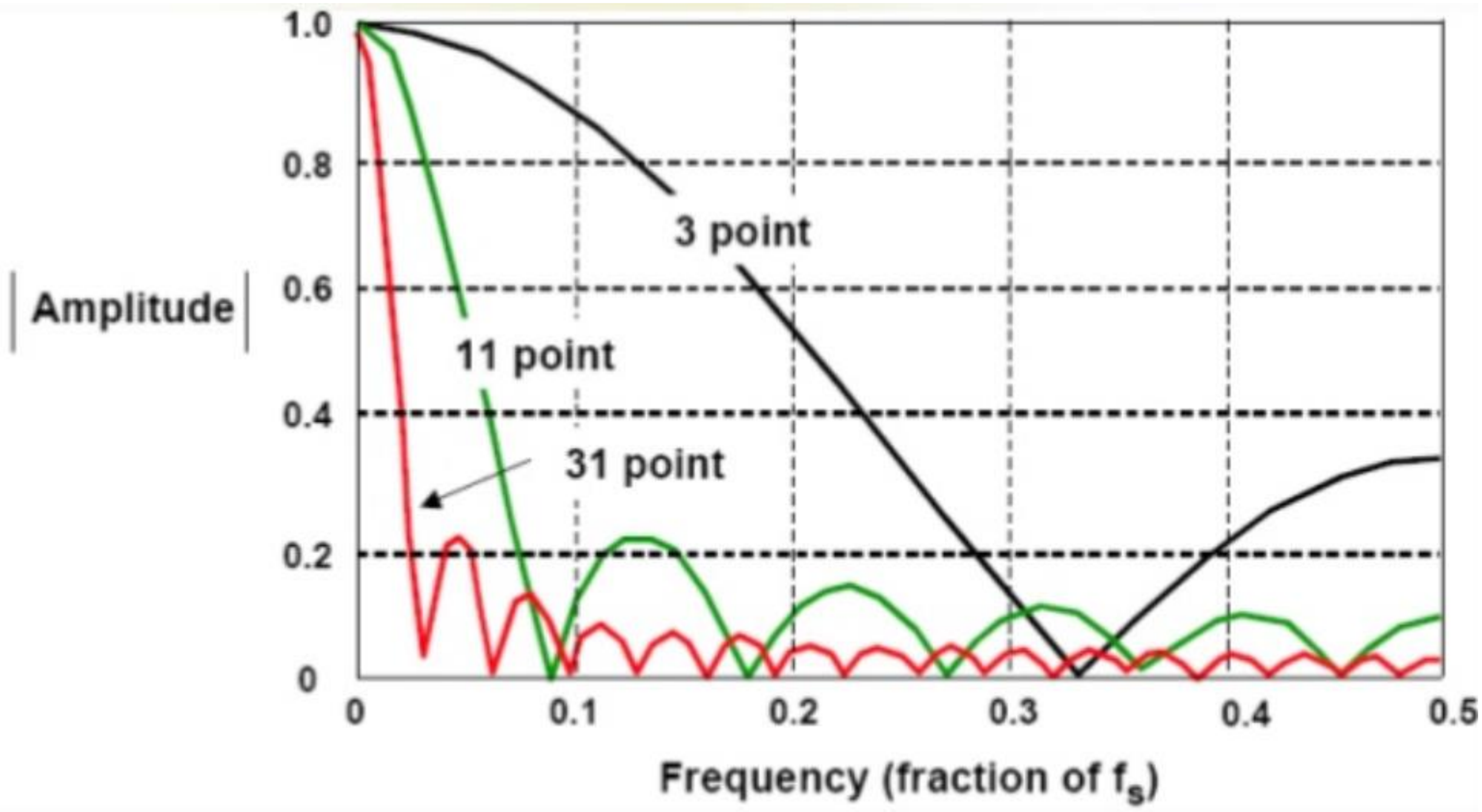
Moving Average Filter: its simplicity vs weaknesses:



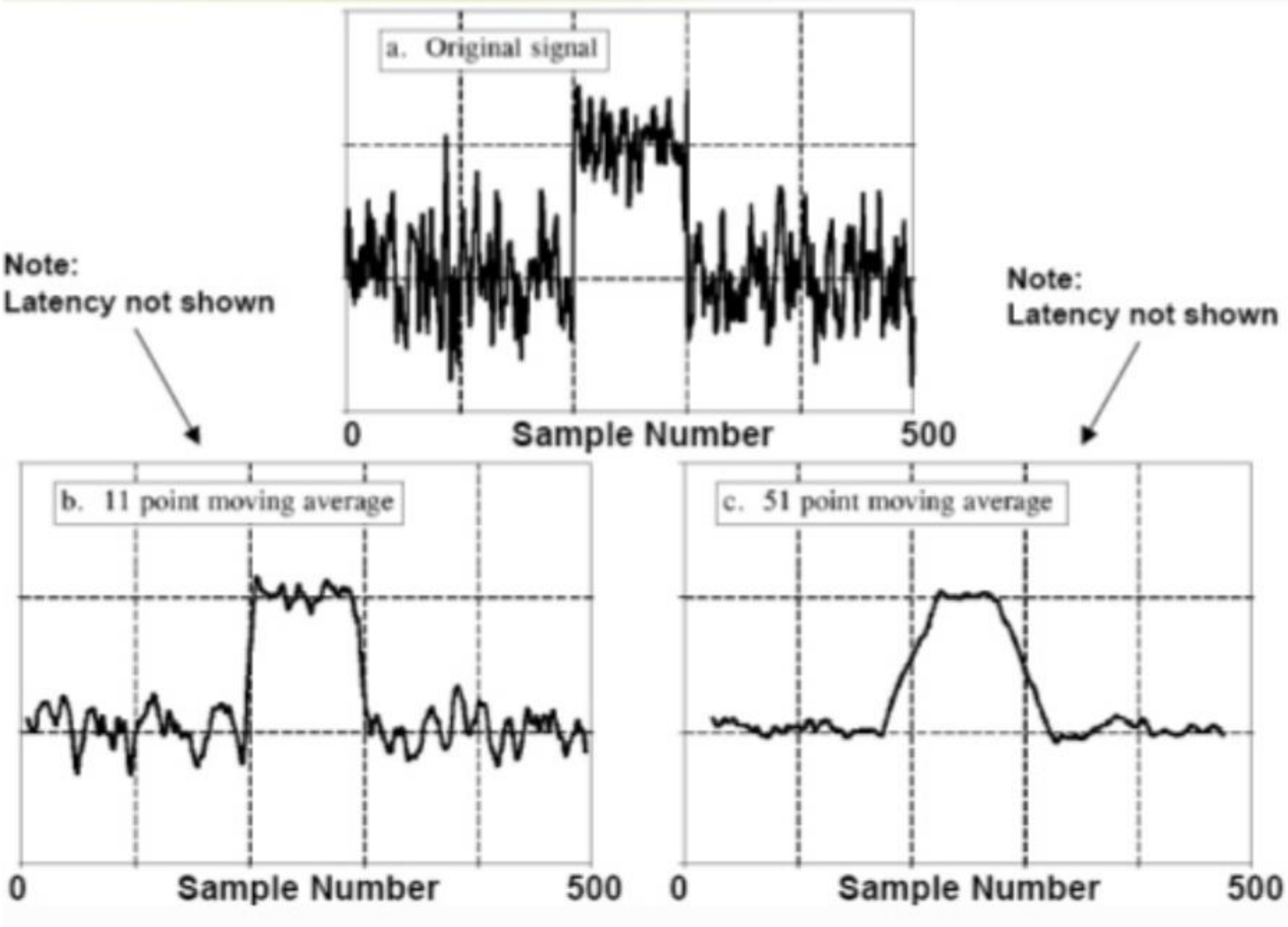
$$y(n) = h(n) \ast x(n) = \sum_{k=0}^{N-1} h(k)\, x(n-k)$$

$$H[f] = \frac{\sin(\pi f M)}{M \sin(\pi f)}$$

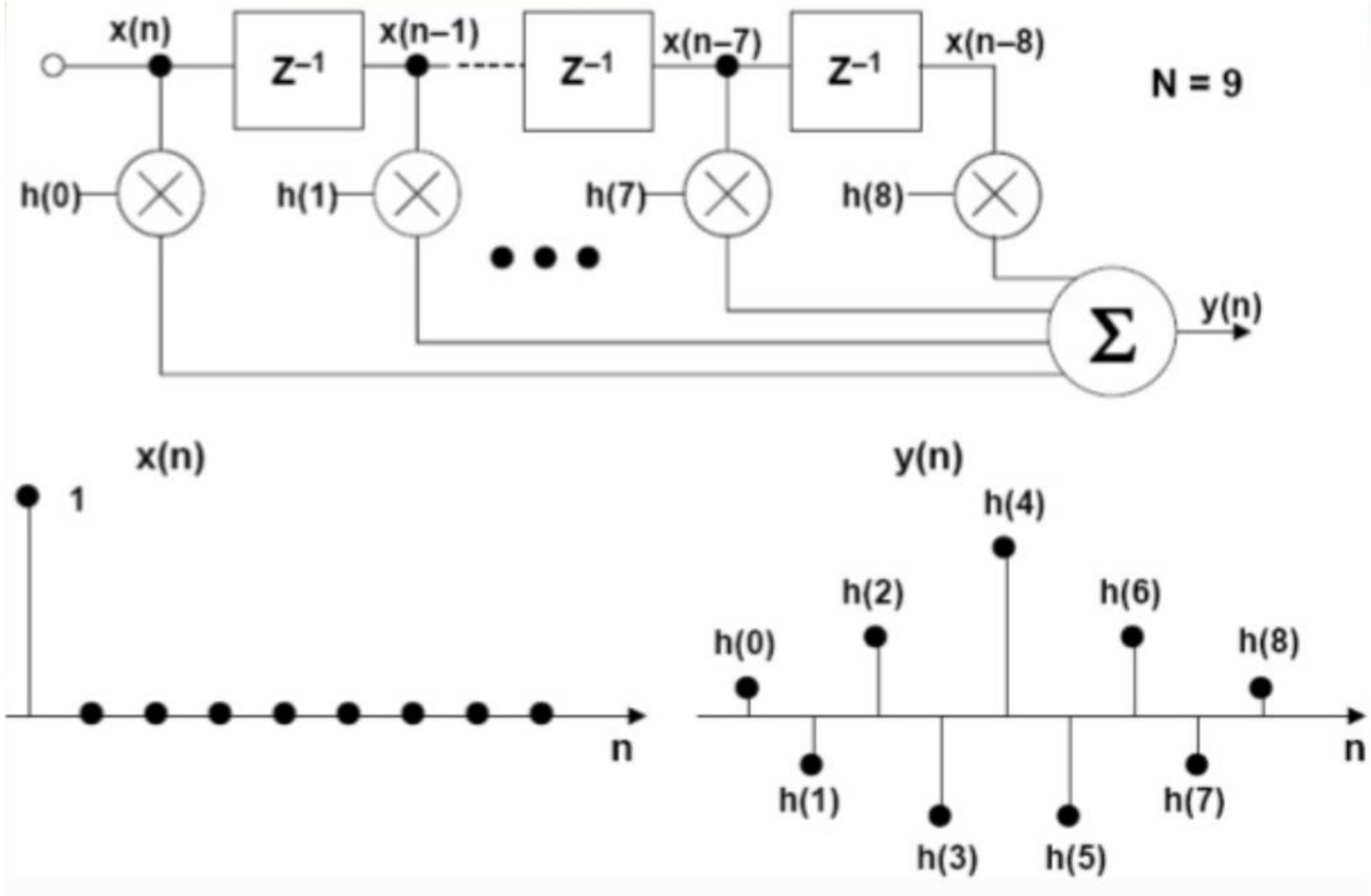




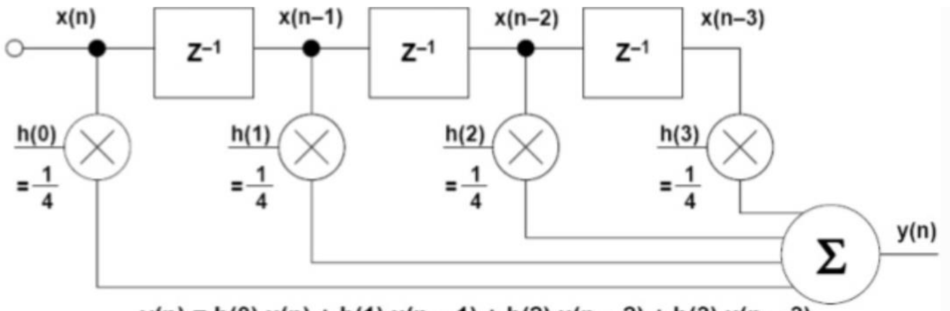
Moving average is an inefficient filtering technic due to its slow roll-off and poor stopband attenuation


a. Original signal
0
Sample Number
500
Note:
Latency not shown
Note:
Latency not shown
b. 11 point moving average
0
Sample Number
500
c. 51 point moving average
0
Sample Number
500

Impulse Response Determines the Filter Coefficients



MAV Filter Characterization:



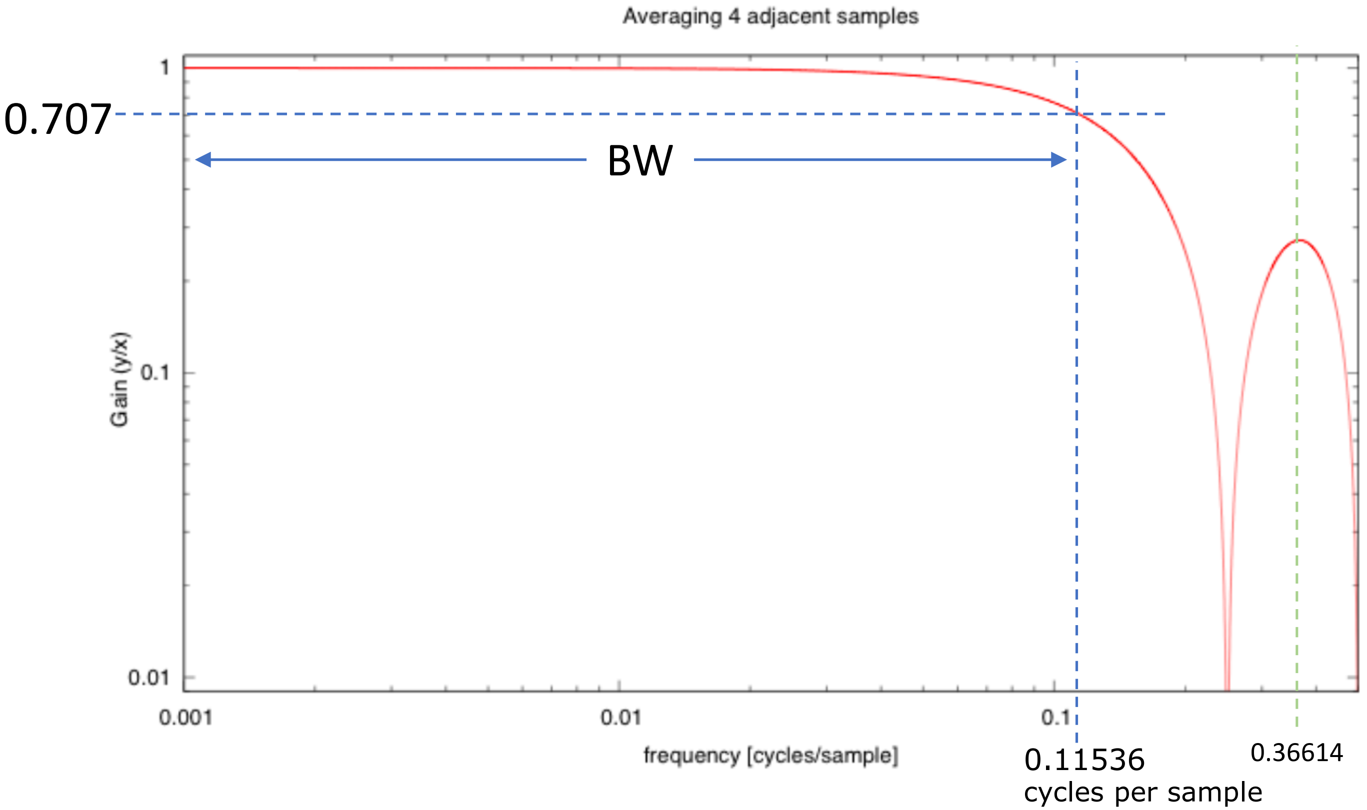
$$y(n) = h(0)\,x(n) + h(1)\,x(n-1) + h(2)\,x(n-2) + h(3)\,x(n-3)$$
$$= \frac{1}{4}x(n) + \frac{1}{4}x(n-1) + \frac{1}{4}x(n-2) + \frac{1}{4}x(n-3)$$
$$= \frac{1}{4}\left[x(n) + x(n-1) + x(n-2) + x(n-3)\right]$$

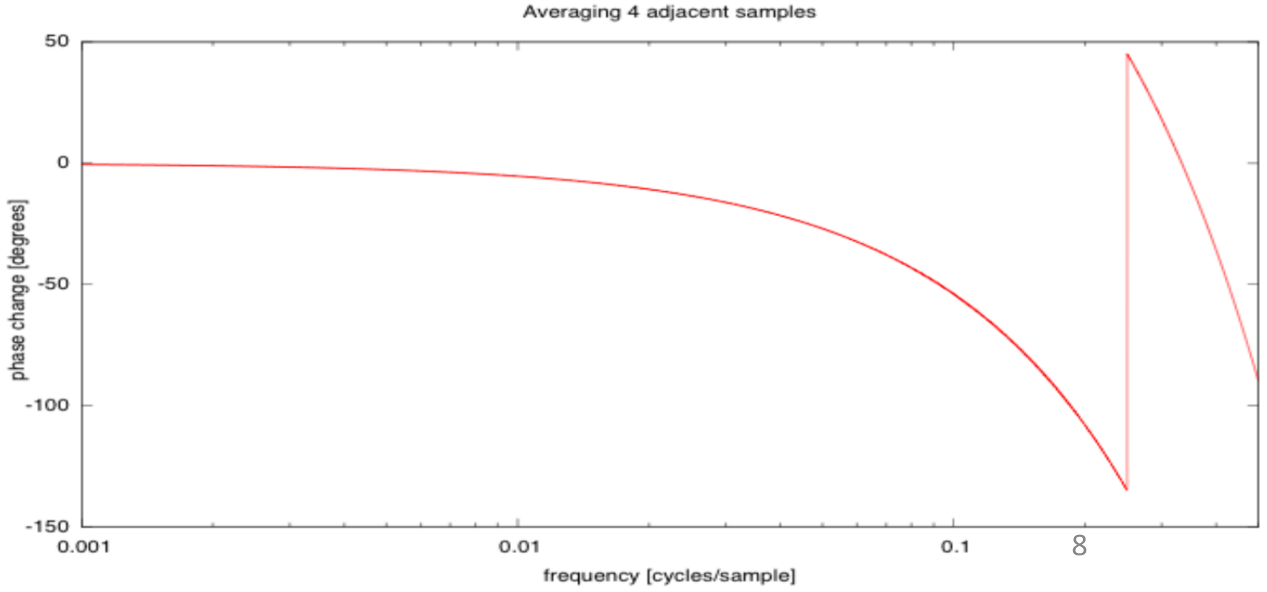
$$H(z) = 1/4(1 + z^{-1} + z^{-2} + z^{-3}).$$

$z=e^{j\omega}$, $f=\omega/2\pi$

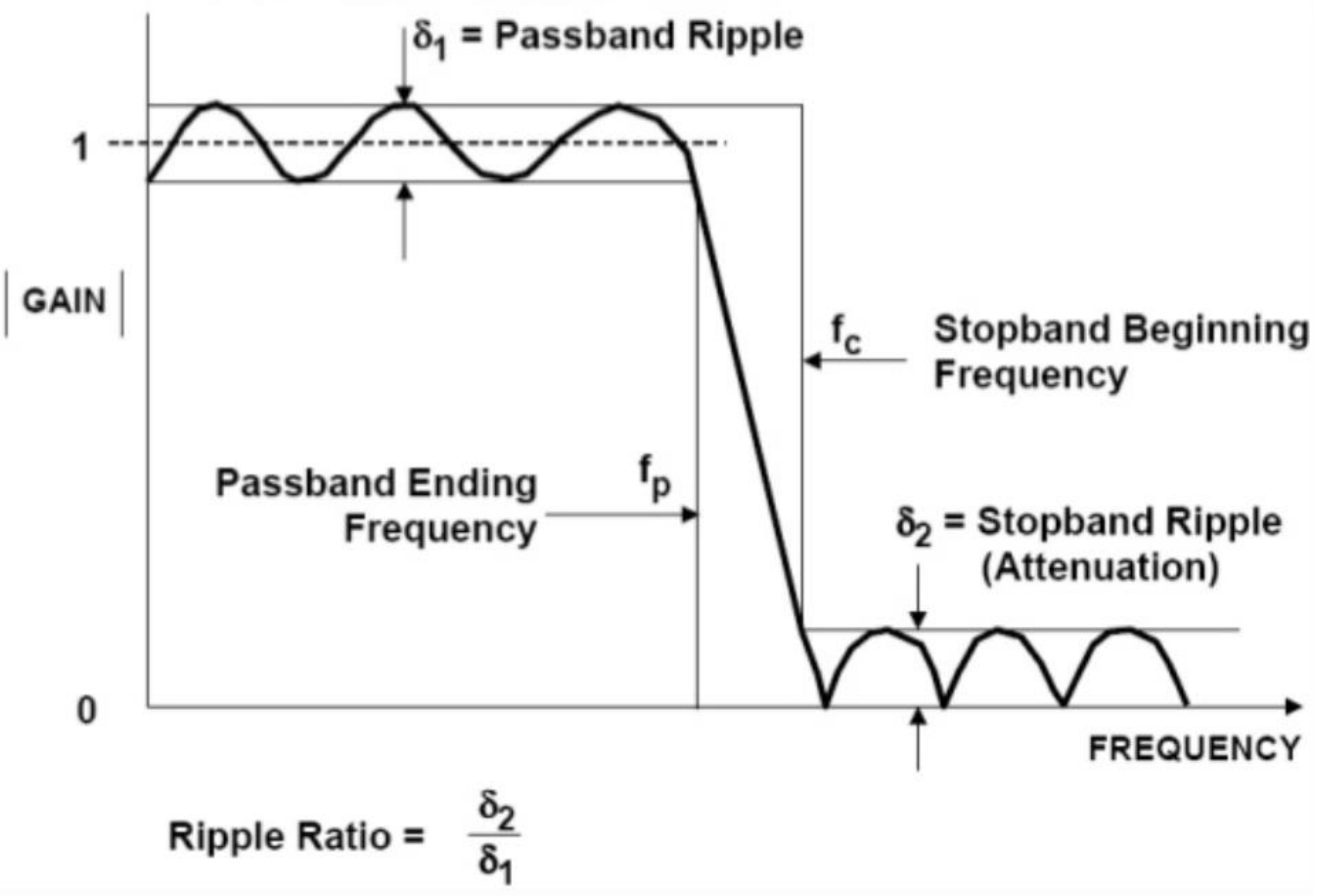
gnuplot source code for the gain plot

```
set title "Averaging 4 adjacent samples"
set key bottom left
set samples 10000

set ylabel "Gain (y/x)"
set logscale y
set yrange [0.009:1.1]

set xlabel "frequency [cycles/sample]"
set logscale x
set xrange [0.001:0.5]

# transfer function
H(z)= (1+z**(-1)+z**(-2)+z**(-3))/4

j=sqrt(-1)
amplitude(omega) = abs(H(exp(j*omega)))
phase(omega) = imag(log(H(exp(j*omega)))) * 180/pi  # phase in degrees

plot amplitude(2*pi*x) notitle
```

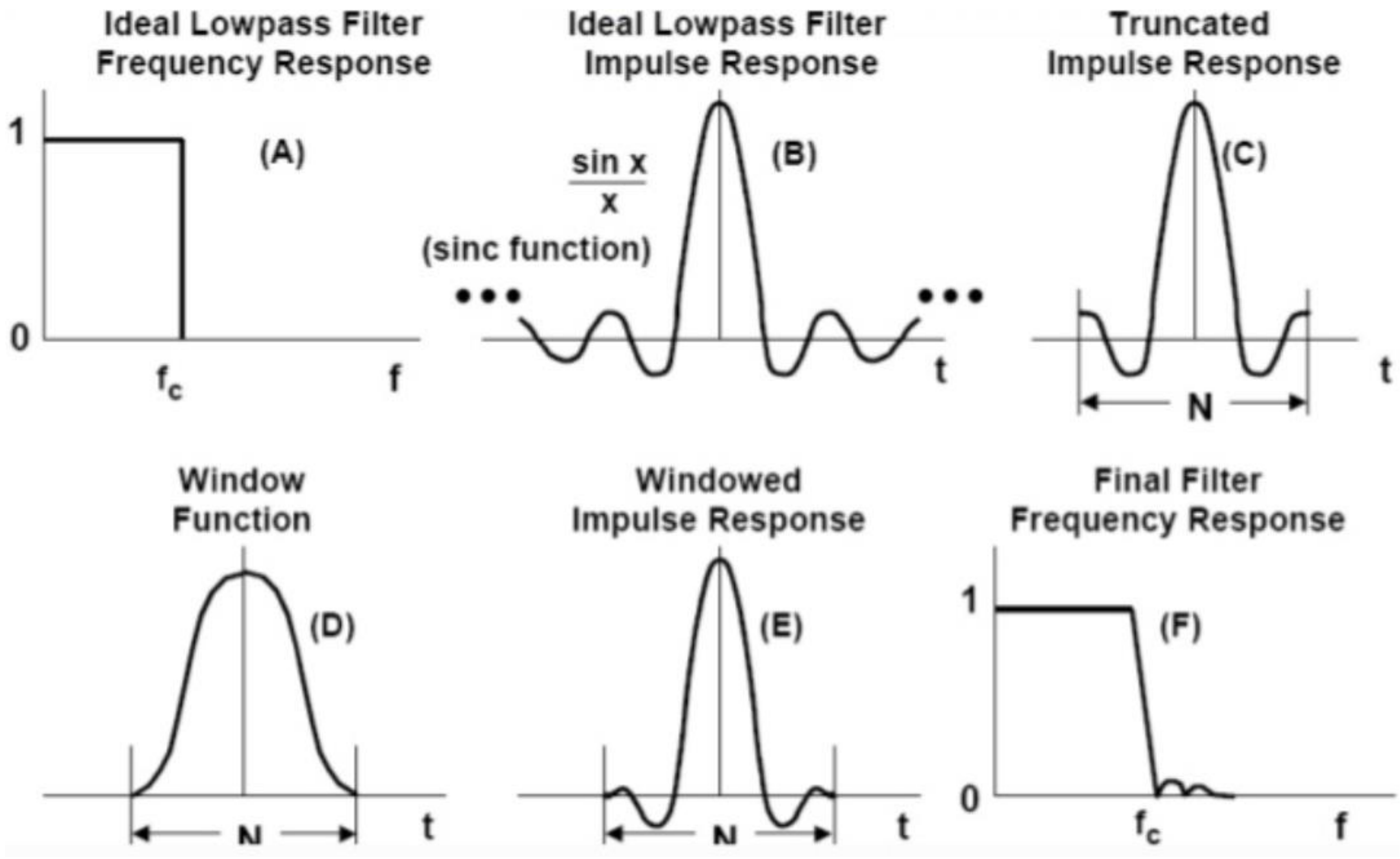




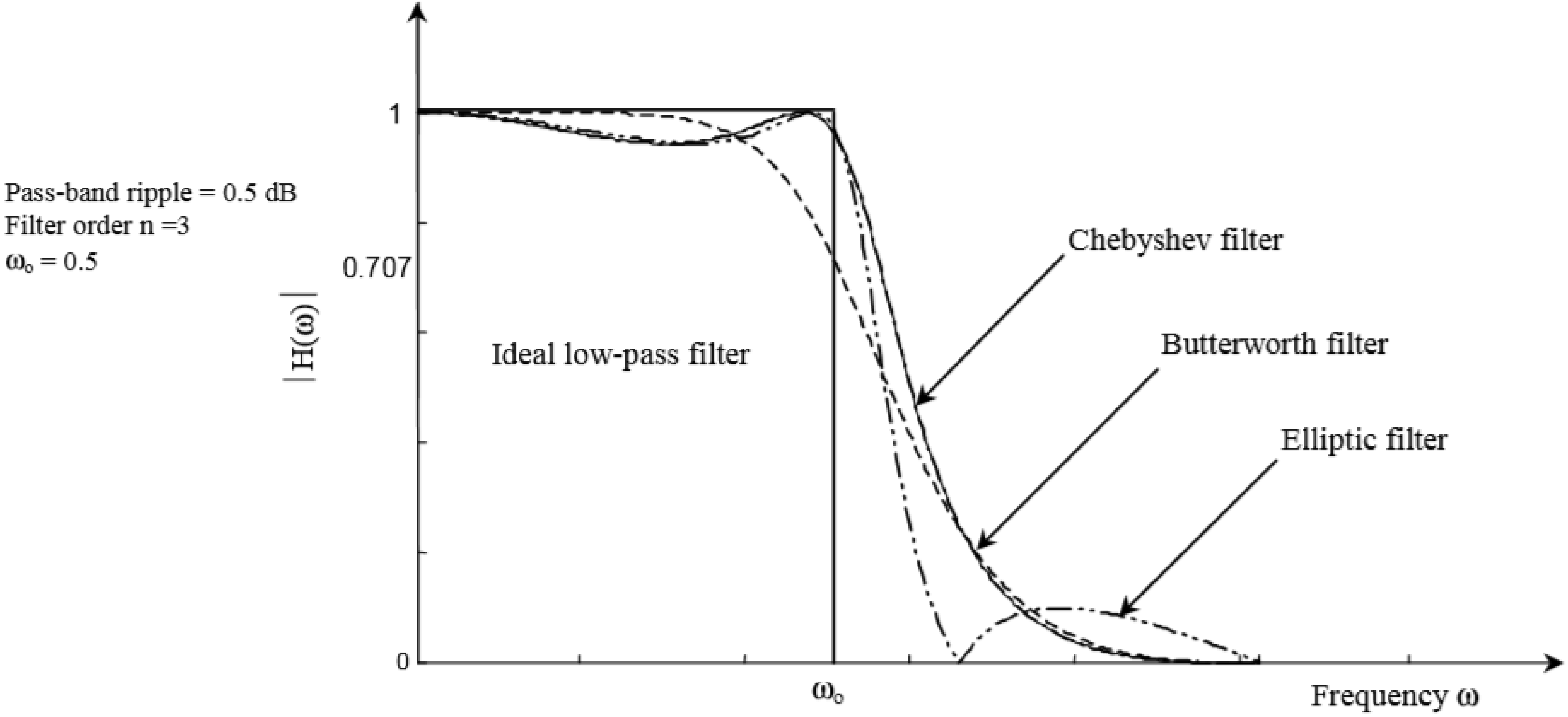
Characteristic Factors of Filters



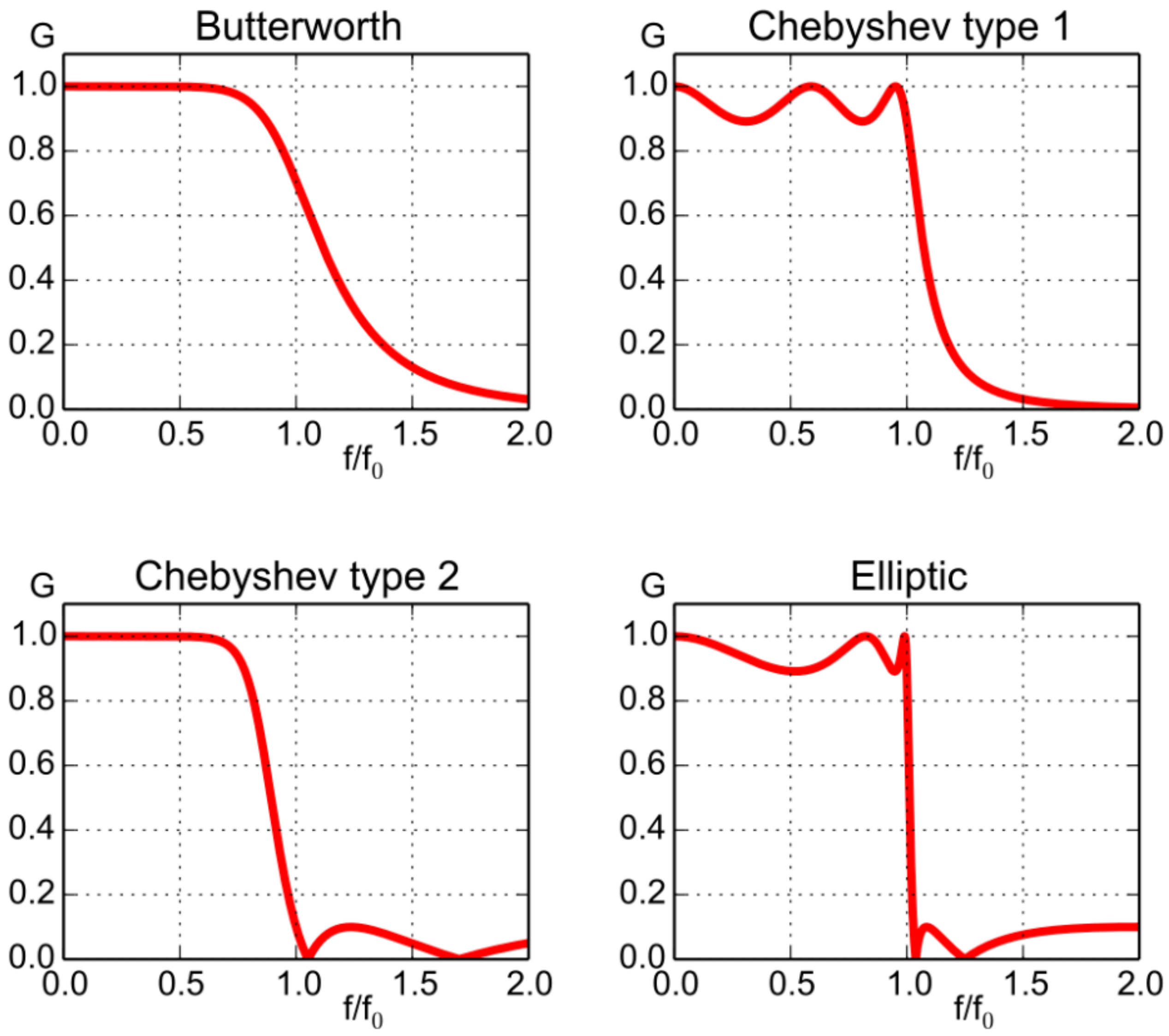
FIR Filter Design Using Windows Sinc Method



Typical frequency responses of various low-pass filters:



Comparison of 5th order Low Pass Filters:



Digital Filter Design with Bilinear Transformation

Analogue filters can be described by a frequency domain transfer function of the general form

$$H(s) = K \frac{(s - z_1)(s - z_2)(s - z_3)\cdots}{(s - p_1)(s - p_2)(s - p_3)\cdots}$$

The frequency response of the filter H(ω), can be obtained by replacing s = j ω. The complete response of the filter is then generated by varying ω between 0 and ∞.

$$H(\omega) = K \frac{(j\omega - z_1)(j\omega - z_2)(j\omega - z_3)\cdots}{(j\omega - p_1)(j\omega - p_2)(j\omega - p_3)\cdots}$$

Bilinear transformation: $s = \frac{2}{T}\left(\frac{1 - z^{-1}}{1 + z^{-1}}\right) \Rightarrow z = \frac{1 + {}^{sT}/_{2}}{1 - {}^{sT}/_{2}}$

$$H(z) = H_a(s)\big|_{s=\frac{2}{T}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)}$$

$$z = \frac{1 + (T_s/2)s}{1 - (T_s/2)s}$$ Setting $s = j\omega$ $$|z| = \left| \frac{1 + j\omega(T_s/2)}{1 - j\omega(T_s/2)} \right| = 1$$

$z = re^{j\Omega}$ and $s = \sigma + j\omega$

$$s = \frac{2}{T_s}\frac{z-1}{z+1} = \frac{2}{T_s}\frac{re^{j\Omega}-1}{re^{j\Omega}+1} = \frac{2}{T_s}\left(\frac{r^2-1}{1+r^2+2r\cos\Omega} + j\frac{2r\sin\Omega}{1+r^2+2r\cos\Omega}\right)$$

$$\sigma = \frac{2}{T_s}\frac{r^2-1}{1+r^2+2r\cos\Omega} \qquad \omega = \frac{2}{T_s}\frac{2r\sin\Omega}{1+r^2+2r\cos\Omega}$$

When r = 1, then σ=0 → $\omega = \frac{2}{T_s}\frac{\sin\Omega}{1+\cos\Omega} = \frac{2}{T_s}\tan\frac{\Omega}{2}$ or $\Omega = 2\tan^{-1}\frac{\omega T_s}{2}$

Verifies that $-\infty < \omega < \infty$ is mapped only into the range $-\pi \le \Omega \le \pi$.

If we apply bilinear transformation then s is replaced by: $F(z) = \dfrac{z-1}{z+1}$

Hence H(z) becomes
$$H(z) = K \frac{\left[\left(\frac{z-1}{z+1}\right) - z_1\right]\left[\left(\frac{z-1}{z+1}\right) - z_2\right]\left[\left(\frac{z-1}{z+1}\right) - z_3\right]\cdots}{\left[\left(\frac{z-1}{z+1}\right) - p_1\right]\left[\left(\frac{z-1}{z+1}\right) - p_2\right]\left[\left(\frac{z-1}{z+1}\right) - p_3\right]\cdots}$$

Frequency response of this (H(z))transfer function is obtained by substituting $z = e^{j\Omega}$ into H(z)

$$F(\Omega) = \frac{e^{j\Omega} - 1}{e^{j\Omega} + 1} = \frac{e^{j\Omega/2} - e^{-j\Omega/2}}{e^{j\Omega/2} + e^{-j\Omega/2}} = \frac{j2\sin(\Omega/2)}{2\cos(\Omega/2)} = j\tan\left(\frac{\Omega}{2}\right)$$

Example:

Transform $H_a(s) = \frac{s+1}{s^2+5s+6}$ into a digital filter for T=1

$$H(z) = H_a(s)\Big|_{s=\frac{2}{T}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)}$$

$$H(z) = H_a\left(\frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}\Big|_{T=1}\right) = H_a\left(2\frac{1-z^{-1}}{1+z^{-1}}\right)$$

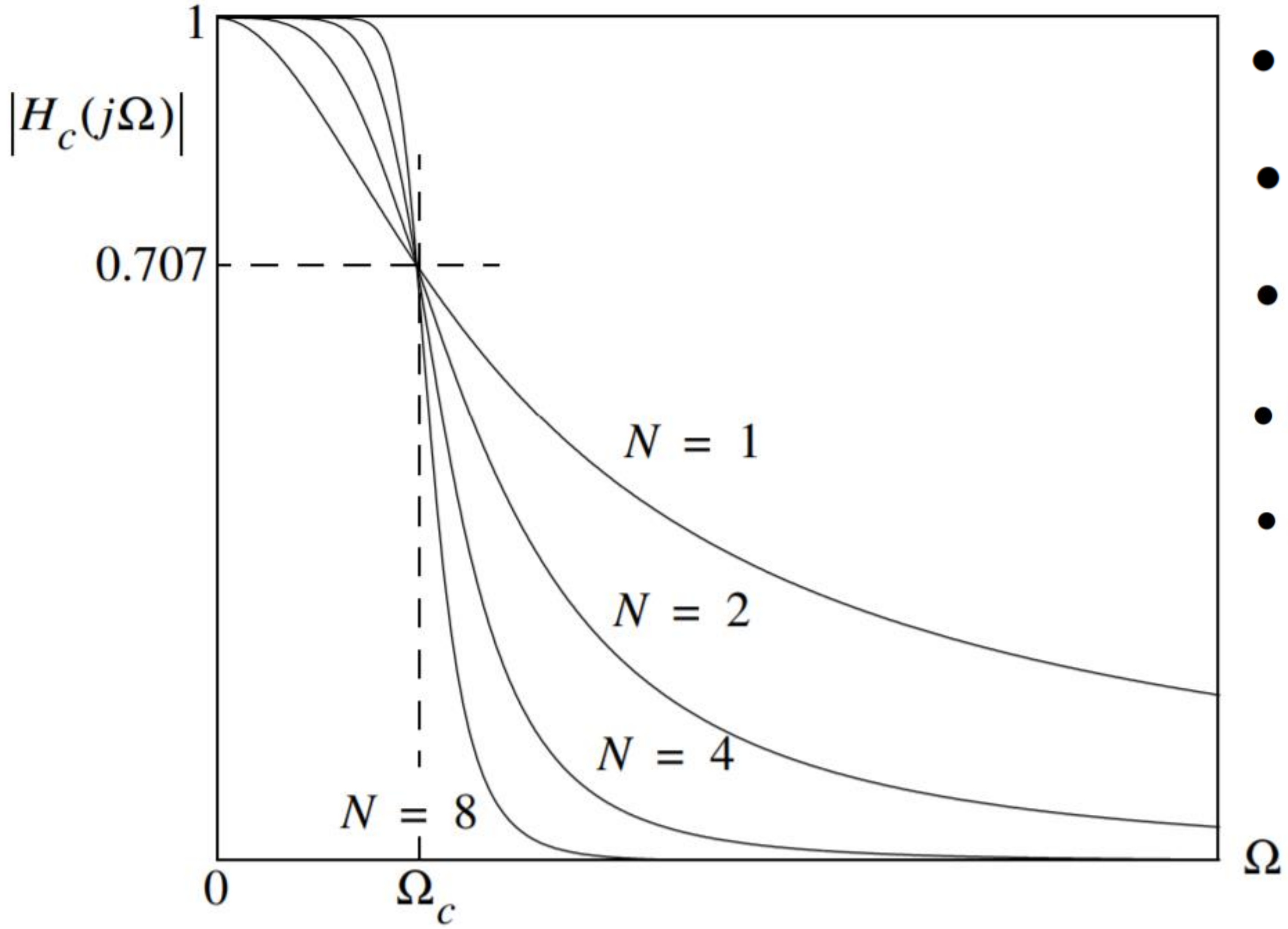
$$H(z) = \frac{3+2z^{-1}-z^{-2}}{20+4z^{-1}} = \frac{0.15+0.1z^{-1}-0.05z^{-2}}{1+0.2z^{-1}}$$

Exercise: Write the pseudo code for implementation of H(z)

Stephen Butterworth (1885–1958)

Magnitude response of Butterworth Filter: $$|H_c(j\Omega)|^2 = \frac{1}{1+\left(\frac{\Omega}{\Omega_c}\right)^{2N}}$$ **(1930)**

Butterworth Magnitude Response for order N = 1, 2, 4, and 8:



- $|H_c(j\Omega)|_{\Omega=0} = 1$, for all N
- $|H_c(j\Omega)|_{\Omega=\Omega_c} = 1/\sqrt{2}$, for all N
- $|H_c(j\Omega)|$ is monotone decreasing for all Ω
- For $\Omega > \Omega_c$ $|H_c(j\Omega)|_{dB}$ has slope $-20N$ dB/decade
- As $N \to \infty$ $|H_c(j\Omega)|$ approaches an ideal lowpass filter

From the form of $|H_c(j\Omega)|^2$ and the causality constraint, we can write

$$H_c(s) = \frac{1}{B_N(s)} = \frac{1}{\prod_{k=1}^{N}(s - s_k)}$$

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

$$\begin{aligned} s_{pk} &= (-1)^{1/(2N)} \cdot (j\omega_c) \\ &= \omega_c \exp\left\{ j\pi \left(\frac{1}{2} + \frac{1+2k}{2N} \right) \right\} \end{aligned}$$

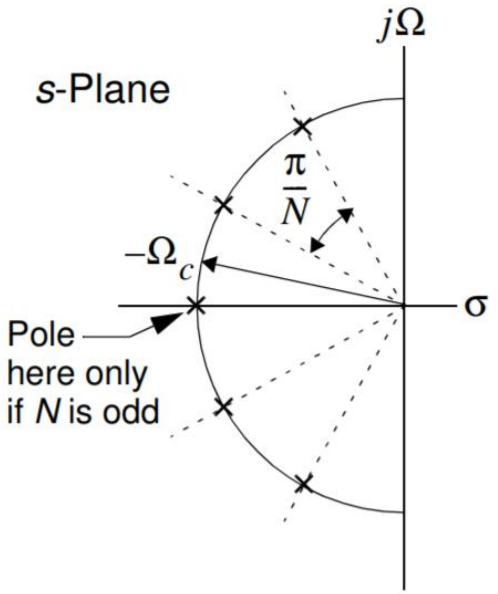
$$k = 0, 1, \ldots, (2N-1)$$

Butterworth pole locations

$B_N(s)$ is an Nth order Butterworth polynomial with roots given by

$$s_k = \Omega_c e^{j\pi[0.5-(2k-1)/(2N)]}, \quad k = 1, 2, \ldots, N$$

Note that $H_c(s)$ has N zeros at infinity and a pole on the negative real axis at $-\Omega_c$ if N is odd

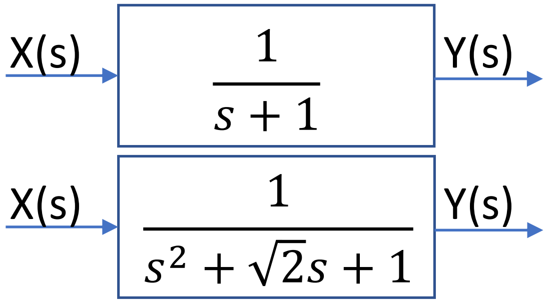


Standard and Factored form Butterworth Polynomials for $\Omega_c = 1$ rad/sec:

Standard Form

$$B_N(s) = a_N s^N + a_{N-1} s^{N-1} + \cdots + a_1 s + a_0$$

a_6	a_5	a_4	a_3	a_2	a_1	a_0	N
					1	1	1
				1	$\sqrt{2}$	1	2
			1	2	2	1	3
		1	2.613	3.414	2.613	1	4
	1	3.236	5.236	5.236	3.236	1	5
1	3.864	7.464	9.141	7.464	3.864	1	6



Quadratic Factored Form

$B_N(s)$	N
$s + 1$	1
$s^2 + \sqrt{2}s + 1$	2
$(s^2 + s + 1)(s + 1)$	3
$(s^2 + 0.76536s + 1)(s^2 + 1.84776s + 1)$	4
$(s + 1)(s^2 + 0.6180s + 1)(s^2 + 1.6180s + 1)$	5
$(s^2 + 0.5176s + 1)(s^2 + \sqrt{2}s + 1)(s^2 + 1.9318s + 1)$	6

Frequency response of this (H(z))transfer function is obtained by substituting $z = e^{j\Omega}$ into H(z)

$$F(\Omega) = \frac{e^{j\Omega} - 1}{e^{j\Omega} + 1} = \frac{e^{j\Omega/2} - e^{-j\Omega/2}}{e^{j\Omega/2} + e^{-j\Omega/2}} = \frac{j2\sin(\Omega/2)}{2\cos(\Omega/2)} = j\tan\left(\frac{\Omega}{2}\right)$$

$$H(\Omega) = K\frac{[j\tan(\Omega/2) - z_1][j\tan(\Omega/2) - z_2][j\tan(\Omega/2) - z_3]\cdots}{[j\tan(\Omega/2) - p_1][j\tan(\Omega/2) - p_2][j\tan(\Omega/2) - p_3]\cdots}$$

The function H(Ω) takes all values of the frequency response of the analogue filter, but compressed into the range $0 \le \Omega \le \pi$.

The "warping" effect of the tan function:

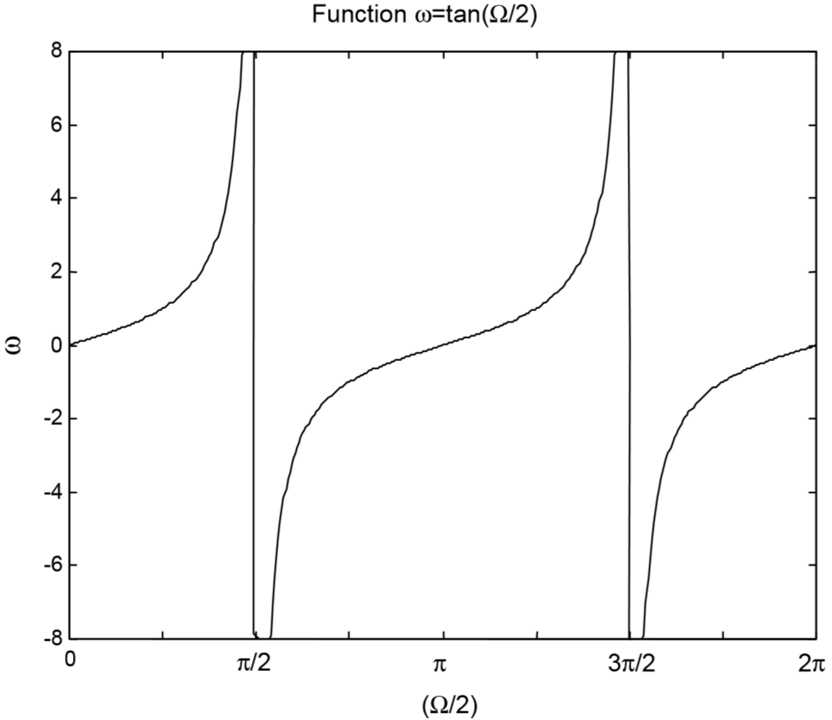
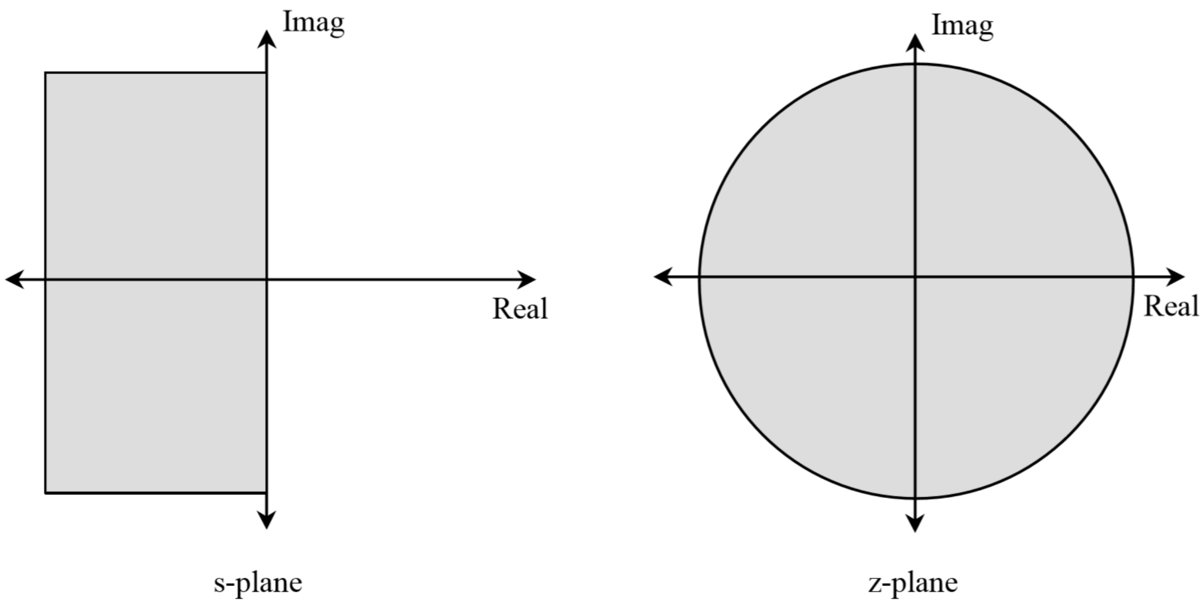


Illustration of s-plane to z-plane mapping using the bilinear z-transform:



If you wish to design a digital low-pass filter with a cut-off frequency Ω_c. We first transform this frequency to the analogue-domain cut-off frequency ω_{ac}, using the pre-warping relationship of

$$\omega_{ac} = k \cdot \tan\left(\frac{\Omega_c}{2}\right) \qquad k = 1 \quad or \quad \frac{2}{T}$$

then proceed to design the analogue filter using the corresponding cut-off frequency, obtained from the above equation. After the analogue filter has been transformed using the bilinear z-transform, the resulting digital filter will have its cutoff frequency in the correct place. Since pre-warping is performed in the beginning of the design procedure, and bilinear transformation is performed at the end, the value of k is unimportant.

IIR Filter Design Using the Bilinear z-transform:

1- Use the digital filter specification to determine a suitable normalized frequency-domain transfer function H(s).
2- Determine the cut-off frequency of the digital filter Ω_c.
3- Obtain the equivalent analogue filter cut-off frequency ω_{ac} using the pre-warping function
4- Denormalize the analogue filter by frequency scaling H(s), with one of the appropriate frequency transformations e.g. $s \Rightarrow s/\omega_{ac}$ etc.
5- Apply the bilinear z-transform to obtain the digital filter transfer function H(z) by replacing s with (z - 1)/(z + 1)

Frequency Transformations for band shaping:

The design of analogue filters other than low-pass is usually achieved by designing a low-pass filter of the desired class. This is accomplished by substituting the frequency-domain transfer function H(s) with one of the relevant frequency transformations listed below. Where ω_2 and ω_1 are the band-edge frequencies of the desired filter and are also positive parameters satisfying $\omega_2 > \omega_1$

Low-pass to *Low-pass* transformation: $$s \Rightarrow \frac{s}{\omega_{ac}}$$

Low-pass to *High-pass* transformation $$s \Rightarrow \frac{\omega_{ac}}{s}$$

Low-pass to *Band-pass* transformation $$s \Rightarrow \frac{s^2 + \omega_1\omega_2}{(\omega_2 - \omega_1)s}$$

Low-pass to *Band-stop* transformation $$s \Rightarrow \frac{(\omega_2 - \omega_1)s}{s^2 + \omega_1\omega_2}$$

Example:

Design a digital filter equivalent of a 2^{nd} order Butterworth low-pass filter with a cut-off frequency f_c = 100 Hz and a sampling frequency f_s = 1000 samples/sec. Derive the finite difference equation and draw the realisation structure of the filter. Given that the analogue prototype of the frequency-domain transfer function $H(s)$ for a Butterworth filter is:

$$H(s) = \frac{1}{s^2 + \sqrt{2} \cdot s + 1}$$

The normalized cut-off frequency of the digital filter is given by : $\Omega_c = \frac{2\pi f_c}{f_s} = \frac{2\pi 100}{1000} = 0.628$

Now determine the equivalent analogue filter cut-off frequency ω_{ac}, using the pre-warping function. Let K = 1

$$\omega_{ac} = K \cdot \tan\left(\frac{\Omega_c}{2}\right) = 1 \cdot \tan\left(\frac{0.628}{2}\right)$$

$$\omega_{ac} = 0.325 \quad rads/\sec$$

Denormalization of the Butterworth filter transfer function H(s) with the corresponding lowpass to low-pass frequency transformation:

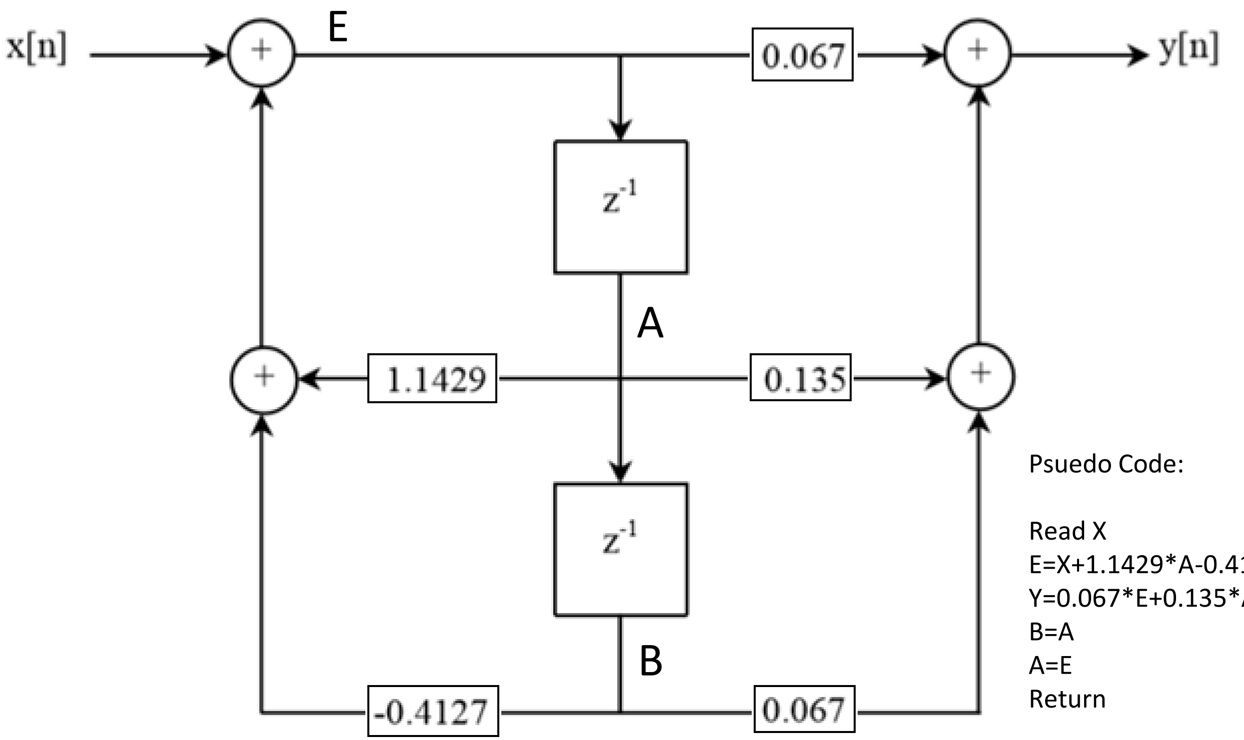
$$H(s)=\frac{1}{\left[\frac{s}{0.325}\right]^2+\sqrt{2}\cdot\left[\frac{s}{0.325}\right]+1}$$

$$s=\frac{z-1}{z+1}\equiv\frac{1-z^{-1}}{1+z^{-1}} \quad \rightarrow \quad H(z)=\frac{1}{\frac{1}{0.325^2}\cdot\left[\frac{1-z^{-1}}{1+z^{-1}}\right]^2+\frac{\sqrt{2}}{0.325}\cdot\left[\frac{1-z^{-1}}{1+z^{-1}}\right]+1}$$

$$H(z)=\frac{Y(z)}{X(z)}=\frac{0.067+0.135z^{-1}+0.067z^{-2}}{1-1.1429z^{-1}+0.4127z^{-2}}$$

$$y(n)=1.1429y(n-1)-0.4127y(n-2)+0.067x(n)+0.135x(n-1)+0.067x(n-2)$$

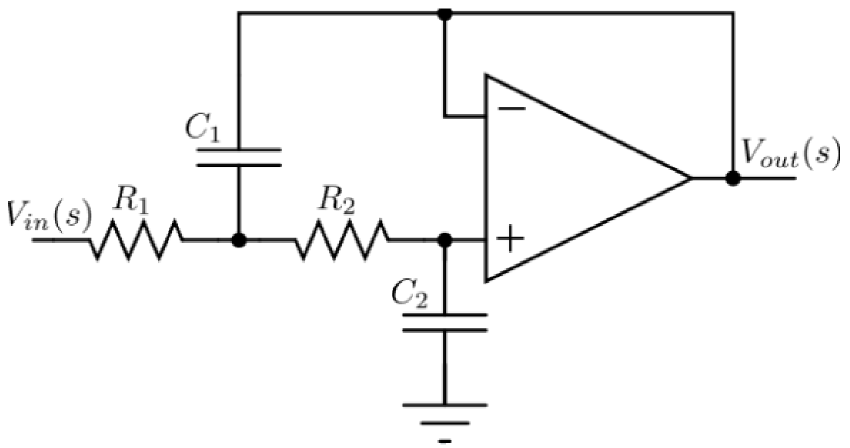
Direct realization of a 2nd order Butterworth equivalent filter



Psuedo Code:

Read X
E=X+1.1429*A-0.4127*B
Y=0.067*E+0.135*A+0.067*B
B=A
A=E
Return

Analog circuit equivalent of the 2nd order Butterworth filtering system:



$$H(s)=\frac{V_{out}(s)}{V_{in}(s)}=\frac{1}{1+C_2(R_1+R_2)s+C_1C_2R_1R_2s^2}$$

Example:

Design a digital low-pass Butterworth filter with a 3dB cut-off frequency of 2kHz and minimum attenuation of 30dB at 4.25kHz for a sampling rate of 10kHz.

$f_s = 10^4$ Hz; hence $T = 10^{-4}$ sec.
$\omega_{d1} = 2\pi \times 2 \times 10^3$ rads/sec; $\omega_{d2} = 2\pi \times 4.25 \times 10^3$ rads/sec.

$$\omega_{a1} = \frac{2}{T}\tan\frac{\omega_{d1}T}{2} = 2 \times 10^4 \tan 0.2\pi = 0.72654 \times \frac{2}{T} = 14.53 \times 10^3 \text{ rads/sec}$$

$$\omega_{a2} = \frac{2}{T}\tan\frac{\omega_{d2}T}{2} = 2 \times 10^4 \tan 0.425\pi = 83.31 \times 10^3 \text{ rads/sec}$$

$$n \approx \frac{log A}{log \frac{\omega_a}{\omega_c}} = \frac{log_{10} 10\sqrt{10}}{log_{10}\frac{83.31}{14.53}} = 1.98$$

2nd order Butterworth filter characteristics: $G(s) = \frac{\omega_c^2}{s^2 + s\sqrt{2}\omega_c + \omega_c^2}$

Substitute $\omega_{a1} = \omega_c = 0.72654\frac{2}{T}$ in above equation and use the bilinear transformation $s = \frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}$:

$$G(z) = \frac{(\frac{2}{T})^2 0.72654^2}{(\frac{2}{T})^2(\frac{1-z^{-1}}{1+z^{-1}})^2 + \sqrt{2}(\frac{2}{T})^2 0.72654(\frac{1-z^{-1}}{1+z^{-1}}) + (\frac{2}{T})^2(0.72654)^2}$$

$$G(z) = \frac{0.52786}{(\frac{1-z^{-1}}{1+z^{-1}})^2 + 1.02749(\frac{1-z^{-1}}{1+z^{-1}}) + 0.52786}$$

$$G(z) = 0.52786 \frac{1 + 2z^{-1} + z^{-2}}{2.55535 - 0.94427z^{-1} + 0.50038z^{-2}}$$

$$G(z) = 0.20657 \frac{1 + 2z^{-1} + z^{-2}}{1 - 0.36953z^{-1} + 0.19582z^{-2}}$$

$$\boxed{y[k] = 0.20657x[k] + 0.41314x[k-1] + 0.20657x[k-2] + 0.36953y[k-1] - 0.19582y[k-2]}$$

Validation of the frequency response:

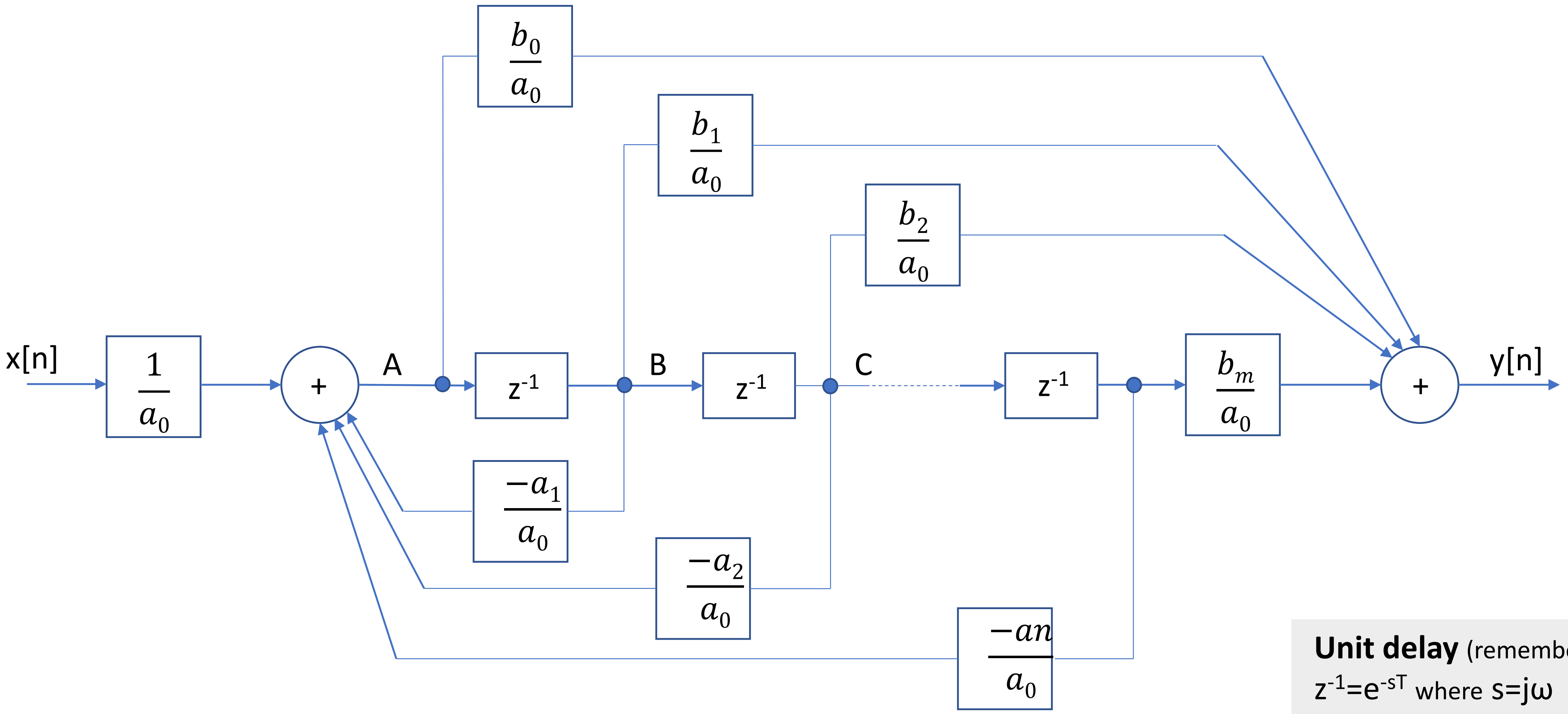
a) DC Gain: $$G(z) = 0.20657 \frac{z^2 + 2z + 1}{z^2 - 0.36953z + 0.19582}$$ $\omega_d T = 0 \rightarrow z = 1$ which gives $|G(1)| = 1$

b) -3dB cut-off frequency @ΩT=0.4π:

$$G(e^{0.4\pi j}) = 0.20657 \frac{\cos 0.8\pi + j \sin 0.8\pi + 2(\cos 0.4\pi + j \sin 0.4\pi) + 1}{\cos 0.8\pi + j \sin 0.8\pi - 0.36953(\cos 0.4\pi + j \sin 0.4\pi) + 0.19582} = 0.20657 \frac{0.8092 + 2.4899j}{-0.72739 + 0.23634j}$$

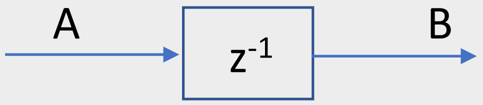
that yields |G|=0.707

Reminder: Direct Programming (canonical form) of $H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + \dots\dots + b_m z^{-m}}{a_0 + a_1 z^{-1} + \dots\dots + a_n z^{-n}}$



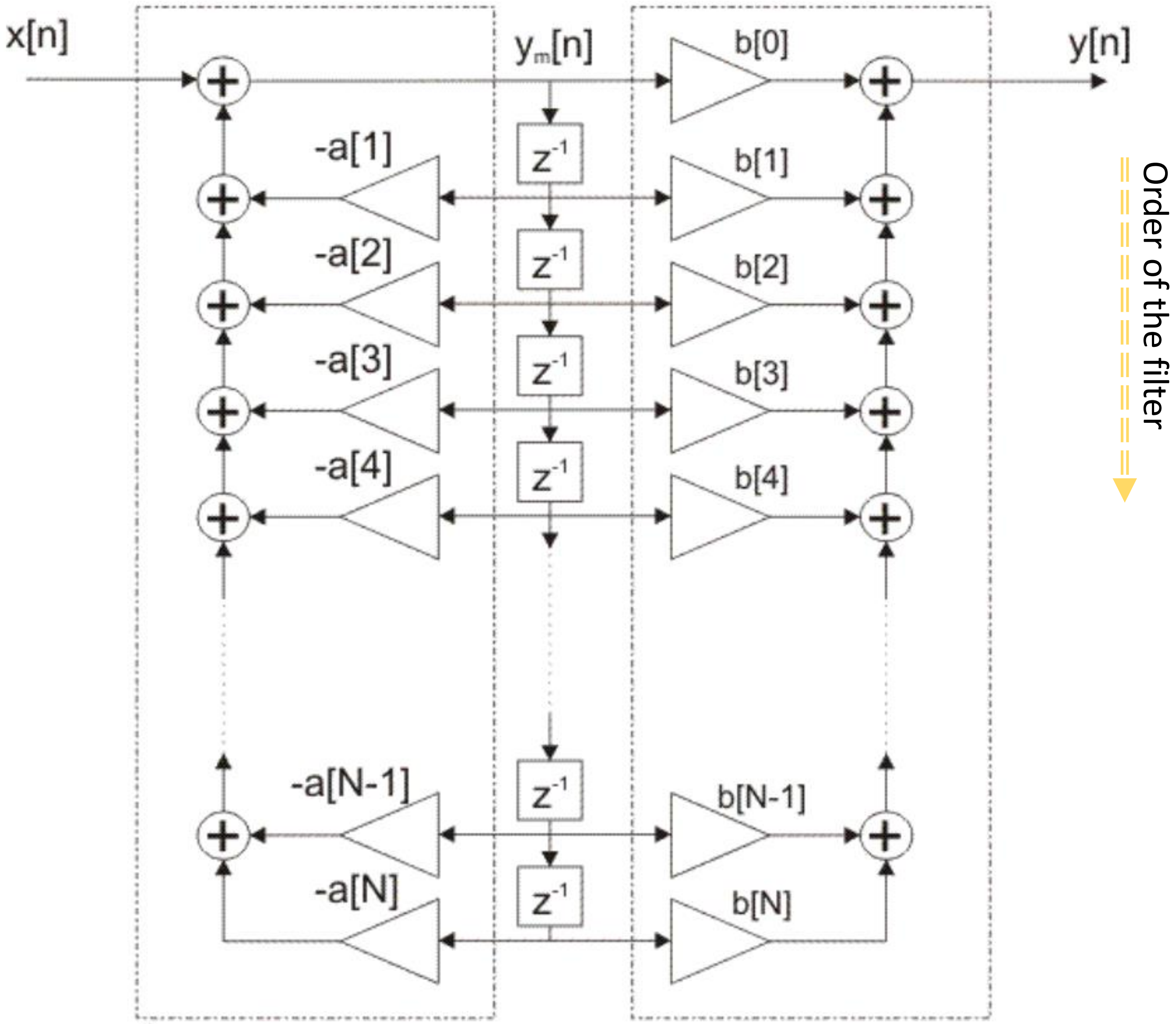
Unit delay (remember):

$z^{-1} = e^{-sT}$ where $s = j\omega$



$B = A \cdot z^{-1} \Rightarrow B(k) = A(k-1)$

Block diagram of direct Butterworth realization in canonical form:



$\frac{1}{A(z)}$ - recursive part

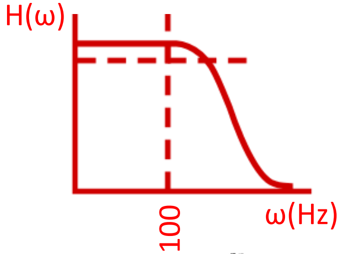
$B(z)$ - non-recursive part

2^{nd} order Butterworth 100Hz Lowpass Filter with DFT Analysis in Python

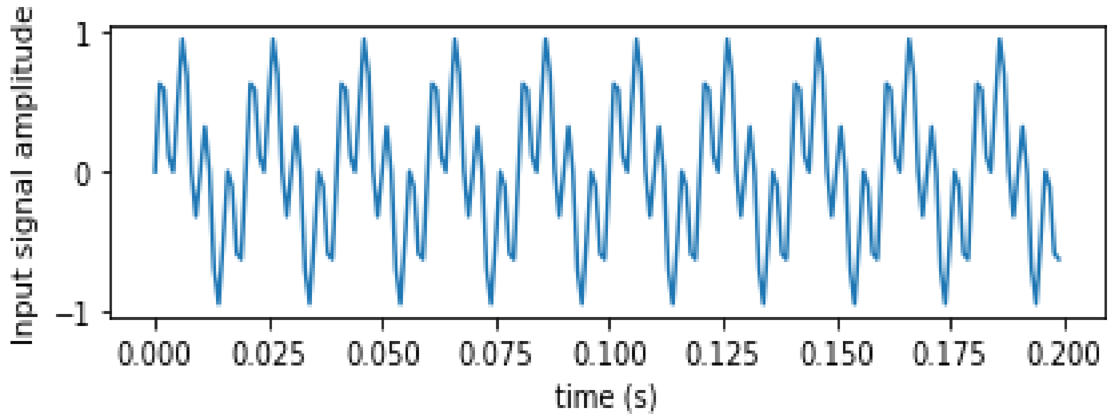
(Based on difference equations only)

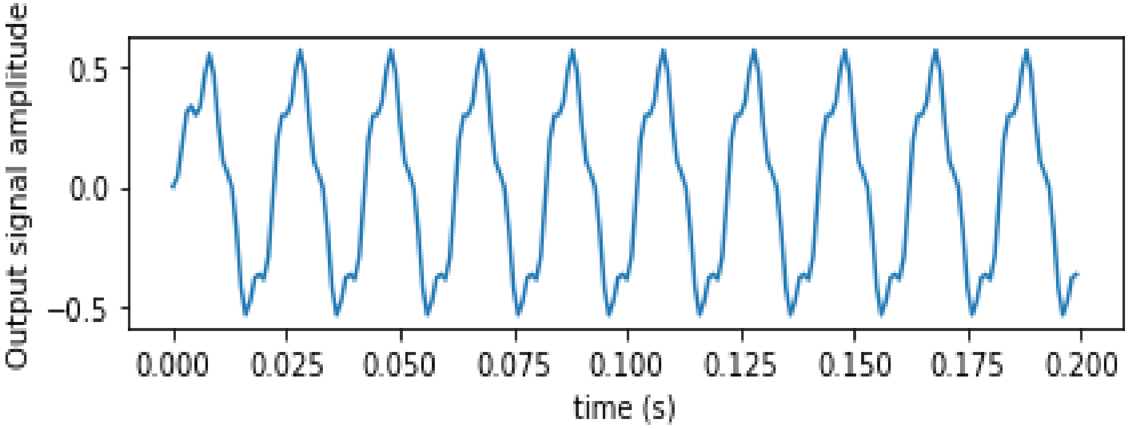
```
import numpy as np
import matplotlib.pyplot as plt

fs = 1000
time = np.arange(0,1,1/fs)

def DFT(x): # To calculate DFT of a 1D real-valued signal x

   N = len(x)
   n = np.arange(N)
   k = n.reshape((N, 1))
   e = np.exp(-2j * np.pi * k * n / N)
   X = np.dot(e, x)
   return X

sound=[]
for i in range (0,len(time)):
   sound.append(
              0.5*np.sin(2*np.pi*(50)*i/fs)
             +0.5*np.sin(2*np.pi*(200)*i/fs)
             #+0.25*(np.random.rand()-0.5)
             )

X = DFT(sound)

# calculate the frequency
N = len(X)
n = np.arange(N)
T = N/fs
freq = n/T

n_oneside = N//2
# get the one side frequency
f_oneside = freq[:n_oneside]

# normalize the amplitude
X_oneside =X[:n_oneside]/n_oneside

plt.figure(figsize = (12, 4))
plt.subplot(221)
plt.xlabel('time (s)')
plt.ylabel('Input signal amplitude')
plt.plot(time[0:200],sound[0:200])
plt.subplot(223)
plt.stem(f_oneside, abs(X_oneside), 'b', \
       markerfmt=" ", basefmt="-b")
plt.xlabel('Freq (Hz)')
plt.ylabel('DFT of the Input')

# 2nd order Butterworth filter with 100Hz cutoff frequency
A=0
B=0
fsound=[]
for i in range (0,N):
   X=sound[i]
   E=X+1.1429*A-0.4127*B
   Y=0.067*E+0.135*A+0.067*B
   fsound.append(Y)
   B=A
   A=E

plt.subplot(222)
plt.xlabel('time (s)')
plt.ylabel('Output signal amplitude')
plt.plot(time[0:200],fsound[0:200])

X1 = DFT(fsound)
# calculate the frequency
N1= len(X1)
n1 = np.arange(N1)
T1 = N1/fs
freq1 = n1/T1
n_oneside1 = N1//2
# get the one side frequency
f_oneside1 = freq1[:n_oneside1]

# normalize the amplitude
X_oneside1 =X1[:n_oneside1]/n_oneside1

plt.subplot(224)
plt.stem(f_oneside1, abs(X_oneside1), 'b', \
       markerfmt=" ", basefmt="-b")
plt.xlabel('Freq (Hz)')
plt.ylabel('DFT of the Output')

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

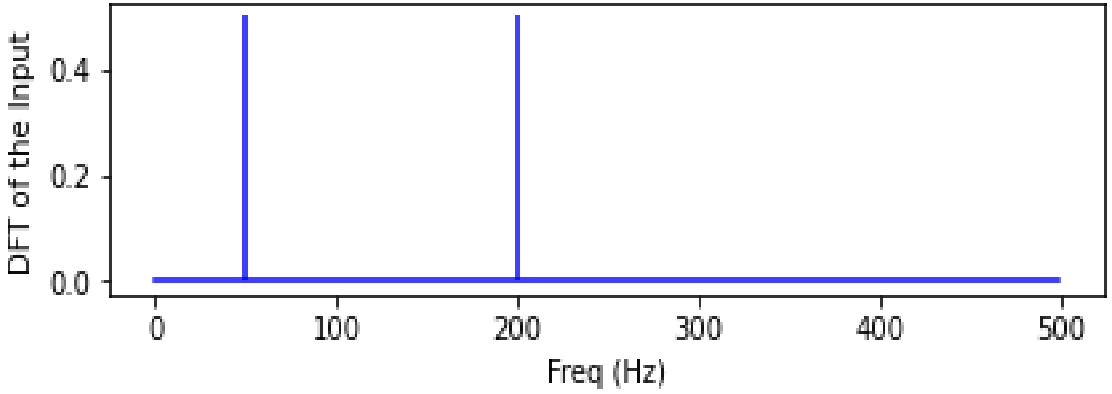
Input Signal:

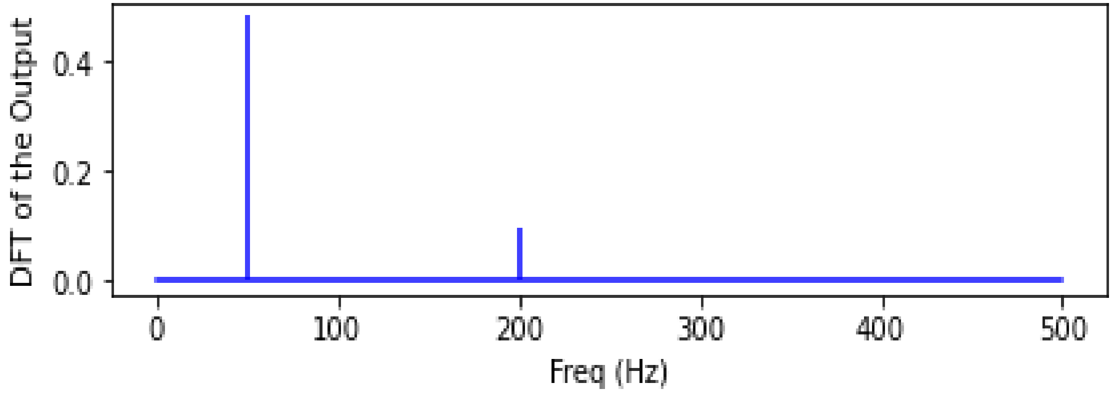


Output Signal:









Python Code For Butterworth Filter Application Example

(By using Scipy library)

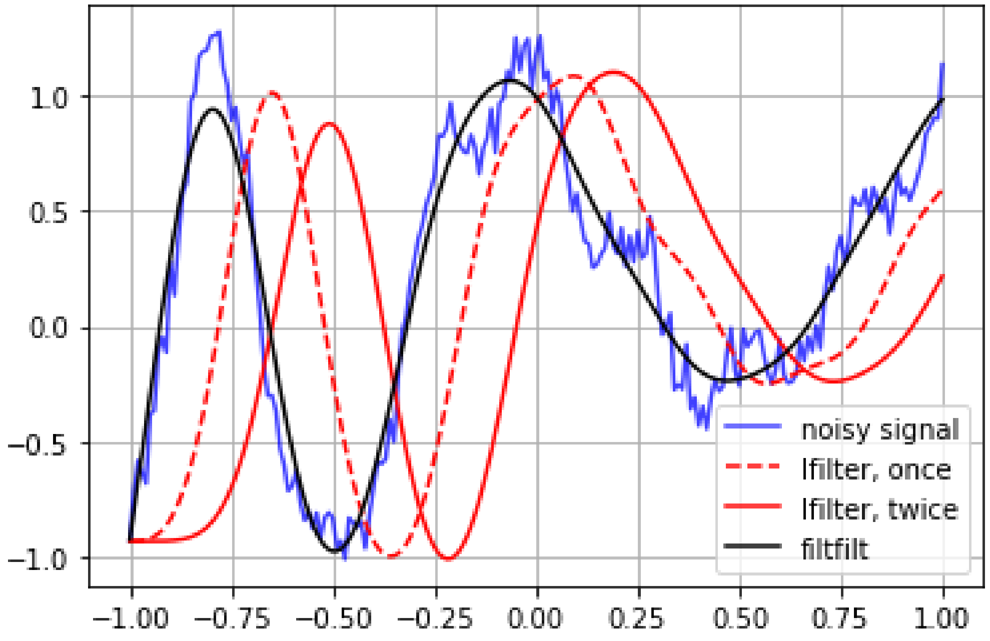
```
from scipy import signal
import matplotlib.pyplot as plt
import numpy as np
t = np.linspace(-1, 1, 201)
x = (np.sin(2*np.pi*0.75*t*(1-t) + 2.1) +
    0.1*np.sin(2*np.pi*1.25*t + 1) +
    0.18*np.cos(2*np.pi*3.85*t))
xn = x + np.random.randn(len(t)) * 0.08

b, a = signal.butter(3, 0.05)

zi = signal.lfilter_zi(b, a)
z, _ = signal.lfilter(b, a, xn, zi=zi*xn[0])

z2, _ = signal.lfilter(b, a, z, zi=zi*z[0])

y = signal.filtfilt(b, a, xn)

plt.figure
plt.plot(t, xn, 'b', alpha=0.75)
plt.plot(t, z, 'r--', t, z2, 'r', t, y, 'k')
plt.legend(('noisy signal', 'lfilter, once', 'lfilter, twice',
        'filtfilt'), loc='best')
plt.grid(True)
plt.show()
```

Noisy signal

Normalized cutoff frequency wn=fc/(fs/2)

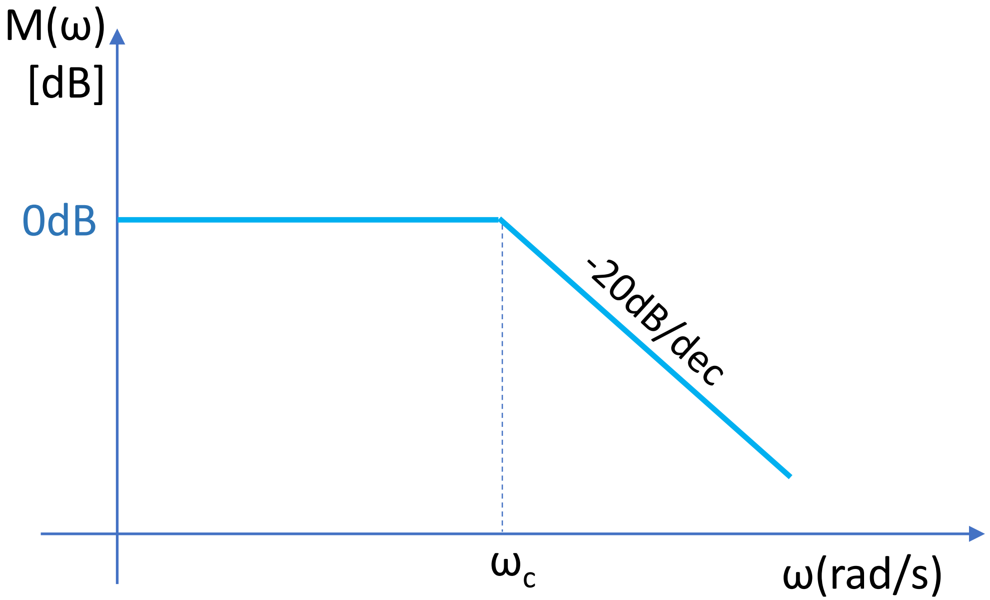
Create an order 3 lowpass butterworth filter

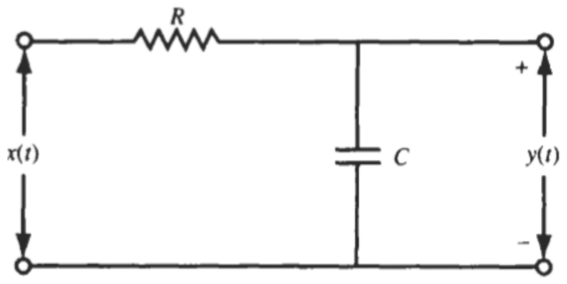
Apply the filter to xn. Use lfilter_zi to choose the initial condition of the filter



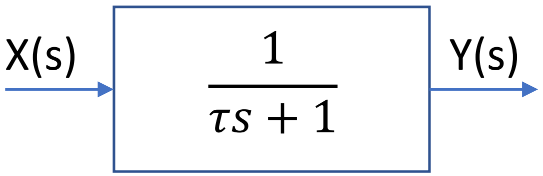
Example:

Design a low-pass discrete-time filter by the bilinear transformation method such that its 3-dB bandwidth is π/4.





$$\omega_{3\text{ dB}} = 1/RC$$



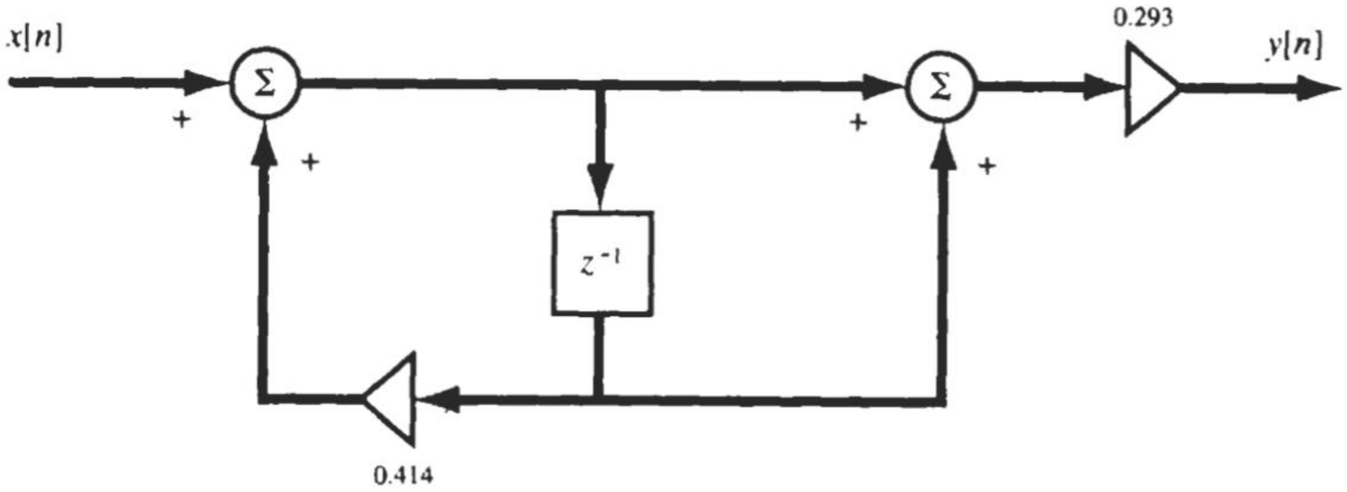
τ=RC

$\Omega_{3\text{ dB}} = \pi/4$ corresponds to $\quad \omega_{3\text{ dB}} = \frac{2}{T_s}\tan\frac{\Omega_{3\text{ dB}}}{2} = \frac{2}{T_s}\tan\frac{\pi}{8} = \frac{0.828}{T_s}$

$$H_c(s) = \frac{0.828/T_s}{s + 0.828/T_s}$$

Bilinear transform→ $$H_d(z) = \frac{0.828/T_s}{\frac{2}{T_s}\frac{1-z^{-1}}{1+z^{-1}} + \frac{0.828}{T_s}} = \frac{0.293(1+z^{-1})}{1-0.414z^{-1}}$$

Simulation of an *RC* filter by the bilinear transformation method.



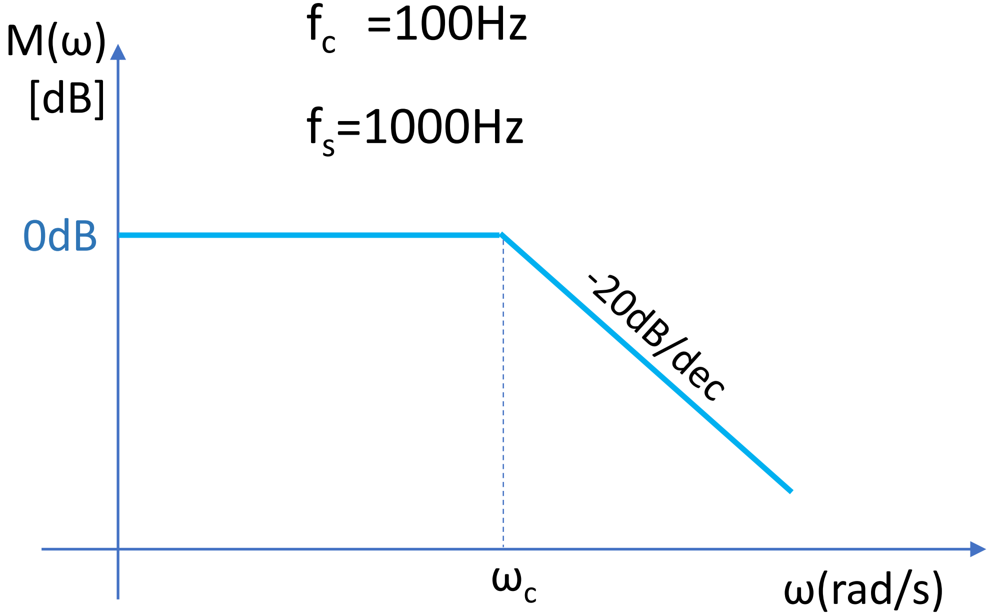
The frequency response of the discrete-time filter is $$H_d(\Omega) = \frac{0.293(1+e^{-j\Omega})}{1-0.414e^{-j\Omega}}$$

At $\Omega = 0, H_d(0) = 1$, and at $\Omega = \pi/4, |H_d(\pi/4)| = 0.707 = 1/\sqrt{2}$ which is the desired response

Exercise:

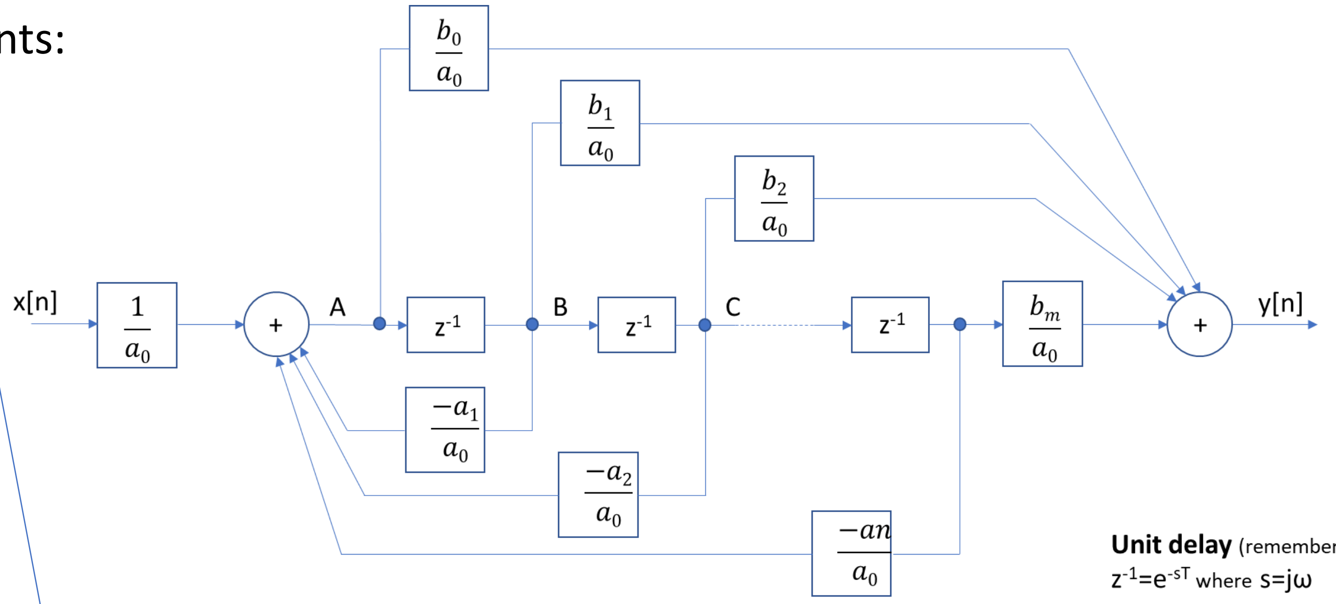
f_c =100Hz

f_s=1000Hz



a) Find the transfer function of 1st order Butterworth filter
b) Draw the discrete time equivalent system in terms of unit delays
c) Write the pseudo code for implementation of this digital LPF
d) Verify H(Ω) at DC and cut-off frequency

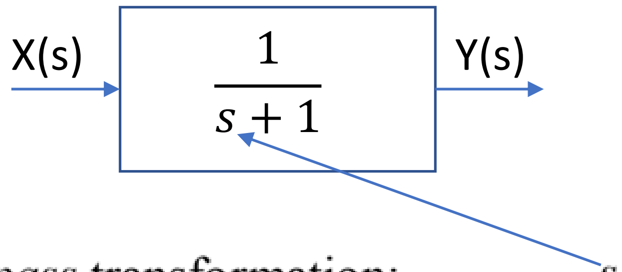
Hints:



Unit delay (remember):
$z^{-1}=e^{-sT}$ where $s=j\omega$

$$H(z) = H_a(s)\big|_{s=\frac{2}{T}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)}$$

$$\omega_{3\,\mathrm{dB}} = \frac{2}{T_*}\tan\frac{\Omega_{3\,\mathrm{dB}}}{2}$$



Low-pass to *Low-pass* transformation: $s \Rightarrow \dfrac{s}{\omega_{ac}}$