SEC1315 Digital Signal Processing and Its Applications

UNIT-1

I. Introduction

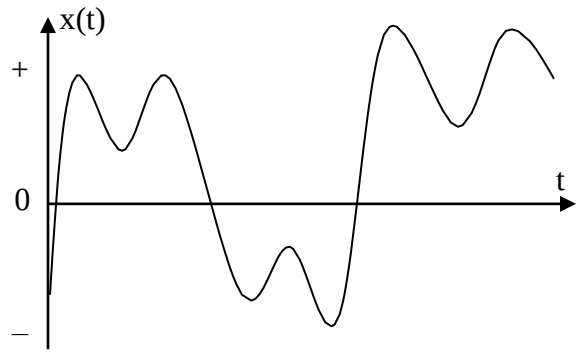
Introduction to Digital Signal Processing: Discrete time signals & sequences, Linear shift invariant systems, Stability, and Causality, Linear constant coefficient difference equations, Frequency domain representation of discrete time signals and systems.

Contents:

Sampling theory
Discrete-time signals
Transformation of the independent variable
Discrete-time systems
Linear constant coefficient difference equations
Fourier analysis of discrete-time signals and systems
Frequency response of discrete-time system
Properties of the discrete-time Fourier transform (DTFT)

Sampling theory

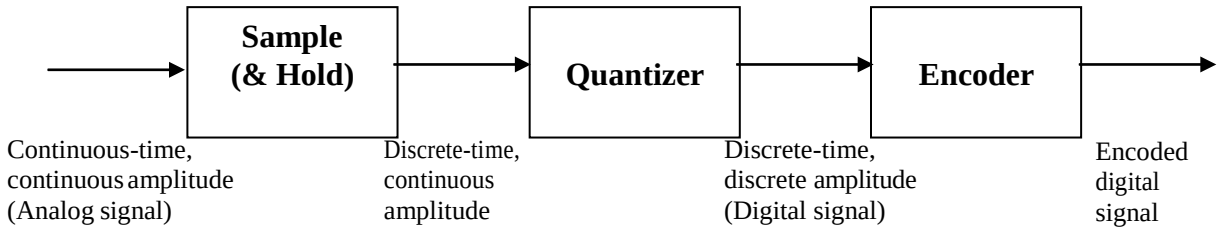
Illustrative example A continuous-time random signal is shown. Based on this several important concepts are shown below. The signal is a continuous-time signal with continuous amplitude. Such a signal is also called an **analog signal**.



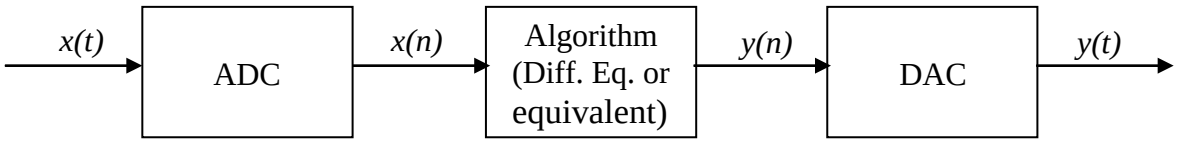
	x(t)‡										
4	8										
3	7										
2	6										
1	5										
0	4										‹Time
–1	3										
–2	2										
–3	1										
–4	0										
nT‹		0	1T	2T	3T	4T	5T	6T	7T	8T	‹Time
n‹		0	1	2	3	4	5	6	7	8	
x(n)‹		5.5	2.8	3.8	5.3	1.5	4.6	8.4	6.9	7.3	›Sampled signal. Discrete-time signal – time is discrete, amplitude is continuous.
		5	2	3	5	1	4	**7**	6	7	›Quantized. Quantization noise (error). Digital signal – both time and amplitude are discrete.
		101	010	011	101	001	100	111	110	111	Encoded to 3 bits/sample.
								‡Note this particular point exhibits saturation (out of range). Rounded down to 7, *not 8*.			

If we were to represent every sample value with infinite precision (for example, x(1) = 2.8--, instead of being approximated as 2 or 3) then we would need registers and memory words of arbitrarily large size. However, owing to a finite word length we round off the sample values (in this case x(1) = 2.8-- will be rounded to 2). This introduces quantization noise or error.

The procedure of generating a discrete-time signal from an analog signal is shown in the following block diagram. In the digital signal processing course we are mostly dealing with *discrete–time* rather than *digital* signals and systems, the latter being a subset of the former.



The three boxes shown above can be represented by an analog to digital converter (ADC). A complete digital signal processing (DSP) system consists of an ADC, a DSP algorithm (e.g., a difference equation) and a digital to analog converter (DAC) shown below.



As the name implies discrete-time signals are defined only at discrete instants of time. Discrete-time signals can arise by sampling analog signals such as a voice signal or a temperature signal in telemetry. Discrete-time signals may also arise naturally, e.g., the number of cars sold on a specific day in a year, or the closing DJIA figure for a specific day of the year.

AT&T's *T1* Stream The voice signal is band limited to 3.3 kHz, sampled at 8000 Hz (8000 samples per second), quantized and encoded into 8 bits per sample. Twenty four such voice channels are combined to form the *T1* stream or signal.

$$\text{Sampling interval} = \frac{1}{8000\ Hz} = 0.125 \text{ msec.}$$

$$\text{Bit rate for each channel} = 8000\ \frac{samples}{\text{sec}} \times 8\ \frac{bits}{sample} = 64000 \text{ bits/sec.}$$

Bit rate for T1 = 64000 bits/sec per channel x 24 channels = 1 544 000 bits/sec.

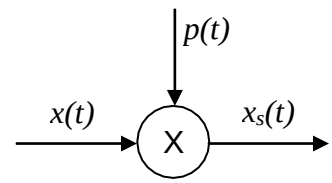
Commercial examples CD, Super Audio CD (SACD), DVD Audio (DVD-A), Digital audio broadcasting - 32 kHz, and Digital audio tape (DAT) - 48 kHz.

Commercial Audio Examples

	CD	**Super Audio CD (SACD)**	**DVD-Audio (DVD-A)**
Sampling Rate	**44.1 kHz**	**2.8224 MHz**	**44.1, 88.2 or 48, 96, 192 kHz**
Coding	**16-bit PCM per sample** **With 2 channels the bit rate is 1.4112 Mbits/sec, but additional error control bits etc., raise it to 4.3218 Mbits/sec.**	**1-bit DSD (Direct Stream Digital – like Delta modulation)**	**12-, 20-, 24-bit**

Pulse-train sampling For pulse-train sampling of the signal $x(t)$ by the rectangular pulse-train $p(t)$ resulting in the sampled signal $x_s(t)$, we have

$$x_s(t) = x(t)\, p(t)$$



The Fourier series of $p(t)$ is given by $p(t) = \sum_{n=-\infty}^{\infty} C_n e^{jn2\pi F_s t}$,

where $F_s = 1/T$ and the Fourier coefficients, are

$$C_n = \frac{1}{T}\int_{-T/2}^{T/2} p(t) e^{-jn2\pi F_s t} dt$$

Thus

$$x_s(t) = \sum_{n=-\infty}^{\infty} C_n x(t) e^{jn2\pi F_s t}$$

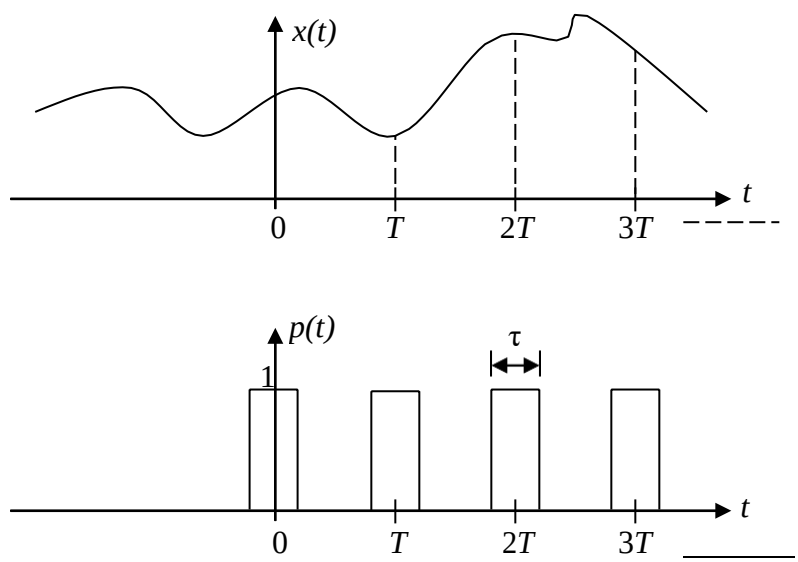
The Fourier spectrum of $x_s(t)$ is given by

$$X_s(F) = \int_{-\infty}^{\infty} x_s(t) e^{-j2\pi F t} dt = \int_{-\infty}^{\infty}\sum_{-\infty}^{\infty} C_n x(t) e^{jn2\pi F_s t} e^{-j2\pi F t} dt$$

Interchanging the order of integration and summation yields the ***aliasing formula***

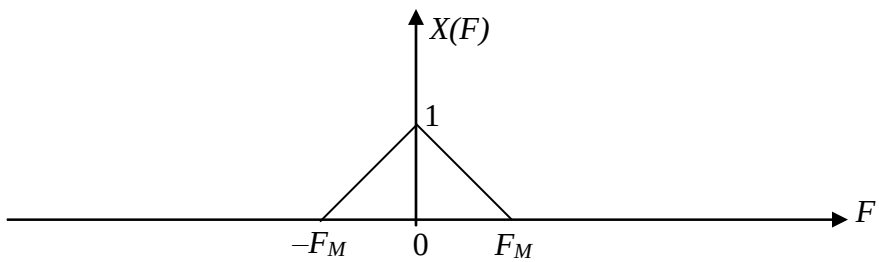
$$X_s(F) = \sum_{n=-\infty}^{\infty} C_n \int_{-\infty}^{\infty} x(t)\, e^{-j2\pi(F-nF_s)t} dt = \sum_{n=-\infty} C_n X(F - nF_s)$$

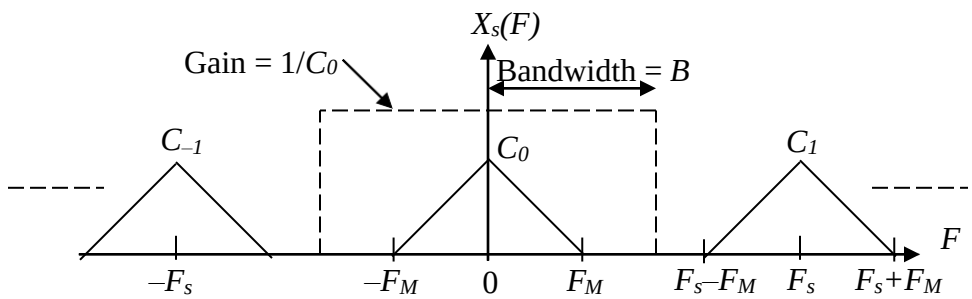
$$= \ldots + C_{-2} X(F + 2F_s) + C_{-1} X(F + 1F_s) + C_0 X(F) + C_1 X(F - 1F_s) + C_2 X(F - 2F_s) + \ldots$$



The sampled signal spectrum $X_s(F)$ is sketched below. For convenience of illustration we have assumed the base band spectrum, $X(F)$, to be real valued; the maximum value of $|X(F)|$ is taken to be 1. $X_s(F)$ consists of replicas of $X(F)$, scaled by the Fourier coefficients C_n and repeated at intervals of F_s. Specifically, the replica at the origin is simply $X(F)$ scaled by C_0. Note that the magnitudes, $|C_n|$, have even symmetry. In this case, since $F_M \le F_s - F_M$, there is no overlap among the replicas in the graph of $X_s(F)$. As a result the original signal $x(t)$ can be

recovered by passing $x_s(t)$ through a low pass filter with a bandwidth B that satisfies the condition $F_M \le B \le F_s - F_M$, and a gain of $1/C_0$.





Impulse-train sampling If $p(t)$ is an impulse-train then the Fourier coefficients are given by

$$C_n = \frac{1}{T}\int_{-T/2}^{T/2} \delta(t) e^{-jn2\pi F_s t} dt = \frac{1}{T} \cdot 1 = F_s \text{ for all } n$$

so that the *aliasing formula* becomes

$$X_s(F) = F_s \sum_{n=-\infty}^{\infty} X(F - nF_s)$$

Alternative derivation It can be shown that

$$X_s(j\Omega) = \frac{\Omega_s}{2\pi} \sum_{n=-\infty} X(j(\Omega - n\Omega_s)) = F_s \sum_{n=-\infty} X(j(\Omega - n\Omega_s))$$

Note that some use the notation $X(\Omega)$ instead of $X(j\Omega)$, so that the above equation is written as

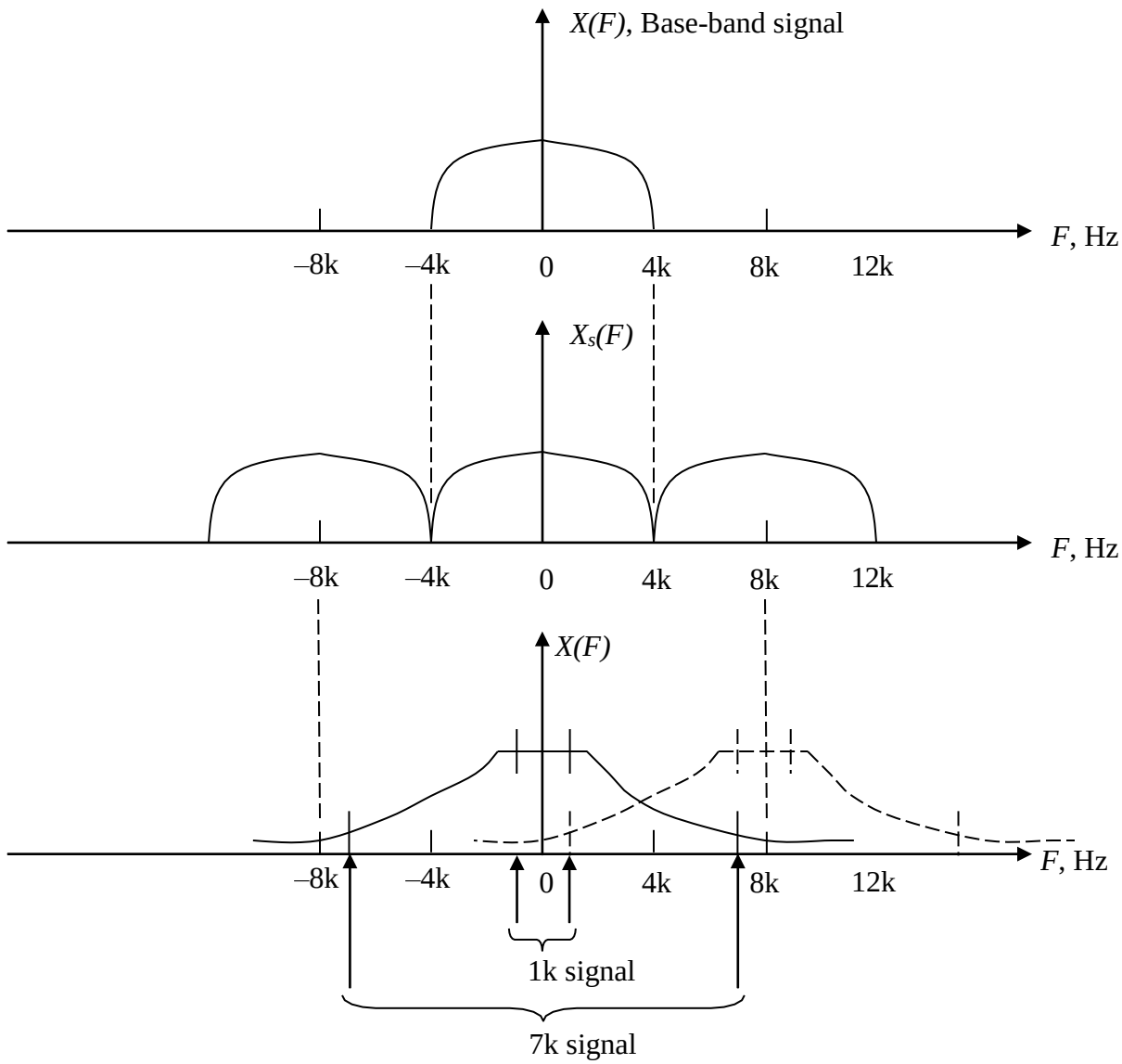
$$X_s(\Omega) = \frac{\Omega_s}{2\pi} \sum_{n=-\infty}^{\infty} X(\Omega - n\Omega_s) = F_s \sum_{n=-\infty}^{\infty} X(\Omega - n\Omega_s)$$

About terminology The highest frequency in the signal is called the *Nyquist frequency*. The minimum sampling rate that, in theory, allows perfect signal recovery is called the *Nyquist rate*. Thus Nyquist rate is twice the Nyquist frequency.

A signal, however, is generally sampled at more than twice the Nyquist frequency. One half the sampling rate is called the *folding frequency*. As an example, if a voice signal is band limited to 3.3 kHz and sampled at 8 kHz then the Nyquist frequency = 3.3 kHz, the Nyquist rate = 6.6 kHz, and the folding frequency = 4 kHz.

To add to the ambiguity, some references use the phrase Nyquist frequency to refer to one half the sampling rate.

Aliasing Illustration using a signal spectrum band-limited to 4 kHz and a sampling rate of 8 kHz. The signal is perfectly band-limited to 4 kHz so that there is no overlapping in the spectrum of the sampled signal.



The signal has a genuine, desirable, 1 kHz component. Since it is not perfectly band-limited to 4 kHz it has another genuine but undesirable 7 kHz component. Due to the first pair of replicas (centered at 8 kHz and –8 kHz) this 7kHz component appears as if it were a 1kHz component – in other words the 7 kHz component is an alias of 1kHz. Thus the first (lowest) alias of the 1 kHz frequency is given by 8 kHz – 1 kHz = 7 kHz.

Due to the second pair of replicas at 16 kHz and –16 kHz the 15k component in the original signal also appears as if it were a 1k component – it is another alias of the 1k. This second alias of the 1 kHz frequency is given by 16 kHz – 1 kHz = 15 kHz. The next alias is (3 x 8 – 1) kHz = 23 kHz.

In general, for any frequency F_1 within the base band (in this case any frequency from 0 to 4000 Hz) its aliases are given by

$$\text{Alias} = kF_s - F_1, \qquad k \text{ is an integer} > 0$$

Aliasing example As an example, let a certain base band signal be band-limited to 8 Hz and let the sampling frequency be F_s = 16 Hz. We do not expect frequencies higher than 8 Hz. Then for the base band frequency of, say, F_1 = 4 Hz, the aliases, F_2, are given by

$$F_2 = kF_s - F_1, \qquad k = 1, 2, \ldots$$

Setting k = 1 gives the lowest alias of F_1, that is, F_2 = 1 x 16 – 4 = 12 Hz. The next alias is 2 x 16 – 4 = 28 Hz. Consider two waveforms with frequencies of 4 Hz and 12 Hz given by

$$x_1(t) = 1 \cos 2\pi 4t \qquad \text{and} \qquad x_2(t) = 1 \cos 2\pi 12t$$

The sampled versions are

$$x_1(n) = \cos 2\pi 4nT = \cos 2\pi 4n(1/16) = \cos (n\pi/2) \text{ and}$$
$$x_2(n) = \cos 2\pi 12nT = \cos 2\pi 12n(1/16) = \cos (n3\pi/2)$$

whose first few samples are tabulated below:

n	0	1	2	3	4	5	6	7
$x_1(n)$	1	0	–1	0	1	0	–1	0
$x_2(n)$	1	0	–1	0	1	0	–1	0

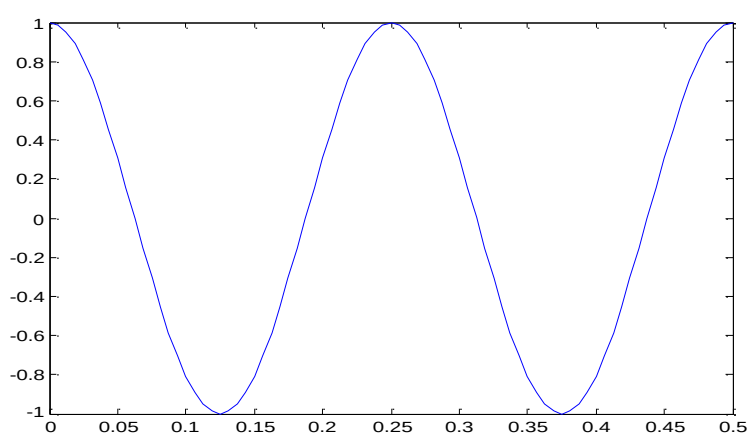
These two sequences are seen to have the same digital frequency and cannot be distinguished from each other *as far as the frequency is concerned*. When passed through a smoothing filter, they will both appear as 4 Hz signals. This is true even if the *amplitudes* of the underlying analog waveforms are different.

In MATLAB

```
%Plot x1(t) = cos 2π4t as t goes from 0 to 0.5 sec (2 cycles) in steps of T = 1/160 sec.
t = 0: 1/160: 0.5; x1 = cos (2*pi*4*t); plot(t, x1)

%Repeat with step size of 1/16 sec.
t = 0: 1/16: 0.5; x1 = cos (2*pi*4*t); plot(t, x1)

%Produce samples of x1(t) as t goes from 0 to 0.5 in steps of 1/16 sec.
t = 0: 1/16: 0.5; x1 = cos (2*pi*4*t)

%Produce samples of x1(n) = cos nπ/2 as n goes from 0 to 8 (2 cycles) in steps of 1
%(T = 1/16 sec.)
n = 0: 1: 8; x1 = cos (n*pi/2)

%Plot x1(n) = cos nπ/20 as n goes from 0 to 80 (2 cycles) in steps of 1 (T = 1/160 sec.)
n = 0: 1: 80; x1 = cos (n*pi/20); plot(n, x1, 'bo'); grid          %Blue circles and grid

%Stem plot x1(n) = cos nπ/2 as n goes from 0 to 8 (2 cycles) in steps of 1 (T = 1/16 sec.)
n = 0: 1: 8; x1 = cos (n*pi/2); stem(n, x1)

%Stem plot x1(n) = cos nπ/20 as n goes from 0 to 80 (2 cycles) in steps of 1
%(T = 1/160 sec.)
n = 0: 1: 80; x1 = cos (n*pi/20); stem(n, x1)

%Titles, labels and grid. Stem plot x1(n) = cos nπ/20 as n goes from 0 to 80 (2 cycles)
%MATLAB won"t accept the kind of single quote in title „Sampled Cosine"
n = 0: 1: 80; x1 = cos (n*pi/20);
stem(n, x1); title(„Sampled Cosine"); xlabel(„n"), ylabel(„x1"); grid
```

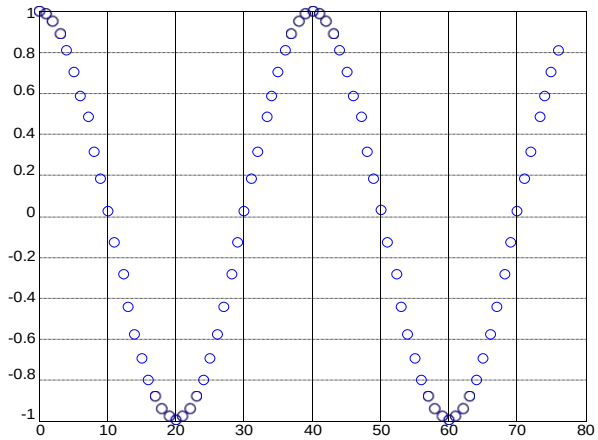
```
%Plot x1(t) = cos 2π4t as t goes from 0 to 0.5 sec (2 cycles) in steps of T = 1/160 sec.
t = 0: 1/160: 0.5; x1 = cos (2*pi*4*t); plot(t, x1)
```

x1 = cos (2*pi*4*t) – 4Hz Cosine *plotted* at T = 1/160 sec.



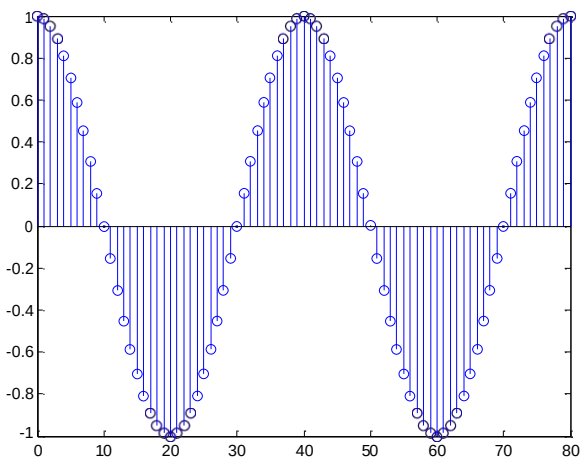
%Plot $x_1(n) = \cos n\pi/20$ as n goes from 0 to 80 (2 cycles) in steps of 1 (T = 1/160 sec.)
n = 0: 1: 80; x1 = cos (n*pi/20); plot(n, x1, 'bo'); grid %Blue circles and grid

x1 = cos (n*pi/20) – 4 Hz Cosine *sampled* at 160 samples per second



%Stem plot $x_1(n) = \cos n\pi/20$ as n goes from 0 to 80 (2 cycles) in steps of 1
%(T = 1/160 sec.)
n = 0: 1: 80; x1 = cos (n*pi/20); stem(n, x1)

Stem plot of x1 = cos (n*pi/20) – 4 Hz Cosine *sampled* at 160 samples per second



Aliasing and digital frequency With F_s = 16 Hz the base band signal must be band-limited to 8 Hz. And we do not expect frequencies higher than 8 Hz. Consider the 3 signals $x_1(t)$, $x_2(t)$, and $x_3(t)$ of frequencies 4Hz, 12Hz and 28Hz, respectively, where x_2 and x_3"s frequencies are the aliases of x_1"s. The continuous and discrete-time signals are given in table below with a sampling rate of 16 samples/sec.

Aliasing and Digital Frequency			
Analog frequency Cycles/sec	**Analog signal**	**Discrete-time signal**	**Digital frequency Cycles/sample**
4	$x_1(t) = \cos 2\pi 4t$	$x_1(n) = \cos 2\pi(1/4)n$	**0.25**
12	$x_2(t) = \cos 2\pi 12t$	$x_2(n) = \cos 2\pi(3/4)n$	**0.75**
28	$x_3(t) = \cos 2\pi 28t$	$x_3(n) = \cos 2\pi(7/4)n$	**1.75**
8	$x_4(t) = \cos 2\pi 8t$	$x_4(n) = \cos 2\pi(1/2)n$	**0.5**
16/3	$x_5(t) = \cos 2\pi(16/3)t$	$x_5(n) = \cos 2\pi(1/3)n$	**1/3**

If the Nyquist criterion is to be satisfied (for perfect signal reconstruction) we never expect any digital frequencies higher than 0.5 cycle/sample (or π rad./sample) in the base band signal. This corresponds to taking 2 samples for every cycle (or a digital frequency of half a cycle per sample) – the Nyquist criterion. Digital frequencies higher than 0.5 cycle/sample ($x_2(n)$ and $x_3(n)$ in this example) are actually disallowed.

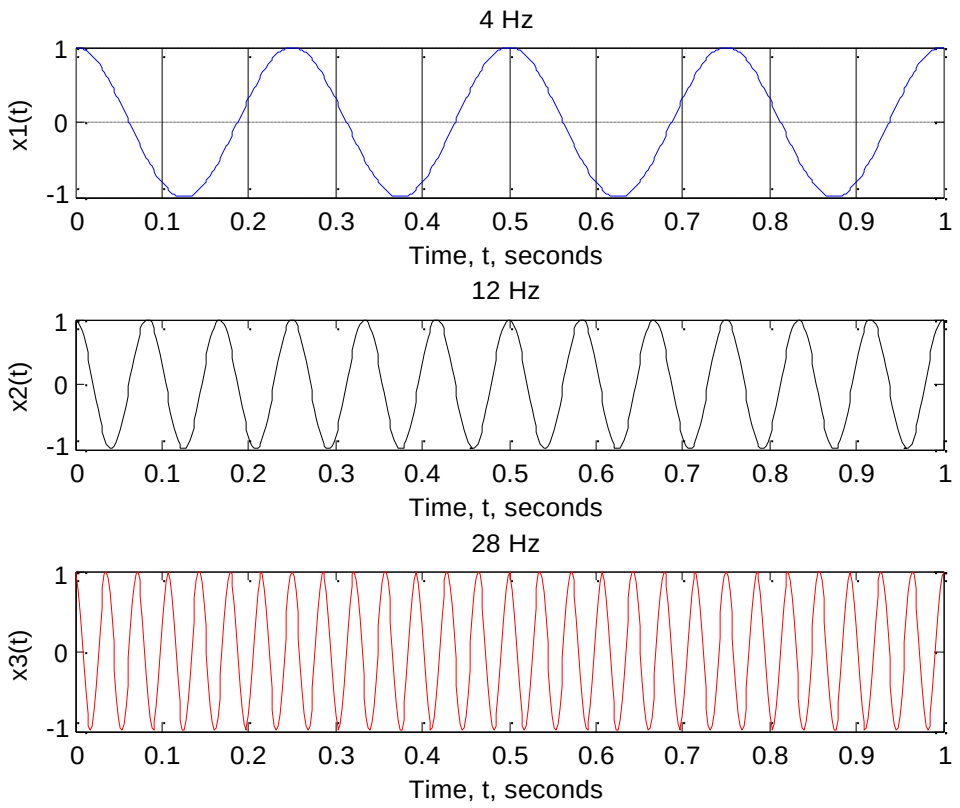
When plotted $x_1(n)$, $x_2(n)$, and $x_3(n)$ cannot be distinguished from one another as far as the digital frequency is concerned. They all have a period = 4 (samples) and a frequency of 0.25 cycle per sample, though we know that $x_2(n)$ has a frequency of 0.75 cycle/sample and $x_3(n)$ has a frequency of 1.75 cycle/sample.

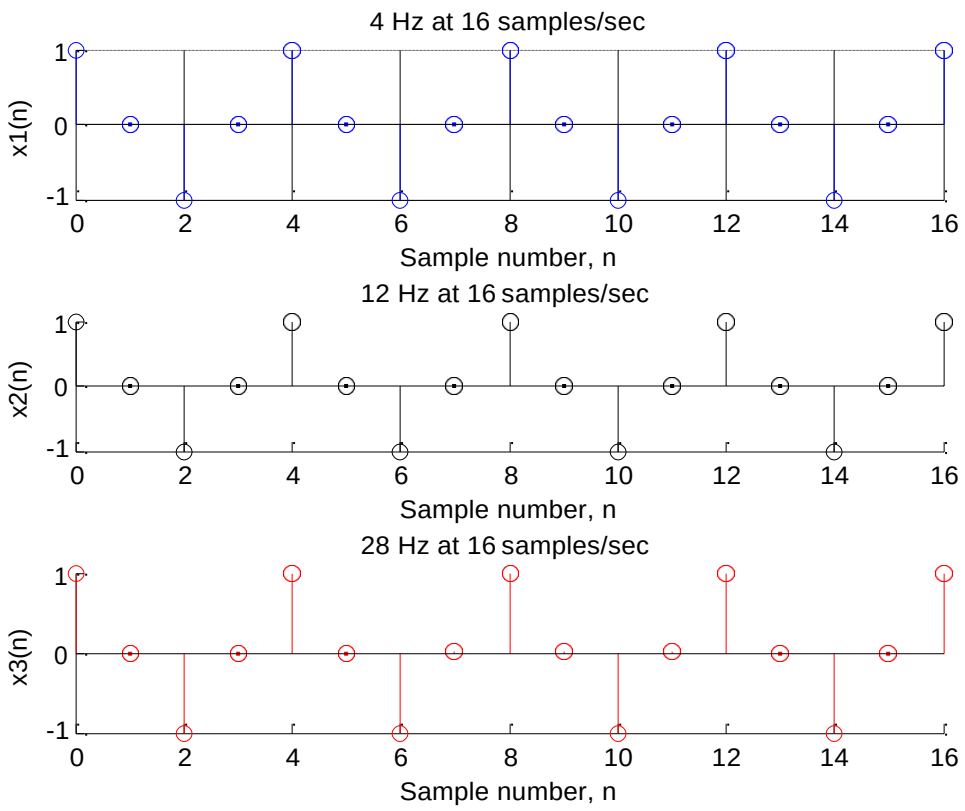
In general any digital frequency above 0.5 cycle/sample (π rad./sample) is an alias (or shows up as an alias of some base band frequency). It actually has more cycles per sample than is apparent in a plot of the sampled data.

We demonstrate below the phenomenon of aliasing using the three waveforms $x_1(t)$, $x_2(t)$, and $x_3(t)$ and the corresponding sequences $x_1(n)$, $x_2(n)$, and $x_3(n)$.

In MATLAB:

```
%Aliasing demo

%First-------------------------------------------
%Plot the continuous-time waveforms x1(t), x2(t), and x3(t) over a 1-second interval
t = 0: 1/500: 1;
x1 = cos (2*pi*4*t); %4 Hz
subplot(3, 1, 1), plot(t, x1, 'b');          %subplot(3, 1, 1) – 3 rows, 1 column, #1
xlabel ('Time, t, seconds'), ylabel('x1(t)');
title ('4 Hz')
grid;

x2 = cos (2*pi*12*t); %12 Hz
subplot(3, 1, 2), plot(t, x2, 'k');          %subplot(3, 1, 2) – 3 rows, 1 column, #2
xlabel ('Time, t, seconds'), ylabel('x2(t)');
title ('12 Hz')

x3 = cos (2*pi*28*t); %28 Hz
subplot(3, 1, 3), plot(t, x3, 'r');          %subplot(3, 1, 3) – 3 rows, 1 column, #3
xlabel ('Time, t, seconds'), ylabel('x3(t)');
title ('28 Hz')

%Second-------------------------------------------
%Plot the sequences x1(n), x2(n), and x3(n)
n = 0: 1: 16;
%4 Hz sampled at 16 Hz
x1 = cos (n*pi/2);
subplot(3, 1, 1), stem(n, x1, 'bo');      %subplot(3, 1, 1) – 3 rows, 1 column, #1
xlabel ('Sample number, n'), ylabel('x1(n)');
title ('4 Hz at 16 samples/sec')
grid;
%12 Hz sampled at 16 Hz
x2 = cos (3*n*pi/2);
subplot(3, 1, 2), stem(n, x2, 'ko');      %subplot(3, 1, 2) – 3 rows, 1 column, #2
xlabel ('Sample number, n'), ylabel('x2(n)');
title ('12 Hz at 16 samples/sec')
%28 Hz sampled at 16 Hz
x3 = cos (7*n*pi/2);
subplot(3, 1, 3), stem(n, x3, 'ro');      %subplot(3, 1, 3) – 3 rows, 1 column, #3
xlabel ('Sample number, n'), ylabel('x3(n)');
title ('28 Hz at 16 samples/sec')
%-------------------------------------------
```


4 Hz
x1(t)
Time, t, seconds
12 Hz
x2(t)
Time, t, seconds
28 Hz
x3(t)
Time, t, seconds


4 Hz at 16 samples/sec
x1(n)
Sample number, n
12 Hz at 16 samples/sec
x2(n)
Sample number, n
28 Hz at 16 samples/sec
x3(n)
Sample number, n

Discrete-time signals

Definition A **discrete-time signal** is a sequence, that is, a function defined on the positive and negative integers.

The sequence $x(n) = x_R(n) + j\, x_I(n)$ is a complex (valued) sequence if $x_I(n)$ is not zero for all n. Otherwise, it is a real (valued) sequence.

Examples of discrete-time signals represented in functional form are given below. (**Exercise**: Plot these signals for several values of n.)

$$x_1(n) = 2 \cos 3n$$
$$x_2(n) = 3 \sin (0.2\pi n)$$

Alternatively, if a signal is non-zero over a finite (small enough) interval, we can list the values of the signal as the elements of a sequence. For example

$$x_3(n) = \{5,\quad 2,\quad -1,\quad \underset{\uparrow}{1},\quad -1/2,\quad 4\}$$

The arrow indicates the value at $n = 0$. We omit the arrow when the first entry represents the value for $n = 0$. The above sequence is a *finite length sequence*. It is assumed that all values of the signal not listed are zero. In the above example $x(0) = 1$, $x(1) = -1/2$, $x(-4) = x(3) = 0$, etc.

Definition A discrete-time signal whose values are from a finite set is called a **digital signal**.

Classification of discrete-time signals (Along lines similar to continuous-time signals)

(Omit) Discrete-time Energy and Power signals The *energy* E of a discrete-time signal $x(n)$ is given by

$$E = \lim_{N \to \infty} \sum_{n=-N}^{N} x(n)\, x^*(n)$$

where x^* is the complex conjugate of x. If $x(n)$ is a real sequence then $x(n)\, x^*(n) = x^2(n)$. The above definition can also be written as

$$E = \lim_{N \to \infty} \sum_{n=-N}^{N} |x(n)|^2 \qquad \text{(there are } 2N+1 \text{ terms here)}$$

The *average power* P of the signal is

$$P = \lim_{N \to \infty} \frac{1}{2N+1} \sum_{n=-N}^{N} |x(n)|^2$$

If E is finite but non zero (i.e., $0 < E < \infty$) the signal is an *energy signal*. It is a *power signal* if E is infinite but P is finite and nonzero (i.e., $0 < P < \infty$). Clearly, when E is finite, $P = 0$. If E is infinite P may or may not be finite.

If neither E nor P is finite, then the signal is neither an energy nor a power signal.

The terms "power" and "energy" are used independently of whether the quantity $\sum_{n=-N}^{N} |x(n)|^2$ (or, $\int_{t_1}^{t_2} |x(t)|^2 dt$, in the continuous time case) actually is related to physical energy. Even if such a relationship exists, these quantities, Σ (.) and $\int$ (.), may have the wrong dimensions and scaling. Still it is convenient to use these terms in a general fashion. It is helpful to imagine that $x(t)$ or $x(n)$ is the voltage across, or, current through, a 1-ohm resistor.

Example 1.2.1 For the signal $x(n) = 1$ for all n,

$$E = \lim_{N\to\infty} \sum_{n=-N}^{N} |x(n)|^2 = \sum_{n=-\infty}^{\infty} 1^2 = \infty \text{ which is infinite energy}$$

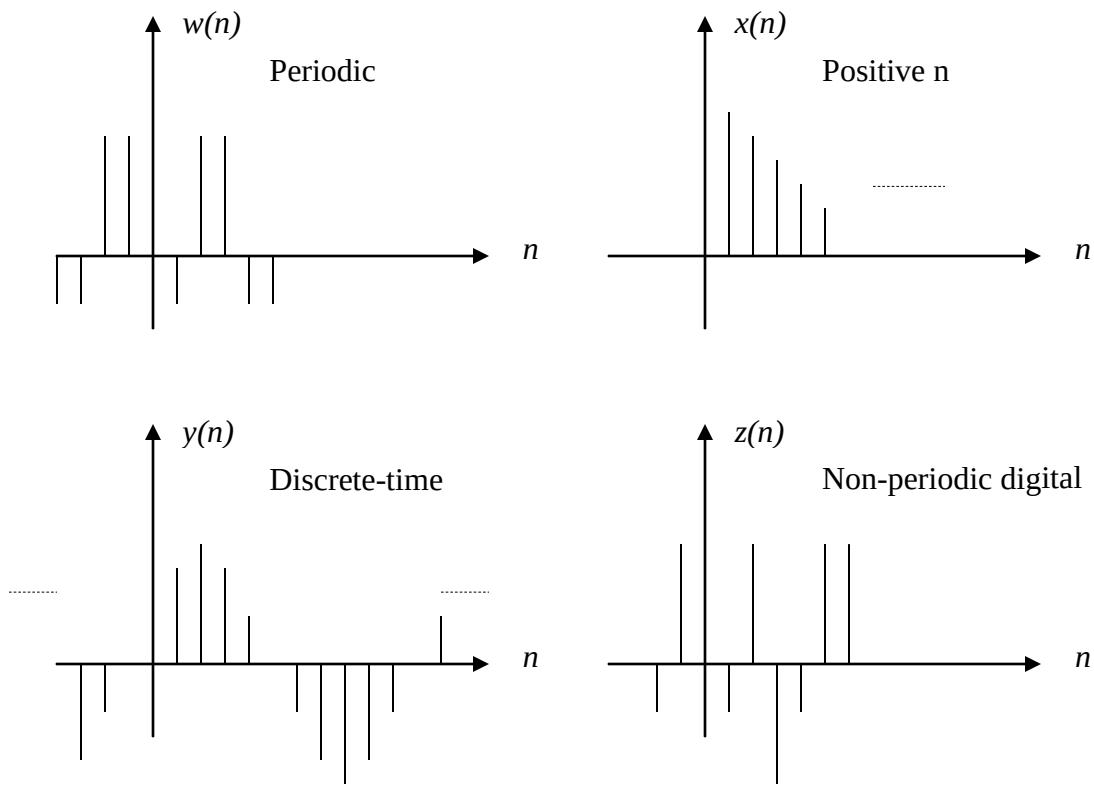
$$P = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{N} |x(n)| = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{N} 1^2 = \lim_{N\to\infty} \frac{2N+1}{2N+1} = 1 \text{ which is finite}$$

Thus $x(n)$ is a power signal.

Example 1.2.2 For the signal $x(n) = n$ both E and P are infinite. This is neither an energy nor a power signal.

(End of Omit)

Examples of discrete-time signals In these examples *w(n)* and *z(n)* take on only a finite number of different values – hence digital. But *x(n)* and *y(n)* take on a countable infinite number of values – they are not digital. (Figure)



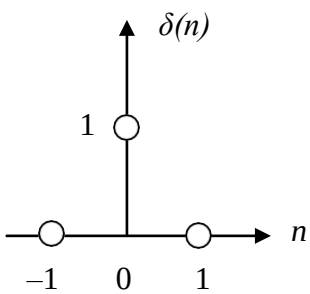
Important discrete-time signals If a continuous-time signal $x(t)$ is sampled at T-second intervals, the result is the sequence $\{x(nT)\}$. For convenience we shall drop the T and the braces and use just $x(n)$ to represent the sequence.

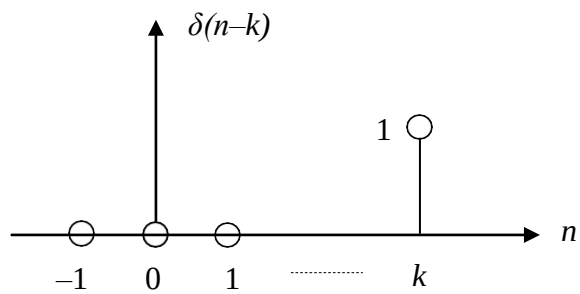
1) The unit sample sequence (discrete-time impulse, aka Kronecker delta)

$$\delta(n) = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases}$$

Whereas $\delta(n)$ is somewhat similar to the continuous-time impulse function $\delta(t)$ – the Dirac delta – we note that the magnitude of the discrete impulse is finite. Thus there are no analytical difficulties in defining $\delta(n)$. It is convenient to interpret the delta function as follows:

$$\delta(\text{argument}) = \begin{array}{l} 1 \text{ when argument} = 0 \\ 0 \text{ when argument} \neq 0 \end{array}$$

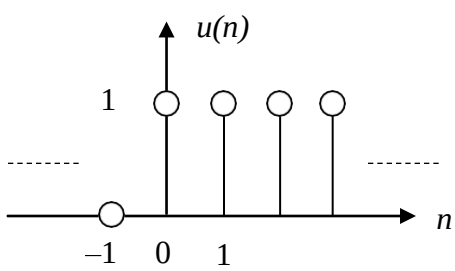


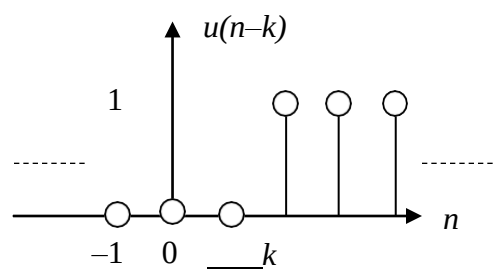


2) The unit step sequence

$$u(n) = \begin{cases} 1, & n \geq 0 \\ 0, & n < 0 \end{cases}$$

$$u(\text{argument}) = \begin{array}{l} 1, \text{ if argument} \geq 0 \\ 0, \text{ if argument} < 0 \end{array}$$



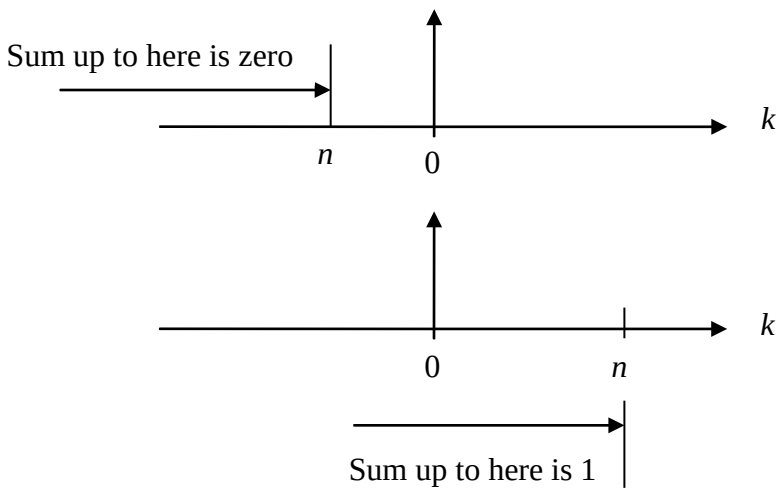


a) The discrete delta function can be expressed as the first difference of the unit step function:

$$\delta(n) = u(n) - u(n-1)$$

b) The sum from $-\infty$ to n of the δ function gives the unit-step:

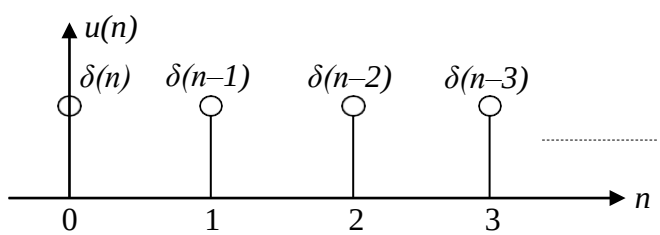
$$\sum_{k=-\infty}^{n} \delta(k) = \begin{cases} 0 & \text{if } n < 0 \\ 1 & \text{if } n \geq 0 \end{cases} = u(n)$$



Results (a) and (b) are like the continuous-time derivative and integral respectively.

c) By inspection of the graph of *u(n)*, shown below, we can write:

$$u(n) = \delta(n) + \delta(n-1) + \delta(n-2) + \ldots = \sum_{\lambda=0}^{\infty} \delta(n-\lambda)$$



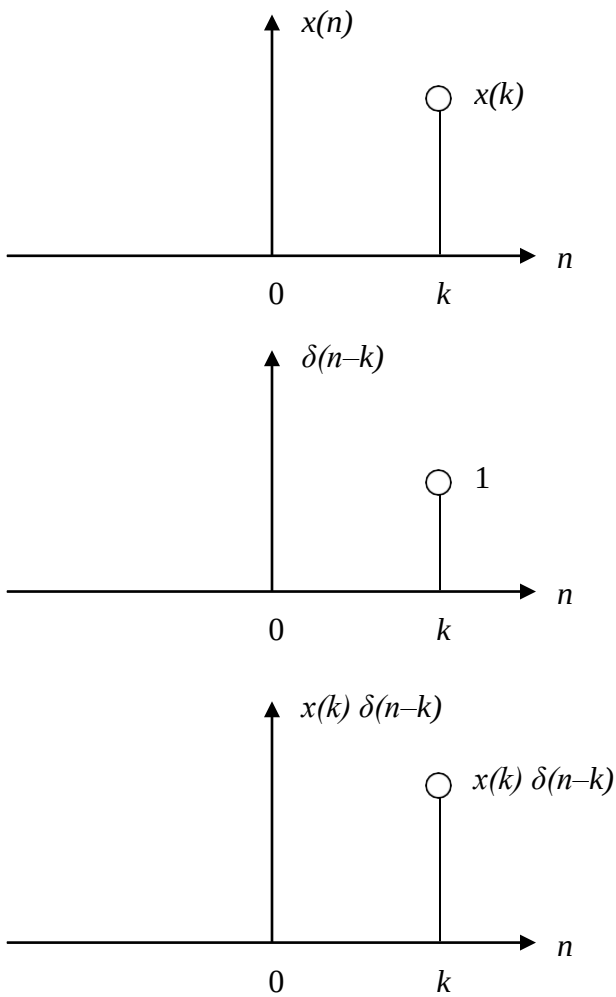
d) For any arbitrary sequence *x(n)*, we have

$$x(n)\,\delta(n-k) = x(k)\,\delta(n-k)$$

that is, the multiplication will pick out just the one value *x(k)*.

If we find the infinite sum of the above we get the sifting property:

$$\sum_{n=-\infty}^{\infty} x(n)\ \delta(n-k) = x(k)$$



e) We can write *x(n)* as follows:

$$x(n) = \ldots + x(-1)\ \delta(n+1) + x(0)\ \delta(n) + x(1)\ \delta(n-1) + x(2)\ \delta(n-2) + \ldots$$

This can be verified to be true for all *n* by setting in turn

$\ldots, n = -2, n = -1, n = 0, n = 1, n = 2$, etc. ...

The above can be written compactly as

$$x(n) = \sum_{k=-\infty}^{\infty} x(k)\ \delta(n-k)$$

This is a weighted-sum of delayed unit sample functions.

3) The real exponential sequence Consider the familiar continuous time signal

$$x(t) = e^{-\alpha t} = e^{-t/\tau}, \qquad t \geq 0$$

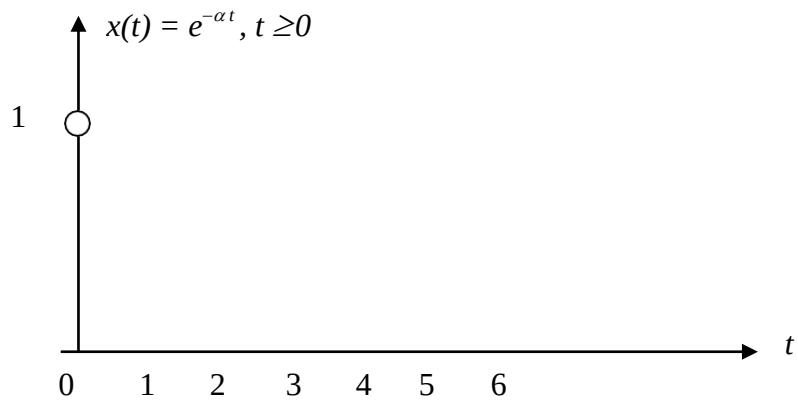
The sampled version is given by setting $t = nT$

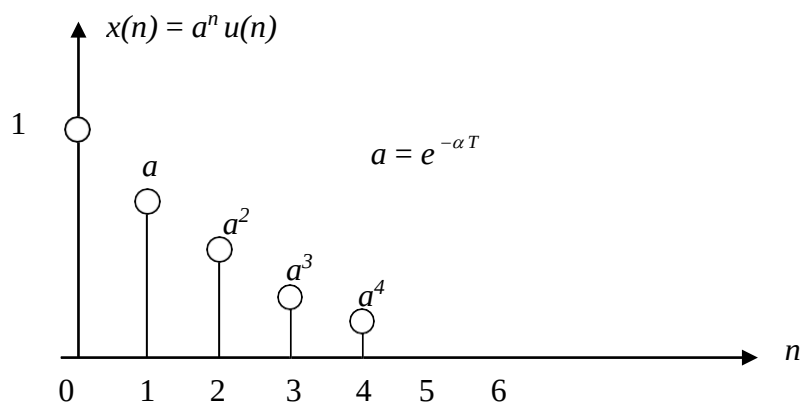
$$x(nT) = e^{-\alpha nT} = \left(e^{-\alpha T}\right)^n, \qquad nT \geq 0$$

Dropping the T from $x(nT)$ and setting $e^{-\alpha T} = a$ we can write

$$x(n) = a^n, \qquad n \geq 0$$

The sequence can also be defined for both positive and negative n, by simply writing $x(n) = a^n$ for all n.





***4)* The sinusoidal sequence** Consider the continuous-time sinusoid $x(t)$

$$x(t) = A \sin 2\pi F_0 t = A \sin \Omega_0 t$$

F_0 and Ω_0 are the analog frequency in Hertz (or cycles per second) and radians per second, respectively. The sampled version is given by

$$x(nT) = A \sin 2\pi F_0 nT = A \sin \Omega_0 nT$$

We may drop the T from $x(nT)$ and write

$$x(n) = A \sin 2\pi F_0 nT = A \sin \Omega_0 nT, \text{ for all } n$$

We may write $\Omega_0 T = \omega_0$ which is the digital frequency in radians (per sample), so that

$$x(n) = A \sin \omega_0 n = A \sin 2\pi f_0 n, \text{ for all } n$$

Setting $\omega_0 = 2\pi f_0$ gives $f_0 = \omega_0/2\pi$ which is the digital frequency in cycles per sample. In the analog domain the horizontal axis is calibrated in seconds; "second" is one unit of the independent variable, so Ω_0 and F_0 are in "per second". In the digital domain the horizontal axis is calibrated in samples; "sample" is one unit of the independent variable, so ω_0 and f_0 are in "per sample".

Classification of discrete-time signals (cont'd)

Periodic signal The discrete-time signal $x(n)$ is periodic if, for some integer $N > 0$

$$x(n+N) = x(n) \text{ for all } n$$

The smallest value of N that satisfies this relation is the (**fundamental**) **period** of the signal. If there is no such integer N, then $x(n)$ is an aperiodic signal.

Given that the continuous-time signal $x_a(t)$ is periodic, that is, $x_a(t) = x_a(t+T_0)$ for all t, and that $x(n)$ is obtained by sampling $x_a(t)$ at T second intervals, $x(n)$ will be periodic if T_0/T is a rational number but not otherwise. If $T_0/T = N/L$ for integers $N \geq 1$ and $L \geq 1$ then $x(n)$ has exactly N samples in L periods of $x_a(t)$ and $x(n)$ is periodic with period N.

Periodicity of sinusoidal sequences The sinusoidal sequence $\sin(2\pi f_0 n)$ has several major differences from the continuous-time sinusoid as follows:

a) The sinusoid $x(n) = \sin(2\pi f_0 n)$ or $\sin(\omega_0 n)$ is periodic if f_0, that is, $\omega_0/2\pi$, is rational. If f_0 is not rational the sequence is not periodic. Replacing n with $(n+N)$ we get

$$x(n+N) = \sin(2\pi f_0 (n+N)) = \sin 2\pi f_0 n.\cos 2\pi f_0 N + \cos 2\pi f_0 n.\sin 2\pi f_0 N$$

Clearly $x(n+N)$ will be equal to $x(n)$ if $f_0 N = m$, an integer or $f_0 = m/N$. The fundamental period is obtained by choosing m as the smallest integer that yields an integer value for N. For example, if $f_0 = 15/25$, which in reduced fraction form is 3/5, then we can choose m = 3 and get $N = 5$ as the period. If f_0 is rational then $f_0 = p/q$ where p and q are integers. If p/q is in reduced fraction form then q is the period as in the above example.

On the other hand if f_0 is irrational, say $f_0 = \sqrt{2}$, then N will not be an integer, and thus $x(n)$ is aperiodic.

Note: In expressions like $\sin \omega n$, $\sin 2\pi f n$, $e^{j\omega n}$ and $e^{j2\pi f n}$ we shall refer to ω or f as the frequency even when the signal concerned is not periodic by the definition above.

b) The sinusoidal sequences $\sin \omega_0 n$ and $\sin((\omega_0+2\pi k)n)$ for $0 \leq \omega_0 \leq 2\pi$ are identical. This can be shown using the identity

$$\begin{aligned}\sin((\omega_0+2\pi k)n) &= \sin(\omega_0 n+2\pi kn)\\ &= \sin \omega_0 n \cos 2\pi kn + \cos \omega_0 n \sin 2\pi kn \qquad \text{QED}\end{aligned}$$

Similarly, $\cos \omega_0 n$ and $\cos((\omega_0+2\pi k)n)$ are the same. Therefore in considering sinusoidal sequences for analysis purposes, ω_0 can be restricted to the range $0 \leq \omega_0 \leq \pi$ without any loss of generality.

c) For $\pi < \omega_0 < 2\pi$, based on the same trigonometric identities,

sin $\omega_0 n$ is the negative of sin $((2\pi-\omega_0)n)$, and
cos $\omega_0 n$ is the same as cos $((2\pi-\omega_0)n)$

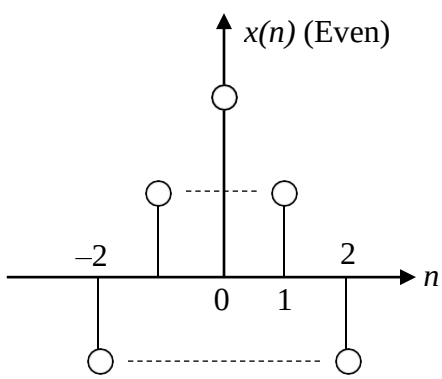
The sum of two discrete-time periodic sequences is also periodic. Let $x(n)$ be the sum of two periodic sequences, $x_1(n)$ and $x_2(n)$, with periods N_1 and N_2 respectively. Let p and q be two integers such that

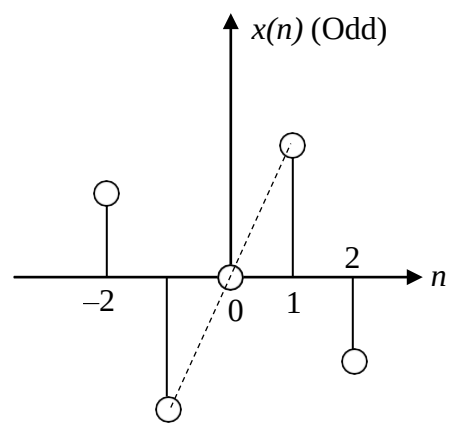
$$pN_1 = qN_2 = N \qquad (p \text{ and } q \text{ can always be found})$$

Then $x(n)$ is periodic with period N since, for all n,

$$\begin{aligned} x(n+N) &= x_1(n+N) + x_2(n+N) \\ &= x_1(n+pN_1) + x_2(n+qN_2) \\ &= x_1(n) + x_2(n) \\ &= x(n) \text{ for all } n \end{aligned}$$

Odd and even sequences The signal $x(n)$ is an *even* sequence if $x(n) = x(-n)$ for all n, and is an *odd* sequence if $x(n) = -x(-n)$ for all n.

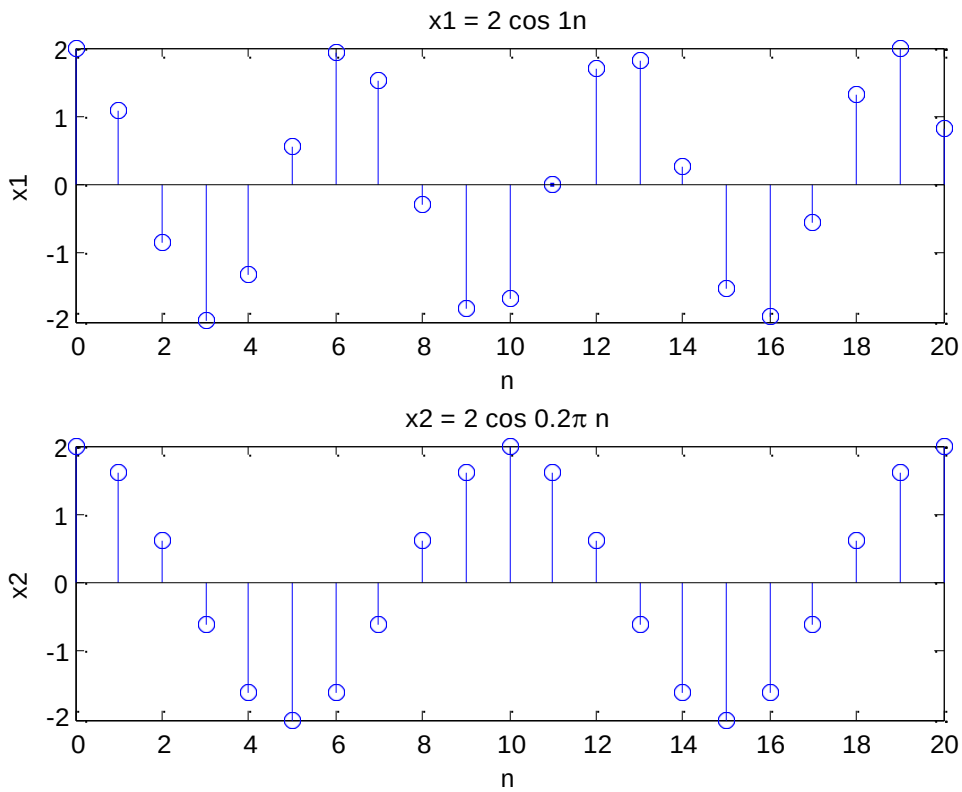




The *even* part of $x(n)$ is determined as $x_e(n) = \dfrac{x(n) + x(-n)}{2}$ and the *odd* part of $x(n)$ is given by $x_o(n) = \dfrac{x(n) - x(-n)}{2}$. The signal $x(n)$ then is given by $x(n) = x_e(n)+x_o(n)$.

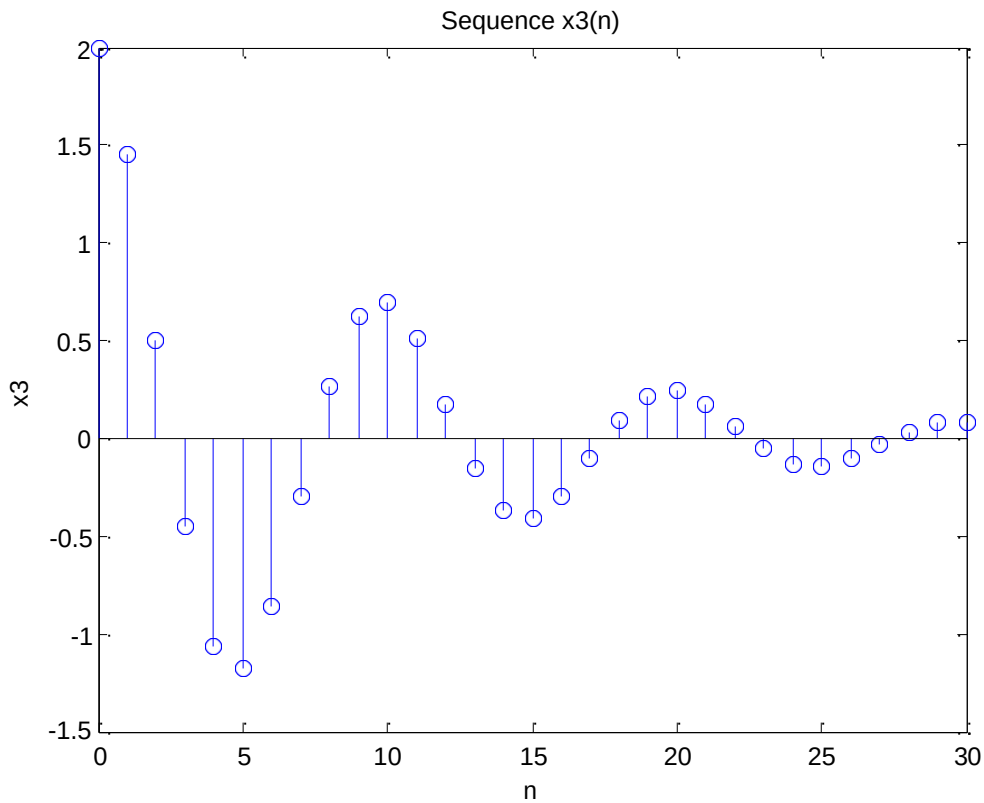
Example 1.2.3 Plot the sequences $x_1(n) = 2 \cos n$ and $x_2(n) = 2 \cos (0.2\pi n)$. What are their "frequencies"? →hich of them is truly periodic and what is its periodicity?
Solution The MATLAB program segment follows:

```
N = 21; n = 0: N-1;
%
%Nonperiodic
x1= 2*cos(1*n);
subplot(2, 1, 1), stem(n, x1);
xlabel('n'), ylabel('x1'); title('x1 = 2 cos 1n');
%
%Periodic
x2 = 2*cos(0.2*pi*n);
subplot(2, 1, 2), stem(n, x2);
xlabel('n'), ylabel('x2'); title('x2 = 2 cos 0.2\pi n');
```



Example 1.2.4 Plot the sequence $x_3(n) = 2(0.9)^n \cos 0.2\pi n$.
Solution The MATLAB program segment follows:

```
n = [0: 30];
%
%".^" stands for element-by-element exponentiation
%".*" stands for element-by-element multiplication
x3 = 2* ((0.9) .^n) .*cos(0.2*pi*n);
stem(n, x3);
xlabel('n'), ylabel('x3'); title('Sequence x3(n)');
```



Transformation of the independent variable

Shifting and folding (reflecting about the vertical axis) Given the sequence *x(n)*, where *n* is the independent variable, we have the following two transformation operations:

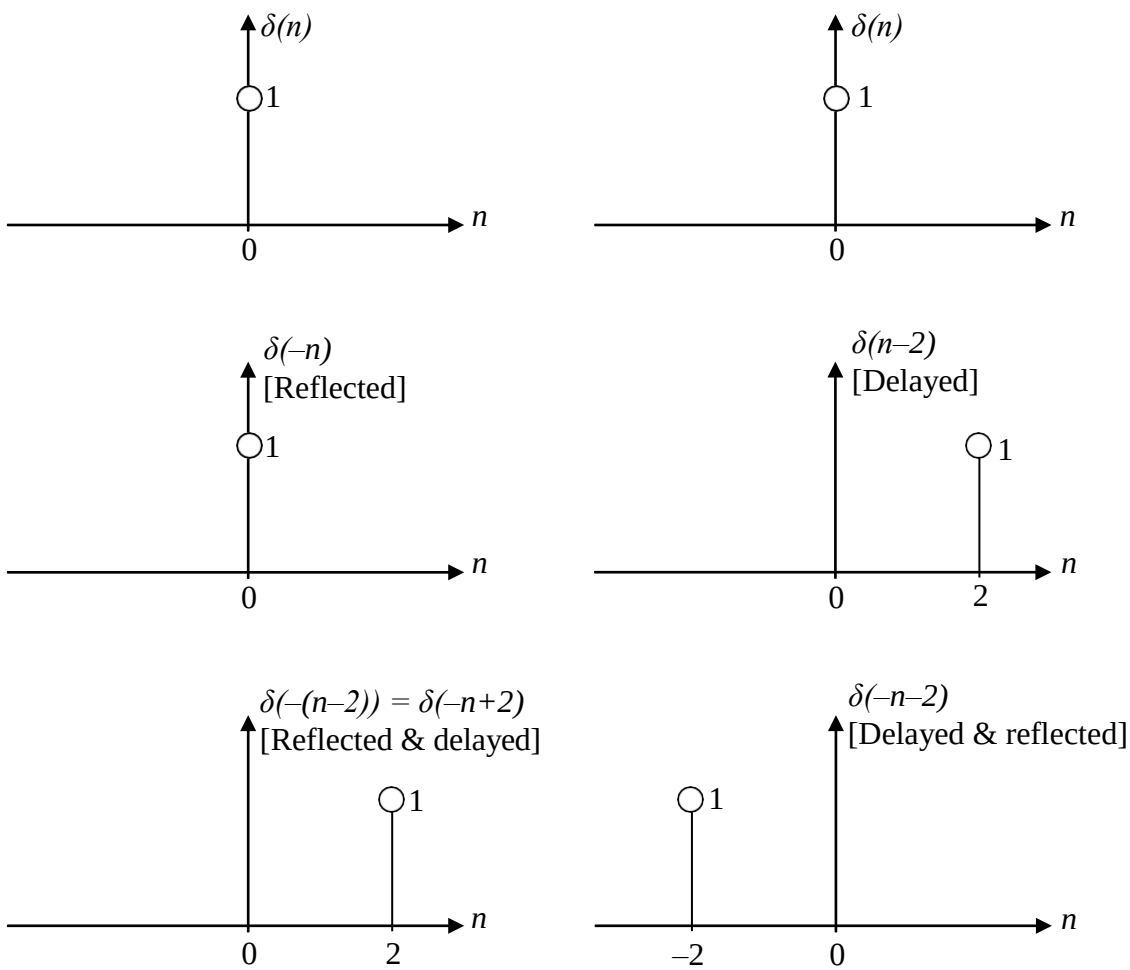
- **Shifting in time** by *k* units, where *k* is an integer, is denoted by *x(n–k)*. The sequence *x(n–k)* represents the sequence *x(n)* shifted by *k* samples, to the right if *k* is positive, or to the left if *k* is negative. Parenthesize *n* and replace it by (*n–k*) for *k* units of delay, or by (*n*+*k*) for *k* units of time advancement.
- **Folding (**a.k.a. **time reversal** or **reflecting about the vertical axis)** is denoted by *x(–n)*. The signal *x(–n)* corresponds to reflecting *x(n)* about the time origin *n* = 0. Reverse the sign of *n* (replace *n* with –*n*).

As in the case of continuous-time signals the operations of shifting and folding are *not commutative*. In other words, the result of first shifting and then folding is not the same as that of first folding and then shifting.

A third operation is **scaling**, of which more will be said in a later unit.

Example 1.3.1 [Delta function] Given the delta function $\delta(n)$, first reflect then shift (delay) by 2 units. The other possibility is first to shift (delay) by 2 units and then reflect.

The result of "reflect then shift" is shown below left. *Reflect* means to change the sign of n; then *shifting* is done by replacing (n) by (n–2). The result is: (1) $\delta(n) \prec \delta(-n)$ and (2) $\delta(-n) = \delta(-(n)) \prec \delta(-(n-2)) = \delta(-n+2)$.



To continue the example, the second possibility, the result of "shift then reflect" is shown above right. *Shift* means replacing (n) by (n–2); then *reflect* by changing the sign of n. The result is $\delta(-n-2)$. The result is: (1) $\delta(n) = \delta((n)) \prec \delta((n-2))$ and (2) $\delta((n-2)) = \delta(n-2) \prec \delta(-n-2)$.

Note that the two end results are not the same. This serves to illustrate that the two operations of shifting and reflecting are not commutative:

$$\text{Fold}(\text{Shift}(\delta(n))) \neq \text{Shift}(\text{Fold}(\delta(n)))$$

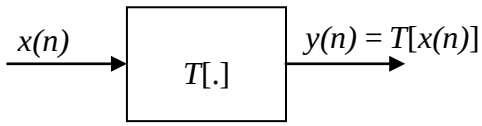
Discrete-time systems

Definition A ***discrete-time system*** is a mapping from the set of acceptable discrete-time signals, called the input set, to a set of discrete-time signals called the output set.

Definition A discrete-time system is ***deterministic*** if its output to a given input does not depend upon some random phenomenon. If it does, the system is called a ***random (stochastic) system***.

Definition A ***digital system*** is a mapping which assigns a digital output signal to every acceptable digital input signal.

A discrete-time system can be thought of as a transformation or operator, T, that maps an input sequence $x(n)$ to an output sequence $y(n)$ shown thus:



In what follows we focus on the presence or absence of the following properties in discrete-time systems: linearity, shift invariance, causality and stability.

Filter Some refer to a linear time-invariant (LTI) system simply as a **filter**, that is, a filter is a system T with a single input and a single output signal that is both linear and time-invariant.

Linearity

Definition A discrete-time system $T[.]$ is linear if the response to a weighted sum of inputs $x_1(n)$ and $x_2(n)$ is a weighted sum (with the same weights) of the responses of the inputs separately for all weights and all acceptable inputs. Thus the system $y(n) = T[x(n)]$ is linear if for all a_1, a_2, $x_1(n)$ and $x_2(n)$ we have

$$T[a_1x_1(n)+a_2x_2(n)] = a_1T[x_1(n)] + a_2T[x_2(n)]$$

Another way of saying this is that if the inputs $x_1(n)$ and $x_2(n)$ produce the outputs $y_1(n)$ and $y_2(n)$, respectively, then the input $a_1x_1(n) + a_2x_2(n)$ produces the output $a_1\, y_1(n) + a_2\, y_2(n)$. This is called the **superposition principle**. The a_1, a_2, $x_1(n)$ and $x_2(n)$ may be complex-valued. The above definition combines two properties, viz.,

1. **Additivity**, that is, $T[x_1(n)+x_2(n)] = T[x_1(n)] + T[x_2(n)]$, and
2. **Scaling** (or **homogeneity**), that is, $T[c\, x(n)] = c\, T[x(n)]$

The procedure of checking for linearity is:

1. Find outputs $y_1(n)$ and $y_2(n)$ corresponding to inputs $x_1(n)$ and $x_2(n)$
2. Form the sum $a_1\, y_1(n) + a_2y_2(n)$
3. Find output $y_3(n)$ corresponding to input $a_1x_1(n) + a_2x_2(n)$
4. Compare the results of steps 2 and 3

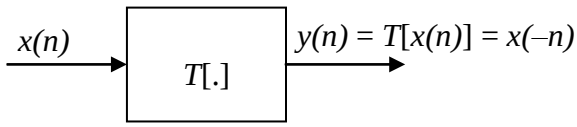
Examples of linear systems:

1. $y(n) = x(n) + x(n-1) + x(n-2)$
2. $y(n) = y(n-1) + x(n)$
3. $y(n) = 0$
4. $y(n) = n\,x(n)$ (But time-varying)

Examples of nonlinear systems:

1. $y(n) = x^2(n)$
2. $y(n) = 2\,x(n)+3$. This is a ***linear equation*** though! This system is made up of a linear part, $2\,x(n)$, and a zero-input response, 3. This is called an ***incrementally linear system***, for it responds linearly to *changes* in the input.

Example 1.4.1 Determine if the system $y(n) = T[x(n)] = x(-n)$ is linear or nonlinear.



Answer Determine the outputs $y_1(.)$ and $y_2(.)$ corresponding to the two input sequences $x_1(n)$ and $x_2(n)$ and form the weighted sum of outputs:

$$y_1(n) = T[x_1(n)] = x_1(-n)$$
$$y_2(n) = T[x_2(n)] = x_2(-n)$$

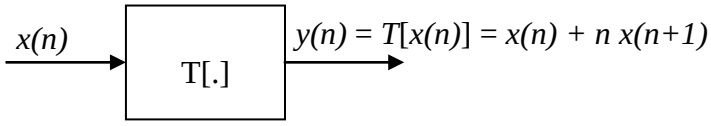
The weighted sum of outputs = $a_1\,x_1(-n) + a_2\,x_2(-n)$ < (A).

Next determine the output y_3 due to a weighted sum of inputs:

$$y_3(n) = T[a_1\,x_1(n) + a_2\,x_2(n)] = a_1\,x_1(-n) + a_2\,x_2(-n) \quad < \text{(B)}$$

Check if (A) and (B) are equal. In this case (A) and (B) are equal; hence the system is linear.

Example 1.4.2 Examine $y(n) = T[x(n)] = x(n) + n\,x(n+1)$ for linearity.



Answer The outputs due to $x_1(n)$ and $x_2(n)$ are:

$$y_1(n) = T[x_1(n)] = x_1(n) + n\,x_1(n+1)$$
$$y_2(n) = T[x_2(n)] = x_2(n) + n\,x_2(n+1)$$

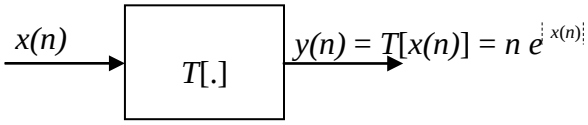
The weighted sum of outputs = $a_1\,x_1(n) + a_1\,n\,x_1(n+1) + a_2\,x_2(n) + a_2\,n\,x_2(n+1)$ < (A)

The output due to a weighted sum of inputs is

$$\begin{aligned} y_3(n) &= T[a_1\,x_1(n) + a_2\,x_2(n)] \\ &= a_1\,x_1(n) + a_2\,x_2(n) + n\,(a_1\,x_1(n+1) + a_2\,x_2(n+1)) \\ &= a_1\,x_1(n) + a_2\,x_2(n) + n\,a_1\,x_1(n+1) + n\,a_2\,x_2(n+1) \quad < \text{(B)} \end{aligned}$$

Since (A) and (B) are equal the system is linear.

Example 1.4.3 Check the system $y(n) = T[x(n)] = ne^{|x(n)|}$ for linearity.



Answer The outputs due $x_1(n)$ and $x_2(n)$ are:

$$y_1(n) = T[x_1(n)] = n\, e^{|x_1(n)|}$$
$$y_2(n) = T[x_2(n)] = n\, e^{|x_2(n)|}$$

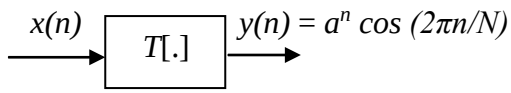
The weighted sum of the outputs $= a_1\, ne^{|x_1(n)|} + a_2\, ne^{|x_2(n)|}$ < (A)

The output due to a weighted sum of inputs is

$$y_3(n) = T[a_1\, x_1(n) + a_2\, x_2(n)] = n\, e^{|a_1 x_1(n) + a_2 x_2(n)|} \quad < \text{(B)}$$

We can specify a_1, a_2, $x_1(n)$, $x_2(n)$ such that (A) and (B) are not equal. Hence nonlinear.

Example 1.4.4 Check the system $y(n) = T[x(n)] = a^n \cos(2\pi n/N)$ for linearity.



Answer Note that the input is $x(n)$. Clearly $y(n)$ is independent of $x(n)$. The outputs due to $x_1(n)$ and $x_2(n)$ are:

$$y_1(n) = T[x_1(n)] = a^n \cos(2\pi n/N)$$
$$y_2(n) = T[x_2(n)] = a^n \cos(2\pi n/N)$$

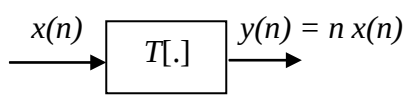
The weighted sum of the outputs $= b_1\, a^n \cos(2\pi n/N) + b_2\, a^n \cos(2\pi n/N)$ < (A)

The output due to a weighted sum of inputs is

$$y_3(n) = T[b_1\, x_1(n) + b_2\, x_2(n)] = a^n \cos(2\pi n/N) \quad < \text{(B)}$$

(A) and (B) are not equal, so the system is not linear. (But (A) $= (b_1+b_2)\, a^n \cos(2\pi n/N)$ and this is equal to (B) within a constant scaling factor.)

Example 1.4.5 Check the system $y(n) = T[x(n)] = n\, x(n)$ for linearity.



Answer For the two arbitrary inputs $x_1(n)$ and $x_2(n)$ the outputs are

$$y_1(n) = T[x_1(n)] = n\, x_1(n)$$
$$y_2(n) = T[x_2(n)] = n\, x_2(n)$$

For the weighted sum of inputs $a_1\, x_1(n) + a_2\, x_2(n)$ the output is

$$y_3(n) = T[a_1\, x_1(n) + a_2\, x_2(n)] = n\,(a_1\, x_1(n) + a_2\, x_2(n))$$
$$= a_1\, n\, x_1(n) + a_2\, n\, x_2(n)$$
$$= a_1\, y_1(n) + a_2\, y_2(n).$$ Hence the system is linear.

Example 1.4.6 Check $y(n) = T[x(n)] = x^2(n)$ for linearity.

Answer For $x_1(n)$, the output is $y_1(n) = x_1^2(n)$ and for $x_2(n)$, $y_2(n) = x_2^2(n)$.

The weighted sum of outputs $= a_1\, x_1^2(n) + a_2\, x_2^2(n)$ < (A)

The output due to a weighted sum of inputs is

$$y_3(n) = T[a_1 x_1(n) + a_2 x_2(n)] = a_1^2 x_1^2(n) + a_2^2 x_2^2(n) + 2\, a_1 a_2 x_1(n)\, x_2(n) \quad \text{(B)}$$

It is possible to specify a_1, a_2, $x_1(n)$, and $x_2(n)$ so that (A) and (B) are not equal. Hence the system is nonlinear.

Example 1.4.7 Determine if $y(n) = T[x(n)] = 2\, x(n) + 3$ is linear.
Answer For $x_1(n)$, the output is $y_1(n) = 2\, x_1(n) + 3$ and for $x_2(n)$, it is $y_2(n) = 2\, x_2(n) + 3$.
Weighted sum of outputs $= a_1 (2\, x_1(n) + 3) + a_2 (2\, x_2(n) + 3)$

$$= 2\, a_1 x_1(n) + 3\, a_1 + 2\, a_2 x_2(n) + 3\, a_2 \quad \text{(A)}$$

The output due to a weighted sum of inputs is

$$y_3(n) = T[a_1 x_1(n) + a_2 x_2(n)] = 2\,(a_1 x_1(n) + a_2 x_2(n)) + 3$$
$$= 2\, a_1 x_1(n) + 2\, a_2 x_2(n) + 3 \quad \text{(B)}$$

It is possible to specify a_1, a_2, $x_1(n)$, and $x_2(n)$ such that (A) and (B) are unequal. Hence the system is nonlinear.

(Since the output $y(n) = 3$ if $x(n) = 0$ we see that the system violates the **"zero-in / zero-out" property** of linear systems. Hence nonlinear.)

Example 1.4.8 Determine if $y(n) = T[x(n)] = \mathrm{Re}\{x(n)\}$ is linear.
Answer Note that the input signals as well as the scaling constants a_1 and a_2 are allowed to be *complex*. Let $x_1(n) = r(n) + j\, s(n)$. Then $y_1(n) = \mathrm{Re}\{x_1(n)\} = r(n)$.

The scaling / homogeneity property says that, if the response to $x_1(n)$ is $y_1(n)$, then the response to $x_2(n) = a.x_1(n)$ is $a.y_1(n)$ where a is *any* constant. Let $a = j$, and choose $x_2(n) = a.x_1(n) = j.x_1(n) = j.(r(n) + j\, s(n)) = -s(n) + j\, r(n)$. The corresponding output is $y_2(n) = \mathrm{Re}\{x_2(n)\} = -s(n)$.

Thus if $y_1(n) = T[.] = T[x_1(n)] = r(n)$, then

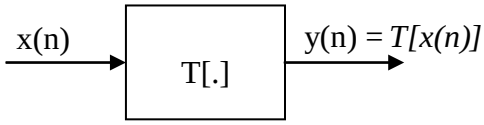
$$y_2(n) = T[.] = T[x_2(n)] = T[j\, x_1(n)] = -s(n).$$

This is not equal to $j\, r(n)$ as would be expected from the homogeneity property. Hence the system is nonlinear.

Shift-invariance (time-invariance)

Definition A discrete time system $y(n) = T[x(n)]$ is shift-invariant if, for all $x(n)$ and all n_0, we have: $T[x(n-n_0)] = y(n-n_0)$.

This means that applying a time delay (or advance) to the input of a system is equivalent to applying it to the output.



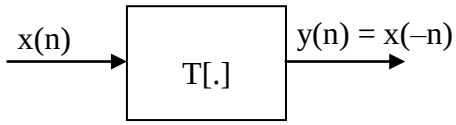
The procedure for determining shift-invariance is:

Step 1. Determine output $y(n)$ corresponding to input $x(n)$.
Step 2. Delay the output $y(n)$ by n_0 units, resulting in $y(n-n_0)$.
Step 3. Determine output $y(n, n_0)$ corresponding to input $x(n-n_0)$.
Step 4. Determine if $y(n, n_0) = y(n-n_0)$. If equal, then the system is shift-invariant; otherwise it is time-varying.

When we suspect that the system is time-varying a very useful alternative approach is to find a counter-example to disprove time-invariance, i.e., use intuition to find an input signal for

which the condition of shift-invariance is violated and that suffices to show that a system is not shift-invariant.

Example 1.4.9 Test if $y(n) = T[x(n)] = x(-n)$ is shift-invariant.



Answer Find output for x(n), delay it by n_0, and compare with the output for $x(n-n_0)$. The output for x(n) is

$$y(n) = T[x(n)] = x(-n)$$

Delaying y(n) by n_0 gives

$$y(n-n_0) = x(-(n-n_0)) = x(-n+n_0) \quad < \text{(A)}$$

As an aside this amounts to reflecting first and then shifting.

The output for $x(n-n_0)$ is denoted $y(n, n_0)$ and is given by

$$y(n, n_0) = T[x(n-n_0)] = x(-n-n_0) \quad < \text{(B)}$$

As an aside this amounts to shifting first and then reflecting.

(A) and (B) are not equal. That is, $y(n, n_0) \neq y(n-n_0)$, so the system is time-varying.

Example 1.4.10 Examine $y(n) = T[x(n)] = x(n) + n\,x(n+1)$ for time invariance.
Answer Notice that the difference equation has a **time-varying coefficient**, n. The output y(n) corresponding to x(n) is already given above. Delaying y(n) by n_0 gives

$$y(n-n_0) = x(n-n_0) + (n-n_0)\,x(n-n_0+1) \quad < \text{(A)}$$

Compare with $y(n, n_0) = T[x(n-n_0)] = x(n-n_0) + n\,x(n-n_0+1) \quad < \text{(B)}$

(A) $\neq$ (B), so the system is time varying.

Example 1.4.11 Check for time invariance of the system $y(n) = T[x(n)] = n\,x(n)$.
Answer We shall do this by counterexample(s) as well as by the formal procedure. The formal procedure is:

$$y(n) = T[x(n)] = n\,x(n)$$

Delay this by n_0 to get $y(n-n_0) = (n-n_0)\,x(n-n_0) \quad < \text{(A)}$

Compare with $y(n, n_0) = T[x(n-n_0)] = n\,x(n-n_0) \quad < \text{(B)}$

Since (A) $\neq$ (B), the system is time-varying.

Alternative We expect that it is time varying since the equation has a time varying coefficient. Find a counter example to show that the system is time varying. For input $x(n) = \delta(n)$, the output is

$$y(n) = n\,\delta(n) = 0 \text{ for all } n$$

For input $x(n-1) = \delta(n-1)$, the output is

$$y(n, 1) = n\,\delta(n-1) = 1\,\delta(n-1)$$

Thus while $x(n-1)$ is a shifted version of $x(n)$, $y(n, 1)$ is not a shifted version of $y(n)$. So the system is time-varying.

Another counter-example If $x(n) = u(n)$, then

$$y(n) = n\,u(n)$$

But if the input is $x(n-2) = u(n-2)$, then

$$y(n, 2) = n\,u(n-2) = (n-2+2)\,u(n-2) = \underbrace{(n-2)\,u(n-2)}_{\text{Delayed version}} + \underbrace{2\,u(n-2)}_{\text{Extra term}}$$

The extra term shows that $y(n, 2) \neq y(n-2)$. So the system is time-varying.

Example 1.4.12 Check for time invariance the system $y(n) = T[x(n)] = \cos(x(n))$.
Answer The output $y(n)$ corresponding to input $x(n)$ is

$$y(n) = T[x(n)] = \cos(x(n))$$

Delay this output by n_0 to get $y(n-n_0) = \cos(x(n-n_0))$ < (A)

For the input $x(n-n_0)$ the output is

$$y(n, n_0) = T[x(n-n_0)] = \cos(x(n-n_0)) \quad < \text{(B)}$$

Since (A) and (B) are equal we have $y(n, n_0) = y(n-n_0)$. Therefore the system is time-invariant.

Example 1.4.13 The system $y(n) = T[x(n)] = g(n)\,x(n)$ needs to be tested for time-invariance.
Answer Note that the coefficient $g(n)$ is time-varying. Hence, the system is time-varying. The output $y(n)$ due to input $x(n)$ is

$$y(n) = T[x(n)] = g(n)\,x(n)$$

Delay this by n_0 to get $y(n-n_0) = g(n-n_0)\,x(n-n_0)$ < (A)

The output $y(n, n_0)$ corresponding to $x(n-n_0)$ is given by

$$y(n, n_0) = T[x(n-n_0)] = g(n)\,x(n-n_0) \quad < \text{(B)}$$

(A) $\neq$ (B), i.e., $y(n, n_0) \neq y(n-n_0)$. Hence, the system is time-varying.

Example 1.4.14 Check for time–invariance the system $y(n) = a^n \cos(2\pi n/N)$.
Answer The output consists simply of a time-varying coefficient and is independent of the input $x(n)$. The output $y(n)$ due to input $x(n)$ is

$$y(n) = T[x(n)] = a^n \cos(2\pi n/N)$$

Delay this by n_0 to get $y(n-n_0) = a^{n-n_0} \cos(2\pi(n-n_0)/N)$ ◄ (A)

The output $y(n, n_0)$ due to input $x(n-n_0)$ is given by

$$y(n, n_0) = T[x(n-n_0)] = a^n \cos(2\pi n/N) \text{ ◄ (B)}$$

(A) ≠ (B), that is, $y(n, n_0) \neq y(n-n_0)$. Hence, the system is time-varying.

Example 1.4.15 Check the system $y(n) = n\, e^{|x(n)|}$ for time-invariance.
Answer Note time-varying coefficient, *n*. The output *y(n)* due to input *x(n)* is

$$y(n) = T[x(n)] = n\, e^{|x(n)|}$$

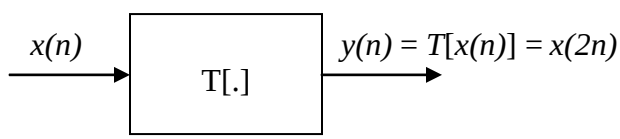
Delay this by n_0 to get $y(n-n_0) = (n-n_0)\, e^{|x(n-n_0)|}$ ◄ (A)

The output $y(n, n_0)$ due to input $x(n-n_0)$ is given by

$$y(n, n_0) = T[x(n-n_0)] = n e^{|x(n-n_0)|} \text{ ◄ (B)}$$

(A) and (B) are not equal. Hence, the system is time-varying.

Example 1.4.16 Test the system *y(n) = x(2n)* for time-invariance.



Answer This system represents ***time scaling***. That is, *y(n)* is a time-compressed version of *x(n)*, compressed by a factor of 2. For example, the value of *x* that occurred at 2*n* is occurring at *n* in the case of *y*. Intuitively, then, any time shift in the input will also be compressed by a factor of 2, and it is for this reason that the system is not time–invariant.

This is demonstrated by counter-example (Oppenheim & Willsky, p. 52). It can also be shown by following the formal procedure, which we shall do first below. For the input *x(n)* the output is

$$y(n) = T[x(n)] = x(2n)$$

Delay this output by n_0 to get

$$y(n-n_0) = x(2(n-n_0)) = x(2n-2n_0) \text{ ◄ (A)}$$

Next, for the input $x(n) = x(n-n_0)$ the output is

$$y(n, n_0) = x(2n) = x(2n-n_0) \text{ ◄ (B)}$$

(A) and (B) are not equal. So the system is *not* time-variant.

By counter example To show that the system *y(n) = x(2n)* is *not* time–invariant by way of a counter example consider the *x(n)* below:

$$x(n) = \quad 1, \qquad -2 \le n \le 2$$

0, otherwise

We shall show that $y(n, 2) \neq y(n-2)$. We express $x(n)$ in terms of unit step functions as

$$x(n) = u(n+2) - u(n-3)$$

We determine $y(n-2)$ by first obtaining $y(n)$ and then delaying it by 2 units:

$y(n)$ < $y(n) = x(2n) = u(2n+2) - u(2n-3)$

Delay < $y(n-2) = x(2(n-2)) = u(2(n-2)+2) - u(2(n-2)-3) = u(2n-2) - u(2n-7)$

Next we determine $y(n, 2)$ by first obtaining $x(n-2)$ and then the corresponding output $y(n,2)$:

$x(n-2)$ < $x(n-2) = u((n-2)+2) - u((n-2)-3) = u(n) - u(n-5) = x_2(n)$, say

$y(n, 2)$ < $y(n, 2) = x_2(2n) = u(2n) - u(2n-5)$

We can easily sketch $y(n, 2)$ and $y(n-2)$ and see that $y(n, 2) \neq y(n-2)$, and therefore the system $y(n) = x(2n)$ is *not* time–invariant.

Alternatively, this can be done entirely graphically.

Application of linearity – Convolution

An arbitrary sequence, $x(n)$, can be written as the weighted sum of delayed unit sample functions:

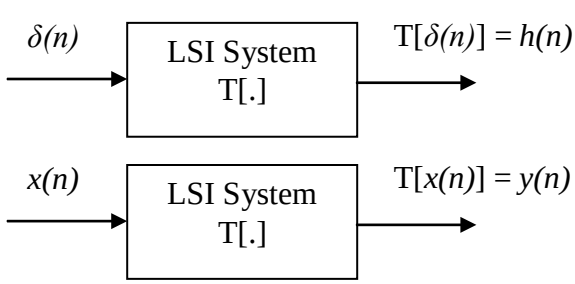
$$x(n) = \ldots + x(-2)\,\delta(n+2) + x(-1)\,\delta(n+1) + x(0)\,\delta(n) + x(1)\,\delta(n-1) + \ldots$$

$$= \sum_{k=-\infty}^{\infty} x(k)\,\delta(n-k)$$

So the response of a linear system to input $x(n)$ can be written down using the linearity principle, i.e., linear superposition. For a linear shift-invariant system whose impulse response is $T[\delta(n)] = h(n)$ the reasoning goes like this

- For an input $\delta(n)$ the output is $h(n)$. For an input $x(0)\,\delta(n)$ the output is $x(0)\,h(n)$ by virtue of scaling.
- For an input $\delta(n-1)$ the output is $h(n-1)$ by virtue of shift-invariance. For an input $x(1)\,\delta(n-1)$ the output is $x(1)\,h(n-1)$ by virtue of scaling.
- Therefore for an input of $x(0)\,\delta(n) + x(1)\,\delta(n-1)$ the output is $x(0)\,h(n) + x(1)\,h(n-1)$ by virtue of additivity.

This reasoning can be extended to cover all the terms that make up $x(n)$. In general the response to $x(k)\,\delta(n-k)$ is given by $x(k)\,h(n-k)$.



Given that

$$h(n) = T[\delta(n)], \qquad \text{and} \qquad x(n) = \sum_{k=-\infty}^{\infty} x(k)\ \delta(n-k)$$

we have

$$y(n) = T[x(n)] = T\left[\sum_{k=-\infty}^{\infty} x(k)\ \delta(n-k)\right]$$

Since $T[.]$ is linear we can apply linearity a countable infinite number of times to write

$$y(n) = \sum_{k=-\infty}^{\infty} T\left[x(k)\delta(n-k)\right] = \sum_{k=-\infty}^{\infty} x(k)T[\delta(n-k)]$$

In above equation since the system is shift-invariant we write $T[\delta(n-k)] = h(n-k)$. Else write $h_k(n)$ or $h(n, k)$ in place of $h(n-k)$. Thus for a linear shift-invariant system

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k)$$

Note that if the system is not specified to be shift-invariant we would leave the above result in the form

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n, k) \qquad \text{or} \qquad y(n) = \sum_{k=-\infty}^{\infty} x(k)h_k(n)$$

Then if shift-invariance is invoked we replace $h(n, k)$ with $h(n-k)$.

As in the case of continuous-time systems, the impulse response, $h(n)$, is determined assuming that the system has no initial energy; otherwise the linearity property does not hold, so that $y(n)$, as determined using the above equation, corresponds to only the forced response of the system.

The sum $\sum_{k=-\infty}^{\infty} x(k)h(n, k)$ is called the **convolution sum**, and is denoted $x(n) * h(n)$.

A *discrete-time linear shift-invariant system is completely characterized* by its unit sample response $h(n)$.

Theorem If a discrete-time system linear shift-invariant, T[.], has the unit sample response $T[\delta(n)] = h(n)$ then the output $y(n)$ corresponding to any input $x(n)$ is given by

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)\ h(n-k) = \sum_{k=-\infty}^{\infty} x(n-k)\ h(k)$$
$$= x(n) * h(n) \qquad = h(n) * x(n)$$

The second summation is obtained by setting $m = n-k$; then for $k = -\infty$ we have $m = +\infty$, and for $k = \infty$ we have $m = -\infty$. Thus

$$\sum_{k=-\infty}^{\infty} x(k)\ h(n-k) = \underbrace{\sum_{m=\infty}^{-\infty} x(n-m)\ h(m)}_{} = \sum_{k=-\infty}^{\infty} x(n-k)\ h(k)$$

m is a dummy variable. The order of summation (forward or backward) makes no difference. Hence change m to k and switch limits

Example 1.4.17 [**Linear Convolution]** Given the input $\{x(n)\}$ ={1, 2, 3, 1} and the unit sample response $\{h(n)\}$ = {4, 3, 2, 1} find the response $y(n) = x(n) * h(n)$.

Answer Since $x(k) = 0$ for $k < 0$ and $h(n - k) = 0$ for $k > n$, the convolution sum becomes

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

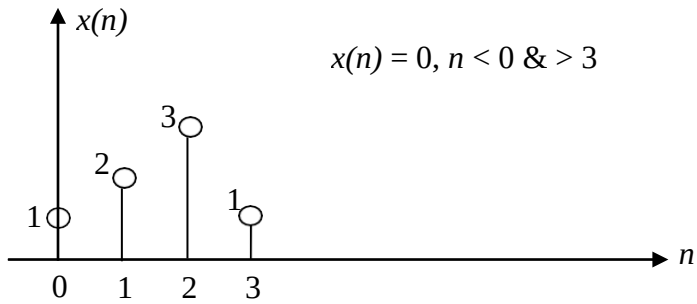
Now $y(n)$ can be evaluated for various values of n; for example, setting $n = 0$ gives $y(0)$. See table below. The product terms shown in ***bold italics*** need not be calculated; they are zero because the signal values involved are zero.

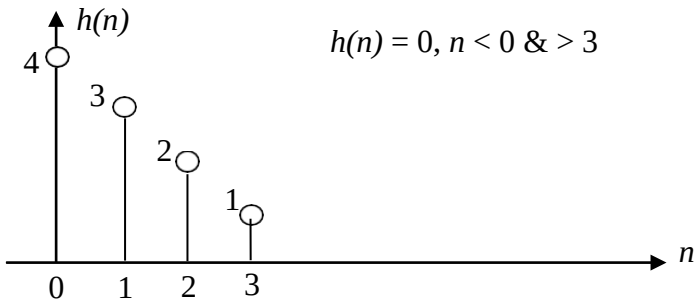
Linear Convolution of {x(n)} ={1, 2, 3, 1} and {h(n)} = {4, 3, 2, 1}

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

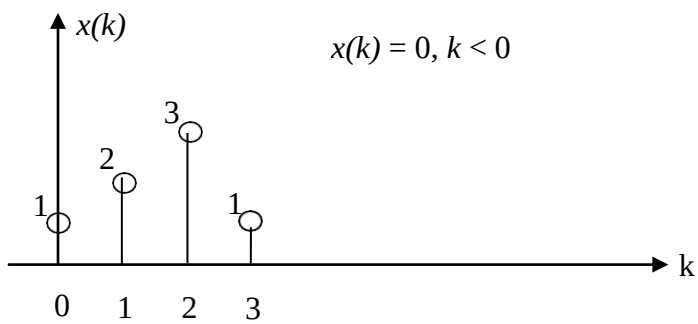
n = 0	$y(0) = \sum_{k=0}^{0} x(k)\, h(0-k)$	= x(0) h(0) = 1 . 4 = 4
n = 1	$y(1) = \sum_{k=0}^{1} x(k)\, h(1-k)$	= x(0) h(1) + x(1) h(0) = 1 . 3 + 2 . 4 = 11
n = 2	$y(2) = \sum_{k=0}^{2} x(k)\, h(2-k)$	= x(0) h(2) + x(1) h(1) + x(2) h(0) = 1 . 2 + 2 . 3 + 3 . 4 = 20
n = 3	$y(3) = \sum_{k=0}^{3} x(k)\, h(3-k)$	= x(0) h(3) + x(1) h(2) + x(2) h(1) + x(3) h(0) = 1 . 1 + 2 . 2 + 3 . 3 + 1 . 4 = 18
n = 4	$y(4) = \sum_{k=0}^{4} x(k)\, h(4-k)$	= x(0) ***h(4)*** + x(1) h(3) + x(2) h(2) + x(3) h(1) + ***x(4)*** h(0) = 1 . 0 + 2 . 1 + 3 . 2 + 1 . 3 + 0 . 4 = 11
n = 5	$y(5) = \sum_{k=0}^{5} x(k)\, h(5-k)$	= ***x(0) h(5) + x(1) h(4)*** + x(2) h(3) + x(3) h(2) + ***x(4) h(1) + x(5) h(0)*** = 3 . 1 + 1 . 2 = 5
n = 6	$y(6) = \sum_{k=0}^{6} x(k)\, h(6-k)$	= ***x(0) h(6) + x(1) h(5) + x(2) h(4)*** + x(3) h(3) + ***x(4) h(2) + x(5) h(1) + x(6) h(0)*** = 1 . 1 = 1
n = 7	$y(7) = \sum_{k=0}^{7} x(k)\, h(7-k)$	= ***x(0) h(7) + x(1) h(6) + x(2) h(5) + x(3) h(4) + x(4) h(3) + x(5) h(2) + x(6) h(1) + x(7) h(0)*** = 0
y(n) = 0 for n < 0 and n > 6		‡Product terms in ***bold italics*** are zero.

Sketches of the two sequences *x(n)* and *h(n)* are shown below.

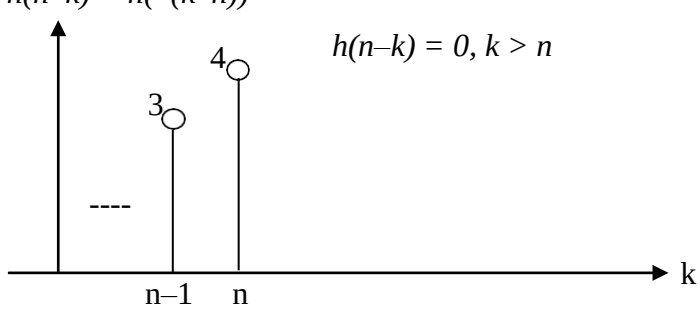




To do the convolution we need the sequences *x(k)* and *h(n–k)*, *k* being the independent variable. Of these *x(k)* is simply *x(n)* with *k* replacing *n*, shown below.



h(n–k) = h(–(k–n))



The sequence *h(–k)* is the reflected version of *h(k)*. If *h(–k)* is delayed by *n* samples we get *h(–(k–n))* that is *h(n–k)*, shown above.

For each value of *n* the sequences *x(k)* and *h(n–k)* are multiplied point by point and the products are added, yielding the value of *y (.)* for the corresponding *n*.

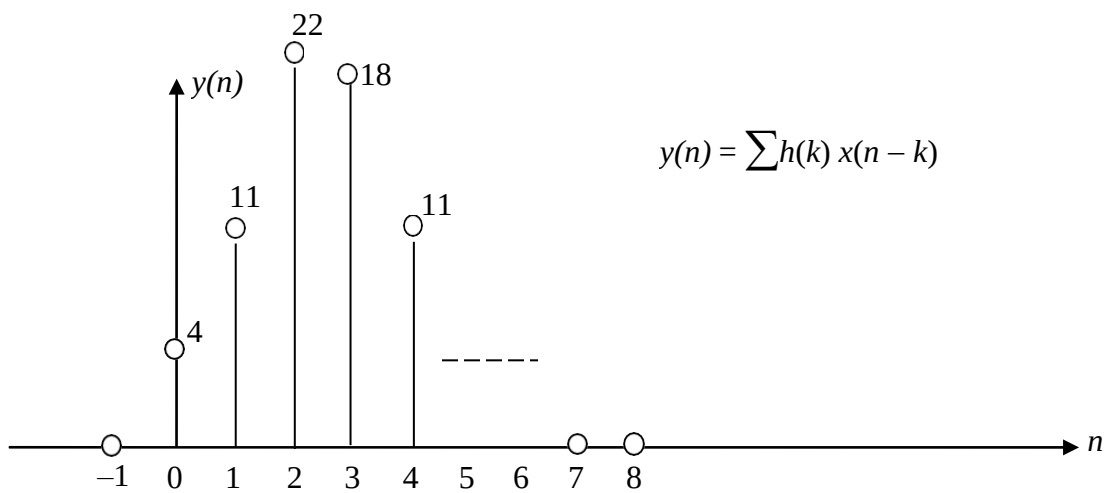
Tabular method The following tabular method uses the second of the two forms, viz., y(n) = $\sum h(k)\, x(n-k)$, for the convolution sum.

Tabular method of linear convolution

	k<	–3	–2	–1	0	1	2	3	4	5	6	7
	h(k) <				4	3	2	1				
n$	x(k) <				1	2	3	1				
0	x(**0**–k)	1	3	2	1							
1	x(**1**–k)		1	3	2	1						
2	x(**2**–k)			1	3	2	1					
3	x(**3**–k)				1	3	2	1				
4	x(**4**–k)					1	3	2	1			
5	x(**5**–k)						1	3	2	1		
6	x(**6**–k)							1	3	2	1	
7	x(**7**–k)								1	3	2	1
.	.											

n	y(n)
0	4
1	11
2	20
3	18
4	11
5	5
6	1
7	0
.	.

The sequence *y(n)* is shown plotted below. Note that L_y, the length of *y(n)*, equals the sum of the lengths of *x(n)* and *h(n)* minus 1: $L_y = L_x + L_h - 1$.

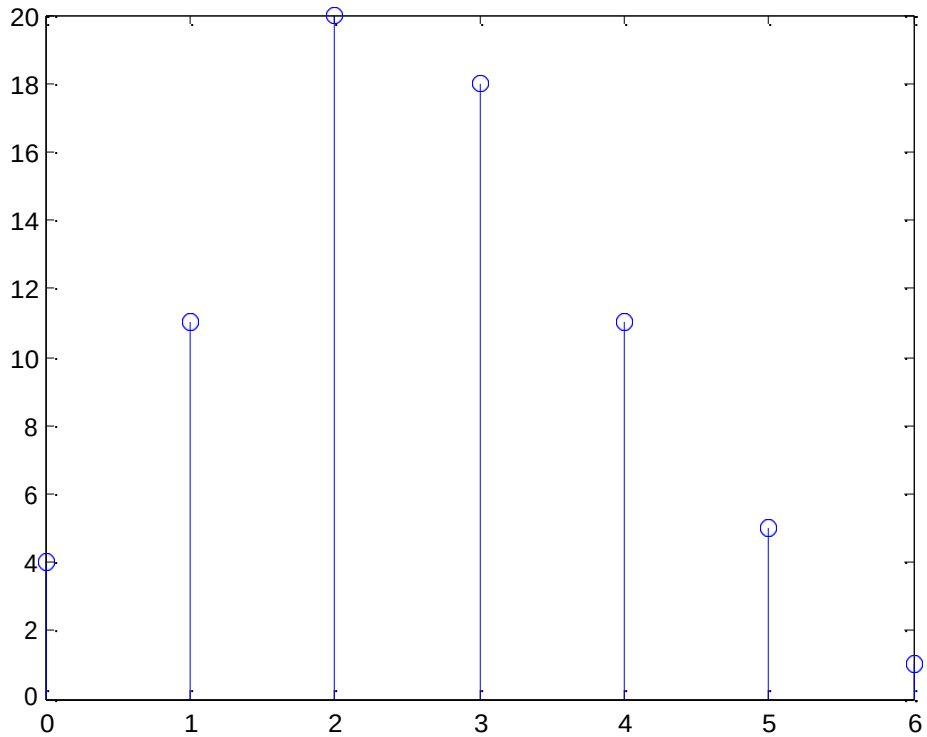


$$y(n) = \sum h(k)\, x(n-k)$$

MATLAB (Linear Convolution)

```
xn = [1, 2, 3, 1]; hn = [4, 3, 2, 1];
yn = conv(xn, hn)
M = length(xn) + length(hn) - 1;
m = 0: M-1;
stem(m, yn)
```

The result is: yn = 4 11 20 18 11 5 1



Matrix-vector multiplication [Proakis, Study Guide, p. 12] When the sequences *x(n)* and *h(n)* are of finite duration (as in this case) their linear convolution can also be implemented by using matrix-vector multiplication. We show below the formulation using the form

$$y(n) = \sum h(k)\, x(n-k)$$

as in the tabular method above. We arrange *h(.)* and *y(.)* as vectors (the latter with a length $L_y = L_x + L_h - 1$) and the (Toeplitz) matrix X (of size L_y rows and L_h columns) formed from the various shifted versions, *x(n–k)*:

$$\mathrm{h}(.) = \begin{bmatrix} 4 \\ 3 \\ 2 \\ 1 \end{bmatrix}, \qquad \mathrm{y}(.) = \begin{bmatrix} y(0) \\ y(1) \\ y(2) \\ y(3) \\ y(4) \\ y(5) \\ y(6) \end{bmatrix} \text{ and } \qquad \mathrm{X}(.,.) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 3 & 2 & 1 & 0 \\ 1 & 3 & 2 & 1 \\ 0 & 1 & 3 & 2 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Note that the Toeplitz matrix may be seen (except for the zeros) in the center part of the table given under "Tabular method". →e then evaluate *y* as the product *X* . *h*:

$$\begin{bmatrix} y(0) \\ y(1) \\ y(2) \\ y(3) \\ y(4) \\ y(5) \\ y(6) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 3 & 2 & 1 & 0 \\ 1 & 3 & 2 & 1 \\ 0 & 1 & 3 & 2 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 4 \\ 3 \\ 2 \\ 1 \end{bmatrix}$$

MATLAB has a function called *toeplitz* that generates the Toeplitz matrix. We could, of course, use the alternative form $\sum x(k)\, h(n-k)$ in which case we evaluate $y = H \,.\, x$.

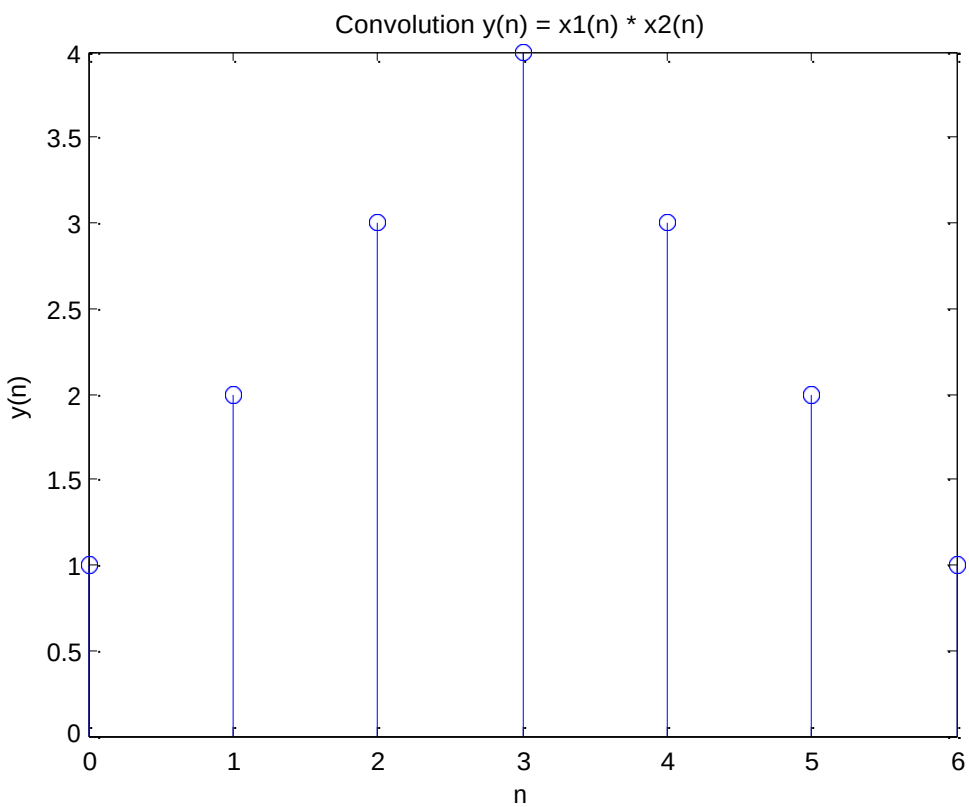
Example 1.4.18 [**Linear Convolution]** Given the finite length sequences below find $y(n) = x_1(n)*x_2(n)$.

$$x_1(n) = \begin{cases} 1, & 0 \le n \le 3, \\ 0, & \text{otherwise} \end{cases}$$

$$x_2(n) = \begin{cases} 1, & 0 \le n \le 3 \\ 0, & \text{otherwise} \end{cases}$$

Solution The MATLAB segment follows:

```
%Define sequences
n = 0:3; x1 = ones(size(n)), x2 = ones(size(n)),
%Length of output
M = length(x1) + length(x2) - 1; m = 0: M-1;
yn = conv(x1,x2); stem(m,yn)
xlabel('n'), ylabel('y(n)'), title('Convolution y(n) = x1(n) * x2(n)');
```



Example 1.4.19 Given $x_1(n) = n\,\{u(n+10) - u(n-20)\}$ and $x_2(n) = \cos(0.1\pi n)\,\{u(n) - u(n-30)\}$, find $y(n) = x_1(n) * x_2(n)$.

Solution The intervals over which the sequences are non-zero are defined below:

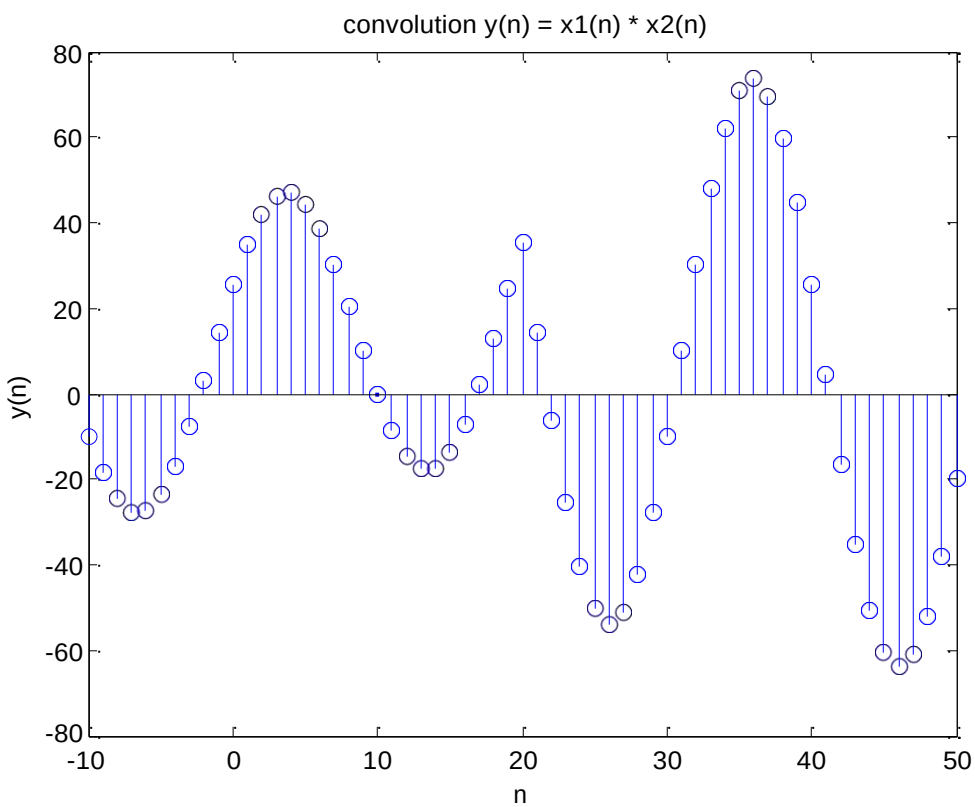
$$x_1(n) = \begin{cases} n, & N_0 \le n \le N_1, \quad N_0 = -10, N_1 = 20 \\ 0, & \text{otherwise} \end{cases}$$

$$x_2(n) = \begin{cases} \cos(0.1\pi n), & N_2 \le n \le N_3, \quad N_2 = 0,\ N_3 = 30 \\ 0, & \text{otherwise} \end{cases}$$

$$y(n) = \begin{cases} \text{Non-zero}, & N_4 \le n \le N_5, \quad N_4 = N_0 + N_2,\ N_5 = N_1 + N_3 \\ 0, & \text{otherwise} \end{cases}$$

For (the summation limits) N_4 and N_5 see DSP-HW. The MATLAB segment and plot are given below:

```
%First sequence over the range N0 to N1
N0 = -10; N1 = 20; n1 = N0:N1; x1 = n1;
%Second sequence over the range N2 to N3
N2 = 0; N3 = 30; n2 = N2:N3; x2 = cos(0.1 * pi * n2);
%Convolution limits
N4 = N0+N2; N5 = N1+N3; n = N4:N5;
yn = conv(x1,x2); stem(n, yn);
xlabel('n'), ylabel('y(n)'), title('Convolution y(n) = x1(n) * x2(n)');
```



Causality

The constraints of linearity and time-invariance define a class of systems that is represented by the convolution sum. The additional constraints of stability and causality define a more restricted class of linear time-invariant systems of practical importance.

Definition A discrete-time system is **causal** if the output at $n = n_0$ depends only on the input for $n \leq n_0$.

The word "causal" has to do with cause and effect; in other words, for the system to act up there must be an actual cause. A causal system does not anticipate future values of the input but only responds to actual, present, input. As a result, if two inputs to a causal system are identical up to some point in time n_0 the corresponding outputs must also be equal up to this same time. The synonyms of "causal" are "(physically) realizable" and "non-anticipatory".

We digress below to introduce memory-less versus dynamic systems and then resume with causality.

(Aside) Systems with and without memory A system is said to be **memory-less** or **static** if its output for each value of n is dependent only on the input at that same time but not on *past* or *future* inputs.

Examples of static systems

1. $y(n) = x(n)$ ← the identity system
2. $y(n) = a\, x(n) - x^2(n)$
3. A resistor R: $y(t) = R\, x(t)$ ($y(t)$ is voltage and $x(t)$ is current)

In many physical systems, memory is directly associated with storage of energy. A resistor has no storage of energy. But a circuit with capacitors and/or inductors has storage of energy and is a **dynamic system**, i.e., has **memory**. However, while storage of energy has to do with past inputs only, a static system is independent not only of *past* but also of *future* inputs.

Examples of systems with memory, i.e., dynamic systems:

1. $y(n) = \sum_{k=-\infty}^{n} x(k)$. This is an accumulator or summer. The output $y(n)$ depends on values of $x(.)$ prior to n such as $x(n-1)$ etc.
2. $y(n) = x(n-1)$. This is a delay element.
3. A capacitor C: $y(t) = \frac{1}{C}\int_{-\infty}^{t} x(\tau)\, d\tau$, ($y(.)$ is voltage and $x(.)$ is current).

(End of Aside)

Getting back to causality, *all memory-less systems are causal* since the output responds only to the current value of the input. In addition, some dynamic systems (such as the three listed above) are also causal.

An example of a noncausal system is $y(n) = x(n) + x(n+1)$ since the output depends on a future value, $x(n+1)$.

Although causal systems are of great importance, they are not the only systems that are of practical importance. For example, causality is not often an essential constraint in applications in which the *independent variable is not time*, such as in image processing. Moreover, in processing data that have been *recorded previously* (non real-time), as often happens with speech, geophysical, or meteorological signals, to name a few, we are by no means constrained to causal

processing. As another example, in many applications, including *historical* stock market analysis and demographic studies, we may be interested in determining a slowly varying trend in data that also contain higher frequency fluctuations about that trend. In this case, a commonly used approach is to average data over an interval in order to smooth out the fluctuations and keep only the trend. An example of such a noncausal averaging system is

$$y(n) = \frac{1}{2M+1} \sum_{k=-M}^{M} x(k)$$

Definition A discrete-time sequence $x(n)$ is called causal if it has zero values for $n < 0$, i.e., $x(n) = 0$ for $n < 0$.

Theorem A *linear shift-invariant system* with impulse response $h(n)$ is causal if and only if $h(n)$ is zero for $n < 0$.

Proof of the "If" part By convolution the output $y(n)$ is given by

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)\, h(n-k)$$

If $h(n) = 0$ for $n < 0$, then $h(n-k) = 0$ for $n-k < 0$ or $k > n$. So

$$y(n) = \sum_{k=-\infty}^{n} x(k)\, h(n-k) + \underbrace{\sum_{k=n+1}^{\infty} x(k)\, h(n-k)}_{=0}$$

$$= \sum_{k=-\infty}^{n} x(k)\, h(n-k)$$

Thus $y(n)$ at any time n is a weighted sum of the values of the input $x(k)$ for $k \le n$, that is, only the present and past inputs. Therefore, the system is causal.

Proof of the "Only If" part This is proved by contradiction, that is, if $h(n)$ is non zero for $n < 0$ then the system is noncausal. Let $h(n)$ be nonzero for $n < 0$:

$h(n)$ is non zero for $n < 0$

- $h(n-k)$ is non zero for $n-k < 0$ or $k > n$
- the 2nd sum above, $\sum_{k=n+1}^{\infty} x(k)\, h(n-k)$, is non zero
- $y(n)$ then depends on $x(n+1)$ and other future terms
- Hence, the system is noncausal

Example 1.4.20 Check the causality of the system $y(n) = x(-n)$.
Answer If n is some positive value then $y(n)$ depends only on past values of the input $x(.)$. But if n is negative, say $n = -2$, then

$y(-2) = x(-(-2)) = x(2)$, a future value of input

Hence the system is noncausal.

Example 1.4.21 Check the system $y(n) = x(n) \cos(n+1)$ for causality.
Answer Note that $x(.)$ is the input, not the cos (.). In this system, the output at any time n equals the input at that same time multiplied by a number that varies with time. We can write the equation as $y(n) = g(n)\, x(n)$, where $g(n) = \cos(n+1)$ is a time-varying function. Only the current

value of the input $x(.)$ determines the current output $y(.)$. Therefore the system is causal (and also memoryless).

Example 1.4.22 Check the system $y(n) = T[x(n)] = ne^{|x(n)|}$ for causality. **Answer** Causal.

Example 1.4.23 Check the system $y(n) = T[x(n)] = a^n \cos(2\pi n/N)$ for causality. **Answer** Causal.

Example 1.4.24 Check the system $y(n) = T[x(n)] = \cos(x(n))$ for causality. **Answer** Causal.

Example 1.4.25 Check the system $y(n) = T[x(n)] = x(-n+2)$ for causality.
Answer For $n \le 0$ the argument of x, viz., $-n+2$ will be ≥ 2. For example, if $n = 0$ we have

$y(0) = x(2)$, a future value of input

Hence the system is noncausal.

Example 1.4.26 Check the system $y(n) = T[x(n)] = \sum_{k=n_0}^{n} x(k)$ for causality. **Answer** Causal.

Example 1.4.27 Check the system $y(n) = T[x(n)] = \sum_{k=n-n_0}^{n+n_0} x(k)$ for causality.

Answer $y(n) = \sum_{k=n-n_0}^{n+n_0} x(k) = \left\{ \sum_{k=n-n_0}^{n} x(k) \right\} + \underbrace{x(n+1) + x(n+2) + \ldots + x(n+n_0)}_{\text{Future terms}}$

The system is noncausal since $y(n)$ depends on future values of the input.

(Aside) A **streamable** system is one that is either causal or *can be made causal* by adding an overall delay. The system $y(n) = x(n+1)$ is not causal but can be made so and is therefore streamable. But $y(n) = x(-n)$ is not streamable since it can"t be made causal.

Bounded input bounded output stability

Definition A sequence $x(n)$ is bounded if there exists a finite M such that $|x(n)| < M$ for all n. (Note that, as expressed here, M is a bound for negative values of $x(.)$ as well. Another way of writing this is $-M < x(n) < M$.)

As an example, the sequence $x(n) = [1+\cos 5\pi n]\, u(n)$ is bounded with $|x(n)| \le 2$. The sequence $x(n) = \left[\frac{(1+n)\sin 10n}{1+(0.8)^n} \right] u(n)$ is unbounded.

Definition A discrete-time system is bounded input-bounded output (BIBO) stable if every bounded input sequence $x(n)$ produces a bounded output sequence. That is, if $|x(n)| \le M < \infty$, then $|y(n)| \le L < \infty$.

BIBO stability theorem A *linear shift invariant system* with impulse response $h(n)$ is bounded input-bounded output stable if and only if S, defined below, is finite.

$$S = \sum_{k=-\infty}^{\infty} |h(k)| < \infty$$

i.e., the unit sample response is *absolutely summable*.

Proof of the "If" part Given a system with impulse response *h(n)*, let *x(n)* be such that $|x(n)| \le M$. Then the output *y(n)* is given by the convolution sum:

$$y(n) = \sum_{k=-\infty}^{\infty} h(k)x(n-k)$$

so that

$$|y(n)| = \left| \sum_{k=-\infty}^{\infty} h(k)\, x(n-k) \right|$$

Using the triangular inequality that the sum of the magnitudes ≥ the magnitude of the sum, we get

$$|y(n)| \le \sum_{k=-\infty}^{\infty} |h(k)x(n-k)|$$

Using the fact that the magnitude of a product is the product of the magnitudes,

$$|y(n)| \le \sum_{k=-\infty}^{\infty} |h(k)|\,|x(n-k)|$$

$$\le M \sum_{k=-\infty}^{\infty} |h(k)|$$

Thus, a sufficient condition for the system to be stable is that the unit sample response must be absolutely summable; that is,

$$\sum_{k=-\infty}^{\infty} |h(k)| < \infty \qquad \text{QED}$$

Proof of the "Only If" part That it is also a necessary condition can be seen by considering as input, the following bounded signal (this is the signum function),

$$x(k) = \operatorname{sgn}[h(n-k)] = \begin{cases} 1 & \text{where } h(n-k) > 0 \\ 0 & \text{where } h(n-k) = 0 \\ -1 & \text{where } h(n-k) < 0 \end{cases}$$

Or, equivalently,

$$x(n-k) = \operatorname{sgn}[h(k)] = \begin{cases} 1 & \text{where } h(k) > 0 \\ 0 & \text{where } h(k) = 0 \\ -1 & \text{where } h(k) < 0 \end{cases}$$

In the above we have implied that $M = 1$ (since M is some arbitrary finite number), and that $|x(n)| \le 1$. If, however, $|x(n)| \le M$ where M is finite but not equal to 1 we then will multiply the signum function by M. In either case *x(n)* is a bounded input. Thus

$$y(n) = \sum_{k=-\infty}^{\infty} h(k)\, x(n-k) = \sum_{k=-\infty}^{\infty} h(k) \operatorname{sgn}[h(k)] = \sum_{k=-\infty}^{\infty} |h(k)|$$

Clearly, if *h(n)* is not absolutely summable, *y(n)* will be unbounded. QED

For a causal system the BIBO stability condition becomes

$$\sum_{k=0}^{\infty} |h(k)| < \infty$$

Example 1.4.28 Evaluate the stability of the linear shift-invariant system with the unit sample response $h(n) = a^n u(n)$.
Answer Evaluate

$$S = \sum_{k=-\infty}^{\infty} |h(k)| = \sum_{k=-\infty}^{\infty} |a^k u(k)| = \sum_{k=0}^{\infty} |a^k| = \left| \sum_{k=0}^{\infty} |a|^k \right|$$

Here we have used the fact that the magnitude of a product ($|a^k|$) is the product of the magnitudes ($|a|^k$). The summation on the right converges if $|a| < 1$ so that S is finite,

$$S = \frac{1}{1-|a|}$$

and the system is BIBO stable.

Example 1.4.29 Evaluate the stability of the system $y(n) = T[x(n)] = n\, x(n)$.
Answer (Note that this is not a shift-invariant system, though it is linear as we determined elsewhere. The BIBO stability theorem applies to LSI systems. Note that if $x(n) = \delta(n)$ we get the result $y(n) = h(n) = 0$, for all n!)

If a system is suspected to be unstable, a useful strategy to verify this is to look for a specific bounded input that leads to an unbounded output. One such bounded input is $x(n) = u(n)$, the unit step sequence. The output then is $y(n) = n\, u(n)$, and as $n \to \infty$, $y(n)$ grows without bound. Hence the system is BIBO unstable.

Example 1.4.30 Examine $y(n) = T[x(n)] = e^{x(n)}$ for stability.
Answer This system is nonlinear as determined elsewhere. Its unit sample response is given by setting $x(n) = \delta(n)$ and is given below

$$h(n) = \begin{cases} e, & n = 0 \\ 1, & n \neq 0 \end{cases}$$

Here we are unable to find a bounded input that results in an unbounded output. So we proceed to verify that all bounded inputs result in bounded outputs. Let $x(n)$ be an arbitrary signal bounded by B an arbitrary positive number,

$$|x(n)| < B \qquad \text{or} \qquad -B < x(n) < B \qquad \text{for all } n$$

Then $y(n) = e^{x(n)}$ must satisfy the condition $e^{-B} < y(n) < e^{B}$. Or, the output is guaranteed to be bounded by e^{B}. Thus the system is BIBO stable.

Example 1.4.31 Check for stability the system $y(n) = a^n \cos(2\pi n/N)$
Answer The output is independent of the input. The system is stable for $|a| \leq 1$, otherwise it is unstable.

Example 1.4.32 $y(n) = x^2(n)$
Answer For any $|x(n)| < B$, we have $y(n) = x^2(n) < B^2$. Hence the system is stable.

Convolution - Properties

Property 1 The convolution operation is **commutative**, that is, $x(n) * h(n) = h(n) * x(n)$.

A linear shift-invariant system with input $x(n)$ and unit sample response $h(n)$ will have the same output as a linear shift-invariant system with input $h(n)$ and unit sample response $x(n)$.

Property 2 The convolution of $h(n)$ with $\delta(n)$ is, by definition, equal to $h(n)$.

That is, the convolution of any function with the δ function gives back the original function. Stated another way: The **identity sequence** for the convolution operator is the unit sample, or

$$x(n) * \delta(n) = \delta(n) * x(n) = x(n)$$

Property 3 The convolution of a delayed unit sample sequence with $x(n)$

$$x(n) * \delta(n-k) = x(n-k)$$

Property 4 Associativity

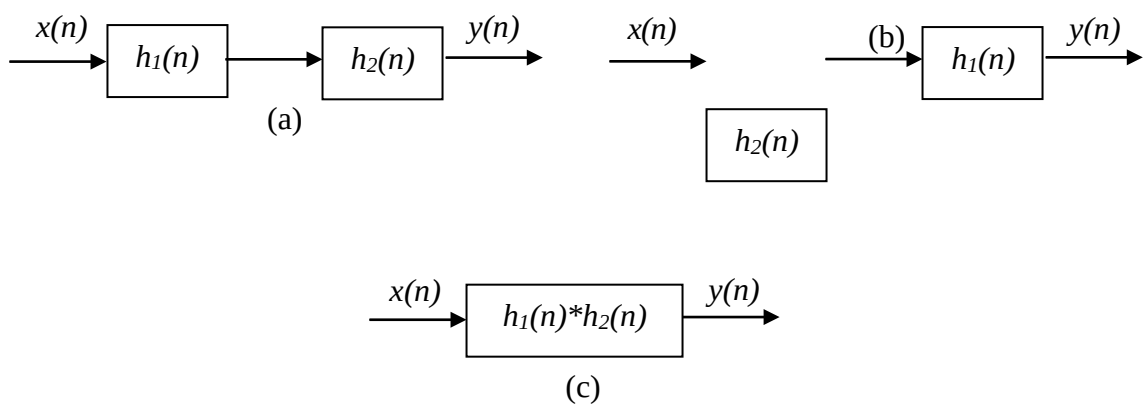
$$x(n) * (y(n) * z(n)) = (x(n) * y(n)) * z(n) = x(n) * y(n) * z(n)$$

Property 5 Distributivity over sequence addition.

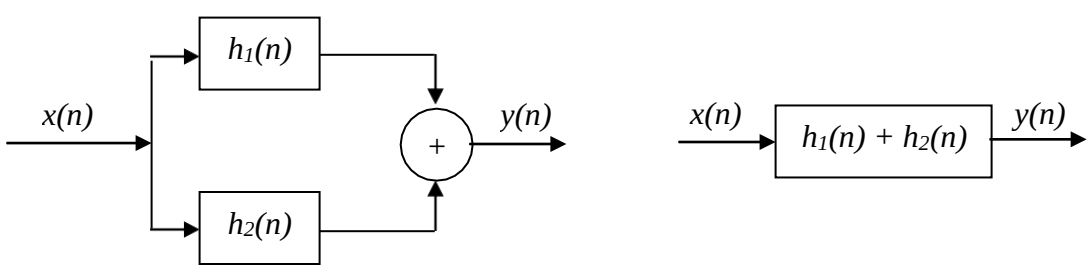
$$x(n) * (y(n) + z(n)) = x(n) * y(n) + x(n) * z(n)$$

Block diagram manipulation The properties of commutativity, associativity and distributivity enable us to determine the impulse response of series or parallel combinations of systems in terms of their individual impulse responses, as below:

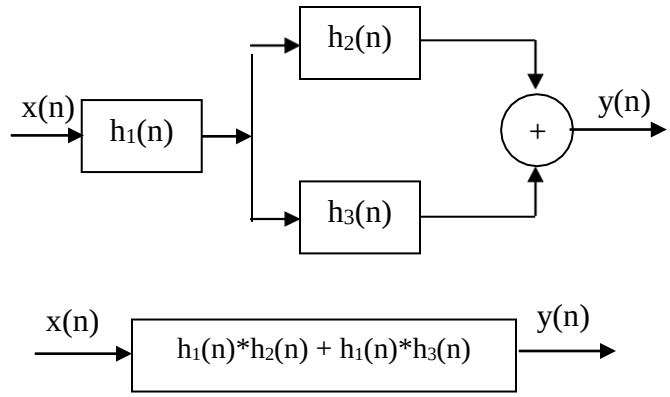
(1) The following three LSI systems have identical unit sample response:



(2) The following two LSI systems are equivalent:



(3) A combination of the above two is the series parallel combination:



Convolution – Overlap-and-add See Unit II.

Linear constant coefficient difference equations

A subclass of linear shift-invariant systems is that for which the input *x(n)* and output *y(n)* satisfy an Nth order linear constant coefficient difference equation, given by

$$a_0\, y(n) + a_1\, y(n-1) + \ldots + a_N\, y(n-N)$$
$$= b_0\, x(n) + b_1\, x(n-1) + \ldots + b_M\, x(n-M), \qquad a_0 \neq 0$$

This may be written in the more compact, though daunting, form

$$\sum_{k=0}^{N} a_k\, y(n-k) = \sum_{r=0}^{M} b_r\, x(n-r), \qquad a_0 \neq 0$$

If the system is causal, then we can rearrange the above equation to compute *y(n)* iteratively from the present input *x(n)* and the past inputs *x(n–1)*, *x(n–2)*, …, *x(n–M)* and the past outputs *y(n–1)*, *y(n–2)*, …, *y(n–N)*:

$$y(n) = -\sum_{k=1}^{N} \left(\frac{a_k}{a^0} \right) y(n-k) + \qquad \text{—} \quad ($$

If we think of the input as beginning at $n = 0$, then *y(n)* can be computed for all $n \geq 0$ once *y(–1)*, *y(–2)*, …, *y(–N)* are specified. This is an iterative solution.

(Aside) Some write the difference equation with the terms *y(n–1)* through *y(n–N)* on the right hand side with positive (symbolic) coefficients and the coefficient of *y(n)* = 1, thus

$$y(n) = b_0\, x(n) + b_1\, x(n-1) + \ldots + b_M\, x(n-M) + a_1\, y(n-1) + \ldots + a_N\, y(n-N)$$

If we need to use this form we shall use different symbols altogether as follows:

$$y(n) = \beta_0\, x(n) + \beta_1\, x(n-1) + \ldots + \beta_M\, x(n-M) + \alpha_1\, y(n-1) + \ldots + \alpha_N\, y(n-N)$$

(End of Aside)

Example 1.5.1 [First order system] For the first order system $y(n) - a\,y(n-1) = x(n)$ find the output sequence $y(n)$ assuming $y(n) = 0$ for all $n < 0$ and $x(n) = \delta(n)$. This corresponds to calculating the impulse response assuming zero initial conditions.

$$y(n) = a\,y(n-1) + x(n) = a\,y(n-1) + \delta(n)$$
$$n = 0:\ \ y(0) = a\,y(-1) + \delta(0) = a.\,0 + 1 = 1$$
$$n = 1:\ \ y(1) = a\,y(0) + \delta(1) = a\,.\,1 + 0 = a$$
$$n = 2:\ y(2) = a\,y(1) + \delta(2) = a\,.\,a + 0 = a^2$$
$$n = 3\ \ \ldots$$

Continuing the process we have $y(n) = a^n$, $n \geq 0$. This is also the unit sample response $h(n) = a^n u(n)$. It is not always possible to express $y(n)$ as an analytical expression (closed form) as above.

Note It is also possible to recast the above problem as a noncausal or negative-time system with $y(n) = 0$ for $n \geq 0$. In this case, solving for $y(n-1)$, we have

$$y(n-1) = \frac{1}{a}\big(y(n) - x(n)\big)$$

This can be recast (by letting $(n-1) = m$ etc.) as

$$y(n) = \frac{1}{a}\big(y(n+1) - x(n+1)\big),\ n < 0$$

The solution now is $y(n) = -a^n$, for $n < 0$, or the impulse response is $h(n) = -a^n u(-n-1)$.

Note Unless stated otherwise we shall generally assume that the difference equation represents a causal system.

Other techniques for solving difference equations Among other techniques is a method paralleling the procedure for solving linear constant coefficient differential equations, which involves finding and combining the **particular** and **homogeneous solutions**. Another method uses the ***z*-transform** (paralleling the Laplace transform). The **state variable approach** provides another *formulation* of the problem and solutions in the time as well as frequency domains.

Example 1.5.2 (Moving average filter) The three-term average $y(n) = \dfrac{x(n) + x(n-1) + x(n-2)}{3}$ as a lead-in to FIR and IIR, see below.

"FIR", "IIR", "Recursive" and "Nonrecursive" In the first example above the impulse response $h(n) = a^n$, $n \geq 0$ lasts for all positive time and is of infinite duration. In the second example (moving average) $h(n) = \{1/3, 1/3, 1/3\}$ which is of finite duration.

Definition If the unit sample response of a linear shift invariant system is of infinite duration, the system is said to be an **infinite impulse response (IIR)** system.

Definition If the unit sample response of a linear shift invariant system is of finite duration, the system is said to be a **finite impulse response (FIR)** system.

Theorem A causal linear shift invariant system characterized by

$$\sum_{k=0}^{N} a_k\, y(n-k) \;=\; \sum_{r=0}^{M} b_r\, x(n-r)$$

represents a finite impulse response (FIR) system if $a_0 \neq 0$, and $a_k = 0$ for $k = 1, 2, \ldots, N$. [Otherwise it could represent either an IIR or FIR system.] This is equivalent to saying that for an FIR system $N = 0$. For an FIR system then we have

$$a_0\, y(n) = \sum_{r=0}^{M} b_r\, x(n-r), \text{ or}$$

$$y(n) = \sum_{r=0}^{M} \left(\frac{b_r}{a_0}\right) x(n-r)$$

The above difference equation is identical to the convolution sum, and the (b_r/a_0) terms can be recognized as $h(r)$, the value of the unit sample response at time r, i.e., we can set $(b_r/a_0) = h_r = h(r)$. So the impulse response, $h(n)$, is given by

$$h(n) = \begin{cases} (b_n/a_0), & 0 \le n \le M \\ 0, & \text{otherwise} \end{cases}$$

which, obviously, is of finite duration.

Note: If the above difference equation were written so that $a_0 = 1$, we have $y(n) = \sum_{r=0}^{M} b_r' x(n-r)$.

In this case the impulse response consists simply of the coefficients b_r' of the $x(n-r)$ terms.

Iterative solution with initial conditions The LTI discrete-time system can be characterized by

$$\sum_{k=0}^{N} a_k\, y(n-k) = \sum_{r=0}^{M} b_r\, x(n-r), \qquad a_0 \neq 0, \text{ and } n \ge 0 \qquad \rightarrow \text{(A)}$$

Here N is the order of the difference equation. When written out in full the equation is

$$a_0\, y(n) + a_1\, y(n-1) + \ldots + a_N\, y(n-N)$$
$$= b_0\, x(n) + b_1\, x(n-1) + \ldots + b_M\, x(n-M), \qquad a_0 \neq 0, \text{ and } n \ge 0$$

The equation can be divided through by a_0 so that the coefficient of $y(n)$ is 1 or, alternatively, we could impose the equivalent condition that $a_0 = 1$.

An **alternative form** of the above equation is sometimes given as

$$\sum_{k=0}^{N} a_k\, y(n+k) = \sum_{r=0}^{M} b_r\, x(n+r), \qquad n \ge 0 \quad \rightarrow \text{(A‘‘)}$$

In this form, if the system is causal, we must have $M \le N$.

The solution to either one of the above equations can be determined (by analogy with the differential equation) as the sum of two components: (1) the **homogeneous solution**, which depends on the **initial conditions** assumed to be known, and (2) the **particular solution**, which depends on the **input**.

Computing $y(n)$ for successive values of n one after another is called an iterative solution. To obtain $y(n)$ for $n \ge 0$ from Eq. (A) we rearrange it as

$$y(n) = -\sum_{k=1}^{N} \left(\frac{a_k}{a^0}\right) y(n-k) + \qquad \text{—} \quad (\qquad , \rightarrow \text{(B)}$$

and evaluate $y(n)$ for $n = 1, 2, \ldots$ in an iterative manner. →e need the initial conditions $y(-1), y(-2), \ldots, y(-N)$. The initial conditions needed to solve for $y(n)$ using Eq. (A") in a similar fashion are $y(0), y(1), \ldots, y(N-1)$.

We may assume that the system described by Eq. (A) is in a condition of initial rest, that is, if $x(n) = 0$ for $n < 0$, then $y(n) = 0$ for $n < 0$ as well. With initial rest the system (A) is LTI *and* causal.

An equation of the form (A) or (B) is called a **recursive equation** since it specifies a recursive procedure to determine the output $y(n)$ in terms of the input and *previous output values*.

In the special case where $N = 0$, Eq. (B) reduces to

$$y(n) = \quad — \quad (\quad , \quad \rightarrow \text{(C)}$$

Here $y(n)$ is an explicit function of the present and previous values of the input only. Eq. (C) is called a **non-recursive equation**, since we do not recursively use previously computed values of the *output* in order to calculate the present value of the output.

Example 1.5.3 Find the solution to

$$y(n) - \frac{3}{\ }y(n-1) + \frac{1}{\ }y(n-2) = \frac{1}{\ } \quad , \qquad n \geq 0$$

with the initial conditions $y(-1) = 4$ and $y(-2) = 10$.

Answer Note that the input is $x(n) = ($. This is the iterative solution in the time domain. We can write $y(n) = \frac{3}{\ }y(n-1) - \frac{1}{\ }y(n-2) + \frac{1}{\ }$, and solve for $y(n)$ starting with $n = 0$:

y(n) – (y(n–1) + (y(n–2) = (, n ≥ 0

y(–1) = 4, and y(–2) = 10

n	**y(n) = (3/2) y(n–1) – (1/2) y(n–2) + (1/4)^n**	**y(n)**
0	**y(n) = (3/2) y(n–1) – (1/2) y(n–2) + (1/4)^n** **y(0) = (3/2) y(0–1) – (1/2) y(0–2) + (1/4)^0** **= (3/2) y(–1) – (1/2) y(–2) + (1/4)^0** **= (3/2) (4) – (1/2) (10) + (1)**	**2**
1	y(n) = (3/2) y(n–1) – (1/2) y(n–2) + (1/4)^n y(1) = (3/2) y(1–1) – (1/2) y(1–2) + (1/4)^1 = (3/2) y(0) – (1/2) y(–1) + (1/4)^1 = (3/2) (2) – (1/2) (4) + (1/4)	5/4
2	**y(n) = (3/2) y(n–1) – (1/2) y(n–2) + (1/4)^n** **y(2) = (3/2) y(2–1) – (1/2) y(2–2) + (1/4)^2** **= (3/2) y(1) – (1/2) y(0) + (1/4)^2** **= (3/2) (5/4) – (1/2) (2) + (1/16)**	**15/16**
3	y(n) = (3/2) y(n–1) – (1/2) y(n–2) + (1/4)^n y(3) = (3/2) y(3–1) – (1/2) y(3–2) + (1/4)^3 = (3/2) y(2) – (1/2) y(1) + (1/4)^3 = (3/2) (15/16) – (1/2) (5/4) + (1/64)	51/64
4	**y(n) = (3/2) y(n–1) – (1/2) y(n–2) + (1/4)^n** **y(4) = (3/2) y(4–1) – (1/2) y(4–2) + (1/4)^4** **= (3/2) y(3) – (1/2) y(2) + (1/4)^4** **= (3/2) (51/64) – (1/2) (15/16) + (1/256)**	**?**
.	Etc.	.

The solution is

$$y(n) = \left\{ 2, \frac{5}{4}, \frac{15}{16}, \frac{51}{64}, \ldots \right\}$$

This procedure does not, in general, yield an analytical expression for *y(n)*, that is, a closed-form solution. But it is easily implemented on a digital computer.

Example 1.5.4 [MATLAB] [Zero initial conditions] In the context of MATLAB, we may use *filter(b, a, x)* to generate the sequence *y(n)*. In MATLAB the coefficients of *y(.)* and *x(.)* are numbered slightly differently as below:

$$a_1\, y(n) + a_2\, y(n-1) + a_3\, y(n-2) + \ldots = b_1\, x(n) + b_2\, x(n-1) + b_3\, x(n-2) + \ldots$$

From the difference equation

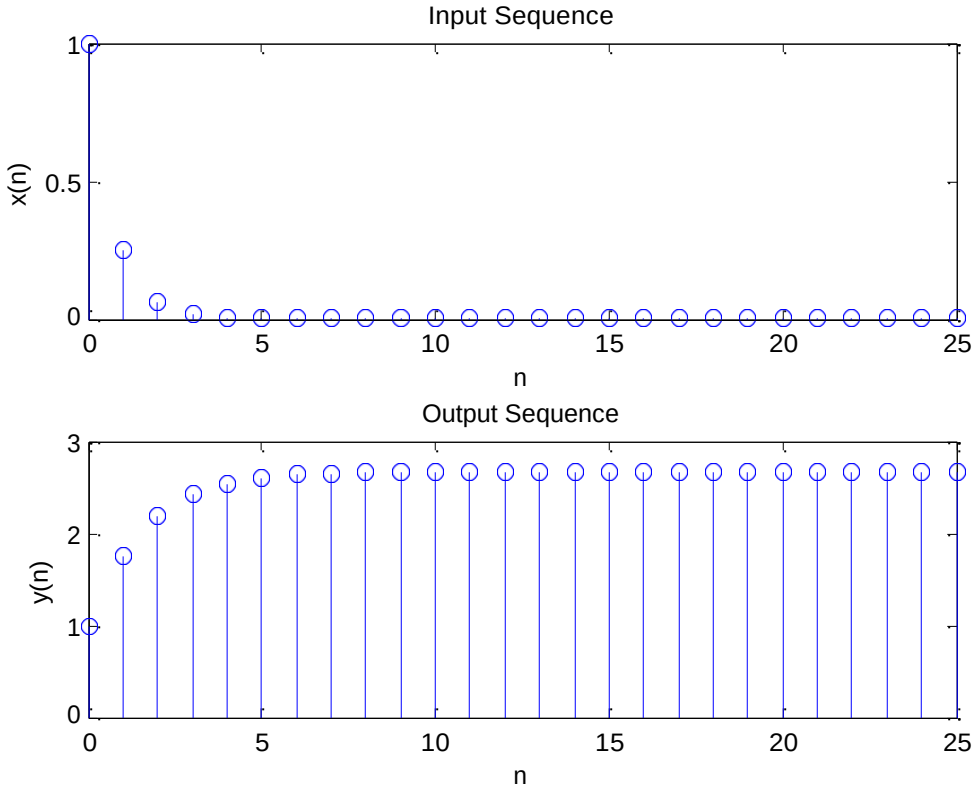
$$y(n) - (3/2)\, y(n-1) + (1/2)\, y(n-2) = (1/4)^n, \qquad n \geq 0$$

we note that the input is $x(n) = (1/4)^n$ and the coefficients of *y(.)* and *x(.)* give us the *a* and *b* vectors, respectively: *a* = [1, -1.5, 0.5] and *b* = [1]. The following segment generates the output with initial conditions taken as zero.

```
%Zero initial conditions
b = [1]; a = [1, -1.5, 0.5];
n = 0:25; xn = (1/4) .^n; %".^" means element-by-element exponentiation
yn = filter(b, a, xn), %Or, yn = filter(1, a, xn)
subplot(2, 1, 1), stem(n, xn);
xlabel('n'), ylabel('x(n)'); title('Input Sequence');
subplot(2, 1, 2), stem(n, yn);
xlabel('n'), ylabel('y(n)'); title('Output Sequence');
```

The solution is:

yn = [1 1.75 2.1875 2.4219 2.5430 2.6045 2.6355 … 2.6667 2.6667 2.6667]



Example 1.5.5 [MATLAB] [Non-zero initial conditions] Find the solution to the difference equation

$$y(n) - (3/2)y(n-1) + (1/2)\ y(n-2) = (1/4)^n, \qquad n \geq 0$$

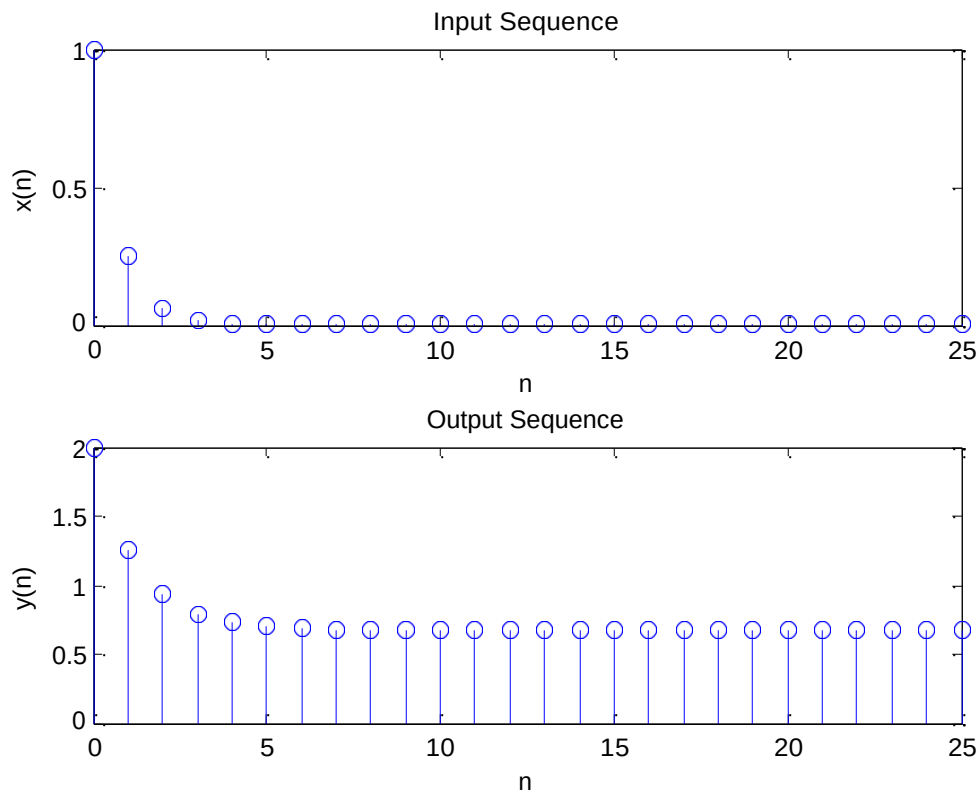
with the initial conditions $y(-1) = 4$ and $y(-2) = 10$.

If the problem specifies non-zero initial conditions, they must first be converted to "equivalent initial conditions" for the *filter* function to work. In this problem $y(-1) = 4$ and $y(-2) = 10$. These are specified as a vector *yic* = [4, 10] and used to generate the equivalent initial conditions *eic* by the function *filtic(b, a, yic)*. The equivalent initial conditions are then used to generate the filter output through *filter(b, a, xn, eic)*. The MATLAB segment follows:

```
%Non-zero initial conditions
b = [1]; a = [1, -1.5, 0.5]; yic = [4, 10];
n = 0:25; xn = (1/4) .^n; %".^" means element-by-element exponentiation
%
%Equivalent initial conditions
eic = filtic(b, a, yic);
yn = filter(b, a, xn, eic)
subplot(2, 1, 1), stem(n, xn);
xlabel('n'), ylabel('x(n)'); title('Input Sequence');
subplot(2, 1, 2), stem(n, yn);
xlabel('n'), ylabel('y(n)'); title('Output Sequence');
```

The output is:

yn = [2 1.25 0.9375 0.7969 0.7305 0.6982 0.6824…0.6667 0.6667 0.6667]



(Omit) Recursive realization of FIR system – a simple example The moving average of the signal *x(n)* is given by

$$y(n) = \frac{x(n)+x(n-1)+x(n-2)+\ldots+x(n-M)}{M+1}$$

Since past values of *y(.)* are not used in computing *y(n)* this is a nonrecursive implementation of the FIR filter. The impulse response is

$$\begin{array}{l} n = \\ h(n) = \end{array} \begin{array}{c} 0 \qquad\qquad\qquad\qquad\qquad\qquad M \\ \left\{ \frac{1}{M+1}, \frac{1}{M+1}, \frac{1}{M+1}, \ldots, \frac{1}{M+1} \right\} \end{array}$$

A total of M+1 terms

which consists of (*M+1*) samples or coefficients. We recognize, by replacing *n* by *(n–1)* in the above equation, that

$$y(n-1) = \frac{x(n-1)+x(n-2)+\ldots+x(n-M)+x(n-M-1)}{M+1} \qquad < (1)$$

By adding and subtracting $\frac{x(n-M-1)}{M+1}$ on the right hand side of the equation for *y(n)* we can arrive at a recursive implementation of the moving average:

$$y(n) = \frac{x(n)+x(n-1)+x(n-2)+\ldots+x(n-M)+x(n-M-1)-x(n-M-1)}{M+1}$$

Using eq. (1) we can write *y(n)* as

$$y(n) = y(n-1) + \frac{x(n)-x(n-M-1)}{M+1}$$

which clearly is a recursive implementation since it involves *y(n–1)*, a past value of *y*.

(End of Omit)

Further nomenclature If we take $a_0 = 1$ our standard difference equation becomes

$$y(n) + a_1\, y(n-1) + \ldots + a_N\, y(n-N) = b_0\, x(n) + b_1\, x(n-1) + \ldots + b_M\, x(n-M)$$

This is an Nth order difference equation.

A **moving average (MA)** filter is one with its output dependent on the present and previous inputs. The average here is a *weighted average*, the weights being the *b* coefficients. In terms of the above difference equation this corresponds to $N = 0$ and the difference equation becomes

$$y(n) = b_0\, x(n) + b_1\, x(n-1) + \ldots + b_M\, x(n-M)$$

There is some variation on the form of this equation. As given here it has (M+1) coefficients and M delay elements. Sometimes it is more convenient to use this with M coefficients rather than (M+1). Either way its *order* is $N = 0$. Note that we have tied the *order* to the oldest term in *y(.)* and not to the oldest term in *x(.)*. See ***Aside*** below.

An **autoregressive (AR)** filter is one with its output dependent on the present input (but not previous inputs) and previous outputs – this is a *purely recursive* system. In terms of the above difference equation this corresponds to $M = 0$ and is of the form

$$y(n) = b_0\, x(n) - a_1\, y(n-1) - \ldots - a_N\, y(n-N)$$

More generally, an **autoregressive moving average** (**ARMA)** filter has its output depend on the present input, M previous inputs, and N previous outputs. In terms of the linear constant coefficient difference equation this has the form

$$y(n) = b_0\, x(n) + b_1\, x(n-1) + \ldots + b_M\, x(n-M) - a_1\, y(n-1) - \ldots - a_N\, y(n-N)$$

(Aside) With $a_0 = 1$ we have

$$H(z) = \frac{\sum_{i=0}^{M} b^i z^{-i}}{1 + \sum_{i=1}^{N} a_i z^{-i}}$$

This represents an IIR filter if at least one of a_1 through a_N is nonzero, and all the roots of the denominator are not canceled exactly by the roots of the numerator. For example, in the system, $H(z) = (1 - z^{-8})/(1 - z^{-1})$, the single pole at $z = 1$ is canceled exactly by the zero at $z = 1$, making *H(z)* a finite polynomial in z^{-1}, that is, an FIR filter.

In general, there are M finite zeros and N finite poles. There is no restriction that M should be less than or greater than or equal to N. In most cases, especially digital filters derived from analog designs, M will be less than or equal to N. Systems of this type are called N^{th} order systems.

When $M > N$, the order of the system is no longer unambiguous. In this case, *H(z)* may be taken to be an Nth order system in cascade with an FIR filter of order ($M - N$).

When $N = 0$, as in the case of an FIR filter, according to our convention the *order* is 0; it is more useful in this case to focus on M and call it an FIR filter of M stages or (M+1) coefficients.

(End of Aside)

Fourier analysis of discrete-time signals and systems

Note For the discrete-time Fourier transform some authors (Oppenheim & Schafer, for instance) use the symbol $X(e^{j\omega})$ while others (Proakis, for instance) use the symbol $X(\omega)$. The symbol ω is used for digital frequency (radians per sample or just radians) and the symbol Ω for the analog frequency (radians/sec). Some authors, on the other hand, use just the opposite of our convention, that is, ω for the analog frequency (radians/sec) and Ω for the digital frequency (radians).

Discrete-time Fourier transform (DTFT) For the continuous-time signal $x(t)$, the Fourier transform is

$$\mathcal{F}\{x(t)\} = X(\Omega) = \int_{-\infty}^{\infty} x(t)\, e^{-j\Omega t}\, dt$$

The *impulse-train* sampled version, $x_s(t)$, is given by

$$x_s(t) = x(t) \sum_{n=-\infty}^{\infty} \delta(t - nT)$$

So the Fourier transform of $x_s(t)$ is given by

$$X_s(\Omega) = \int_{-\infty}^{\infty} x_s(t) e^{-j\Omega t} dt = \int_{-\infty}^{\infty} \left(x(t) \sum_{n=-\infty}^{\infty} \delta(t - nT) \right) e^{-j\Omega t} dt$$

$$= \sum_{n=-\infty}^{\infty} x(nT)\, e^{-j\Omega nT}$$

where the last step follows from the sifting property of the δ function. Replace ΩT by ω the discrete-time frequency variable, that is, the **digital frequency**. Note that Ω has units of radians/second, and ω has units of radians (/sample). This change of notation gives the **discrete-time Fourier transform**, $X(\omega)$, of the discrete-time signal $x(n)$, obtained by sampling $x(t)$, as

$$X(\omega) = \mathcal{F}\{x(n)\} = \sum_{n=-\infty}^{\infty} x(n)\, e^{-j\omega n}$$

Note that this defines the discrete-time Fourier transforms of *any* discrete-time signal $x(n)$. The transform exists if $x(n)$ satisfies a relation of the type

$$\sum_{n=-\infty}^{\infty} |x(n)| < \infty \qquad \text{or} \qquad \sum_{n=-\infty}^{\infty} |x(n)|^2 < \infty$$

These conditions are sufficient to guarantee that the sequence has a discrete-time Fourier transform. As in the case of continuous-time signals there are signals that neither are absolutely summable nor have finite energy, but still have a discrete-time Fourier transform. (See also p. 22, O & S.)

Discrete-time Fourier transform of (non-periodic) sequences The Fourier transform of a general discrete-time sequence tells us what the frequency content of that signal is.

Definition The Fourier transform $X(e^{j\omega})$ of the sequence $x(n)$ is given by

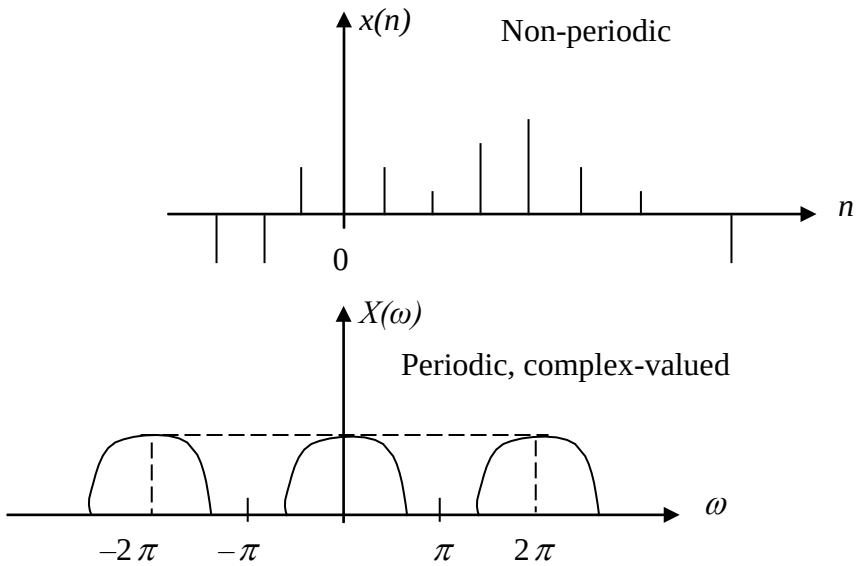
$$\mathcal{F}\{x(n)\} = X(\omega) = \sum_{n=-\infty}^{\infty} x(n)\, e^{-j\omega n} \qquad < \text{(A)}$$

The inverse Fourier transform is given by

$$\mathcal{F}^{-1}\{X(\omega)\} = x(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{j\omega})e^{j\omega n}d\omega \qquad < \text{(B)}$$

Equations (A) and (B) are called the Fourier transform pair for a sequence *x(n)* with *X(ω)* thought of as the *frequency content* of the sequence *x(n)*. Equation (A) is the analysis equation and equation (B) is the synthesis equation. Since *X(ω)* is a periodic function of ω, we can think of *x(n)* as the Fourier coefficients in the Fourier series representation of *X(ω)*. That is, equation (A), in fact, expresses *X(ω)* in the form of a Fourier series.

The sketch below sums up the relationship between the time and frequency domains. The periodicity is 2π. From the relation $\omega = \Omega T$ we can deduce that at the point $\omega = 2\pi$ on the horizontal axis $\Omega = \omega/T = 2\pi/T = 2\pi F_s = \Omega_s$. In other words, in terms of the analog frequency variable the point $\omega = 2\pi$ corresponds to $\Omega = \Omega_s$ or $F = F_s$.



(Omit) Relationship to the z-transform The *z*-transform *X(z)* and the Fourier transform *X(ω)* are given by

$$X(z) = \sum_{n=-\infty}^{\infty} x(n)\, z^{-n} \qquad \text{and} \qquad X(\omega) = \sum_{n=-\infty}^{\infty} x(n)\, e^{-j\omega n}$$

Comparing the two we deduce the relationship as

$$X(z)\big|_{z=e^{j\omega}} \equiv X(\omega) = \sum_{n=-\infty}^{\infty} x(n)\, e^{-j\omega n}$$

The *z*-transform evaluation on the unit circle gives the Fourier transform of the sequence *x(n)*. The *z*-transform of *x(n)* can be viewed as the Fourier transform of the sequence $\{x(n)\, r^{-n}\}$, that is, *x(n)* multiplied by an exponential sequence r^{-n}. This can be seen by setting $z = r\, e^{j\omega}$ in the defining equation of *X(z)*:

$$X(z) = \sum_{n=-\infty}^{\infty} x(n)\left(r\, e^{j\omega}\right)^{-n} = \sum_{n=-\infty}^{\infty} x(n)\, r^{-n}\, e^{-j\omega n}$$

z — *x(n)* multiplied by the exponential sequence r^{-n}

Z-transform of a periodic sequence Consider a sequence *x(n)* that is periodic with period *N* so that *x(n) = x(n+kN)* for any integer value of *k*. Such a sequence cannot be represented by its z-transform, since there is no value of *z* for which the z-transform will converge.
(End of Omit)

Example 1.6.1 [Cf. Example 4.2.3, Proakis, 4th Ed.] For the exponential sequence $x(n) = a^n u(n)$, $|a| < 1$, the DTFT is

$$X(e^{j\omega}) = \sum_{n=0}^{\infty} a^n e^{-j\omega n} = \sum_{n=0}^{\infty} \left(ae^{-j\omega}\right)^n = \frac{1}{1-ae^{-j\omega}} = \frac{1}{1-a(\cos\omega - j\sin\omega)}$$

We shall put this in the form $X(\omega)$= *Magnitude* $\{X\}\, e^{j\,Phase\{X\}} = |X(\omega)|\, e^{j\angle X(\omega)}$ from which the magnitude and phase will be extracted. The denominator (Dr.) is

$$\text{Dr.} = 1 - a\cos\omega + ja\sin\omega = \sqrt{(1-a\cos\omega)^2 + a^2\sin^2\omega}\; e^{j\tan^{-1}\left|\frac{a\sin\omega}{1-a\cos\omega}\right|}$$

Thus

$$X(\omega) = |X(\omega)|\, e^{j\angle X(\omega)} = \frac{1}{\sqrt{(1+a^2-2a\cos\omega)}} e^{-j\tan^{-1}\left(\frac{a\sin\omega}{1-a\cos\omega}\right)}$$

The magnitude and phase are: $|X(\omega)| = \dfrac{1}{\sqrt{(1+a^2-2a\cos\omega)}}$ and $\angle X(\omega) = -\tan^{-1}\left(\dfrac{a\sin\omega}{1-a\cos\omega}\right)$

Plots of $|X|$ and $\angle X$ are shown. Note that $X(\omega)$ is periodic and that the magnitude is an even function of ω and the phase is an odd function. (See below on the notation $|X|$ and $\angle X$).

The value of $X(e^{j\omega})$ at $\omega = 0$ is

$$|X(\omega)|_{\omega=0} = \frac{1}{\sqrt{(1+a^2-2a\cos 0)}} = \frac{1}{1-a}$$

$$\angle X(\omega)\Big|_{\omega=0} = -\tan^{-1}\left(\frac{a\sin 0}{1-a\cos 0}\right) = 0$$

Similarly, at $\omega = \pi$ we have $|X(\omega)|_{\omega=\pi} = 1/(1+a)$ and $\angle X(\omega)|_{\omega=\pi} = 0$.

Phase angle of a complex number In calculating the phase angle of a complex number, $z = Re(z) + j\,Im(z)$, a hand held calculator, typically uses the formula $\tan^{-1}\{\mathrm{Im}(z)/\mathrm{Re}(z)\}$, and returns an answer in the range $-\pi/2 <$ angle $\le \pi/2$. Thus for both $(1-j)$ and $(-1+j)$ the phase angle so calculated is $\pi/4$. However, $(1-j)$ is in the 4th quadrant with $\angle(1-j) = -\pi/4$ whereas $(-1+j)$ is in the 2nd quadrant with $\angle(-1+j) = 3\pi/4$. MATLAB has a function *angle* which takes into account the real and imaginary parts separately (instead of their ratio) and calculates the "four-quadrant inverse tangent".

Example 1.6.2 [MATLAB *fplot*] To illustrate the magnitude and phase plots of the DTFT we take $a = 0.8$ in the exponential sequence $x(n) = a^n u(n)$, $|a| < 1$, treated above. Thus $x(n) = (0.8)^n u(n)$. We need only plot over the interval $-\pi \le \omega \le \pi$ or $0 \le \omega \le 2\pi$. To show, visually, the periodicity we have plotted over $-3\pi \le \omega \le 3\pi$. We have

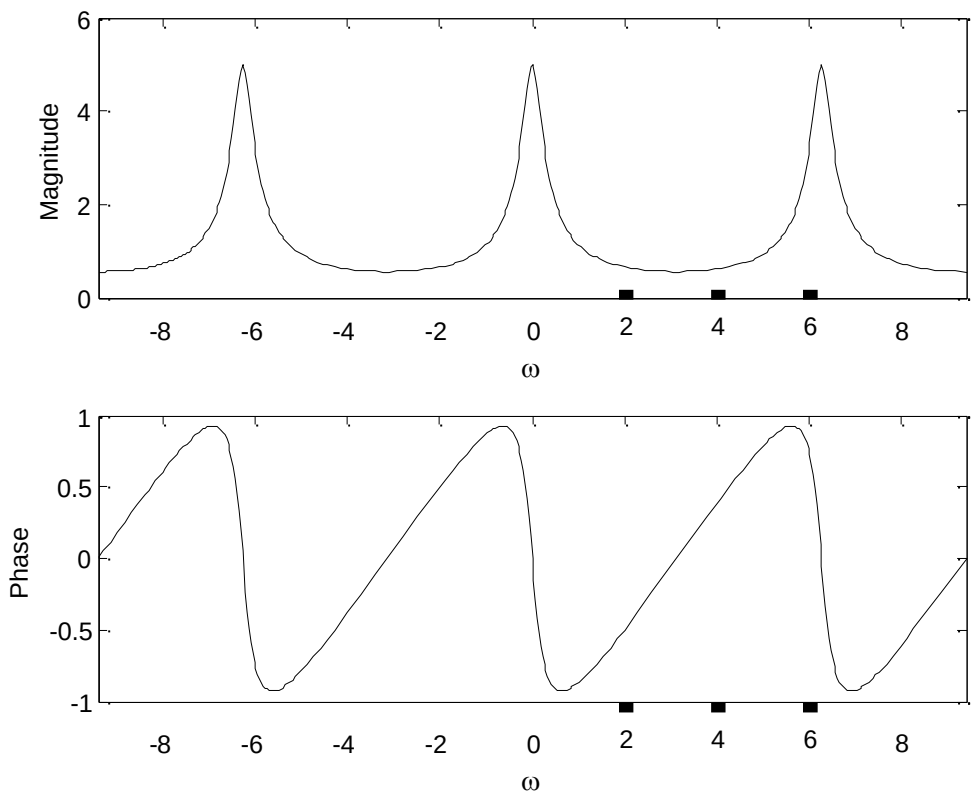
$$X(e^{j\omega}) = \sum_{n=0}^{\infty} a^n e^{-j\omega n} = \sum_{n=0}^{\infty} \left(ae^{-j\omega}\right)^n = \frac{1}{1-ae^{-j\omega}} = \frac{1}{1-0.8\,e^{-j\omega}}$$

In the MATLAB program segment that follows we write an algebraic expression for $X(e^{j\omega}) = \dfrac{1}{1-0.8\,e^{-j\omega}}$ as (1) / (1-0.8*exp(-j*w)). Note that we have used „w" for ω and the plot ranges from -2π to 2π. Both ω and the phase, $\angle X(\omega)$, are in radians. The parameter 'k' means that the plot/display is in black "color".

```
subplot(2,1,1);fplot('abs((1)/(1-0.8*exp(-j*w)))', [-3*pi,3*pi], 'k');
xlabel('\omega');ylabel('Magnitude');
subplot(2,1,2);fplot('angle((1)/(1-0.8*exp(-j*w)))', [-3*pi,3*pi], 'k');
xlabel('\omega');ylabel('Phase');
```

$$\left|X(\omega)\right|_{\omega=0} = \frac{1}{1-a} = \frac{1}{1-0.8} = 5$$

$$\left|X(\omega)\right|_{\omega=\pi} = 1/(1+a) = 1/1.8 = 0.555$$



Example 1.6.3 [MATLAB *freqz*] We repeat the frequency response plots using the *freqz* function. Taking a = 0.8 as before, we have $x(n) = (0.8)^n u(n)$ and

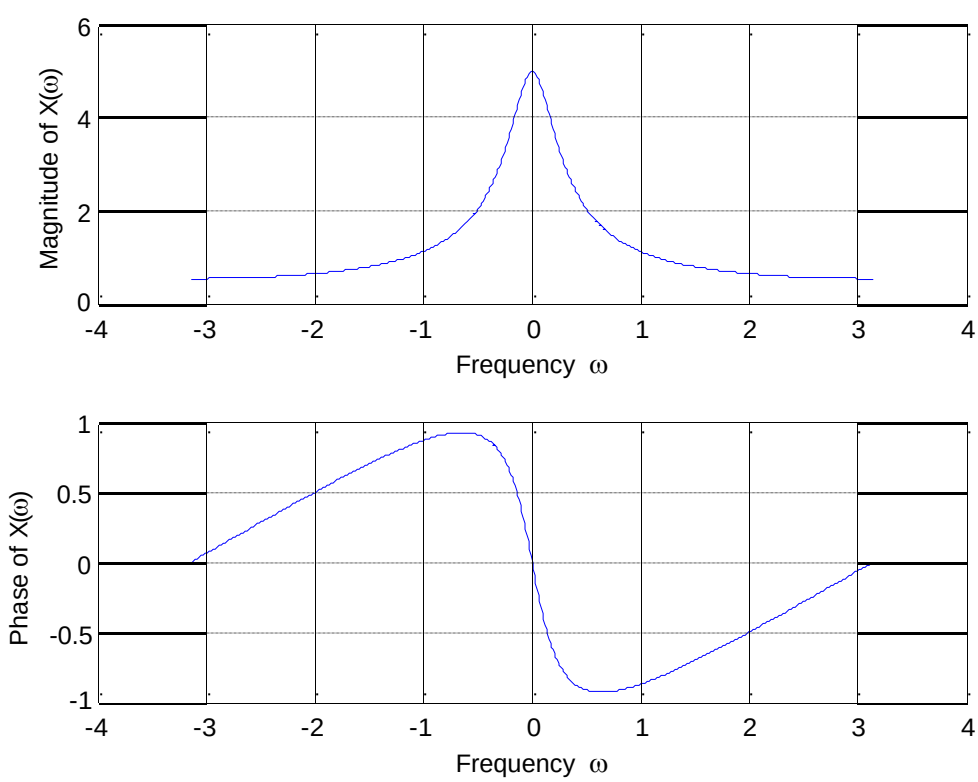
$$X(e^{j\omega}) = \sum_{n=0}^{\infty} a^n e^{-j\omega n} = \sum_{n=0}^{\infty} \left(ae^{-j\omega}\right)^n = \frac{1}{1-ae^{-j\omega}} = \frac{1}{1-0.8\,e^{-j\omega}}$$

Instead of writing an algebraic expression for $X(e^{j\omega}) = \dfrac{1}{1-0.8\,e^{-j\omega}}$ as in the previous example, we shall now specify the numerator and denominator in terms of their coefficients. We shall use the following convention to specify the parameters of the function

$$X(e^{j\omega}) = \frac{b(1)+b(2)e^{-j\omega}+b(3)e^{-j2\omega}+\ldots}{a(1)+a(2)e^{-j\omega}+a(3)e^{-j2\omega}+\ldots} = \frac{1}{1-0.8e^{-j\omega}}$$

Here the vectors b and a specify, respectively, the numerator and denominator coefficients. In our example $b(1)$ = 1, $a(1)$ = 1, and $a(2)$ = –0.8. The MATLAB segment and the corresponding plots follow. Note that the plot goes from $-\pi$ to π, not -3π to 3π.

```
b = [1]; %Numerator coefficient
a = [1, -0.8]; %Denominator coefficients
w = -pi: pi/256: pi; %A total of 512 points
[Xw] = freqz(b, a, w);
subplot(2, 1, 1), plot(w, abs(Xw));
xlabel('Frequency \omega'), ylabel('Magnitude of X(\omega)'); grid
subplot(2, 1, 2), plot(w, angle(Xw));
xlabel('Frequency \omega'), ylabel('Phase of X(\omega)'); grid
```



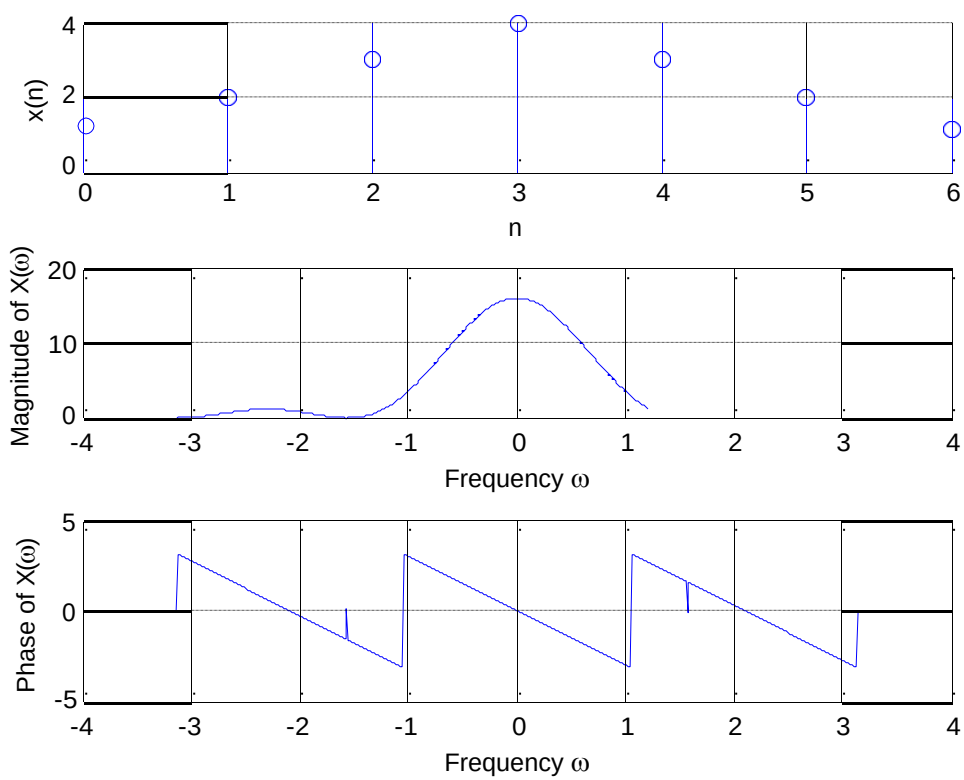
Example 1.6.4 Find the DTFT, $X(\omega)$, for $x(n) = \{1, 2, 3, 4, 3, 2, 1\}$.
Solution The DTFT is

$$X(e^{j\omega}) = \sum_{n=0}^{6} x(n)\, e^{-j\omega n} = 1 + 2e^{-j\omega 1} + 3e^{-j\omega 2} + 4e^{-j\omega 3} + 3e^{-j\omega 4} + 2e^{-j\omega 5} + 1e^{-j\omega 6}$$

$$= (2\cos 3\omega + 4\cos 2\omega + 6\cos\omega + 4)\, e^{-j3\omega}$$

The numerator vector is $b = [1, 2, 3, 4, 3, 2, 1]$ and the denominator vector is $a = \{1\}$. The MATLAB segment plots the sequence and the frequency response.

```
%Sketch of sequence
n = 0:1:6; xn = [1, 2, 3, 4, 3, 2, 1];
subplot (3, 1, 1), stem(n, xn)
xlabel('n'), ylabel('x(n)'); grid
%Frequency response
b = [1, 2, 3, 4, 3, 2, 1]; %Numerator coefficients
a = [1]; %Denominator coefficients
w = -pi: pi/256: pi; %A total of 512 points
[Xw] = freqz(b, a, w);
subplot(3, 1, 2), plot(w, abs(Xw));
xlabel('Frequency \omega'), ylabel('Magnitude of X(\omega)'); grid
subplot(3, 1, 3), plot(w, angle(Xw));
xlabel('Frequency \omega'), ylabel('Phase of X(\omega)'); grid
```



Example 1.6.5 Find the DTFT, $X(\omega)$, for
(a) $x(n) = \{1, 2, 3, 4, 3, 2, 1, 0\}$
(b) $x(n) = \{1, 2, 3, 4, 3, 2, 1, 0, \ldots, 0\}$
Solution !?!

The following examples are repeated in Unit II – DFS &DFT.

Example 1.6.6 Obtain the 7-point DFT of the sequence *x(n)* = {1, 2, 3, 4, 3, 2, 1} by taking 7 samples of its DTFT uniformly spaced over the interval $0 \le \omega \le 2\pi$.

Solution The sampling interval in the frequency domain is $2\pi/7$. From Example 4 we have

$$X(e^{j\omega}) \text{ or } X(\omega) = 1 + 2e^{-j\omega 1} + 3e^{-j\omega 2} + 4e^{-j\omega 3} + 3e^{-j\omega 4} + 2e^{-j\omega 5} + 1e^{-j\omega 6}$$

$$= (2\cos 3\omega + 4\cos 2\omega + 6\cos\omega + 4)\, e^{-j3\omega}$$

The DFT, $X(k)$, is given by replacing ω with $k(2\pi/7)$ where k is an index ranging from 0 to 6:

$$\text{DFT} = X(\omega)\Big|_{\omega = 2\pi k/7} = X(2\pi k/7), \qquad k = 0 \text{ to } 6$$

This is denoted XkfromDTFT in the MATLAB segment below.

MATLAB:

```
w = 0: 2*pi/7: 2*pi-0.001
XkfromDTFT = (4+6*cos(w)+4*cos(2*w)+2*cos(3*w)) .* exp(-j*3*w)
```

MATLAB solution:

```
XkfromDTFT = [16, (-4.5489 - 2.1906i), (0.1920 + 0.2408i), (-0.1431 - 0.6270i),
        (-0.1431 + 0.6270i), (0.1920 - 0.2408i), (-4.5489 + 2.1906i)]
```

This is the 7-point DFT obtained by sampling the DTFT at 7 points uniformly spaced in $(0, 2\pi)$. It should be the same as the DFT directly obtained, for instance, by using the *fft* function in MATLAB:

MATLAB:

```
xn = [1 2 3 4 3 2 1]
Xkusingfft = fft(xn)
```

MATLAB solution:

```
Xkusingfft = [16, (-4.5489 - 2.1906i), (0.1920 + 0.2408i), (-0.1431 - 0.6270i),
        (-0.1431 + 0.6270i), (0.1920 - 0.2408i), (-4.5489 + 2.1906i)]
```

It can be seen that "↓kfromDTFT" = "↓kusingfft".

Example 1.6.7 Obtain the 7-point inverse DTFT *x(n)* by finding the 7-point inverse DFT of *X(k)*:

$$X(2\pi k/7) = [16, (-4.5489 - 2.1906i), (0.1920 + 0.2408i), (-0.1431 - 0.6270i),$$
$$(-0.1431 + 0.6270i), (0.1920 - 0.2408i), (-4.5489 + 2.1906i)]$$

MATLAB:

```
Xk = [16, (-4.5489 - 2.1906i), (0.1920 + 0.2408i), (-0.1431 - 0.6270i),(-0.1431 +
        0.6270i), (0.1920 - 0.2408i), (-4.5489 + 2.1906i)]
xn = ifft(Xk)
```

MATLAB solution:

```
xn = [1.0000    2.0000    3.0000    4.0000    3.0000    2.0000    1.0000]
```

This is the original sequence we started with in Example 4.

Example 1.6.8 What will be the resulting time sequence if the DTFT of the *7-point* sequence is sampled at 6 (or fewer) uniformly spaced points in (0, 2π) and its inverse DFT is obtained?
Solution The sampling interval in the frequency domain now is $2\pi/6$. From Example 4 we have

$$X(e^{j\omega}) \text{ or } X(\omega) = 1 + 2e^{-j\omega 1} + 3e^{-j\omega 2} + 4e^{-j\omega 3} + 3e^{-j\omega 4} + 2e^{-j\omega 5} + 1e^{-j\omega 6}$$
$$= (2\cos 3\omega + 4\cos 2\omega + 6\cos\omega + 4)\, e^{-j3\omega}$$

The DFT then is given by

$$\text{DFT} = X(\omega)\big|_{\omega = 2\pi k/6} = X(2\pi k/6), \qquad k = 0 \text{ to } 5$$

This is denoted Xk6point in the MATLAB segment below.

MATLAB:

```
w = 0: 2*pi/6: 2*pi-0.001
Xk6point = (4+6*cos(w)+4*cos(2*w)+2*cos(3*w)) .* exp(-j*3*w)
```

MATLAB solution:

```
Xk6point = [16, (-3.0000 - 0.0000i), (1.0000 + 0.0000i), 0, (1.0000 + 0.0000i),
        (-3.0000 - 0.0000i)]
```

This is the 6-point DFT obtained by sampling the DTFT at 6 points uniformly spaced in (0, 2π).

Example 1.6.9 Obtain the 6-point inverse DTFT *x(n)* by finding the 6-point inverse DFT of Xk6point:

$$X(2\pi k/6) = [16, (-3.0000 - 0.0000i), (1.0000 + 0.0000i), 0, (1.0000 + 0.0000i), (-3.0000 - 0.0000i)]$$

MATLAB:

```
Xk6point = [16, (-3.0000 - 0.0000i), (1.0000 + 0.0000i), 0, (1.0000 + 0.0000i), (-
        3.0000 - 0.0000i)]
xn = ifft(Xk6point)
```

MATLAB solution:

```
xn = [2    2    3    4    3    2]
```

Comparing with the original 7-point sequence, xn = [1 2 3 4 3 2 1], we see the consequence of *under-sampling* the continuous-ω function $X(\omega)$: the corresponding time domain sequence *x(n)* is said to suffer *time-domain aliasing*. This is similar to the situation that occurs when a continuous-time function *x(t)* is under-sampled: the corresponding frequency domain function $X_s(\omega)$ contains *frequency-domainaliasing*.

Example 1.6.10 What will be the resulting time sequence if the DTFT of the *7-point* sequence is sampled at 8 (or more) uniformly spaced points in (0, 2π) and its inverse DFT is obtained?
Solution The sampling interval in the frequency domain now is $2\pi/8$. From Example 4 we have

$$X(e^{j\omega}) \text{ or } X(\omega) = 1 + 2e^{-j\omega 1} + 3e^{-j\omega 2} + 4e^{-j\omega 3} + 3e^{-j\omega 4} + 2e^{-j\omega 5} + 1e^{-j\omega 6}$$
$$= (2\cos 3\omega + 4\cos 2\omega + 6\cos\omega + 4)\, e^{-j3\omega}$$

The DFT then is given by

$$\text{DFT} = X(\omega)\big|_{\omega = 2\pi k/8} = X(2\pi k/8), \qquad k = 0 \text{ to } 7$$

This is denoted Xk8point in the MATLAB segment below.

MATLAB:

```
w = 0: 2*pi/8: 2*pi-0.001
Xk8point = (4+6*cos(w)+4*cos(2*w)+2*cos(3*w)) .* exp(-j*3*w)
```

MATLAB solution:

```
Xk8point = [16, (-4.8284 - 4.8284i), (0.0000 - 0.0000i), (0.8284 - 0.8284i), 0,
        (0.8284 + 0.8284i), (0.0000 - 0.0000i), (-4.8284 + 4.8284i)]
```

This is the 8-point DFT obtained by sampling the DTFT at 8 points uniformly spaced in (0, 2π).

Example 1.6.11 Obtain the 8-point inverse DTFT *x(n)* by finding the 8-point inverse DFT of Xk8point:

$$X(2\pi k/8) = [16, (-4.8284 - 4.8284i), (0.0000 - 0.0000i), (0.8284 - 0.8284i), 0,$$
$$(0.8284 + 0.8284i), (0.0000 - 0.0000i), (-4.8284 + 4.8284i)]$$

MATLAB:

```
Xk8point = [16, (-4.8284 - 4.8284i), (0.0000 - 0.0000i), (0.8284 - 0.8284i), 0,
        (0.8284 + 0.8284i), (0.0000 - 0.0000i), (-4.8284 + 4.8284i)]
xn = ifft(Xk8point)
```

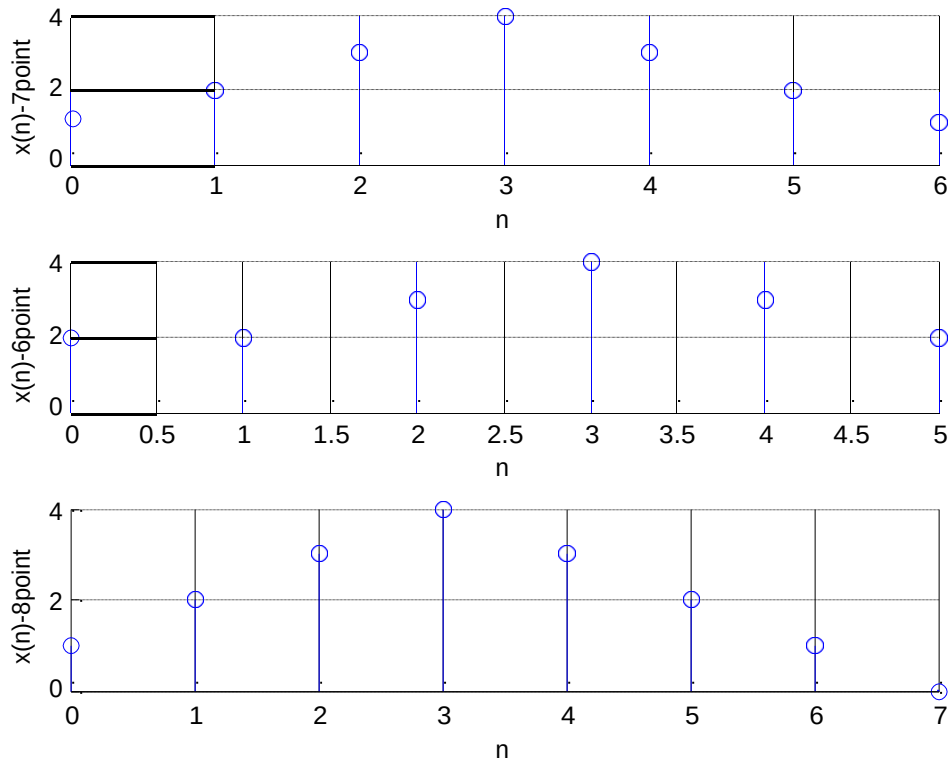
MATLAB solution:

```
xn = [1    2    3    4    3    2    1    0]
```

We see that the original 7-point sequence has been preserved with an *appended zero*. The original sequence and the *zero-padded sequence* (with any number of zeros) have the same DTFT. This is a case of *over-sampling* the continuous-ω function $X(\omega)$: there is no *time-domain aliasing*. This is similar to the situation that occurs when a continuous-time function *x(t)* is over-sampled: the corresponding frequency domain function $X_s(\omega)$ is free from *frequency-domain aliasing*.

```
%Sketch of sequences
n = 0:1:6; xn = [1, 2, 3, 4, 3, 2, 1];
```

```
subplot (3, 1, 1), stem(n, xn)
xlabel('n'), ylabel('x(n)-7point'); grid
%
n = 0:1:5; xn = [2     2     3     4     3     2];
subplot (3, 1, 2), stem(n, xn)
xlabel('n'), ylabel('x(n)-6point'); grid
%
n = 0:1:7; xn = [1, 2, 3, 4, 3, 2, 1, 0];
subplot (3, 1, 3), stem(n, xn)
xlabel('n'), ylabel('x(n)-8point'); grid
```



Frequency response of discrete-time system

For a linear shift-invariant system with impulse response *h(n)*, the Fourier transform *H(ω)* gives the *frequency response*. Consider the input sequence $x(n) = e^{j\omega n}$ for $-\infty < n < \infty$, i.e., a complex exponential of radian frequency ω and magnitude 1, applied to a linear shift-invariant system whose unit sample response is *h(n)*. Using convolution we obtain the output *y(n)* as

$$y(n) = h(n) * x(n) = \sum_{k=-\infty}^{\infty} h(k)\, x(n-k) = \sum_{k=-\infty}^{\infty} h(k)\, e^{j\omega(n-k)} = e^{j\omega n} \underbrace{\sum_{k=-\infty}^{\infty} h(k)\, e^{-j\omega k}}_{H(\omega)}$$

$$= H(\omega)\, e^{j\omega n}$$

Thus we see that *H(ω)* describes the change in complex amplitude of a complex exponential as a function of frequency. The quantity *H(ω)* is called the **frequency response** of the system. In

general, $H(\omega)$ is complex valued and may be expressed either in the Cartesian form or the polar from as

$$H(\omega) = H_R(\omega) + j\, H_I(\omega) \qquad \text{or} \qquad H(\omega) = \hat{H}(\omega)\, e^{j\angle H(\omega)}$$

where H_R and H_I are the real part and imaginary part respectively. $\hat{H}(\omega)$ is loosely called the magnitude and $\angle H(\omega)$ is loosely called the phase. Strictly speaking, $\hat{H}(\omega)$ is called the **zero-phase frequency response**; note that $\hat{H}(\omega)$ is *real valued* but may be positive or negative. We may use the symbol $|H(\omega)|$ for the *magnitude* which is strictly non-negative. If $\hat{H}(\omega)$ is positive then

$$\text{Magnitude} = |H(\omega)| = \hat{H}(\omega) \qquad \& \qquad \text{Phase} = \angle H(\omega)$$

If $\hat{H}(\omega)$ is negative then

$$\text{Magnitude} = |H(\omega)| = |\hat{H}(\omega)| = -\hat{H}(\omega) \qquad \& \qquad \text{Phase} = \angle H(\omega) \pm \pi$$

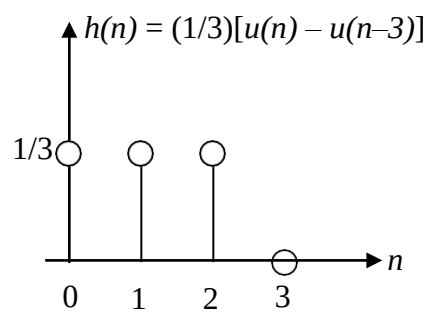
We shall often loosely use the symbol $|H(\omega)|$ to refer to $\hat{H}(\omega)$ as well with the understanding that when the latter is negative we shall take its absolute value (the magnitude) and accordingly adjust $\angle H(\omega)$ by $\pm\pi$.

Example 1.7.1 [Moving average filter] The impulse response of the LTI system

$$y(n) = \frac{x(n) + x(n-1) + x(n-2)}{3}$$

is

$$h(n) = \begin{cases} 1/3, & n = 0, 1, 2 \\ 0, & \text{otherwise} \end{cases}$$



The frequency response is obtained below.

$$H(\omega) = \sum_{k=-\infty}^{\infty} h(k)\, e^{-j\omega k} = \sum_{k=0}^{2} (1/3)\, e^{-j\omega k} = \frac{1}{3}\left(e^{-j\omega 0} + e^{-j\omega 1} + e^{-j\omega 2}\right)$$

$$= \frac{e^{-j\omega}}{3}\left(e^{j\omega} + 1 + e^{-j\omega}\right) = \frac{e^{-j\omega}}{3}\left(1 + 2\left(e^{j\omega} + e^{-j\omega}\right)\right) = \frac{(1 + 2\cos\omega)}{3} e^{-j\omega}$$

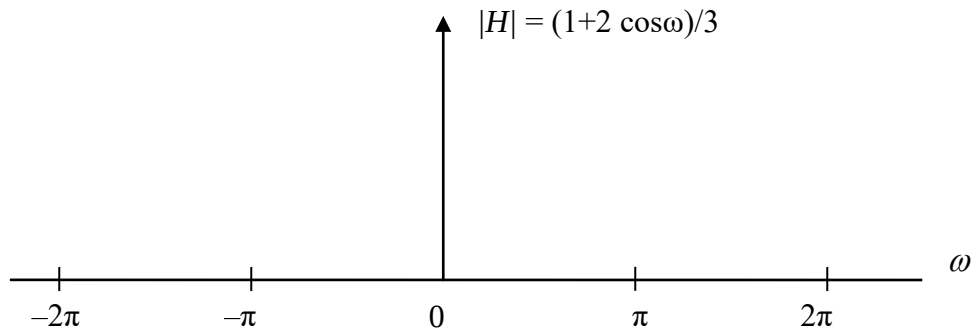
which is already in the polar form $H(\omega) = |H(\omega)|\, e^{j\angle H(\omega)}$, so that

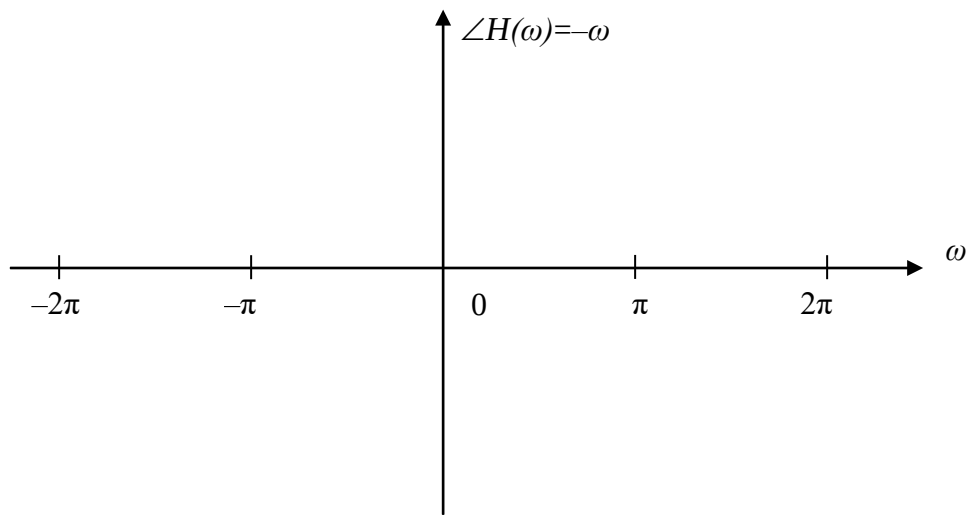
$$|H(\omega)| = (1 + 2\cos\omega)/3 \qquad \text{and} \qquad \angle H(\omega) = -\omega$$

The zero crossings of the magnitude plot occur where $|H(\omega)| = (1 + 2\cos\omega)/3 = 0$, or $\omega = \cos^{-1}(-1/2) = 2\pi/3 = 120^0$. A frequency of $\omega = 2\pi/3$ rad./sample ($f = 1/3$ cycle/sample) is totally stopped (filtered out) by the filter. The corresponding digital signal is $x_5(n) = \cos 2\pi(1/3)n$. The underlying continuous-time signal, $x_5(t)$, depends on the sampling frequency. If,

for example, the sampling frequency is 16Hz, then $x_5(t) = \cos 2\pi(16/3)t$, and a frequency of 16/3 Hz will be totally filtered out. If the sampling frequency is 150Hz, then $x_5(t) = \cos 2\pi(150/3)t$, and a frequency of 50 Hz will be eliminated.

In calibrating the horizontal axis in terms of the cyclic frequency, F, we use the relation $\omega = \Omega T = 2\pi FT = 2\pi F/F_s$ from which the point $\omega = 2\pi$ corresponds to $F = F_s$.





(Aside) The system $y(n) = \frac{x(n)}{3} + \frac{x(n-1)}{3} + \frac{x(n-2)}{3}$ is a crude low pass filter, but the attenuation does not increase monotonically with frequency. In fact, the highest possible frequency, $F_s/2$ Hz, (or π rad/sample) is not well attenuated at all. The following is a slight variation of the three-term moving average:

$$y(n) = \frac{x(n)}{4} + \frac{x(n-1)}{2} + \frac{x(n-2)}{4}$$

Its magnitude response is a "raised cosine" (with no zero crossing, monotonically decreasing but wider than the 3-term).

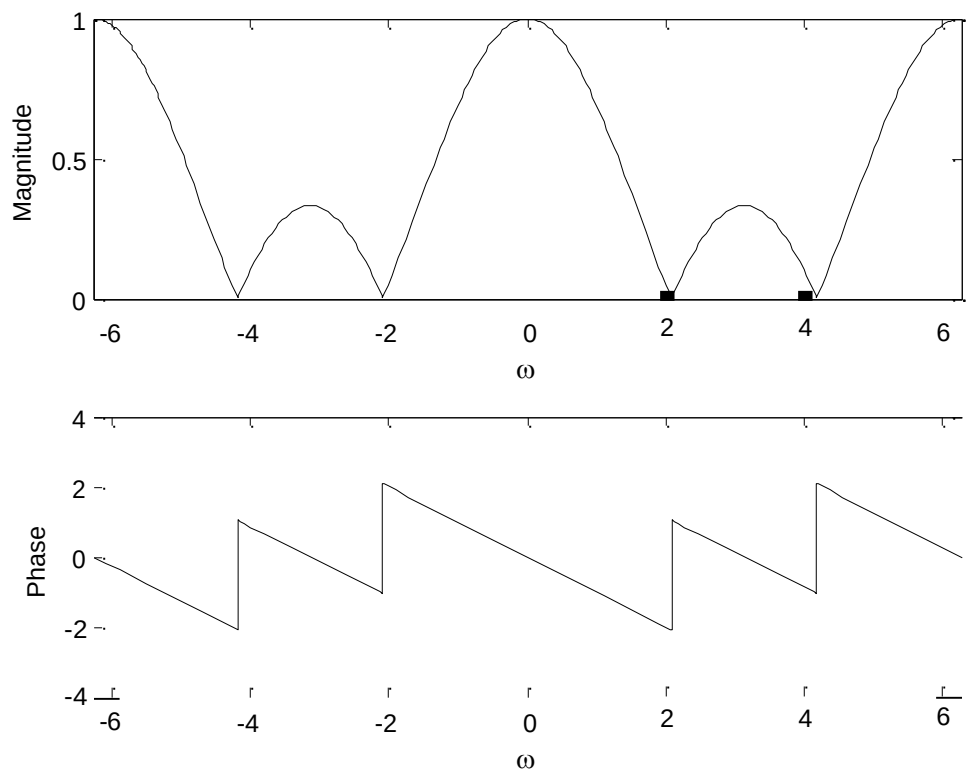
(End of Aside)

Example 1.7.2 [MATLAB *fplot*] [Moving average filter]

$$H(\omega) = \frac{1}{3}\left(1 + e^{-j\omega 1} + e^{-j\omega 2}\right).$$

The program follows. Note that in MATLAB we use „w" for ω and the plot ranges over $(-2\pi, 2\pi)$. Both ω and the phase, $\angle X(\omega)$, are in radians.

```
subplot(2,1,1);fplot('abs((1/3)*(1+exp(-j*w)+exp(-j*2*w)))', [-2*pi,2*pi], 'k');
xlabel('\omega');ylabel('Magnitude');
subplot(2,1,2);fplot('angle((1/3)*(1+exp(-j*w)+exp(-j*2*w)))', [-2*pi,2*pi], 'k');
xlabel('\omega');ylabel('Phase');
```

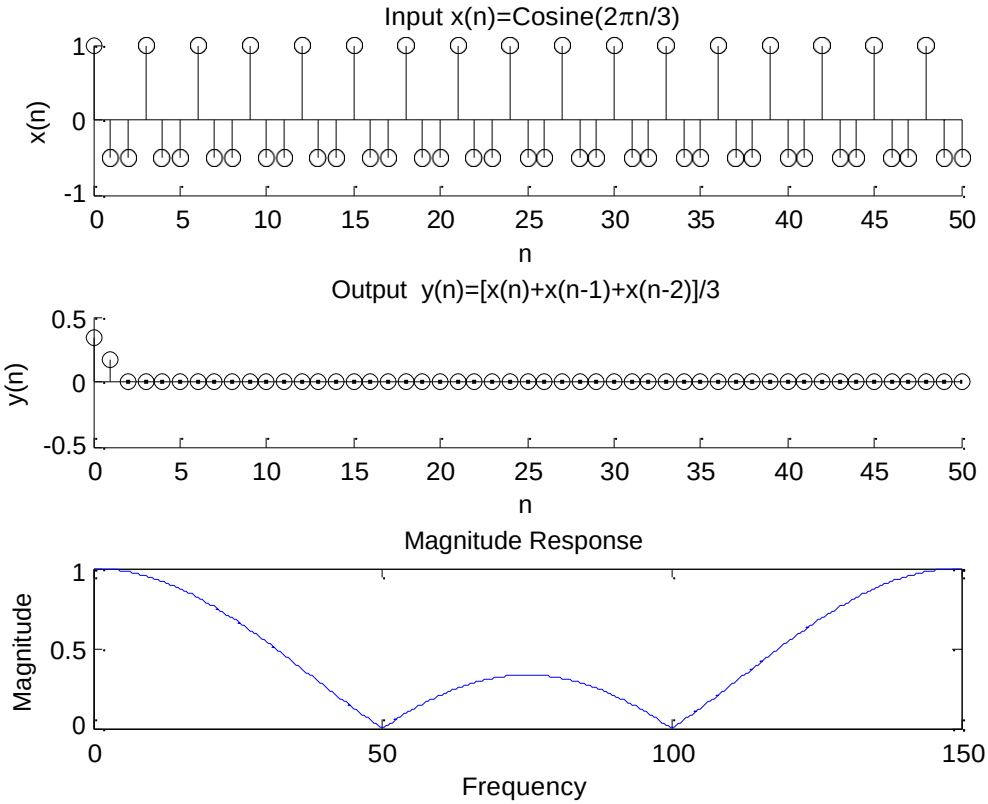


[**Homework]** Plot the output signal $y(n) = \dfrac{x(n) + x(n-1) + x(n-2)}{3}$ for several values of $n \geq 0$ where the input is $x(n) = \cos 2\pi(1/3)n$. Take $x(-1) = x(-2) = 0$.

Example 1.7.3 [MATLAB *filter, freqz*] [Moving average filter] Consider a signal $x(t) = \cos(2\pi 50t)$ sampled at 150 Hz. The corresponding $x(n) = \cos(2\pi n/3)$ is the input to the moving average filter. The following MATLAB segment plots $x(n)$ vs. n and $y(n)$ vs. n.

```
n=0:1:50;
x=cos(2*pi*n/3);
b=[1/3, 1/3, 1/3]; a=[1]; %Filter coefficients
y=filter(b, a, x);
subplot(3, 1, 1), stem(n, x, 'ko');title('Input x(n)=Cosine(2\pin/3)');
```

```
xlabel('n'), ylabel('x(n)')
subplot(3, 1, 2), stem(n, y, 'ko');title('Output y(n)=[x(n)+x(n-1)+x(n-2)]/3');
xlabel('n'), ylabel('y(n)');
w=0: pi/256: 2*pi; h=freqz(b, a, w);
subplot(3, 1, 3), plot(w*75/pi, abs(h));title('Magnitude Response');
xlabel('Frequency'), ylabel('Magnitude');
```



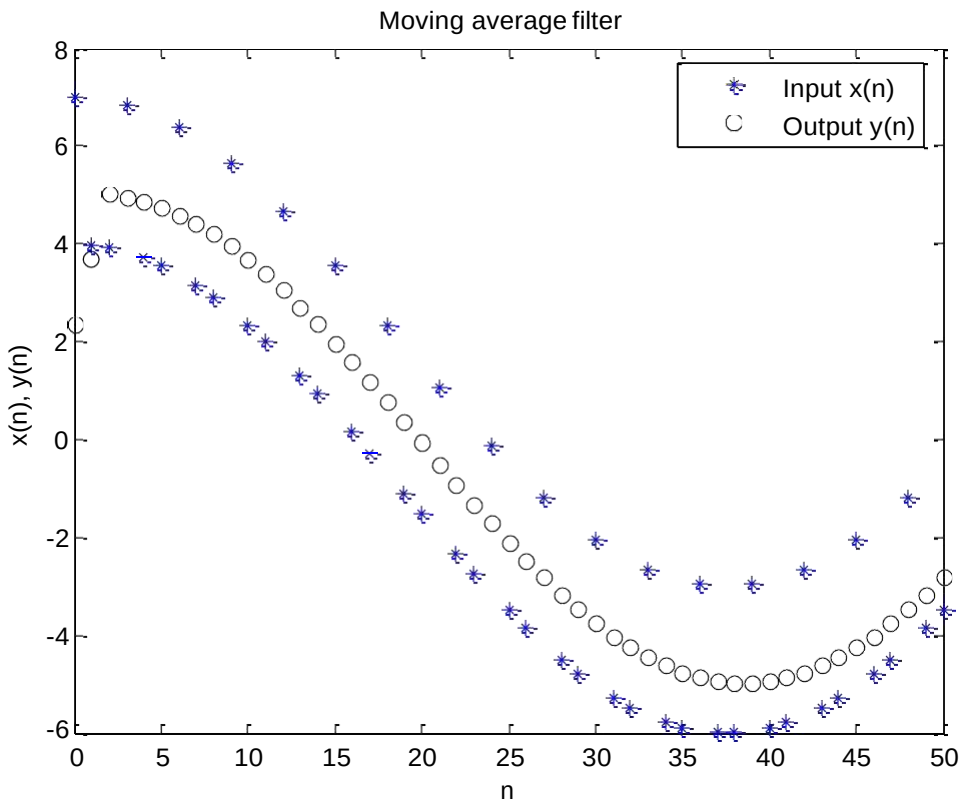
Example 1.7.4 [MATLAB *filter*] [Moving average filter] As an extension of above example, consider the signal consisting of a 2 Hz desirable component plus a noise component of 50 Hz (with a smaller amplitude), sampled at 150 Hz.

$x(t) = 5 \cos(2\pi 2t) + 2 \cos(2\pi 50t)$

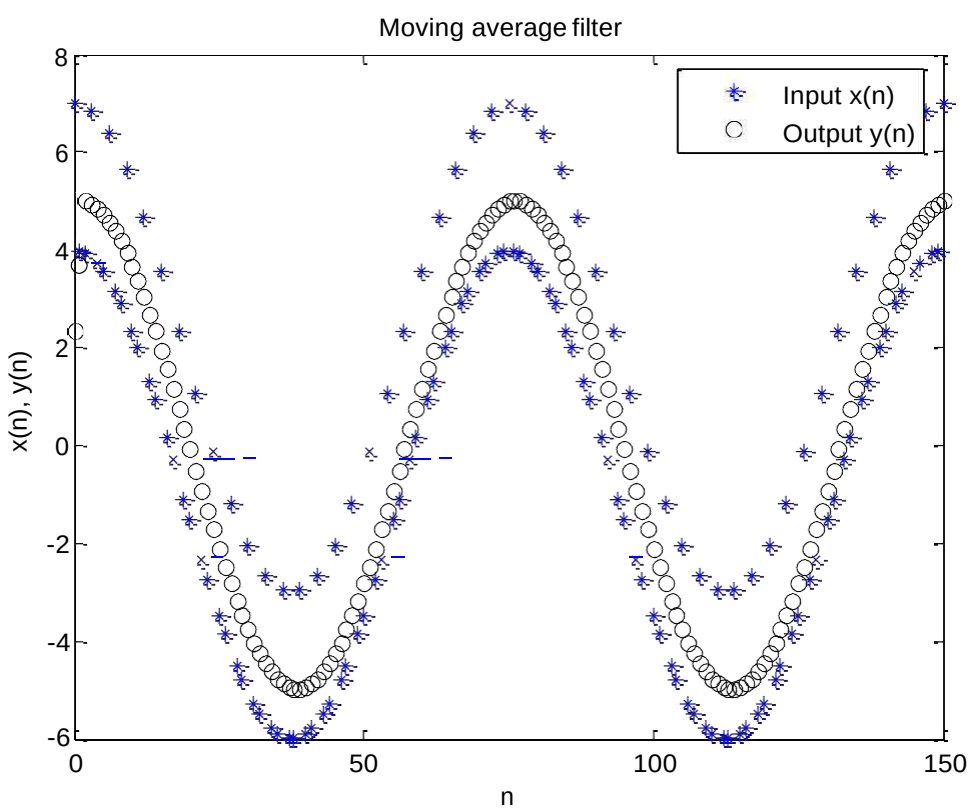
$x(n) = 5 \cos(2\pi n/75) + 2 \cos(2\pi n/3)$

In the following MATLAB segment we show both sequences, *x(n)* and *y(n)*, on the same (multi)plot so that they have the same scale. The smoothing action of the filter is easily discernible in the multiplot.

```
n = 0: 1: 50;
x = 5*cos(2*pi*n/75) + 2*cos(2*pi*n/3);
b = [1/3, 1/3, 1/3]; a = [1]; %Filter coefficients
y = filter(b, a, x);
plot(n, x, 'b*', n, y, 'ko');
legend ('Input x(n)', 'Output y(n)');
title ('Moving average filter');
xlabel ('n'), ylabel('x(n), y(n)');
```



The same plot is shown below over a longer period.

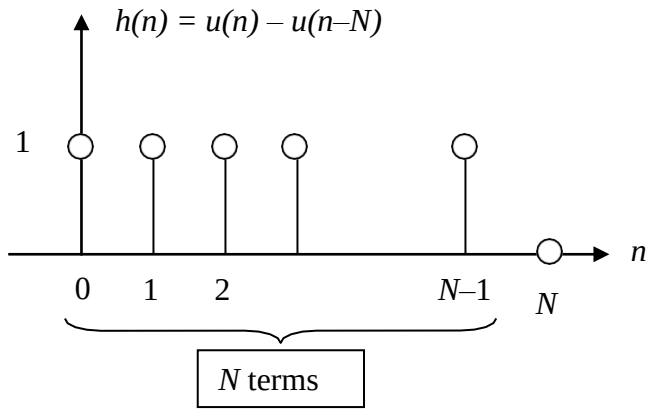


[**Homework]** Examples 5.1.1, 5.1.2, 5.1.3 and 5.1.4, Proakis, 4th Ed.

Sketches of ideal digital low pass and high pass filters *Ref. p. 23, O&S for LP filter.*

Example 1.7.5 [2003] [O & S, p. 20, and L.C. Ludeman, p. 51] A linear time-invariant system has unit sample response $h(n) = u(n) - u(n-N)$. Find the amplitude and phase spectra.

Note that this would be an N-term moving average filter if $h(n) = (1/N)[\, u(n) - u(n-N)]$.



$$h(n) = \begin{cases} 1, & 0 \le n \le N-1 \\ 0, & \text{elsewhere} \end{cases}$$

$$H(\omega) = \sum_{k=-\infty}^{\infty} h(k) e^{-j\omega k} = \sum_{k=0}^{N-1} 1 e^{-j\omega k} = \frac{1 - e^{-j\omega N}}{1 - e^{-j\omega}}$$

$$= \frac{e^{-j\omega N/2}(e^{j\omega N/2} - e^{-j\omega N/2})}{e^{-j\omega/2}(e^{j\omega/2} - e^{-j\omega/2})} = \frac{\sin(\omega N/2)}{\sin(\omega/2)} e^{-j\omega(N-1)/2}$$

$$|H(\omega)| = \frac{\sin(\omega N/2)}{\sin(\omega/2)} \quad \text{and} \quad \angle H(e^{j\omega}) = -\omega(N-1)/2$$

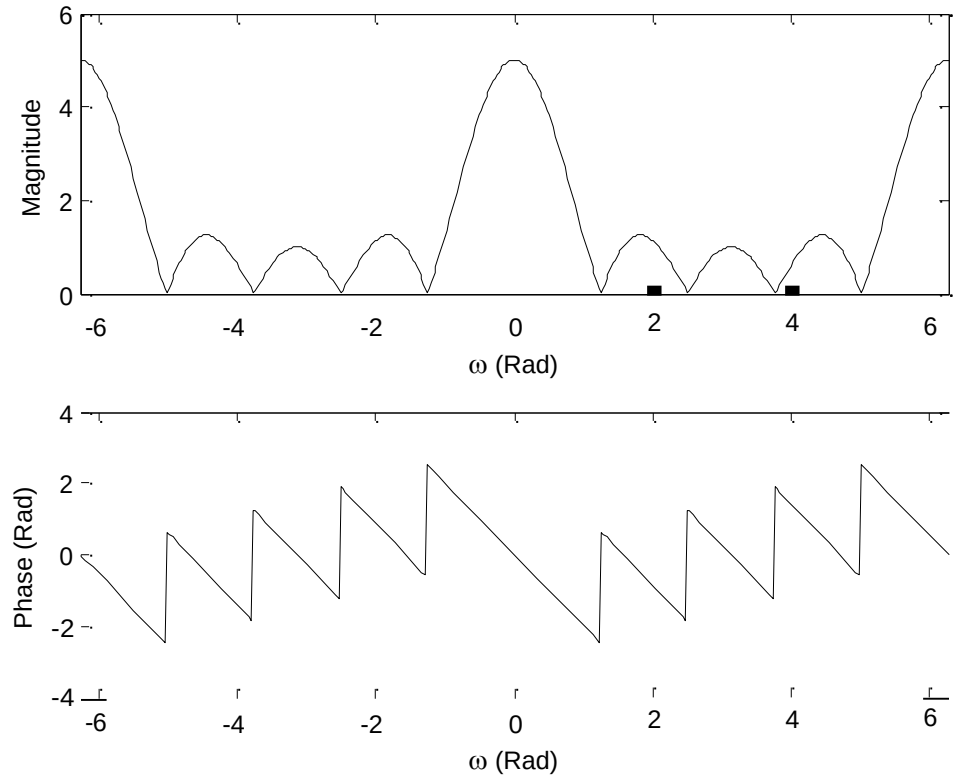
See the plots in Oppenheim and Schafer.

Example 1.7.6 [MATLAB *fplot*] [5-term moving average filter] The program for the case where $N = 5$ follows.

$$H(\omega) = \left(1 + e^{-j\omega 1} + e^{-j\omega 2} + e^{-j\omega 3} + e^{-j\omega 4}\right) = \frac{1 - e^{-j\omega 5}}{1 - e^{-j\omega}}$$

Note that in MATLAB we use „w" for ω and the plot ranges from -2π to 2π. Both ω and the phase, $\angle X(\omega)$, are in radians.

```
subplot(2,1,1);fplot('abs((1-exp(-j*w*5))/(1-exp(-j*w)))', [-2*pi,2*pi], 'k');
xlabel('\omega (Rad)');ylabel('Magnitude');
subplot(2,1,2);fplot('angle((1-exp(-j*w*5))/(1-exp(-j*w)))', [-2*pi,2*pi], 'k');
xlabel('\omega (Rad)');ylabel('Phase (Rad)');
```



Properties of the discrete-time Fourier transform (DTFT)

For the DTFT Oppenheim & Schafer use the symbol $X(e^{j\omega})$ while Proakis uses $X(\omega)$.

(1) Periodicity $X(\omega)$ is periodic with period 2π, that is, $X(\omega+2\pi) = X(\omega)$ for all ω. Since $e^{j\omega n}$ is periodic in ω with period 2π, it follows that $X(\omega)$ is also periodic with the same period. Replacing ω with $(\omega+2\pi)$ gives

$$X(\omega+2\pi) = \sum_{n=-\infty}^{\infty} x(n)\, e^{-j(\omega+2\pi)n} = \sum_{n=-\infty}^{\infty} x(n)\, e^{-j\omega n} e^{-j2\pi n} = \sum_{n=-\infty}^{\infty} x(n)\, e^{-j\omega n}\, 1$$

$$= X(\omega) \text{ for all } \omega$$

As a result, while in the continuous time case Ω ranges from $-\infty$ to ∞, in the discrete-time case we need only consider values of ω over the range 0 to 2π (or, $-\pi$ to π, or any 2π-long interval).

(2) Linearity The discrete Fourier-transform is a linear operation. If $\mathcal{F}\{x_1(n)\} = X_1(\omega)$ and $\mathcal{F}\{x_2(n)\} = X_2(\omega)$, then $\mathcal{F}\{a_1\, x_1(n) + a_2\, x_2(n)\} = a_1\, X_1(\omega) + a_2\, X_2(\omega)$ for any constants a_1 and a_2.

(3) Time shifting Time shift results in phase shift. If $\mathcal{F}\{x(n)\} = X(\omega)$, then $\mathcal{F}\{x(n-k)\} = e^{-j\omega k} X(\omega)$.

Proof We have

$$\mathcal{F}\{x(n-k)\} = \sum_{n=-\infty}^{\infty} x(n-k)\, e^{-j\omega n}$$

On the right hand side set $n-k = m$, so that $n = m+k$ and the limits $n = -\infty$ to ∞ change to $m = -\infty$ to $+\infty$. Then

$$\mathcal{F}\{x(n-k)\} = \sum_{m=-\infty}^{\infty} x(m)\, e^{-j\omega(m+k)} = e^{-j\omega k} \sum_{m=-\infty}^{\infty} x(m)\, e^{-j\omega m} = e^{-j\omega k} X(\omega) \quad \text{QED}$$

(4) Frequency shifting Multiplication in the time domain by a complex exponential results in frequency shifting. Given $\mathcal{F}\{x(n)\} = X(\omega)$, then $\mathcal{F}\left\{e^{j\omega_0 n} x(n)\right\} = X(\omega - \omega_0)$.

Proof We have

$$\mathcal{F}\left\{e^{j\omega_0 n} x(n)\right\} = \sum_{n=-\infty}^{\infty} e^{j\omega_0 n} x(n) e^{-j\omega n} = \sum_{n=-\infty}^{\infty} x(n)\, e^{-j(\omega-\omega_0)n} = X(\omega-\omega_0) \qquad \text{QED}$$

Alternatively, using the synthesis equation,

$$\mathcal{F}^{-1}\{X(\omega-\omega_0)\} = \frac{1}{2} \int_{2\pi} X(\omega - \omega_0)\, e^{j\omega n} d\omega$$

Set $\omega-\omega_0 = \lambda$ so that $\omega = \lambda+\omega_0$ and the limits $\omega = 0$ to 2π change to $\lambda = -\omega_0$ to $(-\omega_0+2\pi)$, which amounts to any interval of length 2π. Also $d\omega = d\lambda$. Then

$$\mathcal{F}^{-1}\{X(\omega-\omega_0)\} = \frac{1}{2} \int_{2\pi} X(\lambda)\, e^{j(\lambda+\omega_0)n} d\lambda$$

$$= e^{j\omega_0 n} \frac{1}{2\pi} \int_{2\pi} X(\lambda)\, e^{j\lambda n} d\lambda = e^{j\omega_0 n} x(n) \qquad \text{QED}$$

(5) Time reversal corresponds to frequency reversal. Given $\mathcal{F}\{x(n)\} = X(\omega)$, then $\mathcal{F}\{x(-n)\} = X(-\omega)$.

Proof We have

$$\mathcal{F}\{x(-n)\} = \sum_{n=-\infty}^{\infty} x(-n)\, e^{-j\omega n}$$

On the right hand side set $m = -n$ so that the limits $n = -\infty$ to ∞ change to $m = \infty$ to $-\infty$, and

$$\mathcal{F}\{x(-n)\} = \sum_{m=\infty}^{-\infty} x(m)\, e^{j\omega m}$$

Since this is a summation the limits can be written in reverse order, and we have

$$\mathcal{F}\{x(-n)\} = \sum_{m=-\infty}^{\infty} x(m)\, e^{-j(-\omega)m} = X(-\omega) = X(e^{-j\omega}) \qquad \text{QED}$$

***(6)* Differentiation in frequency** $\mathcal{F}\{n\, x(n)\} = j \dfrac{dX(e^{j\omega})}{d\omega}$. Since

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n) e^{-j\omega n}$$

we differentiate both sides with respect to ω to get

$$\frac{dX(e^{j\omega})}{d\omega} = \sum_{n=-\infty}^{\infty} x(n) \frac{de^{-j\omega n}}{d\omega} = \sum_{n=-\infty}^{\infty} x(n)(-jn)\, e^{-j\omega n}$$

or

$$j\frac{dX(e^{j\omega})}{d\omega} = \sum_{n=-\infty}^{\infty} n\, x(n)\, e^{-j\omega n} = \mathcal{F}\{n\, x(n)\} \qquad \text{QED}$$

(7) Convolution If $y(n)$ represents the convolution of two discrete-time signals $x(n)$ and $h(n)$, that is, $y(n) = x(n)*h(n)$, then

$$Y(e^{j\omega}) = \mathcal{F}\{x(n)*h(n)\} = X(e^{j\omega}) \,.\, H(e^{j\omega})$$

From the definition of the Fourier transform

$$Y(e^{j\omega}) = \sum_{n=-\infty}^{\infty} y(n)\, e^{-j\omega n} = \sum_{n=-\infty}^{\infty} \{x(n)*\, h(n)\} e^{-j\omega n}$$

$$= \sum_{n=-\infty}^{\infty} \left\{ \sum_{k=-\infty}^{\infty} h(k) x(n-k) \right\} e^{-j\omega n}$$

Interchanging the order of summation

$$Y(e^{j\omega}) = \sum_{k=-\infty}^{\infty} h(k) \left\{ \sum_{n=-\infty}^{\infty} x(n-k)\, e^{-j\omega n} \right\}$$

The inner sum (I.S.) is handled thus: Let $(n-k) = \lambda$. Then as n goes from $-\infty$ to ∞, λ goes from $-\infty$ to ∞ as well. Further $n = \lambda + k$. Thus the inner sum becomes

$$\text{I.S.} = \sum_{n=-\infty}^{\infty} x(n-k)\, e^{-j\omega n} = \sum_{\lambda=-\infty}^{\infty} x(\lambda)\, e^{-j\omega(\lambda+k)} = \left(\sum_{\lambda=-\infty}^{\infty} x(\lambda)\, e^{-j\omega\lambda} \right) e^{-j\omega k} = e^{-j\omega k} X(e^{j\omega})$$

Thus we have

$$Y(e^{j\omega}) = \sum_{k=-\infty}^{\infty} h(k) \left\{ e^{-j\omega k} X\left(e^{j\omega}\right) \right\} = X(e^{j\omega}) \sum_{k=-\infty}^{\infty} h(k) e^{-j\omega k} = X(e^{j\omega}) \,.\, H(e^{j\omega})$$

The function $H(e^{j\omega})$ is referred to as the **frequency response** of the system.

(8) **Multiplication of two sequences** Let $y(n)$ be the product of two sequences $x_1(n)$ and $x_2(n)$ with transforms $X_1(e^{j\omega})$ and $X_2(e^{j\omega})$, respectively. Then

$$Y(e^{j\omega}) = \mathcal{F}\{x_1(n).\, x_2(n)\} = X_1(e^{j\omega}) * X_2(e^{j\omega})$$

$$= \frac{1}{2\pi} \int_{2\pi} X_1\left(e^{j\theta}\right) X_2\left(e^{j(\omega-\theta)}\right) d\theta$$

This is called **periodic convolution** since both $X_1(e^{j\omega})$ and $X_2(e^{j\omega})$ are periodic functions.

(9) Parseval's Theorem If $x(n)$ and $X(e^{j\omega})$ are a Fourier transform pair, then

$$\sum_{n=-\infty}^{\infty} |x(n)|^2 = \frac{1}{2\pi} \int_{2\pi} \left|X\left(e^{j\omega}\right)\right|^2 d\omega$$

Proof The energy E of the discrete-time sequence is defined as

$$E = \sum_{n=-\infty}^{\infty} |x(n)|^2 = \sum_{n=-\infty}^{\infty} x(n)\, x^*(n)$$

Making the substitution $x^*(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X^*\left(e^{j\omega}\right) e^{-j\omega n}\, d\omega$, we get

$$E = \sum_{n=-\infty}^{\infty} x(n) \left\{ \frac{1}{2\pi} \int_{-\pi}^{\pi} X^*\left(e^{j\omega}\right) e^{-j\omega n} d\omega \right\}$$

Interchanging the order of integration and summation,

$$E = \frac{1}{2\pi}\int_{-\pi}^{\pi} X^*\left(e^{j\omega}\right)\underbrace{\left(\sum_{n=-\infty}^{\infty} x(n)\,e^{-j\omega n}\right)}_{=X\left(e^{j\omega}\right)} d\omega = \frac{1}{2\pi}\int_{-\pi}^{\pi} X^*\left(e^{j\omega}\right)X\left(e^{j\omega}\right)d\omega$$

$$= \frac{1}{2\pi}\int_{-\pi}^{\pi}\left|X\left(e^{j\omega}\right)\right|^2 d\omega \qquad \text{QED}$$

(10) Scaling – expansion in time For continuous-time Fourier transforms we have the property that if $x(t) \leftrightarrow X(\Omega)$, then $x(at) \text{ — } \frac{1}{|a|}X\left(\frac{\Omega}{a}\right)$. This amounts to time-compression if $a > 1$ and to time-expansion if $a < 1$.

The relation between time- and frequency-scaling in discrete time takes on a somewhat different form though. However, there is a result that does closely parallel $x(at) \text{ — } \frac{1}{|a|}X\left(\frac{\Omega}{a}\right)$.

Define $y(n) = x(n/k)$, where k is a positive integer, as

$$y(n) = \begin{cases} x(n/k), & \text{if } n \text{ is a multiple of } k \\ 0, & \text{if } n \text{ is not a multiple of } k \end{cases}$$

This is time expansion. For example, if $k = 3$, then $y(n)$ is obtained from $x(n)$ by placing $k–1 = 2$ zeros between successive values of $x(n)$. That is,

$$y(0) = x(0), \quad y(1) = y(2) = 0, \quad y(3) = x(1), \quad y(4) = y(5) = 0, \ldots \quad \text{Etc.}$$

Since $y(n)$ equals zero unless n is a multiple of k, i.e., unless $n = rk$, where r = all integers from $-\infty$ to $+\infty$, we have the Fourier transform of $y(n)$ as:

$$Y\left(e^{j\omega}\right) = \sum_{n=-\infty}^{\infty} y(n)\,e^{-j\omega n} = \sum_{r=-\infty}^{\infty} y(rk)\,e^{-j\omega rk}$$

Since $y(rk) = x(rk/k) = x(r)$, we have

$$Y\left(e^{j\omega}\right) = \sum_{r=-\infty}^{\infty} x(r)\,e^{-j\omega rk} = \sum_{r=-\infty}^{\infty} x(r)\,e^{-j(k\omega)r} = X\left(e^{jk\omega}\right)$$

Thus we have the relation: Given $x(n) \text{ — } X\left(e^{j\omega}\right)$, then $x(n/k) \text{ — } X\left(e^{jk\omega}\right)$.

Example 1.8.1 [2003] (Frequency response) A discrete-time system is given by

$$y(n) – 5\,y(n–1) = x(n) + 4\,x(n–1)$$

where $x(n)$ is the input and $y(n)$ is the output. Determine its magnitude and phase response as a function of frequency.

Solution The frequency response is obtained by taking the discrete-time Fourier transform of both sides of the difference equation:

$$\mathcal{F}\{y(n) – 5\,y(n–1)\} = \mathcal{F}\{x(n) + 4\,x(n–1)\}$$

With $\mathcal{F}\{y(n)\} = Y\left(e^{j\omega}\right)$ and $\mathcal{F}\{y(n–k)\} = e^{-j\omega k}\,Y\left(e^{j\omega}\right)$ the above equation becomes

$$Y(e^{j\omega}) - 5\,e^{-j\omega}\,Y(e^{j\omega}) = X(e^{j\omega}) + 4\,e^{-j\omega}\,X(e^{j\omega})$$

$$Y(e^{j\omega})\,(1 - 5\,e^{-j\omega}) = X(e^{j\omega})\,(1 + 4\,e^{-j\omega})$$

$$\frac{Y(e^{j\omega})}{X(e^{j\omega})} = H(e^{j\omega}) = \frac{1+4e^{-j\omega}}{1-5e^{-j\omega}} = \frac{\sqrt{(1+4\cos\omega)^2 + (4\sin\omega)^2}\; e^{j\tan^{-1}\left(\frac{-4\sin\omega}{1+4\cos\omega}\right)}}{\sqrt{(1-5\cos\omega)^2 + (5\sin\omega)^2}\; e^{j\tan^{-1}\left(\frac{5\sin\omega}{1-5\cos\omega}\right)}}$$

$$= \sqrt{\frac{(1+4\cos\omega)^2 + (4\sin\omega)^2}{(1-5\cos\omega)^2 + (5\sin\omega)^2}}\; e^{j\left(\tan^{-1}\left(\frac{-4\sin\omega}{1+4\cos\omega}\right) - \tan^{-1}\left(\frac{5\sin\omega}{1-5\cos\omega}\right)\right)}$$

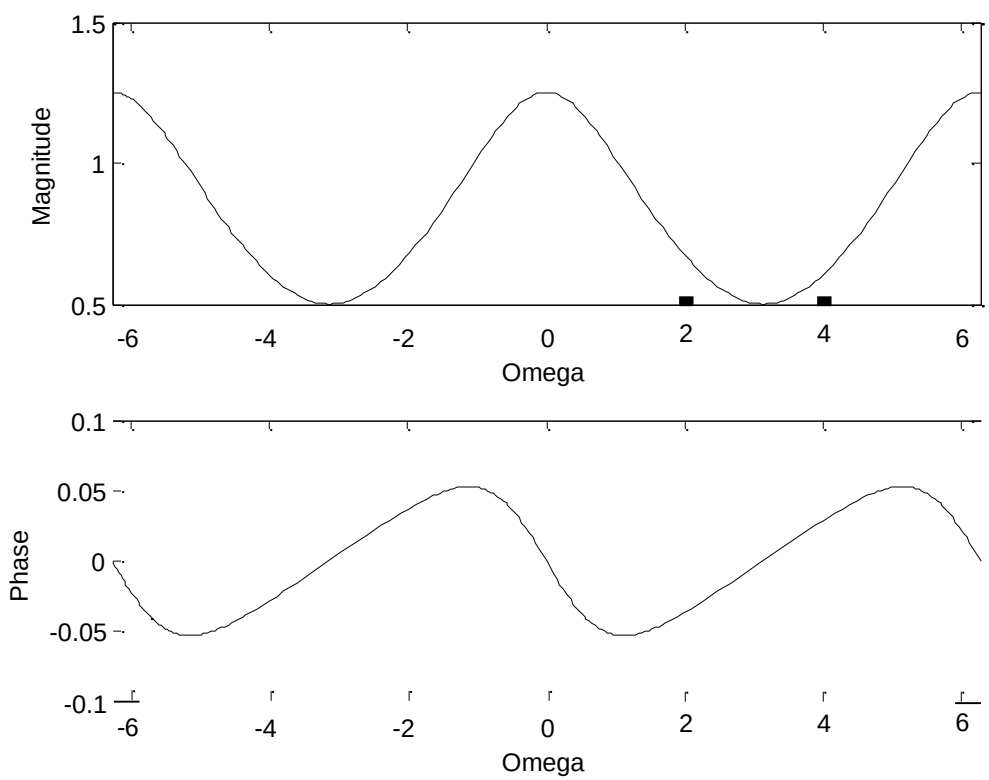
With the notation $H(\omega) = |H(\omega)|\, e^{j\angle H(\omega)}$ we can identify the magnitude and phase, respectively, as follows:

$$|H(\omega)| = \sqrt{\frac{(1+4\cos\omega)^2 + (4\sin\omega)^2}{(1-5\cos\omega)^2 + (5\sin\omega)^2}} = \sqrt{\frac{17+8\cos\omega}{26-10\cos\omega}}$$

$$\angle H(e^{j\omega}) = -\left\{\tan^{-1}\left(\frac{4\sin\omega}{1+4\cos\omega}\right) + \tan^{-1}\left(\frac{5\sin\omega}{1-5\cos\omega}\right)\right\}$$

The MATLAB program follows for $H(\omega) = \dfrac{1+4e^{-j\omega}}{1-5e^{-j\omega}}$. Note that in MATLAB we use „w" for ω and the plot ranges from -2π to 2π. Both ω and the phase, $\angle X(\omega)$, are in radians.

```
subplot(2,1,1);fplot('abs((1+4*exp(-j*w))/(1-5*exp(-j*w)))', [-2*pi,2*pi], 'k');
xlabel('Omega');ylabel('Magnitude');
subplot(2,1,2);fplot('angle((1+4*-exp(-j*w))/(1-5*exp(-j*w)))', [-2*pi,2*pi], 'k');
xlabel('Omega');ylabel('Phase');
```



Example 1.8.2 [Convolution] If $X(e^{j\omega})$ = DTFT{$x(n)$} and $y(n) = x(n)*x(-n)$ then from Properties 5 and 7 $Y(e^{j\omega})$= DTFT{$y(n)$} = DTFT{$x(n)*x(-n)$}= $X(e^{j\omega})\,X(e^{-j\omega})$. This result may also be obtained from the defining equation

$$Y(e^{j\omega}) = \sum_{n=-\infty}^{\infty}\{x(n)*x(-n)\}e^{-j\omega n}$$

The convolution within the braces is $x(n)*x(-n) = \sum_{k=-\infty}^{\infty} x(k)x(-(n-k))$, so that

$$Y(e^{j\omega}) = \sum_{n=-\infty}^{\infty}\left\{\sum_{k=-\infty}^{\infty} x(k)x(k-n)\right\}e^{-j\omega n}$$

$$= \sum_{k=-\infty}^{\infty} x(k)\sum_{n=-\infty}^{\infty} x(k-n)e^{-j\omega n}$$

Set $k-n = m$ etc.

Example 1.8.3 [2009] [Convolution] If $X(e^{j\omega})$ = DTFT{$x(n)$} and $y(n) = x(n) * x^*(-n)$ find $Y(e^{j\omega})$= DTFT{$y(n)$}.

Example 1.8.4 [2008] Find a difference equation to implement a filter with unit sample response

$$h(n) = (1/4)^n \cos(n\pi/3)\,u(n)$$

Solution In the math manipulation it would help to write $\cos(n\pi/3)$ in its exponential form. You should also memorize the two standard z-transform pairs: $\mathcal{Z}\{\cos(\omega_0 n)\,u(n)\}$ and $\mathcal{Z}\{a^n\cos(\omega_0 n)\,u(n)\}$.

Hint Either use DTFT: Find the DTFT $\mathcal{F}\{h(n)\} = H(e^{j\omega}) = \dfrac{Nr(\omega)}{Dr(\omega)}$.

1) Based on the relation $\dfrac{Y(e^{j\omega})}{X(e^{j\omega})} = H(e^{j\omega})$ set $\dfrac{Y(e^{j\omega})}{X(e^{j\omega})} = \dfrac{Nr(\omega)}{Dr(\omega)}$,
2) Cross-multiply to get $Y(e^{j\omega})\,Dr(\omega) = X(e^{j\omega})Nr(\omega)$,
3) Take inverse DTFT using time-shifting property (see below), and
4) Rearrange terms.

Or, use z-transforms: Find the z-transform $\mathcal{Z}\{h(n)\} = H(z) = \dfrac{Nr(z)}{Dr(z)}$.

1) Based on the relation $\dfrac{Y(z)}{X(z)} = H(z)$ set $\dfrac{Y(z)}{X(z)} = \dfrac{Nr(z)}{Dr(z)}$
2) Cross-multiply to get $Y(z)\,Dr(z) = X(z)\,Nr(z)$,
3) Take inverse z-transform using time-shifting property, and
4) Rearrange terms.

Example 1.8.5 [2008] Find the inverse DTFT of $X(e^{j\omega}) = \dfrac{1}{1-(1/3)\,e^{-j10\omega}}$.

Hint Use the result, derived in an earlier example, that for the exponential sequence $x(n) = a^n u(n)$, $|a| < 1$, the DTFT is

$$X(e^{j\omega}) = \sum_{n=0}^{\infty} a^n e^{-j\omega n} = \sum_{n=0}^{\infty}\left(ae^{-j\omega}\right)^n = \frac{1}{1-ae^{j\omega}} \quad \text{with } a = 1/3$$

Then use the scaling property that if $x(n) \text{ — } X\left(e^{j\omega}\right)$, then $x(n/k) \text{ — } X\left(e^{jk\omega}\right)$ with $k = 10$.

Alternatively, use the defining equation

$$\mathcal{F}^{-1}\{X(\omega)\} = x(n) = \frac{1}{2\pi}\int_{-\pi} X\left(e^{j\omega}\right) e^{j\omega n} d\omega$$

Example 1.8.6 [Scaling – compression in time] Given $x(n) \text{ — } X\left(e^{j\omega}\right)$ and $y(n) = x(2n)$, find $Y\left(e^{j\omega}\right)$.

Solution This is a *specific* result, not a general property. We have

$$Y\left(e^{j\omega}\right) = \sum_{n=-\infty}^{\infty} y(n)\, e^{-j\omega n} = \sum_{n=-\infty}^{\infty} x(2n)\, e^{-j\omega n} = \sum_{n \text{ even}}^{\infty} x(n)\, e^{-j\omega n/2}$$

$$= \sum_{n=-\infty}^{\infty} \frac{1}{2}\left\{x(n) + (-1)^n x(n)\right\} e^{-j\omega n/2} = \frac{1}{2}\sum_{n=-\infty}^{\infty} x(n)\, e^{-j\omega n/2} + \frac{1}{2}\sum_{n=-\infty}^{\infty} x(n)\left((-1)^n e^{-j\omega/2}\right)$$

$$= \frac{1}{2}\sum_{n=-\infty}^{\infty} x(n)\left[e^{-j(\omega/2)}\right]^n + \frac{1}{2}\sum_{n=-\infty}^{\infty} x(n)\left[-e^{-j(\omega/2)}\right]^n$$

Notationally, we may write this as

$$Y\left(e^{j\omega}\right) = \frac{1}{2} X\left(e^{j\omega/2}\right) + \frac{1}{2} X\left(-e^{j\omega/2}\right)$$

The *z*-transform and realization of digital filters

Review of z-transforms, Applications of z-transforms, Solution of difference equations of digital filters, Block diagram representation of linear constant-coefficient difference equations, Basic structures of IIR systems, Transposed forms, Basic structures of FIR systems, System function.

Contents:

Introduction

For continuous-time systems the **Laplace transform** is an extension of the **Fourier transform**. The Laplace transform can be applied to a broader class of signals than the Fourier transform can, since there are many signals for which the Fourier transform does not converge but the Laplace transform does. The Laplace transform allows us, for example, to perform transform analysis of unstable systems and to develop additional insights and tools for LTI system analysis.

The ***z*-transform** is the discrete-time counterpart of the Laplace transform. The *z*-transform enables us to analyze certain discrete-time signals that do not have a **discrete-time Fourier transform**. The motivations and properties of the *z*-transform closely resemble those of the Laplace transform. However, as with the relationship of the continuous time versus the discrete-time Fourier transforms, there are distinctions between the Laplace transform and the *z*-transform.

Definition The **two-sided (bilateral) *z*-transform**, $X(z)$, of the sequence $x(n)$ is defined as

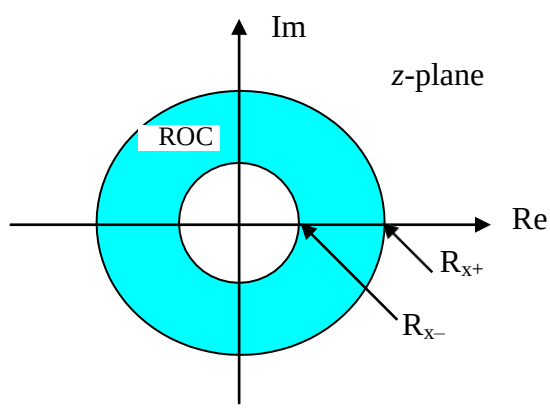
$$X(z) = \mathfrak{Z}\{x(n)\} = \sum_{n=-\infty}^{\infty} x(n) z^{-n}$$

where $z = r\,e^{j\omega}$ is the complex variable. The above power series is a **Laurent series**.

The one–sided (unilateral) z-transform is defined as

$$X_+(z) = \sum_{n=0}^{\infty} x(n) z^{-n}$$

The unilateral *z*-transform is particularly useful in analyzing causal systems specified by linear constant-coefficient difference equations with nonzero initial conditions into which inputs are stepped. It is extensively used in digital control systems.



The region of convergence (ROC) is the set of *z* values for which the above summation converges. In general the ROC is an annular region in the complex *z*-plane given by

$$\text{ROC} = R_{x-} < |z| < R_{x+}$$

Relationship between the *z*-transform and the discrete-time Fourier transform Setting $z = r\,e^{j\omega}$ in the definition gives us

$$X(z)\Big|_{z=re^{j\omega}} = \sum_{n=-\infty}^{\infty} x(n)\left(re^{j\omega}\right)^{-n} = \sum_{n=-\infty}^{\infty} [r^{-n} x(n)] e^{-j\omega n}$$

If $|z| = r = 1$, then the z-transform, evaluated on the unit circle, gives the discrete-time Fourier transform of the sequence $x(n)$, i.e.,

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n} = X(z)\Big|_{z=e^{j\omega}}$$

Example 1.1.1 The positive-time signal

$$x(t) = \begin{cases} e^{-\alpha t}, & t \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

is sampled at T-second intervals resulting in the sequence $x(nT)$ or $x(n)$

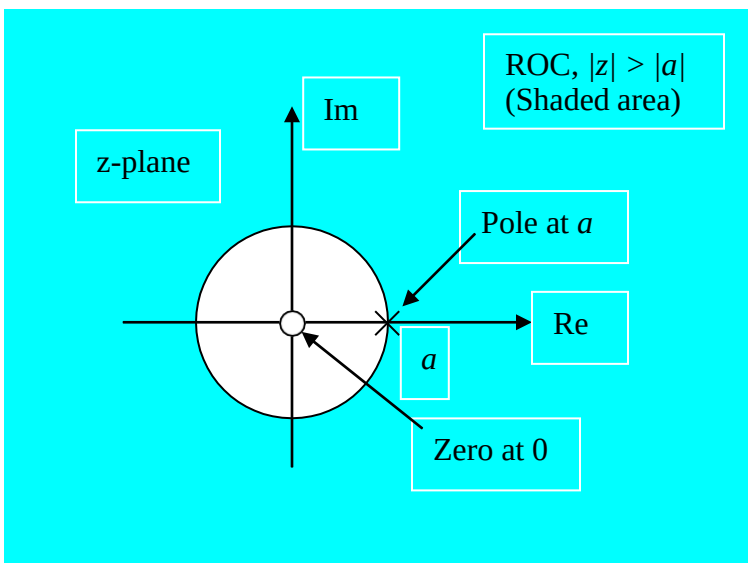
$$x(n) = e^{-\alpha t}\Big|_{t=nT} = e^{-\alpha nT} = \left(e^{-\alpha T}\right)^n = a^n, \; a = e^{-\alpha T} \quad \text{and } n \geq 0$$

$$x(n) = \begin{cases} a^n, & n \geq 0 \\ 0, & n < 0 \end{cases}$$

If $a < 1$ this sequence decays exponentially to 0 as $n \to \infty$. Substituting $x(n)$ into the defining equation, the z-transform is

$$\mathfrak{Z}\{x(n)\} = X(z) = \sum_{n=0}^{\infty} a^n z^{-n} = \sum_{n=0}^{\infty} (az^{-1})^n = \frac{1}{1-az^{-1}}, \qquad |az^{-1}| < 1$$

$$= \frac{1}{1-az^{-1}} = \frac{z}{z-a}, \qquad |z| > |a|$$



The ROC is $|z| > |a|$. This $X(z)$ is a rational function (a ratio of polynomials in z). The roots of the numerator polynomial are the zeros of $X(z)$ and the roots of the denominator polynomial are the poles of $X(z)$.

This is a right-sided sequence. *Right-sided sequences have a ROC that is the exterior of a circle* with radius R_{x-} ($|z| > |a|$ in this case). If the ROC is the exterior of a circle it is a right-sided sequence.

Definition A **right-sided sequence** $x(n)$ is one for which $x(n) = 0$ for all $n < n_0$ where n_0 is positive or negative but finite. If $n_0 \geq 0$ then $x(n)$ is a *causal* or *positive-time sequence.*

Example 1.1.2 The negative-time sequence $x(n) = -b^n\, u(-n-1)$. Recall that the unit step sequence $u(.) = 1$ if the argument of $u(.)$ is ≥ 0, i.e., if $(-n-1) \geq 0$ or $n \leq -1$.

$$x(n) = \begin{cases} -b^n, & n \leq -1 \\ 0, & \text{otherwise} \end{cases}$$

If $b > 1$ this sequence decays exponentially to 0 as $n \to -\infty$. The z-transform is,

$$\mathfrak{Z}\{x(n)\} = X(z) = \sum_{n=-\infty}^{\infty} x(n) z^{-n} = \sum_{n=-\infty}^{-1} -b^n z^{-n} = -\sum_{n=-\infty}^{-1} (bz^{-1})^n$$

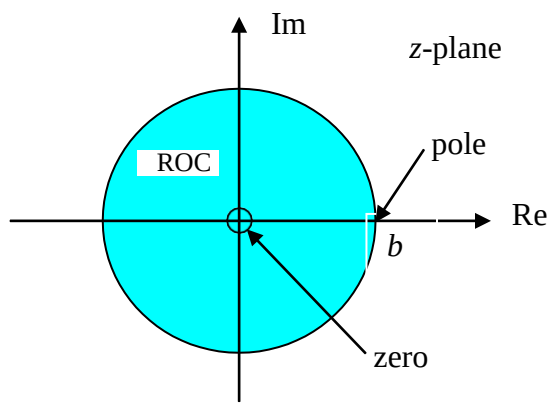
Let $n = -m$ and change the limits accordingly to get,

$$X(z) = -\sum_{m=\infty}^{1} (bz^{-1})^{-m} = 1 - \sum_{m=0}^{\infty} \left(b^{-1}z\right)^m$$

We added 1 in the last step above to make up for the $m = 0$ term within the summation. The result is,

$$X(z) = 1 - \frac{1}{(1 - zb^{-1})}, \quad |z\, b^{-1}| < 1$$

$$= \frac{z}{z - b}, \qquad \text{ROC is } |z| < |b|$$



This is a left-sided sequence. Such a sequence has a region of convergence which is the interior of a circle, $|z| < R_{x+}$. In this case the ROC is $|z| < |b|$.

Note that if $b = a$ then the two examples above have exactly the same $X(z)$. So what makes the difference? The region of convergence makes the difference.

Definition A **left-sided sequence** $x(n)$ is one for which $x(n) = 0$ for all $n \geq n_0$, where n_0 is positive or negative but finite. If $n_0 \leq 0$ then $x(n)$ is an *anticausal* or *negative-time sequence*.

Example 1.1.3 [Two-sided sequence] This is the sum of the positive- and negative-time sequences of the previous two examples.

$$y(n) = \begin{cases} a^n, & n \geq 0 \\ -b^n, & n < 0 \end{cases} \qquad = a^n\, u(n) - b^n\, u(-n-1)$$

Substituting into the defining equation,

$$Y(z) = \mathfrak{Z}\{y(n)\} = \sum_{n=-\infty}^{\infty}[a^n u(n) - b^n u(-n-1)]z^{-n} = \sum_{n=0}^{\infty} a^n z^{-n} - \sum_{n=-\infty}^{-1} b^n z^{-n}$$

Now, from Examples 1 and 2,

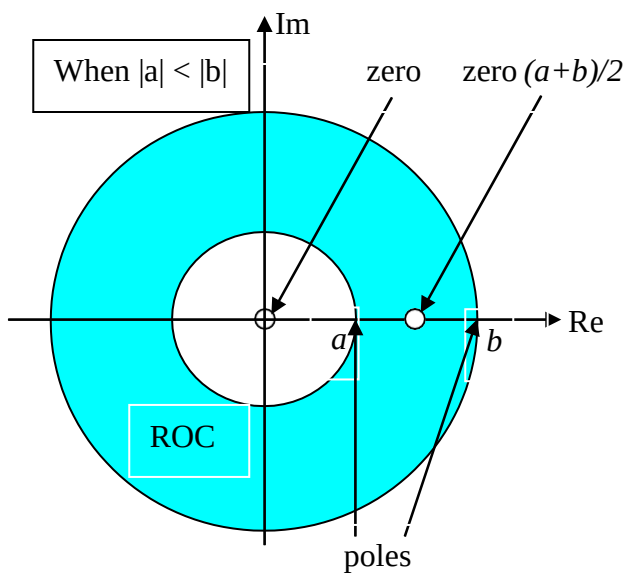
$$\sum_{n=0}^{\infty} a^n z^{-n} = \frac{z}{z-a} \quad (\text{ROC } |z| > |a|) \qquad \& \qquad \sum_{n=-\infty}^{-1} b^n z^{-n} = \frac{z}{z-b} \quad (\text{ROC } |z| < |b|)$$

So, the desired transform $Y(z)$ has a region of convergence equal to the intersection of the two separate ROC's $|z| > |a|$ and $|z| < |b|$. Thus

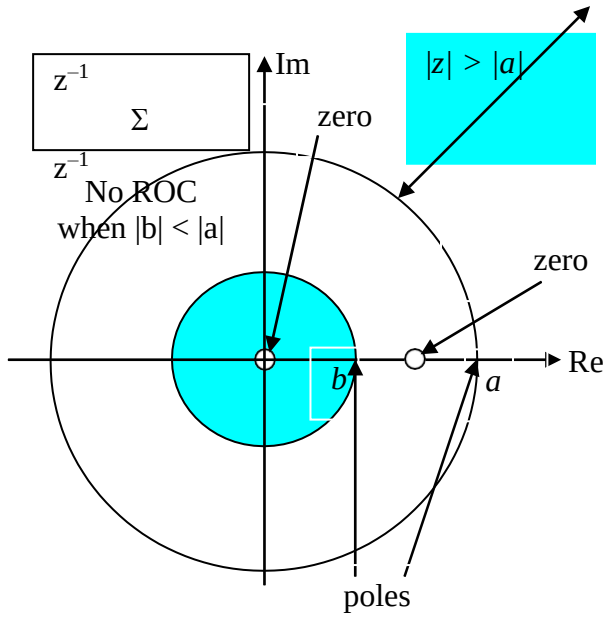
$$Y(z) = \frac{z}{z-a} + \frac{z}{z-b}, \text{ with ROC } \{|z| > |a|\} \cap \{|z| < |b|\}$$

$$= \frac{z(2z-a-b)}{(z-a)(z-b)}, \text{ with ROC } |a| < |z| < |b|$$

The ROC is the overlap of the shaded regions, that is, the annular region between $|a|$ and $|b|$. The two zeros are at 0 and $(a+b)/2$, and the two poles at a and b.



If $|b| < |a|$ the transform does not converge.



In the above three examples we may express the z-transform both as a ratio of polynomials in z (i.e., positive powers) and as a ratio of polynomials in z^{-1} (negative powers). From the definition of the z-transform, we see that for sequences which are zero for $n < 0$, $X(z)$ involves only negative powers of z. However, reference to the poles and zeros is always in terms of the roots of the numerator and denominator expressed as polynomials in z. Also, it is sometimes convenient to refer to $X(z)$, written as a ratio of polynomials in z (i.e., positive power of z), as having poles at infinity if the degree of the numerator exceeds the degree of the denominator or zeros at infinity if the numerator is of smaller degree than the denominator.

Example 4.1.4 [Finite-length sequence] Only a finite number of sequence values are non-zero, as given below.

$$x(n) = \begin{cases} 0 \text{ for } n < N_1 \text{ and for } n > N_2, \text{ where } N_1 \text{ and } N_2 \text{ are finite} \\ \text{non-zero for } N_1 \le n \le N_2 \end{cases}$$

By the defining equation we have

$$X(z) = \sum_{n=N_1}^{N_2} x(n) z^{-n} = x(N_1) z^{-N_1} + \ldots + x(N_2) z^{-N_2}$$

Convergence of this expression requires simply that $|x(n)| < \infty$ for $N_1 \le n \le N_2$. Then z may take on all values except $z = \infty$ if N_1 is negative and $z = 0$ if N_2 is positive. Thus the ROC is at least $0 < |z| < \infty$ and it may include either $z = 0$ or $z = \infty$ depending on the sign of N_1 and N_2.

Important properties of z-transforms

The proofs are easily obtained by using the basic z-transform definition and transformations in the summation. [Sec 2.3 Oppenheim & S]

(1) Linearity If $\mathcal{Z}[x(n)] = X(z)$ with ROC $r_{x1} < |z| < r_{x2}$ and $\mathcal{Z}[y(n)] = Y(z)$ with ROC $r_{y1} < |z| < r_{y2}$ then $\mathcal{Z}[a\,x(n) + b\,y(n)] = a\,X(z) + b\,Y(z)$ with ROC at least the overlap of the ROC's of $X(z)$ and $Y(z)$. If there is any pole-zero cancellation due to the linear combination, then the ROC may be larger.

***(2)* Translation (Time-shifting)** If $\mathcal{Z}[x(n)] = X(z)$ with ROC $r_1 < |z| < r_2$ then $\mathcal{Z}[x(n-k)] = z^{-k}X(z)$ with the same ROC except for the possible addition or deletion of $z = 0$ or $z = \infty$ due to z^{-k}.

Example Given $x(n) = \{1, 2\}$ and $x_2(n) = x(n+2)$ find $X(z)$ and $X_2(z)$ and their respective ROCs. $X(z) = 1 + 2z^{-1}$, ROC: entire z-plane except $z = 0$; $X_2(z) = z^2(1 + 2z^{-1}) = z^2 + 2z$, ROC: entire z-plane except $z = \infty$.

(3) Multiplication by a complex exponential sequence (Scaling in the *z*-domain) If $\mathcal{Z}[x(n)] = X(z)$ with ROC $r_1 < |z| < r_2$ then $\mathcal{Z}[a^n x(n)] = X(z)\big|_{z \to (z/a)}$ with ROC $|a|\,r_1 < |z| < |a|\,r_2$.

Example Given $x(n) = \{1, 2\}$ and $x_2(n) = 0.5^n x(n-2)$ find $X(z)$ and $X_2(z)$ and their respective ROCs.

(4) Multiplication by a ramp If $\mathcal{Z}[x(n)] = X(z)$ with ROC $r_1 < |z| < r_2$ then

$$\mathcal{Z}[n\,x(n)] = -z\frac{dX(z)}{dz} \text{ with ROC } r_1 < |z| < r_2$$

Example Given $x(n) = \{1, 2\}$ and $x_2(n) = (1 + n + n^2)\,x(n)$ find $X(z)$ and $X_2(z)$ and their respective ROCs. $\mathcal{Z}[x_2(n)] = \mathcal{Z}[1] + \mathcal{Z}[n\,x(n)] + \mathcal{Z}[n\,[n\,x(n)]] = \ldots$

(5) Time reversal If $\mathcal{Z}[x(n)] = X(z)$ with ROC $r_1 < |z| < r_2$ then $\mathcal{Z}[x(-n)] = X(z^{-1})$ with ROC $(1/r_2) < |z| < (1/r_1)$

Example Given $x(n) = 2^{-n}u(n)$ and $X(z) = z\,X(z^{-1})$ determine $x_2(n)$.

$x(n) = (0.5)^n u(n)$, $X(z) = \dfrac{z}{z - 0.5}$, ROC: $0.5 < |z|$

$\mathcal{Z}^{-1}\{X(z^{-1})\} = x(-n)$; $x_2(n) = \mathcal{Z}^{-1}\{z\,X(z^{-1})\} = x(-(n+1)) = 2^{-(-(n+1))}u(-(n+1))$

(6) Convolution in time domain leads to multiplication in frequency domain Given $\mathcal{Z}[x(n)] = X(z)$ with ROC $z \in R_x$ and $\mathcal{Z}[y(n)] = Y(z)$ with ROC $z \in R_y$ and $x(n)*y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k)$ then

$\mathcal{Z}[x(n)*y(n)] = X(z).Y(z)$ with ROC $z \in R_x \cap R_y$.

***(7)* Multiplication in time domain leads to convolution in frequency domain** If $\mathcal{Z}[x(n)] = X(z)$ with ROC $r_{x1} < |z| < r_{x2}$ and $\mathcal{Z}[y(n)] = Y(z)$ with ROC $r_{y1} < |z| < r_{y2}$ then

$$\mathcal{Z}[x(n).y(n)] = \frac{1}{j2\pi}\oint_{C_2} X(v)Y\left(\frac{z}{v}\right)v^{-1}dv, \qquad \text{ROC } r_{x1}\,r_{y1} < |z| < r_{x2}\,r_{y2}.$$

where $\oint_{C_2}$ is a complex contour integral and C_2 is a closed contour in the intersection of the ROCs of $X(v)$ and $Y(z/v)$.

(8) Initial Value Theorem If $x(n)$ is a causal sequence with z transform $X(z)$, then

$$x(0) = \lim_{z \to \infty} X(z)$$

(9) Final Value Theorem If $\mathfrak{Z}[x(n)] = X(z)$ and the poles of $X(z)$ are all inside the unit circle then the value of $x(n)$ as $n \to \infty$ is given by

$$x(n)\big|_{n \to \infty} = \lim_{z \to 1} [(z-1)\, X(z)]$$

Some also give this as $x(n)\big|_{n \to \infty} = \lim_{z \to 1} [(1 - z^{-1})\, X(z)]$

Transforms of some useful sequences

(1) The **unit sample** $\delta(n)$:

$$\mathfrak{Z}[\delta(n)] = \sum_{n=-\infty}^{\infty} \delta(n) z^{-n} = 1.\, z^0 = 1, \qquad \text{ROC all } z$$

Delayed unit sample $\delta(n-k)$:

$$\mathfrak{Z}[\delta(n-k)] = \sum_{n=-\infty}^{\infty} \delta(n-k) z^{-n} = 1.\, z^{-k} = z^{-k}, \qquad \text{ROC } |z| > 0 \text{ if } k > 0 \ (|z| < \infty \text{ if } k < 0)$$

(2) **Unit step** $u(n)$ (positive time):

$$\mathfrak{Z}[u(n)] = \sum_{n=-\infty}^{\infty} u(n)\, z^{-n} = \sum_{n=0}^{\infty} 1\, z^{-n} = 1 + z^{-1} + z^{-2} + \ldots$$

$$= \frac{1}{1 - z^{-1}} = \frac{z}{z-1}, \qquad \text{ROC } |z^{-1}| < 1 \text{ or } |z| > 1$$

(3) **Unit step** $-u(-n-1)$ (negative time):

$$\mathfrak{Z}[-u(-n-1)] = \sum_{n=-\infty}^{\infty} -u(-n-1) z^{-n} = \sum_{n=-\infty}^{-1} (-1)\, z^{-n} = -\sum_{n=-\infty}^{-1} z^{-n} = 1 - \sum_{n=-\infty}^{0} z^{-n}$$

$$= 1 - (1 + z + z^2 + \ldots) = 1 - \frac{1}{1-z} = \frac{z}{z-1}, \qquad \text{ROC } |z| < 1$$

(4) **Exponential** $a^n u(n)$, derived in earlier example:

$$\mathfrak{Z}[\, a^n u(n)\,] = \frac{z}{z-a}, \qquad \text{ROC } |z| > |a|$$

(5) **Exponential** $-b^n u(-n-1)$; negative time; derived earlier:

$$\mathfrak{Z}[-b^n u(-n-1)] = \frac{z}{z-b}, \qquad \text{ROC } |z| < |b|$$

(6) **Unit ramp** $n\, u(n)$. Given that $\mathfrak{Z}[u(n)] = U(z) = \dfrac{z}{z-1}$

$$\mathfrak{Z}[n\, u(n)] = -z \frac{dU(z)}{dz} = -z \frac{d}{dz}\left(\frac{z}{z-1}\right) = \frac{-z[(z-1).1 - z.1]}{(z-1)^2}$$

$$= \frac{z}{(z-1)^2} \qquad \text{ROC } |z| > 1, \text{ same as that of } U(z)$$

(7) **Sinusoid** $\sin \omega_0 n\, u(n)$: $\mathfrak{Z}\{\sin \omega_0 n\, u(n)\} = \dfrac{z \sin \omega_0}{z^2 - 2z \cos \omega_0 + 1}$, ROC $|z| > 1$

$$\mathcal{Z}\{\sin\omega_0 n\, u(n)\} = \sum_{n=0}^{\infty} \sin\omega_0 n\, z^{-n} = \sum_{n=0}^{\infty} \left(\frac{e^{j\omega_0 n} - e^{-j\omega_0 n}}{j2} \right) z^{-n}$$

$$= \frac{1}{j2} \left(\sum_{n=0}^{\infty} (e^{j\omega_0} z^{-1})^n - \sum_{n=0}^{\infty} (e^{-j\omega_0} z^{-1}) \right)$$

$$= \frac{1}{j2} \left(\frac{1}{1 - e^{j\omega_0} z^{-1}} - \frac{1}{1 - e^{-j\omega_0} z^{-1}} \right), \text{ ROC } |e^{\pm j\omega_0} z^{-1}| < 1 \text{ or } |z| > 1$$

$$= \frac{1}{j2} \left(\frac{z}{z - e^{j\omega_0}} - \frac{z}{z - e^{-j\omega_0}} \right)$$

$$= \frac{z}{j2} \left(\frac{(z - e^{-j\omega_0}) - (z - e^{j\omega_0})}{z^2 - z(e^{j\omega_0} + e^{-j\omega_0}) + 1} \right)$$

$$= \frac{z}{j2} \left(\frac{(e^{j\omega_0} - e^{-j\omega_0})}{z^2 - z(e^{j\omega_0} + e^{-j\omega_0}) + 1} \right)$$

Using the identities

$$\cos\omega_0 n = \frac{e^{j\omega_0} + e^{-j\omega_0}}{2} \quad \text{and} \quad \sin\omega_0 n = \frac{e^{j\omega_0} - e^{-j\omega_0}}{j2}$$

we have

$$\mathcal{Z}\{\sin\omega_0 n\, u(n)\} = \frac{z \sin\omega_0}{z^2 - 2z\cos\omega_0 + 1}, \qquad \text{ROC } |z| > 1$$

As an extension, using property #3,

$$\mathcal{Z}\{a^n \sin\omega_0 n\, u(n)\} = \frac{(z/a)\sin\omega_0}{(z/a)^2 - 2(z/a)\cos\omega_0 + 1} = \frac{az\sin\omega_0}{z^2 - 2az\cos\omega_0 + a^2}, \qquad \text{ROC } |z| > a$$

(8) **Cosinusoid** $\cos\omega_0 n\, u(n)$. Using the relation $\cos\omega_0 n = \dfrac{e^{j\omega_0 n} + e^{-j\omega_0 n}}{2}$ and a procedure similar to that for the sinusoid we get

$$\mathcal{Z}\{\cos\omega_0 n\, u(n)\} = \frac{z^2 - z\cos\omega_0}{z^2 - 2z\cos\omega_0 + 1}, \qquad \text{ROC } |z| > 1$$

As an extension, using property #3,

$$\mathcal{Z}\{a^n \cos\omega_0 n\, u(n)\} = \frac{(z/a)^2 - (z/a)\cos\omega_0}{(z/a)^2 - 2(z/a)\cos\omega_0 + 1} = \frac{z^2 - za\cos\omega_0}{z^2 - 2az\cos\omega_0 + a^2}, \qquad \text{ROC } |z| > a$$

Region of convergence and stability

Suppose *x(n)* is a causal sequence that can be written as a sum of complex exponentials. This takes in a wide class of signals including sinusoids, exponentials, and products thereof. Let

$$x(n) = \sum_{i=1}^{N} a_i^n\, u(n)$$

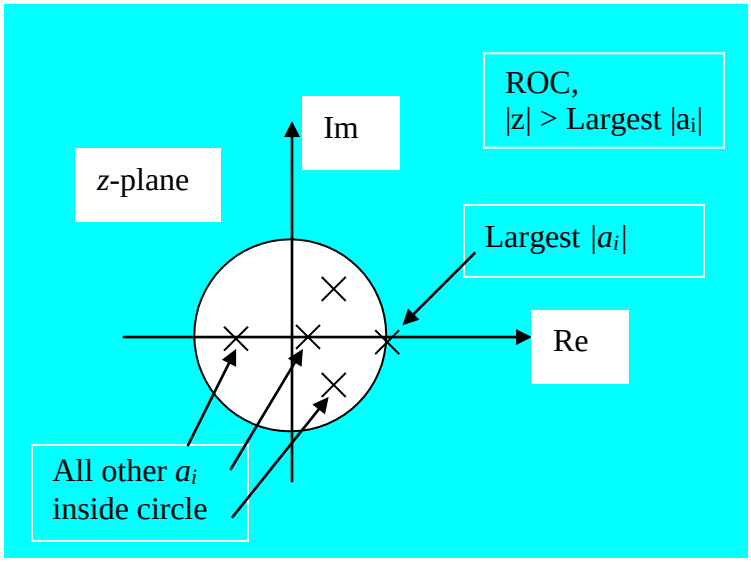
Taking the *z* transform of *x(n)* gives

$$\mathcal{Z}[x(n)] = X(z) = \sum_{i=1}^{N} \frac{z}{z - a_i}$$

The region of convergence R is the intersection of the regions of convergence for each exponential as follows:

$$R = \bigcap_{i=1}^{N} R_i \text{ where } R_i = \{z\colon |z| > |a_i|\}$$

Therefore, $R = \{z\colon |z| >$ largest of $|a_i|\}$ as shown here (Figure)



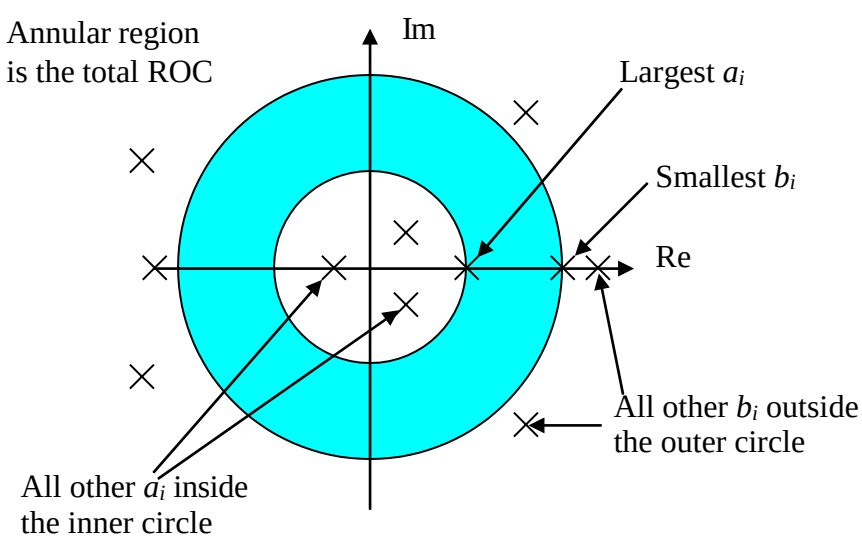
Since the ROC for a translated exponential remains the same as that for the original exponential, all right-sided sequences that are sums of translated exponentials have ROCs similar to that expressed above.

By a similar argument all left-sided sequences expressible as a sum of translated complex exponentials have a ROC, L, given by

$$L = \{z\colon |z| < \text{smallest of } |b_i|\}$$

If we have a combination of right- and left-sided sequences, the corresponding ROC is the intersection of R and L. Therefore the total ROC becomes an annular region as shown below and given by

$$R_{Total} = R \cap L = \{z\colon \text{Largest of } |a_i| < |z| < \text{smallest of } |b_i|\}$$



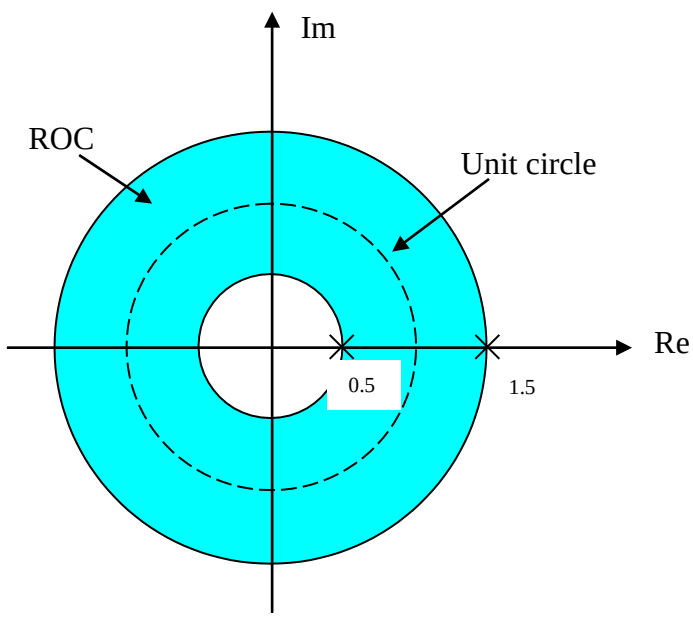
The stability of a system with an impulse response that is the sum of translated right- and left-sided sequences can be determined from the region of convergence. Assume that *h(n)* is the unit sample response of a causal or non-causal linear shift-invariant system. Let $\mathfrak{Z}[h(n)] = H(z)$, the so-called system function. Then:

Theorem A linear shift-invariant system with system function *H(z)* is BIBO stable if and only if the ROC for *H(z)* contains the unit circle.

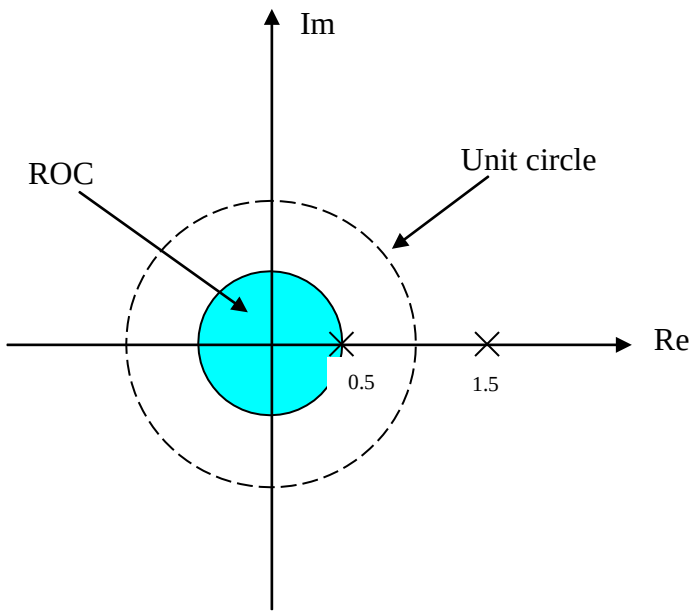
This theorem can be used to determine stability for a given *H(z)* without obtaining the impulse response or checking outputs for all bounded input signals.

Illustration of stability and causality For A system function with 2 poles at, say, $z = 0.5$, and $z = 1.5$, there are three possible regions of convergence.

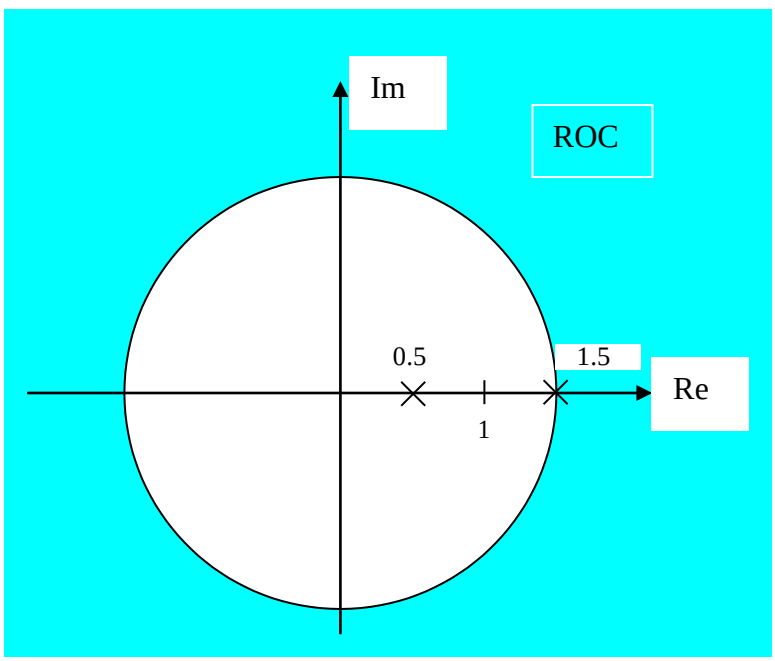
(1) ROC is $0.5 < |z| < 1.5$. Here the system is stable since the unit circle is inside the region of convergence. The impulse response, *h(n)*, is two-sided, so the system is noncausal.



(2) ROC is $|z| < 0.5$. Here the system is not stable. The impulse response, *h(n)*, is left-sided, so the system is noncausal.



(3) ROC is $|z| > 1.5$. Here the system is not stable. The impulse response, *h(n)*, is right-sided, so the system may be causal.



Inverse *z*-transform by partial fractions

(Aside) Comparison of inverse z-transform methods A limitation of the **power series method** is that it does not lead to a closed form solution (although this can be deduced in simple cases), but it is simple and lends itself to computer implementation. However, because of its recursive nature care should be taken to minimize possible build-up of numerical errors when the number of data points in the inverse *z*-transform is large, for example by using double precision.

Both the **partial fraction method** and the **inversion integral method** require the evaluation of residues albeit performed in different ways. The partial fraction method requires the evaluation of the residues of *X(z)* or $\frac{X(z)}{z}$. The complex inversion integral requires the evaluation of the residues of $X(z)z^{n-1}$. In many instances evaluation of the complex inversion integral is needlessly difficult and involved.

Both the partial fraction method and the inversion integral method lead to closed form solutions. The main disadvantage is having to factorize the denominator polynomial of *X(z)* when it is of order greater than 2. Another disadvantage is multiple order poles and the resulting differentiation(s) when determining residues.

The partial fraction method directly generates the coefficients of parallel structures for digital filters. The inversion integral method is widely used in the analysis of quantization errors in discrete-time systems.

(End of Aside)

As in Laplace transforms, in order to expand a rational function into partial fractions, the degree of the numerator should be less than the degree of the denominator – proper fraction. If it is not then we perform long division as below where *Q(z)* is the quotient and $N_1(z)$ is the remainder.

$$X(z) = \frac{N(z)}{D(z)} = Q(z) + \frac{N_1(z)}{D(z)}$$

Long Division

$$\begin{array}{rrl} & Q(z) & \leftarrow \text{Quotient} \\ \text{Denominator} \rightarrow \quad D(z) & \boxed{\begin{array}{l} N(z) \\ \text{---} \end{array}} & \leftarrow \text{Numerator} \\ & N_1(z) & \leftarrow \text{Remainder} \end{array}$$

The long division is done until we get a remainder polynomial $N_1(z)$ whose degree is less than the degree of the denominator *D(z)*. We then obtain *x(n)* as

$$x(n) = \mathcal{Z}^{-1}\{X(z)\} = \mathcal{Z}^{-1}\{Q(z)\} + \mathcal{Z}^{-1}\left\{\frac{N_1(z)}{D(z)}\right\}$$

Since $N_1(z)/D(z)$ is a proper fraction it can be expanded into partial fractions. The overall inverse transform is obtained by looking up a table of *z*-transform pairs.

However, there is an alternative available in the case of *z*-transforms which is not available in Laplace transforms. This is a result of the fact that *z*-transforms are characterized by a *z* in the numerator (as can be verified by looking at a table of *z*-transforms). Therefore, instead of expanding *X(z)* we may, instead, expand [*X(z)/z*] into partial fractions giving

$$\frac{X(z)}{z} = \frac{A}{z-z_1} + \frac{B}{z-z_2} + \ldots$$

so that *X(z)* is given by

$$X(z) = \frac{Az}{z-z_1} + \frac{Bz}{z-z_2} + \ldots$$

This can be inverted by a simple look-up of a table of transforms. Note also that in some cases $X(z) = N(z)/D(z)$ may not be a proper fraction but [*X(z)/z*] is and, therefore, this method obviates the need for long division of *N(z)* by *D(z)*. (In still other cases even [*X(z)/z*] may not be a proper fraction. See later under "General procedure for partial fraction expansion".)

Example 1.5.1 (See also long division later). Find the inverse z-transform, using partial fractions, of

$$X(z) = \frac{2z^2 - 3z}{z^2 - 3z + 2} = \frac{N(z)}{D(z)}$$

This is not a proper fraction since the degree of the numerator is not less than the degree of the denominator. However, $(X(z)/z)$ is a proper fraction

$$\frac{X(z)}{z} = \frac{2z - 3}{z^2 - 3z + 2} = \frac{2z - 3}{(z-1)(z-2)}$$

which has the partial fraction expansion

$$\frac{X(z)}{z} = \frac{1}{(z-1)} + \frac{1}{(z-2)} \qquad \text{or} \qquad X(z) = \frac{z}{(z-1)} + \frac{z}{(z-2)}$$

By looking up a table of z-transforms the inverse z-transform is

$$\mathcal{Z}^{-1}\{X(z)\} = x(n) = u(n) + 2^n u(n)$$

Note that we are giving here the causal solution that corresponds to a ROC $|z| > 2$ (not $1 < |z| < 2$ or $|z| < 1$) so that $x(n)$ is a right-sided sequence.

The **alternative** method is to divide $N(z)$ by $D(z)$ as below (as is standard practice in Laplace transforms). Note that in this long division the numerator and denominator polynomials are arranged in the order of decreasing powers of z. There are three other ways (all of them wrong) of arranging the two polynomials for the long division.

Long Division

		2	←Quotient
Denominator→	$z^2 - 3z + 2$	$2z^2 - 3z$	←Numerator
		$2z^2 - 6z + 4$	
		$3z - 4$	←Remainder

Thus $X(z)$ can be expressed as

$$X(z) = 2 + \frac{3z-4}{(z-1)(z-2)} = 2 + X_1(z) \qquad \text{where } X_1(z) = \frac{3z-4}{(z-1)(z-2)}$$

$X_1(z)$ is a proper fraction and can be expanded into partial fractions as below:

$$X_1(z) = \frac{3z-4}{(z-1)(z-2)} = \frac{A}{z-1} + \frac{B}{z-2}$$

Solving for A and B we get $A = 1$ and $B = 2$, so that $X(z)$ may be written

$$X(z) = 2 + \frac{1}{z-1} + \frac{2}{z-2}$$

Taking the inverse z-transform we get

$$x(n) = \mathcal{Z}^{-1}\{X(z)\} = \mathcal{Z}^{-1}\left\{2 + \frac{1}{z-1} + \frac{2}{z-2}\right\}$$

$$= \mathcal{Z}^{-1}\{2\} + \mathcal{Z}^{-1}\left\{\frac{1}{z-1}\right\} + \mathcal{Z}^{-1}\left\{\frac{2}{z-2}\right\}$$

A term like $\mathcal{Z}^{-1}\left\{\frac{1}{z-1}\right\}$ is handled by writing $\frac{1}{z-1}$ as $\frac{z^{-1}.z}{z-1}$. We know that $\mathcal{Z}^{-1}\left\{\frac{z}{z-1}\right\} = u(n)$, so

$$\mathcal{Z}^{-1}\left\{z^{-1}\frac{z}{z-1}\right\} = u(n-1)$$

Similarly $\mathcal{Z}^{-1}\left(\frac{2}{z-2}\right) = 2 \,.\, 2^{n-1}\, u(n-1)$. Thus

$$x(n) = 2\,\delta(n) + u(n-1) + 2 \,.\, 2^{n-1}\, u(n-1)$$

This can be verified to be equivalent to $x(n) = u(n) + 2^n\, u(n)$ obtained earlier.

In MATLAB (Partial fractions) The partial fractions may be computed by using the *residuez* function. In this method $X(z)$ is arranged as a ratio of polynomials in negative powers of z and, in the denominator, the leading coefficient $a_0 \neq 0$. See "General procedure for partial fraction expansion" later.

$$X(z) = \frac{2-3z^{-1}}{1-3z^{-1}+2z^{-2}} = K + \frac{R_1}{1-p_1 z^{-1}} + \frac{R_2}{1-p_2 z^{-1}}$$

We define the coefficient vectors b = [2, -3] and a = [1, -3, 2]; $R = [R_1, R_2]$ represents the residues (partial fraction constants), $p = [p_1, p_2]$ the poles and K a constant.

```
%Partial fractions
b = [2, -3], a = [1, -3, 2],
[R, p, K] = residuez (b, a)
```

The MATLAB results returned are

```
R =
   1
   1
p =
   2
   1
K =
   []
```

The MATLAB output tells us that the poles are at $z = 2$ and $z = 1$ and the corresponding residues are, respectively, 1 and 1. Further $K = 0$. Therefore,

$$X(z) = \frac{2-3z^{-1}}{1-3z^{-1}+2z^{-2}} = 0 + \frac{1}{1-2z^{-1}} + \frac{1}{1-1z^{-1}} = \frac{z}{z-2} + \frac{z}{z-1}$$

Note that the $X(z) = \frac{1}{1-2z^{-1}} + \frac{1}{1-1z^{-1}}$ obtained by the *residuez* function and $\frac{X(z)}{z} = \frac{1}{(z-1)} + \frac{1}{(z-2)}$ are the same since $X(z)$ has no repeated poles. This won't be the case if $X(z)$ has repeated poles.

Example 1.5.2 Find if the discrete LTI system described by

$$y(n) - y(n-1) + 0.5\, y(n-2) = x(n) + x(n-1)$$

is BIBO stable or not. Find its transfer function and impulse response. Sketch its pole-zero plot.
Solution Take the z-transform of both sides:

$$\mathcal{Z}\{y(n) - y(n-1) + 0.5\, y(n-2)\} = \mathcal{Z}\{x(n) + x(n-1\}$$
$$Y(z) - z^{-1} Y(z) + 0.5\, z^{-2} Y(z) = X(z) + z^{-1} X(z)$$
$$Y(z)\,(1 - z^{-1} + 0.5\, z^{-2}) = X(z)\,(1 + z^{-1})$$

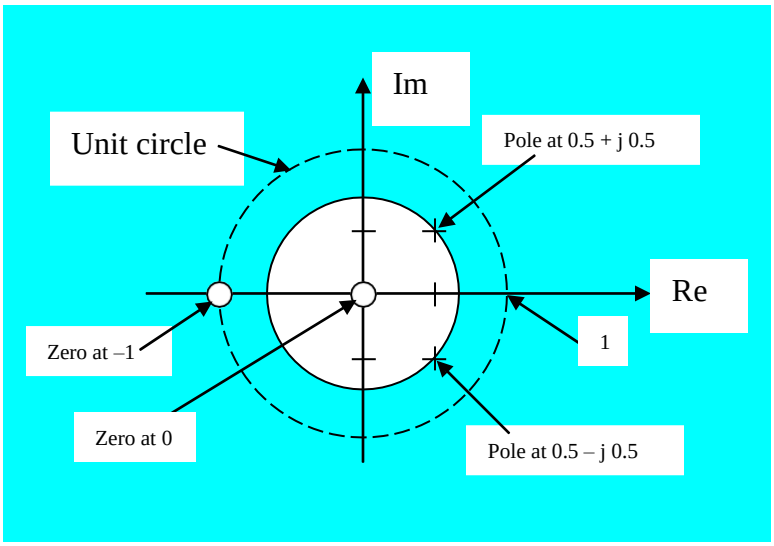
$$H(z) = \frac{Y(z)}{X(z)} = \frac{1+z^{-1}}{1-z^{-1}+0.5z^{-2}} = \frac{(z+1)/z}{(z^2-z+0.5)/z^2} = \frac{z(z+1)}{z^2-z+0.5}$$

The denominator has roots at

$$z_1, z_2 = \frac{-(-1)\pm\sqrt{(-1)^2-4.1.(0.5)}}{2} = \frac{1\pm\sqrt{1-2}}{2} = \frac{1\pm j1}{2} = 0.5 \pm j0.5$$

$$= (0.5 + j0.5) \text{ and } (0.5 - j0.5)$$

Thus the transfer function $H(z)$ has zeros at $z = 0$, $z = 1$, and poles at $z = 0.5 \pm j0.5$. For a causal system (right-sided sequence, $h(n)$) the region of convergence is $|z| > |0.5 + j0.5|$ or $|z| > 0.707$. (Figure)



The impulse response is given by $h(n) = \mathcal{Z}^{-1}\{H(z)\}$. We need partial fractions for $H(z)$; we shall instead handle $H(z)/z$:

$$\frac{H(z)}{z} = \frac{z+1}{z^2-z+0.5} = \frac{z+1}{(z-0.5-j0.5)(z-0.5+j0.5)}$$

$$= \frac{A}{z-0.5-j0.5} + \frac{A^*}{z-0.5+j0.5}$$

Solving for A and A^*, we get

$$A = \left.\frac{z+1}{z-0.5+j0.5}\right]_{z=0.5+j0.5} = \frac{0.5+j0.5+1}{0.5+j0.5-0.5+j0.5} = \frac{1.5+j0.5}{j}$$

$$= 0.5 - j1.5 = \sqrt{5/2}\; e^{-j\tan^{-1}3}$$

$$A^* = 0.5 + j1.5$$

Thus we have

$$\frac{H(z)}{z} = \frac{A}{z-(0.5+j0.5)} + \frac{A^*}{z-(0.5-j0.5)}$$

$$H(z) = A\,\frac{z}{z-\underbrace{(0.5+j0.5)}_{=a}} + A^*\,\frac{z}{z-\underbrace{(0.5-j0.5)}_{=b}}$$

where

$$a = 0.5 + j0.5 = \sqrt{0.5^2 + 0.5^2}\; e^{j\tan^{-1}1} = \frac{1}{\sqrt{2}} e^{j\pi/4}$$

$$b = 0.5 - j0.5 = \frac{1}{\sqrt{2}} e^{-j\pi/4}$$

The inverse z-transform is

$$h(n) = \begin{cases} A\,a^n + A^*\,b^n, & n \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

So that for $n \geq 0$,

$$h(n) = A\left(\frac{1}{\sqrt{2}} e^{j\pi/4}\right)^n + A^*\left(\frac{1}{\sqrt{2}} e^{-j\pi/4}\right)^n$$

$$= A\left(\frac{1}{\sqrt{2}}\right)^n e^{j\pi n/4} + A^*\left(\frac{1}{\sqrt{2}}\right)^n e^{-j\pi n/4}$$

$$= \left(-\frac{1}{2} - j\frac{3}{2}\right)\left(\frac{1}{\sqrt{2}}\right)^n e^{j\pi n/4} + \left(-\frac{1}{2} + j\frac{3}{2}\right)\left(\frac{1}{\sqrt{2}}\right)^n e^{-j\pi n/4}$$

$$= -\frac{1}{2}\left(\frac{1}{\sqrt{2}}\right)^n (e^{j\pi n/4} + e^{-j\pi n/4}) + j\frac{3}{2}\left(\frac{1}{\sqrt{2}}\right)^n (-e^{j\pi n/4} + e^{-j\pi n/4})$$

$$= \left(\frac{1}{\sqrt{2}}\right)^n \underbrace{\left[\frac{e^{j\pi n/4} + e^{-j\pi n/4}}{2}\right]}_{\cos(\pi n/4)} + j\frac{3}{2}\left(\frac{1}{\sqrt{2}}\right)^n \left[\frac{-e^{j\pi n/4} + e^{-j\pi n/4}}{j2}\right] j2$$

$$= \left(\frac{1}{\sqrt{2}}\right)^n \cos(\pi n/4) + 3\left(\frac{1}{\sqrt{2}}\right)^n \underbrace{\left[\frac{e^{j\pi n/4} - e^{-j\pi n/4}}{j2}\right]}_{\sin(\pi n/4)}$$

To sum up,

$$h(n) = \begin{cases} \left(\frac{1}{\sqrt{2}}\right)^n [\cos(\pi n/4) + 3\sin(\pi n/4)], & n \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

Alternatively, since the two terms in

$$h(n) = A\frac{e^{j\pi n/4}}{(\sqrt{2})^n} + A^*\frac{e^{-j\pi n/4}}{(\sqrt{2})^n}, \qquad n \geq 0$$

are complex conjugates of each other we can write

$$h(n) = 2\,\mathrm{Re}\left(A\frac{e^{jn\pi/4}}{(\sqrt{2})^n}\right), \qquad n \geq 0$$

Alternative In the above solution the impulse response initially contains complex numbers; these have been algebraically manipulated into sine and cosine terms. A more direct way to obtain the impulse response in a form that contains no complex numbers is to use results #7 and #8 in "Transforms of some useful sequences" and manipulate the transform

$$H(z) = \frac{z(z+1)}{z^2 - z + 0.5} = \frac{z^2 + z}{z^2 - z + 0.5}$$

into those forms. Comparing the denominator of $H(z)$ with the denominator of the transforms of the sine and cosine functions

$$\mathcal{Z}\left\{a^n \sin \omega_0 n \; u(n)\right\} = \frac{az \sin \omega_0}{z^2 - 2az \cos \omega_0 + a^2}, \qquad \text{ROC } |z| > a$$

and

$$\mathcal{Z}\left\{a^n \cos \omega_0 n \; u(n)\right\} = \frac{z^2 - za \cos \omega_0}{z^2 - 2az \cos \omega_0 + a^2}, \qquad \text{ROC } |z| > a$$

we get

$$z^2 - z + 0.5 = z^2 - 2az \cos\omega_0 + a^2$$

from which

$$a^2 = 0.5 \rightarrow a = \frac{1}{\sqrt{2}} \qquad 2a \cos\omega_0 = 1 \rightarrow \omega_0 = \pi/4,$$

$$\cos \omega_0 = \frac{1}{\sqrt{2}}, \quad a \cos \omega_0 = ½, \sin \omega_0 = \frac{1}{\sqrt{2}} \qquad a \sin \omega_0 = ½$$

The numerators of the two transforms then are

$$az \sin\omega_0 = \frac{1}{\sqrt{2}} z \frac{1}{\sqrt{2}} = \frac{1}{2} z \quad \text{and} \quad z^2 - az \cos\omega_0 = z^2 - \frac{1}{\sqrt{2}} z = z^2 - \frac{1}{\sqrt{2}} z + \frac{1}{2}$$

In light of these we manipulate the numerator of $H(z)$ so that it will contain $\frac{1}{2} z$ and $z^2 - \frac{1}{2} z$

$$\text{Numerator} = z^2 + z = \left(z^2 - \frac{1}{2} z\right) + \left(\frac{3}{2} z\right)$$

$$\text{Denominator} = z^2 - z + 0.5 = z^2 - 2\left(\frac{1}{\sqrt{2}}\right)|z| + \left(\frac{1}{\sqrt{2}}\right)^2$$

Thus

$$H(z) = \frac{z^2 + z}{z^2 - z + 0.5} = \frac{\left(z^2 - \frac{1}{2} z\right) + \left(\frac{3}{2} z\right)}{z^2 - z + 0.5} = \frac{z^2 - \frac{1}{2} z}{z^2 - z + 0.5} + 3 \frac{\frac{1}{2} z}{z^2 - z + 0.5}$$

We have, in effect, arranged $H(z)$ as

$$H(z) = \frac{z^2 - \left(\frac{1}{\sqrt{2}}\right)|z| + \left(\frac{1}{\sqrt{2}}\right)^2}{z^2 - 2\left(\frac{1}{\sqrt{2}}\right)|z| + \left(\frac{1}{\sqrt{2}}\right)^2} + 3 \frac{\left(\frac{1}{2}\right)|z| + \left(\frac{1}{2}\right)}{z^2 - 2\left(\frac{1}{\sqrt{2}}\right)|z| + \left(\frac{1}{\sqrt{2}}\right)^2}$$

Therefore,

$$h(n) = \mathcal{Z}^{-1}\{H(z)\} = \mathcal{Z}^{-1}\left\{\frac{z^2 - \frac{1}{2} z}{z^2 - z + 0.5}\right\} + 3\,\mathcal{Z}^{-1}\left\{\frac{\frac{1}{2} z}{z^2 - z + 0.5}\right\}$$

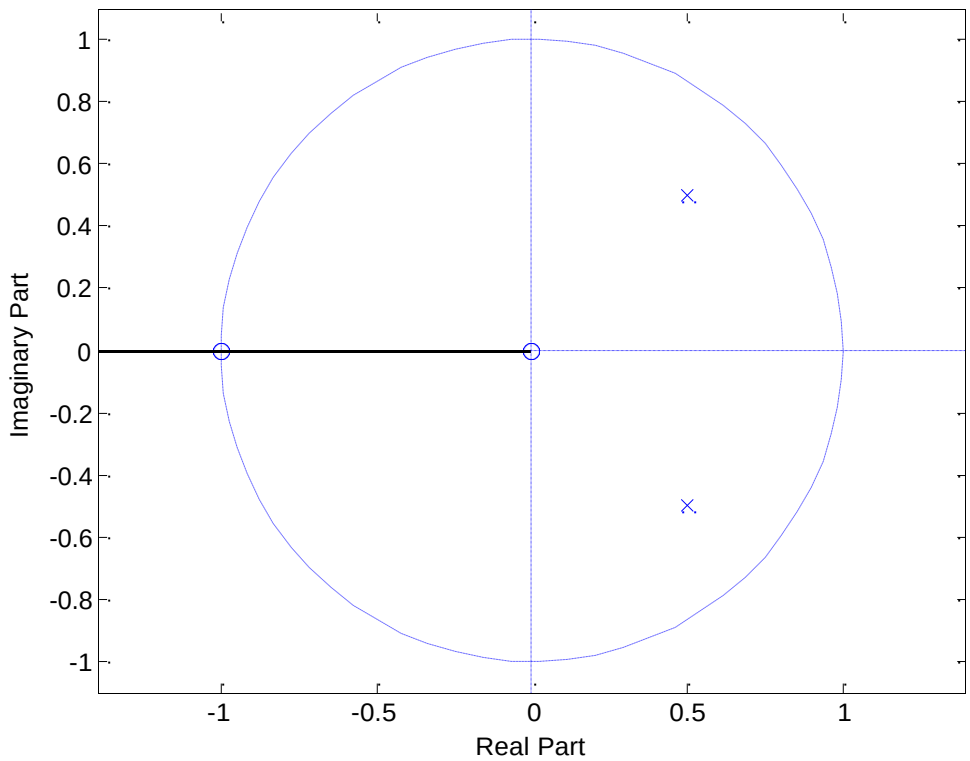
$$= \left(\frac{1}{\sqrt{2}}\right)^n \cos(n\pi/4)\,u(n) + 3\left(\frac{1}{\sqrt{2}}\right)^n \sin(n\pi/4)\,u(n)$$

In MATLAB (Pole-zero plot) It is convenient to specify the transfer function as a ratio of polynomials in z^{-1}

$$H(z) = \frac{1+z^{-1}}{1-z^{-1}+0.5z^{-2}}$$

The numerator coefficients, from left to right, are $\{b_i,\ i = 0 \text{ to } M\}$, specifying the vector $b = [b_0, b_1] = [1, 1]$. Similarly, the denominator coefficients are $\{a_i,\ i = 0 \text{ to } N\}$ from left to right, (with $a_0 = 1$) specifying the vector $a = [1, a_1, a_2] = [1, -1, 0.5]$.

```
%Pole-zero plot
b = [1, 1]; a = [1, -1, 0.5]; zplane (b, a)
```



In MATLAB (Partial fractions) The partial fractions may be computed by using the *residuez* function as below. Note that $H(z)$ is arranged as a ratio of polynomials in negative powers of z.

$$H(z) = \frac{1+z^{-1}}{1-z^{-1}+0.5z^{-2}} = K + \frac{R_1}{1-p_1^{-1}} + \frac{R_2}{1-p_2^{-1}}$$

We define the coefficient vectors $b = [1, 1]$ and $a = [1, -1, 0.5]$; $R = [R_1, R_2]$ represents the residues (partial fraction constants), $p = [p_1, p_2]$ the poles and K a constant.

```
%Partial fractions
b = [1, 1], a = [1, -1, 0.5],
```

```
[R, p, K] = residuez (b, a)
```

The MATLAB results returned are

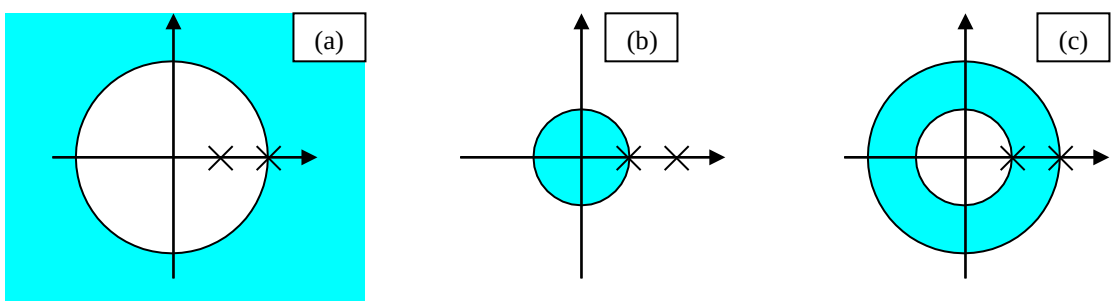
```
R =
  0.5 - 1.5i
  0.5 + 1.5i
p =
  0.5 + 0.5i
  0.5 - 0.5i
K =
   []
```

Therefore,

$$H(z) = \frac{1+z^{-1}}{1-z^{-1}+0.5z^{-2}} = 0 + \frac{0.5-j1.5}{1-(0.5+j0.5)z^{-1}} + \frac{0.5+j1.5}{1-(0.5-j0.5)z^{-1}}$$

Example 1.5.3 Find the inverse transform of $X(z) = \dfrac{z}{3z^2 - 4z + 1}$, where the ROC is (a) $|z| > 1$, (b) $|z| < (1/3)$, (c) $(1/3) < |z| < 1$.

Solution The three possible regions of convergence are shown below. The example shows that the inverse transform, $x(n)$, is unique only when the ROC is specified.



(a) For ROC ≡ $|z| > 1$

$$\frac{X(z)}{z} = \frac{1}{3\left(z^2 - \frac{4}{3}z + \frac{1}{3}\right)} = \frac{1}{3\left(z^2 - \frac{1}{3}z - \frac{1}{3}\right)} = \frac{1}{3z(z-1) - (z-1)\left(\frac{1}{3}\right)}$$

$$= \frac{1}{3(z-1)(z-(1/3))} = \frac{A}{z-1} + \frac{B}{z-(1/3)}$$

$$A = \left.\frac{1}{3(z-(1/3))}\right|_{z=1} = 1/2, \quad \text{and} \quad B = \left.\frac{1}{3(z-1)}\right|_{z=1/3} = -1/2$$

$$\frac{X(z)}{z} = \frac{(1/2)}{z-1} + \frac{(-1/2)}{z-(1/3)}$$

$$X(z) = \frac{(1/2)z}{z-1} + \frac{(-1/2)z}{z-(1/3)}$$

The inverse is

$$x(n) = \mathcal{Z}^{-1}\{X(z)\} = \mathcal{Z}^{-1}\left\{\frac{(1/2)z}{z-1} + \frac{(-1/2)z}{z-(1/3)}\right\} = \mathcal{Z}^{-1}\left\{\frac{(1/2)z}{z-1}\right\} + \mathcal{Z}^{-1}\left\{\frac{(-1/2)z}{z-(1/3)}\right\}$$

The ROC is outside the largest pole signifying a right-sided sequence for each pole. The inverse becomes

$$x(n) = \frac{1}{2}(1)^n u(n) - \frac{1}{2}\left(\frac{1}{3}\right)^n u(n) = u(n) - \frac{1}{2}\left(\frac{1}{3}\right)^n u(n)$$

(b) For ROC ≡ $|z| < (1/3)$. The partial fraction expansion does not change. Since the ROC is inward of the smallest pole, $x(n)$ consists of two negative-time sequences.

$$x(n) = \mathcal{Z}^{-1}\{X(z)\} = \mathcal{Z}^{-1}\left\{\frac{(1/2)z}{z-1}\right\} + \mathcal{Z}^{-1}\left\{\frac{(-1/2)z}{z-(1/3)}\right\}$$

$$= \frac{1}{2}(-1^n)\, u(-n-1) - \frac{1}{2}\left(-(1/3)^n\right) u(-n-1)$$

$$= (1/2)\left(-1 + (1/3)^n\right) u(-n-1)$$

(c) For ROC ≡ $(1/3) < |z| < 1$. The partial fraction expansion stays the same. The pole at $z = 1$ corresponds to a negative-time sequence (left-sided sequence) while the pole at $z = 1/3$ gives a positive-time sequence (right-sided sequence).

$$x(n) = \mathcal{Z}^{-1}\{X(z)\} = \mathcal{Z}^{-1}\left\{\frac{(1/2)z}{z-1} + \frac{(-1/2)z}{z-(1/3)}\right\} = \mathcal{Z}^{-1}\left\{\frac{(1/2)z}{z-1}\right\} + \mathcal{Z}^{-1}\left\{\frac{(-1/2)z}{z-(1/3)}\right\}$$

$$= \frac{1}{2}(-1^n)\, u(-n-1) - \frac{1}{2}\left(\frac{1}{3}\right)^n u(n) = -\frac{1}{2}\, u(-n-1) - \frac{1}{2}\left(\frac{1}{3}\right)^n u(n)$$

The overall result is a two-sided sequence.

Example 1.5.4 (Sometimes there is no z in the numerator to factor out, but we still can divide $X(z)$ by z as in this example.) Find $x(n)$ for $X(z) = \dfrac{z+1}{3z^2 - 4z + 1}$ where the ROC is $|z| > 1$.

Solution

$$\frac{X(z)}{z} = \frac{z+1}{z(3z^2 - 4z + 1)} = \frac{z+1}{3z(z-1)(z-(1/3))} = \frac{A}{z} + \frac{B}{z-1} + \frac{C}{z-(1/3)}$$

$$A = \left.\frac{z+1}{3(z-1)(z-(1/3))}\right|_{z=0} = \frac{1}{3(-1)(-1/3)} = 1$$

$$B = \left.\frac{z+1}{3z(z-(1/3))}\right|_{z=1} = \ldots = 1$$

$$C = \left.\frac{z+1}{3z(z-1)}\right|_{z=1/3} = \ldots = -2$$

$$\frac{X(z)}{z} = \frac{1}{z} + \frac{1}{z-1} - \frac{2}{z-(1/3)}$$

$$X(z) = 1 + \frac{z}{z-1} - 2\frac{z}{z-(1/3)}$$

$$x(n) = \delta(n) + u(n) - 2\left(\frac{1}{3}\right)^n u(n)$$

Example 4.5.5 [2002] Find the inverse z-transform of $X(z) = \dfrac{(z-1)^2}{z^2 - 0.1z - 0.56}$.

Solution The roots of the quadratic in the denominator are given by

$$z_1, z_2 = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{-(-0.1) \pm \sqrt{(-0.1)^2 - 4\,(1)(-0.56)}}{2\,(1)}$$

$$= \frac{0.1 \pm \sqrt{0.01 + 2.24}}{2} = \frac{0.1 \pm \sqrt{2.25}}{2} = \frac{0.1 \pm 1.5}{2} = 0.8 \text{ and } -0.7$$

$$X(z) = \frac{(z-1)^2}{(z - 0.8)(z + 0.7)}$$

$$\frac{X(z)}{z} = \frac{(z-1)^2}{z\,(z - 0.8)\,(z + 0.7)} = \frac{A}{z} + \frac{B}{z - 0.8} + \frac{C}{z + 0.7}$$

$$A = \left.\frac{(z-1)^2}{(z - 0.8)\,(z + 0.7)}\right|_{z=0} = \ldots = -1/0.56 = -1.79$$

$$B = \left.\frac{(z-1)^2}{z\,(z + 0.7)}\right|_{z=0.8} = \ldots = 1/30$$

$$C = \left.\frac{(z-1)^2}{z\,(z - 0.8)}\right|_{z=-0.7} = \ldots = 2.75$$

$$\frac{X(z)}{z} = \frac{-1.79}{z} + \frac{(1/30)}{z - 0.8} + \frac{2.75}{z + 0.7}$$

$$X(z) = -1.79 + \frac{(1/30)\,z}{z - 0.8} + \frac{2.75\,z}{z + 0.7}$$

$$x(n) = \mathcal{Z}^{-1}\{X(z)\} = 1.79\,\delta(n) + \frac{1}{30}(0.8)^n\,u(n) + 2.75\,(-0.7)^n\,u(n)$$

Example 1.5.6 Find the inverse z-transform of $X(z) = \dfrac{1}{[z - (1/2)][z - (1/4)]}$, $|z| > ½$.

Solution

$$\frac{X(z)}{z} = \frac{1}{z[z - (1/2)][z - (1/4)]} = \frac{A}{z} + \frac{B}{z - (1/2)} + \frac{C}{z - (1/4)}$$

$$A = \left.\frac{1}{(z - (1/2))(z - (1/4))}\right|_{z=0} = \ldots = 8$$

$$B = \left.\frac{1}{z(z - (1/4))}\right|_{z=1/2} = \ldots = 8$$

$$C = \left.\frac{1}{z(z - (1/2))}\right|_{z=1/4} = \ldots = -16$$

$$\frac{X(z)}{z} = \frac{8}{z} + \frac{8}{z - (1/2)} - \frac{16}{z - (1/4)}$$

$$X(z) = 8 + \frac{8\,z}{z - (1/2)} - \frac{16\,z}{z - (1/4)}$$

$$x(n) = 8\,\delta(n) + 8\left(\frac{1}{2}\right)^n u(n) - 16\left(\frac{1}{4}\right)^n u(n)$$

Example 1.5.7 Find the inverse of $X(z) = \dfrac{z^4 + z^2}{[z-(1/2)][z-(1/4)]}$ for ROC ½ < |z| < ∞

$$\frac{X(z)}{z} = \frac{z^3 + z}{[z-(1/2)][z-(1/4)]} = \frac{z^3 + z}{z^2 - (3/4)z + (1/8)}$$

There is a pole at $z = \infty$. The numerator degree is 3 and is greater than the denominator degree. By long division we reduce the numerator degree by 2 so that the resulting numerator degree is less than that of the denominator degree.

$$\frac{X(z)}{z} = \frac{z^3 + z}{z^2 - (3/4)z + (1/8)} = z + \frac{3}{4} + \left(\frac{(23/16)z - (3/32)}{z^2 - (3/4)z + (1/8)}\right)$$

(Note that in the long division leading to the above result the numerator and denominator polynomials are arranged in the order of decreasing powers of z. There are three other ways (all of them wrong) of arranging the two polynomials for the long division.)

The proper fraction part can now be expanded into partial fractions:

$$\frac{(23/16)z - (3/32)}{z^2 - (3/4)z + (1/8)} = \frac{A}{(z-(1/2))} + \frac{B}{z-(1/4)}$$

$$A = \left.\frac{(23/16)z - (3/32)}{z-(1/4)}\right|_{z=1/2} = 5/2$$

$$B = \left.\frac{(23/16)z - (3/32)}{z-(1/2)}\right|_{z=1/4} = -17/16$$

$$\frac{X(z)}{z} = z + \frac{3}{4} + \frac{(5/2)}{(z-(1/2))} + \frac{(-17/16)}{z-(1/4)}$$

$$X(z) = z^2 + \frac{3z}{4} + \frac{(5/2)z}{(z-(1/2))} + \frac{(-17/16)z}{z-(1/4)}$$

$$x(n) = \delta(n+2) + \frac{3}{4}\delta(n+1) + \left\{\frac{5}{2}\left(\frac{1}{2}\right)^n - \frac{17}{16}\left(\frac{1}{4}\right)^n\right\} u(n)$$

The answer has values for $n = -1$ and $n = -2$ due to the pole at $z = \infty$. The resulting $x(n)$ is not a causal sequence.

In MATLAB (Partial fractions) The transform $X(z)$ represents a noncausal sequence.

$$X(z) = \frac{z^4 + z^2}{z^2 - (3/4)z + (1/8)} = \frac{1 + z^{-2}}{z^{-2} - (3/4)z^{-3} + (1/8)z^{-4}}$$

Partial fractions cannot be computed by using the *residuez* function directly on $X(z)$ since $a_0 = 0$. However,

$$X(z) = z^2 + \frac{3}{4}z + \left(\frac{(23/16)z^2 - (3/32)z}{z^2 - (3/4)z + (1/8)}\right) = z^2 + \frac{3}{4}z + X_1(z)$$

where

$$X_1(z) = \left(\frac{(23/16)z^2 - (3/32)z}{z^2 - (3/4)z + (1/8)}\right) = \frac{(23/16) - (3/32)z^{-1}}{1 - (3/4)z^{-1} + (1/8)z^{-2}}$$

On which we may use the *residuez* function.

Example 4.5.8 Assuming that $H(z) = \dfrac{z+4}{z-5}$ is a causal system function, prove the following independently of each other

(a) $h(n) = -(4/5)\,\delta(n) + (9/5)\,5^n u(n)$

(b) $h(n) = 5^n u(n) + (4)\,5^{n-1} u(n-1)$

(a) $h(n) = \delta(n) + (9)\,5^{n-1} u(n-1)$

Solution (a) $\dfrac{H(z)}{z} = \dfrac{(-4/5)}{z} + \dfrac{(9/5)}{z-5}$, **(b)** $H(z) = \dfrac{z}{z-5} + \dfrac{4}{z-5}$, **(c)** By long division $H(z) = 1 + \dfrac{9}{z-5}$.

Example 4.5.9 Partial fractions can be obtained with the *z*-transform, say *H(z)*, expressed as a ratio of polynomials in negative powers of *z*. This amounts to expanding *H(z)/z* into partial fractions. Here is an example:

$$H(z) = \frac{8z^3 - 4z^2 + 11z - 2}{\left(z - (1/4)\right)\left(z^2 - z + (1/2)\right)}$$

This example is from "Parallel realization of IIR filters", towards the end of this Unit where we obtain

$$\frac{H(z)}{z} = \frac{16}{z} + \frac{8}{\left(z - (1/4)\right)} + \frac{(-16)\,z + 20}{\left(z^2 - z + (1/2)\right)}$$

However, we may also proceed with negative powers of *z* as below (we may view $z^{-1} = p$ as a new variable):

$$H(z) = \frac{8z^3 - 4z^2 + 11z - 2}{\left(z - (1/4)\right)\left(z^2 - z + (1/2)\right)} = \frac{8 - 4z^{-1} + 11z^{-2} - 2z^{-3}}{\left(1 - 0.25z^{-1}\right)\left(1 - z^{-1} + 0.5z^{-2}\right)}$$

$$= \frac{8 - 4z^{-1} + 11z^{-2} - 2z^{-3}}{1 - 1.25z^{-1} + 0.75z^{-2} - 0.125z^{-3}}$$

By long division we reduce the degree of the numerator by 1 and then expand the proper fraction part into partial fractions:

Long Division

		16	←Quotient
Denominator→	$-0.125z^{-3} + 0.75z^{-2} - 1.25z^{-1} + 1$	$-2z^{-3} + 11z^{-2} - 4z^{-1} + 8$	←Numerator
		$-2z^{-3} + 12z^{-2} - 20z^{-1} +$	
		16	←Remainder
		$-z^{-2} \quad + 16z^{-1} \; -8$	

$$H(z) = 16 + \frac{-8 + 16z^{-1} - z^{-2}}{1 - 1.25z^{-1} + 0.75z^{-2} - 0.125z^{-3}}$$

Let

$$\frac{-8 + 16z^{-1} - z^{-2}}{1 - 1.25z^{-1} + 0.75z^{-2} - 0.125z^{-3}} = \frac{A}{\left(1 - 0.25z^{-1}\right)} + \frac{Bz^{-1} + C}{\left(1 - z^{-1} + 0.5z^{-2}\right)}$$

Comparing coefficients of like powers of z in the numerators on both sides

z^0: $\quad A + C = -8$

z^1: $\quad -A + B - 0.25C = 16$

z^3: $\quad 0.5A - 0.25B = -1$

which give $A = 8$, $B = 20$, and $C = -16$, and

$$H(z) = 16 + \frac{8}{\left(-0.25z^{-1}\right)} + \frac{-16 + 20z^{-1}}{\left(1 - z^{-1} + 0.5z^{-2}\right)}$$

In MATLAB The partial fractions may be computed by using the *residuez* function:

$$H(z) = \frac{8 - 4z^{-1} + 11z^{-2} - 2z^{-3}}{1 - 1.25z^{-1} + 0.75z^{-2} - 0.125z^{-3}} = K + \frac{R_1}{1 - p_1^{-1}} + \frac{R_2}{1 - p_2^{-1}} + \frac{R_3}{1 - p_3^{-1}}$$

We define the coefficient vectors b = [8, -4, 11, -2] and a = [1, -1.25, 0.75, -0.125]; R represents the residues (partial fraction constants), p the poles and K a constant. Note that in the numerator z^{-3} means $M = 3$, and in the denominator z^{-3} means $N = 3$; since M is not less than N this is not a proper rational function, so that K will have a nonzero element(s).

```
%Partial fractions
b = [8, -4, 11, -2], a = [1, -1.25, 0.75, -0.125],
[R, p, K] = residuez (b, a)
```

The MATLAB results returned are

```
 R =
  -8 -12i
  -8 +12i
   8
p =
   0.5 + 0.5i
   0.5 - 0.5i
   0.25
K =
    16
```

Therefore,

$$H(z) = 16 + \frac{-8 - j12}{1 - (0.5 + j50)\ z^{-1}} + \frac{-8 + j12}{1 - (0.5 - j50)\ z^{-1}} + \frac{8}{1 - 0.25\ z^{-1}}$$

Inverse *z*-transform when there are repeated roots With repeated roots, that is, a *k*-th order pole at $z = a$ we have $X(z)$ in the form

$$X(z) = \frac{z}{(z-a)^k}, \qquad \text{ROC} \quad |z| > |a|$$

The table below gives the inverse *z*-transforms for several values of *k* and for the general case of arbitrary *k*.

Repeated Roots

$X(z)$	$x(n) = \mathcal{Z}^{-1}[X(z)]$ for ROC $\lvert z\rvert > \lvert a\rvert$
$\frac{z}{(z-a)}$	$a^n u(n)$
$\frac{z}{(z-a)^2}$	$\frac{n}{1!}a^{n-1}u(n)$
$\frac{z}{(z-a)^3}$	$\frac{n(n-1)}{2!}a^{n-2}u(n)$
$\frac{z}{(z-a)^4}$	$\frac{n(n-1)(n-2)}{3!}a^{n-3}u(n)$
…	…
$\frac{z}{(z-a)^k}$	$\frac{n(n-1)(n-2)\ldots(n-k-2)}{(k-1)!}a^{n-k-1}u(n)$

General procedure for partial fraction expansion Since $X(z)/z$ must be rational, it takes the form

$$\frac{X(z)}{z} = \frac{\beta_K z^K + \beta_{K-1} z^{K-1} + \ldots\ldots\ldots + \beta_1 z + \beta_0}{\alpha_L z^L + \alpha_{L-1} z^{L-1} + \square + \alpha_1 z + \alpha_0}$$

If $K < L$ then no adjustment is needed. The partial fraction expansion is straightforward.

If $K \geq L$ then divide until the remainder polynomial in *z* has a degree of $L-1$ or less:

$$\frac{X(z)}{z} = (c_{K-L} z^{K-L} + \ldots + c_1 z + c_0) + \frac{d_{L-1} z^{L-1} + \ldots\ldots\ldots + d_1 z + d_0}{\alpha_L z^L + \alpha_{L-1} z^{L-1} + \square + \alpha_1 z + \alpha_0}$$

The first part of the above expression, $(c_{K-L} z^{K-L} + \ldots + c_1 z + c_0)$, will eventually contribute δ functions to the output sequence some of which are time-advanced so that the resulting $x(n)$ will be noncausal. The second part – the proper fraction – is expanded into partial fractions. Assume we have *one repeated pole* of order *m*, call it z_1, and that all the rest are distinct, call them $z_{m+1}, z_{m+2}, \ldots, z_L$. Then let

$$\Psi(z) = \frac{d_{L-1} z^{L-1} + \ldots\ldots\ldots + d_1 z + d_0}{\alpha_L z^L + \alpha_{L-1} z^{L-1} + \square + \alpha_1 z + \alpha_0}$$

$$= \frac{A_m}{(z-z_1)^m} + \frac{A_{m-1}}{(z-z_1)^{m-1}} + \ldots + \frac{A_1}{(z-z_1)} + \sum_{j=m+1}^{L} \frac{B_j}{(z-z_j)}$$

The coefficients A_j (*m* of them) and B_j ($L-m$ of them) are found as follows:

$$A_j = \frac{1}{(m-j)!}\left\{\frac{d^{m-j}}{dz^{m-j}}\left[(z-z_1)^m \Psi(z)\right]\right\}\Bigg|_{z=z_1}, \quad j = 1, 2, \ldots, m$$

$$B_j = [(z - z_j)\Psi(z)]\Big|_{z=z_j}, \qquad j = m+1, m+2, \ldots, L$$

In the resulting $x(n)$ the contribution of the A_j terms is a number of exponentials multiplied by n, $(n-1)$, $(n-2)$, etc., and the contribution of the B_j terms is a number of complex exponentials.

Example 1.5.10 Find the inverse of

$$H(z) = \frac{z}{z^3 + 2z^2 + 1.25z + 0.25} = \frac{z^{-2}}{1 + 2z^{-1} + 1.25z^{-2} + 0.25z^{-3}}, \quad |z| > 1$$

Solution This transform is a proper rational function. We shall use this example to give a summary of the three styles of obtaining partial fractions: (1) Expanding $H(z)$ directly, (2) Expanding $H(z)/z$ and (3) Expanding $H(z^{-1})$ as in MATLAB (also Mitra).

When the poles of $H(z)$ are distinct the partial fraction coefficients returned by the MATLAB function ***residuez*** are the same as in expanding $H(z)/z$. However, when there are repeated poles it makes a difference in the coefficients as well as in the final analytical forms of the inverse transforms in these two methods. In addition, directly expanding $H(z)$ results in an analytical form that is still different from the other two. In any event the three inverse transforms are the same as far as the actual sequence values are concerned.

(1) Expanding *H(z)* We have

$$H(z) = \frac{z}{z^3 + 2z^2 + 1.25z + 0.25} = \frac{z}{(z+1)(z+0.5)^2} = \frac{A}{(z+1)} + \frac{B_2}{(z+0.5)^2} + \frac{B_1}{(z+0.5)}$$

There is a pole at $z = -1$ and a repeated pole at $z = -0.5$. The coefficients A, B_2 and B_1 are given by

$$A = \frac{z}{(z+0.5)^2}\Big|_{z=-1} = \frac{-1}{(-1+0.5)^2} = -4$$

$$B_2 = \frac{z}{(z+1)}\Big|_{z=-0.5} = \frac{-0.5}{(-0.5+1)} = -1$$

$$B_1 = \left\{\frac{d}{dz}\left(\frac{z}{z+1}\right)\right\}\Big|_{z=-0.5} = \left\{\frac{(z+1).1 - z(1)}{(z+1)^2}\right\}\Big|_{z=-0.5} = \frac{(-0.5+1)-(-0.5)}{(-0.5+1)^2} = 4$$

Thus

$$H(z) = \frac{(-4)}{(z+1)} + \frac{(-1)}{(z+0.5)^2} + \frac{4}{(z+0.5)}$$

Taking the inverse z-transform,

$$h(n) = -4\,\mathcal{Z}^{-1}\left\{\frac{1}{z+1}\right\} - 1\mathcal{Z}^{-1}\left\{\frac{1}{(z+0.5)^2}\right\} + 4\,\mathcal{Z}^{-1}\left\{\frac{1}{z+0.5}\right\}$$

$$= -4\,\mathcal{Z}^{-1}\left\{z^{-1}\frac{z}{z+1}\right\} - 1\mathcal{Z}^{-1}\left\{z^{-1}\frac{z}{(z+0.5)^2}\right\} + 4\mathcal{Z}^{-1}\left\{z^{-1}\frac{z}{z+0.5}\right\}$$

$$= -4\left((-1)^n u(n)\right)\Big|_{n\to n-1} - 1\left(\frac{n}{1!}(-0.5)^{n-1}u(n)\right)\Big|_{n\to n-1} + 4\left((-0.5)^n u(n)\right)\Big|_{n\to n-1}$$

$$= 4\,(-1)^n u(n-1) - 4\,(n-1)(-0.5)^n u(n-1) - 8\,(-0.5)^n u(n-1)$$

(2) Expanding $H(z)/z$ We have

$$\frac{H(z)}{z} = \frac{1}{z^3 + 2z^2 + 1.25z + 0.25} = \frac{z}{(z+1)(z+0.5)^2} = \frac{C}{(z+1)} + \frac{D_2}{(z+0.5)^2} + \frac{D}{(z+0.5)}$$

The coefficients C, D_2 and D_1 are given by

$$C = \left.\frac{1}{(z+0.5)^2}\right|_{z=-1} = \frac{1}{(-1+0.5)^2} = 4$$

$$D_2 = \left.\frac{1}{(z+1)}\right|_{z=-0.5} = \frac{1}{(-0.5+1)} = 2$$

$$D_1 = \left.\left\{\frac{d}{dz}\left(\frac{1}{z+1}\right)\right\}\right|_{z=-0.5} = \left.\left\{\frac{(z+1).0 - 1(1)}{(z+1)^2}\right\}\right|_{z=-0.5} = \frac{-1}{(-0.5+1)^2} = -4$$

Thus

$$\frac{H(z)}{z} = \frac{4}{(z+1)} + \frac{2}{(z+0.5)^2} + \frac{(-4)}{(z+0.5)}$$

$$H(z) = 4\frac{z}{(z+1)} + 2\frac{z}{(z+0.5)^2} - 4\frac{z}{(z+0.5)}$$

Taking the inverse z-transform,

$$h(n) = 4\,\mathcal{Z}^{-1}\left\{\frac{z}{z+1}\right\} + 2\,\mathcal{Z}^{-1}\left\{\frac{z}{(z+0.5)^2}\right\} - 4\,\mathcal{Z}^{-1}\left\{\frac{z}{z+0.5}\right\}$$

$$= 4\,(-1)^n u(n) + 2\,\frac{n}{1!}(-0.5)^{n-1}\,u(n) - 4\,(-0.5)^n u(n)$$

$$= 4\,(-1)^n u(n) - 4\,n(-0.5)^n u(n) - 4\,(-0.5)^n u(n)$$

(3) Expanding $H(z^{-1})$ as in MATLAB We start with $H(z)$ expressed as a ratio of polynomials in negative powers of z. However, for the sake of continuity we have

$$H(z) = \frac{z^{-2}}{z^3 + 2z^2 + 1.25z + 0.25} = \frac{z^{-2}}{1 + 2z^{-1} + 1.25z^{-2} + 0.25z^{-3}}$$

$$H(z) = \frac{z^{-2}}{(z+1)(z+0.5)} = \frac{z^{-2}}{(1+1z^{-1})(1+0.5z^{-1})^2}$$

$$= \frac{E}{(1+1z^{-1})} + \frac{F_1}{(1+0.5z^{-1})} + \frac{F_2}{(1+0.5z^{-1})^2}$$

(We have ordered the coefficients in the order in which MATLAB displays them). We can define $z^{-1} = v$ so that $z = -1$ corresponds to $v = -1$ and $z = -0.5$ to $v = -2$. The transform now appears as

$$H(v) = \frac{v^2}{(1+1v)(1+0.5v)^2} = \frac{E}{(1+1v)} + \frac{F_1}{(1+0.5v)} + \frac{F_2}{(1+0.5v)^2}$$

The coefficients E, F_2 and F_1 are given by

$$E = \left.\frac{z^{-2}}{(1+0.5z^{-1})^2}\right|_{z^{-1}=-1} = \frac{1^2}{(1+0.5(-1))^2} = 4$$

$$F_2 = \left.\frac{z^{-2}}{(1+1z^{-1})}\right|_{z^{-1}=-2} = \frac{(-2)^2}{(1+1.(-2))} = -4$$

$$F_1 = \left\{ \frac{d}{dv}\left(\frac{v^2}{1+1v} \right) \right\}\Bigg|_{v=-2} = \left\{ \frac{(1+v).2v - v^2.(1)}{(1+v)^2} \right\}\Bigg|_{v=-2} = \frac{(1-2)2(-2)-(-2)^2}{(1-2)^2} = 0$$

Therefore,

$$H(z) = \frac{4}{(1+1z^{-1})} + \frac{0}{(1+0.5z^{-1})} + \frac{(-4)}{(1+0.5z^{-1})^2}$$

$$= 4\frac{z}{z+1} - 4\frac{z^2}{(z+0.5)^2} = 4\frac{z}{z+1} - 4z\frac{z}{(z+0.5)^2}$$

Taking the inverse z-transform,

$$h(n) = 4\,\mathcal{Z}^{-1}\left\{ \frac{z}{z+1} \right\} - 4\mathcal{Z}^{-1}\left\{ \frac{z}{(z+0.5)^2} \right\}$$

$$= 4\,(-1)^n u(n) - 4\left\{ \frac{n}{1!}(-0.5)^{n-1} u(n) \right\}\Bigg|_{n \to n+1}$$

$$= 4\,(-1)^n u(n) - 4\,(-0.5)^{-1}\left\{ \frac{n}{1!}(-0.5)^n u(n) \right\}\Bigg|_{n \to n+1}$$

$$= 4\,(-1)^n u(n) + 8\,(n+1)(-0.5)^{n+1} u(n+1)$$

In MATLAB This particular set of partial fractions may be computed by using the *residuez* function:

$$H(z) = \frac{z^{-2}}{1+2z^{-1}+1.25z^{-2}+0.25z^{-3}} = K + \frac{E}{1-zp_1^{-1}} + \frac{F_1}{1-p_2^{-1}} + \frac{F_2}{(1-p_2 z^{-1})^2}$$

We define the coefficient vectors b = [0, 0, 1] and a = [1, 2, 1.25, 0.25]; R = [E, F_1, F_2] represents the residues (keyed to the above partial fraction coefficients), p the poles and K a constant.

```
%Partial fractions
b = [0, 0, 1], a = [1, 2, 1.25, 0.25],
[R, p, K] = residuez (b, a)
```

The MATLAB results returned are

```
R =
   4
  -0 + 0i
  -4 - 0i
p =
  -1
  -0.5 + 0i
  -0.5 - 0i
K =
     []
```

Therefore,

$$H(z) = \frac{z^{-2}}{1+2z^{-1}+1.25z^{-2}+0.25z^{-3}} = 0 + \frac{4}{1-(-1)z^{-1}} + \frac{0}{1-(-0.5)z^{-1}} + \frac{(-4)}{(1-(-0.5)z^{-1})^2}$$

$$= \frac{4}{1+1z^{-1}} + \frac{(-4)}{\left(1+0.5z^{-1}\right)^2}$$

which agrees with the hand-calculated results.

Example 1.5.11 Find the inverse of

$$X(z) = \frac{z^3 - z^2 + z - \frac{1}{16}}{\left(z-\frac{1}{2}\right)^2\left(z-\frac{1}{4}\right)} = \frac{z^3 - z^2 + z - \frac{1}{16}}{z^3 - (5/4)z^2 + (1/2)z - (1/16)}, \text{ for } |z| > ½$$

$$\frac{X(z)}{z} = \frac{z^3 - z^2 + z - (1/16)}{z\left[z-(1/2)\right]^2\left[z-(1/4)\right]} = \frac{A}{z} + \frac{B_2}{\left(z-(1/2)\right)^2} + \frac{B_1}{\left(z-(1/2)\right)} + \frac{C}{z-(1/4)}$$

$$A = \left.\frac{z^3 - z^2 + z - (1/16)}{\left[z-(1/2)\right]^2\left[z-(1/4)\right]}\right|_{z=0} = \frac{-(1/16)}{\left[-(1/2)\right]^2\left[-(1/4)\right]} = 1$$

$$C = \left.\frac{z^3 - z^2 + z - (1/16)}{z\left[z-(1/2)\right]^2}\right|_{z=1/4} = \frac{(1/4)^3 - (1/4)^2 + (1/4) - (1/16)}{(1/4)\left[(1/4)-(1/2)^2\right]} = 9$$

$$B_2 = \left.\frac{z^3 - z^2 + z - (1/16)}{z\left[z-(1/4)\right]}\right|_{z=1/2} = \frac{(1/2)^3 - (1/2)^2 + (1/2) - (1/16)}{(1/2)\left[(1/2)-(1/4)\right]} = 5/2$$

$$B_1 = \left.\left\{\frac{d}{dz}\left(\frac{z^3 - z^2 + z - (1/16)}{z(z-1/4)}\right)\right\}\right|_{z=1/2} = \left.\left\{\frac{d}{dz}\left(\frac{z^3 - z^2 + z - (1/16)}{z^2 - (z/4)}\right)\right\}\right|_{z=1/2}$$

$$= \left.\left\{\left(\frac{\left[z^2-(z/4)\right](3z^2-2z+1) - \left[z^3-z^2+z-(1/16)\right]\left[2z-(1/4)\right]}{\left[z^2-(z/4)\right]^2}\right)\right\}\right|_{z=1/2}$$

$$= -9$$

$$\frac{X(z)}{z} = \frac{1}{z} + \frac{(5/2)}{\left(z-(1/2)\right)^2} + \frac{(-9)}{\left(z-(1/2)\right)} + \frac{9}{z-(1/4)}$$

$$X(z) = 1 + \frac{5}{2}\frac{z}{\left(z-(1/2)\right)^2} - 9\frac{z}{\left(z-(1/2)\right)} + 9\frac{z}{z-(1/4)}$$

Taking the inverse z-transform,

$$x(n) = \mathcal{Z}^{-1}\{X(z)\}$$

$$= \mathcal{Z}^{-1}\{1\} + (5/2)\mathcal{Z}^{-1}\left\{\frac{z}{\left(z-(1/2)\right)^2}\right\} - 9\mathcal{Z}^{-1}\left\{\frac{z}{\left(z-(1/2)\right)}\right\} + 9\mathcal{Z}^{-1}\left\{\frac{z}{z-(1/4)}\right\}$$

$$= \delta(n) + \frac{5}{2}n\left(\frac{1}{2}\right)^{n-1}u(n) - 9\left(\frac{1}{2}\right)^n u(n) + 9\left(\frac{1}{4}\right)^n u(n)$$

Other possibilities If we choose to expand *X(z)*, rather than $X(z)/z$, into partial fractions, we need to perform long division to reduce the degree of the numerator by 1 resulting in

$$X(z) = 1 + \frac{(z^2/4) + (z/2)}{z^3 - (5/4)z^2 + (1/2)z - (1/16)} = 1 + X_1(z)$$

where $X_1(z)$ is the proper fraction part of the above

$$X_1(z) = \frac{(z^2/4) + (z/2)}{z^3 - (5/4)z^2 + (1/2)z - (1/16)}$$

Either $X_1(z)$ itself or $X_1(z)/z$ may now be expanded into partial fractions.

In MATLAB The partial fractions may be computed by using the *residuez* function:

$$X(z) = \frac{1 - z^{-1} + z^{-2} - (1/16)z^{-3}}{1 - (5/4)z^{-1} + (1/2)z^{-2} - (1/16)z^{-3}} = K + \frac{R_1}{1 - p_1 z^{-1}} + \frac{R_2}{1 - p_2 z^{-1}} + \frac{R_3}{\left(1 - p_2 z^{-1}\right)^2}$$

We define the coefficient vectors b = [1, -1, 1, -1/16] and a = [1, -5/4, 1/2, -1/16]; R represents the residues (partial fraction coefficients), p the poles and K a constant. Note that in the numerator z^{-3} means $M = 3$, and in the denominator z^{-3} means $N = 3$; since M is not less than N this is not a proper rational function, so that K will have a nonzero element(s).

```
%Partial fractions
b = [1, -1, 1, -1/16], a = [1, -5/4, 1/2, -1/16],
[R, p, K] = residuez (b, a)
```

The MATLAB results returned are

```
 R =
  -14.0000
    5.0000
    9.0000
p =
    0.5000
    0.5000
    0.2500
K =
     1
```

Therefore,

$$X(z) = \frac{1 - z^{-1} + z^{-2} - (1/16)z^{-3}}{1 - (5/4)z^{-1} + (1/2)z^{-2} - (1/16)z^{-3}} = 1 + \frac{9}{1 - 0.25z^{-1}} + \frac{(-14)}{1 - 0.5z^{-1}} + \frac{5}{\left(1 - 0.5z^{-1}\right)^2}$$

Example 1.5.12 Find the inverse of $X(z) = \dfrac{z^2 + z}{\left[z - (1/2)\right]^3 \left[z - (1/4)\right]}$ for ROC $|z| > 1/2$

$$\frac{X(z)}{z} = \frac{z + 1}{\left[z - (1/2)\right]^3 \left[z - (1/4)\right]} = \frac{A_3}{\left(z - (1/2)\right)^3} + \frac{A_2}{\left(z - (1/2)\right)^2} + \frac{A_1}{\left(z - (1/2)\right)} + \frac{B}{z - (1/4)}$$

$$A_3 = \left.\frac{z + 1}{\left[z - (1/4)\right]}\right|_{z = 1/2} = \frac{(1/2) + 1}{\left[(1/2) - (1/4)\right]} = 6$$

$$A_2 = \frac{1}{(3-2)!}\left\{\frac{d}{dz}\left(\frac{z+1}{\left[z - (1/4)\right]}\right)\right\}\Bigg|_{z=1/2} = \ldots = -20$$

$$A_1 = \frac{1}{(3-1)!}\left\{\frac{d^2}{dz^2}\left(\frac{z+1}{\left[z - (1/4)\right]}\right)\right\}\Bigg|_{z = 1/2} = \ldots = 80$$

$$B = \left.\frac{z + 1}{\left[z - (1/2)\right]^3}\right|_{z = 1/4} = \ldots = -80$$

$$\frac{X(z)}{z} = \frac{6}{\left(z - (1/2)\right)^3} + \frac{(-20)}{\left(z - (1/2)\right)^2} + \frac{80}{\left(z - (1/2)\right)} + \frac{(-80)}{z - (1/4)}$$

$$X(z) = 6\frac{z}{(z-(1/2))^3} \quad 20\frac{z}{(z-(1/2))^2} + 80\frac{z}{(z-(1/2))} \quad 80\frac{z}{z-(1/4)}$$

$$x(n) = \mathcal{Z}^{-1}\{X(z)\}$$

$$= 6\mathcal{Z}^{-1}\left\{\frac{z}{(z-(1/2))^3}\right\} - 20\mathcal{Z}^{-1}\left\{\frac{z}{(z-(1/2))^2}\right\} + 80\mathcal{Z}^{-1}\left\{\frac{z}{(z-(1/2))}\right\} - 80\mathcal{Z}^{-1}\left\{\frac{z}{z-(1/4)}\right\}$$

$$= 6\frac{n(n-1)}{2!}\left(\frac{1}{2}\right)^{n-2}u(n) - 20\,n\left(\frac{1}{2}\right)^{n-1}u(n) + 80\left(\frac{1}{2}\right)^{n}u(n) - 80\left(\frac{1}{4}\right)^{n}u(n)$$

The *u(n)* may be factored out etc.

MATLAB

$$X(z) = \frac{z^{-2} + z^{-3}}{1-(7/4)z^{-1} + (9/8)z^{-2} - (5/16)z^{-3} + (1/32)z^{-4}}$$

```
%Partial fractions
b = [0, 0, 1, 1], a = [1, -7/4, 9/8, -5/16, 1/32],
[R, p, K] = residuez (b, a)
```

The MATLAB results returned are

```
 R =
  1.0e+002 *

   1.44 + 0.0i
  -0.88 - 0.0i
   0.24
  -0.80
p =
   0.5 + 0.0i
   0.5 - 0.0i
   0.5
   0.25
K =
     []
```

$$X(z) = \frac{144}{(1-(1/2)z^{-1})} + \frac{(-88)}{(1-(1/2)z^{-1})^2} + \frac{24}{(1-(1/2)z^{-1})^3} + \frac{(-80)}{1-(1/4)z^{-1}}$$

Relationships among system representations

A discrete-time linear shift-invariant system can be characterized by its unit sample response, a difference equation, a system function, or a frequency response. Assume that a system is described by the linear constant coefficient difference equation

$$\sum_{k=0}^{N} a_k y(n-k) = \sum_{r=0}^{M} b_r x(n-r)$$

System function Take the *z*-transform of both sides of the above equation

$$\mathcal{Z}\left\{\sum_{k=0}^{N} a_k y(n-k)\right\} = \mathcal{Z}\left\{\sum_{r=0}^{M} b_r x(n-r)\right\}, \text{ or}$$

$$\sum_{k=0}^{N} a_k \, \mathcal{Z}\{y(n-k)\} = \sum_{r=0}^{M} b_r \, \mathcal{Z}\{x(n-r)\}, \text{ or}$$

$$\sum_{k=0}^{N} a_k z^{-k} \, Y(z) = \sum_{r=0}^{M} b_r z^{-r} \, X(z), \text{ or}$$

$$Y(z) \sum_{k=0}^{N} a_k z^{-k} = X(z) \sum_{r=0}^{M} b_r z^{-r}$$

The system function is $H(z) = \dfrac{Y(z)}{X(z)} = \dfrac{\sum_{r=0}^{M} b_r z^{-r}}{\sum_{k=0}^{N} a_k z^{-k}}$

Unit sample response If $x(n) = \delta(n)$ then $X(z) = \mathcal{Z}[x(n)] = \mathcal{Z}[\delta(n)] = 1$. The corresponding $y(n)$ is the unit sample response $h(n)$. We have

$$\frac{Y(z)}{X(z)} = H(z), \quad \text{or} \qquad Y(z) = H(z).X(z) = H(z).1 = H(z)$$

So, given $H(z)$, the system function, the unit sample response is $h(n) = \mathcal{Z}^{-1}[H(z)]$.

The difference equation from the *H(z)* The system function $H(z)$ is first written in terms of negative powers of z and set equal to $\dfrac{Y(z)}{X(z)}$. Then cross-multiply and take the inverse z-transform to get the difference equation.

Frequency response of the system is the Fourier transform (DTFT) of the unit sample response $h(n)$:

$$H(e^{j\omega}) = \sum_{n=-\infty}^{\infty} h(n)\, e^{-j\omega n}$$

Compare this with the system function $H(z)$ defined as the z-transform of the unit sample response $h(n)$

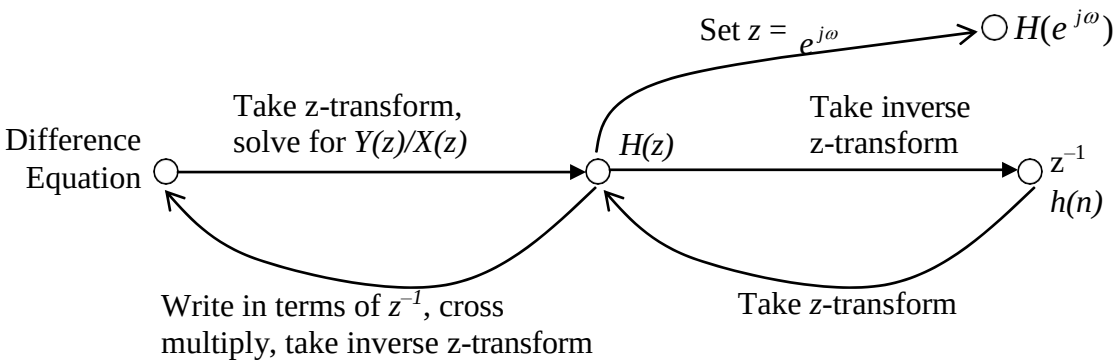
$$H(z) = \sum_{n=-\infty}^{\infty} h(n)\, z^{-n}$$

Thus the frequency response, if it exists, can be obtained by replacing the z in $H(z)$ by $e^{j\omega}$ as follows:

$$H(e^{j\omega}) \triangleq \sum_{n=-\infty}^{\infty} h(n) e^{-j\omega n} = H(z)\Big|_{z=e^{j\omega}}$$

The system is implicitly BIBO-stable.

The above relationships for a *stable, causal system* represented by a *linear constant coefficient difference equation* are summarized in diagram below.



Example 1.6.1 Find the impulse response of $y(n) = a\, y(n-1) + x(n)$.

Solution Note that we have solved this in the time domain earlier. Taking the z-transform of both sides (with zero initial conditions),

$$Y(z) = a\, z^{-1}\, Y(z) + X(z), \text{ or}$$

$$H(z) = \frac{Y(z)}{X(z)} = = \frac{1}{1 - az^{-1}} = \frac{z}{z-a}$$

Assume causality. Then from the table of transforms, $h(n) = \mathcal{Z}^{-1}[H(z)] = a^n\, u(n)$.

Causality in terms of the *z*-transform, *H(z)*, and the ROC A causal LTI system has an impulse response $h(n) = 0$ for $n < 0$, and is, therefore, a right-sided sequence. This also implies that the ROC of $H(z)$ is the exterior of a circle in the z-plane. For a causal system the power series

$$H(z) = \sum_{n=0}^{\infty} h(n) z^{-n} = h(0) + h(1)\, z^{-1} + h(2)\, z^{-2} + \ldots \quad \rightarrow \text{Eq. (1)}$$

does not include any positive powers of z. Consequently, the ROC includes $z = \infty$. Therefore, we have the principle:

> *A discrete-time LTI system is causal if and only if the ROC of its system function is the exterior of a circle, and includes $z = \infty$.*

The initial value theorem says that for a causal sequence, $h(n)$, the initial value is given by

$$h(0) = \lim_{z\to\infty} H(z)$$

This may be seen by setting $z \rightarrow \infty$ in Eq. (1) making all terms go to zero except the term $h(0)$. Thus, for a causal sequence, $h(n)$, if $h(0)$ is finite, then, $\lim_{z\to\infty} H(z)$ is finite. Consequently, with $H(z)$ expressed as a ratio of polynomials in z (positive powers of z), *the order of the numerator polynomial cannot be greater than the order of the denominator polynomial* (if it were there

would be positive powers of z in the power series of $H(z)$, corresponding to non-zero $h(n)$ for negative n; also $z = \infty$ would not be included in the ROC); or, *equivalently, the number of finite zeros of H(z) cannot be greater than the number of finite poles.*

The above discussion is summed up as follows: A discrete-time LTI system with rational system function $H(z)$ is causal if and only if

1. The ROC is the exterior of a circle outside the outermost pole, and,
2. With $H(z)$ expressed as a ratio of polynomials in z, (positive powers of z), the order of the numerator is not greater than the order of the denominator.

Condition 1 alone is not enough because the sequence may be right-sided but not causal.

If $H(z)$ is represented as a ratio of polynomials in z as

$$H(z) = \frac{\beta_K z^K + \beta_{K-1} z^{K-1} + \dots\dots\dots + \beta_1 z + \beta_0}{\alpha_L z^L + \alpha_{L-1} z^{L-1} + \square + \alpha_1 z + \alpha_0} \qquad \rightarrow \text{Eq. (2)}$$

then $L \geq K$ if the system is causal – in other words denominator degree $\geq$ numerator degree. On the other hand, if we write $H(z)$ as the ratio of polynomials in z^{-1} (negative powers of z) as

$$H(z) = \frac{b_0 + b_1 z^{-1} + \dots\dots\dots + b_{M-1} z^{-M+1} + b_M z^{-M}}{a_0 + a_1 z^{-1} + \square + a_{N-1} z^{-N+1} + a_N z^{-N}}$$

$$= \frac{b_0 + (b_1 / z) + \dots\dots\dots + (b_{M-1} / z^{M-1}) + (b_M / z^M)}{a_0 + (a_1 / z) + \square + (a_{N-1} / z^{N-1}) + (a_N / z^N)}$$

then, if the system is (to be) causal, $a_0 \neq 0$. This is seen by setting $z \rightarrow \infty$, and requiring that $h(0) = (b_0/a_0)$ be finite. This is illustrated with an example where $a_0 = 0$, e.g.,

$$H(z) = \frac{1 + z^{-1} + z^{-2}}{0 + z^{-1} + z^{-2}} = \frac{z^2 + z + 1}{z + 1}$$

which, by long division, can be seen to contain z^1 – a positive power of z – hence non-casual.)

Note When $H(z)$ is written as a ratio of polynomials in z (positive power of z), as in Eq. (2), we have required that $L \geq K$ for causality. These L and K are not to be confused with the N and M contained in the difference equation. Consider, for example, the system

$$y(n) + a\, y(n-1) = x(n) + b\, x(n-3)$$

where, according to the notation of the difference equation, $N = 1$ and $M = 3$. Apparently M is greater than N and this is allowable. In other words, there is no restriction on the relative values of N and M. For, the transfer function is given by

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1 + b z^{-3}}{1 + a z^{-1}} = \frac{z^{-3}(z^3 + b)}{z^{-1}(z + a)} = \frac{z^3 + b}{z^2 (z + a)}$$

and it is seen that the numerator degree ($K = 3$) is not greater than the denominator degree ($L = 3$). Thus the system is causal.

As another example consider $y(n) + a\, y(n-1) = x(n) + b\, x(n+1)$ which is non-causal because of the $x(n+1)$ term. The transfer function is

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1 + b z^{1}}{1 + a z^{-1}} = \frac{z(b + z^{-1})}{1 + a z^{-1}} = \frac{(b + z^{-1})}{z^{-1} + a z^{-2}} = \frac{(b + z^{-1})}{0 + z^{-1} + a z^{-2}}$$

Note that, when the numerator and denominator are expressed in terms of negative powers of z, "a_0" = 0. On the other hand, when the numerator and denominator are expressed in terms of positive powers of z, we have

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_2 z^2}{z + a}$$

with the numerator degree greater than the denominator degree.

(Omit) Rational transfer function; LTI system Given the system with the Nth order difference equation,

$$a_0\, y(n) + a_1\, y(n-1) + \ldots + a_N\, y(n-N)$$
$$= b_0\, x(n) + b_1\, x(n-1) + \ldots + b_M\, x(n-M), \qquad a_0 \neq 0$$

we may write it in the more compact form

$$\sum_{k=0}^{N} a_k\; y(n-k) = \sum_{r=0}^{M} b_r\; x(n-r), \qquad a_0 \neq 0$$

(Note that some authors take the coefficient of $y(n)$, a_0, to be 1. In the above difference equation we may divide through by a_0 so that the coefficient of $y(n)$ is 1).

We can find the transfer function of the system by taking the z-transform on both sides of the equation. We note that in finding the impulse response of a system and, consequently, in finding the transfer function, the system must be initially relaxed ("zero initial conditions"). Thus, if we assume zero initial conditions, we can use the linearity and time-shift properties to get

$$Y(z) \sum_{k=0}^{N} a_k\; z^{-k} = X(z) \sum_{r=0}^{M} b_r\, z^{-r}$$

so that

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{r=0}^{M} b_r z^{-r}}{\sum_{k=0}^{N} a_k z^{-k}} \qquad \text{Eq. (1)}$$

The corresponding impulse response can be found as $h(n) = \mathcal{Z}^{-1}\{H(z)\}$. The poles of the system transfer function are the same as the characteristic values of the corresponding difference equation. For the system to be stable, the poles must lie within the unit circle in the z-plane. Consequently, for a stable, causal function, the ROC includes the unit circle.

The system function, $H(z)$, is a rational function:

$$H(z) = \frac{N(z)}{D(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots\ldots\ldots + b_M z^{-M}}{a_0 + a_1 z^{-1} + a_2 z^{-2} + \ldots\ldots\ldots + a_N z^{-N}} = \frac{\sum_{k=0}^{M} b_k z^{-k}}{\sum_{k=0}^{N} a_k z^{-k}}$$

Here $N(z)$ and $D(z)$ stand for numerator and denominator respectively. If $a_0 \neq 0$ and $b_0 \neq 0$, we can avoid the negative powers of z by factoring out $b_0 z^{-M}$ and $a_0 z^{-N}$ as follows:

$$H(z) = \frac{N(z)}{D(z)} = \frac{b_0 z^{-M}}{a_0 z^{-N}} \cdot \frac{z^M + (b_1/b_0) z^{M-1} + \ldots\ldots + (b_M/b_0)}{z^N + (a_1/a_0) z^{N-1} + \ldots\ldots\ldots + (a_N/a_0)}$$

Since $N(z)$ and $D(z)$ are polynomials in z, they can be expressed in factored form as

$$H(z) = \frac{N(z)}{D(z)} = \left(\frac{b_0}{a_0}\right) z^{N-M} \cdot \frac{(z - z_1)(z - z_2)\ldots\ldots\ldots(z - z_M)}{(z - p_1)(z - p_2)\ldots\ldots\ldots(z - p_N)}$$

$$= C\, z^{N-M} \cdot \frac{\prod_{k=1}^{M}(z - z_k)}{\prod_{k=1}^{N}(z - p_k)}, \text{ where } C = (b_0/a_0)$$

Thus $H(z)$ has M ***finite zeros*** at $z = z_1, z_2, \ldots, z_M$, and N ***finite poles*** at $z = p_1, p_2, \ldots, p_N$, and $|N–M|$ zeros (if $N > M$) or poles (if $N < M$) at the origin $z = 0$. Poles and zeros may also occur at $z = \infty$. A pole exists at $z = \infty$ if $H(\infty) = \infty$, and a zero exists at $z = \infty$ if $H(\infty) = 0$. If we count the poles and zeros at $z = 0$ and $z = \infty$ as well as the N poles and M zeros, we find that $H(z)$ has exactly the same number of poles and zeros.

By definition the ROC of $H(z)$ should not (can not) contain any poles.

Proper rational function Taking $a_0 = 1$, we have

$$H(z) = \frac{N(z)}{D(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots\ldots\ldots + b_M z^{-M}}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots\ldots\ldots + a_N z^{-N}}$$

This is called a **proper** rational function if $a_N \neq 0$ and $M < N$. This amounts to saying that the number of finite zeros is less than the number of finite poles. (*Finite* zeros and poles exclude those at $z = 0$). This condition is related to partial fraction expansion and has nothing to do with causality.

(End of Omit)

Example 1.6.2 Give the pole-zero plot for $H(z) = \dfrac{z}{z^2 - z^1 - 1} = \dfrac{z^{-1}}{1 - z^{-1} - z^{-2}}$

Solution The denominator has roots (poles) at

$$z_1, z_2 = \frac{-(-1) \pm \sqrt{(-1)^2 - 4(1)(-1)}}{2} = \frac{1 \pm \sqrt{1+4}}{2} = \frac{1 \pm \sqrt{5}}{2} = 1.62 \text{ and } -0.62$$

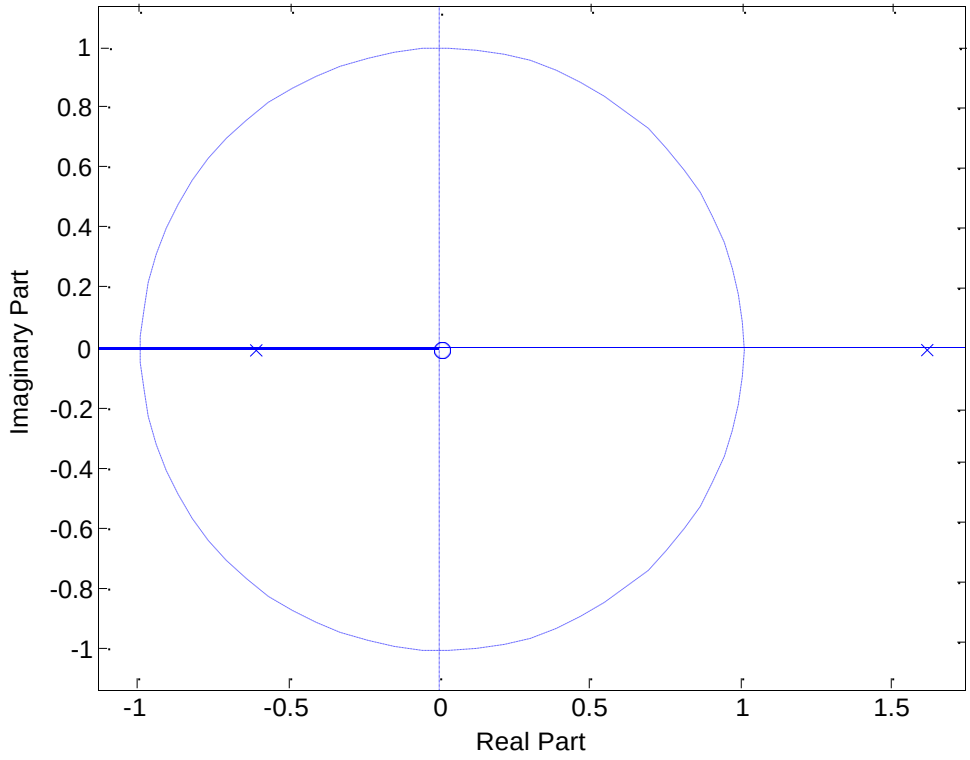
There is a zero at $z = 0$. Further, since the denominator degree is greater than the numerator degree by 1 it is clear that $H(\infty) = 0$, so that there is an additional zero at $z = \infty$.

In MATLAB the transfer function is specified as a ratio of polynomials in z^{-1}

$$H(z) = \frac{0 + 1.z^{-1}}{1 - 1.z^{-1} - 1.z^{-2}}$$

The numerator coefficients, $\{b_i, i = 0 \text{ to } M\}$ and the denominator coefficients $\{a_i, i = 0 \text{ to } N\}$ are specified as the two vectors $b = [0, 1]$ and $a = [1, -1, -1]$.

```
%Pole-zero plot
b = [0, 1]; a = [1, -1, -1]; zplane (b, a)
```


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
Imaginary Part
-1
-0.5
0
0.5
1
1.5
Real Part

$$H(z) = z^{-1} + 2z^{-2} + 3z^{-3} + 4z^{-4} + 5z^{-5} + 6z^{-6} + 7z^{-7} + 8z^{-8} + 9z^{-9}$$

Solution From

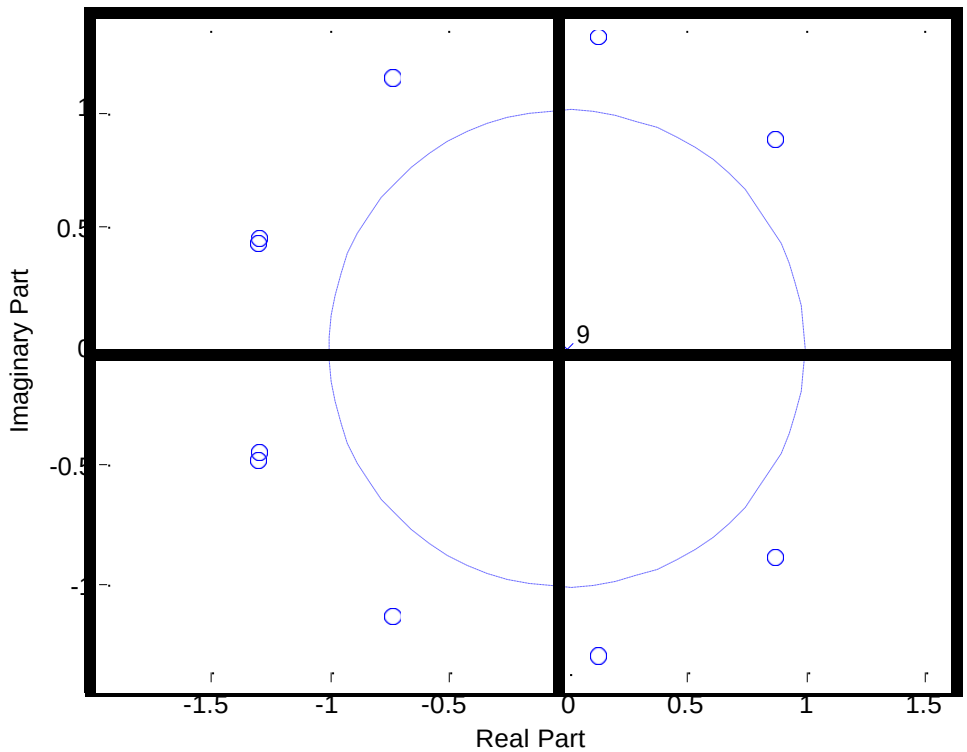
$$H(z) = \frac{z^8 + 2z^7 + 3z^6 + 4z^5 + 5z^4 + 6z^3 + 7z^2 + 8z + 9}{z^9}$$

we can see that there are 9 poles at $z = 0$ and 8 zeros at sundry places and an additional zero at $z = \infty$ owing to the denominator degree being greater than the numerator degree by 1.

For the MATLAB segment the numerator and denominator coefficients are taken from

$$H(z) = \frac{0 + z^{-1} + 2z^{-2} + 3z^{-3} + 4z^{-4} + 5z^{-5} + 6z^{-6} + 7z^{-7} + 8z^{-8} + 9z^{-9}}{1}$$

```
%Pole-zero plot
b = [0: 9]; a = [1, 0]; zplane (b, a)
```



Example 1.6.4 Give the pole-zero plot for

$$y(n) = x(n) + 0.81\, x(n-1) - 0.81\, x(n-2) - 0.45\, y(n-2)$$

Solution The zeros are

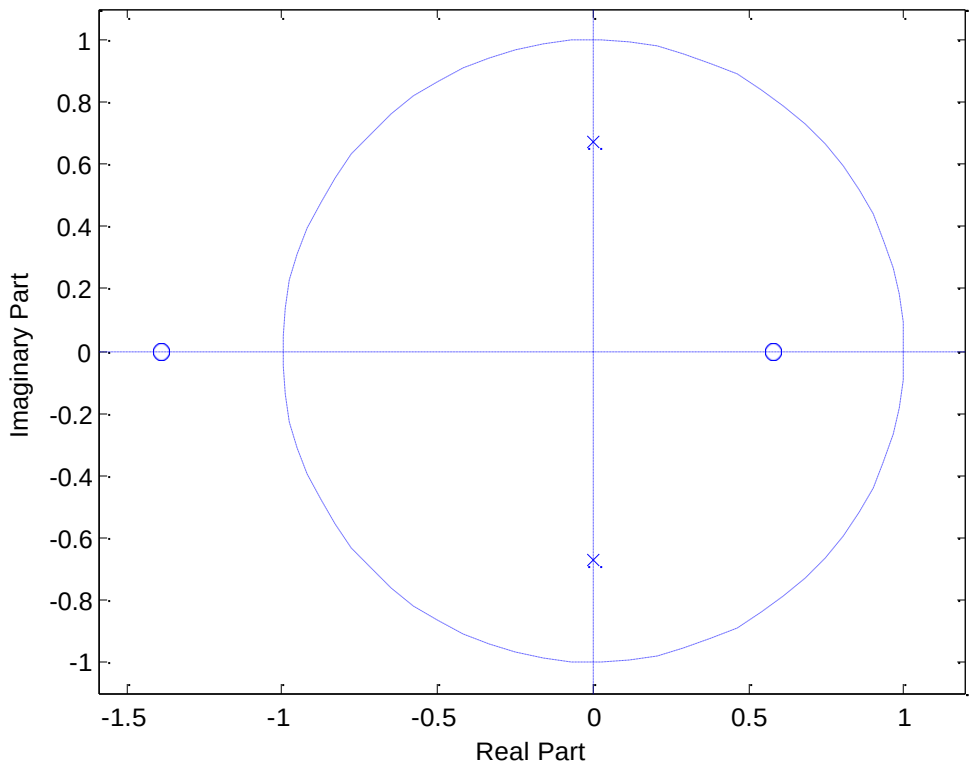
$$z_1, z_2 = \frac{-0.81 \pm \sqrt{(0.81)^2 - 4\,(1)(-0.81)}}{2\,(1)} = 0.5819 \text{ and } -1.3919$$

The poles are given by

$$z_1, z_2 = \pm\, j0.67082$$

For the MATLAB program the coefficient vectors are b = [1, 0.81, -0.81] and a = [1, 0, 0.45].

```
%Pole-zero plot
b = [1, 0.81, -0.81]; a = [1, 0, 0.45]; zplane (b, a)
```



From the general form $H(z)$ in Eq.(1) we can obtain two important special forms: (1) the all-zero system, and (2) the all-pole system. (There are, of course, trivial poles or zeros present.)

The all-zero system If $a_k = 0$ for $1 \le k \le N$, we have $H(z) = \dfrac{\sum_{r=0}^{M} b_r z^{-r}}{a_0}$. Either take $a_0 = 1$ or consider that a_0 is absorbed in the b_r coefficients, so that

$$H(z) = b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_M z^{-M} = \frac{b_0 z^M + b_1 z^{M-1} + \ldots + b_M}{z^M}$$

In this case, $H(z)$ contains M zeros and an Mth order pole at the origin $z = 0$. Since the system contains only ***trivial poles*** (at $z = 0$) and M ***non-trivial zeros***, it is called an all-zero system. Such a system has a finite-duration impulse response (FIR), and is called an FIR system or a moving average (MA) system. Note that the corresponding deference equation is

$$a_0\, y(n) = b_0\, x(n) + b_1\, x(n-1) + \ldots + b_M\, x(n-M)$$

The all-pole system On the other hand, if $b_k = 0$ for $1 \le k \le M$, we have

$$H(z) = \frac{b_0}{\sum_{k=0}^{N} a_k z^{-k}} = \frac{b_0}{a_0 + a_1 z^{-1} + a_2 z^{-2} + \square + a_N z^{-N}}$$

$$= \left(\frac{b_0}{a_0}\right) \cdot \frac{z^N}{z^N + (a_1/a_0) z^{N-1} + (a_2/a_0) z^{N-2} + \ldots + (a_N/a_0)}$$

Here again, either take $a_0 = 1$ or imagine that it is absorbed in the other coefficients viz., b_0, a_1, a_2, …, a_N. Thus

$$H(z) = \frac{b_0 z^N}{z^N + a_1 z^{N-1} + a_2 z^{N-2} + \ldots + a_N}$$

Here $H(z)$ has N poles and an Nth order zero at the origin $z = 0$. We usually do not make reference to these ***trivial zeros***. As a result this system function contains only ***non-trivial poles*** and the corresponding system is called an all-pole system. Due to the presence of the poles, the impulse response of such system is infinite in duration, and hence it is an IIR system. (We can divide the numerator into the denominator and thereby expand $H(z)$ into an infinite series from which it is evident that $h(n)$ is of infinite duration). Note that the corresponding deference equation is

$$a_0\, y(n) + a_1\, y(n-1) + \ldots + a_N\, y(n-N) = b_0\, x(n)$$

The pole-zero system The general form, though, contains both poles and zeros and the system is called a pole-zero system with N poles and M zeros,

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \square + b_M z^{-M}}{a_0 + a_1 z^{-1} + a_2 z^{-2} + \ldots\ldots\ldots + a_N z^{-N}}$$

Poles and/or zeros at $z = 0$ and $z = \infty$ are implied but are not counted explicitly. Due to the presence of poles, the pole-zero system is an IIR system.

Inverse *z*-transform by power series expansion (long division)

If the z-transform is expressed as a *rational function* (a ratio of polynomials in z or z^{-1}) we can use long division to expand it into a power series. If the transform is expressed as an *irrational function* we can use the appropriate power series expansion formula available in mathematical

tables such as the CRC Tables. Note that if the transform is expressed as an irrational function then the partial fraction expansion method of inversion won't work.

By definition the z-transform of the sequence $x(n)$ is given by

$$X(z) = \sum_{n=-\infty}^{\infty} x(n)\, z^{-n} = \ldots + x(-2)\, z^{2} + x(-1)\, z^{1} + x(0)\, z^{0} + x(1)\, z^{-1} + \ldots$$

This is a power series (Laurent series). So by long division we obtain the power series expansion of $X(z)$ and then, by comparison with the power series definition given above, we can identity the sequence $x(n)$. In particular the coefficient of z^{-k} is the sequence value $x(k)$.

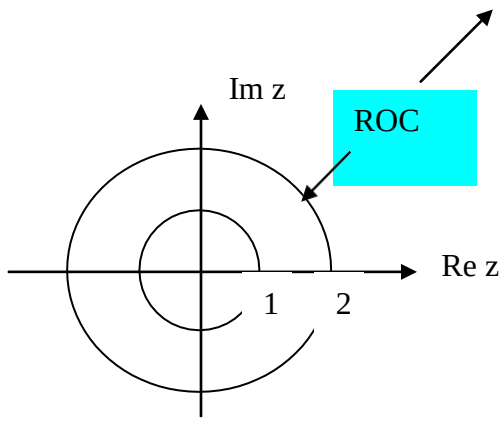
The method is useful in obtaining a quick look at the first few values of the sequence $x(n)$. This approach does not assure an analytical solution. The ROC will determine whether the series has positive or negative exponents. For right-sided sequences the $X(z)$ will be obtained with primarily negative exponents, while left-sided sequences will have primarily positive exponents. For an annular ROC, a Laurent expansion would give both positive- and negative-time terms. This last possibility is illustrated in the example below by taking a little help from partial fractions.

Example 1.7.1 Find the inverse transform, by long division, of

$$X(z) = \frac{2z^2 - 3z}{(z-1)(z-2)} = \frac{2z^2 - 3z}{z^2 - 3z + 2}$$

where the ROC is (a) $|z| > 2$, (b) $|z| < 1$, (c) $1 < |z| < 2$

Solution (a) ROC is $|z| > 2$. We expect a right-sided sequence, with predominantly negative exponents of z. For the long division arrange numerator and denominator as *decreasing powers*



of z and then divide; or as increasingly negative power of z i.e., z^{-1} and then divide.

$$
\begin{array}{lll}
 & 2 \quad +3z^{-1}+5z^{-2} + 9z^{-3} + \ldots & \leftarrow Q(z) \\
D(z) \rightarrow \quad z^2 - 3z + 2 & \left|\begin{array}{l} 2z^2 - 3z \\ 2z^2 - 6z + 4 \end{array}\right. & \leftarrow N(z) \\
 & \quad 3z - 4 & \\
 & \quad 3z - 9 + \quad 6z^{-1} & \\
 & \qquad 5 - \quad 6z^{-1} & \\
 & \qquad 5 - 15z^{-1} + 10z^{-2} & \\
 & \qquad\qquad 9z^{-1} - 10z^{-2} & \\
 & \qquad\qquad 9z^{-1} - 27z^{-2} + 18z^{-3} & \\
 & \qquad\qquad\qquad 17z^{-2} - 18z^{-3} & \\
 & \qquad\qquad\qquad \ldots &
\end{array}
$$

Thus $X(z) = 2 + 3z^{-1} + 5z^{-2} + 9z^{-3} + \ldots$ By comparison with the defining equation

$$X(z) = \ldots x(-1)z^{1} + x(0) + x(1)z^{-1} + x(2)z^{-2} + \ldots$$

we see that the sequence values are

$x(-2) = x(-1) = 0$, or $x(n) = 0$ for $n < 0$, and
$x(0) = 2$, $x(1) = 3$, $x(2) = 5$, etc.

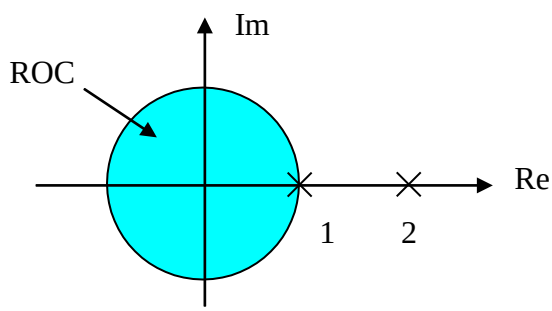
Alternatively, it is also possible to write $X(z)$ as a ratio of polynomials in z^{-1}

$$X(z) = \frac{2 - 3z^{-1}}{1 - 3z^{-1} + 2z^{-2}}$$

Note that the polynomials are written in the order of increasing negative powers of z, that is, z^{-1}. Long division gives (the same answer as obtained earlier):

$$\begin{array}{rl|l}
 & & 2 + 3z^{-1} + 5z^{-2} + 9z^{-3} + \ldots & \leftarrow Q(z) \\
D(z) \rightarrow & 1 - 3z^{-1} + 2z^{-2} & 2 - 3z^{-1} & \leftarrow N(z) \\
 & & 2 - 6z^{-1} + 4z^{-2} & \\
\hline
 & & \quad 3z^{-1} - 4z^{-2} & \\
 & & \quad 3z^{-1} - 9z^{-2} + 6z^{-3} & \\
\hline
 & & \qquad 5z^{-2} - 6z^{-3} & \\
 & & \qquad \ldots & \\
\hline
\end{array}$$

Solution (b) The ROC is $|z| < 1$. We expect a left-sided sequence with predominantly positive exponents of z. For the long division the polynomials are written in the order of increasing powers of z (or decreasingly negative powers of z, i.e., z^{-1}).



$$\begin{array}{rl|l}
 & & -(3/2)z - (5/4)z^2 - (9/8)z^3 - \ldots & \leftarrow Q(z) \\
D(z) \rightarrow & 2 - 3z + z^2 & -3z + 2z^2 & \leftarrow N(z) \\
 & & -3z + (9/2)z^2 - (3/2)z^3 & \\
\hline
 & & \quad -(5/2)z^2 + (3/2)z^3 & \\
 & & \quad -(5/2)z^2 + (15/4)z^3 - (5/4)z^4 & \\
\hline
 & & \qquad -(9/4)z^3 + (5/4)z^4 & \\
 & & \qquad -(9/4)z^3 + (27/8)z^4 - (9/8)z^5 & \\
\hline
\end{array}$$

Thus $X(z) = -(3/2)z - (5/4)z^2 - (9/8)z^3 - \ldots = \ldots - (9/8)z^3 - (5/4)z^2 - (3/2)z$. By comparing with the defining equation

$$X(z) = \ldots x(-3)z^{3} + x(-2)z^{2} + x(-1)z + x(0) + x(1)z^{-1} + \ldots$$

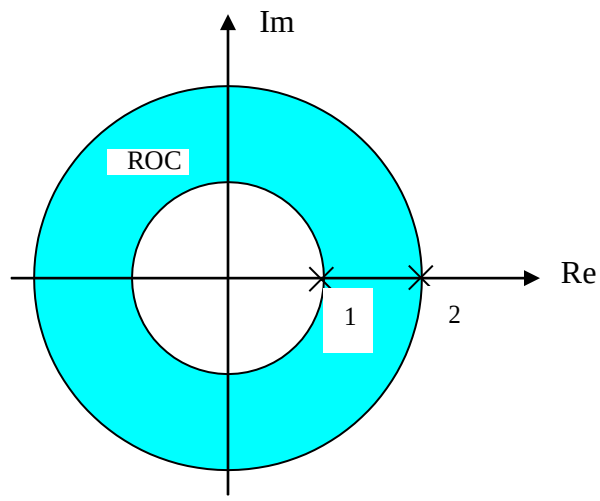
we see that the sequence is given by

$$x(-1) = -3/2,\ x(-2) = -5/4,\ x(-3) = -9/8,\ \ldots\text{etc., and } x(n) = 0 \text{ for } n \geq 0$$

The other way of long division is shown below:

$$\begin{array}{lll} & & -(3/2)z - (5/4)z^2 - (9/8)z^3 - \ldots \quad \leftarrow Q(z) \\ D(z) \rightarrow & 2z^{-2} - 3z^{-1} + 1 & \boxed{\begin{array}{l} -3z^{-1} + 2 \\ -3z^{-1} + (9/2) - (3/2)z \end{array}} \quad \leftarrow N(z) \\ & & \ldots \end{array}$$

(Omit) Solution (c) The ROC is $1 < |z| < 2$. We expect a two-sided sequence with both positive and negative exponents of z. Looking at the pole-zero configuration, the pole at $z = 1$ implies a right-sided sequence and the pole at $z = 2$ a left-sided sequence. Obviously just a single long division cannot give both the left-sided and the right-sided sequences simultaneously. We shall obtain the partial fraction expansion first and then proceed with the division to obtain the sequences separately. These two sequences are then combined into one sequence to get the solution. Note that we do this only to illustrate the method of long division. But once we use partial fractions the utility of long division is nullified.



$$\frac{X(z)}{z} = \frac{2z-3}{(z-1)(z-2)} = \frac{A}{z-1} + \frac{B}{z-2}$$

$$A = \left.\frac{2z-3}{z-2}\right|_{z=1} = (2 \,.\, 1 - 3) / (1 - 2) = 1$$

$$B = \left.\frac{2z-3}{z-1}\right|_{z=2} = (2 \,.\, 2 - 3) / (2 - 1) = 1$$

$$\frac{X(z)}{z} = \frac{1}{z-1} + \frac{1}{z-2}$$

$$X(z) = \frac{z}{z-1} + \frac{z}{z-2}$$

For the term $\dfrac{z}{z-1}$ we have a right-sided sequence given by long division thus:

$$
\begin{array}{rl|l}
 & & 1 + z^{-1} + z^{-2} + z^{-3} + \ldots\ldots & \leftarrow Q(z) \\
D(z) \rightarrow & z - 1 & z & \leftarrow N(z) \\
 & & z - 1 & \\
\hline
 & & \quad 1 & \\
 & & \quad 1 - z^{-1} & \\
\hline
 & & \qquad z^{-1} & \\
 & & \qquad z^{-1} - z^{-2} & \\
\hline
 & & \qquad\qquad z^{-2} & \\
 & & \ldots & \\
\hline
\end{array}
$$

The corresponding sequence is $x_R(n) = \begin{cases} 1, & n \geq 0 \\ 0, & \text{otherwise} \end{cases}$

For the term $\dfrac{z}{z-2}$ we have a left sided sequence

$$
\begin{array}{rl|l}
 & & -(1/2)z - (1/4)z^2 - (1/8)z^3 \ldots\ldots & \leftarrow Q(z) \\
D(z) \rightarrow & -2 + z & z & \leftarrow N(z) \\
 & & z - (1/2)z^2 & \\
\hline
 & & \quad (1/2)z^2 & \\
 & & \quad (1/2)z^2 - (1/4)z^3 & \\
\hline
 & & \qquad\qquad (1/4)z^3 & \\
 & & \qquad\qquad (1/4)z^3 - (1/8)z^4 & \\
\hline
 & & \qquad\qquad\qquad (1/8)z^4 & \\
 & & \ldots & \\
\hline
\end{array}
$$

The corresponding sequence is $x_L(n) = \begin{cases} -2^{-n}, & n < 0 \\ 0, & \text{otherwise} \end{cases}$

The complete sequence is then

$$x(n) = x_R(n) + x_L(n) = \begin{cases} 1, & n \geq 0 \\ -2^{-n}, & n < 0 \end{cases}$$

(End of Omit)

Computation of frequency response

Let the system function be given by

$$H(z) = \frac{\sum_{r=0}^{M} b_r z^{-r}}{\sum_{k=0}^{N} a_k z^{-k}}$$

The frequency response is $H(e^{j\omega})$ or $H(\omega) = H(z)\big|_{z = e^{j\omega}}$. Thus

$$H(e^{j\omega}) = |H(\omega)|\, e^{j\angle H(\omega)} = \frac{\sum_{r=0}^{M} b_r e^{-j\omega r}}{\sum_{k=0}^{N} a_k e^{-j\omega k}} = \frac{\sum_{r=0}^{M} b_r \cos\omega r - j\sum_{r=0}^{M} b_r \sin\omega r}{\sum_{k=0}^{N} a_k \cos\omega k - j\sum_{k=0}^{N} a_k \sin\omega k}$$

$$= \frac{\sum_{r=0}^{M} b_r \cos\omega r - j\sum_{r=0}^{M} b_r \sin\omega r}{\sum_{k=0}^{N} a_k \cos\omega k - j\sum_{k=0}^{N} a_k \sin\omega k} = \frac{A - jB}{C - jD}$$

where

$$A = \sum_{r=0}^{M} b_r \cos\omega r, \qquad B = \sum_{r=0}^{M} b_r \sin\omega r, \qquad C = \sum_{k=0}^{N} a_k \cos\omega k, \qquad D = \sum_{k=0}^{N} a_k \sin\omega k.$$

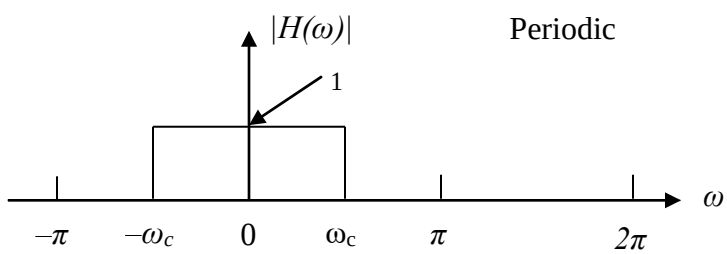
The magnitude and phase of $H(e^{j\omega})$ are given, respectively, by

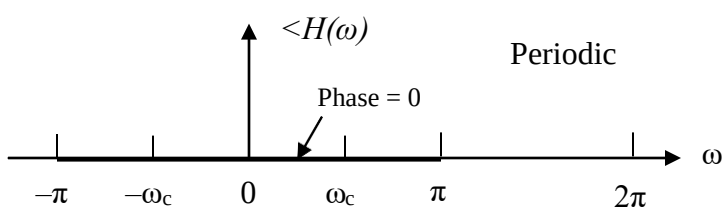
$$|H(\omega)| = \sqrt{\frac{A^2 + B^2}{C^2 + D^2}} \qquad \text{and} \qquad \angle H(\omega) = \tan^{-1}\left(\frac{-B}{A}\right) - \tan^{-1}\left(\frac{-D}{C}\right)$$

Theorem The frequency response $H(e^{j\omega})$ for a BIBO-stable system will always converge.

Accordingly every BIBO-stable system will have a frequency response and a describable steady-state response to sinusoidal inputs. But, the converse of this statement is not true, that is, the fact that $H(e^{j\omega})$ exists does not imply that the system is stable.

Example 1.8.1 [The ideal low pass filter] For the $H(\omega)$ given in figure below find $h(n)$, the unit sample response.





Solution The unit sample response is the inverse DTFT of $H(\omega)$

$$h(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H(\omega) e^{j\omega n} d\omega = \frac{1}{2\pi}\int_{-\omega_c}^{\omega_c} 1 e^{j\omega n} d\omega = \frac{1}{2\pi}\left.\frac{e^{j\omega n}}{jn}\right|_{-\omega_c}^{\omega_c} = \frac{1}{\pi n}\frac{e^{j\omega_c n} - e^{-j\omega_c n}}{j2}$$

$$= \frac{\sin\omega_c n}{\pi n}, \qquad \text{for all } n$$

It is seen that $h(n) \neq 0$ for negative n so that the ideal low pass filter is *noncausal*. Moreover, although $h(n)$ tails off as n goes from 0 to ∞ and from 0 to $-\infty$, it can be shown that $\sum_{n=-\infty}^{\infty} |h(n)|$ is not finite. This means that the ideal low pass filter is *not BIBO-stable either*.

Example 4.8.2 [2002] A discrete system is given by the difference equation

$$y(n) - 5\,y(n-1) = x(n) + 4\,x(n-1)$$

where $x(n)$ is the input and $y(n)$ is the output. Determine the magnitude and phase response as a function of frequency for $\omega \leq \pi$. (Note that the system is not stable since it has a pole at $z = 5$, which is outside the unit circle. The fact that the steady state frequency response exists does not mean that the system is stable.)

Solution [See also Unit I] Taking the z-transform and with a dose of algebra we find the transfer function

$$H(z) = \frac{Y(z)}{X(z)} = \frac{z+4}{z-5}$$

The frequency response is given by

$$H(\omega) = H(z)\Big|_{z=e^{j\omega}} = \frac{z+4}{z-5}\Bigg|_{z=e^{j\omega}} = \frac{e^{j\omega}+4}{e^{j\omega}-5} = \frac{N(\omega)}{D(\omega)}$$

$$N(\omega) = e^{j\omega} + 4 = \cos\omega + 4 + j\sin\omega = |N(\omega)|\, e^{j\angle N(\omega)}$$

$$|N(\omega)| = \sqrt{(\cos\omega+4)^2 + (\sin\omega)^2} \qquad \text{and} \qquad \angle N(\omega) = \tan^{-1}\left|\frac{\sin\omega}{\cos\omega+4}\right|$$

$$D(\omega) = e^{j\omega} - 5 = \cos\omega - 5 + j\sin\omega = |D(\omega)|\, e^{j\angle D(\omega)}$$

$$|D(\omega)| = \sqrt{(\cos\omega-5)^2 + (\sin\omega)^2} \qquad \text{and} \qquad \angle D(\omega) = \tan^{-1}\left|\frac{\sin\omega}{\cos\omega-5}\right|$$

$$|H(\omega)| = \frac{|N(\omega)|}{|D(\omega)|} = \frac{\sqrt{(\cos\omega+4)^2 + (\sin\omega)^2}}{\sqrt{(\cos\omega-5)^2 + (\sin\omega)^2}}$$

$$\angle H(\omega) = \angle N(\omega) - \angle D(\omega) = \tan^{-1}\left|\frac{\sin\omega}{\cos\omega+4}\right| - \tan^{-1}\left|\frac{\sin\omega}{\cos\omega-5}\right|$$

The frequency response can be plotted. Note that $|H(\omega)|$ is an even function and $\angle H(\omega)$ is an odd function of ω.

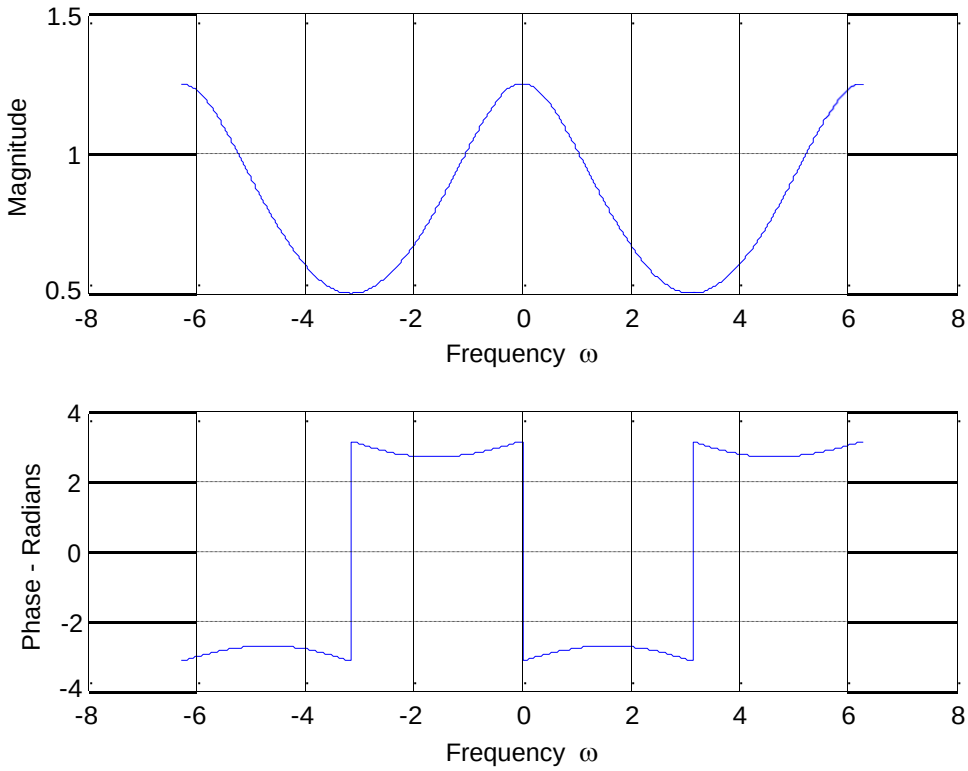
Using MATLAB:

$$H(\omega) = \frac{e^{j\omega}+4}{e^{j\omega}-5} = \frac{1+4e^{-j\omega}}{1-5e^{-j\omega}} = \frac{b(1) + b(2)e^{-j\omega} + b(3)e^{-j2\omega} + \ldots}{a(1) + a(2)e^{-j\omega} + a(3)e^{-j2\omega} + \ldots}$$

Here the vectors b and a specify, respectively, the numerator and denominator coefficients. In our example $b(1) = 1$, $b(2) = 4$, $a(1) = 1$, and $a(2) = -5$. The MATLAB segment and the corresponding plots follow. Note that the plot goes from -2π to 2π. Compare with the solution obtained in Unit I using a different function.

```
b = [1, 4]; %Numerator coefficients
a = [1, -5]; %Denominator coefficients
w = -2*pi: pi/256: 2*pi;
[h] = freqz(b, a, w);
subplot(2, 1, 1), plot(w, abs(h));
xlabel('Frequency \omega'), ylabel('Magnitude'); grid
```

```
subplot(2, 1, 2), plot(w, angle(h));
xlabel('Frequency \omega'), ylabel('Phase - Radians'); grid
```



Example 1.8.3 Assume $H(z) = \dfrac{12z^2 - 1}{6z^2 - z - 1}$ is a causal system. Find the difference equation and the frequency response.

Solution Arrange $H(z)$ in terms of negative powers of z

$$H(z) = \frac{Y(z)}{X(z)} = \frac{z^2(12 - z^{-2})}{z^2(6 - z^{-1} - z^{-2})} = \frac{(12 - z^{-2})}{(6 - z^{-1} - z^{-2})}$$

Cross multiplying

$$Y(z)\,(6 - z^{-1} - z^{-2}) = X(z)\,(12 - z^{-2})$$

$$6Y(z) - z^{-1}Y(z) - z^{-2}Y(z) = 12X(z) - z^{-2}X(z)$$

Taking the inverse z-transform

$$6y(n) - y(n-1) - y(n-2) = 12x(n) - x(n-2)$$

$$y(n) = \frac{1}{6}y(n-1) + \frac{1}{6}y(n-2) + 2x(n) - \frac{1}{6}x(n-2)$$

The poles of $H(z)$ are located at

$$z_1, z_2 = \frac{-(-1) \pm \sqrt{(-1)^2 - 4(6)(-1)}}{2(6)} = \frac{1 \pm \sqrt{1 + 24}}{12} = \frac{1 \pm 5}{12} = 0.5 \text{ and } -1/3$$

and are inside the unit circle. This being a causal system, the ROC is $|z| > ½$ and contains the unit circle. The system is stable, and the frequency response is meaningful. It is given by

$$H(\omega) = H(z)\Big|_{z=e^{j\omega}} = \frac{12z^2-1}{6z^2-z-1}\Bigg|_{z=e^{j\omega}} = \frac{12(e^{j\omega})^2-1}{6(e^{j\omega})^2-e^{j\omega}-1} = \frac{N(\omega)}{D(\omega)}$$

where

$$N(\omega) = 12(e^{j\omega})^2-1 = 12e^{j2\omega}-1 = 12\cos 2\omega + j12\sin 2\omega - 1$$

$$= \sqrt{(12\cos 2\omega-1)^2+(12\sin 2\omega)^2}\; e^{j\tan^{-1}\left(\frac{12\sin 2\omega}{12\cos 2\omega-1}\right)}$$

$$D(\omega) = 6(e^{j\omega})^2 - e^{j\omega} - 1 = 6e^{j2\omega} - e^{j\omega} - 1$$

$$= 6\cos 2\omega + j6\sin 2\omega - \cos\omega - j\sin\omega - 1$$

$$= \sqrt{(6\cos 2\omega-\cos\omega-1)^2+(6\sin 2\omega-\sin\omega)^2}\; e^{j\tan^{-1}\left(\frac{6\sin 2\omega-\sin\omega}{6\cos 2\omega-\cos\omega-1}\right)}$$

The magnitude response is given by

$$|H(\omega)| = \frac{\sqrt{(12\cos 2\omega-1)^2+(12\sin 2\omega)^2}}{\sqrt{(6\cos 2\omega-\cos\omega-1)^2+(6\sin 2\omega-\sin\omega)^2}}$$

The phase response is given by

$$\angle H(\omega) = \tan^{-1}\left(\frac{12\sin 2\omega}{12\cos 2\omega-1}\right) - \tan^{-1}\left(\frac{6\sin 2\omega-\sin\omega}{6\cos 2\omega-\cos\omega-1}\right)$$

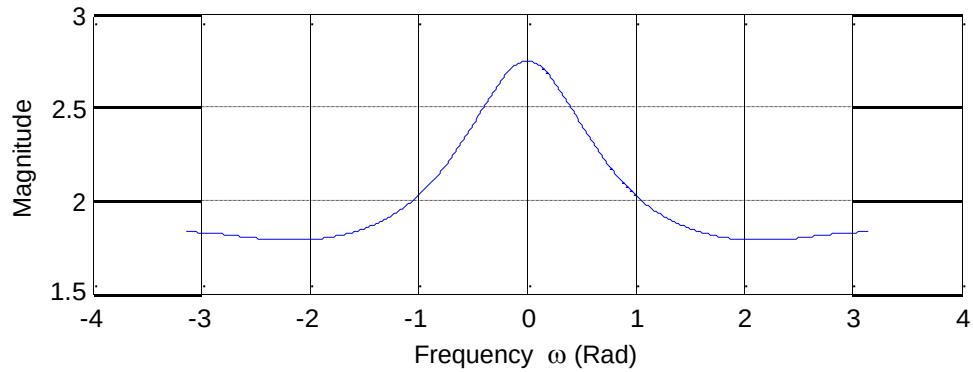
Using MATLAB:

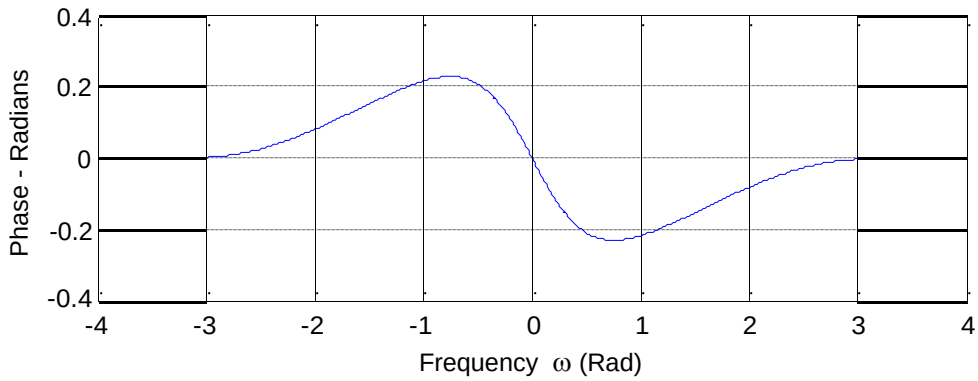
$$H(z) = \frac{12z^2-1}{6z^2-z-1} = \frac{12-z^{-2}}{6-z^{-1}-z^{-2}}$$

$$H(\omega) = \frac{12-e^{-j2\omega}}{6-e^{-j\omega}-e^{-j2\omega}} = \frac{b(1)+b(2)e^{-j\omega}+b(3)e^{-j2\omega}+\ldots}{a(1)+a(2)e^{-j\omega}+a(3)e^{-j2\omega}+\ldots}$$

Here the vectors *b* and *a* specify, respectively, the numerator and denominator coefficients. In our example *b(1)* = 12, *b(2)* = 0, *b(3)* = –1, *a(1)* = 6, *a(2)* = –1 and *a(3)* = –1. The MATLAB segment and the corresponding plots follow. Note that the plot goes from $-\pi$ to π.

```
b = [12, 0, -1]; %Numerator coefficients
a = [6, -1, -1]; %Denominator coefficients
w = -pi: pi/256: pi;
[h] = freqz(b, a, w);
subplot(2, 1, 1), plot(w, abs(h));
xlabel('Frequency \omega (Rad)'), ylabel('Magnitude'); grid
subplot(2, 1, 2), plot(w, angle(h));
xlabel('Frequency \omega (Rad)'), ylabel('Phase - Radians'); grid
```





Example 1.8.4 Discuss the stability of $H(z) = \dfrac{z^{-1}}{1 - z^{-1} - z^{-2}}$ assuming it is a causal system. Find the difference equation and the frequency response.

(b) Determine the frequency, magnitude and phase responses and time delay for the system $y(n) + (1/4)\, y(n-1) = x(n) - x(n-1)$.

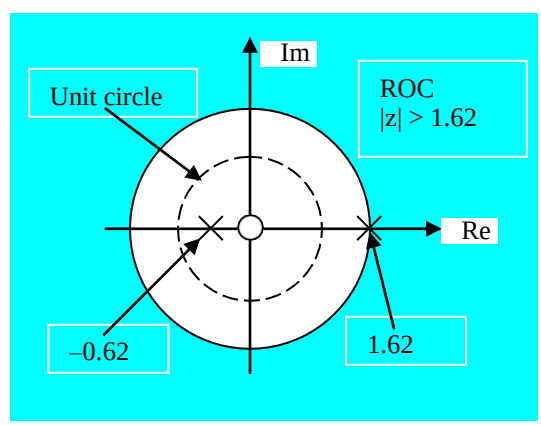
Solution

(a) Find the ROC and the poles:

$$H(z) = \frac{z^{-1}}{1 - z^{-1} - z^{-2}} = \frac{z}{z^{2} - z^{1} - 1}$$

There is a zero at $z = 0$. The denominator has roots at

$$z_1, z_2 = \frac{-(-1) \pm \sqrt{(-1)^2 - 4(1)(-1)}}{2} = \frac{1 \pm \sqrt{1+4}}{2} = \frac{1 \pm \sqrt{5}}{2} = 1.62 \text{ and } -0.62$$



The pole locations are shown here. For the system to be causal the ROC is the exterior of a circle with radius = 1.62. In this case ROC does not include the unit circle. (Equivalently, all the poles do not lie within the unit circle). Hence the system is not stable.

(b) Taking the *z*-transform on both sides of $y(n) + (1/4)\, y(n-1) = x(n) - x(n-1)$ we get

$$Y(z) + (1/4)\, z^{-1}\, Y(z) = X(z) - z^{-1}\, X(z), \text{ or}$$
$$Y(z)\,\{1 + (1/4)\, z^{-1}\} = X(z)\,\{1 - z^{-1}\}, \text{ or}$$

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1 - z^{-1}}{1 + (1/4)z^{-1}} = \frac{z-1}{z+0.25}$$

There is a single zero at $z = 1$ and a single pole at $z = -0.25$ which is inside the unit circle – hence stable. The frequency response is given by

$$H(\omega) = H(z)\,\Big|_{z=e^{j\omega}} = \frac{z-1}{z+0.25}\Big|_{z=e^{j\omega}} = \frac{e^{j\omega}-1}{e^{j\omega}+0.25} = \frac{N(\omega)}{D(\omega)}$$

where

$$N(\omega) = e^{j\omega} - 1 = \cos\omega + j\sin\omega - 1 = \sqrt{(\cos\omega - 1)^2 + \sin^2\omega}\;\; e^{j\tan^{-1}\left(\frac{\sin\omega}{\cos\omega - 1}\right)}$$

$$D(\omega) = e^{j\omega} - 0.25 = \cos\omega + j\sin\omega - 0.25$$
$$= \sqrt{(\cos\omega - 0.25)^2 + \sin^2\omega}\;\; e^{j\tan^{-1}\left(\frac{\sin\omega}{\cos\omega - 0.25}\right)}$$

The magnitude response is given by

$$|H(\omega)| = \frac{\sqrt{(\cos\omega - 1)^2 + \sin^2\omega}}{\sqrt{(\cos\omega - 0.25)^2 + \sin^2\omega}}$$

The phase response is given by

$$\angle H(\omega) = \tan^{-1}\left(\frac{\sin\omega}{\cos\omega - 1}\right) - \tan^{-1}\left(\frac{\sin\omega}{\cos\omega - 0.25}\right)$$

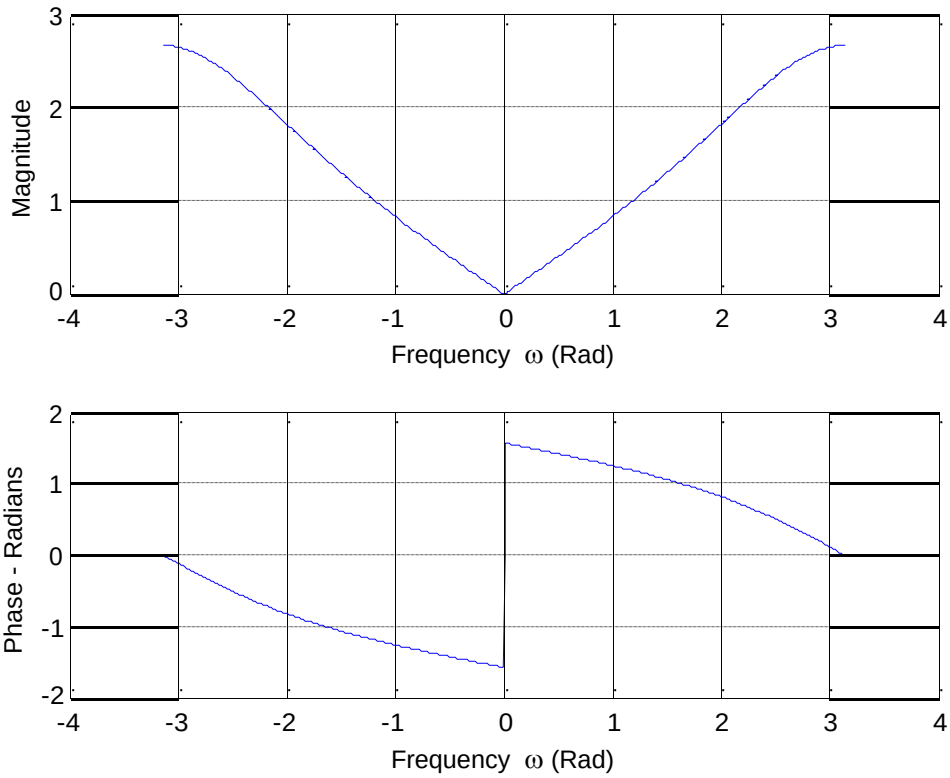
The time (group) delay is given by $-\dfrac{d}{d\omega}\angle H(\omega)$.

Using MATLAB:

$$H(\omega) = \frac{e^{j\omega}-1}{e^{j\omega}+0.25} = \frac{1 - e^{-j\omega}}{1 + 0.25e^{-j\omega}} = \frac{b(1) + b(2)e^{-j\omega} + b(3)e^{-j2\omega} + \ldots}{a(1) + a(2)e^{-j\omega} + a(3)e^{-j2\omega} + \ldots}$$

Here the vectors *b* and *a* specify, respectively, the numerator and denominator coefficients. In our example $b(1) = 1$, $b(2) = -1$, $a(1) = 1$, $a(2) = 0.25$. The MATLAB segment and the corresponding plots follow. Note that the plot goes from $-\pi$ to π.

```
b = [1, -1]; %Numerator coefficients
a = [1, 0.25]; %Denominator coefficients
w = -pi: pi/256: pi;
[h] = freqz(b, a, w);
subplot(2, 1, 1), plot(w, abs(h));
xlabel('Frequency \omega (Rad)'), ylabel('Magnitude'); grid
subplot(2, 1, 2), plot(w, angle(h));
xlabel('Frequency \omega (Rad)'), ylabel('Phase - Radians'); grid
```



Z-transforms with initial conditions

To solve the Nth order difference equation

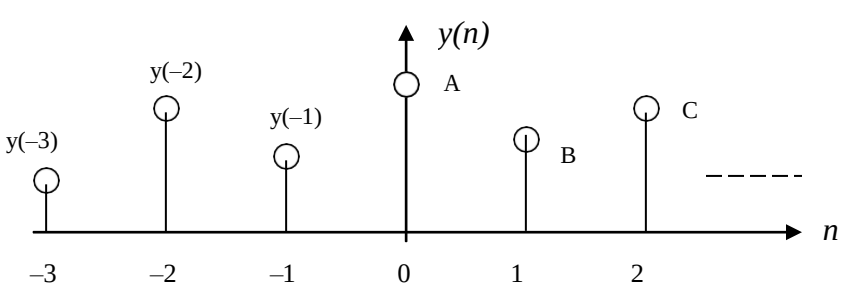
$$y(n) = -\sum_{k=1}^{N} a_k y(n-k) + \sum_{r=0}^{M} b_r x(n-r)$$

with (non-zero) initial conditions we need N initial conditions on the output $y(n)$ and M initial conditions on the input $x(n)$. Usually the input is applied suddenly (i.e., it is stepped into the system) at $n = 0$, so that no initial conditions are needed for it, that is, $x(n) = 0$ for $n < 0$. The output $y(n)$, however, in general, will have non-zero initial conditions for $n = -1$ to $-N$.

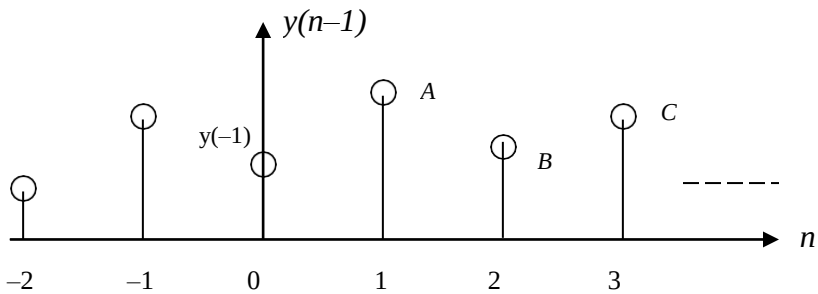
We are solving for $y(n)$ for $n \geq 0$, so that $Y(z) = \mathfrak{Z}\{y(n)\}$ is the one-sided z-transform. The difference equation contains other terms like $y(n-1)$, $y(n-2)$, etc. which are delayed versions of $y(n)$. Suppose $N = 3$, then we shall have $y(n-1)$, $y(n-2)$, and $y(n-3)$ to deal with. The transform of $y(n-1)$ is handled as follows. First, for the sequence $y(n)$ as shown below we define

$$Y_+(z) = \mathfrak{Z}\{y(n),\ n \geq 0\} = A + Bz^{-1} + Cz^{-2} + \ldots$$

We shall refer to this loosely as just $Y(z)$ when there is no possibility of confusion.



We then obtain $y(n-1)$ by delaying the sequence by one unit, shown below.



As can be seen from the graph

$$\mathcal{Z}\{y(n-1),\ n \geq 0\} = y(-1)\, z^0 + \underbrace{Az^{-1} + Bz^{-2} + Cz^{-3} + \ldots}_{z^{-1}Y_+(z)} = z^{-1}Y_+(z) + y(-1)$$

In a similar fashion

$$\mathcal{Z}\{y(n-2),\ n \geq 0\} = y(-2)z^0 + y(-1)\, z^{-1} + \underbrace{Az^{-2} + Bz^{-3} + Cz^{-4} + \ldots}_{z^{-2}Y_+(z)}$$

$$= z^{-2}Y_+(z) + y(-2) + y(-1)z^{-1}$$

and by extension

$$\mathcal{Z}\{y(n-3),\ n \geq 0\} = z^{-3}Y_+(z) + y(-3) + y(-2)z^{-1} + y(-1)z^{-2}$$

For $N = 3$ this would be the last. But we can generalize

$$\mathcal{Z}\{y(n-k),\ n \geq 0\} = z^{-k}Y_+(z) + y(-k) + y(-\overline{k-1})z^{-1} + \ldots + y(-1)z^{-(k-1)}$$

In the case of the input $x(n)$, since it is applied suddenly at $n = 0$, the initial conditions are zero, that is, $x(-1) = x(-2) = \ldots = x(-M) = 0$, so that

$$\mathcal{Z}\{x(n-k)\, u(n)\} = z^{-k} X_+(z)$$

With this intuitive background we give below the mathematical derivation of the z-transform of the delayed truncated sequence.

***Z*-transform of delayed truncated sequence (initial conditions)** The one-sided z-transform of $x(n)$ is

$$X_+(z) = \mathcal{Z}\{x(n)\, u(n)\} = \sum_{n=0}^{\infty} x(n)\, z^{-n}$$

Given the sequence $x(n)$, we delay it by k units, and then truncate it to the left of $n = 0$ to get $x(n-k)\, u(n)$. We want find the z-transform of $x(n-k)\, u(n)$.

$$\mathcal{Z}\{x(n-k)\, u(n)\} = \sum_{n=-\infty}^{\infty} x(n-k)\, u(n)\, z^{-n} = \sum_{n=0}^{\infty} x(n-k)\, z^{-n}$$

If we let $n-k = r$, then $n = r+k$, and the summation limits $n = 0$ to ∞ become $r = -k$ to ∞. Then

$$\mathcal{Z}\{x(n-k)\, u(n)\} = \sum_{r=-k}^{\infty} x(r)\, z^{-(r+k)} = z^{-k} \sum_{r=-k}^{\infty} x(r)\, z^{-r}$$

$$= z^{-k}\left[\underbrace{\sum_{r=0}^{\infty} x(r)\, z^{-r}}_{X_+(z)} + \underbrace{\sum_{r=-k}^{-1} x(r)\, z^{-r}}_{\text{IC}}\right] = z^{-k}\left[X_+(z) + \sum_{r=-k}^{-1} x(r)\, z^{-r}\right]$$

$$= z^{-k}\left[X_+(z) + x(-k)\, z^{k} + x(-k-1)\, z^{k-1} + \ldots + x(-1)\, z^{1}\right]$$
$$= z^{-k} X_+(z) + \underbrace{x(-k) + x(-k-1)\, z^{-1} + \ldots + x(-1)\, z^{-(k-1)}}_{\text{Due to Initial Conditions}}$$

We shall loosely refer to $X_+(z)$ as $X(z)$ and write the result as

$$\mathcal{Z}\{x(n-k)\, u(n)\} = z^{-k} X(z) + \underbrace{x(-k) + x(-k-1)\, z^{-1} + \ldots + x(-1)\, z^{-(k-1)}}_{\text{Due to Initial Conditions}}$$

The above result is used to solve linear constant coefficient difference equations with *inputs that are stepped into a system*. Suppose we want the solution of

$$\sum_{k=0}^{N} a_k y(n-k) = \sum_{r=0}^{M} b_r x(n-r), \qquad n \geq 0$$

subject to the initial conditions

$$\{y(i),\ i = -1, -2, \ldots, -N\} \qquad \text{and} \qquad \{x(i),\ i = -1, -2, \ldots, -M\}$$

We take the z-transform of the equation using the result derived above for delayed-truncated sequences

$$\mathcal{Z}\left\{\sum_{k=0}^{N} a_k y(n-k)\right\} = \mathcal{Z}\left\{\sum_{r=0}^{M} b_r x(n-r)\right\}, \qquad n \geq 0$$

$$\sum_{k=0}^{N} a_k Z\{y(n-k)\} = \sum_{r=0}^{M} b_r Z\{x(n-r)\}, \qquad n \geq 0$$

where we have used Z to mean $\mathcal{Z}$ the z-transformation operation. The left hand side is

$$\begin{aligned}\text{LHS} &= a_0\, \mathcal{Z}\{y(n)\} + a_1\, \mathcal{Z}\{y(n-1)\} + a_2\, \mathcal{Z}\{y(n-2)\} + \ldots + a_N\, \mathcal{Z}\{y(n-N)\} \\ &= a_0\, Y(z) + a_1\, \{z^{-1} Y(z) + y(-1)\} + a_2\, \{z^{-2} Y(z) + y(-2) + y(-1)\, z^{-1}\} + \\ &\qquad \ldots + a_N\, \{z^{-N} Y(z) + y(-N) + y(-(N-1))\, z^{-1} + \ldots + y(-1) z^{-N+1}\}\end{aligned}$$

(Note that in terms of the derivation earlier all of the $Y(z)$'s are $Y_+(z)$'s, i.e. , one-sided transforms). All the $Y(z)$ terms can be grouped together under a summation, and all the remaining terms, due to the initial conditions $\{y(i),\ i = -1, -2, \ldots, -N\}$, can be grouped together so that the above can be written as

$$\text{LHS} = \sum_{k=0}^{N} a_k z^{-k} Y(z) + \underbrace{g\{z^{-1}, y(-1), y(-2), \ldots, y(-N)\}}_{\text{Initial condition terms}}$$

By following a similar procedure the right hand side can also be written as follows (here again the $X(z)$'s are $X_+(z)$'s, i.e. , one-sided transforms):

$$\text{RHS} = \sum_{r=0}^{M} b_r z^{-r} X(z) + \underbrace{h\{z^{-1}, x(-1), x(-2), \ldots, x(-M)\}}_{\text{Initial condition terms}}$$

Writing out in full, LHS = RHS becomes

$$\sum_{k=0}^{N} a_k z^{-k} Y(z) + g\{\ldots\} = \sum_{r=0}^{M} b_r z^{-r} X(z) + h\{\ldots\}$$

Factoring out *Y(z)* and *X(z)* and rearranging we have

$$Y(z) \sum_{k=0}^{N} a_k z^{-k} = X(z) \sum_{r=0}^{M} b_r z^{-r} + h\{\ldots\} - g\{\ldots\}$$

$$Y(z) = X(z) \frac{\sum_{r=0}^{M} b_r z^{-r}}{\sum_{k=0}^{N} a_k z^{-k}} + \frac{h\{\ldots\} - g\{\ldots\}}{\sum_{k=0}^{N} a_k z^{-k}}$$

Taking the inverse *z*-transform we get

$$y(n) = \mathcal{Z}^{-1}\left\{ X(z) \frac{\sum_{r=0}^{M} b_r z^{-r}}{\sum_{k=0}^{N} a_k z^{-k}} \right\} + \mathcal{Z}^{-1}\left\{ \frac{h\{\ldots\} - g\{\ldots\}}{\sum_{k=0}^{N} a_k z^{-k}} \right\}$$

To summarize: to solve for *y(n)* we take the *z*-transform of the linear constant coefficient difference equation using initial conditions, manipulate in the *z*-domain to get *Y(z)* and then take the inverse *z*-transform of *Y(z)* to get *y(n)*.

Example 1.9.1 Find the solution to

$$y(n) - \frac{3}{2} y(n-1) + \frac{1}{2} y(n-2) = \left(\frac{1}{4}\right)^n, \qquad n \geq 0$$

with initial conditions $y(-1) = 4$, $y(-2) = 10$.

Solution There are three methods of solution:

1. Find the iterative solution in the discrete-time domain. In general this will not give an analytical (closed) form of solution.
2. Solve in the discrete-time domain (homogeneous solution + particular solution).
3. Solve in the frequency domain as we do below.

For an *input sequence x(n)* that is stepped into a system, specified in words like $x(n) = 0$ for $n < 0$, the initial conditions are clearly zero and do not matter. But for an *output sequence y(n)* where the initial conditions *y(–1), y(–2)* are explicitly given to be non-zero we need to use the above derived "***z-transform for delayed truncated sequence***". In particular we have

$$\mathcal{Z}\{y(n)\} = Y(z)$$

$$\mathcal{Z}\{y(n-1)\} = z^{-1}\left[Y(z) + y(-1)\, z^{1}\right]$$

$$\mathcal{Z}\{y(n-2)\} = z^{-2}\left[Y(z) + y(-2)\, z^{2} + y(-1)\, z^{1}\right]$$

Taking the *z*-transform of the difference equation we get

$$\mathcal{Z}\left\{ y(n) - \frac{3}{2} y(n-1) + \frac{1}{2} y(n-2) \right\} = \mathcal{Z}\left\{ \left(\frac{1}{4}\right)^n \right\} \qquad n \geq 0$$

$$\mathcal{Z}\{y(n)\} - \frac{3}{2}\mathcal{Z}\{y(n-1)\} + \frac{1}{2}\mathcal{Z}\{y(n-2)\} = \frac{z}{z - (1/4)}$$

$$Y(z) - \frac{3}{2} z^{-1}\left[Y(z) + y(-1)\, z^{1}\right] + \frac{1}{2} z^{-2}\left[Y(z) + y(-2) z^{2} + y(-1) z^{1}\right] = \frac{z}{z - (1/4)}$$

Plugging in the initial conditions $y(-1) = 4$ and $y(-2) = 10$

$$Y(z) - \frac{3}{2}\left[z^{-1}Y(z)+4\right] + \frac{1}{2}\left[z^{-2}Y(z)+10+4z^{-1}\right] = \frac{z}{z-(1/4)}$$

$$Y(z)\left[1-\frac{3}{2}z^{-1}+\frac{1}{2}z^{-2}\right] - 6 + 5 + 2z^{-1} = \frac{z}{z-(1/4)}$$

$$Y(z)\left[1-1.5z^{-1}+0.5z^{-2}\right] = \frac{z}{z-(1/4)} + 1 - 2z^{-1} = \frac{2z^2-(9/4)z+(1/2)}{z\left(z-(1/4)\right)}$$

$$Y(z)\,(1-z^{-1})\,(1-0.5z^{-1}) = \frac{2z^2-(9/4)z+(1/2)}{z\left(z-(1/4)\right)}$$

$$Y(z)\,\frac{(z-1)\left(z-(1/2)\right)}{z^2} = \frac{2z^2-(9/4)z+(1/2)}{z\left(z-(1/4)\right)}$$

$$Y(z) = \frac{z\left(2z^2-(9/4)z+(1/2)\right)}{\left(z-(1/4)\right)(z-1)\left(z-(1/2)\right)}$$

$$\frac{Y(z)}{z} = \frac{\left(2z^2-(9/4)z+(1/2)\right)}{\left(z-(1/4)\right)(z-1)\left(z-(1/2)\right)} = \frac{A}{z-(1/4)} + \frac{B}{z-1} + \frac{C}{z-(1/2)}$$

$$A = \left.\frac{\left(2z^2-(9/4)z+(1/2)\right)}{(z-1)\left(z-(1/2)\right)}\right|_{z=1/4} = 1/3$$

$$B = \left.\frac{\left(2z^2-(9/4)z+(1/2)\right)}{\left(z-(1/4)\right)\left(z-(1/2)\right)}\right|_{z=1} = 2/3$$

$$C = \left.\frac{\left(2z^2-(9/4)z+(1/2)\right)}{\left(z-(1/4)\right)(z-1)}\right|_{z=1/2} = 1$$

$$\frac{Y(z)}{z} = \frac{(1/3)}{z-(1/4)} + \frac{(2/3)}{z-1} + \frac{1}{z-(1/2)}$$

$$Y(z) = \frac{1}{3}\frac{z}{z-(1/4)} + \frac{2}{3}\frac{z}{z-1} + \frac{z}{z-(1/2)}$$

$$y(n) = \left[\frac{1}{3}\left(\frac{1}{4}\right)^n + \frac{2}{3}1^n + \left(\frac{1}{2}\right)^n\right]u(n)$$

The iterative solution for this problem was obtained in Unit I. The time-domain solution was covered in HW (Extra). The solution is repeated below

$$y(n) = \left\{2, \frac{5}{4}, \frac{15}{16}, \frac{51}{64}, \ldots\right\}$$

Example 4.9.2 [2002] Solve the following linear difference equation

$$y(n) + \frac{1}{2}y(n-1) - \frac{1}{4}y(n-2) = 0$$

given $y(-1) = y(-2) = 1$.

Solution Note that the output is a natural response corresponding to the specified initial conditions. There is no forced response since $x(n) = 0$.

The iterative solution is

$$y(n) = \left\{1, -\frac{3}{4}, \frac{3}{8}, -\frac{1}{4}, \frac{7}{32}, \ldots\right\}$$

For the solution in the frequency domain we take the z-transform of the difference equation

$$\mathfrak{Z}\left(y(n) + \frac{1}{2} y(n-1) - \frac{1}{4} y(n-2) \right) = \mathfrak{Z}(0) \qquad n \geq 0$$

$$\mathfrak{Z}\{y(n)\} + \frac{1}{2}\,\mathfrak{Z}\{y(n-1)\} - \frac{1}{4}\,\mathfrak{Z}\{y(n-2)\} = 0$$

$$Y(z) + \frac{1}{2}\, z^{-1}\left[Y(z) + y(-1)\, z^{1}\right] - \frac{1}{4}\, z^{-2}\left[Y(z) + y(-2)\, z^{2} + y(-1)\, z^{1}\right] = 0$$

$$Y(z) + \frac{1}{2}\, z^{-1}\left[Y(z) + 1\, z^{1}\right] - \frac{1}{4}\, z^{-2}\left[Y(z) + 1z^{2} + 1z^{1}\right] = 0$$

$$Y(z)\left[1 + 0.5\, z^{-1} - 0.25\, z^{-2}\right] = -0.25 + 0.25 z^{-1}$$

$$Y(z) = \frac{(-0.25) z\,(z-1)}{z^{2} + 0.5z - 0.25}$$

$$\frac{Y(z)}{z} = \frac{(-0.25)(z-1)}{z^{2} + 0.5z - 0.25}$$

The denominator on the right hand side has roots at

$$z_1, z_2 = \frac{-0.5 \pm \sqrt{(0.5)^2 - 4(1)(-0.25)}}{2} = \frac{-1 \pm \sqrt{5}}{4} = 0.309 \text{ and } -0.809$$

$$\frac{Y(z)}{z} = \frac{(-0.25)(z-1)}{(z-0.309)(z+0.809)} = \frac{A}{z-0.309} + \frac{B}{z+0.809}$$

$$A = \left.\frac{(-0.25)(z-1)}{(z+0.809)}\right|_{z=0.309} = 0.155$$

$$B = \left.\frac{(-0.25)(z-1)}{(z-0.309)}\right|_{z=-0.809} = -0.405$$

$$Y(z) = \frac{0.155z}{z-0.309} - \frac{0.405z}{z+0.809}$$

$$y(n) = 0.155\ (0.309)^{n} - 0.405(-0.809)^{n}, \qquad n \geq 0$$

The first few values of the sequence are $y(n) = \{-0.25, 0.376, -0.25, 0.219, \ldots\}$ and should be compared with the iterative solution.

In the context of MATLAB, we may use *filter(b, a, x)* to generate the sequence *y(n)*. The coefficients of *y(.)* and *x(.)* are numbered slightly differently as below:

$$a_1\, y(n) + a_2\, y(n-1) + a_3\, y(n-2) + \ldots = b_1\, x(n) + b_2\, x(n-1) + b_3\, x(n-2) + \ldots$$

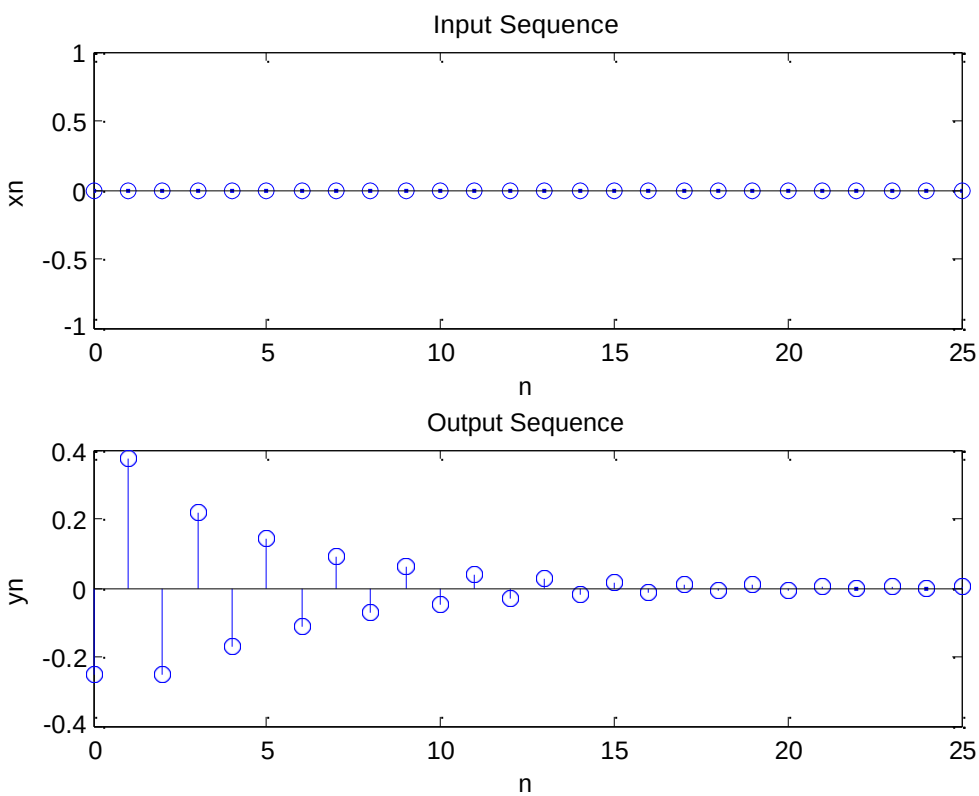
From the difference equation

$$y(n) + \frac{1}{2} y(n-1) - \frac{1}{4} y(n-2) = 0, \qquad n \geq 0$$

we note that the input is $x(n) = 0$ and the coefficients of *y(.)* and *x(.)* give us the *a* and *b* vectors: $a = [1, 0.5, -02.5]$ and $b = [1]$. The non-zero initial conditions $y(-1) = y(-2) = 1$ must first be converted to *equivalent initial conditions* for the *filter* function to work. We specify the vector $yic = [y(-1), y(-2)] = [1, 1]$ and generate the equivalent initial conditions *eic* by the function *filtic(b, a, yic)*. The equivalent initial conditions are then used to generate the filter output through *filter(b, a, xn, eic)*. The MATLAB segment follows:

```
%Non-zero initial conditions
b = [1], a = [1, 0.5, -0.25], yic = [1, 1],
n = 0:25, xn = 0 .*n
%
%Equivalent initial conditions
eic = filtic(b, a, yic),
yn = filter(b, a, xn, eic)
subplot(2, 1, 1), stem(n, xn);
xlabel('n'), ylabel('xn'); title('Input Sequence');
subplot(2, 1, 2), stem(n, yn);
xlabel('n'), ylabel('yn'); title('Output Sequence');
```

The output is:
yn = [-0.25 0.375 -0.25 0.2188 -0.1719 0.1406 -0.1133 … 0.0031 -0.0025 0.0020]



Example 1.9.3 Find the response sequence for the filter defined by

$$y(n) - 7\,y(n-1) + 126\,y(n-2) = x(n)$$

Assume the system is initially relaxed. Obtain the system function and plot its poles and zeros.
Solution The phrase "initially relaxed" means that the initial conditions are zero, that is, $y(n) = 0$ for $n < 0$ and $x(n) = 0$ for $n < 0$. The question doesn't specify what the input $x(n)$ is, so assume $\delta(n)$. (What will the output be if both the input *and* the initial conditions are zero?)

Steady-state and transient responses for a first order system

We consider sinusoidal inputs. Although the presentation is only for a first order system, the relationship established for the steady-state response in terms of the transfer function of the system is a general result for stable systems and sinusoidal inputs.

The system is

$$y(n) = a\, y(n-1) + x(n), \qquad n \geq 0$$

with the initial condition $y(-1)$ and the input $x(n) = \cos\omega_0 n\, u(n)$. (We have considered the time-domain behavior of this system in Unit I). Assume $|a| < 1$in order to have a stable system. The system function is obtained with zero initial conditions,

$$Y(z) = a\, z^{-1}\, Y(z) + X(z), \text{ or}$$

$$H(z) = \frac{Y(z)}{X(z)} = = \frac{1}{1 - az^{-1}} = \frac{z}{z-a}$$

The solution of the difference equation is obtained by taking the z-transform and using the given initial condition

$$Y(z) = a\,[z^{-1}\, Y(z) + y(-1)] + X(z)$$

$$Y(z)\left[1 - az^{-1}\right] = a\, y(-1) + X(z)$$

$$Y(z) = a\, y(-1)\,\frac{1}{1 - az^{-1}} + X(z)\,\frac{1}{1 - az^{-1}} = a\, y(-1)\, H(z) + X(z)\, H(z)$$

Since $X(z) = \mathcal{Z}\{x(n)\} = \mathcal{Z}\{\cos\omega_0 n\, u(n)\} = \dfrac{z^2 - z\cos\omega_0}{z^2 - 2z\cos\omega_0 + 1}$, we have

$$Y(z) = a\, y(-1)\, H(z) + \frac{z^2 - z\cos\omega_0}{z^2 - 2z\cos\omega_0 + 1}\, H(z) = Y_1(z) + Y_2(z)$$

$$= Y_1(z) \qquad\qquad = Y_2(z)$$

Here $Y_1(z)$ is the *zero-input response* due to the initial condition(s)

$$Y_1(z) = a\, y(-1) H(z) = \frac{a\, y(-1)\, z}{z - a}$$

and $Y_2(z)$ is the *forced response* due to the input $x(n)$

$$Y_2(z) = \frac{z^2 - z\cos\omega_0}{z^2 - 2z\cos\omega_0 + 1}\, H(z) = \frac{z\left(z^2 - z\cos\omega_0\right)}{(z-a)\left(z^2 - 2z\cos\omega_0 + 1\right)}$$

$Y_1(z)$ is already in a convenient form for taking the inverse, but $Y_2(z)$ must be expanded into partial fractions as below.

$$\frac{Y_2(z)}{z} = \frac{z\,(z - \cos\omega_0)}{(z-a)\left(z^2 - 2z\cos\omega_0 + 1\right)}$$

$$= \frac{A}{z-a} + \frac{B}{z - e^{j\omega_0}} + \frac{B^*}{z - e^{-j\omega_0}}$$

$$A = \left.\frac{z\,(z - \cos\omega_0)}{\left(z^2 - 2z\cos\omega_0 + 1\right)}\right|_{z=a}$$

$$= \frac{a\,(a - \cos\omega_0)}{a^2 - 2a\cos\omega_0 + 1}$$

Factoring of…

$$z^2 - 2z\cos\omega_0 + 1$$

$$= z^2 - 2z\left|\left(\frac{e^{j\omega_0} + e^{-j\omega_0}}{2}\right)\right| + 1$$

$$= z^2 - ze^{j\omega_0} - ze^{-j\omega_0} + 1$$

$$= z^2 - ze^{j\omega_0} - ze^{-j\omega_0} + e^{j\omega_0}e^{-j\omega_0}$$

$$= z(z - e^{j\omega_0}) - e^{-j\omega_0}(z - e^{j\omega_0})$$

$$= (z - e^{j\omega_0})(z - e^{-j\omega_0})$$

$$B = \left.\frac{z\,(z-\cos\omega_0)}{(z-a)\left(z-e^{-j\omega_0}\right)}\right|_{z=e^{j\omega_0}}$$

$$= \frac{e^{j\omega_0}\left(e^{j\omega_0}-\cos\omega_0\right)}{\left(e^{j\omega_0}-a\right)\left(e^{j\omega_0}-e^{-j\omega_0}\right)} = \frac{e^{j\omega_0}\left(\dfrac{e^{j\omega_0}+e^{-j\omega_0}}{2}\right)}{\left(e^{j\omega_0}-a\right)\left(e^{j\omega_0}-e^{-j\omega_0}\right)}$$

$$= \frac{e^{j\omega_0}\left(\dfrac{e^{j\omega_0}-e^{-j\omega_0}}{2}\right)}{\left(e^{j\omega_0}-a\right)\left(e^{j\omega_0}-e^{-j\omega_0}\right)}$$

$$= \frac{e^{j\omega_0}}{2\left(e^{j\omega_0}-a\right)} = \frac{1}{2}H(z)\Big|_{z=e^{j\omega_0}}$$

$$B^* = \frac{e^{-j\omega_0}}{2\left(e^{-j\omega_0}-a\right)}$$

So

$$Y(z) = Y_1(z) + Y_2(z) = Y_1(z) + \frac{Y_2(z)}{z}z$$

$$= \frac{a\,y(-1)\,z}{z-a} + \frac{A\,z}{z-a} + \frac{B\,z}{z-e^{j\omega_0}} + \frac{B^*\,z}{z-e^{-j\omega_0}}$$

Taking the inverse z-transform we get

$$y(n) = \underbrace{a\,y(-1)\;a^n u(n)}_{y_1(n)} + A\,a^n u(n) + B\left(e^{j\omega_0}\right)^n u(n) + B^*\left(e^{-j\omega_0}\right)^n u(n)$$

$$= y_2(n)$$

$$y(n) = [a\,y(-1) + A]\,a^n u(n) + \underbrace{B\left(e^{j\omega_0}\right)^n u(n) + B^*\left(e^{-j\omega_0}\right)^n u(n)}_{\boxed{\text{These are complex conjugates, hence } = 2\,\mathrm{Re}\left[Be^{j\omega_0 n}\right]u(n)}}$$

Using the fact that $B = \dfrac{\left(e^{j\omega_0}\right)}{2\left(e^{j\omega_0}-a\right)} = \dfrac{1}{2}H(z)\Big|_{z=e^{j\omega_0}}$

$$2\,\mathrm{Re}\left[Be^{j\omega_0 n}\right] = 2\,\mathrm{Re}\left[\frac{1}{2}H(z)\Big|_{z=e^{j\omega_0}} e^{j\omega_0 n}\right] = \mathrm{Re}\left[H(z)\Big|_{z=e^{j\omega_0}} e^{j\omega_0 n}\right]$$

Since $H(z)\Big|_{z=e^{j\omega_0}} = H(j\omega_0)$ we can write

$$2\,\mathrm{Re}\left[Be^{j\omega_0 n}\right] = \mathrm{Re}\left[H(j\omega_0)\,e^{j\angle H(j\omega_0)}e^{j\omega_0 n}\right] = \mathrm{Re}\left[H(j\omega_0)\,e^{j\angle H(j\omega_0)}e^{j\omega_0 n}\right]$$

$$= \mathrm{Re}\left[\left|H(j\omega_0)\right|e^{j(\omega_0 n+\angle H(j\omega_0))}\right]$$

$$= \left|H(j\omega_0)\right|\cos\left(\omega_0 n + \angle H(j\omega_0)\right)$$

Thus $y(n)$ becomes

$$y(n) = \underbrace{[a\,y(-1) + A]\,a^n u(n)}_{\textit{Transient response}} + \underbrace{\left|H(j\omega_0)\right|\cos\left(\omega_0 n + \angle H(j\omega_0)\right)u(n)}_{\textit{Steady-state response}}$$

Since $|a| < 1$ the transient term will eventually go to zero as $n \to \infty$. Even if the initial condition is zero, $y(-1) = 0$, there is still a transient response $Aa^n u(n)$ which eventually dies down.

If there is a nonzero initial condition, $y(-1)$, but the input $x(n) = 0$, the solution becomes $y(n) = y_1(n) = a\, y(-1)\, a^n u(n)$ which also dies down as $n \to \infty$.

Realization of digital filters

Given $H(z)$, the system function, or $h(n)$, the impulse response, the difference equation may be obtained. This difference equation could be implemented by *computer program*, *special purpose digital circuitry*, or *special programmable integrated circuit*. This direct evaluation of the difference equation is not the only possible realization of the digital filter. Alternative realizations of the digital filter are possible by breaking up the direct realization in some form.

Direct Form realization of IIR filters An important class of linear shift invariant systems can be characterized by the following rational system function (where $X(z)$ is the input, $Y(z)$ the output and we have taken $a_0 = 1$ in comparison with the earlier representation):

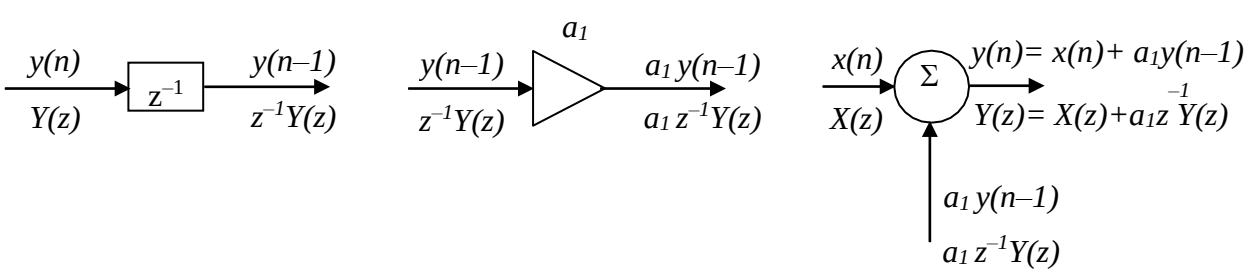
$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + \cdots + b_{M-1} z^{-M+1} + b_M z^{-M}}{1 + a_1 z^{-1} + \cdots + a_{N-1} z^{-N+1} + a_N z^{-N}} = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}}$$

By cross multiplying and taking the inverse z-transform we get the difference equation

$$y(n) = -\sum_{k=1}^{N} a_k y(n-k) + \sum_{r=0}^{M} b_r x(n-r)$$

$$= -a_1 y(n-1) - a_2 y(n-2) - \ldots - a_N y(n-N) + b_0 x(n) + b_1 x(n-1) + \ldots + b_M x(n-M)$$

To construct a filter structure we shall need three types of block diagram elements: a delay element, a multiplier and an adder, illustrated below:



We can construct a realization of the filter called the Direct Form I by starting with $y(n)$ and generating all the delayed versions $y(n-1), y(n-2), \ldots, y(n-N)$; similarly starting with $x(n)$ and generating all the delayed versions $x(n-1), x(n-2), \ldots, x(n-M)$. We then multiply the above terms by the respective coefficients and add them up. This is shown below (next page).

This is an Nth order system N being the order of the difference equation. There is no restriction as to whether M should be less than or greater than or equal to N. The total number of delay elements = $(N+M)$. It is not in canonic form because it uses more than the minimum

possible number of delay elements. It is called "Direct Form" because the multipliers are the actual filter coefficients $\{a_1, a_2, \ldots, a_N, b_0, b_1, b_2, \ldots, b_M\}$.

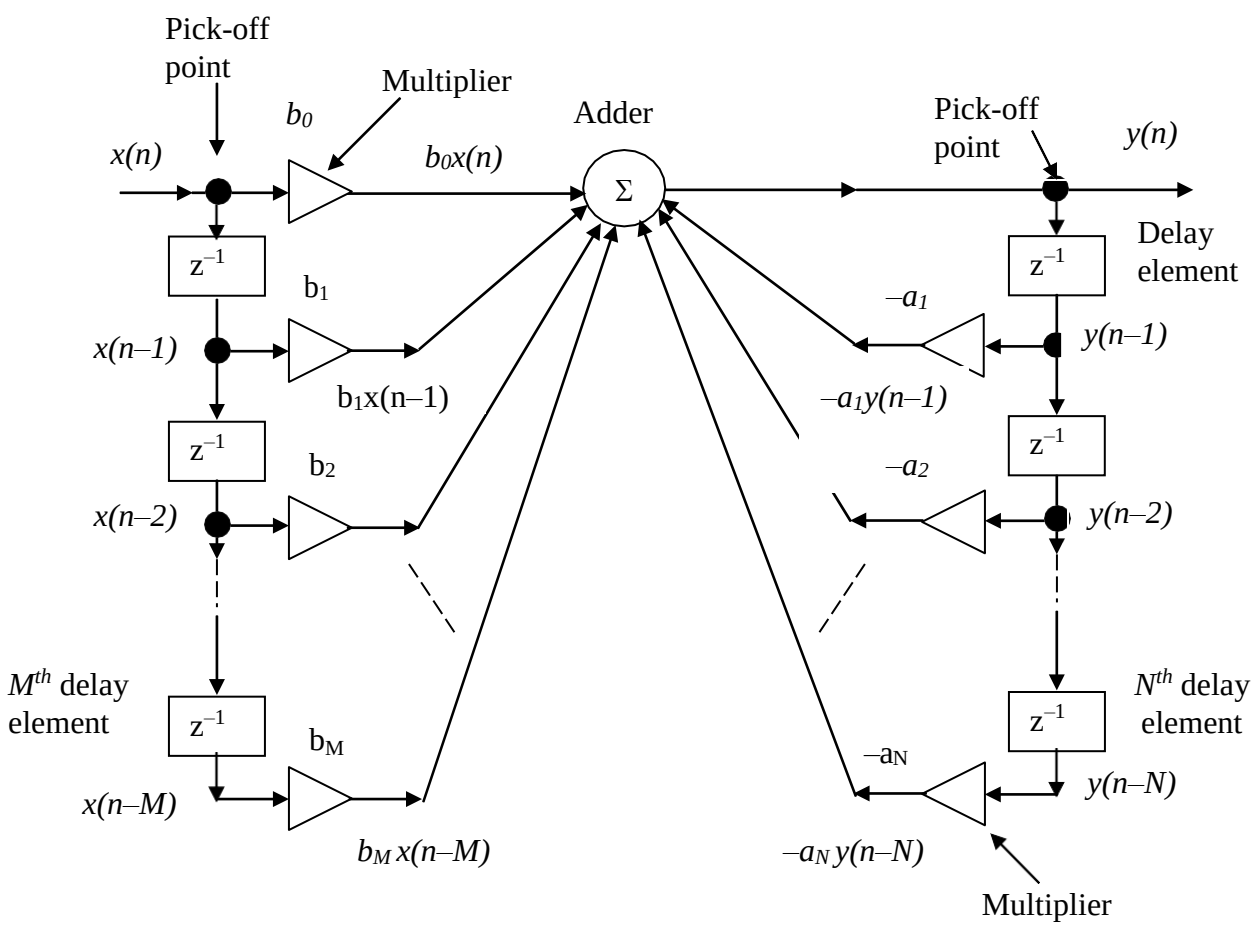
The difference equation of this realization (or structure) continues to be

$$y(n) = \underbrace{-a_1\, y(n-1) - a_2\, y(n-2) - \ldots - a_N\, y(n-N)}_{N \text{ multiplications}}$$

$$+ \underbrace{b_0\, x(n) + b_1\, x(n-1) + \ldots + b_M\, x(n-M)}_{(M+1) \text{ multiplications}}$$

and will be referred to as the Direct Form I difference equation. The total number of multiplications can be counted and is seen to be $(N+M+1)$. We can also count and see that there are $(N+M)$ additions. Finally, to calculate the value $y(n)$ we need to store N past values of $y(.)$, and M past values of $x(.)$, that is, a total of $(N+M)$ storage locations (storage for the present value of $x(.)$ is not counted).

Direct Form I



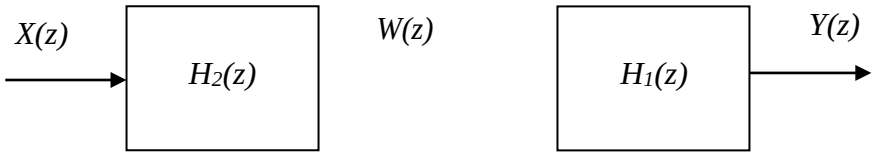
Rearrangement of Direct Form I The above diagram of Direct Form I, or the corresponding expression for $H(z)$, is sometimes rearranged as below. This shows visually that the transfer

function $H(z)$ is arranged as a cascade of an *all-zero system*, $H_2(z)$, followed by an *all-pole system*, $H_1(z)$:

$$H(z) = \frac{Y(z)}{X(z)} = \left(\sum_{k=0}^{M} b_k z^{-k}\right)\left(\frac{1}{1+\sum_{k=1}^{N} a_k z^{-k}}\right) = H_2(z)\, H_1(z)$$

$H_2(z)$ $\qquad$ $H_1(z)$

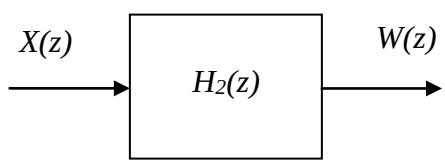
The overall block diagram then is shown thus:



The all-zero system is $H_2(z) = \dfrac{W(z)}{X(z)} = \left(\sum_{k=0}^{M} b_k z^{-k}\right)$ from which, by cross-multiplying and taking the inverse z-transform, we get the difference equation below:

$$W(z) = H_2(z)\, X(z) = X(z)\left(\sum_{k=0}^{M} b_k z^{-k}\right)$$

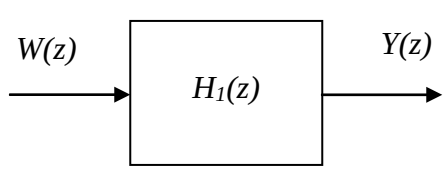
$$w(n) = b_0\, x(n) + b_1\, x(n-1) + \ldots + b_M\, x(n-M)$$



The all-pole system is $H_1(z) = \dfrac{Y(z)}{W(z)} = \left(\dfrac{1}{1+\sum_{k=1}^{N} a_k z^{-k}}\right)$ from which, by cross-multiplying and taking the inverse z-transform, we get the difference equation as below:

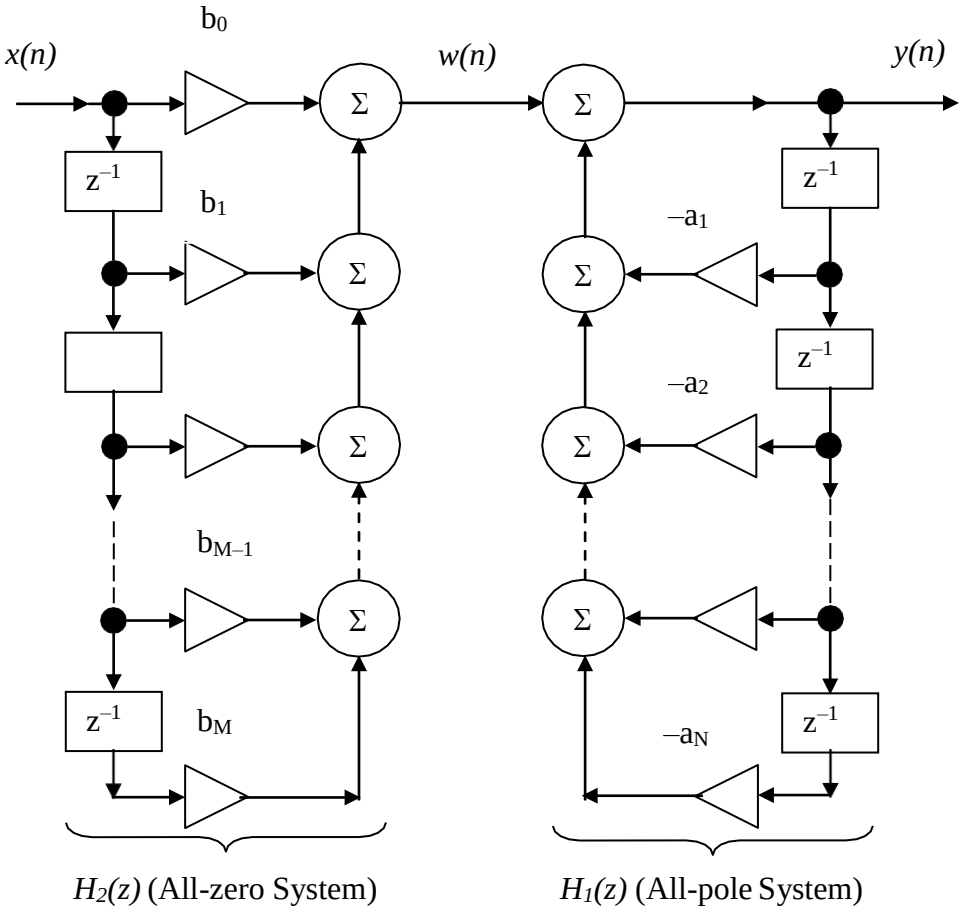
$$Y(z) = H_1(z)\, W(z) = W(z)\left(\frac{1}{1+\sum_{k=1}^{N} a_k z^{-k}}\right)$$

$$y(n) = w(n) - a_1\, y(n-1) - a_2\, y(n-2) - \ldots - a_N\, y(n-N)$$



Even though it seems that there are two equations, one for *w(n)* and another for *y(n)*, there is, in effect, only one since *w(n)* in the second equation is simply a short hand notation for the first equation and can be eliminated from the equation for *y(n)*.

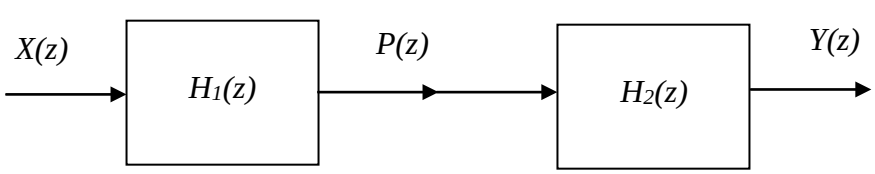
Overall, the Direct Form I has the following alternative appearance:



Derivation of Direct Form II The transfer function *H(z)* can be written as the product of the two transfer functions $H_1(z)$ and $H_2(z)$ as follows where we have reversed the sequence of the two blocks:

$$H(z) = \frac{Y(z)}{X(z)} = \left(\frac{1}{1 + \sum_{k=1}^{N} a_k z^{-k}} \right) \left(\sum_{k=0}^{M} b_k z^{-k} \right) = H_1(z)\, H_2(z)$$

$H_1(z)$ $H_2(z)$

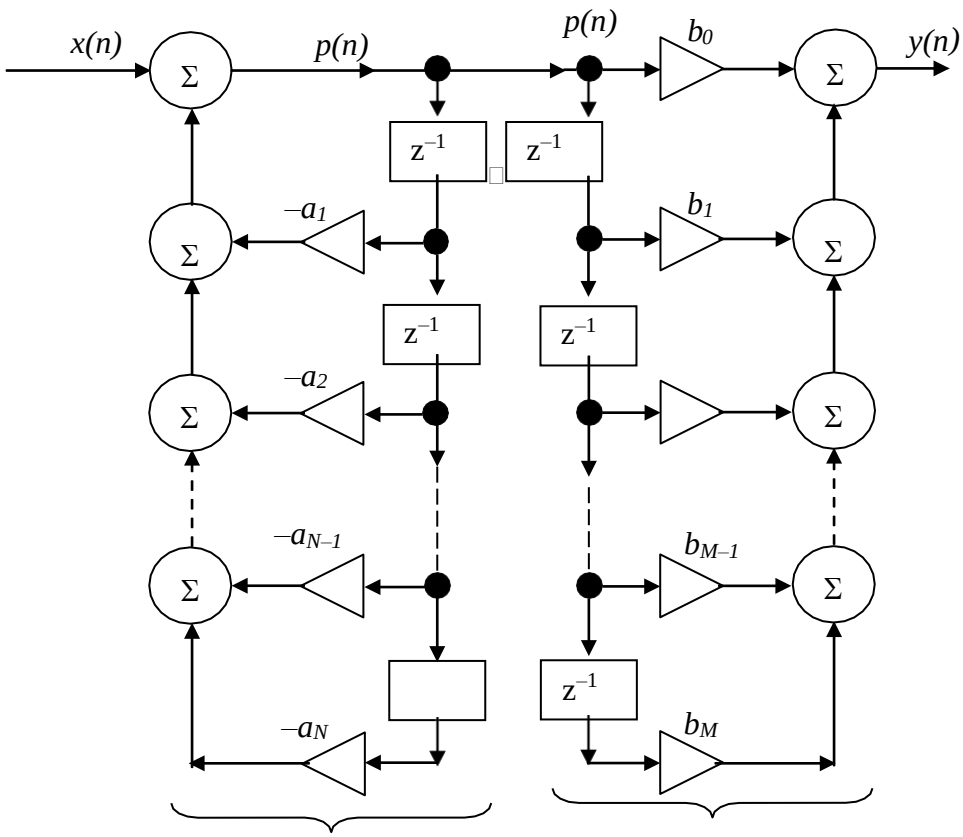


$$P(z) = H_1(z)\,X(z) = \left(\frac{1}{1+\sum_{k=1}^{N} a_k z^{-k}}\right) X(z) \qquad \text{and} \qquad Y(z) = H_2(z)\,P(z) = \left(\sum_{k=0}^{M} b_k z^{-k}\right) P(z)$$

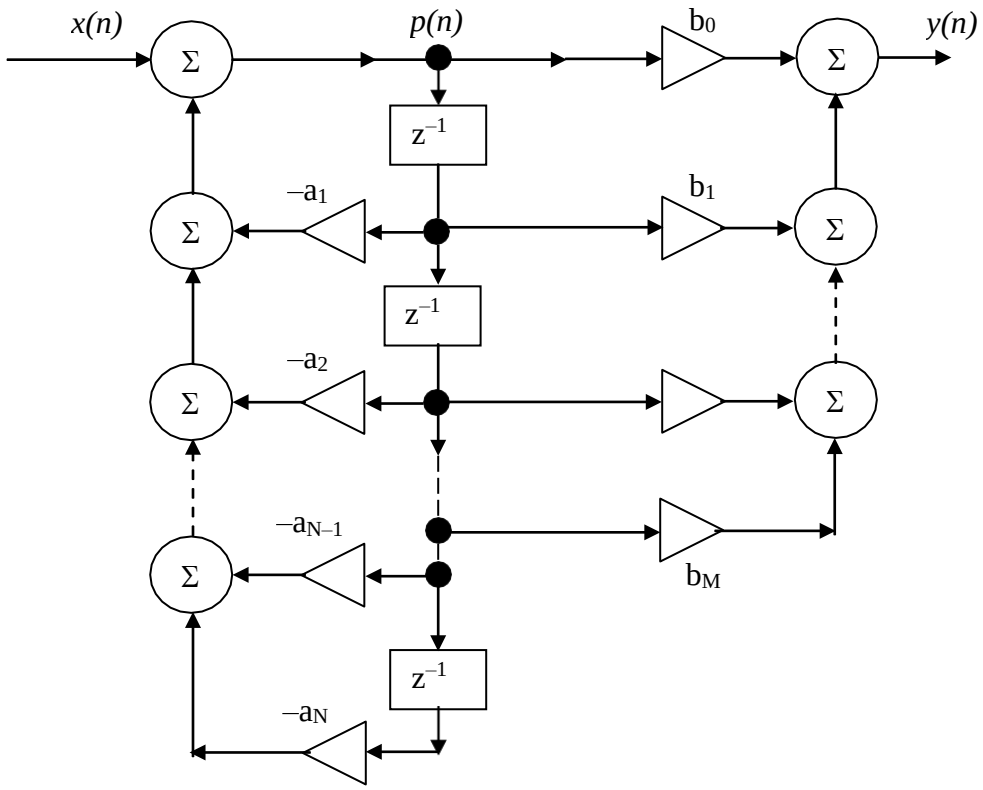
Cross-multiplying, taking the inverse z-transform of the above two and rearranging, we have

$$p(n) = x(n) - \sum_{k=1}^{N} a_k\, p(n-k), \text{ and} \qquad y(n) = \sum_{r=0}^{M} b_r\, p(n-r)$$

The two equations are realized as below:



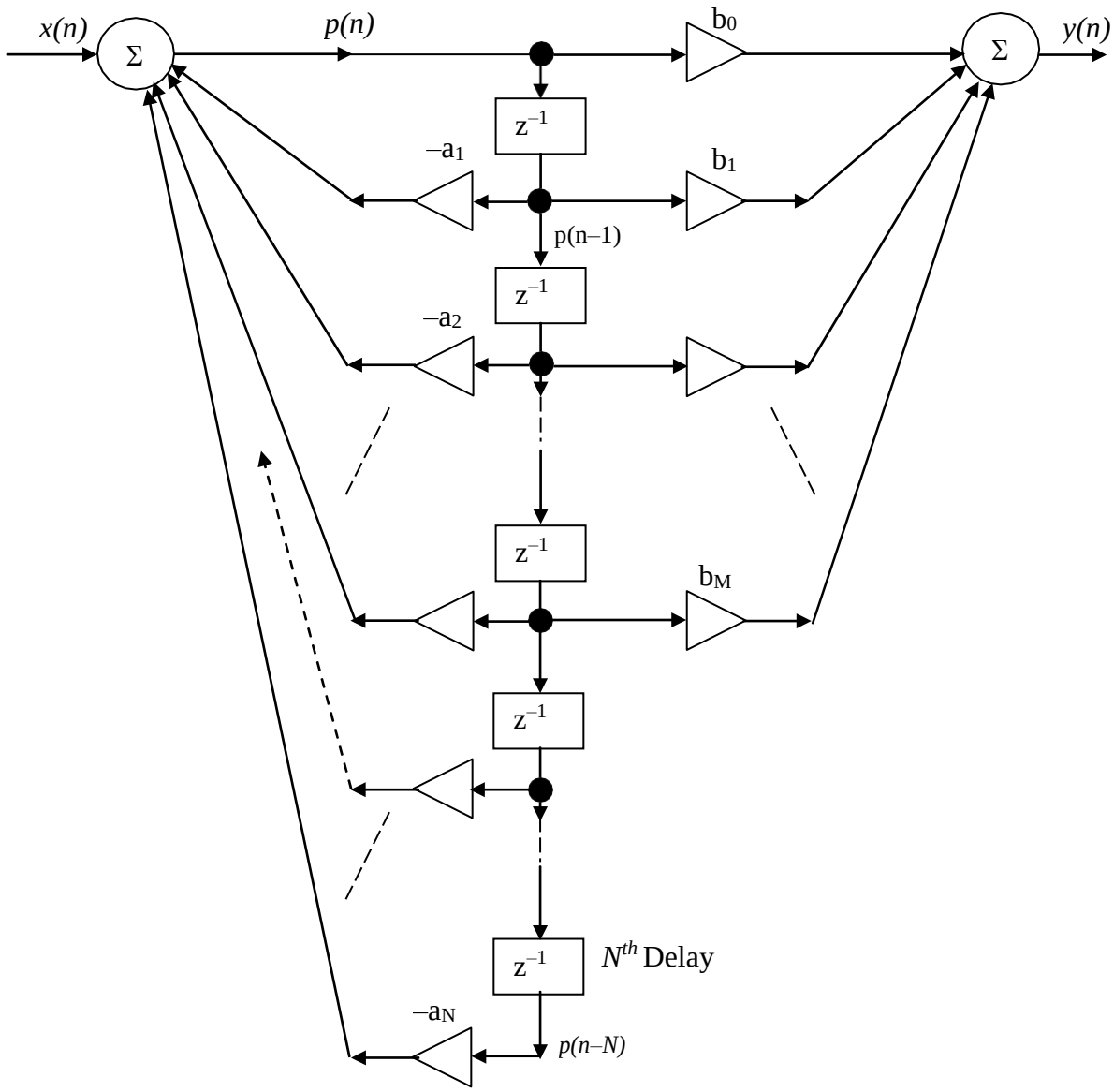
The two branches of delay elements in the middle of the above block diagram can be replaced by just one branch containing either *N* or *M* (whichever is larger) delay elements, resulting in the Direct Form II shown below:



(Aside) This diagram of the filter structure is properly called a *block diagram*. There is another representation called the *signal flow graph*. See below for signal flow graphs and transposed structures.

In the above diagram each column of adders on each side can be replaced by a single adder resulting in the more familiar form shown below. There are now only two adders.

Direct Form II



The number of delay elements = max $\{N, M\}$ – this is the minimum possible, hence called *a* **canonic form**. The multipliers are the actual coefficients from the difference equation. Hence this is also a direct form.

The numbers of multipliers and adders are also the minimum possible, but this does not mean that it is the best realization from other considerations like immunity to round off and quantization errors.

The difference equations are:

$p(n) = x(n) - a_1\, p(n-1) - a_2\, p(n-2) - \ldots - a_N\, p(n-N)$, and

$y(n) = b_0\, p(n) + b_1\, p(n-1) + \ldots + b_M\, p(n-M)$

The above equations show that in order to generate $y(n)$ we need the present value of $x(.)$ and N (or M or whichever is larger) past values of $p(.)$. This requires N storage locations not counting the present value of $x(.)$. We also see that the number of multiplications = $N+M+1$, and number of additions = $N+M$.

Comparing the difference equations of Direct Forms I and II:

- To compute $y(n)$ in DF I we need the past N outputs, the present input, and the past M inputs.
- To compute $y(n)$ in DF II we need the N (or M) values of $p(n-k)$ for $k = 1,2, \ldots, N$, and the present input.

This illustrates the concept of the state of a system.

Definition The **state of a system** is the minimum information required that along with the input allows the determination of the output.

From the above discussion the N (or M) values $p(n-k), k = 1, 2, \ldots, N$, make up the state of the system.

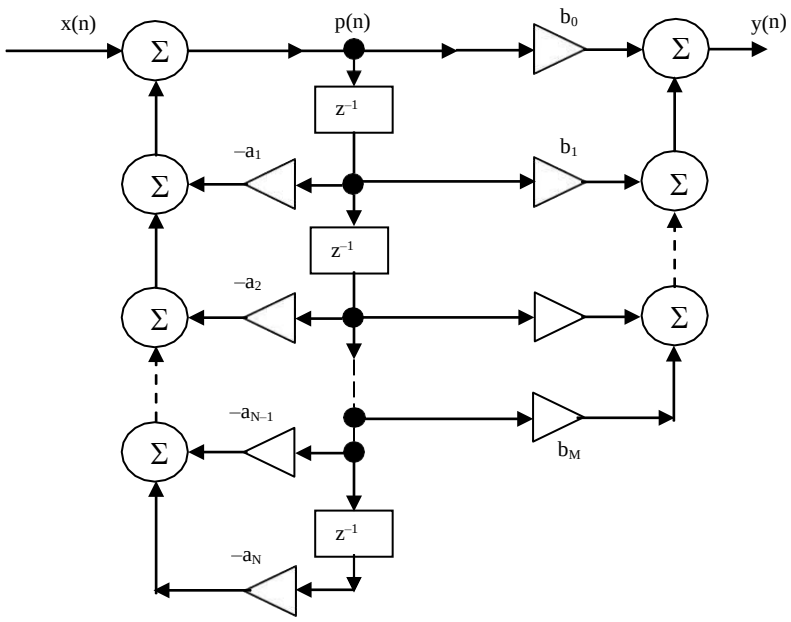
Comparison of Direct Forms I and II

	Direct Form I	Direct Form II
No. of multiplications	$N+M+1$	$N+M+1$
No. of additions	$N+M$	$N+M$
No. of storage locations	$N+M$	N
State	$y(n-k)$, $k = 1$ to N $x(n-k)$, $k = 1$ to M	$p(n-k)$, $k = 1$ to N

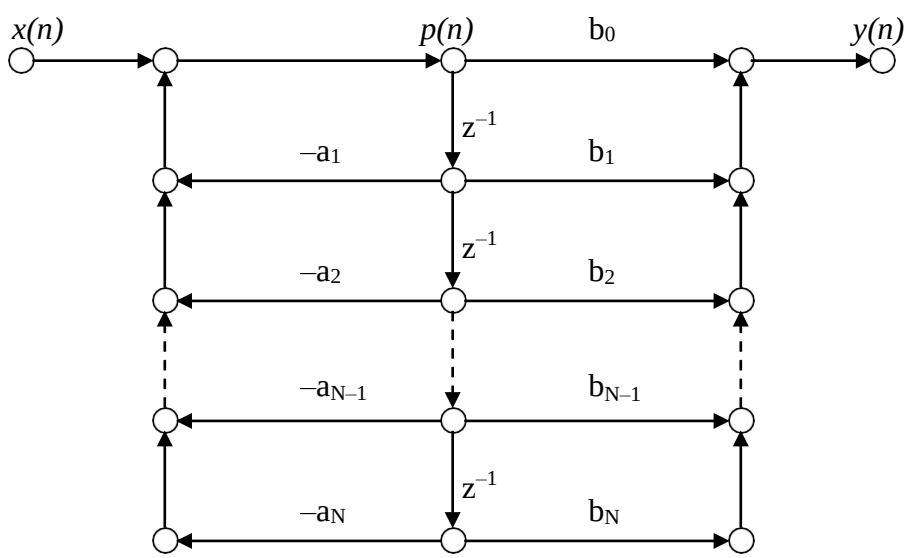
Signal flow graph A signal flow graph representation of a difference equation is essentially the same as its *block diagram* representation, except for a few notational differences. The signal flow graph is a network of directed *branches* that connect at *nodes*. Associated with each node is a variable or node value. Each branch has an input signal and an output signal. In a linear signal flow graph the output of a branch is a linear transformation (such as a constant or a delay shown alongside the branch) of the input to the branch. Sometimes this branch transmittance is unity and not explicitly shown. By definition, the value at each node in a graph is the sum of the outputs of all the branches entering the node.

Source nodes have no entering branches and *sink nodes* have only entering branches.

We repeat below the Direct Form II block diagram developed earlier.



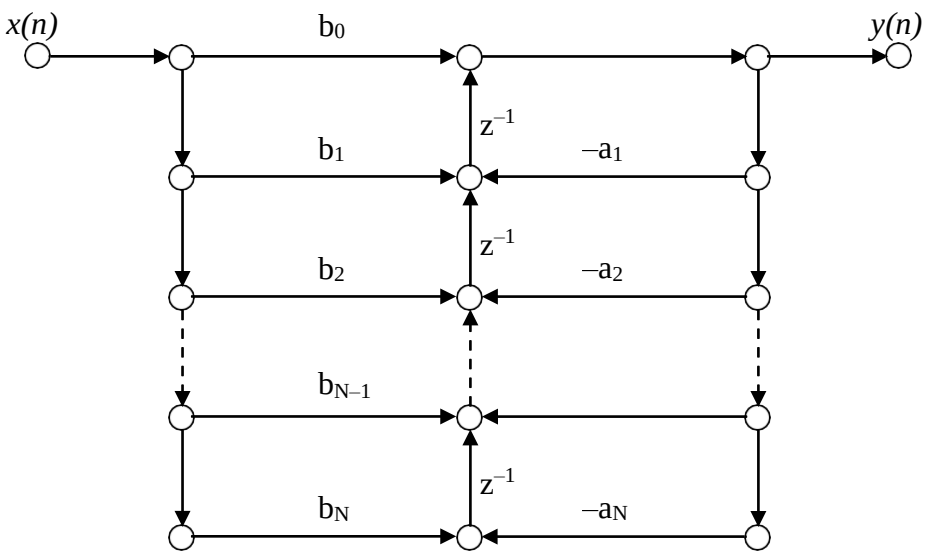
The signal flow graph corresponding to the above structure is shown below. With regard to the "*a*" and "*b*" coefficients we have arbitrarily taken $M = N$, but M and N are independent of each other. In this figure the bottom-most right and left nodes are unnecessary.



Transposition (Flow graph reversal) leaves the overall system function between input and output unchanged and leads to a set of transposed system structures that provide some useful alternatives.

Transposition of a flow graph is accomplished by reversing the directions of all branches while keeping the branch transmittances as they were and reversing the roles of the input and output so that source nodes become sink nodes and vice versa. Clearly, if a signal flow graph configuration is transposed, the number of delay branches and the number of coefficients remain the same. Thus a canonic structure remains canonic in the process of transposition.

The result of applying the transposition theorem to the signal flow graph of the direct form II structure (and reorienting the diagram so that the input and output still appear on the left and right sides, respectively) is shown below. This will be referred to as the "*direct form II transposed*".



Comparison of the direct form II with its transposed version shows that

- The direct form II structure implements the poles first and then the zeros
- The transposed direct form II structure implements the zeros first and then the poles

These differences can become important in the presence of quantization in finite precision digital implementations or in the presence of noise in discrete-time analog implementations.

When the transposition theorem is applied to cascade or parallel structures, the individual second-order systems are replaced by transposed structures.

Example 1.11.1 Develop a canonic direct form realization of the transfer function

$$H(z) = \frac{6z^5 + 8z^3 - 4}{2z^5 + 6z^4 + 10z^3 + 8z}$$

Solution Write numerator and denominator as polynomials in negative powers of z with the leading term (a_0) in the denominator equal to 1

$$H(z) = \frac{z^5(6 + 8z^{-2} - 4z^{-5})}{z^5(2 + 6z^{-1} + 10z^{-2} + 8z^{-4})} = \frac{(6 + 8z^{-2} - 4z^{-5})}{(2 + 6z^{-1} + 10z^{-2} + 8z^{-4})}$$

$$= \frac{(6 + 8z^{-2} - 4z^{-5})}{2(1 + 3z^{-1} + 5z^{-2} + 4z^{-4})} = \frac{3 + 4z^{-2} - 2z^{-5}}{1 + 3z^{-1} + 5z^{-2} + 4z^{-4}}$$

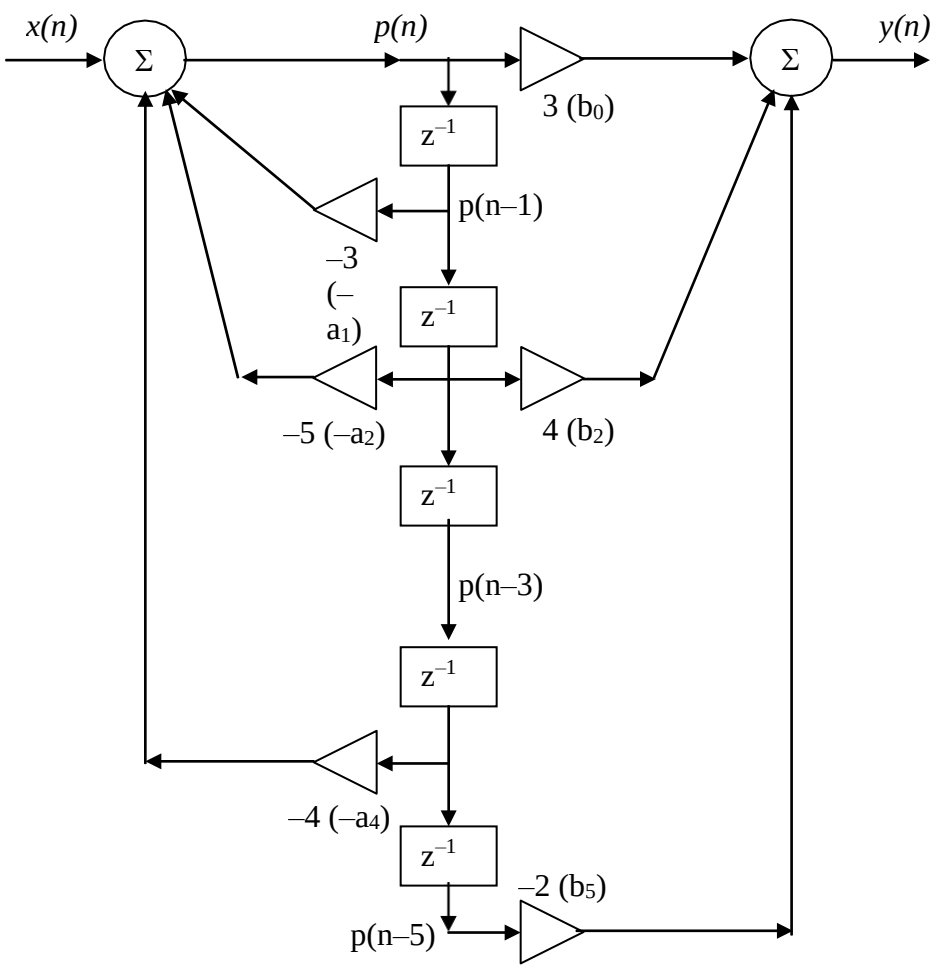
Making the following comparison with the standard notation

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3} + b_4 z^{-4} + b_5 z^{-5}}{1 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3} + a_4 z^{-4}} = \frac{3 + 4z^{-2} - 2z^{-5}}{1 + 3z^{-1} + 5z^{-2} + 4z^{-4}}$$

we identify the following parameters:

$b_0 = 3$, $b_1 = 0$, $b_2 = 4$, $b_3 = 0$, $b_4 = 0$, $b_5 = -2$

$a_1 = 3$, $a_2 = 5$, $a_3 = 0$, $a_4 = 4$



By inspection of the above structure we can write the difference equations

$p(n) = x(n) - 3\,p(n-1) - 5\,p(n-2) - 4\,p(n-4)$ and

$y(n) = 3\,p(n) + 4\,p(n-2) - 2\,p(n-5)$

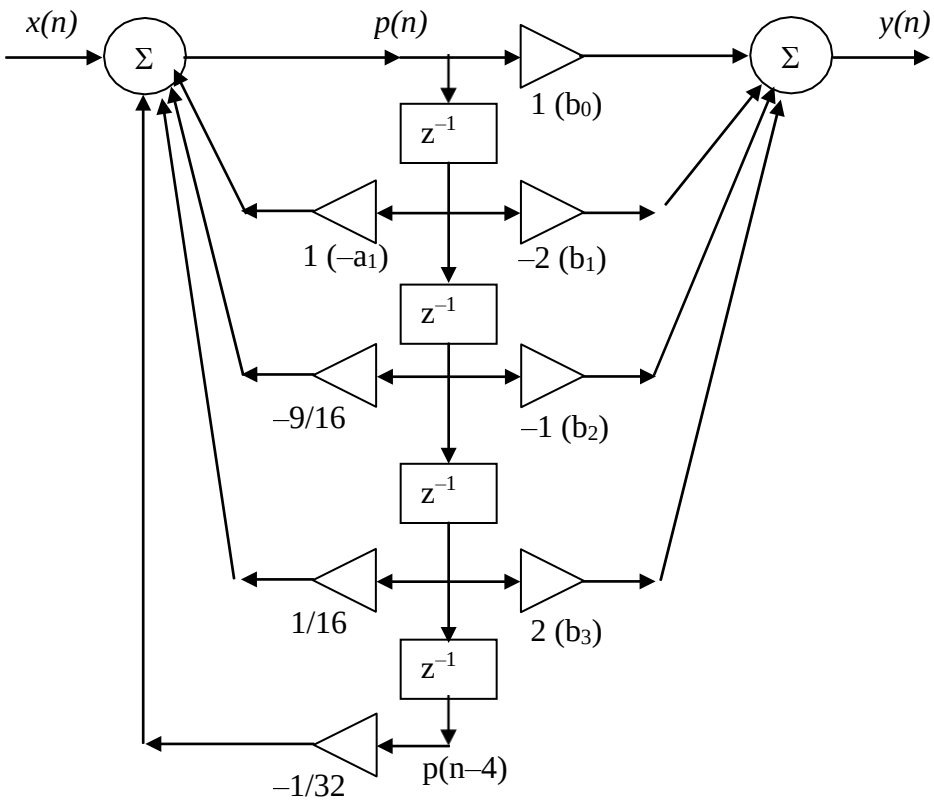
Example 1.11.2 [Ludeman, 5.1] A system is specified by its transfer function as

$$H(z) = \frac{(z-1)(z-2)(z+1)}{\left[z-\left(\frac{1}{2}+j\frac{1}{2}\right)\right]\left[z-\left(\frac{1}{2}-j\frac{1}{2}\right)\right]\left[z-\frac{1}{4}\right]\left[z+\frac{1}{4}\right]}$$

Realize the system in the following forms: (a) Direct Form I, and (b) Direct Form II.

Solution We need to express *H(z)* as a ratio of polynomials in negative powers of *z* with the leading term (a_0) in the denominator equal to 1. Multiplying out the factors in the numerator and denominator and rearranging

$$H(z) = \frac{(z^2-1)(z^2-2z)}{\left[z^2-z+\frac{1}{2}\right]\left[z^2+16\right]} = \frac{1-2z^{-1}-z^{-2}+2z^{-3}}{1-z^{-1}+\frac{9}{16}z^{-2}-\frac{1}{16}z^{-3}+\frac{1}{32}z^{-4}}$$



By inspection of the above structure we can write the difference equations

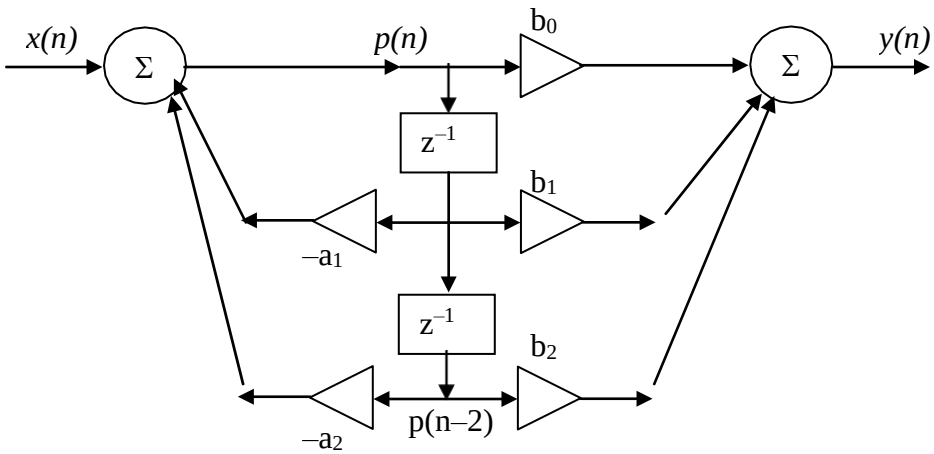
p(n) = ?
y(n) = ?

Biquadratic section When $M = N = 2$ we have the biquadratic section, an important building block. Thus $H(z)$ is a ratio of two quadratics in z^{-1} given by

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

The corresponding DF II structure is shown below with the difference equations.

Biquadratic section – DF II



$$p(n) = x(n) - a_1\, p(n-1) - a_2\, p(n-2)$$
$$y(n) = b_0\, p(n) + b_1\, p(n-1) + b_2\, p(n-2)$$

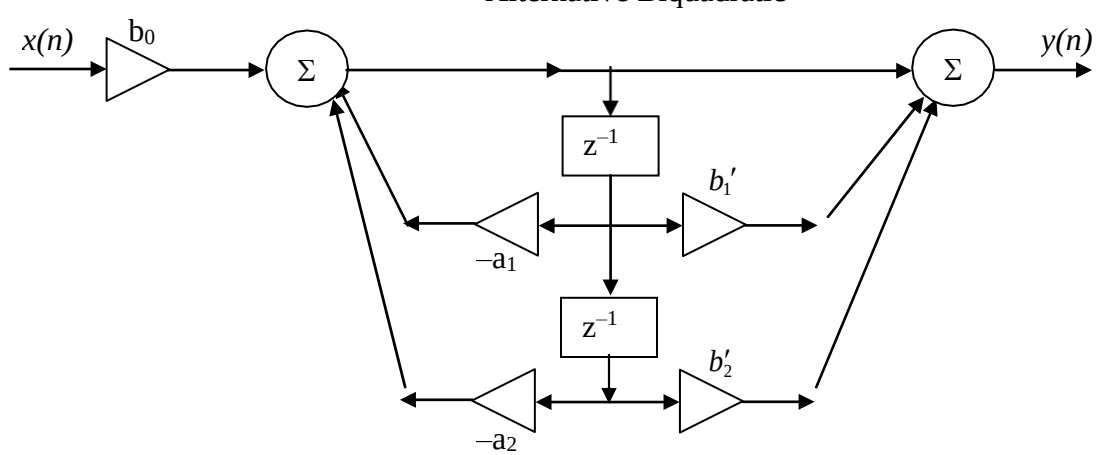
The biquadratic may also be implemented in the DF I with the difference equation:

DF I: $y(n) = b_0\, x(n) + b_1\, x(n-1) + b_2\, x(n-2) - a_1\, y(n-1) - a_2\, y(n-2)$

Shown below is an alternative realization of the biquadratic made possible by factoring out the coefficient b_0. This is more convenient for amplitude scaling by tuning the value of b_0.

$$H(z) = \frac{b_0 (1 + b_1' z^{-1} + b_2' z^{-2})}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

Alternative Biquadratic



As an exercise label the signal out of the left adder as $w(n)$ and write down the difference equations.

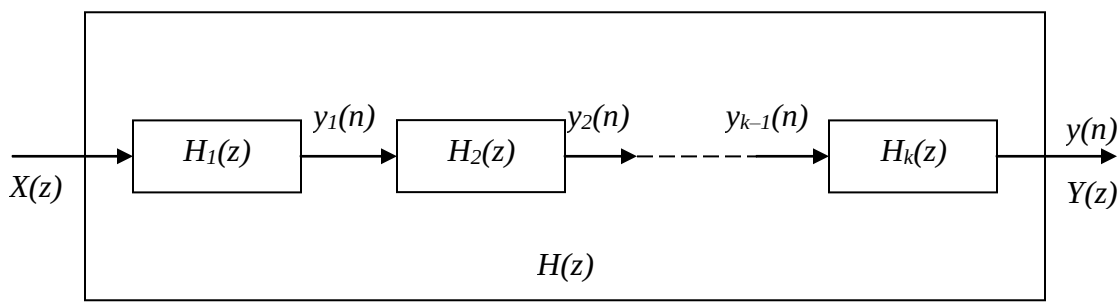
Cascade realization of IIR filters Many different realizations exist depending on how we choose to write and rearrange the given transfer function. Two very important ways of decomposing the transfer function are the *cascade* and *parallel* decompositions.

In the cascade realization $H(z)$ is broken up into a product of transfer functions H_1, H_2, …, H_k, each a rational expression in z^{-1} as follows:

$$\frac{Y(z)}{X(z)} = H(z) = H_k(z)\, H_{k-1}(z) \ldots H_2(z)\, H_1(z)$$

so that $Y(z)$ can be written as

$$Y(z) = H_k(z)\, H_{k-1}(z) \ldots H_2(z)\, H_1(z)\, X(z)$$



Although $H(z)$ could be broken up in many different ways, the most common cascade realization is to require each of the k product H's to be a *biquadratic section.* In many cases the design procedure yields a product of biquadratic expressions so no further work is necessary to put $H(z)$ in the required form. The product terms $H_i(z)$ could take various forms, depending on the actual problem. Some possible forms are

$$H_i(z) = \frac{b_0}{1 + a_1 z^{-1} + a_2 z^{-2}}, \qquad H_i(z) = \frac{b_0 + b_1 z^{-1}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

$$H_i(z) = b_0 + b_1 z^{-1} + b_2 z^{-2}, \qquad H_i(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1}}$$

$$H_i(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \qquad \text{(Biquadratic)}$$

$$H_i(z) = \frac{b_0 + b_1 z^{-1}}{1 + a_1 z^{-1}} \qquad \text{(Bilinear)}$$

Each of the $H_i(z)$ could then be realized using either the direct form I or II.

Different structures are obtained by changing the ordering of the sections and by changing the pole-zero pairings. In practice due to the finite word-length effects, each such cascade realization behaves differently from the others.

Example 1.11.3 Develop two different cascade canonic realizations of the following causal IIR transfer function

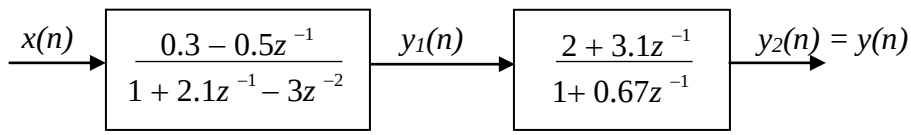
$$H(z) = \frac{z(0.3z - 0.5)(2z + 3.1)}{(z^2 + 2.1z - 3)(z + 0.67)}$$

Solution Write in terms of negative powers of z:

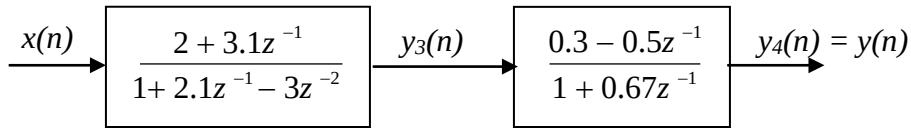
$$H(z) = \frac{z\, z^2 (0.3 - 0.5z^{-1})(2 + 3.1z^{-1})}{z^3 (1 + 2.1z^{-1} - 3z^{-2})(1 + 0.67z^{-1})} = \frac{(0.3 - 0.5z^{-1})(2 + 3.1z^{-1})}{(1 + 2.1z^{-1} - 3z^{-2})(1 + 0.67z^{-1})}$$

Two (of several) different cascade arrangements, based on how the factors are paired, are shown below in block diagram form. Note that the intermediate signal $y_1(n)$ is different from the intermediate signal $y_3(n)$.

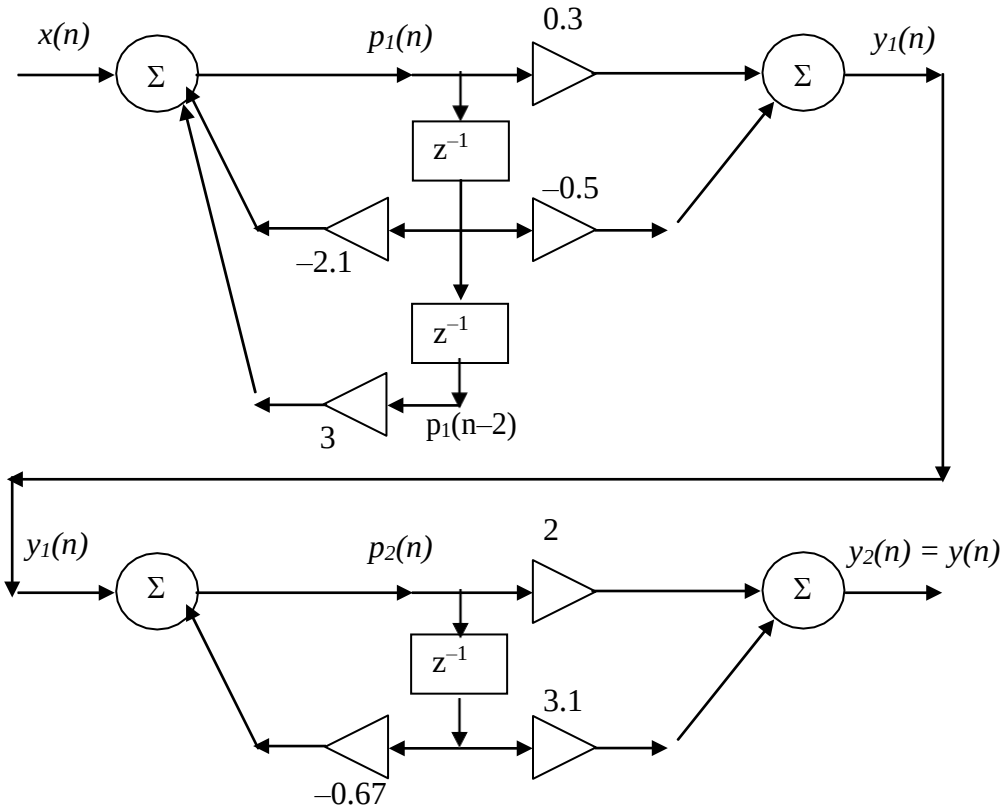
Cascade – *A*



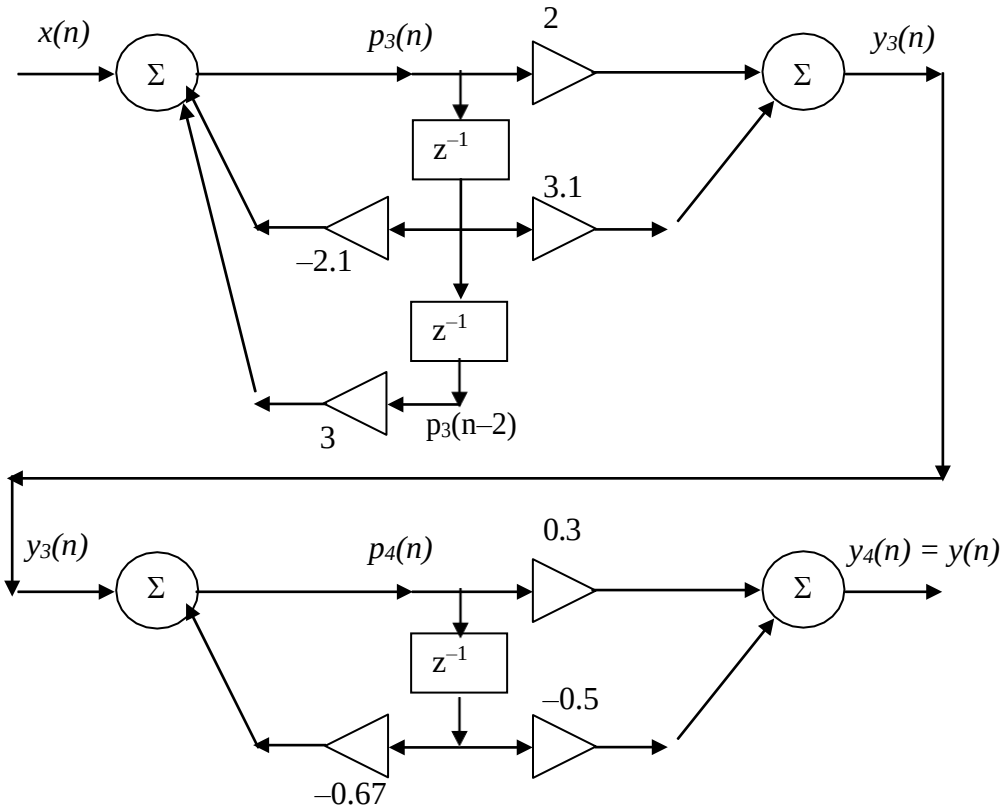
Cascade – *B*



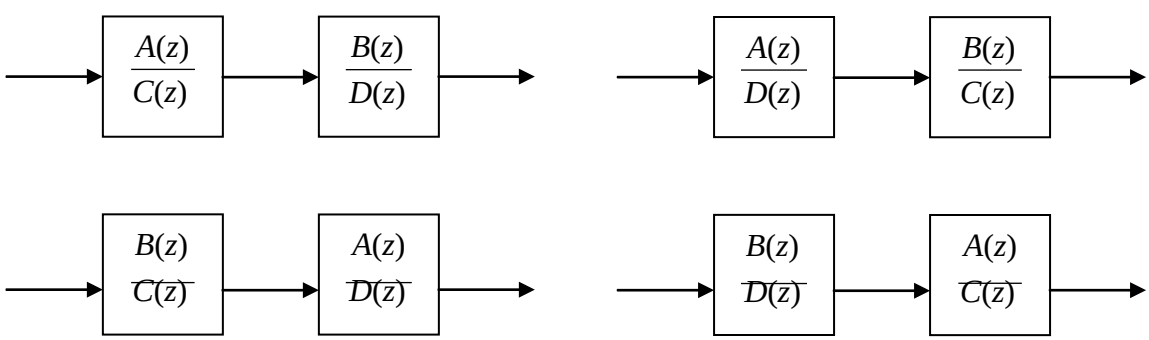
Cascade *A* is shown below using the direct form II for each block separately:



Cascade *B* is shown below using the direct form II for each block separately:



Note in this example that if $H(z) = \dfrac{A(z)\,B(z)}{C(z)\,D(z)}$, then depending on the pole-zero pairings and the sequence order of the blocks in the cascade we can have 4 different implementations (structures). These are equivalent from input to output though not at the intermediate point between the blocks. Moreover the quadratic ($z^2 + 2.1z - 3$) has real roots and so can be split into two factors each of which can be combined with the other factor ($z + 0.67$) in the denominator. This results in more than the 4 structures shown here.



Example 1.11.4 [Ludeman, 5.1] A system is specified by its transfer function as

$$H(z) = \frac{(z-1)(z-2)(z+1)\,z}{\left[z-\left(\frac{1}{2}+j\frac{1}{2}\right)\right]\left[z-\left(\frac{1}{2}-j\frac{1}{2}\right)\right]\left[z-\left(\frac{1}{4}+j\frac{1}{4}\right)\right]\left[z-\left(\frac{1}{4}-j\frac{1}{4}\right)\right]}$$

Realize the system as a cascade of two biquadratic sections.

Solution In the numerator all the zeros are real-valued, so any two factors can be combined to produce a quadratic; thus there are *C(4, 2)* ways of doing this. But in the denominator a pair of factors must be combined so as to produce real-valued coefficients, i.e., combine two factors with complex conjugate poles.

In general, whether in the numerator or denominator, a pair of factors associated with a pair of complex conjugate roots must be combined so that the resulting quadratic expression will have real coefficients.

Parallel realization of IIR filters The transfer function H(z) is written as a sum of transfer functions $H_1(z)$, $H_2(z)$, …, $H_k(z)$ obtained by partial fraction expansion:

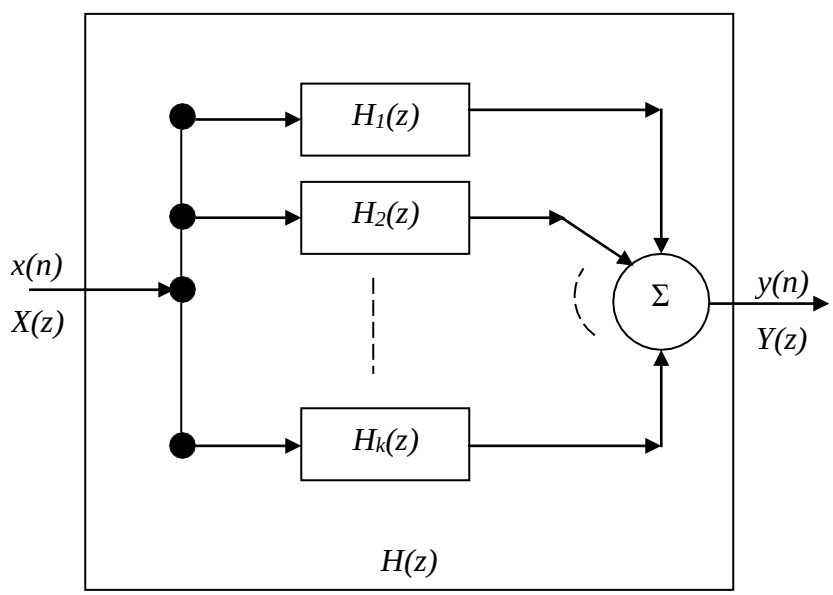
$$\frac{Y(z)}{X(z)} = H(z) = H_1(z) + H_2(z) + \ldots + H_{k-1}(z) + H_k(z)$$

Thus

$$Y(z) = H(z)\,X(z) = [H_1(z) + H_2(z) + \ldots + H_{k-1}(z) + H_k(z)]\,X(z)$$
$$= H_1(z)\,X(z) + H_2(z)\,X(z) + \ldots + H_{k-1}(z)\,X(z) + H_k(z)\,X(z)$$

and is shown in block diagram fashion below. Note that the outputs $y_1(n)$, $y_2(n)$, …, $y_k(n)$ are independent of each other; they are not coupled as in the case of the cascade structure.

Based on whether *H(z)/z* or *H(z)* is the starting point for partial fractions we have parallel forms I and II (S. K. Mitra). Both of these methods are illustrated below.



Parallel Form I (This corresponds to expanding *H(z)/z* instead of *H(z)* into partial fractions). This structure arises when the $H_i(z)$ are all selected to be of the form

$$H_i(z) = \frac{b_0 + b_1 z^{-1}}{1 + a_1 z^{-1} + a_2 z^{-2}} = \frac{b_0 z^2 + b_1 z}{z^2 + a_1 z + a_2} = \frac{z(b_0 z + b_1)}{z^2 + a_1 z + a_2}$$

where the quadratic terms are used for each pair of complex conjugate poles so that the coefficients of the corresponding $H_i(z)$ will be real. Each $H_i(z)$ is then realized as either a DF I or DF II. Special cases of $H_i(z)$ are

$$H_i(z) = b_0, \quad H_i(z) = \frac{b_0}{1+a_1 z^{-1}} = \frac{b_0 z}{z+a_1}, \quad H_i(z) = b_1 z^{-1}$$

Parallel Form II (This corresponds to expanding $H(z)$ directly into partial fractions). This structure arises when the $H_i(z)$ are all selected to be of the form

$$H_i(z) = \frac{b_1 z^{-1} + b_2 z^{-2}}{1+a_1 z^{-1} + a_2 z^{-2}} = \frac{b_1 z + b_2}{z^2 + a_1 z + a_2}$$

Example 4.11.5 Obtain the parallel realization for

$$H(z) = \frac{8z^3 - 4z^2 + 11z - 2}{(z-(1/4))(z^2 - z + (1/2))}$$

Solution For the **Parallel Form I** we expand $H(z)/z$. Note that in the denominator the factor $(z^2 - z + (1/2))$ represents a complex conjugate pair of poles at $((1/2) \pm j(1/2))$.

$$\frac{H(z)}{z} = \frac{8z^3 - 4z^2 + 11z - 2}{z(z-(1/4))(z^2 - z + (1/2))} = \frac{A}{z} + \frac{B}{(z-(1/4))} + \frac{Cz + D}{(z^2 - z + (1/2))}$$

$$A = \left.\frac{8z^3 - 4z^2 + 11z - 2}{(z-(1/4))(z^2 - z + (1/2))}\right|_{z=0} = \frac{-2}{(-1/4)(1/2)} = 16$$

$$B = \left.\frac{8z^3 - 4z^2 + 11z - 2}{z(z^2 - z + (1/2))}\right|_{z=1/4} = \ldots = \ldots = 8$$

To determine C and D

$$\frac{8z^3 - 4z^2 + 11z - 2}{z(z-(1/4))(z^2 - z + (1/2))}$$

$$= \frac{A(z-(1/4))(z^2 - z + (1/2)) + Bz(z^2 - z + (1/2)) + (Cz + D)z(z-(1/4))}{z(z-(1/4))(z^2 - z + (1/2))}$$

Putting $A = 16$ and $B = 8$, and equating the numerators on both sides

$$8z^3 - 4z^2 + 11z - 2$$

$$= 16(z-(1/4))(z^2 - z + (1/2)) + 8z(z^2 - z + (1/2)) + (Cz + D)z(z-(1/4))$$

Equating the coefficients of like powers of z on both sides we have

z^3: $8 = 16 + 8 + C \quad \rightarrow C = -16$

z^0: $-2 = 16(-1/4)(1/2)$ which is an identity $\quad \rightarrow$ doesn't help

z^1: $11 = 16(1/2) + 16(-1/4)(-1) + 8(1/2) + D(-1/4) \quad \rightarrow D = 20$

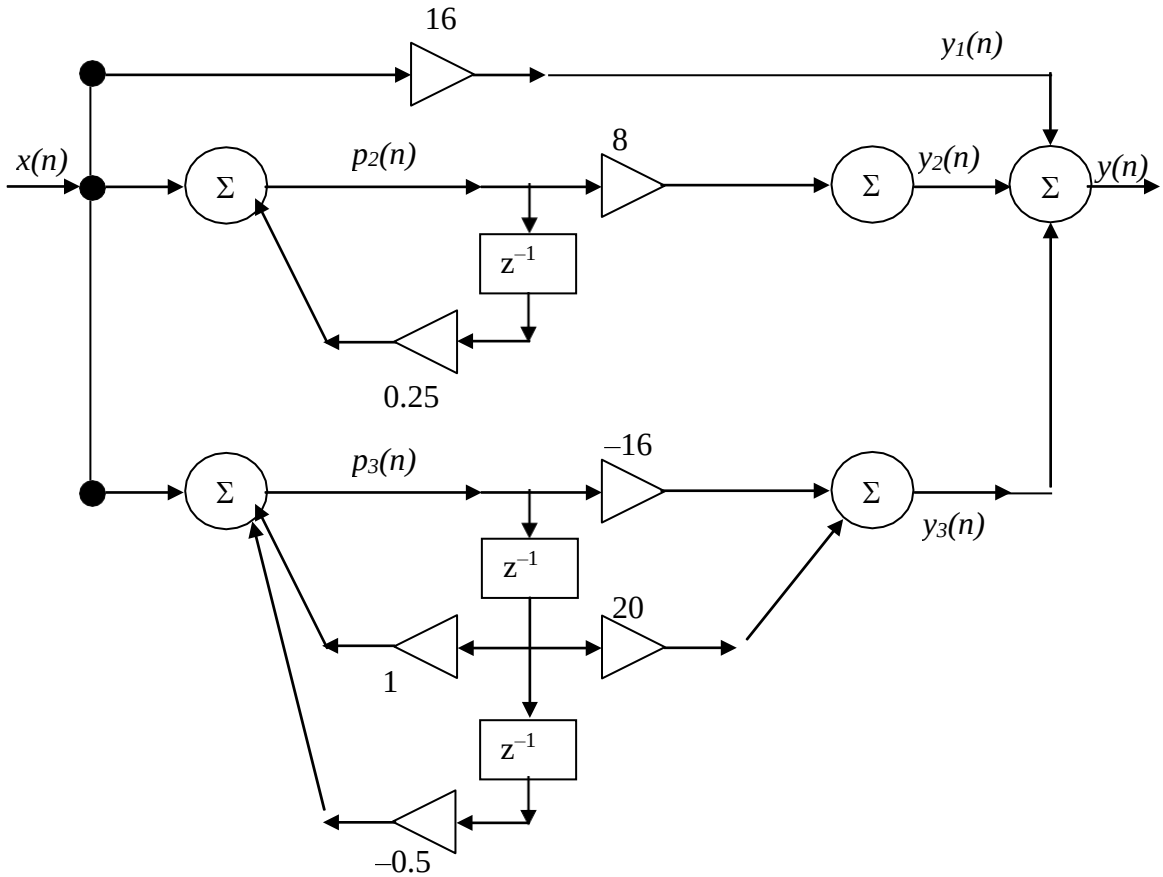
Therefore we have

$$\frac{H(z)}{z} = \frac{16}{z} + \frac{8}{(z-(1/4))} + \frac{(-16)z + 20}{(z^2 - z + (1/2))}$$

$$H(z) = 16 + \frac{8z}{(z-(1/4))} + \frac{z(20 - 16z)}{(z^2 - z + (1/2))}$$

$$H(z) = 16 + \frac{8}{1 - 0.25z^{-1}} + \frac{-16 + 20z^{-1}}{1 - z^{-1} + 0.5z^{-2}} = H_1(z) + H_2(z) + H_3(z)$$

The corresponding parallel form I diagram is shown below.



HW Identify other signals on the diagram as needed and write down the implementation equations in full.

$$p_2(n) = x(n) + 0.25\, p_2(n-1),$$
$$y_2(n) = 8\, p_2(n)$$

$$p_3(n) = x(n) + 1\, p_3(n-1) - 0.5\, p_3(n-2),$$
$$y_3(n) = -16\, p_3(n) + 20\, p_3(n-1)$$

$$y(n) = y_1(n) + y_2(n) + y_3(n)$$

For the **Parallel Form II** we expand $H(z)$

$$H(z) = \frac{8z^3 - 4z^2 + 11z - 2}{\left(z-(1/4)\right)\left(z^2 - z + (1/2)\right)} = \frac{8z^3 - 4z^2 + 11z - 2}{z^3 - 1.25z^2 + 0.75z - 0.125}$$

By long division we reduce the degree of the numerator by 1

Long Division

		8	←Quotient
Denominator→	$z^3 - 1.25z^2 + 0.75z - 0.125$	$8z^3 - 4z^2 + 11z - 2$	←Numerator
		$8z^3 - 10z^2 + 6z - 1$	
		$6z^2 + 5z - 1$	←Remainder

$$H(z) = 8 + \frac{6z^2 + 5z - 1}{z^3 - 1.25z^2 + 0.75z - 0.125} = 8 + \frac{6z^2 + 5z - 1}{(z - (1/4))(z^2 - z + (1/2))}$$

The proper fraction part can now be expanded into partial fractions. Let

$$\frac{6z^2 + 5z - 1}{(z - (1/4))(z^2 - z + (1/2))} = \frac{A}{(z - (1/4))} + \frac{Bz + C}{(z^2 - z + (1/2))}$$

$$A = \left.\frac{6z^2 + 5z - 1}{(z^2 - z + (1/2))}\right|_{z = 1/4} = \ldots = \frac{\ldots}{\ldots} = 2$$

To determine B and C

$$\frac{6z^2 + 5z - 1}{(z - (1/4))(z^2 - z + (1/2))} = \frac{A(z^2 - z + (1/2)) + (Bz + C)(z - (1/4))}{(z - (1/4))(z^2 - z + (1/2))}$$

Putting $A = 2$, and equating the numerators on both sides

$$6z^2 + 5z - 1 = 2(z^2 - z + (1/2)) + (Bz + C)(z - (1/4))$$

Equating the coefficients of like powers of z on both sides we have

z^2: $\quad 6 = 2 + B \quad \rightarrow B = 4$

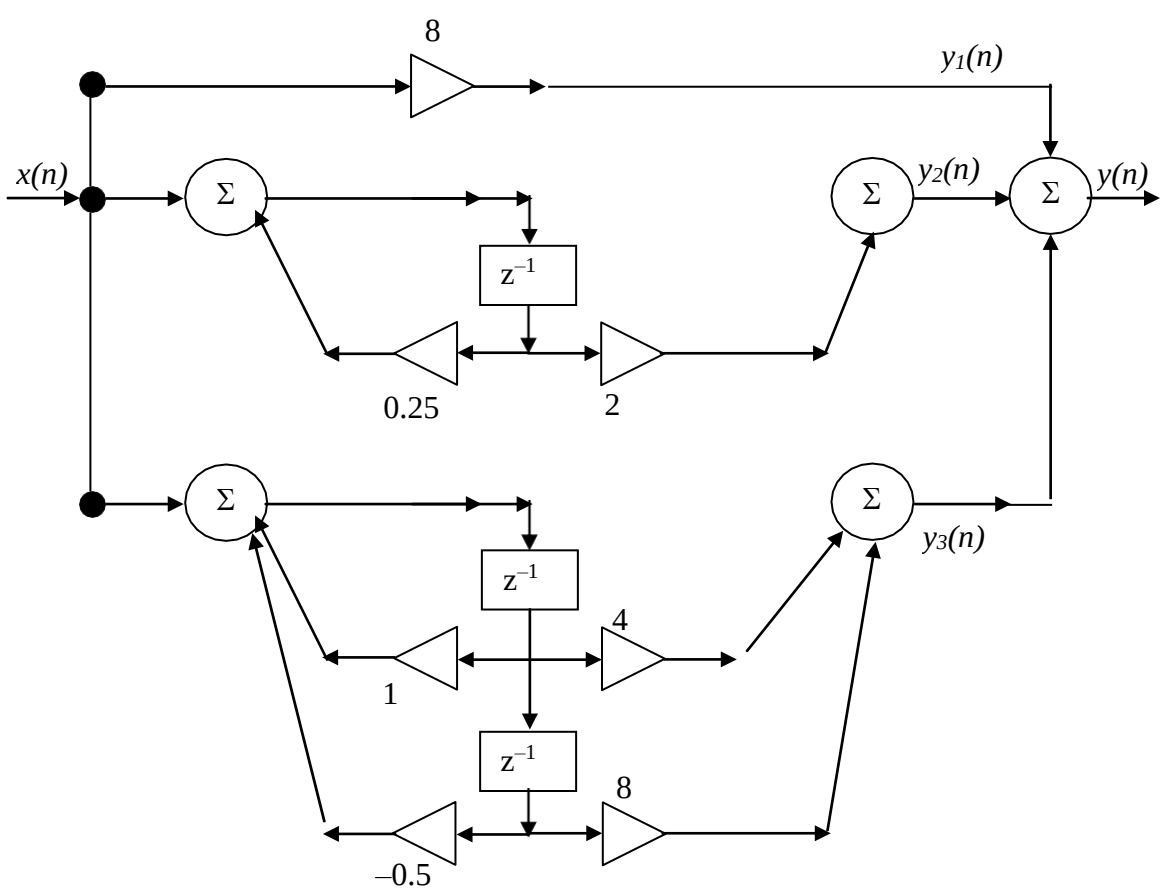
z^0: $\quad -1 = 2\,(1/2) + C\,(-1/4) \quad \rightarrow C = 8$

Therefore we have

$$H(z) = 8 + \frac{2}{(z - (1/4))} + \frac{4z + 8}{(z^2 - z + (1/2))}$$

$$H(z) = 8 + \frac{2z^{-1}}{1 - 0.25z^{-1}} + \frac{4z^{-1} + 8z^{-2}}{1 - z^{-1} + 0.5z^{-2}} = H_1(z) + H_2(z) + H_3(z)$$

The corresponding parallel form II diagram is shown below.



HW Identify other signals on the diagram as needed and write down the implementation equations in full.

Example 4.11.6 [Ludeman, 5.1] A system is specified by its transfer function as

$$H(z) = \frac{(z-1)(z-2)(z+1)\,z}{\left(z-\left(\frac{1}{2}+j\frac{1}{2}\right)\right)\left(z-\left(\frac{1}{2}-j\frac{1}{2}\right)\right)\left(z-\frac{1}{4}+j\frac{1}{4}\right)}$$

Implement the parallel realization in constant, linear and biquadratic sections.
Solution For the Parallel Form I expand *H(z)/z* and for the Parallel Form II expand *H(z)*. In the denominator keep the complex conjugate roots together.

Realization of FIR filters A causal FIR filter is characterized by its transfer function H(z) given by

$$\frac{Y(z)}{X(z)} = H(z) = \sum_{r=0}^{M} b_r z^{-r} = b_0 + b_1 z^{-1} + \ldots + b_M z^{-M}$$

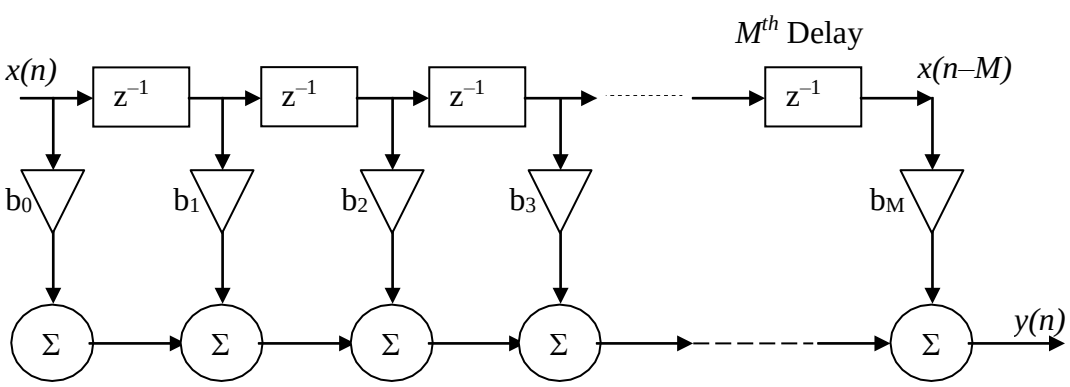
or, by the corresponding difference equation

$$y(n) = \sum_{r=0}^{M} b_r x(n-r) = b_0\, x(n) + b_1\, x(n-1) + b_2\, x(n-2) \ldots + b_M\, x(n-M)$$

Note that some use the notation below with *M* coefficients instead of *M + 1*

$$y(n) = \sum_{r=0}^{M-1} b_r x(n-r) = b_0\, x(n) + b_1\, x(n-1) + b_2\, x(n-2) \ldots + b_{M-1}\, x(n-M+1)$$

We see that the output *y(n)* is a weighted sum of the present and past input values; it does not depend on past output values such as *y(n–1)*, etc. The block diagram is shown below. It is also called a **tapped delay line** or a **transversal filter**.



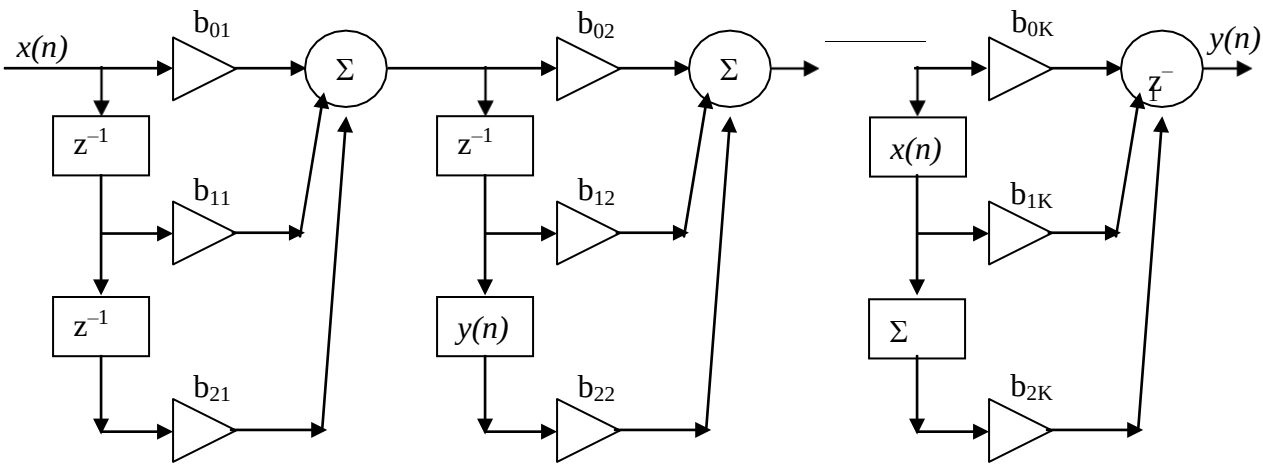
It can be seen that this is the same as the direct form I or II shown earlier for the IIR filter, except that the coefficients a_1 through a_N are zero and $a_0 = 1$; further the delay elements are arranged in a horizontal line. As in earlier diagrammatic manipulation the multipliers can all feed into the rightmost adder and the remaining adders removed.

Other simplifications are possible based on the symmetry of the coefficients $\{b_r\}$, as we shall see in FIR filter design.

Cascade realization of FIR filters The simplest form occurs when the system function is factored in terms of quadratic expressions in z^{-1} as follows:

$$H(z) = \prod_{i=1}^{K} H_i(z) = \prod_{i=1}^{K} \left(b_{0i} + b_{1i} z^{-1} + b_{2i} z^{-2}\right)$$

Selecting the quadratic terms to correspond to the complex conjugate pairs of zeros of $H(z)$ allows a realization in terms of real coefficients b_{0i}, b_{1i} and b_{2i}. Each quadratic could then be realized using the direct form (or alternative structures) as shown below.



Parallel realization of FIR filters These are based on interpolation formulas by Lagrange, Newton, or Hermite methods. In general, these realizations require more multiplications, additions and delays than the others.

The Lattice structure – Introduction

In order to introduce the Lattice structure consider the all-pole IIR filter (b_1 through b_M are zero)

$$H(z) = \frac{b_0}{\sum_{k=0}^{N} a_k z^{-k}} = \frac{b_0}{a_0 + a_1 z^{-1} + a_2 z^{-2} + \square + a_N z^{-N}}$$

We shall take $a_0 = 1$ and $b_0 = 1$ so that

$$H(z) = \frac{1}{1 + \sum_{k=1}^{N} a_k z^{-k}} = \frac{1}{1 + a_1 z^{-1} + a_2 z^{-2} + \square + a_N z^{-N}}$$

We shall write the subscript k on the coefficients a as an index within parentheses: thus a_k becomes $a(k)$; further we shall also incorporate the order N of the filter as a subscript on the coefficients a: thus for an Nth order filter we shall have the set of coefficients $a_N(k)$, with $k = 1$ to N. With this notational refinement the system function goes through the following steps

$$H(z) = \frac{1}{1 + \sum_{k=1}^{N} a(k) z^{-k}} = 1 + a(1) z^{-1} + a(2) z^{-2} + \ldots + a(N) z^{-N}$$

$$H(z) = \frac{1}{1+\sum_{k=1}^{N} a_N(k)\, z^{-k}} = \frac{1}{1+a_N(1)\, z^{-1} + a_N(2)\, z^{-2} + \cdots + a_N(N)\, z^{-N}} = \frac{1}{A_N(z)}$$

where $A_N(z)$ denotes the denominator polynomial.

Using the above notation we shall consider the implementation of a first order filter, that is $N = 1$. We then have

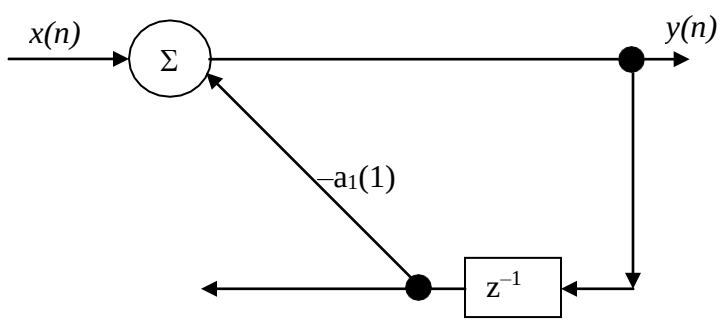
$$H_1(z) = \frac{1}{1+\sum_{k=1}^{1} a_1(k)\, z^{-k}} = \frac{1}{1+a_1(1) z^{-1}} = \frac{1}{A_1(z)}$$

Note that we have also awarded a subscript to the system function H, that is, $H(z)$ is written $H_1(z)$ just to remind ourselves that we are only dealing with a first order system. The difference equation can be written down from

$$H_1(z) = \frac{Y(z)}{X(z)} = \frac{1}{1+\sum_{k=1}^{1} a_1(k) z^{-k}} = \frac{1}{1+a_1(1)\, z^{-1}}$$

$$y(n) = x(n) - a_1(1)\, y(n-1) \qquad or \qquad x(n) = y(n) + a_1(1)\, y(n-1)$$

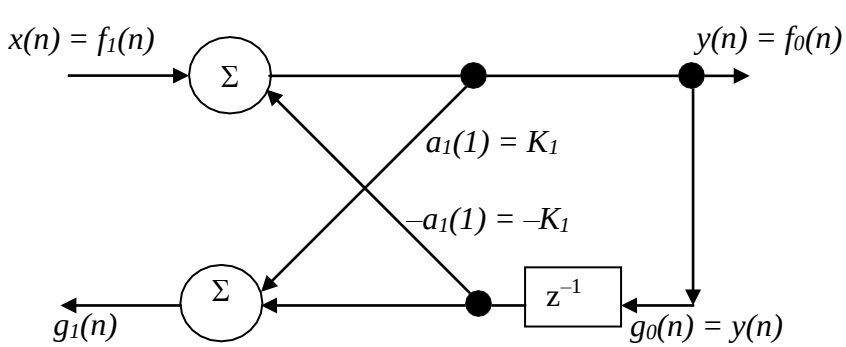
This difference equation is implemented by the following structure (which can be visually verified). Concerning the labeling of the diagram note that $a_1(1)$ is not a signal but a multiplier that multiplies the signal coming out of the delay element and before it reaches the adder (we have previously used a triangular symbol for the multiplier).



An embellished version of the above structure is shown below and is used as a building block in the lattice structure. This being a first order filter the relationship between the direct form coefficient $a_1(1)$ and the **lattice coefficient** K_1 is obvious, that is, $a_1(1) = K_1$. The implementation equations, in terms of the lattice coefficient, are

$f_0(n) = f_1(n) - K_1\, g_0(n-1)$ [which amounts to $y(n) = x(n) - a_1(1)y(n-1)$]
$g_1(n) = K_1\, f_0(n) + g_0(n-1)$ [which amounts to $g_1(n) = K_1\, y(n) + y(n-1)$]

At this point the need for the additional symbols $f(.)$ and $g(.)$ and the equation for $g_1(n)$ is not obvious, but they become more useful as we increase the order of the filter and the relationship between the coefficients of the two structures becomes more involved.

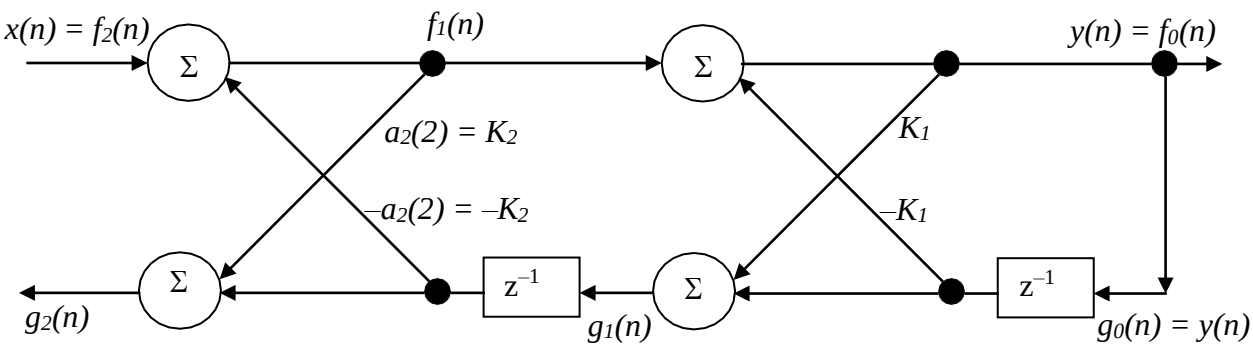


Consider the second order (all-pole) filter ($N = 2$) whose transfer function is

$$H_2(z) = \frac{Y(z)}{X(z)} = \frac{1}{1 + \sum_{k=1}^{2} a_2(k)\, z^{-k}} = \frac{1}{1 + a_2(1)z^{-1} + a_2(2)z^{-2}} = \frac{1}{A_2(z)}$$

$$y(n) = x(n) - a_2(1)\, y(n-1) - a_2(2)\, y(n-2)$$

The corresponding lattice structure is obtained by adding a second stage at the left end of the previous first order structure:



We can write an equation for $y(n)$ in terms of the lattice coefficients K_1 and K_2 and the signal values $x(n)$, $y(n-1)$ and $y(n-2)$:

$$y(n) = x(n) - K_1\,(1+K_2)\, y(n-1) - K_2\, y(n-2)$$

Comparing this equation with the direct form equation for $y(n)$ given above we have the relationship between the direct form and lattice coefficients

$$a_2(1) = K_1\,(1 + K_2) \quad \text{and} \quad a_2(2) = K_2 \qquad a_2(0) = 1$$

Note that we have thrown in a freebie in the form of $a_2(0) = 1$ for future (actually it corresponds to the leading term in the denominator polynomial, $A_2(z)$).

Discrete Fourier series

Properties of discrete Fourier series, DFS representation of periodic sequences, Discrete Fourier transforms, Properties of DFT, Linear convolution of sequences using DFT, Computation of DFT, Relation between z-transform and DFS.

Contents:

Fourier analysis - Recapitulation

(1) The *Fourier series* (FS) of a continuous-time periodic signal, *x(t)*, with fundamental period T_0, is given by the *synthesis equation*

$$x(t) = \sum_{k=-\infty}^{\infty} X_k e^{j2\pi k F_0 t}$$

The Fourier coefficients, X_k, are given by the *analysis equation*

$$X_k = \frac{1}{T_0}\int_{T_0} x(t)\, e^{-j2\pi k F_0 t}\, dt$$

The fundamental frequency, F_0 (Hz), and the period, T_0 (seconds), are related by $F_0 = 1/T_0$.

(2) The *Fourier transform* (FT) of a continuous-time aperiodic signal, *x(t)*, is given by the *analysis equation*

$$X(F) = \int_{-\infty}^{\infty} x(t) e^{-j2\pi F t}\, dt \qquad \text{or} \qquad X(\Omega) = \int_{-\infty}^{\infty} x(t)\, e^{-j\Omega t}\, dt$$

Here Ω and F are analog frequencies, with $\Omega = 2\pi F$. The *inverse Fourier transform* is given by the *synthesis equation*

$$x(t) = \int_{-\infty}^{\infty} X(F)\, e^{j2\pi F t}\, dF \qquad \text{or} \qquad x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(\Omega)\, e^{j\Omega t}\, d\Omega$$

(3) The *Fourier series* (DTFS/DFS) for a discrete-time periodic signal (periodic sequence), *x(n)*, with fundamental period N is given by the *synthesis equation*

$$x(n) = \sum_{k=0}^{N-1} X_k e^{j2\pi k n/N}, \qquad 0 \le n \le N-1$$

The Fourier coefficient X_k are given by the *analysis equation*

$$X_k = \frac{1}{N}\sum_{n=0}^{N-1} x(n)\, e^{-j2\pi k n/N}, \qquad 0 \le k \le N-1$$

This is called the **discrete-time Fourier series (DTFS)** or just **discrete Fourier series (DFS)** for short. The sequence of coefficients, X_k, also is periodic with period N.

These two equations are derived below.

(Note that if the factor $(1/N)$ is associated with *x(n)* rather than with X_k the two DFS equations are identical to the two DFT equations which are derived below in their *standard form*.)

(4) The *Fourier transform* (DTFT) of a finite energy discrete-time aperiodic signal (aperiodic sequence), *x(n)*, is given by the *analysis equation* (some write $X(e^{j\omega})$ instead of $X(\omega)$)

$$X(\omega) = \sum_{n=-\infty}^{\infty} x(n)\, e^{-j\omega n}$$

Certain convergence conditions apply to this analysis equation concerning the type of signal *x(n)*. We shall call this the **discrete-time Fourier transform (DTFT)**. Physically $X(\omega)$ represents the frequency content of the signal *x(n)*. $X(\omega)$ is periodic with period 2π.

The *inverse discrete-time Fourier transform* is given by the *synthesis equation*

$$x(n) = \frac{1}{2\pi}\int_{2\pi} X(\omega)\, e^{j\omega n}\, d\omega$$

The basic difference between the Fourier transform of a continuous-time signal and the Fourier transform of a discrete-time signal is this: For continuous time signals the Fourier transform, and hence the spectrum of the signal, have a frequency range $(-\infty, \infty)$; in contrast, for a discrete-time signal the frequency range of the DTFT is unique over the interval of $(-\pi, \pi)$ or, equivalently, $(0, 2\pi)$.

Since $X(\omega)$ is a periodic function of the frequency variable ω, it has a Fourier series expansion; in fact, the Fourier coefficients are the $x(n)$ values.

Discrete Fourier series

Let $x(n)$ be a real periodic discrete-time sequence of period N. If $x(n)$ can be expressed as a weighted sum of complex exponentials, the response of a linear system to $x(n)$ is easily determined by superposition. By analogy with the Fourier series representation of a periodic *continuous-time* signal, we can expect that we can obtain a similar representation for the periodic *discrete-time* sequence $x(n)$. That is, we seek a representation for $x(n)$ of the form

$$x(n) = \sum_k X_k e^{jk\omega_0 n} \qquad \text{for all } n$$

Here X_k are the Fourier coefficients and $\omega_0 = 2\pi/N$ is the fundamental (digital) frequency (as $\Omega_0 = 2\pi/T_0 = 2\pi F_0$ is in the case of continuous-time Fourier series). With $k\omega_0 = \omega_k = k\left(\frac{2\pi}{N}\right)$, the above is also written

$$k \qquad k \qquad \sum_k X_k e^{jk2\pi n/N}$$

The function $e^{jk2\pi n/N}$ is periodic in k with a periodicity of N and there are only N distinct functions in the set $\left\{e^{jk2\pi n/N}\right\}$ corresponding to $k = 0, 1, 2, ..., N-1$. Thus the representation for $x(n)$ contains only N terms (as opposed to infinitely many terms in the continuous-time case)

$$x(n) = \sum_{k=<N>} X_k e^{jk2\pi n/N}$$

The summation can be done over any N consecutive values of k, indicated by the summation index $k = <N>$. For the most part, however, we shall consider the range $0 \le k \le N-1$, and the representation for $x(n)$ is then written as

$$x(n) = \sum_{k=0}^{N-1} X_k e^{jk2\pi n/N} \text{ for all } n$$

This equation is the **discrete-time Fourier series (DTFS)** or just **discrete Fourier series (DFS)** of the periodic sequence $x(n)$ with coefficients X_k.

The coefficients X_k or $X(k)$ are given by (we skip the algebra – S&S)

$$X_k = \frac{1}{N}\sum_{n=0}^{N-1} x(n)\, e^{-j2\pi kn/N}, \qquad 0 \le k \le N-1 \qquad \text{(B)}$$

Note that the sequence of Fourier coefficients $\{X_k\}$ is periodic with period = N. That is, $X_k = X_{k+N}$. The coefficients can be interpreted to be a sequence of finite length, given by Eq. (B) *for* $k = 0, 1, 2, ..., N-1$ *only* and zero otherwise, or as a periodic sequence defined *for all* k by Eq. (B). Clearly both of these interpretations are equivalent.

Because the Fourier series for discrete-time periodic signals is a finite sum defined entirely by the values of the signal over one period, the series always converges. The Fourier series provides an *exact* alternative representation of the time signal, and issues such as *convergence* or the *Gibbs phenomena* do not arise.

The periodic sequence $X(k)$ has a convenient interpretation as samples on the unit circle, equally spaced in angle, of the z-transform of *one period* of $x(n)$. Let $x_1(n)$ represent one period of $x(n)$. That is,

$$x_1(n) = \begin{cases} x(n), & 0 \le n \le N-1 \\ 0, & \text{otherwise} \end{cases}$$

Then $X_1(z) = \sum_{n=-\infty}^{\infty} x_1(n)\, z^{-n} = \sum_{n=0}^{N-1} x_1(n)\, z^{-n}$, and $X(k) = X^1(z)\big|_{z=e^{j2\pi k/N}}$. This then corresponds to sampling the z-transform $X_1(z)$ at N points equally spaced in angle around the unit circle, with the first such sample occurring at $z = 1$. (Note that the periodic sequence $x(n)$ cannot be represented by its z-transform since there is no value of z for which the z-transform will converge. However, $x_1(n)$ does have a z-transform.)

Properties of discrete Fourier series

Properties of discrete Fourier series (DFS) for periodic sequences The following notation is used:

p = periodic; e = even; o = odd

$W_N = e^{-j2\pi/N}$
Re [.] = Real part of
Im [.] = Imaginary part of
|.| = Magnitude of
Arg (.) = Argument of

The following properties should be noted.

	Sequence	DFS
1	$x_p(n+m)$	$W_N^{-km} X_p(k)$
2	$x_p^*(n)$	$X_p^*(-k)$
3	$x_p^*(-n)$	$X_p^*(k)$

	Sequence	DFS
4	Re $[x_p(n)]$	$X_{pe}(k)$
5	j Im $[x_p(n)]$	$X_{po}(k)$

Example 2.3.1 Show that DFS $\{x_p(n+m)\} = W_N^{-km} X_p(k)$.
Solution We have

$$\text{DFS } \{ x_p(n+m)\} = \sum_{n=0}^{N-1} x_p(n+m) W_N^{kn}$$

Set $n+m = \lambda$ so that $n = \lambda - m$ and the limits $n = 0$ to $N-1$ become $\lambda = m$ to $N-1+m$. Then the RHS becomes

$$= \sum_{\lambda=m}^{N-1+m} x_p(\lambda)\, W_N^{k\lambda} W_N^{-km}$$

Since $x_p(\lambda)$ is periodic with period N the summation can be done over any interval of length N. Thus

$$\text{DFS } \{ x_p(n+m)\} = \sum_{\lambda=0}^{N-1} x_p(\lambda)\, W_N^{k\lambda} W_N^{-km} = W_N^{-km} \sum_{\lambda=0}^{N-1} x_p(\lambda)\, W_N^{k\lambda}$$

$$= W_N^{-km} X_p(k) \qquad \text{QED}$$

Example 2.3.2 Show that DFS $\{ x_p^*(n)\} = X_p^*(-k)$.

Solution We have

$$\text{DFS}\{ x_p^*(n)\} = \sum_{n=0}^{N-1} x_p^*(n)\, W_N^{kn} = \left\{ \left(\sum_{n=0}^{N-1} x_p^*(n) W_N^{kn} \right)^* \right\}^*$$

$$= \left\{ \sum_{n=0}^{N-1} x_p(n)\, W_N^{-kn} \right\}^* = \left\{ X_p(-k) \right\}^* = X_p^*(-k) \qquad \text{QED}$$

Based on the properties above we can show that for a real periodic sequence $x_p(n)$, the following symmetry properties of the discrete Fourier series hold:

1. Re $[X_p(k)]$ = Re $[X_p(-k)]$
2. Im $[X_p(k)]$ = –Im $[X_p(-k)]$
3. $|X_p(k)| = |X_p(-k)|$
4. arg $X_p(k)$ = – arg $X_p(-k)$

The discrete Fourier transform (DFT)

(Omit) The discrete Fourier transform (DFT) derived from the Fourier series The exponential Fourier series of a continuous time periodic signal *x(t)* with fundamental period T_0 is given by the synthesis equation

$$x(t) = \sum_{k=-\infty}^{\infty} X_k\, e^{j2\pi k F_0 t} \qquad < (1)$$

where the Fourier coefficients X_k are given by the analysis equation

$$X_k = \frac{1}{T_0} \int_{T_0} x(t)\, e^{-j2\pi k F_0 t}\, dt \qquad < (2)$$

with the fundamental frequency F_0 and the period T_0 related by F_0 (Hz) = $1/T_0$ (sec).

To obtain finite-sum approximations for the above two equations, consider the analog periodic signal *x(t)* shown in Figure and its sampled version $x_s(nT)$. Using $x_s(nT)$, we can approximate the integral for X_k by the sum

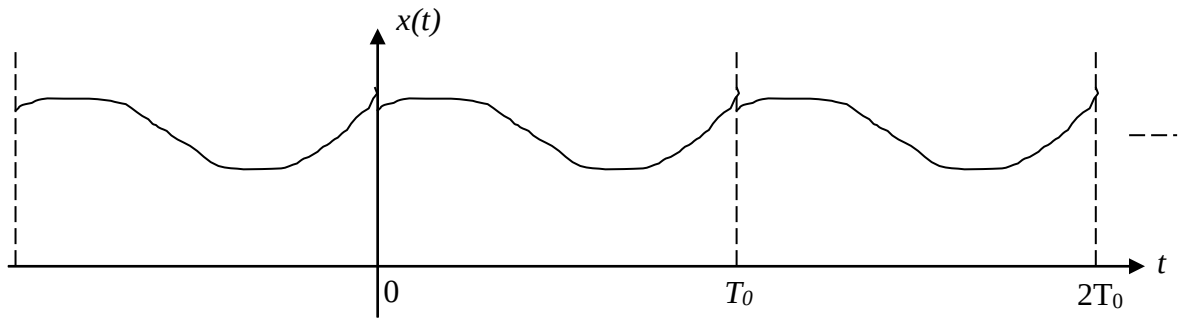
$$X_k = \frac{1}{T_0} \sum_{n=0}^{N-1} x_s(nT)\, e^{-j2\pi k F_0 nT}\, T, \qquad k = 0, 1, \ldots, N-1$$

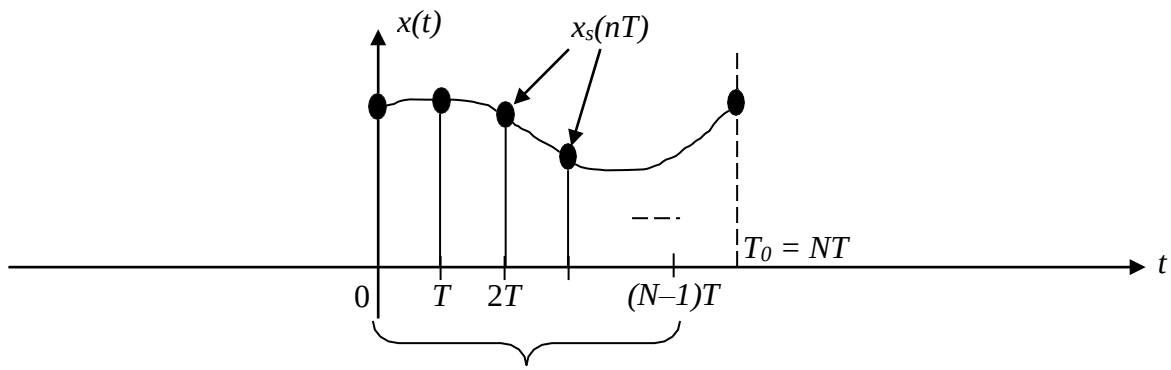
$$= \frac{1}{N} \sum_{n=0}^{N-1} x(n)\, e^{-j2\pi k n/N}, \qquad k = 0, 1, \ldots, N-1$$

where we used the relation $F_0T = 1/N$, and approximated dt (or Δt) by T, and have used the shorthand notation $x(n) = x_s(nT)$. (This procedure is similar to that used in a typical introduction to integral calculus).

A finite series approximation for $x(t)$ is obtained by truncating the series for $x(t)$ in equation (1) to N terms and substituting $t = nT$ and $F_0 = 1/TN$. This will necessarily give the discrete sequence $x(n)$ instead of the continuous function $x(t)$:

$$x(n) \text{ or } x_n = \sum_{k=0}^{N-1} X_k \, e^{j2\pi kn/N}, \quad n = 0, 1, \ldots, N-1$$





The above two equations define the discrete Fourier transform (DFT) pair. *A slight adjustment of the (1/N) factor is needed so as to conform to* ***standard usage***. The adjustment consists of moving the (1/N) factor from one equation to the other. Then the direct DFT of the time series $x_0, x_1, \ldots, x_{N-1}$ is defined as

$$X_k = \sum_{n=0}^{N-1} x_n e^{-j2\pi kn/N}, \; k = 0, 1, \ldots, N-1 \quad (3)$$

And the inverse DFT is defined as

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{j2\pi kn/N}, \; n = 0, 1, \ldots, N-1 \quad (4)$$

It can be shown that substituting equation (3) into equation (4) produces an identity, so that the two equations are indeed mutually inverse operations and therefore constitute a valid transform pair.

(End of Omit)

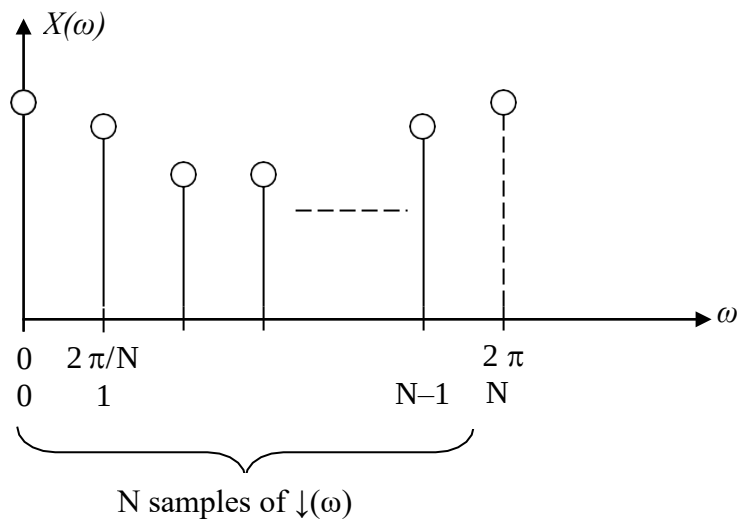
The discrete Fourier transform as a discretized (sampled) version of the DTFT A finite-duration sequence $x(n)$ of length N (the length N may have been achieved by zero-padding a sequence of shorter length) has a Fourier transform denoted $X(\omega)$ or $X(e^{j\omega})$,

$$X(\omega) = \sum_{n=0}^{N-1} x(n) e^{-j\omega n}, \qquad 0 \le \omega < 2\pi$$

$X(\omega)$ is a continuous function of ω and has a period of 2π. If we take N samples of $X(\omega)$ at equally spaced frequencies $\{\omega_k = 2\pi k/N,\ k = 0, 1, ..., N–1\}$, along the interval $[0, 2\pi)$, the resulting samples are (Figure)

$$X(k) \equiv X\left(\frac{2\pi k}{N}\right) = \sum_{n=0}^{N-1} x(n)\, e^{-j2\pi kn/N},$$

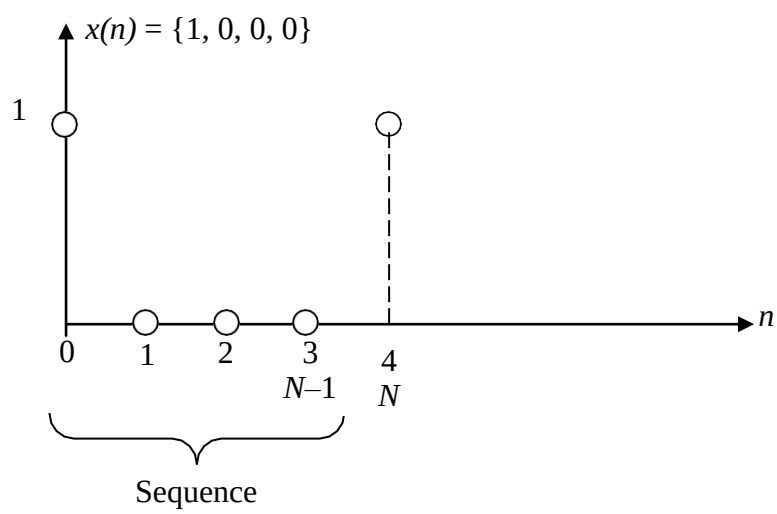
Since these frequency samples are obtained by evaluating the Fourier transform $X(\omega)$ at N equally spaced discrete frequencies, the above relation is called the **discrete Fourier transform (DFT)** of $x(n)$. In other words, $X(k)$ are discrete samples of the continuous $X(\omega)$.



The corresponding **inverse discrete Fourier transform (IDFT)** is given by

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)\, e^{j2\pi kn/N},\ n = 0, 1, \ldots, N–1$$

Example 2.4.1 Find the DFT of the **unit sample** $x(n) = \{1, 0, 0, 0\}$. (Aside. What is the DTFT of $x(n) = \{1, 0, 0, 0\}$?)



Solution The number of samples is $N = 4$. The DFT is given by

$$X(k) = \sum_{n=0}^{N-1} x(n)\, e^{-jk2\pi n/N}, \qquad 0 \le k \le N–1$$

$$= \sum_{n=0}^{4-1} x(n)\, e^{-jk2\pi n/4}, \qquad 0 \le k \le 3$$

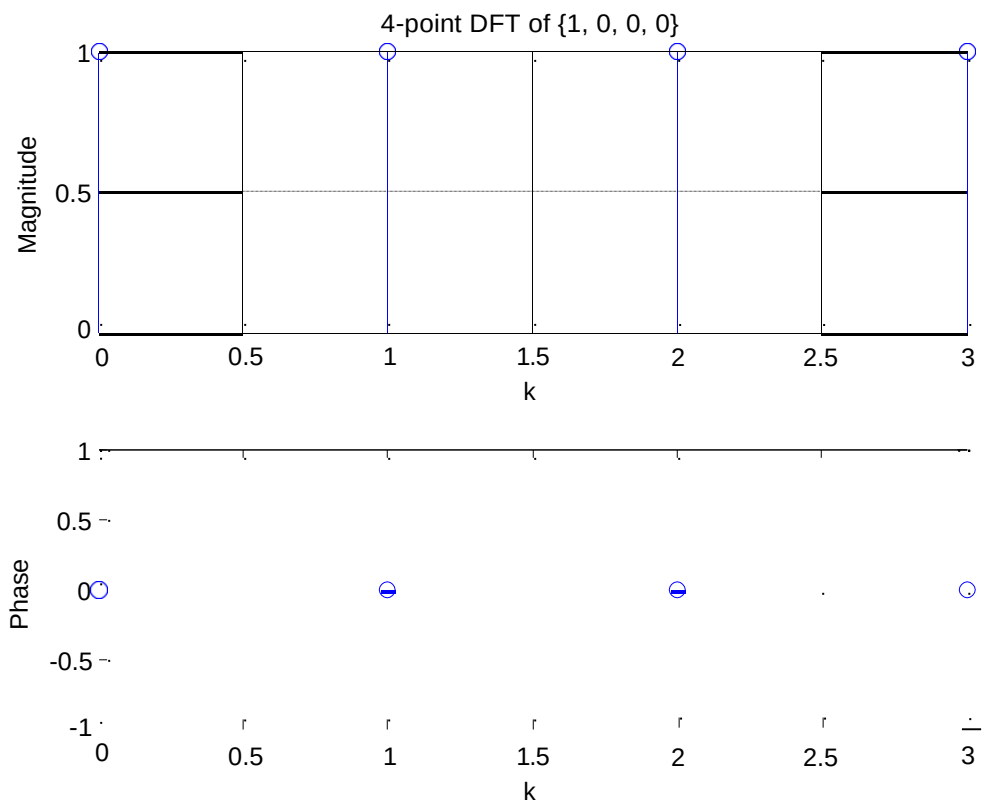
$$= \sum_{n=0}^{3} x(n)\, e^{-jk2\pi n/4}, \qquad k = 0, 1, 2, 3$$

$$\sum_{n=0}^{3} x(n)\, e^{-jk2\pi n/4},\ k = 0, 1, 2, 3$$

$k = 0$	$X(0) = \sum_{n=0}^{3} x(n)\, e^{-j0.2\pi n/4} = \sum_{n=0}^{3} x(n).1 = \sum_{n=0}^{3} x(n)$ $= x(0) + x(1) + x(2) + x(3) = 1 + 0 + 0 + 0 = 1$
$k = 1$	$X(1) = \sum_{n=0}^{3} x(n)\, e^{-j1.2\pi n/4} = \sum_{n=0}^{3} x(n)\, e^{-j\pi n/2} = x(0)\, e^{-j0} = 1 . 1 = 1$
$k = 2$	$X(2) = \sum_{n=0}^{3} x(n)\, e^{-j2.2\pi n/4} = \sum_{n=0}^{3} x(n)\, e^{-j\pi n} = x(0)\, e^{-j0} = 1 . 1 = 1$
$k = 3$	$X(3) = \sum_{n=0}^{3} x(n)\, e^{-j3.2\pi n/4} = \sum_{n=0}^{3} x(n)\, e^{-j3\pi n/2} = x(0)\, e^{-j0} = 1 . 1 = 1$

The DFT is $X(k) = \{1, 1, 1, 1\}$ and contains all (four) frequency components. In this example $X(k)$ is real-valued.

In MATLAB use *fft(x)* for the DFT. The magnitude and phase plots of $X(k)$ and the program segment follow.




```
x= [1, 0, 0, 0]; X= fft(x);
Mag = abs(X); Phase = angle(X);
```

```
k=0:3;
subplot(2, 1, 1), stem(k, Mag, 'bo');    %Two rows, one column, #1
xlabel ('k'), ylabel('Magnitude');
title ('4-point DFT of \{1, 0, 0, 0\}')
grid;
subplot(2, 1, 2), stem(k, Phase, 'bo'); %Two rows, one column, #2
xlabel ('k'), ylabel('Phase');
```

Matrix formulation To facilitate computation the DFT equations may be arranged as a matrix-vector multiplication. We define the twiddle factor $W_N = e^{-j2\pi/N}$, which for $N = 4$ becomes $W_4 = e^{-j2\pi/4}$. The equations are rewritten using the twiddle factor

$$X(k) = \sum_{n=0}^{3} x(n)\, e^{-jk2\pi n/4} = \sum_{n=0}^{3} x(n) W_4^{kn}, \qquad k = 0, 1, 2, 3$$

There are a total of 4 values of *X(.)* ranging from *X(0)* to *X(3)*:

$$\begin{aligned}
X(0) &= x(0)W_4^0 + x(1)W_4^0 + x(2)W_4^0 + x(3)W_4^0 \\
X(1) &= x(0)W_4^0 + x(1)W_4^1 + x(2)W_4^2 + x(3)W_4^3 \\
X(2) &= x(0)W_4^0 + x(1)W_4^2 + x(2)W_4^4 + x(3)W_4^6 \\
X(3) &= x(0)W_4^0 + x(1)W_4^3 + x(2)W_4^6 + x(3)W_4^9
\end{aligned}$$

These equations can be put in matrix from:

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W_4^1 & W_4^2 & W_4^3 \\ 1 & W_4^2 & W_4^4 & W_4^6 \\ 1 & W_4^3 & W_4^6 & W_4^9 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix}$$

This last form is perhaps the most convenient to perform the actual computations by plugging in the twiddle factors W_4^m and the signal values *x(.)*. Note that

$$W_N^{k+(N/2)} = -W_N^k \qquad \text{and} \qquad W_N^{mN+k} = W_N^k$$

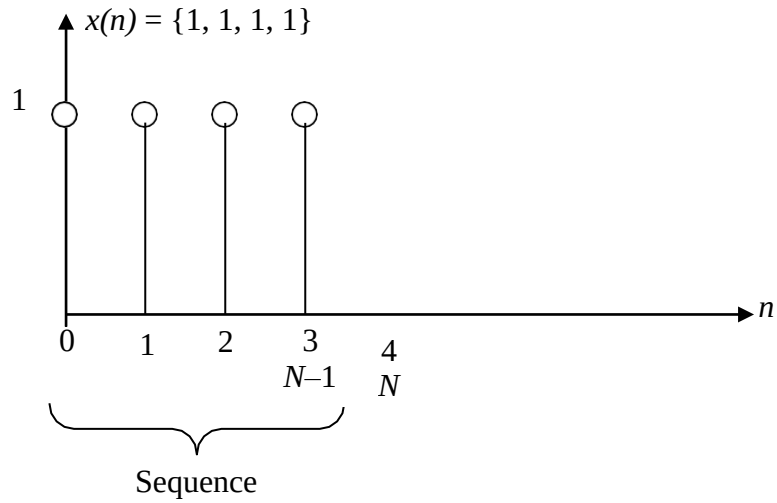
and

$$W_4^1 = e^{-j2\pi/4} = -j,\ W_4^2 = (-j)^2 = -1,\ W_4^3 = (-j)^3 = j, \text{ etc.}$$

The above matrix form then can be written,

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Example 2.4.2 Find the DFT of the **"dc" sequence** *x(n)* = {1, 1, 1, 1}. (Aside. What is the DTFT of *x(n)* = {1, 1, 1, 1}? Give 4 samples of *X(ω)* at intervals of $2\pi/4$ starting at $\omega = 0$.) (Compare Proakis, 3rd Ed., Ex. 5.1.2)



Solution The number of samples is $N = 4$. The DFT is given by

$$X(k) = \sum_{n=0}^{N-1} x(n)\, e^{-jk2\pi n/N}, \qquad 0 \le k \le N-1$$

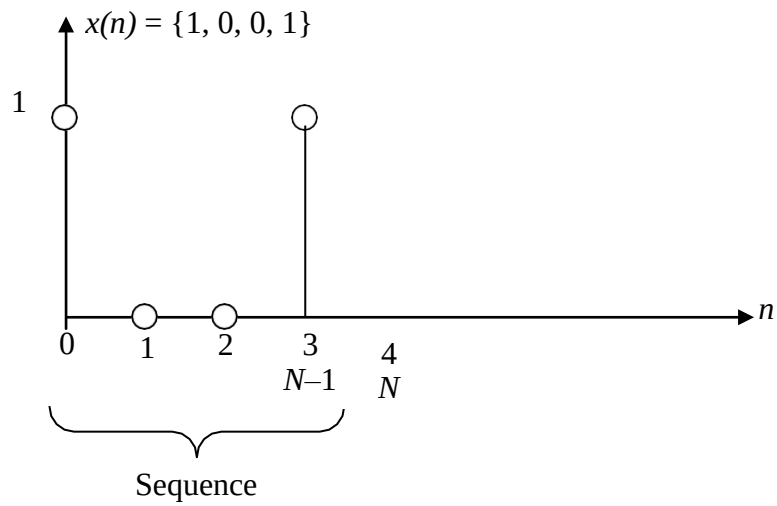
$$= \sum_{n=0}^{3} x(n)\, e^{-jk2\pi n/4}, \qquad k = 0, 1, 2, 3$$

$$X(k) == \sum_{n=0}^{3} x(n)\, e^{-jk2\pi n/4},\ k = 0, 1, 2, 3$$

$k = 0$	$X(0) = \sum_{n=0}^{3} x(n)\, e^{-j0.2\pi n/4} = \sum_{n=0}^{3} x(n)\,.\,1 = \sum_{n=0}^{3} x(n)$ $= x(0) + x(1) + x(2) + x(3) = 1 + 1 + 1 + 1 = 4$
$k = 1$	$X(1) = \sum_{n=0}^{3} x(n) e^{-j1.2\pi n/4} = \sum_{n=0}^{3} x(n) e^{-j\pi n/2} = \sum_{n=0}^{3} 1.(e^{-j\pi/2})^n = \sum_{n=0}^{3} (-j)^n$ $= 1 - j + (-j)^2 + (-j)^3 = 1 - j + j^2 - j^3 = 1 - j - 1 + j = 0$
$k = 2$	$X(2) = \sum_{n=0}^{3} x(n) e^{-j2.2\pi n/4} = \sum_{n=0}^{3} 1.(e^{-j\pi})^n = 1 - 1 + 1 - 1 = 0$
$k = 3$	$X(3) = \sum_{n=0}^{3} x(n) e^{-j3.2\pi n/4} = \sum_{n=0}^{3} 1.e^{-j3\pi n/2} = 1 + j - 1 - j = 0$

The DFT is $X(k) = \{4, 0, 0, 0\}$ and contains only the "dc" component and no other. Here again $X(k)$ is real-valued.

Example 2.4.3 Find the DFT of the sequence $x(n) = \{1, 0, 0, 1\}$



Solution The number of samples is $N = 4$. The DFT is given by

$$X(k) = \sum_{n=0}^{N-1} x(n)\, e^{-jk2\pi n/N}, \qquad 0 \le k \le N-1$$

$$= \sum_{n=0}^{3} x(n)\, e^{-jk2\pi n/4}, \qquad k = 0, 1, 2, 3$$

$$X(k) = \sum_{n=0}^{3} x(n)\, e^{-jk2\pi n/4},\ k = 0, 1, 2, 3$$

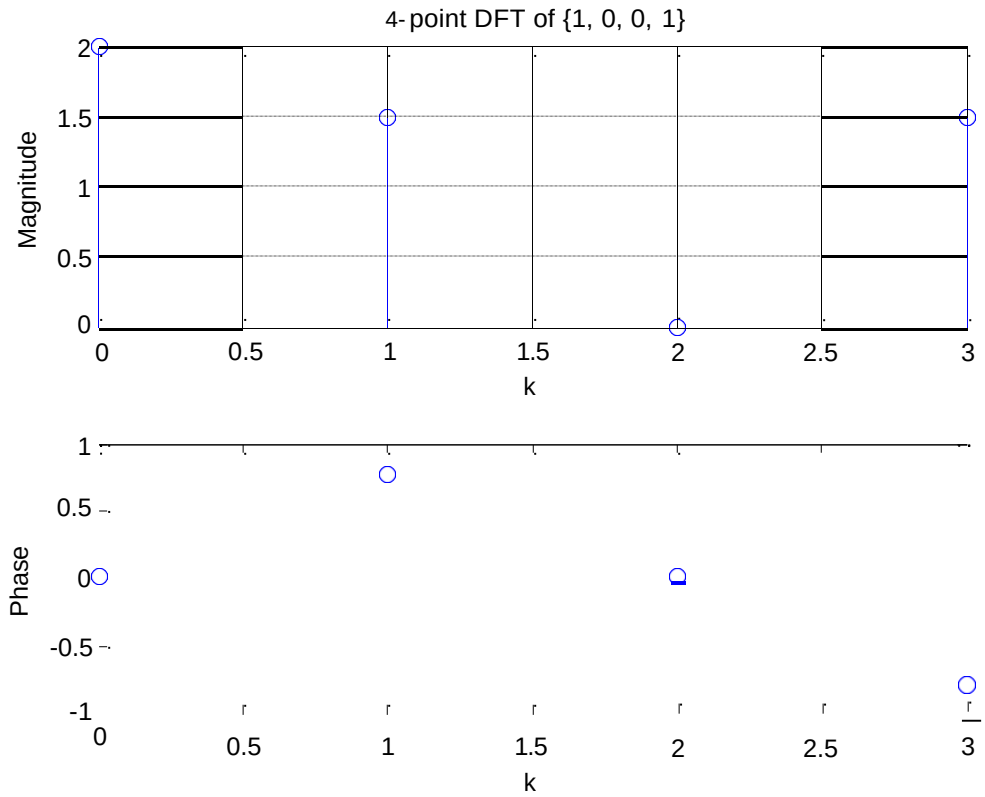
$k = 0$	$X(0) = \sum_{n=0}^{3} x(n)\, e^{-j0.2\pi n/4} = \sum_{n=0}^{3} x(n).1 = \sum_{n=0}^{3} x(n)$ $= x(0) + x(1) + x(2) + x(3) = 1 + 0 + 0 + 1 = 2$
$k = 1$	$X(1) = \sum_{n=0}^{3} x(n)\, e^{-j1.2\pi n/4} = \sum_{n=0}^{3} x(n)\, e^{-j\pi n/2} = \sum_{n=0}^{3} x(n)(e^{-j\pi/2})^n = \sum_{n=0}^{3} x(n)(-j)^n$ $= 1 . 1 + 0 + 0 + 1 . (-j)^3 = 1 + j = \sqrt{2}\, e^{j\pi/4} = \sqrt{2}\angle\pi/4$
$k = 2$	$X(2) = \sum_{n=0}^{3} x(n)\, e^{-j2.2\pi n/4} = \sum_{n=0}^{3} x(n)\, e^{-j\pi n} = x(0)\, e^{-j0} + x(3)\, e^{-j\pi 3}$ $= 1 . 1 + 1 . (-1) = 0$
$k = 3$	$X(3) = \sum_{n=0}^{3} x(n)\, e^{-j3.2\pi n/4} = \sum_{n=0}^{3} x(n)\, e^{-j3\pi n/2} = x(0)\, e^{-j0} + x(3)\, e^{-j3\pi 3/2}$ $= 1 . 1 + 1 . (-j) = 1 - j = \sqrt{2}\, e^{-j\pi/4} = \sqrt{2}\angle -\pi/4$

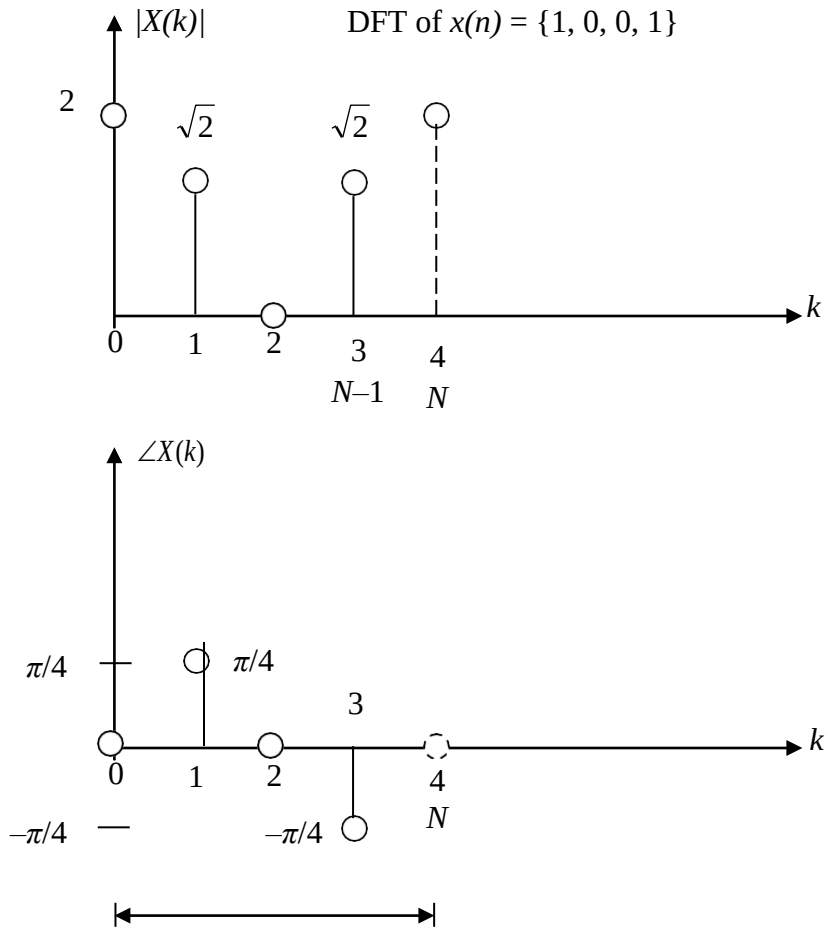
The DFT is $X(k) = \{2, \sqrt{2}e^{j\pi/4}, 0, \sqrt{2}e^{-j\pi/4}\}$. In general $X(k)$ is complex-valued and has a magnitude and a phase. See figure.

In MATLAB use *fft(x)* for the DFT and *ifft(X)* for the IDFT. The magnitude and phase plots and the program segment follow.

```
x = [1, 0, 0, 1]; X = fft(x); Mag = abs(X); Phase = angle(X);
k = 0:3;
subplot(2, 1, 1), stem(k, Mag, 'bo');    %Two rows, one column, #1
```

```
xlabel ('k'), ylabel('Magnitude');
title ('4-point DFT of \{1, 0, 0, 1\}')
grid;
subplot(2, 1, 2), stem(k, Phase, 'bo'); %Two rows, one column, #2
xlabel ('k'), ylabel('Phase');
```



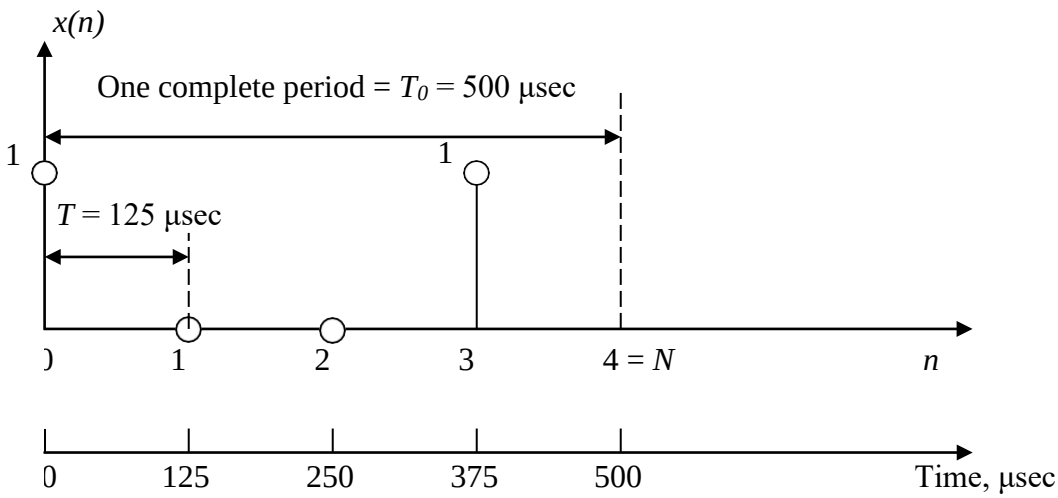


One period. Corresponds to:

1. $\omega = 0$ to 2π radians, or
2. $\Omega = 0$ to Ω_s rad/sec, or
3. $F = 0$ to F_s Hz, or
4. A sequence length of N (= 4)

The frequency components $\{X(k), k = 0, 1, 2, 3\}$ are "harmonics". The spacing between successive components, denoted by F_0, is the **resolution of the DFT** and is given by $F_0 = F_s/N$: it is the sampling interval in the *frequency domain*. It is determined by the sampling frequency F_s or sampling interval T in the *time domain* ($F_s = 1/T$) and the number of samples, N. Note that N is the number of time domain samples as well as the number of frequency domain samples.

Record length, sampling time and frequency resolution of the DFT We shall use Example 3 above to illustrate. Suppose that the sampling frequency is 8000 Hz, then the sampling time is T = (1/8000) sec = 125 μsec. The time domain samples $x(n)$ are spaced 125 μsec apart. In the frequency domain the DFT values, $X(k)$, are spaced (8000/4) = 2000 Hz apart. The DFT spectrum is re-plotted below with these parameters.

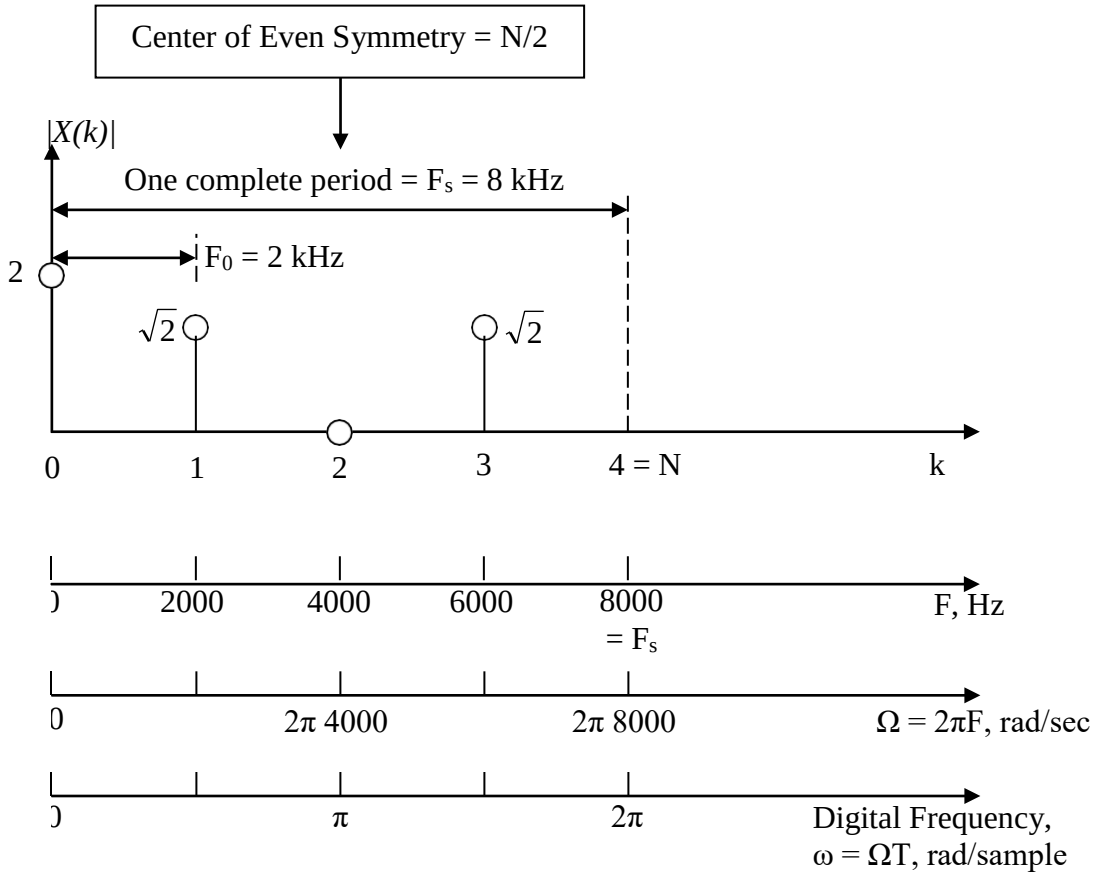


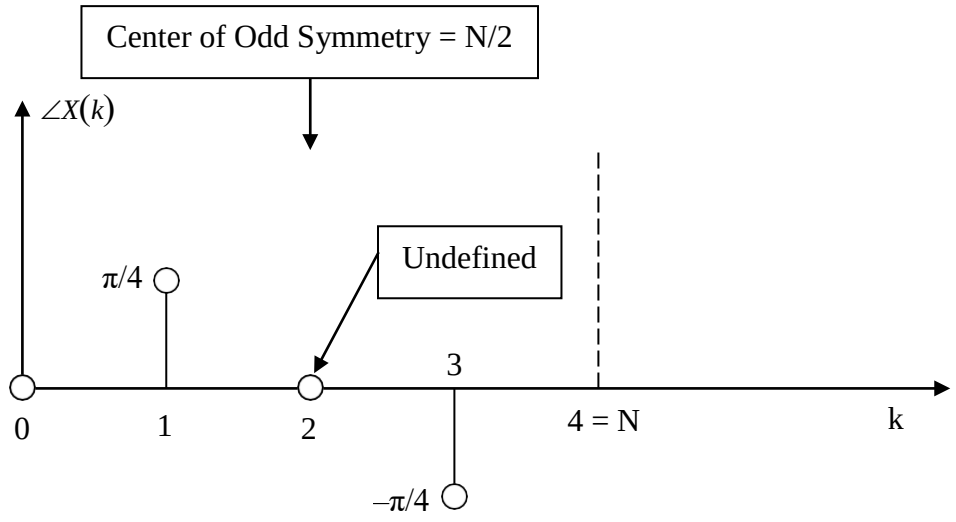
The terms introduced in this example and their interrelationships are summarized below:

Record length (one period) = T_0 seconds = N samples
Sampling interval = T seconds = $1/F_s$
Sampling interval = T sec = (T_0/N) seconds

Sampling frequency (one period) = F_s Hz
Frequency resolution of the DFT = F_0 Hz = $1/T_0$
Frequency resolution of the DFT = F_0 Hz = (F_s/N) Hz

The situation in the frequency domain is shown below.





Example 2.4.4 Find the inverse discrete Fourier transform of $X(k)$ = {3, (2+j), 1, (2–j)}.
Solution The number of samples is N = 4. The IDFT is given by the synthesis equation

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)\, e^{j2\pi k n/N}, \qquad n = 0, 1, \ldots, N-1$$

$$= \frac{1}{4}\sum_{k=0}^{3} X(k)\, e^{j2\pi k n/4},$$

$$= \frac{1}{4}\sum_{k=0}^{3} X(k)\, e^{jkn\pi/2}, \qquad 0 \le n \le 3$$

The calculations for {$x(n)$, n = 0 to 3} are shown in table below.

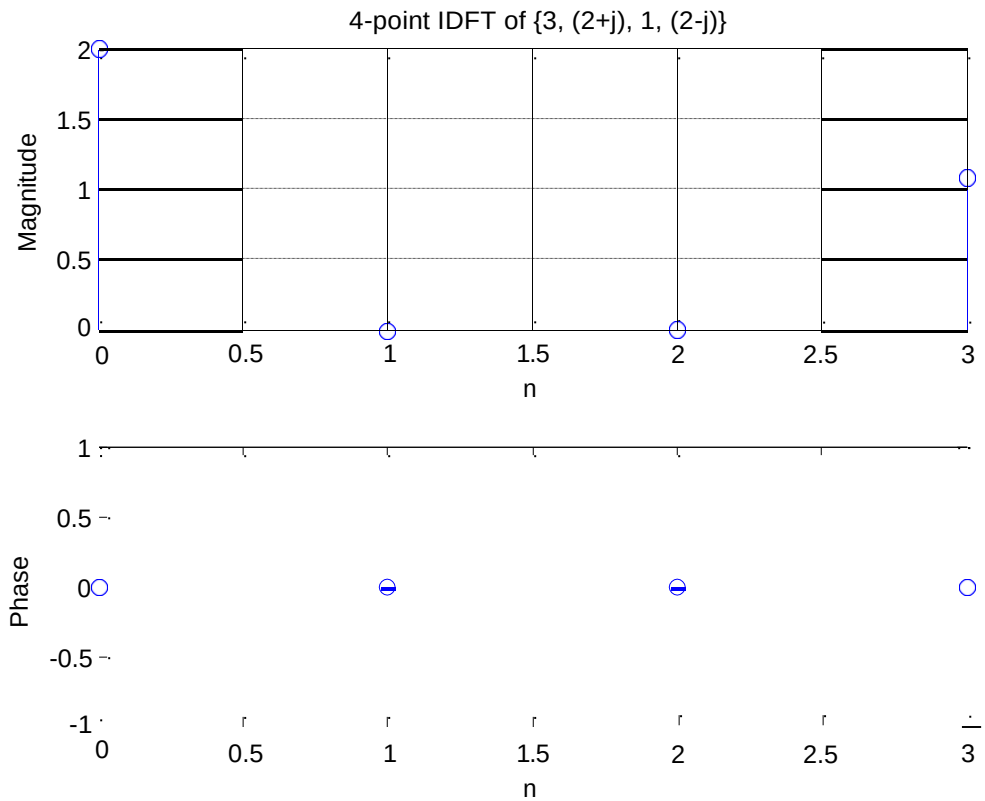
$$x(n) = \frac{1}{4}\sum_{k=0}^{3} X(k)\, e^{jkn\pi/2}$$

$n = 0$	$x(0) = \frac{1}{4}\sum_{k=0}^{3} X(k)\, e^{jk0\pi/2} = \frac{1}{4}\sum_{k=0}^{3} X(k)$ = (1/4) {X(0) + X(1) + X(2) + X(3)} = (1/4) {3 + 2+j + 1 + 2–j} = 2
$n = 1$	$x(1) = (1/4)\sum_{k=0}^{3} X(k)\, e^{jk1\pi/2} = (1/4)\sum_{k=0}^{3} X(k)\left(e^{j\pi/2}\right)^k = (1/4)\sum_{k=0}^{3} X(k)\,(j)^k$ = (1/4) {X(0) $(j)^0$ + X(1) $(j)^1$ + X(2) $(j)^2$ + X(3) $(j)^3$} = (1/4) {3 . 1 + (2+j) . j + 1 . (–1) + (2–j) . (–j)} = 0
$n = 2$	$x(2) = (1/4)\sum_{k=0}^{3} X(k)\, e^{jk2\pi/2} = (1/4)\sum_{k=0}^{3} X(k)\, e^{jk\pi} = (1/4)\sum_{k=0}^{3} X(k)\,(-1)^k$ = (1/4) {X(0) (1) + X(1) (–1) + X(2) (1) + X(3) (–1)} = (1/4) {X(0) – X(1) + X(2) – X(3)} = (1/4) {3 – (2+j) + 1 . (–1) – (2–j)} = 0
$n = 3$	$x(3) = (1/4)\sum_{k=0}^{3} X(k)\, e^{jk3\pi/2} = (1/4)\sum_{k=0}^{3} X(k)\left(e^{j3\pi/2}\right)^k = (1/4)\sum_{k=0}^{3} X(k)\,(-j)^k$ = (1/4) {3 . 1 + (2+j). (–j) + 1 . (–1) + (2–j). j} = (1/4) {3 –j2 + 1 –1 + j2 + 1} = 1

Thus $x(n)$ = {2, 0, 0, 1}.

In MATLAB use *ifft(X)* for the IDFT. The magnitude and phase plots of $x(n)$ and the program segment follow.

```
X = [3, (2+j), 1, (2-j)]; x = ifft(X); Mag = abs(x); Phase = angle(x);
n = 0:3;
subplot(2, 1, 1), stem(n, Mag, 'bo');    %Two rows, one column, #1
xlabel ('n'), ylabel('Magnitude');
title ('4-point IDFT of \{3, (2+j), 1, (2-j)\}')
grid;
subplot(2, 1, 2), stem(n, Phase, 'bo'); %Two rows, one column, #2
xlabel ('n'), ylabel('Phase');
```



Matrix formulation Here again to facilitate computation the IDFT equations may be arranged as a matrix-vector multiplication.

$$x(n) = \frac{1}{4}\sum_{n=0}^{3} X(k)\, e^{jk2\pi n/4} = \frac{1}{4}\sum_{n=0}^{3} X(k)\left(W_4^*\right)^{kn}, \qquad n = 0, 1, 2, 3$$

There are a total of 4 values of *x(.)* ranging from *x(0)* to *x(3)*:

$$\begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \left(W_4^*\right)^1 & \left(W_4^*\right)^2 & \left(W_4^*\right)^3 \\ 1 & \left(W_4^*\right)^2 & \left(W_4^*\right)^4 & \left(W_4^*\right)^6 \\ 1 & \left(W_4^*\right)^3 & \left(W_4^*\right)^6 & \left(W_4^*\right)^9 \end{bmatrix} \begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix}$$

This last form is perhaps the most convenient to perform the actual computations by plugging in the twiddle factors $\left(W_4^*\right)^m$ and the values *X(.)*. The above matrix form then can be written,

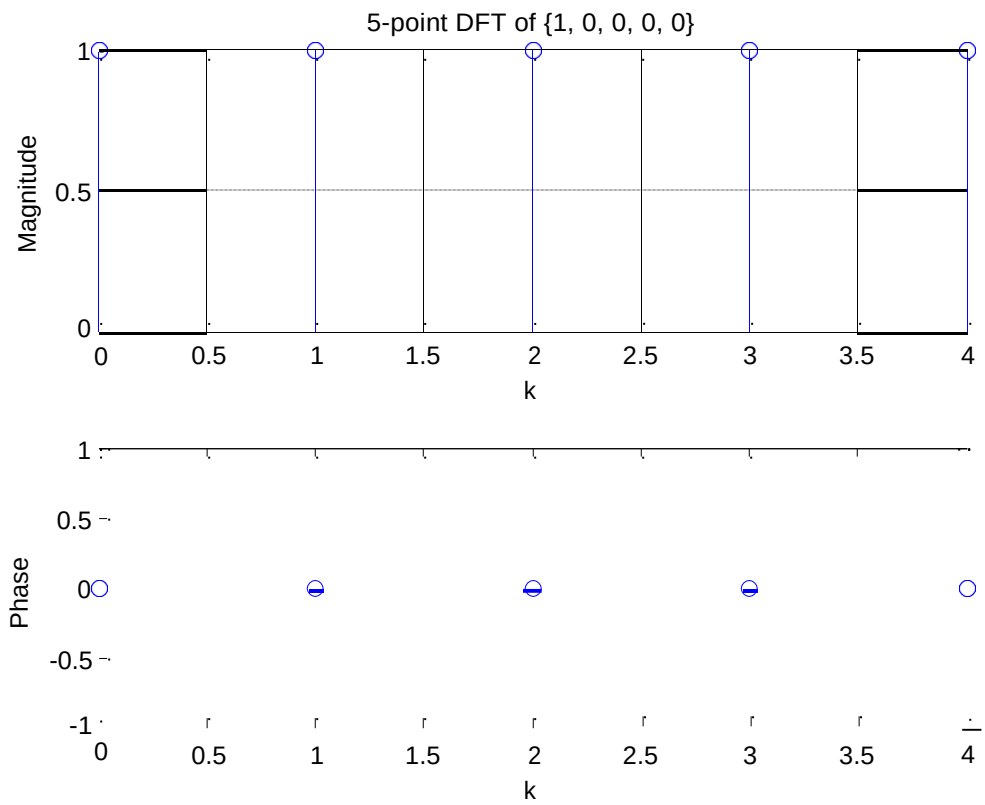
$$\begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \left(W_4^*\right)^1 & \left(W_4^*\right)^2 & \left(W_4^*\right)^3 \\ 1 & \left(W_4^*\right)^2 & \left(W_4^*\right)^4 & \left(W_4^*\right)^6 \\ 1 & \left(W_4^*\right)^3 & \left(W_4^*\right)^6 & \left(W_4^*\right)^9 \end{bmatrix} \begin{bmatrix} 3 \\ 2+j \\ 1 \\ 2-j \end{bmatrix}$$

Example 2.4.5 [*N* not an integral power of 2] Using MATLAB find the 5-point DFT of

$$x(n) = \{1, 0, 0, 0, 0\}$$

Solution

```
x = [1, 0, 0, 0, 0]; X = fft(x),
Mag = abs(X); Phase = angle(X);
k = 0:4;
subplot(2, 1, 1), stem(k, Mag, 'bo');    %Two rows, one column, #1
xlabel ('k'), ylabel('Magnitude');
title ('5-point DFT of \{1, 0, 0, 0, 0\}')
grid;
subplot(2, 1, 2), stem(k, Phase, 'bo'); %Two rows, one column, #2
xlabel ('k'), ylabel('Phase');
```

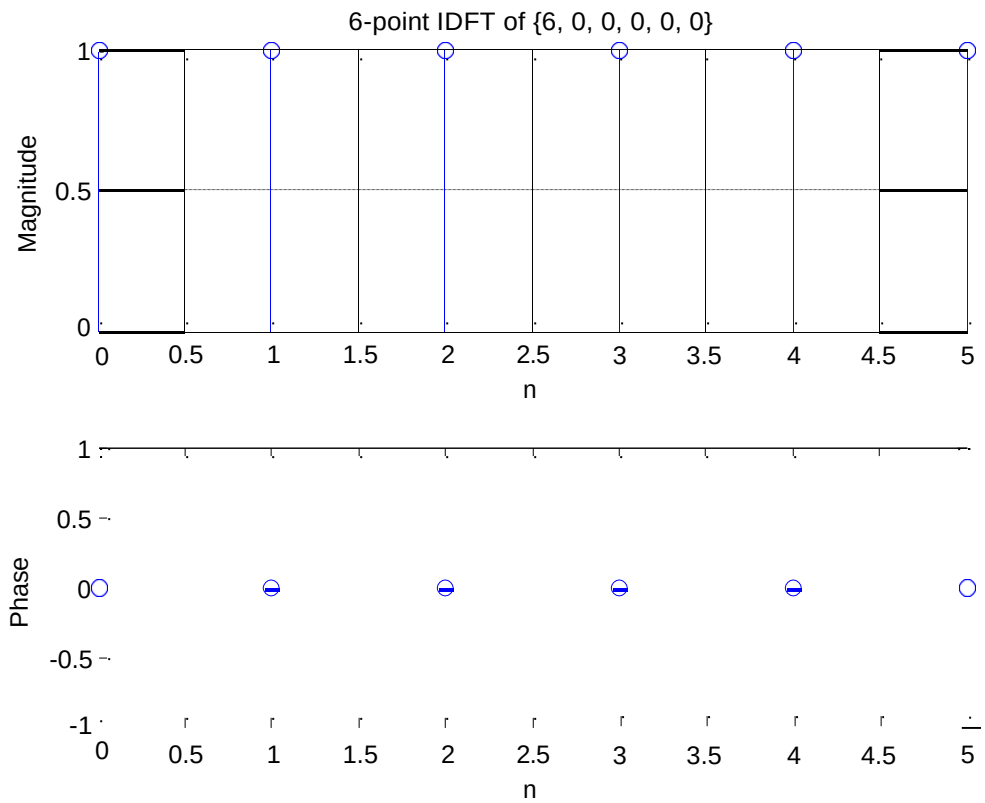


Example 2.4.6 [*N* not an integral power of 2] Using MATLAB find the 6-point IDFT of

$X(k) = \{6, 0, 0, 0, 0, 0\}$

Solution

```
X = [6, 0, 0, 0, 0, 0]; x = ifft(X); Mag = abs(x); Phase = angle(x);
n = 0:5;
subplot(2, 1, 1), stem(n, Mag, 'bo');    %Two rows, one column, #1
xlabel ('n'), ylabel('Magnitude');
title ('6-point IDFT of \{6, 0, 0, 0, 0, 0\}')
grid;
subplot(2, 1, 2), stem(n, Phase, 'bo'); %Two rows, one column, #2
xlabel ('n'), ylabel('Phase');
```



Example 2.4.7 Consider a sequence $x(n) = \{2, -1, 1, 1\}$ and the sampling time T = 0.5 sec. Compute its DFT and compare it with its DTFT.

Solution The record length of the sequence is $T_0 = 4T = 2$ sec.

The DFT is a sequence of 4 values given by

$$X(k) = \sum_{n=0}^{3} x(n)\, e^{-jk2\pi n/4}, \qquad k = 0, 1, 2, 3$$

The periodicity of $X(k)$ is 4. The frequency resolution of the DFT is $1/T_0 = 0.5$ Hz.

The DTFT, $X(\omega)$, is a continuous function of ω

$$X(\omega) = \sum_{n=-\infty}^{\infty} x(n)\, e^{-j\omega n} = \sum_{n=0}^{3} x(n)\, e^{-j\omega n} = 2\, e^{j0} - 1\, e^{-j\omega} + 1\, e^{-j2\omega} + 1\, e^{-j3\omega}$$

$$= 2 - e^{-j\omega} + e^{-j2\omega} + e^{-j3\omega}$$

The periodicity of $X(\omega)$, in terms of ω, is 2π. In terms of the Hertz frequency the periodicity is the sampling frequency = $F_s = 1/T = 2$ Hz.

The DFT is a sampled version of the DTFT, sampled at 4 points along the frequency axis spaced 0.5 Hz apart.

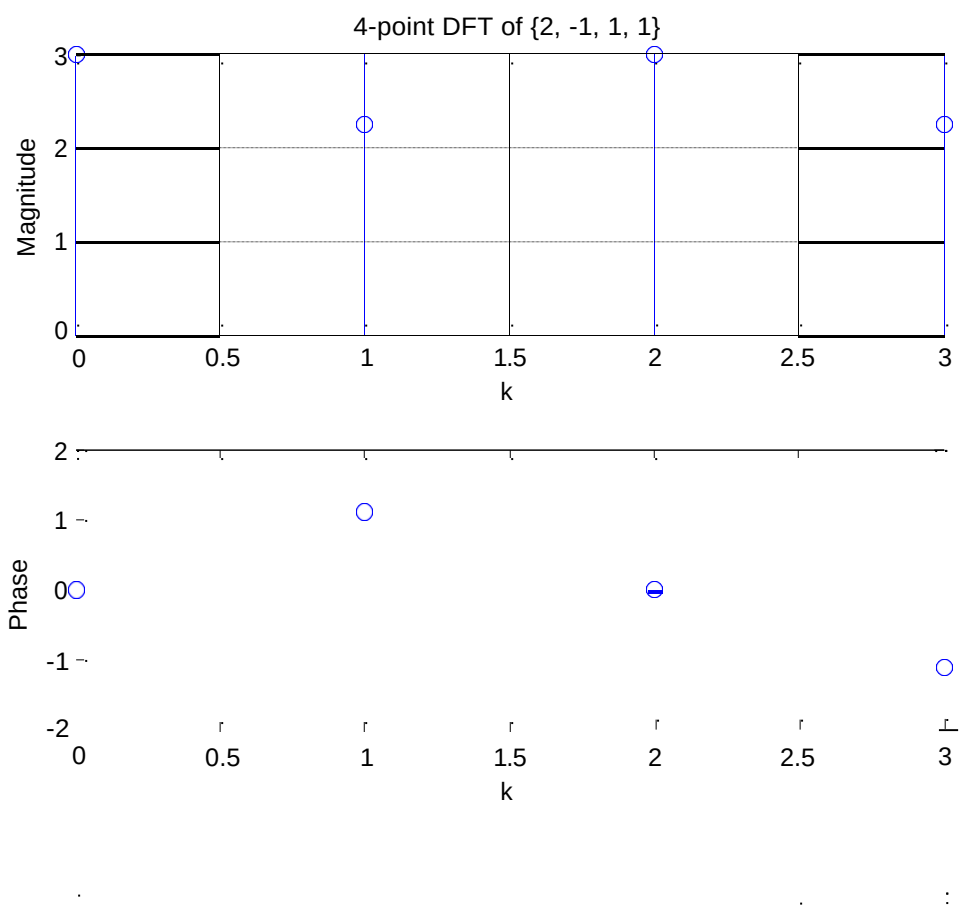
You should evaluate completely both $X(k)$ (a set of 4 numbers) and $X(\omega)$ (magnitude and phase). Note that $X(\omega)$ may be evaluated directly at $\omega = 0$, $\pi/2$, π, and $3\pi/2$ by plugging in the ω values into the expression given above; these are then the DFT numbers as well. The MATLAB solutions are given below.

In MATLAB: The magnitude and phase plots of the DFT can be generated by the following segment:

```
x = [2, -1, 1, 1]; X = fft(x); Mag = abs(X); Phase = angle(X);
k = 0:3;
subplot(2, 1, 1), stem(k, Mag, 'bo');    %Two rows, one column, #1
xlabel ('k'), ylabel('Magnitude');
title ('4-point DFT of \{2, -1, 1, 1\}')
grid;
subplot(2, 1, 2), stem(k, Phase, 'bo'); %Two rows, one column, #2
xlabel ('k'), ylabel('Phase');
```

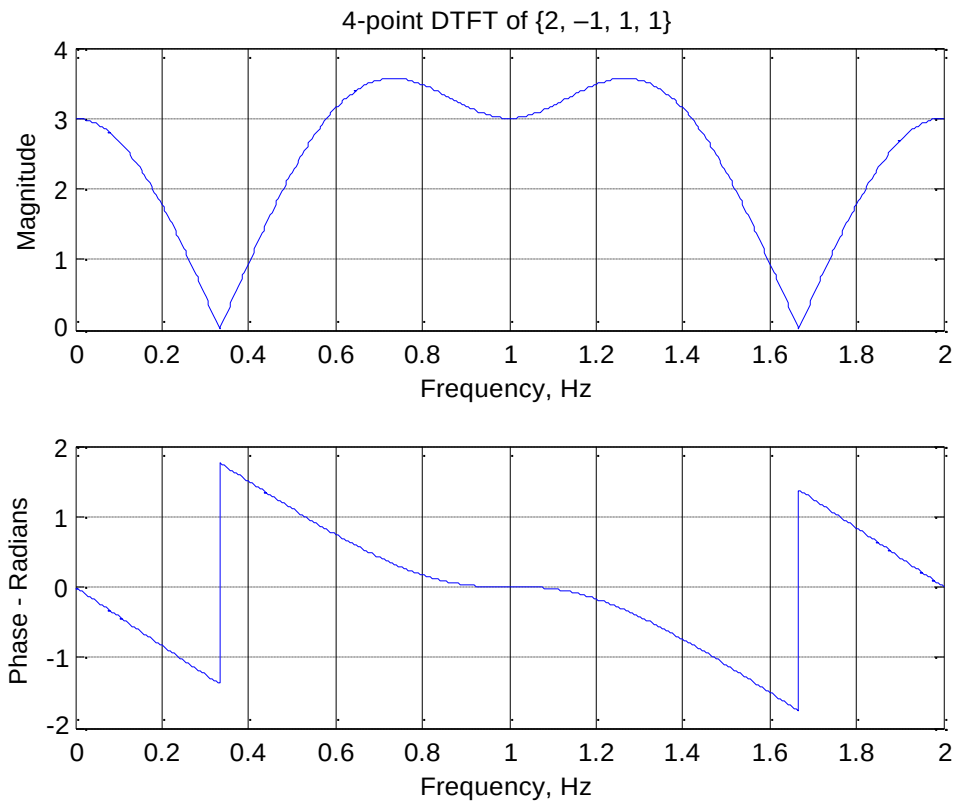
The MATLAB solution:

X = 3, (1 + j2), 3, (1 - j2)
Mag = 3, 2.2361, 3, 2.2361
Phase = 0, 1.1071, 0, -1.1071



The magnitude and phase plots of the DTFT can be generated by the following segment:

```
b = [2, -1, 1, 1]; %Numerator coefficients
a = [1]; %Denominator coefficient
w = 0: pi/256: 2*pi; %A total of 512 points
[h] = freqz(b, a, w);
subplot(2, 1, 1), plot(w/pi, abs(h));
xlabel('Frequency, Hz'), ylabel('Magnitude'); grid
title ('4-point DTFT of \{2, –1, 1, 1\}')
subplot(2, 1, 2), plot(w/pi, angle(h));
xlabel('Frequency, Hz'), ylabel('Phase - Radians'); grid
```



Example 2.4.8 Compute the discrete Fourier transform of the following finite length sequences considered to be of length N.

1) $x(n) = \delta(n+n_0)$, $\quad 0 < n_0 < N$

2) $x(n) = a^n$, $\quad 0 < a < 1$

Solution See Ramesh Babu 3.16.

Note that $x(n) = \delta(n+n_0)$ would be zero everywhere except at $n = -n_0$ which is not in the range $[0, N)$. So make it $\delta(n-n_0)$.

Example 2.4.9 [2008] Compute the N-point DFT $X(k)$ of the sequence

$$x(n) = \cos(2\pi n/N), \qquad 0 \le n \le N-1$$

for $0 \le k \le N-1$.

Solution Express $\cos(2\pi n/N)$ as $(e^{j2\pi n/N} + e^{-j2\pi n/N}) / 2$.

Example 2.4.10 Obtain the 7-point DFT of the sequence $x(n) = \{1, 2, 3, 4, 3, 2, 1\}$ by taking 7 samples of its DTFT uniformly spaced over the interval $0 \le \omega \le 2\pi$.

Solution The sampling interval in the frequency domain is $2\pi/7$. From Example 4 we have

$$X(e^{j\omega}) \text{ or } X(\omega) = 1 + 2e^{-j\omega 1} + 3e^{-j\omega 2} + 4e^{-j\omega 3} + 3e^{-j\omega 4} + 2e^{-j\omega 5} + 1e^{-j\omega 6}$$

$$= (2\cos 3\omega + 4\cos 2\omega + 6\cos\omega + 4)\, e^{-j3\omega}$$

The DFT, $X(k)$, is given by replacing ω with $k(2\pi/7)$ where k is an index ranging from 0 to 6:

$$\text{DFT} = X(\omega)\big|_{\omega = 2\pi k/7} = X(2\pi k/7), \qquad k = 0 \text{ to } 6$$

This is denoted XkfromDTFT in the MATLAB segment below.

MATLAB:

```
w = 0: 2*pi/7: 2*pi-0.001
XkfromDTFT = (4+6*cos(w)+4*cos(2*w)+2*cos(3*w)) .* exp(-j*3*w)
```

MATLAB solution:

```
XkfromDTFT = [16, (-4.5489 - 2.1906i), (0.1920 + 0.2408i), (-0.1431 - 0.6270i),
        (-0.1431 + 0.6270i), (0.1920 - 0.2408i), (-4.5489 + 2.1906i)]
```

This is the 7-point DFT obtained by sampling the DTFT at 7 points uniformly spaced in $(0, 2\pi)$. It should be the same as the DFT directly obtained, for instance, by using the *fft* function in MATLAB:

MATLAB:

```
xn = [1 2 3 4 3 2 1]
Xkusingfft = fft(xn)
```

MATLAB solution:

```
Xkusingfft = [16, (-4.5489 - 2.1906i), (0.1920 + 0.2408i), (-0.1431 - 0.6270i),
        (-0.1431 + 0.6270i), (0.1920 - 0.2408i), (-4.5489 + 2.1906i)]
```

It can be seen that "↓kfromDTFT" = "↓kusingfft".

Example 2.4.11 Obtain the 7-point inverse DTFT *x(n)* by finding the 7-point inverse DFT of *X(k)*:

$$X(2\pi k/7) = [16, (-4.5489 - 2.1906i), (0.1920 + 0.2408i), (-0.1431 - 0.6270i),$$
$$(-0.1431 + 0.6270i), (0.1920 - 0.2408i), (-4.5489 + 2.1906i)]$$

MATLAB:

```
Xk = [16, (-4.5489 - 2.1906i), (0.1920 + 0.2408i), (-0.1431 - 0.6270i),(-0.1431 +
        0.6270i), (0.1920 - 0.2408i), (-4.5489 + 2.1906i)]
xn = ifft(Xk)
```

MATLAB solution:

```
xn = [1.0000    2.0000    3.0000    4.0000    3.0000    2.0000    1.0000]
```

This is the original sequence we started with in Example 4.

Example 2.4.12 What will be the resulting time sequence if the DTFT of the *7-point* sequence is sampled at 6 (or fewer) uniformly spaced points in (0, 2π) and its inverse DFT is obtained?
Solution The sampling interval in the frequency domain now is $2\pi/6$. From Example 4 we have

$$X(e^{j\omega}) \text{ or } X(\omega) = 1 + 2e^{-j\omega 1} + 3e^{-j\omega 2} + 4e^{-j\omega 3} + 3e^{-j\omega 4} + 2e^{-j\omega 5} + 1e^{-j\omega 6}$$
$$= (2\cos 3\omega + 4\cos 2\omega + 6\cos\omega + 4)\, e^{-j3\omega}$$

The DFT then is given by

$$\text{DFT} = X(\omega)\big|_{\omega = 2\pi k/6} = X(2\pi k/6), \qquad k = 0 \text{ to } 5$$

This is denoted Xk6point in the MATLAB segment below.

MATLAB:

```
w = 0: 2*pi/6: 2*pi-0.001
Xk6point = (4+6*cos(w)+4*cos(2*w)+2*cos(3*w)) .* exp(-j*3*w)
```

MATLAB solution:

```
Xk6point = [16, (-3.0000 - 0.0000i), (1.0000 + 0.0000i), 0, (1.0000 + 0.0000i),
        (-3.0000 - 0.0000i)]
```

This is the 6-point DFT obtained by sampling the DTFT at 6 points uniformly spaced in (0, 2π).

Example 2.4.13 Obtain the 6-point inverse DTFT *x(n)* by finding the 6-point inverse DFT of Xk6point:

$$X(2\pi k/6) = [16, (-3.0000 - 0.0000i), (1.0000 + 0.0000i), 0, (1.0000 + 0.0000i), (-3.0000 - 0.0000i)]$$

MATLAB:

```
Xk6point = [16, (-3.0000 - 0.0000i), (1.0000 + 0.0000i), 0, (1.0000 + 0.0000i), (-
        3.0000 - 0.0000i)]
xn = ifft(Xk6point)
```

MATLAB solution:

```
xn = [2    2    3    4    3    2]
```

Comparing with the original 7-point sequence, xn = [1 2 3 4 3 2 1], we see the consequence of *under-sampling* the continuous-ω function $X(\omega)$: the corresponding time domain sequence *x(n)* is said to suffer *time-domain aliasing*. This is similar to the situation that occurs when a continuous-time function *x(t)* is under-sampled: the corresponding frequency domain function $X_s(\omega)$ contains *frequency-domainaliasing*.

Example 2.4.14 What will be the resulting time sequence if the DTFT of the *7-point* sequence is sampled at 8 (or more) uniformly spaced points in (0, 2π) and its inverse DFT is obtained?
Solution The sampling interval in the frequency domain now is $2\pi/8$. From Example 4 we have

$$X(e^{j\omega}) \text{ or } X(\omega) = 1 + 2e^{-j\omega 1} + 3e^{-j\omega 2} + 4e^{-j\omega 3} + 3e^{-j\omega 4} + 2e^{-j\omega 5} + 1e^{-j\omega 6}$$
$$= (2\cos 3\omega + 4\cos 2\omega + 6\cos\omega + 4)\, e^{-j3\omega}$$

The DFT then is given by

$$\text{DFT} = X(\omega)\big|_{\omega = 2\pi k/8} = X(2\pi k/8), \qquad k = 0 \text{ to } 7$$

This is denoted Xk8point in the MATLAB segment below.

MATLAB:

```
w = 0: 2*pi/8: 2*pi-0.001
Xk8point = (4+6*cos(w)+4*cos(2*w)+2*cos(3*w)) .* exp(-j*3*w)
```

MATLAB solution:

```
Xk8point = [16, (-4.8284 - 4.8284i), (0.0000 - 0.0000i), (0.8284 - 0.8284i), 0,
        (0.8284 + 0.8284i), (0.0000 - 0.0000i), (-4.8284 + 4.8284i)]
```

This is the 8-point DFT obtained by sampling the DTFT at 8 points uniformly spaced in $(0, 2\pi)$.

Example 2.4.15 Obtain the 8-point inverse DTFT *x(n)* by finding the 8-point inverse DFT of Xk8point:

$$X(2\pi k/8) = [16, (-4.8284 - 4.8284i), (0.0000 - 0.0000i), (0.8284 - 0.8284i), 0,$$
$$(0.8284 + 0.8284i), (0.0000 - 0.0000i), (-4.8284 + 4.8284i)]$$

MATLAB:

```
Xk8point = [16, (-4.8284 - 4.8284i), (0.0000 - 0.0000i), (0.8284 - 0.8284i), 0,
        (0.8284 + 0.8284i), (0.0000 - 0.0000i), (-4.8284 + 4.8284i)]
xn = ifft(Xk8point)
```

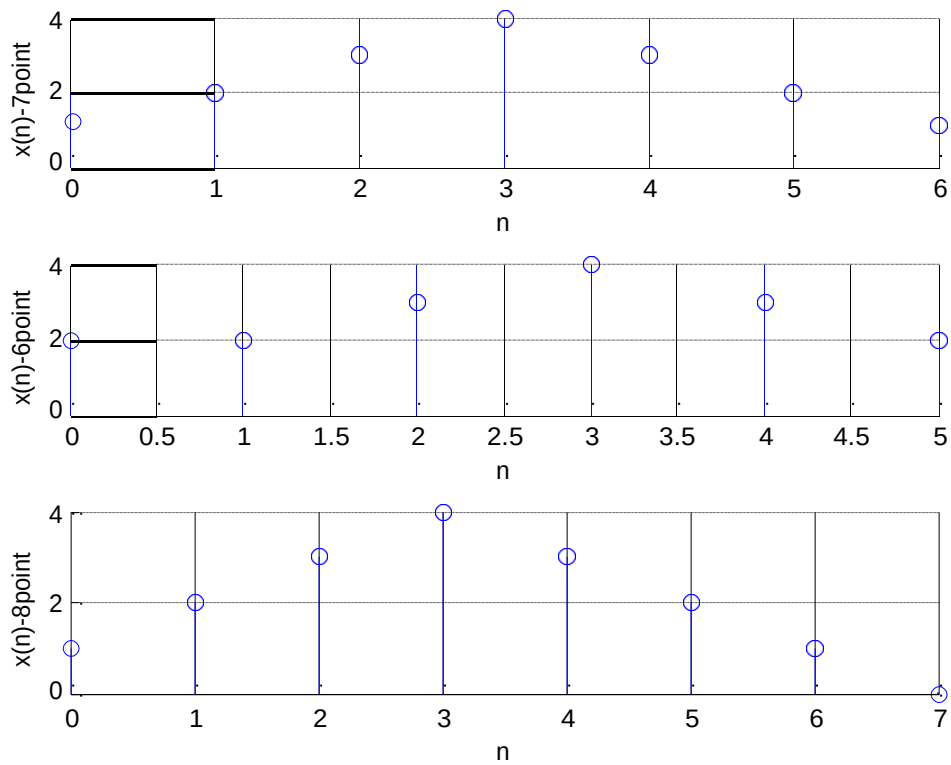
MATLAB solution:

```
xn = [1     2     3     4     3     2     1     0]
```

We see that the original 7-point sequence has been preserved with an *appended zero*. The original sequence and the *zero-padded sequence* (with any number of zeros) have the same DTFT. This is a case of *over-sampling* the continuous-ω function $X(\omega)$: there is no *time-domain aliasing*. This is similar to the situation that occurs when a continuous-time function *x(t)* is over-sampled: the corresponding frequency domain function $X_s(\omega)$ is free from *frequency-domain aliasing*.

```
%Sketch of sequences
n = 0:1:6; xn = [1, 2, 3, 4, 3, 2, 1];
subplot (3, 1, 1), stem(n, xn)
xlabel('n'), ylabel('x(n)-7point'); grid
%
n = 0:1:5; xn = [2     2     3     4     3     2];
```

```
subplot (3, 1, 2), stem(n, xn)
xlabel('n'), ylabel('x(n)-6point'); grid
%
n = 0:1:7; xn = [1, 2, 3, 4, 3, 2, 1, 0];
subplot (3, 1, 3), stem(n, xn)
xlabel('n'), ylabel('x(n)-8point'); grid
```



Properties of DFT

The properties of the DFT (for finite duration sequences) are essentially similar to those of the DFS for periodic sequences and result from the *implied* periodicity in the DFT representation.

(1) Periodicity If $x(n)$ and $X(k)$ are an N-point DFT pair, then $X(k)$ (and $x(n)$) is periodic with a periodicity of N. That is

$$X(k+N) = X(k) \text{ for all } k$$

This can be proved by replacing k by $k+N$ in the defining equation for $X(k)$.

(2) **Linearity** For two sequences $x_1(n)$ and $x_2(n)$ defined on $[0, N-1]$, if $x_3(n) = a\, x_1(n) + b\, x_2(n)$ then

$$\text{DFT}\,\{x_3(n)\} = \text{DFT}\,\{a\, x_1(n) + b\, x_2(n)\} = a\ \text{DFT}\,\{x_1(n)\} + b\ \text{DFT}\,\{x_2(n)\}$$

If one of the two sequences $x_1(n)$ and $x_2(n)$ is shorter than the other then the shorter one must be padded with zeros to make both sequences of the same length.

(3) Circular shift or circular translation of a sequence (The sequence wraps around). The circular shift, by an amount n_0 to the right, of the sequence $x(n)$ defined on $[0, N-1]$ is denoted by $x((n-n_0)_{mod\ N})$ or $x((n-n_0))_N$. For example, if $x(n)$ is

$$x(n) = \{x(0), x(1), x(2), \ldots, x(N-3), x(N-2), x(N-1)\}$$

then $x((n-2))_N$ is given by

$$x((n-2))_N = \{x(N-2), x(N-1), x(0), x(1), x(2), \ldots, x(N-3)\}$$

The operation can be thought of as wrapping the part that falls outside the region of interest around to the front of the sequence, or equivalently, just a straight (linear) translation of its *periodic extension*.

Example 2.5.1 Given $x(n) = \{1, 2, 2, 0\}$. Here $N = 4$. The circular shift of $x(n)$ by one unit to the right is $x((n-1))_4$, and is given by

$$x((n-1))_4 = \{0, 1, 2, 2\}$$

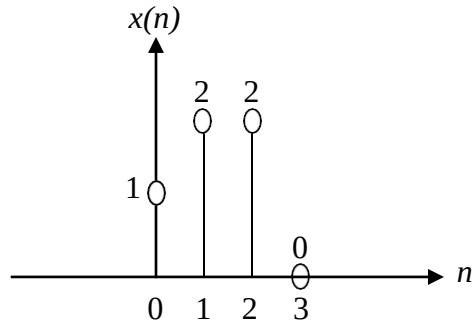
where the 0 has been wrapped around to the start and the other values are shifted one unit to the right.

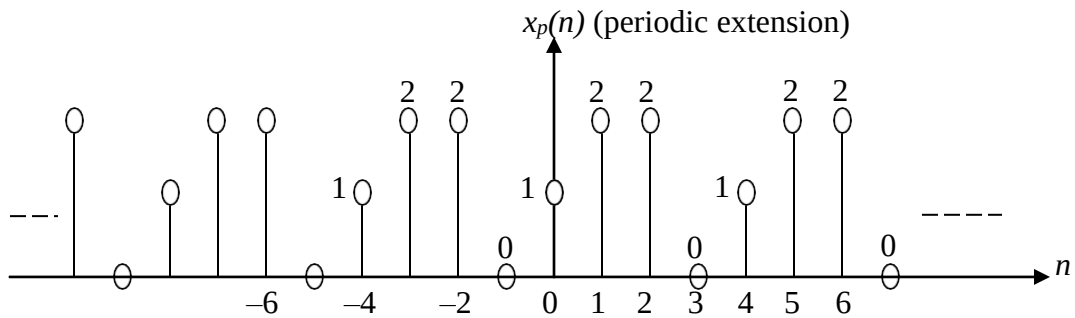
Alternatively, we can view this as a straight translation of the periodic extension outside the range [0, 3] of the given sequence. The periodic extension $x_p(n)$ is shown below:

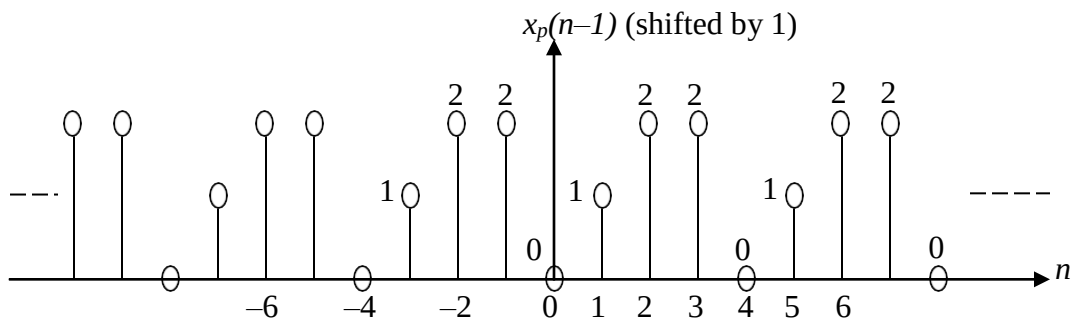
$$x_p(n) = \{\ldots\ 2,\ 2,\ 0,\ 1,\ 2,\ 2,\ 0,\ \underset{\substack{\uparrow \\ n=0}}{1},\ 2,\ 2,\ 0,\ 1,\ 2,\ 2,\ 0,\ 1,\ \ldots\}$$

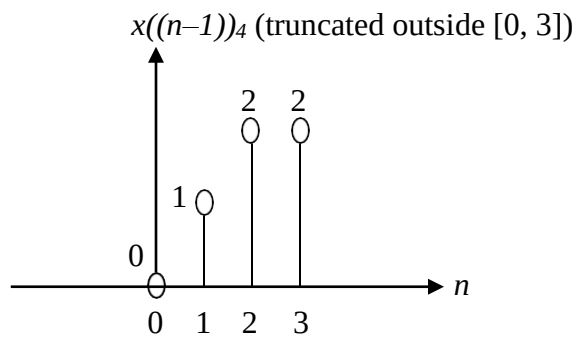
The periodic extension, when shifted to the right by one unit, appears as below; and the circularly shifted version $x((n-1))_4$ is the shaded part defined over $0 \le n \le 3$ only:

$$x_p(n-1) = \{\ldots\ 1,\ 2,\ 2,\ 0,\ 1,\ 2,\ 2,\ \underbrace{\overset{x((n-1))_4}{\overset{\uparrow}{\underset{n=0}{0}},\ 1,\ 2,\ 2,}}\ 0,\ 1,\ 2,\ 2,\ 0,\ \ldots\}$$

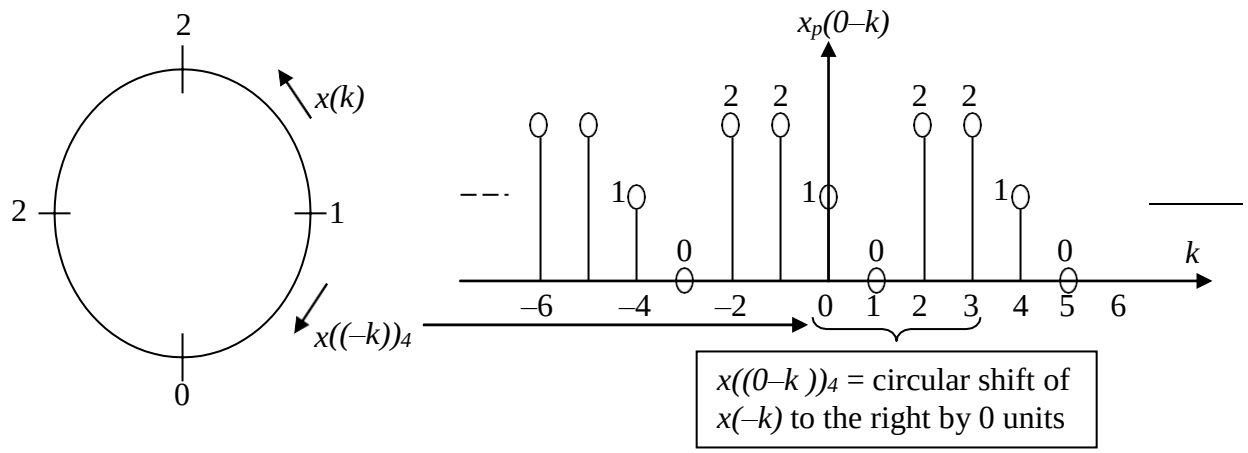


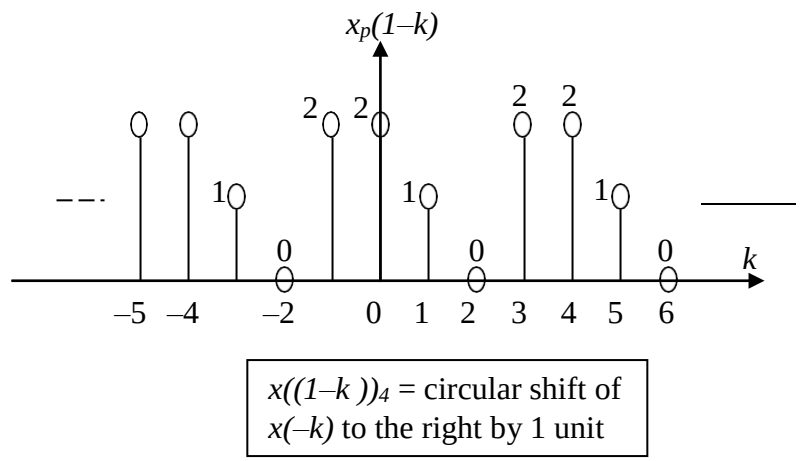






Example 2.5.2 Given $x(n) = \{1, 2, 2, 0\}$, sketch $x((n–k))_4$ where k is the independent variable.





DFT of circularly-shifted sequence Given that DFT $\{x(n)\} = X(k)$, then

$$\text{DFT } \{x((n–m))_N\} = W_N^{km} X(k)$$

Conversely, if the $X(k)$ is circularly shifted, the resulting inverse transform will be the multiplication of the inverse of $X(k)$ by a complex exponential: that is, if DFT $\{x(n)\} = X(k)$, then

$$\text{DFT } \{W_N^{-ln} x(n)\} = X((k–l))_N$$

Note from the above property that a shift in the frequency domain values $X(k)$ generally results in a *complex-valued* inverse sequence $x(n)$ even though the original sequence in the time domain could have been *real-valued*.

Circular convolution The N-point circular convolution of two sequences $x_1(n)$ and $x_2(n)$ denoted by $x_1(n) ©_N x_2(n)$ is defined as follows:

$$x_1(n) ©_N x_2(n) \equiv \sum_{k=0}^{N-1} x_1(k)\, x_2((n-k))_N = \sum_{k=0}^{N-1} x_1((n-k))_N\, x_2(k)$$

where $x_1((n-k))_N$ is the reflected and circularly translated version of $x_1(n)$. Note that k is the independent variable, so that $x_1(-k)$ is the reflected version and $x_1(-(k-n))_N$ is simply the reflected version shifted right by n units; the "mod N" makes it a circular shift instead of a linear shift.

If $X_1(k)$ and $X_2(k)$ represent the N-point DFTs of $x_1(n)$ and $x_2(n)$ respectively, i.e.,

$$X_1(k) = \mathrm{DFT}_N\{x_1(n)\} \quad \text{and} \quad X_2(k) = \mathrm{DFT}_N\{x_2(n)\}$$

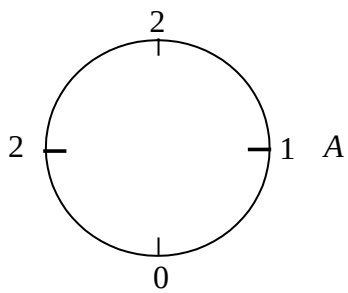
Then

$$\mathrm{IDFT}_N\{X_1(k)\, X_2(k)\} = x_1(n) \circledcirc_N x_2(n), \qquad \text{or} \qquad \mathrm{DFT}_N\{x_1(n) \circledcirc_N x_2(n)\} = X_1(k)\, X_2(k)$$

This property is used to perform circular convolution of two sequences by first obtaining their DFTs, multiplying the two DFTs, then taking the inverse DFT of the product.

Example 2.5.3 [Circular convolution] For the two sequences $x_1(n) = \{1, 2, 2, 0\}$ and $x_2(n) = \{0, 1, 2, 3\}$ find $y(n) = x_1(n) \circledcirc_4 x_2(n)$.

Solution We use the form $y(n) = x_1(n) \circledcirc_N x_2(n) = \sum_{k=0}^{3} x_1((n-k))_4\, x_2(k)$ which uses the circularly shifted version $x_1((n-k))_4$. The values of the sequence $x_1(k) = \{1, 2, 2, 0\}$ are arranged on a circle in counterclockwise direction starting at point A. The sequence $x_1((-k))_4$ is then read off in the clockwise direction starting at A. Thus $x_1((-k))_4 = \{1, 0, 2, 2\}$. See figure below.



As an alternative we may also obtain the sequence $x_1((-k))_4$ by *periodically* extending $x_1(k)_4$, reflecting it about $k = 0$, and truncating it outside the range $0 \le k \le 3$.

The value

$$y(0) = \sum_{k=0}^{3} x_1((0-k))_4\, x_2(k)$$

is obtained by lining up $x_1((-k))_4$ below $x_2(k)$, multiplying and adding:

$x_2(k)$	0	1	2	3
$x_1((-k))_4$	1	0	2	2

Thus $y(0) = (0)\,(1) + (1)\,(0) + (2)\,(2) + (3)\,(2) = 10$.

For $n = 1$ the value

$$y(1) = \sum_{k=0}^{3} x_1((1-k))_4\, x_2(k)$$

is obtained as follows: the sequence $x_1((1-k))_4$ is obtained from $x_1((-k))_4$ by shifting the latter to the right by 1 with wrap around; we then line up $x_1((1-k))_4$ below $x_2(k)$, multiply and add to get $y(1)$:

$x_2(k)$	0	1	2	3
$x_1((1-k))_4$	2	1	0	2

The result is $y(1)$ = (0) (2) +(1) (1) +(2) (0) +(3) (2) = 7.

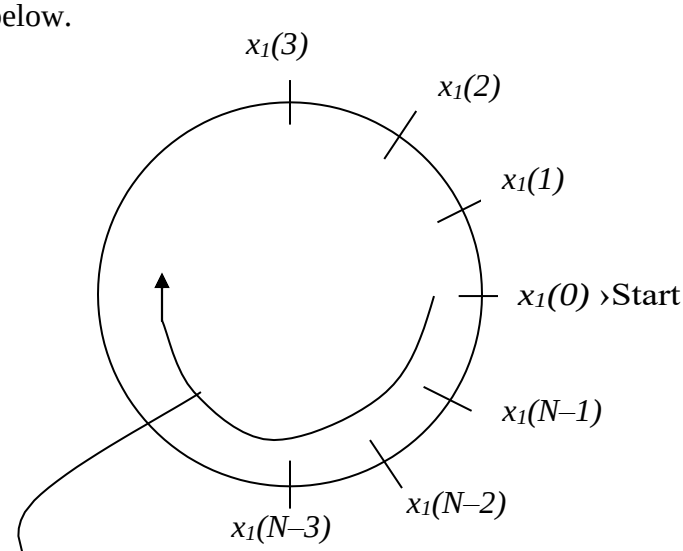
The procedure is continued for successive values of n, at each step using the circularly-shifted-by-1 version of the previous $x_1((n-k))_4$.

Circular convolution – Matrix method The circular convolution of the two sequences $x_1(n)$ and $x_2(n)$ is given by:

$$x_1(n) \circledcirc_N x_2(n) \equiv \sum_{k=0}^{N-1} x_1((n-k))_N x_2(k)$$

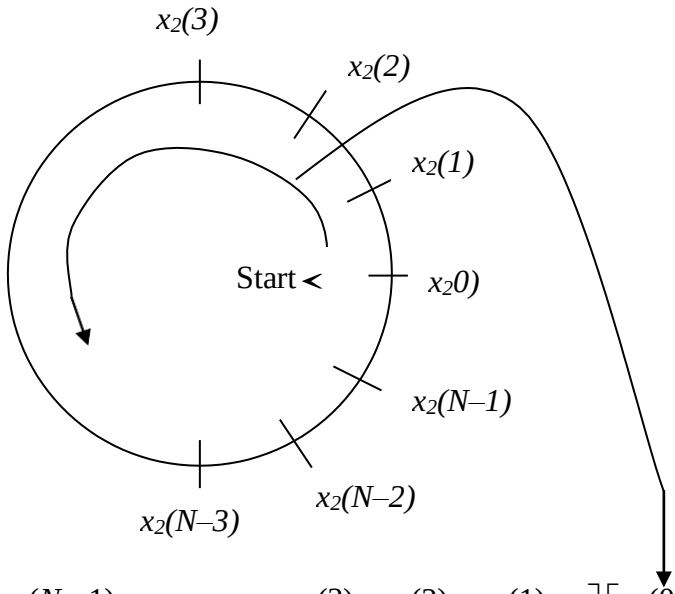
- Step 1. By *zero-padding* make sure the two sequences are of the same length, say, N.
- Step 2. Arrange the various circularly shifted versions of $x_1(.)$ as a matrix and $x_2(.)$ as a vector; then multiply to get the vector $x_3(.)$ which is the desired result.

The matrix formed by the shifted versions of $x_1(.)$ is shown below. It displays somewhat more terms than is possible to show in the complete multiplication equation shown farther down below.



$$\begin{bmatrix} x_1(0) & x_1(N-1) & x_1(N-2) & . & x_1(3) & x_1(2) & x_1(1) \\ x_1(1) & x_1(0) & x_1(N-1) & . & x_1(4) & x_1(3) & x_1(2) \\ x_1(2) & x_1(1) & x_1(0) & . & x_1(5) & x_1(4) & x_1(3) \\ . & . & . & . & . & . & . \\ x(N-2) & x(N-3) & x(N-4) & . & x(1) & x(0) & x(N-1) \\ x_1^1(N-1) & x_1^1(N-2) & x_1^1(N-3) & . & x_1^1(2) & x_1^1(1) & x_1^1(0) \end{bmatrix}$$

The complete multiplication step is shown below:

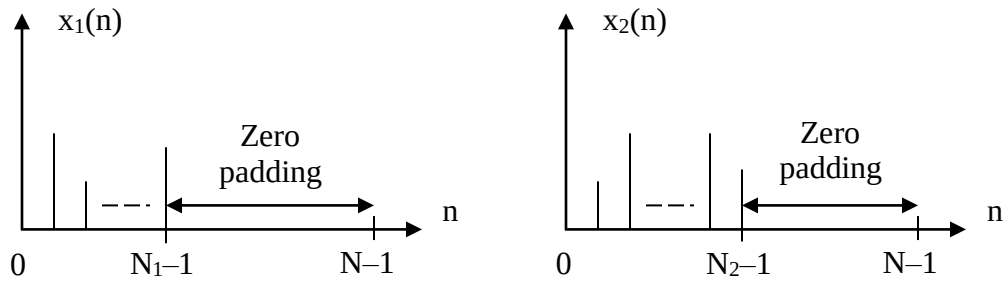


$$\begin{bmatrix} x_1(0) & x_1(N-1) & . & x_1(3) & x_1(2) & x_1(1) \\ x_1(1) & x_1(0) & . & x_1(4) & x_1(3) & x_1(2) \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ x_1(N-2) & x_1(N-3) & . & x_1(1) & x_1(0) & x_1(N-1) \\ x_1(N-1) & x_1(N-2) & . & x_1(2) & x_1(1) & x_1(0) \end{bmatrix} \begin{bmatrix} x_2(0) \\ x_2(1) \\ . \\ . \\ x_2(N-2) \\ x_2(N-1) \end{bmatrix} = \begin{bmatrix} x_3(0) \\ x_3(1) \\ . \\ . \\ x_3(N-2) \\ x_3(N-1) \end{bmatrix}$$

As an example the element $x_3(0)$ is given by

$$x_3(0) = x_1(0)\, x_2(0) + x_1(N-1)\, x_2(1) + \ldots + x_1(2)\, x_2(N-2) + x_1(1)\, x_2(N-1)$$

Carrying out circular convolution to obtain linear convolution If both signals $x_1(n)$ and $x_2(n)$ are of finite lengths N_1 and N_2 respectively, and defined on $[0, N_1–1]$ and $[0, N_2–1]$, respectively, as shown below, the value of N needed so that circular and linear convolution are the same on $[0, N–1]$ can be shown to be $N \geq N_1 + N_2 – 1$.



Example 2.5.4 [Circular and linear convolution] (a) Determine the 4-point circular convolution of the sequences

$$x_1(n) = [1, 2, 3, 1] \text{ and } x_2(n) = [4, 3, 2, 1]$$

(b) Evaluate the linear convolution of the above sequences.
(c) Evaluate the linear convolution of the above sequences using circular convolution.
Solution
(a) The 4-point circular convolution is given by

$$y(n) = x_1(n) \circledcirc_4 x_2(n) = \{15, 16, 21, 18\}$$

(b) The linear convolution was done in Unit I:

$$y(n) = x_1(n) * x_2(n) = \{4, 11, 20, 18, 11, 5, 1\}$$

(c) Length of sequence $x_1(n) = N_1 = 4$; length of sequence $x_2(n) = N_2 = 4$. Let $N = N_1 + N_2 – 1 = 4 + 4 – 1 = 7$ be the length of each of the zero-padded sequences $x_1'(.)$ and $x_2'(.)$.

$$x_1(n) \circledcirc_N x_2(n) \equiv \sum_{k=0}^{N-1} x_1((n-k))_N x_2(k)$$

Example 2.5.5 [2007] Compute the circular convolution of the sequences $x_1(n) = \{1, 2, 0, 1\}$ and $x_2(n) = \{2, 2, 1, 1\}$ using the DFT approach.
Solution The sequences are of the same length, so no zero padding is needed. The length of $\{x_1(n) \circledcirc_N x_2(n)\}$ is 4 (= N). Use the property that if $x_1(n) \leftrightarrow X_1(k)$ and $x_2(n) \leftrightarrow X_2(k)$ and $x_3(n) = x_1(n) \circledcirc_N x_2(n)$, then $x_3(n) \leftrightarrow X_3(k) = X_1(k)\, X_2(k)$:

$$x_3(n) = x_1(n) \circledcirc_N x_2(n) = \text{IDFT}\{X_3(k)\} = \text{IDFT}_N\{X_1(k)\, X_2(k)\} \text{ with } N = 4$$

where $X_1(k)$ and $X_2(k)$ are the N-point DFTs of $x_1(n)$ and $x_2(n)$, respectively. The following steps are involved in computing $x_1(n) \circledcirc_N x_2(n)$:

1. Find $X_1(k) = \text{DFT}_4\{x_1(n)\}$ and $X_2(k) = \text{DFT}_4\{x_2(n)\}$
2. Compute the product $X_1(k)\, X_2(k)$

3. Compute $x_1(n) \circledcirc_N x_2(n) = \text{IDFT}\{X_1(k) X_2(k)\}$

Example 2.5.6 Compute the linear convolution of the sequences $x(n) = \{1, 2, 0, 1\}$ and $y(n) = \{2, 2, 1, 1\}$ using the DFT approach.
Solution The length of $x(n)*y(n)$ is 7 (= 4+4–1). We zero-pad the sequences to a length of 7 each and perform circular convolution of the 7-point sequences; the result will be the same as the linear convolution of the original 4-point sequences. The following steps are involved in computing $x(n)*y(n)$:

1. Augment the sequences $x(.)$ and $y(.)$ by zero-padding: $x_a(n) = \{1, 2, 0, 1, 0, 0, 0\}$ and $y_a(n) = \{2, 2, 1, 1, 0, 0, 0\}$
2. Find $X_a(k) = \text{DFT}_7\{x_a(n)\}$ and $Y_a(k) = \text{DFT}_7\{y_a(n)\}$.
3. Compute the product $X_a(k)\, Y_a(k)$
4. Compute $x(n)*y(n) = x(n) \circledcirc_7 y(n) = \text{IDFT}\{X(k)\, Y(k)\}$

Example 2.5.7 Compute the linear convolution of the sequences $x(n) = \{1, 2\}$ and $y(n) = \{2, 2, 1\}$ using the DFT approach.
Solution The length of $x(n)*y(n)$ is 4 (= 2+3–1). We zero-pad the sequences to a length of 4 each and perform circular convolution of the 4-point sequences; the result will be the same as the linear convolution of the original 2- and 3-point sequences. The following steps are involved in computing $x(n)*y(n)$:

1. Augment the sequences $x(.)$ and $y(.)$ by zero-padding: $x_a(n) = \{1, 2, 0, 0\}$ and $y_a(n) = \{2, 2, 1, 0\}$
2. Find $X_a(k) = \text{DFT}_4\{x_a(n)\}$ and $Y_a(k) = \text{DFT}_4\{y_a(n)\}$.
3. Compute the product $X_a(k)\, Y_a(k)$
4. Compute $x(n)*y(n) = x_a(n) \circledcirc_4 y_a(n) = \text{IDFT}\{X_a(k)\, Y_a(k)\}$

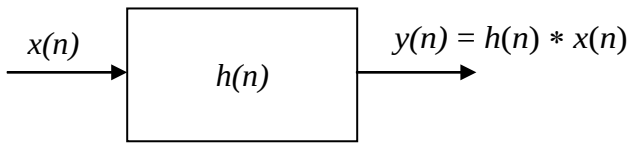
Convolution – Overlap-and-add The response, $y(n)$, of a LTI system, $h(n)$, can be obtained by linear convolution

$$y(n) = h(n) * x(n)$$

Let the impulse response $\{h(n), n = 0 \text{ to } M-1\}$ be of finite length M. The input sequence $\{x(n), n = 0 \text{ to } S-1\}$ is long but of finite length S. Recall that

$$\text{Length } \{y(n)\} = \text{Length } \{h(n)\} + \text{Length } \{x(n)\} - 1 = M + S - 1,$$

Further, let $h(n)$ be defined to be zero everywhere except over the interval $[N_1, N_2]$. Similarly, let $x(n)$ be defined to be non-zero over $[N_3, N_4]$. Then $y(n)$ is non-zero over $[(N_1 + N_3), (N_2 + N_4)]$.



One way to perform the convolution in pseudo real time (i.e., real time with a finite delay) is by sectionalizing the input. We divide $x(n)$ into K sections of length M each, where $K = \lceil S/M \rceil$:

$$x_1(n) = \begin{cases} x(n), & 0 \le n \le M-1 \\ 0, & \text{elsewhere} \end{cases}$$

…

$$x_i(n) = \begin{cases} x(n), & (i-1)M \le n \le iM-1 \\ 0, & \text{elsewhere} \end{cases}$$

…

$$x_K(n) = \begin{cases} x(n), & (K-1)M \le n \le KM-1 \\ 0, & \text{elsewhere} \end{cases}$$

In the Kth section (the last section) zeros may have to be appended. If, for instance, $x(n) = \{3, -1, 0, 1, 3, 2, 0, 1, 2, 1\}$ with $S = 10$ and $h(n) = \{1, 1, 1\}$ with $M = 3$, we have $K = \lceil 10/3 \rceil = 4$, with the 4th section containing two appended zeros, and the sections are

$$x_1(n) = \{3, -1, 0\}$$
$$x_2(n) = \{1, 3, 2\}$$
$$x_3(n) = \{0, 1, 2\}$$
$$x_4(n) = \{1, 0, 0\}$$

In general, then, $x(n)$ can be written as the sum of all the sections

$$x(n) = \sum_{i=1}^{K} x_i(n)$$

and the output $y(n)$ becomes

$$y(n) = h(n) * x(n) = h(n) * \left\{ \sum_{i=1}^{K} x_i(n) \right\}$$

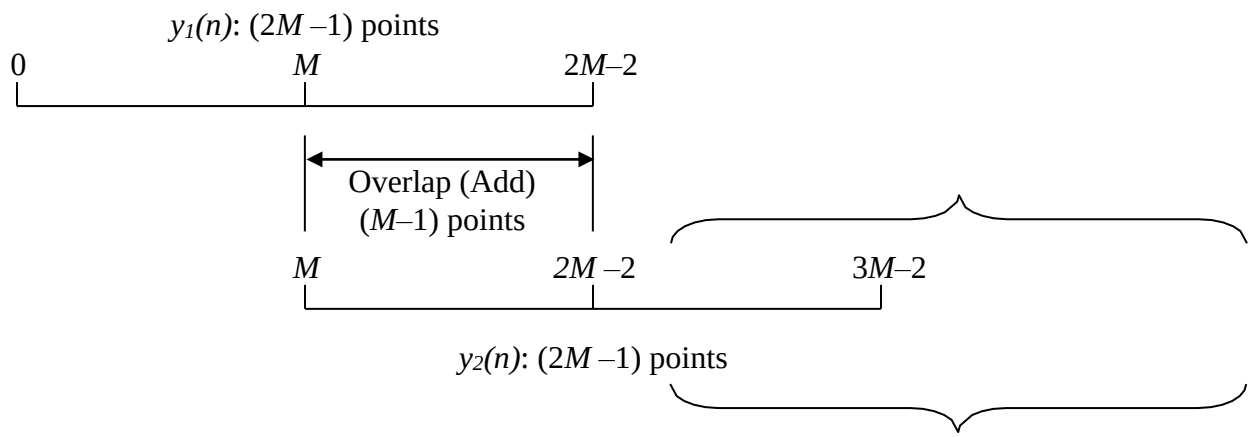
Using the linearity property this becomes

$$y(n) = \sum_{i=1}^{K} \{x_i(n) * h(n)\} = \sum_{i=1}^{K} y_i(n)$$

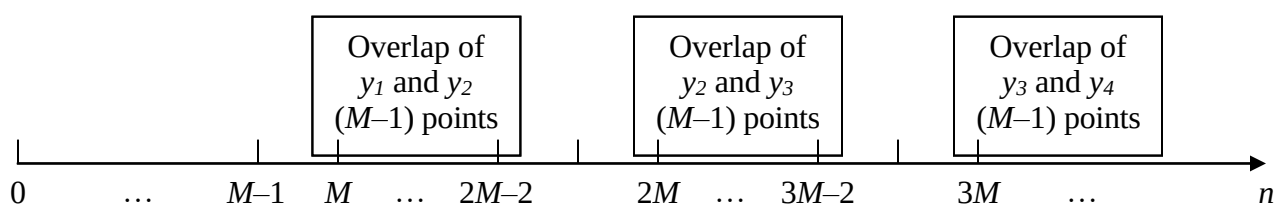
where $y_i(n) = x_i(n) * h(n)$ are the output sections. Let us examine $y_1(n)$ and $y_2(n)$. For $i = 1$ we have

$$y_1(n) = x_1(n) * h(n)$$

Since $x_1(n)$ and $h(n)$ are of lengths M each and they are defined to be non-zero over [0, M–1] and [0, M–1] respectively, the result $y_1(n)$ will be non-zero over [0, 2M–2] and of length (2M–1). Similarly, for $i = 2$, $x_2(n)$ is defined over [M, 2M–1] while $h(n)$ remains unchanged. The resulting $y_2(n)$ then is non-zero from (0+M) to (M–1 + 2M–1), i.e., over [M, 3M–2] with a length of (2M–1). Comparing $y_1(n)$ and $y_2(n)$ it is seen that they overlap in the interval $M \le n \le (2M–2)$, over a range of (2M–2) – M +1 = M–1 points. Consequently the two must be added in this range (see figure). This amounts to adding (M–1) pairs of data.



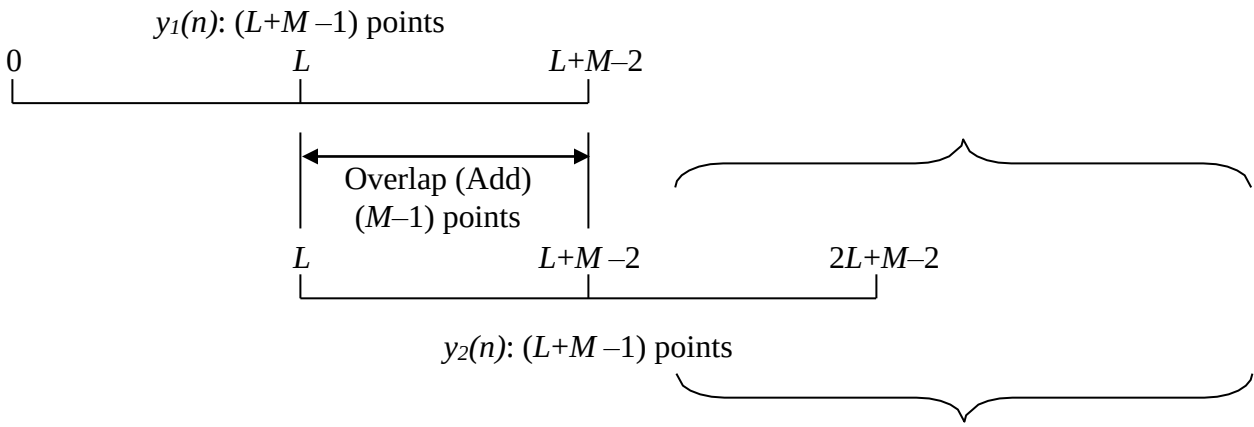
In a similar fashion $x_3(n)$ is defined over $2M \le n \le (3M–1)$, so that $y_3(n)$ is non-zero over [2M, (4M–2)]. Comparing $y_2(n)$ and $y_3(n)$ it is seen that they overlap in the interval $2M \le n \le (3M–2)$, consequently the two must be added in this range. This amounts to adding ((3M–2) – 2M +1) or (M–1) pairs of data.



The overlap interval of $y_1(n)$ and $y_2(n)$ is disjoint from that of $y_2(n)$ and $y_3(n)$. In general, the overlap intervals of successive pairs of $y_i(n)$ are mutually exclusive. Thus we calculate successive $y_i(n)$ for $i = 0$ to K and add each successive $y_i(n)$ to the previous $y_i(n)$ in

the overlap region. Hence the procedure is called the overlap-and-add method. Each convolution could be obtained by using the DFT of size ($2M$–1) or greater so that the resulting circular convolution would be a linear convolution. In principle rather than using the DFT we could zero-pad $h(n)$ and each of the $x_i(n)$'s to a length of ($2M$–1) and perform circular convolution to generate the $y_i(n)$'s which are then overlapped and added.

Input divided into sections of length L In the above development we divided $x(n)$ into K sections of length M each, where $K = \lceil S/M \rceil$. This need not be the case. We could divide $x(n)$ into (some number of) sections of length L each. The situation now looks as below and the overlap occurs over a range of (M–1) points – the same as before.



Once again each new section $x_i(n)$ and $h(n)$ are zero-padded to a length of (L+M –1) and circular convolution performed to generate the new $y_i(n)$'s which are overlapped and added to generate $y(n)$.

Note A little reflection shows that the input sequence $x(n)$ need not be of finite length. As the stream of input samples arrives we could sectionalize it into blocks of size L and proceed as discussed above to generate the stream of blocks of $y(n)$ as a continuous process.

Example 2.5.7 [Ramesh Babu's Example 3.14] Find the output $y(n)$ of a filter with impulse response $h(n)$ = {1, 1, 1} and input $x(n)$ = {3, –1, 0, 1, 3, 2, 0, 1, 2, 1}.

Symmetry properties of the DFT Notation: $R_N(n)$ = 1 in [0, N–1], and 0 elsewhere. Thus $x((n+m))_N R_N(n)$ means the circularly shifted version of the finite length sequence $x(n)$ defined over [0, N–1]. Sometimes the $R_N(n)$ is omitted.

The following properties should be noted.

	Sequence	DFT
1	$x((n+m))_N R_N(n)$	$W_N^{-km} X(k)$
2	$x^*(n)$	$X^*((-k))_N R_N(k)$
3	$x^*((-n))_N R_N(n)$	$X^*(k)$

	Sequence	DFT
4	Re [$x(n)$]	$X_{ep}(k)$
5	j Im [$x(n)$]	$X_{op}(k)$

Example 2.5.8 Show that DFT$\{x((n+m))_N\} = W_N^{-km} X(k)$.

Solution By definition

$$\text{DFT}\ \{x((n+m))_N\} = \sum_{n=0}^{N-1} x((n+m))_N\ W_N^{kn}$$

Set $n+m = \lambda$ so that $n = \lambda - m$ and the limits $n = 0$ to $N-1$ become $\lambda = m$ to $N-1+m$. Then the RHS becomes

$$= \sum_{\lambda=m}^{N-1+m} x((\lambda))_N\ W_N^{k\lambda}\ W_N^{-km}$$

The limits on λ will be changed to 0 to $N-1$ resulting in

$$= \sum_{\lambda=0}^{N-1} x((\lambda))_N\ W_N^{k\lambda}\ W_N^{-km} = W_N^{-km} X(k)$$

Based on the properties above, we can show that for a real sequence the following symmetry properties of the DFT hold:

1. Re $[X(k)]$ = Re $[X((-k))_N]\ R_N(k)$
2. Im $[X(k)]$ = –Im $[X((-k))_N]\ R_N(k)$
3. $|X(k)| = |X((-k))_N|\ R_N(k)$
4. arg $[X(k)]$ = – arg $[X((-k))_N]\ R_N(k)$

Example 2.5.9 [2009] Given that the real-valued sequence $x(n)$ defined over $0 \le n \le N-1$ has the DFT $X(k) = X_R(k) + jX_I(k)$, $0 \le k \le N-1$ show that $X_R(k)$ is an even function and $X_I(k)$ is an odd function of k.

Solution By definition we have

$$X(k) = \sum_{n=0}^{N-1} x(n)\, e^{-jk2\pi n/N}, \qquad 0 \le k \le N-1$$

$$= \sum_{n=0}^{N-1} x(n)\left\{\cos(2\pi kn/N) - j\sin(2\pi kn/N)\right\}, \quad 0 \le k \le N-1$$

$$= \sum_{n=0}^{N-1} x(n)\cos(2\pi kn/N) - j\sum_{n=0}^{N-1} x(n)\sin(2\pi kn/N), \qquad 0 \le k \le N-1$$

With $X_R(k) = \sum_{n=0}^{N-1} x(n)\cos(2\pi kn/N)$ and $X_I(k) = -\sum_{n=0}^{N-1} x(n)\sin(2\pi kn/N)$ the above DFT may be written

$$X(k) = X_R(k) + jX_I(k), \qquad 0 \le k \le N-1$$

Since $\cos(2\pi kn/N)$ is an even function of k, that is, $\cos(2\pi(-k)n/N) = \cos(2\pi kn/N)$ for all k, it follows that $X_R(k)$ is an even function of k, that is, $X_R(-k) = X_R(k)$ for all k.

Similarly, since $\sin(2\pi kn/N)$ is an odd function of k, it follows that $X_I(k)$ is an odd function of k.

Do the above results depend on whether $x(n)$ is real-valued or not?

Filtering through DFT/FFT

Filtering of a sequence $x(n)$ may be done in the discrete time domain using the difference equation. Alternatively, we may work in the frequency domain: Given $x(n)$ we first find its DFT, $X(k)$ and then set selected components of $X(k) = 0$ (this is done so as to preserve the symmetry

properties of $X(k)$ about the mid point of the sequence $k = N/2$). The resulting DFT is denoted $X_f(k)$. We then find the IDFT of $X_f(k)$ which we shall denote as $x_f(n)$, which should be a filtered version of $x(n)$. Thus $x(n)$ has been filtered entirely in the discrete frequency domain.

We illustrate with a signal consisting of two frequency components, a 2 Hz and a 4 Hz. Given the signals $x_1(t) = \cos 2\pi 2t$ and $x_2(t) = \cos 2\pi 4t$, the signal $x(t) = x_1(t) + x_2(t)$ is sampled at 16 Hz. We construct below the discrete-time sequence $x(n)$ and find its 8-point DFT, i.e., the $X(k)$ values. We then filter the signal in the frequency domain, i.e., work on the DFT values instead of on the $x(n)$ values and denote the "filtered" DFT values by $X_f(k)$. We then take the IDFT of $X_f(k)$ resulting in the sequence $x_f(n)$.

$$x(t) = \cos 2\pi 2t + \cos 2\pi 4t$$
$$x(nT) = x(n) = \cos 2\pi 2nT + \cos 2\pi 4nT = \cos 2\pi 2n(1/16) + \cos 2\pi 4n(1/16)$$
$$= \cos(\pi n/4) + \cos(\pi n/2)$$

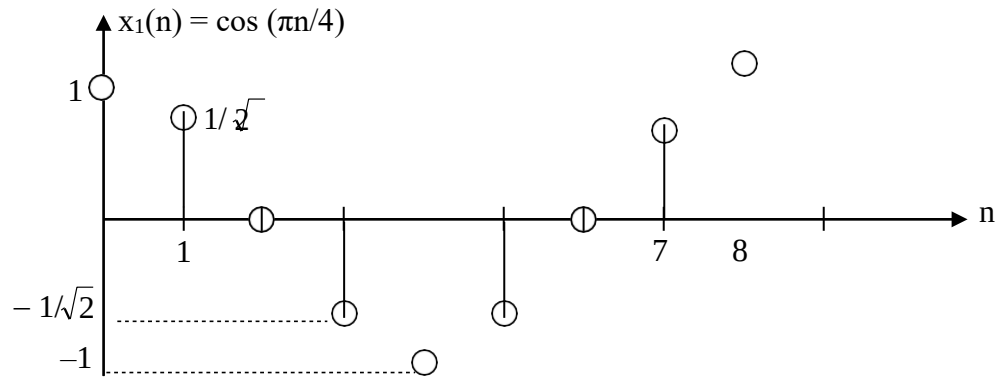
The $\cos 2\pi 2t$ component has an analog frequency of 2 Hz. When sampled at 16 Hz its digital frequency is 1/8 cycles/sample. Similarly, the 4 Hz component, $\cos 2\pi 4t$, sampled at 16 sps, has a digital frequency of 1/4 cycles/sample.

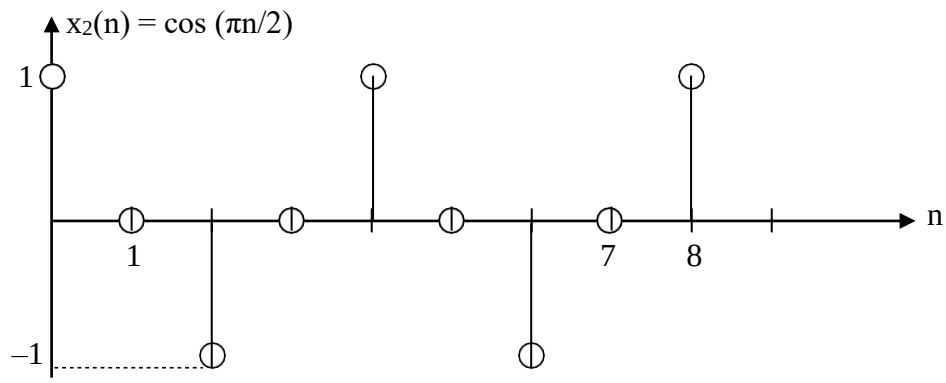
Example 2.6.1 (a) Find the frequency and period of (i) $x_1(n) = \cos(\pi n/4)$ and (ii) $x_2(n) = \cos(\pi n/2)$. Sketch the sequences $x_1(n)$, $x_2(n)$, and $x(n) = x_1(n) + x_2(n)$ for $0 \le n \le 7$.

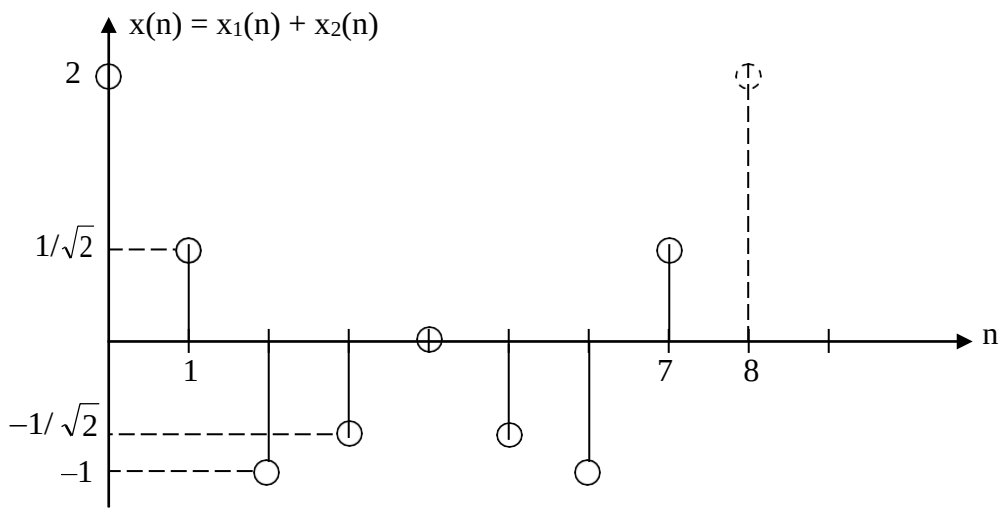
Solution Arrange $\cos(\pi n/4)$ in the format $\cos(2\pi f n)$. Thus, $\cos(\pi n/4) = \cos(2\pi(1/8)n)$, from which the digital frequency is identified as $f = 1/8$ cycle/sample or $\omega = \pi/4$ rad/sample. The sequence values are:

$$x_1(n) = \left\{1, \frac{1}{\sqrt{2}}, 0, -\frac{1}{\sqrt{2}}, -1, -\frac{1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}}\right\}$$
$$x_2(n) = \{1, 0, -1, 0, 1, 0, -1, 0\}$$
$$x(n) = x_1(n) + x_2(n) = \left\{2, \frac{1}{\sqrt{2}}, -1, -\frac{1}{\sqrt{2}}, 0, -\frac{1}{\sqrt{2}}, -1, \frac{1}{\sqrt{2}}\right\}$$

The sequences $x_1(n)$, $x_2(n)$ and $x(n)$ are sketched below.

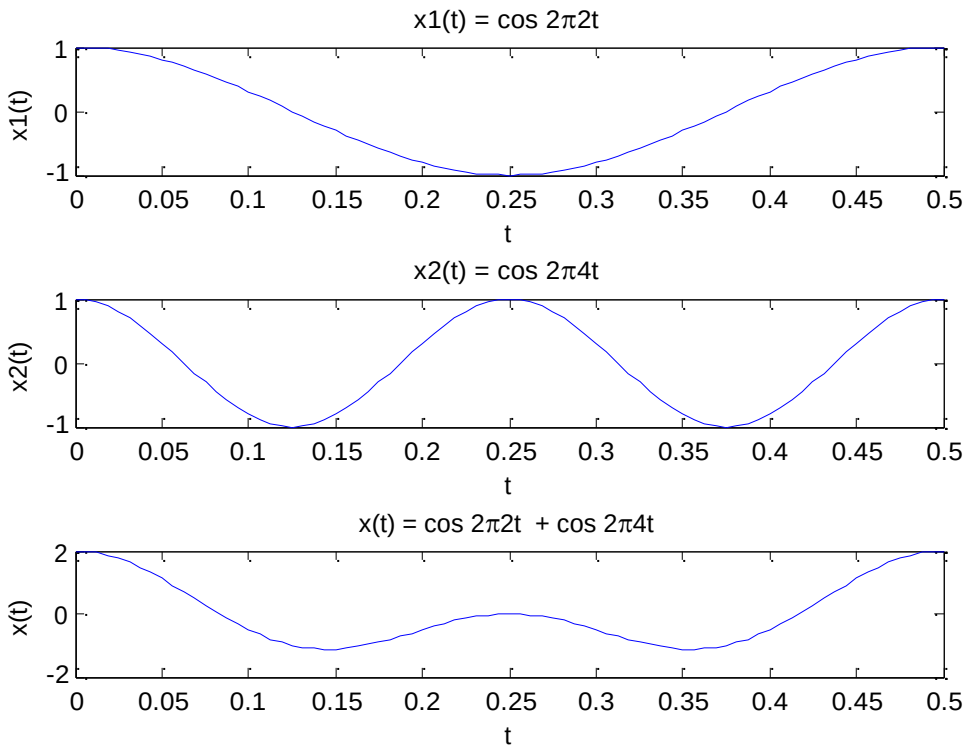






In MATLAB the following segment plots the three functions $x_1(t)$, $x_2(t)$ and $x(t)$.

```
t = 0:1/160:0.5; x1t = cos(2*pi*2*t); x2t = cos(2*pi*4*t);
xt = x1t + x2t;
%
subplot(3,1,1), plot(t,x1t); xlabel('t'), ylabel('x1(t)');
title('x1(t) = cos 2\pi2t');
%
subplot(3,1,2), plot(t,x2t); xlabel('t'), ylabel('x2(t)');
title('x2(t) = cos 2\pi4t');
%
subplot(3,1,3), plot(t,xt); xlabel('t'), ylabel('x(t)');
title('x(t) = cos 2\pi2t + cos 2\pi4t');
```

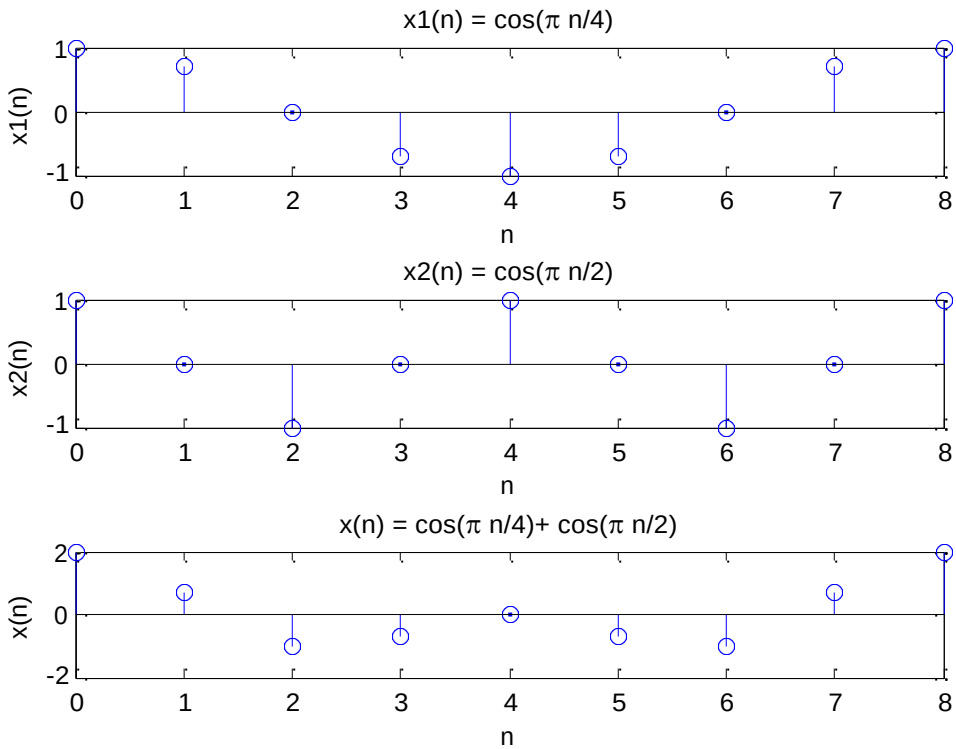


In MATLAB the following segment plots the two functions $x_1(n)$, $x_2(n)$ and $x(n)$.

```
%t = 0:1/160:0.5; x1t = cos(2*pi*2*t); x2t = cos(2*pi*4*t);
%xt = x1t + x2t;
%
n = 0:8; x1n = cos(pi*n/4); x2n = cos(pi*n/2);
xn = x1n + x2n,
%
subplot(3,1,1), stem(n,x1n); xlabel('n'), ylabel('x1(n)');
title('x1(n) = cos(\pi n/4)');
%
```

```
subplot(3,1,2), stem(n,x2n); xlabel('n'), ylabel('x2(n)');
title('x2(n) = cos(\pi n/2)');
%
subplot(3,1,3), stem(n,xn); xlabel('n'), ylabel('x(n)');
title('x(n) = cos(\pi n/4)+ cos(\pi n/2)');
```

The sequence is x(n) = {2, 0.707, -1, -0.707, 0, -0.707, -1, 0.707}



Example 2.6.1 (b) Find the 8-point DFT (8-point FFT using either DIT or DIF, or use the direct calculation) of $x_1(n) = \cos(\pi n/4)$, for $0 \le n \le 7$. Sketch the sequence $X_1(k)$.

In MATLAB the following segment plots the magnitude and phase angle of $X_1(k)$:

```
n = 0:7; x1n = cos(pi*n/4), X1k = fft(x1n),
%
% MX1k = magnitude of X1k, AX1k = phase angle of X1k
MX1k = abs(X1k); AX1k = angle(X1k);
%
subplot(2,1,1), stem(n, abs(X1k)); xlabel('k'), ylabel('|X1(k)|');
title ('Magnitude of X1(k)');
%
subplot(2,1,2), stem(n, angle(X1k)); xlabel('k'), ylabel('<X1(k)');
title ('Phase of X1(k)');
```

The sequence and DFT values are:

$x_1(n)$ = {1 0.707 0 -0.707 -1 -0.707 -0 0.707}
$X_1(k)$ = {-0 (4 - 0i) 0 (-0 - 0i) 0 (-0 + 0i) 0 (4 + 0i)}

$$|X_1(k)| = \{0 \quad 4 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 4\}$$
$$\angle X_1(k) = \{3.1416 \quad -0.0000 \quad 0 \quad -2.9873 \quad 0 \quad 2.9873 \quad 0 \quad 0.0000\}$$

!!! Note that all phase angle values should be zeros, that is, $\angle X_1(k) = 0$ for all k.

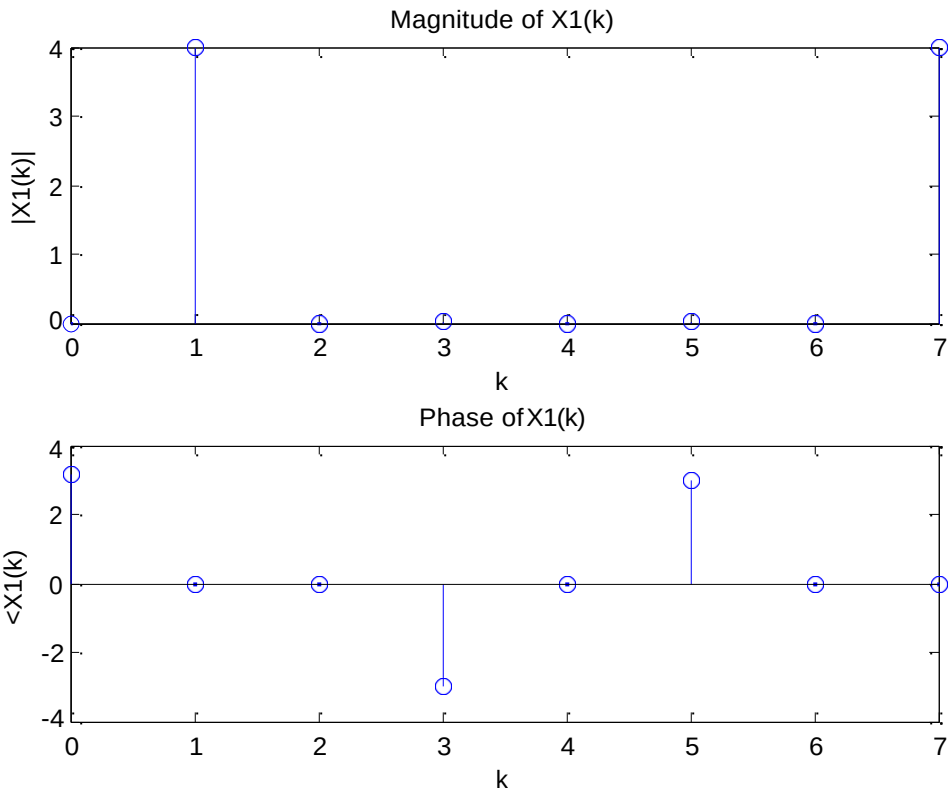
The MATLAB plots for magnitude and phase angle are shown below. The 2 Hz component is indicated by $X_1(1) = 4$ (and, from symmetry considerations, $X_1(7) = 4$).

With regard to the phase angle, strictly speaking, the phase of $X_1(k) = 0$ for all k since $X_1(k)$ is real valued for all k. ***Check out why!!!***: When the input is listed explicitly as

```
x1n = [1, 1/sqrt(2), 0, -1/sqrt(2), -1, -1/sqrt(2), 0, 1/sqrt(2)]
```

the program gives the correct *phase angle* calculations, but not when it is specified implicitly as

```
n = 0:7; x1n = cos(pi*n/4)
```



Example 2.6.1 (c) Find the 8-point DFT of $x(n) = \cos(\pi n/4) + \cos(\pi n/2)$, for $0 \le n \le 7$. Sketch the sequence $X(k)$.

Solution Consider the 8-point sequence $x(n)$ for $n = 0$ to 7 obtained by sampling $x(t)$ at 16 Hz. The length of the sequence is $N = 8$. The sequence values are

$$x(n) = \left\{2, \frac{1}{\sqrt{2}}, -1, -\frac{1}{\sqrt{2}}, 0, -\frac{1}{\sqrt{2}}, 1, \frac{1}{\sqrt{2}}\right\}$$

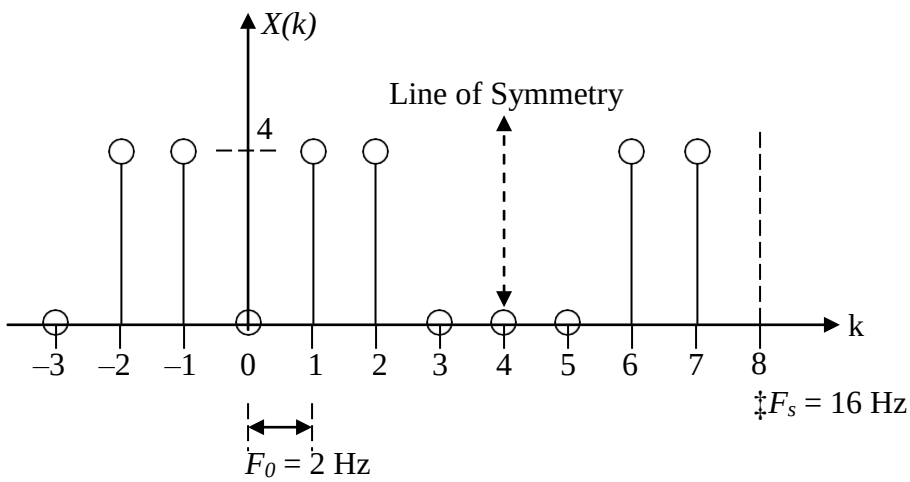
The corresponding DFT is given by

$$X(k) = \sum_{n=0}^{N-1} x(n)\, e^{-j2\pi kn/N}, \qquad k = 0 \text{ to } (N-1)$$

and may be obtained using either the DIT or the DIF form of FFT:

$$X(k) = \{\underset{\substack{\uparrow \\ k=0}}{0}, \quad 4, \quad 4, \quad 0, \quad 0, \quad 0, \quad 4, \quad 4\}$$

The DFT is sketched below: there is a component at 2 Hz. (k = 1) and another at 4 Hz. (k = 2) as would be expected.



Note that the DFT sequence shows a component at 2 Hz and another at 4 Hz, corresponding to k = 1 and 2 respectively.

In MATLAB the following segment plots the magnitude and phase angle of $X(k)$. Note that $x(n)$ is specified by an explicit list.

```
%n = 0:7; xn = cos(pi*n/4) + cos(pi*n/2),
xn = [2, 1/sqrt(2), -1, -1/sqrt(2), 0, -1/sqrt(2), -1, 1/sqrt(2)], Xk = fft(xn),
%
% MXk = magnitude of Xk, AXk = phase angle of Xk
MXk = abs(Xk), AXk = angle(Xk),
%
k = 0:7;
subplot(2,1,1), stem(k, abs(Xk)); xlabel('k'), ylabel('|X(k)|');
title ('Magnitude of X(k)');
%
subplot(2,1,2), stem(k, angle(Xk)); xlabel('k'), ylabel('<X(k)');
title ('Phase of X(k)');
```

The sequence and DFT values are:

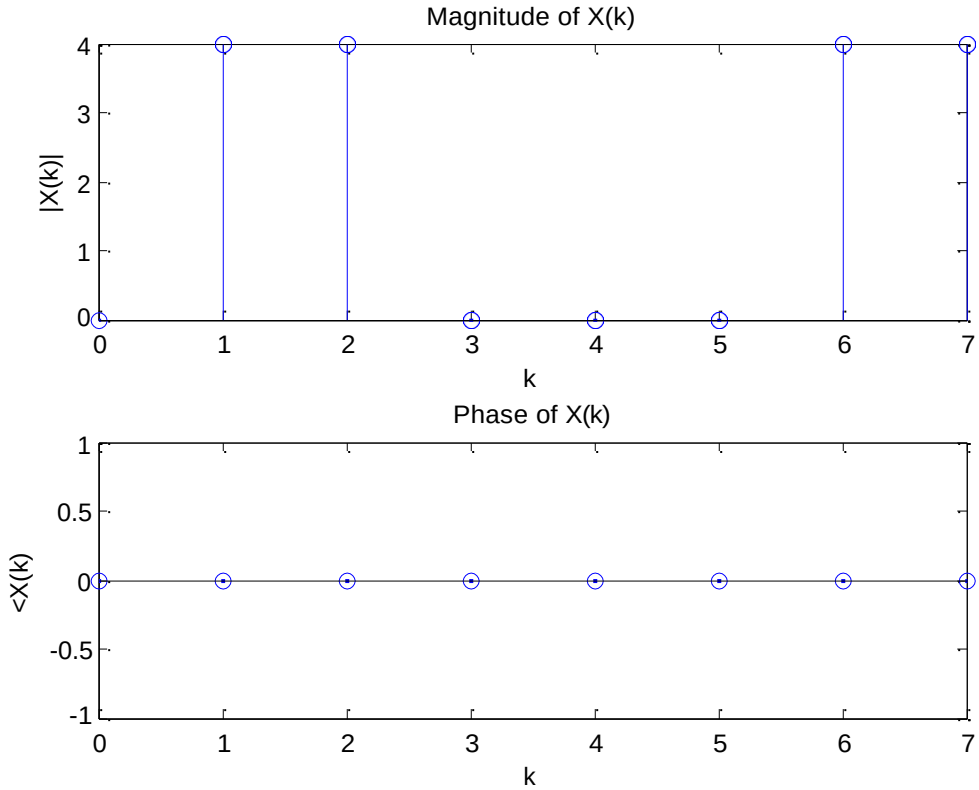
$x(n) = \{2, 0.707, -1, -0.707, 0, -0.707, -1, 0.707\}$

$X(k) = \{0 \quad 4 \quad 4 \quad 0 \quad 0 \quad 0 \quad 4 \quad 4\}$

$|X(k)| = \{0\ 4\ 4 \quad 0\ 0\ 0\ 4\ 4\}$

$\angle X(k) = \{0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0\}$

The MATLAB plots for magnitude and phase angle are shown below. The 2 Hz component is indicated by $X(1) = 4$ (and, from symmetry considerations, $X(7) = 4$). Similarly, the 4 Hz component is indicated by $X(2) = X(6) = 4$.



Example 2.6.1 (d) [Filtering] Next we would like to filter the sequence $x(n) = \cos(\pi n/4) + \cos(\pi n/2)$ so that the 4 Hz component is removed.

Solution In terms of the DFT sequence $X(k)$ this means setting $X(2)$ and $X(6)$ to zero. In order to preserve the symmetry properties of the DFT we should set *both* $X(2)$ and $X(6) = 0$, not just $X(2)$. The resulting DFT sequence is denoted $X_f(k)$ and is given by

$$X_f(k) = \{\underset{\substack{\uparrow \\ k=0}}{0}, \quad 4, \quad 0, \quad 0, \quad 0, \quad 0, \quad 0, \quad 4\}$$

We next find the inverse DFT of the above $X_f(k)$ by using either the DIT or the DIF form of the FFT. The result is denoted by $x_f(n)$. Using the IDFT formula we have

$$x_f(n) = \frac{1}{N}\sum_{k=0}^{N-1} X_f(k) e^{j2\pi kn/N}, \qquad n = 0, 1, \ldots, N-1$$

$$= \frac{1}{8}\sum_{k=0}^{7} X_f(k) e^{j2\pi kn/8}, \qquad n = 0 \text{ to } 7$$

It will be seen that $x_f(n)$ is equal to the original $x_1(n)$ component (= cos πn/4, from the 2 Hz. component; see plot of $x_1(n)$ earlier); that is

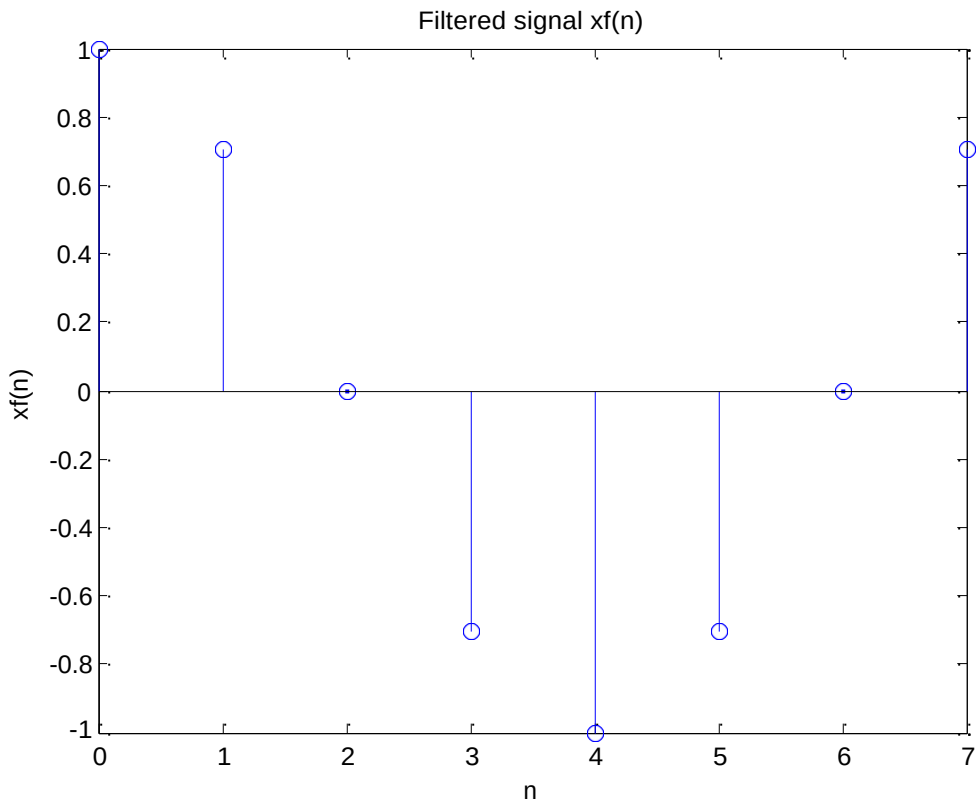
$$x_f(n) = x_1(n) = \left\{1, \frac{1}{\sqrt{2}}, 0, -\frac{1}{\sqrt{2}}, -1, -\frac{1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}}\right\}$$

This is low pass filtering where we have selectively removed the 4 Hz component.

In MATLAB the following segment finds the inverse DFT, $x_f(n)$, from the given $X_f(k)$.

```
Xfk = [0, 4, 0, 0, 0, 0, 0, 4],
n = 0:7; xfn = ifft(Xfk),
stem(n, xfn); xlabel('n'), ylabel('xf(n)');
title ('Filtered signal xf(n)');
```

The filtered version is $x_f(n)$ = {1 0.707 0 -0.707 -1 -0.707 0 0.707}



To remove all frequency components above 2 Hz (in this example the 4, 6 and 8 Hz components), we set $X(2) = X(6) = 0$ for the 4 Hz, $X(3) = X(5) = 0$ for the 6 Hz, and $X(4) = 0$ for the 8 Hz, once again preserving symmetry. In this example, of course, there are no 6 or 8 Hz components.

Similarly high pass filtering is done by deleting $X(1)$ and $X(7)$ – set them to zero – preserving symmetry once again. In this case $X_{fHP}(k)$ = {0, 0, 4, 0, 0, 0, 4, 0}.

Picket-fence effect

Example 2.7.1 The signal $x(t) = \cos 2\pi 2t$ is sampled at 16 Hz.
(a) What frequency components do you expect to see in its DFT?
(b) Take 8 samples and calculate the 8-point DFT.
Solution (b) Using $x(n) = \cos 2\pi 2n(1/16) = \cos(\pi n/4)$, the sequence values are

$x(n)$ = {1, 1/sqrt(2), 0, -1/sqrt(2), -1, -1/sqrt(2), 0, 1/sqrt(2)}

Note that the average value of the sequence (the dc component) is zero. The MATLAB program follows:

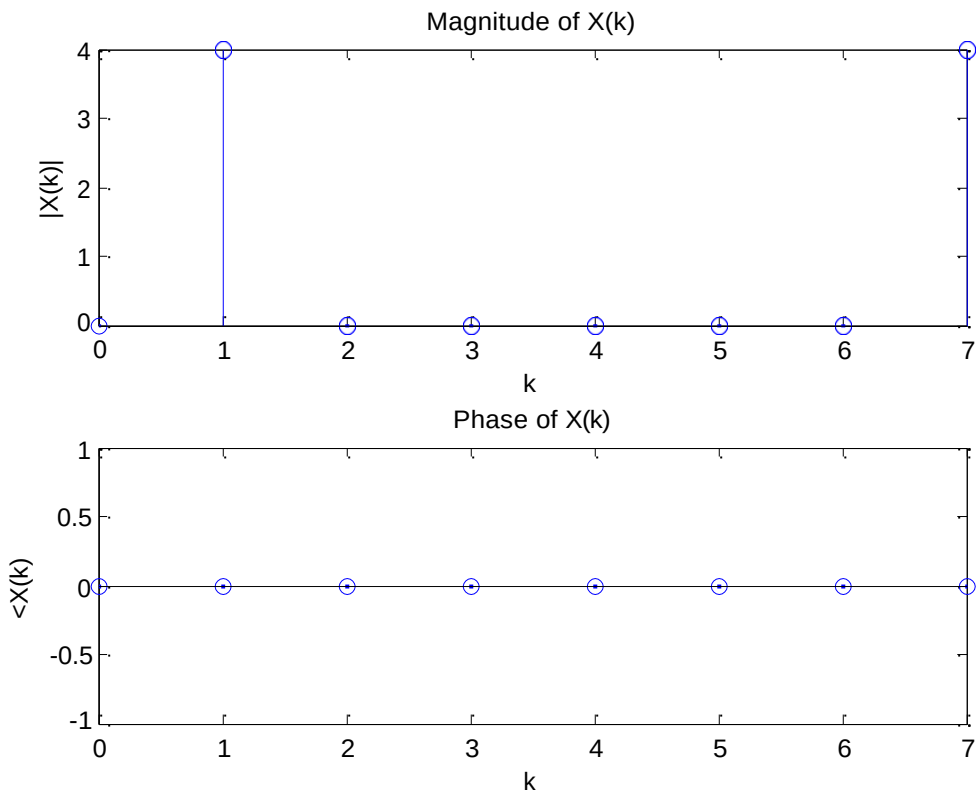
```
xn = [1, 1/sqrt(2), 0, -1/sqrt(2), -1, -1/sqrt(2), 0, 1/sqrt(2)]; Xk = fft(xn),
%
% MXk = magnitude of Xk, AXk = phase angle of Xk
MXk = abs(Xk); AXk = angle(Xk);
%
k = 0:7;
subplot(2,1,1), stem(k, abs(Xk)); xlabel('k'), ylabel('|X(k)|');
title ('Magnitude of X(k)');
%
subplot(2,1,2), stem(k, angle(Xk)); xlabel('k'), ylabel('<X(k)');
title ('Phase of X(k)');
```

The DFT values are:

$$X(k) = \{0 \quad 4 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 4\}$$
$$|X(k)| = \{0 \quad 4 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 4\}$$
$$\angle X(k) = \{0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0\}$$

The frequency resolution is F_s/N = 16/8 = 2 Hz. The table below shows that the 8-point DFT contains a component at 2 Hz, corresponding to k = 1. The DFT values are all real numbers symmetrically disposed about k = 4, the center of symmetry.

	k=0							
	$							
$X(k)$ =	{0,	4,	0,	0,	0,	0,	0,	4}
Hz <	0	2	4	6	8	10	12	14
					(F_s/2)			



Example 2.7.2 [Zero-padding] The first 8 sample values of a 2-Hz cosine, *x(t)* = cos 2π2*t*, obtained at a sampling rate of 16 samples/second are given below:

x(n) = {1, 1/sqrt(2), 0, -1/sqrt(2), -1, -1/sqrt(2), 0, 1/sqrt(2)}

Find the 16-point DFT using zero padding.
Solution The zero-padded sequence is given by

x(n) = {1, 1/sqrt(2), 0, -1/sqrt(2), -1, -1/sqrt(2), 0, 1/sqrt(2), 0, 0, 0, 0, 0, 0, 0, 0}

Note that when zero padded the average value of the sequence is no longer zero. The frequency resolution of the DFT is

$$\text{Frequency resolution} = \frac{Sampling\ Frequency}{Number\ of\ point\ s} = \frac{F_s}{N} = \frac{16\ Hz}{16} = 1\ \text{Hz}.$$

The 2 Hz component corresponds to *X(k)* with *k* = 2.

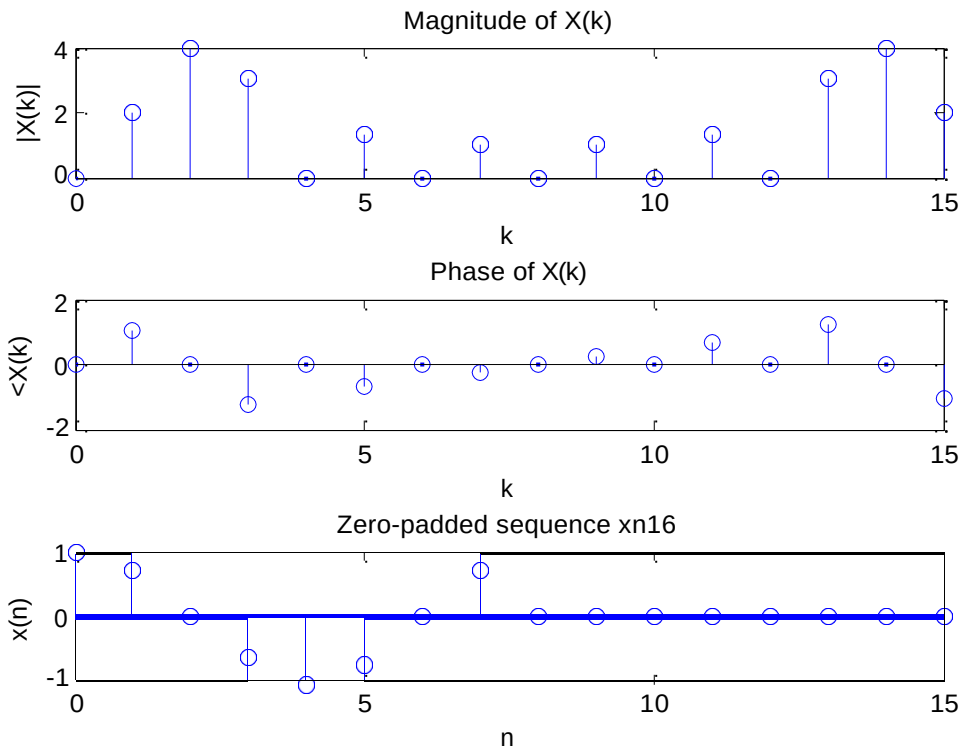
In MATLAB:

```
xn = [1, 1/sqrt(2), 0, -1/sqrt(2), -1, -1/sqrt(2), 0, 1/sqrt(2)],
Xk = fft(xn, 16),          %x(n) is zero-padded to a length of 16
%
% MXk = magnitude of Xk, AXk = phase angle of Xk
MXk = abs(Xk), AXk = angle(Xk),
%
k = 0:15;
subplot(3,1,1), stem(k, abs(Xk)); xlabel('k'), ylabel('|X(k)|');
title ('Magnitude of X(k)');
```

```
%
subplot(3,1,2), stem(k, angle(Xk)); xlabel('k'), ylabel('<X(k)');
title ('Phase of X(k)');
%
xn16 = ifft(Xk),
n = 0:15;
subplot(3,1,3), stem(n, xn16); xlabel('n'), ylabel('x(n)');
title ('Zero-padded sequence xn16');
```

The DFT (reproduced from the MATLAB output) is

X(k) = {0, (1+1.7654i), 4, (1-2.8478i), 0, (1-0.8478i), 0, (1-0.2346i),
0, {1+0.2346i), 0, (1+0.8478i), 0, (1+2.8478i), 4, (1-1.7654i)}



Example 2.7.3 [Without zero-padding] Given all 16 sample values of a 2-Hz cosine, *x(t)* = cos $2\pi 2t$, obtained at a sampling rate of 16 samples/second find its 16-point DFT.
Solution The frequency resolution of the DFT is

$$\text{Frequency resolution} = \frac{Sampling\ Frequency}{Number\ of\ points} = \frac{F_s}{N} = \frac{16\ Hz}{16} = 1\ \text{Hz}.$$

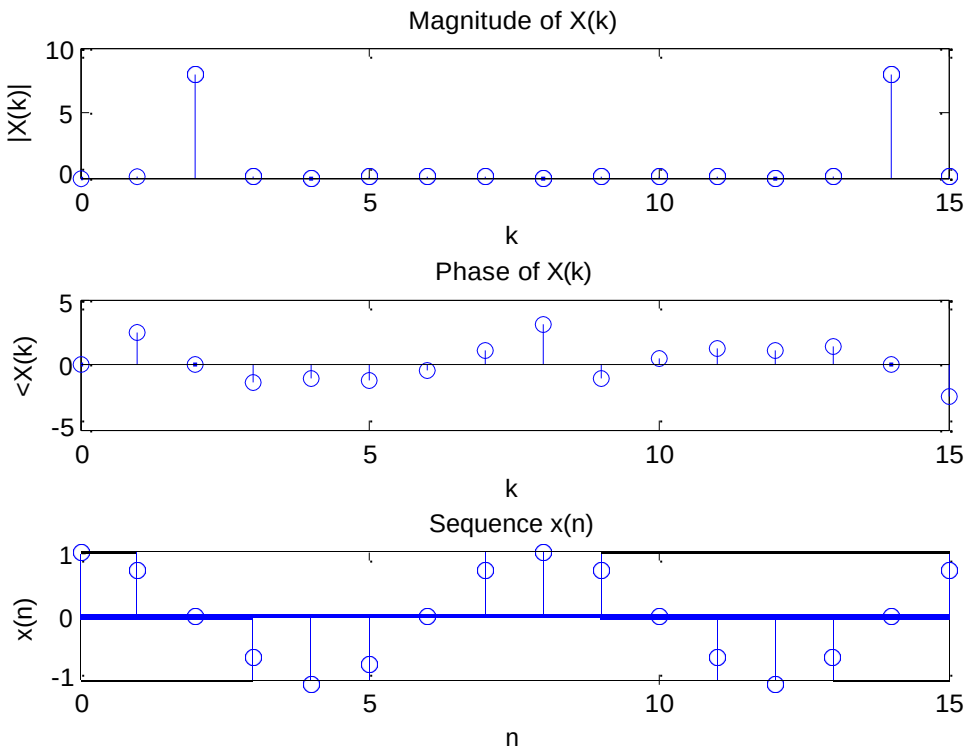
The 2 Hz component corresponds to *X(k)* with *k* = 2.

```
t = 0: 1/16: 15/16; xn = cos (2*pi*2*t), Xk = fft(xn),
%
% MXk = magnitude of Xk, AXk = phase angle of Xk
MXk = abs(Xk), AXk = angle(Xk),
%
k = 0:15;
subplot(3,1,1), stem(k, abs(Xk)); xlabel('k'), ylabel('|X(k)|');
```

```
title ('Magnitude of X(k)');
%
subplot(3,1,2), stem(k, angle(Xk)); xlabel('k'), ylabel('<X(k)');
title ('Phase of X(k)');
%
n = 0:15;
subplot(3,1,3), stem(n, xn); xlabel('n'), ylabel('x(n)');
title ('Sequence x(n)');
```

The DFT (reproduced from the MATLAB output) is

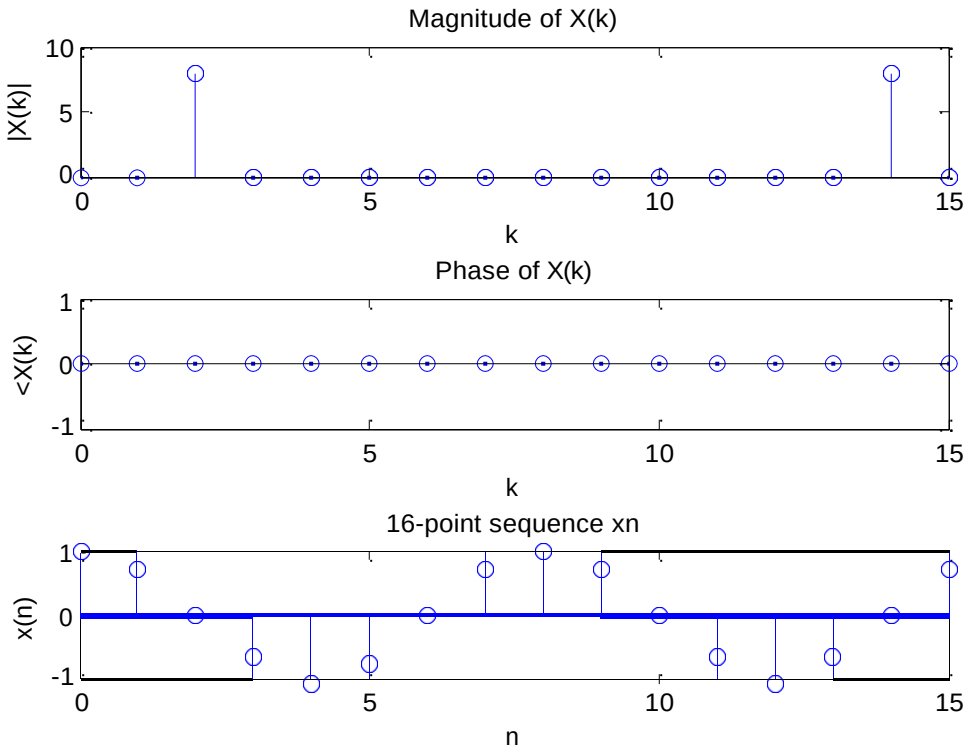
X(k) = {0, (-0+0i), (8-0i), (0-0i), (0-0i), (0-0i), (0-0i), (0+0i),
-0, (0-0i), (0+0i), (0+0i), (0+0i), (0+0i), (8+0i), (-0-0i)}



Specifying the sequence values explicitly as below produces correct phase values (rather than implicitly by *n = 0: 1 : 15; x = cos (pi*n/4), X = fft(x)*)

```
%Listing the sequence values explicitly
xn1 = [1, 1/sqrt(2), 0, -1/sqrt(2), -1, -1/sqrt(2), 0, 1/sqrt(2)],
xn2 = [1, 1/sqrt(2), 0, -1/sqrt(2), -1, -1/sqrt(2), 0, 1/sqrt(2)],
xn = [xn1,xn2], Xk = fft(xn),
%
% MXk = magnitude of Xk, AXk = phase angle of Xk
MXk = abs(Xk), AXk = angle(Xk),
%
k = 0:15
subplot(3,1,1), stem(k, abs(Xk)); xlabel('k'), ylabel('|X(k)|');
title ('Magnitude of X(k)');
%
```

```
subplot(3,1,2), stem(k, angle(Xk)); xlabel('k'), ylabel('<X(k)');
title ('Phase of X(k)');
%
n = 0:15;
subplot(3,1,3), stem(n, xn); xlabel('n'), ylabel('x(n)');
title ('16-point sequence xn');
```



The 16-point DFT is $X(k)$ = {0, 0, 8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0}. The DFT contains a component at 2 Hz. The DFT values are all real numbers symmetrically disposed about k = 8, the center of symmetry.

k <	0		2						8							15
X(k)	{0	0	8	0	0	0	0	0	0	0	0	0	0	0	8	0}
Hz <			2						8							

Example 2.7.4 The signal x(t) = cos $2\pi 2t$ is sampled at 12 Hz (still satisfies the sampling theorem). Calculate (1) the 6-point DFT and (2) the 8-point DFT. Compare the results.
Solution The resulting sequence is $x(n)$ = cos $2\pi 2n(1/12)$ = cos $(\pi n/3)$.
(1) 6-point DFT.

```
n = 0: 1 : 5; x = cos (pi*n/3), X = fft(x)
```

The samples are

$$x(n) = \left\{1, \frac{1}{2}, -\frac{1}{2}, -1, -\frac{1}{2}, \frac{1}{2}\right\}$$

The average value of the sequence (the dc component) is zero. The frequency resolution is F_s/N = 12/6 = 2 Hz. The DFT is

$$X(k) = \{0, 3, 0, 0, 0, 3\}$$

which does show a component at 2 Hz.

(2) 8-point DFT.

n = 0: 1 : 7; x = cos (pi*n/3), X = fft(x)

The samples are

$$x(n) = \left\{1, \frac{1}{2}, -\frac{1}{2}, -1, -\frac{1}{2}, \frac{1}{2}, 1, \frac{1}{2}\right\}$$

Note that the average value of the sequence (the dc component) is *not* zero. The frequency resolution is F_s/N = 12/8 = 1.5 Hz. The DFT is

$$X(k) = \{1.5, (2.5607 + j2.5607), -j1.5, (0.4393 - j0.4393), 0.5, (0.4393 + j0.4393), j1.5, (2.5607 - j2.5607)\}$$

It is not possible to know that there is a component at 2 Hz.

Example 2.7.5 For the signal $x(t) = \cos 2\pi 2t + \cos 2\pi 4t$ choose a sampling frequency of 12Hz (still satisfies sampling theorem). Find the N-point DFT for (a) $N = 6$, (b) $N = 8$, and (c) $N = 12$.

(a) For $N = 6$

$$x(n) = \cos(2\pi 2n/12) + \cos(2\pi 4n/12) = \cos(\pi n/3) + \cos(2\pi n/3)$$

n = 0: 1 : 5; x = cos (pi*n/3) + cos (2*pi*n/3), X = fft(x)

The samples are

$$x(n) = \{2, 0, -1, 0, -1, 0\}$$

The dc component is zero. The frequency resolution is F_s/N = 12/6 = 2 Hz. The DFT is

$$X(k) = \{0, 3, 3, 0, 3, 3\}$$

Both the 2 Hz and the 4 Hz components show up.

k<	0			3		5
X(k)	{0	3	3	0	3	3
Hz<	0	2	4	6	8	10

(b) For $N = 8$

$$x(n) = \cos(2\pi 2n/12) + \cos(2\pi 4n/12) = \cos(\pi n/3) + \cos(2\pi n/3)$$

n = 0: 1 : 7; x = cos (pi*n/3) + cos (2*pi*n/3), X = fft(x)

The samples are

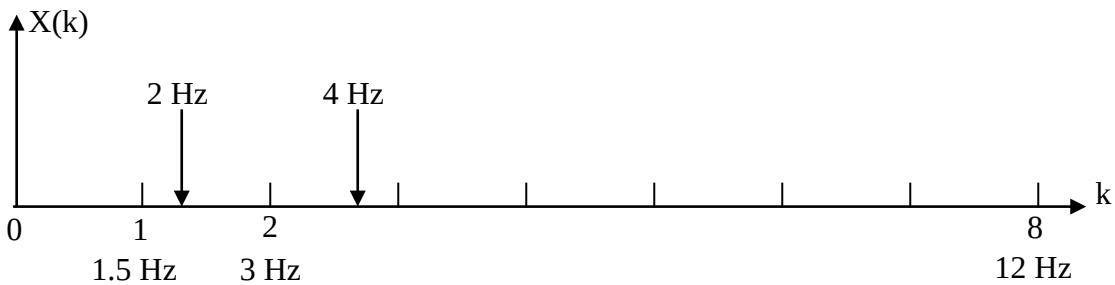
$$x(n) = \{2, 0, -1, 0, -1, 0, 2, 0\}$$

This particular set has a *nonzero* dc component. The frequency resolution is $F_s/N = 12/8 = 1.5$ Hz. The DFT is

$$X(k) = \{2, (3 + j3), 0, (3 - j3), 2, (3 + j3), 0, (3 - j3)\}$$

Neither the 2 Hz nor the 4 Hz component can show up.

k <	0				4			7
X(k)	{2	3+j3	0	3–j3	2	3+j3	0	3–j3
Hz <	0	1.5	3	4.5	6	7.5	9	10.5



Example 2.7.6 For the signal $x(t) = \cos 2\pi 2t + \cos 2\pi 3t + \cos 2\pi 4t$ choose a sampling frequency of 12Hz (still satisfies sampling theorem). Find the N-point DFT for (a) $N = 6$, (b) $N = 8$.

(a) For $N = 6$

$$x(n) = \cos(2\pi 2n/12) + \cos(2\pi 3n/12) + \cos(2\pi 4n/12)$$
$$= \cos(\pi n/3) + \cos(\pi n/2) + \cos(2\pi n/3)$$

n = 0: 1 : 5; x = cos (pi*n/3) + cos (πn/2) + cos (2*pi*n/3), X = fft(x)

The samples are

$$x(n) = \{3, 0, -2, 0, 0, 0\}$$

The dc component is not zero. The frequency resolution is $F_s/N = 12/6 = 2$ Hz. The DFT is

$$X(k) = \{1, (4 + j1.732), (4 - j1.732), 1, (4 + j1.732), (4 - j1.732)\}$$

Both the 2 Hz and the 4 Hz components show up, but the 3 Hz component is missing.

k <	0			3		5
X(k)	{1	4 + j1.732	4 - j1.732	1	4 + j1.732	4 - j1.732
Hz <	0	2	4	6	8	10

Fast Fourier transform

Fast Fourier Transform (FFT) – Radix-2 decimation in time and decimation in frequency FFT Algorithms, Inverse FFT, and FFT for composite N.

Contents:

Introduction

For a finite-duration sequence *x(n)* of length *N*, the DFT sum may be written as

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn},\ k = 0, 1, \ldots, N-1$$

where $W_N = e^{-j2\pi/N}$. There are a total of *N* values of *X(.)* ranging from *X(0)* to *X(N–1)*. The calculation of *X(0)* involves no multiplications at all since every product term involves $W_N^0 = e^{-j0} = 1$. Further, the first term in the sum always involves W_N^0 or $e^{-j0} = 1$ and therefore does not require a multiplication. Each *X(.)* calculation other than *X(0)* thus involves (*N–1*) complex multiplications. And each *X(.)* involves (*N–1*) complex additions. Since there are *N* values of *X(.)* the overall DFT requires $(N-1)^2$ complex multiplications and *N(N–1)* complex additions. For large *N* we may round these off to N^2 complex multiplications and the same number of complex additions.

Each complex multiplication is of the form

$$(A + jB)(C + jD) = (AC - BD) + j(BC + AD)$$

and therefore requires four real multiplications and two real additions. Each complex addition is of the form

$$(A + jB) + (C + jD) = (A + C) + j(B + D)$$

and requires two real additions. Thus the computation of all *N* values of the DFT requires $4N^2$ real multiplications and $4N^2 (= 2N^2 + 2N^2)$ real additions.

Efficient algorithms which reduce the number of multiply-and-add operations are known by the name of **fast Fourier transform** (FFT). The Cooley-Tukey and Sande-Tukey FFT algorithms exploit the following properties of the **twiddle factor** (phase factor), $W_N = e^{-j2\pi/N}$ (the factor $e^{j2\pi/N}$ is called the Nth principal root of 1):

1. Symmetry property $W_N^{k+\frac{N}{2}} = -W_N^k$
2. Periodicity property $W_N^{k+N} = W_N^k$

To illustrate, for the case of *N* = 8, these properties result in the following relations:

$$W_8^0 = -W_8^4 = 1 \qquad W_8^1 = -W_8^5 = \frac{1-j}{\sqrt{2}}$$

$$W_8^2 = -W_8^6 = -j \qquad W_8^3 = -W_8^7 = -\frac{(1+j)}{\sqrt{2}}$$

The use of these properties reduces the number of complex multiplications from N^2 to $\frac{N}{2}\log_2 N$ (actually the number of multiplications is less than this because several of the multiplications by W_N^r are really multiplications by ±1 or ±*j* and don't count); and the number of complex additions are reduced from N^2 to $N \log_2 N$. Thus, with each complex multiplication requiring four real multiplications and two real additions and each complex addition requiring two real additions, the computation of all *N* values of the DFT requires

$$\text{Number of real multiplications} = 4\left(\frac{N}{2}\log_2 N\right) = 2N\log_2 N$$

$$\text{Number of real additions} = 2N\log_2 N + 2\left(\frac{N}{2}\log_2 N\right) = 3N\log_2 N$$

We can get a rough comparison of the speed advantage of an FFT over a DFT by computing the number of multiplications for each since these are usually more time consuming than additions. For instance, for $N = 8$ the DFT, using the above formula, would need $8^2 = 64$ complex multiplications, but the radix-2 FFT requires only 12 ($= \frac{8}{2}\log_2 8 = 4 \times 3$).

Number of multiplications: DFT vs. FFT

No. of points	**No. of complex multiplications**		**No. of real multiplications**	
N	**DFT**	**FFT**	**DFT**	**FFT**
32	1024	80	4096	320
128	16384	448	65536	1792
1024	1048576	5120	4194304	20480

We consider first the case where the length N of the sequence is an integral power of 2, that is, $N = 2^v$ where v is an integer. These are called **radix-2 algorithms** of which the **decimation-in-time (DIT)** version is also known as the **Cooley-Tukey algorithm** and the **decimation-in-frequency (DIF)** version is also known as the **Sande-Tukey algorithm**. We show first how the algorithms work; their derivation is given later.

For a radix of ($r = 2$), the **elementary computation** (*EC*) known as the **butterfly** consists of a single complex multiplication and two complex additions.

If the number of points, N, can be expressed as $N = r^m$, and if the computation algorithm is carried out by means of a succession of r-point transforms, the resultant FFT is called a **radix-*r* algorithm**. In a radix-r FFT, an elementary computation consists of an r-point DFT followed by the multiplication of the r results by the appropriate twiddle factor. The number of *EC*s required is

$$C_r = \frac{N}{r}\log_r N$$

which decreases as r increases. Of course, the complexity of an *EC* increases with increasing r. For $r = 4$, the *EC* requires three complex multiplications and several complex additions.

Suppose that we desire an N-point DFT where N is a composite number that can be factored into the product of integers

$$N = N_1 N_2 \ldots N_m$$

If, for instance, $N = 64$ and $m = 3$, we might factor N into the product $64 = 4 \times 4 \times 4$, and the 64-point transform can be viewed as a three-dimensional $4 \times 4 \times 4$ transform.

If N is a prime number so that factorization of N is not possible, the original signal can be *zero-padded* and the resulting new composite number of points can be factored.

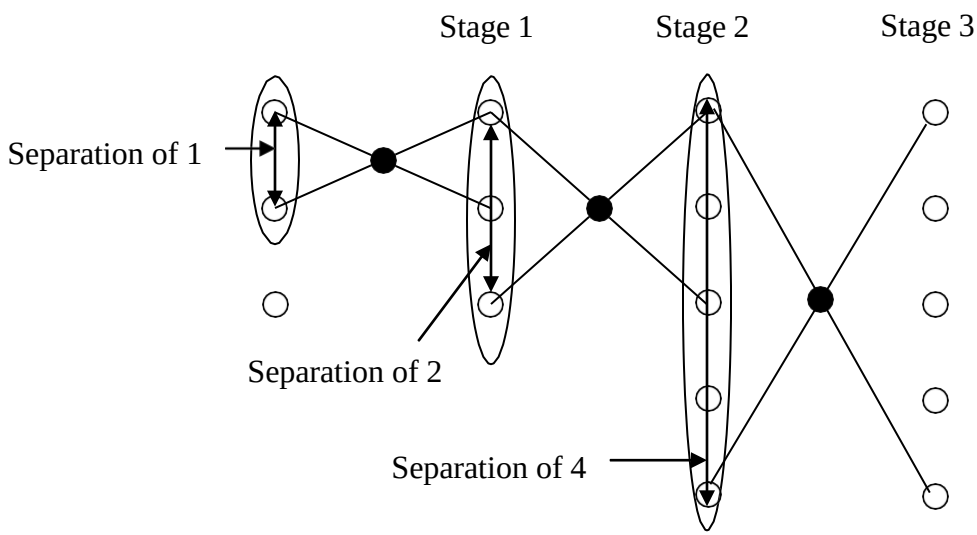
Radix-2 decimation-in-time FFT (Cooley-Tukey)

Procedure and important points

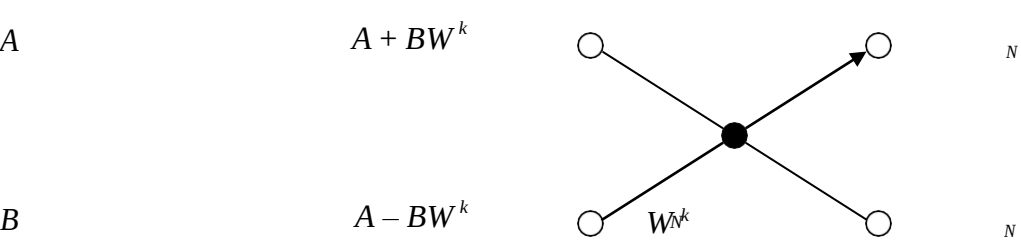
1. The number of input samples is $N = 2^v$ where v is an integer.
2. The input sequence is shuffled through bit-reversal. The index n of the sequence $x(n)$ is expressed in binary and then reversed.
3. The number of stages in the flow graph is given by $v = \log_2 N$.
4. Each stage consists of $N/2$ butterflies.
5. Inputs/outputs for each butterfly are separated as follows:

   Separation = 2^{m-1} samples where m = stage index, stages being numbered from left to right (that is, $m = 1$ for stage 1, $m = 2$ for stage 2 etc.).

   This amounts to separation increasing from left to right in the order 1, 2, 4, …, $N/2$.



6. The number of complex additions = $N \log_2 N$ and the number of complex multiplications is $\frac{N}{2} \log_2 N$.
7. The elementary computation block in the flow graph, called the butterfly, is shown here. This is an **in-place calculation** in that the outputs $(A + BW_N^k)$ and $(A - BW_N^k)$ can be computed and stored in the same locations as A and B.



Example 2.2.1 Radix-2, 8-point, decimation-in-time FFT for the sequence

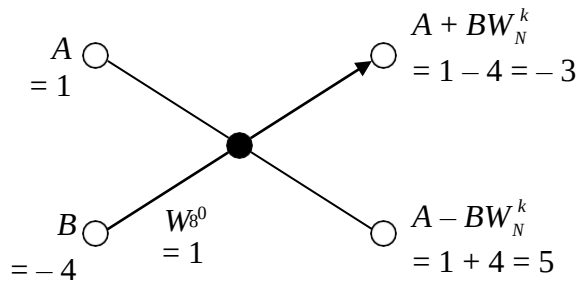
$n <$	0	1	2	3	4	5	6	7
$x(n) =$	{1,	2	3	4	–4	–3	–2	–1}

Solution The twiddle factors are

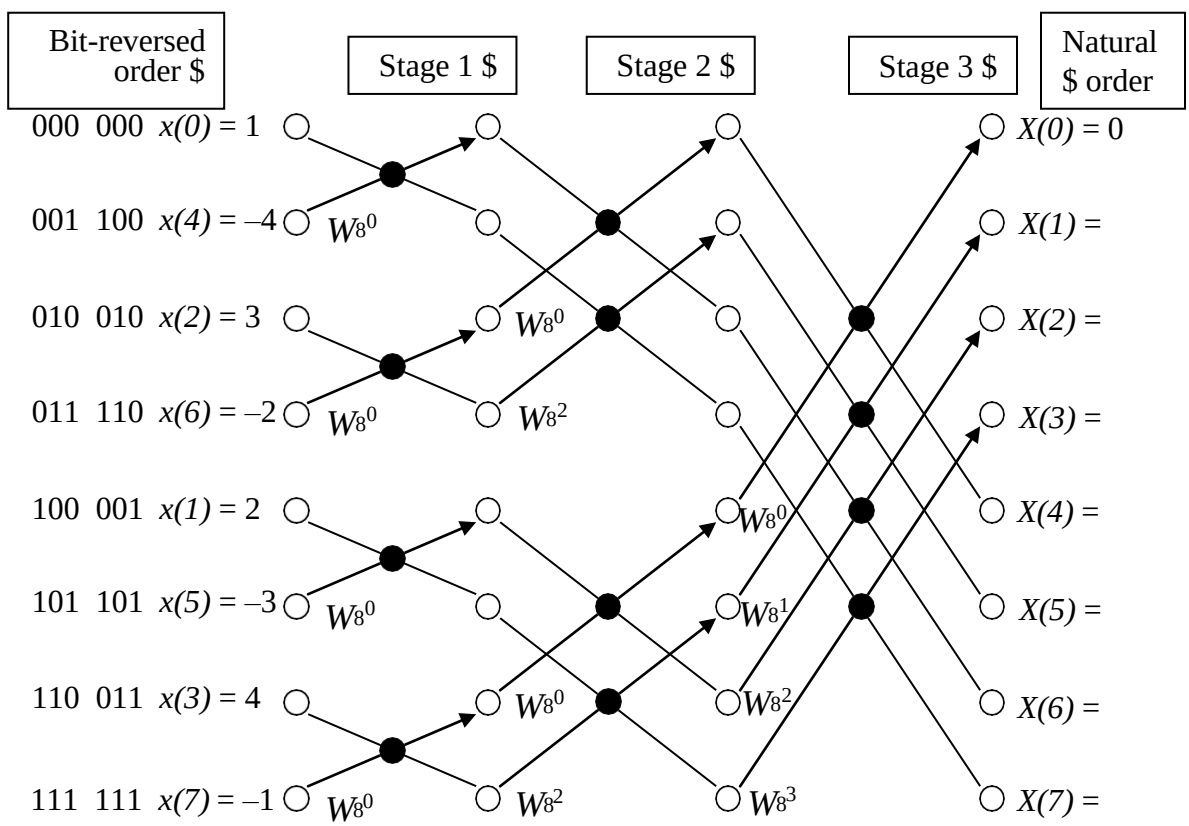
$$W_8^0 = 1 \qquad W_8^1 = e^{-j2\pi/8} = e^{-j\pi/4} = \frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}}$$

$$W_8^2 = \left(e^{-j2\pi/8}\right)^2 = e^{-j\pi/2} = -j \qquad W_8^3 = \left(e^{-j2\pi/8}\right)^3 = e^{-j3\pi/4} = -\frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}}$$

One of the elementary computations is shown below:



The signal flow graph follows:



8-point FFT using DIT			
Results of the first stage			
Input	Stage 1	Stage 2	Stage 3 (Output)
1	1 + (–4) . 1 = –3		
– 4	1 – (–4) . 1 = 5		
3	3 + (–2) . 1 = 1		
– 2	3 – (–2) . 1 = 5		
2	2 + (–3) . 1 = –1		
– 3	2 – (–3) . 1 = 5		
4	4 + (–1) . 1 = 3		
– 1	4 – (–1) . 1 = 5		

Results of the second stage			
Input	Stage 1	Stage 2	Stage 3 (Output)
1	–3	–3 + 1 . 1 = –2	
– 4	5	5 + 5 . (–j) = 5 $2e^{-j\pi/4}$	
3	1	–3 – 1 . 1 = –4	
– 2	5	5 – 5 . (–j) = 5 $2e^{j\pi/4}$	
2	–1	–1 + 3 . 1 = 2	
– 3	5	5 + 5 . (–j) = 5 $2e^{-j\pi/4}$	
4	3	–1 – 3 . 1 = –4	
– 1	5	5 – 5 . (–j) = 5 $2e^{j\pi/4}$	

Results of the third stage			
Input	Stage 1	Stage 2	Stage 3 (Output)
1	–3	–2	–2 + 2 . 1 = 0
– 4	5	$5\ 2e^{-j\pi/4}$	$5\ 2e^{-j\pi/4} + 5\ 2e^{-j\pi/4} . e^{-j\pi/4} = 5 - j12.07$
3	1	–4	$-4 + (-4) . (-j) = -4 + j4 = 4\ 2e^{j3\pi/4}$
– 2	5	$e^{j\pi/4}$	$5\ 2e^{j\pi/4} + 5\ 2e^{j\pi/4} . e^{-j3\pi/4} = 5 - j\ 2.07$
2	–1	2	–2 – 2 . 1 = –4
– 3	5	$5\ 2e^{-j\pi/4}$	$5\ 2e^{-j\pi/4} - 5\ 2e^{-j\pi/4} . e^{-j\pi/4} = 5 + j2.07$
4	3	–4	$-4 - (-4) . (-j) = -4 - j4 = 4\ 2e^{-j3\pi/4}$
– 1	5	$e^{j\pi/4}$	$5\ 2e^{j\pi/4} - 5\ 2e^{j\pi/4} . e^{-j3\pi/4} = 5 + j12.07$

The DFT is $X(k)$ = {0, (5 – j12.07), (–4 + j4), (5 – j2.07), –4, (5 + j2.07), (–4 – j4), (5 + j12.07)}

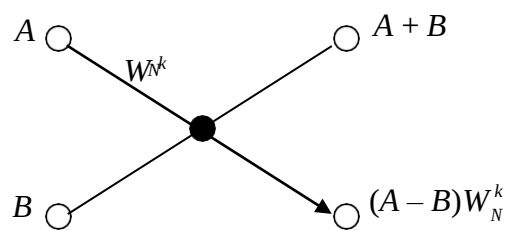
The MATLAB program:

```
x = [1, 2, 3, 4, -4, -3, -2, -1], X = fft(x)
```

Radix-2 decimation-in-frequency FFT (Sande-Tukey)

Procedure and important points

1. The number of input samples is $N = 2^v$ where v is an integer.
2. The input sequence is in natural order; the output is in bit-reversed order.
3. The number of stages in the flow graph is given by $v = \log_2 N$.
4. Each stage consists of $N/2$ butterflies.
5. Inputs/outputs for each butterfly are separated in the reverse order from that of the DIT. The separation decreases *from left to right* in the order $N/2, \ldots, 4, 2, 1$.
6. The number of complex additions = $N \log_2 N$ and the number of complex multiplications is $\frac{N}{2}\log_2 N$.
7. The basic computation block in the flow graph of the DIF FFT is the butterfly shown here. This is an **in-place calculation** in that the two outputs $(A + B)$ and $(A - B)\, W_N^k$ can be computed and stored in the same locations as A and B.



Example 2.3.1 Radix-2, 8-point, decimation-in-frequency FFT for the sequence

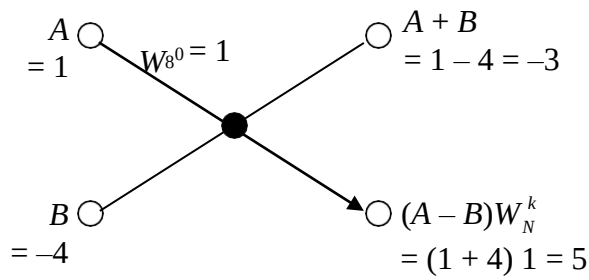
$n <$	0	1	2	3	4	5	6	7
$x(n) =$	{1,	2	3	4	–4	–3	–2	–1}

Solution The twiddle factors are the same as in the DIT FFT done earlier (both being 8-point DFTs):

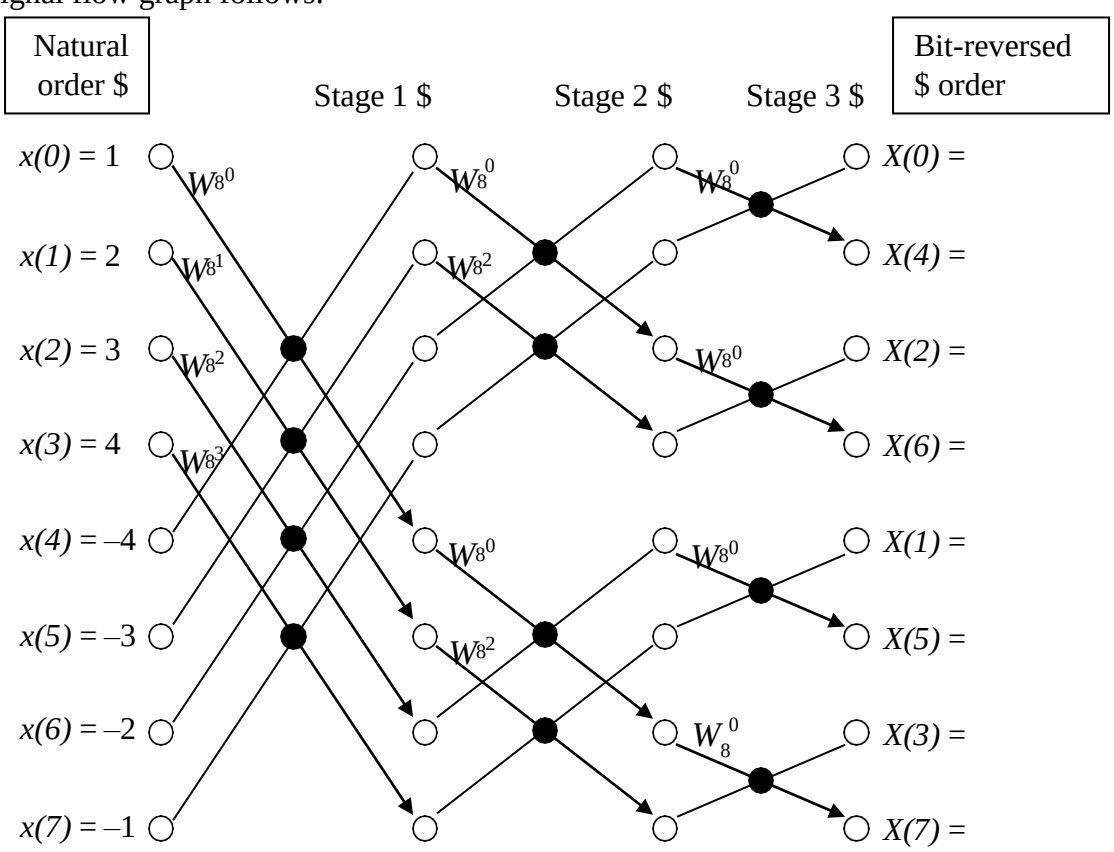
$$W_8^0 = 1 \qquad W_8^1 = e^{-j2\pi/8} = \frac{1}{\sqrt{2}}j \quad \frac{1}{\sqrt{2}}$$

$$W_8^2 = \left(e^{-j2\pi/8}\right)^2 = e^{-j\pi/2} = -j \qquad W_8^3 = \left(e^{-j2\pi/8}\right)^3 = e^{-j3\pi/2} = -\frac{1}{\sqrt{2}}j \quad \frac{1}{\sqrt{2}}$$

One of the elementary computations is shown below:



The signal flow graph follows:



8-point FFT using DIF			
Results of the first stage			
Input	Stage 1	Stage 2	Stage 3 (Output)
1	$1 + (-4) = -3$		
2	$2 + (-3) = -1$		
3	$3 + (-2) = 1$		
4	$4 + (-1) = 3$		
– 4	$(1 - (-4))\ 1 = 5$		
– 3	$(2 - (-3))\ e^{-j\pi/4} = 5\ e^{-j\pi/4}$		
– 2	$(3 - (-2))\ (-j) = -j5$		
– 1	$(4 - (-1))\ e^{-j3\pi/4} = 5\ e^{-j3\pi/4}$		

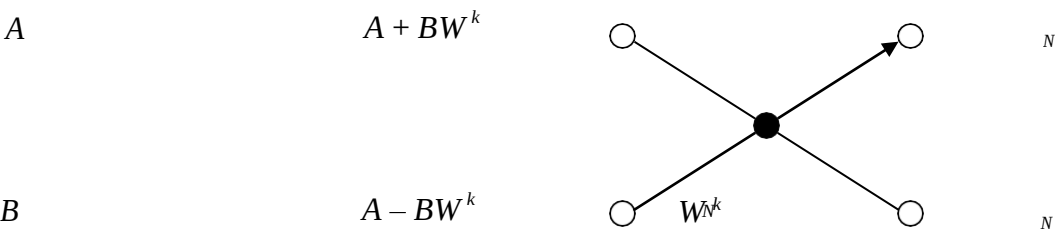
Results of the second stage			
Input	Stage 1	Stage 2	Stage 3 (Output)
1	-3	$-3 + 1 = -2$	
2	-1	$-1 + 3 = 2$	
3	1	$(-3\ -1)\ 1 = -4$	
4	3	$(-1\ -3)\ (-j) = j4$	
– 4	5	$5 + (-j5) = 5\sqrt{2}e^{-j\pi/4}$	
– 3	$5\ e^{-j\pi/4}$	$5\ e^{-j\pi/4} + 5e^{-j3\pi/4} = -j5\quad 2$	
– 2	$-j5$	$(5 - (-j5))\ 1 = 5\sqrt{2}e^{j\pi/4}$	
– 1	$5\ e^{-j3\pi/4}$	$(5\ e^{-j\pi/4} - 5e^{-j3\pi/4})\ (-j) = -j5\quad 2$	

Results of the third stage			
Input	Stage 1	Stage 2	Stage 3 (Output)
1	-3	-2	$-2 + 2 = 0$
2	-1	2	$(-2\ -2)\ 1 = -4$
3	1	-4	$-4 + j4 = -4 + j4 = 4\quad 2e^{j3\pi/4}$
4	3	j4	$(-4 - j4)\ 1 = -4 - j4 = 4\quad 2e^{-j3\pi/4}$
– 4	5	$5\sqrt{2}e^{-j\pi/4}$	$5\sqrt{2}e^{-j\pi/4} + (-j5\sqrt{2}) = 5 - j12.07$
– 3	$5\ e^{-j\pi/4}$	$-j5\sqrt{2}$	$(5\sqrt{2}e^{-j\pi/4} - (-j5\sqrt{2}))\ 1 = 5 + j2.07$
– 2	$-j5$	$5\sqrt{2}e^{j\pi/4}$	$5\quad 2e^{j\pi/4} + (-j5\quad 2) = 5 - j2.07$
– 1	$5\ e^{-j3\pi/4}$	$-j5\sqrt{2}$	$(5\sqrt{2}e^{j\pi/4} - (-j5\sqrt{2}))\ 1 = 5 + j12.07$

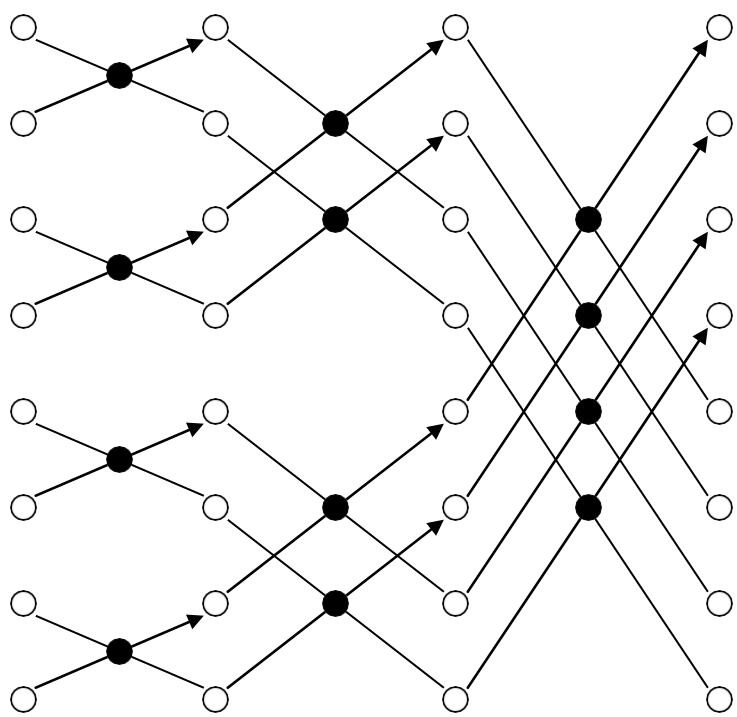
The DFT is $X(k)$ = {0, (5 – j12.07), (–4 + j4), (5 – j2.07), –4, (5 + j2.07), (–4 – j4), (5 + j12.07)}

The MATLAB progarm is the same as shown in Example 1.

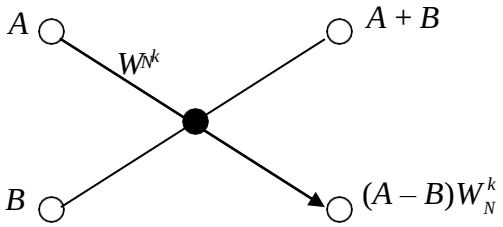
(DIT Template)
The elementary computation (Butterfly):



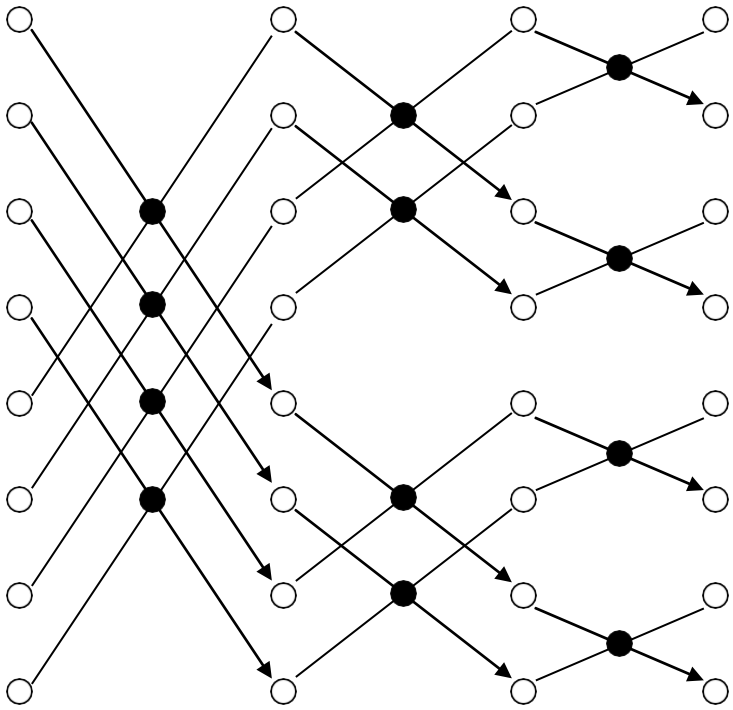
The signal flow graph:



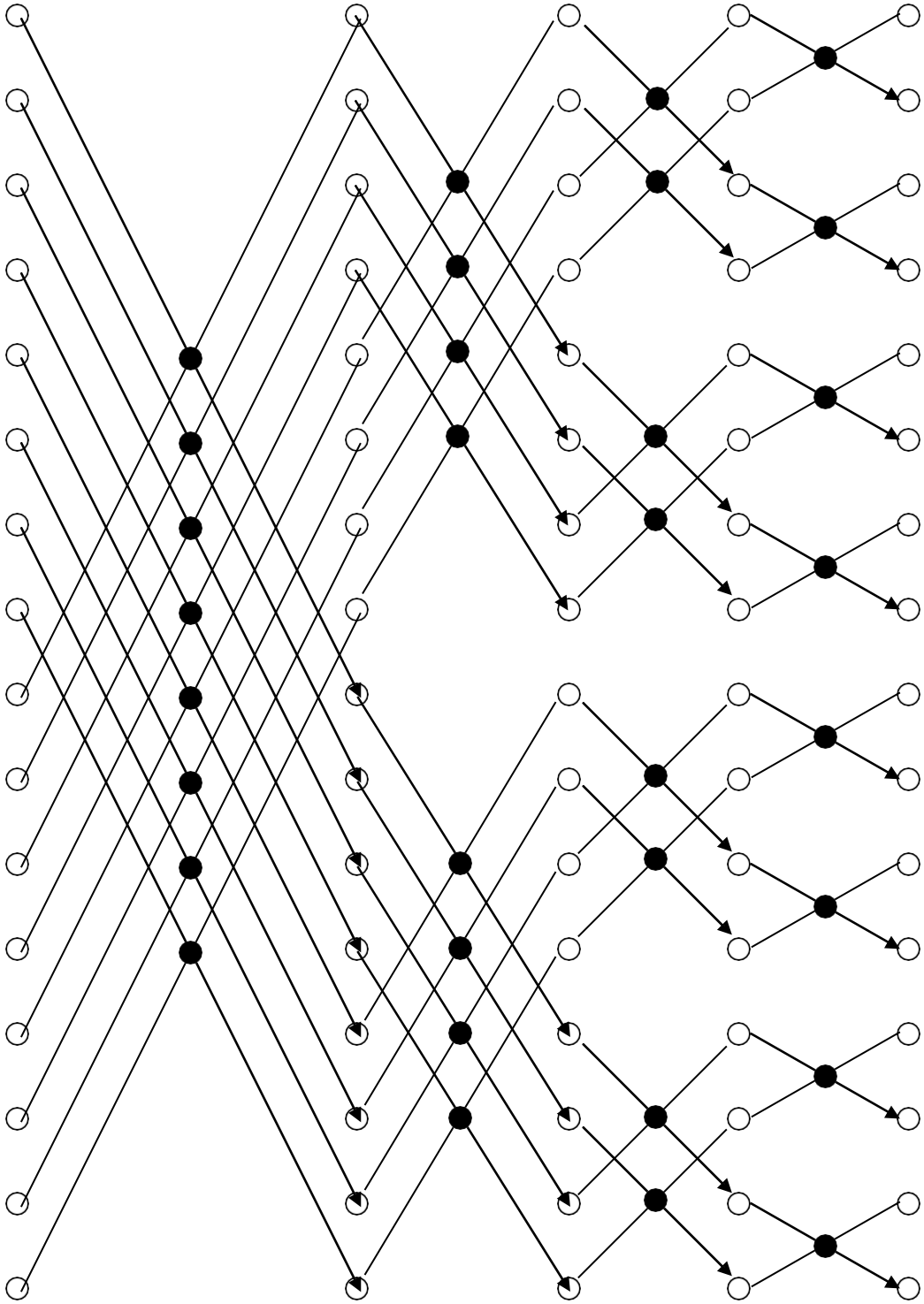
(DIF Template)
The elementary computation (Butterfly):



The signal flow graph:



16-point DIF FFT



Inverse DFT using the FFT algorithm

The inverse DFT of an N-point sequence $\{X(k), k = 1, 2, \ldots, (N-1)\}$ is defined as

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k) W_N^{-kn}, \qquad n = 0, 1, \ldots, N-1$$

where $W_N = e^{-j2\pi/N}$. Take the complex conjugate of $x(n)$ and multiply by N to get

$$N\,x^*(n) = \sum_{k=0}^{N-1} X^*(k) W_N^{kn}$$

The right hand side of the above equation is simply the DFT of the sequence $X^*(k)$ and can be computed by using any FFT algorithm. The desired output sequence is then found by taking the conjugate of the result and dividing by N

$$x(n) = \frac{1}{N}\left(\sum_{k=0}^{N-1} X^*(k) W_N^{kn}\right)^*$$

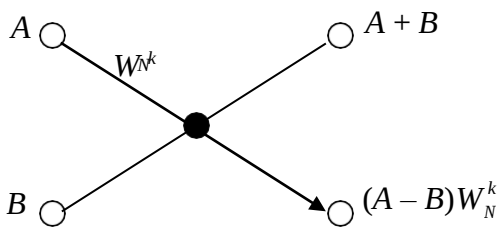
Example 2.4.1 Given the DFT sequence *X(k)* = {0, (–1–j), j, (2+j), 0, (2–j), –j, (–1+j)} obtain the IDFT *x(n)* using the DIF FFT algorithm.

Solution This is an 8-point IDFT. The 8-point twiddle factors are, as calculated earlier,

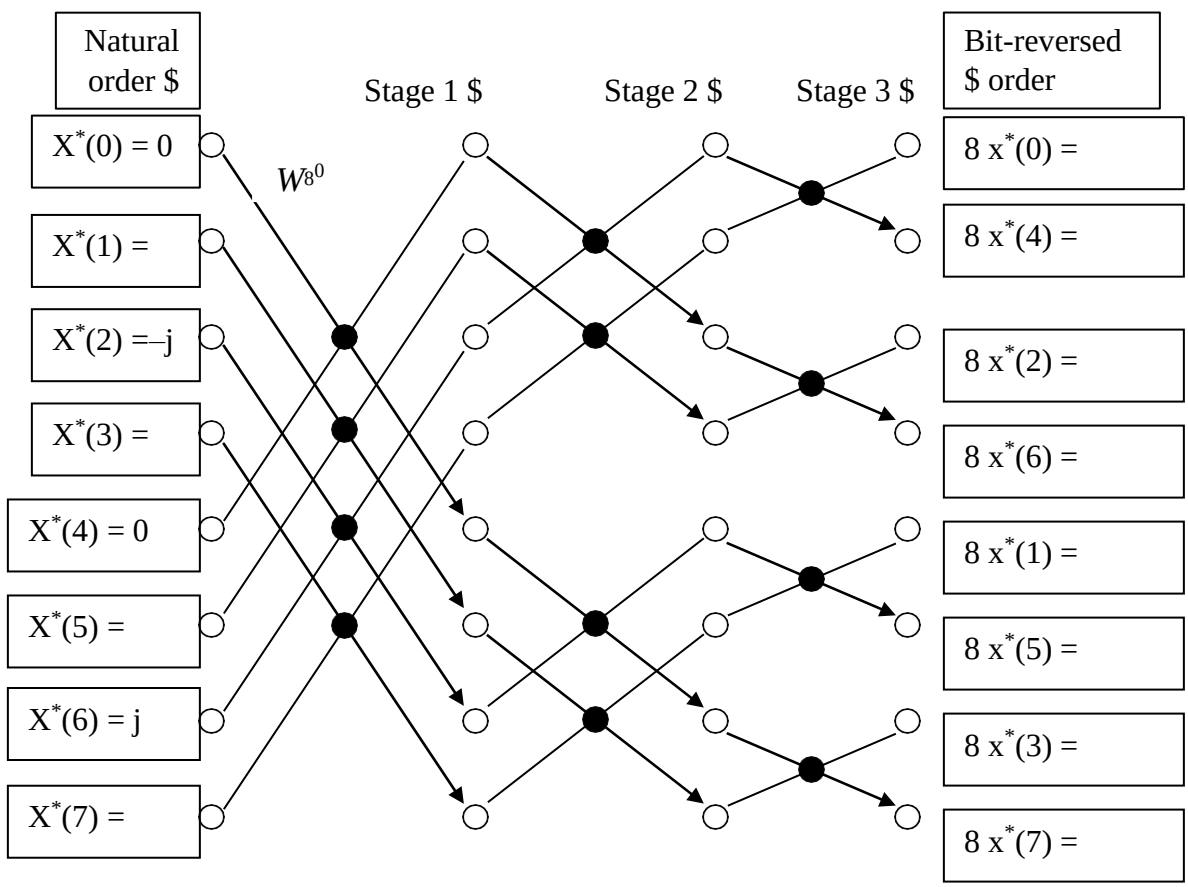
$$W_8^0 = 1 \qquad W_8^1 = e^{-j2\pi/8} = \frac{1}{\sqrt{2}}j \quad \frac{1}{\sqrt{2}}$$

$$W_8^2 = \left(e^{-j2\pi/8}\right)^2 = e^{-j\pi/2} = -j \qquad W_8^3 = \left(e^{-j2\pi/8}\right)^3 = e^{-j3\pi/2} = -\frac{1}{\sqrt{2}}j \quad \frac{1}{\sqrt{2}}$$

The elementary computation (Butterfly) is shown below:



The signal flow graph follows:



8-point IDFT using DIF FFT			
Results of the first stage			
Input $X^*(k)$	Stage 1	Stage 2	Stage 3 (Output)
0	0 + 0 = 0		
–1+j	–1+j + 2+j = 1+j2		
–j	–j + j = 0		
2–j	2–j + (–1–j) = 1–j2		
0	(0 – 0) 1 = 0		
2+j	$(-1+j-(2+j))\, e^{-j\pi/4} = -3\, e^{-j\pi/4}$		
j	(–j – j) (–j) = –2		
–1–j	$(2-j-(-1-j))\, e^{-j3\pi/4} = 3\, e^{-j3\pi/4}$		

Results of the second stage			
Input	Stage 1	Stage 2	Stage 3 (Output)
0	0	0 + 0 = 0	
–1+j	1+j2	1+j2 + 1–j2 = 2	
–j	0	(0 – 0) 1 = 0	
2–j	1–j2	(1+j2 – (1–j2)) (–j) = 4	
0	0	0 + (–2) = –2	
2+j	$-3\, e^{-j\pi/4}$	$-3\, e^{-j\pi/4} + 3\, e^{-j3\pi/4} = -3 \quad 2$	
j	–2	(0 – (–2)) 1 = 2	
–1–j	$3\, e^{-j3\pi/4}$	$(-3\, e^{-j\pi/4} - 3 e^{-j3\pi/4})\,(-j) = 3 \quad 2$	

Results of the third stage				
Input	Stage 1	Stage 2	Stage 3 (Output)	
0	0	0	0 + 2 = 2	← $8\, x^*(0)$
–1+j	1+j2	2	(0 – 2) 1 = – 2	← $8\, x^*(4)$
–j	0	0	0 + 4 = 4	← $8\, x^*(2)$
2–j	1–j2	4	(0 – 4) 1 = – 4	← $8\, x^*(6)$
0	0	–2	$-2 + (-3\sqrt{2}) = -6.24$	← $8\, x^*(1)$
2+j	$-3\, e^{-j\pi/4}$	$-3\, 2$	$(-2-(-3\sqrt{2}))\, 1 = 2.24$	← $8\, x^*(5)$
j	–2	2	2 + 3 2 = 6.24	← $8\, x^*(3)$
–1–j	$3\, e^{-j3\pi/4}$	3 2	(2 – 3 2) 1 = – 2.24	← $8\, x^*(7)$

The output at stage 3 gives us the values $\{8\, x^*(n)\}$ in bit-reversed order:

$$\{8x^*(n)\}_{bit\ rev\ order} = \{2, -2, 4, -4, -6.24, 2.24, 6.24, -2.24\}$$

The IDFT is given by arranging the data in normal order, taking the complex conjugate of the sequence and dividing by 8:

$$\{8\, x^* \quad _{normalorder} = \{2, -6.24, 4, 6.24, -2, 2.24, -4, -2.24\}$$

$$x(n) = \left\{ \frac{1}{4}, \frac{-6.24}{8}, \frac{1}{2}, \frac{6.24}{8}, -\frac{1}{4}, \frac{2.24}{8}, -\frac{1}{2}, \frac{-2.24}{8} \right\}$$

$$x(n) = \{0.25, -0.78, 0.5, 0.78, -0.25, 0.28, -0.5, -0.28\}$$

Note Because of the conjugate symmetry of $\{X(k)\}$, we should expect the sequence $\{x(n)\}$ to be *real-valued*.

The MATLAB program:

```
X = [0, (-1-j), j, (2+j), 0, (2-j), -j, (-1+j)], x = ifft(X)
```

Example 2.4.2 Given the DFT sequence $X(k) = \{0, (1-j), j, (2+j), 0, (2-j), (-1+j), -j\}$ obtain the IDFT $x(n)$ using the DIF FFT algorithm.

Solution There is no conjugate symmetry in $\{X(k)\}$. Using MATLAB

```
X = [0, 1-1j, 1j, 2+1j, 0, 2-1j, -1+1j, -1j]
x = ifft(X)
```

The IDFT is

$$x(n) = \{0.5, (-0.44 + 0.037i), (0.375 - 0.125i), (0.088 + 0.14i), (-0.75 + 0.5i), (0.44 + 0.21i), (-0.125 - 0.375i), (-0.088 - 0.39i)\}$$

2.7 FFT with general radix

As mentioned in the introduction, if the number of points, N, can be expressed as $N = r^m$, and if the computation algorithm is carried out by means of a succession of r-point transforms, the resultant FFT is called a **radix-*r* algorithm**. In a radix-r FFT, an **elementary computation** (*EC*) consists of an r-point DFT followed by the multiplication of the r results by the appropriate twiddle factor. The number of *EC*s required is

$$C_r = \frac{N}{r}\log_r N$$

which decreases as r increases.

Of course, the complexity of an *EC* increases with increasing r. For $r = 2$, the *EC* (the butterfly) consists of a single complex multiplication and two complex additions; for $r = 4$, the *EC* requires three complex multiplications and several complex additions.

Suppose that we desire an N-point DFT where N is a composite number that can be factored into the product of integers

$$N = N_1 N_2 \ldots N_m$$

If, for instance, $N = 64$ and $m = 3$, we might factor N into the product 64 = 4 x 4 x 4, and the 64-point transform can be viewed as a three-dimensional 4 x 4 x 4 transform.

If N is a prime number so that factorization of N is not possible, the original signal can be *zero-padded* and the resulting new composite number of points can be factored.

We illustrate in the table below the situation for $N = 64$. Since $64 = 2^6$, we can have a radix-2 FFT; alternatively, since $64 = 4^3$, we can also have a radix-4 FFT.

$N = 64 = 2^6 = 4^3 = 8^2$			
	Radix-2	Radix-4	Radix-8
No. of stages	$\log_2 64 = 6$	$\log_4 64 = 3$	$\log_8 64 = 2$
No. of *EC*s per stage	64/2 = 32	64/4 = 16	64/8 = 8

UNIT -2

IIR Digital Filters

Analog filter approximations – Butterworth and Chebyshev, Design of IIR digital filters from analog filters, Bilinear transformation method, Step and Impulse invariance techniques, Spectral transformations, Design examples: Analog-Digital transformations

Contents:

Introduction

Nomenclature With $a_0 = 1$ in the linear constant coefficient difference equation,

$$a_0\, y(n) + a_1\, y(n-1) + \ldots + a_N\, y(n-N)$$
$$= b_0\, x(n) + b_1\, x(n-1) + \ldots + b_M\, x(n-M), \qquad a_0 \neq 0$$

we have,

$$H(z) = \frac{\sum_{i=0}^{M} b_i z^{-i}}{1 + \sum_{i=1}^{N} a_i z^{-i}}$$

This represents an IIR filter if at least one of a_1 through a_N is nonzero, and all the roots of the denominator are not canceled exactly by the roots of the numerator. In general, there are M finite zeros and N finite poles. There is no restriction that M should be less than or greater than or equal to N. In most cases, especially digital filters derived from analog designs, $M \leq N$. Systems of this type are called N^{th} order systems. This is the case with IIR filter design in this Unit.

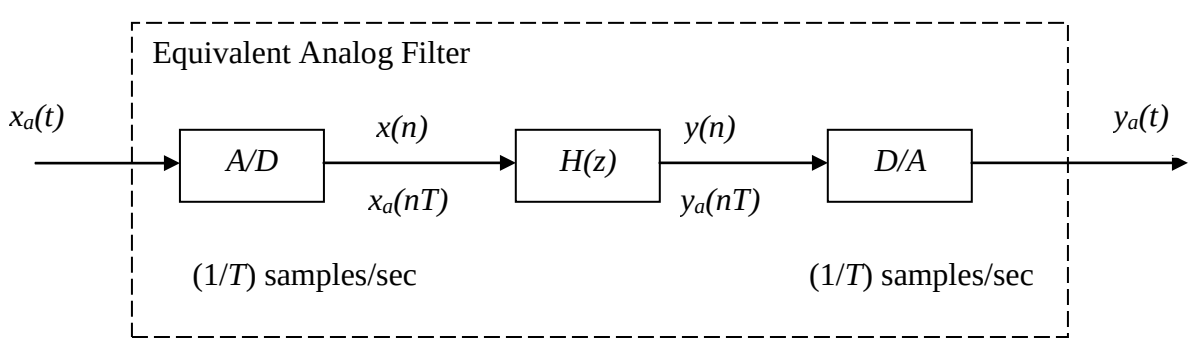
When $M > N$, the order of the system is no longer unambiguous. In this case, $H(z)$ may be taken to be an Nth order system in cascade with an FIR filter of order $(M - N)$.

When $N = 0$, as in the case of an FIR filter, according to our convention the *order* is 0. However, it is more meaningful in such a case to focus on M and call the filter an FIR filter of M stages or $(M+1)$ coefficients.

Example The system $H(z) = (1 - z^{-8})/(1 - z^{-1})$ is not an IIR filter. Why (verify)?

IIR filter design An analog filter specified by the Laplace transfer function, $H_a(s)$, may be designed to either frequency domain or time domain specifications. Similarly, a digital filter, $H(z)$, may be required to have either (1) a given frequency response, or (2) a specific time domain response to an impulse, step, or ramp, etc.

Analog design using digital filters, $\omega_i = \Omega_i T$ Another possibility is that a digital filter may be required to simulate a continuous-time (analog) system. To simulate an analog filter the discrete-time filter is used in the *A/D – H(z) – D/A* structure shown below. The A/D converter can be thought of roughly as a sampler and coder, while the D/A converter, in many cases, represents a decoder and holder followed by a low pass filter (*smoothing filter*). The A/D converter may be preceded by a low pass filter, also called an *anti-aliasing filter* or *pre-filter*.



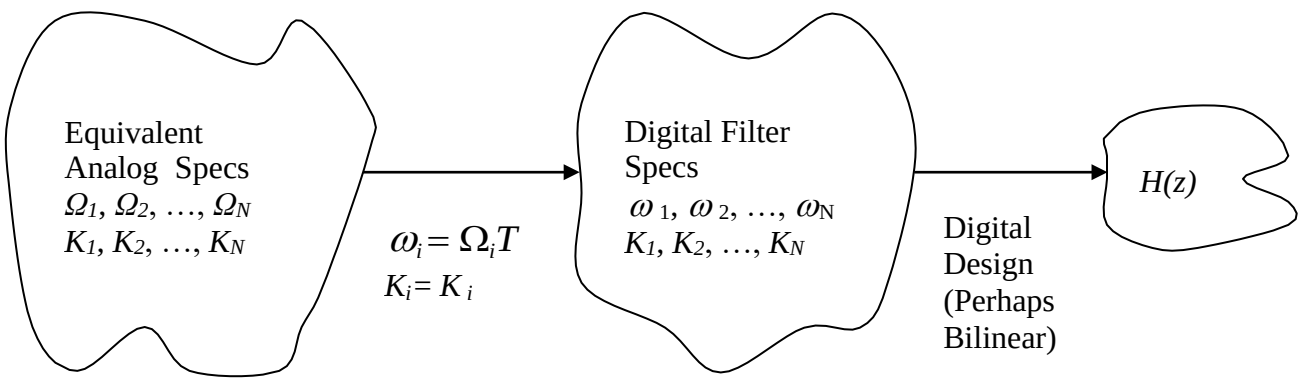
We will usually be given a set of analog requirements with critical frequencies $\Omega_1, \Omega_2, \ldots, \Omega_N$ in radians/sec., and the corresponding frequency response magnitudes $K_1, K_2, \ldots, K_N$ in dB. The sampling rate $1/T$ of the *A/D* converter will be specified or can be determined from the input signals under consideration.

The general approach for the design is to first convert the analog requirements to digital requirements and then design the digital filter using the bilinear transformation. The conversion of the analog specifications to digital specifications is through the formula $\omega_i = \Omega_i T$. To show that this is true, suppose that the input to the equivalent analog filter is $x_a(t) = \sin \Omega_i t$. The output of the *A/D* converter with sampling rate $1/T$ becomes

$$x(n) = x_a(nT) = \sin\Omega_i nT = \sin(\Omega_i T)n = \sin \omega_i n$$

Thus, the magnitude of the discrete-time sinusoidal signal is the same as the continuous time sinusoid, while the digital frequency ω_i is given in terms of the analog frequency Ω_i by $\omega_i = \Omega_i T$.

Thus, the specifications for the digital filter become $\omega_1, \omega_2, \ldots, \omega_N$ with the corresponding frequency response magnitudes $K_1, K_2, \ldots, K_N$. The digital frequency, ω, is in units of radians. The procedure is conceptually shown in figure below.



There are various techniques for designing $H(z)$:

1. Numerical approximation (numerical solution) to the derivative operation or the integration operation (this latter results in the **bilinear transformation** aka bilinear *z*-transformation – BZT).
2. **Time domain invariance**, e.g., impulse invariance and step-invariance methods.

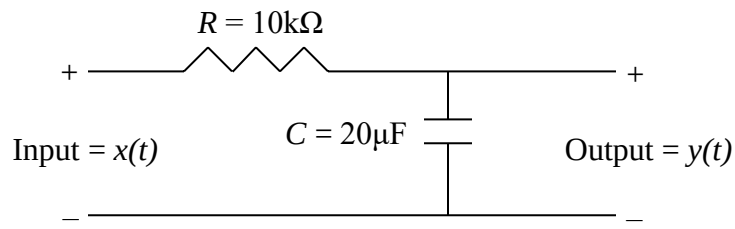
The focus is on the low pass analog filter because once designed it can be transformed into an equivalent quality high pass, band pass or band stop filter by frequency transformation. The Butterworth, Chebyshev and elliptic filters are used as a starting point in designing digital filters. We approximate the magnitude part of the frequency response, not the phase. Butterworth and Chebyshev filters are actually special cases of the more difficult elliptic filter.

Because a constant divided by an Nth order polynomial in Ω falls off as Ω^N it will be an approximate low pass function as Ω varies from 0 to ∞. Therefore, an all-pole analog filter $H(s) = 1/D(s)$ is a good and simple choice for a low pass filter form and is used in both the Butterworth and the type I Chebyshev filters. Moreover, for a given denominator order, having the numerator constant (order zero) gives (for a given number of filter coefficients) the maximum attenuation as $\Omega \rightarrow \infty$.

The normalized analog, low pass, Butterworth filter

As a lead-in to digital filter design we look at a simple analog low pass filter – an RC filter, and its frequency response.

Example 3.2.1 Find the transfer function, $H_a(s)$, impulse response, $h_a(t)$, and frequency response, $H_a(j\Omega)$, of the following system.



Solution This is a voltage divider. The transfer function is given by

$$H_a(s) = \frac{Y(s)}{X(s)} = \frac{(1/sC)}{R+(1/sC)} = \frac{(1/RC)}{s+(1/RC)} = \frac{5}{s+5}$$

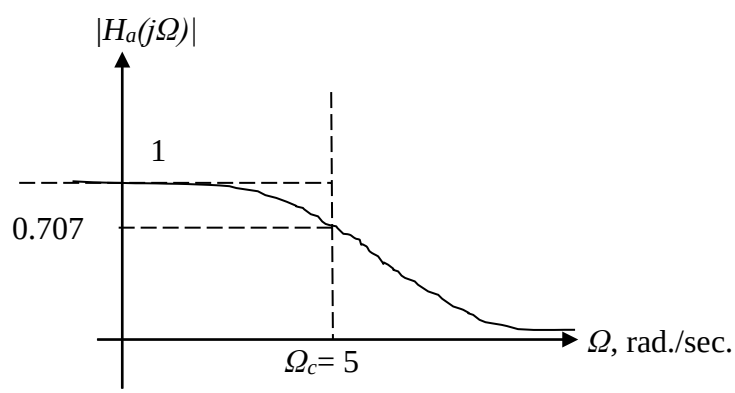
Taking the inverse Laplace transform gives the impulse response,

$$h_a(t) = 5\, e^{-5t} u(t)$$

The frequency response is

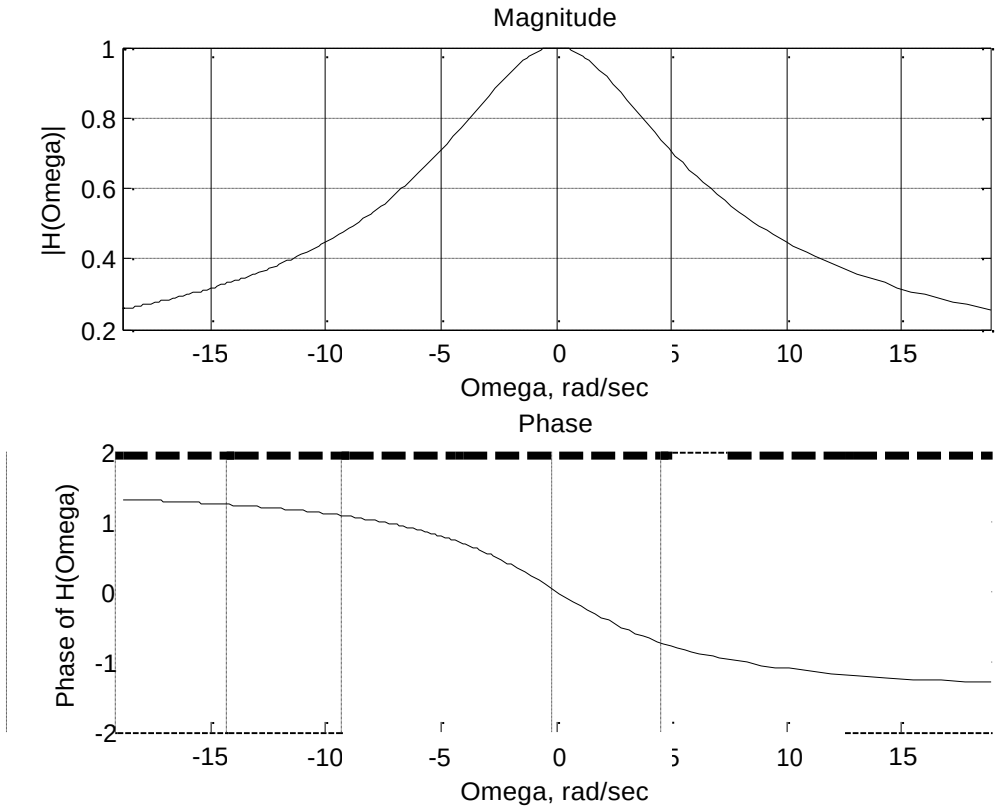
$$H_a(j\Omega) = H_a(s)\big|_{s=j\Omega} = \frac{5}{j\Omega+5} = \frac{1}{\sqrt{1+(\Omega/5)^2}} e^{-j\tan^{-1}(\Omega/5)}$$

The cut-off frequency is $\Omega_c = 5$ rad/sec. The gain at $\Omega = 0$ is 1.



The MATLAB plots of frequency response of $H_a(j\Omega) = 5/(j\Omega+5)$ are shown below. We use the function ***fplot***. The analog frequency, Omega (Ω), extends from 0 to ∞; however, the plots cover the range 0 to 6π rad/sec.

```
subplot(2, 1, 1), fplot('abs(5/(5+j*Omega))', [-6*pi, 6*pi], 'k');
xlabel ('Omega, rad/sec'), ylabel('|H(Omega)|'); grid; title ('Magnitude')
%
subplot(2, 1, 2), fplot('angle(5/(5+j*Omega))', [-6*pi, 6*pi], 'k');
xlabel ('Omega, rad/sec'), ylabel('Phase of H(Omega)'); grid; title ('Phase')
```



If we adjust the values of the components R and C so that $1/RC = 1$, we would have $H_a(s) = \frac{1}{s+1}$ which is a *normalized filter* with cut-off frequency $\Omega_c = 1$ rad/sec and gain of 1 at $\Omega = 0$. Such a normalized LP filter could be transformed to another LP filter with a different cut-off frequency of, say, $\Omega_c = 10$ rad/sec by the *low pass to low pass transformation* $s \rightarrow (s/10)$. The transfer function then becomes

$$H_a(s) = \left.\frac{1}{s+1}\right|_{s \rightarrow (s/10)} = \frac{10}{s+10}$$

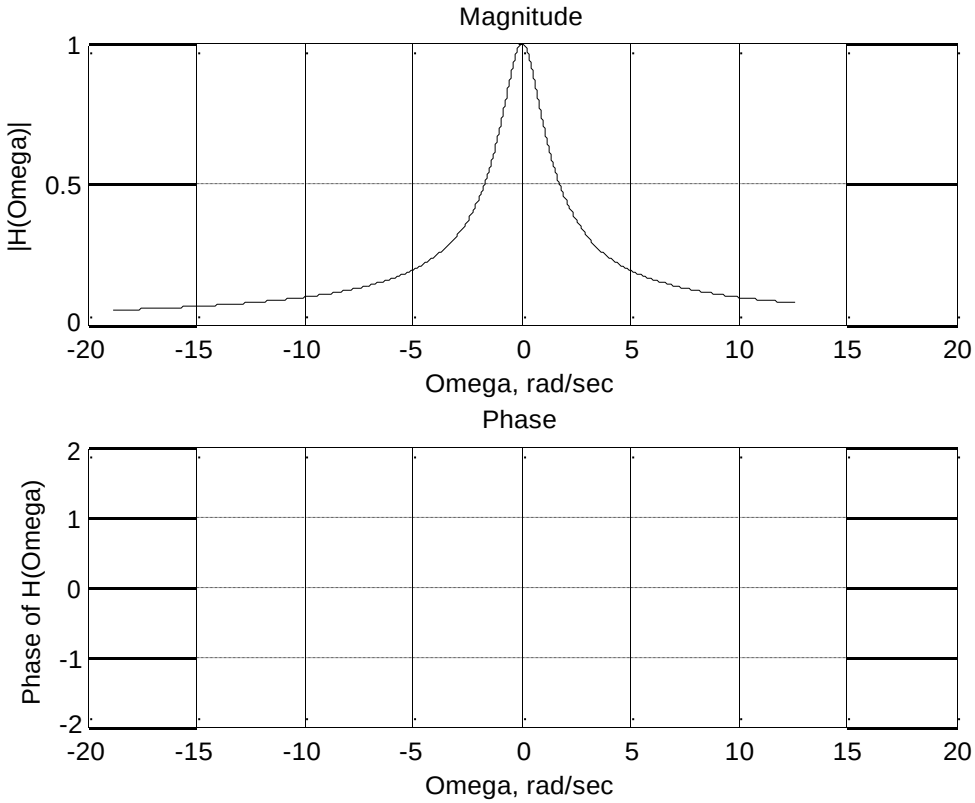
which still has a dc gain of 1. The gain of this filter could be *scaled* by a multiplier, say, K, so that

$$H_a(s) = K\frac{10}{s+10}$$

which has a dc gain of K and a cut-off frequency of $\Omega_c = 10$ rad/sec.

The frequency response of the normalized filter $H_a(s) = 1/(s+1)$ is $H_a(j\Omega) = 1/(j\Omega+1)$. The corresponding MATLAB plots are shown below using the function ***plot***. Omega is a *vector*, consequently we use "./" instead of "/" etc.

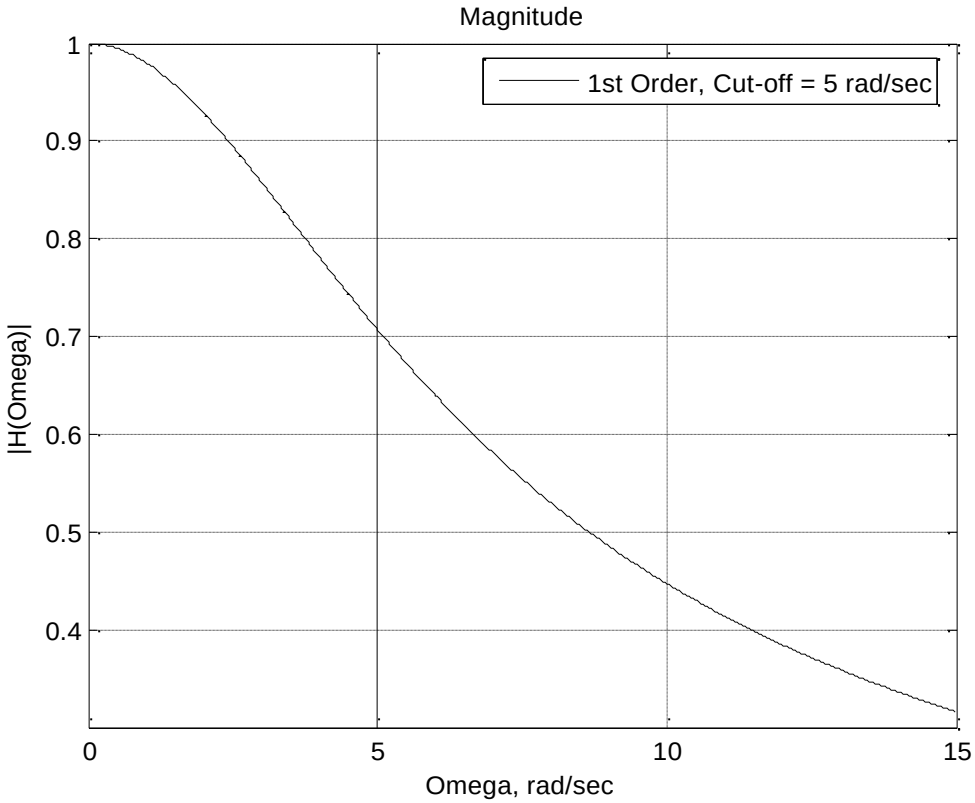
```
Omega = -6*pi: pi/256: 6*pi; H = 1./(1.+ j .*Omega);
subplot(2, 1, 1), plot(Omega, abs(H), 'k');
xlabel ('Omega, rad/sec'), ylabel('|H(Omega)|'); grid; title ('Magnitude')
subplot(2, 1, 2), plot(Omega, angle(H), 'k');
xlabel ('Omega, rad/sec'), ylabel('Phase of H(Omega)'); grid; title ('Phase')
```



Butterworth filter The filter $H_a(s) = 5/(s + 5)$ is a first order Butterworth filter with cut-off frequency $\Omega_c = 5$ rad/sec. Its magnitude response is given by

$$|H(j\Omega)| = \frac{1}{\sqrt{1+(\Omega/5)^2}}$$

```
Omega = 0: pi/256: 15; H1 = 1./sqrt(1.+ (Omega/5) .^2);
plot(Omega, H1, 'k'); legend ('1st Order, Cut-off = 5 rad/sec');
xlabel ('Omega, rad/sec'), ylabel('|H(Omega)|'); grid; title ('Magnitude')
```

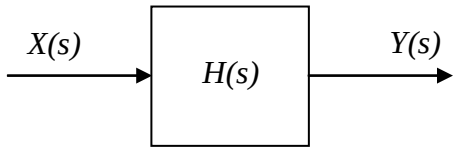

Magnitude
1st Order, Cut-off = 5 rad/sec
|H(Omega)|
1
0.9
0.8
0.7
0.6
0.5
0.4
0
5
10
15
Omega, rad/sec

Frequency response analysis The frequency response analysis in the analog frequency (Ω) domain is given by the following equations which may be used to illustrate, qualitatively, the effect of LP, HP or BP analog filters on a signal.

$$Y(s) = H(s)\,X(s)$$

$$Y(j\Omega) = H(j\Omega)\,X(j\Omega) = |H(j\Omega)|\,e^{j\angle H(j\Omega)}\,|X(j\Omega)|\,e^{j\angle X(j\Omega)} = |H(j\Omega)|\,|X(j\Omega)|\,e^{j(\angle H(j\Omega)+\angle X(j\Omega))}$$

$$|Y(\Omega)| = |H(\Omega)|\,|X(\Omega)| \quad \text{and} \quad \angle Y(\Omega) = \angle H(\Omega) + \angle X(\Omega)$$

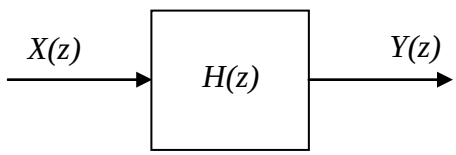


Similarly, if we have a digital filter $H(z)$ the frequency response analysis in the digital frequency (ω) domain is given by the following equations which may be used to illustrate, qualitatively, the effect of LP, HP or BP digital filters on a signal.

$$Y(z) = H(z)\,X(z)$$

$$Y(j\omega) = H(j\omega)\,X(j\omega) = |H(j\omega)|\,e^{j\angle H(j\omega)}\,|X(j\omega)|\,e^{j\angle X(j\omega)} = |H(j\omega)|\,|X(j\omega)|\,e^{j(\angle H(j\omega)+\angle X(j\omega))}$$

$$|Y(\omega)| = |H(\omega)|\,|X(\omega)| \quad \text{and} \quad \angle Y(\omega) = \angle H(\omega) + \angle X(\omega)$$

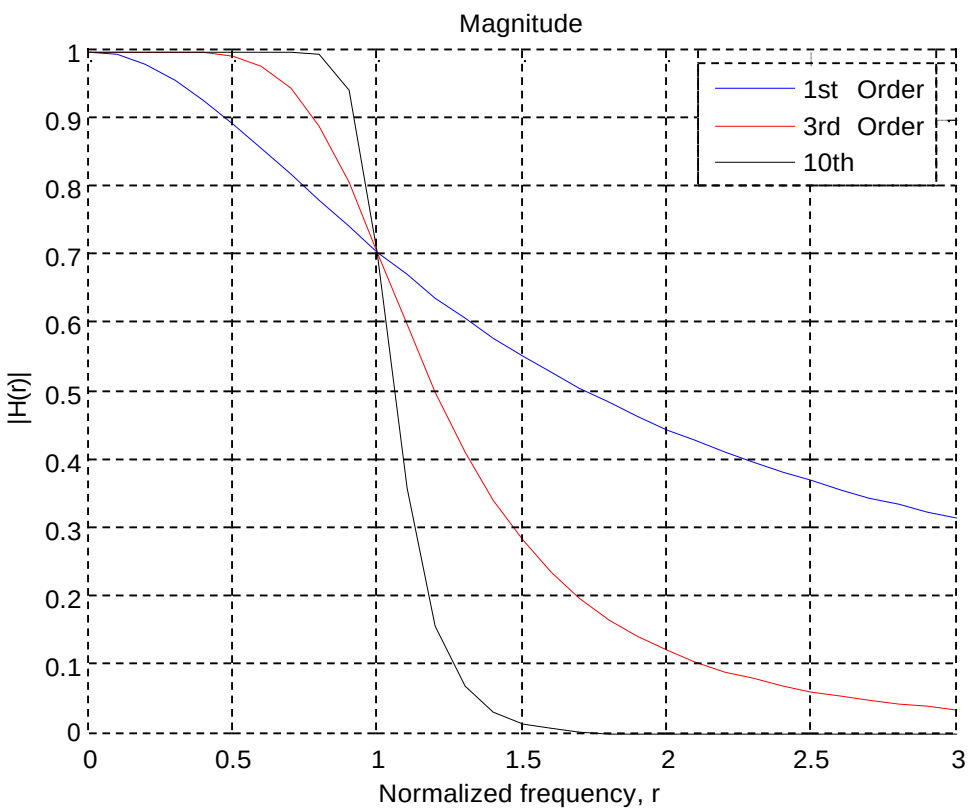


The Nth order Butterworth filter In general the magnitude response of the Nth order Butterworth filter with cut-off frequency Ω_c is given by

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

With the normalized frequency variable defined as $r = \Omega / \Omega_c$, the MATLAB segment below plots the magnitude response for 1st, 3rd, and 10th order filters, that is, N = 1, 3, and 10, respectively. Note that as the filter order increases the response becomes flatter on either side of the cut-off; and the transition (cut-off) becomes sharper.

```
r = 0: 0.1: 3;
H1 = 1./sqrt(1.+ r .^2);
H3 = 1./sqrt(1.+ r .^6);
H10 = 1./sqrt(1.+ r .^20);
plot (r, H1, r, H3, 'r', r, H10, 'k');
legend ('1st Order', '3rd Order', '10th Order');
xlabel ('Normalized frequency, r'), ylabel('|H(r)|'); grid; title ('Magnitude')
```



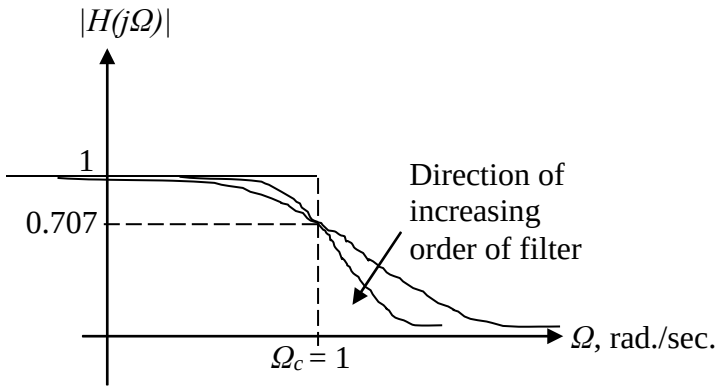
Writing down the Nth order filter transfer function $H(s)$ from the pole locations Let us look at some analog Butterworth filter theory.

(1) $|H(j\Omega)| = \dfrac{1}{\sqrt{1+(\Omega/\Omega_c)^{2N}}}$ decreases monotonically with Ω. No ripples.

(2) Poles lie on the unit circle in the s-plane (for the Chebyshev filter, in contrast, they lie on an ellipse).

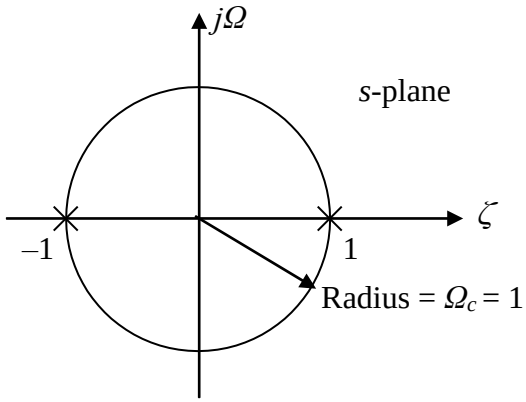
(3) The transition band is wider (than in the case of the Chebyshev filter).

(4) For the same specifications the Butterworth filter has more poles (or, is of higher order) than the Chebyshev filter. This means that the Butterworth filter needs more components to build.



The normalized analog Butterworth filter has a gain of $|H(j\Omega)| = 1$ at $\Omega = 0$, and a cut-off frequency of $\Omega_c = 1$ rad/sec. Given the order, N, of the filter we want to be able to write down its transfer function from the pole locations on the Butterworth circle.

Example 3.2.2 Given the order N of the filter, divide the unit circle into $2N$ equal parts and place poles on the unit circle at $(360^0/2N)$ apart. The $H(s)$ will be made up of the N poles in the left half plane only. Remember complex valued poles must occur as complex conjugate pairs. There will

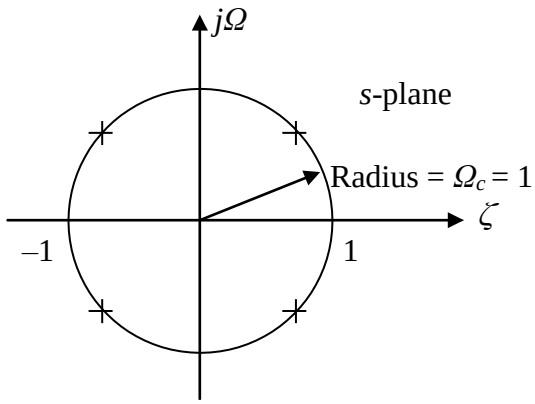


be no poles on the imaginary axis. Since the N poles must lie on the left half semicircle, when N is odd the odd pole must be at $s = -1$. Thus, for $N = 1$, there is one pole, $s_1 = -1$, and $H(s)$ is given by

$$H(s) = \frac{1}{s - s_1} = \frac{1}{s - (-1)} = \frac{1}{s + 1}$$

Example 3.2.3 Filter order $N = 2$ so that $2N = 4$ and $360^0/4 = 90^0$. The pole plot is shown above. The poles are at

$$s_1 = \left(-\frac{1}{\sqrt{2}} + j\frac{1}{\sqrt{2}}\right) \quad \text{and} \quad s_2 = \left(-\frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}}\right),$$



so that

$$H(s) = \frac{1}{(s - s_1)(s - s_2)} = \frac{1}{\left[s + \left(-\frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}}\right)\right]\left[s + \left(-\frac{1}{\sqrt{2}} + j\frac{1}{\sqrt{2}}\right)\right]}$$

Denominator is

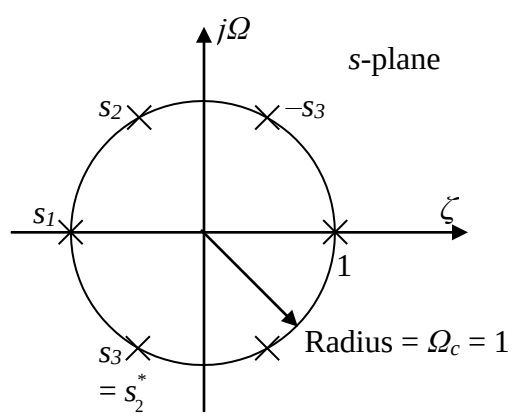
$$\text{Dr.} = \left(s + \frac{1}{\sqrt{2}}\right)^2 - \left(j\frac{1}{\sqrt{2}}\right)^2 = s^2 + \frac{1}{2} + 2\frac{1}{\sqrt{2}}s - j^2\frac{1}{2} = s^2 + \sqrt{2}s + 1$$

So

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

Example 3.2.4 Filter order $N = 3$, so that $2N = 6$ and $360^0/6 = 60^0$. Poles are at

$$s_{1,\,2,\,3} = -1, \left(-\frac{1}{2} + j\frac{\sqrt{3}}{2}\right), \text{ and } \left(-\frac{1}{2} - j\frac{\sqrt{3}}{2}\right) \quad \frac{1}{s^2 + \sqrt{2}\,s + 1}$$



$$H(s) = \frac{1}{(s+1)\left(s+\frac{1}{2}-j\frac{\sqrt{3}}{2}\right)\left(s+\frac{1}{2}+j\frac{\sqrt{3}}{2}\right)}$$

Denominator is

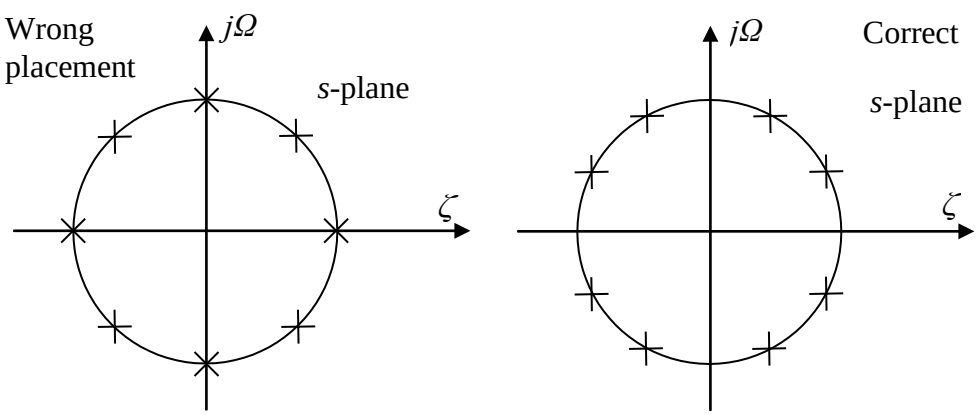
$$\text{Dr.} = (s+1)\left[\left(s+\frac{1}{2}\right)^2 - j^2\frac{3}{4}\right] = (s+1)\left(s^2+s+1\right)$$

So

$$H(s) = \frac{1}{(s+1)\left(s^2+s+1\right)}$$

Example 3.2.5 Filter order $N = 4$, so that $2N = 8$ and $360^0/8 = 45^0$. Poles are at

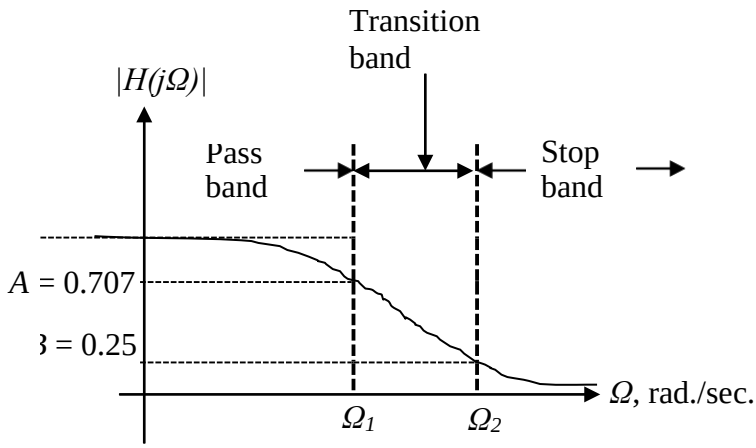
$$s = (-\cos 22.5^0 \pm j \sin 22.5^0) = (-\cos\alpha \pm j\sin\alpha)\ (-\cos 67.5^0 \pm j\sin 67.5^0) = (-\cos\beta \pm j\sin\beta)$$



$$H(s) = \frac{1}{\left[(s+\cos\alpha)^2 - j^2\sin^2\alpha\right]\left[(s+\cos\beta)^2 - j^2\sin^2\beta\right]}$$

$$H(s) = \frac{1}{\left(s^2+1.848s+1\right)\left(s^2+0.765s+1\right)}$$

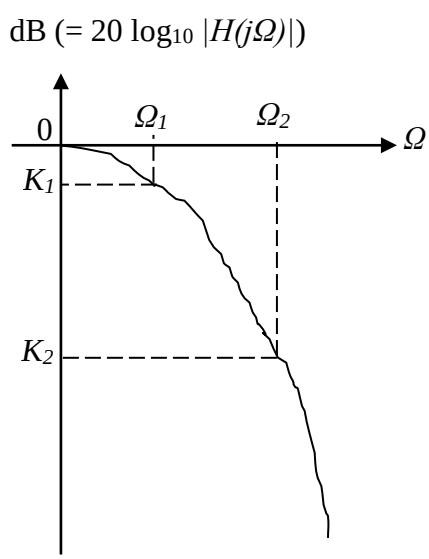
Determining the order and transfer function from the specifications A typical *magnitude* response specification is sketched below. The magnitudes at the critical frequencies Ω_1 and Ω_2 are A and B, respectively. Typically Ω_1 is in the pass band or is the edge of the pass band and Ω_2 is in the stop band or is the edge of the stop band. For illustrative purposes we have arbitrarily



taken $A = 0.707$ (thus Ω_1 is the cut-off frequency, but this need not be the case) and $B = 0.25$.

The *log-magnitude* specification is diagrammed below. Note that $(20 \log A) = K_1$ dB and $(20 \log B) = K_2$ dB. Thus the analog filter specifications are

$$0 \geq 20\log_{10}|H(j\Omega)| \geq K_1 \text{ for all } \Omega \leq \Omega_1$$
$$20\log_{10}|H(j\Omega)| \leq K_2 \text{ for all } \Omega \geq \Omega_2$$



With the magnitude $|H(j\Omega)|$ given by the Butterworth function,

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

and using the equality condition at the critical frequencies in the above specifications the order, N, of the filter is given by

$$N = \left\lceil \frac{\log_{10}\left(\dfrac{10^{-K_1/10}-1}{10^{-K_2/10}-1}\right)}{2\log_{10}\left(\dfrac{\Omega_1}{\Omega_2}\right)} \right\rceil \qquad \rightarrow \text{(3.16, Ludeman)}$$

The result is rounded to the next larger integer. For example, if $N = 3.2$ by the above calculation then it is rounded up to 4, and the order of the required filter is $N = 4$. In such a case the resulting filter would exceed the specification at both Ω_1 and Ω_2. The cut-off frequency Ω_c is determined from one of the two equations below.

$$\Omega_c = \frac{\Omega_1}{\sqrt[2N]{10^{-K_1/10}-1}} \qquad \text{or} \qquad \Omega_c = \frac{\Omega_2}{\sqrt[2N]{10^{-K_2/10}-1}} \qquad \rightarrow \text{(3.17 Ludeman)}$$

The equation on the left will result in the specification being met exactly at Ω_1 while the specification is exceeded at Ω_2. The equation on the right results in the specification being met exactly at Ω_2 and exceeded at Ω_1.

Note that the design equation for N may be written in the alternative form

$$N = \frac{\log_{10}\left(\dfrac{10^{-K_2/10}-1}{10^{-K_1/10}-1}\right)}{2\log_{10}\left(\dfrac{\Omega_2}{\Omega_1}\right)}$$

Example 3.2.6 What is the order and transfer function of the analog Butterworth filter that satisfies the following specification?

$\Omega_1 = 200$ rad/sec $\qquad K_1 = -1$ dB

$\Omega_2 = 600$ rad/sec $\qquad K_2 = -30$ dB

Solution The order N is given by

$$N = \left\lceil \frac{\log_{10}\left(\dfrac{10^{-K_1/10}-1}{10^{-K_2/10}-1}\right)}{2\log_{10}\left(\dfrac{\Omega_1}{\Omega_2}\right)} \right\rceil = \left\lceil \frac{\log_{10}\left(\dfrac{10^{-(-1)/10}-1}{10^{-(-30)/10}-1}\right)}{2\log_{10}\left(\dfrac{200}{600}\right)} \right\rceil = \left\lceil \frac{\log_{10}\left(\dfrac{10^{0.1}-1}{10^{3}-1}\right)}{2\log_{10}\left(\dfrac{200}{600}\right)} \right\rceil$$

$$= \left\lceil \frac{\log_{10}\left(\dfrac{1.2589-1}{1000-1}\right)}{2\log_{10}\left(\dfrac{200}{600}\right)} \right\rceil = \left\lceil \frac{\log_{10}\left(\dfrac{0.2589}{999}\right)}{2\log_{10}\left(\dfrac{200}{600}\right)} \right\rceil = \left\lceil \frac{\log_{10}\left(\dfrac{0.2589}{999}\right)}{2\log_{10}\left(\dfrac{200}{600}\right)} \right\rceil$$

$$= \left\lceil \frac{\log_{10}(0.00025918)}{2\log_{10}(0.3333333)} \right\rceil = \left\lceil \frac{-3.586}{2(-0.47712)} \right\rceil = \lceil 3.76 \rceil = 4$$

Now, as in an earlier example, locate 8 poles uniformly on the unit circle, making sure to satisfy all the requirements … and write down the transfer function, $H(s)$, of the normalized Butterworth filter (with a cut-off frequency of 1 rad/sec),

$$H(s) = \frac{1}{\left(s^2+1.848s+1\right)\left(s^2+0.765s+1\right)}$$

Next, determine the cutoff frequency Ω_c that corresponds to the given specifications and the order $N = 4$ determined above

$$\Omega_c = \frac{\Omega_1}{\sqrt[2N]{10^{-K_1/10}-1}} = \frac{200}{\sqrt[2(4)]{10^{-(-1)/10}-1}} = \frac{200}{\sqrt[8]{1.26-1}} = \frac{200}{0.8446} = 236.8 \text{ rad/sec}$$

Finally, we make the substitution $s \to (s / 236.8)$ in $H(s)$ and thereby move the cutoff frequency from 1 rad/sec to 236.8 rad/sec resulting in the transfer function $H_a(s)$

$$\left| \frac{1}{s + 1.848s + 1)\left(s + 0.765s + 1\right)} \right|_{s \to (s/236.8)}$$

$$= \frac{1}{((s/236.8)^2 + 1.848(s/236.8) + 1)\left((s/236.8)^2 + 0.765(s/236.8) + \right)1}$$

$$= \frac{(236.8)^4}{\left(s^2 + 1.848(236.8)s + 236.8^2\right)\left(s^2 + 0.765(2326.8)s + 236.8^2\right)}$$

$$= \ldots$$

(Aside) **The more general N^{th} order Butterworth filter** has the magnitude response given by

$$|H(j\Omega)| = \frac{1}{\sqrt{1 + \varepsilon^2 (\Omega/\Omega_1)^{2N}}}$$

The parameter ε has to do with pass band attenuation and Ω_1 is the pass band edge frequency (not necessarily the same as the 3 dB cut-off frequency Ω_c). MATLAB takes $\varepsilon = 1$ in which case $\Omega_1 = \Omega_c$. See DSP-HW. [See Cavicchi, Ramesh Babu].

(End of Aside)

Time domain invariance

Given an analog filter's response to a specific input we require that the response of the digital filter (to be designed) to the digital version of the analog input should be the same as the analog response at sampling instants. If the input is an impulse function the corresponding design is called an impulse invariant design, if the input is a step function the corresponding design is called a step invariant design.

Impulse-invariant design If $h_a(t)$ represents the response of an analog filter $H_a(s)$ to a unit impulse $6(t)$, then the unit sample response of a discrete-time filter used in an *A/D – H(z) – D/A* structure is selected to be the sampled version of $h_a(t)$. That is, we are preserving the response to an impulse. Therefore the discrete-time filter is characterized by the system function, $H(z)$, given by

$$H(z) = \mathcal{Z}\{h(n)\} = \mathcal{Z}\left[\{h_a(t)\}\big|_{t=nT}\right]$$

If we are given an analog filter with system function $H_a(s)$ the corresponding impulse-invariant digital filter, $H(z)$, is seen from above to be

$$H(z) = \mathcal{Z}\left[\left(L^{-1}\{H_a(s)\}\right)\big|_{t=nT}\right], \text{ where } L^{-1} \text{ means Laplace inverse}$$

Note that at this point we have not specified how $H_a(s)$ was obtained, but rather we have shown how to obtain the digital filter $H(z)$ from any given $H_a(s)$ using impulse invariance.

Example 5.3.1 [Low pass filter] For the analog filter $H_a(s) = \dfrac{A}{s+\alpha}$ find the $H(z)$ corresponding to the impulse invariant design using a sample rate of $1/T$ samples/sec.

Solution The analog system's impulse response is $h_a(t) = \mathcal{L}^{-1}\left(\dfrac{A}{s+\alpha}\right) = Ae^{-\alpha t}u(t)$. The corresponding $h(n)$ is then given by

$$h(n) = h_a(t)\big|_{t=nT} = Ae^{-\alpha nT}u(nT) = A\left(e^{-\alpha T}\right)^n u(n) = Aa^n u(n)$$

where, as previously, we have set $e^{-\alpha T} = a$. The discrete-time filter, then, is given by the z-transform of $h(n)$

$$H(z) = \mathcal{Z}\{h(n)\} = \mathcal{Z}\left\{A\left(e^{-\alpha T}\right)^n u(n)\right\} = \frac{Az}{z - e^{-\alpha T}} = \frac{Az}{z-a}$$

$$= \frac{A}{1 - e^{-\alpha T}z^{-1}} = \frac{A}{1 - az^{-1}}$$

which has a pole at $z = e^{-\alpha T} = a$. In effect, the pole at $s = -\alpha$ in the s-plane is mapped to a pole at $z = e^{-\alpha T} = a$ in the z-plane. (**HW** What is the difference equation?)

$$\frac{Y(z)}{X(z)} = \frac{A}{1 - az^{-1}} \quad \rightarrow \quad Y(z)(1 - az^{-1}) = A\,X(z)$$

$$\rightarrow \quad y(n) - ay(n-1) = Ax(n) \quad \rightarrow \quad y(n) = Ax(n) + ay(n-1)$$

Relationship between the s-plane and the z-plane ($z = e^{sT}$) We can extend the above procedure to the case where $H_a(s)$ is given as a sum of N terms with distinct poles as

$$H_a(s) = \sum_{k=1}^{N} \frac{A_k}{s + \alpha_k}$$

For this case the impulse invariant design, $H(z)$, is given by

$$H(z) = \mathcal{Z}\left\{L^{-1}\left\{\sum_{k=1}^{N} \frac{A_k}{s+\alpha_k}\right\}\Big|_{t=nT}\right\} = \sum_{k=1}^{N} \frac{A_k z}{z - e^{-\alpha_k T}} = \sum_{k=1}^{N} \frac{A_k}{1 - e^{-\alpha_k T}z^{-1}}$$

where L^{-1} means Laplace inverse. We observe that a pole at $s = -\alpha_k$ in the s-plane gives rise to a pole at $z = e^{-\alpha_k T}$ in the z-plane and the coefficients in the partial fraction expansion of $H_a(s)$ and $H(z)$ are equal. If the analog filter is stable, corresponding to $-\alpha_k$ being in the left half plane, then the magnitude of $e^{-\alpha_k T}$ will be less than unity, so that the corresponding pole of the digital filter is inside the unit circle, and as a result the digital filter also is stable.

While the poles in the s-plane "map" to poles in the z-plane according to the relationship $z = e^{sT}$, it is important to recognize that the impulse invariance design procedure does not correspond to a *mapping* (*transformation*) of the s-plane to the z-plane by that relationship or in fact by any relationship. (An example of a transformation is where we actually make a *substitution*, say, $s = \dfrac{2}{T}\left(\dfrac{1 - z^{-1}}{1 + z^{-1}}\right)$ which, of course, is the bilinear transformation). For example, the zeros of $H_a(s)$ do not map to zeros of $H(z)$ according to this relation. See also matched z-transform later.

We can explore the relationship $z = e^{sT}$ keeping in mind that it only applies to poles and that it is not a transformation. Set $s = \zeta + j\Omega$ and $z = r\,e^{j\omega}$ in $z = e^{sT}$ to get $r\,e^{j\omega} = e^{(\sigma + j\Omega)T} = e^{\sigma T}e^{j\Omega T}$ so that $r = e^{\sigma T}$ and $\omega = \Omega T$.

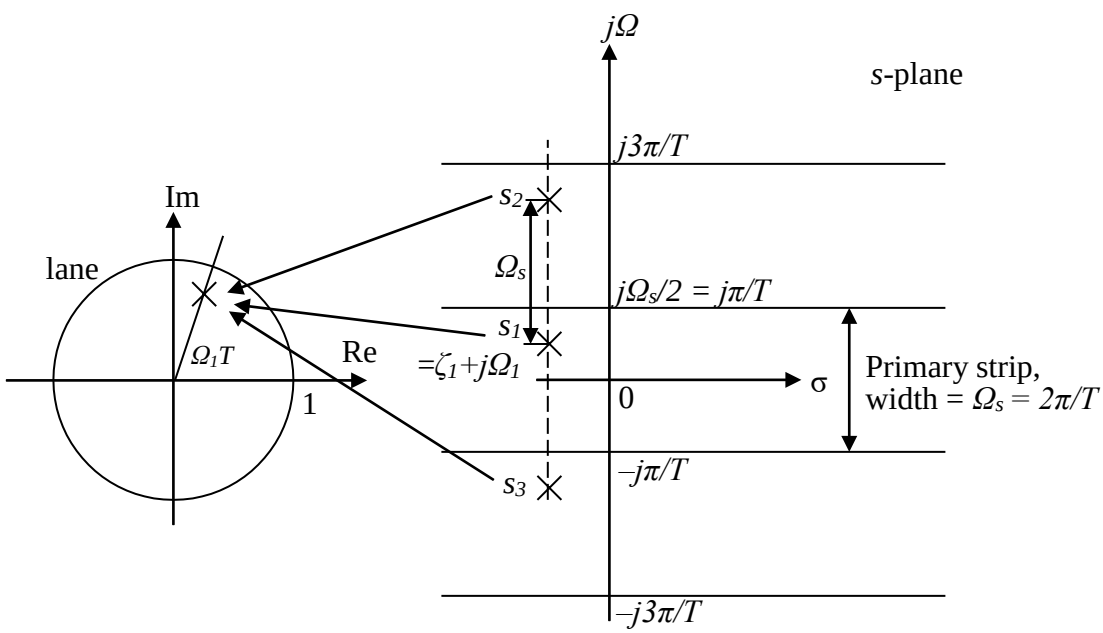
The above relations can be used to show that poles in the left half of the primary strip in the s-plane map into poles within the unit circle in the z-plane as shown in the figure for $s = s_1$.

Mapping of poles, $z = e^{sT}$

s-plane pole $s = \zeta + j\Omega$	z-plane pole $z = e^{sT} = r\,e^{j\omega}$	r	ω
0	1	1	0
$j\Omega_s/2$	–1	1	π
$-\infty + j\Omega_s/2$	–0	0	π
$-\infty - j\Omega_s/2$	–0	0	π
$-j\Omega_s/2$	–1	1	π

For $s_1 = \zeta_1 + j\Omega_1$ we have $r = e^{\sigma_1 T}$ and $\omega = \Omega_1 T$. However, poles at s_2 and s_3 (which are a distance Ω_s from s_1) also will be mapped to the same pole that s_1 is mapped to. In fact, an infinite number of s-plane poles will be mapped to the same z-plane pole in a many-to-one relationship. These frequencies differ by $\Omega_s = 2\pi F_s = 2\pi/T$ (F_s is the sampling frequency in Hertz). This is called **aliasing (of the poles)** and is a drawback of the impulse-invariant design. The analog system poles will not be aliased in this manner if, in the first place, they are confined to the "primary strip" of width $\Omega_s = 2\pi F_s = 2\pi/T$ in the s-plane.

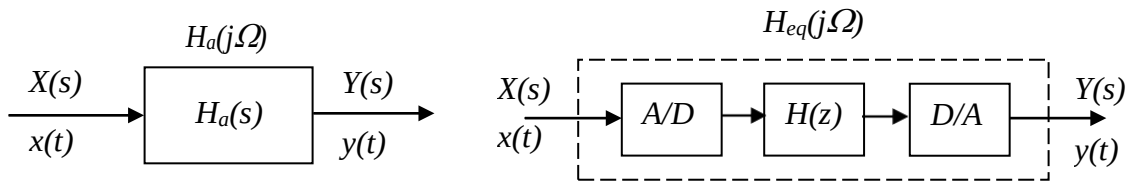
In a similar fashion poles located in the right half of the primary strip in the s-plane will be mapped to the outside of the unit circle in the z-plane. Here again the mapping of the s-plane poles to the z-plane poles is many-to-one.



Owing to the aliasing, the impulse invariant design is suitable for the design of low pass and band pass filters but not for high pass filters.

(Omit) Matched z-transform In this method we apply the mapping $z = e^{sT}$ not only to the poles but also to the zeros of $H_a(s)$. As a result the observations made above are valid for the matched ransform method of filter design.

Frequency response of the equivalent analog filter Going back to the impulse invariant design of the first order filter, how does the frequency response of the *A/D – H(z) – D/A* structure using this *H(z)* compare to the frequency response of the original system specified by $H_a(s)$?



Note $H_a(s) = \dfrac{A}{s+\alpha}$ and $H(z) = \dfrac{Az}{z - e^{-\alpha T}} = \dfrac{Az}{z-a}$ with $a = e^{-\alpha T}$. For the analog filter we have

$$H_a(j\Omega) = \left.\frac{A}{s+\alpha}\right|_{s=j\Omega} = \frac{A}{j\Omega+\alpha} = \frac{A}{\sqrt{\alpha^2+\Omega^2}}\, e^{-j\tan^{-1}(\Omega/\alpha)}$$

$$|H_a(j\Omega)| = \frac{A}{\sqrt{\alpha^2+\Omega^2}}, \qquad -\infty < \Omega < \infty$$

For future reference note that $|H_a(j0)| = A/\alpha$.

To obtain the equivalent frequency response of the *A/D – H(z) – D/A* structure one must first find the frequency response of the discrete-time filter specified by *H(z)*. This is given by

$$H(e^{j\omega}) = H(z)\Big]_{z=e^{j\omega}} = \frac{Ae^{j\omega}}{e^{j\omega} - e^{-\alpha T}}, \qquad -\pi < \omega < \pi \text{ (because periodic)}$$

The analog frequency response of the *equivalent* analog filter is then determined by replacing ω by ΩT. Note, however, that since the digital frequency, ω, is restricted to $(-\pi, \pi)$, the analog frequency, Ω, is correspondingly is restricted to $(-\pi/T, \pi/T)$. We get

$$H_{eq}(j\Omega) = H(e^{j\omega})\Big]_{\omega=\Omega T} = \frac{Ae^{j\Omega T}}{e^{j\Omega T} - e^{-\alpha T}} = \frac{A}{1 - e^{-\alpha T}e^{-j\Omega T}}, \qquad \Omega T < \pi \text{ or } \Omega < \pi/T$$

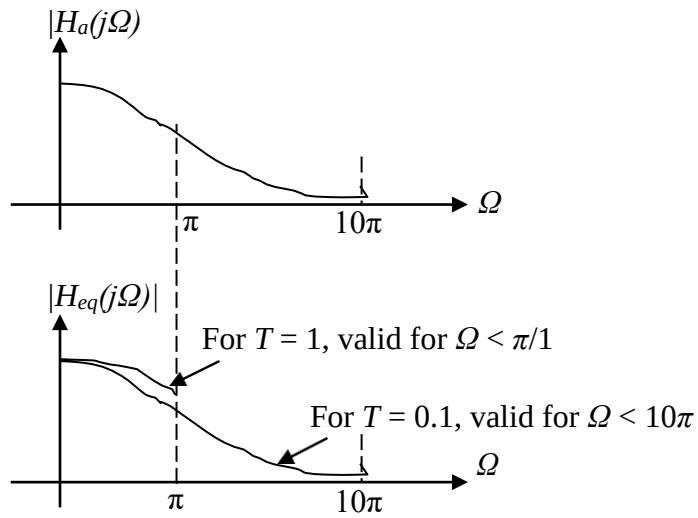
$$\text{Denominator} = 1 - e^{-\alpha T}e^{-j\Omega T} = 1 - e^{-\alpha T}(\cos\Omega T - j\sin\Omega T) = (1 - e^{-\alpha T}\cos\Omega T) + je^{-\alpha T}\sin\Omega T$$

So
$$H_{eq}(j\Omega) = \frac{A}{(1 - e^{-\alpha T}\cos\Omega T) + je^{-\alpha T}\sin\Omega T}, \qquad \Omega < \pi/T$$

$$|H_{eq}(j\Omega)| = \frac{A}{\sqrt{1 + e^{-2\alpha T} - 2e^{-\alpha T}\cos\Omega T}}, \qquad \Omega < \pi/T$$

Note that $|H_{eq}(j0)| = \dfrac{A}{1 - e^{-\alpha T}} = \dfrac{A}{1-a}$.

We can plot $|H_{eq}(j\Omega)|$ and $|H_a(j\Omega)|$ for, say, $\alpha = 1$ and different values of T say $T = 0.1$ and $T = 1$, remembering that $|H_{eq}(j\Omega)|$ is periodic, the basic period going from $-\pi/T < \Omega < \pi/T$. Ideally the two plots should be very close (in shape, over the range of frequencies of interest) but it will be found that the smaller the value of T, the closer the two plots are. Thus $T = 0.1$ will result in a closer match than $T = 1$. Therefore, using the impulse invariant design, good results are obtained provided the time between samples (T) is selected small enough. What is small enough may be difficult to assess when the $H_a(s)$ has several poles; and when it is found, it may be so small that implementation may be costly. In general, other transformational methods such as the bilinear allow designs with sample rates that are less than those required by the impulse invariant method and also allow flexibility with respect to selection of sample rate size.



Example 3.3.2 [Impulse invariant design of 2nd order Butterworth filter] Obtain the impulse invariant digital filter corresponding to the 2nd order Butterworth filter $H_a(s) = \dfrac{4}{s^2 + 2\sqrt{2}s + 4}$ with sampling time $T = 1$ sec.

Solution Note that we are using $T = 1$ sec. simply for the purpose of comparing with the bilinear design done later with $T = 1$ sec. It is important to remember that in bilinear design calculations the value of T is immaterial since it gets cancelled in the *design process*; but in impulse invariant design there is no such cancellation, so the value of T is critical (the smaller, the better).

$$H_a(s) = \frac{4}{s^2 + 2\sqrt{2}s + 4} = \frac{4}{s^2 + 2\sqrt{2}s + \left(\sqrt{2}\right)^2 + \left(\sqrt{2}\right)^2}$$

$$= \frac{4}{\left(s + \sqrt{2}\right)^2 + \left(\sqrt{2}\right)^2} = 2\sqrt{2}\left\{\frac{\sqrt{2}}{\left(s + \sqrt{2}\right)^2 + \left(\sqrt{2}\right)^2}\right\}$$

The expression in braces is in familiar form and can be converted to its impulse invariant digital filter equivalent. See 3(c) in HW.

Step invariant design Here the response of the digital filter to the unit step sequence, $u(n)$, is chosen to be samples of the analog step response. In this way, if the analog filter has good step response characteristics, such as small rise-time and low peak over-shoot, these characteristics would be preserved in the digital filter. Clearly this idea of waveform invariance can be extended to the preservation of the output wave shape for a variety of inputs.

(Omit) **Problem** Given the analog system $H_a(s)$, let $h_a(t)$ be its impulse response and let $p_a(t)$ be its step response. The system $H_a(s)$ is given to be continuous-time linear time-invariant. Let also

$h(n)$ be the unit sample response,
$p(n)$ be the step response, and,
$H(z)$ be the system function,

of a discrete-time linear shift-invariant filter. Then,

(a) If $h(n) = h_a(nT)$, does $p(n) = \sum_{k=-\infty}^{n} h_a(kT)$?

(b) If $p(n) = p_a(nT)$, does $h(n) = h_a(nT)$?

Solution (a) If $h(n) = h_a(nT)$, does $p(n) = \sum_{k=-\infty}^{n} h_a(kT)$? We know that

$$u(n) = \sum_{k=-\infty}^{n} \delta(k)$$

This is seen to be true by writing it out in full as

$$u(n) = 6(-\infty) + 6(-\infty+1) + \ldots + 6(-1) + 6(0) + 6(1) + \ldots + 6(n)$$

where n is implicitly some positive integer. Take, for instance, $n = 3$; then, from the above equation $u(3) = 6(0) = 1$, all the other terms being zero. In other words $u(n)$ is a linear combination of unit sample functions. And, since the response to $6(k)$ is $h(k)$, therefore, the response to $u(n)$ is a linear combination of the unit sample responses $h(k)$. That is,

$$p(n) = \sum_{k=-\infty}^{n} h(k) = \sum_{k=-\infty}^{n} h_a(kT)$$

Therefore, the answer to the above question is, Yes.

(b) If $p(n) = p_a(nT)$, does $h(n) = h_a(nT)$? Since $6(n) = u(n) - u(n-1)$, the response of the digital system, $H(z)$, to the input $6(n)$ is

$$h(n) = p(n) - p(n-1) = p_a(nT) - p_a(nT-T) \neq h_a(nT)$$

Therefore, the answer to the above question is, No.

Example 3.3.3 [LP filter] [Step Invariance] Consider the continuous-time system $H_a(s) = \dfrac{A}{s+\alpha}$ with unit step response $p_a(t)$. Determine the system function, $H(z)$, i.e., the z-transform of the unit sample response $h(n)$ of a discrete-time system designed from this system on the basis of step-invariance, such that $p(n) = p_a(nT)$, where

$$p(n) = \sum_{k=-\infty}^{n} h(k) \qquad \text{and} \qquad p_a(t) = \int_{-\infty}^{t} h_a(\tau)\, d\tau$$

Solution Since $p_a(t) = \int_{-\infty}^{t} h_a(\tau)\, d\tau$ we have

$$\mathcal{L}[p_a(t)] = P_a(s) = \frac{H_a(s)}{s} = \frac{A}{s(s+\alpha)} = \frac{K_1}{s} + \frac{K_2}{s+\alpha}$$

where K_1 and K_2 are the coefficients of the partial fraction expansion, given by

$$K_1 = \left.\frac{A}{s+\alpha}\right|_{s=0} = \frac{A}{\alpha} \qquad \text{and} \qquad K_2 = \left.\frac{A}{s}\right|_{s=-\alpha} = -\frac{A}{\alpha}$$

Therefore,

$$P_a(s) = \frac{A/\alpha}{s} - \frac{A/\alpha}{s+\alpha} \leftrightarrow \qquad p_a(t) = \frac{A}{\alpha}e^{-0t}u(t) - \frac{A}{\alpha}e^{-\alpha t}u(t)$$

from which we write $p(n) = p_a(nT)$ and hence $P(z)$ etc. Equivalently, we may reason as follows. The correspondence between s-plane poles and z-plane poles is

$$\frac{1}{s+\alpha} \leftrightarrow \frac{1}{1-e^{-\alpha T}z^{-1}}$$

Thus we get $P(z)$ as below,

$$P(z) = \frac{(A/\alpha)}{1-e^{-0T}z^{-1}} - \frac{(A/\alpha)}{1-e^{-\alpha T}z^{-1}} = \frac{A}{\alpha}\left(\frac{z}{z-1} - \frac{z}{z-e^{-\alpha T}}\right) = \frac{A}{\alpha}\left(\frac{z(z-e^{-\alpha T}) - z(z-1)}{(z-1)(z-e^{-\alpha T})}\right)$$

$$= \frac{A}{\alpha}\left(1-e^{-\alpha T}\right)\frac{z}{(z-1)(z-e^{-\alpha T})}$$

However, what we need is $H(z)$. Since $6(n) = u(n) - u(n-1)$, and we know the response to $u(n)$, therefore, the response to $u(n) - u(n-1)$ is given by $h(n) = p(n) - p(n-1)$, and taking the z-transform of this last equation,

$$H(z) = P(z) - z^{-1}P(z) = (1-z^{-1})P(z) = \left(\frac{z-1}{z}\right)P(z)$$

$$= \left(\frac{z-1}{z}\right)\frac{A}{\alpha}\left(1-e^{-\alpha T}\right)\frac{z}{(z-1)(z-e^{-\alpha T})}$$

$$= \frac{A}{\alpha}\left(1-e^{-\alpha T}\right)\frac{1}{(z-e^{-\alpha T})}$$

Alternatively, we may also obtain the transfer function as the ratio of output and input transforms, $H(z) = P(z)/U(z)$.

Frequency-response analysis As we did in the case of the impulse-invariant design, here also we can compare $|H_a(j\Omega)|$ with $|H_{eq}(j\Omega)|$. For the system $H_a(s) = \frac{A}{s+\alpha}$ the frequency response is already evaluated as $|H_a(j\Omega)| = \frac{A}{\sqrt{\alpha^2+\Omega^2}}$. We need the frequency response, $|H_{eq}(j\Omega)|$, of the equivalent analog filter when the above $H(z)$ is used in a $A/D - H(z) - D/A$ structure. Start with the above $H(z)$ and set $z = e^{j\omega}$ to get

$$H(e^{j\omega}) = H(z)\Big|_{z=e^{j\omega}} = \frac{A}{\alpha}\left(1-e^{-\alpha T}\right)\left.\frac{1}{(z-e^{-\alpha T})}\right|_{z=e^{j\omega}} = \frac{A}{\alpha}\left(1-e^{-\alpha T}\right)\frac{1}{(e^{j\omega}-e^{-\alpha T})}$$

For the equivalent analog filter we get $H_{eq}(j\Omega)$ by setting $\omega = \Omega T$ in $H(e^{j\omega})$.

Bilinear transformation

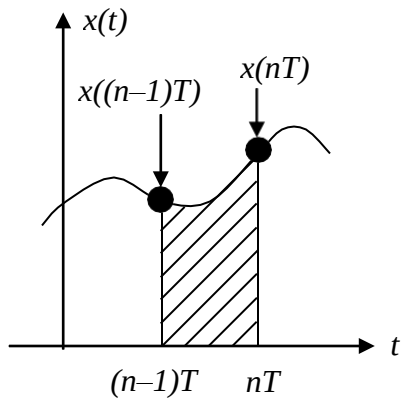
One approach to the *numerical solution* of an ordinary linear constant-coefficient differential equation is based on the application of the *trapezoidal rule* to the *first order approximation of an integral* (or *integration*). Consider the following equivalent pair of equations

$$\frac{dy}{dt} = x(t) \qquad \Rightarrow \qquad \int dy = \int x(t)dt$$

Here $\int dy$ = area shown shaded and is given by

$$y(n) - y(n-1) = \left(\frac{x(n) + x(n-1)}{2}\right)T$$

where we have used the trapezoidal rule to compute the area under a curve.



Taking the z transform of the above we get

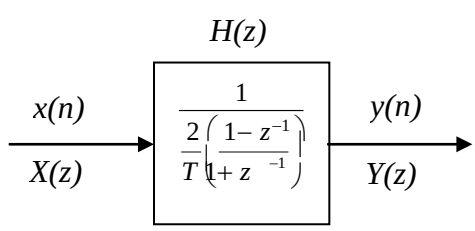
$$Y(z) - z^{-1}Y(z) = \frac{T}{2}X(z)(1 + z^{-1})$$

Rearranging terms gives

$$\frac{Y(z)}{X(z)} = \frac{T}{2}\left(\frac{1 + z^{-1}}{1 - z^{-1}}\right) = \frac{1}{\frac{2}{T}\left(\frac{1 - z^{-1}}{1 + z^{-1}}\right)}$$

Thus the continuous time system $y(t) = \int x(t)dt$ represented by the following block diagrams

is replaced in the discrete-time domain by the following block.



In other words, given the Laplace transfer function $H_a(s)$, the corresponding digital filter is given by replacing s with $\frac{2(1 - z^{-1})}{T(1 + z^{-1})}$, or

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

This is called *bilinear transformation* (both numerator and denominator are first order polynomials), also known as *bilinear z-transformation* (BZT).

Here again, note that at this point we have not specified how $H_a(s)$ was obtained, but rather we are showing how to obtain the digital filter $H(z)$ from any given $H_a(s)$ using bilinear transformation.

Example 3.4.1 [LP filter] [Bilinear] Design a digital filter based on the analog system $H_a(s) = \dfrac{A}{s+\alpha}$, using the bilinear transformation. Give the difference equation. Use $T = 2$ sec.

Solution

$$H(z) = H_a(s)\Big]_{s=\frac{2(1-z^{-1})}{T(1+z^{-1})}} = \frac{A}{\frac{2(1-z^{-1})}{T(1+z^{-1})}+\alpha} = \frac{A}{(\alpha+1)+(\alpha-1)z^{-1}}$$

Example 2 [Ludeman, p. 178] Apply bilinear transformation to the 2nd order Butterworth filter $H_a(s) = \dfrac{4}{s^2+2\sqrt{2}s+4}$ with $T = 1$ sec. Obtain (1) $H(z)$, (2) the difference equation and (3) $H(e^{j\omega})$.

Solution

1. With $T = 1$ second, $s = \dfrac{2}{T}\left(\dfrac{1-z^{-1}}{1+z^{-1}}\right)$ becomes $s = \dfrac{2(1-z^{-1})}{1+z^{-1}}$, and

$$H(z) = H_a(s)\Big]_{s=\frac{2(1-z^{-1})}{1(1+z^{-1})}} = \frac{4}{\left[\frac{2(1-z^{-1})}{(1+z^{-1})}\right]^2 + 2\sqrt{2}\left[\frac{2(1-z^{-1})}{(1+z^{-1})}\right]+4}$$

$$= \frac{1+2z^{-1}+z^{-2}}{3.4142135+0.5857865\,z^{-2}}$$

2. The difference equation is obtained from

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1+2z^{-1}+z^{-2}}{3.4142135+0.5857865\,z^{-2}}$$

Cross-multiply and take inverse z-transform to get

$$3.4142135\,y(n) + 0.5857865\,y(n-2) = x(n) + 2x(n-1) + x(n-2)$$

By rearranging and scaling $y(n)$ can be realized as

$$y(n) = 0.2928932\,[x(n) + 2\,x(n-1) + (n-2)] - 0.1715729\,y(n-2)$$

3. Frequency response

$$H(z)\Big]_{z=e^{j\omega}} = \frac{1+2e^{-j\omega}+e^{-j2\omega}}{3.4142135+0.5857865e^{-j2\omega}} = \frac{N(\omega)}{D(\omega)}$$

Numerator $= N(\omega) = 1+2e^{-j\omega}+e^{-j2\omega} = e^{-j\omega}(e^{j\omega}+2+e^{-j\omega}) = e^{-j\omega}(2+2\cos\omega)$

$= 2(1+\cos\omega)\,e^{-j\omega}$

Denominator $= D(\omega) = 3.4142135 + 0.5857865\,e^{-j2\omega} = A + B\,e^{-j2\omega}$

$= (A + B\cos 2\omega) - j\,B\sin 2\omega$

$$H(e^{j\omega}) = \frac{2(1+\cos\omega)}{\sqrt{(A+B\cos 2\omega)^2 + (B\sin 2\omega)^2}} e^{j\left[-\omega - \tan^{-1}\left(\frac{-B\sin 2\omega}{A+B\cos 2\omega}\right)\right]}$$

$$= \sqrt{(A+B\cos 2\omega)^2 + (B\sin 2\omega)^2}\, e^{j\tan^{-1}\left(\frac{-B\sin 2\omega}{A+B\cos 2\omega}\right)}$$

The magnitude of the frequency response is

$$|H(e^{j\omega})| = \frac{2(1+\cos\omega)}{\sqrt{(A+B\cos 2\omega)^2 + (B\sin 2\omega)^2}}$$

Plot $20\log_{10}|H(e^{j\omega})|$ vs ω for $\omega = 0$ *to* π.

Relationship between the *s*- and *z*-planes The bilinear transformation is

$$s = \frac{2(1-z^{-1})}{T(1+z^{-1})} \qquad \text{or} \qquad z = \frac{1+(sT/2)}{1-(sT/2)}$$

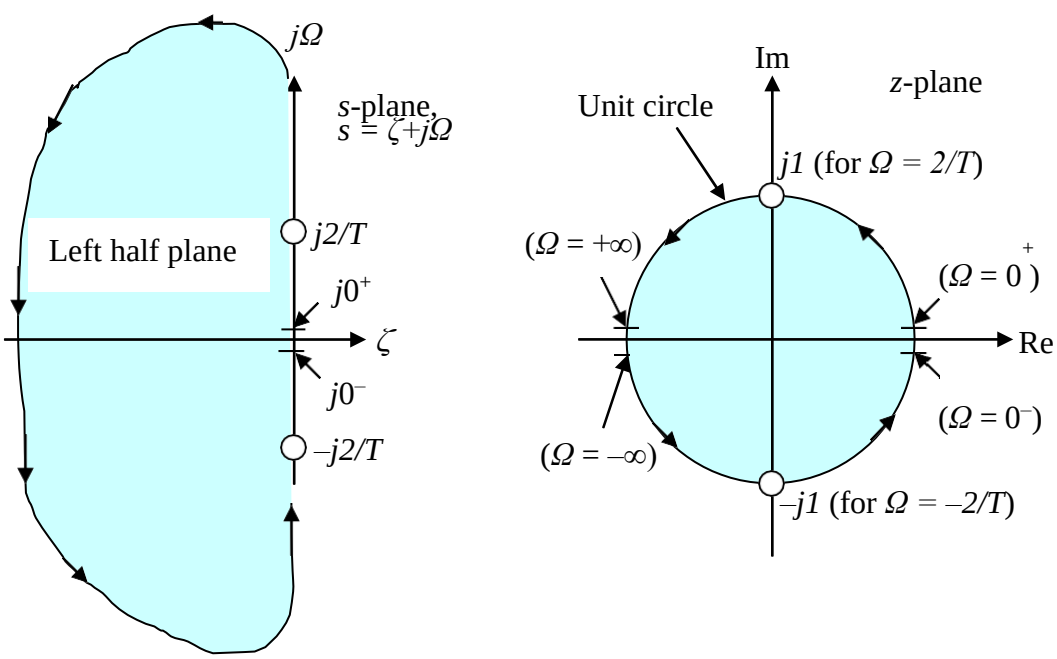
We can map a couple of points on the $j\Omega$ axis by setting $s = j\Omega$, so that

$$z = \frac{1+(j\Omega T/2)}{1-(j\Omega T/2)}$$

Thus $\Omega = 0$ maps to $z = 1$ and $\Omega = 2/T$ maps to $z = j1 = 1\, e^{j\pi/2}$ as shown in figure below. The transformation has the following properties:

1. The entire $j\Omega$ axis of the *s*-plane goes on to the unit circle of the *z*-plane.
2. The entire left half of the *s*-plane is mapped into the inside of the unit circle of the *z*-plane. (In contrast, in impulse invariant design the *poles* in the left halves of infinitely many strips of width Ω_s in the *s*-plane are mapped into the inside of the unit circle in the *z*-plane.)

So a stable analog filter, with all of its poles in the left half plane, would be transformed into a stable digital filter with all of its poles in the unit circle. The frequency response is evaluated on the $j\Omega$ axis in the *s*-plane and on the unit circle in the *z*-plane. While the frequency responses of the analog filter and digital filter have the same amplitudes there is a nonlinear relationship between corresponding digital and analog frequencies.



Nonlinear relationship of frequencies in bilinear transformation

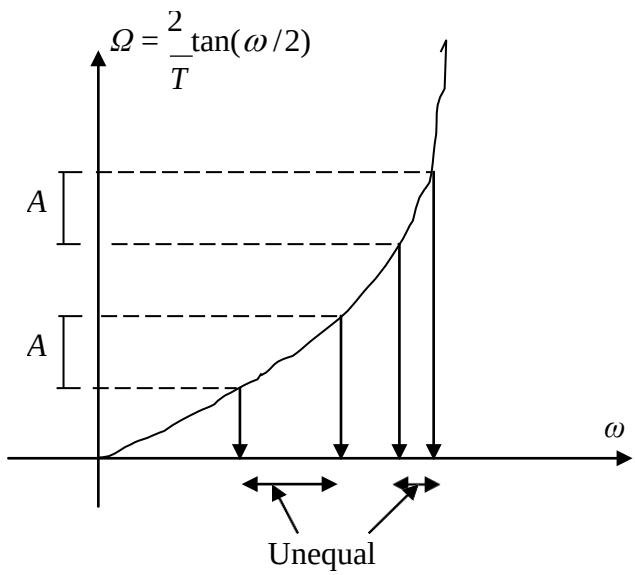
In the bilinear transformation the analog and digital frequencies are non-linearly related. Setting $s = j\Omega$ and $z = e^{j\omega}$ in $s = \frac{2}{T}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)$, we get

$$j\Omega = \frac{2}{T}\left(\frac{1-e^{-j\omega}}{1+e^{-j\omega}}\right) = \frac{2}{T}\,\frac{e^{-j\omega/2}(e^{j\omega/2}-e^{-j\omega/2})}{e^{-j\omega/2}(e^{j\omega/2}+e^{-j\omega/2})} = \frac{2}{T}\,\frac{j\left((e^{j\omega/2}-e^{-j\omega/2})/2j\right)}{\left((e^{j\omega/2}+e^{-j\omega/2})/2\right)}$$

or

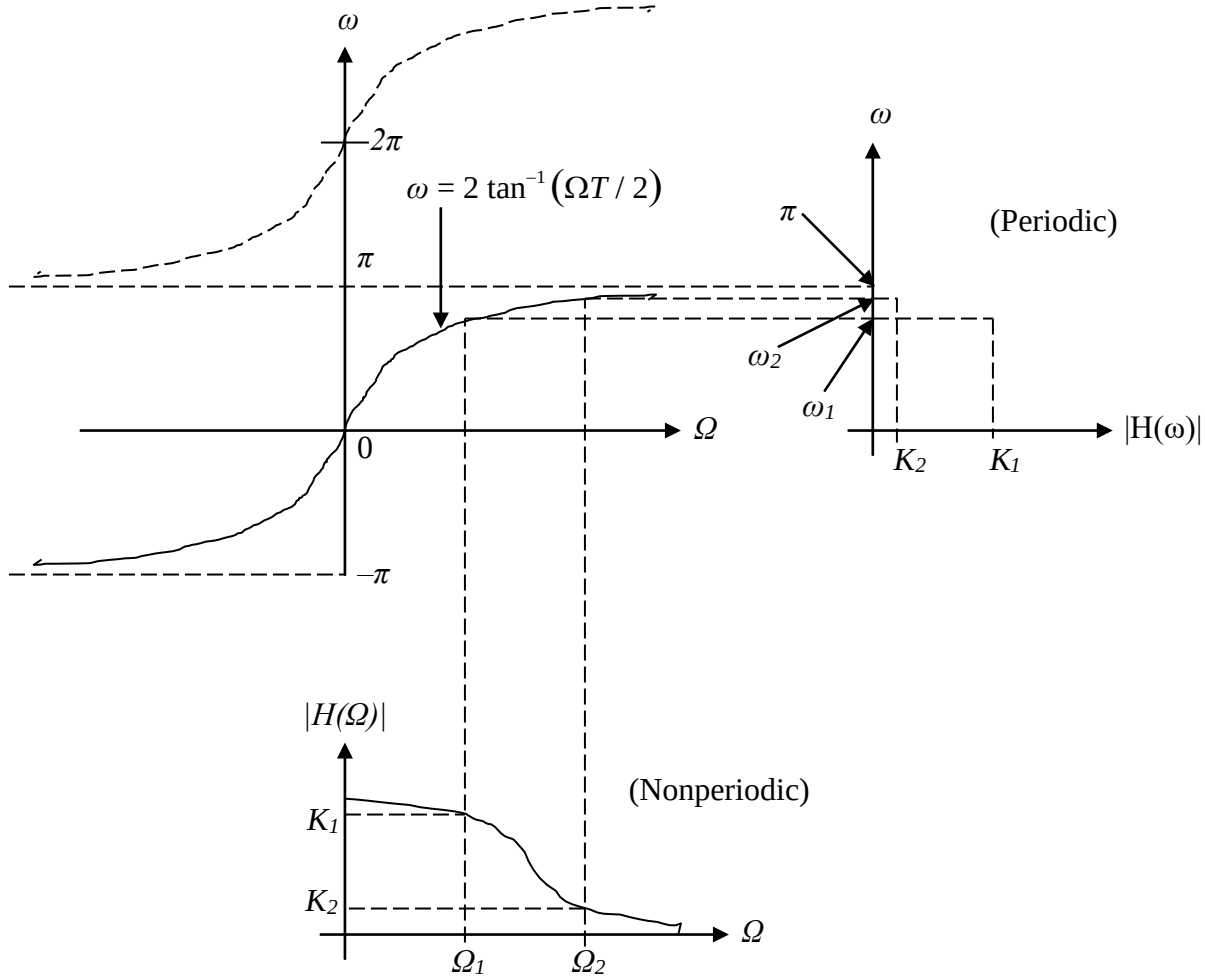
$$\Omega = \frac{2}{T}\,\frac{\sin\omega/2}{\cos\omega/2} = \frac{2}{T}\tan\left(\frac{\omega}{2}\right) \quad \text{or} \quad \omega = 2\tan^{-1}\left(\frac{\Omega T}{2}\right)$$

First we sketch Ω (the analog frequency) as a function of ω (the digital frequency) as given by $\Omega = \frac{2}{T}\tan\left(\frac{\omega}{2}\right)$ to show qualitatively the distortion of the frequency scale that occurs due to the nonlinear nature of the relationship.



Equally spaced pass bands A are pushed together or warped on the higher frequency end of the digital frequency scale. This effect is normally compensated for by pre-warping the analog filter before applying bilinear transformation.

Because of warping the relationship between Ω_1 and Ω_2 on the one hand and ω_1 and ω_2 on the other is not linear. The digital frequencies ω_1 and ω_2 are pushed in towards the origin ($\omega = 0$). In this process $\Omega = \infty$ is transformed to $\omega = \pi$.



If the bilinear transformation is applied to the system $H_a(s)$ with critical frequency Ω_c, the digital filter will have a critical frequency $\omega_c = 2\tan^{-1}(\Omega_c T/2)$. If the resulting $H(z)$ is used in an A/D–$H(z)$–D/A structure, the equivalent critical frequency (of the equivalent analog filter) is obtained by replacing ω_c with $\Omega_{ceq}T$:

$$\omega_c \Rightarrow \Omega_{ceq}T = 2\tan^{-1}\left(\frac{\Omega_c T}{2}\right) \qquad \text{or} \qquad \Omega_{ceq} = \frac{2}{T}\tan^{-1}\left(\frac{\Omega_c T}{2}\right)$$

If $(\Omega_c T/2)$ is so small that $\tan^{-1}(\Omega_c T/2) \approx \Omega_c T/2$, then we have

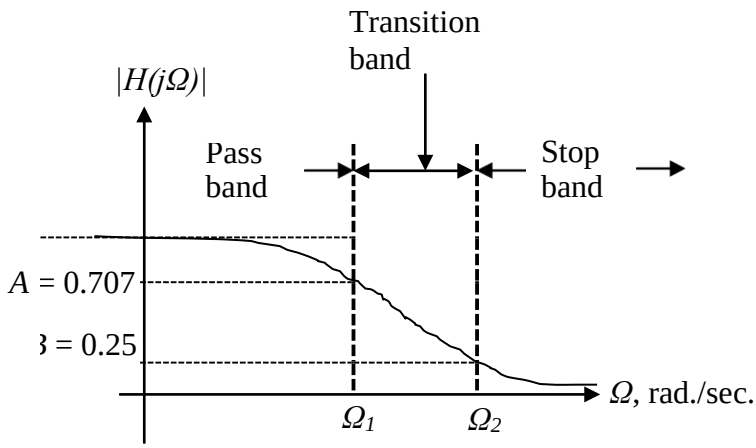
$$\Omega_{ceq} = \frac{2}{T}\left(\frac{\Omega_c T}{2}\right) = \Omega_c$$

If this condition is not satisfied, then the warping of the critical frequency (in the bilinear design) is compensated for by *pre-warping*.

Digital filter design – The Butterworth filter

Before taking up design we reproduce below some of the material relating filter specifications to the filter order and the cut-off frequency.

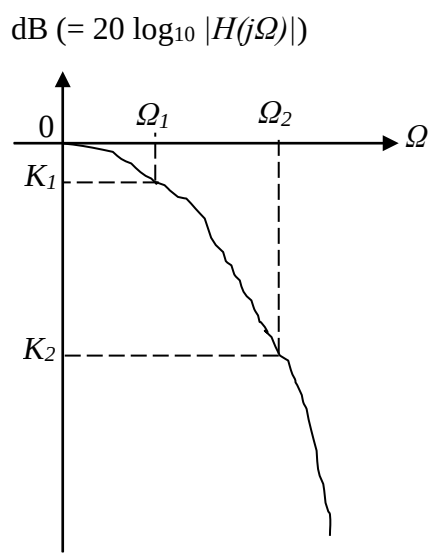
A typical *magnitude* response specification is sketched below. The magnitudes at the critical frequencies Ω_1 and Ω_2 are A and B, respectively. For illustrative purposes we have



arbitrarily taken $A = 0.707$ (thus Ω_1 is the cut-off frequency, but this need not be the case) and $B = 0.25$.

The *log-magnitude* specification is diagrammed below. Note that $(20 \log A) = K_1$ and $(20 \log B) = K_2$. Thus the analog filter specifications are

$$0 \geq 20\log_{10}|H(j\Omega)| \geq K_1 \text{ for all } \Omega \leq \Omega_1$$
$$20\log_{10}|H(j\Omega)| \leq K_2 \text{ for all } \Omega \geq \Omega_2$$



With the magnitude $|H(j\Omega)|$ given by the Butterworth function,

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

and using the equality condition at the critical frequencies in the above specifications the order, N, of the filter is given by

$$N = \left\lceil \frac{\log_{10}\left[\dfrac{10^{-K_1/10}-1}{10^{-K_2/10}-1}\right]}{2\log_{10}\left(\dfrac{\Omega_1}{\Omega_2}\right)} \right\rceil \quad \rightarrow \text{(3.16, Ludeman)}$$

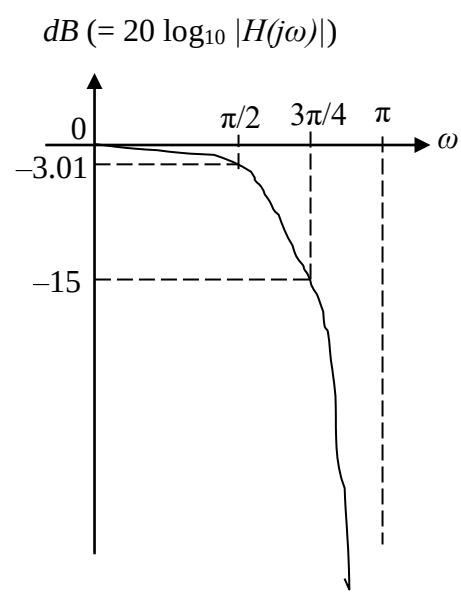
The result is rounded to the next larger integer. For example, if $N = 3.2$ by the above calculation then it is rounded up to 4, and the order of the required filter is $N = 4$. In such a case the resulting filter would exceed the specification at both Ω_1 and Ω_2. The cut-off frequency Ω_c is determined from one of the two equations below.

$$\Omega_c = \frac{\Omega_1}{\sqrt[2N]{10^{-K_1/10}-1}} \quad \text{or} \quad \Omega_c = \frac{\Omega_2}{\sqrt[2N]{10^{-K_2/10}-1}} \quad \rightarrow \text{(3.17 Ludeman)}$$

The equation on the left will result in the specification being met exactly at Ω_1 while the specification is exceeded at Ω_2. The equation on the right results in the specification being met exactly at Ω_2 and exceeded at Ω_1.

Example 3.6.1 Design and realize a digital low pass filter using the ***bilinear transformation*** method to satisfy the following characteristics:

(a) A monotonic pass band and stop band
(b) –3.01 dB cut off frequency of 0.5π rad.
(c) Magnitude down at least 15 dB at 0.75π rad.



Note that the given frequencies are digital frequencies. The required frequency response is shown. We use bilinear transformation on an analog prototype.

Step 1 Pre-warp the critical digital frequencies $\omega_1 = 0.5\pi$ and $\omega_2 = 0.75\pi$ using $T = 1$ sec. That is, we find the analog frequencies Ω'_1 and Ω'_2 that correspond to ω_1 and ω_2:

$$\Omega'_1 = \frac{2}{T}\tan\left(\frac{\omega_1}{2}\right) = 2\tan\frac{0.5\pi}{2} = 2.0 \text{ rad / sec}$$

$$\Omega'_2 = \frac{2}{T}\tan\left(\frac{\omega_2}{2}\right) = 2\tan\frac{0.75\pi}{2} = 4.8284 \text{ rad / sec}$$

Step 2 Design LP analog filter with critical frequencies Ω_1' and Ω_2' that satisfy

$$0 \geq 20 \log |H_a(j\,\Omega_1')| \geq -3.01 \text{ dB} = K_1, \text{ and}$$
$$20 \log |H_a(j\,\Omega'_2)| \leq -15 \text{ dB} = K_2$$

The Butterworth filter satisfies the monotonic property and has an order N and critical frequency Ω_c determined by Eq. 3.16 and 3.17 of Ludeman

$$N = \left\lceil \frac{\log_{10}\left|\left(\dfrac{10^{-K_1/10}-1}{10^{-K_2/10}-1}\right)\right|}{2\log_{10}\left|\left(\dfrac{\Omega_1}{\Omega_2}\right)\right|} \right\rceil \quad \text{and} \quad \Omega_c = \frac{\Omega_1}{\sqrt[2N]{10^{-K_1/10}-1}}$$

Plugging in numerical values,

$$N = \left\lceil \frac{\log_{10}\left|\left(\dfrac{10^{+3.01/10}-1}{10^{+15/10}-1}\right)\right|}{2\log_{10}\left|\left(\dfrac{2}{4.828}\right)\right|} \right\rceil = \left\lceil \frac{\log_{10}\left(\dfrac{2-1}{31.62-1}\right)}{2(-0.3827)} \right\rceil = \left\lceil \frac{-1.486}{2\ (-0.3827)} \right\rceil = \lceil 1.941 \rceil = 2$$

$$\Omega_c = \frac{2.0}{\sqrt[4]{10^{+3.01/10}-1}} = \frac{2}{\sqrt[4]{2-1}} = 2 \text{ rad / sec}$$

Note in this case that $\Omega_c = \Omega_1$.

Therefore, the required *pre-warped*, *normalized*, *unit bandwidth*, analog filter of order 2 using the Butterworth Table 3.1b (or the Butterworth circle) is

$$H_a(s) = \frac{1}{s^2 + \sqrt{2}\,s + 1} \text{(with a cut off frequency = 1 rad / sec)}$$

Since we need a cut-off frequency of $\Omega_c = 2$ rad/sec, we next use the low pass to low pass transformation $s \rightarrow s/2$ in order to move the cut-off frequency from 1 to 2 rad/sec.

$$H_a(s) = \left.\frac{1}{s^2 + \sqrt{2}\,s + 1}\right|_{s \to s/2} = \frac{1}{(s/2)^2 + \sqrt{2}(s/2) + 1}$$

$$= \frac{4}{s^2 + 2\sqrt{2}\,s + 4} \text{ (with a cut-off frequency = 2 rad/sec.)}$$

Step 3 Applying the bilinear transformation to $H_a(s)$ with $T = 1$ will transform the pre-warped analog filter into a digital filter with system function H(z) that will satisfy the given digital requirements:

$$H(z) = H_a(s)\Big]_{s \to \frac{2(1-z^{-1})}{1+z^{-1}}} = \frac{4}{\left[\dfrac{2(1-z^{-1})}{1+z^{-1}}\right]^2 + 2\sqrt{2}\left[\dfrac{2(1-z^{-1})}{1+z^{-1}}\right] + 4}$$

$$= \frac{1 + 2z^{-1} + z^{-2}}{3.414 + 0.585\,z^{-2}}$$

HW: Obtain the difference equation. Plot $|H(e^{j\omega})|$ and $\angle H(e^{j\omega})$ vs ω.

Bilinear transformation: Cancellation of sampling time in warping and pre-warping The digital specifications are the set of critical frequencies $\{\omega_1, \omega_2, \ldots, \omega_N\}$ and the corresponding set of magnitude requirements $\{K_1, K_2, \ldots, K_N\}$. When an analog filter is used as the prototype for

the bilinear transformation method the relationship between digital and analog frequencies is nonlinear and governed by

$$\Omega = \frac{2}{T}\tan\left(\frac{\omega}{2}\right) \qquad \text{and} \qquad \omega = 2\tan^{-1}\left(\frac{\Omega T}{2}\right)$$

Therefore, to get the proper digital frequency, we must design an analog filter with analog critical frequencies Ω_i: $i = 1, 2, \ldots, N$ given by

$$\Omega_i = \frac{2}{T}\tan\left(\frac{\omega_i}{2}\right), \qquad i = 1, 2, \ldots, N$$

This operation will be referred to as pre-warping. The corresponding analog magnitude requirements are not changed and remain the same as the corresponding digital requirements. An analog filter $H_a(s)$ is then designed to satisfy the pre-warped specifications given by $\Omega_1, \Omega_2, \ldots, \Omega_N$ and $K_1, K_2, \ldots, K_N$. The bilinear transformation is then applied to $H_a(s)$, i.e.,

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

As the T in the Ω_i equation and the T in the bilinear transform cancel in the procedure described above for low pass filter design, it is convenient to just use $T = 1$ in both places. This is easily seen since if the Ω_i comes from an analog-to-analog transformation of an $H_a(s)$ with a unit radian cut-off frequency, we have $s \to (s/\Omega_i)$, and when the bilinear transformation $s \to \frac{2(1-z^{-1})}{T(1+z^{-1})}$ is used the cascade of transformations is given by

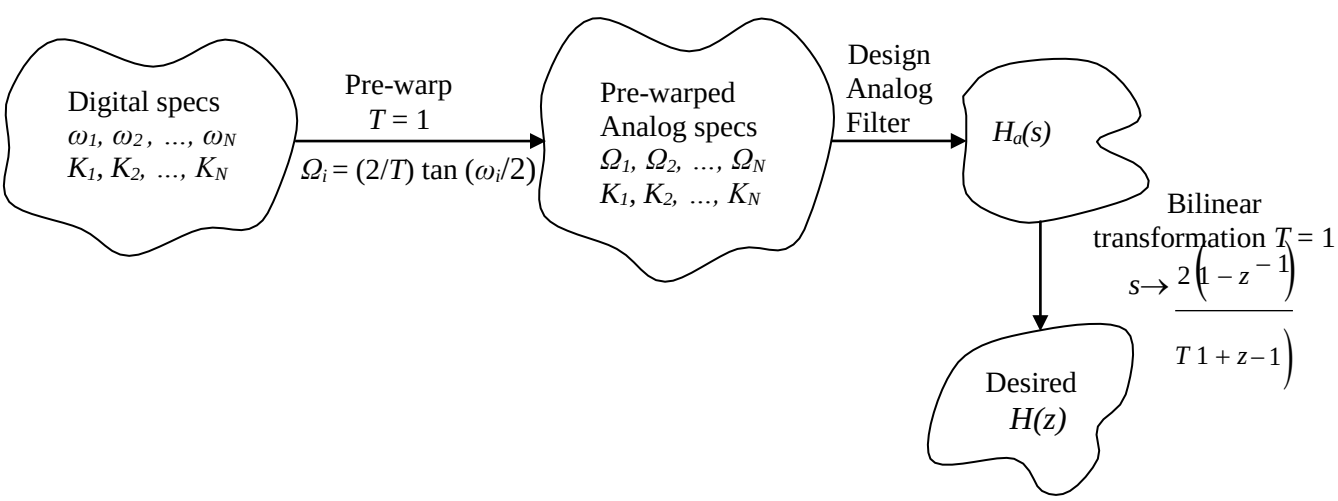
$$s \to \frac{2(1-z^{-1})}{T(1+z^{-1})\Omega_i} = \frac{2(1-z^{-1})}{T(1+z^{-1})\left[\frac{2}{T}\tan\frac{\omega_i}{2}\right]} = \frac{(1-z^{-1})}{(1+z^{-1})\tan\frac{\omega_i}{2}}$$

This does not contain a T. Thus it is immaterial what value of T is used as long as it is the same in both steps (which it is).

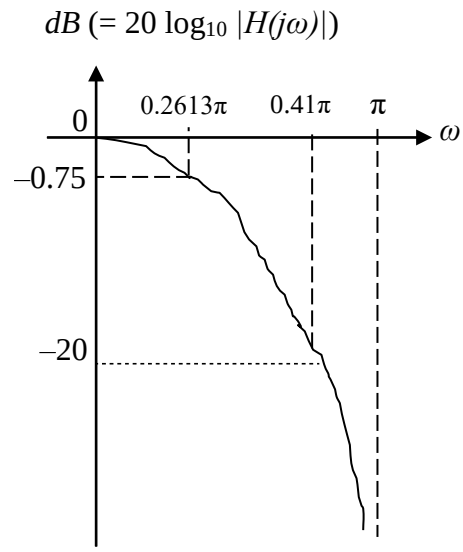
The procedure for the design of a digital filter using the bilinear transformation consists of:

Step 1: Pre-warping the digital specifications
Step 2: Designing an analog filter to meet the pre-warped specs
Step 3: Applying the bilinear transformation

In the process T is arbitrarily set to 1, but it can be set equal to any value (e.g., $T = 2$), since it cancels in the design. The design process is shown by the figure below.



Example 3.6.2 Design a digital low pass filter with pass band magnitude characteristic that is constant to within 0.75 dB for frequencies below $\omega = 0.2613\pi$ and stop band attenuation of at least 20 dB for frequencies between $\omega = 0.41\pi$ and π.



Use ***bilinear transformation***. Determine the transfer function $H(z)$ for the lowest order Butterworth design which meets these specifications. Draw the cascade form realization.

Step 1: Pre-warp $\omega_1 = 0.2613\pi$, $\omega_2 = 0.41\pi$ with $T = 1$ sec.

$$\Omega_1 = \frac{2}{T}\tan\frac{\omega_1}{2} = 2\tan\frac{0.2613\pi}{2} = 0.8703 \text{ rad / sec}$$

$$\Omega_2 = \frac{2}{T}\tan\frac{\omega_2}{2} = 2\tan\frac{0.41\pi}{2} = 1.501 \text{ rad / sec}$$

Step 2: Design $H_a(s)$

$$N = \left\lceil \frac{\log_{10}\left[\dfrac{10^{+0.75/10}-1}{10^{+20/10}-1}\right]}{2\log_{10}\left(\dfrac{0.8703}{1.501}\right)} \right\rceil = \left\lceil \frac{\log_{10}\left(\dfrac{1.19-1}{100-1}\right)}{2(-0.2967)} \right\rceil = \left\lceil \frac{-2.7203}{2(-0.2967)} \right\rceil = \lceil 5.75 \rceil = 6$$

$$\Omega_c = \frac{\Omega_1}{\sqrt[2N]{10^{-K_1/10}-1}} = \frac{0.8703}{\sqrt[12]{10^{+0.75/10}-1}} = \frac{0.8703}{0.8701} = 1 \text{ rad/sec}$$

Let the left-half plane poles be denoted s_1^*, s_1, s_2^*, s_2, s_3^*, and s_3.

$$H_a(s) = \frac{1}{(s-s_1^*)(s-s_1)}\frac{1}{(s-s_2^*)(s-s_2)}\frac{1}{(s-s_3^*)(s-s_3)}$$

Since $\Omega_c = 1$ the LP to LP transformation $s \rightarrow (s/1)$ results in the same $H_a(s)$ as given above.

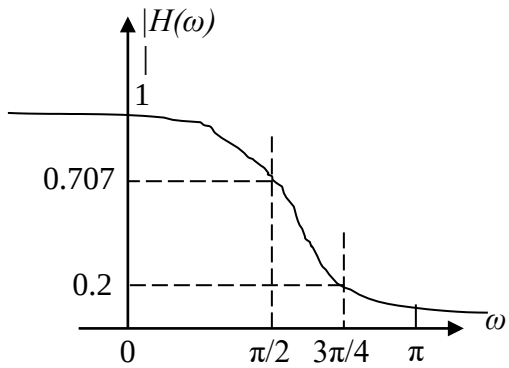
Step 3: $H(z) = H_a(s)\Big]_{s \rightarrow \frac{2(1-z^{-1})}{1+z^{-1}}}$

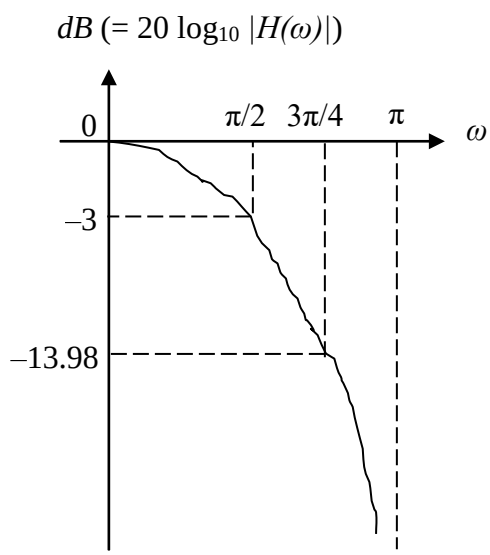
$$= \left.\frac{1}{(s-s_1)(s-s_1^*)}\right|_{s \to \frac{2(1-z^{-1})}{1+z^{-1}}} \left.\frac{1}{(s-s_2)(s-s_2^*)}\right|_{s \to \frac{2(1-z^{-1})}{1+z^{-1}}} \left.\frac{1}{(s-s_3)(s-s_3^*)}\right|_{s \to \frac{2(1-z^{-1})}{1+z^{-1}}}$$

Example 3.6.3 Determine *H(z)* for a Butterworth filter satisfying the following constraints. Use the ***impulse invariance*** technique.

$$\sqrt{0.5} \le |H(e^{j\omega})| \le 1, \text{ for } 0 \le \omega \le \pi/2$$

$$|H(e^{j\omega})| \le 0.2, \qquad \text{for } 3\pi/4 \le \omega \le \pi$$





Solution The critical frequencies are $\omega_1 = \pi/2$, $\omega_2 = 3\pi/4$. Use $\omega = \Omega T$ to determine the analog frequencies Ω_1 and Ω_2. Note T is not given. Take $T = 1$, so that $\omega = \Omega .1 = \Omega$. (This corresponds to the pre-warping step of the bilinear transformation with $\Omega = (2/T)\tan(\omega/2)$).

We have $\omega = \Omega .1$, so that the critical frequencies are

$K_1 = -3.01$ dB
$\Omega_1 = \omega_1 = \pi/2$ rad/sec. ($= \omega_c$)

$K_2 = -13.98$ dB
$\Omega_2 = \omega_2 = 3\pi/4$ rad/sec.

The order of the filter is given by

$$N = \left\lceil \frac{\log_{10}\left(\dfrac{10^{-K_1/10}-1}{10^{-K_2/10}-1}\right)}{2\log_{10}\left(\dfrac{\Omega_1}{\Omega_2}\right)} \right\rceil = \left\lceil \frac{\log_{10}\left(\dfrac{10^{+3.01/10}-1}{10^{-13.98/10}-1}\right)}{2\log_{10}\left(\dfrac{\pi/2}{3\pi/4}\right)} \right\rceil = \left\lceil \frac{\log_{10}\left(\dfrac{2-1}{25.003-1}\right)}{2\log_{10}\left(\dfrac{2}{3}\right)} \right\rceil$$

$$= \left\lceil \frac{-13.8}{2\,(-0.176)} \right\rceil = \lceil 3.919 \rceil = 4$$

Ω_c is already known to be $\pi/2$ rad/sec.

The 4th order normalized Butterworth filter (with unit bandwidth) is

$$H_a(s) \text{ (normalized)} = \frac{1}{s^4 + 2.613\,s^3 + 3.414\,s^2 + 2.613\,s + 1}$$

Using the low pass to low pass analog transformation $s \to (s/\Omega_c)$ or $s \to (s/1.57)$ we get the Butterworth filter satisfying the required specs:

$$H_a(s) = \frac{1}{\left(\dfrac{s}{1.57}\right)^4 + 2.613\left(\dfrac{s}{1.57}\right)^3 + 3.414\left(\dfrac{s}{1.57}\right)^2 + 2.613\left(\dfrac{s}{1.57}\right) + 1}$$

For the impulse invariant design we need the poles of $H_a(s)$, so the above form is not much help. We need to use the factored form:

$$H_a(s) \text{ (normalized)} = \frac{1}{(s^2 + 0.76536\,s + 1)(s^2 + 1.84776\,s + 1)}$$

And with $s \to (s/1.57)$, we have

$$H_a(s) = \left[\frac{1}{\left(\dfrac{s}{1.57}\right)^2 + 0.76536\left(\dfrac{s}{1.57}\right) + 1}\right]\left[\frac{1}{\left(\dfrac{s}{1.57}\right)^2 + 1.84776\left(\dfrac{s}{1.57}\right) + 1}\right]$$

Put this into partial fraction form. We need the poles individually since we need to use the relation

$$(s = s_1) \to (z = e^{s_1 T}), \text{ with } T = 1 \text{ of course}$$

to get $H(z)$ from the $H_a(s)$. Once we get the $H_a(s)$ we can then combine complex conjugate pole pairs to biquadratic form and then implement as a parallel form with two biquadratics in it. Alternatively, memorize relations 3(c) and 3(d). This latter gives the biquadratics directly.

This same problem will next be solved using the ***bilinear transformation*** to show the difference.

Step 1: Pre-warping according to $\Omega = \dfrac{2}{T}\tan\dfrac{\omega}{2}$ with $T = 1$ gives

$$\Omega_1 = 2\tan\frac{\omega_1}{2} = 2\tan\frac{(\pi/2)}{2} = 2 \text{ rad / sec}$$

$$\Omega_2 = 2\tan\frac{\omega_2}{2} = 2\tan\frac{(3\pi/4)}{2} = 4.828 \text{ rad / sec}$$

Step 2: Design $H_a(s)$

$$N = \frac{\left\lceil \log_{10}\left(\frac{10^{+19.980}-1}{10^{-1}-1}\right) \right\rceil}{2\log_{10}\left(\frac{2}{4.828}\right)} = ?$$

$$\Omega_c = \frac{\Omega_1}{\sqrt[2N]{10^{-K_1/10}-1}} = ?$$

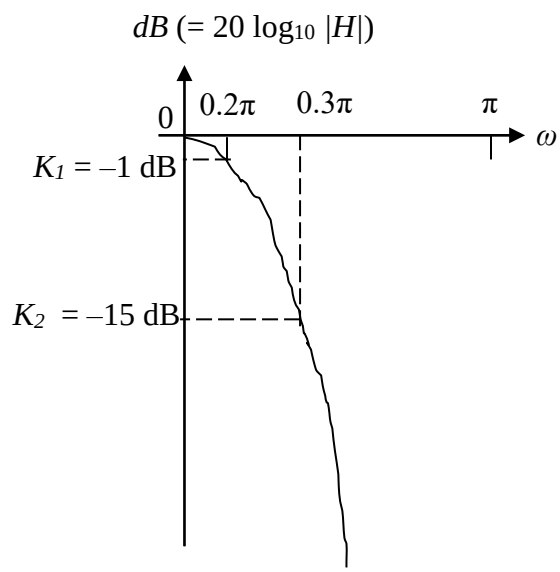
Step 3: $H(z) = H_a(s)\Big]_{s \to \frac{2(1-z^{-1})}{1+z^{-1}}}$

Example 3.6.4 Design a low pass digital filter by applying ***impulse invariance*** to an appropriate Butterworth continuous-time filter. The digital filter specs are:

$$-1 \text{ dB} \le 20 \text{ Log } |H(\omega)| \le 0, \qquad 0 \le |\omega| \le 0.2\pi$$
$$20 \text{ Log } |H(\omega)| \le -15 \text{ dB}, \qquad 0.3\pi \le |\omega| \le \pi$$

Solution First convert the digital frequencies ω to analog frequencies Ω. The mapping between ω and Ω is linear in the absence of aliasing. We shall use $\omega = \Omega T$ with $T = 1$. Thus the specs become

$$-1 \text{ dB} \le 20 \text{ Log } |H_a(\Omega)| \le 0, \qquad 0 \le |\Omega| \le 0.2\pi$$
$$20 \text{ Log } |H_a(\Omega)| \le -15 \text{ dB}, \qquad 0.3\pi \le |\Omega| \le \pi$$



$$N = \frac{\left\lceil \log_{10}\left(\frac{10^{-K_1/10}-1}{10^{-K_2/10}-1}\right) \right\rceil}{2\log_{10}\left(\frac{\Omega_1}{\Omega_2}\right)} = \frac{\left\lceil \log_{10}\left(\frac{10^{-(-1)/10}-1}{10^{-(-15)/10}-1}\right) \right\rceil}{2\log_{10}\left(\frac{0.2\pi}{0.3\pi}\right)} = \frac{\left\lceil \log_{10}\left(\frac{1.259-1}{31.6228-1}\right) \right\rceil}{2\log_{10}(0.6666)}$$

$$= \left\lceil \frac{\log_{10}\left(\frac{0.2589}{30.6228}\right)}{2(-0.1761)} \right\rceil = \left\lceil \frac{\log_{10}(0.0085)}{2(-0.1761)} \right\rceil = \left\lceil \frac{-2.0729}{2(-0.3827)} \right\rceil = \lceil 5.885 \rceil = 6$$

$$\Omega_c = \frac{\Omega_1}{\sqrt[2N]{10^{-K_1/10}-1}} = \frac{0.2\pi}{\sqrt[12]{1.2589-1}} = \frac{0.2\pi}{\sqrt[12]{0.2589}} = \frac{0.2\pi}{0.8935} = 0.703 \text{ rad/sec}$$

Let the left-half plane poles be denoted s_1, s_1^*, s_2, s_2^*, s_3, and s_3^*.

$$H_a(s) = \frac{1}{(s-s_1)(s-s_1^*)(s-s_2)(s-s_2^*)(s-s_3)(s-s_3^*)}$$

Since $\Omega_c = 0.703$ the LP to LP transformation s → (s/0.703) results in

$$H_a(s) = \left.\frac{1}{(s-s_1)(s-s_1^*)(s-s_2)(s-s_2^*)(s-s_3)(s-s_3^*)}\right|_{s \to \frac{s}{0.703}}$$

To be completed

Example 3.6.5 [2003] [The Butterworth circle and the bilinear transformation] Refer to Oppenheim & Schafer, Sec. 5.1.3 and Sec. 5.2.1. The bilinear transformation is given by

$$s = \frac{2(1-z^{-1})}{T(1+z^{-1})} \qquad \text{or} \qquad z = \frac{1+(sT/2)}{1-(sT/2)}$$

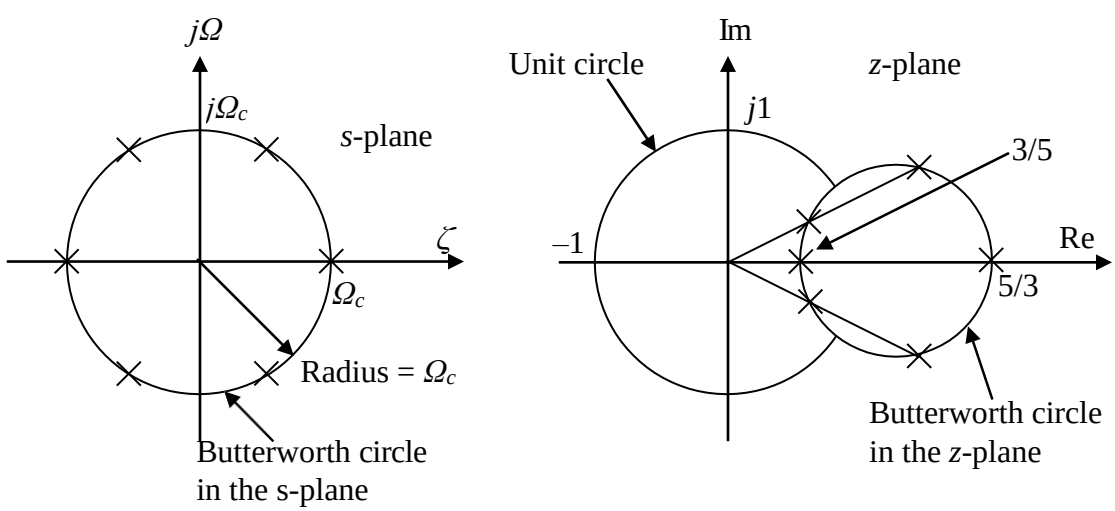
This last equation is used to map the poles on the Butterworth circle in the s-plane into poles on the Butterworth circle in the *z*-plane. For the normalized Butterworth filter with a cut-off frequency of 1 rad/sec., the Butterworth circle in the *s*-plane has unit radius. If the cut-off frequency is Ω_c instead of 1, then the circle has a radius of Ω_c. This is the case in the example on pp. 212-214 of Oppenheim & Schafer where the order *N* of the filter is 3, the radius of the Butterworth circle in the *s*-plane is Ω_c, and $\Omega_c T = ½$ which corresponds to a sampling frequency of twice the cut-off frequency (Figure 5.14).

For the two poles at $s = -\Omega_c$ and $s = \Omega_c$ and $\Omega_c T = ½$ we get

$$s = -\Omega_c: \qquad z = \frac{1-(\Omega_c T/2)}{1+(\Omega_c T/2)} = \frac{1-(1/4)}{1+(1/4)} = 3/5$$

$$s = \Omega_c: \qquad z = \frac{1+(\Omega_c T/2)}{1-(\Omega_c T/2)} = \frac{1+(1/4)}{1-(1/4)} = 5/3$$

Both of these *z*-plane poles are on the real axis as shown in figure below.



The other s-plane poles are similarly mapped to z-plane poles, though the algebra involved is a little more. Note that the three poles in the left-half of the s-plane are mapped into the inside of the unit circle in the z-plane.

5.7 Analog design using digital filters

When we are required to simulate an analog filter using the *A/D – H(z) – D/A* structure, the specifications consist of the analog frequencies$\{\Omega_1, \Omega_2, \ldots, \Omega_N\}$, the corresponding magnitudes $\{K_1, K_2, \ldots, K_N\}$ and the sampling time T. We convert to digital specs using the relation $\omega_i = \Omega_i T$.

Example 5.7.1 [Bilinear] [4.2, p. 180, Ludeman] Design a digital filter $H(z)$ that when used in an *A/D-H(z)-D/A* structure gives an equivalent low-pass analog filter with (a) –3.01 dB cut-off frequency of 500Hz, (b) monotonic stop and pass bands, (c) magnitude of frequency response down at least 15 dB at 750 Hz, and (d) sample rate of 2000 samples/sec.
Solution
Step 0 First we convert the analog Ω's to digital ω's using $\omega_i = \Omega_i T$.

$$\Omega_1 = 2\pi F_1 = 2\pi\, 500 = 1000\,\pi \text{ rad/sec.}, \qquad K_1 = -3.01 \text{ dB}$$
$$\Omega_2 = 2\pi F_2 = 2\pi\, 750 = 1500\,\pi \text{ rad/sec.}, \qquad K_2 = -15 \text{ dB}$$

Thus

$$\omega_1 = \Omega_1 T = 1000\,\pi\,\frac{1}{2000} = 0.5\,\pi \text{ rad}, \qquad K_1 = -3.01 \text{ dB}$$
$$\omega_2 = \Omega_2 T = 1500\,\pi\,\frac{1}{2000} = 0.75\,\pi \text{ rad}, \qquad K_2 = -15 \text{ dB}$$

Step 1 Pre-warping. Use $T = 1$. (In Steps 1 and 3 we could have used $T = 1/2000$ but the two occurrences of T would cancel out).

$$\Omega'_1 = \frac{2}{T}\tan\left(\frac{\omega_1}{2}\right) = 2.0 \text{ rad/sec.} \qquad \text{and} \qquad \Omega'_2 = \frac{2}{T}\tan\left(\frac{\omega_2}{2}\right) = 4.828 \text{ rad/sec.}$$

Step 2 Design $H_a(s)$, i.e., determine the low pass Butterworth filter (see earlier example).

$$N = \left\lceil \frac{\log_{10}\left(\dfrac{10^{+3.01/10}-1}{10^{+15/10}-1}\right)}{2\log_{10}\left(\dfrac{2}{4.828}\right)} \right\rceil = \left\lceil \frac{\log_{10}\left(\dfrac{2-1}{31.62-1}\right)}{2(-0.3827)} \right\rceil = \left\lceil \frac{-1.486}{2\ (-0.3827)} \right\rceil = \lceil 1.941 \rceil = 2$$

$$\Omega_c = \frac{2.0}{\sqrt[4]{10^{+3.01/10}-1}} = \frac{2}{\sqrt[4]{2-1}} = 2 \text{ rad / sec}$$

Do analog low pass to low pass transformation $s \to (s/\Omega_c)$, i.e., $s \to (s/2)$ in order to move the cut-off frequency from 1 to 2 rad/sec. This gives the $H_a(s)$ with pre-warped specs and $\Omega_c = 2$ rad/sec.

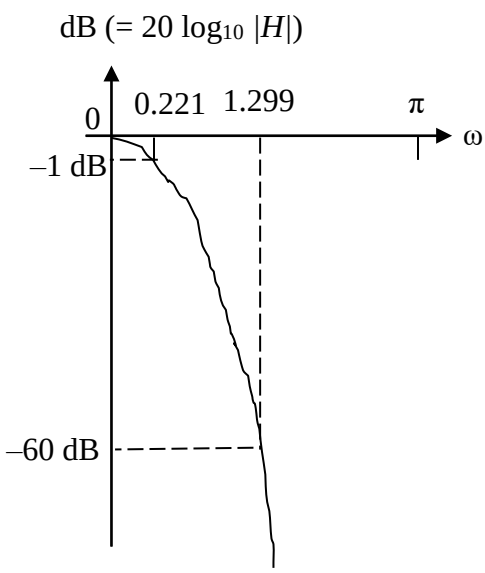
$$H_a(s) = \left.\frac{1}{s^2+\sqrt{2}\,s+1}\right|_{s \to s/2} = \frac{1}{(s/2)^2+\sqrt{2}(s/2)+1}$$
$$= \frac{4}{s^2+2\sqrt{2}\,s+4} \quad \text{(with a cut-off frequency = 2 rad/sec.)}$$

Step 3 Applying the bilinear transformation $s \to \frac{2}{T}\frac{(1-z^{-1})}{(1+z^{-1})}$ to $H_a(s)$ with $T = 1$ will transform the pre-warped analog filter into a digital filter with system function $H(z)$ that will satisfy the given requirements:

$$H(z) = H_a(s)\Big|_{s \to \frac{2(1-z^{-1})}{(1+z^{-1})}} = \frac{4}{\left[\frac{2(1-z^{-1})}{1+z^{-1}}\right]^2 + 2\sqrt{2}\left[\frac{2(1-z^{-1})}{1+z^{-1}}\right] + 4} = \frac{1+2z^{-1}+z^{-2}}{3.414+0.585z^{-2}}$$

Example 3.7.2 [2002] Derive the Butterworth digital filter having the following specs:

Pass band: 0 to 4411 rad/sec.
Maximum ripple in pass band: 1 dB
Stop band: beyond 25975 rad/sec.
Minimum attenuation in stop band: 60 dB
Sampling frequency: 20 kHz



Solution Even though pass band *ripple* may suggest a Chebyshev filter we shall comply with the request for a Butterworth filter. We use ***bilinear transformation***.

Analog frequency specs are given. Convert them to digital by using the relation $\omega = \Omega T$ and $T = (1/20000)$ sec.

$\Omega_1 = 4411$ rad/sec. becomes $\omega_1 = \Omega_1 T = 4411/20000 = 0.221$ rad
$\Omega_2 = 25975$ rad/sec. becomes $\omega_2 = \Omega_2 T = 25975/20000 = 1.299$ rad

Now, starting from these digital specs the design proceeds in 3 steps as usual.

Step 1 Pre-warp the critical digital frequencies ω_1 and ω_2 using $T = 1$ sec., to get

$$\Omega'_1 = \frac{2}{T}\tan\left(\frac{\omega_1}{2}\right) = 2\tan\frac{0.221}{2} = 2\tan 0.11 = 0.221 \text{ rad/sec.}$$

$$\Omega'_2 = \frac{2}{T}\tan\left(\frac{\omega_2}{2}\right) = 2\tan\frac{1.299}{2} = 2\tan 0.650 = 1.519 \text{ rad/sec.}$$

Step 2 Design an analog low pass filter with critical frequencies Ω_1' and Ω_2' to satisfy

$$0 \geq 20 \log |H_a(j\,\Omega_1')| \geq -1 \text{ dB} = K_1, \text{ and}$$
$$20 \log |H_a(j\,\Omega_2')| \leq -60 \text{ dB} = K_2$$

The Butterworth filter of order N and cut-off frequency Ω_c is given by equations (3.16) and (3.17) of Ludeman:

$$N = \frac{\log_{10}\left(\dfrac{10^{-K_1/10}-1}{10^{-K_2/10}-1}\right)}{2\log_{10}\left(\dfrac{\Omega_1}{\Omega_2}\right)} = \frac{\log_{10}\left(\dfrac{10^{+1/10}-1}{10^{+60/10}-1}\right)}{2\log_{10}\left(\dfrac{0.221}{1.519}\right)} = \frac{\log_{10}\left(\dfrac{1.259-1}{10^{6}-1}\right)}{2\log_{10}(0.145)}$$
$$= \frac{\log_{10}(2.59 \cdot 10^{-7})}{2(-0.837)} = \frac{-6.587}{-1.674} = 3.934 = 4$$

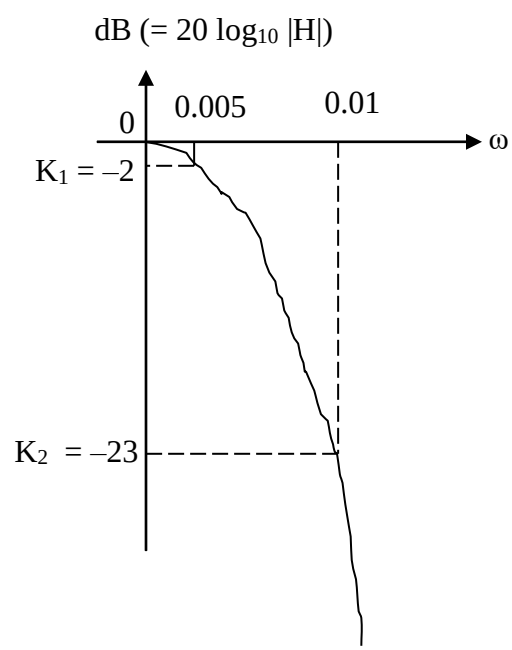
$$\Omega_c = \frac{\Omega_1}{\sqrt[2N]{10^{-K_1/10}-1}} = \frac{0.221}{\sqrt[8]{10^{+1/10}-1}} = \frac{0.221}{\sqrt[8]{1.259-1}} = \frac{0.221}{0.845} = 0.262 \text{ rad/sec.}$$

Therefore the required pre-warped Butterworth (analog) filter using Table 3.1b (Ludeman) and the analog low-pass to low pass transformation from Table 3.2, $s \rightarrow (s/\Omega_c)$, that is, $s \rightarrow (s/0.262)$, is

$$H_a(s) = \left.\frac{1}{s^4 + 2.613\,s^3 + 3.414\,s^2 + 2.613\,s + 1}\right|_{s \rightarrow (s/0.262)}$$

Example 3.7.3 Design a digital LPF using ***bilinear transformation*** with the following specifications, and a Butterworth approximation:

– 2 dB at 5 rad/sec.,
– 23 dB at 10 rad/sec.,
Sampling frequency = 1000 per sec.

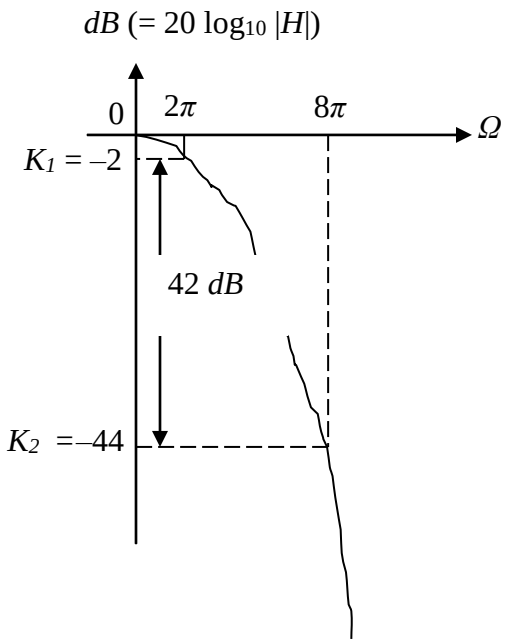


Solution Convert the analog specs to digital using $\omega = \Omega T$, with T = 1/1000 sec. The critical frequencies are

$$\omega_1 = \Omega_1 T = 5/1000 = 0.005 \text{ rad}; \qquad K_1 = -2 \text{ dB}$$
$$\omega_2 = \Omega_2 T = 10/1000 = 0.01 \text{rad}; \qquad K_2 = -23 \text{ dB}$$

Now apply Steps 1, 2 and 3 of the bilinear transformation design process.

Example 5.7.4 [2003] Design a digital filter that will pass a 1 Hz signal with attenuation less than 2 dB and suppress 4 Hz signal down to at least 42 dB from the magnitude of the 1 Hz signal.



Solution All the specs are given in the analog domain. The sampling period T is not specified. Since 1 Hz is in the pass band and 4 Hz in the stop band we shall use some multiple of 4 Hz, say, 20 Hz as the sampling frequency. Thus T = 1/20. We shall employ the ***impulse invariance*** method.

Step 0 Convert the analog specs to digital by using $\omega = \Omega T = 2\pi FT$. Thus

$$\Omega_1 = 2\pi .1 = 2\pi \text{ rad/sec., and}$$
$$\Omega_2 = 2\pi . 4 = 8\pi \text{ rad/sec.}$$

so that $\omega_1 = 2\pi T = 2\pi (1/20) = 0.1\ \pi$ rad., and $\omega_2 = 8\pi T = 8\pi (1/20) = 0.4\ \pi$ rad.

Step 1 Convert the digital frequencies ω_1 and ω_2 back to analog frequencies. Since we are using impulse invariance this involves using the same formula $\omega = \Omega T$ and we get the same analog frequencies as before, viz., $\Omega_1 = 2\pi$ rad/sec., and $\Omega_2 = 8\pi$ rad/sec. (Note that the value of T is irrelevant up to this point. We could have used a value of T = 1 in Steps 0 and 1, resulting in awkward values for the ω's, like 2π and 8π when we expect values between 0 and π; but this is not a problem for the design).

Step 2 Determine the order of the analog Butterworth filter.

$$N = \left\lceil \frac{\log_{10}\left(\frac{10^{-K_1/10}-1}{10^{-K_2/10}-1}\right)}{2\log_{10}\left(\frac{\Omega_1}{\Omega_2}\right)} \right\rceil = \left\lceil \frac{\log_{10}\left(\frac{10^{+2/10}-1}{10^{-44/10}-1}\right)}{2\log_{10}\left(\frac{2\pi}{8\pi}\right)} \right\rceil = \left\lceil \frac{\log_{10}\left(\frac{1.585-1}{25118.8-1}\right)}{2(-0.602)} \right\rceil$$

$$= \left\lceil \frac{-4.633}{-1.204} \right\rceil = \lceil 3.848 \rceil = 4$$

Cut–off frequency Ω_c is determined next:

$$\Omega_c = \frac{\Omega_1}{\sqrt[2N]{10^{-K_1/10}-1}} = \frac{2\pi}{\sqrt[8]{10^{+2/10}-1}} = \frac{2\pi}{\sqrt[8]{1.585-1}} = \frac{2\pi}{0.93515} = 6.719$$

Step 3 The normalized filter $H_a(s)$ of order 4 is

$$H_a(s) = \frac{1}{(s^2 + 1.848s + 1)\,(s^2 + 0.765s + 1)}$$

We may break it down into partial fractions now (before making the transformation $s \to (s/\Omega_c)$.

$$H_a(s) = \frac{As + B}{s^2 + 1.848s + 1} + \frac{Cs + D}{s^2 + 0.765s + 1}$$

Determine A, B, C, and D and then substitute $s \to (s/\Omega_c)$.

Alternatively, we can get the individual pole locations from the Butterworth circle.

$$H_a(s) = \frac{1}{(s + 0.924 + j0.383)(s + 0.924 - j0.383)(s + 0.383 + j0.924)(s + 0.383 - j0.924)}$$

$$= \frac{A}{s - s_1} + \frac{A^*}{s - s_1^*} + \frac{C}{s - s_2} + \frac{C^*}{s - s_2^*}$$

Determine A, A^*, C and C^*. Next find $H_a(s)\big|_{s \to (s/\Omega_c)}$ which is the analog prototype. From this we can find the $H(z)$ by mapping the s-plane poles to z–plane poles by the relation: $(s = s_1) \to (z = e^{s_1 T})$. Here at last we must specify T; we could use $T = 1/20$. In general, the smaller the value of T the better.

$$H_a(z) = A\frac{z}{z - e^{s_1 T}} + A^*\frac{z}{z - e^{s_1^* T}} + C\frac{z}{z - e^{s_2 T}} + C^*\frac{z}{z - e^{s_2^* T}}$$

If we were to use the ***bilinear transformation*** use some value of T like 1/20 (justifying it on the basis of the sampling theorem) in $\omega = \Omega T$ in Step 0.

Example 5.7.5 [Low pass filter] Design a digital low pass filter to approximate the following transfer function:

$$H_a(s) = \frac{1}{s^2 + \sqrt{2}\,s + 1}$$

Using the ***BZT (Bilinear z-transform)*** method obtain the transfer function, $H(z)$, of the digital filter, assuming a 3 dB cut-off frequency of 150 Hz and a sampling frequency of 1.28 kHz.

Solution This problem is a slight variation from the pattern we have followed so far in that it specifies *one* critical frequency (cut-off frequency) and the filter order instead of *two* critical frequencies with the filter order unknown.

Step 0 Convert the analog specs to digital

$$\Omega_c = 2\pi F_c = 2\pi\,150 = 300\,\pi \text{ rad/sec.}$$

The corresponding digital frequency is

$$\omega_c = \Omega_c T = 300\,\pi\,\frac{1}{1280} = 0.2343\,\pi \text{ rad.}$$

Step 1 Pre-warp (with $T = 1$)

$$\Omega'_c = \frac{2}{T}\tan\left(\frac{\omega_c}{2}\right) = 0.7715$$

Step 2 $H_a(s)$ is given to be of order 2. We only need to do the analog low pass to low pass transformation.

$$H_a(s)\Big|_{s\to\frac{s}{0.7715}} = \frac{1}{\left(\frac{s}{0.7715}\right)^2 + \sqrt{2}\left(\frac{s}{0.7715}\right) + 1}$$

Step 3 Applying the BZT (with $T = 1$)

$$H(z) = H_a(s)\Big|_{s\to\frac{2(1-z^{-1})}{T(1+z^{-1})}} = \frac{1}{\left[\frac{2(1-z^{-1})}{(1+z^{-1})\,0.7715}\right]^2 + \sqrt{2}\left[\frac{2(1-z^{-1})}{(1+z^{-1})\,0.7715}\right] + 1}$$

$$= \frac{0.0878\,(1 + 2z^{-1} + z^{-2})}{1 - 1.0048\,z^{-1} + 0.3561\,z^{-2}}$$

Example 3.7.6 [Low pass filter] Design a low pass digital filter derived from a second order Butterworth analog filter with a 3 dB cut-off frequency of 50 Hz. The sampling rate of the system is 500 Hz.

Solution (This is similar to Example 10). The second order Butterworth analog filter is

$$H_a(s) = \frac{1}{s^2 + \sqrt{2}\,s + 1}$$

Step 0 Convert the analog specs to digital

$$\Omega_c = 2\pi F_c = 2\pi\,50 = 100\,\pi \text{ rad/sec.}$$

The corresponding digital frequency is

$$\omega_c = \Omega_c T = 100\,\pi\,\frac{1}{500} = 0.2\pi \text{ rad.}$$

Step 1 Pre-warp (with $T = 1$)

$$\Omega'_c = \frac{2}{T}\tan\left(\frac{\omega_c}{2}\right) = 2\tan\left(\frac{0.2\pi}{2}\right) = 0.6498$$

Step 2 $H_a(s)$ is given to be of order 2. We only need to do the analog low pass to low pass transformation.

$$= \frac{1}{\left(\frac{s}{0.6498}\right)^2 + \sqrt{2}\left(\frac{s}{0.6498}\right) + 1} = \frac{0.42229}{s^2 + 0.91901s + 0.42229}$$

Step 3 Applying the BZT (with $T = 1$)

$$H(z) = H_a(s)\Big|_{s\to\frac{2(1-z^{-1})}{T(1+z^{-1})}} = \frac{0.42229}{\left[\frac{2(1-z^{-1})}{(1+z^{-1})}\right]^2 + 0.91901\left[\frac{2(1-z^{-1})}{(1+z^{-1})}\right] + 0.42229}$$

$$= \frac{0.42229\left(1+z^{-1}\right)^2}{4\left(1-z^{-1}\right)^2 + (0.91901)(2)\left(1-z^{-1}\right)\left(1+z^{-1}\right) + 0.42229\left(1+z^{-1}\right)^2}$$

The denominator is

$$\text{Dr} = 4\left(1-2z^{-1}+z^{-2}\right) + 1.83802\left(1-z^{-2}\right) + 0.42229\left(1+2z^{-1}+z^{-2}\right)$$

$$\text{Dr} = (4 + 1.83802 + 0.42229) + z^{-1}(-8 + 0.84458) + z^{-2}(4 - 1.83802 + 0.42229)$$

$$\text{Dr} = 6.26031 - 7.15542\, z^{-1} + 2.58427\, z^{-2}$$

$$H(z) = \frac{0.42229(1+2z^{-1}+z^{-2})}{6.26031 - 7.15542\, z^{-1} + 2.58427\, z^{-2}}$$

$$= \frac{0.06746(1+2z^{-1}+z^{-2})}{6.26031 - 1.14298\, z^{-1} + 0.4128\, z^{-2}}$$

Summary of IIR filter design

Bilinear	**Bilinear**
Digital specs given: ω_1, ω_2 (rad.) K_1, K_2 (*dB*) (1) Pre-warp $\Omega' = \frac{2}{T}\tan\left(\frac{\omega}{2}\right)$, T = 1 sec. Calculate Ω'_1, Ω'_2 and K_1, K_2 *(2)* Find filter order N, cut-off frequency Ω_c Find normalized filter $H_a(s)$ Find $H_a(s)\big\|_{s \to (s/\Omega_c)}$ (3) $H(z) = H_a(s)\big\|_{s \to \left(\frac{2(1-z^{-1})}{T(1+z^{-1})}\right)}$ with T = 1 sec.	Analog specs given: Ω_1, Ω_2 (rad./sec.), K_1, K_2 (dB), and T Step 0. (Preparation) Use $\omega = \Omega T$ to convert Ω to ω Get ω_1, ω_2, and K_1, K_2 Then (1) (2) As in the left side column (3)
Impulse invariance Digital specs given: ω_1, ω_2, K_1, K_2, and T (or T = 1) *(1)* Determine Ω_1, Ω_2 using $\omega = \Omega T$ *(2)* Find filter order N, cut-off frequency Ω_c Find normalized filter $H_a(s)$ Find $H_a(s)\big\|_{s \to (s/\Omega_c)}$ *(3)* Find individual pole locations. Then $(s = s_1) \to (z = e^{s_1 T})$ for given T	

Frequency Transformation

Frequency transformation is useful for converting a frequency-selective filter from one type to another.

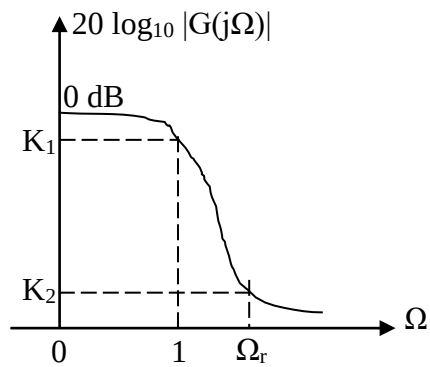
Analog-to-analog transformations Suppose we are given a continuous-time normalized low pass filter *G(s)* with a cut-off frequency of 1 rad/sec. Then what is the effect of the

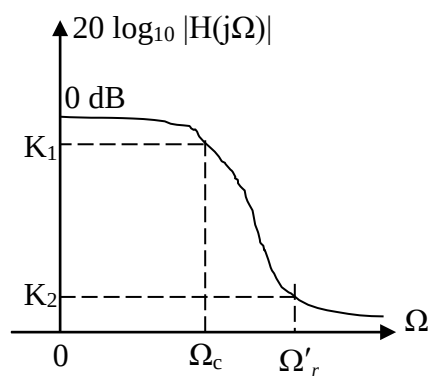
transformation $s' = s\Omega_c$ where s' represents the transformed frequency variable? In other words, we make the substitution $s \to s'/\Omega_c$ in the transfer function. Since $s' = s\Omega_c$ implies $\Omega' = \Omega\,\Omega_c$, it follows that the frequency range $0 \le \Omega \le 1$ is mapped into the range $0 \le \Omega' \le \Omega_c$. Thus $G(s')$ represents a low pass filter with a cut-off frequency of Ω_c. In the rest of what follows, rather than use an explicitly different symbol s' we shall instead indicate such transformation by $s \to s/\Omega_c$ and the resulting transfer function by $H(s) = G(s)\big]_{s \to s/\Omega_c}$.

1. **Low-pass to low-pass transformation** Given the prototype low pass filter $G(s)$ with unit band width (i.e., cut-off frequency = 1 rad/sec.) and unity gain at $\Omega = 0$ (i.e., $|G(j0)| = 1$), the transformation $s \to s/\Omega_c$ gives us a new filter, $H(s)$, with cut-off frequency of Ω_c. The filter $H(s)$ is given by

$$H(s) = G(s)\big]_{s \to s/\Omega_c}$$

The critical frequency Ω_r of the filter $G(s)$ is transformed to Ω'_r of the filter $H(s)$, given by $\Omega'_r = \Omega_r\,\Omega_c$. Both $G(s)$ and $H(s)$ are low pass filters.





More generally, the transformation

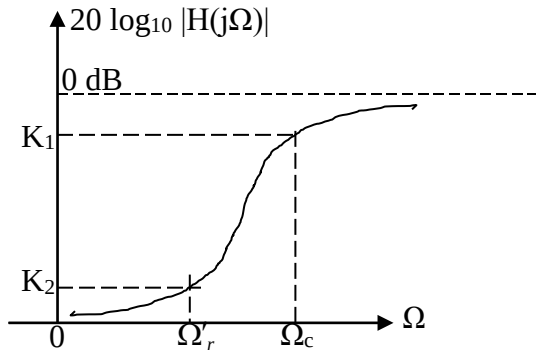
$$s \to s\frac{\Omega_c}{\Omega'_c}$$

transforms a low pass filter with a cut-off (or critical) frequency Ω_c to a low pass filter with a cut-off (or critical) frequency Ω'_c.

2. **Low-pass to high-pass transformation** Given $G(s)$ as above the transformation $s \to \Omega_c/s$ will transform the low-pass $G(s)$ into a high-pass filter $H(s)$ with cut-off frequency of Ω_c:

$$H(s) = G(s)\big]_{s \to \Omega_c/s}$$

The critical frequency Ω_r of the filter $G(s)$ is changed to Ω'_r of $H(s)$, given by $\Omega'_r = \Omega_c/\Omega_r$.



There are similar transformations from low-pass to band-pass and low-pass to band-stop. Refer to table 3.2, page 128, Ludeman.

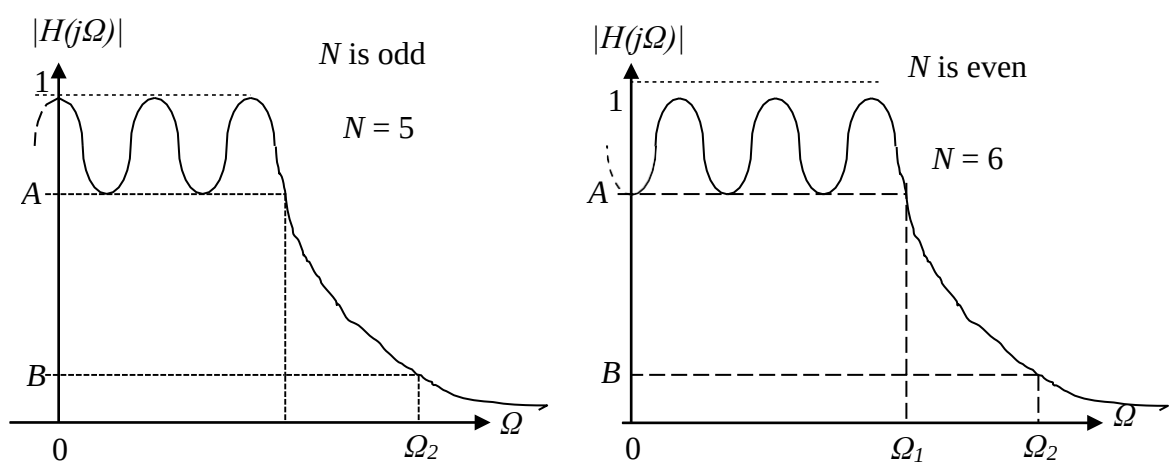
The Low-pass prototype analog unit bandwidth filter could be any analog filter such as Butterworth, Chebyshev etc. of any order, any ripple etc.

Digital-to-digital transformations Similarly, a set of transformations can be found that take a low-pass digital filter and turn it into another low-pass or high-pass or band-pass or band stop digital filter. Refer to Sec 4.3, p. 181, Ludeman.

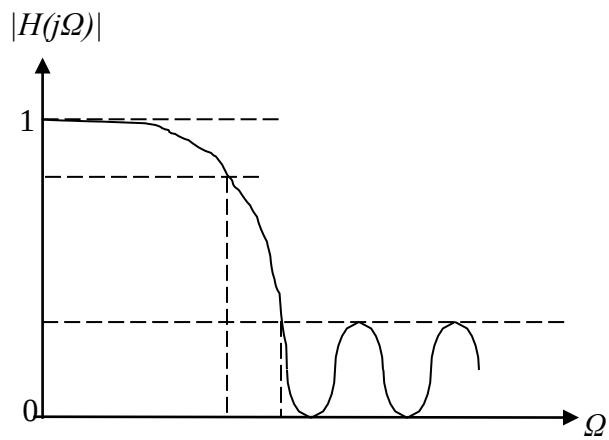
The Chebyshev filter

The Butterworth filter provides a good approximation to the ideal low pass characteristics for values of Ω near zero, but has a low fall-off rate in the transition band. The Chebyshev filter has ripples in either the pass band or the stop band but has a sharper cut-off in the transition band. Thus, for filters of the same order, the Chebyshev filter has a smaller transition band than the Butterworth filter. We now look at the analog low pass Chebyshev filter. There are two types of the Chebyshev filter.

Type I (Chebyshev I) is an *all-pole filter*. It has equiripple behavior in the pass band and monotonically decreases in the stop band. For N = order of the filter, the magnitude response looks as below (N = 5 and N = 6 illustrated). The magnitude, $|H(j\Omega)|$, is an even symmetric function of Ω.

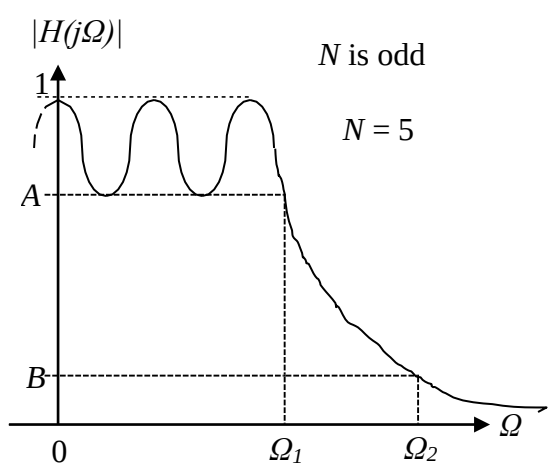


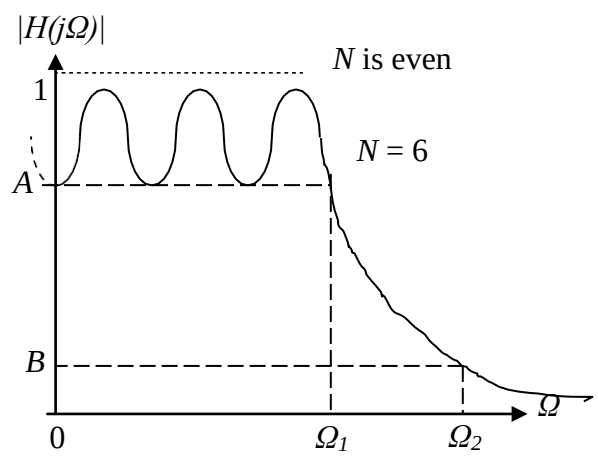
Type II (Chebyshev II or **Inverse Chebyshev)** filter has *both poles and zeros*. It has a monotonically decreasing shape in the pass band and an equiripple behavior in the stop band.



Design of the Chebyshev I filter A typical *magnitude* response specification is sketched below (shown for $N = 5$ and $N = 6$). The magnitudes at the critical frequencies Ω_1 and Ω_2 are A and B, respectively. Typically Ω_1 is in the pass band or is the edge of the pass band and Ω_2 is in the stop band or is the edge of the stop band. In terms of the *log-magnitude* the analog filter specifications are as below. Note that $(20 \log A) = K_1$ dB and $(20 \log B) = K_2$ dB. If A and B are less than 1, K_1 and K_2 are negative.

$$0 \geq 20\log_{10}|H(j\Omega)| \geq K_1 \text{ for all } \Omega \leq \Omega_1$$
$$20\log_{10}|H(j\Omega)| \leq K_2 \text{ for all } \Omega \geq \Omega_2$$





The magnitude characteristic of the Nth order Chebyshev I filter is given by

$$|H(j\Omega)| = \frac{1}{\sqrt{1 + \varepsilon^2 C_N^2(\Omega / \Omega_1)}}, \qquad N = 1, 2, \ldots$$

where ε has to do with pass band ripple and $C_N(x)$ is the Nth order Chebyshev cosine polynomial defined as

$$C_N(x) = \begin{cases} \cos\left(N \cos^{-1} x\right), & x \le 1 \text{ (pass band)} \\ \cosh\left(N \cosh^{-1} x\right), & x > 1 \text{ (outside the pass band)} \end{cases}$$

Chebyshev polynomials are also defined by the recursion formula

$$C_N(x) = 2xC_{N-1}(x) - C_{N-2}(x)$$

with $C_0(x) = 1$ and $C_1(x) = x$. Using this recursion formula we get, for $N = 2$, $C_2(x) = 2xC_1(x) - C_0(x) = 2x^2 - 1$.

Chebyshev Polynomials, $C_N(x)$	
N	$C_N(x)$
0	1
1	x
2	$2x^2 - 1$
3	$4x^3 - 3x$
…	…

[Aside If the frequencies are normalized, that is, for a normalized filter, $\Omega_1 = 1$ rad/sec and the magnitude characteristic of the Nth order Chebyshev I filter is given by

$$|H(j\Omega)| = \frac{1}{\sqrt{1 + \varepsilon^2 C_N^2(\Omega)}}, \qquad N = 1, 2, \ldots$$

End of Aside]

At $\Omega = 0$ we have

$$C_N(0) = \cos\left(N \cos^{-1} 0\right) = \cos\left(N\pi/2\right) = \begin{cases} 0, & N \text{ odd} \\ \pm 1, & N \text{ even} \end{cases}$$

As a consequence, on the vertical axis ($\Omega = 0$) the magnitude curve starts at $|H(j0)| = 1$ for odd N and at $|H(j0)| = A = \dfrac{1}{\sqrt{1+\varepsilon^2}}$ for even N.

At $\Omega = \Omega_1$ we have $C_N(1) = \cos\left(N \cos^{-1} 1\right) = \cos(0) = 1$ for all N. The corresponding magnitude is

$$|H(j\Omega_1)| = A = \frac{1}{\sqrt{1 + \varepsilon^2}}, \qquad \text{for all } N$$

This equation is used to compute ε from the given $|H(j\Omega)|$.

The order, N, of the filter is given by

$$N = \left\lceil \frac{\cosh^{-1}\sqrt{\dfrac{10^{-K_2/10} - 1}{10^{-K_1/10} - 1}}}{\cosh^{-1}\left(\dfrac{\Omega_2}{\Omega_1}\right)} \right\rceil$$

The symbol $\lceil\ \rceil$ means that the computed result is rounded to the next larger integer. For example, if $N = 3.2$ by the above calculation then it is rounded up to 4, and the order of the required filter is $N = 4$. In such a case the resulting filter would exceed the specification at both Ω_1 and Ω_2.

Example 3.9.1 [Filter order] Determine the order of a Chebyshev I filter to have an attenuation of no more than 1 dB for $|\Omega| \le 1000$ rad/sec and at least 10 dB for $|\Omega| \ge 5000$ rad/sec.

Solution The specifications as given are

$$0 \ge 20\log_{10}|H(j\Omega)| \ge -1 \text{ dB for all } \Omega \le 1000 \text{ rad/sec}$$
$$20\log_{10}|H(j\Omega)| \le -10 \text{ dB all } \Omega \ge 5000 \text{ rad/sec}$$

The design parameters are identified as

$$K_1 = -1 \text{ dB}, \quad \Omega_1 = 1000 \text{ rad/sec}$$
$$K_2 = -10 \text{ dB}, \quad \Omega_2 = 5000 \text{ rad/sec}$$

The filter order is given by

$$N = \left\lceil \frac{\cosh^{-1}\sqrt{\dfrac{10^{-K_2/10}-1}{10^{-K_1/10}-1}}}{\cosh^{-1}\left(\dfrac{\Omega_2}{\Omega_1}\right)} \right\rceil = \left\lceil \frac{\cosh^{-1}\sqrt{\dfrac{10^{10/10}-1}{10^{1/10}-1}}}{\cosh^{-1}\left(\dfrac{5000}{1000}\right)} \right\rceil = \left\lceil \frac{\cosh^{-1}\sqrt{\dfrac{10-1}{1.2589-1}}}{\cosh^{-1}(5)} \right\rceil$$

$$= \left\lceil \frac{\cosh^{-1} 5.8956}{\cosh^{-1} 5} \right\rceil = \left\lceil \frac{2.4601}{2.2924} \right\rceil = \lceil 1.073 \rceil = 2$$

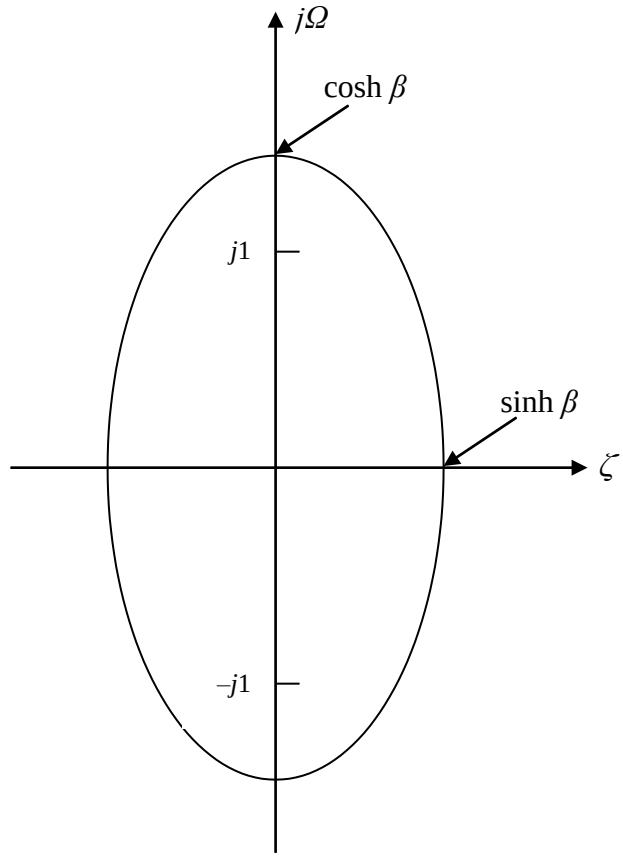
[On the HP 15C, $\cosh^{-1} 5.8956$ is obtained by: (1) Enter Radian mode, (2) Enter 5.8956, (3) g, (4) HYP^{-1}, and (5) COS]

Pole locations and transfer function The poles of the Chebyshev I filter are related to those of the Butterworth filter of the same order and are located on an ellipse in the *s*-plane. If *N* is odd there will be a pole on the negative real axis. In order to find the pole locations and hence the transfer function we introduce the parameter β

$$\beta = \frac{1}{N}\sinh^{-1}(1/\varepsilon)$$

The poles of *H(s)*, $s_k = (\zeta_k + j\Omega_k)$, $k = 0, 1, \ldots, (N-1)$, are given by

$$\zeta_k = \sinh\beta \sin\left(\frac{2k-1}{N}\frac{\pi}{2}\right) \quad \text{and} \quad \Omega_k = \cosh\beta \cos\left(\frac{2k-1}{N}\frac{\pi}{2}\right)$$



Note that if the $\sinh\beta$ and $\cosh\beta$ terms were not present we would have the pole locations of the normalized Butterworth filter (on the unit circle), that is,

$$\zeta_k = \sin\left(\frac{2k-1}{N}\frac{\pi}{2}\right) \quad \text{and} \quad \Omega_k = \cos\left(\frac{2k-1}{N}\frac{\pi}{2}\right)$$

with $\sigma_k^2 + \Omega_k^2 = 1$ which is the unit circle. Thus, the hyperbolic sine and cosine terms are scale factors which, when applied to the Butterworth pole coordinates, give the pole coordinates of a Chebyshev I filter of the same order. The Chebyshev poles are located on an ellipse in the *s*-plane described by

$$\frac{\sigma_k^2}{\sinh^2\beta} + \frac{\Omega_k^2}{\cosh^2\beta} = 1$$

The major axis of the ellipse is on the imaginary ($j\Omega$) axis and the minor axis is on the real axis and the foci are at $\Omega = \pm 1$. The 3 dB cut-off frequency occurs at the point where the ellipse intersects the $j\Omega$ axis, that is, at $\Omega = \cosh\beta$.

In putting together the transfer function, $H(s)$, we rely on the symmetry of pole positions and make use of the left half plane poles only. Finally, the pole positions are scaled by the actual "cut-off frequency" Ω_1. This last step amounts to $s \to s/\Omega_1$ (in the case of the Butterworth design this was $s \to s/\Omega_c$).

Example 3.9.2 [Pole locations and transfer function] Find the pole locations and the transfer function of the Chebyshev I filter designed in above example, that is, with an attenuation of no more than 1 dB for $|\Omega| \le 1000$ rad/sec and at least 10 dB for $|\Omega| \ge 5000$ rad/sec.

Solution The filter order has been determined above as $N = 2$. Further, we know that $|H(j\Omega_1)| = 1/\sqrt{1+\varepsilon^2}$ and $20\log|H(j\Omega_1)| = -1$ dB. Thus

$$20\log\left(1/\sqrt{1+\varepsilon^2}\right) = -1$$

Solving for ε we get $\varepsilon = 0.5088$. Since N is even $|H(j0)| = 1/\sqrt{1+\varepsilon^2} = 0.8913$ is the starting point on the vertical axis.

With regard to the pole locations, if it were a Butterworth filter of order 2 the poles are located at $\zeta_k = \sin\left(\frac{(2k-1)\pi}{2N}\right)$ and $\Omega_k = \cos\left(\frac{(2k-1)\pi}{2N}\right)$, $k = 0, 1$

$$s_{0,1} = \sin\left(\frac{\pi}{4}\right) \pm j\cos\left(\frac{\pi}{4}\right) = \left(-1/\sqrt{2}\right) \pm j\left(1/\sqrt{2}\right)$$

The Chebyshev I poles are then obtained from the Butterworth poles by scaling the real and imaginary parts, respectively, by $\sinh\beta$ and $\cosh\beta$ and then scaling both parts by Ω_1:

$$s_{0,1} = \Omega_1\left\{\left(-1/\sqrt{2}\right)\sinh\beta \pm j\left(1/\sqrt{2}\right)\cosh\beta\right\}$$

where $\Omega_1 = 1000$ rad/sec., and

$$\beta = \frac{1}{N}\sinh^{-1}(1/\varepsilon) = \frac{1}{2}\sinh^{-1}(1/0.5088) = 0.7140$$

Thus the Chebyshev I pole locations are

$$s_{0,1} = -\frac{1000}{\sqrt{2}}\sinh 0.714 \pm j\frac{1000}{\sqrt{2}}\cosh 0.714$$

$$= -\frac{1000}{\sqrt{2}}\sinh 0.714 \pm j\frac{1000}{\sqrt{2}}\cosh 0.714$$

$$= -(707.11)(0.7762) \pm j(707.11)(1.2659) = -548.86 \pm j\,895.15$$

Hence

$$H(s) = \frac{K}{(s-s_0)(s-s_1)} = \frac{K}{(s-(-548.86+j895.15))(s-(-548.86-j895.15))}$$

$$= \frac{K}{(s+548.86)^2 + (895.15)^2}$$

Since N is even the constant K will be adjusted to achieve $|H(j0)| = 1/\sqrt{1+\varepsilon^2} = 0.8913$. (If N were odd, K would be adjusted to achieve $|H(j0)| = 1$.)

$$|H(j0)| = \frac{K}{(548.86)^2 + (895.15)^2} = 0.8913$$

which yields $K = 982694.6$. The filter then is

$$H(s) = \frac{982694.6}{(s+548.86)^2 + (895.15)^2}$$

Example 3.9.3 [Ramesh Babu] Determine the order of a Chebyshev I filter to have a gain of –3 dB or better for $|F| \le 1000$ Hz and a gain of –16 dB or less for $|F| \ge 2000$ Hz. Find the pole locations and the transfer function.

Solution The specifications as given are

$$0 \ge 20\log_{10}|H(j\Omega)| \ge -3 \text{ dB for all } \Omega \le 2\pi\,(1000) \text{ rad/sec}$$
$$20\log_{10}|H(j\Omega)| \le -16 \text{ dB all } \Omega \ge 2\pi\,(2000) \text{ rad/sec}$$

The design parameters are identified as

$$K_1 = -3 \text{ dB}, \quad \Omega_1 = 2000\,\pi \text{ rad/sec}$$
$$K_2 = -16 \text{ dB}, \quad \Omega_2 = 4000\,\pi \text{ rad/sec}$$

The filter order is given by

$$N = \left\lceil \frac{\cosh^{-1}\sqrt{\dfrac{10^{-K_2/10}-1}{10^{-K_1/10}-1}}}{\cosh^{-1}\left(\dfrac{\Omega_2}{\Omega_1}\right)} \right\rceil = \left\lceil \frac{\cosh^{-1}\sqrt{\dfrac{10^{16/10}-1}{10^{3/10}-1}}}{\cosh^{-1}\left(\dfrac{4000\pi}{2000\pi}\right)} \right\rceil = \left\lceil \frac{\cosh^{-1}\sqrt{\dfrac{39.81-1}{1.995-1}}}{\cosh^{-1}(2)} \right\rceil$$
$$= \left\lceil \frac{\cosh^{-1} 6.2445}{\cosh^{-1} 2} \right\rceil = \left\lceil \frac{2.5184}{1.3169} \right\rceil = \lceil 1.9122 \rceil = 2$$

Further $|H(j\Omega_1)| = 1/\sqrt{1+\varepsilon^2}$ and $20\log|H(j\Omega_1)| = -3$ dB implies that $|H(j\Omega_1)| = 1/\sqrt{2}$.

From these two conditions it follows that $\varepsilon = 1$. Since N is even $|H(j0)| = 1/\sqrt{1+\varepsilon^2} = 1/\sqrt{2}$.

The poles of $H(s)$ are given by $s_k = \zeta_k + j\Omega_k$, $k = 0, 1, \ldots, 2N-1$ and

$$\zeta_k = \sin\left(\frac{(2k-1)\pi}{2N}\right)\sinh\beta \quad \text{and} \quad \Omega_k = \cos\left(\frac{(2k-1)\pi}{2N}\right)\cosh\beta$$

$$\beta = \frac{1}{N}\sinh^{-1}(1/\varepsilon) = \frac{1}{2}\sinh^{-1}(1/1) = \frac{0.88137}{2} = 0.44069$$

Thus, one of the left half plane poles is

$$s_1 = \sigma_1 + j\Omega_1 = \sin\left(\frac{-\pi}{4}\right)\sinh 0.44069 + j\cos\left(\frac{-\pi}{4}\right)\cosh 0.44069$$
$$= -\frac{1}{\sqrt{2}}0.45509 + j\frac{1}{\sqrt{2}}1.09868 = -0.3218 + j0.77689$$

Scaling this by $\Omega_1 = 2000\,\pi$ rad/sec results in

$$s_1 = 2000\,\pi\,(-0.3218 + j0.77689) = -2021.9 + j4881.3 = -a + jb$$

where $a = 2021.9$ and $b = 4881.3$. The other left half plane pole is the conjugate of the above, at

$$s_2 = -2021.9 - j4881.3 = -a - jb$$

The transfer function is $H(s) = 1/D(s)$, where the denominator, $D(s)$, is put together from the poles as follows

$$D(s) = (s - s_1)(s - s_2) = \big(s - (-2021.9 + j4881.3)\big)\big(s - (-2021.9 - j4881.3)\big)$$
$$= (s + 2021.9)^2 + (4881.3)^2$$

For convenience we shall write this as $D(s) = (s+a)^2 + b^2 = s^2 + 2sa + a^2 + b^2$, so that

$$H(s) = \frac{K}{D(s)} = \frac{K}{s^2 + 2s \quad a^2 + b^2}$$

where the constant K is adjusted so as to make $|H(j0)| = 1/\sqrt{2}$:

$$|H(j0)| = \frac{K}{a^2 + b^2} = 1/\sqrt{2} \quad \rightarrow \quad K = \frac{a^2 + b^2}{\sqrt{2}}$$

Thus

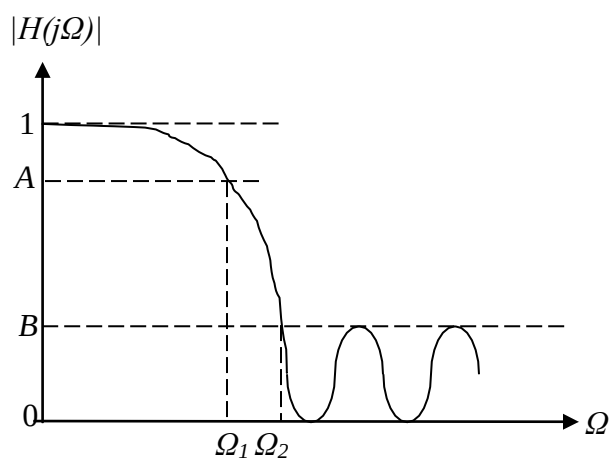
$$H(s) = \frac{a^2 + b^2}{\sqrt{2}} \frac{1}{s^2 + 2s \quad a^2 + b^2}$$

$$= 19739005.5 \frac{1}{s^2 + 4043.8s + 27915169.3}$$

Design of the Chebyshev II filter A typical *magnitude* response specification is sketched below. The magnitudes at the critical frequencies Ω_1 and Ω_2 are A and B, respectively. Typically Ω_1 is in the pass band or is the edge of the pass band and Ω_2 is in the stop band or is the edge of the stop band. In terms of the *log-magnitude* the analog filter specifications are as below. Note that (20 log A) = K_1 dB and (20 log B) = K_2 dB. If A and B are less than 1, K_1 and K_2 are negative.

$$0 \geq 20\log_{10}|H(j\Omega)| \geq K_1 \text{ for all } \Omega \leq \Omega_1$$

$$20\log_{10}|H(j\Omega)| \leq K_2 \text{ for all } \Omega \geq \Omega_2$$



The magnitude characteristic of the Nth order Chebyshev II filter is given by

$$|H(j\Omega)| = \frac{1}{\sqrt{1 + \varepsilon^2 \frac{C_N^2(\Omega_2/\Omega_1)}{C_N^2(\Omega_2/\Omega)}}}, \qquad N = 1, 2, \ldots$$

where ε has to do with pass band attenuation and $C_N(x)$ is the Nth order Chebyshev polynomial.

At $\Omega = \Omega_1$ we have

$$|H(j\Omega_1)| = \frac{1}{\sqrt{1 + \varepsilon^2 \frac{C_N^2(\Omega_2/\Omega_1)}{C_N^2(\Omega_2/\Omega_1)}}} = \frac{1}{\sqrt{1 + \varepsilon^2}} = A, \qquad \text{for all } N$$

At $\Omega = 0$, $C_N(\Omega_2/\Omega) = C_N(\infty) \rightarrow \infty$, and the magnitude becomes

$$|H(j0)| = \frac{1}{\sqrt{1+\varepsilon^2 \frac{C_N^2(\Omega_2/\Omega_1)}{C_N^2(\Omega_2/0)}}} = 1 \quad \text{for all } N$$

The magnitude curve starts at $|H(j0)| = 1$ on the vertical axis for all N.

At $\Omega = \Omega_2$ we have $C_N(\Omega_2/\Omega_2) = C_N(1) = \cos\left(N \cos^{-1} 1\right) = \cos(0) = 1$

$$|H(j\Omega_2)| = \frac{1}{\sqrt{1+\varepsilon^2 \frac{C_N^2(\Omega_2/\Omega_1)}{C_N^2(\Omega_2/\Omega_2)}}} = \frac{1}{\sqrt{1+\varepsilon^2 \frac{C_N^2(\Omega_2/\Omega_1)}{1}}}$$

$$= \frac{1}{\sqrt{1+\varepsilon^2 C_N^2(\Omega_2/\Omega_1)}} = B, \quad \text{for all } N$$

The order, N, of the filter is given by

$$N = \left\lceil \frac{\cosh^{-1}\sqrt{\frac{10^{-K_2/10}-1}{10^{-K_1/10}-1}}}{\cosh^{-1}\left(\frac{\Omega_2}{\Omega_1}\right)} \right\rceil$$

The symbol $\lceil\ \rceil$ means that the computed result is rounded to the next larger integer. For example, if $N = 3.2$ by the above calculation then it is rounded up to 4, and the order of the required filter is $N = 4$. In such a case the resulting filter would exceed the specification at both Ω_1 and Ω_2.

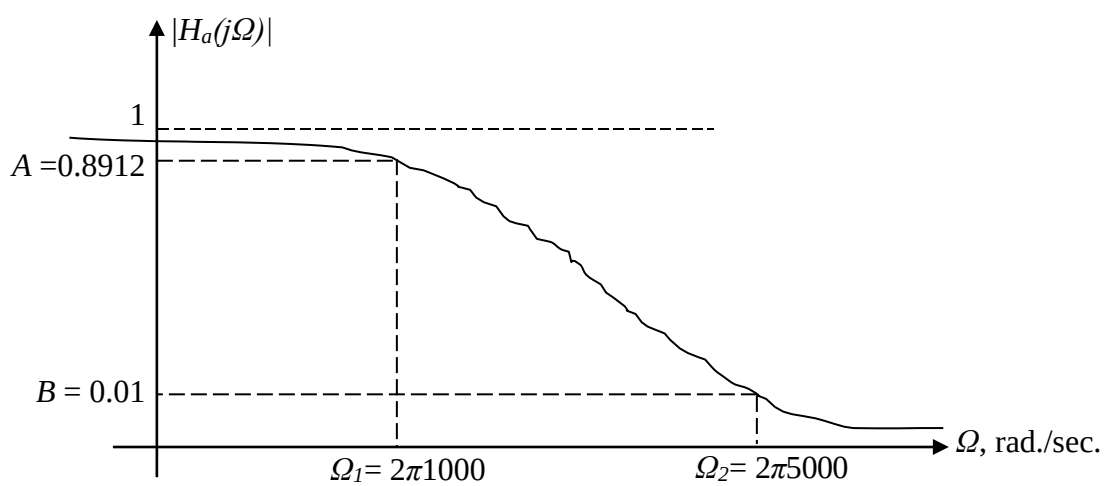
Example 3.9.4 If an analog low pass filter is to have an attenuation of 1 dB at cut-off frequency of 1 kHz, and a maximum stop band ripple of 0.01 for $|F| > 5$ kHz, determine the required the filter order for (a) a Butterworth filter, (b) a Chebyshev I filter, and (c) a Chebyshev II filter.

Solution The specifications are the same for all three cases but the magnitude characteristic differs from one case to the next.

(a) The Butterworth magnitude (absolute value) characteristic is sketched below.

$\Omega_1 = 2\pi F_1 = 2\pi\, 1000$ rad/sec., $\quad K_1 = -1$ dB $\quad \rightarrow A = 0.8912$

$\Omega_2 = 2\pi F_2 = 2\pi\, 5000$ rad/sec., $\quad B = 0.01 \quad \rightarrow K_2 = -40$ dB



The relation between the absolute values and the dB figures ($K_1 = 20 \log A$ and $K_2 = 20 \log B$) is used to compute $A = 10^{K_1/20} = 10^{-1/20} = 0.8912$ and $K_2 = 20 \log B = 20 \log 0.01 = -40$ dB.

The Butterworth filter order is given by

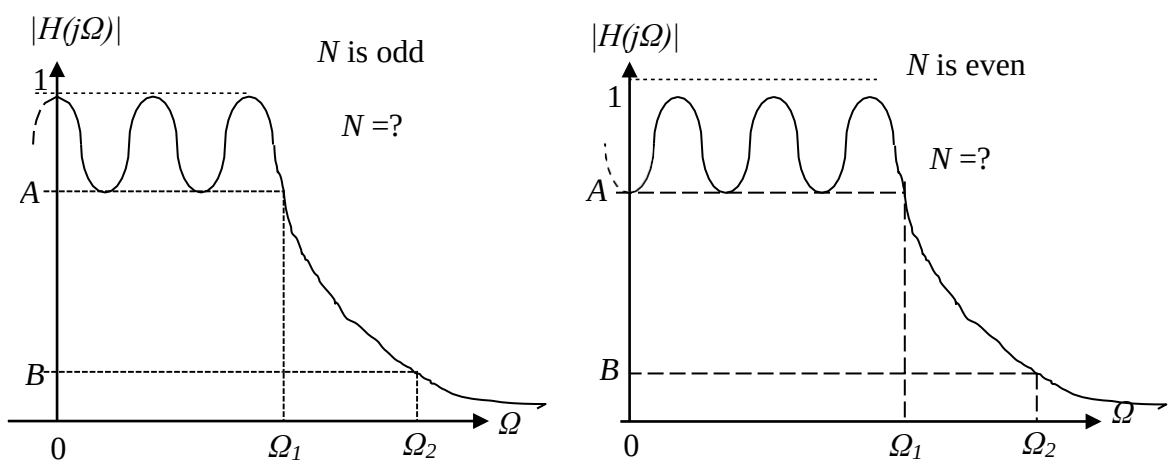
$$N = \left\lceil \frac{\log_{10}\left(\dfrac{10^{-K_1/10}-1}{10^{-K_2/10}-1}\right)}{2\log_{10}\left(\dfrac{\Omega_1}{\Omega_2}\right)} \right\rceil = \left\lceil \frac{\log_{10}\left(\dfrac{10^{-(-1)/10}-1}{10^{-(-40)/10}-1}\right)}{2\log_{10}\left(\dfrac{1}{5}\right)} \right\rceil = \left\lceil \frac{\log_{10}\left(\dfrac{10^{0.1}-1}{10^{4}-1}\right)}{2\log_{10}\left(\dfrac{1}{5}\right)} \right\rceil$$

$$= \left\lceil \frac{\log_{10}\left(\dfrac{1.25892-1}{10000-1}\right)}{2\log_{10}(0.2)} \right\rceil = \left\lceil \frac{-4.5868}{-1.3979} \right\rceil = \lceil 3.28 \rceil = 4$$

(b) The Chebyshev I specs and magnitude (absolute value) characteristic are diagrammed below. As earlier we compute $A = 10^{K_1/20} = 10^{-1/20} = 0.8912$ and $K_2 = 20 \log B = 20 \log 0.01 = -40$ dB.

$\Omega_1 = 2\pi F_1 = 2\pi\, 1000$ rad/sec., $\quad K_1 = -1$ dB $\quad \rightarrow A = 0.8912$

$\Omega_2 = 2\pi F_2 = 2\pi\, 5000$ rad/sec., $\quad B = 0.01 \quad \rightarrow K_2 = -40$ dB



The Chebyshev I filter order is given by

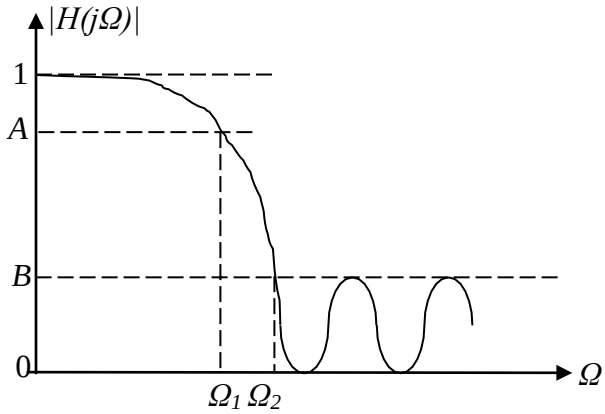
$$N = \left\lceil \frac{\cosh^{-1}\sqrt{\dfrac{10^{-K_2/10}-1}{10^{-K_1/10}-1}}}{\cosh^{-1}\left(\dfrac{\Omega_2}{\Omega_1}\right)} \right\rceil = \left\lceil \frac{\cosh^{-1}\sqrt{\dfrac{10^{-(-40)/10}-1}{10^{-(-1)/10}-1}}}{\cosh^{-1}\left(\dfrac{5}{1}\right)} \right\rceil = \left\lceil \frac{\cosh^{-1}\sqrt{\dfrac{10000-1}{10^{0.1}-1}}}{\cosh^{-1}(5)} \right\rceil$$

$$= \left\lceil \frac{\cosh^{-1}\sqrt{38617.3}}{\cosh^{-1}(5)} \right\rceil = \left\lceil \frac{\cosh^{-1}(196.5)}{\cosh^{-1}(5)} \right\rceil = \left\lceil \frac{5.9738}{2.2924} \right\rceil = \lceil 2.6059 \rceil = 3$$

(c) The Chebyshev II specs and magnitude (absolute value) characteristic are shown below. As earlier we compute $A = 10^{K_1/20} = 10^{-1/20} = 0.8912$ and $K_2 = 20 \log B = 20 \log 0.01 = -40$ dB.

$\Omega_1 = 2\pi F_1 = 2\pi 1000$ rad/sec., $\quad K_1 = -1$ dB $\quad \rightarrow A = 0.8912$

$\Omega_2 = 2\pi F_2 = 2\pi 5000$ rad/sec., $\quad B = 0.01 \quad \rightarrow K_2 = -40$ dB



The Chebyshev II filter order is given by

$$N = \left\lceil \frac{\cosh^{-1}\sqrt{\dfrac{10^{-K_2/10}-1}{10^{-K_1/10}-1}}}{\cosh^{-1}\left(\dfrac{\Omega_2}{\Omega_1}\right)} \right\rceil = \left\lceil \frac{\cosh^{-1}\sqrt{\dfrac{10^{-(-40)/10}-1}{10^{-(-1)/10}-1}}}{\cosh^{-1}\left(\dfrac{5}{1}\right)} \right\rceil = \left\lceil \frac{\cosh^{-1}\sqrt{\dfrac{10000-1}{10^{0.1}-1}}}{\cosh^{-1}(5)} \right\rceil$$

$$= \left\lceil \frac{\cosh^{-1}\sqrt{38617.3}}{\cosh^{-1}(5)} \right\rceil = \left\lceil \frac{\cosh^{-1}(196.5)}{\cosh^{-1}(5)} \right\rceil = \left\lceil \frac{5.9738}{2.2924} \right\rceil = \lceil 2.6059 \rceil = 3$$

Example 3.9.5 Determine the system function *H(z)* of the lowest order Chebyshev filter that meets the following specs. Use the impulse invariance method.

(a) 0.5 dB ripple in the pass band, $0 \le |\omega| \le 0.24\pi$

(b) At least 50 db attenuation in the stop band, $0.35\pi \le |\omega| \le \pi$

Solution We assume the Chebyshev I filter. The procedure is similar for the Chebyshev II. The specs are:

$\omega_1 = 0.24\pi$ rad., $\quad K_1 = -0.5$ dB $\rightarrow \quad A = 10^{K_1/20} = 0.94406$

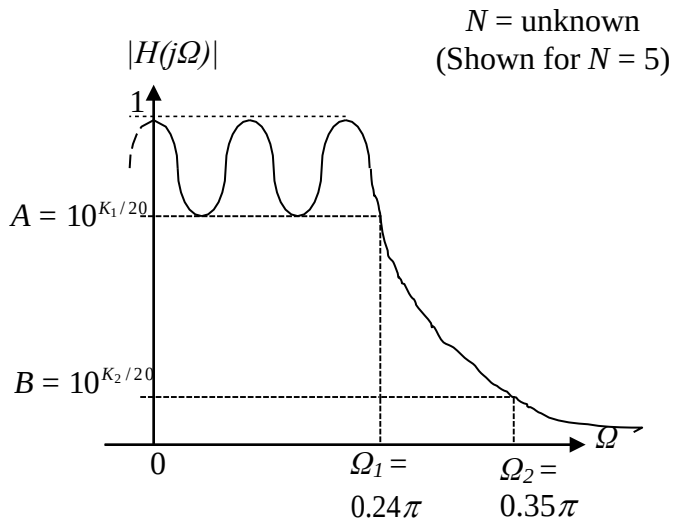
$\omega_2 = 0.35\pi$ rad., $\quad K_2 = -50$ dB $\rightarrow \quad B = 10^{K_2/20} = 0.0031622$

The sampling time, *T*, is not specified. Since this is impulse invariant design, *T* should be very small – the smaller the better. Strictly for convenience we shall use $T = 1$ sec., and convert the digital specs to analog using the relation $\omega = \Omega T$.

$\Omega_1 = \omega_1/T = 0.24\pi$ rad/sec, $\quad K_1 = -0.5$ dB $\rightarrow \quad A = 10^{K_1/20} = 0.94406$

$\Omega_2 = \omega_2/T = 0.35\pi$ rad/sec, $\quad K_2 = -50$ dB $\rightarrow \quad B = 10^{K_2/20} = 0.0031622$

The magnitude (absolute value) characteristic is diagrammed below.



The Chebyshev I filter order is given by

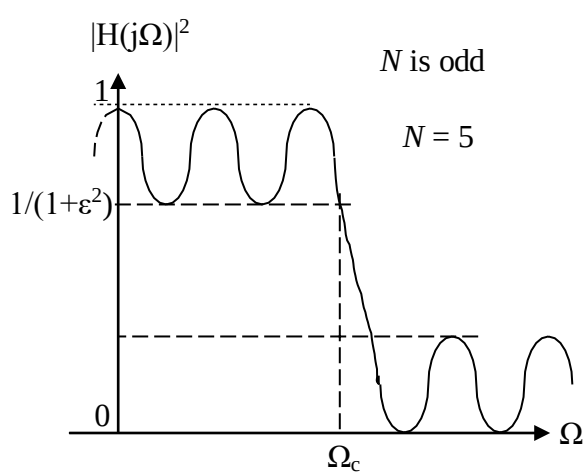
$$N = \left\lceil \frac{\cosh^{-1}\sqrt{\dfrac{10^{-K_2/10}-1}{10^{-K_1/10}-1}}}{\cosh^{-1}\left(\dfrac{\Omega_2}{\Omega_1}\right)} \right\rceil = \left\lceil \frac{\cosh^{-1}\sqrt{\dfrac{10^{-(-50)/10}-1}{10^{-(-0.5)/10}-1}}}{\cosh^{-1}\left(\dfrac{0.35}{0.24}\right)} \right\rceil = \left\lceil \frac{\cosh^{-1}\sqrt{\dfrac{100000-1}{10^{0.05}-1}}}{\cosh^{-1}(1.4583)} \right\rceil$$

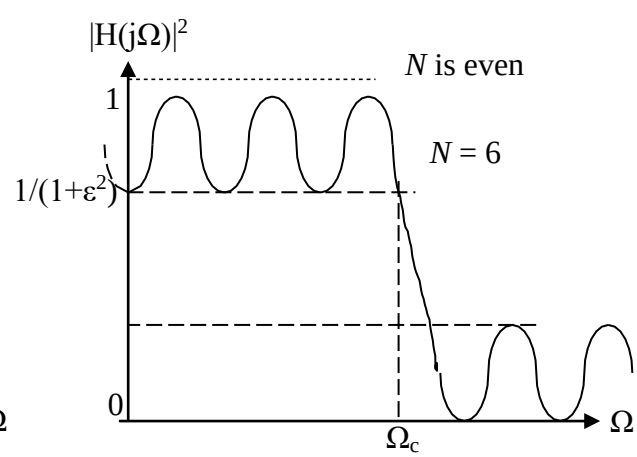
$$= \left\lceil \frac{\cosh^{-1}\sqrt{819539.96}}{\cosh^{-1}(1.4583)} \right\rceil = \left\lceil \frac{\cosh^{-1}(905.28)}{\cosh^{-1}(1.4583)} \right\rceil = \left\lceil \frac{7.5014}{0.9242} \right\rceil = \lceil 8.117 \rceil = 9$$

Determine the poles and transfer function *H(s)*, *h(t)*, *h(n)*, and *H(z)*.

The Elliptic (or Cauer) filter

An approximation to the ideal low-pass characteristic, which, for a given order of filter, has an even smaller transition band than the Chebyshev filter, can be obtained in terms of Jacobi elliptic sine functions. The resulting filter is called an elliptic filter. The magnitude characteristic of the elliptic filter has ripples in both the pass band and the stop band.





UNIT-3
FIR digital filters

Characteristics of FIR digital filters, Frequency response, Design of FIR digital filters using Window techniques, Frequency sampling technique, Comparison of IIR and FIR filters.

Contents:

FIR - Recapitulation

Nomenclature With $a_0 = 1$ in the linear constant coefficient difference equation,

$$a_0\, y(n) + a_1\, y(n-1) + \ldots + a_N\, y(n-N)$$
$$= b_0\, x(n) + b_1\, x(n-1) + \ldots + b_M\, x(n-M), \qquad a_0 \neq 0$$

we have,

$$H(z) = \frac{\sum_{i=0}^{M} b^i z^{-i}}{1 + \sum_{i=1}^{N} a_i z^{-i}}$$

This represents an IIR filter if at least one of a_1 through a_N is nonzero, and all the roots of the denominator are not canceled exactly by the roots of the numerator. In general, there are M finite zeros and N finite poles. There is no restriction that M should be less than or greater than or equal to N. In most cases, especially digital filters derived from analog designs, $M \leq N$. Systems of this type are called N^{th} order systems. This is the case with IIR filter design.

When $M > N$, the order of the system is no longer unambiguous. In this case, $H(z)$ may be taken to be an Nth order system in cascade with an FIR filter of order $(M - N)$.

When $N = 0$, as in the case of an FIR filter, according to our convention the *order* is 0. However, it is more meaningful in such a case to focus on M and call the filter an FIR filter of M stages or $(M+1)$ coefficients.

Example The system $H(z) = (1 - z^{-8})/(1 - z^{-1})$ is an FIR filter. Why (verify)?

An FIR filter then has only the "b" coefficients and all the "a" coefficients (except a_0 which equals 1) are zero. An example is the three-term moving average filter $y(n) = (1/3)\, x(n) + (1/3)\, x(n-1) + (1/3) x(n-2)$. In general the difference equation of an FIR filter can be written

$$y(n) = \sum_{r=0}^{M} b_r\, x(n-r) = b_0 x(n) + b_1\, x(n-1) + \ldots + b_M\, x(n-M) \qquad \rightarrow (1)$$

There are $(M + 1)$ coefficients; some use only M coefficients. This equation describes a **nonrecursive** implementation. Its impulse response $h(n)$ is made up of the coefficients $\{b_r\} = \{b_0, b_1, \ldots, b_M\}$

$$h(n) = \begin{cases} b_n, \text{ for } 0 \leq n \leq M \\ 0, \text{ elsewhere} \end{cases} = \{b_0, b_1, \ldots, b_M\}$$

Equivalently, the finite length impulse response can also be written in the form of a weighted sum of δ functions as was done in Unit I $\left(\textit{for example}, x(n) = \sum_{k=-\infty}^{\infty} x(k)\delta(n-k) \right)$

$$h(n) = \sum_{r=0}^{M} b_r \delta(n-r) = b_0\, \delta(n) + b_1 \delta(n-1) + \ldots + b_M\, \delta(n-M)$$

The difference equation (1) is also equivalent to a direct convolution of the input and the impulse response:

$$y(n) = \sum_{r=0}^{M} b_r\, x(n-r) = \sum_{r=0}^{M} b(r) x(n-r)$$

where we have written b_r as $b(r)$, i.e., the subscript in b_r is written as an index in $b(r)$.

The transfer function $H(z)$ of the FIR filter can be obtained either from the difference equation or from the impulse response $h(n)$:

$$H(z) = \sum_{n=0}^{M} h(n) z^{-n} = b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_M z^{-M}$$

$$= \frac{b_0 z^M + b_1 z^{M-1} + \ldots + b_{M-1} z^1 + b_M}{z^M}$$

$$= \frac{b_0 \left(z^M + \frac{b_1}{b_0} z^{M-1} + \square + \frac{b_{M-1}}{b_0} z^1 + \frac{b_M}{b_0} \right)}{z^M}$$

The transfer function has M nontrivial zeros and an Mth order (trivial) pole at $z = 0$. This is considered as an *all-zero system*.

We may obtain the frequency response $H(e^{j\omega})$ or $H(\omega)$ of the FIR filter either from $H(z)$ as

$$H(e^{j\omega}) = H(z)\Big|_{z=e^{j\omega}}$$

or, from the impulse response, $h(n)$, as the discrete-time Fourier transform (DTFT) of $h(n)$:

$$H(e^{j\omega}) = \sum_{n=-\infty}^{\infty} h(n) e^{-j\omega n} = \sum_{n=0}^{M} b_n e^{-j\omega n} = b_0 + b_1 e^{-j\omega} + b_2 e^{-j2\omega} + \ldots + b_M e^{-jM\omega}$$

The inverse DTFT of $H(\omega)$ is of course the impulse response, given by

$$h(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} H(\omega) e^{j\omega n} d\omega$$

The basic design problem is to determine the impulse response $h(n)$, or, the coefficients b_r, for $r = 0$ to M, required to achieve a desired $H(\omega)$. These coefficients are of course the constants that appear in the numerator of the transfer function $H(z)$. The various transformations used in IIR filter design cannot be used here since they usually yield IIR functions, i.e., with both numerator and denominator coefficients.

Characteristics of FIR digital filters

Illustration The equations of the three-term **moving average filter** are repeated below

$$y(n) = \frac{x(n) + x(n-1) + x(n-2)}{3}$$

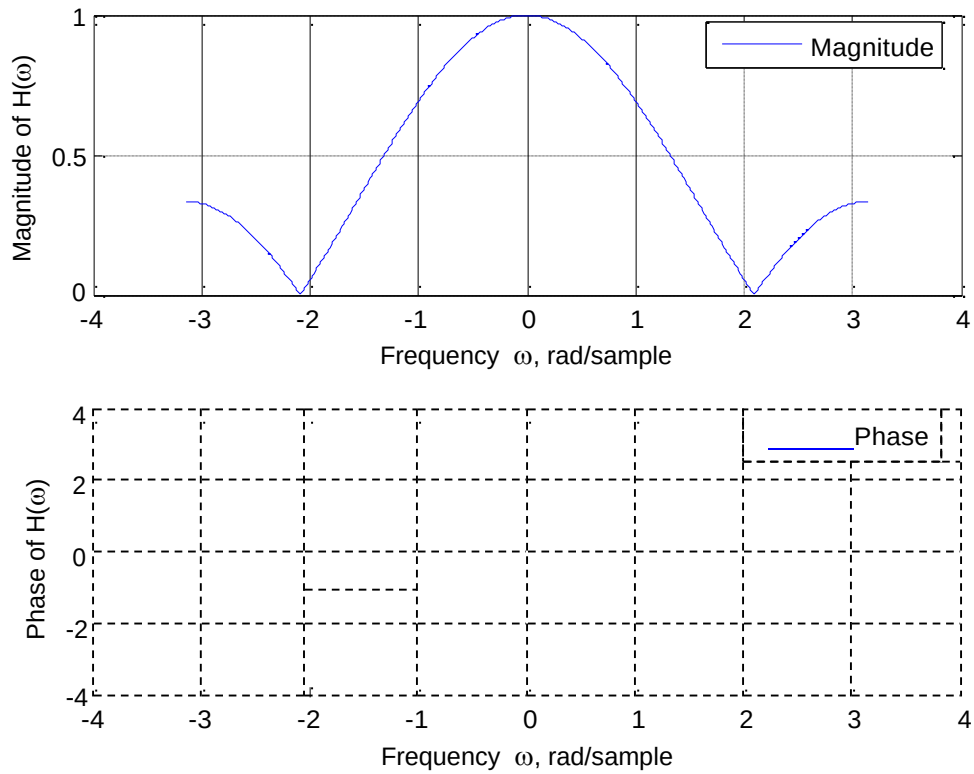
$$\frac{Y(z)}{X(z)} = H(z) = \frac{1 + z^{-1} + z^{-2}}{3}$$

$$H(e^{j\omega}) = \frac{1 + e^{-j\omega} + e^{-j2\omega}}{3} = \frac{e^{-j\omega}(e^{j\omega} + 1 + e^{-j\omega})}{3} = \left(\frac{1 + 2\cos\omega}{3}\right) e^{-j\omega}$$

This is a crude low pass filter with linear phase, $\angle H(\omega) = -\omega$.

```
%Magnitude and phase response of 3-coefficient moving average filter
%Filter coefficients: h(n) = {1/3, 1/3, 1/3}
b3=[1/3, 1/3, 1/3],
a=[1]
w=-pi: pi/256: pi;
Hw3=freqz(b3, a, w);
subplot(2, 1, 1), plot(w, abs(Hw3)); legend ('Magnitude');
xlabel('Frequency \omega, rad/sample'), ylabel('Magnitude of H(\omega)'); grid
```

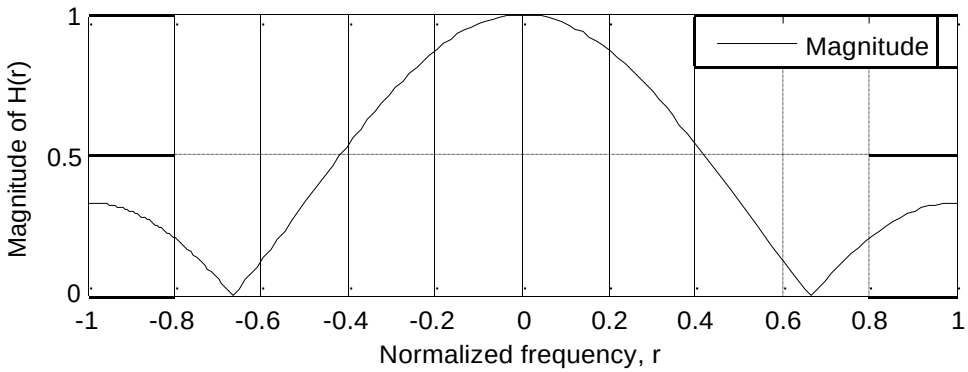
```
subplot(2, 1, 2), plot(w, angle(Hw3)); legend ('Phase');
xlabel('Frequency \omega, rad/sample'), ylabel('Phase of H(\omega)'); grid
```

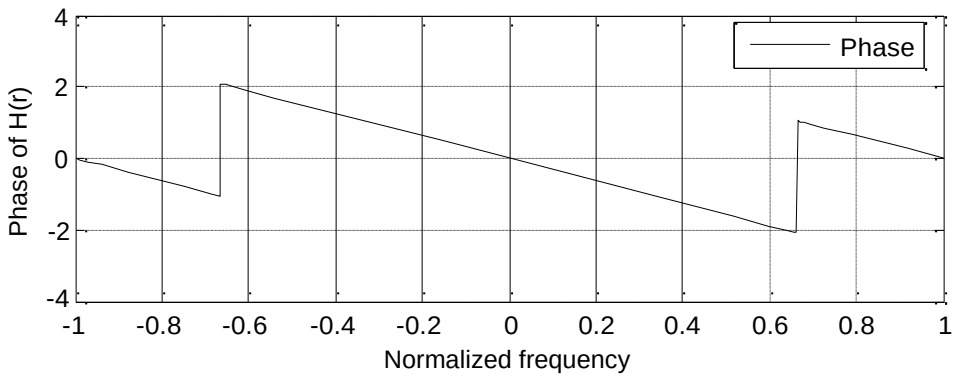


Normalized frequency We define the $r = \omega/\pi$. As ω goes from $-\pi$ to π the variable r goes from –1 to 1. This corresponds to a frequency range of $-F_s/2$ to $F_s/2$ Hz. In terms of the normalized frequency the frequency response of the three-term moving average filter becomes

$$H(r) = \frac{1 + e^{-j\pi r} + e^{-j2\pi r}}{3}$$

```
%Magnitude and phase response of 3-coefficient moving average filter
%Filter coefficients: h(n) = {1/3, 1/3, 1/3}
subplot(2,1,1); fplot('abs((1/3)*(1+exp(-j*pi*r)+exp(-j*2*pi*r)))', [-1, 1], 'k');
legend ('Magnitude');
xlabel('Normalized frequency, r'); ylabel('Magnitude of H(r)'); grid
subplot(2,1,2);fplot('angle((1/3)*(1+exp(-j*pi*r)+exp(-j*2*pi*r)))', [-1, 1], 'k');
legend ('Phase');
xlabel('Normalized frequency');ylabel('Phase of H(r)'); grid
```


Magnitude
Magnitude of H(r)
1
0.5
0
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
Normalized frequency, r


Phase
Phase of H(r)
4
2
0
-2
-4
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
Normalized frequency

We illustrate below the characteristics of several types of FIR filter. The filter length N may be an odd (preferred) or an even number. Further, we are typically interested in linear phase. This requires the impulse response to have either even or odd symmetry about its center.

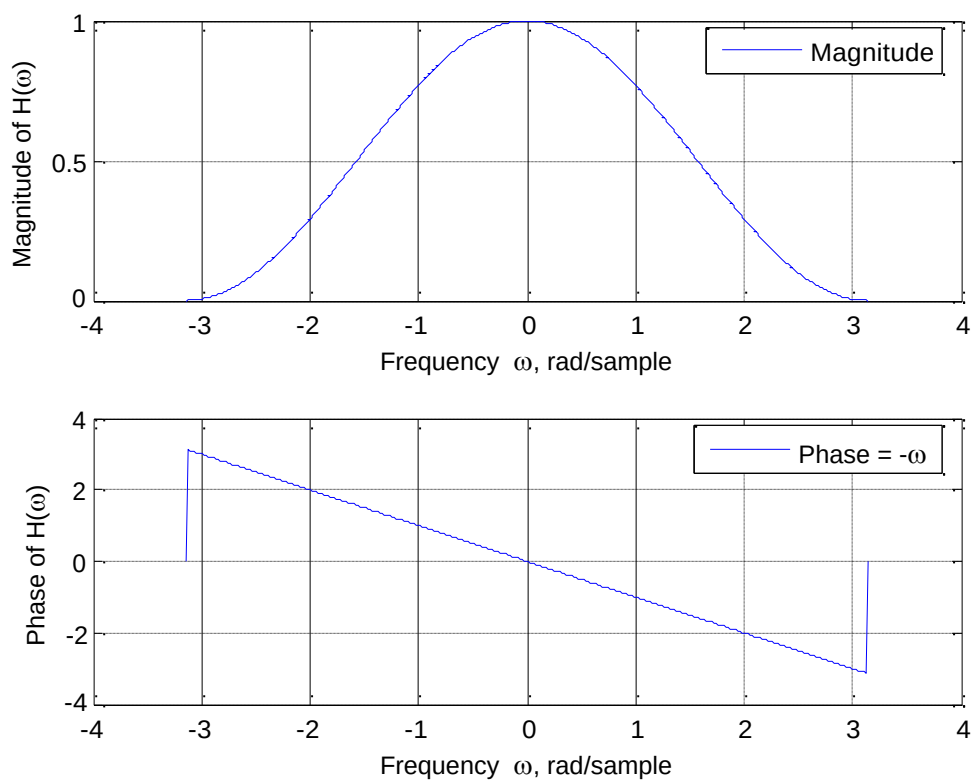
Example 4.2.1 Find the frequency response of the following FIR filters

A. $h(n) = \{0.25, 0.5, 0.25\}$ Even symmetry
B. $h(n) = \{0.5, 0.3, 0.2\}$ No symmetry
C. $h(n) = \{0.25, 0.5, -0.25\}$ No symmetry
D. $h(n) = \{0.25, 0, -0.25\}$ Odd symmetry

Solution

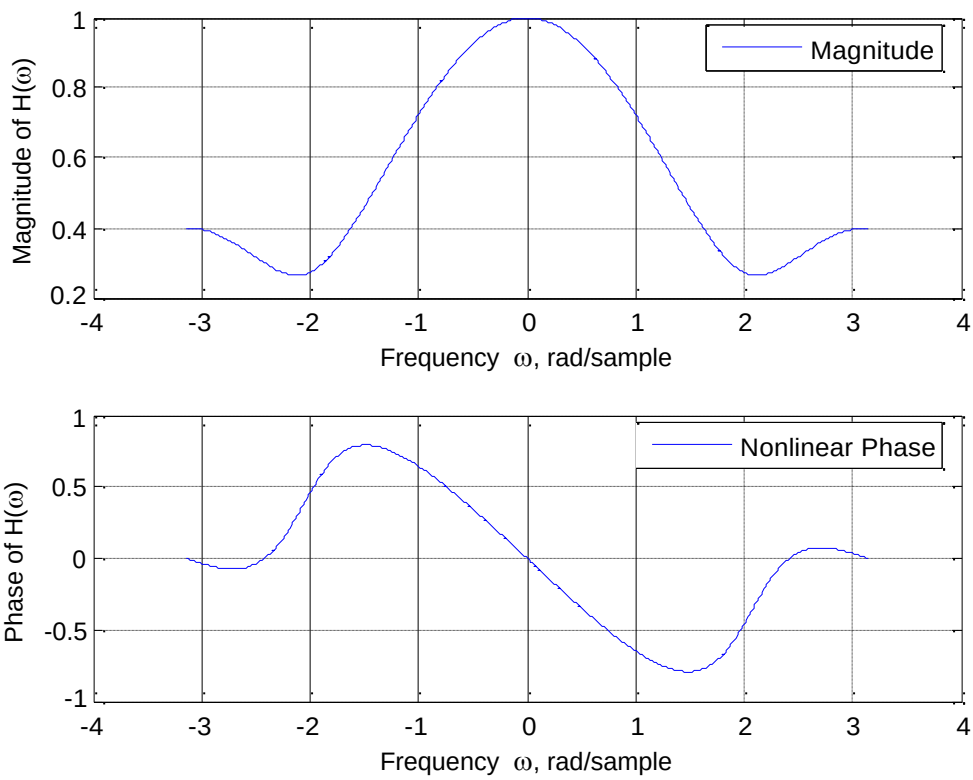
(A) The sequence $h(n) = \{0.25, 0.5, 0.25\}$ has even symmetry.

```
%Magnitude and phase response of h(n) = {0.25, 0.5, 0.25}
%Filter coefficients – Even symmetry
b3=[0.25, 0.5, 0.25],
a=[1]
w=-pi: pi/256: pi;
Hw3=freqz(b3, a, w);
subplot(2, 1, 1), plot(w, abs(Hw3)); legend ('Magnitude');
xlabel('Frequency \omega, rad/sample'), ylabel('Magnitude of H(\omega)'); grid
subplot(2, 1, 2), plot(w, angle(Hw3)); legend ('Phase = -\omega');
xlabel('Frequency \omega, rad/sample'), ylabel('Phase of H(\omega)'); grid
```



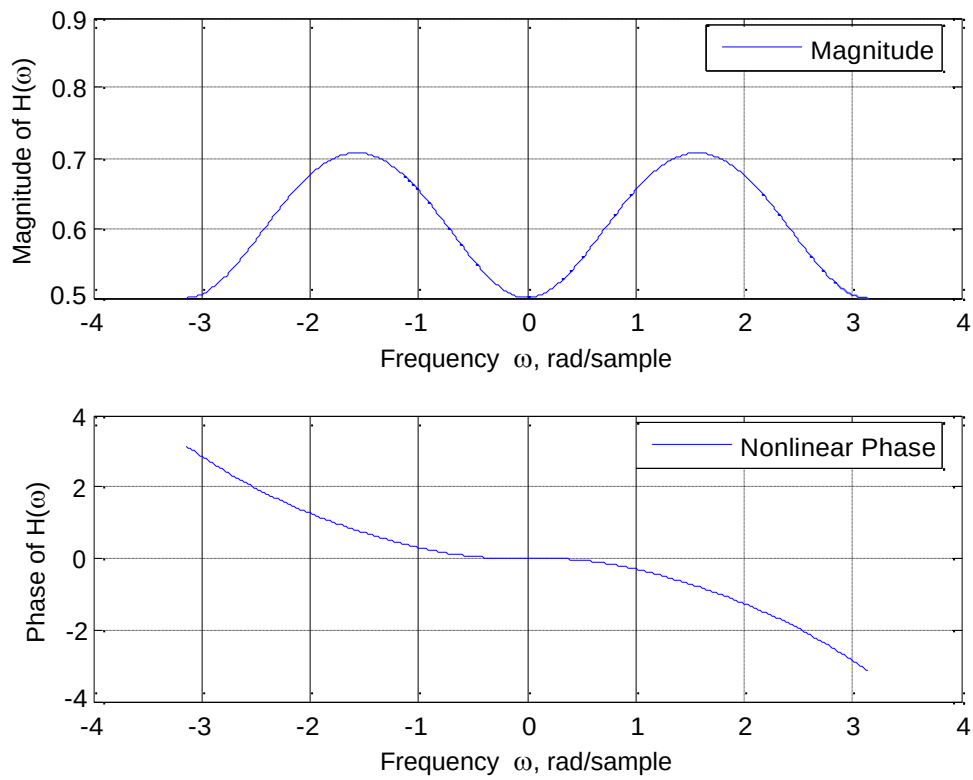
(B) The sequence *h(n)* = {0.5, 0.3, 0.2}is not symmetric.

```
%Magnitude and phase response of h(n) = {0.5, 0.3, 0.2}
%Filter coefficients – No symmetry
b3=[0.5, 0.3, 0.2],
a=[1]
w=-pi: pi/256: pi;
Hw3=freqz(b3, a, w);
subplot(2, 1, 1), plot(w, abs(Hw3)); legend ('Magnitude');
xlabel('Frequency \omega, rad/sample'), ylabel('Magnitude of H(\omega)'); grid
subplot(2, 1, 2), plot(w, angle(Hw3)); legend ('Nonlinear Phase');
xlabel('Frequency \omega, rad/sample'), ylabel('Phase of H(\omega)'); grid
```



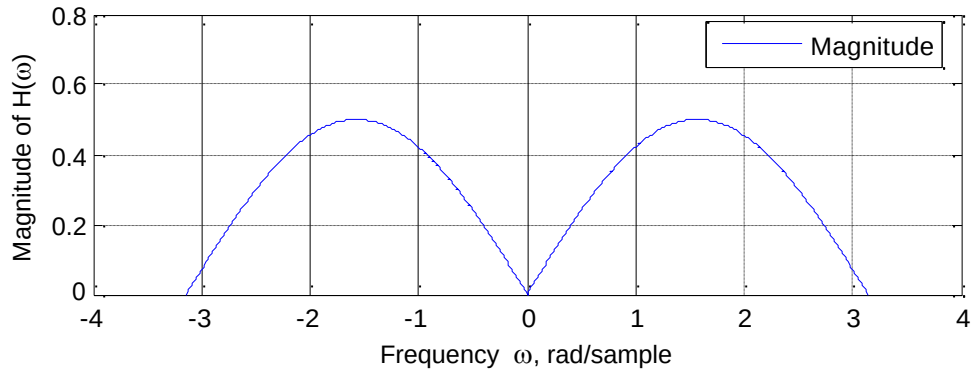
(C) The sequence *h(n)* = {0.25, 0.5, –0.25} is not symmetric.

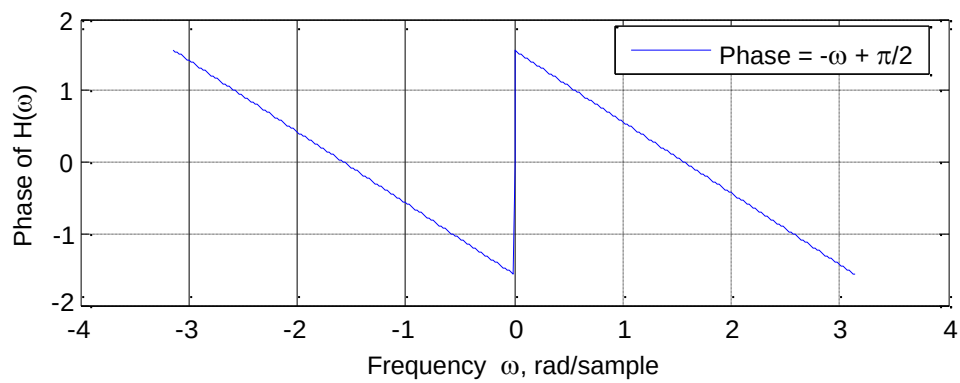
```
%Magnitude and phase response of h(n) = {0.25, 0.5, –0.25}
%Filter coefficients – This is not odd symmetry
b3=[0.25, 0.5, -0.25],
a=[1]
w=-pi: pi/256: pi;
Hw3=freqz(b3, a, w);
subplot(2, 1, 1), plot(w, abs(Hw3)); legend ('Magnitude');
xlabel('Frequency \omega, rad/sample'), ylabel('Magnitude of H(\omega)'); grid
subplot(2, 1, 2), plot(w, angle(Hw3)); legend ('Nonlinear Phase');
xlabel('Frequency \omega, rad/sample'), ylabel('Phase of H(\omega)'); grid
```



(D) The sequence *h(n)* = {0.25, 0, –0.25} has odd symmetry.

```
%Magnitude and phase response of h(n) = {0.25, 0, –0.25}
%Filter coefficients – This is odd symmetry
b3=[0.25, 0, -0.25],
a=[1]
w=-pi: pi/256: pi;
Hw3=freqz(b3, a, w);
subplot(2, 1, 1), plot(w, abs(Hw3)); legend ('Magnitude');
xlabel('Frequency \omega, rad/sample'), ylabel('Magnitude of H(\omega)'); grid
subplot(2, 1, 2), plot(w, angle(Hw3)); legend ('Phase = -\omega + \pi/2 ');
xlabel('Frequency \omega, rad/sample'), ylabel('Phase of H(\omega)'); grid
```





Frequency response

Realization of linear phase FIR filters An important special subset of FIR filters has a *linear phase* characteristic. Linear phase results if the *impulse response is symmetric about its center*. For a causal filter whose impulse response begins at 0 and ends at N–1, this symmetry is expressed thus

Even: $h(n) = h(N-1-n)$, for $n = 0, 1, \ldots, (N-1)$ – a total of N points
Odd: $h(n) = -h(N-1-n)$, for $n = 0, 1, \ldots, (N-1)$ – a total of N points

This symmetry allows the transfer function to be rewritten so that only half the number of multiplications is required for the resulting realization.

Linear phase – phase and delay distortion Assume a low pass filter with frequency response $H(e^{j\omega})$ given by

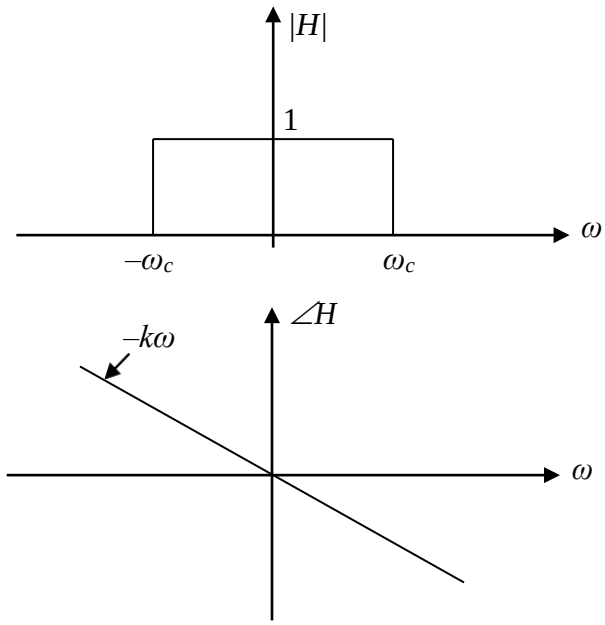
$$H(e^{j\omega}) = \begin{cases} 1e^{-j\omega k}, & |\omega| < \omega_c \\ 0, & \omega_c < |\omega| < \pi \end{cases}$$

where k is an integer. This is a linear phase filter with the slope of the phase "curve" in the pass band being $-k$. Let $X(e^{j\omega})$ represent the Fourier transform of an input sequence $x(n)$. Then the transform of the output sequence $y(n)$ is given by $Y(e^{j\omega}) = X(e^{j\omega}) \cdot H(e^{j\omega})$. If $X(e^{j\omega})$ is entirely within the pass band of $H(e^{j\omega})$ then

$$Y(e^{j\omega}) = X(e^{j\omega}) \cdot e^{-j\omega k}$$

So the output signal $y(n)$ can be obtained as the inverse F-transform of $Y(e^{j\omega})$ as

$$y(n) = x(n-k), \text{ a delayed version of } x(n)$$

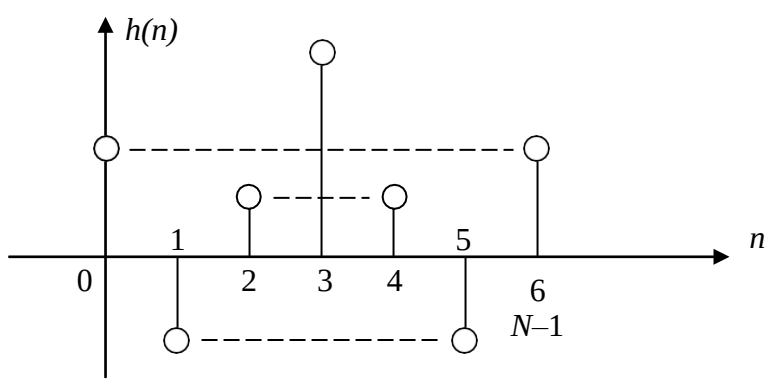


Thus the linear phase filter did not alter the shape of the original signal, simply translated (delayed) it by k samples. If the phase response had not been linear, the output signal would have been a distorted version of $x(n)$.

It can be shown that a causal IIR filter cannot produce a linear phase characteristic and that only special forms of causal FIR filters can give linear phase.

Theorem If $h(n)$ represents the impulse response of a discrete time system, a necessary and sufficient condition for linear phase is that $h(n)$ have a finite duration N, and that it be symmetric about its midpoint.

Example 4.3.1 (a) For the FIR filter of length $N = 7$ with impulse response $h(n)$ let $h(n) = h(N-1-n)$. Show that the filter has a linear phase characteristic. **(b)** Repeat for $N = 8$.



Solution (a) For $N = 7$, the positive symmetry relation $h(n) = h(N-1-n)$ leads to $h(n) = h(6-n)$ which means that $h(0) = h(6)$, $h(1) = h(5)$, and $h(2) = h(4)$, as shown in figure above.

$$H(z) = \sum_{n=0}^{6} h(n) z^{-n} \qquad \text{and} \qquad H(e^{j\omega}) = H(z)\big|_{z=e^{j\omega}} = \sum_{n=0}^{6} h(n) e^{-j\omega n}$$

$$\begin{aligned} H(e^{j\omega}) &= h(0) + h(1)\, e^{-j\omega} + h(2)\, e^{-j2\omega} + h(3) e^{-j3\omega} \\ &\quad + h(4)\, e^{-j4\omega} + h(5) e^{-j5\omega} + h(6)\, e^{-j6\omega} \\ &= e^{-j3\omega} \{ h(0) e^{j3\omega} + h(1) e^{j2\omega} + h(2) e^{j\omega} + h(3) \\ &\quad + h(4) e^{-j\omega} + h(5)\, e^{-j2\omega} + h(6) e^{-j3\omega} \} \end{aligned}$$

Since $h(0) = h(6)$, etc., we can write

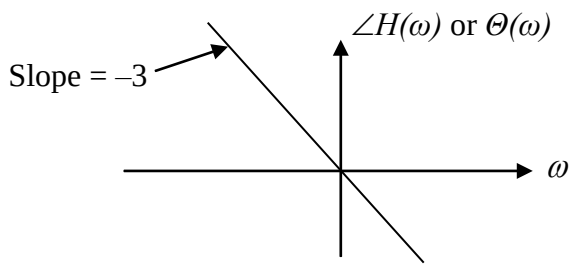
$$\begin{aligned} H(e^{j\omega}) &= e^{-j3\omega} \{ h(0)(e^{j3\omega} + e^{-j3\omega}) + h(1)(e^{j2\omega} + e^{-j2\omega}) + h(2)(e^{j\omega} + e^{-j\omega}) + h(3) \} \\ &= e^{-j3\omega} \{ \underbrace{2h(0)\cos 3\omega}_{= a(3)} + \underbrace{2h(1)\cos 2\omega}_{= a(2)} + \underbrace{2h(2)\cos\omega}_{= a(1)} + \underbrace{h(3)}_{= a(0)} \} \\ &= e^{-j3\omega} \sum_{k=0}^{3} a(k) \cos k\omega \text{, with } a(0) = h(3) \text{ and } a(k) = 2h(3-k),\ k = 1, 2, 3 \end{aligned}$$

The coefficients, in general, are given by

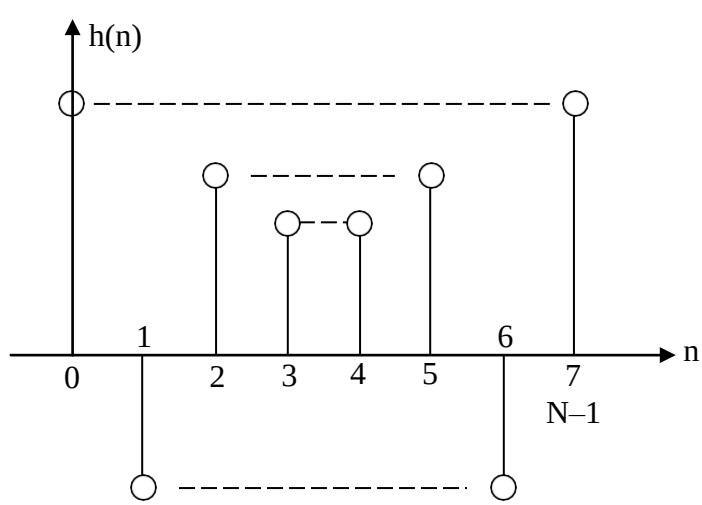
$$a(0) = h\left(\frac{N-1}{2}\right) \text{ and } a(k) = 2h\left(\frac{N-1}{2} - k\right), \text{ for } k = 1, 2, \ldots, (N-1)/2$$

$$H(e^{j\omega}) = \pm \left| H(e^{j\omega}) \right| e^{j\angle H(e^{j\omega})} = e^{-j3\omega} \sum_{k=0}^{3} a(k) \cos k\omega$$

where $\pm \left| H(e^{j\omega}) \right| = \sum_{k=0}^{3} a(k) \cos k\omega$ and $\angle H(e^{j\omega}) = \Theta(\omega) = -3\omega$. The phase response is obviously linear, with slope = –3 = – (N– 1)/2 which means that the delay is an integer number of samples.



(b) For N = 8, the positive symmetry relation $h(n) = h(N-1-n)$ leads to $h(n) = h(7-n)$, which means $h(0) = h(7)$, $h(1) = h(6)$, $h(2) = h(5)$, and $h(3) = h(4)$ as shown in figure below.



$$H(z) = \sum_{n=0}^{7} h(n) z^{-n} \qquad \text{and} \qquad H(e^{j\omega}) = H(z)\big|_{z=e^{j\omega}} = \sum_{n=0}^{7} h(n) e^{-j\omega n}$$

$$\begin{aligned} H(e^{j\omega}) &= h(0) + h(1)\, e^{-j\omega} + h(2)\, e^{-j2\omega} + h(3) e^{-j3\omega} \\ &\quad + h(4)\, e^{-j4\omega} + h(5) e^{-j5\omega} + h(6)\, e^{-j6\omega} + h(7) e^{-j7\omega} \\ &= e^{-j7\omega/2} \{ h(0) e^{j7\omega/2} + h(1)\, e^{j5\omega/2} + h(2)\, e^{j3\omega/2} + h(3) e^{j\omega/2} \\ &\quad + h(4)\, e^{-j\omega/2} + h(5)\, e^{-j3\omega/2} + h(6) e^{-j5\omega/2} + h(7) e^{-j7\omega/2} \} \end{aligned}$$

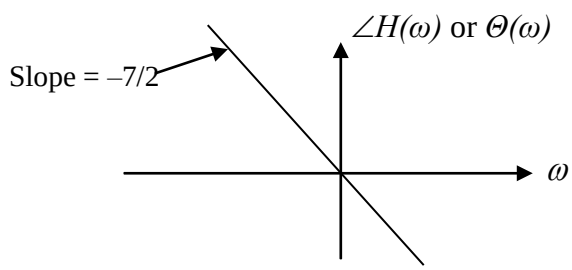
Since $h(0) = h(7)$, etc., we can write

$$\begin{aligned} H(e^{j\omega}) &= e^{-j7\omega/2} \{ h(0)(e^{j7\omega/2} + e^{-j7\omega/2}) + h(1)\,(e^{j5\omega/2} + e^{-j5\omega/2}) \\ &\quad + h(2)(e^{j3\omega/2} + e^{-j3\omega/2}) + h(3)(e^{j\omega/2} + e^{-j\omega/2}) \} \\ &= e^{-j7\omega/2} \Big[\underbrace{2h(0) \cos\left(\frac{7\omega}{2}\right)}_{= b(4)} + \underbrace{2h(1) \cos\left(\frac{5\omega}{2}\right)}_{= b(3)} + \underbrace{2h(2) \cos\left(\frac{3\omega}{2}\right)}_{= b(2)} + \underbrace{2h(3) \cos\left(\frac{\omega}{2}\right)}_{= b(1)} \Big] \end{aligned}$$

With $b(k) = 2h((N/2) - k)$, for $k = 1, 2, \ldots, N/2$, we can write

$$H(e^{j\omega}) = \pm \left| H(e^{j\omega}) \right| e^{j\angle H(e^{j\omega})} = e^{-j7\omega/2} \sum_{k=1}^{4} b(k) \cos[(k - 1/2)\omega]$$

where $\pm \left| H(e^{j\omega}) \right| = \sum_{k=1}^{4} b(k) \cos[(k - 1/2)\omega]$ and $\angle H(e^{j\omega}) = \Theta(\omega) = -7\omega/2$. The phase, $\angle H(e^{j\omega})$, is clearly linear. However, the slope of the phase curve is (–7/2), which is not an integer. The non-integer delay will cause the values of the sequence to be changed, which, in some cases, may be undesirable.



Implementation For a causal filter whose impulse response has even symmetry:

$$h(n) = h(N-1-n), \qquad \text{for } n = 0, 1, \ldots, (N-1) \text{ – a total of } N \text{ points}$$

the transfer function $H(z) = \mathcal{Z}\{h(n)\} = \sum_{n=0}^{N-1} h(n) z^{-n}$ can be written, depending on whether N is even or odd, as follows.

For even *N* The difference equation is derived starting from $H(z)$,

$$H(z) = \sum_{n=0}^{(N/2)-1} h(n)\left(z^{-n} + z^{-(N-1-n)}\right)$$

Since $Y(z) = H(z)\,X(z)$, we can write

$$Y(z) = \left(\sum_{n=0}^{(N/2)-1} h(n)\left(z^{-n} + z^{-(N-1-n)}\right)\right) X(z)$$

$$= h(0)\left(1 + z^{-(N-1)}\right) X(z) + h(1)\left(z^{-1} + z^{-(N-2)}\right) X(z)$$

$$+ \ldots + h\left(\frac{N}{2} - 1\right)\left(z^{-\left(\frac{N}{2}-1\right)} + z^{-\frac{N}{2}}\right) X(z)$$

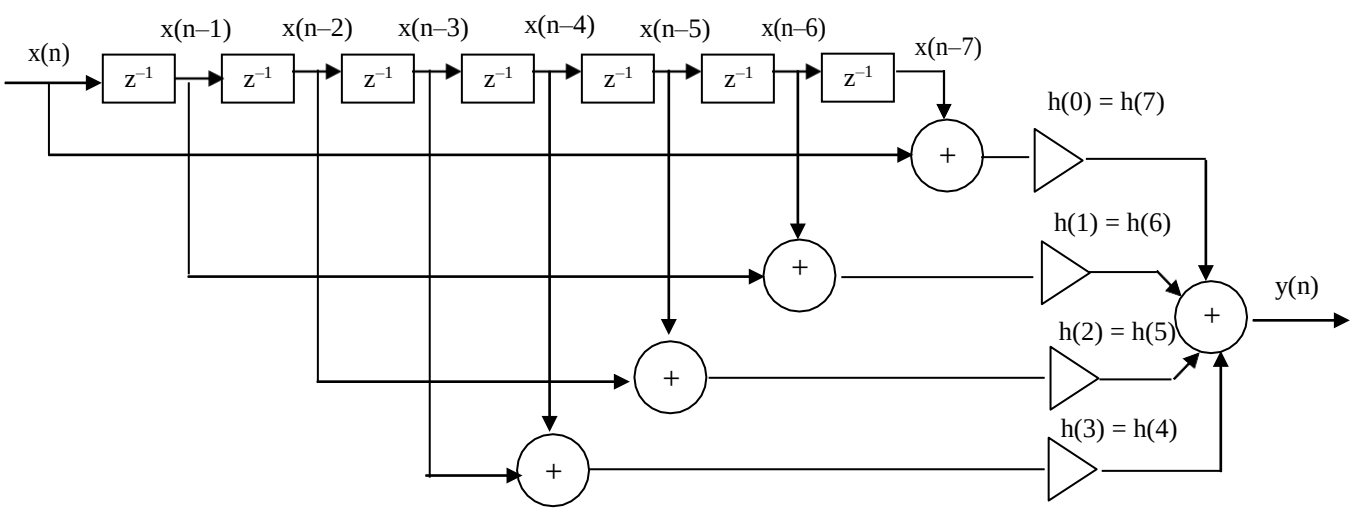
Taking the inverse z-transform of the above we get y(n) as

$$y(n) = h(0)\left[x(n) + x\left(n - N - 1\right)\right] + h(1)\left[x(n-1) + x\left(n - N - 2\right)\right]$$

$$+ \ldots + h\left(\frac{N}{2} - 1\right)\left[x\left(n - \left(\frac{N}{2} - 1\right)\right) + x\left(n - \frac{N}{2}\right)\right]$$

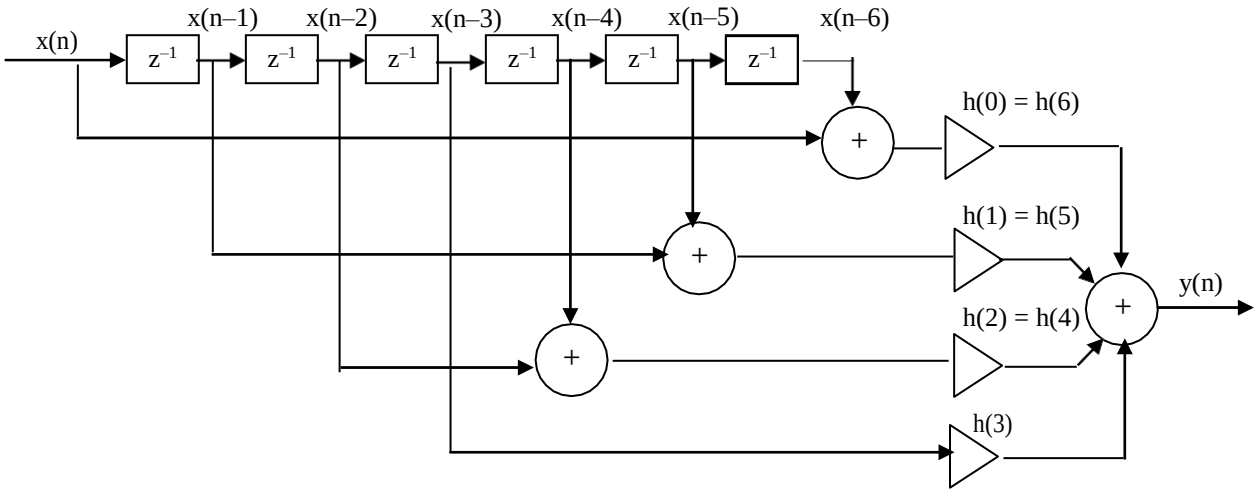
The delayed versions of $x(n)$ are added in pairs and then multiplied by coefficients $h(.)$. This is shown in figure below for $N = 8$. Note that there are an odd number (= 7) of delay elements. There are $N/2 = 4$ multiplications and $(N/2) + 1 = 4 + 1 = 5$ adders (actually the number of two-operand additions is 4 + 3 = 7).

Figure for $N = 8$



For odd *N* We need not derive the equations (they would be necessary if we were writing a computer program to automate it). For $N = 7$, there are $N - 1 = 6$ delay elements – an even number of delay elements. There are $(N + 1)/2 = (7 + 1)/2 = 4$ multiplications and 4 adders (the number of two-operand additions is 6).

Figure for $N = 7$



Properties of FIR digital filters The sinusoidal steady state transfer function of a digital filter is periodic in the sampling frequency. We have

$$H(e^{j\omega}) = H(z)\big|_{z=e^{j\omega}} = \sum_{n=-\infty}^{\infty} h(n)e^{-j\omega n}$$

in which *h(n)* represents the terms of the unit pulse response. The above expression can be decomposed into real and imaginary components by writing

$$H(e^{j\omega}) = \sum_{n=-\infty}^{\infty} h(n)\cos\omega n - j\sum_{n=-\infty}^{\infty} h(n)\sin\omega n = H_R(\omega) + j\,H_I(\omega)$$

where the real and imaginary parts of the transfer function are given by

$$H_R(\omega) = \sum_{n=-\infty}^{\infty} h(n)\cos\omega n \qquad \text{and} \qquad H_I(\omega) = -\sum_{n=-\infty}^{\infty} h(n)\sin\omega n$$

These expressions for $H_R(\omega)$ and $H_I(\omega)$ show that

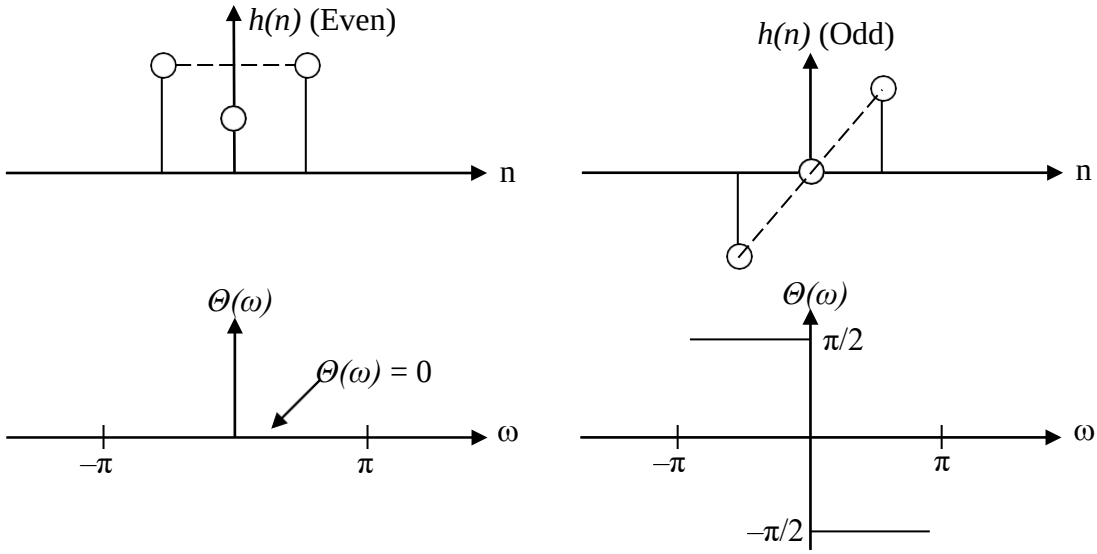
1. $H_R(\omega)$ is an even function of frequency and $H_I(\omega)$ is an odd function of frequency.
2. If *h(n)* is an even sequence, the imaginary part of the transfer function, $H_I(\omega)$, will be zero. (The even sequence, *h(n)*, multiplied by the odd sequence $\sin \omega n$ will yield an odd sequence. An odd sequence summed over symmetric limits yields zero.) In this case

$$H(e^{j\omega}) = \sum_{n=-\infty}^{\infty} h(n)\cos\omega n = H_R(\omega)$$

3. Similarly, if *h(n)* is an odd sequence, the real part of the transfer function, $H_R(\omega)$, will be zero

$$H(e^{j\omega}) = -j\sum_{n=-\infty}^{\infty} h(n)\sin\omega n = j\,H_I(\omega)$$

Thus an even unit pulse response yields a real-valued transfer function and an odd unit pulse response yields on imaginary-valued transfer function. Recall that a real transfer function has a phase shift of 0 or $\pm\pi$ radians, while an imaginary transfer function has a phase shift of $\pm\pi/2$ radians as shown in figures below. So, by making the unit pulse response either even or odd, we can generate a transfer function that is either real or imaginary.



Two types of applications In designing digital filters we are usually interested in one of the following two situations:

1. **Filtering** We are interested in the amplitude response of the filter (e.g., low pass, band pass, etc.) without phase distortion. This is realized by using a real valued transfer function, i.e., $H(e^{j\omega}) = H_R(\omega)$, with $H_I(\omega) = 0$.
2. **Filtering plus quadrature phase shift** These applications include integrators, differentiators, and Hilbert transform devices. For all of these the desired transfer function is imaginary, i.e., $H(e^{j\omega}) = j\,H_I(\omega)$, with $H_R(\omega) = 0$

FIR Filter Design Procedure

1. Decide whether $H_R(\omega)$ or $H_I(\omega)$ is to be set equal to zero. Typically,
   - $H_d(\omega) = H_R(\omega) + j\,0$ for filtering, and
   - $H_d(\omega) = 0 + j\,H_I(\omega)$ for integrators, differentiators and Hilbert transformers
2. Expand $H_d(\omega)$ into Fourier series $h_d(n)$. This is the desired impulse response.
3. Decide on the length N of the impulse response duration. Truncate the sequence $h_d(n)$ to N samples $\{h_t(n), n = -(N-1)/2 \text{ to } (N-1)/2\}$. Even values of N result in delays of half-sample periods; odd values of N avoid this problem.
4. Apply window function $\{w(n), n = -(N-1)/2 \text{ to } (N-1)/2\}$
5. Find the transfer function $H(z) = z^{-(N-1)/2}\,H_t(z)$ and the frequency response $H(\omega)$. If not satisfactory the value of N may have to be increased or a different window function may be tried.

Phase delay and group delay If we consider a signal that consists of several frequency components (such as a speech waveform or a modulated signal) the **phase delay** of the filter is

the amount of time delay each frequency component of the signal suffers in going through the filter. Mathematically, the phase delay τ_p is given by secant

$$\tau_p = -\frac{\Theta(\omega)}{\omega}$$

The **group delay** on the other hand is the average time delay the composite signal suffers at each frequency. The group delay τ_g is given by the slope (tangent) at ω

$$\tau_g = -\frac{d\Theta(\omega)}{d\omega}$$

where $\Theta(\omega) = \angle H(e^{j\omega})$ of the filter.

A nonlinear phase characteristic will cause phase distortion, which is undesirable in many applications, for example, music, data transmission, video and biomedicine.

A filter is said to have a linear phase response if its phase response satisfies one of the following relationships:

$$\Theta(\omega) = -k\omega \rightarrow \text{(A)} \quad \text{or} \quad \Theta(\omega) = \beta - k\omega \rightarrow \text{(B)}$$

where k and β are constants. If a filter satisfies equation (A) its group delay and phase delay are the same constant k. It can be shown that for condition (A) to be satisfied the impulse response of the filter must have positive symmetry (aka even symmetry or just symmetry). The phase response in this case is simply a function of the filter length N:

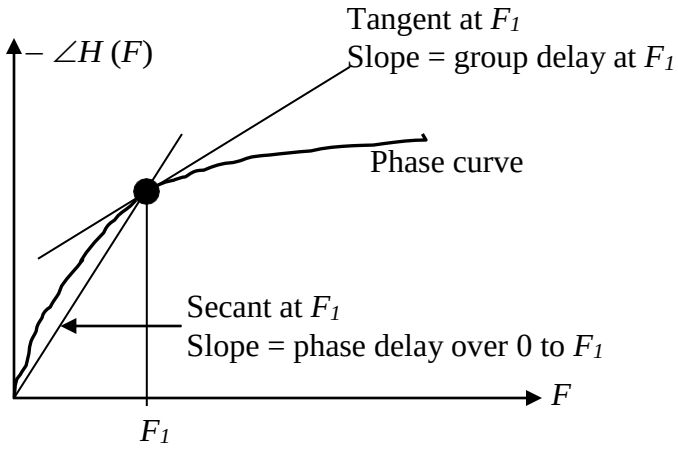
$$h(n) = h(N-1-n), \quad n = 0, 1, 2, \ldots, (N-1)/2 \quad \text{for } N \text{ odd}$$
$$n = 0, 1, 2, \ldots, (N/2) - 1 \quad \text{for } N \text{ even}$$

$$k = (N-1)/2$$

If equation (B) is satisfied the filter will have a constant group delay only. In this case, the impulse response $h(n)$ has negative symmetry (aka odd symmetry or antisymmetry):

$$h(n) = -h(N-1-n)$$
$$k = (N-1)/2$$
$$\beta = \pi/2$$

Analog filter background of phase and group delay Phase delay "at a given frequency" is the slope of the secant line from dc to the particular frequency and is a sort of overall average delay parameter. Phase delay is computed over the *frequency range* representing the major portion of the input signal spectrum (0 to F_1 in the figure below).



The group delay at a given frequency represents the slope of the tangent line at the particular frequency and represents a *local* or *narrow range* (neighborhood of F_1 in the figure) delay parameter.

A case of significance involving both phase delay and group delay is that of a ***narrow band modulated signal***. When a narrow band modulated signal is passed through a filter, the carrier is delayed by a time equal to the phase delay, while the envelope (or intelligence) is delayed by a time approximately equal to the group delay. Since the intelligence (modulating signal) represents the desired information contained in such signals, strong emphasis on good group delay characteristics is often made in filters designed for processing modulated waveforms.

Summary of symmetry

Positive symmetry (or just **"symmetry"** or **even symmetry** about the middle) is characterized by $h(n) = h(N-1-n)$. Show that for positive symmetry

a) For N odd (Type I):

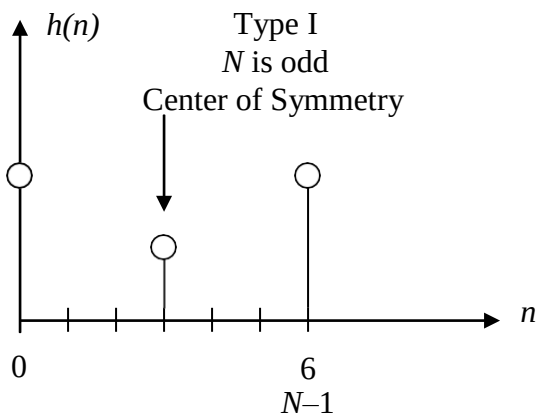
$$H(e^{j\omega}) = e^{-j\omega(N-1)/2} \sum_{k=0}^{(N-1)/2} a(k) \cos k\omega$$

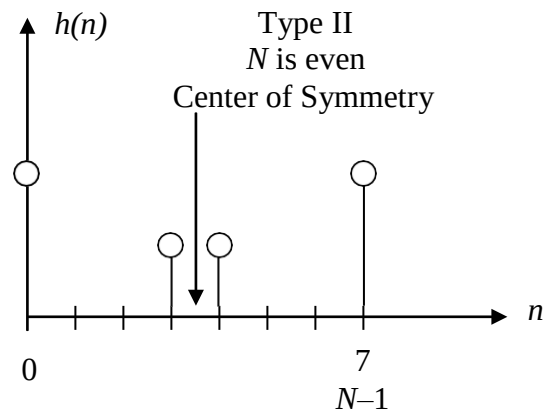
$$a(0) = h\left(\frac{N-1}{2}\right) \;\&\; a(k) = 2h\left(\frac{N-1}{2} - k\right), \; k \neq 0$$

b) For N even (Type II):

$$H(e^{j\omega}) = e^{-j\omega(N-1)/2} \sum_{k=1}^{N/2} b(k) \cos[(k - 1/2)\omega]$$

$$b(k) = 2h\left(\frac{N}{2} - k\right)$$





Summary of symmetry, cont'd

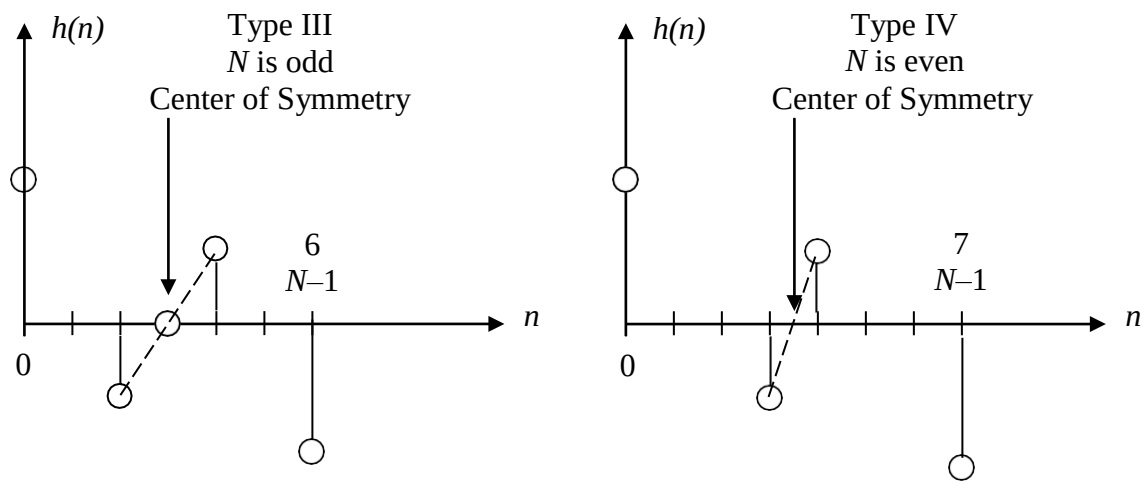
Negative symmetry (or **"antisymmetry"** or **odd symmetry** about the middle) is characterized by $h(n) = -h(N-1-n)$. Show that for negative symmetry

a) For N odd (Type III): $$H(e^{j\omega}) = e^{-j\left(\omega\frac{N-1}{2}-\frac{\pi}{2}\right)(N-1)/2}\sum_{k=0} a(k)\sin k\omega$$

$$a(0) = h\left(\frac{N-1}{2}\right) \;\&\; a(k) = 2h\left(\frac{N-1}{2}-k\right),\ k \neq 0$$

b) For N even (Type IV): $$H(e^{j\omega}) = e^{-j\left(\omega\frac{N-1}{2}-\frac{\pi}{2}\right)N/2}\sum_{k=1} b(k)\sin[(k-1/2)\omega]$$

$$b(k) = 2h\left(\frac{N}{2}-k\right)$$



Qualitative nature of symmetry

Type I Positive symmetry, N is odd. To illustrate take $N = 5$:

$$H(e^{j\omega}) = e^{-j\omega(N-1)/2}\sum_{k=0}^{(N-1)/2} a(k)\cos k\omega = e^{-j2\omega}\sum_{k=0}^{2} a(k)\cos k\omega$$

$$= e^{-j2\omega}[a(0) + a(1)\cos\omega + a(2)\cos 2\omega]$$

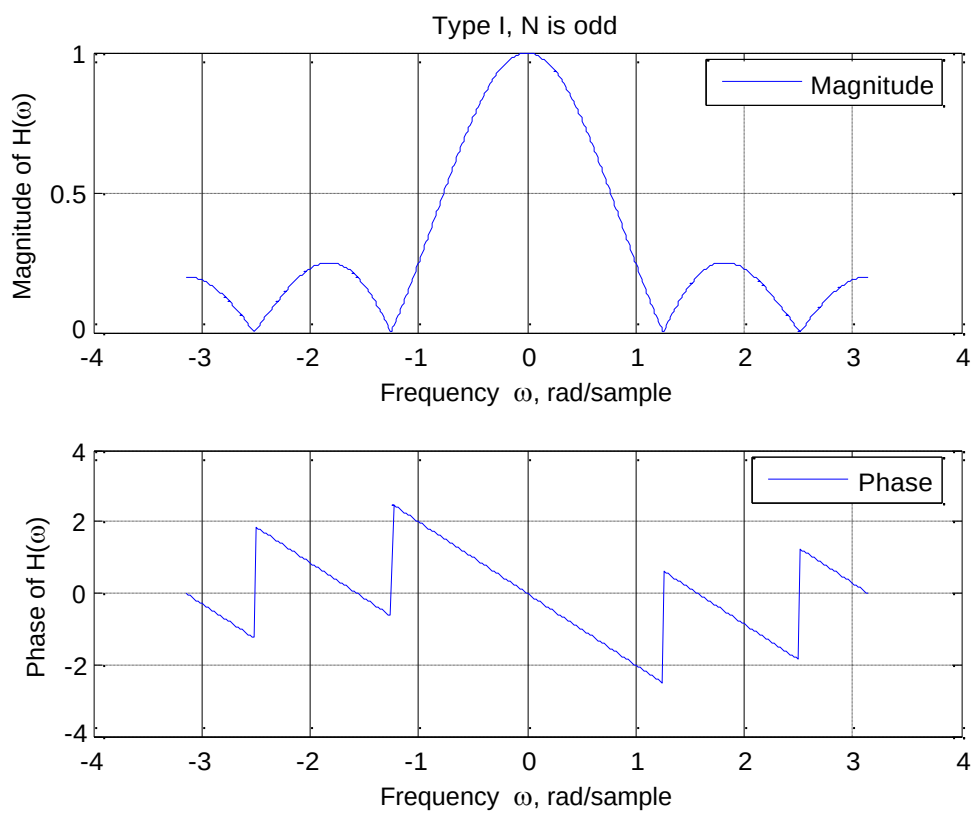
We have to add up *a(0)*, and the two cosine terms. It is clear that at $\omega = 0$ all the cosine terms are at their positive peak, so that when added the response of $|H(e^{j\omega})|$ vs. ω would indicate a low pass filter. Consider

$$y(n) = \frac{x(n) + x(n-1) + x(n-2) + x(n-3) + x(n-4)}{5}$$

$$H(e^{j\omega}) = H(z)\Big|_{z=e^{j\omega}} = \frac{1 + e^{-j\omega} + e^{-j2\omega} + e^{-j3\omega} + e^{-j4\omega}}{5}$$

$$= e^{-j2\omega}\left(\frac{e^{j2\omega} + e^{j\omega} + 1 + e^{-j\omega} + e^{-j2\omega}}{5}\right) = \left(\frac{1 + 2\cos\omega + 2\cos 2\omega}{5}\right)e^{-j2\omega}$$

```
%Frequency response of moving average filter h(n) = {0.2, 0.2, 0.2, 0.2, 0.2}
b5 = [0.2, 0.2, 0.2, 0.2, 0.2], a = [1]
w=-pi: pi/256: pi; Hw5=freqz(b5, a, w);
subplot(2, 1, 1), plot(w, abs(Hw5)); legend ('Magnitude'); title ('Type I, N is odd');
xlabel('Frequency \omega, rad/sample'), ylabel('Magnitude of H(\omega)'); grid
subplot(2, 1, 2), plot(w, angle(Hw5)); legend ('Phase');
xlabel('Frequency \omega, rad/sample'), ylabel('Phase of H(\omega)'); grid
```



Type II Positive symmetry, N is even. Take $N = 6$:

$$H(e^{j\omega}) = e^{-j\omega(N-1)/2}\sum_{k=1}^{N/2} b(k)\cos[(k-1/2)\omega] = e^{-j5\omega/2}\sum_{k=1}^{3} b(k)\cos[(k-1/2)\omega]$$

$$= e^{-j5\omega/2}[b(1)\cos\omega/2 + b(2)\cos 3\omega/2 + b(3)\cos 5\omega/2]$$

At $\omega = \pi$, corresponding to half the sampling frequency (maximum possible frequency), all the cosine terms will be zero. Thus this type of filter is unsuitable as a high-pass filter. It should be ok as a low pass filter. Consider

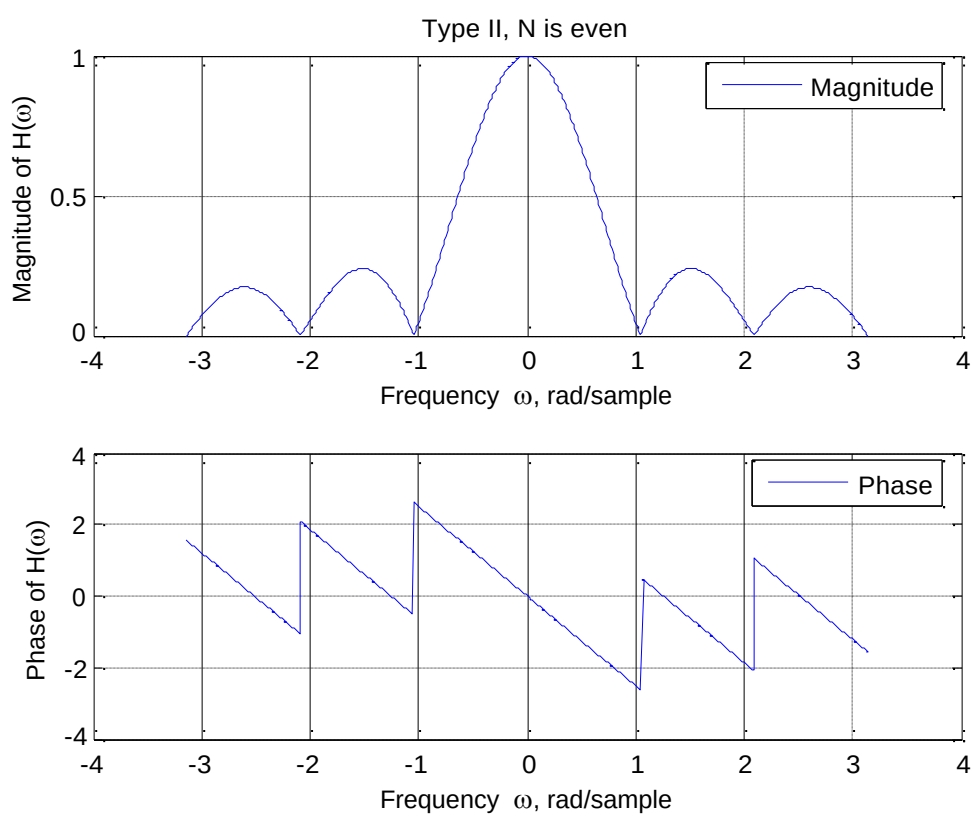
$$y(n) = \frac{x(n) + x(n-1) + x(n-2) + x(n-3) + x(n-4) + x(n-5)}{6}$$

$$H(e^{j\omega}) = H(z)\Big|_{z=e^{j\omega}} = \frac{1 + e^{-j\omega} + e^{-j2\omega} + e^{-j3\omega} + e^{-j4\omega} + e^{-j5\omega}}{6}$$

$$= e^{-j(5/2)\omega}\left(\frac{e^{j(5/2)\omega} + e^{j(3/2)\omega} + e^{j(1/2)\omega} + e^{-j(1/2)\omega} + e^{-j(3/2)\omega} + e^{-j(5/2)\omega}}{6}\right)$$

$$= \left(\frac{\cos(\omega/2) + \cos(3\omega/2) + \cos(5\omega/2)}{3}\right) e^{-j(5/2)\omega}$$

```
%Frequency response of moving average filter h(n) = {1/6, 1/6, 1/6, 1/6, 1/6, 1/6}
b6 = [1/6, 1/6, 1/6, 1/6, 1/6, 1/6], a = [1]
w=-pi: pi/256: pi; Hw6=freqz(b6, a, w);
subplot(2, 1, 1), plot(w, abs(Hw6)); legend ('Magnitude');
title ('Type II, N is even');
xlabel('Frequency \omega, rad/sample'), ylabel('Magnitude of H(\omega)'); grid
subplot(2, 1, 2), plot(w, angle(Hw6)); legend ('Phase');
xlabel('Frequency \omega, rad/sample'), ylabel('Phase of H(\omega)'); grid
```



Type III Negative symmetry, N is odd. This introduces a 90^0 (= $\pi/2$) phase shift. Because of the sine terms $|H|$ is always zero at $\omega = 0$ and at $\omega = \pi/2$ (half the sampling frequency). Therefore the filter is unsuitable as a low pass or a high pass filter. To illustrate take $N = 5$ and

$$h(n) = \{0.2, 0.2, 0, -0.2, -0.2\}$$

$$H(e^{j\omega}) = e^{-j\left(\omega\frac{N-1}{2}-\frac{\pi}{2}\right)}\sum_{k=0}^{(N-1)/2} a(k)\sin k\omega = e^{-j\left(\omega\frac{5-1}{2}-\frac{\pi}{2}\right)}\sum_{k=0}^{(5-1)/2} a(k)\sin k\omega$$

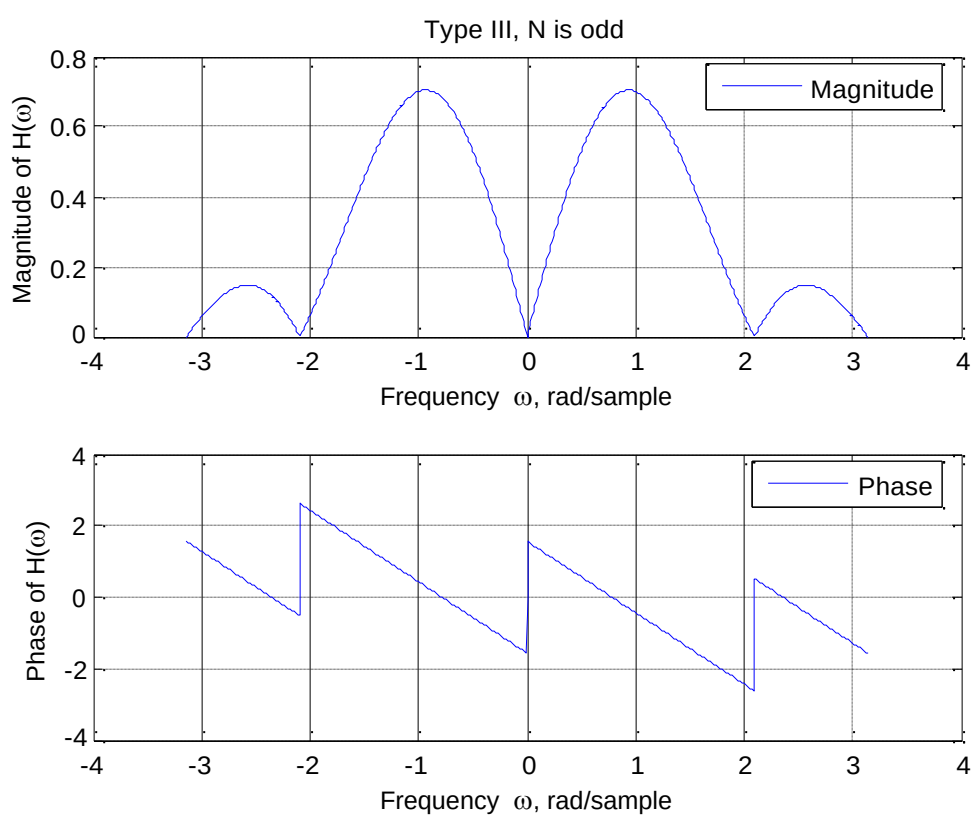
$$= e^{-j2\omega+j(\pi/2)}\sum_{k=0}^{2} a(k)\sin k\omega$$

$$a(0) = h\left(\frac{N-1}{2}\right) = h(2) = 0$$

$$a(k) = 2h\left(\frac{N-1}{2}-k\right) = 2h(2-k),\ k \neq 0$$

Etc.

```
%Frequency response of Type III filter, h(n) = {0.2, 0.2, 0, -0.2, -0.2}
b5 = [0.2, 0.2, 0, -0.2, -0.2], a = [1]
w=-pi: pi/256: pi; Hw5=freqz(b5, a, w);
subplot(2, 1, 1), plot(w, abs(Hw5)); legend ('Magnitude');
title ('Type III, N is odd');
xlabel('Frequency \omega, rad/sample'), ylabel('Magnitude of H(\omega)'); grid
subplot(2, 1, 2), plot(w, angle(Hw5)); legend ('Phase');
xlabel('Frequency \omega, rad/sample'), ylabel('Phase of H(\omega)'); grid
```



Type IV Negative symmetry, N is even. This introduces a 90^0 (= $\pi/2$) phase shift. Because of the sine terms $|H|$ is always zero at $\omega = 0$. Therefore the filter is unsuitable as a low pass filter. To illustrate take $N = 6$ and

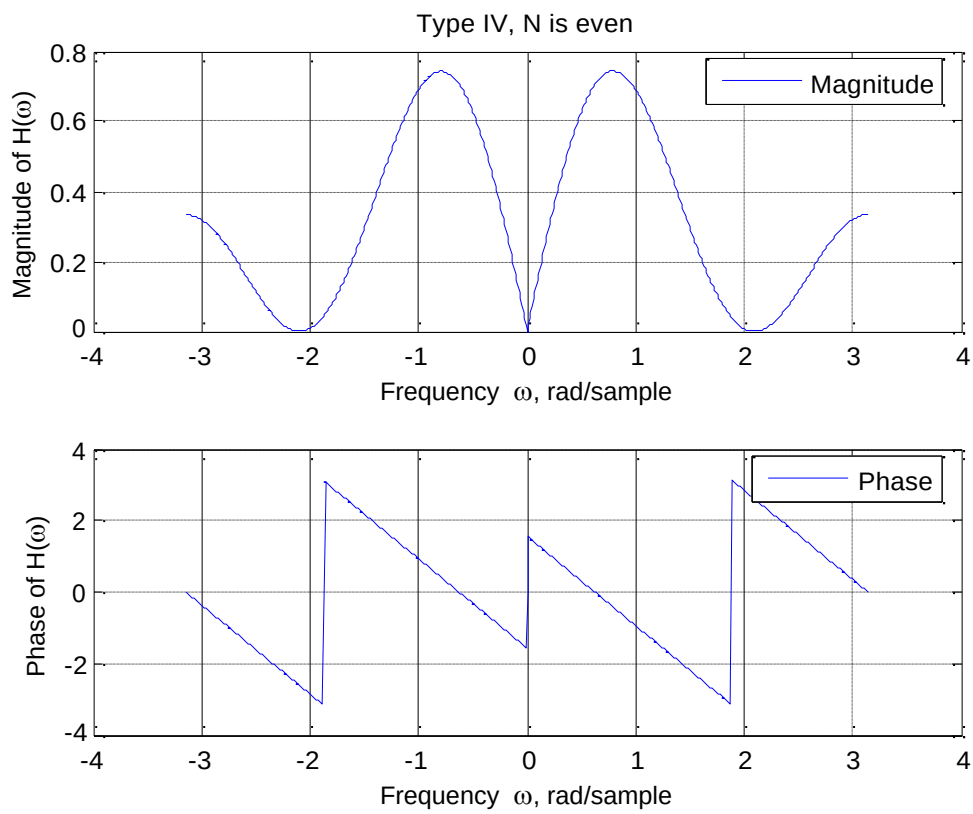
$$h(n) = \{1/6, 1/6, 1/6, -1/6, -1/6, -1/6\}$$

$$H(e^{j\omega}) = e^{-j\left(\frac{\omega N-1}{2}-\frac{\pi}{2}\right)\Big|_{N/2}} \sum_{k=1}^{3} b(k)\sin[(k-1/2)\omega] = e^{-j\left(\frac{\omega 6-1}{2}-\frac{\pi}{2}\right)\Big|_{6/2}} \sum_{k=1} b(k)\sin[(k-1/2)\omega]$$

$$= e^{-j3\omega+j(\pi/2)} \sum_{k=1} b(k)\sin[(k-1/2)\omega] \quad \text{and} \quad b(k) = 2h(3-k),\ k = 1 \text{ to } 3$$

Etc.

```
%Frequency response of Type IV filter h(n) = {1/6, 1/6, 1/6, -1/6, -1/6, -1/6}
b6 = [1/6, 1/6, 1/6, -1/6, -1/6, -1/6], a = [1]
w=-pi: pi/256: pi; Hw6=freqz(b6, a, w);
subplot(2, 1, 1), plot(w, abs(Hw6)); legend ('Magnitude');
title ('Type IV, N is even');
xlabel('Frequency \omega, rad/sample'), ylabel('Magnitude of H(\omega)'); grid
subplot(2, 1, 2), plot(w, angle(Hw6)); legend ('Phase');
xlabel('Frequency \omega, rad/sample'), ylabel('Phase of H(\omega)'); grid
```



Types III and IV are often used to design **differentiators** and **Hilbert transformers** because of the 90^0 phase shift that each one can provide.

The phase delay for Type I and II filters or group delay for all four types of filters is expressible in terms of the number of coefficients of the filter and so can be corrected to give a zero phase or group delay response.

Types I and II: $\tau_p = \tau_g = -\Theta(\omega)/\omega = \left(\frac{N-1}{2}\right)T$

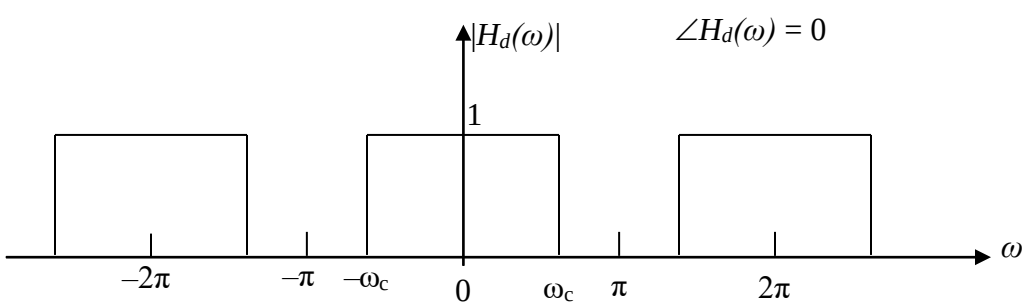
Types III and IV: $\tau_g = -\frac{d\Theta(\omega)}{d\omega} = \left(\frac{N-1}{2}\right)T$

	Magnitude ($\lvert H(\omega)\rvert$) response at		
	$\omega = 0$ rad.	$\omega = \pi$ rad.	
Type I	Max		Low pass
Type II		Zero	OK as LP Not OK as HP filter
Type III	Zero	Zero	90^0 Phase shift
Type IV	Zero		90^0 Phase shift

Design of FIR digital filters – The Fourier series and windowing method

This method of filter design originates with the observation that the sinusoidal steady state transfer function of a digital filter is periodic in the sampling frequency. Since $H(\omega)$ is a continuous and periodic function of ω we can expand it into a Fourier series. The resulting Fourier coefficients are the impulse response, $h(n)$. The major disadvantage is that one cannot easily specify in advance the exact values for pass band and stop band attenuation/ripple levels, so it may be necessary to check several alternate designs to get the required one.

Consider the ideal low pass filter with frequency response $H_d(e^{j\omega})$ or $H_d(\omega)$ as shown below. The subscript d means that it is the *desired* or *ideal* filter.



The impulse response is given by

$$h_d(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(e^{j\omega})\, e^{j\omega n} d\omega = \frac{1}{2\pi}\int_{-\omega_c}^{\omega_c} 1 e^{j\omega n} d\omega$$

$$= \begin{cases} \dfrac{\sin n\omega_c}{\pi n}, & -\infty \le n \le \infty,\ n \ne 0 \\ \dfrac{\omega_c}{\pi}, & n = 0 \end{cases}$$

This is a non-causal infinite impulse response sequence. It is made a finite impulse response sequence by truncating it symmetrically about $n = 0$; it is made causal by shifting the truncated sequence to the right so that it starts at $n = 0$. The shifting results in a time delay and we shall ignore it for now. The truncation results in the sequence $h_t(n)$ where the subscript t means truncation but we shall ignore the subscript

$$h(n) = \begin{cases} h_d(n), & -(N-1)/2 \le n \le (N-1)/2 \\ 0, & \text{otherwise} \end{cases}$$

In general $h(n)$ can be thought of as obtained by multiplying $h_d(n)$ with a window function $w(n)$ as follows

$$h(n) = h_d(n) \,.\, w(n)$$

For the $h(n)$ obtained by simple truncation as above the window function is a rectangular window given by

$$w(n) = \begin{cases} 1, & -(N-1)/2 \le n \le (N-1)/2 \\ 0, & \text{otherwise} \end{cases}$$

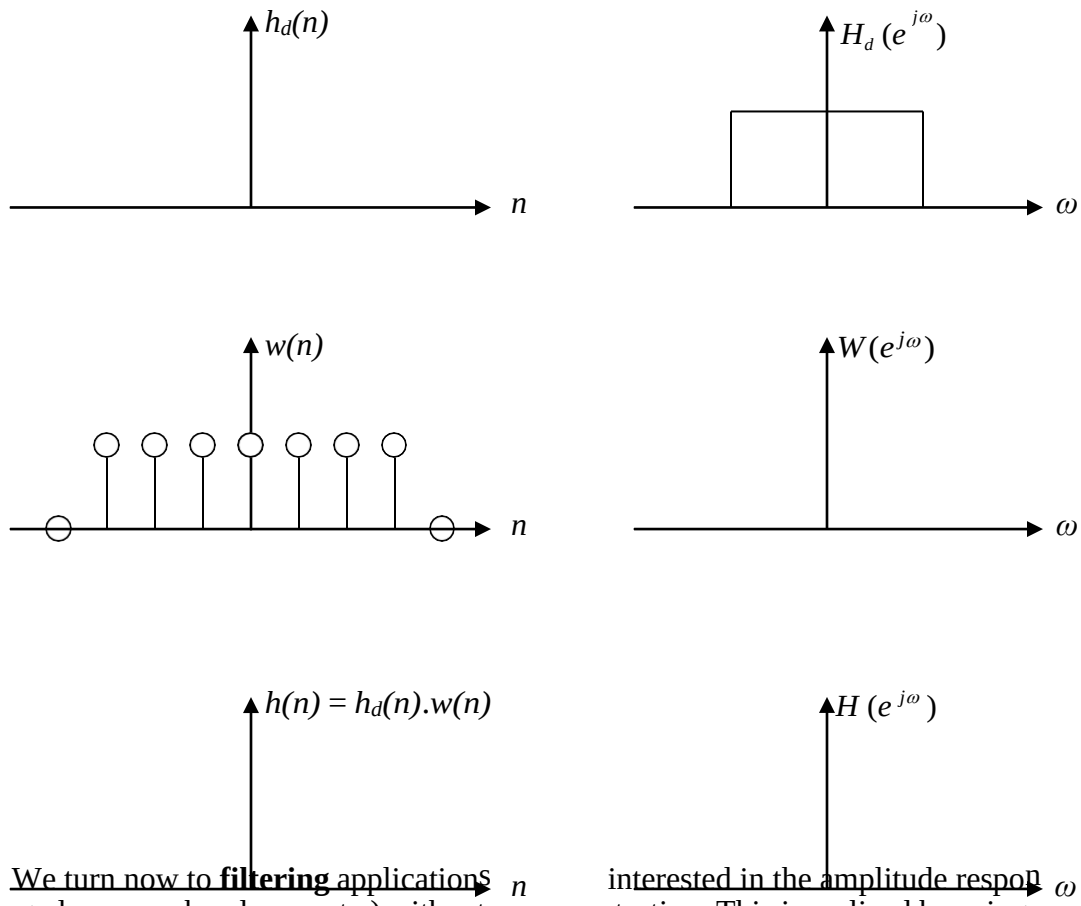
Let $H(e^{j\omega})$, $H_d(e^{j\omega})$ and $W(e^{j\omega})$ represent the Fourier transforms of $h(n)$, $h_d(n)$ and $w(n)$ respectively. Then the frequency response $H(e^{j\omega})$ of the resulting filter is the convolution of $H_d(e^{j\omega})$ and $W(e^{j\omega})$ given by

$$H(e^{j\omega}) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(e^{j\theta})W(e^{j(\omega-\theta)})\,d\theta = H_d(e^{j\omega}) * W(e^{j\omega})$$

The convolution produces a smeared version of the ideal low pass filter. In other words, $H(e^{j\omega})$ is a smeared version of $H_d(e^{j\omega})$. In general, the wider the main lobe of $W(e^{j\omega})$, the more spreading or smearing, whereas the narrower the main lobe (larger N), the closer $H(e^{j\omega})$ comes to $H_d(e^{j\omega})$.

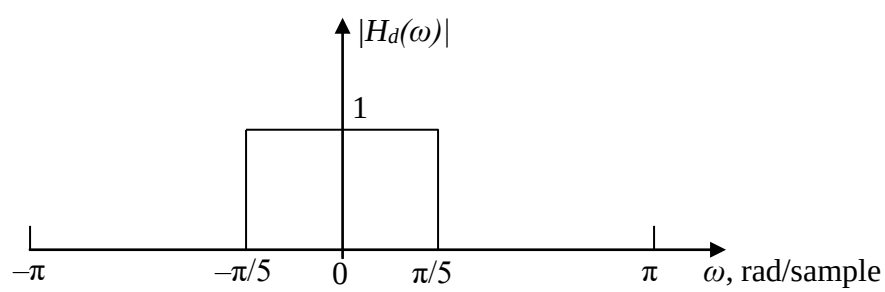
For any arbitrary window the *transition band* of the filter is determined by the width of the *main lobe* of the window. The *side lobes* of the window produce *ripples* in both pass band and stop band.

In general, we are left with a trade-off of making N large enough so that smearing is minimized, yet the number of filter coefficients (= N) is not too large for a reasonable implementation. Some commonly used windows are the rectangular, Bartlett (triangular), Hanning, Hamming, Blackman, and Kaiser windows.



We turn now to **filtering** applications interested in the amplitude respon filter (e.g., low pass, band pass, etc.) without tortion. This is realized by using valued transfer function, i.e., $H(e^{j\omega}) = H_R(\omega)$, with $H_I(\omega) = 0$.

Example 4.4.1 [Design of 9-coefficient LP FIR filter] Design a nine-coefficient (or 9-point or 9-tap) FIR digital filter to approximate an ideal low-pass filter with a cut-off frequency $\omega_c = 0.2\pi$. The magnitude response, $|H_d(\omega)|$, is given below. Take $\angle H_d(\omega) = 0$.



Solution The impulse response of the desired filter is

$$h_d(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(e^{j\omega})\, e^{j\omega n} d\omega = \frac{1}{2\pi}\int_{-0.2\pi}^{0.2\pi} 1.e^{j\omega n}\, d\omega$$

$$= \frac{1}{2\pi}\left[\frac{e^{j\omega n}}{jn}\right]_{-0.2\pi}^{0.2\pi} = \frac{(e^{j0.2\pi n} - e^{-j0.2\pi n})}{2\pi jn} = \frac{\sin 0.2\pi n}{\pi n}$$

Since $h_d(n) \neq 0$ for $n < 0$ this is a noncausal filter (also, it is not BIBO stable – see Unit IV). The rest of the design is aimed at coming up with a noncausal approximation of the above impulse response.

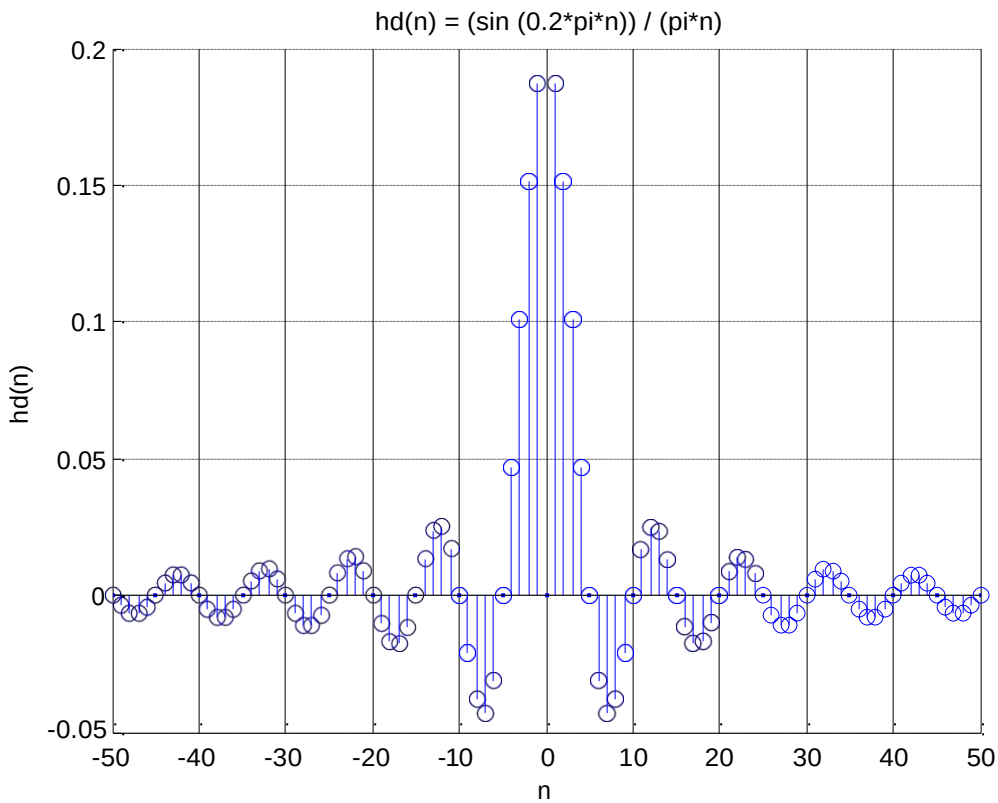
For a rectangular window of length 9, the corresponding impulse response is obtained by evaluating $h_d(n)$ for $-4 \le n \le 4$ on a calculator. In the MATLAB segment below, which generates the $h_d(n)$ coefficients for $-50 \le n \le 50$, division by zero for $n = 0$ causes "NaN" (*Not a Number*), while all the other coefficients are correct. In generating the frequency response $|H(\omega)|$ we copy and paste all the coefficients except for $h_d(0)$ which is entered by hand.

$$h_d(0) = \left.\frac{\frac{d(\sin 0.2\pi n)}{dn}}{\frac{d(\pi n)}{dn}}\right|_{n=0} = \left.\frac{0.2\pi \cos 0.2\pi n}{\pi}\right|_{n=0} = 0.2$$

Aside (MATLAB) The segment below generates and stem-plots the $h_d(n)$ coefficients.

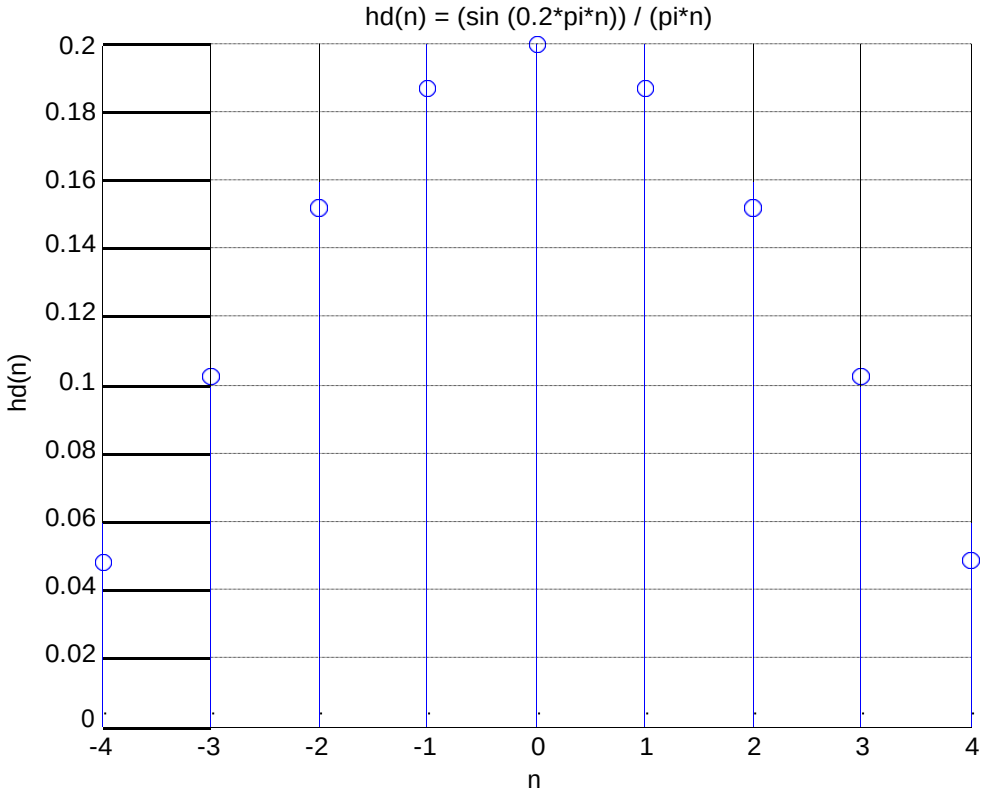
```
%Calculate hdn = (sin (0.2*pi*n)) / (pi*n) and stem plot
n = -50: 50, hdn = (sin(0.2*pi*n)) ./(pi*n), stem(n, hdn)
xlabel('n'), ylabel('hd(n)'); grid; title ('hd(n) = (sin (0.2*pi*n)) / (pi*n)')
```

n = -50 to 50
Warning: Divide by zero.
hdn = -0, -0.0038, -0.0063, -0.0064, -0.0041, 0, 0.0043 0.0070 0.0072,
0.0046 -0, -0.0048 -0.0080 -0.0082 -0.0052 0, 0.0055 0.0092, 0.0095
0.0060 -0, -0.0065 -0.0108 -0.0112 -0.0072 0, 0.0078, 0.0132 0.0138
0.0089 -0, -0.0098 -0.0168 -0.0178 -0.0117 0, 0.0134 0.0233 0.0252
0.0170 -0, -0.0208 -0.0378 -0.0432 -0.0312, 0, ***0.0468 0.1009 0.1514***
0.1871 NaN 0.1871 0.1514 0.1009, 0.0468 0.0000 -0.0312 -0.0432
-0.0378 -0.0208 -0.0000 0.0170 0.0252, 0.0233 0.0134 0.0000 -0.0117
-0.0178 -0.0168 -0.0098 -0.0000 0.0089, 0.0138 0.0132 0.0078 0.0000
-0.0072 -0.0112 -0.0108 -0.0065 -0.0000, 0.0060 0.0095 0.0092 0.0055
0.0000 -0.0052 -0.0082 -0.0080 -0.0048, -0.0000 0.0046 0.0072 0.0070
0.0043 0.0000 -0.0041 -0.0064 -0.0063, -0.0038 -0.0000



Note 1 The segment below is used to get a quick look at the $h_d(n)$ coefficients and their symmetry. The MATLAB problem with $n = 0$ may be avoided by replacing n with $(n – 0.001)$. This will affect the other coefficients very slightly (which is not a serious problem as far as demonstrating the even symmetry of $h_d(n)$) but the accuracy of the coefficients is somewhat compromised in the third or fourth significant digit.

```
%Calculate hdn = (sin (0.2*pi*n)) / (pi*n) and stem plot
n = -4: 4, hdn = (sin(0.2*pi*(n-0.001) ) ) ./(pi*(n-0.001) ), stem(n, hdn),
xlabel('n'), ylabel('hd(n)'); grid; title ('hd(n) = (sin (0.2*pi*n)) / (pi*n)')

hdn = 0.0467, 0.1009, 0.1513, 0.1871, 0.2, 0.1871, 0.1514, 0.1010, 0.0468
```



Note 2 As an alternative to the above, one could write a custom program to calculate *all* coefficients exactly including $h_d(0)$.

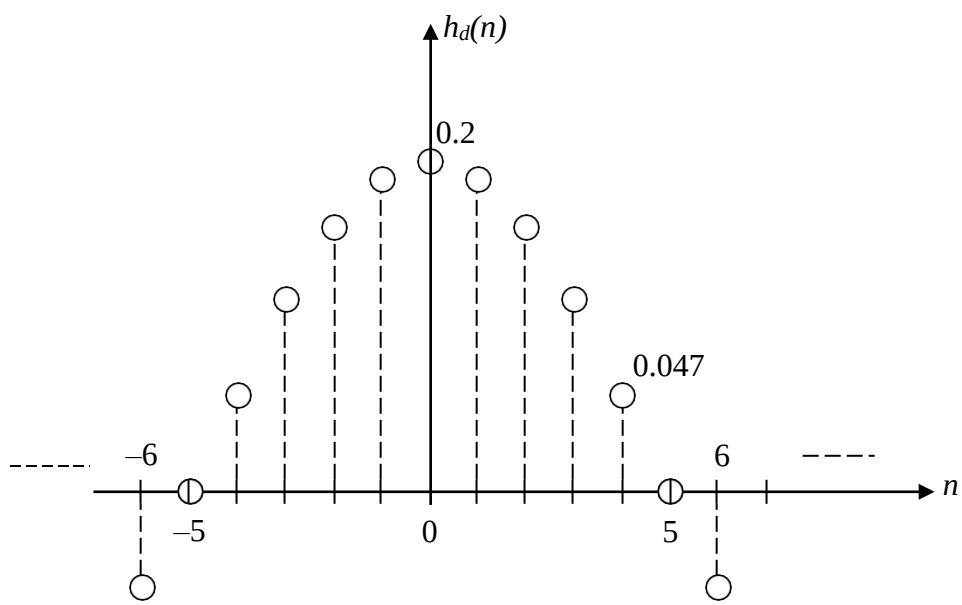
End of Aside

The values are shown in table below:

n =	–4	–3	–2	–1	0	1	2	3	4
$h_t(n)$ =	{0.047,	0.101,	0.151,	0.187,	0.2,	0.187,	0.151,	0.101,	0.047}

By a rectangular window of length 9 we mean that we retain the above 9 values of $h_d(.)$ and *truncate* the rest outside the window. Thus

$$h_t(-4) = 0.047,\ h_t(-3) = 0.101,\ \ldots,\ h_t(0) = 0.2,\ \ldots,\ h_t(4) = 0.047$$



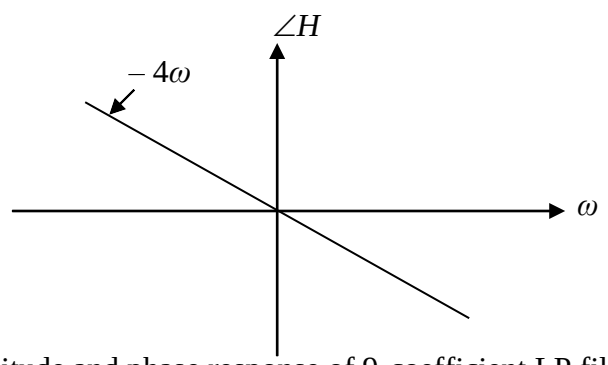
The transfer function of this filter is

$$H_t(z) = \sum_{n=-4}^{4} h_d(n) z^{-n} = 0.047\, z^{4} + 0.101\, z^{3} + 0.151\, z^{2} + 0.187\, z^{1} + 0.2\, z^{0}$$
$$+ 0.187\, z^{-1} + 0.151\, z^{-2} + 0.101\, z^{-3} + 0.047\, z^{-4}$$

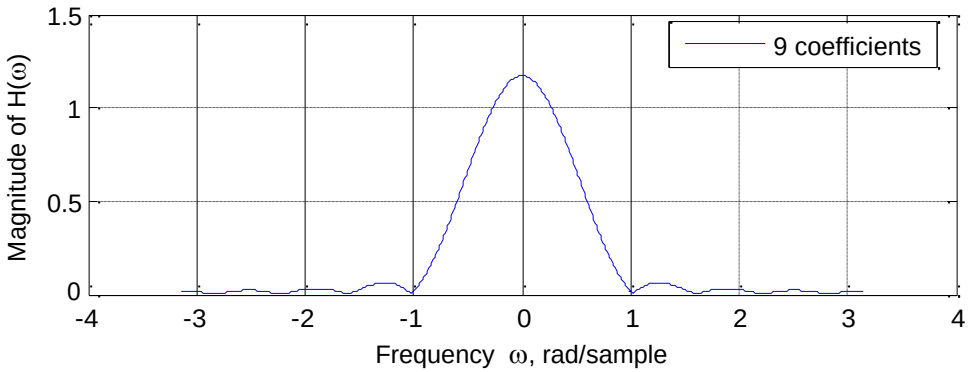
The causal filter is then given by *delaying* the sequence $h_t(n)$ by 4 samples. That is, $h(n) = h_t(n-4)$, and the resulting transfer function is

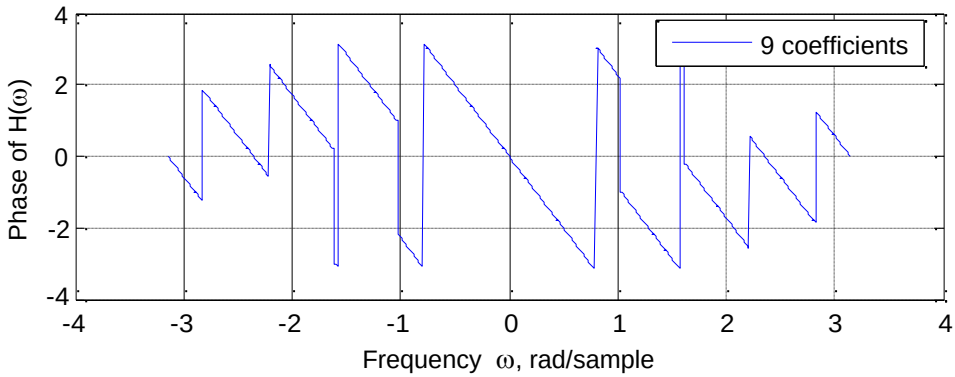
$$H(z) = z^{-4} H_t(z)$$
$$= 0.047\,(1 + z^{-8}) + 0.101\,(z^{-1} + z^{-7}) + 0.151\,(z^{-2} + z^{-6})$$
$$+ 0.187\,(z^{-3} + z^{-5}) + 0.2\, z^{-4}$$

We may obtain the frequency response of this realizable (causal) filter by setting $z = e^{j\omega}$, that is, $H(e^{j\omega}) = H(z)\big|_{z=e^{j\omega}}$. Because of the *truncation* the magnitude $|H|$ will only be approximately equal to $|H_d|$ – Gibbs phenomenon, see comparison of 9 coefficients versus 101 coefficients below. Further, because of the *delay* the phase $\angle H = -4\omega$ whereas, as originally specified, $\angle H_d = 0$. The slope $\dfrac{d\angle H(\omega)}{d\omega} = -4$, showing that the filter introduces a delay of 4 samples.

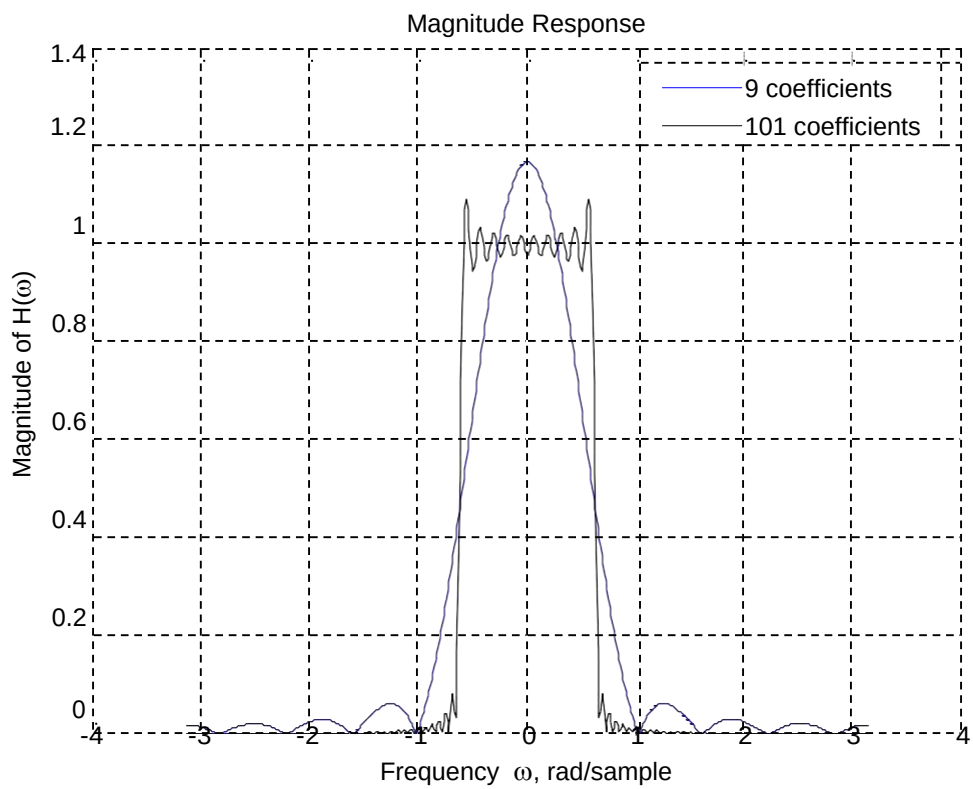



```
%Magnitude and phase response of 9-coefficient LP filter
%Filter coefficients
b9=[0.0468, 0.1009, 0.1514, 0.1871, 0.2, 0.1871, 0.1514, 0.1009, 0.0468],
a=[1]
w=-pi: pi/256: pi;
Hw9=freqz(b9, a, w);
subplot(2, 1, 1), plot(w, abs(Hw9)); legend ('9 coefficients');
xlabel('Frequency \omega, rad/sample'), ylabel('Magnitude of H(\omega)'); grid
subplot(2, 1, 2), plot(w, angle(Hw9)); legend ('9 coefficients');
xlabel('Frequency \omega, rad/sample'), ylabel('Phase of H(\omega)'); grid
```


1.5
1
0.5
0
Magnitude of H(ω)
-4
-3
-2
-1
0
1
2
3
4
Frequency ω, rad/sample
9 coefficients


4
2
0
-2
-4
Phase of H(ω)
-4
-3
-2
-1
0
1
2
3
4
Frequency ω, rad/sample
9 coefficients

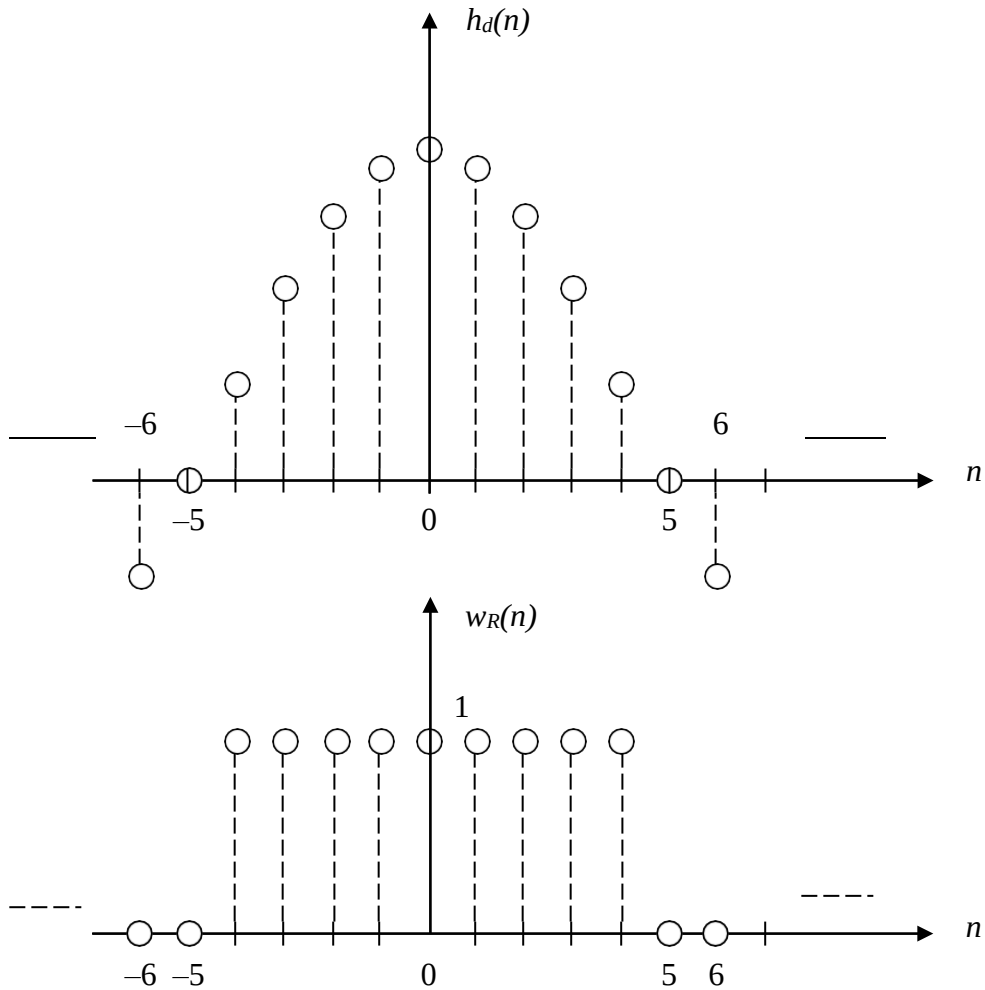
```
%Comparison of 9 coefficients vs. 101 coefficients
%Filter coefficients
b9=[0.0468, 0.1009, 0.1514, 0.1871, 0.2, 0.1871, 0.1514, 0.1009, 0.0468],
a=[1]
b101 = [-0.0 -0.0038 -0.0063 -0.0064 -0.0041 0.0, 0.0043    0.0070    0.0072
0.0046   -0.0, -0.0048   -0.0080   -0.0082   -0.0052    0.0, 0.0055    0.0092,
0.0095    0.0060   -0.0, -0.0065  -0.0108    -0.0112   -0.0072    0.0, 0.0078
0.0132    0.0138    0.0089 -0.0, -0.0098     -0.0168   -0.0178   -0.0117    0.0,
0.0134    0.0233    0.0252    0.0170  -0.0,  -0.0208   -0.0378   -0.0432  -0.0312
0.0, 0.0468    0.1009    0.1514    0.1871       0.2    0.1871    0.1514    0.1009 0.0468
0.0  -0.0312    -0.0432   -0.0378   -0.0208   -0.0    0.0170    0.0252, 0.0233
0.0134    0.0   -0.0117   -0.0178   -0.0168   -0.0098  -0.0    0.0089, 0.0138
0.0132    0.0078       0.0 -0.0072  -0.0112   -0.0108   -0.0065   -0.0, 0.0060
0.0095    0.0092     0.0055    0.0  -0.0052   -0.0082   -0.0080   -0.0048, -0.0
0.0046    0.0072    0.0070    0.0043    0.0   -0.0041   -0.0064   -0.0063, -0.0038  -
0.0]
w=-pi: pi/256: pi;
Hw9=freqz(b9, a, w);
Hw101=freqz(b101, a, w);
plot(w, abs(Hw9), w, abs(Hw101), 'k')
legend ('9 coefficients', '101 coefficients');
title('Magnitude Response');
xlabel('Frequency \omega, rad/sample'), ylabel('Magnitude of H(\omega)'); grid
```



Rectangular window In this example the sequence $h_d(n)$, which extends to infinity on both sides, has been truncated to 9 terms. This truncation process can be thought of as multiplying the

infinitely long sequence by a *window function* called the rectangular window, $w_R(n)$. The figure below shows both $h_d(n)$ and $w_R(n)$ in the undelayed form, that is, symmetrically disposed about $n = 0$.

Specifying the window function The interval over which the window function is defined



depends on whether we first delay $h_d(n)$ and then truncate it or the other way around. If, instead of truncating first and then delaying, we adopt the procedure of first delaying $h_d(n)$ and then truncating it, the window function may be defined over the interval $0 \le n \le N-1$, where N is the number of terms retained. With this understanding the rectangular window, $w_R(n)$, is given below.

$$w_R(n) = \begin{cases} 1, & 0 \le n \le N-1 \\ 0, & \text{elsewhere} \end{cases}$$

If, however, we define $w_R(n)$ symmetrically about $n = 0$, as we do later, we have

$$w_R(n) = \begin{cases} 1, & -(N-1)/2 \le n \le (N-1)/2 \\ 0, & \text{elsewhere} \end{cases}$$

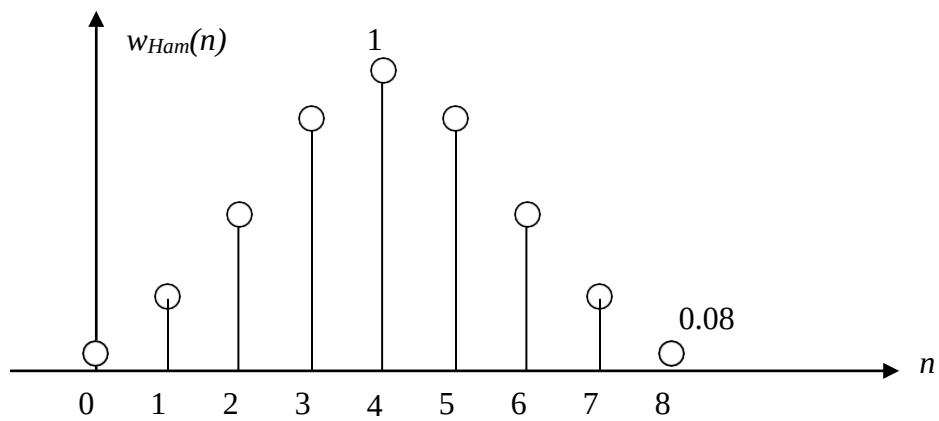
In this case we have implied that N is odd.

Hamming window As an alternative to the rectangular window we shall apply the Hamming window, defined over the interval $0 \le n \le N-1$, by

$$w_{Ham}(n) = \begin{cases} 0.54 - 0.46\cos[2\pi n/(N-1)], & 0 \le n \le N-1 \\ 0, & \text{elsewhere} \end{cases}$$

For $N = 9$, the Hamming window is given by $w_{Ham}(n) = 0.54 - 0.46\cos(\pi n / 4)$, $0 \le n \le 8$

$n =$	0	1	2	3	4	5	6	7	8
$w_{Ham}(n) =$	{0.08,	0.215,	0.54,	0.865,	1,	0.865,	0.54,	0.215,	0.08}



Imagine that we line up this sequence alongside the $h_d(n)$ given earlier. (This means we should imagine $w_{Ham}(n)$ is moved to the left by 4 samples). We then multiply the two sequences at each point to get the windowed sequence, $h_t(n)$:

$n =$	–4	–3	–2	–1	0	1	2	3	4
$h_d(n) =$	{0.047,	0.101,	0.151,	0.187,	0.2,	0.187,	0.151,	0.101,	0.047}
$w(n) =$	{0.08,	0.215,	0.54,	0.865,	1,	0.865,	0.54,	0.215,	0.08}
$h_t(n) =$	{0.00382,	0.0216,	0.0815,	0.1617,	0.2,	0.1617,	0.0815,	0.0216,	0.00382}

The transfer function is given by

$$H_t(z) = 0.00382\, z^{4} + 0.0216\, z^{3} + 0.0815\, z^{2} + 0.1617\, z^{1} + 0.2 + 0.1617\, z^{-1} + 0.0815\, z^{-2} + 0.0216\, z^{-3} + 0.00382\, z^{-4}$$

Delaying by 4 sample periods we get

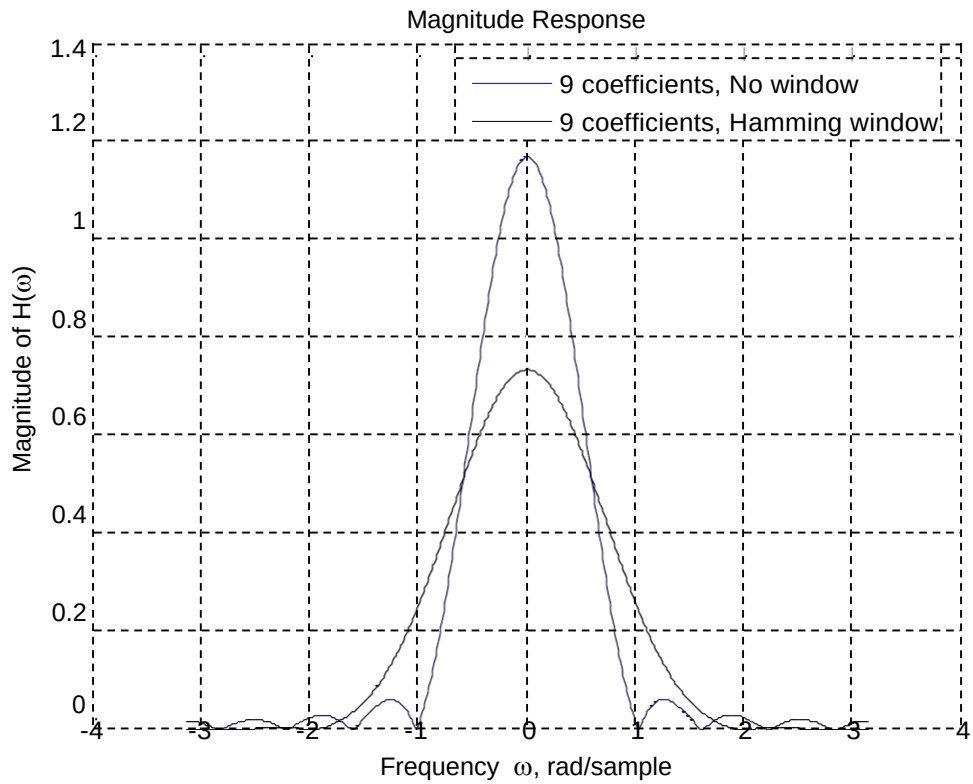
$$H(z) = z^{-4} H_t(z) = 0.00382\,(1 + z^{-8}) + 0.0216\,(z^{-1} + z^{-7}) + 0.0815\,(z^{-2} + z^{-6}) + 0.1617\,(z^{-3} + z^{-5}) + 0.2\, z^{-4}$$

HW Write the corresponding difference equation and show the filter structure.

We compare below the 9-tap Hamming windowed filter to the filter without the window.

```
%Comparison of no window vs. Hamming window
%Nine-tap filter coefficients, no window
b9=[0.0468, 0.1009, 0.1514, 0.1871, 0.2, 0.1871, 0.1514, 0.1009, 0.0468],
%
%Nine-tap, Hamming-windowed filter coefficients
```

```
b9Ham=[0.00382, 0.0216, 0.0815, 0.1617, 0.2, 0.1617, 0.0815, 0.0216, 0.00382]
a=[1]
w=-pi: pi/256: pi;
Hw9=freqz(b9, a, w);
Hw9Ham=freqz(b9Ham, a, w);
plot(w, abs(Hw9), w, abs(Hw9Ham), 'k')
legend ('9 coefficients, No window', '9 coefficients, Hamming window');
title('Magnitude Response');
xlabel('Frequency \omega, rad/sample'), ylabel('Magnitude of H(\omega)'); grid
```



Example 4.4.2 [Low pass filter] [2003] Design a low pass FIR filter that approximates the following frequency response

$$H(F) = \begin{cases} 1, & 0 \le F \le 1000 \text{ Hz} \\ 0, & \text{elsewhere in } 0 \le F \le F_s/2 \end{cases}$$

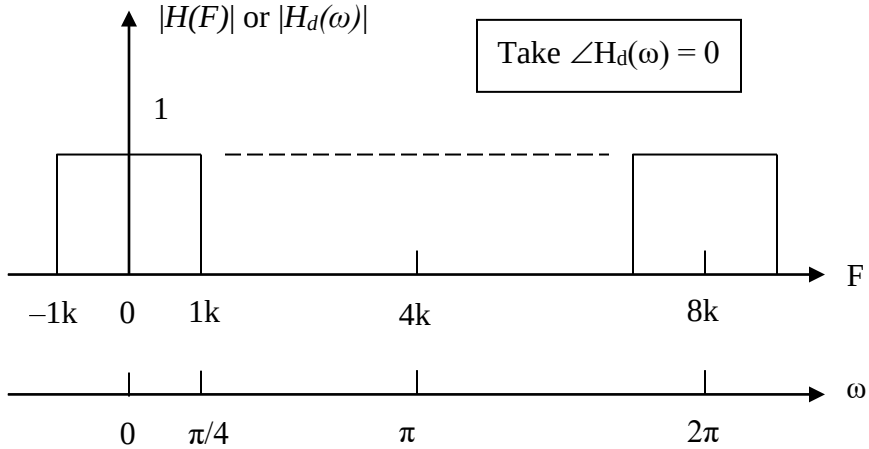
where the sampling frequency F_s is 8000 sps. The impulse response duration is to be limited to msec. Draw the filter structure.

Solution Note the specs are given in Hertz. On the digital frequency scale ω goes from 0 to 2π, with 2π corresponding to the sampling frequency of F_s = 8000 Hz or to $\Omega_s = 2\pi$ 8000 rad/sec. Based on this 1000 Hz corresponds to $(1/8)2\pi = \pi/4$. Or we may use the relation $\omega = \Omega T$ to convert the analog frequency 1000 Hz to the digital frequency. Thus

$$\omega = \Omega T = 2\pi F\, T = 2\pi\, 1000(1/8000) = \pi/4$$

Thus the specifications are restated on the ω scale as

$$H_d(e^{j\omega}) = \begin{cases} 1, & -\pi/4 \le \omega \le \pi/4 \\ 0, & \text{elsewhere in the range } [-\pi \text{ to } \pi] \end{cases}$$

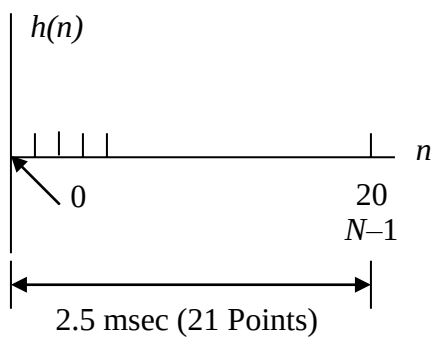


We can evaluate the impulse response

$$h_d(n) = \frac{1}{2\pi}\int_{-\pi} H_d(e^{j\omega})\, e^{j\omega n} d\omega = \frac{1}{2\pi}\int_{-\pi/4}^{\pi/4} 1.e^{j\omega n} d\omega = \frac{1}{2\pi}\left[\frac{e^{j\omega n}}{jn}\right]_{-\pi/4}^{\pi/4}$$

$$= \frac{(e^{j\pi n/4} - e^{-j\pi n/4})}{2\pi jn} = \frac{\sin 0.25\pi n}{\pi n}$$

We need to decide the number of coefficients, that is, the filter length, needed. The problem specifies the impulse response duration is to be limited to 2.5 msec. At 8000



samples/sec., the sampling period T = 1/8000 = 0.125 msec. So the duration of 2.5msec translate to 2.5/0.125 = 20 sample periods. Arranging these on a linear scale we see that the filter length N = 21 or we need 21 coefficients. Therefore determine the values $h_d(-10)$ through $h_d(10)$,

$$h_t(n) = \frac{\sin 0.25\pi n}{\pi n}, \quad -10 \le n \le 10$$

$h_t(0)$ = 0.25 by L'Hopital's rule

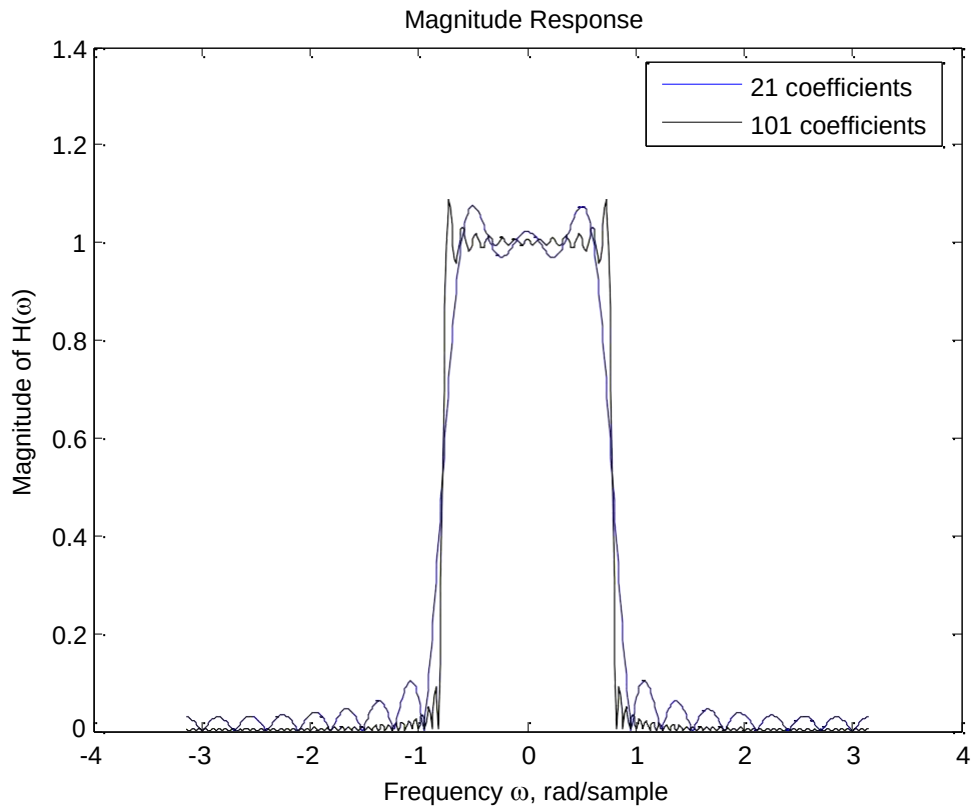
```
%Calculate hdn = (sin (0.25*pi*n)) / (pi*n) and stem plot
n = -50: 50, hdn = (sin(0.25*pi*n)) ./(pi*n), stem(n, hdn)

n = -50 to 50
Warning: Divide by zero.
```

```
hdn = [0.0064    0.0046   -0.0000   -0.0048   -0.0069   -0.0050    0.0000    0.0052
0.0076    0.0055   -0.0000   -0.0058   -0.0084   -0.0061    0.0000    0.0064
0.0094    0.0068   -0.0000   -0.0073   -0.0106   -0.0078    0.0000    0.0083
0.0122    0.0090   -0.0000   -0.0098   -0.0145   -0.0107    0.0000    0.0118
0.0177    0.0132   -0.0000   -0.0150   -0.0227   -0.0173    0.0000    0.0205
0.0318    0.0250   -0.0000   -0.0322   -0.0531   -0.0450    0.0000    0.0750
0.1592    0.2251       NaN    0.2251    0.1592    0.0750    0.0000   -0.0450  -
0.0531   -0.0322   -0.0000    0.0250    0.0318    0.0205    0.0000   -0.0173  -
0.0227   -0.0150   -0.0000    0.0132    0.0177    0.0118    0.0000   -0.0107
  -0.0145   -0.0098   -0.0000    0.0090    0.0122    0.0083    0.0000 -0.0078 -
0.0106   -0.0073   -0.0000    0.0068    0.0094    0.0064    0.0000   -0.0061  -
0.0084   -0.0058   -0.0000    0.0055    0.0076    0.0052    0.0000   -0.0050  -
0.0069   -0.0048   -0.0000    0.0046    0.0064]
```

n =	0	±1	±2	±3	±4	±5	±6	±7	±8	±9	±10
$h_t(n)$ =	0.25	0.2251	0.1592	0.075	0	-0.045	0.0531	-0.0322	0	0.025	0.0318

Then take the z transform $H_t(z) = \sum_{n=-10}^{10} h(n) z^{-n}$. Then determine the transfer function, $H(z)$, of the realizable FIR filter as $H(z) = z^{-10} H_t(z)$ from which the filter structure can be drawn. The frequency response is compared below for 21 and 101 coefficients. The 21-tap filter is not that bad.

```
%21-tap filter coefficients
b21=[0.0318    0.0250   -0.0000   -0.0322   -0.0531   -0.0450    0.0000    0.0750
0.1592    0.2251       0.25    0.2251    0.1592    0.0750    0.0000 -0.0450 -0.0531
-0.0322 -0.0000    0.0250    0.0318],
a=[1]
%101-tap filter coefficients
b101=[0.0064    0.0046   -0.0000   -0.0048   -0.0069   -0.0050    0.0000    0.0052
0.0076    0.0055   -0.0000   -0.0058   -0.0084   -0.0061    0.0000    0.0064
0.0094    0.0068   -0.0000   -0.0073   -0.0106   -0.0078    0.0000    0.0083
0.0122    0.0090   -0.0000   -0.0098   -0.0145   -0.0107    0.0000    0.0118
0.0177    0.0132   -0.0000   -0.0150   -0.0227   -0.0173    0.0000    0.0205
0.0318    0.0250   -0.0000   -0.0322   -0.0531   -0.0450    0.0000    0.0750
0.1592    0.2251       0.25    0.2251    0.1592    0.0750    0.0000 -0.0450  -0.0531
-0.0322   -0.0000    0.0250    0.0318    0.0205    0.0000   -0.0173   -0.0227  -
0.0150   -0.0000    0.0132    0.0177    0.0118    0.0000   -0.0107   -0.0145  -
0.0098   -0.0000    0.0090    0.0122    0.0083    0.0000   -0.0078   -0.0106  -
0.0073   -0.0000    0.0068    0.0094    0.0064    0.0000   -0.0061   -0.0084  -
0.0058   -0.0000    0.0055    0.0076    0.0052    0.0000   -0.0050   -0.0069  -
0.0048   -0.0000    0.0046    0.0064],
w=-pi: pi/256: pi;
Hw21=freqz(b21, a, w);
Hw101=freqz(b101, a, w);
plot(w, abs(Hw21), w, abs(Hw101), 'k')
legend ('21 coefficients', '101 coefficients');
title('Magnitude Response');
xlabel('Frequency \omega, rad/sample'), ylabel('Magnitude of H(\omega)');
```



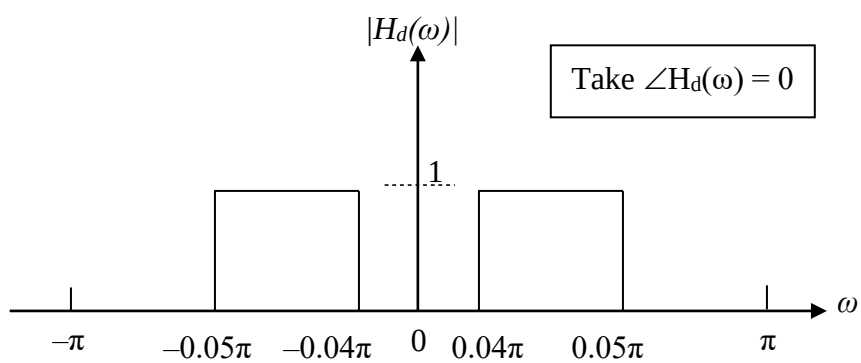
Example 4.4.3 [Very narrow band pass filter] [2003] Design a band pass FIR filter that approximates the following frequency response:

$$H(F) = \begin{cases} 1, & 160 \le F \le 200 \text{ Hz} \\ 0, & \text{elsewhere in } 0 \le F \le F_s/2 \end{cases}$$

when the sampling frequency is 8000 sps. Limit the duration of impulse response to 2 msec. Draw the filter structure.

Solution The sampling frequency F_s = 8000 Hz corresponds to $\omega = 2\pi$ rad. Thus

160 Hz corresponds to $\omega = (160/8000)2\pi = 0.04\ \pi$ rad., and
200 Hz corresponds to $\omega = (200/8000)2\pi = 0.05\ \pi$ rad.



$$H_d(e^{j\omega}) = \begin{cases} 1, & -0.05\pi \le \omega \le -0.04\pi \quad \text{and} \quad 0.04\pi \le \omega \le 0.05\pi \\ 0, & \text{elsewhere in the range } [-\pi \text{ to } \pi] \end{cases}$$

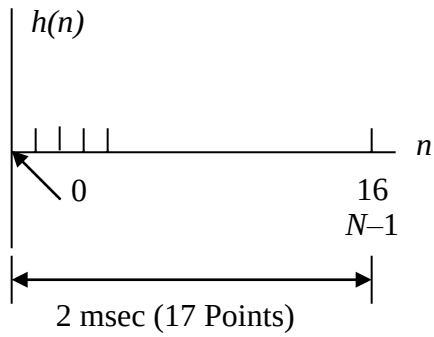
$$h_d(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(e^{j\omega})e^{j\omega n}d\omega = \frac{1}{2\pi}\left(\int_{-0.05\pi}^{-0.04\pi} 1e^{j\omega n}d\omega + \int_{0.04\pi}^{0.05\pi} 1e^{j\omega n}d\omega\right)$$

$$= \frac{1}{2\pi}\left\{\left[\frac{e^{j\omega n}}{jn}\right]_{-0.05\pi}^{-0.04\pi} + \left[\frac{e^{j\omega n}}{jn}\right]_{0.04\pi}^{0.05\pi}\right\}$$

$$= \frac{1}{j2\pi n}\left\{\left(e^{-j0.04\pi n} - e^{-j0.05\pi n}\right) + \left(e^{j0.05\pi n} - e^{j0.04\pi n}\right)\right\}$$

$$= \frac{1}{\pi n}\left\{\frac{e^{j0.05\pi n} - e^{-j0.05\pi n}}{j2} - \frac{e^{j0.04\pi n} - e^{-j0.04\pi n}}{j2}\right\}$$

$$= \frac{\sin 0.05\pi n - \sin 0.04\pi n}{\pi n}$$



Filter length N is determined by the duration of the impulse response. Two milliseconds corresponds to 0.002 / (1/8000) = 16 sample periods. This means that the filter length N = 17, and there will be 17 coefficients. Determine the values of $h_d(n)$ for $-8 \le n \le 8$, so that

$h_t(n) = \{ h_d(-8), h_d(-7), \ldots, h_d(0), \ldots, h_d(8) \}$, and

$$H_t(z) = \sum_{n=-8}^{8} h_t(n)z^{-n}$$

Delay the impulse response by 8 sample periods so that

$h(n) = h_t(n-8)$ and $H(z) = z^{-8}H_t(z)$

$h_d(0) = 0.01$ by L'Hopital's rule

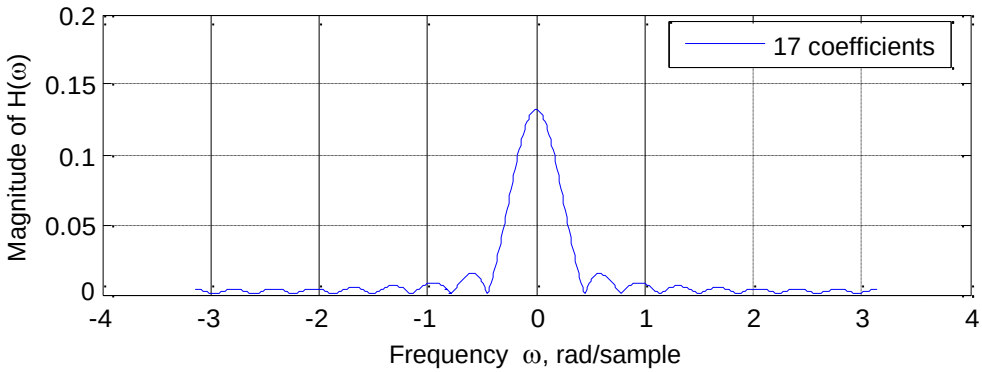
```
%Calculate hdn = (sin(0.05*pi*n) - sin(0.04*pi*n)) /(pi*n) and stem plot
n = -50: 50, hdn = (sin(0.05*pi*n) - sin(0.04*pi*n)) ./(pi*n), stem(n, hdn)

n = -50 to 50
Warning: Divide by zero.
hdn =[0.0064    0.0072    0.0080    0.0085    0.0089    0.0092    0.0092    0.0091
0.0087    0.0082    0.0076    0.0067    0.0058    0.0047    0.0035    0.0022    0.0009
-0.0005   -0.0018   -0.0031   -0.0044   -0.0056 -0.0066 -0.0076 -0.0084  -
0.0090   -0.0095   -0.0097   -0.0098   -0.0097 -0.0094 -0.0088 -0.0082  -
```

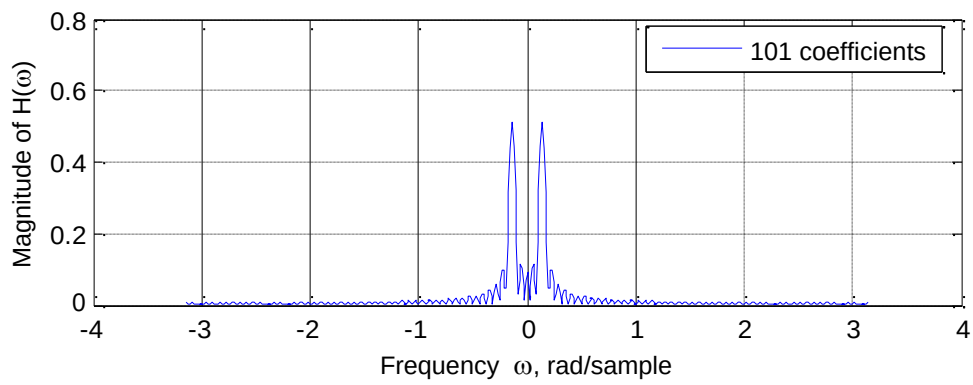
0.0073 -0.0063 -0.0052 -0.0039 -0.0026 -0.0012 0.0002 0.0016
0.0029 ***0.0042 0.0055 0.0066 0.0076 0.0084 0.0091 0.0096 0.0099***
NaN 0.0099 0.0096 0.0091 0.0084 0.0076 0.0066 0.0055 0.0042
0.0029 0.0016 0.0002 -0.0012 -0.0026 -0.0039 -0.0052 -0.0063 -
0.0073 -0.0082 -0.0088 -0.0094 -0.0097 -0.0098 -0.0097 -0.0095 -
0.0090 -0.0084 -0.0076 -0.0066 -0.0056 -0.0044 -0.0031 -0.0018 -
0.0005 0.0009 0.0022 0.0035 0.0047 0.0058 0.0067 0.0076 0.0082
0.0087 0.0091 0.0092 0.0092 0.0089 0.0085 0.0080 0.0072
0.0064]

n =	0	±1	±2	±3	±4	±5	±6	±7	±8
$h_d(n)$ =	0.01	0.0099	0.0096	0.0091	0.0084	0.0076	0.0066	0.0055	0.0042

The frequency responses 17-tap and 101-tap filters are shown below. The 17-tap filter looks more like a low pass filter! Owing to the very narrow pass band a very large number of coefficients is needed before the pass band becomes discernible. In general FIR filters are characterized by a large number of coefficients compared to IIR filters.

```
%Filter coefficients
b17=[0.0042 0.0055 0.0066 0.0076 0.0084 0.0091 0.0096 0.0099
0.01 0.0099 0.0096 0.0091 0.0084 0.0076 0.0066 0.0055 0.0042],
a=[1]
b101=[0.0064 0.0072 0.0080 0.0085 0.0089 0.0092 0.0092 0.0091
0.0087 0.0082 0.0076 0.0067 0.0058 0.0047 0.0035 0.0022 0.0009
-0.0005 -0.0018 -0.0031 -0.0044 -0.0056 -0.0066 -0.0076 -0.0084 -
0.0090 -0.0095 -0.0097 -0.0098 -0.0097 -0.0094 -0.0088 -0.0082 -
0.0073 -0.0063 -0.0052 -0.0039 -0.0026 -0.0012 0.0002 0.0016
0.0029 0.0042 0.0055 0.0066 0.0076 0.0084 0.0091 0.0096 0.0099
0.01 0.0099 0.0096 0.0091 0.0084 0.0076 0.0066 0.0055 0.0042
0.0029 0.0016 0.0002 -0.0012 -0.0026 -0.0039 -0.0052 -0.0063 -
0.0073 -0.0082 -0.0088 -0.0094 -0.0097 -0.0098 -0.0097 -0.0095 -
0.0090 -0.0084 -0.0076 -0.0066 -0.0056 -0.0044 -0.0031 -0.0018 -
0.0005 0.0009 0.0022 0.0035 0.0047 0.0058 0.0067 0.0076 0.0082
0.0087 0.0091 0.0092 0.0092 0.0089 0.0085 0.0080 0.0072
0.0064]
w=-pi: pi/256: pi; Hw17=freqz(b17, a, w); Hw101=freqz(b101, a, w);
subplot(2, 1, 1), plot(w, abs(Hw17)); legend ('17 coefficients');
xlabel('Frequency \omega, rad/sample'), ylabel('Magnitude of H(\omega)'); grid
subplot(2, 1, 2), plot(w, abs(Hw101)); legend ('101 coefficients');
xlabel('Frequency \omega, rad/sample'), ylabel('Magnitude of H(\omega)'); grid
```





Example 4.4.4 [Band pass filter] [2003] Design a band pass filter to pass frequencies in the range 1 to 2 rad/sec using **Hanning window** with $N = 5$. Draw the filter structure and plot its spectrum.

Solution The Hanning window is also known as the **Hann window**. It is a raised cosine, is very similar to the Hamming window, and is given by

$$w_{Han}(n) = \begin{cases} 0.5\{1-\cos(2\pi n/(N-1))\}, & 0 \le n \le N-1 \\ 0, \text{ elsewhere} \end{cases}$$

Note In order to convert the analog frequencies to digital we need the sampling time T. The sampling frequency F_s (or the sampling time T) is not specified. We assume $T = 1$ sec or, what amounts to the same, we assume that the frequencies given are actually digital, that is, 1 to 2 rad/sample instead of 1 to 2 rad/sec. However, the solution below assumes that the frequencies are given correctly, that is, that they are analog, and uses a sampling frequency of $\Omega_s = 4$ rad / sec.

Although there is a specialized version of the sampling theorem for band pass signals we shall simply take the sampling frequency to be twice the highest frequency which is 2 rad / sec. Thus we shall take $\Omega_s = 4$ rad / sec. This then gives us a *high pass* filter rather a band pass. However, if we take, say, $\Omega_s = 8$ rad / sec., we shall have a *band pass* filter. We shall next convert the analog frequency specs to digital (ω):

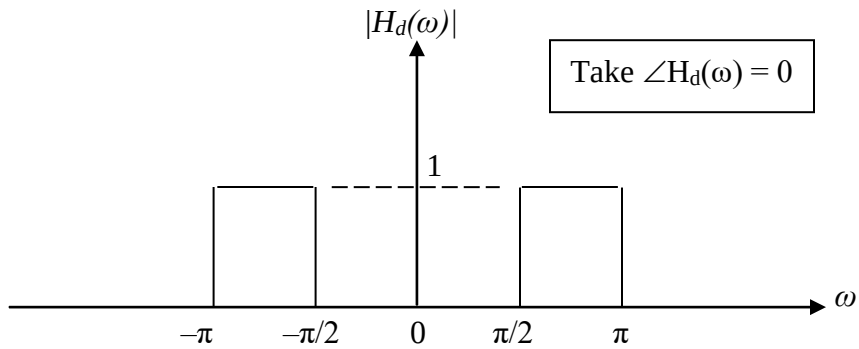
$\Omega_s = 4$ rad / sec corresponds to $\omega = 2\pi$ rad

1 rad / sec corresponds to $\omega = 2\pi/4 = \pi/2$ rad

2 rad / sec corresponds to $\omega = (2\pi/4)\ 2 = \pi$rad

Thus

$$H_d(\omega) = \begin{cases} 1, & -\pi \le \omega \le -\pi/2 \text{ and } \pi/2 \le \omega \le \pi \\ 0, & \text{elsewhere in } [-\pi, \pi] \end{cases}$$



Evaluate impulse response

$$h_d(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(\omega)\, e^{j\omega n} d\omega = \frac{1}{2\pi}\left[\int_{-\pi}^{-\pi/2} 1\, e^{j\omega n} d\omega + \int_{\pi/2}^{\pi} 1\, e^{j\omega n} d\omega\right]$$

$$= \frac{1}{2\pi}\left\{\left[\frac{e^{j\omega n}}{jn}\right]_{-\pi}^{-\pi/2} + \left[\frac{e^{j\omega n}}{jn}\right]_{\pi/2}^{\pi}\right\} = \frac{1}{j2\pi n}\left\{\left(e^{-j\pi n/2} - e^{-j\pi n}\right) + \left(e^{j\pi n} - e^{j\pi n/2}\right)\right\}$$

$$= \frac{1}{\pi n}\left\{\frac{\left(e^{j\pi n} - e^{-j\pi n}\right)}{j2} - \frac{\left(e^{j\pi n/2} - e^{-j\pi n/2}\right)}{j2}\right\} = \frac{\sin(\pi n) - \sin(\pi n/2)}{\pi n}$$

$h_d(0) = 0.5$ by L'Hopital's rule

Determine $h_d(n)$ for $-2 \le n \le 2$, so that

$$h_t(n) = \{h_d(-2), h_d(-1), h_d(0), h_d(1), h_d(2)\}$$

For $N = 5$ the Hanning window is given by

$$w_{Han}(n) = 0.5\left\{1 - \cos\left(2\pi n/5 - 1\right)\right\},\ 0 \le n \le 5-1$$

$$= 0.5\left\{1 - \cos\left(\pi n/2\right)\right\}, \qquad 0 \le n \le 4$$

Thus $w(n) = \{0, 0.5, 1, 0.5, 0\}$. Multiplying $h_t(n)$ and $w_{Han}(n)$ point by point we get

$$h_t(n) = h_t(n)\, w_{Han}(n) = \left\{0, \frac{h_d(-1)}{2}, h_d(0), \frac{h_d(1)}{2}, 0\right\} \text{ and}$$

$$H_t(z) = \sum_{n=-2}^{2} h_t(n) z^{-n}$$

Delay by 2 samples to get $h(n) = h_t(n-2)$ and $H(z) = z^{-2} H_t(z)$. Now draw the direct form structure for $H(z)$. The spectrum is given by $H(e^{j\omega}) = H(z)\big|_{z=e^{j\omega}}$. We compare below filters lengths of 5 and 101 without the Hanning window.

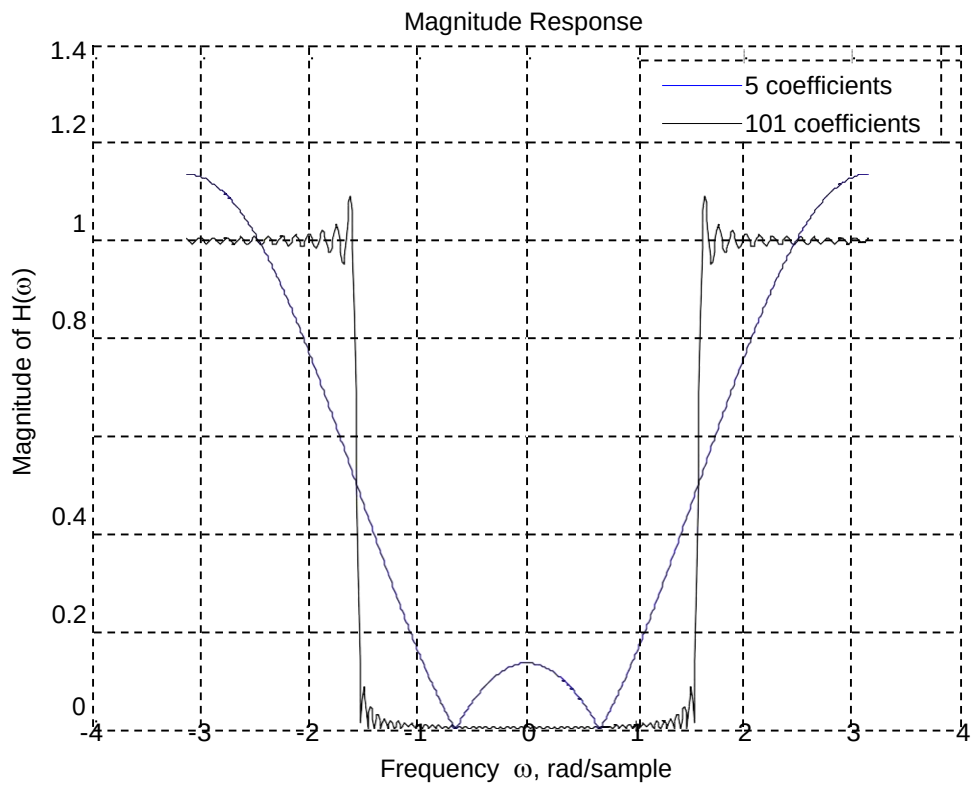
Generate filter coefficients:

```
%Calculate hdn = (sin(pi*n) - sin(pi*n/2)) /(pi*n) and stem plot
n = -50: 50, hdn = (sin(pi*n) - sin(pi*n/2)) ./(pi*n), stem(n, hdn)
```

```
n = -50 to 50
Warning: Divide by zero.
hdn = [0.0000   -0.0065   -0.0000    0.0068   -0.0000   -0.0071   -0.0000    0.0074
0.0000    -0.0078   -0.0000    0.0082   -0.0000   -0.0086   -0.0000    0.0091
0.0000   -0.0096    -0.0000    0.0103   -0.0000   -0.0110   -0.0000    0.0118
0.0000   -0.0127   -0.0000     0.0138   -0.0000   -0.0152   -0.0000    0.0168   -
0.0000   -0.0187   -0.0000    0.0212    -0.0000   -0.0245   -0.0000    0.0289   -
0.0000   -0.0354   -0.0000    0.0455   -0.0000    -0.0637   -0.0000    0.1061   -
0.0000   -0.3183       NaN   -0.3183   -0.0000    0.1061    -0.0000   -0.0637   -
0.0000    0.0455   -0.0000   -0.0354   -0.0000    0.0289   -0.0000    -0.0245   -
0.0000    0.0212   -0.0000   -0.0187   -0.0000    0.0168   -0.0000   -0.0152    -
0.0000    0.0138   -0.0000   -0.0127    0.0000    0.0118   -0.0000   -0.0110   -
0.0000     0.0103   -0.0000   -0.0096    0.0000    0.0091   -0.0000   -0.0086   -
0.0000    0.0082    -0.0000   -0.0078    0.0000    0.0074   -0.0000   -0.0071   -
0.0000    0.0068   -0.0000    -0.0065    0.0000]
```

Generate frequency responses:

```
%5-tap filter coefficients
b5=[0.0000   -0.3183       0.5 -0.3183 -0.0000],
a=[1]
%101-tap filter coefficients
b101=[0.0000   -0.0065   -0.0000    0.0068   -0.0000   -0.0071   -0.0000    0.0074
0.0000    -0.0078   -0.0000    0.0082   -0.0000   -0.0086   -0.0000    0.0091
0.0000   -0.0096    -0.0000    0.0103   -0.0000   -0.0110   -0.0000    0.0118
0.0000   -0.0127   -0.0000     0.0138   -0.0000   -0.0152   -0.0000    0.0168   -
0.0000   -0.0187   -0.0000    0.0212    -0.0000   -0.0245   -0.0000    0.0289   -
0.0000   -0.0354   -0.0000    0.0455   -0.0000    -0.0637   -0.0000    0.1061   -
0.0000   -0.3183       0.5   -0.3183   -0.0000    0.1061    -0.0000 -0.0637  -0.0000
0.0455   -0.0000   -0.0354   -0.0000    0.0289   -0.0000    -0.0245   -0.0000
0.0212   -0.0000   -0.0187   -0.0000    0.0168   -0.0000   -0.0152    -0.0000
0.0138   -0.0000   -0.0127    0.0000    0.0118   -0.0000   -0.0110   -0.0000
0.0103   -0.0000   -0.0096    0.0000    0.0091   -0.0000   -0.0086   -0.0000
0.0082    -0.0000   -0.0078    0.0000    0.0074   -0.0000   -0.0071   -0.0000
0.0068   -0.0000    -0.0065    0.0000],
w=-pi: pi/256: pi;
Hw5=freqz(b5, a, w);
Hw101=freqz(b101, a, w);
plot(w, abs(Hw5), w, abs(Hw101), 'k')
legend ('5 coefficients', '101 coefficients');
title('Magnitude Response');
xlabel('Frequency \omega, rad/sample'), ylabel('Magnitude of H(\omega)'); grid
```



We compare below the frequency responses of the 5-tap filter with and without the Hanning window. It can be seen that the Hanning window aggravates what is already a poor (short) filter length. There may be more to gain by increasing the filter length than by windowing.

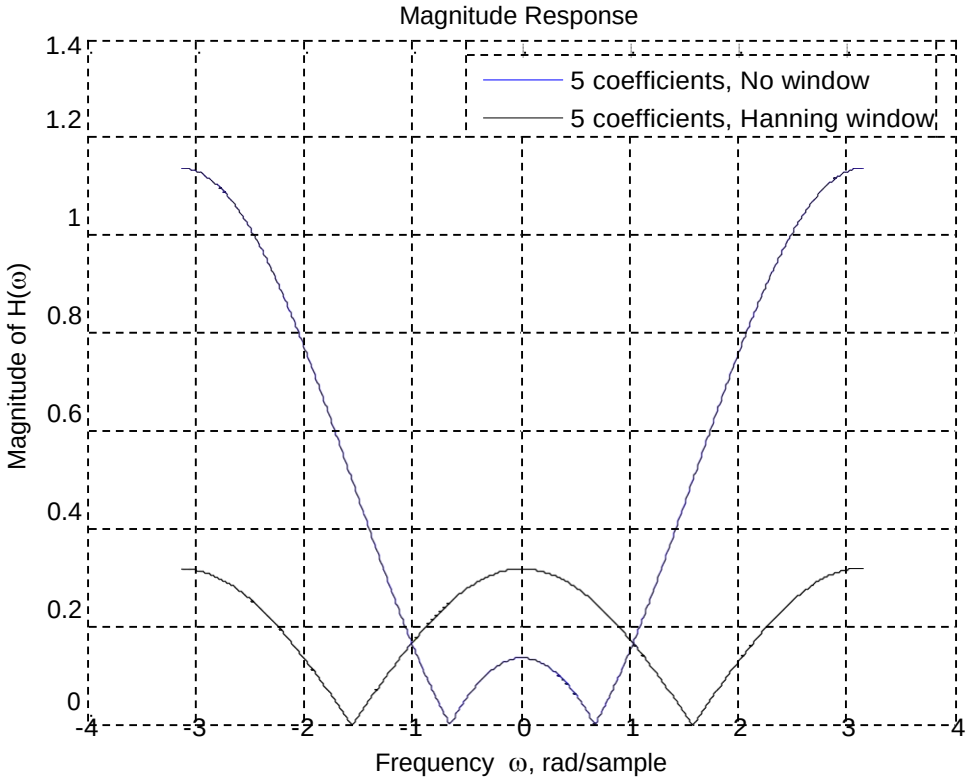
Generate Hanning window:

```
%Generate Hanning window wn = 0.5*(1–cos(pi*n/2)) and stem plot
n = -2: 2, wn = 0.5*(1-cos(pi*n/2)), stem(n, wn)

n = -2 to 2
wn = 1.0000    0.5000         0    0.5000    1.0000
```

Generate frequency responses:

```
%5-tap filter coefficients
b5=[0.0000   -0.3183       0.5 -0.3183  -0.0000],
%Hanning window coefficients
wn = [1.0000    0.5000         0    0.5000    1.0000],
%Windowed coefficients
b5Han = b5 .*wn,
a=[1]
w=-pi: pi/256: pi;
Hw5=freqz(b5, a, w);
Hw5Han=freqz(b5Han, a, w);
plot(w, abs(Hw5), w, abs(Hw5Han), 'k')
legend ('5 coefficients, No window', '5 coefficients, Hanning window ');
```

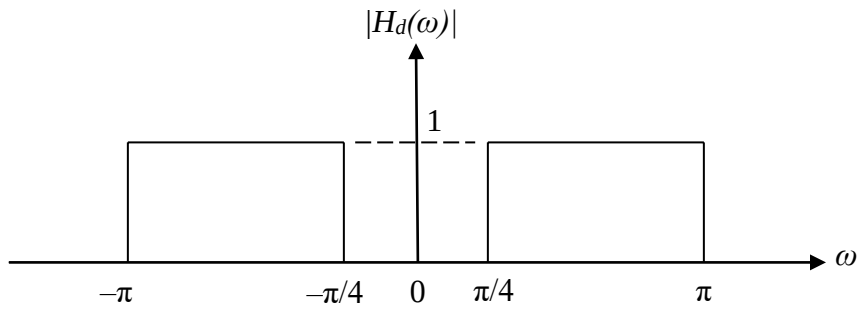
```
title('Magnitude Response');
xlabel('Frequency \omega, rad/sample'), ylabel('Magnitude of H(\omega)'); grid
```

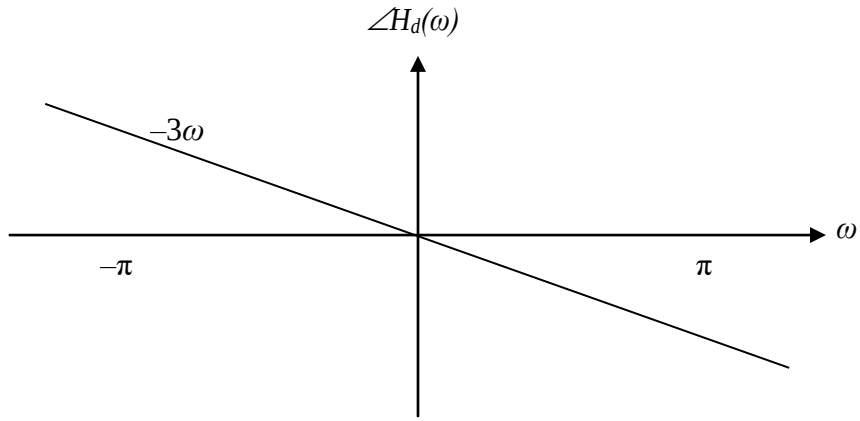


Example 4.4.5 [High pass filter] [2008] Design a high pass linear phase filter with frequency response

$$H_d(e^{j\omega}) = \begin{cases} 1\, e^{-j3\omega}, & \omega_c \le \omega \le \pi \\ 0, & \text{elsewhere in the range } [-\pi \text{ to } \pi] \end{cases}$$

The number of filter coefficients is $N = 7$ and $\omega_c = \pi/4$. Use (a) rectangular window and (b) Hamming window.





$$h_d(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(\omega)\, e^{j\omega n} d\omega = \frac{1}{2\pi}\left[\int_{-\pi}^{-\pi/4} 1\, e^{-j3\omega} e^{j\omega n} d\omega + \int_{\pi/4}^{\pi} 1\, e^{-j3\omega} e^{j\omega n} d\omega\right]$$

$$= \frac{1}{2\pi}\left[\int_{-\pi}^{-\pi/4} e^{j(n-3)\omega} d\omega + \int_{\pi/4}^{\pi} e^{j(n-3)\omega} d\omega\right]$$

$$= \frac{1}{2\pi}\left\{\left[\frac{e^{j(n-3)\omega}}{j(n-3)}\right]_{-\pi}^{-\pi/4} + \left[\frac{e^{j(n-3)\omega}}{j(n-3)}\right]_{\pi/4}^{\pi}\right\}$$

$$= \frac{1}{j2\pi(n-3)}\left\{\left(e^{-j\pi(n-3)/4} - e^{-j\pi(n-3)}\right) - \left(e^{j\pi(n-3)} - e^{j\pi(n-3)/4}\right)\right\}$$

$$= \frac{1}{\pi(n-3)}\left\{\frac{\left(e^{j\pi(n-3)} - e^{-j\pi(n-3)}\right)}{j2} - \frac{\left(e^{j\pi(n-3)/4} - e^{-j\pi(n-3)/4}\right)}{j2}\right\}$$

$$= \frac{\sin(\pi(n-3)) - \sin(\pi(n-3)/4)}{\pi(n-3)}$$

This sequence is centered at $n = 3$. Since the filter length is $N = 7$ we can calculate the 7 coefficients as $\{h_d(n), 0 \le n \le 6\}$. In other words we truncate it outside the interval $0 \le n \le 6$; moreover, there is no need to right-shift the truncated sequence.

In general, one may not know the filter length with certainty and there is no special advantage in specifying the phase, $\angle H_d(\omega)$, as anything but zero. We calculate 101 coefficients of the sequence centered about $n = 0$, that is, $h_d(n) = \dfrac{\sin(\pi n) - \sin(\pi n / 4)}{\pi n}$:

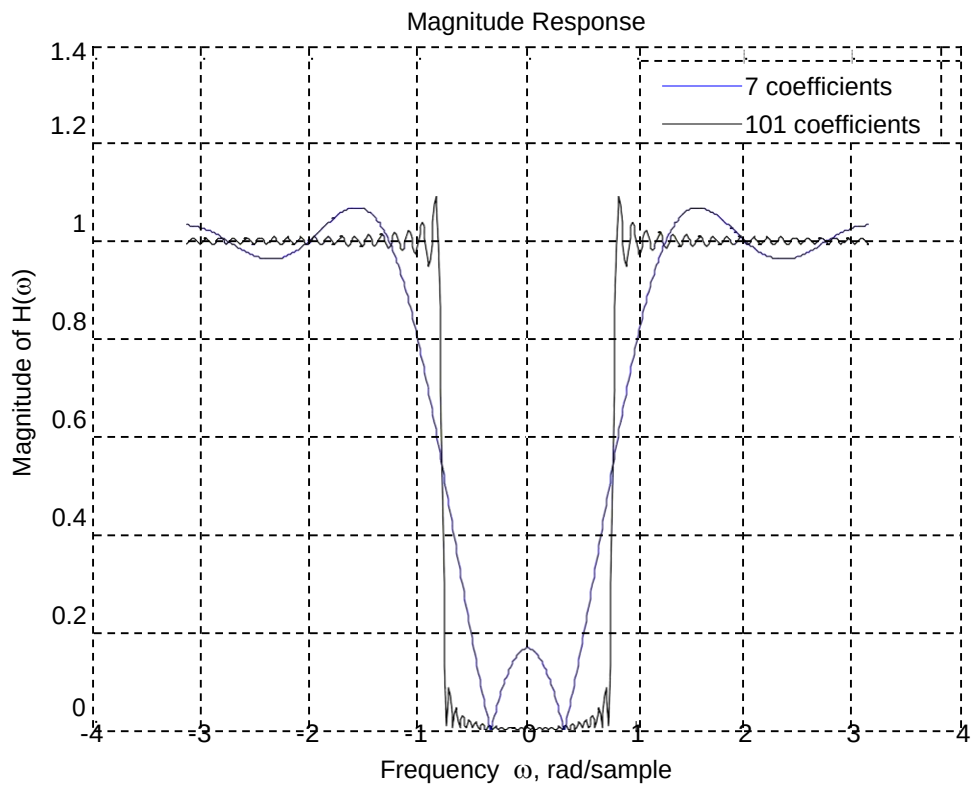
$h_d(0) = 0.75$ by L'Hopital's rule

```
% Generate hdn = (sin(pi*n) - sin(pi*n/4)) /(pi*n) and stem plot
n = -50: 50, hdn = (sin(pi*n) - sin(pi*n/4)) ./(pi*n), stem(n, hdn)
```

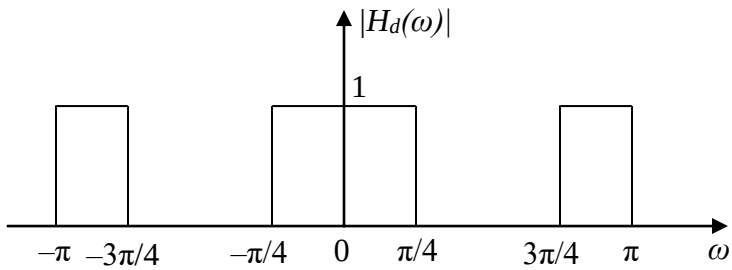
n = –50 to 50
Warning: Divide by zero.
hdn = [-0.0064 -0.0046 -0.0000 0.0048 0.0069 0.0050 -0.0000 -0.0052
-0.0076 -0.0055 -0.0000 0.0058 0.0084 0.0061 -0.0000 -0.0064 -
0.0094 -0.0068 -0.0000 0.0073 0.0106 0.0078 -0.0000 -0.0083 -
0.0122 -0.0090 -0.0000 0.0098 0.0145 0.0107 -0.0000 -0.0118 -
0.0177 -0.0132 -0.0000 0.0150 0.0227 0.0173 -0.0000 -0.0205 -
0.0318 -0.0250 -0.0000 0.0322 0.0531 0.0450 -0.0000 ***-0.0750*** -
0.1592 -0.2251 NaN -0.2251 -0.1592 -0.0750 -0.0000 0.0450
0.0531 0.0322 -0.0000 -0.0250 -0.0318 -0.0205 -0.0000 0.0173
0.0227 0.0150 -0.0000 -0.0132 -0.0177 -0.0118 -0.0000 0.0107
0.0145 0.0098 -0.0000 -0.0090 -0.0122 -0.0083 -0.0000 0.0078
0.0106 0.0073 -0.0000 -0.0068 -0.0094 -0.0064 -0.0000 0.0061
0.0084 0.0058 -0.0000 -0.0055 -0.0076 -0.0052 -0.0000 0.0050
0.0069 0.0048 -0.0000 -0.0046 -0.0064]

The 7-tap and 101-tap filter responses are shown below

```
%7-tap filter coefficients
b7=[-0.0750   -0.1592   -0.2251       0.75 -0.2251 -0.1592 -0.0750],
a=[1]
%101-tap filter coefficients
b101=[-0.0064   -0.0046   -0.0000    0.0048    0.0069    0.0050 -0.0000 -0.0052
-0.0076  -0.0055   -0.0000    0.0058    0.0084    0.0061   -0.0000   -0.0064   -
0.0094   -0.0068    -0.0000    0.0073    0.0106    0.0078   -0.0000   -0.0083   -
0.0122   -0.0090   -0.0000     0.0098    0.0145    0.0107   -0.0000   -0.0118   -
0.0177   -0.0132   -0.0000    0.0150     0.0227    0.0173   -0.0000   -0.0205   -
0.0318   -0.0250   -0.0000    0.0322    0.0531     0.0450   -0.0000   -0.0750   -
0.1592   -0.2251       0.75   -0.2251   -0.1592   -0.0750    -0.0000    0.0450
0.0531    0.0322   -0.0000   -0.0250   -0.0318   -0.0205   -0.0000     0.0173
0.0227    0.0150   -0.0000   -0.0132   -0.0177   -0.0118   -0.0000    0.0107
0.0145    0.0098   -0.0000   -0.0090   -0.0122   -0.0083   -0.0000    0.0078
0.0106     0.0073   -0.0000   -0.0068   -0.0094   -0.0064   -0.0000    0.0061
0.0084    0.0058    -0.0000   -0.0055  -0.0076    -0.0052 -0.0000      0.0050
0.0069    0.0048   -0.0000    -0.0046  -0.0064],
w=-pi: pi/256: pi;
Hw7=freqz(b7, a, w);
Hw101=freqz(b101, a, w);
plot(w, abs(Hw7), w, abs(Hw101), 'k')
legend ('7 coefficients', '101 coefficients');
title('Magnitude Response');
xlabel('Frequency \omega, rad/sample'), ylabel('Magnitude of H(\omega)'); grid
```



Example 4.4.6 [Band-stop filter] Design a band-stop linear phase filter with the following frequency response. The number of filter coefficients is $N = 31$.



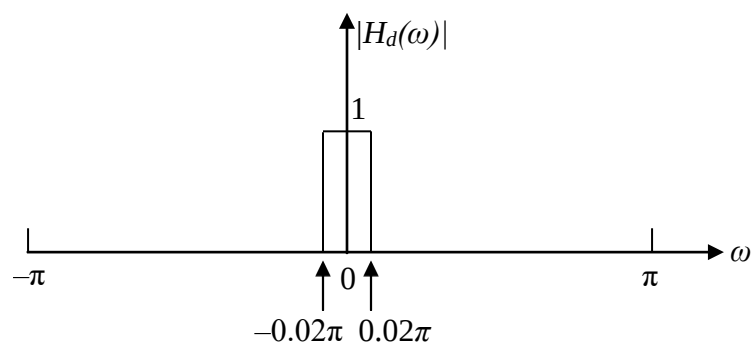
$$h_d(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(\omega)\, e^{j\omega n} d\omega = \frac{1}{2\pi}\left[\int_{-\pi}^{-3\pi/4} 1 e^{j\omega n} d\omega + \int_{-\pi/4}^{\pi/4} 1 e^{j\omega n} d\omega + \int_{3\pi/4}^{\pi} 1 e^{j\omega n}\right]$$

$$= \frac{1}{2\pi}\left\{\left[\frac{e^{j\omega n}}{jn}\right]_{-\pi}^{-3\pi/4} + \left[\frac{e^{j\omega n}}{jn}\right]_{-\pi/4}^{\pi/4} + \left[\frac{e^{j\omega n}}{jn}\right]_{3\pi/4}^{\pi}\right\}$$

$$= \ldots$$

Example 4.4.7 [Design of 9-coefficient narrow-band LP FIR filter] This is a repeat of Example 2.1 with the bandwidth reduced to 0.02π. The band width is *narrower* by a factor of 10 from the band width in that example. The objective is to show that the narrower the band width the larger the number of coefficients needed to achieve the specified frequency response.

Design a nine-coefficient (or 9-point or 9-tap) FIR digital filter to approximate an ideal low-pass filter with a cut-off frequency $\omega_c = 0.02\pi$. The magnitude response, $|H_d(\omega)|$, is given below. Take $\angle H_d(\omega) = 0$.



Solution The impulse response of the desired filter is

$$h_d(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(e^{j\omega})\, e^{j\omega n} d\omega = \frac{1}{2\pi}\int_{-0.02\pi}^{0.02\pi} 1.e^{j\omega n} d\omega$$

$$= \frac{1}{2\pi}\left[\frac{e^{j\omega n}}{jn}\right]_{-0.02\pi}^{\pi} = \frac{(e^{j0.02\pi n} - e^{-j0.02\pi n})}{2\pi jn} = \frac{\sin 0.02\pi n}{\pi n}$$

$$h_d(0) = \left.\frac{\dfrac{d\,(\sin 0.02\pi n)}{dn}}{\dfrac{d\,(\pi n)}{dn}}\right|_{n=0} = \left.\frac{0.02\pi \cos 0.02\pi n}{\pi}\right|_{n=0} = 0.02$$

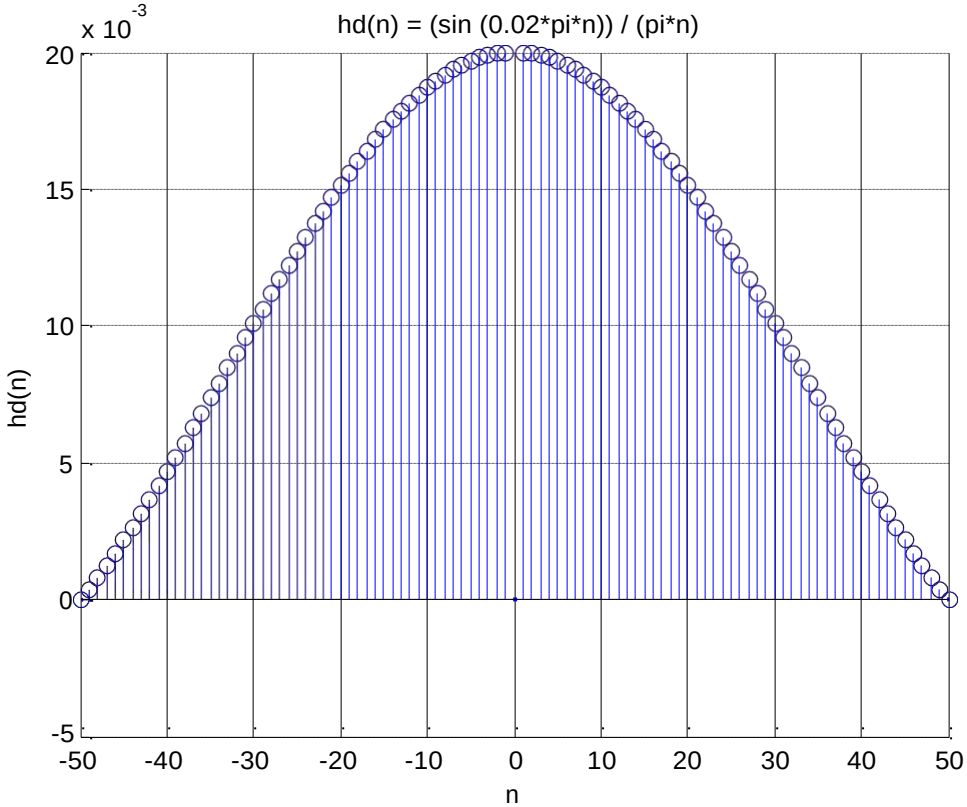
The MATLAB calculation of coefficients follows (good for all *n* except 0 where we fill in the value 0.02):

```
% Generate hdn = (sin (0.02*pi*n)) / (pi*n) and stem plot
n = -50: 50, hdn = (sin(0.02*pi*n)) ./(pi*n), stem(n, hdn)
xlabel('n'), ylabel('hd(n)'); grid; title ('hd(n) = (sin (0.02*pi*n)) / (pi*n)')
```

n = -50 to 50
Warning: Divide by zero.

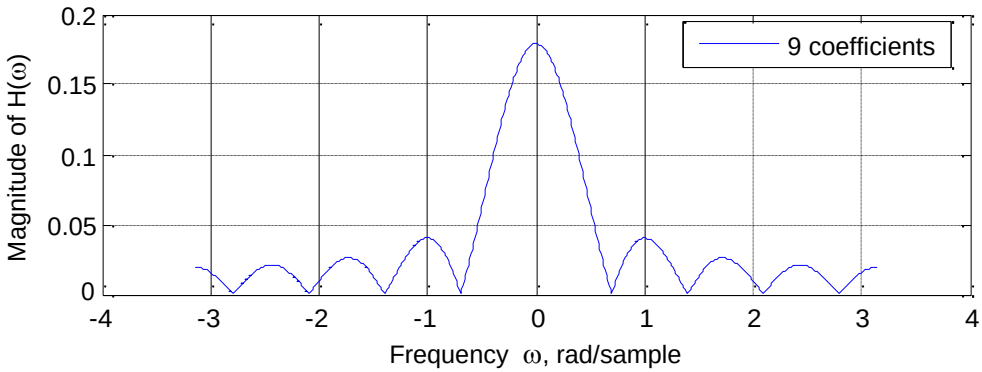
hdn = -0.0000	0.0004	0.0008	0.0013	0.0017	0.0022	0.0027	0.0032	
0.0037	0.0042	0.0047	0.0052	0.0057	0.0063	0.0068	0.0074	0.0079
0.0085	0.0090	0.0095	0.0101	0.0106	0.0112	0.0117	0.0122	0.0127
0.0132	0.0137	0.0142	0.0147	0.0151	0.0156	0.0160	0.0164	0.0168
0.0172	0.0175	0.0178	0.0182	0.0184	0.0187	0.0190	0.0192	0.0194
0.0195	0.0197	***0.0198***	***0.0199***	***0.0199***	***0.0200***	***NaN***	***0.0200***	***0.0199***
0.0199	***0.0198***	0.0197	0.0195	0.0194	0.0192	0.0190	0.0187	0.0184
0.0182	0.0178	0.0175	0.0172	0.0168	0.0164	0.0160	0.0156	0.0151
0.0147	0.0142	0.0137	0.0132	0.0127	0.0122	0.0117	0.0112	0.0106
0.0101	0.0095	0.0090	0.0085	0.0079	0.0074	0.0068	0.0063	0.0057
0.0052	0.0047	0.0042	0.0037	0.0032	0.0027	0.0022	0.0017	0.0013
0.0008	0.0004	-0.0000						

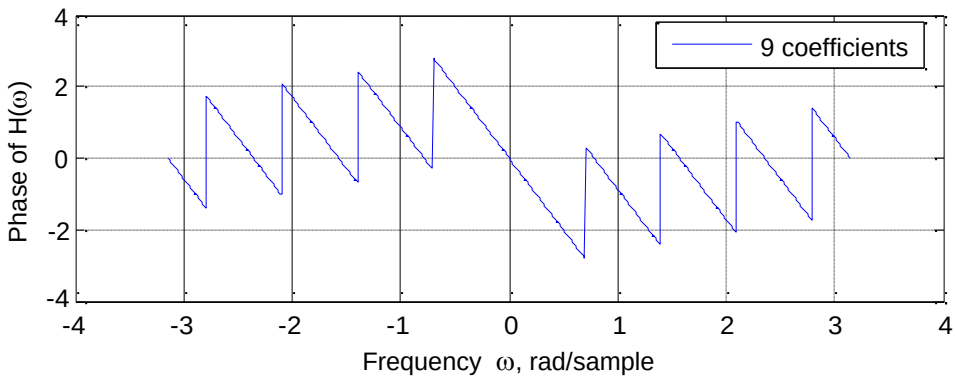
The following MATLAB plot of coefficients is good for all *n* except at *n* = 0.



MATLAB plots of magnitude and phase response of the 9-coefficient narrower band LP filter follow:

```
%Magnitude and phase response of 9-coefficient LP filter
%Filter coefficients
b9ex7= [0.0198 0.0199 0.0199 0.0200 0.02 0.0200 0.0199 0.0199 0.0198],
a=[1]
w=-pi: pi/256: pi;
Hw9ex7=freqz(b9ex7, a, w);
subplot(2, 1, 1), plot(w, abs(Hw9ex7)); legend ('9 coefficients');
xlabel('Frequency \omega, rad/sample'), ylabel('Magnitude of H(\omega)'); grid
subplot(2, 1, 2), plot(w, angle(Hw9ex7)); legend ('9 coefficients');
xlabel('Frequency \omega, rad/sample'), ylabel('Phase of H(\omega)'); grid
```

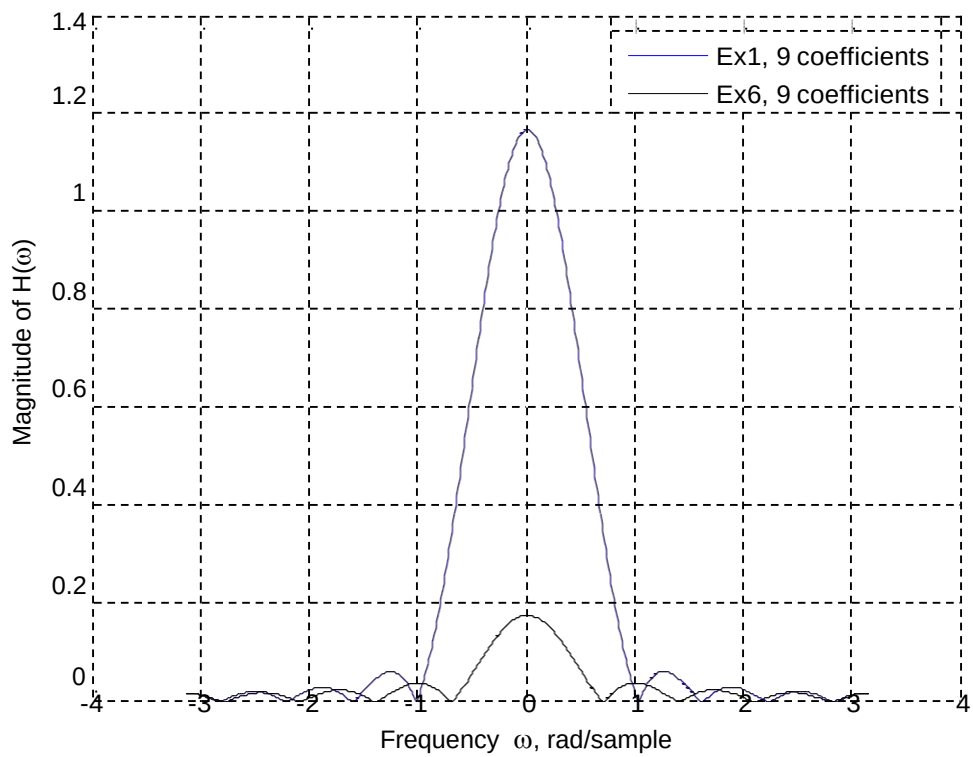

0.2
0.15
0.1
0.05
0
Magnitude of H(ω)
-4
-3
-2
-1
0
1
2
3
4
Frequency ω, rad/sample
9 coefficients


4
2
0
-2
-4
Phase of H(ω)
-4
-3
-2
-1
0
1
2
3
4
Frequency ω, rad/sample
9 coefficients

The MATLAB plots below enable us to compare (only visually) the magnitude responses of the two 9-coefficient filters, the only difference being that Ex 1 has a band width of 0.2π while Ex 2 is very narrow at 0.02π.

```
% Magnitude responses compared: Ex 1 and Ex 7 (9-coefficient LP filters)
%Filter coefficients
b9ex1=[0.0468, 0.1009, 0.1514, 0.1871, 0.2, 0.1871, 0.1514, 0.1009, 0.0468],
b9ex7= [0.0198 0.0199 0.0199 0.0200 0.02 0.0200 0.0199 0.0199 0.0198],
a=[1]
w=-pi: pi/256: pi;
Hw9ex1=freqz(b9ex1, a, w);
Hw9ex7=freqz(b9ex7, a, w);
plot(w, abs(Hw9ex1), w, abs(Hw9ex7), 'k')
legend ('Ex1, 9 coefficients', 'Ex7, 9 coefficients');
xlabel('Frequency \omega, rad/sample'), ylabel('Magnitude of H(\omega)'); grid
```

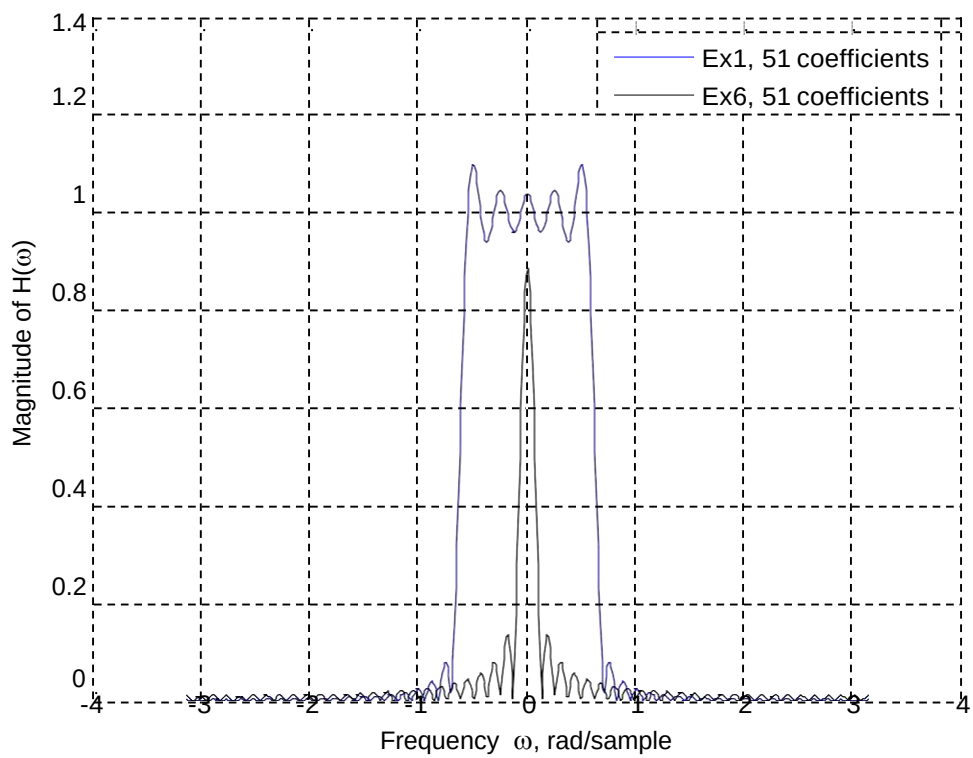
From the plots it clear that the narrow-band filter of Example 6 is no where near either the band width or the gain specified. In contrast, the wider band width filter of Example 1 is relatively much closer to specification.



In the MATLAB plots below we have increased the number of coefficients to 51 for both filters, the only difference being that Ex 1 has a band width of 0.2π while Ex 2 is very narrow at 0.02π.

```
% Magnitude responses compared: Ex 1 and Ex 7 (51-coefficient LP filters)
%Filter coefficients
b51ex1=[0.0, 0.0078    0.0132    0.0138    0.0089 -0.0, -0.0098 -0.0168  -
0.0178   -0.0117    0.0,    0.0134    0.0233    0.0252    0.0170 -0.0, -0.0208  -
0.0378   -0.0432   -0.0312     0.0, 0.0468    0.1009    0.1514    0.1871       0.2
0.1871    0.1514    0.1009 0.0468    0.0   -0.0312 -0.0432 -0.0378 -0.0208  -
0.0    0.0170    0.0252, 0.0233    0.0134    0.0 -0.0117 -0.0178 -0.0168 -
0.0098   -0.0    0.0089, 0.0138    0.0132    0.0078    0.0],
%
b51ex7= [0.0127    0.0132    0.0137    0.0142    0.0147    0.0151    0.0156
0.0160    0.0164    0.0168    0.0172    0.0175    0.0178    0.0182    0.0184    0.0187
0.0190    0.0192    0.0194    0.0195    0.0197    0.0198    0.0199    0.0199    0.0200
0.02    0.0200    0.0199    0.0199    0.0198    0.0197    0.0195    0.0194    0.0192
0.0190    0.0187    0.0184    0.0182    0.0178    0.0175    0.0172    0.0168    0.0164
0.0160    0.0156    0.0151    0.0147    0.0142    0.0137    0.0132    0.0127],
a=[1]
w=-pi: pi/256: pi;
Hw51ex1=freqz(b51ex1, a, w); Hw51ex7=freqz(b51ex7, a, w);
plot(w, abs(Hw51ex1), w, abs(Hw51ex7), 'k')
legend ('Ex1, 51 coefficients', 'Ex7, 51 coefficients');
xlabel('Frequency \omega, rad/sample'), ylabel('Magnitude of H(\omega)'); grid
```

From the plots it clear that the narrow-band filter of Example 6 still has a long way to go. In contrast, the wider band width filter of Example 1 is almost there.

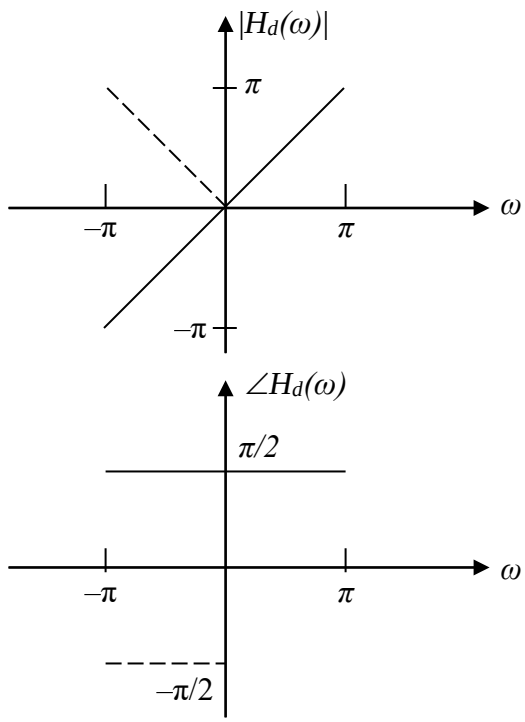


We turn now to **filtering plus quadrature phase shift**. These applications include integrators, differentiators, and Hilbert transform devices. For all of these the desired transfer function is imaginary, i.e., $H(e^{j\omega}) = j\,H_I(\omega)$, with $H_R(\omega) = 0$

Example 4.4.8 [The ideal differentiator] In the analog situation the ideal differentiator is given by the transfer function $H(s) = s$ with the frequency response $H(\Omega) = j\Omega$, for $0 \le \Omega \le \infty$. The digital version of the ideal differentiator may be defined as

$$H_d(e^{j\omega}) \text{ or } H_d(\omega) = j\omega, \qquad -\pi \le \omega \le \pi$$

Note $H_d(\omega) = j\omega = \omega\, e^{j\pi/2}$ has a magnitude of $|H_d(\omega)| = |\omega|$, and a phase $\angle H_d(\omega) = \pm\pi/2$ (see figure).



Since $H_d(\omega)$ is periodic in ω with period 2π, we can expand in into a Fourier series as

$$H_d(\omega) = \sum_{n=-\infty}^{\infty} h_d(n)\, e^{-j\omega n}$$

where the Fourier coefficients $h_d(n)$ (the impulse response) are given by

$$h_d(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(\omega)\, e^{j\omega n}\, d\omega$$

Substituting $H_d(\omega) = j\omega$, we get

$$h_d(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} j\omega\, e^{j\omega n} d\omega = \frac{j}{2\pi}\left[\omega \left.\frac{e^{j\omega n}}{jn}\right|_{-\pi}^{\pi} - \int_{-\pi}^{\pi} 1.\frac{e^{j\omega n}}{jn} d\omega\right]$$

$$= \frac{j}{2\pi}\left[\frac{\left(\pi e^{j\pi n} - (-\pi)e^{-j\pi n}\right)}{jn} - \frac{e^{j\omega n}}{(jn)^2}\Big|_{-\pi}^{\pi}\right]$$

$$= \left[\frac{\left(e^{j\pi n} + e^{-j\pi n}\right)}{2n}\right] - \frac{j}{2\pi}\left[\frac{e^{j\pi n} - e^{-j\pi n}}{j^2 n^2}\right]$$

$$= \frac{\cos n\pi}{n} - \frac{\sin n\pi}{\pi n^2} = \frac{\cos n\pi}{n} - 0, \quad n = \text{non-zero integers}$$

where we have used the fact that $\frac{\sin n\pi}{\pi n^2} = 0$ for non-zero integer values of n. For $n = 0$ the value of $h_d(0)$ is evaluated from the defining equation, viz.,

$$h_d(0) = \frac{1}{2\pi}\int_{-\pi}^{\pi} j\omega\, e^{j\omega 0}\, d\omega = \frac{1}{2\pi}\int_{-\pi}^{\pi} j\omega\, d\omega = \frac{j}{2\pi}\frac{\omega^2}{2}\Big|_{-\pi}^{\pi} = 0$$

Thus

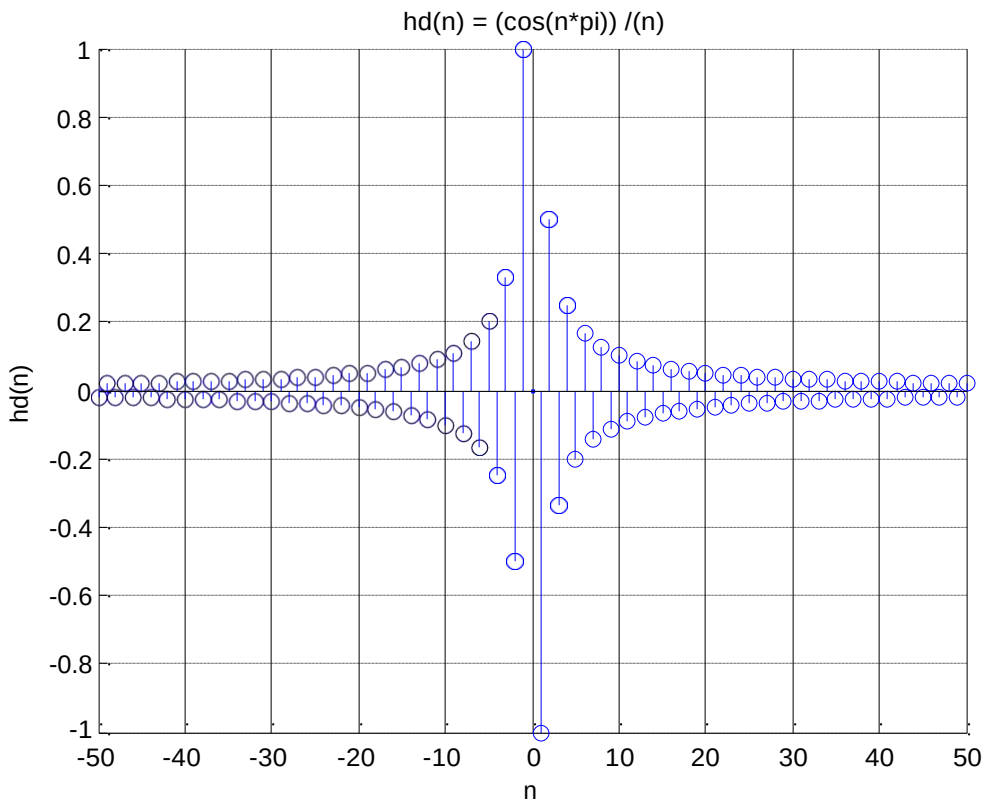
$$h_d(n) = \begin{cases} \dfrac{\cos n\pi}{n}, & n = \text{non-zero integers} \\ 0, & n = 0 \end{cases}$$

As a specific case choose a filter length $N = 9$ and evaluate $h_d(n)$ for $-4 \le n \le 4$.

```
%Ideal Differentiator
% Generate hdn = (cos(n*pi)) /(n) and stem plot
n = -50: 50, hdn = (cos(n*pi)) ./(n), stem(n, hdn)
xlabel('n'), ylabel('hd(n)'); grid; title ('hd(n) = (cos(n*pi)) /(n)')
```

n = –50 to 50
Warning: Divide by zero.
hdn =[-0.0200 0.0204 -0.0208 0.0213 -0.0217 0.0222 -0.0227 0.0233 -0.0238 0.0244 -0.0250 0.0256 -0.0263 0.0270 -0.0278 0.0286 -0.0294 0.0303 -0.0313 0.0323 -0.0333 0.0345 -0.0357 0.0370 -0.0385 0.0400 -0.0417 0.0435 -0.0455 0.0476 -0.0500 0.0526 -0.0556 0.0588 -0.0625 0.0667 -0.0714 0.0769 -0.0833 0.0909 -0.1000 0.1111 -0.1250 0.1429 -0.1667 0.2000 ***-0.2500 0.3333 -0.5000 1.0000 0 -1.0000 0.5000 -0.3333 0.2500*** -0.2000 0.1667 -0.1429 0.1250 -0.1111 0.1000 -0.0909 0.0833 -0.0769 0.0714 -0.0667 0.0625 -0.0588 0.0556 -0.0526 0.0500 -0.0476 0.0455 -0.0435 0.0417 -0.0400 0.0385 -0.0370 0.0357 -0.0345 0.0333 -0.0323 0.0313 -0.0303 0.0294 -0.0286 0.0278 -0.0270 0.0263 -0.0256 0.0250 -0.0244 0.0238 -0.0233 0.0227 -0.0222 0.0217 -0.0213 0.0208 -0.0204 0.0200]

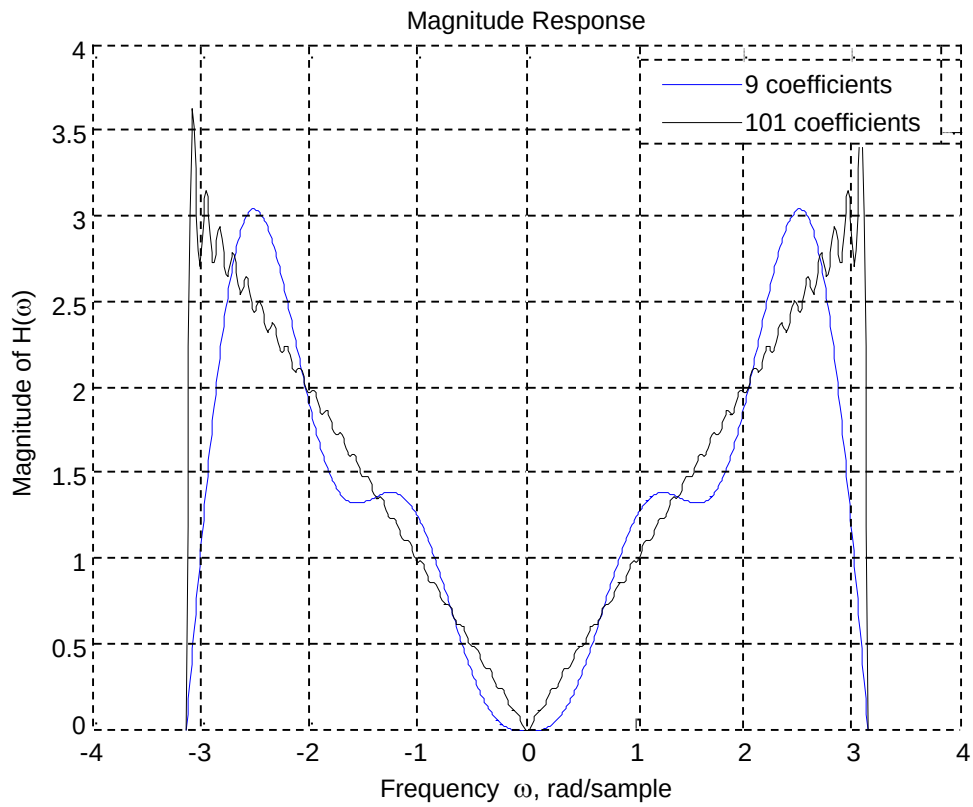
We can see the odd symmetry of $h_d(n)$ from the following stem plot (note that the correct value of $h_d(0) = 0$; the MATLAB segment used here has $h_d(0) = \infty$ which is not correct).



The frequency response

```
%Ideal Differentiator
%9-tap filter coefficients
b9=[ -0.2500    0.3333   -0.5000    1.0000       0   -1.0000    0.5000  -0.3333
0.2500],
a=[1]
%101-tap filter coefficients
b101=[-0.0200    0.0204   -0.0208    0.0213   -0.0217    0.0222   -0.0227    0.0233
-0.0238    0.0244   -0.0250    0.0256   -0.0263    0.0270   -0.0278    0.0286  -
0.0294    0.0303    -0.0313    0.0323   -0.0333    0.0345   -0.0357    0.0370   -
0.0385    0.0400   -0.0417    0.0435   -0.0455    0.0476   -0.0500    0.0526   -
0.0556    0.0588   -0.0625    0.0667    -0.0714    0.0769   -0.0833    0.0909   -
0.1000    0.1111   -0.1250    0.1429   -0.1667    0.2000   -0.2500    0.3333   -
0.5000    1.0000       0   -1.0000    0.5000   -0.3333    0.2500   -0.2000    0.1667 -
0.1429    0.1250   -0.1111    0.1000   -0.0909    0.0833   -0.0769    0.0714   -
0.0667    0.0625   -0.0588    0.0556   -0.0526    0.0500   -0.0476    0.0455   -
0.0435    0.0417   -0.0400    0.0385   -0.0370    0.0357   -0.0345    0.0333   -
0.0323    0.0313   -0.0303    0.0294   -0.0286    0.0278   -0.0270    0.0263   -
0.0256    0.0250   -0.0244    0.0238   -0.0233    0.0227   -0.0222    0.0217   -
0.0213    0.0208   -0.0204    0.0200],
w=-pi: pi/256: pi;
Hw9=freqz(b9, a, w);
Hw101=freqz(b101, a, w);
plot(w, abs(Hw9), w, abs(Hw101), 'k')
legend ('9 coefficients', '101 coefficients');
```

```
title('Magnitude Response');
xlabel('Frequency \omega, rad/sample'), ylabel('Magnitude of H(\omega)'); grid
```



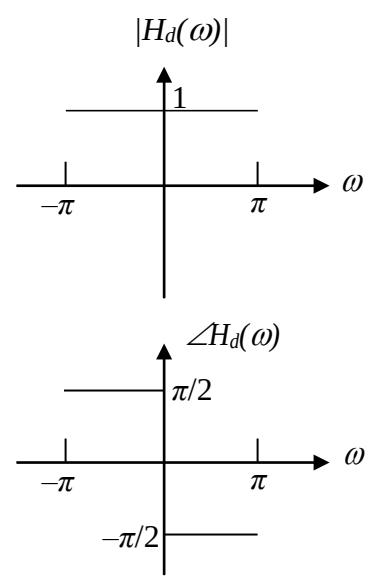
Example 4.4.9 [The Hilbert transformer] This is used to generate signals that are in *phase quadrature* to an input sinusoidal signal (or, more generally, an input narrowband waveform). That is, if the input to a Hilbert transformer is the signal $x_a(t) = \cos \Omega_0 t$, the output is $y_a(t) = \sin \Omega_0 t$. The Hilbert transformer is used in communication systems in various modulation schemes.

The frequency response of the Hilbert transformer is (Figure)

$$H_d(e^{j\omega}) = -j\,\mathrm{sgn}(\omega),\ -\pi \le \omega \le \pi$$

where the sgn(ω) is the signum function defined as

$$\mathrm{sgn}(\omega) = \begin{cases} 1 & \text{if } \omega \text{ is positive} \\ -1 & \text{if } \omega \text{ is negative} \end{cases}$$



Since $j = e^{j\pi/2}$ we may also express $H_d(e^{j\omega})$ also as

$$H_d(e^{j\omega}) = \begin{cases} -e^{j\pi/2} = -j = 1 \text{ at } \angle -\pi/2, & 0 \le \omega \le \pi \\ e^{j\pi/2} = j = 1 \text{ at } \angle \pi/2, & -\pi \le \omega \le 0 \end{cases}$$

Note that the magnitude $|H_d(e^{j\omega})| = 1$ for all ω and $\angle H_d(\omega)$ changes from $\pi/2$ to $-\pi/2$ at $\omega = 0$.

The filter coefficients are the Fourier coefficients given by

$$h_d(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(\omega) e^{j\omega n} d\omega = \frac{1}{2\pi}\left(\int_{-\pi}^{0} j\, e^{j\omega n} d\omega + \int_{0}^{\pi} (-j)\, e^{j\omega n} d\omega\right)$$

$$= \frac{j}{2\pi}\left(\left.\frac{e^{j\omega n}}{jn}\right|_{-\pi}^{0}\right) + \frac{(-j)}{2\pi}\left(\left.\frac{e^{j\omega n}}{jn}\right|_{0}^{\pi}\right)$$

$$= \frac{1}{2\pi n}\left((1-e^{-j\pi n}) - (e^{j\pi n} - 1)\right) = \frac{1}{2\pi n}\left(2 - e_{-j\pi n} - e^{j\pi n}\right)$$

$$= \frac{2 - 2\cos n\pi}{2\pi n} = \frac{1-\cos n\pi}{n\pi}, \qquad n \ne 0$$

For $n = 0$, $h_d(0)$ may be evaluated from the defining equation:

$$h_d(0) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(\omega)\, e^{j\omega 0} d\omega = \frac{1}{2\pi}\left(\int_{-\pi}^{0} j1 d\omega + \int_{0}^{\pi} (-j)1 d\omega\right)$$

$$= \frac{j}{2\pi}\left(\left.\omega^{0}\right|_{-\pi}\right) - \frac{j}{2\pi}\left(\left.\omega^{\pi}\right|_{0}\right) = 0$$

Thus

$$h_d(n) = \begin{cases} 2/n\pi, & n \text{ odd} \\ 0, & n \text{ even (including 0)} \end{cases}$$

As a specific case choose a filter length $N = 9$ and evaluate $h_d(n)$ for $-4 \le n \le 4$.

Generate the filter coefficients:

```
%Hilbert Transform
%Generate hdn = 2/(n*pi) for odd n & hdn = 0 for even n, and stem plot
n = -50: 50, hdn = 2 ./(n*pi), stem(n, hdn)
```

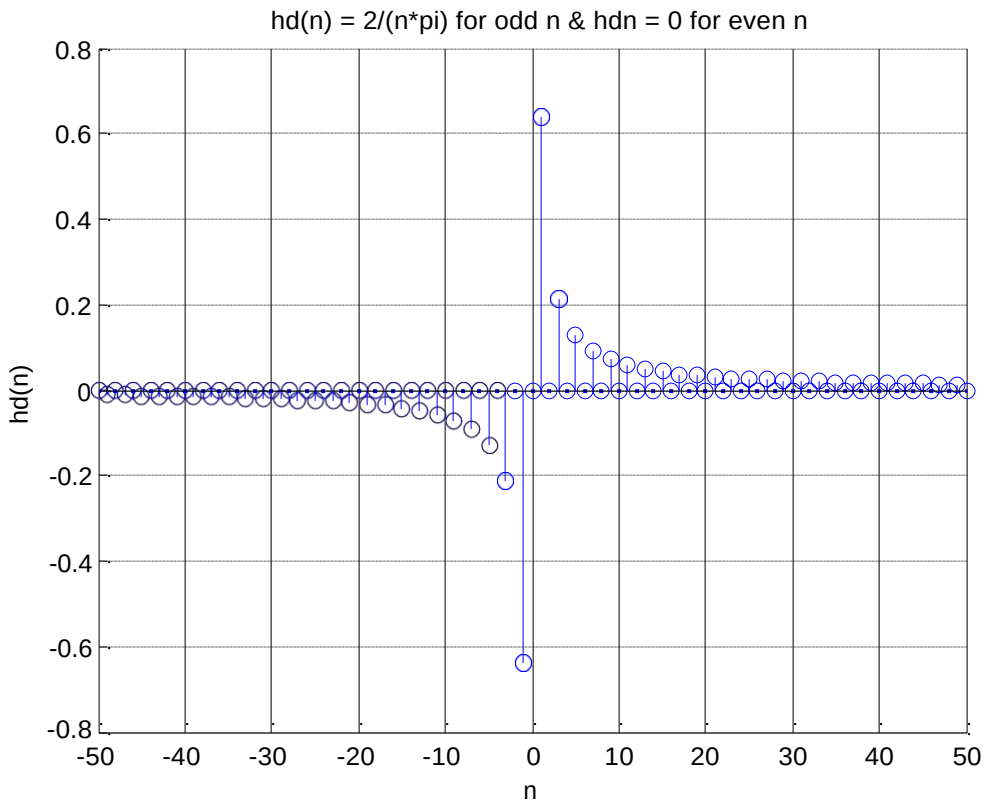
n = –50 to 50 (Entered hdn = 0 by hand for even n)
Warning: Divide by zero.

hdn =[0 -0.0130 0 -0.0135 0 -0.0141 0 -0.0148 0 -0.0155 0 -
0.0163 0 -0.0172 0 -0.0182 0 -0.0193 0 -0.0205 0 -0.0220 0 -
0.0236 0 -0.0255 0 -0.0277 0 -0.0303 0 -0.0335 0 -0.0374 0 -
0.0424 0 -0.0490 0 -0.0579 0 -0.0707 0 -0.0909 0 -0.1273 ***0*** -
0.2122 0 -0.6366 Inf 0.6366 0 0.2122 0 0.1273 0 0.0909 0
0.0707 0 0.0579 0 0.0490 0 0.0424 0 0.0374 0 0.0335 0
0.0303 0 0.0277 0 0.0255 0 0.0236 0 0.0220 0 0.0205 0
0.0193 0 0.0182 0 0.0172 0 0.0163 0 0.0155 0 0.0148 0
0.0141 0 0.0135 0 0.0130 0]

We copy and paste the above coefficients and do a stem plot:

```
%Hilbert Transform
```

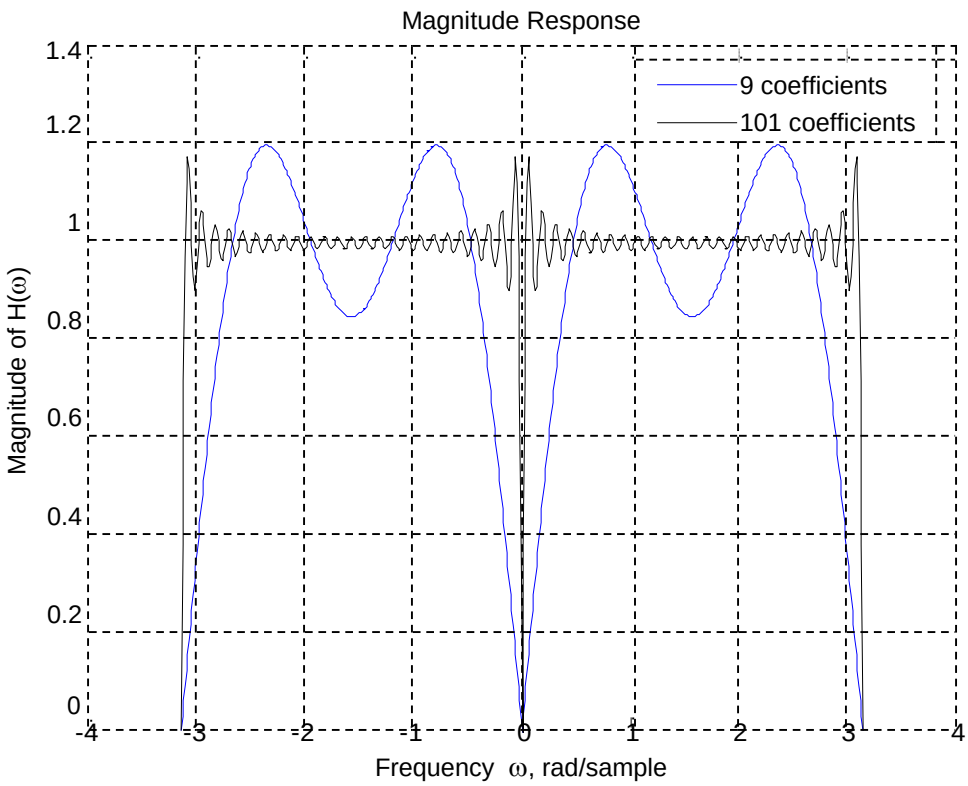
```
%Stem plot hdn = 2/(n*pi) for odd n & hdn = 0 for even n
n = -50: 50,
hdn =[0 -0.0130 0 -0.0135 0 -0.0141 0 -0.0148 0 -0.0155 0  -0.0163
0 -0.0172 0 -0.0182 0 -0.0193 0 -0.0205 0 -0.0220 0 -0.0236 0 -
0.0255   0   -0.0277  0    -0.0303  0    -0.0335  0    -0.0374  0    -0.0424  0    -
0.0490   0   -0.0579  0    -0.0707  0    -0.0909  0    -0.1273  0    -0.2122  0    -
0.6366      0    0.6366    0    0.2122    0    0.1273    0    0.0909    0    0.0707    0
0.0579    0    0.0490    0    0.0424    0    0.0374    0    0.0335    0    0.0303    0
0.0277    0    0.0255    0    0.0236    0    0.0220    0    0.0205    0    0.0193    0
0.0182    0    0.0172    0    0.0163    0    0.0155    0    0.0148    0    0.0141    0
0.0135    0    0.0130    0],
stem(n, hdn)
xlabel('n'), ylabel('hd(n)'); grid;
title ('hd(n) = 2/(n*pi) for odd n & hdn = 0 for even n')
```



Frequency response

```
%Hilbert Transform
%Comparison of 9 coefficients vs. 101 coefficients
%9-tap filter coefficients
b9=[0   -0.2122   0   -0.6366       0    0.6366    0    0.2122    0],
a=[1]
b101 =[ 0 -0.0130 0 -0.0135 0 -0.0141 0 -0.0148 0 -0.0155 0 -
0.0163   0   -0.0172   0   -0.0182   0   -0.0193   0   -0.0205   0   -0.0220   0   -
0.0236   0   -0.0255   0   -0.0277   0   -0.0303   0   -0.0335   0   -0.0374   0   -
0.0424   0   -0.0490   0   -0.0579   0   -0.0707   0   -0.0909   0   -0.1273   0   -
0.2122   0   -0.6366       0    0.6366    0    0.2122    0    0.1273    0    0.0909    0
```

```
0.0707   0   0.0579   0   0.0490   0   0.0424   0   0.0374   0   0.0335   0
0.0303   0   0.0277   0   0.0255   0   0.0236   0   0.0220   0   0.0205   0
0.0193   0   0.0182   0   0.0172   0   0.0163   0   0.0155   0   0.0148   0
0.0141   0   0.0135   0   0.0130   0]
w=-pi: pi/256: pi;
Hw9=freqz(b9, a, w);
Hw101=freqz(b101, a, w);
plot(w, abs(Hw9), w, abs(Hw101), 'k')
legend ('9 coefficients', '101 coefficients');
title('Magnitude Response');
xlabel('Frequency \omega, rad/sample'), ylabel('Magnitude of H(\omega)'); grid
```



Window functions [Ref. S.K. Mitra] We want to see the frequency behavior of window functions by themselves. Note that depending on convenience we shall define the window functions either over $-(N-1)/2 \le n \le (N-1)/2$ or over $0 \le n \le (N-1)$. In the former case the phase function will be zero; in the latter case, used especially in MATLAB, the phase has a negative slope.

There are fixed windows and adjustable windows. Among the fixed windows we have (in addition to the rectangular window) the following tapered windows:

1. Bartlett (triangular) window
2. Hann (aka Hanning or von Hann) window
3. Hamming window
4. Blackman window

These windows result in a fixed amount of ripple in the frequency response of the designed filter. The Kaiser window is an adjustable window which allows some control over the ripple.

Bartlett: $w(n) = 1 - \dfrac{2|n|}{N-1}, \qquad -(N-1)/2 \le n \le (N-1)/2$

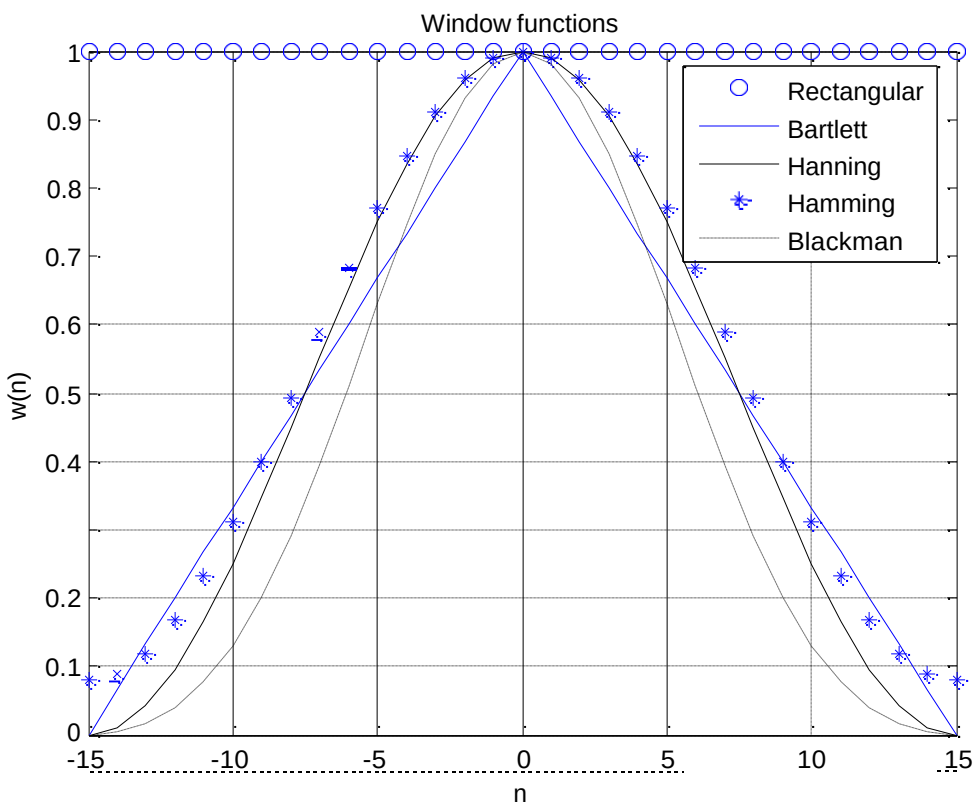
Hann: $w(n) = 0.5 + 0.5\cos\left(\dfrac{2\pi n}{N-1}\right), \quad -(N-1)/2 \le n \le (N-1)/2$

Hamming: $w(n) = 0.54 + 0.46\cos\left(\dfrac{2\pi n}{N-1}\right), \quad -(N-1)/2 \le n \le (N-1)/2$

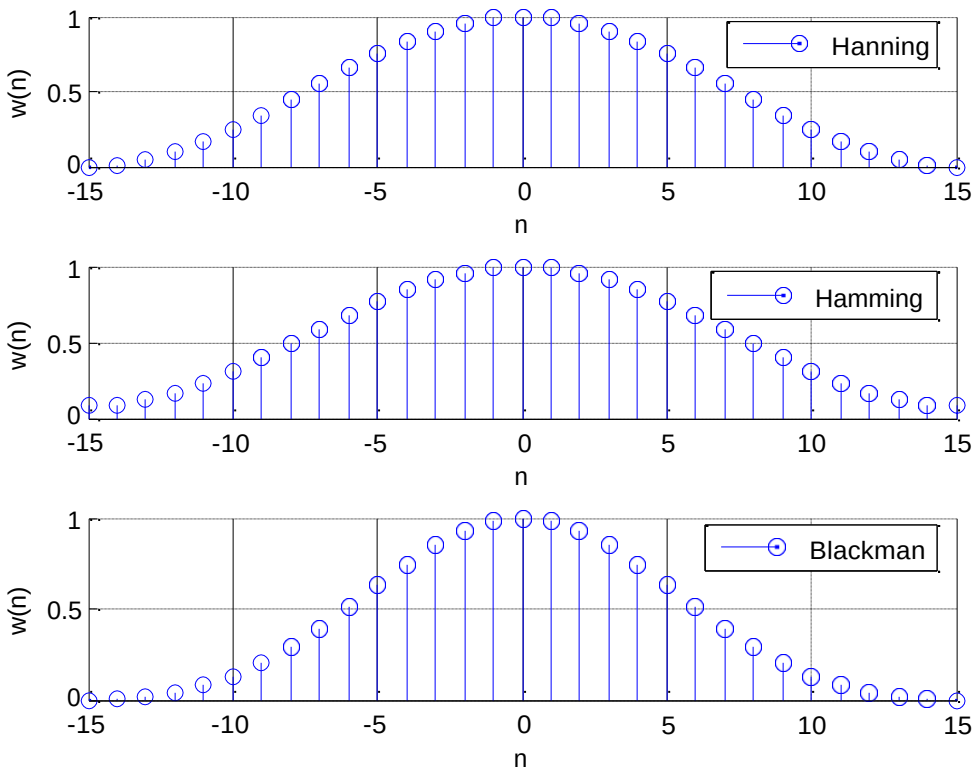
Blackman: $w(n) = 0.42 + 0.5\cos\left(\dfrac{2\pi n}{N-1}\right) + 0.08\cos\left(\dfrac{4\pi n}{N-1}\right), \quad -(N-1)/2 \le n \le (N-1)/2$

The following MATLAB multiplot gives a graphical comparison of the above window functions. Note that all are discrete sequences; to make it easy on the eyes some are plotted as continuous lines and some as discrete.

```
%Window functions defined over n = –(N–1)/2 to (N–1)/2
N = 31; n = -(N-1)/2: (N-1)/2;
wR = n-n+1; %Rectangular
wBa = 1 - 2* abs(n)/(N-1); %Bartlett
wHn = 0.5 + 0.5 * cos(2*pi*n/(N-1)); %Hamming
wHm = 0.54 + 0.46 * cos(2*pi*n/(N-1)); %Hamming
wBl = 0.42 + 0.5 * cos(2*pi*n/(N-1)) + 0.08 * cos(4*pi*n/(N-1)); %Blackman
%Multiplot. All are discrete sequences.
%To make it easy on the eyes some are plotted with continuous lines.
plot (n, wR, 'o', n, wBa, 'b', n, wHn, 'k', n, wHm, 'b*', n, wBl, 'k--');
legend ('Rectangular', 'Bartlett', 'Hanning', 'Hamming', 'Blackman');
xlabel('n'), ylabel('w(n)'); grid; title ('Window functions')
```

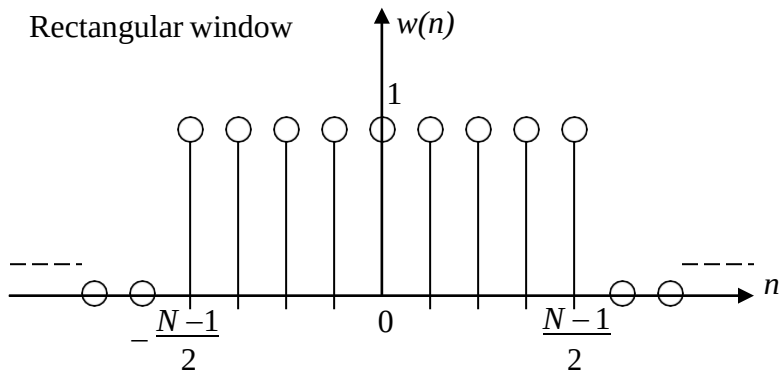


```
%Window functions defined over n = –(N–1)/2 to (N–1)/2
N = 31; n = -(N-1)/2: (N-1)/2;
wHn = 0.5 + 0.5 * cos(2*pi*n/(N-1)); %Hanning
wHm = 0.54 + 0.46 * cos(2*pi*n/(N-1)); %Hamming
wBl = 0.42 + 0.5 * cos(2*pi*n/(N-1)) + 0.08 * cos(4*pi*n/(N-1)); %Blackman
%
subplot(3, 1, 1), stem(n, wHn); legend ('Hanning');
xlabel('n'), ylabel('w(n)'); grid
subplot(3, 1, 2), stem(n, wHm); legend ('Hamming');
xlabel('n'), ylabel('w(n)'); grid
subplot(3, 1, 3), stem(n, wBl); legend ('Blackman');
xlabel('n'), ylabel('w(n)'); grid
```

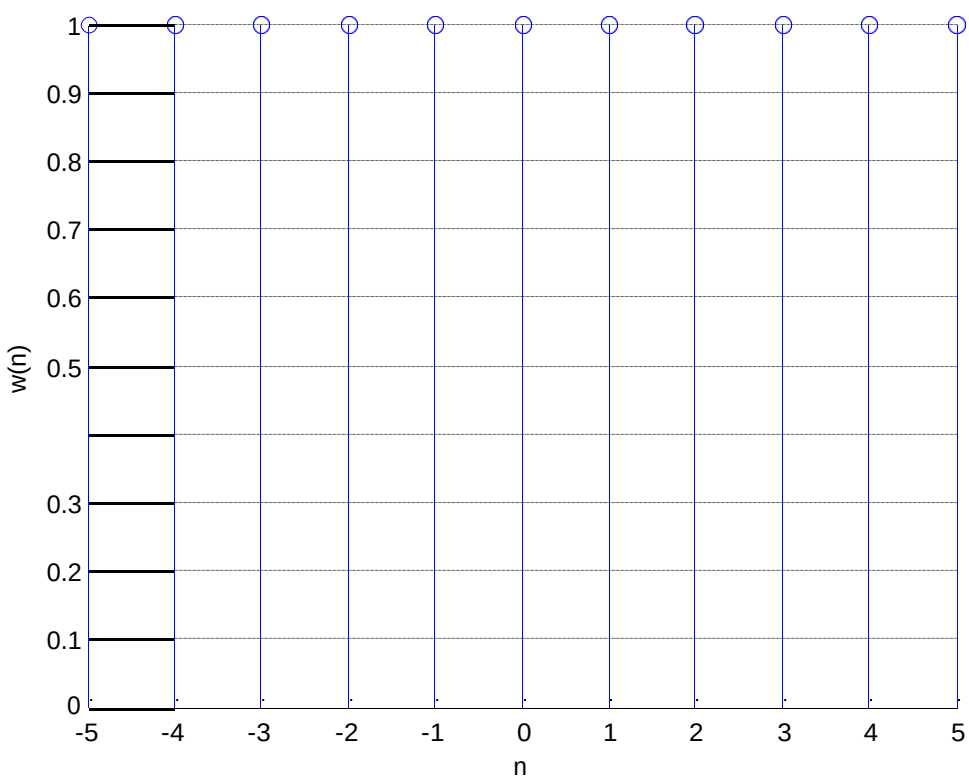


The **rectangular window** defined over $-(N-1)/2 \le n \le (N-1)/2$ is given by

$$w_R(n) = \begin{cases} 1, & -(N-1)/2 \le n \le (N-1)/2 \\ 0, & \text{elsewhere} \end{cases}$$




```
%Rectangular window defined over n = –(N–1)/2 to (N–1)/2
% w(n) = {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}
N = 11; n = -(N-1)/2: (N-1)/2; wn = n-n+1;
stem (n, wn); xlabel('n'), ylabel('w(n)'); grid
```



The Fourier transform (spectrum) of the window is

$$W\left(e^{j\omega}\right) = \sum_{n=-\infty}^{\infty} w(n) e^{-j\omega n} = \sum_{n=-(N-1)/2}^{(N-1)/2} 1 e^{-j\omega n}$$

We can simplify the above in one of two ways. One possibility is

$$W(e^{j\omega}) = e^{j\omega\left(\frac{N-1}{2}\right)} + e^{j\omega\left(\frac{N-3}{2}\right)} + \ldots + e^{j\omega} + 1 + e^{-j\omega} + \ldots + e^{-j\omega\left(\frac{N-1}{2}\right)}$$

$$= e^{j\omega\left(\frac{N-1}{2}\right)} \underbrace{\left(1 + e^{-j\omega} + e^{-j2\omega} + \ldots + e^{-j\omega(N-1)}\right)}_{N \text{ terms}} = e^{j\omega\left(\frac{N-1}{2}\right)} \left[\frac{1 - e^{-j\omega N}}{1 - e^{-j\omega}}\right]$$

$$= e^{j\omega\left(\frac{N-1}{2}\right)} \left[\frac{e^{-j\omega N/2}\left(e^{j\omega N/2} - e^{-j\omega N/2}\right)}{e^{-j\omega/2}\left(e^{j\omega/2} - e^{-j\omega/2}\right)}\right] = \frac{\left(e^{j\omega N/2} - e^{-j\omega N/2}\right)}{\left(e^{j\omega/2} - e^{-j\omega/2}\right)}$$

$$= \frac{\sin(\omega N/2)}{\sin(\omega/2)}$$

Using L'Hopital's rule,

$$W(e^{j\omega})\Big]_{\omega=0} = \frac{(N/2)\cos(\omega N/2)}{(1/2)\cos(\omega/2)}\Bigg|_{\omega=0} = N$$

$$20\log_{10}\left|W(e^{j\omega})\Big]_{\omega=0}\right| = 20\log_{10} N$$

In frequency response plots this is normally regarded as a reference point, that is, as the 0 dB level. That is, the expression for $W(e^{j\omega})$ is *normalized* by dividing it by its value at dc, N in this case:

$$W(e^{j\omega}) = \frac{1}{N}\frac{\sin(\omega N/2)}{\sin(\omega/2)}$$

The second method to simplify $W(e^{j\omega})$ is to combine pairs of terms, one from each end of the expression, into a cosine (or sine, in the case of odd symmetry):

$$W(e^{j\omega}) = \sum_{n=-\infty}^{\infty} w(n)e^{-j\omega n} = \sum_{n=-(N-1)/2}^{(N-1)/2} 1e^{-j\omega n}$$

$$= e^{j\omega\left(\frac{N-1}{2}\right)} + e^{j\omega\left(\frac{N-3}{2}\right)} + \ldots + e^{j\omega} + 1 + e^{-j\omega} + \ldots + e^{-j\omega\left(\frac{N-1}{2}\right)}$$

$$= 1 + 2\cos\omega + 2\cos 2\omega + \ldots + 2\cos\left(\frac{N-1}{2}\omega\right)$$

This consists of $\left(1 + \frac{N-1}{2}\right)$ terms and should be normalized by dividing by N.

Using the equation $W(e^{j\omega}) = \frac{1}{N}\frac{\sin(\omega N/2)}{\sin(\omega/2)}$, its zero-crossings occur when $(\omega N/2)$ equals integer multiples of π, that is,

$$(\omega N/2) = \pm k\pi \qquad \text{or} \qquad \omega = k(2\pi/N), \quad k \neq 0$$

The spectrum between the zero-crossings at $-(2\pi/N)$ and $(2\pi/N)$ is called the ***main lobe***, the remaining lobes are called ***side lobes***. The width of the main lobe is

Width of the main lobe = $2(2\pi/N) = 4\pi/N$
Width of each side lobe = $2\pi/N$

As the length of the window, N, is increased the lobes become narrower; also the height of the main lobe increases (= N). However, with reference to the *normalized* frequency response

$$W(e^{j\omega}) = \frac{1}{N}\frac{\sin(\omega N/2)}{\sin(\omega/2)}$$

the height of the main lobe, $W(e^{j\omega})\Big]_{\omega=0}$ stays at 1 (or 0 dB) while the side lobes keep getting smaller with increasing N.

In the MATLAB segment below the window is defined over $0 \le n \le (N-1)$ rather than over $-(N-1)/2 \le n \le (N-1)/2$.

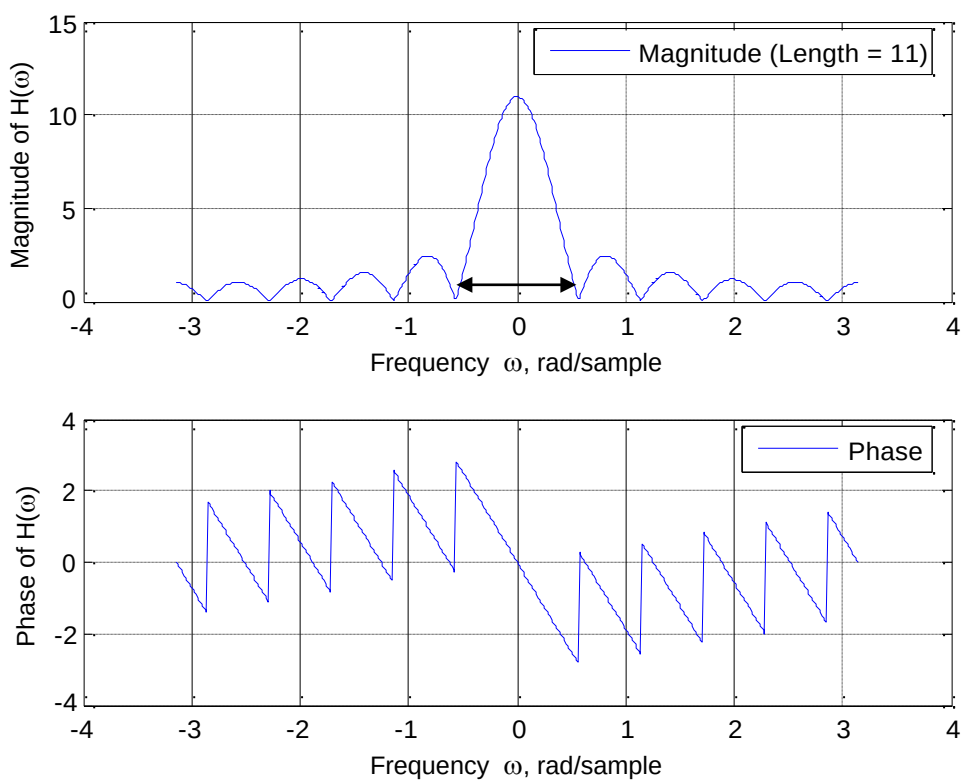
```
%Frequency response of rectangular window defined over n = 0 to N–1
% h(n) = {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}
b11= [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
a= [1]
w=-pi: pi/256: pi;
Hw11=freqz(b11, a, w);
subplot(2, 1, 1), plot(w, abs(Hw11)); legend ('Magnitude (Length = 11)');
xlabel('Frequency \omega, rad/sample'), ylabel('Magnitude of H(\omega)'); grid
subplot(2, 1, 2), plot(w, angle(Hw11)); legend ('Phase');
xlabel('Frequency \omega, rad/sample'), ylabel('Phase of H(\omega)'); grid
```

Note in the plot below that the height of the main lobe is 11 (= N). The width of the main lobe is taken as the separation between the zero crossings on either side of $\omega = 0$:

Width of main lobe = $4\pi/N = 4\pi/11$

From the plot, by eyeballing, we can gather:

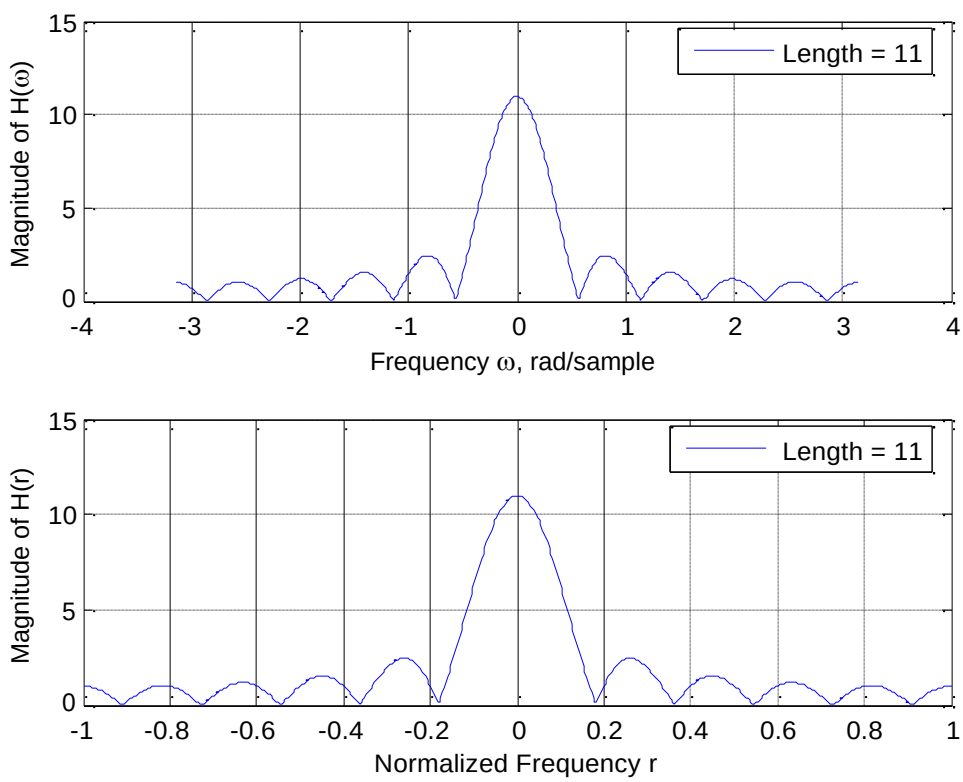
1. For a given window length the side lobes have the same width.
2. For a given window length the magnitude of the side lobes decreases with increasing frequency



Normalized frequency *r* In the MATLAB segment below the window is defined over $0 \le n \le (N-1)$. Further we define the normalized frequency $r = \omega/\pi$. As ω varies from $-\pi$ to π the normalized frequency varies from –1 to 1.

```
%Magnitude response of rectangular window defined over n = 0 to N–1
% h(n) = {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}
b11=[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
a=[1]
w=-pi: pi/256: pi; r = w/pi;
Hw11=freqz(b11, a, w);
subplot(2, 1, 1), plot(w, abs(Hw11)); legend ('Length = 11');
xlabel('Frequency \omega, rad/sample'), ylabel('Magnitude of H(\omega)'); grid
%
%Normalized frequency r = ω/π
Hr11=freqz(b11, a, pi*r);
subplot(2, 1, 2), plot(r, abs(Hr11)); legend (' Length = 11');
xlabel('Normalized Frequency r'), ylabel('Magnitude of H(r)'); grid
```

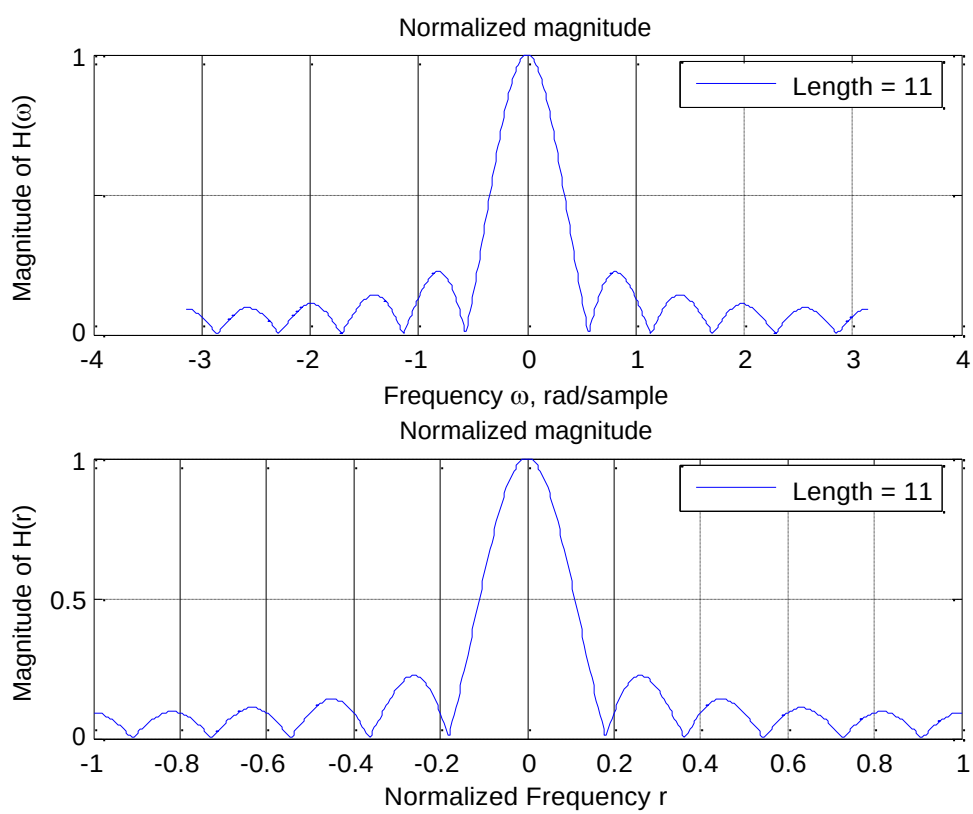
Note in the plot below that as ω goes from $-\pi$ to π the normalized frequency r goes from –1 to 1.



Normalized magnitude In the MATLAB segment below we go another step: we normalize the magnitude by dividing it by N. The window is defined over $0 \le n \le (N-1)$ and the normalized frequency is $r = \omega/\pi$. As ω varies from $-\pi$ to π the normalized frequency varies from –1 to 1.

```
%Magnitude response rectangular window defined over n = 0 to N–1
% h(n) = {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}
N = 11; b11= ones(1, 11); a=[1];
w=-pi: pi/256: pi; r = w/pi;
%
%Normalized magnitude H(ω)/N
Hw11n= freqz(b11, a, w)/N;
subplot(2, 1, 1), plot(w, abs(Hw11n)); legend (' Length = 11');
xlabel('Frequency \omega, rad/sample'), ylabel('Magnitude of H(\omega)');grid;
title ('Normalized magnitude')
%
%Normalized magnitude and normalized frequency
Hr11n= freqz(b11, a, pi*r)/N;
subplot(2, 1, 2), plot(r, abs(Hr11n)); legend (' Length = 11');
xlabel('Normalized Frequency r'), ylabel('Magnitude of H(r)'); grid;
title ('Normalized magnitude')
```

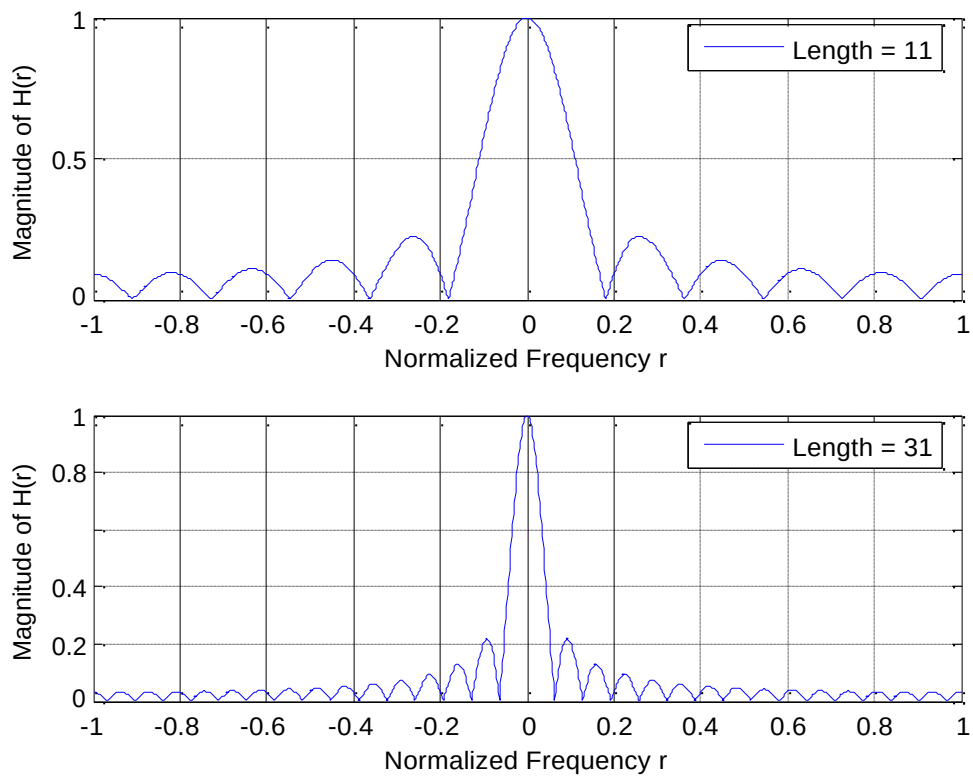
Note in the plot below that the height of the main lobe is 1 since it is normalized.



Comparison of two rectangular windows of lengths 11 and 31 In the MATLAB segment below we go another step: we normalize the magnitude by dividing it by N. The window is defined over $0 \le n \le (N-1)$ and the normalized frequency is $r = \omega/\pi$. As ω varies from $-\pi$ to π the normalized frequency varies from –1 to 1.

```
%Magnitude response rectangular window defined over n = 0 to N–1
% Comparison of two rectangular windows of lengths 11 and 31
N1 = 11; b11= ones(1, N1); N2 = 31; b31= ones(1, N2); a=[1];
w=-pi: pi/512: pi; r = w/pi;
%
%Length 11, normalized magnitude and normalized frequency
Hr11n= freqz(b11, a, pi*r )/N1;
subplot(2, 1, 1), plot(r, abs(Hr11n)); legend ('Length = 11');
xlabel('Normalized Frequency r'), ylabel('Magnitude of H(r)'); grid
%
%Length 31, normalized magnitude and normalized frequency
Hr31n= freqz(b31, a, pi*r)/N2;
subplot(2, 1, 2), plot(r, abs(Hr31n)); legend ('Length = 31');
xlabel('Normalized Frequency r'), ylabel('Magnitude of H(r)'); grid
```

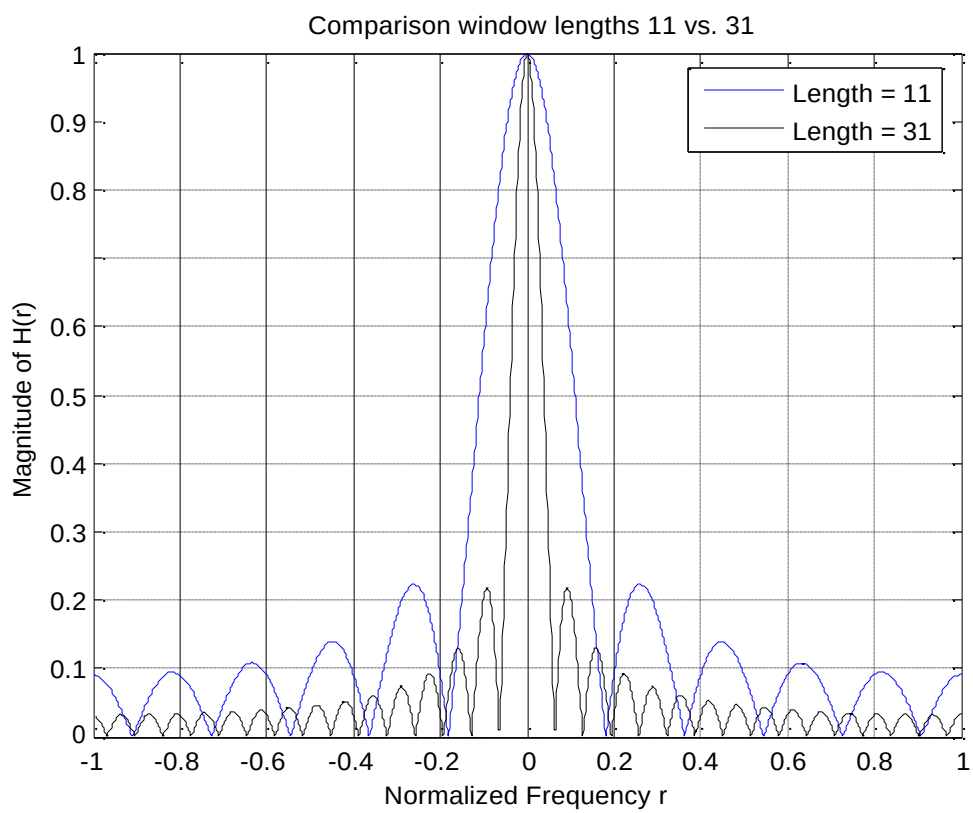
Only the *first* side lobe is of concern since all the other side lobes are smaller. From the plot below the maximum of the first side lobe is a little over 20% of the height of the main lobe in both windows.



Two rectangular windows of lengths 11 and 31 compared on a multi-plot In the MATLAB segment below we compare the two window lengths on the same multi-plot. As before, the window is defined over $0 \le n \le (N-1)$ and the normalized frequency is $r = \omega/\pi$. As ω varies from $-\pi$ to π the normalized frequency varies from –1 to 1.

```
%Magnitude response rectangular window defined over n = 0 to N–1
% Comparison of two rectangular windows of lengths 11 and 31
N1 = 11; b11= ones(1, N1); N2 = 31; b31= ones(1, N2); a=[1];
w=-pi: pi/768: pi; r = w/pi;
%
%Length 11, normalized magnitude and normalized frequency
Hr11n= freqz(b11, a, pi*r )/N1;
%
%Length 31, normalized magnitude and normalized frequency
Hr31n= freqz(b31, a, pi*r)/N2;
%
plot(r, abs(Hr11n), r, abs(Hr31n), 'k'); legend ('Length = 11', 'Length = 31');
title('Comparison window lengths 11 vs. 31');
xlabel('Normalized Frequency r'), ylabel('Magnitude of H(r)'); grid
```

From the plot, by eyeballing, we can gather that for a given window length the magnitude of the side lobes decreases with increasing frequency. Further, as the window length increases the height of a specific side lobe (such as the first side lobe) decreases: for instance the height of the first side lobe for $N = 31$ is smaller than that of the first side lobe for $N = 11$.

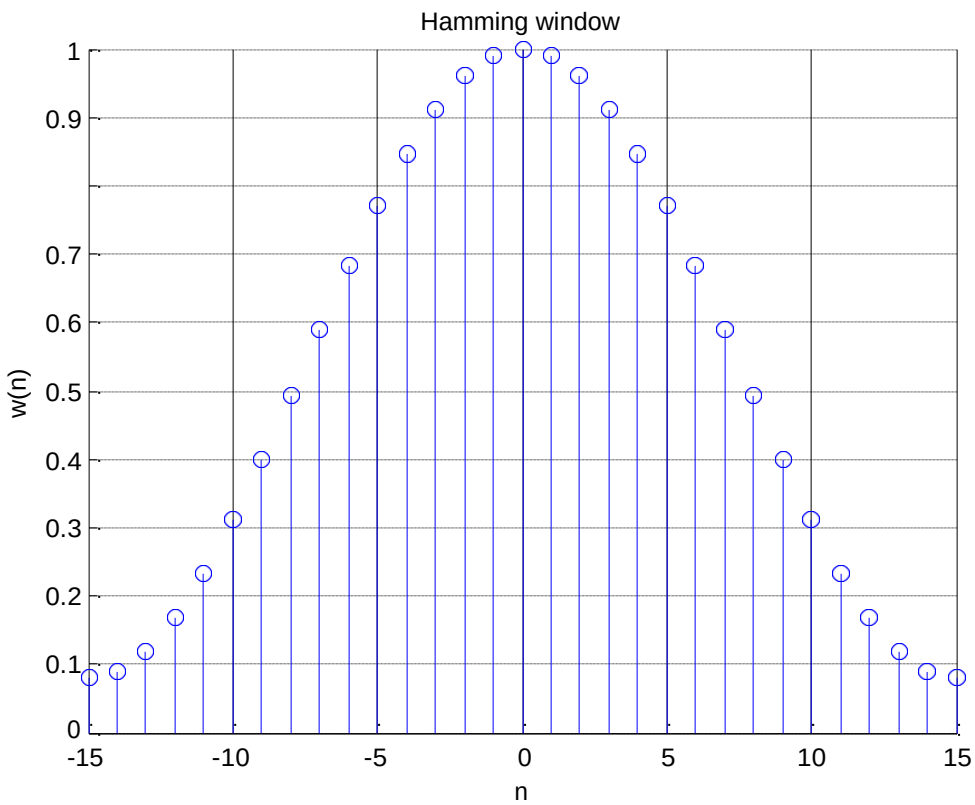


The **Hamming window** defined over $-(N-1)/2 \le n \le (N-1)/2$ is

$$w_{Ham}(n) = \begin{cases} 0.54 + 0.46\cos[2\pi n/(N-1)], & -(N-1)/2 \le n \le (N-1)/2 \\ 0, & \text{elsewhere} \end{cases}$$

```
%Hamming window defined over n = –(N–1)/2 to (N–1)/2
% w(n) = 0.54 + 0.46 cos(2*pi*n/(N-1))
N = 31; n = -(N-1)/2: (N-1)/2; wn = 0.54 + 0.46 * cos(2*pi*n/(N-1)),
stem (n, wn); xlabel('n'), ylabel('w(n)'); grid; title ('Hamming window')
```

n = -5 -4 -3 -2 -1 0 1 2 3 4 5
w(n) = {0.0800 0.0901 0.1198 0.1679 0.2322 0.3100 0.3979 0.4919 0.5881
0.6821 0.7700 0.8478 0.9121 0.9602 0.9899 1.0000 0.9899 0.9602 0.9121
0.8478 0.7700 0.6821 0.5881 0.4919 0.3979 0.3100 0.2322 0.1679 0.1198
0.0901 0.0800}



The Fourier transform (spectrum) of the window is

$$W(e^{j\omega}) = \sum_{n=-\infty}^{\infty} w(n)e^{-j\omega n} = \sum_{n=-(N-1)/2}^{(N-1)/2} \{0.54 + 0.46\cos 2\pi n/(N-1)\}e^{-j\omega n}$$

$$= \sum_{n=-(N-1)/2}^{(N-1)/2} 0.54 e^{-j\omega n} + \sum_{n=-(N-1)/2}^{(N-1)/2} 0.46\{\cos 2\pi n/(N-1)\}e^{-j\omega n}$$

$$= 0.54\,\frac{\sin(\omega N/2)}{\sin(\omega/2)} + \sum_{n=-(N-1)/2} 0.46\left(\frac{e^{j2\pi n/(N-1)} + e^{-j2\pi n/(N-1)}}{2}\right)e^{-j\omega n}$$

$$= \ldots$$

$$= 0.54\ \frac{\sin(\omega N/2)}{\sin(\omega/2)} + 0.23\frac{\sin[(\omega N/2) - \pi N/(N-1)]}{\sin[(\omega/2) - \pi/(N-1)]} + 0.23\ \frac{\sin[(\omega N/2) + \pi N/(N-1)]}{\sin[(\omega/2) + \pi/(N-1)]}$$

The magnitude at dc is

$$W(e^{j\omega})\Big]_{\omega=0} = 0.54\,N + 2\,(0.23)\ \frac{\sin[\pi N/(N-1)]}{\sin[\pi/(N-1)]}$$

This is calculated below for N = 11, 21, 31 and 41:

```
%Magnitude at DC
N = 11:10:41,
WdcN = 0.54*N + 0.46* sin(pi*N./(N-1))./sin(pi./(N-1))
```

```
N =     11    21    31    41
WdcN =     5.4800 10.8800 16.2800 21.6800
```

The width of the main lobe is taken as the separation between the zero crossings on either side of $\omega = 0$. This is obtained by setting $W\left(e^{j\omega}\right) = 0$ and solving for ω; it is given as

Width of main lobe (Hamming) = $8\pi/N$

twice that of the rectangular window.

The Hamming window, defined over the interval $0 \le n \le N-1$, is given by

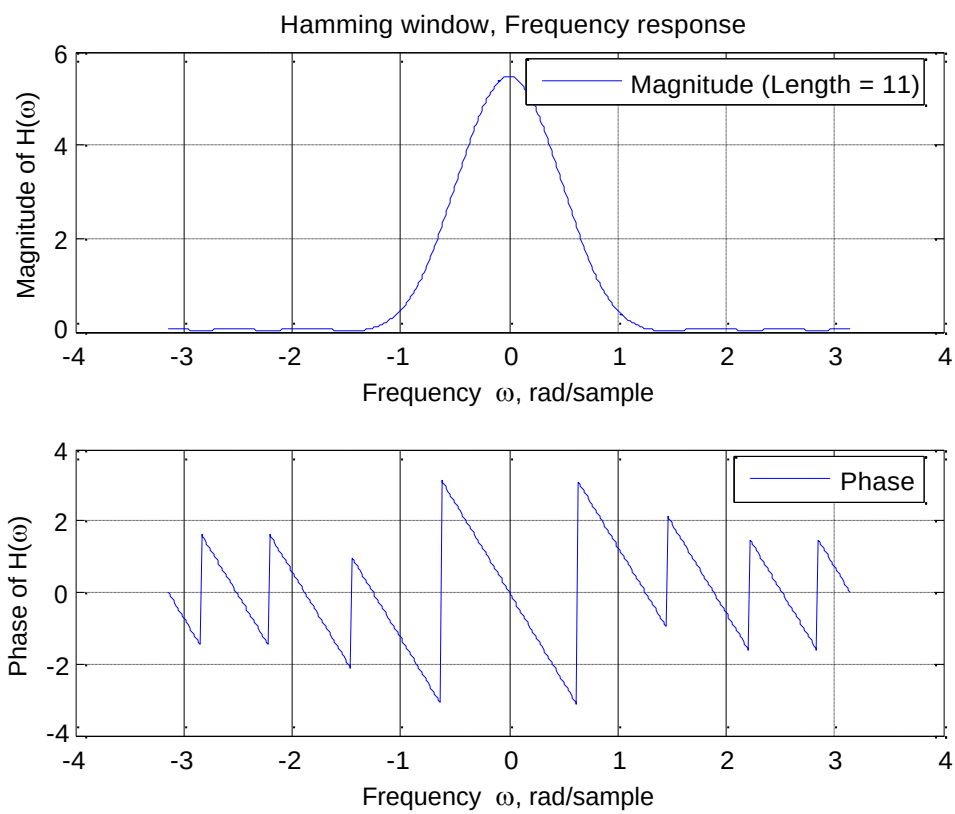
$$w_{Ham}(n) = \begin{cases} 0.54 - 0.46\cos[2\pi n/(N-1)], & 0 \le n \le N-1 \\ 0, & \text{elsewhere} \end{cases}$$

For $N = 11$, the Hamming window is given by $w_{Ham}(n) = 0.54 - 0.46\cos(\pi n / 5)$, $0 \le n \le 10$.

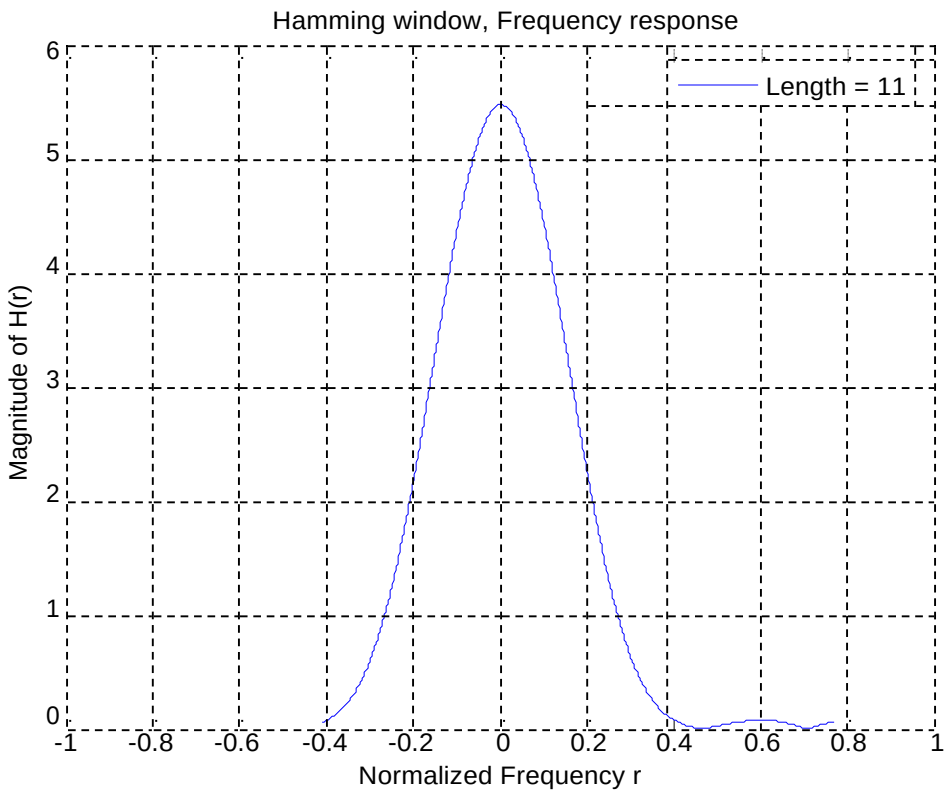
Frequency response of Hamming window:

```
%Magnitude response of 11-point Hamming window defined over n = 0 to N–1
% WHam = b11 = 0.54 - 0.46 *cos((2*pi/(N-1) .*n))
N = 11; n = 0: N-1; b11 = 0.54 - 0.46 *cos((2*pi/(N-1) .*n));
a=[1]
w=-pi: pi/256: pi;
Hw11=freqz(b11, a, w);
subplot(2, 1, 1), plot(w, abs(Hw11)); legend ('Magnitude (Length = 11)');
xlabel('Frequency \omega, rad/sample'), ylabel('Magnitude of H(\omega)'); grid
title('Hamming window, Frequency response');
subplot(2, 1, 2), plot(w, angle(Hw11)); legend ('Phase');
xlabel('Frequency \omega, rad/sample'), ylabel('Phase of H(\omega)'); grid
```

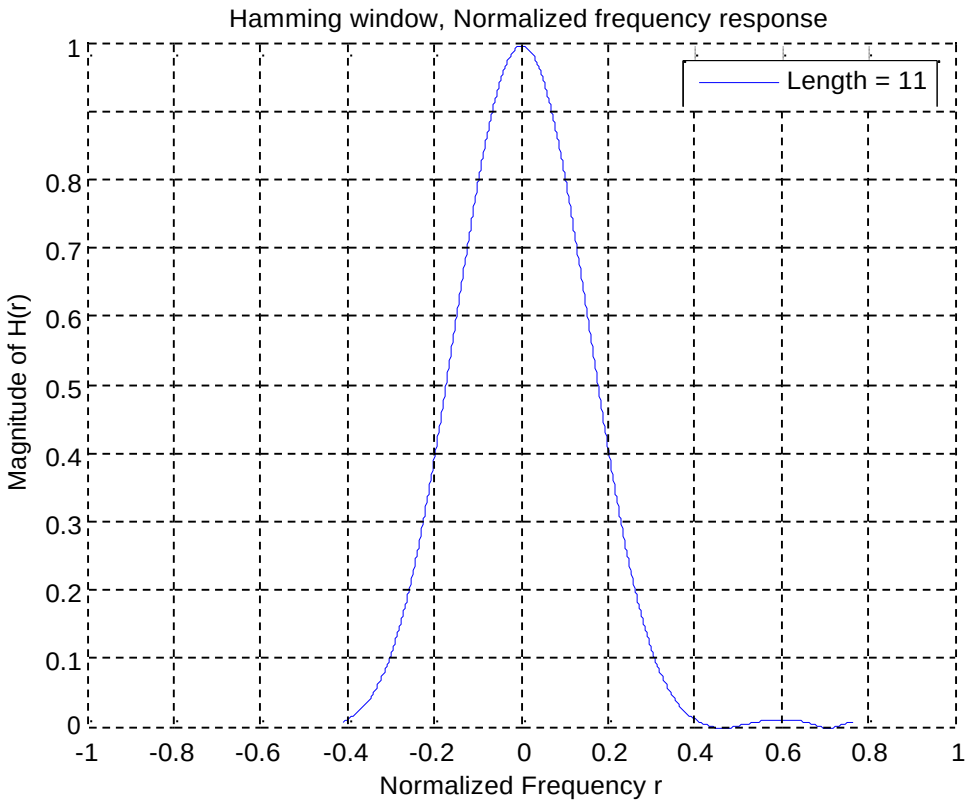
In the phase plot below the phase reaches $-\pi$ in the interval $0 \le \omega \le 1$ and is therefore adjusted by adding 2π; this is not due to a zero-crossing. The same applies to the phase adjustment in the interval $-1 \le \omega \le 0$.



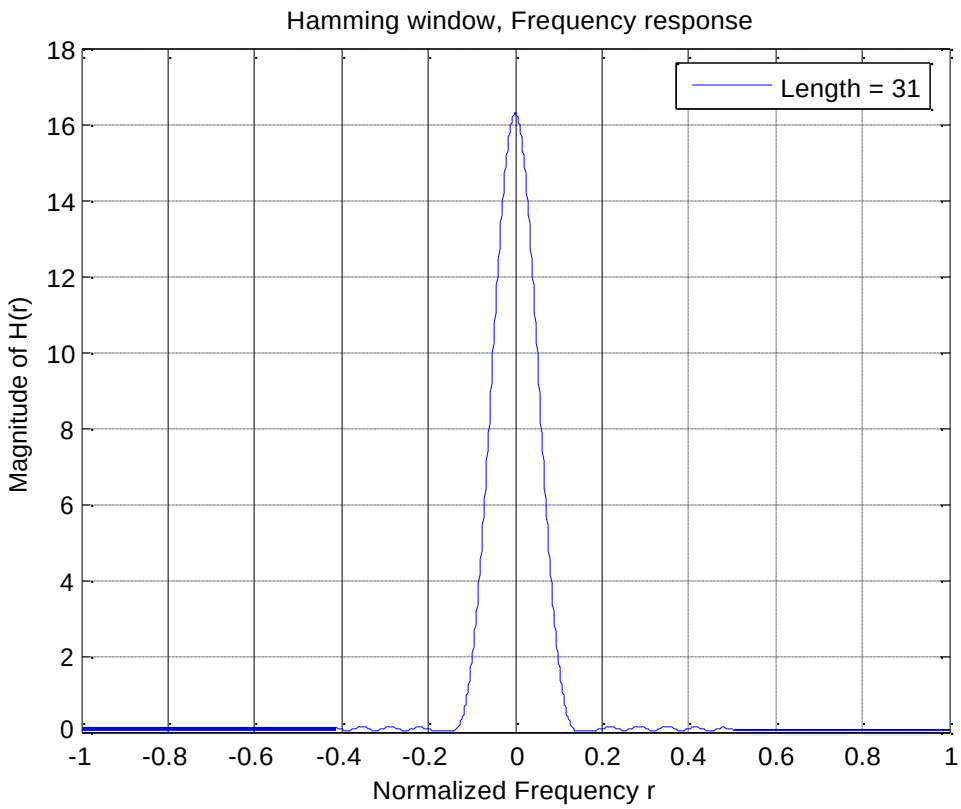
```
%Magnitude response of Hamming window defined over n = 0 to N–1
%Length 11
N = 11; n = 0: N-1; b11 = 0.54 - 0.46 *cos((2*pi/(N-1) .*n)); a=[1];
w=-pi: pi/768: pi; r = w/pi; % normalized frequency
Hr11n= freqz(b11, a, pi*r); % Frequency response
% Plot
plot(r, abs(Hr11n)); legend ('Length = 11');
title('Hamming window, Frequency response');
xlabel('Normalized Frequency r'), ylabel('Magnitude of H(r)'); grid
```



```
%Magnitude response of Hamming window defined over n = 0 to N–1
%Length 11
N = 11; n = 0: N-1; b11 = 0.54 - 0.46 *cos((2*pi/(N-1) .*n)); a=[1];
w=-pi: pi/768: pi; r = w/pi; % normalized frequency
%Magnitude at dc
WdcN = 0.54*N + 0.46* sin(pi*N/(N-1))/sin(pi/(N-1)),
Hr11n= freqz(b11, a, pi*r)/WdcN; % normalized frequency response
% Plot
plot(r, abs(Hr11n)); legend ('Length = 11');
title('Hamming window, Normalized frequency response');
xlabel('Normalized Frequency r'), ylabel('Magnitude of H(r)'); grid
```



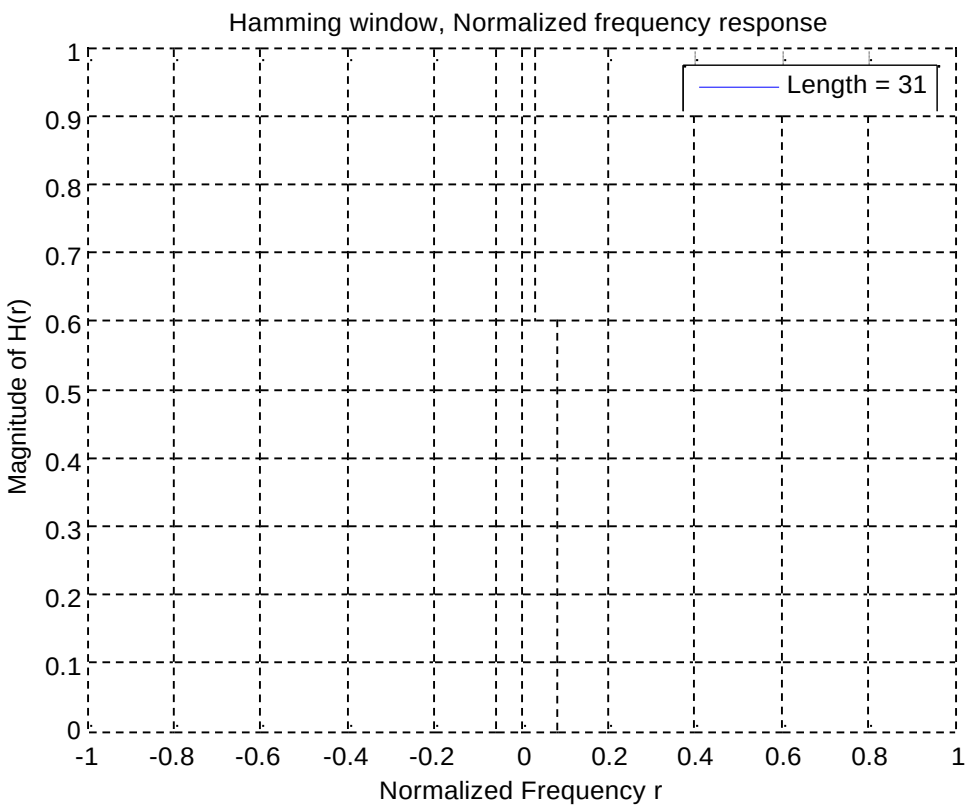
```
%Magnitude response of Hamming window defined over n = 0 to N–1
%Length 31
N = 31; n = 0: N-1; b31 = 0.54 - 0.46 *cos((2*pi/(N-1) .*n)); a=[1];
w=-pi: pi/768: pi; r = w/pi; % normalized frequency
Hr31n= freqz(b31, a, pi*r); % Frequency response
% Plot
plot(r, abs(Hr31n)); legend ('Length = 31');
title('Hamming window, Frequency response');
xlabel('Normalized Frequency r'), ylabel('Magnitude of H(r)'); grid
```



```
%Magnitude response of Hamming window defined over n = 0 to N–1
%Length 31
N = 31; n = 0: N-1; b31 = 0.54 - 0.46 *cos((2*pi/(N-1) .*n)); a=[1];
w=-pi: pi/2048: pi; r = w/pi; % normalized frequency
%Magnitude at dc
WdcN = 0.54*N + 0.46* sin(pi*N/(N-1))/sin(pi/(N-1)),
Hr31n= freqz(b31, a, pi*r)/WdcN; % normalized frequency response
% Plot
plot(r, abs(Hr31n)); legend ('Length = 31');
title('Hamming window, Normalized frequency response');
xlabel('Normalized Frequency r'), ylabel('Magnitude of H(r)'); grid
```

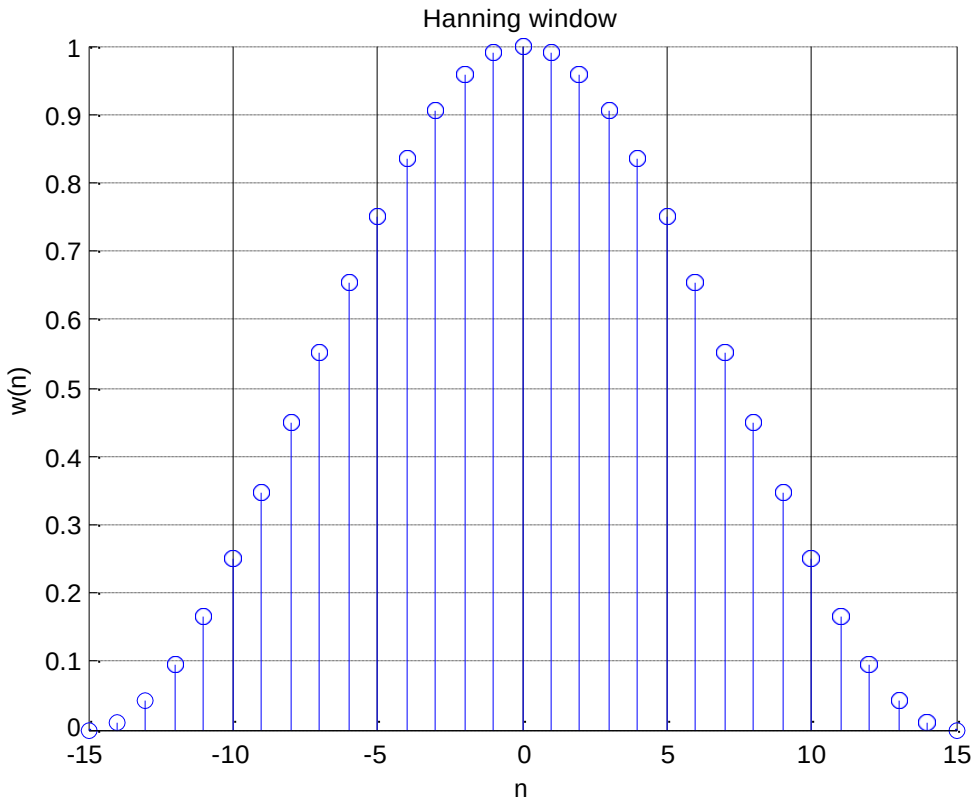
Observations:

1. In the plot below the magnitude of the side lobes appears to stay about the same with increasing frequency for the first few lobes, but beyond $r = 0.6$ there is a discernible (to the eye) fall-off in the height of the side lobes. Relative to the rectangular window, however, we may say the side lobe height stays about the same with increasing frequency.
2. The width of the side lobe appears to stay constant with increasing frequency.



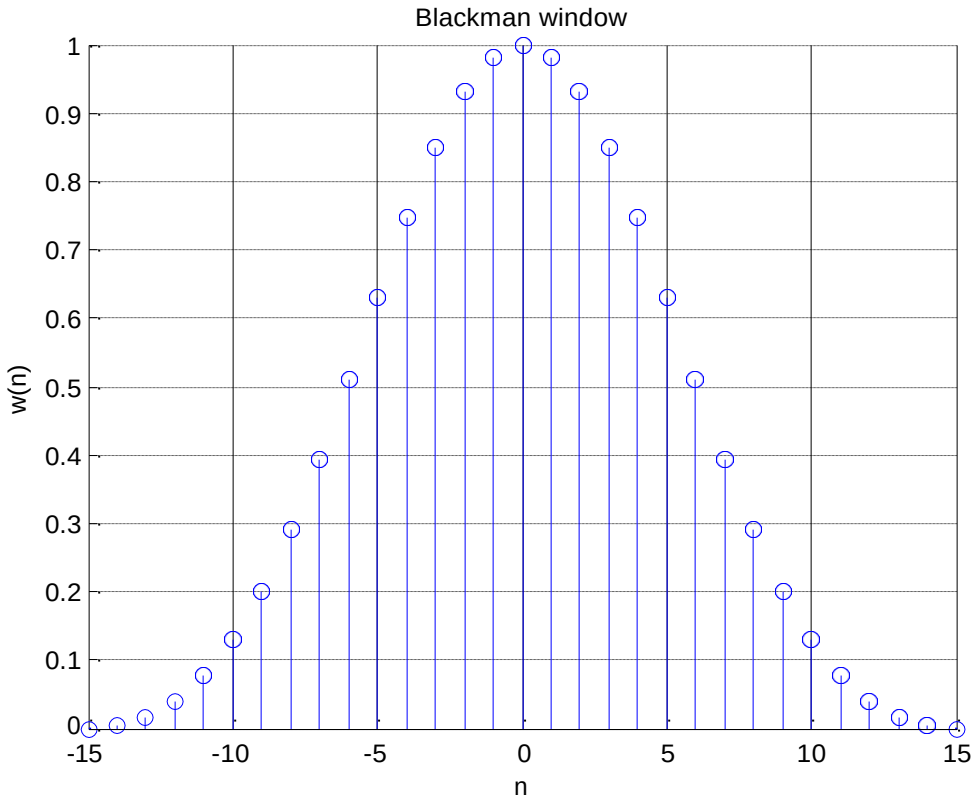
Hanning window

```
%Hanning window defined over n = –(N–1)/2 to (N–1)/2
% w(n) = 0.5 + 0.5 * cos(2*pi*n/(N-1))
N = 31; n = -(N-1)/2: (N-1)/2; wn = 0.5 + 0.5 * cos(2*pi*n/(N-1)),
stem (n, wn); xlabel('n'), ylabel('w(n)'); grid; title ('Hanning window')
```



Blackman window

```
%Blackman window defined over n = –(N–1)/2 to (N–1)/2
% w(n) = 0.5 + 0.5 * cos(2*pi*n/(N-1))
N = 31; n = -(N-1)/2: (N-1)/2;
wn = 0.42 + 0.5 * cos(2*pi*n/(N-1)) + 0.08 * cos(4*pi*n/(N-1)),
stem (n, wn); xlabel('n'), ylabel('w(n)'); grid; title ('Blackman window')
```



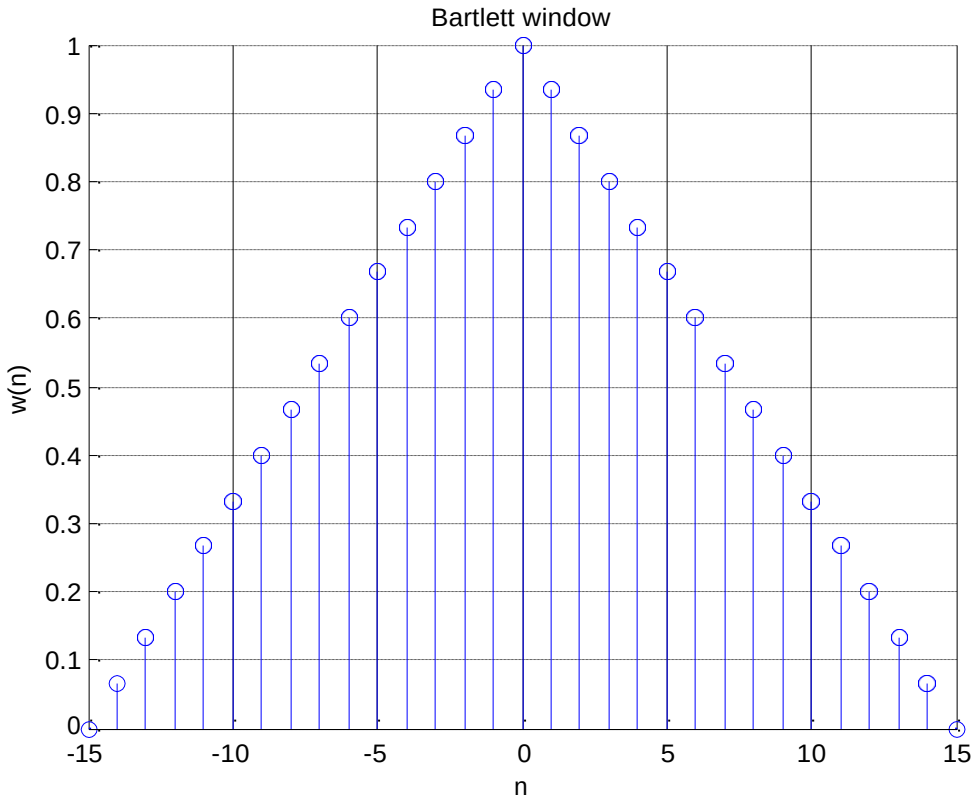
The width of the main lobe is taken as the separation between the zero crossings on either side of $\omega = 0$. This is obtained by setting $W\left(e^{j\omega}\right) = 0$ and solving for ω; it is given as

Width of main lobe (Blackman) = $12\pi/N$

thrice that of the rectangular window.

Bartlett window

```
%Bartlett window defined over n = –(N–1)/2 to (N–1)/2
% w(n) = 1 - 2* abs(n)/(N-1)
N = 31; n = -(N-1)/2: (N-1)/2; wn = 1 - 2* abs(n)/(N-1),
stem (n, wn); xlabel('n'), ylabel('w(n)'); grid; title ('Bartlett window')
```



4.5 Choosing between FIR and IIR filters

IIR Filter	FIR Filter
Use IIR when the only important requirements are 1. **sharp cutoff frequency** and 2. **high throughput** as IIR filters, especially those using elliptic characteristics, will have fewer coefficients than FIR	Use FIR if 1. the **number of coefficients is not a problem** and 2. in particular, if **very little or no phase distortion** is desired Some DSP processors have architectures that are tailored/designed for FIR filters.

The choice between FIR and IIR filters depends largely on the relative advantages of the two types of filters.

(1) Linear phase FIR filters can have exactly linear phase response. This means no phase distortion. This is an important requirement in, for example, data transmission, biomedicine, digital audio, and image processing.

The phase response of IIR filters is nonlinear, especially at the band edges.

(2) Stability FIR filters realized non-recursively are always stable. The stability of IIR filters cannot always be guaranteed.

(3) The effects of finite word length such as round off noise and coefficient quantization errors are much less severe in FIR than in IIR.

(4) FIR requires more coefficients for sharp cut-off filters than IIR. Thus for a given amplitude response specification, more processing time and storage will be required for FIR implementation. However, one can readily take advantage of the computational speed of the FFT and multirate techniques to improve significantly the efficiency of FIR implementation.

(5) Analog filters can be readily transformed into equivalent IIR digital filters meeting similar specifications. This is not possible with FIR filters as they have no analog counterpart. However, with FIR it is easier to synthesize filters of arbitrary frequency response.

(6) In general, FIR is algebraically more difficult to synthesize, if CAD support is not available.

5 Multirate DSP

Decimation, Interpolation, Sampling rate conversion, Filter design and-Implementation of sampling rate conversion.

Contents:

Discrete-time systems with different sampling rates at various parts of the system are called **multirate systems**. They are *linear and time-varying systems*. Integer sampling rate converters change the sampling frequency by an integer factor and rational sampling rate converters by a rational number. Here is a sampling of sampling rates in commercial applications (Mitra):

Sampling Rates	
Digital Audio	**Video**
Broadcasting – 32 kHz **CD – 44.1 kHz** **DAT – 48 kHz**	**Composite Video Signal** • **NTSC – 14.3181818 MHz** • **PAL – 17.734475 MHz** **Digital Component Video Signal** • **Luminance – 13.5 MHz** • **Color difference – 6.75 MHz**

Time and frequency scaling in continuous-time systems

Illustration An audio signal recorded on cassette tape at a certain speed could be played back at a higher speed than that at which it was recorded. This is called *time scaling*, in particular, *compression* in the time domain, and results in an inverse effect in the frequency domain, i.e., an expansion of the frequency spectrum. Similarly when the audio signal is played back at a slower speed than the recording speed we have *expansion* in the time domain resulting in a corresponding compression of the spectrum in the frequency domain.

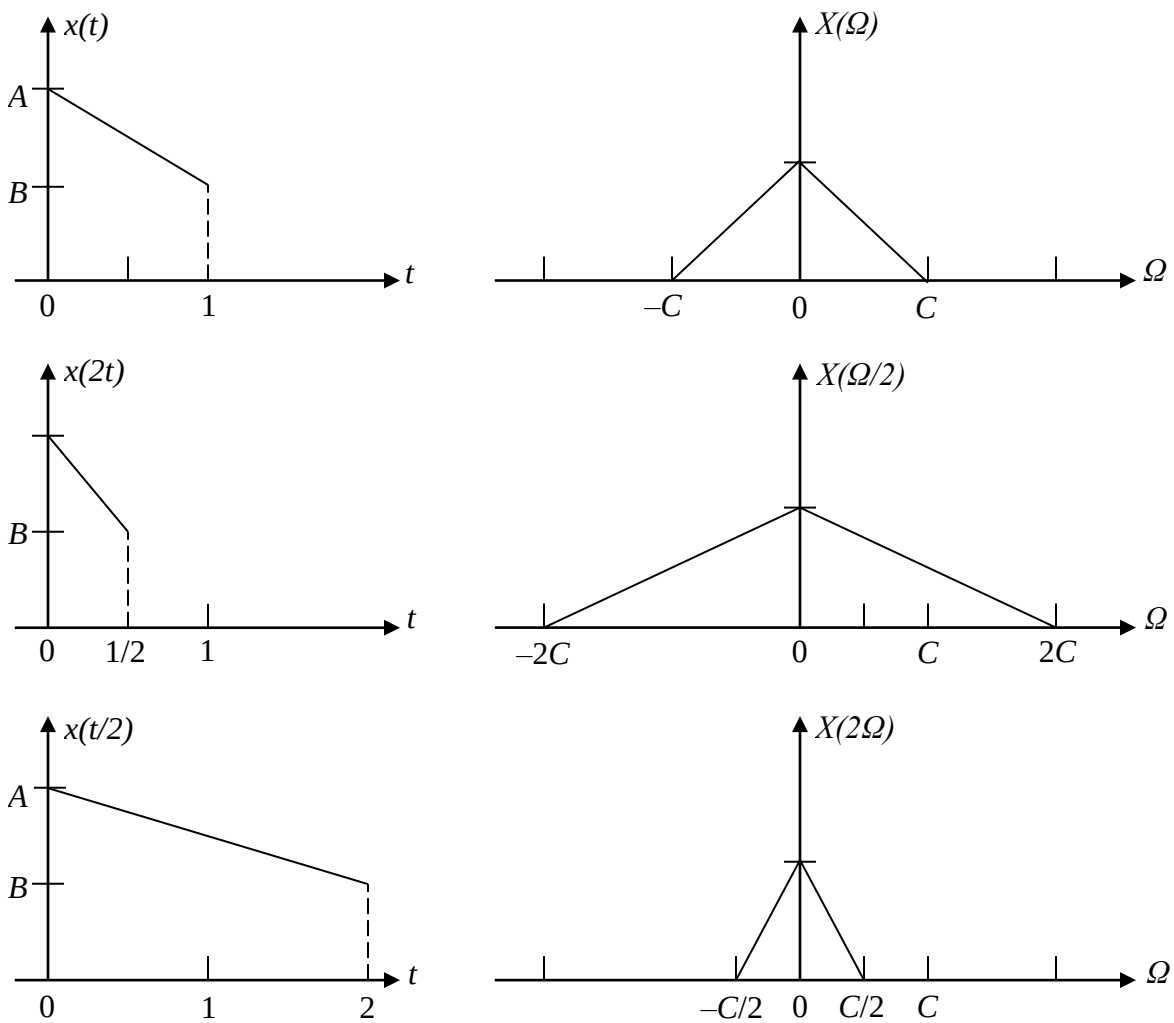
Given the signal $x(t)$ and its Fourier transform $X(\Omega)$, represented notationally by

$$x(t) \text{ — } X(\Omega)$$

then time scaling results in

$$x(at) \text{ — } \frac{1}{|a|} X(\Omega/a)$$

If $a > 1$ the scaling corresponds to **compression in time**. If, for instance, $a = 2$, we may visualize a new signal $y_1(t) = x(2t)$; with $t = 1$, for instance, the value of x, that is, $x(2)$ that occurred at 2 seconds occurs at 1 second in the case of y_1, that is, $y_1(1)$ – which is compression in time.



If $x(t)$ is an audio signal *recorded* on tape then $x(2t)$ could be the signal $x(t)$ *played back* at twice the speed at which $x(t)$ was recorded. The signal $x(2t)$ varies more rapidly than $x(t)$ and the playback frequencies are higher.

If $a < 1$ the scaling corresponds to **expansion in time**. If, for instance, $a = 1/2$, then $x(t/2)$ is the signal $x(t)$ played back at half the speed at which $x(t)$ was recorded. The signal $x(t/2)$ varies slower than $x(t)$ and the playback frequencies are lower. Again, we may visualize this as a new signal $y_2(t) = x(t/2)$; the value of $x(.)$ that occurred at $t/2$ occurs at t in the case of $y_2(.)$ – which is expansion in time.

Time expansion and frequency compression is found in data transmission from space probes to receiving stations on earth. To reduce the amount of noise superposed on the signal, it is necessary to keep the bandwidth of the receiver as small as possible. One means of doing this is to reduce the bandwidth of the signal: store the data collected by the probe, and then transmit it at a slower rate. Since the time-scaling factor is known, the signal can be reproduced at the receiver.

The corresponding operations in the case of discrete-time systems are not quite so straight forward owing to

1. The need to band limit the continuous-time signal prior to sampling, and
2. The need to avoid aliasing in the process of sampling

Example 5.1 Consider the 4 Hz signal $x(t) = \cos 2\pi 4t$ which is obviously band-limited to $F_{max} = 4$ Hz. It is sufficient to sample it at 8 Hz. Alternatively, the signal can be sampled at, say, 16 Hz or 20 Hz etc. Suppose that it has been *over-sampled* by a factor of, say, 6 at $F_s = 48$ Hz to give $x(n) = \cos 2\pi 4n(1/48) = \cos(\pi n/6)$.

(a) If it is desired subsequently to generate from $x(n)$ another signal $x_1(n)$ that is a discrete-time version of $x(t)$ sampled at $F_{s1} = 16$ Hz (sampling rate reduced by a factor of 3), then can we do this by simply dropping two samples of $x(n)$ for every sample that we keep? That is $x_1(n) = x(3n)$. This is called **down-sampling**.
(b) How do we generate from $x(n)$ another signal $x_2(n)$ that is a discrete-time version of $x(t)$ sampled at, say, $F_{s2} = 96$ Hz (sampling rate doubled)? This is called **up-sampling**.
(c) Can we generate from $x(n)$ another signal $x_3(n)$ that is a discrete-time version of $x(t)$ sampled at $F_{s3} = 6$ Hz?

We pick up on this problem again after covering transformation of the independent variable.

Transformation of the independent variable

Time scaling (Refer also to Section 7.5 of Signals and Systems, Oppenheim and Willsky.) Given the sequence $x(n)$, the sequence $y(n) = x(2n)$ is obtained by skipping odd-numbered samples in $x(n)$ and retaining the even-numbered ones. The extension to $y(n) = x(Mn)$ means we retain sample numbers 0, M, $2M$, $3M$, …, and skip the intervening M–1 samples between those we keep. The original sequence $x(n)$ is obtained by sampling a continuous signal $x(t)$ at a certain rate (perhaps over-sampling). The signal $y(n) = x(Mn)$ is then obtained by reducing the sampling rate by a factor of M on the continuous-time signal $x(t)$. This is known as **down-sampling** or **decimation** or sampling rate compression.

Similarly the process of constructing the sequence $y(n) = x(n/L)$ from the sequence $x(n)$ means we derive $y(n)$ by inserting (L–1) sequence points with zero value between points of $x(n)$. This is called **up-sampling** or **interpolation** or sampling rate expansion. (Inserting (L–1) zeros is

just one way of interpolating. It is also possible for the up-sampler to be followed by a digital system that replaces the inserted zeros with more appropriate values based on a linear combination of the *x(n)* samples.)

In general, the result of time scaling a discrete-time signal is not just a stretched or compressed version of the original but possibly a totally different sequence/waveform.

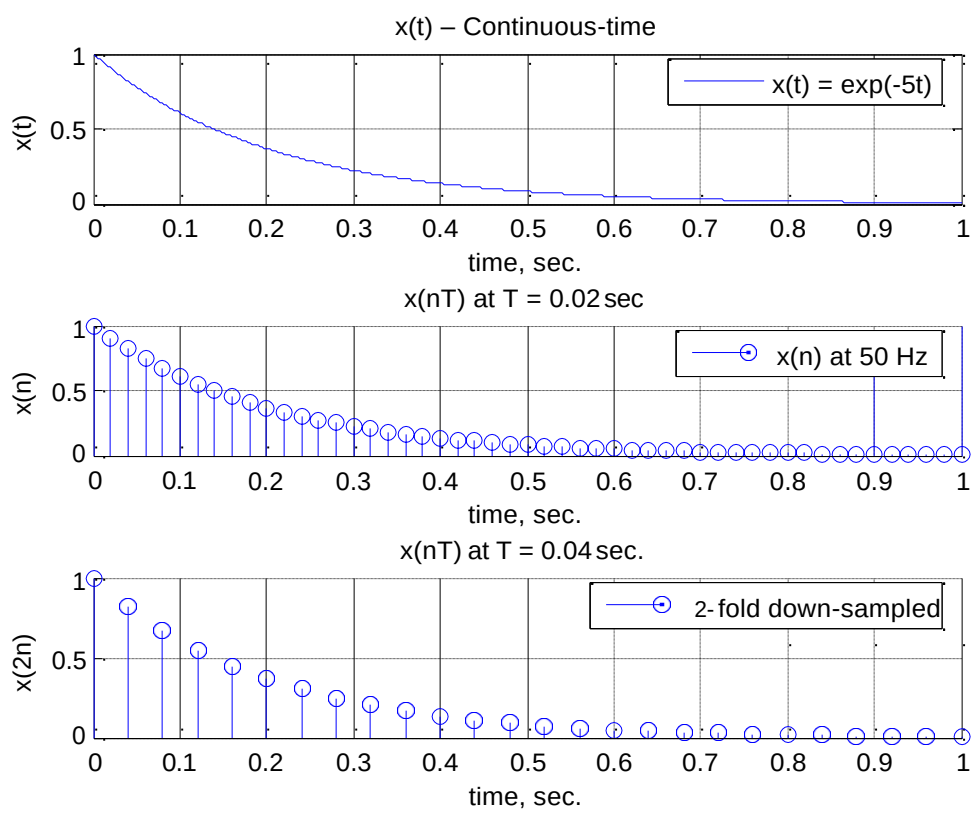
Example 5.2.1 Given that $x(t) = e^{-5t}u(t)$ is sampled at 50 Hz, find an expression for *x(n)*. Plot *x(t)*, *x(n)* and *x(2n)*. Sketch the spectrum of *x(n)*.

Solution The sampling time is $T = 0.02$ sec. Replacing t with nT we get $x(nT) = e^{-5nT}u(nT)$, or

$$x(n) = \left(e^{-0.1}\right)^n u(n) \;=\; (0.905)^n u(n) \,.$$

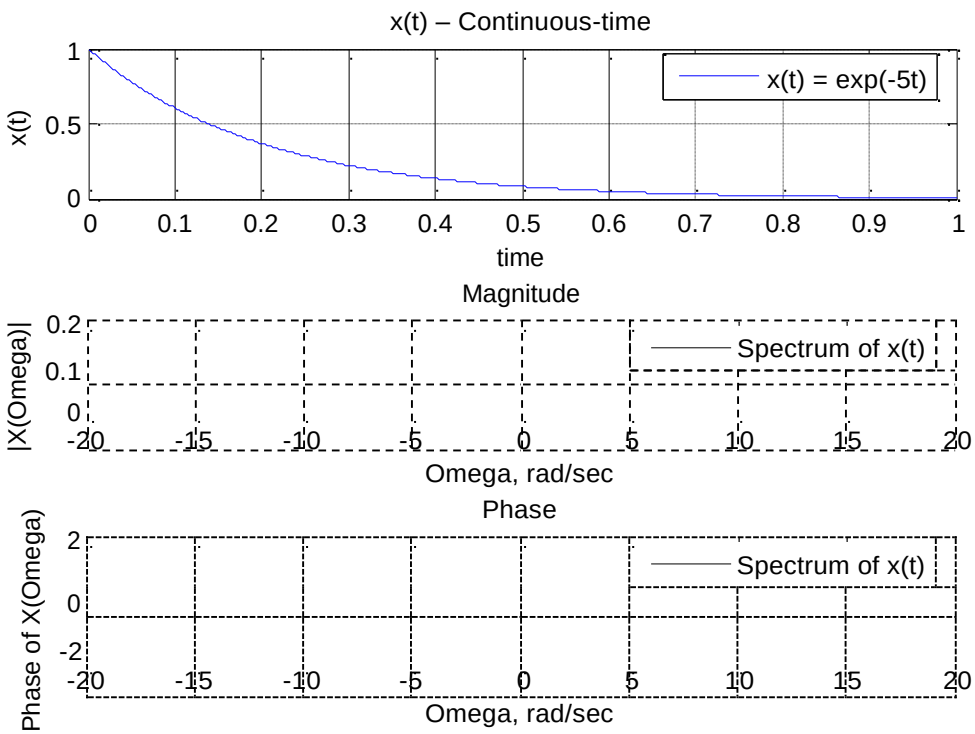
We show below three plots: (1) The continuous-time signal *x(t)*, (2) The sampled (at 50 Hz) version *x(n)*, and (3) *x(2n)*, the 2-fold down-sampled version of *x(n)*; this is equivalent to sampling *x(t)* at 25 Hz.

```
t = 0 : 1/512: 1; xt = exp (-5*t); %x(t) evaluated at 512 points
subplot(3, 1, 1), plot(t, xt); legend ('x(t) = exp(-5t)');
xlabel ('time, sec.'), ylabel('x(t)'); grid; title ('x(t) – Continuous-time')
%
t1 = 0 : 0.02: 1; xn = exp (-5*t1); %Sampled at 50 Hz.
subplot(3, 1, 2), stem(t1, xn); legend ('x(n) at 50 Hz');
xlabel ('time, sec.'), ylabel('x(n)'); grid; title ('x(nT) at T = 0.02 sec')
%
t2 = 0 : 0.04: 1; xt2 = exp (-5*t2); %Sampled at 25 Hz
subplot(3, 1, 3), stem(t2, xt2); legend ('2-fold down-sampled');
xlabel ('time, sec.'), ylabel('x(2n)'); grid; title ('x(nT) at T = 0.04 sec.')
```



Note that $X(s) = \mathcal{L}\left(e^{-5t}u(t)\right) = 1/(s+5)$. Shown below is the MATLAB plot of the magnitude spectrum $|X(j\Omega)|$ of the continuous-time signal $x(t)$ using the function ***plot***. Omega is a vector, consequently we use "./" instead of "/" etc. The main point to be made here is that $X(j\Omega)$ extends asymptotically to ∞, so, strictly speaking, $x(t)$ is not band-limited. Consequently, the spectrum $X(\omega)$ of the sampled signal $x(n)$ (shown later below) has some built-in aliasing.

```
t = 0 : 1/512: 1; xt = exp (-5*t); %x(t) evaluated at 512 points
subplot(3, 1, 1), plot(t, xt); legend ('x(t) = exp(-5t)');
xlabel ('time'), ylabel('x(t)'); grid; title ('x(t) – Continuous-time')
%
Omega = -6*pi: pi/256: 6*pi; X = 1./(5.+ j .*Omega);
subplot(3, 1, 2), plot(Omega, abs(X), 'k'); legend ('Spectrum of x(t)');
xlabel ('Omega, rad/sec'), ylabel('|X(Omega)|'); grid; title ('Magnitude')%
subplot(3, 1, 3), plot(Omega, angle(X), 'k'); legend ('Spectrum of x(t)');
xlabel ('Omega, rad/sec'), ylabel('Phase of X(Omega)'); grid; title ('Phase')
```



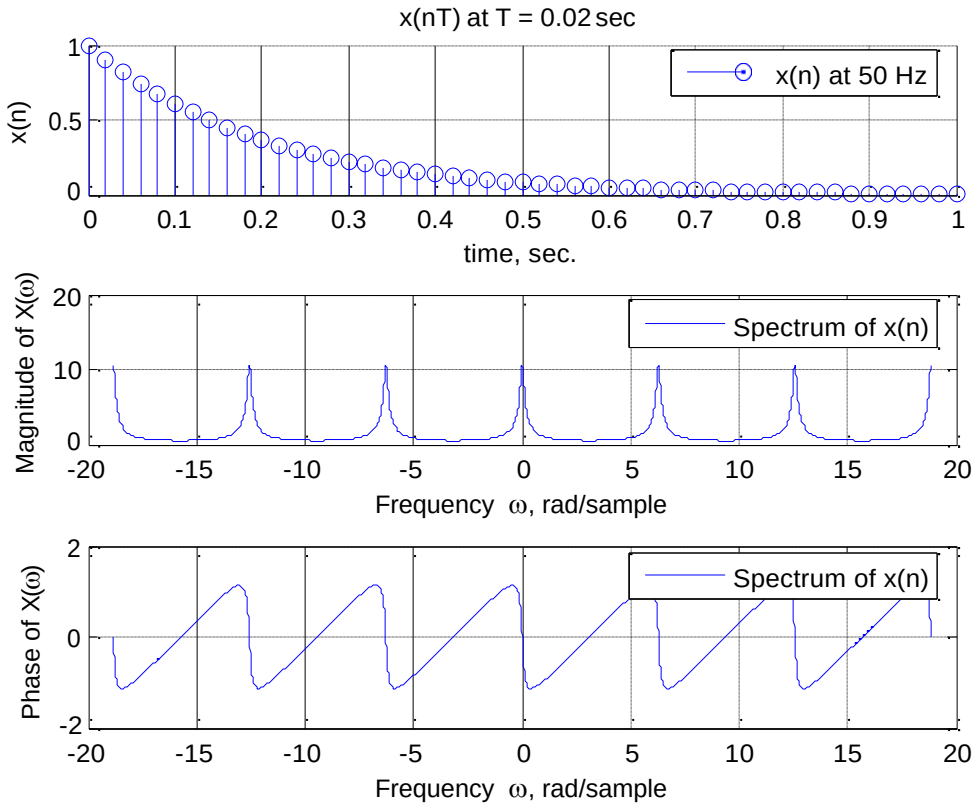
Coming to the discrete-time signal, the spectrum of $x(n) = a^n u(n) = (0.905)^n u(n)$ is its DTFT

$$X(e^{j\omega}) = \sum_{n=0}^{\infty} a^n e^{-j\omega n} = \frac{1}{1 - ae^{-j\omega}} = \frac{1}{1 - 0.905e^{-j\omega}}$$

The MATLAB segment is

```
t1 = 0 : 0.02: 1; xn = exp (-5*t1); %Sampled at 50 Hz.
subplot(3, 1, 1), stem(t1, xn); legend ('x(n) at 50 Hz');
xlabel ('time, sec.'), ylabel('x(n)'); grid; title ('x(nT) at T = 0.02 sec')
%
```

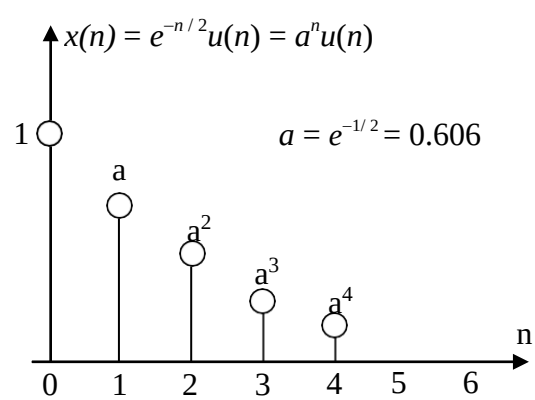
```
b = [1]; %Numerator coefficient
a = [1, -0.905]; %Denominator coefficients
w = -6*pi: pi/256: 6*pi; [Xw] = freqz(b, a, w);
subplot(3, 1, 2), plot(w, abs(Xw)); legend ('Spectrum of x(n)');
xlabel('Frequency \omega, rad/sample'), ylabel('Magnitude of X(\omega)'); grid
subplot(3, 1, 3), plot(w, angle(Xw)); legend ('Spectrum of x(n)');
xlabel('Frequency \omega, rad/sample'), ylabel('Phase of X(\omega)'); grid
```



Example 5.2.2 Given $x(n) = e^{-n/2}u(n)$, find (a) $x(5n/3)$, (b) $x(2n)$, (c) $x(n/2)$.
Answer The sequence

$$x(n) = e^{-n/2}u(n) = \left(e^{-1/2}\right)^n u(n) = (0.606)^n u(n) = a^n u(n)$$

where $a = e^{-1/2} = 0.606$, is sketched below:



(a) With $y(n) = x(5n/3)$, we evaluate $y(n)$ for several values of n (we have assumed here that $x(n)$ is zero if n is not an integer):

$$y(0) = x(5 \,.\, 0/3) = x(0) = e^{-0/2} = 1$$
$$y(1) = x(5 \,.\, 1/3) = x(5/3) = 0$$
$$y(2) = x(5 \,.\, 2/3) = x(10/3) = 0$$
$$y(3) = x(5 \,.\, 3/3) = x(5) = e^{-5/2} = a^5$$
$$\ldots$$
$$y(6) = x(5 \,.\, 6/3) = x(10) = e^{-10/2} = a^{10}$$
$$\ldots$$

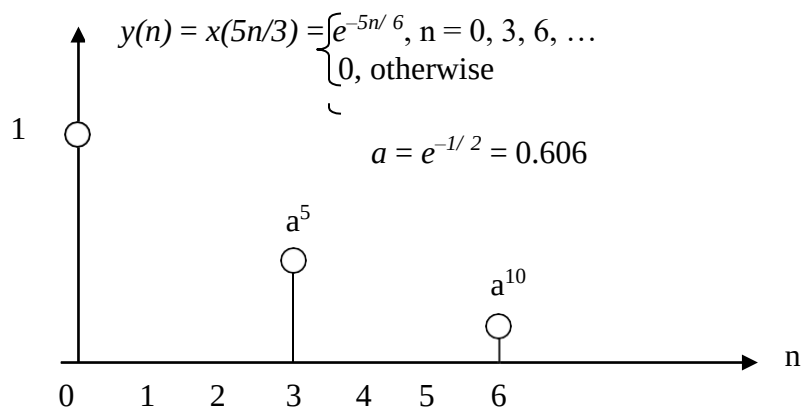
The general expression for $y(n)$ can be written as

$$y(n) = x(5n/3) = e^{-(5n/3)/2}, \qquad n \text{ as specified below}$$

$$= \begin{cases} e^{-5n/6}, & n = 0, 3, 6, \ldots \\ 0, & \text{otherwise} \end{cases}$$

$n =$	0	1	2	3	4	5	6	7	8	9	10
$y(n) =$	1	0	0	a^5	0	0	a^{10}	0	0	a^{15}	0

The sequence is sketched below:



(b) With $y(n) = x(2n)$, we evaluate $y(.)$ for several values of n:

$$y(0) = x(2 \,.\, 0) = x(0) = 1$$
$$y(1) = x(2 \,.\, 1) = x(2) = a^2$$
$$y(2) = x(2 \,.\, 2) = x(4) = a^4$$
$$y(3) = x(2 \,.\, 3) = x(6) = a^6$$
$$\ldots$$

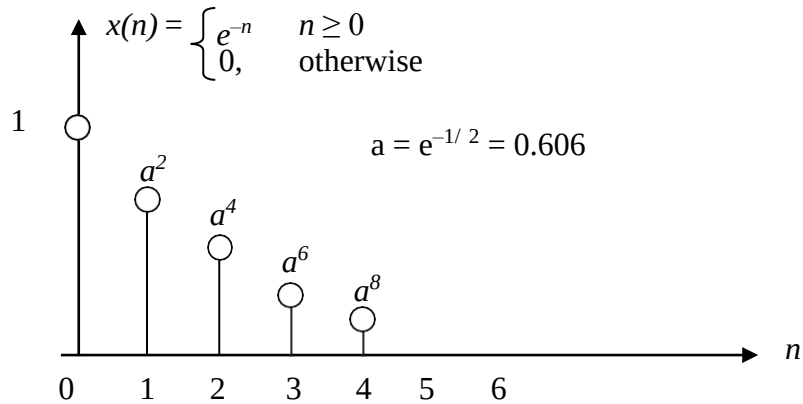
The general expression for $y(n)$ can be written as

$$y(n) = x(2n) = e^{-(2n)/2}, \qquad n \text{ as specified below}$$

$$= \begin{cases} e^{-n}, & n \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

$n =$	0	1	2	3	4	5
$y(n) =$	1	a^2	a^4	a^6	a^8	a^{10}

The sequence *y(.)* is made up of every other sample of *x(.)*. This is **down-sampling** or **decimation** by a factor of 2 (or, compression in time). Note that some of the original sample values have disappeared. The sequence is sketched below.



(c) With $y(n) = x(n/2) = e^{-n/4}u(n)$, we evaluate *y(.)* for several values of *n* (again, we have assumed here that *x(n)* is zero if *n* is not an integer):

y(0) = x(0/2) = x(0) = 1
y(1) = x(1/2) = x(0.5) = 0
y(2) = x(2/2) = x(1) = a
y(3) = x(3/2) = x(1.5) = 0
…

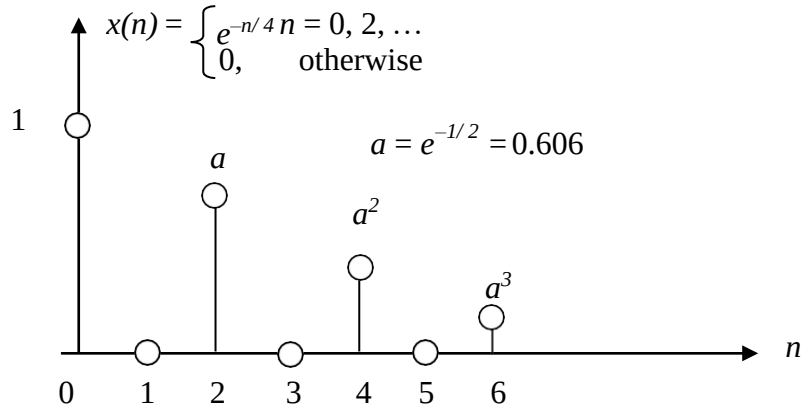
The general expression for *y(n)* can be written as

$$y(n) = x(n/2) = e^{-(n/2)/2}, \qquad n \text{ as specified below}$$

$$= \begin{cases} e^{-n/4}, & n = 0, 2, 4, \ldots \\ 0, & \text{otherwise} \end{cases}$$

$n =$	0	1	2	3	4	5	6
$y(n) =$	1	0	a	0	a^2	0	a^3

The sequence *y(.)* is constructed by inserting one zero between successive samples of *x(.)*. This is **up-sampling** or **interpolation** by a factor of 2 (or expansion in time). The sequence is sketched below:



To get back to the problem raised earlier, given the sequence *x(n)* obtained from *x(t)* at a rate (1/*T*)

$$x(t) \prec x(nT) \prec x(n), \text{ rate } (1/T)$$

we want to obtain the sequence $x'(n)$ which corresponds to a sampling rate $(1/T')$ where $T \neq T'$

$$x(t) \prec x(nT') \prec x'(n), \text{ rate } (1/T')$$

There are two approaches to do this:

1. Convert *x(n)* to *x(t)* and resample at $(1/T')$ to generate $x'(n)$. This is not ideal because of the imperfections in the *A/D-H(z)-D/A* originally involved in generating *x(n)*. Or,
2. Change the sampling rate entirely with discrete-time operations.

Example 5.2.3 Consider the 4 Hz signal $x(t) = \cos 2\pi 4t$ which is obviously band-limited to $F_{max} = 4$ Hz. It is sufficient to sample it at 8 Hz. Suppose that it has been *over-sampled* by, say, a factor of 6 at $F_s = 48$ Hz to give $x(n) = \cos 2\pi 4n(1/48) = \cos(\pi n/6)$.

If it is desired subsequently to generate from *x(n)* another signal $x_1(n)$ that is a discrete-time version of *x(t)* sampled at $F_{s1} = 16$ Hz (sampling rate reduced or down-sampled by a factor of 3), then can we do this by dropping two samples of *x(n)* for every sample that we keep? In this specific example this is possible since a sampling rate of 16 Hz is clearly greater than $2F_{max}$ of 8 Hz. Thus the down-sampled version is obtained by replacing *n* in *x(n)* by 3*n*

$$x_1(n) = x(3n) = \cos(\pi 3n/6) = \cos(\pi n/2) \qquad \prec (1)$$

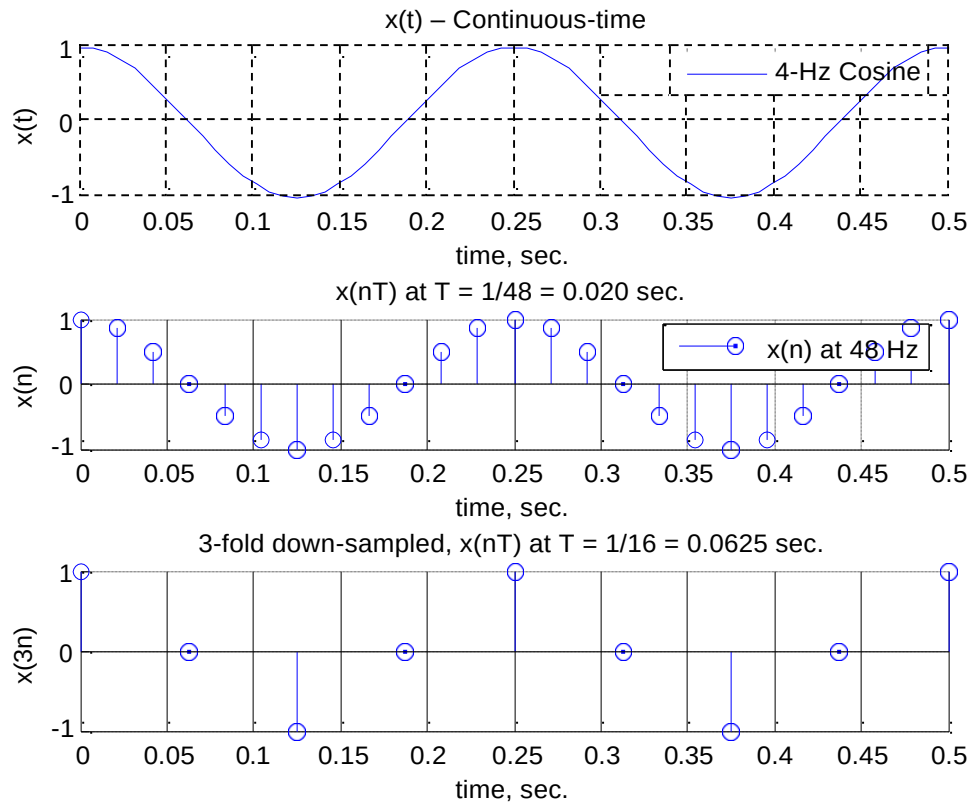
Let us compare this with what we would get if we were to sample $x(t) = \cos 2\pi 4t$ directly at 16 Hz. We simply replace *t* by $nT = n(1/16)$

$$x_1(n) = \cos 2\pi 4n(1/16) = \cos(\pi n/2) \qquad \prec (2)$$

The results in (1) and (2) are the same. (QED)

We show below three plots: (1) The continuous-time signal *x(t)*, (2) The sampled (at 48 Hz) version *x(n)*, and (3) *x(3n)*, the 3-fold down-sampled version of *x(n)*; this is equivalent to sampling *x(t)* at 16 Hz.

```
t = 0 : 1/128: 0.5; xt = cos (2*pi*4*t); %x(t) evaluated at 128 points
subplot(3, 1, 1), plot(t, xt); legend ('4-Hz Cosine');
xlabel ('time, sec.'), ylabel('x(t)'); grid; title ('x(t) – Continuous-time')
%
t1 = 0 : 1/48: 0.5; xn = cos (2*pi*4*t1); %Sampled at 48 Hz
subplot(3, 1, 2), stem(t1, xn); legend ('x(n) at 48 Hz');
xlabel ('time, sec.'), ylabel('x(n)'); grid; title ('x(nT) at T = 1/48 = 0.020 sec.')
%
t3 = 0 : 1/16: 0.5; x1n = cos (2*pi*4*t3); %Sampled at 16 Hz
subplot(3, 1, 3), stem(t3, x1n); xlabel ('time, sec.'), ylabel('x(3n)');
grid; title ('3-fold down-sampled, x(nT) at T = 1/16 = 0.0625 sec.')
```



Alternatively, assuming $x(t)$ is not available, $x_1(n)$ could be obtained as follows:

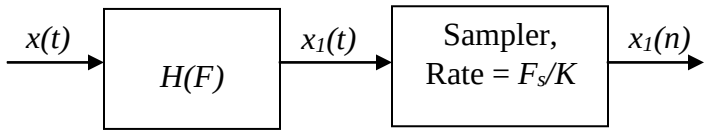
1. Recover $x(t)$ by passing $x(n)$ through a DAC
2. Sample the resulting $x(t)$ at F_{s1} = 16 Hz

We take it, however, that this option is not desirable.

The above analysis assumes that we know the frequency content of the base band signal, $x(t)$. Generally this is not the case. Given the sequence $x(n)$ that was obtained by sampling at a rate, say F_s, we do not know what is the maximum frequency, F_{max}, contained in the underlying analog signal, $x(t)$. Assuming it was originally band-limited and properly sampled, it is safest to assume that the base band signal was band-limited to $F_s/2$ (= F_{max}) and not lower. In such a case simply dropping one or more samples of $x(n)$ for every sample we keep will not work. If we want to reduce the sampling rate by a factor of, say, K, then we would have to band-limit the precursor of $x(n)$ to $(F_s/2)/K = (F_s/2K)$ and then sample it at the K-fold reduced sampling rate to achieve the required decimation. This amounts to down sampling $x(n)$ by a factor of K. (If the signal $x(t)$ originally actually contained a maximum frequency of $F_s/2$ then subsequent down sampling will result in unavoidable loss of information. But if it was band limited to significantly less than $F_s/2$ then down-sampling without loss of information is possible.)

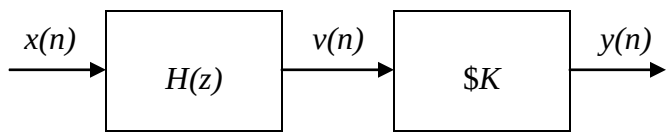
The band-limiting mentioned above may be done either in the continuous-time domain or in the discrete-time domain. The procedure in the continuous time domain is as follows: Imagine that $x(t)$ is recovered from $x(n)$; $x(t)$ is then band-limited to $F_s/2K$ by passing it through an ideal low pass filter described by

$$H(F) = \begin{cases} 1, & 0 \le F < F_s/2K \\ 0, & F_s/2K \le F \le F_s/2 \end{cases}$$



The band-limited signal, denoted $x_1(t)$ is then sampled at the reduced rate of F_s/K to generate $x_1(n)$. This method is generally undesirable because of all the imperfections inherent in originally generating $x(n)$ from $x(t)$ at a sampling rate of F_s, converting $x(n)$ back into $x(t)$, then band-limiting $x(t)$ to $F_s/2K$ to generate $x_1(t)$ and then sampling $x_1(t)$ at a sampling rate of F_s/K to generate $x_1(n)$.

Sampling rate decimation Reducing the sampling rate by an integer factor in the discrete-time domain is shown in the following block diagram. The down arrow in $\$K$ indicates **down sampling** by a factor of K. The filter $H(z)$ is a *digital anti-aliasing filter* whose output $v(n)$ is a low pass filtered version of $x(n)$.



If the filter $H(z)$ is implemented as a linear phase FIR filter with $(M+1)$ coefficients specified as $\{b_r, r = 0 \text{ to } M\}$, (some call it "$M^{th}$ order"), then

$$v(n) = \sum_{r=0}^{M} b_r x(n-r)$$

We desire the output $y(n)$ to be a down-sampled version of $x(n)$, that is

$$y(n) = v(Kn) = \sum_{r=0}^{M} b_r x(Kn-r)$$

Example 5.2.4 Consider the 4 Hz signal $x(t) = \cos 2\pi 4t$ which is obviously band-limited to $F_{max} = 4$ Hz. It is sufficient to sample it at 8 Hz. Suppose instead that it has been over sampled, say, by a factor of 6 at $F_s = 48$ Hz to give $x(n) = \cos 2\pi 4n(1/48) = \cos(\pi n/6)$.

Can we generate from $x(n)$ another signal $x_3(n)$ that is a discrete-time version of $x(t)$ sampled at $F_{s3} = F_s/8 = 6$ Hz? This is down sampling by a factor of 8. We simply replace t by $nT = n(1/6)$ to get

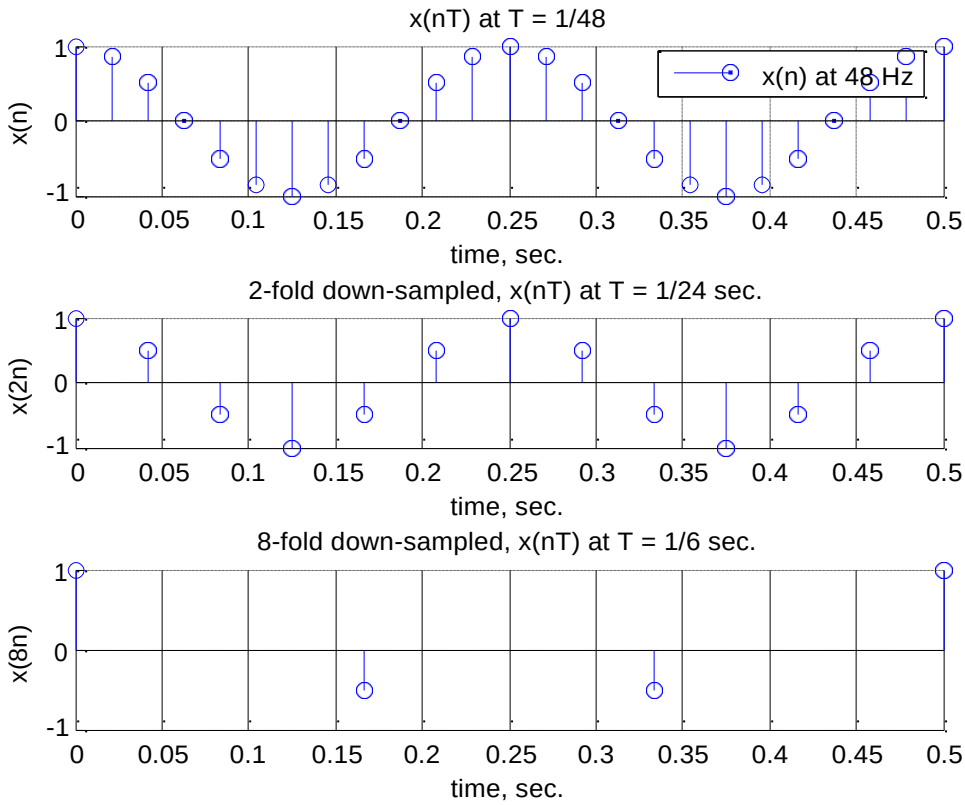
$$x_3(n) = \cos 2\pi 4n(1/6) = \cos(8\pi n/6) = x(8n) = \{x(0), x(8), x(16), \ldots\}$$

In other words, $x_3(n)$ is made up of every 8th sample of $x(n)$. For every sample value of $x(n)$ we keep we discard the next 7 samples. We know, however, that a sampling frequency of 6 Hz does not satisfy the sampling theorem; in this case down sampling has been taken too far.

We show below three plots: (1) The sampled (at 48 Hz) version $x(n)$ – this is repeated from above, (2) $x(2n)$, the 2-fold down-sampled version of $x(n)$; this is equivalent to sampling $x(t)$ at 24 Hz, and (3) $x(8n)$, the 8-fold down-sampled version of $x(n)$; this is equivalent to sampling $x(t)$ at the unacceptably low rate of 6 Hz.

```
t1 = 0 : 1/48: 0.5; xn = cos (2*pi*4*t1); %Sampled at 48 Hz
subplot(3, 1, 1), stem(t1, xn); legend ('x(n) at 48 Hz');
xlabel ('time, sec.'), ylabel('x(n)'); grid; title ('x(nT) at T = 1/48')
%
t2 = 0 : 1/24: 0.5; xt2 = cos (2*pi*4*t2); %Sampled at 24 Hz
```

```
subplot(3, 1, 2), stem(t2, xt2); xlabel ('time, sec.'), ylabel('x(2n)');
grid; title ('2-fold down-sampled, x(nT) at T = 1/24 sec.')
%
t4 = 0 : 1/6: 0.5; x3n = cos (2*pi*4*t4); %Sampled at 6 Hz
subplot(3, 1, 3), stem(t4, x3n); xlabel ('time, sec.'), ylabel('x(8n)');
grid; title ('8-fold down-sampled, x(nT) at T = 1/6 sec.')
```



Example 7.2.5 To show visually a case of down sampling that is not satisfactory, consider $x_4(n)$ generated from $x(n)$ by down sampling by a factor of 12, i.e., $x_4(n) = x(12n)$. This is also obtained by sampling at 48/12 = 4 Hz:

$$x_4(n) = x(nT) = \cos 2\pi 4n(1/4) = \cos (2\pi n) = \cos (12\pi n/6) = x(12n)$$

In this case $\cos (2\pi n) = 1$ for all n, so that

$$x_4(n) = 1 \text{ for all } n$$

which has no resemblance to $x(n)$, making it visually obvious that down sampling has been taken too far. Depending on at what point in the cycle the samples are taken, $x_4(n)$ equals a constant (including 0), for all n.

Down-sampling

Assume that $x(n)$ is obtained from an underlying continuous-time signal $x(t)$ by sampling at F_x Hz. Assume that $x(t)$ was originally band limited to $F_x/2$ Hz. On the digital frequency (ω) scale this amounts to $x(n)$ being band limited to π.

We now wish to generate a signal $y(n)$ by down-sampling $x(n)$ by a factor of M, that is, we are reducing the sampling rate by a factor of M. This amounts to:

1. Converting $x(n)$ to $x(t)$ using a D/A converter.
2. Band limiting $x(t)$ to $F_x/2M$ Hz. Assume that no information is lost due to this band limiting.
3. Resampling $x(t)$ at F_x/M Hz. to produce $y(n)$.

Equivalently the above task is accomplished entirely in the digital domain by

1. Band limiting $x(n)$ to π/M. Assume that no information is lost due to this step.
2. Down-sampling the above $x(n)$ by a factor of M to produce $y(n)$.

We may view $y(n)$ as though it were generated by sampling an underlying analog signal $y(t)$ at a rate $F_y = F_x/M$ Hz.

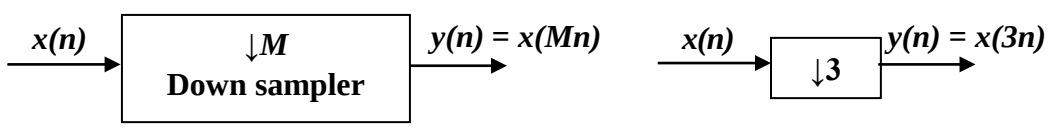
Given the signal $x(n)$ that was obtained at a certain sampling rate the new signal $y(n)$, the down-sampled version of $x(n)$, with a sampling rate that is $(1/M)$ of that of $x(n)$, obtained from $x(n)$, is given by:

$$y(n) = x(Mn)$$

and is made up of every M^{th} sample value of $x(n)$; the intervening $(M–1)$ sample values of $x(n)$ are dropped. This amounts to

$$y(0) = x(0),\ y(1) = x(M),\ y(2) = x(2M),\ y(3) = x(3M),\ \ldots$$

The time between samples of $y(.)$ is M times that between samples of $x(.)$, or the sampling frequency of $y(.)$ is reduced by a factor of M from that of $x(.)$. The block diagram of a down sampler is shown below.



Example 5.3.1 As an example, if $x(n) = a^n u(n)$, $a < 1$, is the sequence:

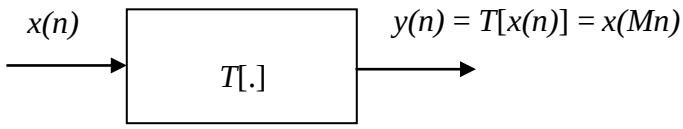
$$\begin{array}{rcccccccccccc} n = & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & & & \\ x(n) = & \{1 & a & a^2 & a^3 & a^4 & a^5 & a^6 & a^7 & a^8 & . & . & .\} \end{array}$$

then $y(n) = x(2n)$, with $M = 2$, is its 2-fold down-sampled version and is obtained by keeping every other sample of $x(n)$ and dropping the samples in between:

$$\begin{array}{rcccccccccc} n = & 0 & 1 & 2 & 3 & 4 & 5 & & & \\ y(n) = & \{1 & a^2 & a^4 & a^6 & a^8 & a^{10} & . & . & .\} \end{array}$$

In this example it is understood that the time between samples of $y(n)$ is twice that between samples of $x(n)$, or, the sampling rate of $y(n)$ is one-half of that of $x(n)$.

Example 7.3.2 Test the system $y(n) = x(Mn)$, where M is a constant, for **time-invariance**.



Solution See also Unit I. For the input $x(n)$ the output is

$$y(n) = T[x(n)] = x(Mn)$$

Delay this output by n_0 to get

$$y(n-n_0) = x(M(n-n_0)) = x(Mn-Mn_0) \quad \text{(A)}$$

Next, for the delayed input $x(n-n_0)$ the output is

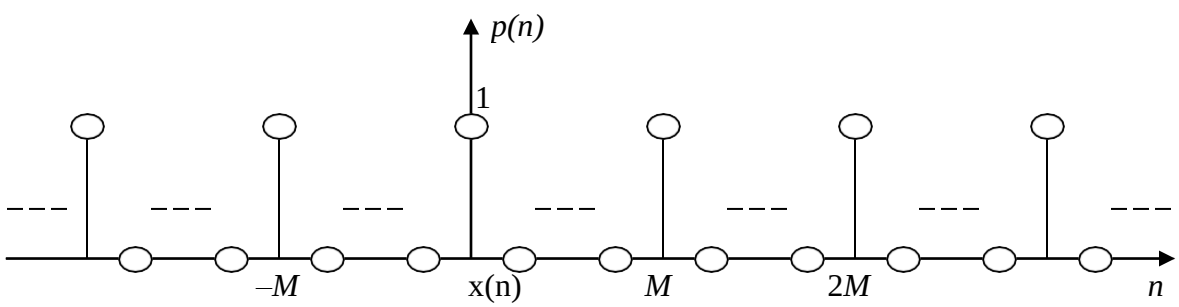
$$y(n, n_0) = T[x(n-n_0)] = x(Mn) = x(Mn-n_0) \quad \text{(B)}$$

→e see that (A) ≠ (B), that is, $y(n-n_0)$ and $y(n, n_0)$ are not equal. Delaying the input is not equivalent to delaying the output. So the system is *not* time-invariant. In other words the down-sampling operation is a time-varying system.

Spectrum of a down-sampled signal Given the signal $x(n)$ whose spectrum is $X(\omega)$ or $X(e^{j\omega})$ we want to find the spectrum of $y(n)$, the down-sampled version of $x(n)$, denoted by $y(n)$ — $Y(\omega)$.

Consider the periodic train of impulses, $p(n)$, with period M

$$p(n) = \begin{cases} 1, & n = 0, \pm M, \pm 2M, \ldots \\ 0, & \text{otherwise} \end{cases}$$



The discrete Fourier series representation (see Example 1 in Unit II) of $p(n)$ is

$$p(n) = \sum_{k=0}^{M-1} P_k \, e^{j2\pi kn/M}, \qquad 0 \le n \le M-1$$

The Fourier coefficients are given by

$$P_k = \frac{1}{M} \sum_{n=0}^{M-1} p(n)\, e^{-j2\pi kn/M} = \frac{1}{M}, \qquad 0 \le k \le M-1$$

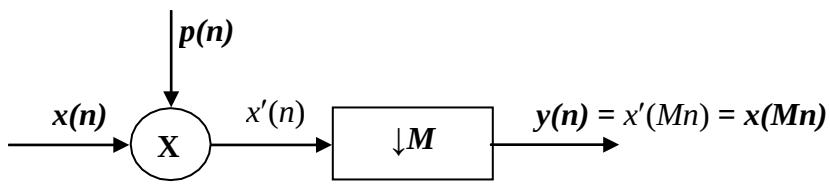
Thus the DFS for $p(n)$ is

$$p(n) = \frac{1}{M}\sum_{k=0}^{M-1} e^{j2\pi kn/M}, \qquad 0 \le n \le M-1$$

Define the signal $x'(n)$

$$x'(n) = x(n)p(n) = \begin{cases} x(n), & n = 0, \pm M, \pm 2M, \ldots \\ 0, & \text{otherwise} \end{cases}$$

The sequence $x'(n)$ consists of values of $x(n)$ whenever $n = 0, \pm M, \pm 2M, \ldots$, and *zeros in between* those points.



Define the down-sampled version $y(n)$

$$y(n) = x'(Mn) = x(Mn)\,p(Mn) = x(Mn)$$

The signal $y(n)$ consists of values of $x(Mn)$ at $n = 0, \pm 1, \pm 2, \ldots$, but *no zeros in between*.

With $y(n) = x'(Mn) = x(Mn)$ our objective is to find the spectrum $Y(\omega)$. Keep in mind that $X(\omega)$ periodic in ω since $x(n)$ is a discrete-time sequence; and the same is true of $Y(\omega)$. Now the z-transform of $y(n)$ is

$$Y(z) = \sum_{n=-\infty}^{\infty} y(n)z^{-n} = \sum_{n=-\infty}^{\infty} x'(Mn)z^{-n}$$

Set $Mn = k$: then $n = k/M$ and the summation limits $n = \{-\infty \text{ to } \infty\}$ become $k = \{-\infty \text{ to } \infty\}$. Thus

$$Y(z) = \sum_{k=-\infty}^{\infty} x'(k)z^{-k/M} = \sum_{n=-\infty}^{\infty} x'(n)z^{-n/M}$$

Here $x'(n) = 0$ except when n is a multiple of M. Substituting $x(n)\,p(n)$ for $x'(n)$ in the above equation,

$$Y(z) = \sum_{n=-\infty}^{\infty} x(n)\,p(n)z^{-n/M}$$

Substituting $\frac{1}{M}\sum_{k=0}^{M-1} e^{j2\pi kn/M}$ for $p(n)$ (from the DFS) in the above equation,

$$Y(z) = \sum_{n=-\infty}^{\infty} x(n)\left[\frac{1}{M}\sum_{k=0}^{M-1} e^{j2\pi kn/M}\right]z^{-n/M} = \frac{1}{M}\sum_{k=0}^{M-1}\sum_{n=-\infty}^{\infty} x(n)e^{j2\pi kn/M}z^{-n/M}$$

$$= \frac{1}{M}\sum_{k=0}^{M-1}\underbrace{\sum_{n=-\infty}^{\infty} x(n)\left(e^{-j2\pi k/M}z^{1/M}\right)^{-n}}_{= X\left(e^{-j2\pi k/M}z^{1/M}\right)}$$

$$= \frac{1}{M}\sum_{k=0}^{M-1} X\left(e^{-j2\pi k/M} z^{1/M}\right)$$

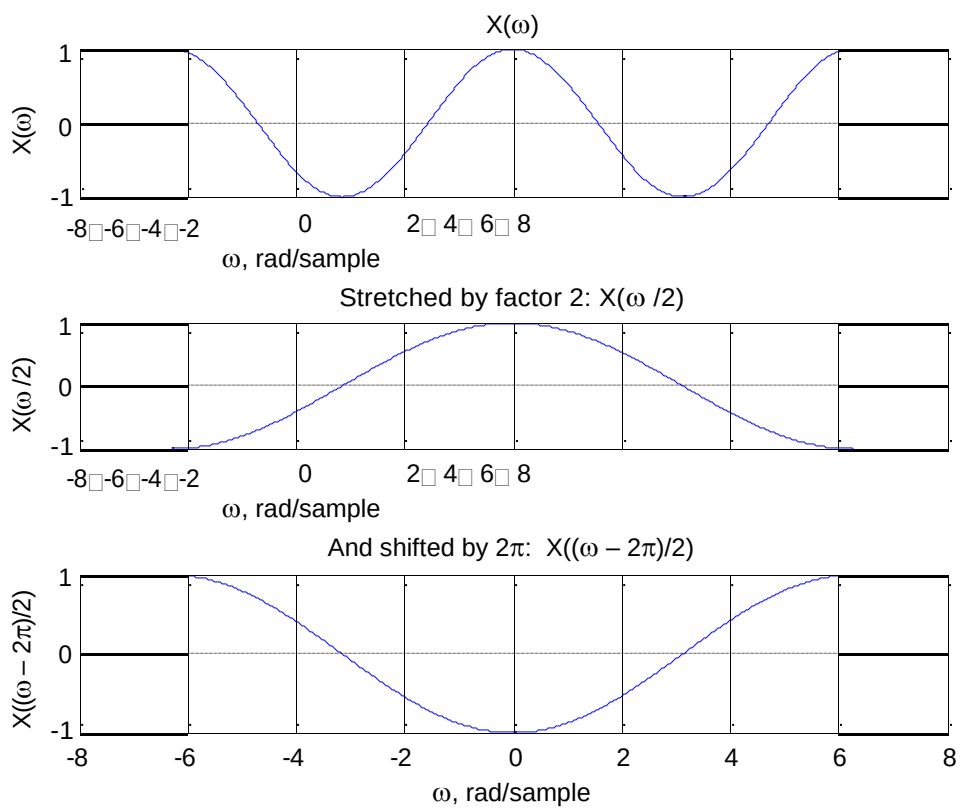
Substituting $z = e^{j\omega}$ gives us the DTFT, $Y(\omega)$ or $Y(e^{j\omega})$,

$$Y(\omega) = Y(z)\Big|_{z=e^{j\omega}} = \frac{1}{M}\sum_{k=0}^{M-1} X\left(e^{-j2\pi k/M} e^{j\omega/M}\right) = \frac{1}{M}\sum_{k=0}^{M-1} X\left(e^{j(\omega-2\pi k)/M}\right)$$

$$= \frac{1}{M}\sum_{k=0}^{M-1} X\left(\frac{\omega-2\pi k}{M}\right)$$

MATLAB. To demonstrate the stretching and shifting of $X(\omega)$ to $X((\omega-2\pi k)/M)$ for $k = 1$ and $M = 2$, that is, $X((\omega-2\pi)/2)$. This is done in 3 steps: (1) $X(\omega)$, (2) $X(\omega/2)$, and (3) $X((\omega-2\pi)/2)$

```
w = -2*pi: pi/256: 2*pi;
subplot(3, 1, 1), plot(w, cos(w));
xlabel ('\omega, rad/sample'), ylabel('X(\omega)'); grid; title ('X(\omega)')
%
subplot(3, 1, 2), plot(w, cos(w/2));
xlabel ('\omega, rad/sample'), ylabel('X(\omega /2)'); grid;
title ('Stretched by factor 2: X(\omega /2)')
%
subplot(3, 1, 3), plot(w, cos((w-2*pi)/2));
xlabel ('\omega, rad/sample'), ylabel('X((\omega – 2\pi)/2)'); grid;
title ('And shifted by 2\pi: X((\omega – 2\pi)/2)')
```



where, for simplicity, we have used the notation $X(\omega)$ instead of $X(e^{j\omega})$. This expression for $Y(e^{j\omega})$ is a sum of M terms. Note that the function $X(\omega-2\pi k)$ is a shifted (by $2\pi k$) version of $X(\omega)$ and $X(\omega/M)$ is a stretched (by a factor M) version of $X(\omega)$. Thus $Y(e^{j\omega})$ is the sum of M uniformly shifted and stretched versions of $X(e^{j\omega})$ each scaled by the factor $(1/M)$. The shifting

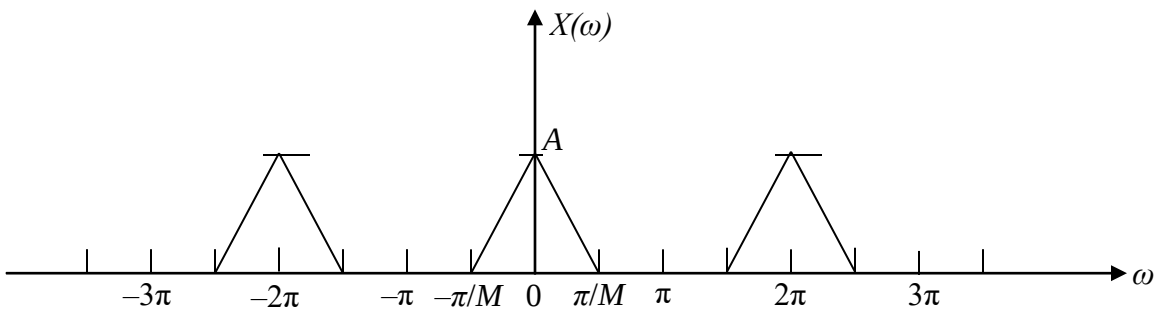
in multiples of 2π corresponds to the factor $(\omega-2\pi k)$ in the argument of $X(.)$, and the stretching by the factor M corresponds to the M in $(\omega-2\pi k)/M$. Note that the amount of shift is also affected by the factor M, that is, the amount of shift doesn't stay at $2\pi k$ but ends up being $2\pi k/M$.

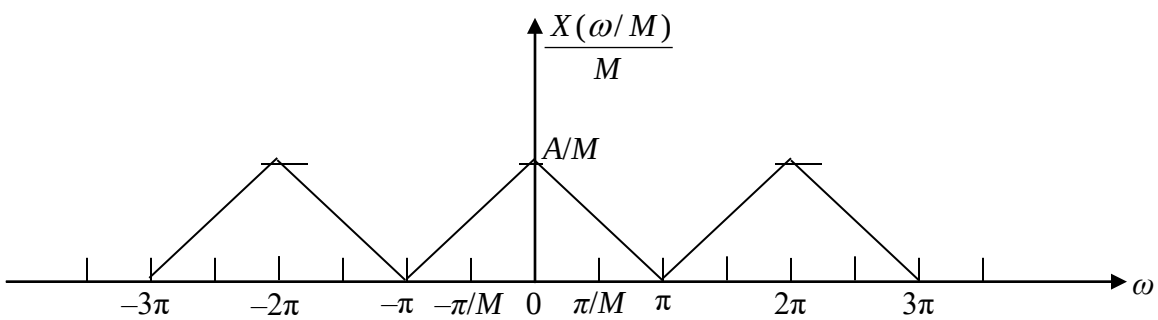
The expression for $Y(e^{j\omega})$ contains a total of M versions of $X(e^{j\omega})$, one original and $(M-1)$ shifted replicas. Each of these is also stretched by a factor of M, so $X(e^{j\omega})$ should have been preshrunk, that is, band limited, to π/M before undertaking the down-sampling. Writing out the expression for $Y(e^{j\omega})$ in full, we have

$$Y(e^{j\omega}) = \frac{1}{M}\sum_{k=0}^{M-1} X\left(\frac{\omega-2\pi k}{M}\right)$$

$$= \frac{1}{M}\left[X\left(\frac{\omega}{M}\right) + X\left(\frac{\omega-2\pi 1}{M}\right) + \ldots + X\left(\frac{\omega-2\pi(M-1)}{M}\right)\right]$$

The first term that makes up $Y(e^{j\omega})$, that is, $\frac{1}{M}X\left(\frac{\omega}{M}\right)$, is shown in the figure below. The figure implicitly uses $M = 2$. In general there will be $(M-1)$ shifted replicas of this term.





In particular, for $M = 2$, we have

$$Y(e^{j\omega}) = \frac{1}{2}\sum_{k=0}^{2-1} X\left(\frac{\omega-2\pi k}{2}\right) = \frac{1}{2}\left[X\left(\frac{\omega}{2}\right) + X\left(\frac{\omega-2\pi}{2}\right)\right]$$

$$= \frac{1}{2}\left[X\left(\frac{\omega}{2}\right) + X\left(\frac{\omega}{2}-\pi\right)\right]$$

This is also written in the form

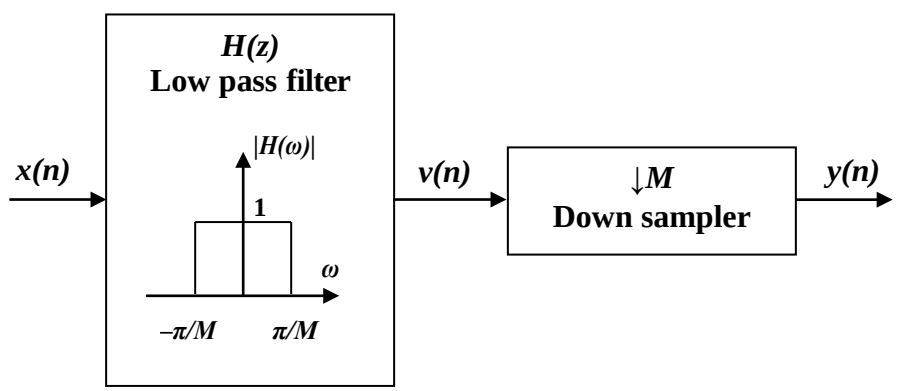
$$Y(e^{j\omega}) = \frac{1}{2}\sum_{k=0}^{2-1} X\left(e^{j(\omega-2\pi k)/2}\right) = \frac{1}{2}\left[X\left(e^{j\omega/2}\right) + X\left(e^{j(\omega-2\pi)/2}\right)\right]$$

$$= \frac{1}{2}\left[X\left(e^{j\omega/2}\right) + X\left(e^{j\omega/2 - j\pi}\right)\right] = \frac{1}{2}\left[X\left(e^{j\omega/2}\right) + X\left(-e^{j\omega/2}\right)\right]$$

To recapitulate, before we decided to down sample $X(\omega)$ was originally band limited to π on the digital frequency scale (that is, F_x /2 Hz). We then band limited it to π/M (that is, F_x /2M Hz) and down sampled by a factor of M.

Aliasing Down-sampling by a factor of M, in itself, is simply retaining every M^{th} sample while dropping all samples in between. If, therefore, prior to down-sampling, the signal $x(n)$ is indeed band-limited to π/M then we generate the down-sampled version $y(n)$ by simply taking every M^{th} sample of $x(n)$. This process is shown below in block diagram fashion. If in this set-up $x(n)$ is not band-limited as required then the spectrum of $y(n)$ will contain overlapping spectral components of $x(n)$ due to stretching, i.e., $X\left(\omega / M\right)$ will overlap $X\left(\overline{\omega - 2}\pi / M\right)$, etc. This results in **aliasing**.

Band-limiting $x(n)$ to π/M (if not done already) is done by an **anti-aliasing filter** (digital low pass filter) with a cut-off frequency of π/M. The general process of *decimation then consists of filtering followed by down sampling* shown in block diagram below.



Unlike an analog anti-aliasing filter associated with an ADC, the filter in this diagram is a *digital* anti-aliasing filter specified as

$$H(\omega) = \begin{cases} 1, & 0 \le |\omega| < \pi/M \\ 0, & \pi/M \le |\omega| \le \pi \end{cases}$$

Note that π corresponds to $F_x/2$ and π/M corresponds to $F_x/2M$ where F_x is the sampling frequency of $x(n)$.

Typically, in order to avoid (delay) distortion, the filter $H(z)$ is a linear phase FIR filter with (N+1) coefficients $\{h(r), r = 0 \text{ to } N\}$. The output, $v(n)$, of the low pass filter is then given by convolution

$$v(n) = \sum_{r=0}^{N} h(r)\, x(n - r)$$

and the decimated signal is

$$y(n) = v(nM) = \sum_{r=0}^{N} h(r)\, x(nM - r)$$

In summary, in order to down sample a signal by a factor of M:

- The signal should have been originally over-sampled by a factor of M (that is originally band limited to π/M and over-sampled). In this case the signal is down-sampled straightaway; no pre-filter is needed. OR
- The signal, assumed originally band limited to π, should be band-limited to π/M by a pre-filter; the signal is then down-sampled. In this case there will be some loss of information.

Example 5.3.3 Consider the signal $x(n) = a^n u(n)$, $a < 1$.

a) Determine the spectrum $X(\omega)$

b) If $x(n)$ is applied to a decimator that reduces the sampling rate by a factor of 2 determine the output spectrum

c) Show that the spectrum in part (b) is simply the Fourier transform of $x(2n)$

d) Plot the spectra of $x(n)$ and $x(2n)$ for $a = 0.905$

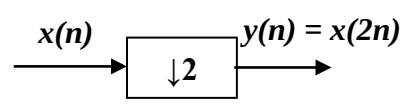
Solution [See also Unit I]

a) The spectrum of $x(n)$ is given by its DTFT

$$X(\omega) = \sum_{n=-\infty}^{\infty} x(n)\, e^{-j\omega n} = \sum_{n=0}^{\infty} a^n\, e^{-j\omega n} = \sum_{n=0}^{\infty} \left(a\, e^{-j\omega}\right)^n$$

$$= \frac{1}{1 - ae^{-j\omega}}, \qquad \left|ae^{-j\omega}\right| < 1$$

This spectrum is not band-limited but we may pretend it is. This may also be obtained as $X(\omega) = X(z)\big|_{z = e^{j\omega}}$.



b) The spectrum of $y(n) = x(2n)$ is given by

$$Y(\omega) = \frac{1}{M} \sum_{k=0}^{M-1} X\left(\frac{\omega - 2\pi k}{M}\right)$$

which, with $M = 2$, becomes

$$Y(\omega) = \frac{1}{2} \sum_{k=0}^{1} X\left(\frac{\omega - 2\pi k}{2}\right) = \frac{1}{2}\left[X\left(\frac{\omega}{2}\right) + X\left(\frac{\omega}{2} - \pi\right)\right]$$

$$= \frac{1}{2}\left[\frac{1}{1 - ae^{-j\omega/2}} + \frac{1}{1 - ae^{-j((\omega/2) - \pi)}}\right] = \frac{1}{2}\left[\frac{1}{1 - ae^{-j\omega/2}} + \frac{1}{1 - ae^{-j\omega/2} e^{j\pi}}\right]$$

$$= \frac{1}{2}\left[\frac{1}{1 - ae^{-j\omega/2}} + \frac{1}{1 + ae^{-j\omega/2}}\right] = \frac{1}{1 - a^2 e^{-j\omega}}$$

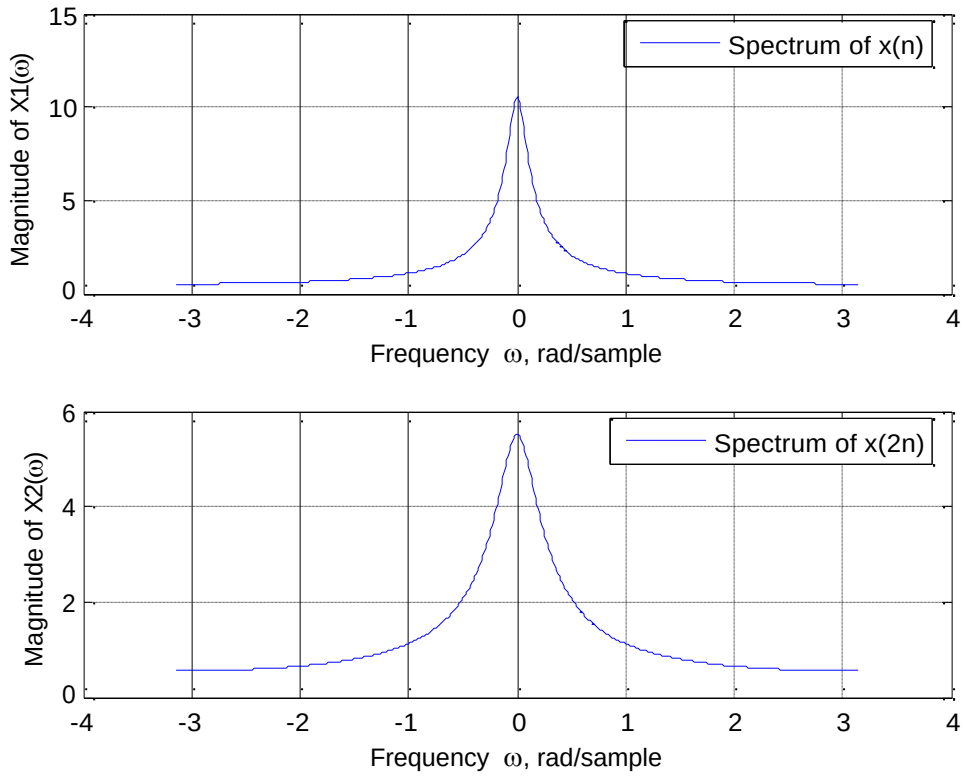
c) The Fourier transform of $y(n) = x(2n) = a^{2n} u(2n) = a^{2n} u(n)$ is

$$Y(\omega) = \sum_{n=0}^{\infty} a^{2n}\, e^{-j\omega n} = \sum_{n=0}^{\infty} \left(a^2\, e^{-j\omega}\right)^n = \frac{1}{1 - a^2 e^{-j\omega}}, \qquad \left|a^2 e^{-j\omega}\right| < 1$$

d) The spectra.

```
b = [1]; %Numerator coefficient
a1 = [1, -0.905]; a2 = [1, -0.819]; %Denominator coefficients
w = -pi: pi/256: pi; %A total of 512 points
[X1w] = freqz(b, a1, w); [X2w] = freqz(b, a2, w)
subplot(2, 1, 1), plot(w, abs(X1w)); legend ('Spectrum of x(n)');
```

```
xlabel('Frequency \omega, rad/sample'), ylabel('Magnitude of X1(\omega)'); grid
subplot(2, 1, 2), plot(w, abs(X2w)); legend ('Spectrum of x(2n)');
xlabel('Frequency \omega, rad/sample'), ylabel('Magnitude of X2(\omega)'); grid
```



Up-sampling

Assume that $x(n)$ is obtained from the continuous-time signal $x(t)$ by sampling at F_x Hz. We now wish to generate a signal $y(n)$ by up-sampling $x(n)$ by a factor of L, that is, we are increasing the sampling rate by a factor of L. This amounts to

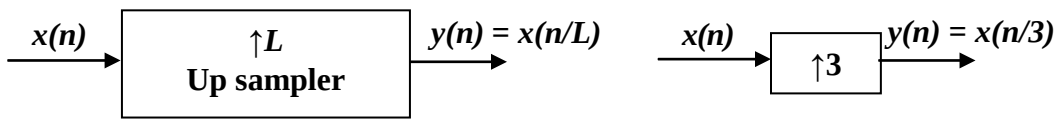
1. Converting $x(n)$ to $x(t)$ using a D/A converter.
2. Resampling $x(t)$ at LF_x Hz to produce $y(n)$.

We may view $y(n)$ as though it were generated by sampling an underlying analog signal $y(t)$ (or $x(t)$ for that matter) at a rate $F_y = LF_x$ Hz. As in the case of down-sampling we prefer to do this entirely in the digital domain.

Given the signal $x(n)$ that was obtained at a certain sampling rate we can obtain a new signal $y(n)$ from $x(n)$ with a sampling rate that is L times that of $x(n)$. The signal $y(n)$, an up-sampled version of $x(n)$, is given by:

$$y(n) = \begin{cases} x(n/L), & n = 0, \pm L, \pm 2L, \ldots \\ 0, & \text{otherwise} \end{cases}$$

and is constructed by placing $(L-1)$ zeros between every pair of consecutive samples of $x(n)$. The time between samples of $y(n)$ is $(1/L)$ of that between samples of $x(n)$, or the sampling frequency of $y(n)$ is increased by a factor of L from that of $x(n)$. The block diagram of an up-sampler is shown below.



Example 5.4.1 As an example, if $x(n) = a^n u(n)$, $a < 1$, is the sequence:

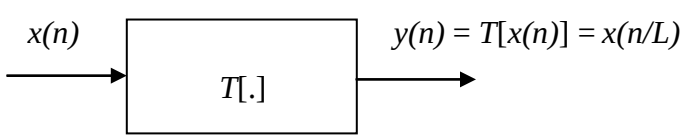
$$\begin{array}{rcccccccccccc} n = & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & & & \\ x(n) = & \{1 & a & a^2 & a^3 & a^4 & a^5 & a^6 & a^7 & a^8 & . & . & .\} \end{array}$$

then $y(n) = x(n/2)$, with $L = 2$, is its 2-fold up-sampled version and is obtained by inserting a 0 between each pair of consecutive values in $x(n)$

$$\begin{array}{rcccccccccccc} n = & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & \\ y(n) = & \{1 & 0 & a & 0 & a^2 & 0 & a^3 & 0 & a^4 & 0 & a^5 & . \quad . \quad .\} \end{array}$$

In this example it is understood that the time between samples of $y(n)$ is one half of that between samples of $x(n)$, or, the sampling rate of $y(n)$ is twice that of $x(n)$.

Example 5.4.2 Test the system $y(n) = x(n/L)$, where L is a constant, for **time-invariance**.



Solution See also Unit I. The system $y(n) = x(n/L)$ in itself only partially defines an up-sampler. But the following goes to show that the up-sampling operation is a time-varying system. For the input $x(n)$ the output is

$$y(n) = T[x(n)] = x(n/L)$$

Delay this output by n_0 to get

$$y(n–n_0) = x((n–n_0)/L) = x((n/L)–(n_0/L)) \quad \text{(A)}$$

Next, for the delayed input $x(n–n_0)$ the output is

$$y(n, n_0) = T[x(n–n_0)] = x(n/L) = x((n/L)–n_0) \quad \text{(B)}$$

→e see that (A) ≠ (B), that is, $y(n–n_0)$ and $y(n, n_0)$ are not equal. Delaying the input is not equivalent to delaying the output. So the system $y(n) = x(n/L)$ is *not* time-invariant. Therefore the up sampler defined by

$$y(n) = \begin{cases} x(n/L), & n = 0, \pm L, \pm 2L, \ldots \\ 0, & \text{otherwise} \end{cases}$$

is *not* time-invariant; it is a time-varying system.

Spectrum of an up-sampled signal Given the signal $x(n)$ whose spectrum is $X(\omega)$ or $X(e^{j\omega})$ we want to find the spectrum of $y(n)$, the up-sampled version of $x(n)$, denoted by $y(n)$ — $Y(\omega)$.

The signal $y(n)$, with a sampling rate that is L times that of $x(n)$, is given by:

$$y(n) = \begin{cases} x(n/L), & n = 0, \pm L, \pm 2L, \ldots \\ 0, & \text{otherwise} \end{cases}$$

We obtain the z-transform and from it the spectrum:

$$Y(z) = \sum_{n=-\infty}^{\infty} y(n) z^{-n} = \sum_{n = 0, \pm L, \pm 2L, \ldots}^{\infty} x(n/L)\, z^{-n} + \sum_{\substack{n = \text{otherthan } \pm kL \\ k = \text{all integers}}}^{\infty} 0\, z^{-n}$$

$$= \sum_{n = 0, \pm L, \pm 2L, \ldots}^{\infty} x(n/L)\, z^{-n}$$

Set $n/L = k$: this leads to $n = kL$, and the summation indices $n = \{0, \pm L, \pm 2L, \pm 3L, \ldots\}$ become $k = \{-\infty \text{ to } \infty\}$, so that

$$Y(z) = \sum_{k=-\infty}^{\infty} x(k)\, z^{-kL} = \sum_{k=-\infty}^{\infty} x(k) \left(z^L\right)^{-k} = X\left(z^L\right)$$

Setting $z = e^{j\omega}$ gives us the spectrum

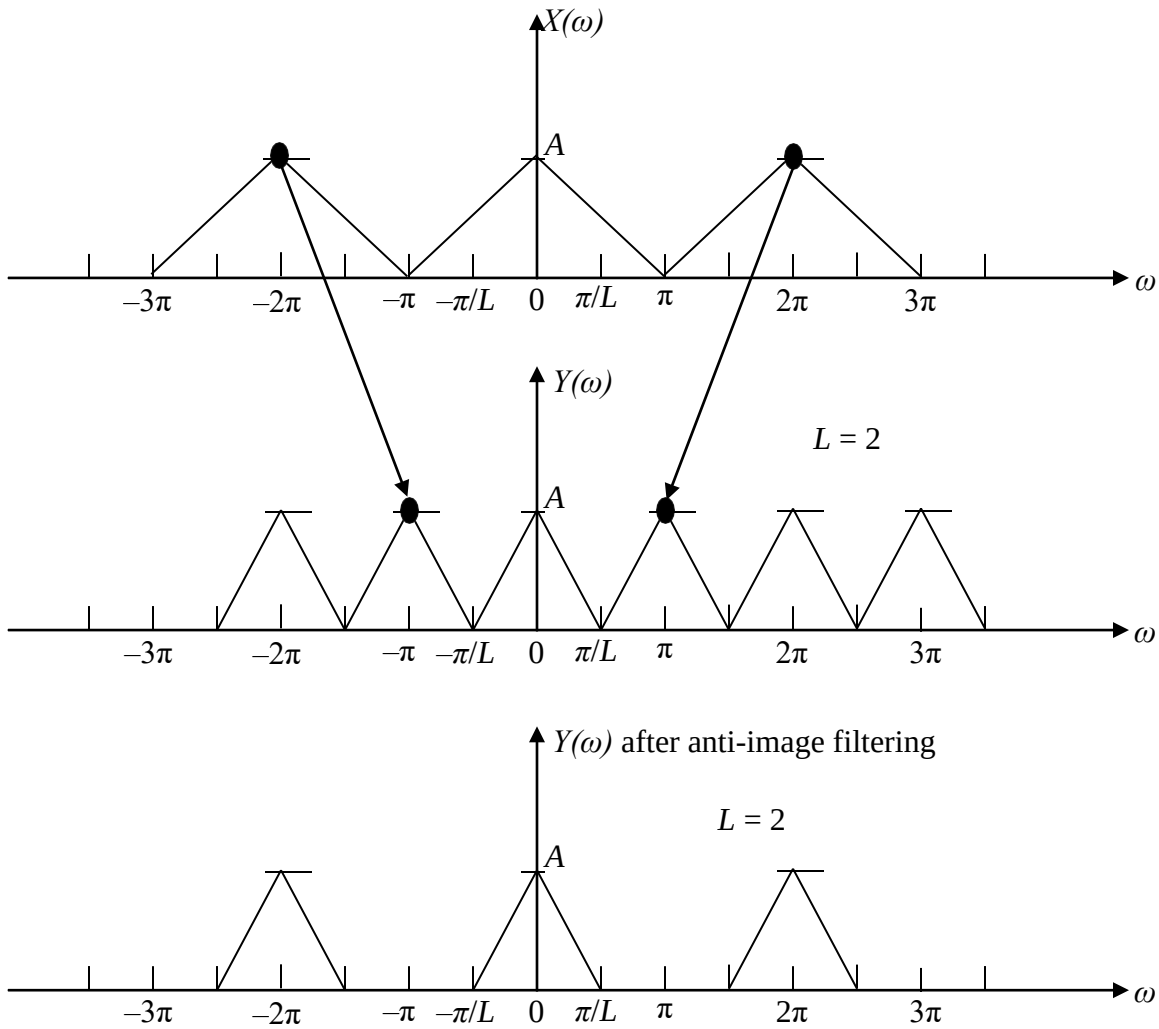
$$Y(e^{j\omega}) = Y(z)\big|_{z=e^{j\omega}} = X\left(e^{j\omega L}\right) \qquad \text{or} \qquad Y(\omega) = X(\omega L)$$

Thus $Y(\omega)$ is an L-fold compressed version of $X(\omega)$; the value of $X(.)$ that occurred at ωL occurs at ω, (that is, at $\omega L/L$) in the case of $Y(.)$. In going from X to Y the frequency values are pushed in toward the origin by the factor L. For example, the frequency ωL is pushed to $\omega L/L$, the frequency π is pushed to π/L, 2π is pushed to $2\pi/L$, etc.

Shown below are the spectra $X(\omega)$ and $Y(\omega)$ for 2-fold up-sampling, that is, $L = 2$. Note that $X(\omega)$ is periodic to start with so that the frequency content of interest is in the base range ($-\pi \le \omega \le \pi$) with replicas of this displaced by multiples of 2π from the origin on either side. Due to

up-sampling the frequency content of $X(\omega)$ in the range ($-\pi \le \omega \le \pi$) is compressed into the range ($-\pi/L \le \omega \le \pi/L$) of $Y(\omega)$, that is, into ($-\pi/2 \le \omega \le \pi/2$), centered at $\omega = 0$. The first replica of $X(\omega)$ in the range ($\pi \le \omega \le 3\pi$), centered at 2π, is compressed to the range ($\pi/2 \le \omega \le 3\pi/2$) of $Y(\omega)$, centered at π; its counterpart, in ($-3\pi \le \omega \le -\pi$), centered at -2π, is compressed to ($-3\pi/2 \le \omega \le -\pi/2$), centered at $-\pi$. If, for the purpose of discussion, we consider the range $(0, 2\pi)$ as one fundamental period then the replica in the range $(\pi/2, 3\pi/2)$ of Y is an **image (spectrum)** and needs to be filtered out with a low pass filter (**anti-imaging filter**) of band-width $\pi/2$. With $L = 2$ this is the only image in $(0, 2\pi)$.

Furthermore, while the spectrum $X(\omega)$ is periodic with a period = 2π, the spectrum $Y(\omega)$, on account of the image, is a 2-fold periodic repetition of the base spectrum in ($-\pi/2 \le \omega \le \pi/2$); the image spectrum is actually spurious/unwanted; further the periodicity of $Y(\omega)$ is still 2π.

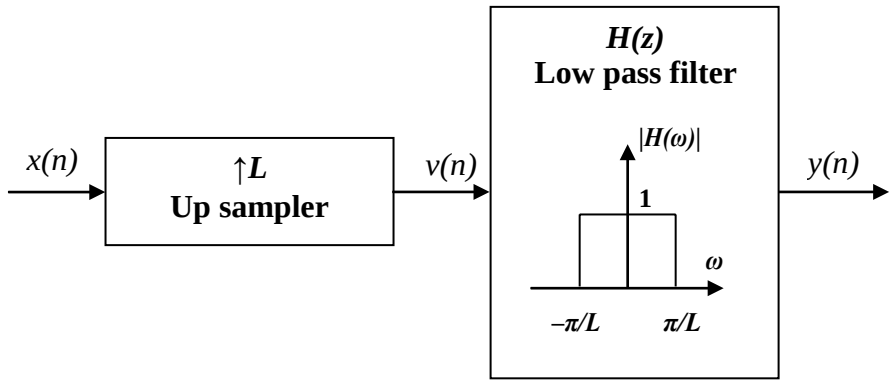


These observations can be extended to larger values of L. For $L = 3$, for instance, there will be two image spectra (a 3-fold periodic repetition of the base spectrum in ($-\pi/3 \le \omega \le \pi/3$), and the anti-imaging filter band width will be $\pi/3$.

In general, up-sampling of $x(n)$ by a factor of L involves

- Inserting L–1 zeros between successive pairs of sample values of $x(n)$.
- The spectrum $Y(\omega)$ of the up-sampled signal is an L-fold compressed version of $X(\omega)$. As a result $Y(\omega)$ contains L–1 images and is an L-fold periodic repetition of the base spectrum in ($-\pi/L \le \omega \le \pi/L$).
- The anti-imaging filter band width is π/L.

The over-all scheme of up-sampling is shown in block diagram below. Unlike an analog anti-imaging filter associated with a DAC, the filter in this diagram is a *digital anti-imaging filter*.



In this diagram the pass band gain of the anti-imaging filter is shown as 1. This gain is actually chosen equal to L to compensate for the fact that the average value of $y(n)$ is $1/L$ times the average value of $x(n)$ due to the presence of the inserted zeros.

$$H(\omega) = \begin{cases} L, & 0 \le |\omega| < \pi/L \\ 0, & \pi/L \le |\omega| \le \pi \end{cases}$$

Note that π corresponds to $F_x/2$ and π/L corresponds to $F_x/2L$ where F_x is the sampling frequency of $x(n)$.

The output of the low pass filter is given by the convolution sum

$$y(n) = \sum_{r=-\infty}^{\infty} h(n-r)\, v(r)$$

where its input is

$$v(r) = \begin{cases} x(r/L), & r = 0, \pm L, \pm 2L, \ldots \\ 0, & \text{otherwise} \end{cases}$$

Now $v(r) = 0$ except at $r = kL$, where k is all integers from $-\infty$ to ∞. Thus we have

$$v(kL) = x(kL/L) = x(k)$$

The convolution sum may be written as

$$\sum_{r=-\infty}^{\infty} h(n-r)\, v(r) = \sum_{k=-\infty}^{\infty} h(n-kL)\, v(kL) = \sum_{k=-\infty}^{\infty} h(n-kL)\, x(k)$$

so that the interpolated signal is

$$y(n) = \sum_{k=-\infty}^{\infty} h(n-kL)\, x(k)$$

Illustration Given the signal $x(n) = \{1, a, a^2, a^3, a^4, a^5, a^6, a^7, a^8, a^9, a^{10}, \ldots\}$, its 2-fold up-sampled version is obtained by inserting a 0 between each pair of consecutive samples in $x(n)$:

$$y(n) = \{1, 0, a, 0, a^2, 0, a^3, 0, a^4, 0, a^5, 0, a^6, 0, a^7, 0, a^8, 0, a^9, 0, a^{10}, \ldots\}$$

Intuitively, even visually, *y(n)* contains higher (or, more) frequencies than *x(n)* because of the inserted zeros. For instance, consider the first two or three samples in each sequence. In the case of *x(n)* the changes from 1 to *a* to a^2 are smoother than the fluctuations in *y(n)* from 1 to 0 to *a* to 0 to a^2; these latter fluctuations are the higher frequencies not originally contained in *x(n)*. It is these higher frequencies that are represented by the image in the spectrum of *y(n)* prior to anti-imaging filtering. The anti-imaging filter removes or smoothes out the higher frequency fluctuations from the up-sampled version; *this smoothing is manifested in the form of the interpolated zeros being replaced by nonzero values*.

Cascading sampling rate converters

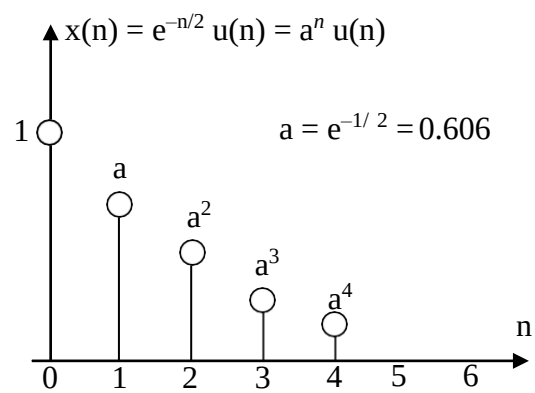
Given a discrete-time signal *x(n)* we may want to convert its sampling rate by a non integer factor, in particular, by a rational number. For instance, we may be interested in *x(3n/2)*. This involves a 2-fold up-sampling and a 3-fold down-sampling for a net down sampling by a factor of 1.5 (= 3/2). The sequence *x(3n/5)* involves a 5-fold up-sampling and a 3-fold down-sampling for a net up-sampling by a factor of 1.67 (= 5/3).

In general, in a cascade of an *M*-fold down-sampler and an *L*-fold up-sampler the positions of the two samplers are inter-changeable with no difference in the input-output behavior *if and only if M and L are co-prime* (*relatively prime*, that is, *M* and *L* do not have a common factor). The sequence *x(3n/2)* may be generated by cascading the up-sampler and the down-sampler in either order, that is, down followed by up or vice versa. However, a cascade of a 6-fold down-sampler (*M* = 6) followed by a 4-fold up-sampler (*L* = 4) is not the same as a cascade of a 4-fold up-sampler followed by a 6-fold down-sampler even though in both cases *M/L* = 6/4. This is because *M* and *L* have a common factor, that is, the rational number *M/L* is not in its reduced form. The ratio *M/L* should be reduced to 3/2; then the 3-fold down-sampler and the 2-fold up-sampler are interchangeable in position.

Example 5.5.1 Given $x(n) = e^{-n/2}\, u(n)$, find $x(5n/3)$.
Answer We borrow this from an earlier section. Our objective is to present the earlier solution and then reformulate it in the context of cascading up- and down-samplers. The sequence

$$x(n) = e^{-n/2}\, u(n) = (e^{-1/2})^n\, u(n) = (0.606)^n\, u(n) = a^n\, u(n)$$

where $a = e^{-1/2} = 0.606$, is sketched below:



With *y(n) = x(5n/3)*, we evaluate *y(.)* for several values of *n* (we have assumed here that *x(n)* is zero if *n* is not an integer):

$y(0) = x(5 \,.\, 0 / 3) = x(0) = e^{-0/2} = 1$
$y(1) = x(5 \,.\, 1 / 3) = x(5 / 3) = 0$
$y(2) = x(5 \,.\, 2 / 3) = x(10 / 3) = 0$
$y(3) = x(5 \,.\, 3 / 3) = x(5) = e^{-5/2} = a^5$
…
$y(6) = x(5 \,.\, 6 / 3) = x(10) = e^{-10/2} = a^{10}$
…

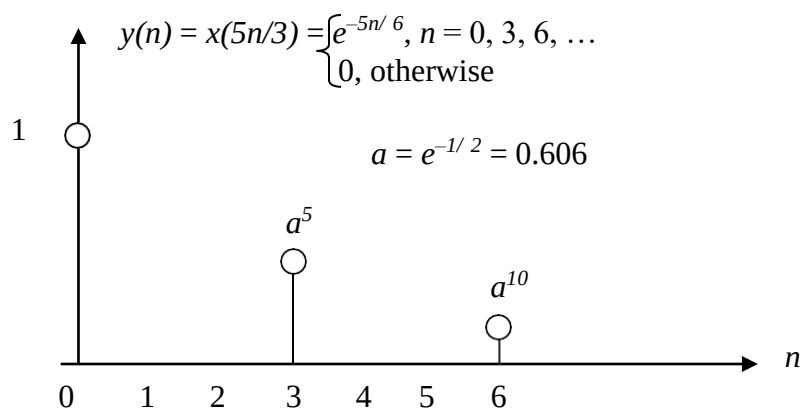
The general expression for $y(n)$ can be written as

$$y(n) = x(5n/3) = e^{-(5n/3)/2}, \qquad n \text{ as specified below}$$

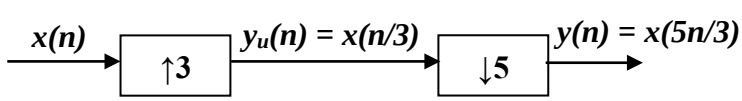
$$= \begin{cases} e^{-5n/6}, & n = 0, 3, 6, \ldots \\ 0, & \text{otherwise} \end{cases}$$

$n =$	0	1	2	3	4	5	6	7	8	9	10	11	12.	.	.
$y(n) =$	{1	0	0	a^5	0	0	a^{10}	0	0	a^{15}	0	0	a^{20}	.	.}

The sequence is sketched below:



We shall recast this problem in terms of cascading the up- and down-samplers. In the expression $y(n) = x(5n/3)$ there is a 3-fold up-sampling and a 5-fold down-sampling. Since the numerator 5 is greater than the denominator 3 there is a ***net down-sampling by a factor of 1.67*** (= 5/3). Let us first do a 3-fold up-sampling of $x(n)$ followed by a 5-fold down-sampling of the resulting sequence. That is, given the sequence $x(n)$



$n =$	0	1	2	3	4	5	6	7	8	9	10	.	.
$x(n) =$	{1	a	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	.	.}

we define $y_u(n) = x(n/3)$, and then $y(n) = y_u(5n) = x(5n/3)$. The sequences $y_u(n)$ and $y(n)$ are given below.

$$y_u(n) = x(n/3) = e^{-n/6}\, u(n/3) = \begin{cases} a^{n/3} & n = 0, 3, 6, \ldots \\ 0, & \text{otherwise} \end{cases}$$

$y_u(n) = x(n/3)$

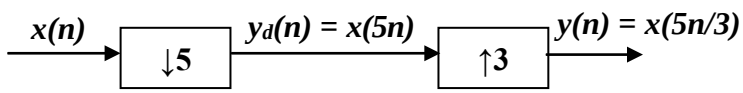
$n =$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	.	.
$y_u(n) =$	{1	0	0	a	0	0	a^2	0	0	a^3	0	0	a^4	0	0	a^5 ..... }				

$$y(n) = y_u(5n) = x(5n/3) = e^{-5n/6}\, u(5n/3) = \begin{cases} a^{5n/3} & n = 0, 3, 6, \ldots \\ 0, & \text{otherwise} \end{cases}$$

$y(n) = y_u(5n) = x(5n/3)$

$n =$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	.	.
$y(n) =$	{1	0	0	a^5	0	0	a^{10}	0	0	a^{15}	0	0	a^{20}	0	0	a^{25} ..... }				

Alternatively, we may first do a 5-fold down sampling followed by a 3-fold up-sampling:



$y_d(n) = x(5n) = \{1, a^5, a^{10}, a^{15}, a^{20}, \ldots\}$
$y(n) = y_d(n/3) = x(5n/3) = \{1, 0, 0, a^5, 0, 0, a^{10}, 0, 0, a^{15}, 0, 0, a^{20}, \ldots\}$

The net effect is that between the first two terms (1 and a^5) of the final output $y(.)$ we have dropped four original terms and inserted two zeros.

Example 5.5.2 Given $x(n) = e^{-n/2}\, u(n)$, find $x(3n/5)$. Here there is a 5-fold up-sampling and a 3-fold down sampling. Since the denominator is bigger there is a ***net up-sampling by a factor of 1.67***.

$$x(n) = \{1, a, a^2, a^3, a^4, a^5, a^6, a^7, a^8, a^9, a^{10}, \ldots\}$$

Method A Up-sampling followed by down sampling is given below. The 5-fold up-sampled signal, $y_u(n)$, is obtained by inserting 4 zeros shown in bold face between every pair of consecutive samples in $x(n)$

$$\begin{aligned} y_u(n) &= x(n/5) \\ &= \{1, \mathbf{0, 0, 0, 0,}\ a, \mathbf{0, 0, 0, 0,}\ a^2, \mathbf{0, 0, 0, 0,}\ a^3, \mathbf{0, 0, 0, 0,}\ a^4, \mathbf{0, 0, 0, 0,} \\ &\qquad a^5, \mathbf{0, 0, 0, 0,}\ a^6, \mathbf{0, 0, 0, 0,}\ a^7, \mathbf{0, 0, 0, 0,}\ a^8, \mathbf{0, 0, 0, 0,}\ a^9, \\ &\qquad \mathbf{0, 0, 0, 0,}\ a^{10}, \ldots\} \end{aligned}$$

The 3-fold down-sampled signal, $y_1(n)$, is obtained by keeping every third sample in $y_u(n)$ and discarding the rest (shown underlined)

$$y_u(n) = = \{1, \underline{0, 0,}\ 0, \underline{0, a,}\ 0, \underline{0, 0,}\ 0, \underline{a^2, 0,}\ 0, \underline{0, 0,}\ a^3, \underline{0, 0,}\ 0, \underline{0, a^4,}\ 0, \underline{0, 0,}\ 0,$$

$\underline{a^5, 0,}$ 0, $\underline{0, 0,}$ a^6, $\underline{0, 0,}$ 0, $\underline{0, a^7,}$ 0, $\underline{0, 0,}$ 0, $\underline{a^8, 0,}$ 0, $\underline{0, 0,}$ a^9, $\underline{0, 0,}$ 0, $\underline{0, a^{10},}$ …}

$$y_1(n) = y_u(3n) = x(3n/5)$$
$$= \{1, 0, 0, 0, 0, a^3, 0, 0, 0, 0, a^6, 0, 0, 0, 0, a^9, 0, \ldots\}$$

Method B Down-sampling followed by up sampling is given below. The 3-fold down-sampled signal, $y_d(n)$, is obtained by keeping every third sample in $x(n)$ and discarding the rest (shown umderlined)

$$x(n) = \{1, \underline{a, a^2,} a^3, \underline{a^4, a^5,} a^6, \underline{a^7, a^8,} a^9, \underline{a^{10}, a^{11},} \ldots\}$$

$$y_d(n) = x(3n) = \{1, a^3, a^6, a^9, a^{12}, \ldots\}$$

The 5-fold up-sampled signal, $y_2(n)$, is obtained by inserting 4 zeros shown in bold face between every pair of samples in $y_d(n)$

$$y_2(n) = y_d(n/5) = x(3n/5)$$

$$y_2(n) = \{1, \mathbf{0, 0, 0, 0,} a^3, \mathbf{0, 0, 0, 0,} a^6, \mathbf{0, 0, 0, 0,} a^9, \mathbf{0, 0, 0, 0,} a^{12}, \ldots\}$$
$$= \{1, 0, 0, 0, 0, a^3, 0, 0, 0, 0, a^6, 0, 0, 0, 0, a^9, 0, 0, 0, 0, a^{12}, \ldots\}$$

It is seen that $y_1(n) = y_2(n)$.

Example 5.5.3 Given that $x(n) = \{1, a, a^2, a^3, a^4, a^5, a^6, a^7, a^8, a^9, a^{10}, \ldots\}$ is the input,

1. Find the output $y_1(n)$ of a cascade of a 2-fold up-sampler followed by a 4-fold down sampler.
2. Find the output $y_2(n)$ of a cascade of a 4-fold down sampler followed by a 2-fold up-sampler.

Solution Note that the down-sampling factor $M = 4$ and the up-sampling factor $L = 2$ are not co-prime since they have a factor in common. The ratio $M/L = 4/2$, as given, is not in its reduced form. As a result we do not expect that $y_1(n)$ and $y_2(n)$ will be equal. Specifically, in the first case we have

$$x(n) = \{1, a, a^2, a^3, a^4, a^5, a^6, a^7, a^8, a^9, a^{10}, \ldots\}$$

Up-sample by inserting a zero (shown bold face) between consecutive samples of $x(n)$ resulting in $y_u(n)$

$$y_u(n) = x(n/2) = \{1, \mathbf{0}, a, \mathbf{0}, a^2, \mathbf{0}, a^3, \mathbf{0}, a^4, \mathbf{0}, a^5, \mathbf{0}, a^6, \mathbf{0}, a^7, \mathbf{0}, a^8, \mathbf{0}, a^9, \mathbf{0}, a^{10}, \ldots\}$$

Down-sample by keeping every fourth sample of $y_u(n)$ and discarding the three samples in between resulting in $y_1(n)$

$$y_1(n) = y_u(4n) = x(4n/2) = \{1, a^2, a^4, a^6, a^8, a^{10}, \ldots\}$$

In the second case

$$x(n) = \{1, a, a^2, a^3, a^4, a^5, a^6, a^7, a^8, a^9, a^{10}, \ldots\}$$

$$y_d(n) = x(4n) = \{1, a^4, a^8, a^{12}, \ldots\}$$

$$y_2(n) = y_d(n/2) = x(4n/2) = \{1, 0, a^4, 0, a^8, 0, a^{12}, \ldots\}$$

It is seen that $y_1(n) \neq y_2(n)$.

Sampling Rate Conversion by a Rational Factor *L*/*M* Here the sampling rate is being converted by a non-integral factor such as 0.6 or 1.5. That is, given $x(n)$ with a sampling rate of F_x we want to obtain $y(n)$ with a sampling rate of F_y of, say, $0.6F_x$ (decimation) or $1.5F_x$ (interpolation).

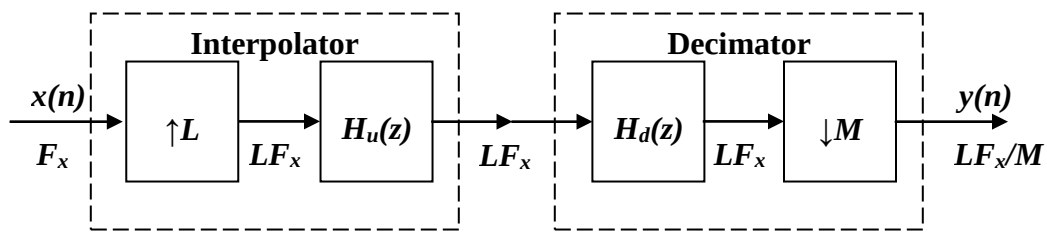
Take, for instance $L/M = 3/5$. Here the basic approach is to first interpolate (up-sample) by a factor of $L = 3$ and then decimate (down-sample) by a factor of $M = 5$. The net effect of the cascade of interpolation followed by decimation is to change the sampling rate by a rational factor L/M, that is,

$$F_y = \left(\frac{L}{M}\right) F_x = \left(\frac{3}{5}\right) F_x = 0.6\, F_x$$

The corresponding signal is given by $y(n) = x(5n/3)$, ignoring the filters involved. (This can also be done by first down-sampling and then up-sampling).

The block diagram of the scheme where the interpolator precedes the decimator is shown below.

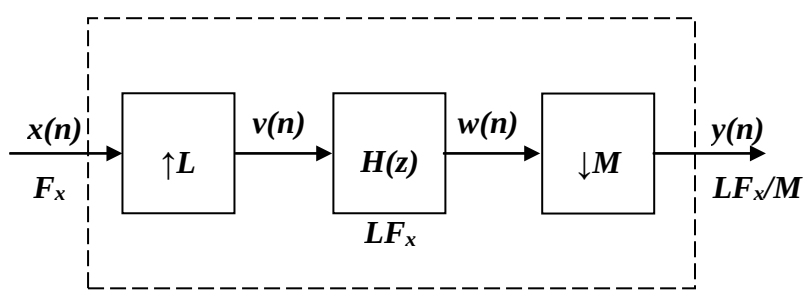
Rational sampling rate conversion



In general, if $L < M$ we have a rational decimator and if $L > M$ we have a rational interpolator. In this set-up interpolation is done before decimation in order to work at the higher sampling rate so as to preserve the original spectral characteristics of $x(n)$. Recall that unless $x(n)$ was originally over-sampled, decimation in itself or decimation prior to interpolation will modify the spectrum of $x(n)$ irrecoverably.

The above configuration has an added benefit that the two filters $H_u(z)$ and $H_d(z)$ in series (which operate at the same sampling rate) can be combined into a single equivalent low pass filter with a frequency response of $H(\omega) = H_u(\omega)H_d(\omega)$. The simplified configuration is shown below.

Rational sampling rate conversion



The bandwidth of the anti-imaging filter $H_u(z)$ is π/L rad., and that of the anti-aliasing filter $H_d(z)$ is π/M rad., so that the bandwidth of the composite anti-imaging and anti-aliasing filter $H(\omega)$ is

$$\omega_c = \min\left(\frac{\pi}{L}, \frac{\pi}{M}\right)$$

and the frequency response is given by

$$H(\omega) = \begin{cases} L, & 0 \le |\omega| < \omega_c \\ 0, & \omega_c \le |\omega| \le \pi \end{cases}$$

In the time domain, the output of the up-sampler, $v(n)$, is given by

$$v(n) = \begin{cases} x(n/L), & n = 0, \pm L, \pm 2L, \ldots \\ 0, & \text{otherwise} \end{cases}$$

and the output of the linear time-invariant filter $H(z)$ is

$$w(n) = \sum_{k=-\infty}^{\infty} h(n-k)\, v(k)$$

Since $v(k) = 0$ except at $k = rL$, where r is an integer between $-\infty$ to ∞, we set $k = rL$. As k goes from $-\infty$ to ∞, r goes from $-\infty$ to ∞, and $v(rL) = x(rL/L) = x(r)$.

$$w(n) = \sum_{r=-\infty}^{\infty} h(n-rL)\, v(rL) = \sum_{r=-\infty}^{\infty} h(n-rL)\, x(r) = \sum_{k=-\infty}^{\infty} h(n-kL)\, x(k)$$

Finally the output of the down sampler is

$$y(n) = w(Mn) = \sum_{k=-\infty}^{\infty} h(Mn-kL)\, x(k)$$

In summary, sampling rate conversion by the factor L/M can be achieved by first increasing the sampling rate by L, accomplished by inserting $L-1$ zeros between successive samples of the input $x(n)$, followed by linear filtering of the resulting sequence to eliminate unwanted images of $X(\omega)$ and, finally, by down-sampling the filtered signal by the factor M to get the output $y(n)$. The sampling rates are related by $F_y = (L/M)F_x$. If $F_y > F_x$, that is, $L > M$, the low pass filter acts as an anti-imaging post-filter to the up-sampler. If $F_y < F_x$, that is, $L < M$, the low pass filter acts as an anti-aliasing pre-filter to the down-sampler.

Example 5.5.4 The signal $x(t) = \cos 2\pi 2t + 0.8 \sin 2\pi 4t$ is sampled at 40 Hz to generate $x(n)$.

a) Give an expression for $x(n)$

b) Design a sampling rate converter to change the sampling frequency of $x(n)$ by a factor of 2.5. Give an expression for $y(n)$.

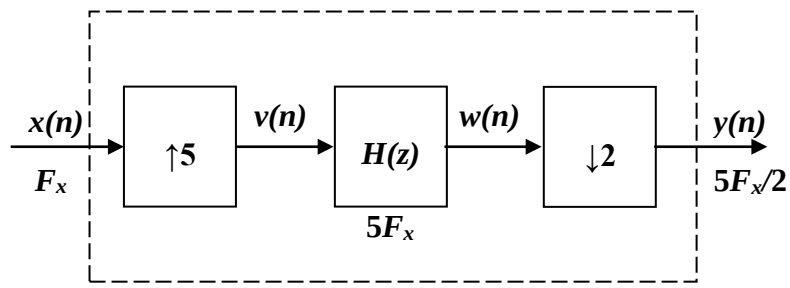
c) Design a sampling rate converter to change the sampling frequency of $x(n)$ by a factor of 0.4. Give an expression for $y(n)$.

Solution

a) $x(n) = \cos 2\pi 2nT + 0.8 \sin 2\pi 4nT = \cos \pi n/10 + 0.8 \sin \pi n/5$

b) The rate conversion factor is $L/M = 2.5 = 5/2$. We do this by first up-sampling by a factor of 5, then down-sampling by a factor of 2. Roughly speaking, $y(n) = x(2n/5)$.

Rational sampling rate conversion

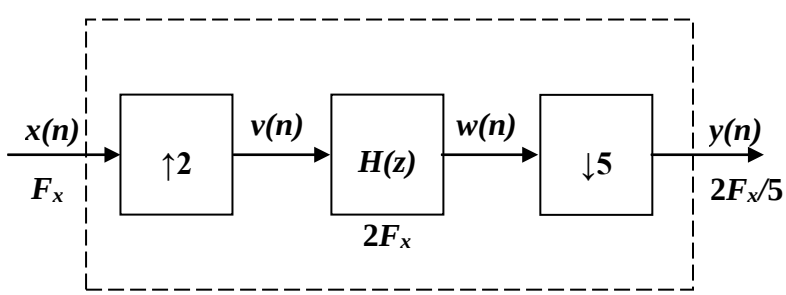


The up-sampling requires an anti-imaging LP filter of bandwidth of $\pi/L = \pi/5$ rad., and a gain of 5. The down-sampling requires an anti-aliasing LP filter of band width $\pi/M = \pi/2$ rad. When the up-sampler precedes the down-sampler we have the following configuration where the filter has the smaller of the two band widths, that is, $\pi/5$ rad. The gain is 5 and is always determined by the up-sampler.

Write equations for $v(n)$, $w(n)$, and $y(n)$ based on the above diagram and assuming $H(z)$ is an FIR filter of N coefficients.

c) The rate conversion factor is $L/M = 0.4 = 2/5$. We do this by first up-sampling by a factor of 2, then down-sampling by a factor of 5. Roughly speaking, $y(n) = x(5n/2)$. Band width of filter = $\pi/5$ rad., and gain = 2, once again determined by the up-sampler.

Rational sampling rate conversion



Write equations for $v(n)$, $w(n)$, and $y(n)$ based on the above diagram and assuming $H(z)$ is an FIR filter N coefficients.

Multistage conversion The composite anti-imaging and anti-aliasing LP filter band width is π/L or π/M whichever is smaller. That is, the filter band width is determined by the larger number of L and M. However, if either L or M is a very large number the filter bandwidth is very narrow.

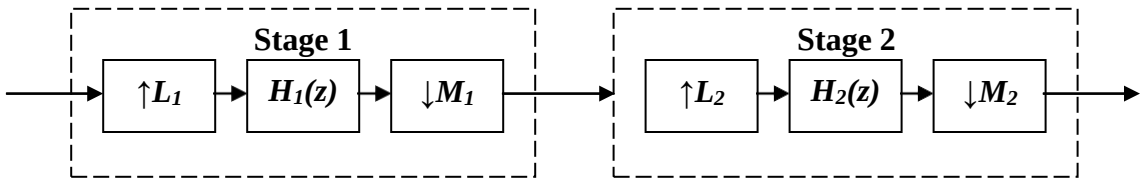
Narrowband FIR (linear phase) filters can require a very large number of coefficients (see Unit VI, FIR Filters, Example 6). This can pose problems in

1. Increased storage space for coefficients,
2. Long computation time, and
3. Detrimental finite word length effects

The latter drawback is minimized by using a multistage sampling rate converter where the conversion ratio L/M is factored into the product of several ratios each of which has its own smaller L and M values. If, for instance, the ratio L/M is split into the product of two ratios

$$\frac{L}{M} = \left(\frac{L_1}{M_1}\right)\left(\frac{L_2}{M_2}\right)$$

where the L's and M's on the right hand side are smaller, we may implement the rate conversion in two stages as shown below



Example 7.5.5 [Rational sampling rate converter][CD, DAT] Digital audio tape (DAT) used in sound recording studios has a sampling rate of 48 kHz, while a compact disc (CD) is recorded at a sampling rate of 44.1 kHz. Design a sampling rate converter that will convert the DAT signal *x(n)* to a signal *y(n)* for CD recording.

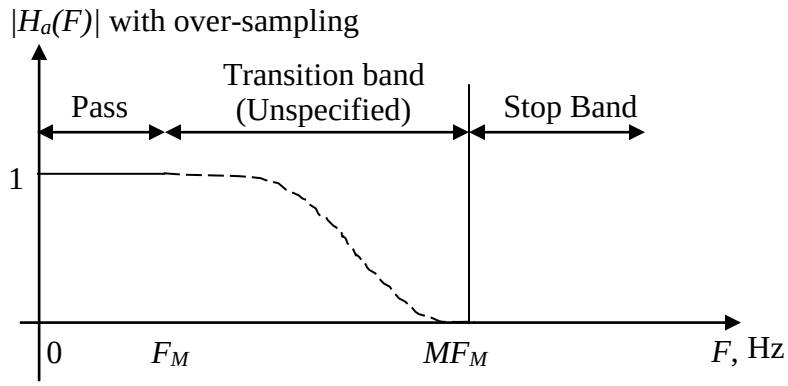
Over-sampling analog-to-digital converter (ADC) [Ref. SKMitra, Sec 4.8.4] A practical difficulty with analog to digital conversion is the need for a low pass analog anti-aliasing prefilter to band limit the signal to less than half of the sampling rate. High-order analog filters are expensive, and they are also difficult to keep in calibration. The combination of over-sampling followed by down-sampling can be used to transfer some of the anti-aliasing burden from the analog into the digital domain, and thereby use a simpler low order analog filter.

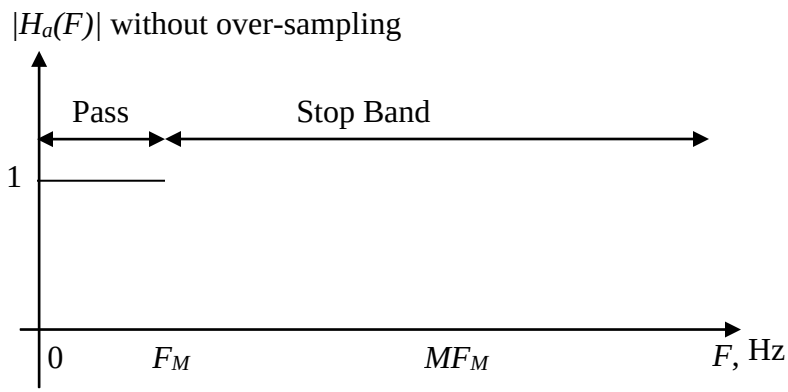
As an example, a typical compact disk encoding system may employ an *over-sampling sigma-delta A/D converter* which over-samples at 3175.2 kHz which is then brought down to the CD sampling rate of 44.1 kHz. This amounts to over-sampling by a factor of 72 (= 3175.2/44.1).

Suppose the range of frequencies of interest in the signal *x(t)* is $0 \le |F| \le F_M$. Normally we would band-limit *x(t)* to the maximum frequency F_M with a sharp cut-off analog low pass filter and sample it at a rate of $F_s = 2F_M$ at least (strictly, $F_s \ge 2F_M$). Suppose that we instead over-sample *x(t)* by an integer factor M, at $F_s = M\,(2F_M)$. This significantly reduces the requirements for the anti-aliasing filter which may be specified more leniently as

$$H_a(s) = \begin{cases} 1, & 0 \le |F| \le F_M \\ 0, & MF_M \le |F| < \infty \end{cases}$$

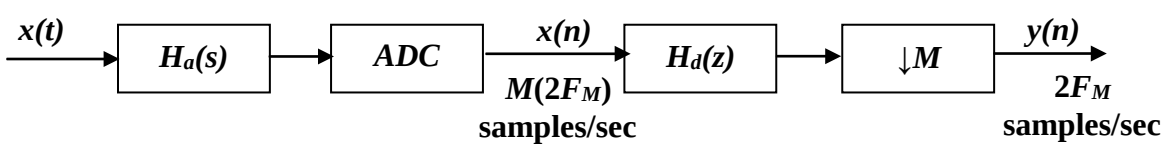
Though this is still an ideal low pass filter in the pass band and stop band, its transition band width is no longer zero and it may be approximated with an inexpensive first- or second-order Butterworth filter as shown below.





We are paying the price of a *higher sampling rate* for the benefit of a *cheaper analog anti-aliasing filter*. The result is a discrete-time signal that is sampled at a much higher rate than $2F_M$. Following the sampling operation, we can reduce this sampling rate to the minimum value using a decimator. The resulting structure of the *over-sampling ADC* is shown in the block diagram below. There are *two* anti-aliasing filters, a low-order analog filter $H_a(s)$ with cut-off frequency F_M rad/sec., and a high-order digital filter $H_d(z)$, with a cut-off frequency (π/M) rad/sample.

Over-sampling AD converter



A second benefit of using the over-sampling ADC is the *reduction in quantization noise*. If q is the quantization step size (precision), then the quantization noise in $x(n)$ is $\sigma_x^2 = q^2/12$, and the noise appearing in the output $y(n)$ in the above scheme is $\sigma_y^2 = \sigma_x^2/M = q^2/12M$, a reduction by a factor of M.

Identities

A sampling rate converter (the $*M* or ‡*L* operation) is a linear time-varying system. On the other hand, the filters $H_d(z)$ and $H_u(z)$ are linear time-invariant systems. In general, the order of a sampling rate converter and a linear time-invariant system *cannot* be interchanged. We derive below several identities, two of which are known as ***noble identities*** (viz., identities 3 and 6, all the others being special cases), which help to swap the position of a filter with that of a down-sampler or up-sampler by *properly modifying the filter*.

Recall that the input-output description of a down-sampler is

$$y(n) = x(Mn) \text{ — } Y(z) = \frac{1}{M}\sum_{k=0}^{M-1} X\left(e^{-j2\pi k/M} z^{1/M}\right) \text{ — } Y(e^{j\omega}) = \frac{1}{M}\sum_{k=0}^{M-1} X\left(e^{j(\omega-2\pi k)/M}\right)$$

and the same for an up-sampler is

$$y(n) = \begin{cases} x(n/L), & n = 0, \pm L, \pm 2L, \ldots \\ 0, & \text{otherwise} \end{cases} \text{ — } Y(z) = X(z^L) \text{ — } Y(e^{j\omega}) = X(e^{j\omega L})$$

which we use in the following development.

Example 7.6.1 Show that the following systems are equivalent.

In the structure on the left the process of generating *y(.)* consists of multiplying every input sample *x(.)* by *a* and then, in the down-sampling process, dropping *(M–1)* of these products for every M^{th} one we keep. The structure on the right is more efficient computationally: the *(M–1)* samples are dropped first and every M^{th} is multiplied by *a*, that is, only the samples that are retained are multiplied. The number of multiplications is reduced by $\frac{100(M-1)}{M}$%.

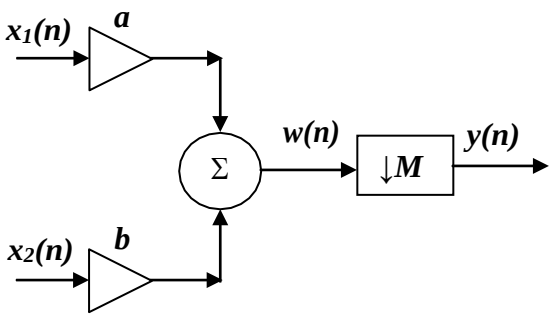
Example 5.6.2 Show that the following systems are equivalent.

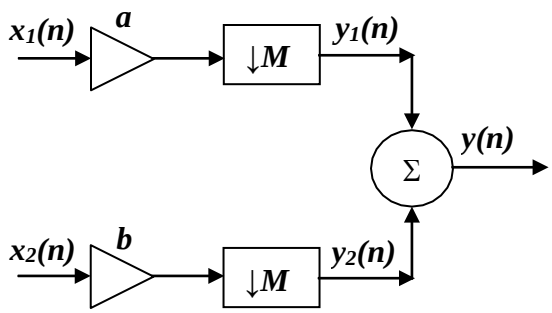
In the structure on the left the process of generating *y(.)* consists of first up-sampling, that is, inserting zeros between consecutive points of *x(n)* and then multiplying by *a*. In the process the (*L*–1) zeros are also multiplied. The structure on the right is more efficient computationally: the sequence *x(n)* is first multiplied and then the zeros inserted. The number of multiplications is reduced by $\frac{100(L-1)}{L}$%.

Identity #1 If we use the notation $\$M\{.\}$ to mean the down-sampling of the signal in braces, then we have

$$\$M\{ax_1(n)+bx_2(n)\} = \$M\{ax_1(n)\} + \$M\{bx_2(n)\} \qquad < (1)$$
$$= a\$M\{x_1(n)\} + b\$M\{x_2(n)\} \qquad < (2)$$

In words, the result of down-sampling the weighted sum of signals equals the weighted sum of the down-sampled signals. In other words, the two block diagrams below are equivalent.





In the diagram on the left the weighted sum of two inputs is down-sampled:

$$w(n) = a\,x_1(n) + b\,x_2(n) \qquad — \qquad W(e^{j\omega}) = \mathrm{a}\,X_1(e^{j\omega}) + \mathrm{b}\,X_2(e^{j\omega})$$

The output and its spectrum are then given by

$$y(n) = w(Mn)$$
$$Y(e^{j\omega}) = \frac{1}{M}\sum_{k=0}^{M-1} W\left(e^{j(\omega-2\pi k)/M}\right)$$
$$= \frac{a}{M}\sum_{k=0}^{M-1} X_1\left(e^{j(\omega-2\pi k)/M}\right) + \frac{b}{M}\sum_{k=0}^{M-1} X_2\left(e^{j(\omega-2\pi k)/M}\right) \qquad < (A)$$

In the diagram on the right the weighted inputs are down-sampled and added to form the output.

$$y_1(n) = a\,x_1(Mn) \qquad — \qquad Y_1(e^{j\omega}) = \frac{a}{M}\sum_{k=0}^{M-1} X_1\left(e^{j(\omega-2\pi k)/M}\right)$$
$$y_2(n) = b\,x_2(Mn) \qquad — \qquad Y_2(e^{j\omega}) = \frac{b}{M}\sum_{k=0}^{M-1} X_2\left(e^{j(\omega-2\pi k)/M}\right)$$

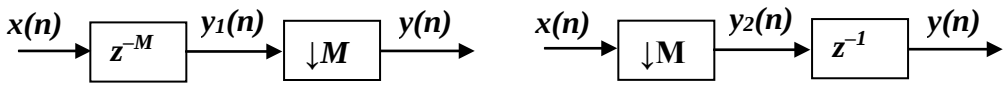
The output and its spectrum are given by

$$y(n) = y_1(n) + y_2(n)$$
$$Y(e^{j\omega}) = Y_1(e^{j\omega}) + Y_2(e^{j\omega}) = \frac{a}{M}\sum_{k=0}^{M-1} X_1\left(e^{j(\omega-2\pi k)/M}\right) + \frac{b}{M}\sum_{k=0}^{M-1} X_2\left(e^{j(\omega-2\pi k)/M}\right) < (B)$$

Equations (A) and (B) are identical. QED.

Eventually, by virtue of Example 6.1, the down-samplers are moved to the upstream side of the multipliers which would correspond to Eq. (2).

Identity #2 A delay of M sample periods before an M-fold down-sampler is the same as a delay of one sample period after the down-sampler.



In the first case (diagram on the left) we have

$$y_1(n) = x(n-M) \quad — \quad Y_1(z) = z^{-M} X(z) \qquad < (1)$$

and

$$y(n) = y_1(Mn) \quad — \quad Y(z) = \frac{1}{M}\sum_{k=0}^{M-1} Y_1\left(e^{-j2\pi k/M} z^{1/M}\right) \qquad < (2)$$

Note from (1) that

$$Y_1\left(e^{-j2\pi k/M} z^{1/M}\right) = Y_1(z)\Big|_{z=e^{-j2\pi k/M} z^{1/M}} = z^{-M} X(z)\Big|_{z=e^{-j2\pi k/M} z^{1/M}}$$

$$= \left(e^{-j2\pi k/M} z^{1/M}\right)^{-M} X\left(e^{-j2\pi k/M} z^{1/M}\right)$$

Substituting this in (2) we have

$$Y(z) = \frac{1}{M}\sum_{k=0}^{M-1}\left(e^{-j2\pi k/M} z^{1/M}\right)^{-M} X\left(e^{-j2\pi k/M} z^{1/M}\right)$$

$$= \frac{1}{M}\sum_{k=0}^{M-1}\left(e^{j2\pi kM/M} z^{-M/M}\right) X\left(e^{-j2\pi k/M} z^{1/M}\right) = \frac{1}{M}\sum_{k=0}^{M-1}\left(1 z^{-1}\right) X\left(e^{-j2\pi k/M} z^{1/M}\right)$$

$$= z^{-1}\frac{1}{M}\sum_{k=0}^{M-1} X\left(e^{-j2\pi k/M} z^{1/M}\right) \qquad < (A)$$

In the second case (diagram on the right) we have

$$y_2(n) = x(nM) \quad — \quad Y_2(z) = \frac{1}{M}\sum_{k=0}^{M-1} X\left(e^{-2\pi k/M} z^{1/M}\right) \qquad < (3)$$

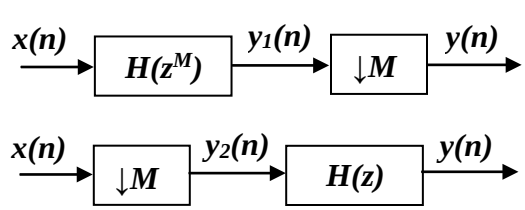
$$y(n) = y_2(n-1) \quad — \quad Y(z) = z^{-1} Y_2(z) \qquad < (4)$$

Substituting from (3) into (4) we have

$$Y(z) = z^{-1} Y_2(z) = z^{-1}\frac{1}{M}\sum_{k=0}^{M-1} X\left(e^{-2\pi k/M} z^{1/M}\right) \qquad < (B)$$

Equations (A) and (B) are identical. QED.

Identity #3 (Noble identity) An M-fold down-sampler followed by a linear time invariant filter $H(z)$ is equivalent to a linear time invariant filter $H(z^M)$ followed by an M-fold down-sampler. Note that the second identity is a special case of this identity with $H(z) = z^{-1}$ and $H(z^M) = z^{-M}$.



For the system consisting of the filter followed by the down-sampler we have

$$Y_1(z) = H(z^M)\, X(z)$$

$$Y(z) = \frac{1}{M}\sum_{k=0}^{M-1} Y_1\left(e^{-j2\pi k/M} z^{1/M}\right)$$

Note that

$$\begin{aligned}
Y_1\left(e^{-j2\pi k/M} z^{1/M}\right) &= Y_1(z)\Big|_{z = e^{-j2\pi k/M} z^{1/M}} \\
&= H\left(z^M\right) X(z)\Big|_{z = e^{-j2\pi k/M} z^{1/M}} \\
&= H\left(\left(e^{-j2\pi k/M} z^{1/M}\right)^M\right) X\left(e^{-j2\pi k/M} z^{1/M}\right) \\
&= H\left(e^{-j2\pi k M/M} z^{M/M}\right) X\left(e^{-j2\pi k/M} z^{1/M}\right) \\
&= H(1z)\, X\left(e^{-j2\pi k/M} z^{1/M}\right) = H(z)\, X\left(e^{-j2\pi k/M} z^{1/M}\right)
\end{aligned}$$

Thus

$$\begin{aligned}
Y(z) &= \frac{1}{M}\sum_{k=0}^{M-1} H(z)\, X\left(e^{-j2\pi k/M} z^{1/M}\right) \\
&= H(z)\, \frac{1}{M}\sum_{k=0}^{M-1} X\left(e^{-j2\pi k/M} z^{1/M}\right) \qquad \text{< (A)}
\end{aligned}$$

For the system consisting of the down sampler followed by the filter we have

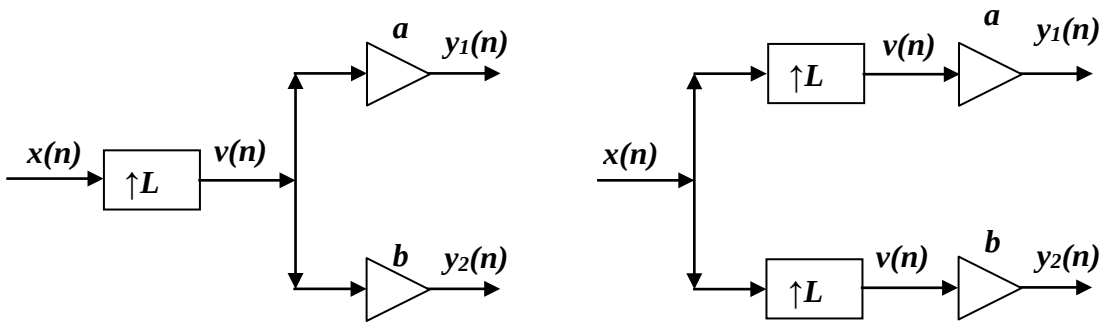
$$y_2(n) = x(nM)$$

$$Y_2(z) = \frac{1}{M}\sum_{k=0}^{M-1} X\left(e^{-2\pi k/M} z^{1/M}\right)$$

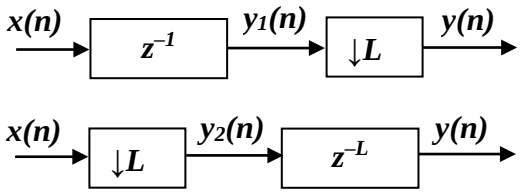
$$Y(z) = H(z)\, Y_2(z) = H(z)\, \frac{1}{M}\sum_{k=0}^{M-1} X\left(e^{-2\pi k/M} z^{1/M}\right) \qquad \text{< (B)}$$

Equations (A) and (B) are identical. Thus the two systems are equivalent.

Identity #4 (This identity contains no summing junction as does identity #1.) Eventually, by virtue of Example 6.2, the up-samplers are moved to the downstream side of the multipliers.



Identity #5 A delay of one sample period before an L-fold up-sampler is the same as a delay of L sample periods after the up-sampler.



For the system consisting of the up-sampler preceded by z^{-1} *we* have

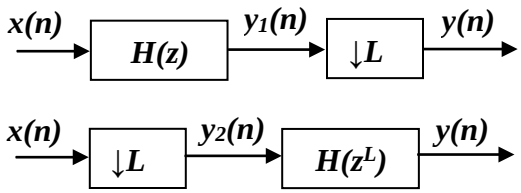
$$Y_1(z) = z^{-1}\, X(z)$$
$$Y(z) = Y_1(z^L) = z^{-L}\, X(z^L) \quad \text{(A)}$$

For the system consisting of the up-sampler followed by z^{-L} we have

$$Y_2(z) = X(z^L)$$
$$Y(z) = z^{-L}\, Y_2(z) = z^{-L}\, X(z^L) \quad \text{(B)}$$

Since equations (A) and (B) are identical the two systems are equivalent.

Identity #6 (Noble identity) An L-fold up-sampler preceded by a linear time invariant filter $H(z)$ is equivalent to a linear time invariant filter $H(z^L)$ preceded by an L-fold up-sampler. Note that the fifth identity is a special case of this identity with $H(z) = z^{-1}$ and $H(z^L) = z^{-L}$.



For the system consisting of the filter followed by the up-sampler we have

$$Y_1(z) = H(z)\, X(z)$$
$$Y(z) = Y_1(z^L) = H(z^L)\, X(z^L) \quad \text{(A)}$$

For the system consisting of the up-sampler followed by the filter we have

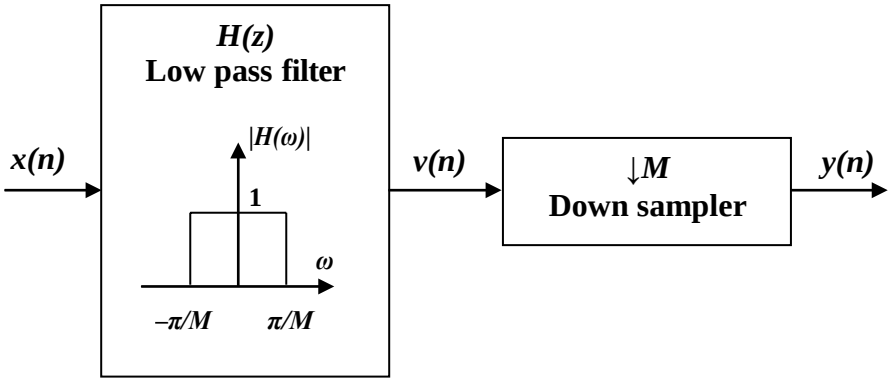
$$Y_2(z) = X(z^L)$$
$$Y(z) = H(z^L)\, Y_2(z) = H(z^L)\, X(z^L) \quad \text{(B)}$$

Since equations (A) and (B) are identical the two systems are equivalent.

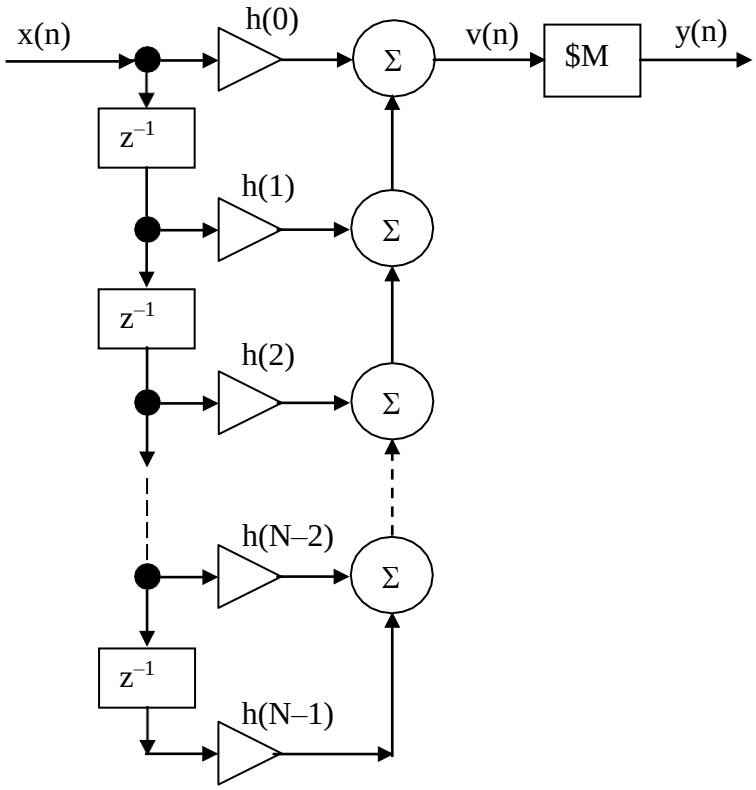
FIR implementation of sampling rate conversion

The anti-aliasing filter in a decimator and the anti-imaging filter in an interpolator may each be either an FIR or an IIR filter, the former being preferred since it offers linear phase. We give here the FIR implementation.

Implementation of Decimator The process of decimation consists of an anti-aliasing (low pass) filter followed by a down-sampler. We repeat below the block diagram developed earlier.



Taking the filter *H(z)* to be an FIR filter, the decimator is implemented as shown below, using a direct form structure for *H(z)*. Note that the coefficients $\{b_i, i = 0 \text{ to } (N-1)\}$, used in earlier formulations, are the same as $\{h(i), i = 0 \text{ to } (N-1)\}$ used in this diagram. Further, the FIR filter here is implemented with *N* coefficients rather than *(N+1)* coefficients.

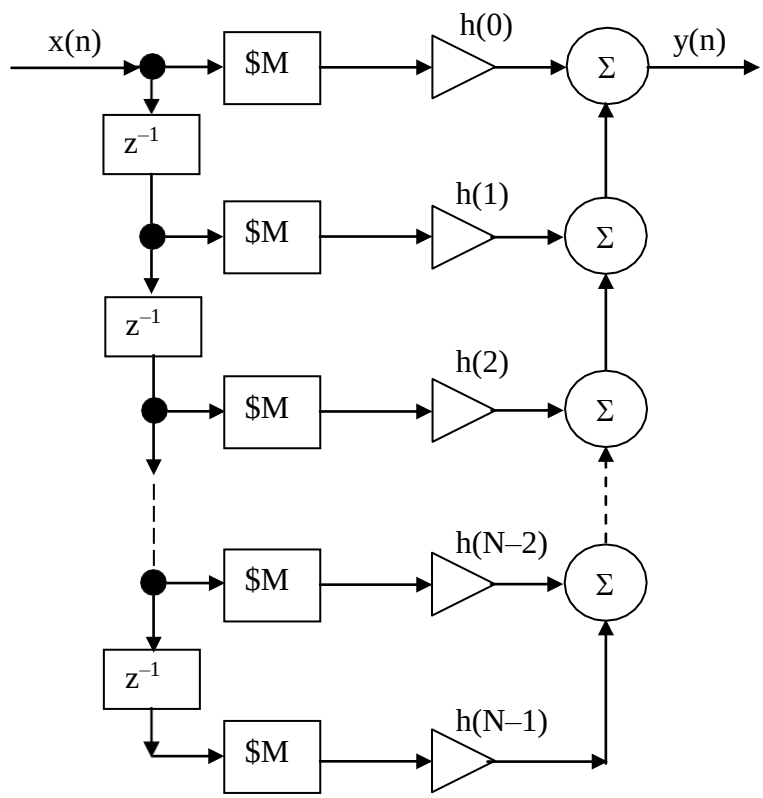


The implementation equations which correspond to the above structure are

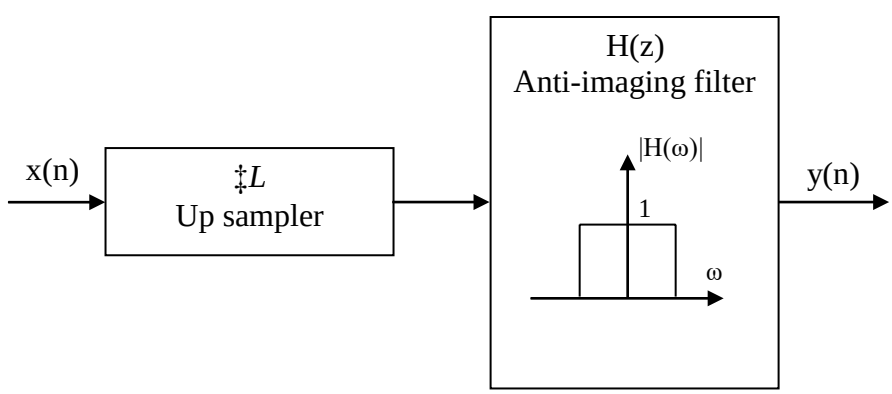
$$v(n) = \sum_{r=0}^{N-1} h(r)\, x(n-r) \qquad \text{and} \qquad y(n) = v(Mn) = \sum_{r=0}^{N-1} h(r)\, x(Mn-r)$$

We first compute $v(n)$ for all values of n. Then $y(n)$ is obtained by retaining every M^{th} value of $v(.)$, dropping the intervening $(M-1)$ values. In other words there are $(M-1)$ computations of $v(.)$ that could be avoided.

We may use identity #1 to move the down-sampler to the left of the adders and the result of Example 6.1 to move it to the upstream side of the multipliers as shown below. As a result the number of multiplications is reduced by $\dfrac{100(M-1)}{M}$%.

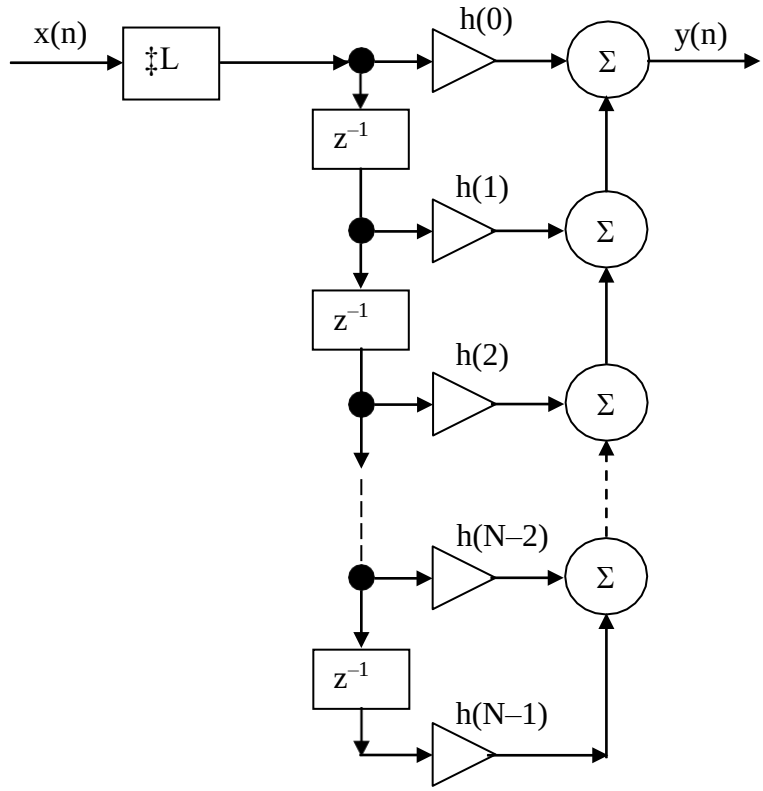


Implementation of Interpolator The process of interpolation consists of an up-sampler followed by an anti-imaging (low pass) filter. We repeat below the block diagram developed earlier.



Taking the filter *H(z)* to be an FIR filter, the interpolator is implemented as shown below, using a direct form structure for *H(z)*. Note that the coefficients $\{b_i, i = 0 \text{ to } (N-1)\}$, used in earlier formulations, are the same as $\{h(i), i = 0 \text{ to } (N-1)\}$ used in this diagram. Further, the FIR filter here is implemented with *N* coefficients rather than *(N+1)* coefficients.

We shall find it more convenient to use the transposed form of the FIR filter rather than the structure actually shown here, so here follows a digression on the transposed structure.

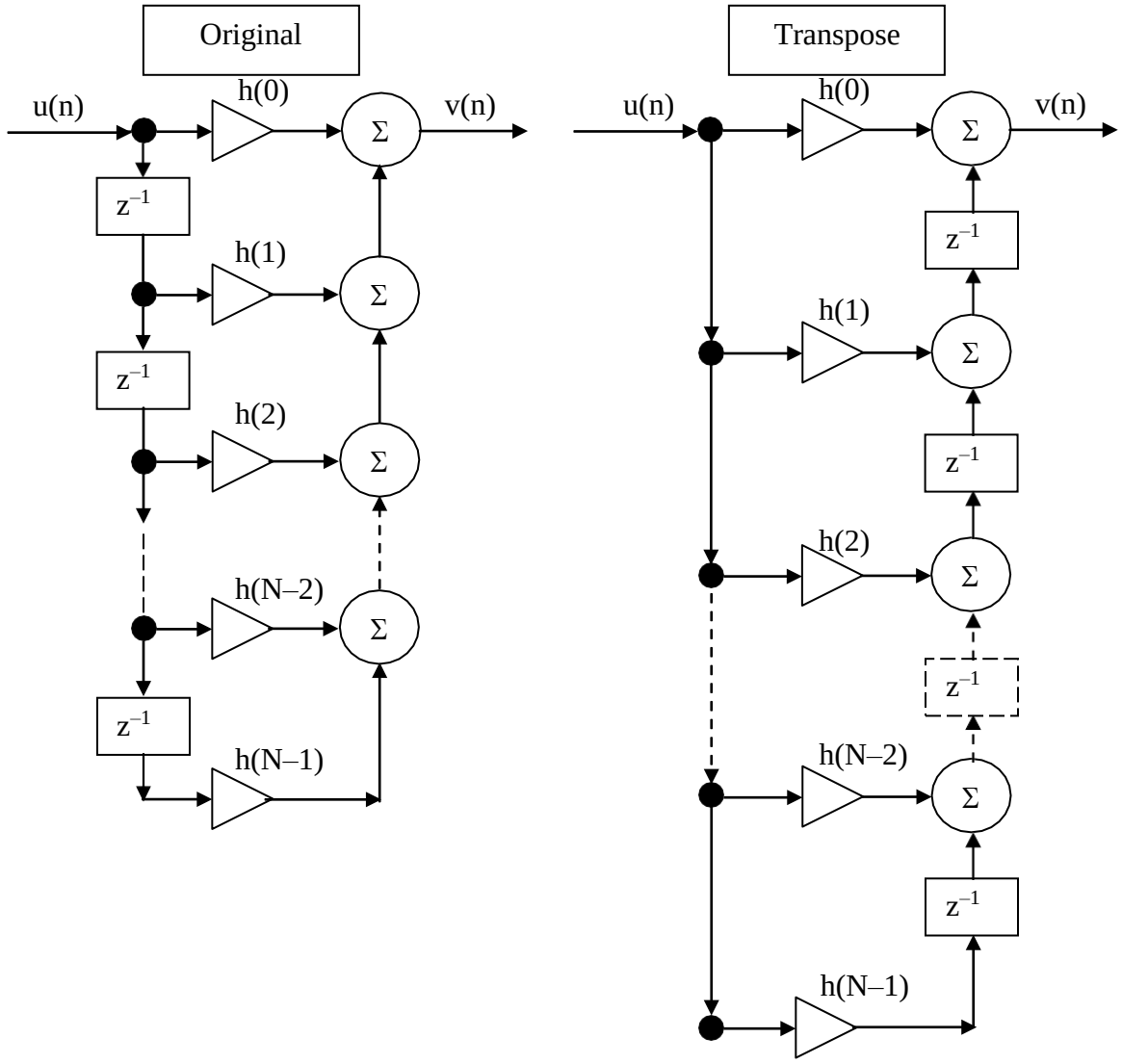


Start of Digression

Transposed Structure According to the transposition theorem the transposed form of a filter has the same transfer function as the filter. The transposed form of a given filter structure is found as follows:

1. Construct the signal flow graph of the filter.
2. Reverse the direction of arrow on every branch.
3. Interchange the inputs and outputs.
4. Reverse the roles of all nodes: an adder becomes a pick-off point and a pick-off point becomes an adder.

If we apply this procedure to the FIR structure in the above interpolator the result is the transpose shown below. The intermediate steps are omitted.



Start of Aside

As an aside note that the FIR structure is simple enough that the following algebraic manipulation can be used to proceed from the original FIR structure to the transpose structure. Let the system function be

$$\frac{V(z)}{U(z)} = H(z) = h(0) + h(1)z^{-1} + h(2)z^{-2} + h(3)z^{-3} + h(4)z^{-4} + \ldots + h(N-1)z^{-(N-1)}$$

This may be rearranged as

$$\begin{aligned} V(z) &= H(z)U(z) \\ &= h(0)U(z) + h(1)z^{-1}U(z) + h(2)z^{-2}U(z) + h(3)z^{-3}U(z) + h(4)z^{-4}U(z) + \ldots \\ &\quad + h(N-1)z^{-(N-1)}U(z) \\ &= h(0)U(z) + z^{-1}(h(1)U(z) + z^{-1}(h(2)U(z) + z^{-1}(h(3)U(z) + \ldots \\ &\quad \ldots + z^{-1}(h(N-2)U(z) + z^{-1}h(N-1)U(z))))) \end{aligned}$$

This last equation, proceeding from right to left, performs the following in the time domain:

- Multiply $u(n)$ by $h(N-1)$, giving $h(N-1)u(n)$
- Delay by 1 unit, giving $h(N-1)u(n-1)$
- Add to $h(N-2)u(n)$, giving $h(N-2)u(n)+h(N-1)u(n-1)$
- Delay by 1 unit, giving $h(N-2)u(n-1)+h(N-1)u(n-2)$
- Add to $h(N-3)u(n)$, giving $h(N-3)u(n)+h(N-2)u(n-1)+h(N-1)u(n-2)$
- Delay by 1 unit, giving $h(N-3)u(n-1)+h(N-2)u(n-2)+h(N-1)u(n-3)$
- …
- Add to $h(0)u(n)$

all of which yields the implementation of the difference equation

$$\begin{aligned} v(n) &= h(0)u(n) + h(1)u(n-1) + h(2)u(n-2) + \ldots \\ &\quad \ldots + h(N-2)u(n-N+2) + h(N-1)u(n-N-1) \end{aligned}$$

Further, the equation

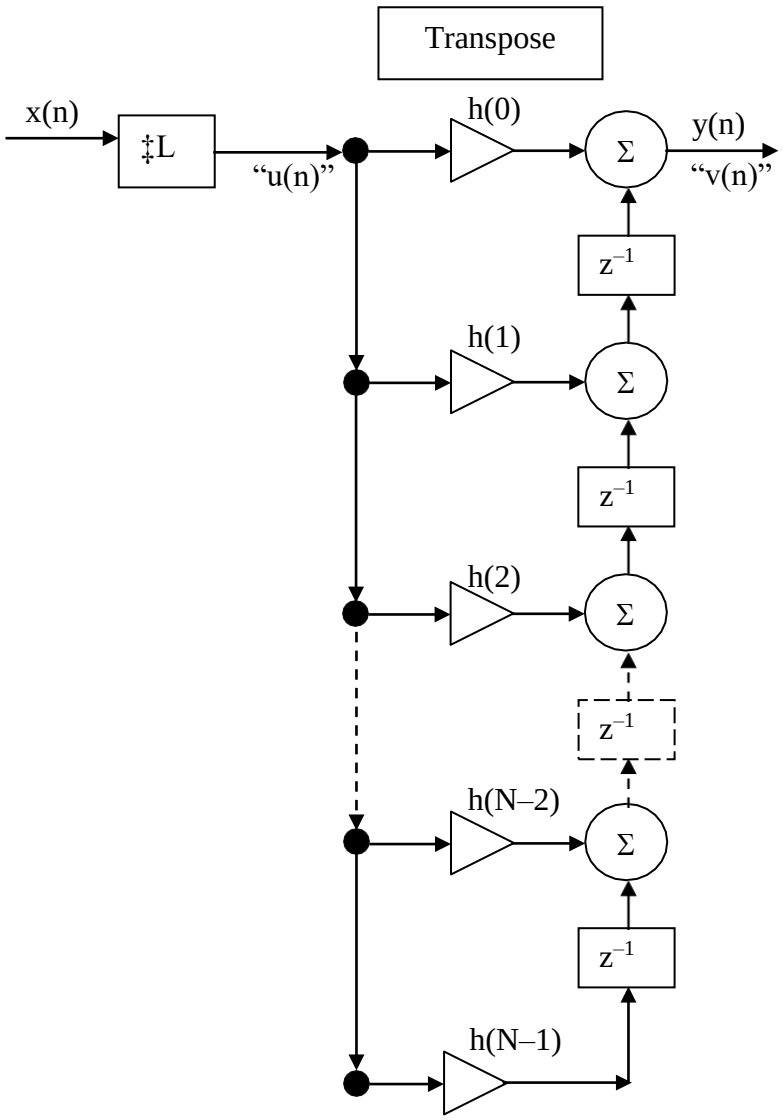
$$\begin{aligned} V(z) &= h(0)U(z) + z^{-1}(h(1)U(z) + z^{-1}(h(2)U(z) + z^{-1}(h(3)U(z) + \ldots \\ &\quad \ldots + z^{-1}(h(N-2)U(z) + z^{-1}h(N-1)U(z))))) \end{aligned}$$

also suggests the transpose structure previously developed according to the rules.

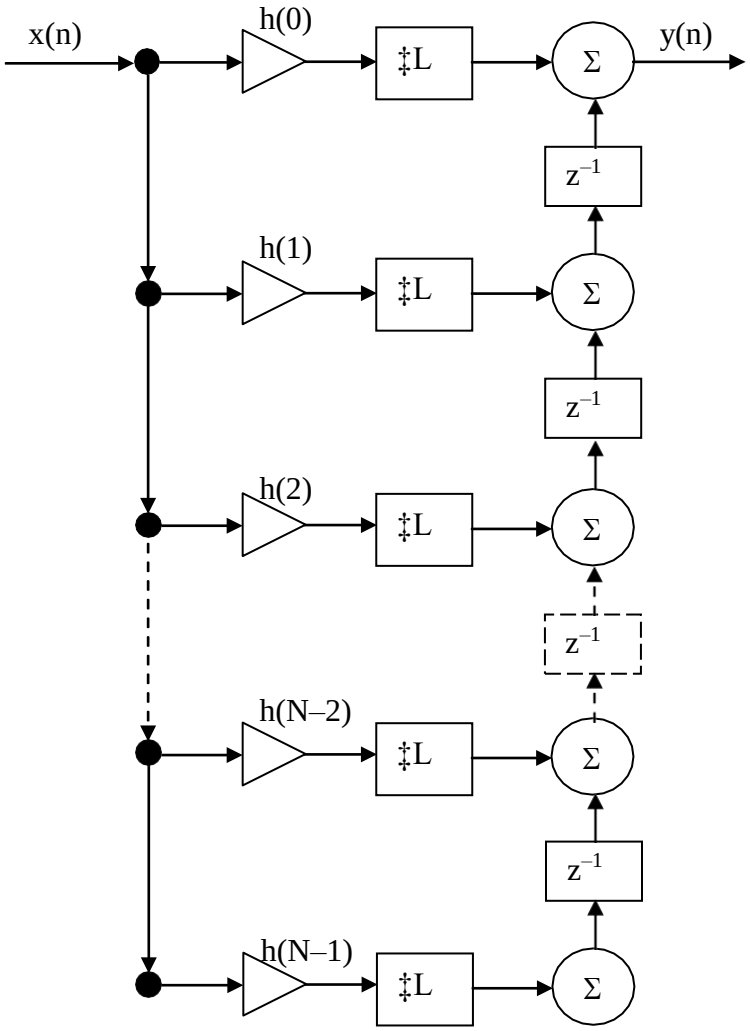
End of Aside

End of Digression

Resumption of Implementation of Interpolator Using the transposed form of the FIR filter the structure of the interpolator appears as below:



We may use identity #4 and the result of Example 6.2 to move the up-sampler to the right of the multipliers as shown below. As a result the number of multiplications is reduced by $\frac{100(L-1)}{L}$ %.



Polyphase structures

The polyphase structure for FIR Filters was developed for the efficient implementation of sampling rate converters; however, it can be used in other applications. Further, the polyphase structure can be developed for any filter, FIR or IIR. We give below an introduction.

Polyphase Structure for FIR Filters The impulse response of the FIR filter *h(n)* is of finite length, *N*. The system function with *N* coefficients is

$$H(z) = \sum_{n=0}^{N-1} h(n) z^{-n} = h(0) + h(1)z^{-1} + h(2)z^{-2} + h(3)z^{-3} + h(4)z^{-4} + \ldots + h(N-1)z^{-(N-1)}$$

We shall use another parameter *M*: we shall divide the number of coefficients into *M* groups (or branches or phases), modulo *M*. In other words the *N* terms in *H(z)* are arranged into *M* branches with each branch containing at most $\left(Int\left(\overline{N-1/M}\right)+1\right)$ terms.

Type 1 polyphase decomposition For illustration, let *N* = 11 and *M* =2 so that one group contains 6 coefficients and the other 5 as developed below:

$$H(z) = \sum_{n=0}^{10} h(n)\, z^{-n} = h(0) + h(1)\, z^{-1} + h(2)\, z^{-2} + h(3)\, z^{-3} + h(4)\, z^{-4} + \ldots + h(10)\, z^{-10}$$

$$= h(0) + h(2)z^{-2} + h(4)z^{-4} + h(6)z^{-6} + h(8)z^{-8} + h(10)z^{-10} \qquad \rightarrow 1^{st} \text{ group}$$

$$+ h(1)z^{-1} + h(3)z^{-3} + h(5)z^{-5} + h(7)z^{-7} + h(9)z^{-9} \qquad \rightarrow 2^{nd} \text{ group}$$

$$= h(0) + h(2)\, z^{-2} + h(4)\, z^{-4} + h(6)\, z^{-6} + h(8)\, z^{-8} + h(10)\, z^{-10}$$

$$+ z^{-1}\{\, h(1) + h(3)\, z^{-2} + h(5)\, z^{-4} + h(7)\, z^{-6} + h(9)\, z^{-8}\,\}$$

Define

$Int\left(\overline{N-1/M}\right)$ = integer part of the argument

$$P_0(z) = h(0) + h(2)z^{-1} + h(4)z^{-2} + h(6)z^{-3} + h(8)z^{-4} + h(10)z^{-5} = \sum_{n=0}^{Int\left(\overline{11-1/2}\right)} h(2n+0)z^{-n}$$

(More generally, $P_0(z) = \sum_{n=0}^{Int\left(\overline{N-1/M}\right)} h(Mn+0)z^{-n}$)

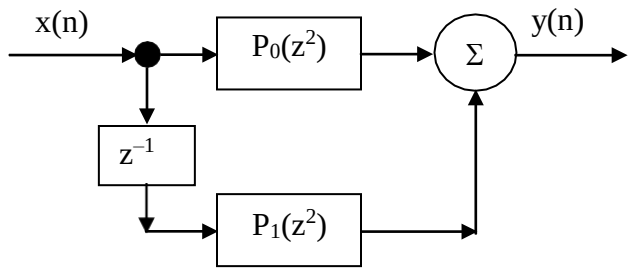
$$P_1(z) = h(1) + h(3)z^{-1} + h(5)z^{-2} + h(7)z^{-3} + h(9)z^{-4} = \sum_{n=0}^{Int\left(\overline{11-1/2}\right)} h(2n+1)z^{-n} \quad (h(11) = 0)$$

(More generally, $P_1(z) = \sum_{n=0}^{Int\left(\overline{N-1/M}\right)} h(Mn+1)\, z^{-n}$)

In this specific case we have

$$\frac{Y(z)}{X(z)} = H(z) = P_0(z^2) + z^{-1} P(z^2) = \sum_{n=0}^{1} z^{-m} P(z^2)$$

This decomposition of H(z) is known as **type 1 polyphase decomposition**. The corresponding structure is shown below. The functions $P_0(z^2)$ and $P_1(z^2)$ can each be implemented as a direct form.



By observing the expressions for $P_0(z)$ and $P_1(z)$ we can further generalize the functions $P_m(z)$ for any m as

$$P_m(z) = \sum_{n=0}^{Int(N-1/M)} h(Mn+m)z^{-n}), \quad m = 0 \text{ to } (M-1)$$

Further, the system function becomes

$$\frac{Y(z)}{X(z)} = H(z) = P_0(z^M) + z^{-1}P_1(z^M) + \ldots + z^{-(M-1)}P_{M-1}(z^M) = \sum_{m=0}^{M-1} z^{-m}P_m(z^M)$$

Example 5.8.1 As another example, let N = 11 and M =3.

$$H(z) = \sum_{n=0}^{10} h(n)\, z^{-n} = h(0) + h(1)\, z^{-1} + h(2)\, z^{-2} + h(3)\, z^{-3} + h(4)\, z^{-4} + \ldots + h(10)\, z^{-10}$$

$$= h(0) + h(3)z^{-3} + h(6)z^{-6} + h(9)z^{-9} \qquad \text{– 1st group}$$

$$+ h(1)z^{-1} + h(4)\, z^{-4} + h(7)\, z^{-7} + h(10)z^{-10} \qquad \text{– 2nd group}$$

$$+ h(2)\, z^{-2} + h(5)\, z^{-5} + h(8)\, z^{-8} \qquad \text{– 3rd group}$$

$$H(z) = h(0) + h(3)\, z^{-3} + h(6)\, z^{-6} + h(9)\, z^{-9}$$

$$+ z^{-1}\{\, h(1) + h(4)\, z^{-3} + h(7)\, z^{-6} + h(10)\, z^{-9}\}$$

$$+ z^{-2}\{\, h(2) + h(5)\, z^{-3} + h(8)\, z^{-6}\}$$

Define

$Int(N-1/M)$ = integer part of the argument

$$P_0(z) = h(0) + h(3)z^{-1} + h(6)z^{-2} + h(9)z^{-3} = \sum_{n=0}^{Int(11-1/3)} h(3n+0)z^{-n}$$

(More generally, $P_0(z) = \sum_{n=0}^{Int(N-1/M)} h(Mn+0)z^{-n}$)

$$P_1(z) = h(1) + h(4)z^{-1} + h(7)z^{-2} + h(10)z^{-3} = \sum_{n=0}^{Int(11-1/3)} h(3n+1)z^{-n}$$

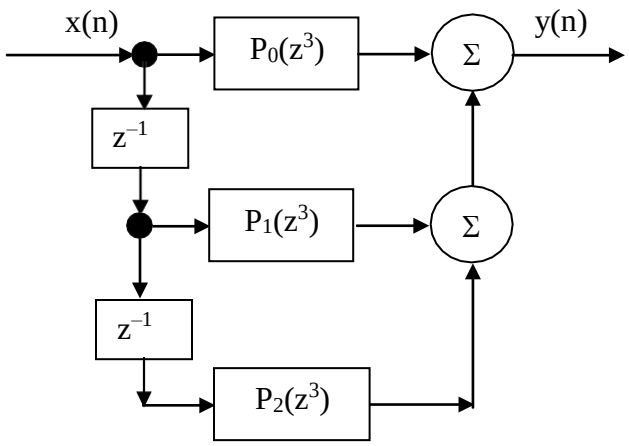
(More generally, $P_1(z) = \sum_{n=0}^{Int(N-1/M)} h(Mn+1)\, z^{-n}$)

$$P_2(z) = h(2) + h(5)z^{-1} + h(8)z^{-2} = \sum_{n=0}^{Int(11-1/3)} h(3n+1)z^{-n} \qquad (h(11) = 0)$$

$$\text{(More generally, } P_2(z) = \sum_{n=0}^{Int(N-1/M)} h(Mn+1)z^{-n}\text{)}$$

In this specific case we have

$$\frac{Y(z)}{X(z)} = H(z) = P_0(z^3) + z^{-1}P_1(z^3) + z^{-2}P_2(z^3) = \sum_{m=0}^{2} z^{-m}P_m(z^3)$$



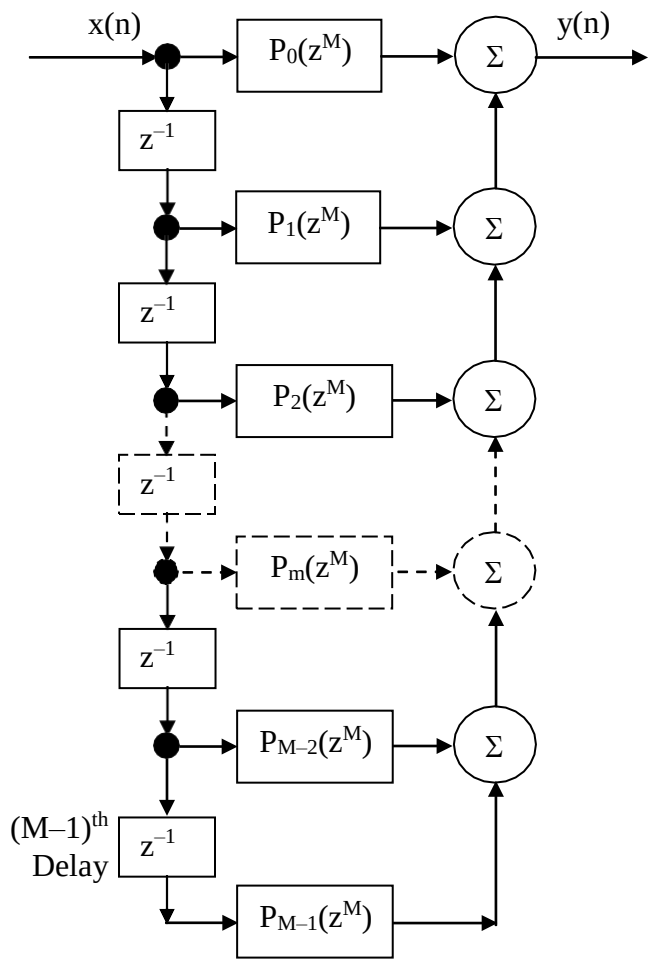
Generalization As mentioned earlier, in the general case for an arbitrary M ($\leq N$) we have

$$P_m(z) = \sum_{n=0}^{Int\left(\frac{N-1}{M}\right)} h(Mn+m)z^{-n}, \qquad m = 0 \text{ to } (M-1)$$

and

$$\frac{Y(z)}{X(z)} = H(z) = P_0(z^M) + z^{-1}P_1(z^M) + \ldots + z^{-(M-1)}P_{M-1}(z^M) = \sum_{m=0}^{M-1} z^{-m}P_m(z^M)$$

This overall operation is known as polyphase filtering.



Type 2 polyphase decomposition Given

$$\frac{Y(z)}{X(z)} = H(z) = P_0(z^M) + z^{-1}P_1(z^M) + \ldots + z^{-(M-1)}P_{M-1}(z^M) = \sum_{m=0}^{M-1} z^{-m}P_m(z^M)$$

set $m = M-1-k$: as m goes from 0 to M–1, k goes from M–1 to 0. Thus

$$H(z) = \sum_{k=M-1}^{0} z^{-(M-1-k)}P_{M-1-k}(z^M)$$

Let

$$Q_k(z^M) = z^{-(M-1-k)}P_{M-1-k}(z^M)$$

This gives us the type 2 polyphase decomposition

$$\sum_{k=M-1}^{0} Q_k(z^M) = \sum_{k=0}^{M-1} Q_k(z^M) = Q_0(z^M) + Q_1(z^M) + \ldots + Q_{M-1}(z^M)$$

Type 3 polyphase decomposition Given

$$\frac{Y(z)}{X(z)} = H(z) = P_0(z^M) + z^{-1}P_1(z^M) + \ldots + z^{-(M-1)}P_{M-1}(z^M) = \sum_{m=0}^{M-1} z^{-m}P_m(z^M)$$

set m = –k: as m goes from 0 to M–1, k goes from 0 to –(M–1) to 0. Thus

$$H(z) = \sum_{k=0}^{-(M-1)} z^{k}P_{-k}(z^M)$$

Let

$$R_k(z^M) = z^{k}P_{-k}(z^M)$$

Example 5.8.2 For the system

$$H(z) = \sum_{n=0}^{6} h(n)\, z^{-n} = h(0) + h(1)z^{-1} + h(2)z^{-2} + h(3)z^{-3} + h(4)z^{-4} + h(5)z^{-5} + h(6)z^{-6}$$

$$= \{ h_0 + h_2 z^{-2} + h_4 z^{-4} + h_6 z^{-6} \} + \{ h_1 z^{-1} + h_3 z^{-3} + h_5 z^{-5} \}$$

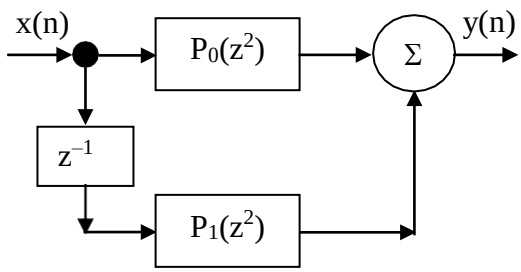
$$= \{ h_0 + h_2 z^{-2} + h_4 z^{-4} + h_6 z^{-6} \} + z^{-1} \{ h_1 + h_3 z^{-2} + h_5 z^{-4} \}$$

$$P_0(z) = h_0 + h_2 z^{-1} + h_4 z^{-2} + h_6 z^{-3}$$

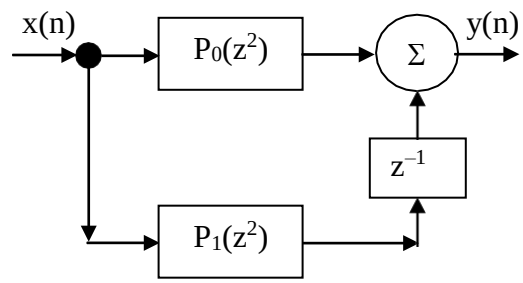
$$P_1(z) = h_1 + h_3 z^{-1} + h_5 z^{-2}$$

The structure is shown below (left). As mentioned earlier $P_0(z^2)$ and $P_1(z^2)$ can each be implemented as a direct form.

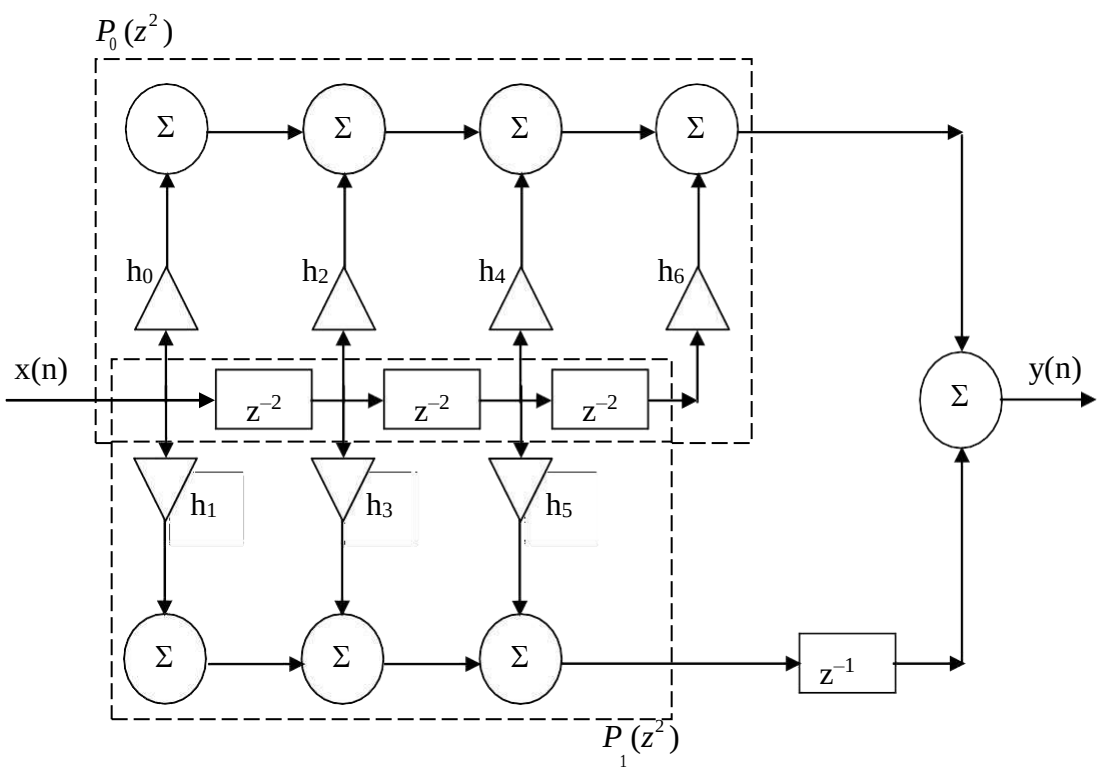
Polyphase realization



Canonical polyphase realization



The structure shown on the right is obtained by moving the delay element z^{-1} to the right of $P_1(z^2)$ these two being in series in the second phase. In this latter case the two systems $P_0(z^2)$ and $P_1(z^2)$ can share the same delay elements (that is, storage locations) even though each has its own set of coefficients, thus resulting in a canonical polyphase realization, shown below.



Polyphase Structure for IIR Filters The anti-aliasing filter in a decimator and the anti-imaging filter in an interpolator may each be either an FIR or an IIR filter. The polyphase structure can be developed for any filter, FIR or IIR, and any finite value of *M*. We now proceed to the case where *h(n)* is an infinitely long sequence:

$$H(z) = \sum_{n=-\infty}^{\infty} h(n)\, z^{-n} = \ldots + h(-M)\, z^{M} + \ldots\, h(-2)\, z^{2} + h(-1)\, z^{1}$$
$$+ h(0) + h(1)\, z^{-1} + h(2)\, z^{-2} + \ldots + h(M-1)\, z^{-(M-1)}$$
$$+ h(M)\, z^{-M} + h(M+1)\, z^{-(M+1)} \ldots$$

Once again we arrange the terms into *M* groups or branches in a modulo-*M* fashion. Each branch contains infinitely many terms. The terms are arranged in tabular form below:

$H(z) = \sum_{n=-\infty}^{\infty} h(n) z^{-n}$					
1st branch‹	…	$+ h(-M)\, z^{M}$	$+ h(0)$	$+ h(M)\, z^{-M}$	…
2nd branch‹	…	$+ h(-M+1)\, z^{M-1}$	$+ h(1)\, z^{-1}$	$+ h(M+1)\, z^{-(M+1)}$	…
	…	…	$+ h(2)\, z^{-2}$	$+ h(M+2)\, z^{-(M+2)}$	…
	…	…	…	…	…
ith branch‹	…	$h(-M+i-1) z^{-(-M+i-1)}$	$h(i-1) z^{-(i-1)}$	$h(M+i-1) z^{-(M+i-1)}$	…
	…	…	…	…	…
	…	$+ h(-2)\, z^{2}$	$+ h(M-2) z^{-(M-2)}$	$+ h(2M-2) z^{-(2M-2)}$	…
Mth branch‹	…	$+ h(-1)\, z^{1}$	$+ h(M-1) z^{-(M-1)}$	$+ h(2M-1) z^{-(2M-1)}$	…

$$\begin{aligned}
H(z) = {} & [\ldots + h(-M) z^{M} + h(0) + h(M) z^{-M} + h(2M) z^{-2M} + \ldots] && 1^{st}\text{ row} \\
& + [\ldots + h(-M+1) z^{M-1} + h(1) z^{-1} + h(M+1) z^{-(M+1)} + \ldots] && 2^{nd}\text{ row} \\
& + [\ldots] + \ldots \\
& + [\ldots + h(-M+i-1)\, z^{-(-M+i-1)} + h(i-1) z^{-(i-1)} + h(M+i-1)\, z^{-(M+i-1)} \\
& \qquad + h(2M+i-1)\, z^{-(2M+i-1)} + \ldots] \\
& \ldots \\
& + [\ldots + h(-2)\, z^{2} + h(M-2)\, z^{-(M-2)} + h(2M-2)\, z^{-(2M-2)} + \ldots] \\
& + [\ldots + h(-1) z^{1} + h(M-1) z^{-(M-1)} + h(2M-1) z^{-(2M-1)} + \ldots] && M^{th}\text{ row}
\end{aligned}$$

We factor out z^{0} from the first row(branch), z^{-1} from the 2nd row, and, in general, $z^{-(i-1)}$ from the ith row to get

$$\begin{aligned}
H(z) = {} & [\ldots + h(-M) z^{M} + h(0) + h(M) z^{-M} + h(2M) z^{-2M} + \ldots] && 1^{st}\text{ row} \\
& + z^{-1} [\ldots + h(-M+1) z^{M} + h(1) + h(M+1) z^{-M} + \ldots] && 2^{nd}\text{ row} \\
& + z^{-2} [\ldots] + \ldots \\
& + z^{-(i-1)} [\ldots + h(-M+i-1) z^{M} + h(i-1) + h(M+i-1) z^{-M} + h(2M+i-1) z^{-2M} + \ldots] \\
& \ldots \\
& + z^{-(M-2)} [\ldots + h(-2)\, z^{M} + h(M-2) + h(2M-2) z^{-M} + \ldots] \\
& + z^{-(M-1)} [\ldots + h(-1)\, z^{M} + h(M-1) + h(2M-1)\, z^{-M} + \ldots]
\end{aligned}$$

Define

$$P_0(z^M) = [\ldots + h(-M)z^M + h(0) + h(M)z^{-M} + h(2M)z^{-2M} + \ldots] = \sum_{n=-\infty}^{\infty} h(nM)z^{-nM}$$

so that

$$P_0(z) = [\ldots + h(-M)z + h(0) + h(M)z^{-1} + h(2M)z^{-2} + \ldots] = \sum_{n=-\infty}^{\infty} h(nM)\, z^{-n}$$

Similarly,

$$P_1(z) = [\ldots + h(-M+1)z + h(1) + h(M+1)z^{-1} + h(2M+1)z^{-2} + \ldots] = \sum_{n=-\infty}^{\infty} h(nM+1)\, z^{-n}$$

In general

$$P_m(z) = \sum_{n=-\infty}^{\infty} h(nM+m)z^{-n}, 0 \text{ to } (M-1)$$

And the system function can now be written

$$H(z) = \sum_{m=0}^{M-1} z^{-m} P_m(z^M)$$

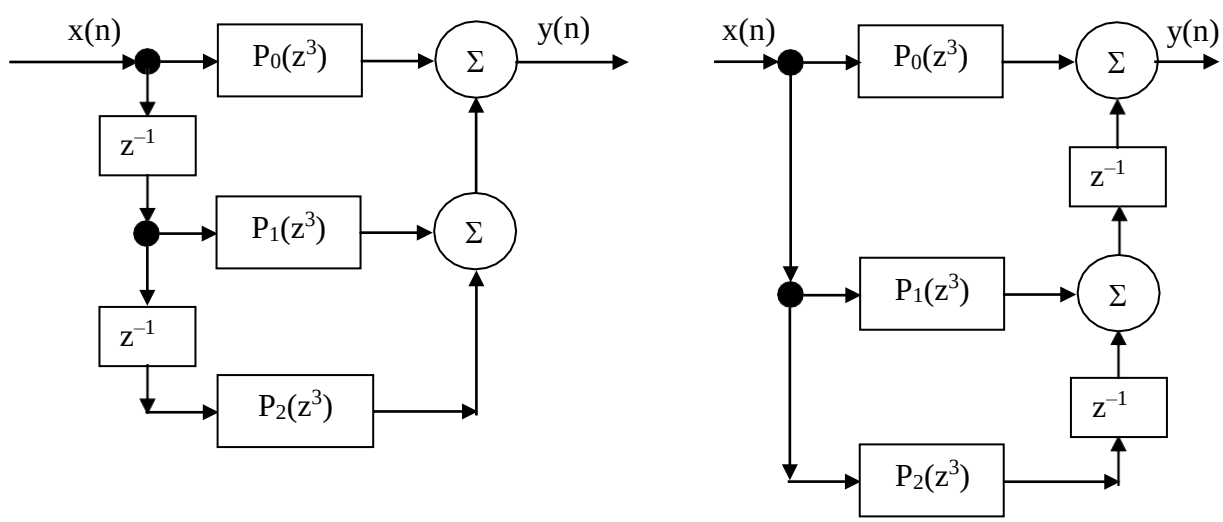
This is called the *M*-component polyphase decomposition of *H(z)*. The *M* functions $P_m(z)$ are the polyphase components of *H(z)*. This overall operation is known as polyphase filtering.

As an example, for $M = 3$, we have

$$\frac{Y(z)}{X(z)} = H(z) = \sum_{m=0}^{2} z^{-m} P_m(z^3) = P_0(z^3) + z_1^{-1} P(z^3) + z_2^{-2} P(z^3)$$

$$Y(z) = P_0(z^3)\, X(z) + z_1^{-1} P(z^3)\, X(z) + z_2^{-2} P(z^3)\, X(z)$$

This last equation leads to the structure below (left):



We may also rearrange the output equation as

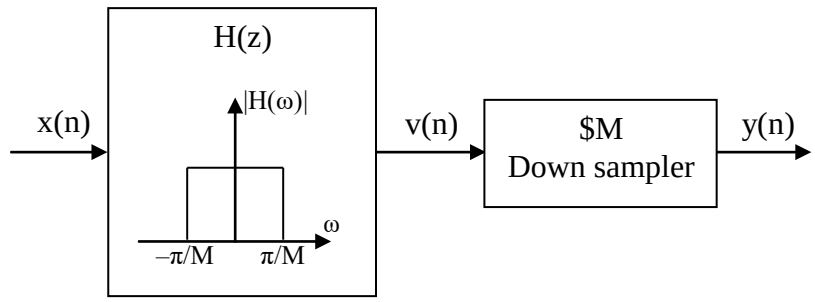
$$Y(z) = P_0(z^3)\, X(z) + z_1^{-1} P(z^3)\, X(z) + z_2^{-2} P(z^3)\, X(z)$$

$$= P_0(z^3)\, X(z) + z_1^{-1} \{ P(z^3)\, X(z) + z_2^{-1} P(z^3)\, X(z) \}$$

This last equation leads to the polyphase structure shown above (right), known as the transpose polyphase structure because it is similar to the transpose FIR filter realization.

Polyphase structure for a decimator

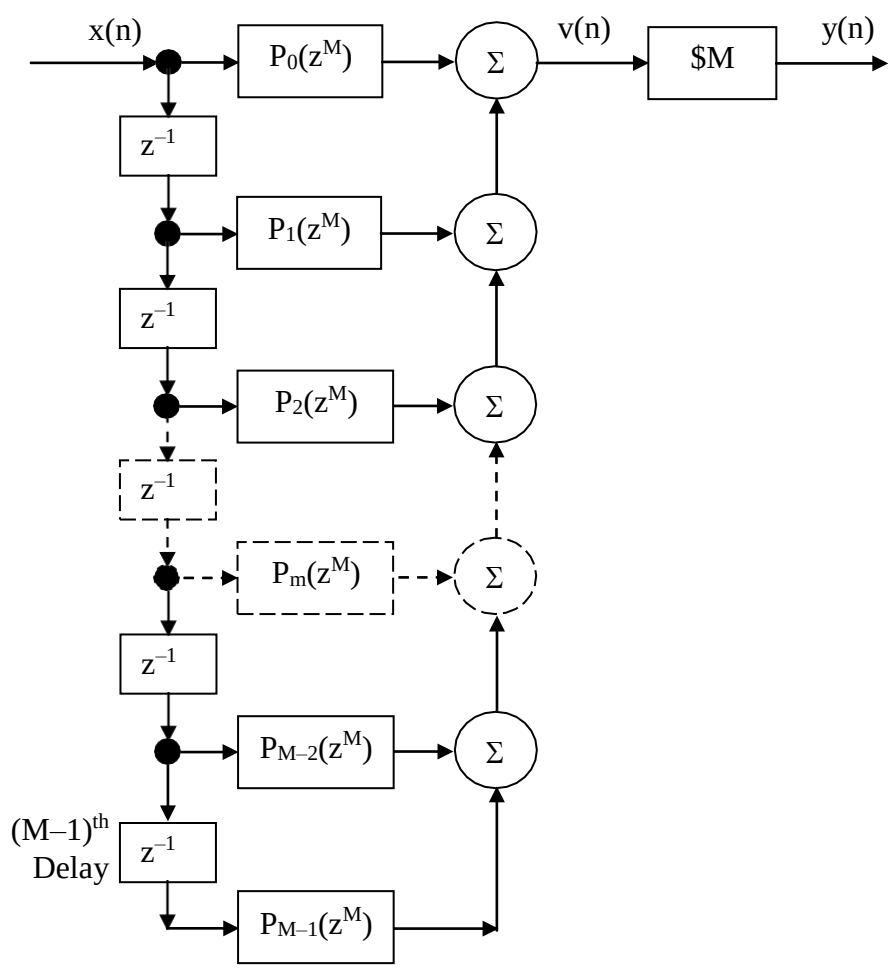
The decimator block diagram is shown below: it consists of an anti-aliasing filter, *H(z)*, which could be an FIR or an IIR filter, followed by an *M*-fold down sampler.



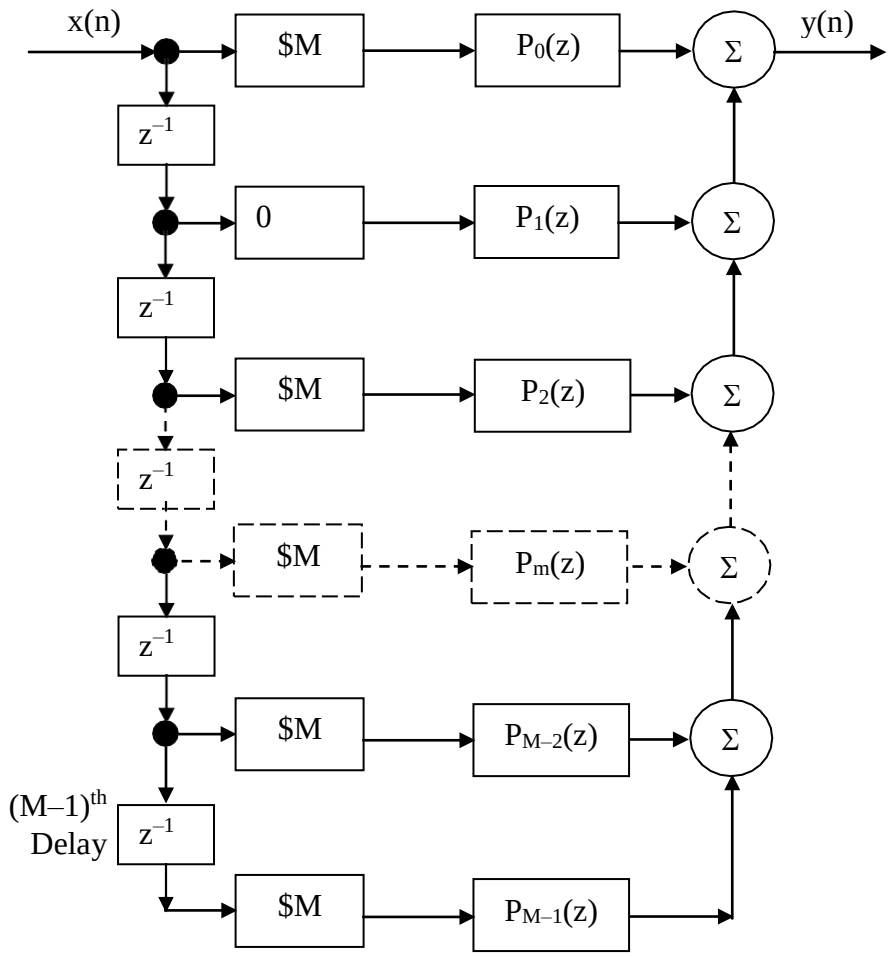
We replace the filter *H(z)* by its *M*-component polyphase decomposition

$$H(z) = \sum_{m=0}^{M-1} z^{-m} P_m(z^M)$$

The sub filters $P_0(z^M)$, $P_1(z^M)$, …, $P_{M-1}(z^M)$ could be FIR or IIR depending on *H(z)*. The block diagram then appears as below.



We may use identity #1 to move the down-sampler to the immediate right of $P_m(z^M)$ in each branch, and then use identity #3 to move the down-sampler from the immediate right to the immediate left of $P_m(.)$ while at the same time changing $P_m(z^M)$ to $P_m(z)$. The result appears as below. It can be seen from this diagram that in this structure the number of multiplications is reduced by a factor of *M*.



****

**** Continuing with the development, comparing

$$P_m(z) = \sum_{r=-\infty}^{\infty} h(rM+m)\, z^{-r}$$

with the defining equation of the z-transform

$$P_m(z) = \sum_{r=-\infty}^{\infty} p_m(r) z^{-r}$$

we identify

$$p_m(r) = h(rM+m)$$

Note that *M* is a constant and *m* is a parameter. Upon substituting for $P_m(z)$ in the system function

$$H(z) = \sum_{m=0}^{M-1} z^{-m} P_m(z^M)$$

we have

$$H(z) = \sum_{m=0}^{M-1} z^{-m} \sum_{r=-\infty}^{\infty} h(rM+m)\left(z^{M}\right)^{-r} = \sum_{m=0}^{M-1}\sum_{r=-\infty}^{\infty} h(rM+m) z^{-(rM+m)}$$

$$= \sum_{m=0}^{M-1}\sum_{r=-\infty}^{\infty} p_m(r)\, z^{-(rM+m)}$$

The output transform is

$$Y(z) = H(z)X(z) = \sum_{m=0}^{M-1}\sum_{r=-\infty}^{\infty} p_m(\mathrm{r})\, X(z)\, z^{-(rM+m)}$$

The output is

$$y(n) = Z^{-1}\left\{\sum_{m=0}^{M-1}\sum_{r=-\infty}^{\infty} p_m(r) X(z) z_{-(rM+m)}\right\} = \sum_{m=0}^{M-1}\sum_{r=-\infty}^{\infty} p_m(r)\, Z_{-1}\left\{X(z)\, z_{-(rM+m)}\right\}$$

$$= \sum_{m=0}^{M-1}\sum_{r=-\infty}^{\infty} p_m(r)\, x(n - rM + m)$$

Define

$$x_m(r) = x(rM - m)$$

Note that M is a constant and m is a parameter.

$$x_m(-r) = x(-rM - m)$$

Shifting the sequence by n units we get

$$x_m(n - r) = x(n - rM - m)$$

The output now may be written

$$y(n) = \sum_{m=0}^{M-1}\sum_{r=-\infty}^{\infty} p_m(r)\, x_m(n - r)$$

Define the operation of polyphase convolution as

$$y(m) = p_m(n) * x_m(n) = \sum_{r=-\infty}^{\infty} p_m(r)\, x_m(n - r)$$

Then

$$y(n) = \sum_{m=0}^{M-1} p_m(n) * x_m(n) = \sum_{m=0}^{M-1} y_m(n)$$

This overall operation is known as polyphase filtering.

UNIT-4 FINITE WORD LENGTH EFFECTS

Finite Wordlength Effects

All the signals and systems are digital in DSP. The digital implementation has finite accuracy. When numbers are represented in digital form, errors are introduced due to their finite accuracy. These errors generate finite precision effects or finite wordlength effects.

Let us consider an example if the first order IIR filter to illustrate how errors are encountered in discretization. Such filter can be described as,

$y(n)=\alpha\, y(n-1)+x(n)$ ..... (1)

The z-transform of above equation gives

$z[y(n)]=\alpha\, z[y(n-1)+x(n)]$

$X(z)=\alpha\, z^{-1}\, Y(z)+X(z)$

Hence the transfer function

$$H(z)=\frac{1}{1-\alpha z^{-1}}=\frac{z}{z-\alpha}$$

Here observe that 'α' is the filter coefficient when this filter is implemented on some DSP processor or software, 'α' can have only discrete values. Let the discrete values of α be represented by α.

Hence the actual transfer function which is implemented is given as,

$$H(z)=\frac{z}{z-\alpha}$$

The transfer function given by above equation is slightly different from H(z). Hence the actual frequency response will be different from desired response.

The input x(n) is obtained by sampling the analog input signal. Since the quantizer takes only fixed(discrete) values of x(n) ,error is introduced. The actual input can be denoted by $x(n)$.

$x(n)=x(n)+e(n)$

Here e(n) is the error introduced during A/D conversion process due to finite wordlength of the quantizer. Similarly error is introduced in the multiplication of α and y(n-1) in equation(1). This is because the product α y(n-1) has to be quantized to one of the available discrete values. This introduces error. These errors generate finite wordlength effects.

Finite Wordlength Effects in IIR Digital Filters

When an IIR filter is implemented in a small system, such as an 8-bit microcomputer, errors arise in representing the filter coefficients and in performing the arithmetic operations indicated by the difference equation. These errors degrade the performance of the filter and in extreme cases lead to instability.

Before implementing an IIR filter, it is important to ascertain the extent to which its performance will be degraded by finite wordlength effects and to find a remedy if the degradation is not acceptable. The effects of these errors can be reduced to acceptable levels by using more bits but this may be at the expense of increased cost.

The main errors in digital IIR filters are:

i. ADC Quantization Noise:

This noise is caused by representing the samples of the input data ,by only a small number of bits.

ii. Coefficient quantization errors:

These errors are caused by representing the IIR filter coefficients by a finite number of bits.

iii. Overflow errors

These errors are caused by the additions or accumulation of partial results in a limited register length.

iv. Product round-offerrors

These errors are caused when the output ,and results of internal arithmetic operations are rounded to the permissible wordlength.

Finite Wordlength Effects in FFT Filters

As in most DSP algorithms, the main errors arising from implementing FFT algorithms using fixed point arithmetic are

i. Round off errors

These errors are produced when the product $W^k B$ is truncated or rounded to the system wordlength.

ii. Overflow errors

These errors result when the output of a butterfly exceeds the permissible wordlength.

iii. Coefficient quantization errors

These errors result from representing the twiddle factors using a limited number of bits.

LIMIT CYCLES

5.2.1 Overflow oscillations

Limit Cycle: The finite wordlength effects are analyzed using the linear model of the digital systems. But nonlinearities are introduced because of quantization of arithmetic operations. Because of these nonlinearities, the stable digital filter under infinite precision may become unstable under finite preision.Because of this instability, oscillating periodic output is generated.Such output is called limit cycle.The limit cycle occur in IIR filters due to feedback paths.

Types of Limit Cycles

There are two types of limit cycles.

(1) Granular and

(2) Overflow.

1. Granular Limit Cycles

The granular limit cycles are of low amplitude. These cycles occur in digital filters when the input signal levels are very low. The granular limit cycles are of two types. They are

i. Inaccessible limit cycles

ii. Accessible limit cycles.

2. Overflow Limit Cycles

Overflow limit cycles occur because of overflow due to addition in digital filters implemented with finite precision. The amplitudes of overflow limit cycles are very large and it can cover complete dynamic range of the register. This further leads to overflow causing cumulative effect. Hence overflow limit cycles are more serious than granular limit cycles.

Transfer Characteristics and Example

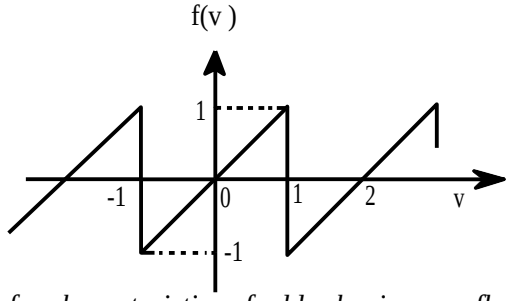


Figure: *Transfer characteristics of adder having overflow limit cycles*

Because of overflow limit cycle oscillations the output fluctuates between minimum and maximum values. The above figure shows the transfer characteristic of an adder that exihibit overflow limit cycle oscillations. Here f(v) indicates addition operation. Consider the addition of following two numbers in sign magnitude form.

$x_1 = 0.111$ i.e., $\frac{7}{8}$ $\qquad$ $x_2 = 0.101$ i.e., $\frac{5}{8}$

Then $x_1 + x_2 = 1.010$ i.e., $-\frac{2}{8}$

Here overflow has occured in addition due to finite precision and the digit before decimal point makes the number negative.

Signal Scaling:

Need for Scaling: Limit cycle oscillations can be avoided by using the nonlinear transfer characteristic. But it introduces distortion in the output. Hence it is better to perform signal scaling such that overflow or underflow does not occur and hence limit cycle oscillations can be avoided.

Implementation of Signal Scaling

Figure shows the direct form-II structure of IIR filter. Let the input $x(n)$ be scaled by a factor s_0 before the summing node to prevent overflow. With the scaling, the transfer function will be,

$$H(z) = s_0 \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} = s_0 \frac{B(z)}{A(z)}$$

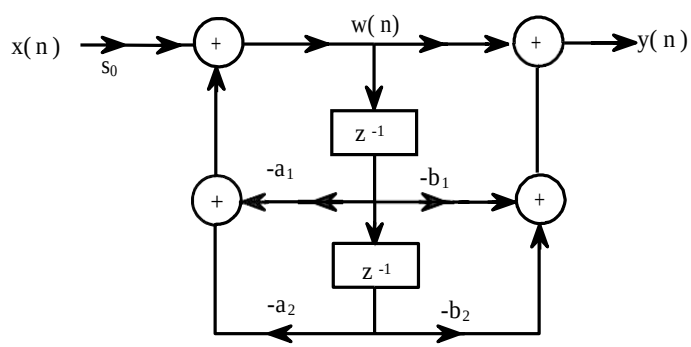


Figure: *Direct form-II realization for second order IIR filter*

Let us define the transfer function

$$H'(z) = \frac{W(z)}{X(z)}$$

From figure we can write above transfer function as,

$$H'(z) = \frac{s_0}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

Since $A(z) = 1 + a_1 z^{-1} + a_2 z^{-2}$

$$H'(z) = \frac{s_0}{A(z)}$$

Since, $H'(z) = \frac{W(z)}{X(z)}$

$$\frac{W(z)}{X(z)} = \frac{s_0}{A(z)}$$

(or) $W(z) = \frac{s_0 X(z)}{A(z)}$

Let $S(z) = \frac{1}{A(z)}$, then above equation becomes,

$$W(z) = s_0 S(z) X(z)$$

Evaluating z-transform on unit circle, we put $z = e^{j\omega}$ in above equation,

$$W(e^{j\omega}) = s_0 S(e^{j\omega}) X(e^{j\omega})$$

Taking inverse Fourier transform of above equation,

$$\omega(n) = \frac{1}{2\pi}\int W(e^{j\omega}) e^{j\omega n} d\omega \qquad = \frac{1}{2\pi}\int s_0 S(e^{j\omega}) X(e^{j\omega}) e^{j\omega n} d\omega$$

Hence $\omega^2(n) = \frac{s_0^2}{4\pi^2}\left|\int S(e^{j\omega}) X(e^{j\omega}) e^{j\omega n} d\omega\right|^2$

Schwartz inequality states that,

$$\left|\int x_1(t) x_2(t) dt\right|^2 \le \int |x_1(t)|^2 dt \int |x_2(t)|^2 dt$$

Using this relation we can write above equation as,

$$\omega^2(n) \le \frac{s_0^2}{4\pi^2}\left[\int |S(e^{j\omega})|^2 d\omega . \int |X(e^{j\omega})|^2 d\omega\right],$$

since, $|e^{j\omega n}|^2 = 1$

$$\therefore \omega^2(n) \le s_0^2 \left[\frac{1}{2\pi}\int |S(e^{j\omega})|^2 d\omega . \frac{1}{2\pi}\int |X(e^{j\omega})|^2 d\omega\right]$$

Parseval's theorem states that $\frac{1}{2\pi}\int |X(e^{j\omega})|^2 = \sum_{n=0}^{\infty} x^2(n)$. Then above equation can be written as,

$$\omega^2(n) \le s_0^2 \left[\frac{1}{2\pi}\int |S(e^{j\omega})|^2 d\omega . \sum_{n=0}^{\infty} x^2(n)\right]$$

We have put $z = e^{j\omega}$. Hence $dz = je^{j\omega}d\omega$ or

$$d\omega = \frac{dz}{je^{j\omega}} = \frac{dz}{jz}, \text{ since } e^{j\omega} = z$$

Putting these values in equation

$$\omega^2(n) \le s_0^2 \left[\frac{1}{2\pi} \int \left|S(e^{j\omega})\right|^2 \cdot \frac{dz}{jz} \cdot \sum_{n=0}^{\infty} x^2(n) \right]$$

$$\le s_0^2 \sum_{n=0}^{\infty} x^2(n) \cdot \frac{1}{2\pi j} \int \left|S(z)\right|^2 \frac{}{z}$$

Here $|S(z)|^2 = S(z).S(z^{-1})$. Then we have,

$$\omega^2(n) \le s_0^2 \sum_{n=0}^{\infty} x^2(n) \cdot \frac{1}{2\pi j} \int S(z).S(z^{-1}) z^{-1} dz$$

Here the integration is executed over a closed contour i.e. $\omega^2(n) \le s_0^2 \sum_{n=0}^{\infty} x^2(n) \cdot \frac{1}{2\pi j} \oint_C S(z).S(z^{-1}) z^{-1} dz$

(or)

$$\omega^2(n) \le \sum_{n=0}^{\infty} x^2(n) \left[s_0^2 \oint_C S(z).S(z^{-1}) z^{-1} dz \right]$$

Here $\omega^2(n)$ represents instantaneous energy of signal after first summing node. And $x^2(n)$ represents instantaneous energy of input signal. Overflow will not occur if

$$\omega^2(n) \le \sum_{n=0}^{\infty} x^2(n)$$

For this equation to be true we get following condition from equation

$$s_0^2 \frac{1}{2\pi j} \oint_C S(z).S(z^{-1}) z^{-1} dz = 1$$

Earlier we have defined $S(z) = \frac{1}{A(z)}$. Hence above condition becomes,

$$s_0^2 \frac{1}{2\pi j} \oint_C \frac{z^{-1} dz}{A(z).A(z^{-1})} = 1$$

(or)

$$s_0^2 = \frac{1}{\frac{1}{2\pi j} \oint_C \frac{z^{-1} dz}{A(z).A(z^{-1})}}$$

Above equation gives the value of scaling factor s_0 to avoid overflow

5.3 ROUND OFF NOISE IN IIR DIGITAL FILTERS

Statistical Model for Analysis of Round-off Error Multiplication:

We perform arithmetic operations like addition and multiplication some errors will be occured. Those errors are called arithmetic errors. The results of arithmetic operations are required to be quantized so that they can occupy one of the finite set of digital levels. Such operation can be visualized as multiplier (or other arithmetic operation) with quantizer at its output.

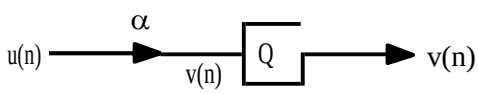


Figure: *Quantization of multiplication or product*

The above process can be represented by a statistical model for error analysis. The output υ (n) and error $e_\alpha(n)$ in product quantization process.i.e.,

$$\upsilon(n) = \upsilon(n) + e_\alpha(n)$$

The statistical model is shown below.

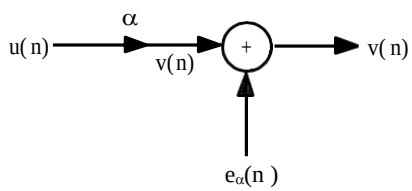


Figure: *Statistical model for analysis of round-off error multiplication*

For the analysis purpose following assumptions are made.

) The error sequence $\{e_{\alpha(n)}\}$ tihe sample sequence of a stationary white noise process.

ii $e_\alpha(n)$ is having uniform distribution over the range of quantizationerror.

iii) The sequence $\{e_\alpha(n)\}$ is uncorrelated with the sequence $\upsilon(n)$ and input sequence x(n).

Computational output round off noise

Product Round-off Errors and its Reduction:

The results of product or multiplication operations are quantized to fit into the finite wordlength,when the digital filters are implemented using fixed point arithmetic. Hence errors generated in such operation are called product round off errors.

The effect of product Round-off errors can be analyzed using the statistical model of the quantization process. The noise due to product round-off errors reduces the signal to noise ratio at the output of the filter. Some times this ratio may be reduces below acceptable levels. Hence it is necessary to reduce the effects of product round-off errors.

There are two solutions available to reduce product round-off errors.

a) Error feedback structures and

b)State space structure.

The error feedback structures use the difference between the unquantized and quantized signal to reduce the round-off noise. The difference between unquantized and quantized signal is fed back to the digital filter structure in such a way that output noise power due to round-off errors is reduced.

First Order Error-feedback Structure to reduce Round-off Error:

The results of product or multiplication operations are quantized to fit into the finite wordlength, when the digital filters are implemented using fixed point arithmetic. Hence errors generated in such operation are called product round off errors.

The effect of product round-off errors can be analyzed using the statistical model of the quantization process.

Let the quantization error signal be given as the difference between unquantized signal y(n) and quantized signal $\upsilon(n)$ .i.e.,

$$e(n) = y(n) - \upsilon(n)$$

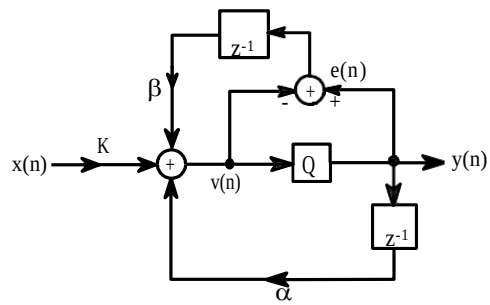


Figure: *First order error feedback structure to reduce product round-off error*

This error signal is fed back in the structure such that round-off noise is reduced. Such structure for first order digital filter is shown in figure.

The incorporation of quantization error feedback as shown in figure helps in reducing the noise power at the output . This statement can be proved mathematically.

Round-off Errors in FFT Algorithms:

FFT is used in large number of applications. Hence it is necessary to analyze the effects due to finite wordlengths in FFT algorithms. The most critical error in FFT computation occurs due to arithmetic round-off errors.

The DFT is used in large number of applications such as filtering, correlation, spectrum analysis etc. In such applications DFT is computed with the help of FFT algorithms. Therefore it is important to study the quantization errors in FFT algorithms. These quantization effects mainly take place because of round-off errors. These errors are introduced when multiplications are performed in fixed point arithmetic.

FFT algorithms require less number of multiplications compared to direct computation of DFT. But it does not mean that quantization errors are also reduced in FFT algorithms.

Let σ_x^2 represents the variance of output DFT coefficients i.e., $|X(k)|$.

For N-point DFT σ_x is given as,

$$\sigma_x^2 = \frac{1}{3N} \qquad .....(1)$$

For direct computation of DFT, the variance of quantization errors in multiplications is given as,

$$\sigma_q^2 = \frac{N}{3}.\Delta^2 \qquad .....(2)$$

Here σ_q^2 is variance of quantization errors and Δ is step size which is given as,

$$\Delta = 2^{-b} \qquad .....(3)$$

And b is the number of bits to represent one level.

Hence equation (2) becomes,

$$\sigma_q^2 = \frac{N}{3}.2^{-2b} \qquad .....(4)$$

The signal to noise power ratio at the output (i.e., DFT coefficients) can be considered as the measure of quantization errors. This ratio is the ratio of variance of DFT coefficients $\left(\sigma_x^2\right)$ to the variance of quantization errors $\left(\sigma_q^2\right)$ i.e.,

Signal to noise ratio in direct computation of $\text{DFT} = \left(\frac{\sigma_x^2}{\sigma_q^2}\right)_{\text{Direct DFT}}$

From equation (1) and equation (2) we have,

$$\left(\frac{\sigma_x^2}{\sigma_q^2}\right)_{\text{Direct DFT}} = \frac{\frac{1}{3N}}{\frac{N}{3}.2^{-2b}} = \frac{2^{2b}}{N^2} \quad - .....(5)$$

When DFT is computed using FFT algorithms, the variance of the signal remains same i.e.,

$$\sigma_x^2 = \frac{1}{3N} \text{ from equation (1)} ..... (6)$$

But with algorithms the variance of the quantization errors is given as,

$$\sigma_q^2 = \frac{2}{3}.2^{-2b} \qquad .....(7)$$

Hence signal to noise ration in FFT algorithms is,

$$\left(\frac{\sigma_x^2}{\sigma_q^2}\right)_{\text{FFT}} = \frac{\frac{1}{3N}}{\frac{2}{3}.2^{-2b}} = \frac{2^{2b}}{2N}$$

In the above expression, the signal to noise ration is inversely proportional to N. whereas in direct DFT computation the signal to noise ratio is inversely proportional to N^2 as given by equation (5). This means quantization errors increase fast with increase in 'N' in direct computation of DFT. But in FFT algorithms the quantization errors increase slowly with increase in 'N'.

Product of Round-off Errors in IIR Digital Filters:

The results of product or multiplication operations are quantized to fit into the finite wordlength, when the digital filters are implemented using fixed point arithmetic. Hence errors generated in such operation are called product round off errors.

Product round-off error analysis is an extensive topic.Our presentation here will be brief and aims to make you aware of the nature of the errors, their effects and how to reduce them if necessary.

The basic operations in IIR filtering are defined by the familiar second- order difference equation:

$$y(n) = \sum_{k=0}^{2} b_k x(n-k) - \sum_{k=1}^{2} a_k y(n-k)$$

Where x(n-k) and y(n-k) are the input and output data samples,and b_k and a_k are the filter coefficients. In practice these variables are often represented as fixed point numbers. Typically , each of the products b_k x(n-k)

and a_k y(n-k) would require more bits to represent than any of the operands. For example, the product of a B-bit data and a B-bit coefficient is 2B bits long.

Truncation or rounding is used to quantize the products back to the permissible wordlength. Quantizing the products leads to errors,popularly known as round-off errors,in the output data and hence a reduction in the SNR. These errors can also lead to small-scale oscillations in the output of the digital filter,even when there is no input to the filter.

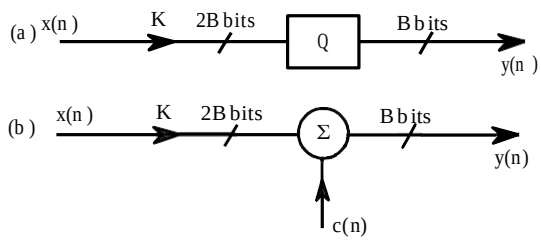


Figure: *Representation of the product quantization error: (a) a block diagram representation of the quantization process; (b) a linear model of the quantization process*

The figure(a) represents a block diagram of the product quantization process,and figure (b) represents a linear model of the effect of product quantization. The model consists of an ideal multiplier,witk infinite precision, in series with an adder fed by a noise sample, e(n), representing the error in the quantized product ,where we have assumed,for simplicity,that x(n),y(n), and K are each represented by B bits. Thus

y(n) = Kx(n) + e(n)

The noise power, due to each product quantization, is given by

$$\sigma_r^2 = \frac{q^2}{12}$$

Where r symbolizes the round-off error and q is the quantization step defined by the wordlength to which product is quantized. The round-off noise is assumed to be a random variable with zero mean and constant variance. Although this assumption may not always be valid, it is useful in assesing the performance of the filter.

Product of Round-off Errors on Filter Performance:

The effects of round-off errors on filter performance depend on the type of filter structure used and the point at which the results are quantized.

The above figure represents the quantization noise model for the direct form building block. It is assumed in the figure that the input data,x(n),output data,y(n),and the filter coefficients are represented as B-bit numbers (including the sign bit). The products are quantized back to B bits after multiplication by rounding (or truncation).

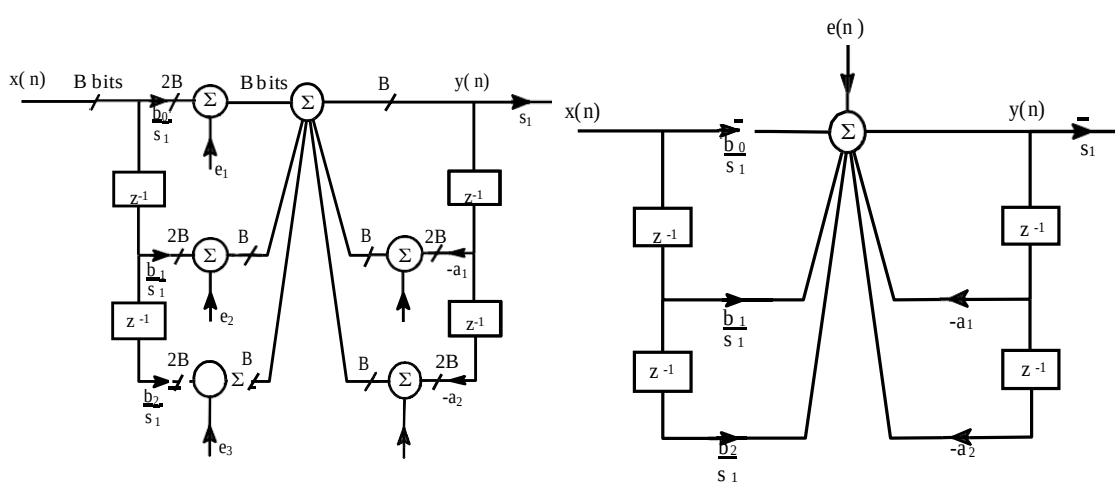


Figure: *Product quantization noise model for the direct form filter section. All the noise sources in (a) have*

been combined in (b) as they feed to the same point

Since all five noise sources, e_1 to e_5 in figure(a),feed to the same point (that is into the middle adder), the total output noise power is the sum of the individual noise powers(figure(b)).

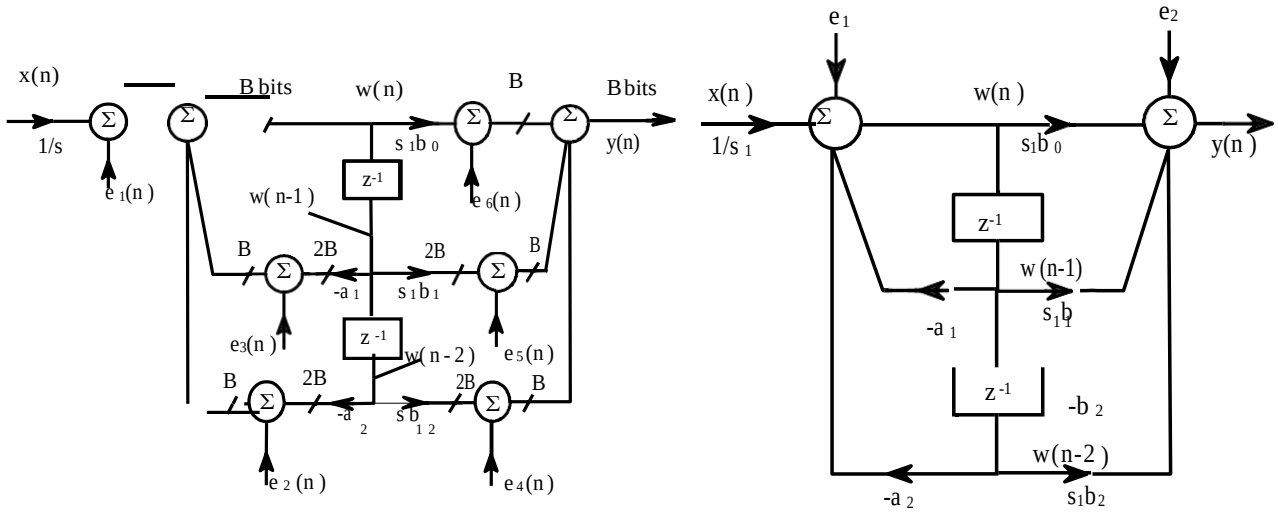


Figure: *Product quantization noise model for the canonic filter section. The noise sources feeding the same point in (a) have been combined in (b)*

$$\sigma_{or}^2 = \frac{5q^2}{12}\left[\frac{1}{2\pi j}\oint_c F(z)F(z^{-1})\frac{dz}{z}\right]s_1^2 = \frac{5q^2}{12}\left[\sum_{k=0}^{\infty} f^2(k)\, s_1^2\right] = \frac{5q^2}{12}\|F(z)\|_2^2 s_1^2$$

Where $F(z) = \dfrac{1}{1+a_1 z^{-1}+a_2 z^{-2}}$

$f(k) = Z^{-1}[F(z)]$ the inverse z-transform of F(z),which is also the impulse response from each noise source to the filter output, $\|\ \|_2^2$ is the L_2 norm squared and $\frac{q^2}{12}$ is the intrinsic product round-off noise power. The total noise power at the filter output is the sum of the product round-off noise and the ADC quantization noise.

$$\sigma_0^2 = \sigma_{0A}^2 + \sigma_{or}^2$$

$$= \frac{q^2}{12}\left[\sum_{k=0}^{\infty} h^2(k) + 5s_1^2\sum_{k=0}^{\infty} f^2(k)\right] = \frac{q^2}{12}\left[\|H(z)\|_2^2 + 5s_1^2\|F(z)\|_2^2\right]$$

For canonic section, figure(a) , the noise model again includes a scale factor as this generates a round-off error of its own. The noise sources $e_1(n)$ to $e_3(n)$ all feed to the left adder, whilst the noise sources $e_4(n)$ to $e_6(n)$ feed directly into the filter output. Combination of the noise sources feeding to the same point leads to the noise model of figure(b). Assuming uncorrelated noise sources, the total noise contribution is simply the sum of the individual noise contributions:

$$\sigma_{or}^2 = \frac{3q^2}{12}\sum_{k=0}^{\infty} f^2(k) + \frac{3q^2}{12} = \frac{3q^2}{12}\left[\|F(z)\|_2^2 + 1\right]$$

Where f(k) is the impulse response from the noise source e_1 to the filter output, and F(z) the corresponding transfer function given by

$$F(z) = s_1 \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} = s_1 H(z)$$

The total noise (ADC+round-off noises) at the filter output is given by

$$\sigma_0^2 = \sigma_{0A}^2 + \sigma_{or}^2$$

$$= \frac{q^2}{12}\left\{3\left[1 + s_1^2 \sum_{k=0}^{\infty} h^2(k)\right] + \sum_{k=0}^{\infty} h^2(k)\right\} = \frac{q^2}{12}\left\{3\left[1 + s_1^2 \|H(z)\|_2^2\right] + \|H(z)\|_2^2\right\}$$

HOMEWORK TO PREVENT OVERFLOW

Prevent Overflow Limit Cycle Oscillations: The overflow limit cycles occur because of overflow due to addition in digital filters implemented with finite precision. The amplitudes of overflow limit cycles are very large and it can cover complete dynamic range of the register.

The specific design of filter coefficients do not assure prevention of overflow limit cycle oscillations. The transfer characteristic can be modified to avoid overflow limit cycle oscillations.

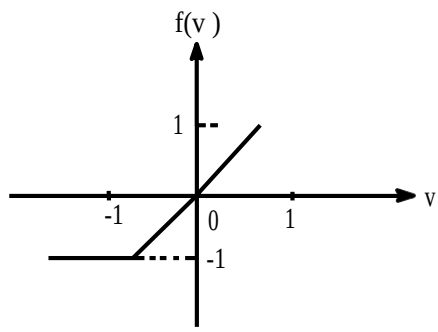


Figure: *Prevention of overflow limit cycle oscillations*

As shown in figure when an overflow or underflow is sensed, the output of the adder is set to its full scale value of ± 1. This prevents oscillatory output. This nonlinearity of the characteristic causes very small distortion in the output because overflow/underflow occurs rarely.

Scaling is also used to prevent overflow limit cycle oscillations. Limit cycle free structures are normally used to avoid the effects of limit cycles.

Characteristics of a Limit Cycle Oscillation with respect to the System by the following Difference equation

$y(n) = 0.95\, y(n-1) + x(n)$ **.**

Let $y_1(n)$ be the output of the system after the product term $0.95\, y(n-1)$ is quantized after rounding. i.e.,

$$y_1(n) = Q\left[0.95 y(n-1)\right] + x(n)$$

Let $$x(n) = \begin{cases} 0.75 & \text{for } n = 0 \\ 0 & \text{for } n \neq 0 \end{cases}$$

Let $b = 4$ bits are used to represent the quantized product excluding sign bit.

With n=0

$y_r(n) = Q_r[0.95y_r(n-1)] + x(n)$

$\therefore\ y_r(0) = Q_r[0.95y_r(-1)] + x(0) = Q_r[0.95 \times 0] + 0.75$

Since $y_r(-1) = 0 = 0.75$

$[0.75]_{10} = [0.11]_2$

$\therefore$ 4-bits rounded value of $[0.11]_2$ will be $[0.1100]_2$ i.e., 0.75 only.

$\therefore\ y_r(0) = 0.75$ after 4 bits rounding

With n = 1

$y_r(1) = Q_r[0.95y_r(0)] + x(1) = Q_r[0.95 \times 0.75] + 0 = Q_r[0.7125]$

$[0.7125]_{10} = [0.1\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0 \cdots]_2$

Note that 0.7125 requires infinite binary digits for its representation. Let us round it to 4 bits.

$\therefore\ Q_r[0.7125]_{10} = [0.1\ 0\ 1\ 1]_2$ upto 4 bits

But decimal equivalent of $[0.1\ 0\ 1\ 1]_2$ is 0.6875.

$\therefore\ y_r(1) = 0.6875$

This means the actual value of $y_r(1) = 0.7125$ is changes to 0.6875 due to 4-bits quantization.

With n = 2

$y_r(2) = Q_r[0.95y_r(1)] + x(2) = Q_r[0.95 \times 0.6875] + 0 = Q_r[0.653125]$

$[0.653125]_{10} = [0.1\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0 \cdots]$

$\therefore\ Q_r[0.653125]_{10} = [0.1\ 0\ 1\ 0]_2$ upto 4 bits $= [0.625]_{10}$

$\therefore\ y_r(2) = 0.625$

With n = 3

$y_r(3) = Q_r[0.95y_r(2)] + x(3) = Q_r[0.95 \times 0.625] + 0 = Q_r[0.59375]$

$[0.59375]_{10} = [0.1\ 0\ 1\ 0\ 1]_2$

$\therefore\ Q_r[0.59375]_{10} = [0.1\ 0\ 1\ 1]_2$ upto 4 bits $= 0.625$

$\therefore\ y_r(3) = 0.625$

Thus $y_r(2) = y_r(3) = \square = 0.625$

Thus the system enters into limit oscillation when $n = 2$.

To calculate dead band

Consider equation

$$y(n-1) \le \frac{6/2}{1-|a|}$$

Here $\delta = \frac{1}{2^b} = \frac{1}{2^4} = 0.0625$

$$y(n-1) \le \frac{0.0625/2}{1-0.95} \le 0.625$$

Dead band = [-0.625, +0.625]

Signal Scaling to Prevent Limit Cycle Oscillations: This is zero input condition. Following table lists the values of $y(n)$ before and after quantization. Here the values are rounded to nearest integer value.

n	$y(n)$ before quantization	$y(n)$ after quantization
-1	12	12
0	10.8	11
1	9.72	10
2	8.748	9
3	7.8732	8
4	7.08588	7
5	6.377292	6
6	5.7395628	6
7	5.1656065	5
8	4.6490459	5

Table: *Values of* $y(n)$ *before and after quantization*

From table observe that if $|y(-1)| \le 5$, $y(n) = y(-1)$ for $n \ge 0$ for zero input. Hence the dead band will be $[-5,5]$.

Since the values are rounded to nearest integer after quantization, the step size will be $\delta = 1$. Hence dead band can also be calculated as follows:

$$y(n-1) = \frac{\delta/2}{1-\alpha}, \text{Here } \alpha = 0.9 \text{ , } y(n-1) = \frac{1/2}{1-0.9} = 5$$

Thus the dead band is $[-5,5]$.

Dynamic Range Scaling to Prevent the Effects of Overflow: The overflow can take place at some internal nodes when the digital filters are implemented by using fixed point arithmetic. Such nodes can be inputs/outputs of address or multipliers. This overflow can take place even if the inputs are scaled. Because of such overflow at intermediate points,produces totally undesired output or oscillations. The overflow can be avoided by scaling the internal signal levels with the help of scaling multipliers. These scaling multipliers are inserted at the apprppriate points in the filter structure to avoid possibilities of overflow. Sometimes these scaling multipliers are absorbed with the existing multipliers in the structure to reduce the total number and complexity.

At which node the overflow will take place is not known in advance. This is because the overflow depends upon type of input signal samples. Hence whenever overflow takes place at some node, the scaling should be done dynamically. Hence dynamic range scaling in the filter structure can avoid the effects of overflow.

Let $u_r(n)$ be the signal sample at r^{th} node in the structure. Then the scaling should ensure that,

$$|u_r(n)| \le 1 \qquad \text{for all r and n.}$$

RADTE OFF BETWEEN ROUND-OFF AND OVERFLOW NOISE, MEASUREMENT OF COEFFICIENT QUANTIZATION EFFECTS THROUGH POLE-ZERO MOVEMENT

Errors in Rounding and Truncation Operations : The computations like multiplication or addition are performed the result is truncated or rounded to nearest available digital level. This operation introduces an error. Hence the performance of the system is changed from expected value.

Truncation Error: This error is introduced whenever the number is represented by reduced number of bits.

Let Q_t(x) be the value after truncation ,then truncation error will be,

$$\varepsilon_r = Q_t(x) - (x)$$

Here x is the original value of the number.

Rounding Error : This error is introduced whenever the number is rounded off to the nearest digital level.The number of bits used to represent the rounded number are generally less than the number of bits required for actual number.

Let Q_r(x) be the value after rounding.Then rounding error will be,

$$\varepsilon_r = Q_r(x) - x$$

Here x is the original value of a number.

Tradeoff between roundoff and overflow noise:

Scaling operation

Scaling is a process of readjusting certain internal gain parameters in order to constrain internal signals to a range appropriate to the hardware with the constraint that the transfer function from input to output should not be changes.

The filter in figure with unscaled node x has the transfer function

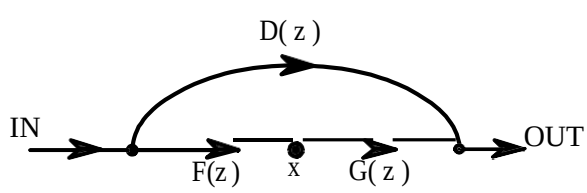
H(z) = D(z)+F(z)G(z) .....(1)

To scale the node x, we divide F(z) by some number β and multiply G(z) by the same number as in figure. Although the transfer function does not change by this operation, the signal level at node x has been changes. The scaling parameter β can be chosen to meet any specific scaling rule such as

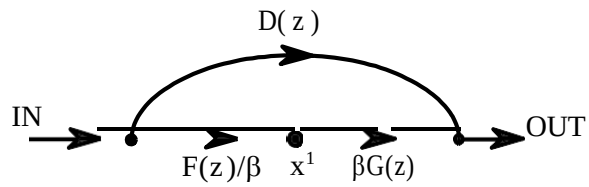
$$l_1 \text{scaling}: \beta = \sum_{i=0}^{\infty} \lfloor f(i) \rfloor \qquad .....(2)$$

$$l_2 \text{ scaling}: \beta = \delta \sqrt{\sum_{i=0}^{\infty} \lfloor f^2(i) \rfloor} \qquad .....(3)$$

Where f(i) is the unit-sample response from input to the node x and the parameter δ can be interpreted to represent the number of standard deviations



(a)



(b)

Figure: *A filter with unscaled node x and (b) A filter with scaled node x'*

Representable in the register at node x if the input is unit-variance white noise. If the input is bound by $|u(n)| \le 1$, then,

$$|x(n)| = \left|\sum_{i=0}^{\infty} f(i)u(n-i)\right| \le \sum_{i=0}^{\infty} |f(i)| \qquad .....(4)$$

Equation represents the true bound on the range of x and overflow is completely avoided by l_1 scaling in (2), which is the most stringent scaling policy.

In many cases, input can be assumed to be white noise. Although we cannot compute the variance at node x. for unit-variance white noise input,

$$E\left[x^2(n)\right] = \sum_{i=0}^{\infty} f^2(i) \qquad .....(5)$$

Since most input signals can be assumed to be white noise, l_2 scaling is commonly used. In addition, (5) can be easily computed. Since (5) is the variance (not a strict bound), there is a possibility of overflow, which can be reduced by increasing δ in (3). For large values of δ , the internal variables are scaled conservatively so that no overflow occurs. However, there is a trade –off between overflow and roundoff noise, since increasing δ deteriorates the output SNR (signal to noise ratio).

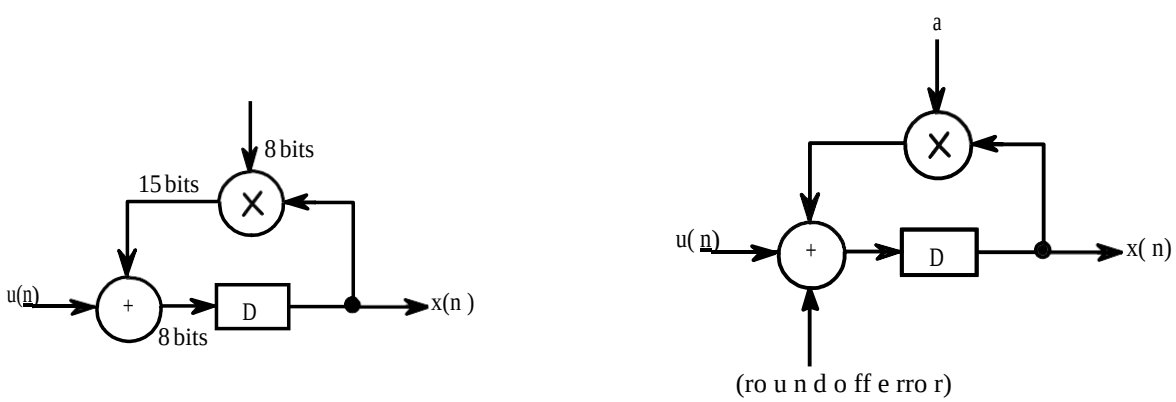


Figure: *Model of roundoff error*

Roundoff Noise: If two W-bit fixed point fraction numbers are multiplies together, the product is (2W-1) bit long. This product must eventually be quantized to W-bits by rounding or truncation. For example, consider the 1st –order IIR filter shown in figure. Assume that the input wordlength is W=8bits. If the multiplier coefficient wordlength is also the same, then to maintain full precision in the output we need to increase the output wordlength by 8 bits per iterations. This is clearly infeasible. The alternative is to roundoff (or truncate) the output to its nearest 8-bit representation.

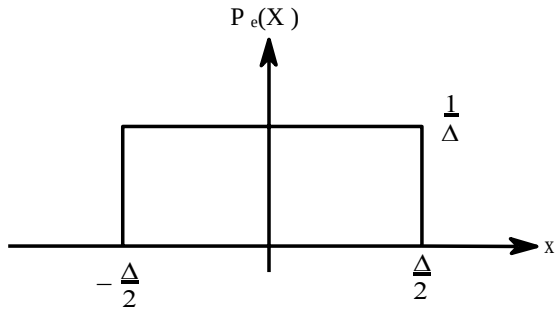


Figure: *Error probability distribution*

The result of such quantization introduces roundoff noise e(n). For mathematical ease a system with roundoff can be modeled as an infinite precision system with an external error input. For example in the previous case (shown in figure) we round off the output of the multiply add operation and an equivalent model is shown in figure.

Although rounding is not a linear operation, its effect at the output can be analyzed using linear system theory with the following assumptions about e(n):

1. E(n) is uniformly distributed white noise.
2. E(n) is a wide –sense stationary random process, i.e., mean and covariance of e(n) are independent of the time index n.
3. E(n) is uncorrelated to all other signals such as input and other noise signals.

Let the wordlength of the output be W-bits, then the roundoff error e(n) can be given by

$$\frac{-2^{-(w-1)}}{2} \le e(n) \le \frac{2^{-(w-1)}}{2} \qquad .....(6)$$

Since the error is assumed to be uniformly distributed over the interval given in (6), the corresponding probability distribution is shown in figure, where Δ is the length of the interval (i.e., $2^{-(w-1)}$).

Let us compute the mean E[e(n)] and variance $E\left[e^2(n)\right]$ of this error function.

$$E[e(n)] = \int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} x P_e(x)dx = \left[\frac{1}{\Delta}\frac{x^2}{2}\right]_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} = 0 \qquad .....(7)$$

Note that since mean is zero, variance is simply $E\left[e^2(n)\right]$

$$E\left[e^2(n)\right] = \int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} x^2 P_e(x)dx = \left[\frac{1}{\Delta}\frac{x^3}{3}\right]_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} = \frac{\Delta^2}{12} = \frac{2^{-2w}}{3} \qquad .....(8)$$

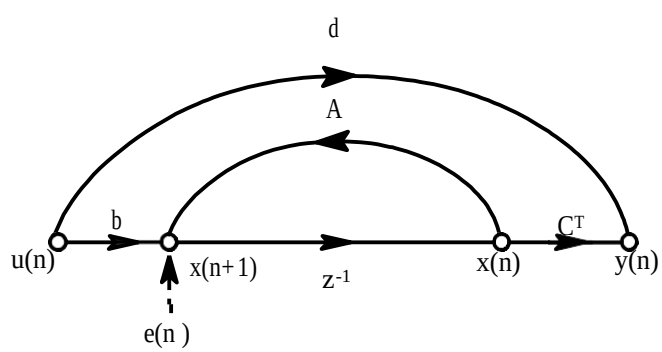


Figure: *Signal flow graph*

In other words (8) can be rewritten as

$$\sigma_e^2 = \frac{2^{-2w}}{3} \qquad .....(9)$$

Where σ_e^2 is the variance of the roundoff error in a finite precision, W-bit wordlength system. Since the variance is proportional to 2^{-2w}, increase in wordlength by 1 bit decreases the error by a factor of 4.

The purpose of analyzing roundoff noise is to determine its effect at the output signal. If the noise variance at the output is not negligible in comparison to the output signal level, the worlength should be increase or some low noise structures should be used. Therefore, we need to compute SNR at the output, not just the noise gain to the output. In the noise analysis, we use a double length accumulator model, which means rounding is performed after two (2w-1)-bit products are added. Also, notice that multipliers are the sources for roundoff noise.

Effects of Coefficient Quantization in FIR filters: Let us consider the effects of coefficient quantization in FIR filters. Consider the transfer function of the FIR filter of length M,

$$H(z) = \sum_{n=0}^{M-1} h(n)z^{-n}$$

The quantization of h(n) takes place during implementation of filter. Let the quantized coefficeints be denoted by $h(n)$ and e(n) be the error in quantization. Then we can write,

$$h(n) = h(n)+e(n)$$

And the new filter transfer function becomes,

$$H(z) = \sum_{n=0}^{M-1} h(n)z^{-n} = \sum_{n=0}^{M-1}\left[h(n) + e(n)\right]z^{-n} = \sum_{n=0}^{M-1} h(n)z^{-n} + \sum_{n=0}^{M-1} e(n)z^{-n} = H(z)+E(z)$$

Where, $E(z)=\sum_{n=0}^{M-1} e(n)z^{-n}$

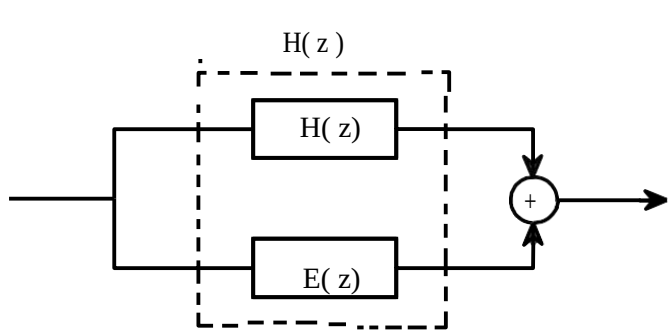


Figure: *Model of FIR filter with quantizer coefficients*

Here observe that the FIR filter with quantized coefficients can be modelled as parallel connection of two FIR filters H(z) and E(z).

In the figure, H(z) is the FIR filter with unquantized coefficients, and E(z) is the FIR filter representing coefficient quantization error.

The frequency response of FIR filter with quantized coefficients as,

$$H(\omega) = H(\omega) + E(\omega)$$

Here $E(\omega)$ is the error in the desired frequency response which is given as,

$$E(\omega) = \sum_{n=0}^{M-1} e(n)e^{-j\omega n}$$

Consider the magnitude of error i.e.,

$$|E(\omega)| = \left|\sum_{n=0}^{M-1} e(n)e^{-j\omega n}\right| \Rightarrow |E(\omega)| \le \sum_{n=0}^{M-1} |e(n)\, e^{-j\omega n}| \Rightarrow \le \sum_{n=0}^{M-1} |e(n)| \quad (\therefore |e^{-j\omega n}| = 1)$$

The upper bound is reached if all the errors have same sign and have the maximum value in the range. If we consider e(n) to be statistically independent random variables, then more realistic bound is given by standard derivation of E(ω) i.e.;

$\sigma_E(\omega)$ is the standard derivation of error in frequency response i.e., E(ω).

DEADBAND EFFECTS

Deadband and Deadband of First Order Filter: Dead band is the range of output amplitudes over which limit cycle oscillations takeplace

Dead band of first order filter

Consider the first order filter,

$$y(n) = \alpha y(n-1) + x(n)$$

Here $\alpha\, y(n-1)$ is the product term. After rounding it to 'b' bits we get,

$$y(n) = Q\left[\alpha y(n-1)\right] + x(n)$$

When limit cycle oscillations take place,

$$Q_r\left[\alpha y(n-1)\right] = \pm y(n-1) \qquad \text{…..(1)}$$

The error due to rounding is less than $\frac{\delta}{2}$. Hence,

$$\left|Q\left[\alpha y(n-1) - \alpha y(n-1)\right]\right| \le \frac{\delta}{2}$$

From equation (1) above equation can be written as,

$$\left|\pm y(n-1) - \alpha y(n-1)\right| \le \frac{\delta}{2}$$

$$\therefore\ y(n-1)\left[1 \mp \alpha\right] \le \frac{\delta}{2}$$

$$\therefore\ y(n-1) \le \frac{\delta/2}{1-|\alpha|}$$

UNIT-4

Finite Word length Effects

Introduction

Practical digital filters must be implemented with finite precision numbers and arithmetic. As a result, both the filter coefficients and the filter input and output signals are in discrete form. This leads to four types of finite word length effects.

Discretization (quantization) of the filter coefficients has the effect of perturbing the location of the filter poles and zeroes. As a result, the actual filter response differs slightly from the ideal response. This *deterministic* frequency response error is referred to as **coefficient quantization error**.

The use of finite precision arithmetic makes it necessary to quantize filter calculations by rounding or truncation. **Roundoffnoise**is thaterror in thefilter output that resultsfrom roundingor truncating calculations within the filter. As the name implies, this error looks like low-level noise at the filter output.

Quantization of the filter calculations also renders the filter slightly nonlinear. For large signals this nonlinearity is negligible and roundoff noise is the major concern. However, for recursive filters with a zero or constant input, this nonlinearity can cause spurious oscillations called **limit cycles**.

With fixed-point arithmetic it is possible for filter calculations to overflow. The term **overflow oscillation**, sometimes also called **adder overflow limit cycle**, refers to a high-level oscillation that can exist in an otherwise stable filter due to the nonlinearity associated with the overflow of internal filter calculations.

In this chapter, we examine each of these finite word length effects. Both fixed-point and floating- point number representations are considered.

Number Representation

In digital signal processing, $(B + 1)$-bit fixed-point numbers are usually represented as two's-complement signed fractions in the format

$$b_0 \cdot b_{-1} b_{-2} \cdots b_{-B}$$

The number represented is then

$$X = -b_0 + b_{-1}2^{-1} + b_{-2}2^{-2} + \cdots + b_{-B}2^{-B} \tag{3.1}$$

where b_0 is the sign bit and the number range is $-1 \leq X < 1$. The advantage of this representation is that the product of two numbers in the range from -1 to 1 is another number in the same range.

Floating-point numbers are represented as

$$X = (-1)^s m 2^c \tag{3.2}$$

where s is the *sign bit*, m is the **mantissa**, and c is the *characteristic* or *exponent*. To make the representation of a number unique, the mantissa is *normalized* so that $0.5 \leq m < 1$.

Although floating-point numbers are always represented in the form of (3.2), the way in which this representation is actually *stored* in a machine may differ. Since $m \geq 0.5$, it is not necessary to store the 2^{-1}-weight bit of m, which is always set. Therefore, in practice numbers are usually stored as

$$X = (-1)^s (0.5 + f) 2^c \tag{3.3}$$

where f is an unsigned fraction, $0 \leq f < 0.5$.

Most floating-point processors now use the IEEE Standard 754 32-bit floating-point format for storing numbers. According to this standard the exponent is stored as an unsigned integer p where

$$p = c + 126 \tag{3.4}$$

Therefore, a number is stored as

$$X = (-1)^s (0.5 + f) 2^{p-126} \tag{3.5}$$

where s is the sign bit, f is a 23-b unsigned fraction in the range $0 \leq f < 0.5$, and p is an 8-b unsigned integer in the range $0 \leq p \leq 255$. The total number of bits is $1 + 23 + 8 = 32$. For example, in IEEE format 3/4 is written $(-1)^0(0.5 + 0.25)2^0$ so $s = 0$, $p = 126$, and $f = 0.25$. The value $X = 0$ is a unique case and is represented by all bits zero (i.e., $s = 0$, $f = 0$, and $p = 0$). Although the 2^{-1}-weight mantissa bit is not actually stored, it does exist so the mantissa has 24 b plus a sign bit.

Fixed-Point Quantization Errors

In fixed-point arithmetic, a multiply doubles the number of significant bits. For example, the product of the two 5-b numbers 0.0011 and 0.1001 is the 10-b number 00.000 110 11. The extra bit to the left of the decimal point can be discarded without introducing any error. However, the least significant four of the remaining bits must ultimately be discarded by some form of quantization so that the result can be stored to 5 b for use in other calculations. In the example above this results in 0.0010 (quantization by rounding) or 0.0001 (quantization by truncating). When a sum of products calculation is performed, the quantization can be performed either after each multiply or after all products have been summed with double-length precision.

We will examine three types of fixed-point quantization—rounding, truncation, and magnitude truncation. If X is an exact value, then the rounded value will be denoted $Q_r(X)$, the truncated value $Q_t(X)$, and the magnitude truncated value $Q_{mt}(X)$. If the quantized value has B bits to the right of the decimal point, the quantization step size is

$$O = 2^{-B} \tag{3.6}$$

Since rounding selects the quantized value nearest the unquantized value, it gives a value which is never more than $\pm O/2$ away from the exact value. If we denote the rounding error by

$$\epsilon_r = Q_r(X) - X \tag{3.7}$$

then

$$-\frac{O}{2} \le \epsilon_r \le \frac{O}{2} \tag{3.8}$$

Truncation simply discards the low-order bits, giving a quantized value that is always less than or equal to the exact value so

$$-O < \epsilon_t \le 0 \tag{3.9}$$

Magnitude truncation chooses the nearest quantized value that has a magnitude less than or equal to the exact value so

$$-O < \epsilon_{mt} < O \tag{3.10}$$

The error resulting from quantization can be modeled as a random variable uniformly distributed over the appropriate error range. Therefore, calculations with roundoff error can be considered error-free calculations that have been corrupted by additive white noise. The mean of this noise for rounding is

$$m_{\epsilon_r} = E\{\epsilon_r\} = \frac{1}{O}\int_{-O/2}^{O/2} \epsilon_r \, d\epsilon_r = 0 \tag{3.11}$$

where $E\{\}$ represents the operation of taking the expected value of a random variable. Similarly, the variance of the noise for rounding is

$$\sigma_{\epsilon_r}^2 = E\{(\epsilon_r - m_{\epsilon_r})^2\} = \frac{1}{O}\int_{-O/2}^{O/2} (\epsilon_r - m_{\epsilon_r})^2 \, d\epsilon_r = \frac{O^2}{12} \tag{3.12}$$

Likewise, for truncation,

$$\begin{aligned} m_{\epsilon_t} &= E\{\epsilon_t\} = -\frac{O}{2} \\ \sigma_{\epsilon_t}^2 &= E\{(\epsilon_t - m_{\epsilon_t})^2\} = \frac{O^2}{12} \end{aligned} \tag{3.13}$$

and, for magnitude truncation

$$\begin{aligned} m_{\epsilon_{mt}} &= E\{\epsilon_{mt}\} = 0 \\ \sigma_{\epsilon_{mt}}^2 &= E\{(\epsilon_{mt} - m_{\epsilon_{mt}})^2\} = \frac{O^2}{3} \end{aligned} \tag{3.14}$$

Floating-Point Quantization Errors

With floating-point arithmetic it is necessary to quantize after both multiplications and additions. The addition quantization arises because, prior to addition, the mantissa of the smaller number in the sum is shifted right until the exponent of both numbers is the same. In general, this gives a sum mantissa that is too long and so must be quantized.

We will assume that quantization in floating-point arithmetic is performed by rounding. Because of the exponent in floating-point arithmetic, it is the relative error that is important. The relative error is defined as

$$\varepsilon_r = \frac{Q_r(X) - X}{X} = \frac{\epsilon_r}{X} \tag{3.15}$$

Since $X = (-1)^s m 2^c$, $Q_r(X) = (-1)^s Q_r(m) 2^c$ and

$$\varepsilon_r = \frac{Q_r(m) - m}{m} = \frac{\epsilon}{m} \tag{3.16}$$

If the quantized mantissa has B bits to the right of the decimal point, $|\epsilon| < O/2$ where, as before, $O = 2^{-B}$. Therefore, since $0.5 \le m < 1$,

$$|\varepsilon_r| < O \tag{3.17}$$

If we assume that ϵ is uniformly distributed over the range from $-O/2$ to $O/2$ and m is uniformly distributed over 0.5 to 1,

$$\begin{aligned} m_{\varepsilon_r} &= E\left[\frac{\epsilon}{m}\right] = 0 \\ \sigma^2_{\varepsilon_r} &= E\left[\frac{\epsilon^2}{m^2}\right] = \frac{2}{O}\int_{1/2}^{1}\int_{-O/2}^{O/2}\frac{\epsilon^2}{m^2}\,d\epsilon\,dm \\ &= \frac{O^2}{6} = (0.167)2^{-2B} \end{aligned} \tag{3.18}$$

In practice, the distribution of m is not exactly uniform. Actual measurements of roundoff noise in [1] suggested that

$$\sigma^2_{\varepsilon_r} \approx 0.23 O^2 \tag{3.19}$$

while a detailed theoretical and experimental analysis in [2] determined

$$\sigma^2_{\varepsilon_r} \approx 0.18 O^2 \tag{3.20}$$

From (3.15) we can represent a quantized floating-point value in terms of the unquantized value and the random variable ε_r using

$$Q_r(X) = X(1 + \varepsilon_r) \tag{3.21}$$

Therefore, the finite-precision product X_1X_2 and the sum $X_1 + X_2$ can be written

$$fl(X_1X_2) = X_1X_2(1 + \varepsilon_r) \tag{3.22}$$

and

$$fl(X_1 + X_2) = (X_1 + X_2)(1 + \varepsilon_r) \tag{3.23}$$

where ε_r is zero-mean with the variance of (3.20).

Roundoff Noise

To determine the roundoff noise at the output of a digital filter we will assume that the noise due to a quantization is stationary, white, and uncorrelated with the filter input, output, and internal variables. This assumption is good if the filter input changes from sample to sample in a sufficiently complex manner. It is not valid for zero or constant inputs for which the effects of rounding are analyzed from a limit cycle perspective.

To satisfy the assumption of a sufficiently complex input, roundoff noise in digital filters is often calculated for the case of a zero-mean white noise filter input signal $x(n)$ of variance σ_x^2. This simplifies calculation of the output roundoff noise because expected values of the form $\{E\ x(n)x(n-k)\}$ are zero for $k \neq 0$ and give σ_x^2 when $k = 0$. This approach to analysis has been found to give estimates of the output roundoff noise that are close to the noise actually observed for other input signals.

Another assumption that will be made in calculating roundoff noise is that the product of two quantization errors is zero. To justify this assumption, consider the case of a 16-b fixed-point processor. In this case a quantization error is of the order 2^{-15}, while the product of two quantization errors is of the order 2^{-30}, which is negligible by comparison.

If a linear system with impulse response $g(n)$ is excited by white noise with mean m_x and variance σ_x^2, the output is noise of mean [3, pp.788–790]

$$m_y = m_x \sum_{n=-\infty}^{\infty} g(n) \tag{3.24}$$

and variance

$$\sigma_y^2 = \sigma_x^2 \sum_{n=-\infty}^{\infty} g^2(n) \tag{3.25}$$

Therefore, if $g(n)$ is the impulse response from the point where a roundoff takes place to the filter output, the contribution of that roundoff to the variance (mean-square value) of the output roundoff noise is given by (3.25) with σ_x^2 replaced with the variance of the roundoff. If there is more than one source of roundoff error in the filter, it is assumed that the errors are uncorrelated so the output noise variance is simply the sum of the contributions from each source.

Roundoff Noise in FIR Filters

The simplest case to analyze is a finite impulse response (FIR) filter realized via the convolution summation

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k) \tag{3.26}$$

When fixed-point arithmetic is used and quantization is performed after each multiply, the result of the N multiplies is N-times the quantization noise of a single multiply. For example, rounding after each multiply gives, from (3.6) and (3.12), an output noise variance of

$$\sigma_o^2 = N\,\frac{2^{-2B}}{12} \tag{3.27}$$

Virtually all digital signal processor integrated circuits contain one or more double-length accumulator registers which permit the sum-of-products in (3.26) to be accumulated without quantization. In this case only a single quantization is necessary following the summation and

$$\sigma_o^2 = \frac{2^{-2B}}{12} \tag{3.28}$$

For the floating-point roundoff noise case we will consider (3.26) for $N=4$ and then generalize the result to other values of N . The finite-precision output can be written as the exact output plus an error term $e(n)$. Thus,

$$\begin{aligned} y(n) + e(n) = (\{&[h(0)x(n)[1 + \varepsilon_1(n)] \\ &+ h(1)x(n-1)[1+\varepsilon_2(n)]][1+\varepsilon_3(n)] \\ &+ h(2)x(n-2)[1+\varepsilon_4(n)]\}\{1+\varepsilon_5(n)\} \\ &+ h(3)x(n-3)[1+\varepsilon_6(n)])[1+\varepsilon_7(n)] \end{aligned} \tag{3.29}$$

In (3.29), $\varepsilon_1(n)$ represents the error in the first product, $\varepsilon_2(n)$ the error in the second product, $\varepsilon_3(n)$ the error in the first addition, etc. Notice that it has been assumed that the products are summed in the order implied by the summation of (3.26).

Expanding (3.29), ignoring products of error terms, and recognizing $y(n)$ gives

$$\begin{aligned} e(n) = \; & h(0)x(n)[\varepsilon_1(n) + \varepsilon_3(n) + \varepsilon_5(n) + \varepsilon_7(n)] \\ &+ h(1)x(n-1)[\varepsilon_2(n) + \varepsilon_3(n) + \varepsilon_5(n) + \varepsilon_7(n)] \\ &+ h(2)x(n-2)[\varepsilon_4(n) + \varepsilon_5(n) + \varepsilon_7(n)] \\ &+ h(3)x(n-3)[\varepsilon_6(n) + \varepsilon_7(n)] \end{aligned} \tag{3.30}$$

Assuming that the input is white noise of variance σ_x^2 so that $E\{x(n)x(n-k)\}$ is zero for $k \neq 0$, and assuming that the errors are uncorrelated,

$$E\{e^2(n)\} = [4h^2(0) + 4h^2(1) + 3h^2(2) + 2h^2(3)]\sigma_x^2\sigma_{\varepsilon r}^2 \tag{3.31}$$

In general, for any N ,

$$\sigma_o^2 = E\{e^2(n)\} = \left[Nh^2(0) + \sum_{k=1}^{N-1}(N+1-k)h^2(k)\right]\sigma_x^2\sigma_{\varepsilon r}^2 \tag{3.32}$$

Notice that if the order of summation of the product terms in the convolution summation is changed, then the order in which the $h(k)$'s appear in (3.32) changes. If the order is changed so that the $h(k)$ with smallest magnitude is first, followed by the next smallest, etc., then the roundoff noise variance is minimized. However, performing the convolution summation in nonsequential order greatly complicates data indexing and so may not be worth the reduction obtained in roundoff noise.

Roundoff Noise in Fixed-Point IIR Filters

To determine the roundoff noise of a fixed-point infinite impulse response (IIR) filter realization, consider a causal first-order filter with impulse response

$$h(n) = a^n u(n) \tag{3.33}$$

realized by the difference equation

$$y(n) = ay(n-1) + x(n) \tag{3.34}$$

Due to roundoff error, the output actually obtained is

$$\hat{y}(n) = Q\{ay(n-1) + x(n)\} = ay(n-1) + x(n) + e(n) \tag{3.35}$$

where $e(n)$ is a random roundoff noise sequence. Since $e(n)$ is injected at the same point as the input, it propagates through a system with impulse response $h(n)$. Therefore, for fixed-point arithmetic with rounding, the output roundoff noise variance from (3.6), (3.12), (3.25), and (3.33) is

$$\sigma_o^2 = \frac{O^2}{12} \sum_{n=-\infty}^{\infty} h^2(n) = \frac{O^2}{12} \sum_{n=0}^{\infty} a^{2n} = \frac{2^{-2B}}{12} \frac{1}{1-a^2} \tag{3.36}$$

With fixed-point arithmetic there is the possibility of overflow following addition. To avoid overflow it is necessary to restrict the input signal amplitude. This can be accomplished by either placing a *scaling* multiplier at the filter input or by simply limiting the maximum input signal amplitude. Consider the case of the first-order filter of (3.34). The transfer function of this filter is

$$H(e^{j\omega}) = \frac{Y(e^{j\omega})}{X(e^{j\omega})} = \frac{1}{e^{j\omega} - a} \tag{3.37}$$

so

$$|H(e^{j\omega})|^2 = \frac{1}{1 + a^2 - 2a\cos(\omega)} \tag{3.38}$$

and

$$|H(e^{j\omega})|_{\max} = \frac{1}{1 - |a|} \tag{3.39}$$

The peak gain of the filter is $1/(1 - |a|)$ so limiting input signal amplitudes to $|x(n)| \le 1 - |a|$ will make overflows unlikely.

An expression for the output roundoff noise-to-signal ratio can easily be obtained for the case where the filter input is white noise, uniformly distributed over the interval from $-(1 - |a|)$ to $(1 - |a|)$ [4, 5]. In this case

$$\sigma_x^2 = \frac{1}{2(1-|a|)} \int_{-(1-|a|)}^{1-|a|} x^2\, dx = \frac{1}{3}(1 - |a|)^2 \tag{3.40}$$

so, from (3.25),

$$\sigma_y^2 = \frac{1}{3} \frac{(1 - |a|)^2}{1 - a^2} \tag{3.41}$$

Combining (3.36) and (3.41) then gives

$$\frac{\sigma_o^2}{\sigma_y^2} = \frac{2^{-2B}}{12} \frac{1}{1-a^2} \frac{1 - a^2}{(1-|a|)^2} = \frac{2^{-2B}}{12} \frac{3}{(1-|a|)^2} \tag{3.42}$$

Notice that the noise-to-signal ratio increases without bound as $|a| \to 1$.

Similar results can be obtained for the case of the causal second-order filter realized by the difference equation

$$y(n) = 2r\cos(\theta)y(n-1) - r^2 y(n-2) + x(n) \tag{3.43}$$

This filter has complex-conjugate poles at $re^{\pm j\theta}$ and impulse response

$$h(n) = \frac{1}{\sin(\theta)} r^n \sin[(n+1)\theta]u(n) \tag{3.44}$$

Due to roundoff error, the output actually obtained is

$$\hat{y}(n) = 2r\cos(\theta)y(n-1) - r^2 y(n-2) + x(n) + e(n) \tag{3.45}$$

There are two noise sources contributing to $e(n)$ if quantization is performed after each multiply, and there is one noise source if quantization is performed after summation. Since

$$\sum_{n=-\infty}^{\infty} h^2(n) = \frac{1+r^2}{1-r^2}\,\frac{1}{(1+r^2)^2 - 4r^2\cos^2(\theta)} \tag{3.46}$$

the output roundoff noise is

$$\sigma_o^2 = \nu\,\frac{2^{-2B}}{12}\,\frac{1+r^2}{1-r^2}\,\frac{1}{(1+r^2)^2 - 4r^2\cos^2(\theta)} \tag{3.47}$$

where $\nu = 1$ for quantization after summation, and $\nu = 2$ for quantization after each multiply.

To obtain an output noise-to-signal ratio we note that

$$H(e^{j\omega}) = \frac{1}{1 - 2r\cos(\theta)e^{-j\omega} + r^2 e^{-j2\omega}} \tag{3.48}$$

and, using the approach of [6],

$$|H(e^{j\omega})|^2_{\max} = \frac{1}{4r^2\left[\mathrm{sat}\left(\frac{r^2}{1+r^2}\cos(\theta)\right) - \frac{1+r^2}{2}\cos(\theta)\right]^2 + \left[\frac{1-r^2}{2}\sin(\theta)\right]^2} \tag{3.49}$$

where

$$\mathrm{sat}(\mu) = \begin{cases} 1 & \mu > 1 \\ \mu & -1 \le \mu \le 1 \\ -1 & \mu < -1 \end{cases} \tag{3.50}$$

Following the same approach as for the first-order case then gives

$$\frac{\sigma_o^2}{\sigma_y^2} = \nu\,\frac{2^{-2B}}{12}\,\frac{1+r^2}{1-r^2}\,\frac{3}{(1+r^2)^2 - 4r^2\cos^2(\theta)} \times \frac{1}{4r^2\left[\mathrm{sat}\left(\frac{r^2}{1+r^2}\cos(\theta)\right) - \frac{1+r^2}{2}\cos(\theta)\right]^2 + \left[\frac{1-r^2}{2}\sin(\theta)\right]^2} \tag{3.51}$$

Figure 3.1 is a contour plot showing the noise-to-signal ratio of (3.51) for $\nu = 1$ in units of the noise variance of a single quantization, $2^{-2B}/12$. The plot is symmetrical about $\theta = 90^\circ$, so only the range from 0° to 90° is shown. Notice that as $r \to 1$, the roundoff noise increases without bound. Also notice that the noise increases as $\theta \to 0^\circ$.

It is possible to design state-space filter realizations that minimize fixed-point roundoff noise [7]–[10]. Depending on the transfer function being realized, these structures may provide a roundoff noise level that is orders-of-magnitude lower than for a nonoptimal realization. The price paid for this reduction in roundoff noise is an increase in the number of computations required to implement the filter. For an Nth-order filter the increase is from roughly $2N$ multiplies for a direct form realization to roughly $(N+1)^2$ for an optimal realization. However, if the filter is realized by the parallel or cascade connection of first- and second-order optimal subfilters, the increase is only to about $4N$ multiplies. Furthermore, near-optimal realizations exist that increase the number of multiplies to only about $3N$ [10].

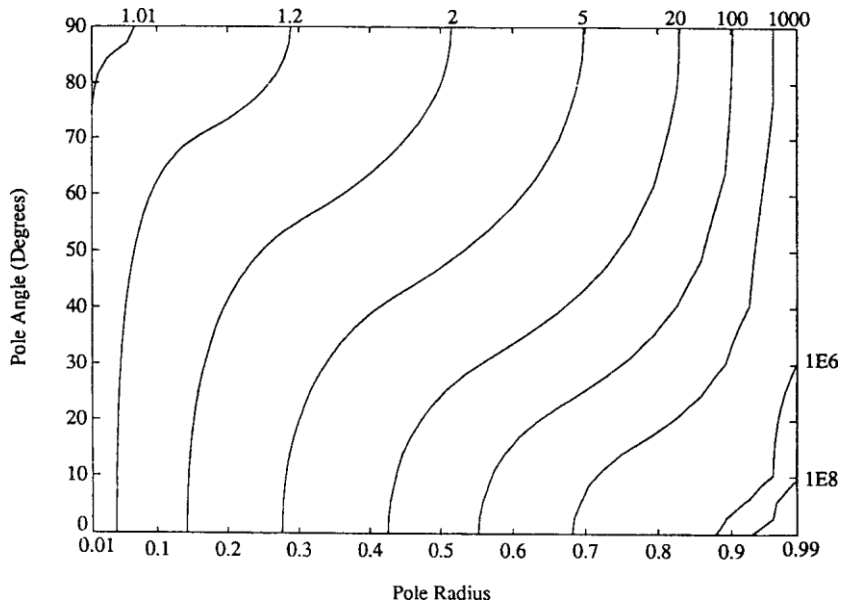


FIGURE 3.1: Normalized fixed-point roundoff noise variance.

Roundoff Noise in Floating-Point IIR Filters

For floating-point arithmetic it is first necessary to determine the injected noise variance of each quantization. For the first-order filter this is done by writing the computed output as

$$y(n) + e(n) = [ay(n-1)(1+\varepsilon_1(n)) + x(n)](1+\varepsilon_2(n)) \tag{3.52}$$

where $\varepsilon_1(n)$ represents the error due to the multiplication and $\varepsilon_2(n)$ represents the error due to the addition. Neglecting the product of errors, (3.52) becomes

$$\begin{aligned} y(n) + e(n) \approx\ & ay(n-1) + x(n) + ay(n-1)\varepsilon_1(n) \\ & + ay(n-1)\varepsilon_2(n) + x(n)\varepsilon_2(n) \end{aligned} \tag{3.53}$$

Comparing (3.34) and (3.53), it is clear that

$$e(n) = ay(n-1)\varepsilon_1(n) + ay(n-1)\varepsilon_2(n) + x(n)\varepsilon_2(n) \tag{3.54}$$

Taking the expected value of $e^2(n)$ to obtain the injected noise variance then gives

$$\begin{aligned} E\{e^2(n)\} = {} & a^2E\{y^2(n-1)\}E\{\varepsilon_1^2(n)\} + a^2E\{y^2(n-1)\}E\{\varepsilon_2^2(n)\} \\ & + E\{x^2(n)\}E\{\varepsilon_2^2(n)\} + E\{x(n)y(n-1)\}E\{\varepsilon_2^2(n)\} \end{aligned} \tag{3.55}$$

To carry this further it is necessary to know something about the input. If we assume the input is zero-mean white noise with variance σ_x^2, then $E\{x^2(n)\} = \sigma_x^2$ and the input is uncorrelated with past values of the output so $E\{x(n)y(n-1)\} = 0$ giving

$$E\{e^2(n)\} = 2a^2\sigma_y^2\sigma_{\varepsilon r}^2 + \sigma_x^2\sigma_{\varepsilon r}^2 \tag{3.56}$$

and

$$\sigma_o^2 = \left[2a^2\sigma_y^2\sigma_{\varepsilon_r}^2 + \sigma_x^2\sigma_{\varepsilon_r}^2\right]\sum_{n=-\infty}^{\infty} h^2(n)$$

$$= \frac{2a^2\sigma_y^2 + \sigma_x^2}{1-a^2}\sigma_{\varepsilon_r}^2 \tag{3.57}$$

However,

$$\sigma_y^2 = \sigma_x^2 \sum_{n=-\infty}^{\infty} h^2(n) = \frac{\sigma_x^2}{1-a^2} \tag{3.58}$$

so

$$\sigma_o^2 = \frac{1+a^2}{(1-a^2)^2}\sigma_{\varepsilon_r}^2\sigma_x^2 = \frac{1+a^2}{1-a^2}\sigma_{\varepsilon_r}^2\sigma_y^2 \tag{3.59}$$

and the output roundoff noise-to-signal ratio is

$$\frac{\sigma_o^2}{\sigma_y^2} = \frac{1+a^2}{1-a^2}\sigma_{\varepsilon_r}^2 \tag{3.60}$$

Similar results can be obtained for the second-order filter of (3.43) by writing

$$\begin{aligned} y(n) + e(n) = {} & ([2r\cos(\theta)y(n-1)(1+\varepsilon_1(n)) - r^2y(n-2)(1+\varepsilon_2(n))] \\ & \times [1+\varepsilon_3(n)] + x(n))(1+\varepsilon_4(n)) \end{aligned} \tag{3.61}$$

Expanding with the same assumptions as before gives

$$\begin{aligned} e(n) \approx {} & 2r\cos(\theta)y(n-1)[\varepsilon_1(n)+\varepsilon_3(n)+\varepsilon_4(n)] \\ & - r^2y(n-2)[\varepsilon_2(n)+\varepsilon_3(n)+\varepsilon_4(n)] + x(n)\varepsilon_4(n) \end{aligned} \tag{3.62}$$

and

$$\begin{aligned} E\{e^2(n)\} = {} & 4r^2\cos^2(\theta)\sigma_{\varepsilon_r}^2 3\sigma_y^2 + r^2\sigma_{\varepsilon_r}^2 3\sigma_y^2 \\ & + \sigma_x^2\sigma_{\varepsilon_r}^2 - 8r^3\cos(\theta)\sigma_{\varepsilon_r}^2 E\{y(n-1)y(n-2)\} \end{aligned} \tag{3.63}$$

However,

$$\begin{aligned} & E\{y(n-1)y(n-2)\} \\ & = E\{[2r\cos(\theta)y(n-2) - r^2y(n-3) + x(n-1)]y(n-2)\} \\ & = 2r\cos(\theta)E\{y^2(n-2)\} - r^2E\{y(n-2)y(n-3)\} \\ & = 2r\cos(\theta)E\{y^2(n-2)\} - r^2E\{y(n-1)y(n-2)\} \\ & = \frac{2r\cos(\theta)}{1+r^2}\sigma_y^2 \end{aligned} \tag{3.64}$$

so

$$E\{e^2(n)\} = \sigma_{\varepsilon_r}^2\sigma_x^2 + \left[3r^4 + 12r^2\cos^2(\theta) - \frac{16r^4\cos^2(\theta)}{1+r^2}\right]\sigma_{\varepsilon_r}^2\sigma_y^2 \tag{3.65}$$

and

$$\begin{aligned} \sigma_o^2 &= E\{e^2(n)\}\sum_{n=-\infty}^{\infty} h^2(n) \\ &= \xi\left[\sigma_{\varepsilon_r}^2\sigma_x^2 + \left[3r^4 + 12r^2\cos^2(\theta) - \frac{16r^4\cos^2(\theta)}{1+r^2}\right]\sigma_{\varepsilon_r}^2\sigma_y^2\right] \end{aligned} \tag{3.66}$$

where from (3.46),

$$\xi = \sum_{n=-\infty}^{\infty} h^2(n) = \frac{1+r^2}{1-r^2}\,\frac{1}{(1+r^2)^2 - 4r^2\cos^2(\theta)} \tag{3.67}$$

Since $\sigma_y^2 = \xi\sigma_x^2$, the output roundoff noise-to-signal ratio is then

$$\frac{\sigma_o^2}{\sigma_y^2} = \xi\left[1 + \xi\left[3r^4 + 12r^2\cos^2(\theta) - \frac{16r^4\cos^2(\theta)}{1+r^2}\right]\right]\sigma_{\varepsilon r}^2 \tag{3.68}$$

Figure 3.2 is a contour plot showing the noise-to-signal ratio of (3.68) in units of the noise variance of a single quantization $\sigma_{\varepsilon r}^2$. The plot is symmetrical about $\theta = 90^\circ$, so only the range from 0° to 90° is shown. Notice the similarity of this plot to that of Fig. 3.1 for the fixed-point case. It has been observed that filter structures generally have very similar fixed-point and floating-point roundoff characteristics [2]. Therefore, the techniques of [7] – [10], which were developed for the fixed-point case, can also be used to design low-noise floating-point filter realizations. Furthermore, since it is not necessary to scale the floating-point realization, the low-noise realizations need not require significantly more computation than the direct form realization.

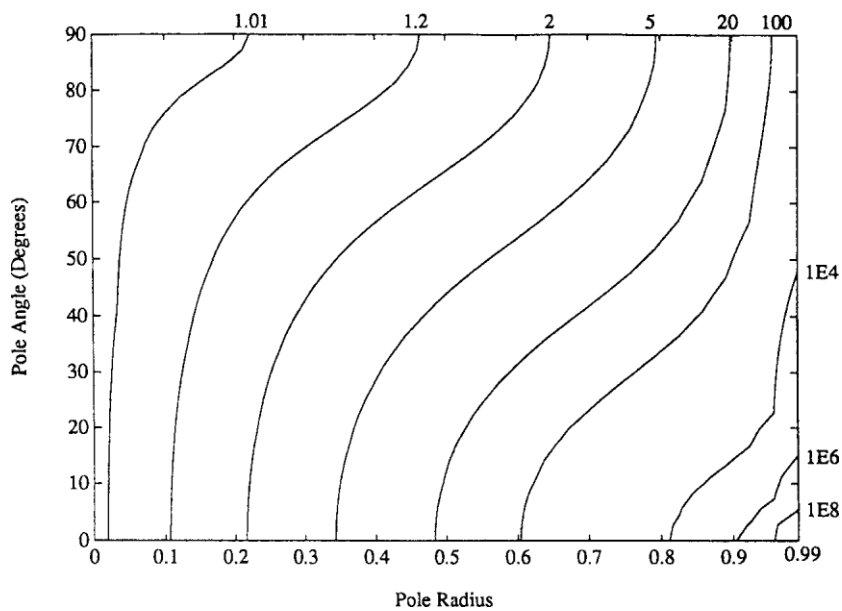


FIGURE 3.2: Normalized floating-point roundoff noise variance.

Limit Cycles

A limit cycle, sometimes referred to as a **multiplier roundoff limit cycle**, is a low-level oscillation that can exist in an otherwise stable filter as a result of the nonlinearity associated with rounding (or truncating) internal filter calculations [11]. Limit cycles require recursion to exist and do not occur in nonrecursive FIR filters.

As an example of a limit cycle, consider the second-order filter realized by

$$y(n) = Q_r\left[\frac{7}{8}y(n-1) - \frac{5}{8}y(n-2) + x(n)\right] \tag{3.69}$$

where $Q_r[\cdot]$ represents quantization by rounding. This is stable filter with poles at $0.4375 \pm j\,0.6585$. Consider the implementation of this filter with 4-b (3-b and a sign bit) two's complement fixed-point arithmetic, zero initial conditions ($y(-1) = y(-2) = 0$), and an input sequence $x(n) = \frac{3}{8}\delta(n)$, where $\delta(n)$ is the unit impulse or unit sample. The following sequence is obtained;

$$\begin{aligned}
y(0) &= Q_r\left[\frac{3}{8}\right] = \frac{3}{8} \\
y(1) &= Q_r\left[\frac{21}{64}\right] = \frac{3}{8} \\
y(2) &= Q_r\left[\frac{32}{64}\right] = \frac{1}{2} \\
y(3) &= Q_r\left[-\frac{1}{32}\right] = 0 \\
y(4) &= Q_r\left[\frac{5}{64}\right] = \frac{1}{8} \\
y(5) &= Q_r\left[-\frac{1}{32}\right] = 0 \\
y(6) &= Q_r\left[-\frac{5}{64}\right] = -\frac{1}{8} \\
y(7) &= Q_r\left[-\frac{7}{64}\right] = -\frac{1}{8} \\
y(8) &= Q_r\left[-\frac{1}{32}\right] = 0 \\
y(9) &= Q_r\left[\frac{5}{64}\right] = \frac{1}{8} \\
y(10) &= Q_r\left[\frac{7}{64}\right] = \frac{1}{8} \\
y(11) &= Q_r\left[\frac{1}{32}\right] = 0 \\
y(12) &= Q_r\left[-\frac{5}{64}\right] = -\frac{1}{8}
\end{aligned} \tag{3.70}$$

Notice that while the input is zero except for the first sample, the output oscillates with amplitude 1/8 and period 6.

Limit cycles are primarily of concern in fixed-point recursive filters. As long as floating-point filters are realized as the parallel or cascade connection of first- and second-order subfilters, limit cycles will generally not be a problem since limit cycles are practically not observable in first- and second-order systems implemented with 32-b floating-point arithmetic [12]. It has been shown that such systems must have an extremely small margin of stability for limit cycles to exist at anything other than underflow levels, which are at an amplitude of less than 10^{-38} [12].

There are at least three ways of dealing with limit cycles when fixed-point arithmetic is used. One is to determine a bound on the maximum limit cycle amplitude, expressed as an integral number of quantization steps [13]. It is then possible to choose a word length that makes the limit cycle amplitude acceptably low. Alternately, limit cycles can be prevented by randomly rounding calculations up or down [14]. However, this approach is complicated to implement. The third approach is to properly choose the filter realization structure and then quantize the filter calculations using magnitude truncation [15, 16]. This approach has the disadvantage of producing more roundoff noise than truncation or rounding [see (3.12)–(3.14)].

Overflow Oscillations

With fixed-point arithmetic it is possible for filter calculations to overflow. This happens when two numbers of the same sign add to give a value having magnitude greater than one. Since numbers with magnitude greater than one are not representable, the result overflows. For example, the two's complement numbers 0.101 (5/8) and 0.100 (4/8) add to give 1.001 which is the two's complement representation of 7/8.

The overflow characteristic of two's complement arithmetic can be represented as $R\{\}$ where

$$R\{X\} = \begin{cases} X - 2 & X \geq 1 \\ X & -1 \leq X < 1 \\ X + 2 & X < -1 \end{cases} \tag{3.71}$$

For the example just considered, $R\{9/8\} = -7/8$.

An overflow oscillation, sometimes also referred to as an *adder overflow limit cycle*, is a high-level oscillation that can exist in an otherwise stable fixed-point filter due to the gross nonlinearity associated with the overflow of internal filter calculations [17]. Like limit cycles, overflow oscillations require recursion to exist and do not occur in nonrecursive FIR filters. Overflow oscillations also do not occur with floating-point arithmetic due to the virtual impossibility of overflow.

As an example of an overflow oscillation, once again consider the filter of (3.69) with 4-b fixed-point two's complement arithmetic and with the two's complement overflow characteristic of (3.71):

$$y(n) = Q_r\left[R\left\{\frac{7}{8}y(n-1) - \frac{5}{8}y(n-2) + x(n)\right\}\right] \tag{3.72}$$

In this case we apply the input

$$\begin{aligned} x(n) &= -\frac{3}{4}\delta(n) - \frac{5}{8}\delta(n-1) \\ &= \left\{-\frac{3}{4}, -\frac{5}{8}, 0, 0, \cdots\right\}, \end{aligned} \tag{3.73}$$

giving the output sequence

$$\begin{aligned} y(0) &= Q_r\left[R\left\{-\frac{3}{4}\right\}\right] = Q_r\left[-\frac{3}{4}\right] = -\frac{3}{4} \\ y(1) &= Q_r\left[R\left\{-\frac{41}{32}\right\}\right] = Q_r\left[-\frac{23}{32}\right] = -\frac{3}{4} \\ y(2) &= Q_r\left[R\left\{\frac{9}{8}\right\}\right] = Q_r\left[-\frac{7}{8}\right] = -\frac{7}{8} \\ y(3) &= Q_r\left[R\left\{-\frac{79}{64}\right\}\right] = Q_r\left[-\frac{49}{64}\right] = -\frac{3}{4} \end{aligned}$$

$$
\begin{aligned}
y(4) &= Q_r\left[\frac{77}{64}\right] = Q_r\left[-\frac{7}{8}\right] = -\frac{7}{8} \\
y(5) &= Q_r\left[\frac{9}{8}\right] = Q_r\left[-\frac{7}{8}\right] = -\frac{7}{8} \\
y(6) &= Q_r\left[\frac{79}{64}\right] = Q_r\left[\frac{51}{64}\right] = \frac{3}{4} \\
y(7) &= Q_r\left[-\frac{9}{64}\right] = Q_r\left[-\frac{7}{64}\right] = -\frac{7}{64} \\
y(8) &= Q_r\left[-\frac{9}{8}\right] = Q_r\left[-\frac{7}{8}\right] = -\frac{7}{8}
\end{aligned}
\tag{3.74}
$$

This is a large-scale oscillation with nearly full-scale amplitude.

There are several ways to prevent overflow oscillations in fixed-point filter realizations. The most obvious is to scale the filter calculations so as to render overflow impossible. However, this may unacceptably restrict the filter dynamic range. Another method is to force completed sums-of-products to saturate at ± 1, rather than overflowing [18, 19]. It is important to saturate only the completed sum, since intermediate overflows in two's complement arithmetic do not affect the accuracy of the final result. Most fixed-point digital signal processors provide for automatic saturation of completed sums if their *saturation arithmetic* feature is enabled. Yet another way to avoid overflow oscillations is to use a filter structure for which any internal filter transient is guaranteed to decay to zero [20]. Such structures are desirable anyway, since they tend to have low roundoff noise and be insensitive to coefficient quantization [21].

Coefficient Quantization Error

Each filter structure has its own finite, generally nonuniform grids of realizable pole and zero locations when the filter coefficients are quantized to a finite word length. In general the pole and zero locations desired in filter do not correspond exactly to the realizable locations. The error in filter performance (usually measured in terms of a frequency response error) resulting from the placement of the poles and zeroes at the nonideal but realizable locations is referred to as coefficient quantization error.

Consider the second-order filter with complex-conjugate poles

$$
\begin{aligned}
\lambda &= re^{\pm j\theta} \\
&= \lambda_r \pm j\lambda_i \\
&= r\cos(\theta) \pm jr\sin(\theta)
\end{aligned}
\tag{3.75}
$$

and transfer function

$$
H(z) = \frac{1}{1 - 2r\cos(\theta)z^{-1} + r^2 z^{-2}} \tag{3.76}
$$

realized by the difference equation

$$
y(n) = 2r\cos(\theta)y(n-1) - r^2 y(n-2) + x(n) \tag{3.77}
$$

Figure 3.3 from [5] shows that quantizing the difference equation coefficients results in a nonuniform grid of realizable pole locations in the z plane. The grid is defined by the intersection of vertical lines corresponding to quantization of $2\lambda_r$ and concentric circles corresponding to quantization of $-r^2$.

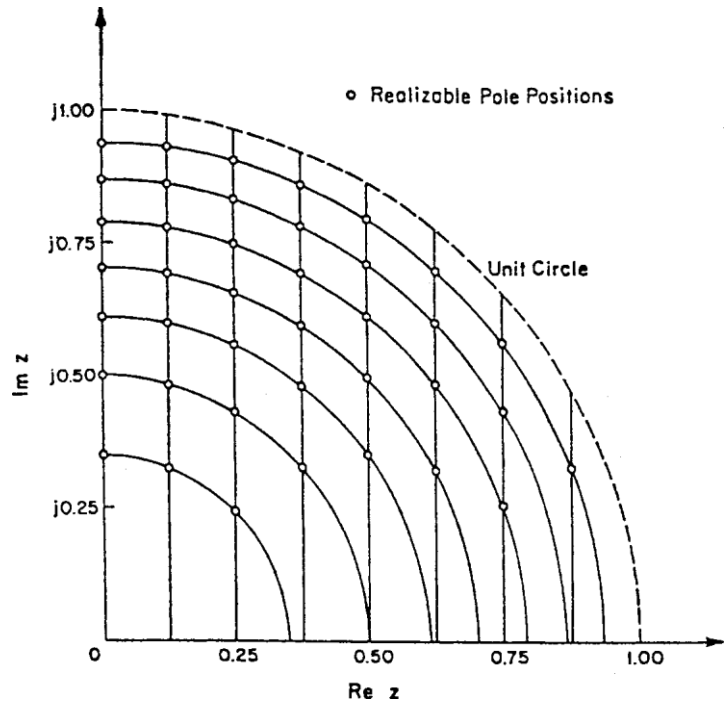


FIGURE 3.3: Realizable pole locations for the difference equation of (3.76).

The sparseness of realizable pole locations near $z = \pm 1$ will result in a large coefficient quantization error for poles in this region.

Figure 3.4 gives an alternative structure to (3.77) for realizing the transfer function of (3.76). Notice that quantizing the coefficients of this structure corresponds to quantizing λ_r and λ_i . As shown in Fig. 3.5 from [5], this results in a uniform grid of realizable pole locations. Therefore, large coefficient quantization errors are avoided for all pole locations.

It is well established that filter structures with low roundoff noise tend to be robust to coefficient quantization, and visa versa [22]– [24]. For this reason, the uniform grid structure of Fig. 3.4 is also popular because of its low roundoff noise. Likewise, the low-noise realizations of [7]– [10] can be expected to be relatively insensitive to coefficient quantization, and digital wave filters and lattice filters that are derived from low-sensitivity analog structures tend to have not only low coefficient sensitivity, but also low roundoff noise [25, 26].

It is well known that in a high-order polynomial with clustered roots, the root location is a very sensitive function of the polynomial coefficients. Therefore, filter poles and zeros can be much more accurately controlled if higher order filters are realized by breaking them up into the parallel or cascade connection of first- and second-order subfilters. One exception to this rule is the case of linear-phase FIR filters in which the symmetry of the polynomial coefficients and the spacing of the filter zeros around the unit circle usually permits an acceptable direct realization using the convolution summation.

Given a filter structure it is necessary to assign the ideal pole and zero locations to the realizable locations. This isgenerally done bysimply rounding or truncating thefilter coefficients to the available number of bits, or by assigning the ideal pole and zero locations to the nearest realizable locations. A more complicated alternative is to consider the original filter design problem as a problem in discrete

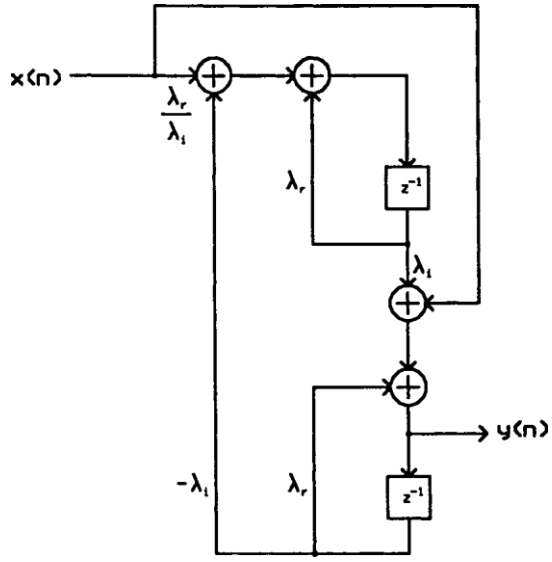


FIGURE 3.4: Alternate realization structure.

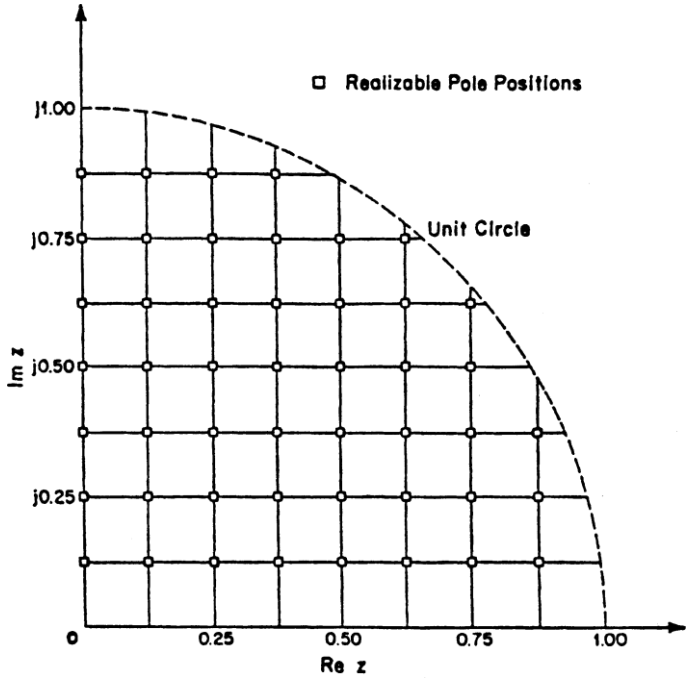


FIGURE 3.5: Realizable pole locations for the alternate realization structure.

optimization, and choose the realizable pole and zero locations that give the best approximation to the desired filter response [27]– [30].

Realization Considerations

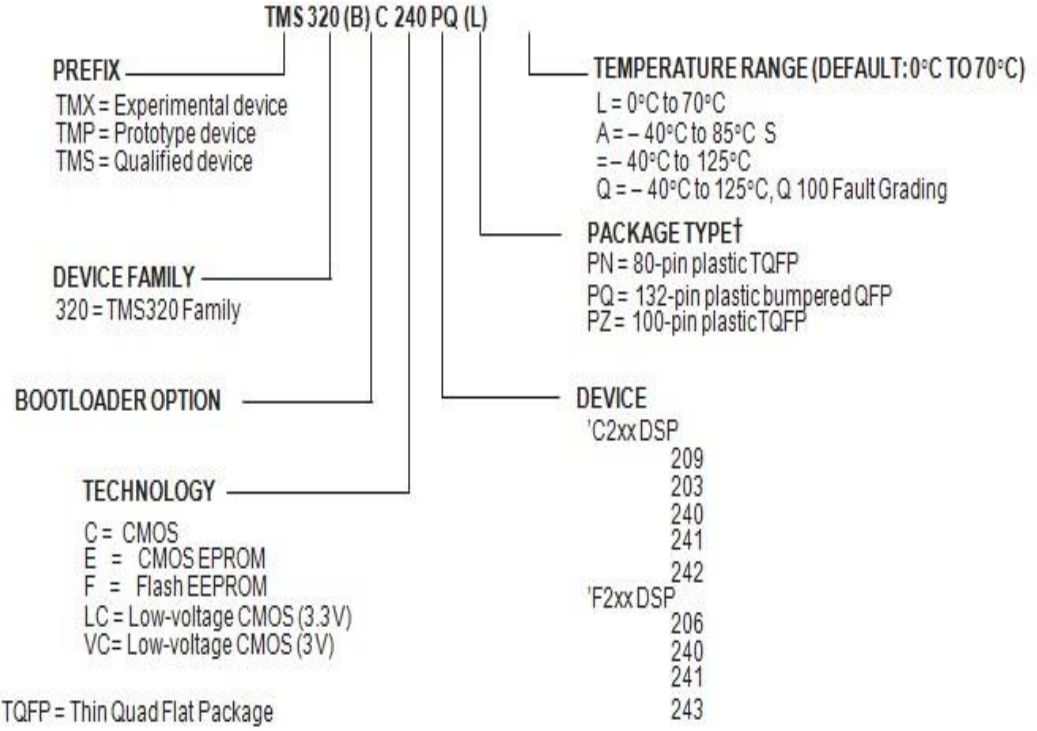
Linear-phase FIR digital filters can generally be implemented with acceptable coefficient quantization sensitivity using the direct convolution sum method. When implemented in this way on a digital signal processor, fixed-point arithmetic is not only acceptable but may actually be preferable to floating-point arithmetic. Virtually all fixed-point digital signal processors accumulate a sum of products in a double-length accumulator. This means that only a single quantization is necessary to compute an output. Floating-point arithmetic, on the other hand, requires a quantization after every multiply and after every add in the convolution summation. With 32-b floating-point arithmetic these quantizations introduce a small enough error to be insignificant for many applications.

When realizing IIR filters, either a parallel or cascade connection of first- and second-order subfilters is almost always preferable to a high-order direct-form realization. With the availability of very low-cost floating-point digital signal processors, like the Texas Instruments TMS320C32, it is highly recommended that floating-point arithmetic be used for IIR filters. Floating-point arithmetic simultaneously eliminates most concerns regarding scaling, limit cycles, and overflow oscillations. Regardless of the arithmetic employed, a low roundoff noise structure should be used for the second-order sections. Good choices are given in [2] and [10]. Recall that realizations with low fixed-point roundoff noise also have low floating-point roundoff noise. The use of a low roundoff noise structure for the second-order sections also tends to give a realization with low coefficient quantization sensitivity. First-order sections are not as critical in determining the roundoff noise and coefficient sensitivity of a realization, and so can generally be implemented with a simple direct form structure.

UNIT -5

Nomenclature- **TMS320C2407 DSP CONTROLLER**

TI device nomenclature also includes a suffix with the device family name. This suffix indicates the package type (for example, PN, PQ, and PZ) and temperature range (for example, L). Figure 16 provides a legend for reading the complete device name for any TMS320x2xx family member.



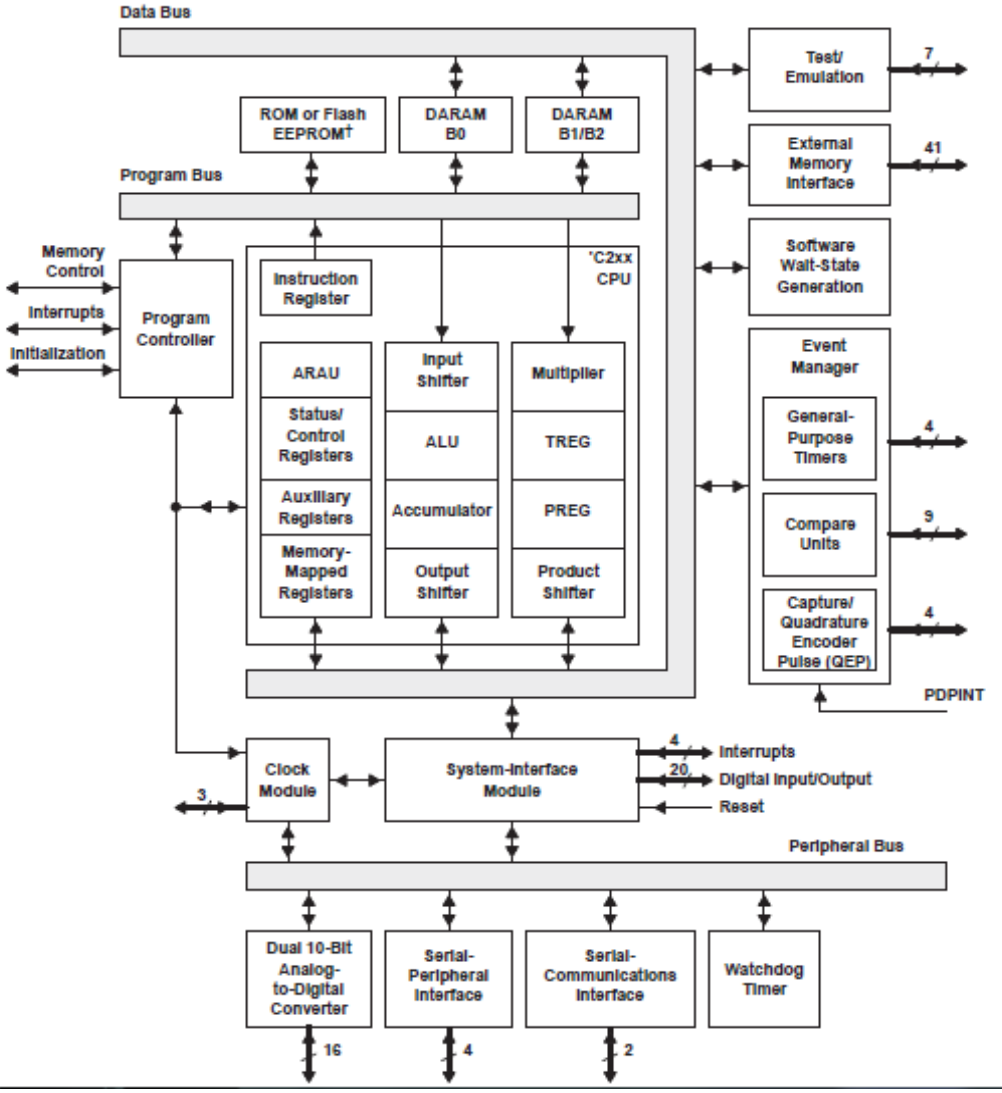
† TQFP = Thin Quad Flat Package

Figure 16. TMS320 Device Nomenclature

TMS 320 family overview -Architectural Overview

- Two Event Managers (A and B)
- General Purpose (GP) timers
- PWM generators for digital motor control
- Analog-to-digital converter
- Controller Area Network (CAN) interface
- Serial Peripheral Interface (SPI) – synchronous serial port
- Serial Communications Interface (SCI) – asynchronous serial port
- General-Purpose bi-directional digital I/O (GPIO) pins
- Watchdog Timer ("time-out" DSP reset device for system integrity

functional block diagram



INTRODUCTION TO THE TMSLF2407 DSP CONTROLLER

1. Introduction

The Texas Instruments TMS320LF2407 DSP Controller (referred to as the LF2407 in this text) is a programmable digital controller with a C2xx DSP central processing unit (CPU) as the core processor. The LF2407 contains the DSP core processor and useful peripherals integrated onto a single piece of silicon. The LF2407 combines the powerful CPU with on-chip memory and peripherals. With the DSP core and control-oriented peripherals integrated into a single chip, users can design very compact and cost-effective digital control systems.

The LF2407 DSP controller offers 40 million instructions per second (MIPS) performance. This high processing speed of the C2xx CPU allows users to compute parameters in real time rather than look up approximations from tables stored in memory. This fast performance is well suited for processing control parameters in applications such as notch filters or sensorless motor control algorithms where a large amount of calculations must be computed quickly.

While the "brain" of the LF2407 DSP is the C2xx core, the LF2407 contains several control-orientated peripherals onboard (see Fig. 1.1). The peripherals on the LF2407 make virtually any digital control requirement possible. Their applications range from analog to digital conversion to pulse width modulation (PWM) generation. Communication peripherals make possible the communication with external peripherals, personal computers, or other DSP processors. Below is a brief listing of the different peripherals onboard the LF2407 followed by a graphical layout depicted in Fig. 1.1.

The LF2407 peripheral set includes:

- Two Event Managers (A and B)
- General Purpose (GP) timers
- PWM generators for digital motor control
- Analog-to-digital converter
- Controller Area Network (CAN) interface
- Serial Peripheral Interface (SPI) – synchronous serial port
- Serial Communications Interface (SCI) – asynchronous serial port
- General-Purpose bi-directional digital I/O (GPIO) pins
- Watchdog Timer ("time-out" DSP reset device for system integrity)

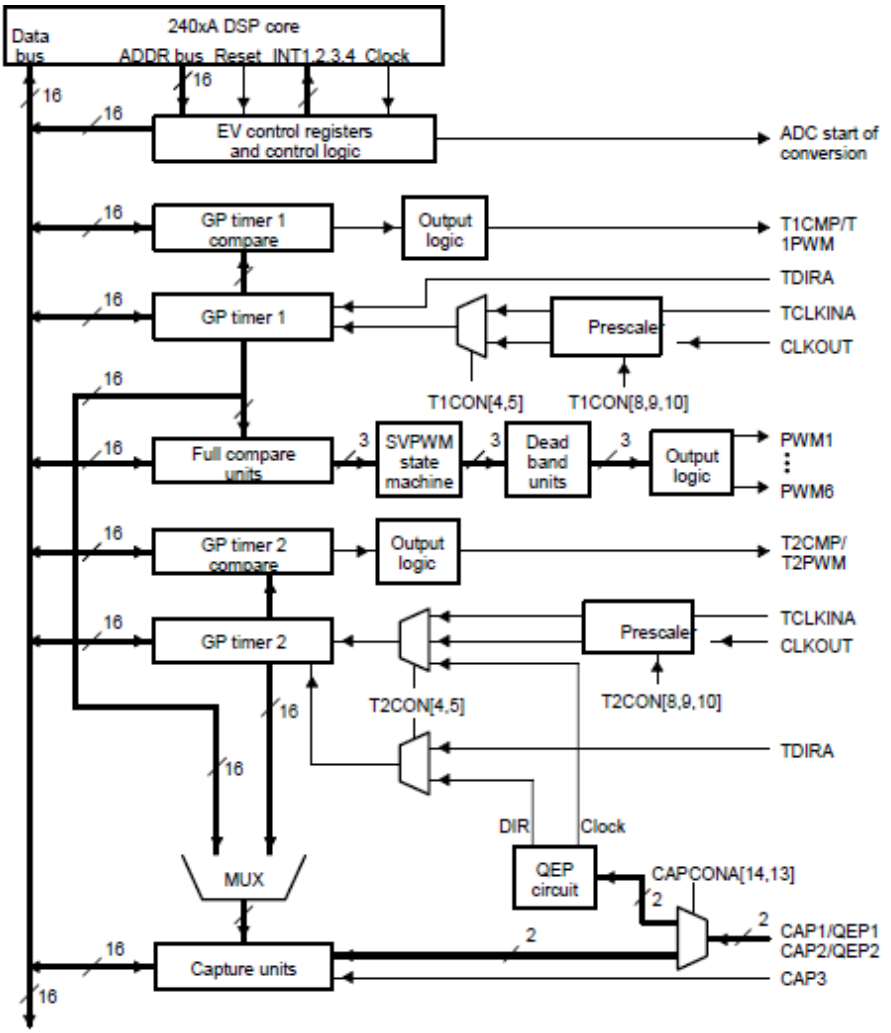


Figure 6.1 Event Manager A (EVA) block diagram. (Courtesy of Texas Instruments)

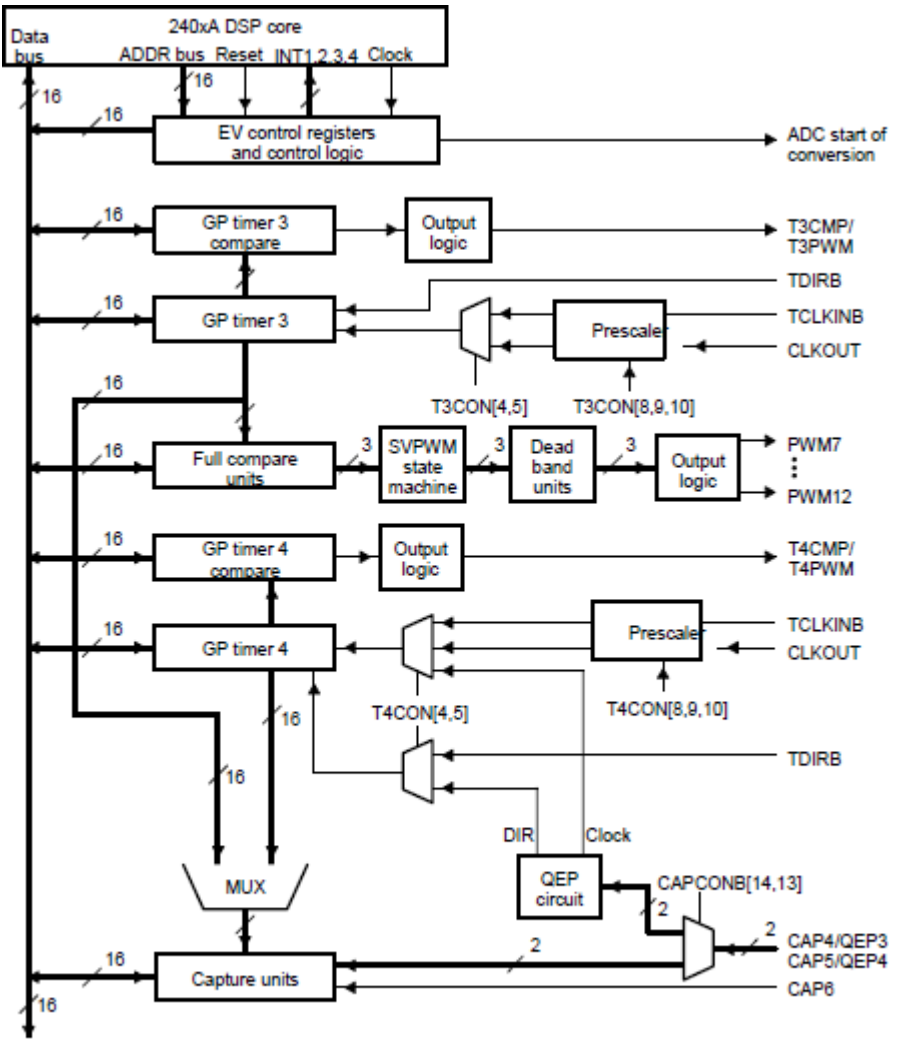


Figure 6.2 Event Manager B (EVB) block diagram. (Courtesy of Texas Instruments)

Brief Introduction to Peripherals

The following peripherals are those that are integrated onto the LF2407 chip. Refer to Fig. 1.1 to view the pin-out associated with each peripheral.

Event Managers (EVA, EVB)

There are two Event Managers on the LF2407, the EVA and EVB. The Event Manager is the most important peripheral in digital motor control. It contains the necessary functions needed to control electromechanical devices. Each EV is composed of functional "blocks" including timers, comparators, capture units for triggering on an event, PWM logic circuits, quadrature-encoder–pulse (QEP) circuits, and interrupt logic.

The Analog-to-Digital Converter (ADC)

The ADC on the LF2407 is used whenever an external analog signal needs to be sampled and converted to a digital number. Examples of ADC applications range from sampling a control signal for use in a digital notch filtering algorithm or using the ADC in a control feedback loop to monitor motor performance. Additionally, the ADC is useful in motor control applications because it allows for current sensing using a shunt resistor instead of an expensive current sensor.

The Control Area Network (CAN) Module

While the CAN module will not be covered in this text, it is a useful peripheral for specific applications of the LF2407. The CAN module is used for multi-master serial communication between external hardware. The CAN bus has a high level of data integrity and is ideal for operation in noisy environments such as in an automobile, or industrial environments that require reliable communication and data integrity.

Serial Peripheral Interface (SPI) and Serial Communications Interface (SCI)

The SPI is a high-speed synchronous communication port that is mainly used for communicating between the DSP and external peripherals or another DSP device. Typical uses of the SPI include communication with external shift registers, display drivers, or ADCs.

The SCI is an asynchronous communication port that supports asynchronous serial (UART) digital communication between the CPU and other asynchronous peripherals that use the standard NRZ (non-return-to-zero) format. It is useful in communication between external devices and the DSP. Since these communication peripherals are not directly related to motion control applications, they will not be discussed further in this text.

Watchdog Timer (WD)

The Watchdog timer (WD) peripheral monitors software and hardware operations and asserts a system reset when its internal counter overflows. The WD timer (when enabled) will count for a specific amount of time. It is necessary for the user's software to reset the WD timer periodically so that an unwanted reset does not occur. If for some reason there is a CPU disruption, the watchdog will generate a system reset. For example, if the software enters an endless loop or if the CPU becomes temporarily disrupted, the WD timer will overflow and a DSP reset will occur, which will cause the DSP program to branch to its initial starting point. Most error conditions that temporarily disrupt chip operation and inhibit proper CPU function can be cleared by the WD function. In this way, the WD increases the reliability of the CPU, thus ensuring system integrity.

General Purpose Bi-Directional Digital I/O (GPIO) Pins

Since there are only a finite number of pins available on the LF2407 device, many of the pins are multiplexed to either their primary function or the secondary GPIO function. In most cases, a pin's second function will be as a general-purpose input/output pin. The GPIO capability of the LF2407 is very useful as a means of controlling the functionality of pins and also provides another method to input or output data to and from the device. Nine 16-bit control registers control all I/O and shared pins. There are two types of these registers:

- I/O MUX Control Registers (MCRx) – Used to control the multiplexer selection that chooses between the primary function of a pin or the general-purpose I/O function.
- Data and Direction Control Registers (PxDATDIR) – Used to control the data and data direction of bi-directional I/O pins.

Joint Test Action Group (JTAG) Port

The JTAG port provides a standard method of interfacing a personal computer with the DSP controller for emulation and development. The XDS510PP or equivalent emulator pod provides the connection between the JTAG module on the LF2407 and the personal computer. The JTAG module allows the PC to take full control over the DSP processor while Code Composer Studio™ is running. Figure 1.2 shows the connection scheme from computer to the DSP board.

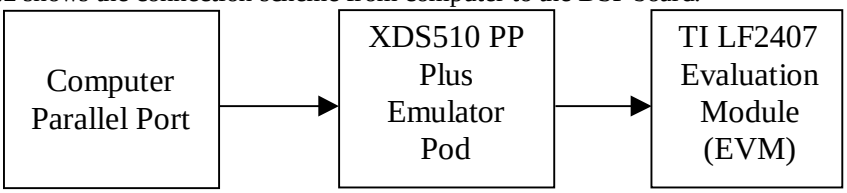


Figure 1.2 PC to DSP connection scheme.

Phase Locked Loop (PLL) Clock Module

The phase locked loop (PLL) module is basically an input clock multiplier that allows the user to control the input clocking frequency to the DSP core. External to the LF2407, a clock reference (can oscillator/crystal) is generated. This signal is fed into the LF2407 and is multiplied or divided by the PLL. This new (higher or lower frequency) clock signal is then used to clock the DSP core. The LF2407's PLL allows the user to select a multiplication factor ranging from 0.5X to 4X that of the external clock signal. The default value of the PLL is 4X.

Memory Allocation Spaces

The LF2407 DSP Controller has three different allocations of memory it can use: Data, Program, and I/O memory space. Data space is used for program calculations, look-up tables, and any other memory used by an algorithm. Data memory can be in the form of the on-chip random access memory (RAM) or external RAM. Program memory is the location of user's program code. Program memory on the LF2407 is either mapped to the off-chip RAM (MP/MC- pin =1) or to the on-chip flash memory (MP/MC- = 0), depending on the logic value of the MP/MC-pin.

I/O space is not really memory but a virtual memory address used to output data to peripherals external to the LF2407. For example, the digital-to-analog converter (DAC) on the Spectrum Digital™ evaluation module is accessed with I/O memory. If one desires to output data to the DAC, the data is simply sent to the configured address of I/O space with the "OUT" command. This process is similar to writing to data memory except that the OUT command is used and the data is copied to and outputted on the DAC instead of being stored in memory.

3. Types of Physical Memory

Random Access Memory (RAM)

The LF2407 has 544 words of 16 bits each in the on-chip DARAM. These 544 words are partitioned into three blocks: B0, B1, and B2. Blocks B1 and B2 are allocated for use only as data memory. Memory block B0 is different than B1 and B2. This memory block is normally configured as Data Memory, and hence primarily used to hold data, but in the case of the B0 block, it can also be configured as Program Memory. B0 memory can be configured as program or data memory depending on the value of the core level "CNF" bit.

- (CNF=0) maps B0 to data memory.
- (CNF=1) maps B0 to program memory.

The LF2407 also has 2K of single-access RAM (SARAM). The addresses associated with the SARAM can be used for both data memory and program memory, and are software configurable to the internal SARAM or external memory.

Non-Volatile Flash Memory

The LF2407 contains 32K of on-chip flash memory that can be mapped to program space if the MP/MC-pin is made logic 0 (tied to ground). The flash memory provides a permanent location to store code that is unaffected by cutting power to the device. The flash memory can be electronically programmed and erased many times to allow for code development. Usually, the external RAM on the LF2407 Evaluation Module (EVM) board is used instead of the flash for code development due to the fact that a separate "flash programming" routine must be performed to flash code into the flash memory. The on-chip flash is normally used in situations where the DSP program needs to be tested where a JTAG connection is not practical or where the DSP needs to be tested as a "stand-alone" device. For example, if a LF2407 was used to develop a DSP control solution to an automobile braking system, it would be somewhat impractical to have a DSP/JTAG/PC interface in a car that is undergoing performance testing.

4. Software Tools

Texas Instrument's Code Composer Studio™ (CCS) is a user-friendly Windows-based debugger for developing and debugging software for the LF2407. CCS allows users to write and debug code in C or in TI assembly language. CCS has many features that can aid in developing code. CCS features include:

- User-friendly Windows environment
- Ability to use code written in C and assembly
- Memory displays and on-the-fly editing capability
- Disassembly window for debugging
- Source level debugging, which allows stepping through and setting breakpoints in original source code
- CPU register visibility and modification
- Real-time debugging with watch windows and continuousrefresh
- Various single step/step over/ step-into command icons
- Ability to display data in graph formats
- General Extension Language (GEL) capability, allows the user to create functions that extend the usefulness of CCS™

1.4.1 Becoming Aquatinted with Code Composer Studio(CCS)

This exercise will help you become familiar with the software and emulation tools of the LF2407 DSP Controller. CCS™, the current emulation and debugging software, is user-friendly and a powerful development tool.

The hardware required for this exercise and all others is the Spectrum Digital TMS320LF2407 EVM package, which includes LF2407 EVM board and the XDS510PP Plus JTAG emulator pole. You will also need a Windows-based

Introduction to the C2xx DSP Core and Code Generation

The heart of the LF2407 DSP Controller is the C2xx DSP core. This core is a 16-bit fixed point processor, meaning that it works with 16-bit binary numbers. One can think of the C2xx as the central processor in a personal computer. The LF2407 DSP consists of the C2xx DSP core plus many peripherals such as Event Managers, ADC, etc., all integrated onto one single chip. This chapter will discuss the C2xx DSP core, subcomponents, and instruction set.

The C2xx core has its own native instruction set of assembly mnemonics or commands. Through the use of CCS and the associated compiler, one has the freedom of writing code in both C language and the native assembly language. However, to write compact, fast executing programs, it is best to compose code in assembly language. Due to this reason, programming in assembly will be the focus of this book. However, we will also include an example of a software tool called VisSim™, by Visual Solutions. VisSim allows users to simulate algorithms and develop code in "block" form. More on VisSim will be presented in the Appendix.

1. The Components of the C2xx DSP Core

The DSP core (like all microprocessors) consists of several subcomponents necessary to perform arithmetic operations on 16-bit binary numbers. The following is a list of the multiple subcomponents found in the C2xx core which we will discuss further:

- A 32-bit central arithmetic logic unit (CALU)
- A 32-bit accumulator (used frequently in programs)
- Input and output data-scaling shifters for the CALU
- A (16-bit by 16-bit) multiplier
- A product-scaling shifter
- Eight auxiliary registers (AR0 – AR7) and an auxiliary register arithmetic unit (ARAU)

Each of the above components is either accessed directly by the user code or is indirectly used during the execution of an assembly command.

Central Arithmetic Logic Unit (CALU)

The C2xx performs 2s-complement arithmetic using the 32-bit CALU. The CALU uses 16-bit words taken from data memory, derived from an immediate instruction, or from the 32-bit multiplier result. In addition to arithmetic operations, the CALU can perform Boolean operations. The CALU is somewhat transparent to

the user. For example, if an arithmetic command is used, the user only needs to write the command and later read the output from the appropriate register. In this sense, the CALU is “transparent” in that it is not accessed directly by the user.

Accumulator

The accumulator stores the output from the CALU and also serves as another input to the CALU (many arithmetic commands perform operations on numbers that are currently stored in the accumulator; versus other memory locations). The accumulator is 32 bits wide and is divided into two sections, each consisting of 16 bits. The high-order bits consist of bits 31 through 16, and the low-order bits are made up of bits 15 through 0. Assembly language instructions are provided for storing the high- and low-order accumulator words to data memory. In most cases, the accumulator is written to and read from directly by the user code via assembly commands. In some instances, the accumulator is also transparent to the user (similar to the CALU operation in that it is accessed “behind the scenes”).

Scaling Shifters

The C2xx has three 32-bit shifters that allow for scaling, bit extraction, extended arithmetic, and overflow-prevention operations. The scaling shifters make possible commands that shift data left or right. Like the CALU, the operation of the scaling shifters is “transparent” to the user. For example, the user needs only to use a shift command, and observe the result. Any one of the three shifters could be used by the C2xx depending on the specific instruction entered. The following is a description of the three shifters:

- **Input data-scaling shifter (input shifter):** This shifter left-shifts 16-bit input data by 0 to 16 bits to align the data to the 32-bit input of the CALU. For example, when the user uses a command such as “ADD 300h, 5”, the input shifter is responsible for first shifting the data in memory address “300h” to the left by five places before it is added to the contents of the accumulator.

- **Output data-scaling shifter (output shifter):** This shifter left-shifts data from the accumulator by 0 to 7 bits before the output is stored to data memory. The content of the accumulator remains unchanged. For example, when the user uses a command such as “SACL 300h, 4”, the output shifter is responsible for first shifting the contents of the accumulator to the left by four places before it is stored to the memory address “300h”.

- **Product-scaling shifter (product shifter)**: The product register (PREG) receives the output of the multiplier. The product shifter shifts the output of the PREG before that output is sent to the input of the CALU. The product shifter has four product shift modes (no shift, left shift by one bit, left shift by four bits, and right shift by six bits), which are useful for performing multiply/accumulate operations, fractional arithmetic, or justifying fractional products.

Multiplier

The multiplier performs 16-bit, 2s-complement multiplication and creates a 32-bit result. In conjunction with the multiplier, the C2xx uses the 16-bit temporary register (TREG) and the 32-bit product register (PREG).

The operation of the multiplier is not as "transparent" as the CALU or shifters. The TREG **always** needs to be loaded with one of the numbers that are to be multiplied. Other than this prerequisite, the multiplication commands do not require any more actions from the user code. The output of the multiply is stored in the PREG, which can later be read by the user code.

Auxiliary Register Arithmetic Unit (ARAU) and Auxiliary Registers

The ARAU generates data memory addresses when an instruction uses indirect addressing to access data memory (more on indirect addressing will be covered later along with assembly programming). Eight auxiliary registers (AR0 through AR7) support the ARAU, each of which can be loaded with a 16-bit value from data memory or directly from an instruction. Each auxiliary register value can also be stored in data memory. The auxiliary registers are mainly used as "pointers" to data memory locations to more easily facilitate looping or repeating algorithms. They are directly written to by the user code and are automatically incremented or decremented by particular assembly instructions during a looping or repeating operation. The auxiliary register pointer (ARP) embedded in status register ST0 references the auxiliary register. The status registers (ST0, ST1) are core level registers where values such as the Data Page (DP) and ARP located. More on the operation and use of auxiliary registers will be covered in subsequent chapters.

Mapping External Devices to the C2xx Core and the Peripheral Interface

Since the LF2407 contains many peripherals that need to be accessed by the C2xx core, the C2xx needs a way to read and write to the different peripherals. To make this possible, peripherals are mapped to data memory (memory will be covered shortly). Each peripheral is mapped to a corresponding block of data memory addresses. Where applicable, each corresponding block contains configuration registers, input registers, output registers, and status registers. Each peripheral is accessed by simply writing to the appropriate registers in data memory, provided the peripheral clock is enabled(see System Configuration registers).

The peripherals are linked to the internal memory interface of the CPU through the PBUS interface shown in Fig. 2.1. All on-chip peripherals are accessed through the Peripheral Bus (PBUS). All peripherals, excluding the WD timer counter, are clocked by the CPU clock (which has a selectable frequency), and must be enabled via the system configuration registers.

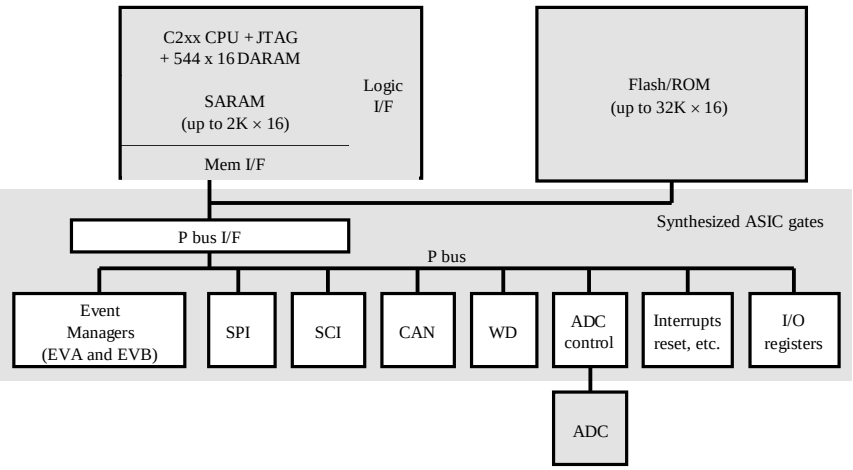


Figure 2.1 Functional block diagram of the LF2407 DSP controller.

System Configuration Registers

The System Control and Status Registers (SCSR1, SCSR2) are used to configure or display fundamental settings of the LF2407. For example, these fundamental settings include the clock speed (clock pre-scale setting) of the LF2407, which peripherals are enabled, microprocessor/microcontroller mode, etc. Bits are controlled by writing to the corresponding data memory address or the logic level on an external pin as with the microprocessor/microcontroller (*MP/MC)* select bit. The bit descriptions of these two registers (mapped to data memory) are listed below.

System Control and Status Register 1 (SCSR1) — Address 07018h

15	14	13	12	11	10	9	8
Reserved	CLKSRC	LPM1	LPM0	CLK PS2	CLK PS1	CLK PS0	Reserved
R–0	RW–0	RW–0	RW–0	RW–1	RW–1	RW–1	R–0
7	6	5	4	3	2	1	0
ADC CLKEN	SCI CLKEN	SPI CLKEN	CAN CLKEN	EVB CLKEN	EVA CLKEN	Reserved	ILLADR
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	R–0	RC–0

Note: *R = read access, W = write access, C = clear, -0 = value after reset.*

Bit 15 **Reserved**

Bit 14 **CLKSRC.** CLKOUT pin source select

0 CLKOUT pin has CPU Clock (40 MHz on a 40-MHz device) as the output

1 CLKOUT pin has Watchdog clock as the output

Bits 13–12 **LPM** (1:0). Low-power mode select

These bits indicate which low-power mode is entered when the CPU executes the IDLE instruction.

Description of the low-power modes:

LPM(1:0)	Low-Power mode selected
00	IDLE1 (LPM0)
01	IDLE2. (LPM1)
1x	HALT (LPM2)

Bits 11–9 **PLL Clock prescale select.** These bits select the PLL multiplication factor for the input clock.

CLK PS2	CLK PS1	CLK PS0	System Clock Frequency
0	0	0	4 x F_{in}
0	0	1	2 x F_{in}
0	1	0	1.33 x F_{in}
0	1	1	1 x F_{in}
1	0	0	0.8 x F_{in}
1	0	1	0.66 x F_{in}
1	1	0	0.57 x F_{in}
1	1	1	0.5 x F_{in}

Note: F_{in} is the input clock frequency.

Bit 8 Reserved

Bit 7 **ADC CLKEN.** ADC module clock enable control bit

0 Clock to module is disabled (i.e., shut down to conserve power)

1 Clock to module is enabled and running normally

Bit 6 **SCI CLKEN.** SCI module clock enable control bit

0 Clock to module is disabled (i.e., shut down to conserve power)

1 Clock to module is enabled and running normally

Bit 5 **SPI CLKEN.** SPI module clock enable control bit

0 Clock to module is disabled (i.e., shut down to conserve power)

1 Clock to module is enabled and running normally

Bit 4 **CAN CLKEN.** CAN module clock enable control bit

0 Clock to module is disabled (i.e., shut down to conserve power)

1 Clock to module is enabled and running normally

Bit 3 **EVB CLKEN.** EVB module clock enable control bit

0 Clock to module is disabled (i.e., shut down to conserve power)

1 Clock to module is enabled and running normally

Bit 2 **EVA CLKEN.** EVA module clock enable control bit

0 Clock to module is disabled (i.e., shut down to conserve power)

1 Clock to module is enabled and running normally

***Note**: In order to modify/read the register contents of any peripheral, the clock to that peripheral must be enabled by writing a 1 to the appropriate bit.*

Bit 1 Reserved

Bit 0 ILLADR. Illegal Address detect bit

If an illegal address has occurred, this bit will be set. It is up to software to clear this bit following an illegal address detect. This bit is cleared by writing a 1 to it and should be cleared as part of the initialization sequence. Note: An illegal address will cause a Non-Maskable Interrupt (NMI).

System Control and Status Register 2 (SCSR2) — Address 07019h

15-8							
Reserved							
RW–0							
7	6	5	4	3	2	1	0
Reserved	I/P QUAL	WD OVERRIDE	XMIF HI–Z	BOOT EN	MP/MC	DON	PON
	RW–0	RC–1	RW–0	RW–BOOT EN pin	RW–MP/MC pin	RW–1	RW–1

***Note:** R = read access, W = write access, C = clear, -0 = value after reset.*

Bits 15–7 **Reserved.** Writes have no effect; reads are undefined

Bit 6 Input Qualifier Clocks.

An input-qualifier circuitry qualifies the input signal to the CAP1–6, XINT1/2, ADCSOC, and PDPINTA/B pins in the 240xA devices. The I/O functions of these pins do not use the input-qualifier circuitry. The state of the internal input signal will change only after the pin is held high/low for 6 (or 12) clock edges. This ensures that a glitch smaller than (or equal to) 5 (or 11) CLKOUT cycles wide will not change the internal pin input state. The user must hold the pin high/low for 6 (or 12) cycles to ensure that the device will see the level change. This bit determines the width of the glitches (in number of internal clock cycles) that will be blocked. Note that the internal clock is not the same as CLKOUT, although its frequency is the same as CLKOUT.

0 The input-qualifier circuitry blocks glitches up to 5 clock cycles long

1 The input-qualifier circuitry blocks glitches up to 11 clock cycles long

Note: *This bit is applicable only for the 240xA devices, not for the 240x devices because they lack an input-qualifier circuitry.*

Bit 5 Watchdog Override. (WD protect bit)

After RESET, this bit gives the user the ability to disable the WD function through software (by setting the WDDIS bit = 1 in the WDCR). This bit is a clear-only bit and defaults to a 1 after reset.

Note: *This bit is cleared by writing a 1 to it.*

0 Protects the WD from being disabled by software. This bit cannot be set to 1 by software. It is a clear-only bit, cleared by writing a 1

1 This is the default reset value and allows the user to disable the WD through the WDDIS bit in the WDCR. Once cleared, however, this bit can no longer be set to 1 by software, thereby protecting the integrity of the WD timer

Bit 4 XMIF Hi-Z Control

This bit controls the state of the external memory interface (XMIF) signals.

0 XMIF signals in normal driven mode; i.e., not Hi-Z (high impedance)

1 All XMIF signals are forced to Hi-Z state

Bit 3 Boot Enable

This bit reflects the state of the BOOT_EN / XF pin at the time of reset. After reset and device has "booted up", this bit can be changed in software to re-enable Flash memory visibility or return to active Boot ROM.

0 Enable Boot ROM — Address space 0000 — 00FF is now occupied by the on-chip Boot ROM Block. Flash memory is totally disabled in this mode. Note: There is no on-chip boot ROM in ROM devices (i.e., LC240xA)

1 Disable Boot ROM — Program address space 0000 — 7FFF is mapped to on-chip Flash memory in the case of LF2407A and LF2406A. In the case of LF2402A, addresses 0000 – 1FFF are mapped

Bit 2 Microprocessor/Microcontroller Select

This bit reflects the state of the MP/MC pin at time of reset. After reset, this bit can be changed in software to allow dynamic mapping of memory on and off chip.

0 Set to Microcontroller mode — Program Address range 0000 — 7FFF is mapped internally (i.e., Flash)

1 Set to Microprocessor mode — Program Address range 0000 — 7FFF is mapped externally (i.e., customer provides external memory device.)

Bits 1–0 SARAM Program/Data Space Select

DON	PON	SARAM status
0	0	SARAM not mapped (disabled), address space allocated to external memory
0	1	SARAM mapped internally to Programspace
1	0	SARAM mapped internally to Data space
1	1	SARAM block mapped internally to both Data and Program spaces. This is the default or reset value

Note: *See memory map for location of SARAM addresses*

Memory

Memory is required to hold programs, perform operations, and execute programming instructions. There are three main blocks of memory which are present on the LF2407 chip: B0, B1, and B2. Additionally, there are two different memory "spaces" (program, data) in which blocks are used. We will discuss exactly what each memory "block" and memory "space" is, and what each is used for.

1. *Memory Blocks and Types*

A block of memory on the LF2407 is simply a specified range of memory addresses (each address consists of a 16-bit word of memory). There are three main memory blocks on the LF2407 that can be specified via the *Linker Command File* (we will discuss the *Linker Command File* and other files types when we cover programming).

The LF2407 has 544 16-bit words of on-chip Double Access Random Access Memory (DARAM) that are divided into three main memory blocks named B0, B1, and B2. In addition to the DARAM, there are also 2000 16-bit words of Single Access Random Access Memory (SARAM). The main difference between DARAM and SARAM is that DARAM memory can be accessed twice per clock cycle and SARAM can only be accessed once per cycle. Thus, DARAM reads and writes twice as fast as SARAM.

In addition to the RAM present on the LF2407, there is also non-volatile Flash memory. Unlike RAM, the Flash memory does not lose its contents when the LF2407 loses power. Flash memory can only be written to by "flashing" the memory, which is a process that can only be done manually by a user. Therefore, Flash memory on the LF2407 is used only to store a program that is to be run. As stated in Chapter 1, it is only necessary to use the Flash memory if the DSP is to be run independently from a PC and JTAG interface. Though we introduce Flash memory, it will not be covered in this text. However, the reader is encouraged to consult the Texas Instruments documentation on Flash memory. Flash memory can prove to be a valuable code development tool when it comes time to test a LF2407 program where having a PC connected is impractical.

2. *Memory Space and Allocation*

There are two ways of using the physical memory on board the LF2407: storing a program or storing data.

A program that is to be run must be stored in memory that is mapped to program space. Likewise, only memory that is in data space may be used to store data. Program memory is written to when a program is loaded into the LF2407. Data memory is normally written to during the execution of a program, where the program might use the data memory as temporary storage for calculation variables and results.

Memory blocks B1 and B2 are configured as data memory. The B0 block is primarily intended to hold data, but can be configured to act as either program *or* data memory, depending on the value of the CNF bit in Status Register ST1. CNF = 0 maps B0 in data memory, while CNF = 1 maps B0 in program memory.

The memory addresses associated with the SARAM can be configured for both data memory and program memory, and are also software configurable to either

access external memory or the internal SARAM. When configured for internal, the SARAM can be used as data or program memory. However, when configured as external, these addresses are used for off-chip program memory. SARAM is useful if more memory is needed for data than the B0, B1, and B2 blocks can provide. The SARAM addresses should be configured to either program or data space via the *Linker Command File*.

The on-chip flash in the LF2407 is mapped to program memory space when the external MP/MC-pin is pulled low. When the MP/MC-pin is pulled high, the program memory is mapped to external memory addresses, access via memory that is physically external to the LF2407. In the case of the Spectrum Digital EVM, external memory is installed on the board and a jumper pulls the MP/MC pin high or low.

2.5.3 Memory Maps

Program Memory

When a program is loaded into the LF2407, the code resides in and is run from program memory space. In addition to storing the user code, the program memory can also store immediate operands and table information. Figure 2.2 shows the various program memory addresses (in hexadecimal) and how they are used.

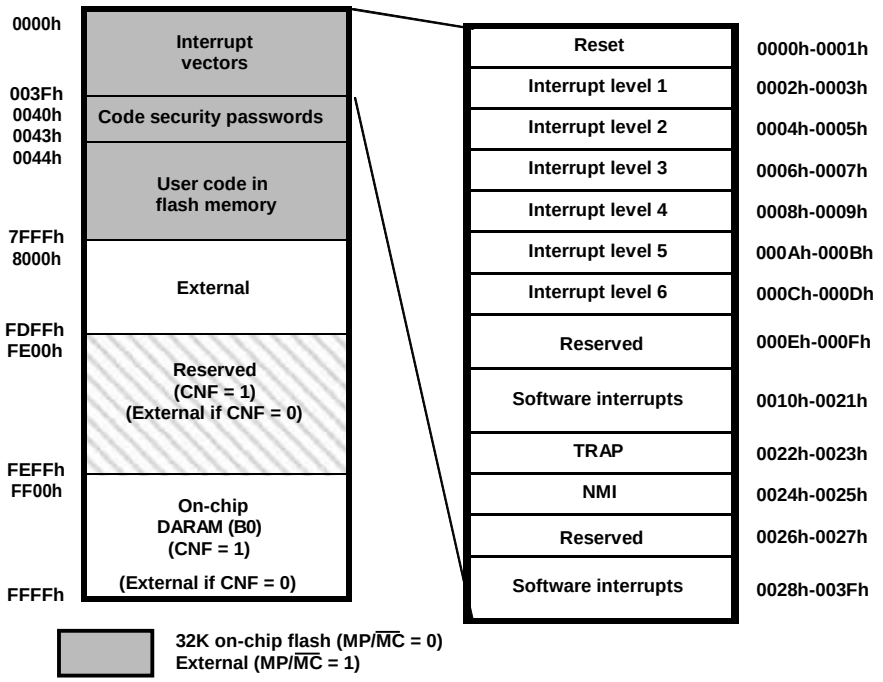


Figure 2.2 Program memory map for LF2407. (Courtesy of Texas Instruments)

Two factors determine the configuration of program memory:

CNF bit:

The CNF bit determines if B0 memory is in on-chip program space:

CNF = 0. The 256 words are mapped as external memory.

CNF = 1. The 256 words of DARAM B0 are configured for program use. At reset, B0 is mapped to data space (CNF = 0).

MP/MC pin:

The level on the MP/MC pin determines if program instructions are read from on-chip Flash/ROM or external memory:

MP/MC = 0. The device is configured in microcontroller mode. The on-chip flash EEPROM is accessible. The device fetches the reset vector from on-chip memory.

MP/MC = 1. The device is configured in microprocessor mode. Program memory is mapped to external memory.

Data Memory

For the execution of a program, it is necessary to store calculation results or look up tables in memory. The memory allocated for this function is called data memory. In order to store a value to a data memory address (*dma*), the corresponding memory block must reside in data memory space. Blocks B1 and B2 discussed earlier permanently reside in data space, while block B0 and the SARAM are configurable for either program or data.

Data memory space has the second functionality of providing an easy way to access on-chip configuration registers and peripherals. Each user configurable peripheral has associated registers in data memory addresses that may be written to or read from as needed. For example, the control registers for the analog-to-digital converter (ADC) are each located in the data memory range of **70A0**h to **70BF**h. The internal data memory includes the memory-mapped registers, DARAM blocks, and peripheral memory-mapped registers. The remaining 32K words of memory (8000h to FFFFh) form part of the external data memory.

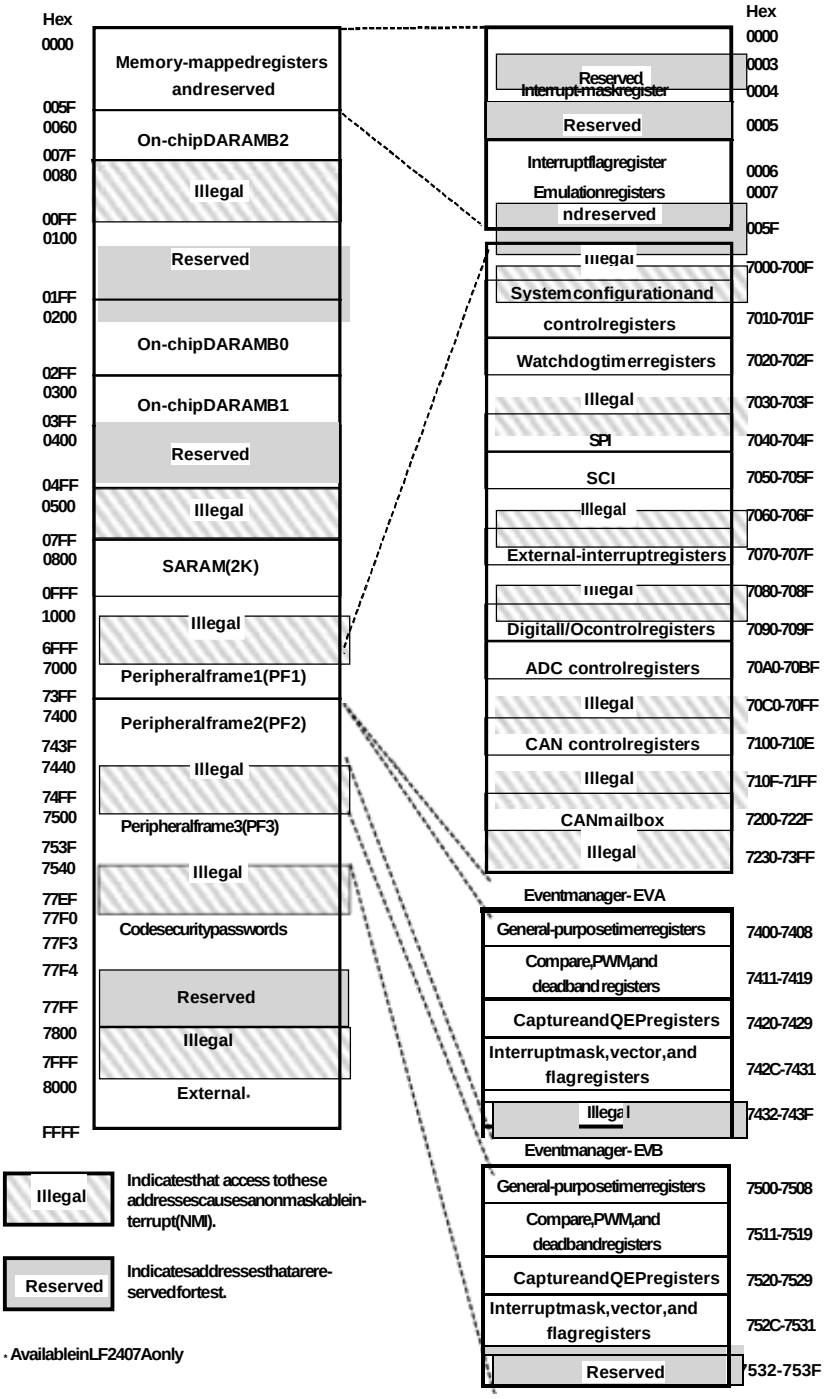


Figure 2.3 Data memory map for the LF2407. (Courtesy of Texas Instruments)

Input/Output (I/O) Space

I/O space is solely used for accessing external peripherals such as the digital-to-analog converter (DAC) on the LF2407 EVM. ***It is not to be confused with the I/O functionality of pins.*** The assembly instruction "*OUT*" is used to write to an address that is mapped to I/O space. Figure 2.4 depicts the basic memory map of the I/O space on the LF2407.

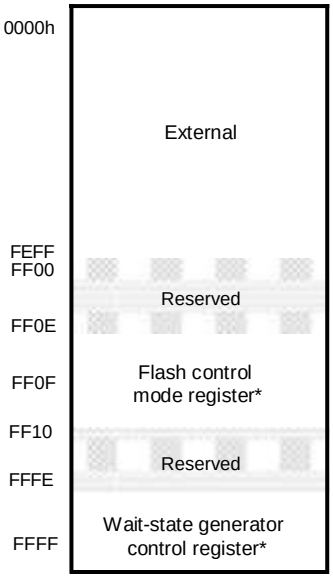


Figure 2.4 Memory map of I/O space. (Courtesy of Texas Instruments)

Within program, data, and I/O space are addresses that are reserved for system functionality and may not be written to. It is important that the user pay attention to what memory ranges are used by the program and where the program is to be loaded. It is important to make sure the *Linker Command File* is configured properly and the correct Data Page (DP) is set to avoid inadvertently writing to an undesired or reserved memory address.

Detailed information on the memory map is given in the Texas Instruments *TMS320LF/LC240xA DSP Controllers Reference Guide - System and Peripherals; Literature Number: SPRU357A.*

6. Memory Addressing Modes

There are three basic memory addressing modes used by the C2xx instruction set. The three modes are:

- Immediate addressing mode (does not actually access memory)
- Direct addressing mode
- Indirect addressing mode

Immediate Addressing Mode

In the immediate addressing mode, the instruction contains a constant to be manipulated by the instruction. Even though the name "immediate addressing" suggests that a memory location is accessed, immediate addressing is simply dealing with a user-specified constant which is usually included in the assembly command syntax. The "#" sign indicates that the value is an immediate address (just a constant). The two types of immediate addressing modes are:

Short-immediate addressing. The instructions that use short-immediate addressing have an 8-bit, 9-bit, or 13-bit constant as the operand.

For example, the instruction:

```
LACL    #44h        ;loads lower bits of accumulator with
                    ;eight-bit constant (44h in this case)
```

Note: *The LACL command will work only with a short 8-bit constant. If you want to load a long 16-bit constant, then use the LACC command.*

Long-immediate addressing. Instructions that use long-immediate addressing have a 16-bit constant as an operand. This 16-bit value can be used as an absolute constant or as a 2s-complement value.

For example, the instruction:

```
LACC    #4444h      ;loads accumulator with up to a 16-bit
                    ;constant (4444h in this case)
```

If you need to use registers or access locations in data memory, you must use either direct or indirect addressing.

Direct Addressing Mode

In direct addressing, data memory is first addressed in blocks of 128 words called data pages. The entire 64K of data memory consists of 512 DPs labeled 0 through 511, as shown in the Fig. 2.5. The current DP is determined by the value in the 9-bit DP pointer in status register ST0. For example, if the DP value is "0 0000 0000", the current DP is 0. If the DP value is "0 0000 0010", the current data page is 2. The DP of a particular memory address can be found easily by dividing the address (in hexadecimal) by 80h. For example:

For the data memory address 0300h, 300h/80h = 6h so the DP pointer is 6h. Likewise, the DP pointer for 200h is 4h.

DP Value	Offset	Data Memory
0000 0000 0 . . 0000 0000 0	000 0000 . . 111 1111	Page 0: 0000h-007Fh
0000 0000 1 . . 0000 0000 1	000 0000 . . 111 1111	Page 1: 0080h-00FFh
0000 0001 0 . . 0000 0001 0	000 0000 . . 111 1111	Page 2: 0100h-017Fh .
. . . . . .	. . . . . .	. . . . . .
1111 1111 1 . . 1111 1111 1	000 0000 . . 111 1111	Page 511: FF80h-FFFFh

Figure 2.5 Data pages and corresponding memory ranges. (Courtesy of Texas Instruments)

In addition to the DP, the DSP must know the particular word being referenced on that page. This is determined by a 7-bit offset. The 7-bit offset is simply the 7 least significant bits (LSBs) of the memory address. The DP and the offset make up the 16-bit memory address (see Fig. 2.6).

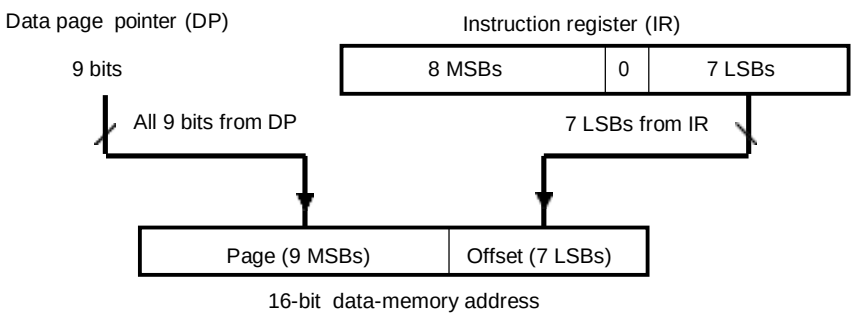


Figure 2.6 Data page and offset make up a 16-bit memory address.

When you use direct addressing, the processor uses the 9 DP bits and the 7 LSBs of the instruction to obtain the true memory address. The following steps should be followed when using direct addressing:

1. Set the DP. Load the appropriate value (from 0 to 511 in decimal or 0-1FF in hex) into the DP. The easiest way to do this is with the LDP instruction. The LDP instruction loads the DP directly to the ST0 register without affecting any other bits of the ST0.

```
LDP     #0E1h       ;sets the data page pointer to E1h
```

or

```
LDP     #225        ;sets the data page pointer to 225 decimal
                    ;which is E1 in hexadecimal
```

2. Specify the offset. For example, if you want the ADD instruction to use the value at the second address of the current data page, you would write: ADD 1h

If the data page points to 300h, then the above instruction will add the contents of 301h to the accumulator

Note: *You do not have to set the data page prior to every instruction that uses direct addressing. If all the instructions in a block of code access the same data page, you can simply load the DP before the block. However, if various data pages are being accessed throughout the block of code, be sure the DP is changed accordingly.*

Indirect Addressing Mode

Indirect addressing is a powerful way of addressing data memory. Indirect addressing mode is not dependent on the current data page as is direct addressing. Instead, when using indirect addressing you load the memory space that you would like to access into one of the auxiliary registers (ARx). The current auxiliary register acts as a pointer that points to a specific memory address.

The register pointed to by the ARP is referred to as the current auxiliary register or current AR. To select a specific auxiliary register, load the 3-bit auxiliary register pointer (ARP) with a value from 0 to 7. The ARP can be loaded with the MAR instruction or by the LARP instruction. An ARP value can also be loaded by using the ARx operand after any instruction that supports indirect addressing as seen below.

Example of using MAR:

```
ADD     * , AR1     ;Adds using current * , then makes AR1 the
                    ;new current AR for future uses
```

Example of using LARP

```
LARP    #2          ;this will make AR2 the current AR
```

The C2xx provides four types of indirect addressing options:

- **No increment or decrement.** The instruction uses the content of the current auxiliary register as the data memory address but neither increments nor decrements the content of the current auxiliary register.
- **Increment or decrement by 1.** The instruction uses the content of the current auxiliary register as the data memory address and then increments or decrements the content of the current auxiliary register by one.

- **Increment or decrement by an index amount.** The value in AR0 is the index amount. The instruction uses the content of the current auxiliary register as the data memory address and then increments or decrements the content of the current auxiliary register by the index amount.

- **Increment or decrement by an index amount using reverse carry.** The value in AR0 is the index amount. After the instruction uses the content of the current auxiliary register as the data memory address, that content is incremented or decremented by the index amount. The addition and subtraction process is accomplished with the carry propagation reversed and is useful in fast Fourier transforms algorithms.

Table 2.1 displays the various operands that are available for use with instructions while using indirect addressing mode.

Table 2.1 Indirect addressing operands.

Operand	Option	Example
*	No increment or decrement	**LT *** loads the temporary register TREG with the content of the data memory address referenced by the current AR.
*+	Increment by 1	**LT *+** loads the TREG with the content of the data memory address referenced by the current AR and then adds 1 to the content of the current AR.
*-	Decrement by 1	**LT *-** loads the TREG with the content of the data memory address referenced by the current AR and then subtracts 1 from the content of the current AR.
*0+	Increment by index amount	**LT *0+** loads the TREG with the content of the data memory address referenced by the current AR and then adds the content of AR0 to the content of the current AR.
*0-	Decrement by index amount	**LT *0-** loads the TREG with the content of the data memory address referenced by the current AR and then subtracts the content of AR0 from the content of the current AR.
*BR0+	Increment by index amount, adding with reverse carry	**LT *BR0+** loads the TREG with the content of the data memory address referenced by the current AR and then adds the content of AR0 to the content of the current AR, adding with reverse carry propagation.
*BR0-	Decrement by index amount, subtracting with reverse carry	**LT *BR0-** loads the TREG with the content of the data memory address referenced by the current AR and then subtracts the content of AR0 from the content of the current AR, subtracting with bit reverse carry propagation.

Example: **LACC** *dma* , you can use several way to address the *dma* (data memory address).

```
LACC    *
```

or

```
LACC    200h
```

or

```
LACC    v           ; where "v" is any variable assigned to data
                    memory
```

where *, 200h, and v are the data memory addresses

Boldface Characters Boldface characters must be included in the syntax.

Example: **LAR** *dma*, **16** ; direct addressing with left shift of 16

```
LAR AR1, 60h, 16 ; load auxiliary AR1 register with the memory
                   contents of 60h that was left shifted
                   16 bits
```

Example: **LACC** *dma*, [shift] ; optional left shift from 0, 15 ; defaults to 0

```
LACC  main_counter,  8  ;  shifts  contents  of  the  variable
                   "main_counter" data 8 places to the left
                   before loading accumulator
```

[ ] An optional operand may be placed in the placed here.

Example: **LACC** *ind* [, shift [, **AR** n]_] Indirect addressing

```
LACC    *           ;load Accum. w/contents of the memory
                    ;location pointed to by the current AR.
LACC    * ,5        ;load Accum. with the contents of the memory
                    ;location pointed to by the current AR after
                    ;the memory contents are left shifted by 5
                    ;bits .
LACC    * ,0, AR3   ;load Accum. with the contents of the memory
                    ;location pointed to by the current AR after
                    ;the memory contents are left shifted by 5
                    ;bits . Now you have the option of choosing
                    ;a new AR. In this case, AR3 will become the
                    ;new AR.
```

[, x1 [, x2]] Operands x1 and x2 are optional, but you cannot include x2 without also including x1.

It is optional when using indirect addressing to modify the data. Once you supply a left shift value from 0…15 (even a shift of 0), then you have the option of changing to a new current auxiliary register (AR).

The # sign is prefix that signifies that the number used is a constant as opposed to memory location.

Example: RPT #15 ; this syntax is using short immediate addressing. It will repeat the next instruction 15+1 times.

```
LACC    #60h        ;this will load the accumulator with the
                    ;constant 60h
LACC    60h         ;However, this instruction will load the
                    ;accumulator with the contents in the data
                    ;memory location 60h, not the constant #60h
```

We will now provide a few examples of using the instruction set. Example 2.1 performs a few arithmetic functions with the DSP core and illustrates the nature of assembly programming. Programming with the assembly instruction set is somewhat different than languages such as C. In a high-level language, to add two numbers we might just code "c = a + b". In assembly, the user must be sure to code everything that needs to happen in order for a task to be executed. Take the following example:

Example 2.1 - Add the two numbers "2" and "3":

```
LDP     #6h         ;loads the proper DP for dma 300h
SPLK    #2, 300h    ;store the number "2" in memory address 300h
LACL    #3          ;load the accumulator with the number "3"
ADD     300h        ;adds contents of 300h ("2") to the contents
                    ;of the accumulator("3"); accumulator = 5
```

Another way:

```
LDP     #6h         ;loads the proper DP for dma 300h
SPLK    #2h, 300h   ;store the number "2h" in memory address
                    ;300h
SPLK    #3h, 301h   ;stores the number "3h" into memory address
                    ;301h
LACL    300h        ;load the accumulator with the contents in
                    ;memory location 300h
ADD     301h        ;adds contents of memory address 301h ("3h")
                    ;to the contents of the accumulator ("2h")
                    ;accumulator = 5h
```

Looping algorithms are very common in all programming languages. In high-level languages, the "For" and "While" loops can be used. However, in assembly, we need a slightly different approach to perform a repeating algorithm. The

GENERAL PURPOSE INPUT/OUTPUT (GPIO) FUNCTIONALITY

3.1 Pin Multiplexing (MUX) and General Purpose I/O Overview

Due to the limited number of physical pins on the LF2407 DSP, it is necessary to multiplex two functions onto most of the pins. That is, each pin can be programmed for either a primary or secondary (GPI/O) function (see Fig. 3.1). Once the pins on the LF2407 are multiplexed, the effective pin-out of the device is doubled. This provides enough effective pin-out for six General Purpose Input Output (GPIO) ports to be configured as the secondary function on most pins. Each Input/Output Port (IOP) consists of eight pins when they are configured to their secondary function.

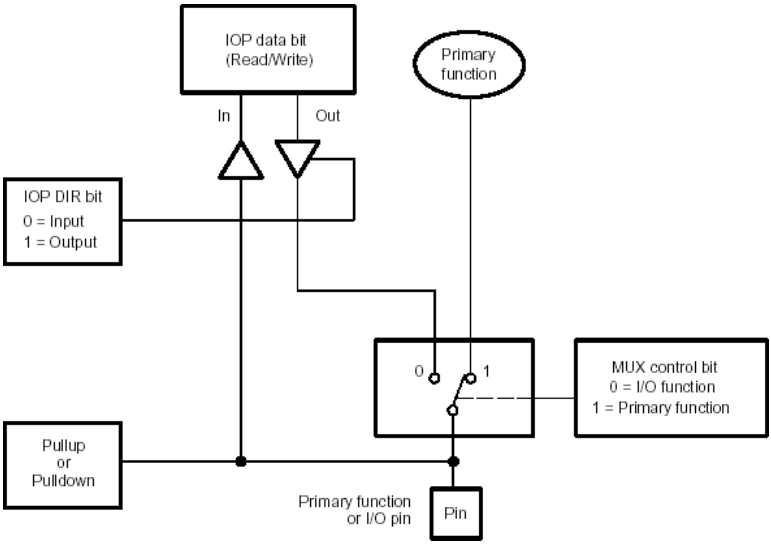


Figure 3.1 Block diagram of the multiplexing of a single pin. (Courtesy of Texas Instruments)

GPIO pins are grouped in sets of eight pins called ports. There are six ports total, ports A through F. Even though the pins are grouped in ports, each pin can be individually configured as primary or secondary (GPIO) functionality; and if GPIO, then either input or output. The multiplexing of primary pin functions with secondary GPIO functions provides a flexible method of controlling both the dedicated andsecondary pin functions.

Each multiplexed pin's primary/secondary functionality is controlled by a corresponding bit in the appropriate MUX control register. Additionally, when the pin is in GPIO mode, there are port data and direction (PxDATDIR) control

registers which control the direction (input or output) and data of the port/pin. If the pin is configured as an output, then the data (voltage) on the pin is determined by what value is written to the pin's data bit. Inversely, if the pin is configured as an input, then the voltage level applied to the pin determines the value of the pin's corresponding data bit.

If the pin is configured as an output pin, it can either be set to a logic high "1" (3.3 Volts) or a logic low "0" (0 Volts) by writing to its corresponding data bit in the corresponding PxDATDIR register. If the pin is configured as an input, the pin's corresponding bit in the appropriate PxDATDIR register will be "1" if 3.3 Volts or "0" if 0 Volts is applied to the pin. The data bits in the PxDATDIR can then be read by the user code and the values used in the program. The input and output ports provide a convenient way to input or output binary data (each pin = 1 bit). For example, a seven-segment display could be controlled by a GPIO port configured as output.

***Note**: There is no relationship between the GPIO pins and the I/O space of the LF2407.*

2. Multiplexing and General Purpose I/O Control Registers

The three MUX control registers and six data/direction control registers are all mapped to data memory (see Table 3.1). They control all dedicated and shared pin functions:

- I/O MUX Control Registers (MCRA, MCRB, MCRC): These 16-bit registers determine whether a pin will operate in its primary function or secondary GPIO function. Two ports are assigned to each MUX control register. For example, the MCRA register controls ports A and B.

- Data and Direction Control registers (PxDATDIR): Once a pin is configured in I/O mode by the appropriate MUX control register, the appropriate PxDATDIR register is used to configure each pin as input or output; and if output, whether the pin is high (3.3 Volts) or low (0 Volts).

Table 3.1 GPIO Control Register Summary

Data Memory Address	Register Name	Description
7090h	MCRA	I/O MUX Control Register A
7092h	MCRB	I/O MUX Control Register B
7094h	MCRC	I/O MUX Control Register C
7098h	PADATDIR	I/O Port A Data and Direction Register
709Ah	PBDATDIR	I/O Port B Data and Direction Register
709Ch	PCDATDIR	I/O Port C Data and Direction Register
709Eh	PDDATDIR	I/O Port D Data and Direction Register
7095h	PEDATDIR	I/O Port E Data and Direction Register
7096h	PFDATDIR	I/O Port F Data and Direction Register

3.2.1 I/O Multiplexing(MUX) Control Registers

I/O MUX Control Register A (MCRA) Configuration

15	14	13	12	11	10	9	8
MCRA.15	MCRA.14	MCRA.13	MCRA.12	MCRA.11	MCRA.10	MCRA.9	MCRA.8
RW –0	RW –0	RW –0	RW –0	RW –0	RW –0	RW –0	RW –0
7	6	5	4	3	2	1	0
MCRA.7	MCRA.6	MCRA.5	MCRA.4	MCRA.3	MCRA.2	MCRA.1	MCRA.0
RW –0	RW –0	RW –0	RW –0	RW –0	RW –0	RW –0	RW –0

Note: *R = read access, W = write access, -0 = value after reset.*

Bit #	Name.bit#	Pin Function Selected	
		(MCA.n = 1) (Primary)	(MCA.n = 0) (Secondary)
0	MCRA.0	SCITXD	IOPA0
1	MCRA.1	SCIRXD	IOPA1
2	MCRA.2	XINT1	IOPA2
3	MCRA.3	CAP1/QEP1	IOPA3
4	MCRA.4	CAP2/QEP2	IOPA4
5	MCRA.5	CAP3	IOPA5
6	MCRA.6	PWM1	IOPA6
7	MCRA.7	PWM2	IOPA7
8	MCRA.8	PWM3	IOPB0
9	MCRA.9	PWM4	IOPB1
10	MCRA.10	PWM5	IOPB2
11	MCRA.11	PWM6	IOPB3
12	MCRA.12	T1PWM/T1CMP	IOPB4
13	MCRA.13	T2PWM/T2CMP	IOPB5
14	MCRA.14	TDIRA	IOPB6
15	MCRA.15	TCLKINA	IOPB7

I/O MUX Control Register B (MCRB) Configuration

15	14	13	12	11	10	9	8
MCRB.15	MCRB.14	MCRB.13	MCRB.12	MCRB.11	MCRB.10	MCRB.9	MCRB.8
RW-1	RW-1	RW-1	RW-1	RW-1	RW-1	RW-1	RW-0
7	6	5	4	3	2	1	0
MCRB.7	MCRB.6	MCRB.5	MCRB.4	MCRB.3	MCRB.2	MCRB.1	MCRB.0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-1	RW-1

Note: *R = read access, W = write access, -0 = value after reset.*

Bit #	Name.bit #	Pin Function Selected	
		(MCB.n = 1) (Primary)	**(MCB.n = 0) (Secondary)**
0	MCRB.0	W/R	IOPC0
1	MCRB.1	BIO	IOPC1
2	MCRB.2	SPISIMO	IOPC2
3	MCRB.3	SPISOMI	IOPC3
4	MCRB.4	SPICLK	IOPC4
5	MCRB.5	SPISTE	IOPC5
6	MCRB.6	CANTX	IOPC6
7	MCRB.7	CANRX	IOPC7
8	MCRB.8	XINT2/ADCSOC	IOPD0
9	MCRB.9	EMU0	Reserved
10	MCRB.10	EMU1	Reserved
11	MCRB.11	TCK	Reserved
12	MCRB.12	TDI	Reserved
13	MCRB.13	TDO	Reserved
14	MCRB.14	TMS	Reserved
15	MCRB.15	TMS2	Reserved

I/O MUX Control Register C (MCRC) Configuration

15	14	13	12	11	10	9	8
Reserved	Reserved	MCRC.13	MCRC.12	MCRC.11	MCRC.10	MCRC.9	MCRC.8
		RW –0	RW –0	RW –0	RW –0	RW –0	RW –0
7	6	5	4	3	2	1	0
MCRC.7	MCRC.6	MCRC.5	MCRC.4	MCRC.3	MCRC.2	MCRC.1	MCRC.0
RW –0	RW –0	RW –0	RW –0	RW –0	RW –0	RW –0	RW –1

Note: *R = read access, W = write access, -0 = value after reset.*

		Pin Function Selected	
it #	**Name.bit#**	**(MCC.n = 1) (Primary)**	**(MCC.n = 0) (Secondary)**
0	MCRC.0	CLKOUT	IOPE0
1	MCRC.1	PWM7	IOPE1
2	MCRC.2	PWM8	IOPE2
3	MCRC.3	PWM9	IOPE3
4	MCRC.4	PWM10	IOPE4
5	MCRC.5	PWM11	IOPE5
6	MCRC.6	PWM12	IOPE6
7	MCRC.7	CAP4/QEP3	IOPE7
8	MCRC.8	CAP5/QEP4	IOPF0
9	MCRC.9	CAP6	IOPF1
10	MCRC.10	T3PWM/T3CMP	IOPF2
11	MCRC.11	T4PWM/T4CMP	IOPF3
12	MCRC.12	TDIRB	IOPF4
13	MCRC.13	TCLKINB	IOPF5
14	MCRC.14	Reserved	IOPF6
15	MCRC.15	Reserved	Reserved

Bits 7–0 IOPBn – Data Bits
If BnDIR = 0, then:
0 Corresponding I/O pin is read as a low
1 Corresponding I/O pin is read as a high
If BnDIR = 1, then:
0 Set corresponding I/O pin low
1 Set corresponding I/O pin high

Port C Data and Direction Control Register (PCDATDIR)

15	14	13	12	11	10	9	8
C7DIR	C6DIR	C5DIR	C4DIR	C3DIR	C2DIR	C1DIR	C0DIR
RW –0	RW –0	RW –0	RW –0	RW –0	RW –0	RW –0	RW –0
7	6	5	4	3	2	1	0
IOPC7	IOPC6	IOPC5	IOPC4	IOPC3	IOPC2	IOPC1	IOPC0
RW –†	RW –†	RW –†	RW –†	RW –†	RW –†	RW –†	RW –x

† The reset value of these bits depends upon the state of the respective pins.

Note: *R = read access, W = write access, -0 = value after reset, x = undefined.*

Bits 15–8 CnDIR – Direction Bits
1 Configure corresponding pin as an input
2 Configure corresponding pin as an output

Bits 7–0 IOPCn – Data Bits
If CnDIR = 0, then:
0 Corresponding I/O pin is read as a low
1 Corresponding I/O pin is read as a high
If CnDIR = 1, then:
0 Set corresponding I/O pin low
1 Set corresponding I/O pin high

Port D Data and Direction Control Register (PDDATDIR)

15-9	8
Reserved	D0DIR
	RW –0
7-1	0
Reserved	IOPD0
	RW –†

† The reset value of this bit depends upon the state of the respective pins.

Note: *R = read access, W = write access, -0 = value after reset.*

Bits 15–9 Reserved

Bit 8 D0DIR – Direction Bits

1 Configure corresponding pin as an input
2 Configure corresponding pin as an output

Bits 7–1 Reserved

Bit 0 IOPD0 – Data Bit

If D0DIR = 0, then:

0 Corresponding I/O pin is read as a low
1 Corresponding I/O pin is read as a high

If D0DIR = 1, then:

0 Set corresponding I/O pin low
1 Set corresponding I/O pin high

Port E Data and Direction Control Register (PEDATDIR)

15	14	13	12	11	10	9	8
E7DIR	E6DIR	E5DIR	E4DIR	E3DIR	E2DIR	E1DIR	E0DIR
RW –0	RW –0	RW –0	RW –0	RW –0	RW –0	RW –0	RW –0
7	6	5	4	3	2	1	0
IOPE7	IOPE6	IOPE5	IOPE4	IOPE3	IOPE2	IOPE1	IOPE0
RW –†	RW –†	RW –†	RW –†	RW –†	RW –†	RW –†	RW –x

† The reset value of these bits depends upon the state of the respective pins.

Note: *R = read access, W = write access, -0 = value after reset, x = undefined.*

Bits 15–8 EnDIR – Direction Bits

1 Configure corresponding pin as an input
2 Configure corresponding pin as an output

Bits 7–0 IOPEn – Data Bits

If EnDIR = 0, then:

0 Corresponding I/O pin is read as a low
1 Corresponding I/O pin is read as a high

If EnDIR = 1, then:

0 Set corresponding I/O pin low
1 Set corresponding I/O pin high

Port F Data and Direction Control Register (PFDATDIR)

15	14	13	12	11	10	9	8
Reserved	F6DIR	F5DIR	F4DIR	F3DIR	F2DIR	F1DIR	F0DIR
	RW –0	RW –0	RW –0	RW –0	RW –0	RW –0	RW –0
7	6	5	4	3	2	1	0
Reserved	IOPF6	IOPF5	IOPF4	IOPF3	IOPF2	IOPF1	IOPF0
	RW –†	RW –†	RW –†	RW –†	RW –†	RW –†	RW –†

† The reset value of these bits depends upon the state of the respective pins.

Note: *R = read access, W = write access, -0 = value after reset.*

Bit 15 Reserved

Bits 14–8 FnDIR – Direction Bits

0 Configure corresponding pin as an input
1 Configure corresponding pin as an output

Bit 7 Reserved

Bits 6–0 IOPFn – Data Bits

If FnDIR = 0, then:

0 Corresponding I/O pin is read as a low
1 Corresponding I/O pin is read as a high

If FnDIR = 1, then:

0 Set corresponding I/O pin low
1 Set corresponding I/O pinhigh

3. Using the General Purpose I/O Ports

The GPIO functionality is relatively simple to use and provides a valuable way of imputing and outputting data to and from the DSP. To use the GPIO functionality of a particular pin or groups of pins, the following steps must be followed to configure the DSP:

1. Set the bits in the appropriate MUX control register to configure the desired pins for GPIO function. This can be done by writing a “0” to the corresponding bits in the appropriate MUX. It may not be absolutely necessary to do this due to the fact that upon a reset (power on) the pins in the LF2407 are by default in their GPIO functionally. However, configuring the MUX register anyway is good programming practice.

2. Now that the desired pins are configured as GPIO, set the Port Data and Direction (PxDATDIR) register(s) that corresponds to the desired pins. When configuring the PxDATDIR, the most significant bits control the direction (input or output) and the lower bits determine (output) or display

INTERRUPTS ON THE TMS320LF2407

1. Introduction to Interrupts

The interrupts on the LF2407 allow the device hardware to trigger the CPU of the LF2407 (CPU=C2xx DSP core) to break from the current task, branch to a new section of code and start a new task, then return back to the initial task. The "new task" referred to in the previous sentence is known as the Interrupt Service Routine (ISR). The ISR is simply a separate user-written subroutine, which the core will branch to every time a certain interrupt occurs.

For example, say the ADC is being used and we want the program to load the conversion value into the accumulator every time the ADC finishes a conversion. The ADC can be configured to generate an interrupt whenever a conversion is finished. When the ADC generates its interrupt, the interrupt signal makes its way through the interrupt hierarchy to the core and the core then branches to the appropriate ISR.

In a more general sense, when an interrupt occurs, the core branches to the ISR (GISR1, GISR2 etc… depending on the interrupt) where an interrupt service routine is located. In the ISR, after the instructions are executed, the interrupt hierarchy is "reset" to allow for future interrupts. This usually entails clearing the peripheral level interrupt flag bit and clearing the INTM bit. These steps ensure that future interrupts of the same origin will be able to pass through to the core. The final instruction in the ISR is the RET command, which instructs the core to return to where it was before the interrupt occurred.

2. Interrupt Hierarchy

This section will explain the different hierarchical levels and how an interrupt request signal propagates through them. The different control registers and their operations will be reviewed.

1. Interrupt Request Sequence

There are two levels of interrupt hierarchy in the LF2407 as seen in Fig. 4.1 below. There is an interrupt flag bit and an interrupt enable bit located in each peripheral configuration register for each event that can generate an interrupt. The peripheral interrupt flag bit is the first bit to be set when an interrupt generating event occurs. The interrupt enable bit acts as a "gate". If the interrupt enable bit is not set, then the setting of the peripheral flag bit will not be able to generate an interrupt signal. If the enable bit is set, then the peripheral flag bit will generate an interrupt signal. That interrupt signal will then leave the peripheral level and go to the next hierarchal level.

Once an interrupt signal leaves the peripheral level, it is then multiplexed through the Peripheral Interrupt Expansion (PIE) module. The PIE module takes the many individual interrupts and groups them into six priority levels (INT1 through INT6). Once an interrupt reaches the PIE, a code identifying the individual interrupt is loaded into the Peripheral Interrupt Vector Register (PIVR). This allows the ISR to determine which interrupt was actually asserted when multiple interrupts from the same level occur. After passing through the PIE module, the interrupt request signal has now entered the upper level of hierarchy or the "CPU level".

The six interrupt groupings from the PIE module feed into the CPU level. The final stage of the CPU level is the CPU itself (C2xx core). From Fig. 4.1, we can see the six interrupt levels and the many individual peripheral interrupts assigned to priority level. Each of the six levels has a corresponding flag bit in the Interrupt Flag Register (IFR). Additionally there is an Interrupt Mask Register (IMR) which acts similar to the interrupt enable bits at the peripheral level. Each of the six bits in the IMR behaves as a "gate" to each of the corresponding six bits in the IFR. If the corresponding bits in both the IFR and IMR are both set, then the interrupt request signal can continue through to the C2xx core itself.

Once the interrupt request signal has entered the CPU level and has passed through the IFR/IMR, there is one more gateway the signal must pass through in order to cause the core to service the interrupt. The Interrupt Mask (INTM) bit must be cleared for the interrupt signal to reach the core. When the core acknowledges a pending interrupt, the INTM bit is automatically set, thereby not allowing any more interrupts from reaching the core while a current interrupt is being serviced.

When the core is finished with the current interrupt, only the flag bit in the IFR is cleared automatically. The INTM bit and the peripheral level flag bit must be cleared "manually" via software. When this is done, the core will acknowledge the highest priority pending interrupt request signal.

Additionally, if an interrupt request signal occurs, but the signal never reaches the core, all flag bits "downstream" of the point where the signal was halted will still remain set until cleared by software. The IFR bits will be cleared if: (1) the interrupt path to the core is opened, and the interrupt is acknowledged normally or
(2) the bit is cleared "manually" by software. If no interrupt request has occurred but the peripheral level IF bit is set and the peripheral IE bit is later set without clearing the IF bit, then an interrupt request signal will be asserted and the corresponding IFR bit will be set.

Furthermore, in the event that two interrupts of different priority groupings (INTx) occur at the same time, the highest priority interrupt will be acknowledged first by the core.

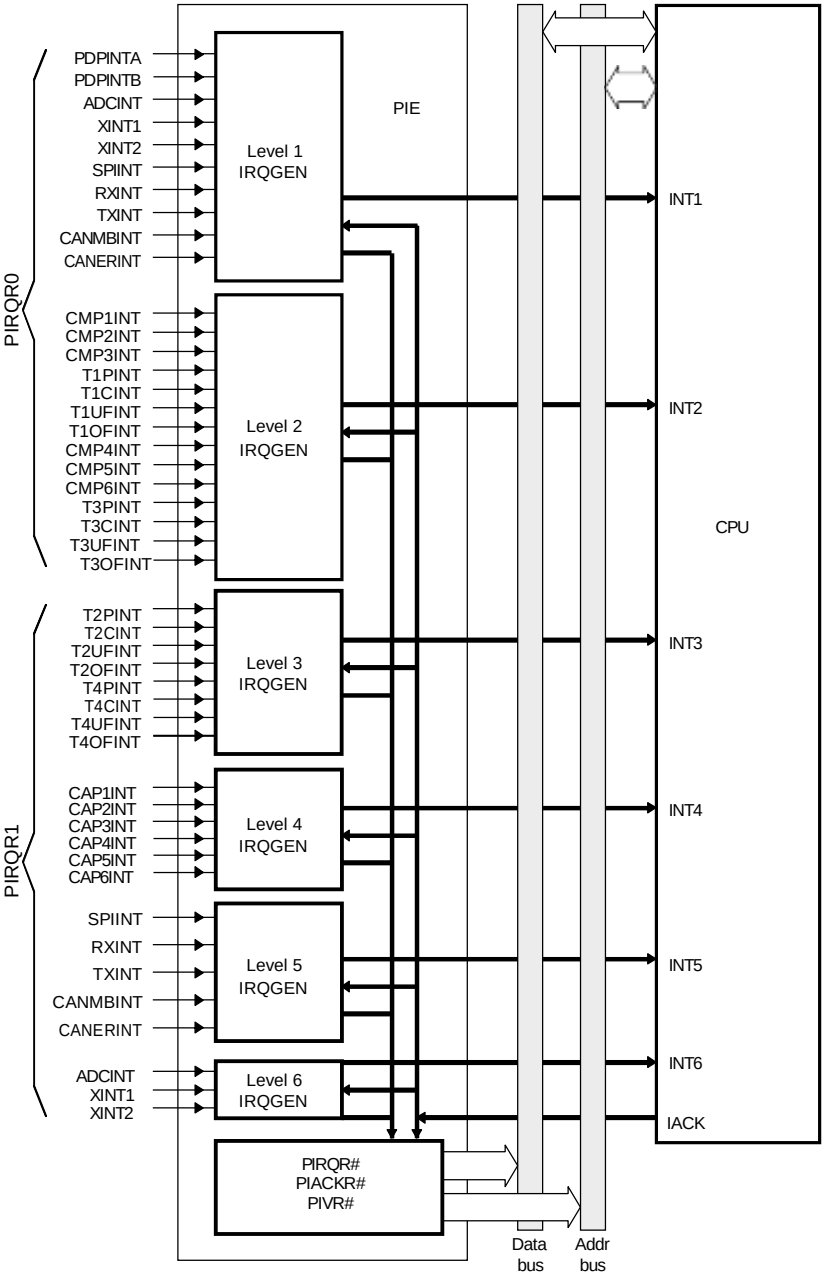


Figure 4.1 Interrupt hierarchy in the LF2407. (Courtesy of Texas Instruments)

4.2.2 Reset and Non-Maskable Interrupts

There are two special interrupts on the LF2407 which have not been covered thus far; the Reset (RS) and the Non-Maskable Interrupt (NMI). Both of these interrupts bypass the usual interrupt hierarchy and feed straight to the DSP core. A reset causes the core to branch to address 0000h in program memory. Resets are activated during power on, when the external RESET pin is brought to logic "0" (O Volts), or by the Watchdog Timer. If the Watchdog is not disabled, it will pull the reset pin to "0" if not periodicallyreset.

When an illegal memory space is written to, the illegal address flag (ILLADR) in System Control and Status Register 1 (SCSR1) will be set. When this flag is set, a non-maskable interrupt (NMI) will be generated, causing the core to branch to address 0024h in program memory. The illegal address flag (ILLADR) will remain set following an illegal address condition until it is cleared by software or a DSP reset.

3. Interrupt Control Registers

This section will review the interrupt control registers. The IFR, IMR, and PIVR registers as well as the INTM bit discussed in the previous section will be presented in more detail. We will not discuss peripheral level interrupt bits in this chapter, as they will be discussed in each section dealing with the specific peripherals.

There are three registers used at the CPU level, the Interrupt Flag Register (IFR), the Interrupt Mask Register (IMR), and the Peripheral Interrupt Vector Register (PIVR). The IFR and IMR control the interrupt signal at the beginning of the CPU level. The PIVR register, while actually loaded in the PIE, provides information about the specific interrupt that occurred at the peripheral level. This information can be used by the ISR in determining the source of the interrupt signal. In addition to these registers, the INTM bit at the CPU level provides the final "gateway" that the interrupt signal must pass through to reach the core itself.

In addition to the peripheral interrupts, there are two External Interrupts (XINT1, XINT2). Their interrupt request operation is exactly like the peripheral interrupts. However, external interrupts are triggered by a logic edge transition on their external pin. The external interrupt control registers will also be discussed.

1. Interrupt Flag Register (IFR)

The IFR is a 16-bit (only 6 bits are really used) register mapped to address 0006h in data memory. The IFR is used to identify and clear pending interrupts at the CPU level and contains the interrupt flag bits for the maskable interrupt priorities INT1–INT6.

A flag bit in the IFR is set to "1" when an individual interrupt request signal makes its way out of the peripheral level and into the CPU level. The particular flag

bit set depends on what priority the individual interrupt is grouped under. After the interrupt is serviced, the IFR bit corresponding to the interrupt is automatically cleared (to "0") by the DSP.

In addition to triggering the CPU level during the standard interrupt process, the IFR can also be read by software. If a desired situation occurred where the INTM bit was set (meaning no interrupt signals make it to the core) and an interrupt signal was generated at the below levels, the corresponding bit in the IFR would still be set. In this situation, the IFR could be read by software to identify pending interrupt requests.

If desired, to "manually" clear a bit in the IFR, software needs to write a "1" to the appropriate bit (see IFR bit descriptions). The flag bits can be thought of as "toggling" when a "1" is written to them. Loading the IFR into the accumulator, then storing the contents of the IFR back into itself clears all bits in the IFR. However, if the peripheral level interrupt flag bit is still set, the corresponding bit in the IFR will immediately become set right after it is cleared.

Notes:

1. *To clear an IFR bit, we must write a one to it, not a zero.*
2. *When an interrupt is acknowledged,* ***only the IFR bit is cleared automatically****. The flag bit in the corresponding peripheral control register is* ***not*** *automatically cleared. If an application requires that the control register flag be cleared, the bit must be cleared by software.*
3. *IFR registers pertain to interrupts at the CPU level only. All peripherals have their own interrupt mask and flag bits in their respective control/configuration registers.*
4. *When an interrupt is requested by the INTR assembly instruction and the corresponding IFR bit is set, the CPU does not clear the bit automatically. If an application then requires that the IFR bit needs to be cleared, the bit must be cleared by software.*

Interrupt Flag Register (IFR) — Address 0006h

15-6	5	4	3	2	1	0
Reserved	INT6 flag	INT5 flag	INT4 flag	INT3 flag	INT2 flag	INT1 flag
0	RW1C–0	RW1C–0	RW1C–0	RW1C–0	RW1C–0	RW1C–0

Note: *0 = always read as zeros, R = read access, W1C = write 1 to this bit to clear it, -0 = value after reset.*

Bits 15–6 **Reserved.** These bits are always read as zeros.

Bit 5 **INT6.** Interrupt 6 flag. This bit is the flag for interrupts connected to interrupt level INT6.

0 No INT6 interrupt is pending

1 At least one INT6 interrupt is pending. Write a 1 to this bit to clear it to 0 and clear the interruptrequest

Bit 4 **INT5.** Interrupt 5 flag. This bit is the flag for interrupts connected to interrupt level INT5.

0 No INT5 interrupt is pending

1 At least one INT5 interrupt is pending. Write a 1 to this bit to clear it to 0 and clear the interruptrequest

Bit 3 **INT4.** Interrupt 4 flag. This bit is the flag for interrupts connected to interrupt level INT4.

0 No INT4 interrupt is pending

1 At least one INT4 interrupt is pending. Write a 1 to this bit to clear it to 0 and clear the interruptrequest

Bit 2 **INT3.** Interrupt 3 flag. This bit is the flag for interrupts connected to interrupt level INT3.

0 No INT3 interrupt is pending

1 At least one INT3 interrupt is pending. Write a 1 to this bit to clear it to 0 and clear the interruptrequest

Bit 1 **INT2.** Interrupt 2 flag. This bit is the flag for interrupts connected to interrupt level INT2.

0 No INT2 interrupt is pending

1 At least one INT2 interrupt is pending. Write a 1 to this bit to clear it to 0 and clear the interruptrequest

Bit 0 **INT1.** Interrupt 1 flag. This bit is the flag for interrupts connected to interrupt level INT1.

0 No INT1 interrupt is pending

1 At least one INT1 interrupt is pending. Write a 1 to this bit to clear it to 0 and clear the interruptrequest

Interrupt Mask Register(IMR)

The Interrupt Mask Register (IMR) is a 16-bit (only 6 bits are used) register located at address 0004h in data memory. It contains a mask bits for each of the six interrupt priority levels INT1–INT6. When an IMR bit is “0”, the corresponding interrupt is “masked”. When an interrupt is masked, the interrupt will be halted at the CPU level; the core will not be able to receive the interrupt request signal, regardless of the INTM bit status. When the interrupt’s IMR bit is set to “1”, the interrupt will be acknowledged if the corresponding IFR bit is “1” and the INTM bit is “0”. The IMR may also be read to identify which interrupts are masked or unmasked.

Interrupt Mask Register (IMR) — Address 0004h

15-6	5	4	3	2	1	0
Reserved	INT6 mask	INT5 mask	INT4 mask	INT3 mask	INT2 mask	INT1 mask
0	RW	RW	RW	RW	RW	RW

Note: *0 = always read as zeros, R = read access, W = write access, bit values are not affected by a device reset.*

Bits 15–6 **Reserved**. These bits are always read as zeros.

Bit 5 **INT6.** Interrupt 6 mask. This bit masks or unmasks interrupt level INT6.
0 Level INT6 is masked
1 Level INT6 is unmasked

Bit 4 **INT5.** Interrupt 5 mask. This bit masks or unmasks interrupt level INT5.
0 Level INT5 is masked
1 Level INT5 is unmasked

Bit 3 **INT4.** Interrupt 4 mask. This bit masks or unmasks interrupt level INT4.
0 Level INT4 is masked
1 Level INT4 is unmasked

Bit 2 **INT3.** Interrupt 3 mask. This bit masks or unmasks interrupt level INT3.
0 Level INT3 is masked
1 Level INT3 is unmasked

Bit 1 **INT2.** Interrupt 2 mask. This bit masks or unmasks interrupt level INT2.
0 Level INT2 is masked
1 Level INT2 is unmasked

Bit 0 **INT1.** Interrupt 1 mask. This bit masks or unmasks interrupt level INT1.
0 Level INT1 is masked
1 Level INT1 is unmasked

Note: *A device reset does not affect The IMR bits.*

Peripheral Interrupt Vector Register(PIVR)

The Peripheral Interrupt Vector Register (PIVR) is a 16-bit read-only register located at address 701Eh in data memory. Each interrupt has a unique code which is loaded into the PIVR when in the PIE module. When a peripheral interrupt signal is passed through the PIE module, the PIVR is loaded with the vector of the pending interrupt which has the highest priority level. This assures that if two interrupts of

different priorities happen simultaneously, the higher priority interrupt will be serviced first.

Peripheral Interrupt Vector Register (PIVR) — Address 701Eh

15	14	13	12	11	10	9	8
V15	V14	V13	V12	V11	V10	V9	V8
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
V7	V6	V5	V4	V3	V2	V1	V0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

Note: *R = read access, -0 = value after reset.*

Bits 15–0 V15–V0. Interrupt vector. This register contains the peripheral interrupt vector of the most recently acknowledged peripheral interrupt.

External Interrupt Control Registers

The external interrupts (XINT1, XINT2) are controlled by the XINT1CR and XINT2CR control registers, respectively. If these interrupts are enabled in their control registers, an interrupt will be generated when the XINT1 or XINT2 logic transition occurs for at least 12 CPU clock cycles.

For example, if XINT1 was configured for generating an interrupt on a low (0 Volts) to high (3.3 Volts) transition and the XINT1 pin only went high for 6 clock cycles, then back down to low, an interrupt request would not occur. However, if the pin was brought high for 12 or more cycles, an interrupt request signal would be generated.

External Interrupt 1 Control Register (XINT1CR) – Address 7070h

15	14–3	2	1	0
XINT1 flag	Reserved	XINT1 polarity	XINT1 priority	XINT1 enable
RC-0	R-0	RW-0	RW-0	RW-0

Note: *R = read access, W = write access, C = clear by writing a 1, -0 = value after reset.*

Bit 15 XINT1 Flag

This bit indicates if the selected transition has been detected on the XINT1 pin and is set whether or not the interrupt is enabled. This bit is cleared by software writing a 1 (writing a 0 has no effect), or by a device reset.

0 No transition detected

1 Transition detected

Note: *the description in the TI user guide can be misleading: this bit is* ***not*** *cleared automatically during the interrupt acknowledge sequence.*

Bits 14–3 **Reserved.** Reads return zero; writes have no effect.

Bit 2 **XINT1 Polarity**

This read/write bit determines if interrupts are generated on the rising edge or the falling edge of a signal on the pin.

0 Interrupt generated on a falling edge (high-to-low transition)

1 Interrupt generated on a rising edge (low-to-high transition)

Bit 1 **XINT1 Priority**

This read/write bit determines which interrupt priority is requested. The CPU interrupt priority levels corresponding to low and high priority are coded into the peripheral interrupt expansion controller. These priority levels are shown in Table 2–2, *240xA Interrupt Source Priority and Vectors,* in Chapter 2 on page 2-9.

0 High priority

1 Low priority

Bit 0 **XINT1 Enable**

This read/write bit enables or disables external interrupt XINT1.

0 Disable interrupt

1 Enable interrupt

External Interrupt 2 Control Register (XINT2CR) – Address 7071h

15	14–3	2	1	0
XINT2 flag	Reserved	XINT2 polarity	XINT2 priority	XINT2 enable
RC-0	R-0	RW-0	RW-0	RW-0

Note: *R = read access, W = write access, C = Clear by writing a 1, -0 = value after reset.*

Bit 15 **XINT2 Flag**

This bit indicates if the selected transition has been detected on the XINT2 pin and is set whether or not the interrupt is enabled. This bit is cleared by software writing a 1 (writing a 0 has no effect), or by a device reset.

0 No transition detected

1 Transition detected

Note: *the description in the TI user guide can be misleading: this bit is* ***not*** *cleared automatically during the interrupt acknowledge sequence.*

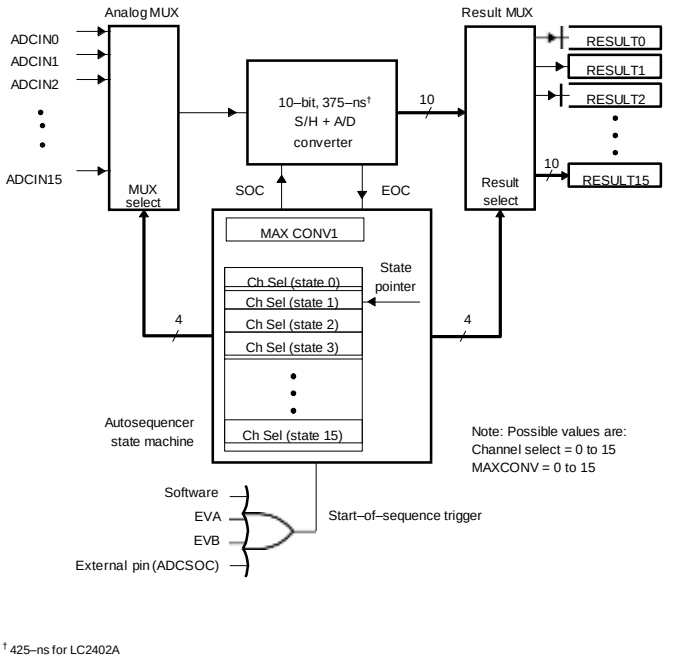


Figure 5.1 Block diagram of ADC in cascaded sequencer mode. (Courtesy of Texas Instruments)

Table 5.1 Comparasion table of dual (SEQ1 and SEQ2) versus cascaded sequencer configuration

Feature	Single 8–state sequencer #1(SEQ1)	Single 8–state sequencer #2 (SEQ2)	Cascaded 16–state sequencer (SEQ)
Start–of–conversion triggers	EVA, software, external pin	EVB, software	EVA, EVB, software, external pin
Maximum number of autoconversions (i.e., sequence length)	8	8	16
Autostop at end–of–sequence (EOS)	Yes	Yes	Yes
Arbitration priority	High	Low	Not applicable
ADC conversion result register locations	0 to 7	8 to 15	0 to 15
CHSELSEQn bit field assignment	CONV00 to CONV07	CONV08 to CONV15	CONV00 to CONV15

THE EVENT MANAGERS (EVA, EVB)

This chapter explains the features and operation of the LF2407 Event Managers (EV1, EV2). There are two identical event managers on board the LF2407 DSP. All control orientated features of the LF2407 are centered in the EV. The event manager peripheral is made up of components such as timers and pulse width modulation (PWM) generators. We start with a brief overview of the EV without getting into too much detail. Since the EV consists of several sub-components, we discuss in detail the operation and functionality of each sub-component separately in subsequent sections.

Overview of the Event Manager(EV)

We start with the EV by reviewing the multiple functional modules of the peripheral. The two EVs (EVA/B) are identical to one another in terms of functionality and register/bit definition, but have different register names and addresses. Since both EV1 and EV2 are identical, only the functionality of EV1 will be explained.

Each EV module in the LF2407 contains the following sub-components:

- Interrupt logic
- Two general-purpose (GP) timers
- Three compare units
- PWM circuits that include space vector PWM circuits, dead-band generation units, and output logic
- Three Capture Units
- Quadrature encoder pulse (QEP) circuit

Figure 6.1 shows a block diagram of the EVA module. Similarly, Fig. 6.2 illustrates the block diagram of EVB.

Like all peripherals, the EV registers occupy a range of 16-bit memory addresses in data memory space. Most of these registers are programmable control and data registers, but read-only status registers are also present. EVA registers are located in the data memory range 7400h to 7431h. EVB registers are located in the range of 7500h to 7531h. Some of the EV memory allocation range is for use by the DSP only. These undefined registers and undefined bits of EV registers will just read zero when read by user software. Writes also have no effect on these registers. As a general rule, one should not write to reserved or illegal addresses in order to avoid an illegal address non-maskable interrupt (NMI) from occurring.

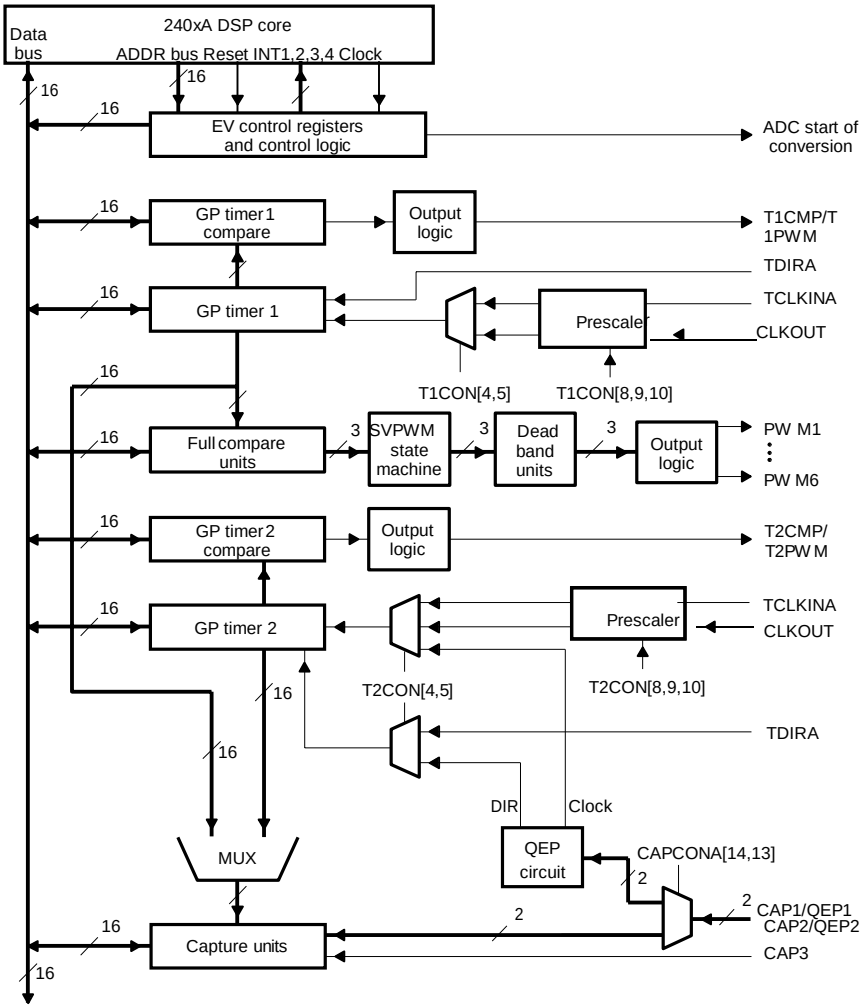


Figure 6.1 Event Manager A (EVA) block diagram. (Courtesy of Texas Instruments)

Event Manager Interrupts

The interrupt system in the EV will be discussed first because each of the sub-modules of the EVs have interrupt flags. The EV interrupt sub-system is slightly different from that of the main interrupt system. Each EV has its own “local” interrupt sub-system which includes its own interrupt mask and flag registers. After the EV interrupts pass through the sub-system, they flow into the PIE just like any other interrupt on the LF2407. The EV interrupts are arranged into three groups (A, B, C). Each group (A,B,C) has its own mask and flag register and is assigned to

)

a particular CPU interrupt priority level at the PIE. EV interrupts happen to be only at the INT2, INT3, and INT4 CPU priority levels.

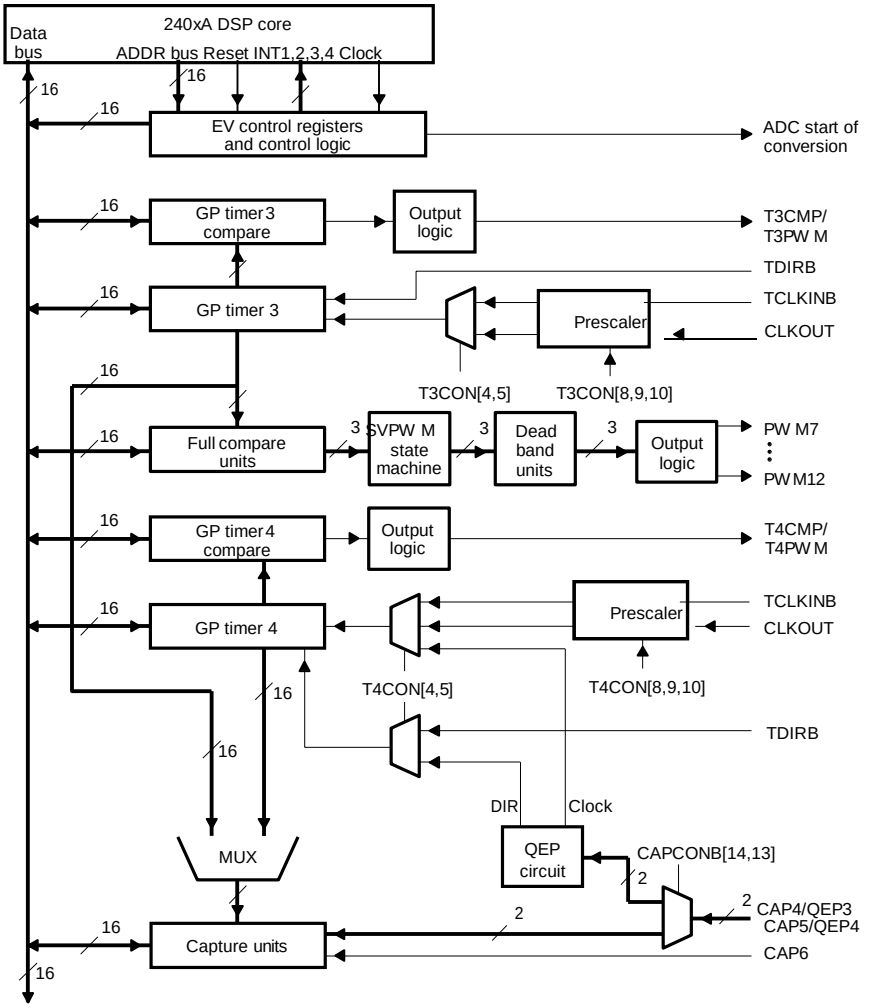


Figure 6.2 Event Manager B (EVB) block diagram. (Courtesy of Texas Instruments)

The following are the sequential steps for interrupt response within the EV:

1. ***Interrupt source***. When an EV interrupt condition occurs, the respective flag bits in registers EVxIFRA, EVxIFRB, or EVxIFRC (x = A or B) are set. As with other peripheral level flags, once set, these flags remain set until *explicitly* cleared by the software. In other words, you must clear

theses flags “manually” through your software in order for future interrupts to be recognized.

2. ***Interrupt enable.*** The EV interrupts can be individually enabled or disabled by the EV interrupt mask registers EVxIMRA, EVxIMRB, and EVxIMRC (x being either EV = A or B). To enable (unmask) an interrupt, the user must set the corresponding bit to “1”. To disable (mask) the interrupt, clear the corresponding bit to “0”. From now on, the interrupt is handled like other peripheral interrupts as discussed earlier in the text.

3. ***PIE request.*** If both interrupt flag bits and interrupt mask bits are set, then the interrupt request is passed to the PIE module. As with any other peripheral interrupts, the PIE module will send the CPU a request for a CPU level interrupt of the appropriate priority level based on the priority of the received interrupts.

4. ***CPU response.*** On receiving a CPU level interrupt request, the respective bit in the CPU interrupt flag register (IFR) will be set. If the corresponding interrupt mask register (IMR) bit is set and INTM bit is cleared, then the CPU recognizes the interrupt and issues an acknowledgement to the PIE module. Following this, the CPU finishes executing the current instruction and branches to the interrupt service routine via the interrupt vector. At this time, the respective IFR bit will be cleared and the INTM bit will be set disabling further interrupt recognition. The interrupt vector contains a branch instruction for the interrupt service routine. From here, the user software controls the interrupt servicing.

5. ***Interrupt software.*** The interrupt software can include two levels of response.

   a. **GISR:** The General Interrupt Service Routine (GISR) should do any context save and read the PIVR register to decide which specific interrupt occurred. Information on PIVR values and their corresponding interrupts can be found in Tables 6.1 and 6.2. Since the PIVR value for each interrupt is unique, it can be used to branch to the interrupt service routine specific to this interrupt condition.

   b. **SISR:** The Specific Interrupt Service Routine (SISR) level will normally reside as a sub-section of the GISR. After executing the interrupt specific service code, the routine should clear the interrupt flag in the EVxIFRA, EVxIFRB, or EVxIFRC that caused the serviced interrupt. Code will return the CPU to the pre-interrupt task after enabling the CPU’s global interrupt bit INTM (clear INTM bit).

)

EVA Interrupts

Table 6.1 EVA Interrupts and Corresponding PIVR Values

Group	Interrupt	Priority within group	Vector (ID)	Description/Source	INT
	PDPINTA	1 (highest)	0020h	Power Drive Protection Interrupt A	1
A	CMP1INT	2	0021h	Compare Unit 1 compare interrupt	2
	CMP2INT	3	0022h	Compare Unit 2 compare interrupt	2
	CMP3INT	4	0023h	Compare Unit 3 compare interrupt	2
	T1PINT	5	0027h	GP Timer 1 period interrupt	2
	T1CINT	6	0028h	GP Timer 1 compare interrupt	2
	T1UFINT	7	0029h	GP Timer 1 underflow interrupt	2
	T1OFINT	8 (lowest)	002Ah	GP Timer 1 overflow interrupt	2
B	T2PINT	1 (highest)	002Bh	GP Timer 2 period interrupt	3
	T2CINT	2	002Ch	GP Timer 2 compare interrupt	3
	T2UFINT	3	002Dh	GP Timer 2 underflow interrupt	3
	T2OFINT	4	002Eh	GP Timer 2 overflow interrupt	3
C	CAP1INT	1 (highest)	0033h	Capture Unit 1 interrupt	4
	CAP2INT	2	0034h	Capture Unit 2 interrupt	4
	CAP3INT	3	0035h	Capture Unit 3 interrupt	4

EVB Interrupts

Table 6.2 EVB Interrupts and Corresponding PIVR Values

Group	Interrupt	Priority within group	Vector (ID)	Description/Source	INT
	PDPINTB	1 (highest)	0019h	Power Drive Protection Interrupt B	1
A	CMP4INT	2	0024h	Compare Unit 4 compare interrupt	2
	CMP5INT	3	0025h	Compare Unit 5 compare interrupt	2
	CMP6INT	4	0026h	Compare Unit 6 compare interrupt	2
	T3PINT	5	002Fh	GP Timer 3 period interrupt	2
	T3CINT	6	0030h	GP Timer 3 compare interrupt	2
	T3UFINT	7	0031h	GP Timer 3 underflow interrupt	2
	T3OFINT	8 (lowest)	0032h	GP Timer 3 overflow interrupt	2
B	T4PINT	1 (highest)	0039h	GP Timer 4 period interrupt	3
	T4CINT	2	003Ah	GP Timer 4 compare interrupt	3
	T4UFINT	3	003Bh	GP Timer 4 underflow interrupt	3
	T4OFINT	4	003Ch	GP Timer 4 overflow interrupt	3
C	CAP4INT	1 (highest)	0036h	Capture Unit 4 interrupt	4
	CAP5INT	2	0037h	Capture Unit 5 interrupt	4
	CAP6INT	3	0038h	Capture Unit 6 interrupt	4

EVA Interrupt Flag Register A (EVAIFRA) — Address 742Fh

15-11			10	9	8
Reserved			T1OFINT FLAG	T1UFINT FLAG	T1CINT FLAG
R–0			RW 1C–0	RW 1C–0	RW 1C–0
7	6-4	3	2	1	0
T1PINT FLAG	Reserved	CMP3INT FLAG	CMP2INT FLAG	CMP1INT FLAG	PDPINTA FLAG
RW 1C–0	R–0	RW 1C–0	RW 1C–0	RW 1C–0	RW 1C–0

Note: *R = read access, W1C = write 1 to clear, -0 = value after reset.*

Bits 15–11 Reserved. Reads return zero; writes have no effect.

Bit 10 **T1OFINT FLAG**. GP Timer 1 overflow interrupt.

Read:	0	Flag is reset
	1	Flag is set
Write:	0	No effect
	1	Resets flag

Bit 9 **T1UFINT FLAG**. GP Timer 1 underflow interrupt.

Read:	0	Flag is reset
	1	Flag is set
Write:	0	No effect
	1	Resets flag

Bit 8 **T1CINT FLAG**. GP Timer 1 compare interrupt.

Read:	0	Flag is reset
	1	Flag is set
Write:	0	No effect
	1	Resets flag

Bit 7 **T1PINT FLAG.** GP Timer 1 period interrupt.

Read:	0	Flag is reset
	1	Flag is set
Write:	0	No effect
	1	Resets flag

Bits 6–4 Reserved. Reads return zero; writes have no effect.

Bit 3 **CMP3INT FLAG**. Compare 3 interrupt.

Read:	0	Flag is reset
	1	Flag is set
Write:	0	No effect
	1	Resets flag

)

Bit 2 **CMP2INT FLAG**. Compare 2 interrupt.

Read:	0	Flag is reset
	1	Flag is set
Write:	0	No effect
	1	Resets flag

Bit 1 **CMP1INT FLAG**. Compare 1 interrupt.

Read:	0	Flag is reset
	1	Flag is set
Write:	0	No effect
	1	Resets flag

Bit 0 **PDPINTA FLAG**. Power drive protection interrupt.

Read:	0	Flag is reset
	1	Flag is set
Write:	0	No effect
	1	Resets flag

EVA Interrupt Flag Register B (EVAIFRB) — Address 7430h

15-4	3	2	1	0
Reserved	T2OFINT FLAG	T2UFINT FLAG	T2CINT FLAG	T2PINT FLAG
R–0	RW 1C–0	RW 1C–0	RW 1C–0	RW 1C–0

Note: *R = read access, W1C = write 1 to clear, -0 = value after reset.*

Bits 15–4 **Reserved**. Reads return zero; writes have no effect.

Bit 3 **T2OFINT FLAG**. GP Timer 2 overflow interrupt.

Read:	0	Flag is reset
	1	Flag is set
Write:	0	No effect
	1	Resets flag

Bit 2 **T2UFINT FLAG**. GP Timer 2 underflow interrupt.

Read:	0	Flag is reset
	1	Flag is set
Write:	0	No effect
	1	Resets flag

Bit 1 **T2CINT FLAG**. GP Timer 2 compare interrupt.

Read:	0	Flag is reset
	1	Flag is set
Write:	0	No effect
	1	Resets flag

Bit 0 **T2PINT FLAG**. GP Timer 2 period interrupt.

Read:	0	Flag is reset
	1	Flag is set
Write:	0	No effect
	1	Resets flag

EVA Interrupt Flag Register C (EVAIFRC) — Address 7431h

15-3	2	1	0
Reserved	CAP3INT FLAG	CAP2INT FLAG	CAP1INT FLAG
R–0	RW 1C–0	RW 1C–0	RW 1C–0

Note: *R = read access, W1C = write 1 to clear, -0 = value after reset.*

Bits 15–3 **Reserved**. Reads return zero; writes have no effect.

Bit 2 **CAP3INT FLAG**. Capture 3 interrupt.

Read:	0	Flag is reset
	1	Flag is set
Write:	0	No effect
	1	Resets flag

Bit 1 **CAP2INT FLAG**. Capture 2 interrupt.

Read:	0	Flag is reset
	1	Flag is set
Write:	0	No effect
	1	Resets flag

Bit 0 **CAP1INT FLAG**. Capture 1 interrupt.

Read:	0	Flag is reset
	1	Flag is set
Write:	0	No effect
	1	Resets flag

EVA Interrupt Mask Register A (EVAIMRA) — Address 742Ch

15-11	10	9	8
Reserved	T1OFINT ENABLE	T1UFINT ENABLE	T1CINT ENABLE
R–0	RW –0	RW –0	RW –0

7	6-4	3	2	1	0
T1PINT ENABLE	Reserved	CMP3INT ENABLE	CMP2INT ENABLE	CMP1INT ENABLE	PDPINTA ENABLE
RW –0	R–0	RW –0	RW –0	RW –0	RW –1

Note: *R = read access, W = write access, value following dash (–) = value after reset.*

)

Bits 15–11 Reserved. Reads return zero; writes have no effect.

Bit 10 T1OFINT ENABLE
0 Disable
1 Enable
Bit 9 T1UFINT ENABLE
0 Disable
1 Enable
Bit 8 T1CINT ENABLE
0 Disable
1 Enable
Bit 7 T1PINT ENABLE
0 Disable
1 Enable
Bits 6–4 Reserved. Reads return zero; writes have no effect.

Bit 3 CMP3INT ENABLE
0 Disable
1 Enable
Bit 2 CMP2INT ENABLE
0 Disable
1 Enable

Bit 1 CMP1INT ENABLE
0 Disable
1 Enable

Bit 0 PDPINTA ENABLE. This is enabled (set to 1) following reset.
0 Disable
1 Enable

EVA Interrupt Mask Register B (EVAIMRB) — Address 742Dh

15-4	3	2	1	0
Reserved	T2OFINT ENABLE	T2UFINT ENABLE	T2CINT ENABLE	T2PINT ENABLE
R–0	RW –0	RW –0	RW –0	RW –0

Note: *R = read access, W = write access, -0 = value after reset.*

Bits 15–4 Reserved. Reads return zero; writes have no effect.

Bit 3 T2OFINT ENABLE
0 Disable
1 Enable

Bit 2 **T2UFINT ENABLE**
0 Disable
1 Enable

Bit 1 **T2CINT ENABLE**
0 Disable
1 Enable

Bit 0 **T2PINT ENABLE**
0 Disable
1 Enable

EVA Interrupt Mask Register C (EVAIMRC) — Address 742Eh

15-3	2	1	0
Reserved	CAP3INT ENABLE	CAP2INT ENABLE	CAP1INT ENABLE
R–0	RW –0	RW –0	RW –0

Note: *R = read access, W = write access, -0 = value after reset.*
Bits 15–3 **Reserved**. Reads return zero; writes have no effect.

Bit 2 **CAP3INT ENABLE**
0 Disable
1 Enable

Bit 1 **CAP2INT ENABLE**
0 Disable
1 Enable

Bit 0 **CAP1INT ENABLE**
0 Disable
1 Enable

EVB Interrupt Flag Register A (EVBIFRA) — Address 752Fh

15-11	10	9	8
Reserved	T3OFINT FLAG	T3UFINT FLAG	T3CINT FLAG
R–0	RW 1C–0	RW 1C–0	RW 1C–0

7	6-4	3	2	1	0
T3PINT FLAG	Reserved	CMP6INT FLAG	CMP5INT FLAG	CMP4INT FLAG	PDPINTB FLAG
RW 1C–0	R–0	RW 1C–0	RW 1C–0	RW 1C–0	RW 1C–0

Note: *R = read access, W1C = write 1 to clear, -0 = value after reset.*

)

Bits 15–11 Reserved. Reads return zero; writes have no effect.

Bit 10 **T3OFINT FLAG**. GP Timer 3 overflow interrupt.

Read:	0	Flag is reset
	1	Flag is set
Write:	0	No effect
	1	Resets flag

Bit 9 **T3UFINT FLAG**. GP Timer 3 underflow interrupt.

Read:	0	Flag is reset
	1	Flag is set
Write:	0	No effect
	1	Resets flag

Bit 8 **T3CINT FLAG**. GP Timer 3 compare interrupt.

Read:	0	Flag is reset
	1	Flag is set
Write:	0	No effect
	1	Resets flag

Bit 7 **T3PINT FLAG.** GP Timer 3 period interrupt.

Read:	0	Flag is reset
	1	Flag is set
Write:	0	No effect
	1	Resets flag

Bits 6–4 Reserved. Reads return zero; writes have no effect.

Bit 3 **CMP6INT FLAG**. Compare 6 interrupt.

Read:	0	Flag is reset
	1	Flag is set
Write:	0	No effect
	1	Resets flag

Bit 2 **CMP5INT FLAG**. Compare 5 interrupt.

Read:	0	Flag is reset
	1	Flag is set
Write:	0	No effect
	1	Resets flag

Bit 1 **CMP4INT FLAG**. Compare 4 interrupt.

Read:	0	Flag is reset
	1	Flag is set
Write:	0	No effect
	1	Resets flag

Bit 0 **PDPINTB FLAG**. Power drive protection interrupt.

Read:	0	Flag is reset
	1	Flag is set
Write:	0	No effect
	1	Resets flag

EVB Interrupt Flag Register B (EVBIFRB) — Address 7530h

15-4	3	2	1	0
Reserved	T4OFINT FLAG	T4UFINT FLAG	T4CINT FLAG	T4PINT FLAG
R–0	RW 1C–0	RW 1C–0	RW 1C–0	RW 1C–0

Note: *R = read access, W1C = write 1 to clear, -0 = value after reset.*

Bits 15–4 **Reserved**. Reads return zero; writes have no effect.

Bit 3 **T4OFINT FLAG**. GP Timer 4 overflow interrupt.

Read:	0	Flag is reset
	1	Flag is set
Write:	0	No effect
	1	Resets flag

Bit 2 **T4UFINT FLAG**. GP Timer 4 underflow interrupt.

Read:	0	Flag is reset
	1	Flag is set
Write:	0	No effect
	1	Resets flag

Bit 1 **T4CINT FLAG**. GP Timer 4 compare interrupt.

Read:	0	Flag is reset
	1	Flag is set
Write:	0	No effect
	1	Resets flag

Bit 0 **T4PINT FLAG**. GP Timer 4 period interrupt.

Read:	0	Flag is reset
	1	Flag is set
Write:	0	No effect
	1	Resets flag

EVB Interrupt Flag Register C (EVBIFRC) — Address 7531h

15-3	2	1	0
Reserved	CAP6INT FLAG	CAP5INT FLAG	CAP4INT FLAG
R–0	RW 1C–0	RW 1C–0	RW 1C–0

Note: *R = read access, W1C = write 1 to clear, -0 = value after reset.*

Bits 15–3 **Reserved**. Reads return zero; writes have no effect.

Bit 2 **CAP6INT FLAG**. Capture 6 interrupt.

Read:	0	Flag is reset
	1	Flag is set
Write:	0	No effect
	1	Resets flag

Bit 1 **CAP5INT FLAG**. Capture 5 interrupt.

Read:	0	Flag is reset
	1	Flag is set
Write:	0	No effect
	1	Resets flag

Bit 0 **CAP4INT FLAG**. Capture 4 interrupt.

Read:	0	Flag is reset
	1	Flag is set
Write:	0	No effect
	1	Resets flag

EVB Interrupt Mask Register A (EVBIMRA) — Address 752Ch

15-11	10	9	8
Reserved	T3OFINT ENABLE	T3UFINT ENABLE	T3CINT ENABLE
R–0	RW –0	RW –0	RW –0

7	6-4	3	2	1	0
T3PINT ENABLE	Reserved	CMP6INT ENABLE	CMP5INT ENABLE	CMP4INT ENABLE	PDPINTB ENABLE
RW –0	R–0	RW –0	RW –0	RW –0	RW –1

Note: *R = read access, W = write access, -n = value after reset.*

Bits 15–11 Reserved. Reads return zero; writes have no effect.

Bit 10 **T3OFINT ENABLE**
0 Disable
1 Enable

Bit 9 **T3UFINT ENABLE**
0 Disable
1 Enable

Bit 8 **T3CINT ENABLE**
0 Disable
1 Enable

Bit 7 **T3PINT ENABLE**
0 Disable
1 Enable

Bits 6–4 Reserved. Reads return zero; writes have no effect.

Bit 3 **CMP6INT ENABLE**
0 Disable
1 Enable

Bit 2 **CMP5INT ENABLE**
0 Disable
1 Enable

Bit 1 **CMP4INT ENABLE**
0 Disable
1 Enable

Bit 0 **PDPINTB ENABLE.** This is enabled (set to 1) following reset.
0 Disable
1 Enable

EVB Interrupt Mask Register B (EVBIMRB) — Address 752Dh

15-4	3	2	1	0
Reserved	T4OFINT ENABLE	T4UFINT ENABLE	T4CINT ENABLE	T4PINT ENABLE
R–0	RW –0	RW –0	RW –0	RW –0

Note: *R = read access, W = write access, -0 = value after reset.*

Bits 15–4 **Reserved**. Reads return zero; writes have no effect.

Bit 3 **T4OFINT ENABLE**
0 Disable
1 Enable

Bit 2 **T4UFINT ENABLE**
0 Disable
1 Enable

Bit 1 **T4CINT ENABLE**
0 Disable
1 Enable

Bit 0 **T4PINT ENABLE**
0 Disable
1 Enable

EVB Interrupt Mask Register C (EVBIMRC) — Address 752Eh

15-3	2	1	0
Reserved	CAP6INT ENABLE	CAP5INT ENABLE	CAP4INT ENABLE
R–0	RW –0	RW –0	RW –0

Note: *R = read access, W = write access, -0 = value after reset.*

Bits 15–3 **Reserved**. Reads return zero; writes have no effect.

Bit 2 **CAP6INT ENABLE**
0 Disable
1 Enable

Bit 1 **CAP5INT ENABLE**
0 Disable
1 Enable

Bit 0 **CAP4INT ENABLE**
0 Disable
1 Enable

General Purpose (GP) Timers

A General Purpose (GP) timer is simply a 16-bit counter, which may be configured to count up, down, or continuously up and down. There are two GP Timers in each EV: Timer1 and Timer2 for EVA and Timer3 and Timer4 for EVB.

All timers use the CPU clock as a general timing reference, but each individual timer may use a “pre-scaled” or frequency reduced time base which is specified in each timer’s control register.

A GP Timer may also be configured to generate an interrupt or trigger another peripheral on certain events such as a timer overflow (timer reached period value), underflow (timer reached zero), or compare (timer value reached compare value). Some examples of uses for the GP Timers include: setting the sampling period for the ADC by triggering the start of conversion; or providing the switching period for the generation of a PWM signal.

Figure 6.3 shows a block diagram of a GP Timer. There are two cases that apply to Fig. 6.3:

1. When “x” = 2, “y” = 1 and “n” = 2
2. When “x” = 4, “y” = 3 and “n” = 4

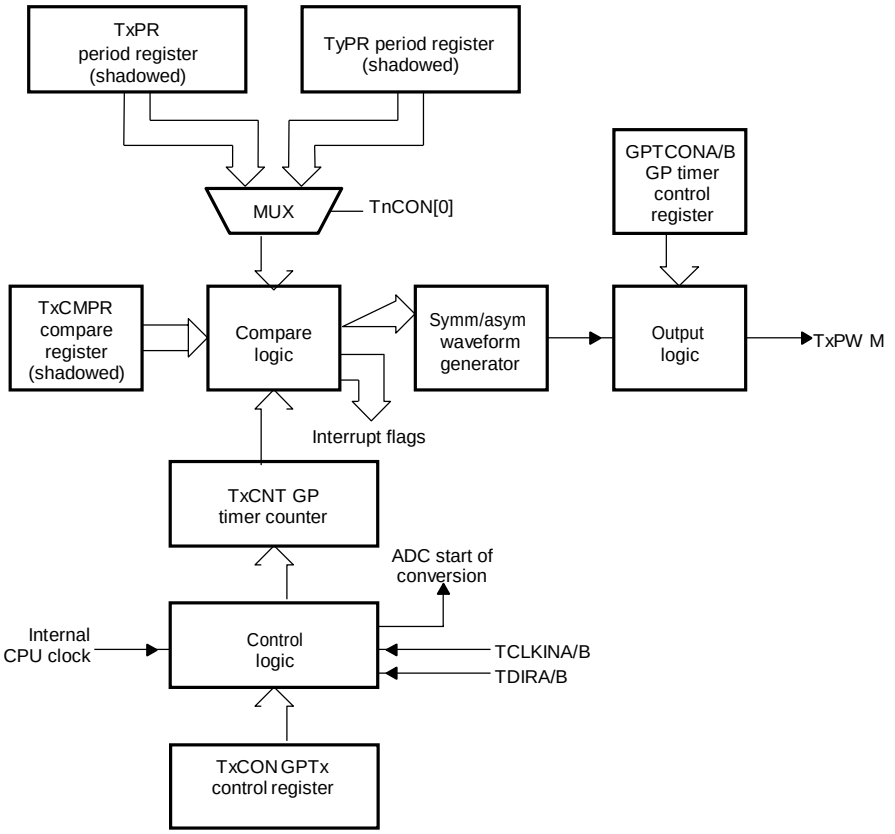


Figure 6.3 General purpose timer configuration diagram.

memory mapped register, another peripheral, or an external pin of the LF2407. Each GP Timer has the following outputs:

- x GP Timer compare outputs TxCMP, x = 1, 2, 3, 4 (external pins on the LF2407)
- x ADC start-of-conversion signal (connected to the ADC module)
- x Underflow, overflow, compare match, and period match signals to its own compare logic and to the compare units (connected to the compare units of the EV)
- x Counting direction indication bits (in the GPTCONA/B registers mapped to data memory)

The General Purpose Timer Control Register (GPTCONA/B), configures the action to be taken by the timers on different timer events, and indicates the counting directions of the GP Timers. GPTCONA/B is readable and writeable, although writing to the status bits in this register has no effect.

2. GP Counting Operation

GP Timers have four possible modes of counting operation:

1. Stop/Hold mode
2. Continuous Up-Counting mode
3. Directional Up/Down-Countingmode
4. Continuous Up/Down-Countingmode

Each timer is configured for desired counting mode in its corresponding Timer Control register (TxCON). Each GP Timer is enabled by setting the Timer Enable bit each timer's control register. When the timer is enabled, the timer counts according to the counting mode specified by the bits in the TxCON. The counting direction of the GP Timers are reflected by their respective bit in GPTCONA/B. When the timer is disabled (enable bit=0), counting is disabled and the prescaler of that timer is reset to the default value of "x/1".

Stop/Hold mode is like the "pause" button for the timer. In stop/hold mode the GP Timer stops and holds at its current state. The timer counter, the compare output, and the pre-scale select all remain unchanged.

Continuous Up-Counting Mode:

The continuous up-counting mode is useful in creating asymmetric PWM signals. In the continuous up-count mode the following events occur:

1. The GP Timer in this mode counts up in sync with the pre-scaled input clock until the value of the timer counter matches that of the period register.
2. On the next rising edge of the input clock after the match, the GP Timer resets to zero and starts counting up again.
3. The period interrupt flag of the timer is set one clock cycle after the match between the timer counter and period register. If the flag is not masked, a

peripheral interrupt request is generated. An ADC start is sent to the ADC module at the same time the flag is set if the period interrupt of this timer has been selected by the appropriate bits in GPTCONA/B to start the ADC.

4. One clock cycle after the GP Timer becomes 00001, the underflow interrupt flag of the timer is set. A peripheral interrupt request is generated by the flag if it is unmasked. An ADC start is sent to the ADC module at the same time if the underflow interrupt flag of this timer has been selected by the appropriate bits in the GPTCONA/B to start the ADC.

The duration of the timer period is (TxPR) + 1 cycles of the scaled clock input except for the first period. The duration of the first period is the same if the timer counter is zero when counting starts. The initial value of the GP Timer can be any value from 0h to FFFFh. When the initial value is greater than the value in the period register, the timer counts up to FFFFh, resets to zero, and continues the operation as if the initial value was zero. The overflow interrupt flag is set one clock cycle after the value in TxCNT matches FFFFh. A peripheral interrupt request is generated by the flag if it isunmasked.

When the initial value in the timer counter is the same as that of the period register, the timer sets the period interrupt flag, resets to zero, sets the underflow interrupt flag, and then continues the operation again as if the initial value was zero. If the initial value of the timer is between zero and the contents of the period register, the timer counts up to the period value and continues to finish the period as if the initial counter value was the same as that of the period register.

The counting direction indication bit in GPTCONA/B is “1” for the timer in this mode. Either the external or internal device clock can be selected as the input clock to the timer. The TDIRA/B input is ignored by the GP Timer in this mode since we are in an up-count only mode. The continuous up-count mode of the GP Timer is particularly useful for the generation of edge-triggered or asynchronous PWM waveforms and sampling periods in many motor and motion control systems. Figure shows the continuous up-counting mode of the GP Timer.

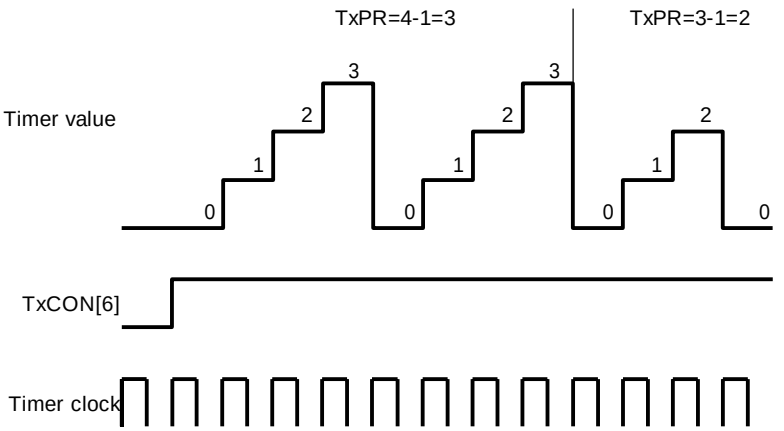


Figure 6.4 Operation of continuous up-counting mode (TxPR = 3 or 2).

Directional Up/Down-Counting Mode:

A GP Timer in directional up/down-counting mode counts either up or down according to the pre-scaled clock and TDIRA/B inputs. The input pin TDIRA/B determines the direction of counting when the GP Timer is in directional up/down-counting mode. When TDIRA/B is high, upward counting is specified; when TDIRA/B is low, downward counting is specified.

When the TDIRA/B pin is held high, the GP Timer will count up until it reaches the value of the period register. When the timer value equals that of its period register the timer will reset to zero and start counting up to the period again. The initial value of the timer can be any value between 0000h to FFFFh. In the case that the initial value of the timer counter is greater than that of the period register, the timer would count up to FFFFh before resetting itself to zero and continuing the counting operation. When TDIRA/B pin is held low, the GP Timer will count down from whatever initial value the counter was at until its count value becomes zero. When its count value becomes zero, the value of the period register is automatically loaded into the count value register and the timer begins counting down to zero.

In the directional up/down mode, the period, underflow, and overflow interrupt flags, interrupts, and associated actions are generated on respective events in the same manner as they are generated in the continuous up-counting mode. The direction of counting is indicated for the timer in this mode by the corresponding direction indication bit in GPTCONA/B: 1 means counting up; 0 means counting down. Either the external clock from the TCLKINA/B pin or the internal device clock can be used as the input clock for the timer in this mode. Figure 6.5 shows the directional up-/down-counting mode of the GPTimers.

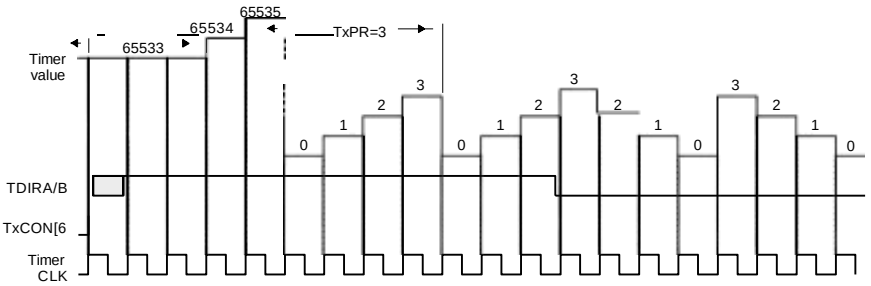


Figure 6.5 GP timer directional up/down-counting mode: prescale factor 1 and TxPR = 3A.

Additionally, the directional up-/down-counting mode of GP Timer 2 and 4 can also be used with the Quadrature Encoder Pulse (QEP) circuits in the EV module. While the QEP circuits are active, they provide both the counting clock and direction for GP Timers 2 or 4.

)

Continuous Up/Down-Counting Mode

The continuous up/down-counting mode is useful in generating symmetric PWM waveforms. This mode of operation is the same as the directional up-/down-counting mode, except for the fact that the TDIRA/B pin has no effect on the counting direction. The counting direction changes from up to down when the timer reaches the period value. The timer direction changes from down to up when the timer reaches zero. Continuous up/down-counting mode is particularly useful in generating centered or symmetric PWM waveforms.

The initial value of the GP Timer counter can be any value from 0h to FFFFh. When the initial value is greater than that of the period register (TxPR), the timer counts up to FFFFh, resets to zero, and continues the operation as if the initial value were zero. If the initial value of the timer counter is the same as that of the period register, the timer counts down to zero and continues again as if the initial value were zero. If the initial value of the timer is between zero and the contents of the period register, the timer will count up to the period value and continue to finish the period as if the initial counter value were the same as that of the period register.

The counting direction indication bit in the GPTCONA/B indicates "1" when the timer counts upward and "0" when the timer is counting downward. Either an external clock reference from the TCLKINA/B pin or the internal CPU clock can be selected as the input clock. Since the change of count direction is automatic in this mode, the TDIRA/B pin has no effect. The period, underflow, and overflow interrupt flags, interrupts, and associated actions are generated on the respective events in the same manner as they are generated in other counting modes. Figure 6.6 shows the continuous up-/down-counting mode of the GP Timer.

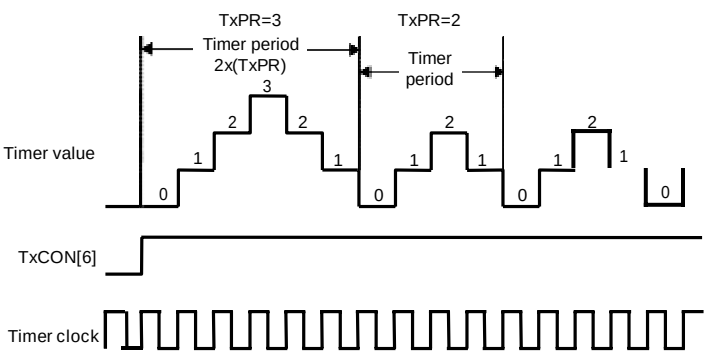


Figure 6.6 Continuous up/down counting mode (timer period register = 3 or 2).

Note: *The period of the timer in this mode is 2*(TxPR) cycles of the scaled clock input, except for the first period.*

3. Control Registers Associated with the General Purpose Timers

Individual Timer Control Registers (TxCON), where x=1,2,3,4

The operational mode of each GP Timer is controlled by the timer's corresponding control register (TxCON). The bits in the TxCON configure:

1. What counting mode the timer is set for
2. Whether the internal (CPU) or an external clock is to be used for the clock reference
3. Which of the eight input clock pre-scale factors (ranging from 1/1 to 1/128) is used
4. When (on which condition) the timer compare register is reloaded
5. Whether the timer is enabled or disabled
6. Whether the timer compare operation is enabled or disabled
7. Which period register is used by timer 2 (its own, or timer 1's period register (EVA))
8. Which period register is used by timer 4 (its own, or timer 3's period register (EVB))

In EVA, GP Timer 2 can be synchronized with GP Timer 1. Additionally, in EVB, GP Timer 4 can be synchronized with GP Timer 3 by configuring T2CON and T4CON, respectively, in the following ways:

EVA:

1. Set the T2SWT1 bit in T2CON to start GP Timer 2 counting with the TENABLE bit in T1CON (both timer counters start simultaneously)
2. Initialize the timer counter in GP Timers 1 and 2 with different values before starting synchronized operation
3. Specify that GP Timer 2 uses the period register of GP Timer 1 as its period register (ignoring its own period register) by setting SELT1PR in T2CON

EVB:

1. Set the T4SWT3 bit in T4CON to start GP Timer 4 counting with the TENABLE bit in T3CON (thus, both timer counters start simultaneously)
2. Initialize the timer counters in GP Timers 3 and 4 with different values before starting synchronized operation
3. Specify that GP Timer 4 uses the period register of GP Timer 3 as its period register (ignoring its own period register) by setting SELT3PR in T4CON

This allows the desired synchronization between GP Timer events. Since each GP Timer starts the counting operation from its current value in the counter register, one GP Timer can be programmed to start with a known delay after the other GP Timer.

)

Timer x Control Register Bit Descriptions (TxCON; x = 1, 2, 3, or 4) —

Addresses: 7404h (T1CON), 7408h (T2CON), 7504h (T3CON), and 7508h (T4CON)

15	14	13	12	11	10	9	8
Free	Soft	Reserved	TMODE1	TMODE0	TPS2	TPS1	TPS0
RW –0	RW –0	RW –0	RW –0	RW –0	RW –0	RW –0	RW –0
7	6	5	4	3	2	1	0
T2SW T1/ T4SW T3†	TENABLE	TCLKS1	TCLKS0	TCLD1	TCLD0	TECMPR	SELT1PR/ SELT3PR†
RW –0	RW –0	RW –0	RW –0	RW –0	RW –0	RW –0	RW –0

† Reserved in T1CON andT3CON

Note: *R = read access, W = write access, -0 = value after reset.*

Bits 15–14 Free, Soft. Emulation control bits.

00 Stop immediately on emulation suspend
01 Stop after current timer period is complete on emulation suspend
10 Operation is not affected by emulation suspend
11 Operation is not affected by emulation suspend

Bit 13 Reserved. Reads return zero, writes have no effect.

Bits 12–11 TMODE1–TMODE0. Count Mode Selection.

00 Stop/Hold
01 Continuous-Up/-Down Count Mode
10 Continuous-Up Count Mode
11 Directional-Up/-Down Count Mode

Bits 10–8 TPS2–TPS0.

Input Clock Prescaler.
000=x/1 , 001=x/2, 010=x/4, 011=x/8, 100=x/16, 101=x/32,110=x/64
111=x/128 ; x = device (CPU) clock frequency

Bit 7 T2SWT1. In the case of EVA, this bit is T2SWT1. (GP Timer 2 start with GP Timer 1.) Start GP Timer 2 with GP Timer 1's timer enable bit. This bit is reserved in T1CON.

T4SWT3. In the case of EVB, this bit is T4SWT3. (GP Timer 4 start with GP Timer 3.) Start GP Timer 4 with GP Timer 3's timer enable bit. This bit is reserved in T3CON.

0 Use own TENABLE bit
1 Use TENABLE bit of T1CON (in case of EVA) or T3CON (in case of EVB) to enable and disable operation ignoring own TENABLE bit

Bit 6 **TENABLE**. Timer enable.

0	Disable timer operation (the timer is put in hold and the prescaler counter is reset)
1	Enable timer operations

Bits 5–4 TCLKS1, TCLKS0. Clock Source Select.

5	4	Source
0	0	Internal
0	1	External
1	0	Reserved
1	1	QEP Circuit† (in case of Timer 2/Timer 4) Reserved (in case of Timer 1/Timer 3)
†		This option is valid only if SELT1PR = 0

Bits 3–2 TCLD1, TCLD0. Timer Compare Register Reload Condition.

00	When counter is 0
01	When counter value is 0 or equals period register value
10	Immediately
11	Reserved

Bit 1 **TECMPR**. Timer compare enable.

0	Disable timer compare operation
1	Enable timer compare operation

Bit 0 **SELT1PR**. In the case of EVA, this bit is SELT1PR (Period register select).

When set to 1 in T2CON, the period register of Timer 1 is chosen for Timer 2 also, ignoring the period register of Timer 2. This bit is a reserved bit in T1CON. **SELT3PR**. In the case of EVB, this bit is SELT3PR (Period register select). When set to 1 in T4CON, the period register of Timer 3 is chosen for Timer 4 also, ignoring the period register of Timer 4. This bit is a reserved bit in T3CON.

0	Use own period register
1	Use T1PR (in case of EVA) or T3PR (in case of EVB) as period register ignoring own period register

Overall GP Timer Control Registers (GPTCONA/B)

The control register GPTCONA/B specifies the action to be taken by the timers on different timer events. This register also has timer direction status bits that display the current direction of the timers. Also, the polarity of the GP Timer compare outputs is configured here. Bits in GPTCONA/B can also configure specific timers to trigger an ADC start signal when an underflow, compare match, or period match occurs. This feature requires that the ADC also be configured to

)

accept the start of conversion signal from the GP Timer. Having the GP Timer trigger provides for automatic synchronization between the GP Timer event and the ADC.

GP Timer Control Register A (GPTCONA) Bit Descriptions — Address 7400h

15	14	13	12-11	10-9	8-7
Reserved	T2STAT	T1STAT	Reserved	T2TOADC	T1TOADC
RW –0	R–1	R–1	RW –0	RW –0	RW –0

6	5-4	3-2	1-0
TCOMPOE	Reserved	T2PIN	T1PIN
RW –0	RW –0	RW –0	RW –0

Note: *R = read access, W = write access, -n = value after reset.*

Bit 15 **Reserved**. Reads return zero; writes have no effect.

Bit 14 **T2STAT**. GP Timer 2 Status. Read only.
0 Countingdownward
1 Counting upward

Bit 13 **T1STAT**. GP Timer 1 Status. Read only.
0 Countingdownward
1 Counting upward

Bits 12–11 Reserved. Reads return zero; writes have no effect.

Bits 10–9 T2TOADC. Start ADC with timer 2 event.
00 No event starts ADC
01 Setting of underflow interrupt flag starts ADC
10 Setting of period interrupt flag starts ADC
11 Setting of compare interrupt flag starts ADC

Bits 8–7 T1TOADC. Start ADC with timer 1 event.
00 No event starts ADC
01 Setting of underflow interrupt flag starts ADC
10 Setting of period interrupt flag starts ADC
11 Setting of compare interrupt flag starts ADC

Bit 6 **TCOMPOE**. Compare output enable. If PDPINTx is active this bit is set to zero.
0 Disable all GP Timer compare outputs (all compare outputs are put in the high-impedance state)
1 Enable all GP Timer compare outputs

Bits 5–4 Reserved. Reads return zero; writes have no effect.

Bits 3–2 T2PIN. Polarity of GP Timer 2 compare output.

00	Forced low
01	Active low
10	Active high
11	Forced high

Bits 1–0 T1PIN. Polarity of GP Timer 1 compare output.

00	Forced low
01	Active low
10	Active high
11	Forced high

GP Timer Control Register B (GPTCONB) Bit Descriptions — Address 7500h

15	14	13	12-11	10-9	8-7
Reserved	T4STAT	T3STAT	Reserved	T4TOADC	T3TOADC
RW–0	R–1	R–1	RW–0	RW–0	RW–0

5-	3-	1-0
TCOMPOE	Reserved	T4PIN	T3PIN
RW–0	RW–0	RW–0	RW–0

Note: *R = read access, W = write access, -n = value after reset.*

Bit 15 **Reserved**. Reads return zero; writes have no effect.

Bit 14 **T4STAT**. GP Timer 4 Status. Read only.

0	Countingdownward
1	Counting upward

Bit 13 **T3STAT**. GP Timer 3 Status. Read only.

0	Countingdownward
1	Counting upward

Bits 12–11 Reserved. Reads return zero; writes have no effect.

Bits 10–9 T4TOADC. Start ADC with timer 4 event.

00	No event starts ADC
01	Setting of underflow interrupt flag starts ADC
10	Setting of period interrupt flag starts ADC
11	Setting of compare interrupt flag starts ADC

Bits 8–7 T3TOADC. Start ADC with timer 3 event.

00	No event starts ADC
01	Setting of underflow interrupt flag starts ADC

)

10 Setting of period interrupt flag starts ADC
11 Setting of compare interrupt flag starts ADC

Bit 6 **TCOMPOE**. Compare output enable. If PDPINTx is active this bit is set to zero.
0 Disable all GP Timer compare outputs (all compare outputs are put in the high-impedance state)
1 Enable all GP Timer compare outputs

Bits 5–4 Reserved. Reads return zero; writes have no effect.

Bits 3–2 T4PIN. Polarity of GP Timer 4 compare output.
00 Forced low
01 Active low
10 Active high
11 Forced high

Bits 1–0 T3PIN. Polarity of GP Timer 3 compare output.
00 Forced low
01 Active low
10 Active high
11 Forced high

GP Timer Compare Registers (TxCMPR), x=1,2,3,4 – User Specified Value

Addresses 7402h (T1CMPR), 7406h (T2CMPR), 7502h (T3CMPR), 7506h (T4CMPR)

The compare register associated with each GP Timer stores the value that will be constantly compared with the current value of the GP Timer. When a compare match occurs, the following events also occur:

1. A transition occurs on the associated compare output according to the bit pattern in GPTCONA/B
2. The corresponding interrupt flag is set
3. A peripheral interrupt request is generated if the interrupt is unmasked
4. The compare operation of a GP Timer can be enabled or disabled by the appropriate bit in TxCON
5. The compare operation and outputs can be enabled in any of the timer counting modes, including the QEP circuit

GP Timer Period Registers (TxPR) – User Specified Value

Addresses 7403h (T1PR), 7407h (T2PR), 7503h (T3PR), 7507h (T4PR)

The period register determines the rate at which the timer resets itself or changes direction (the period of the timer). This register in combination with the input clock frequency (and clock pre-scale factor) determines the frequency of a

the same compare match. The PWM outputs associated with the compare units allow for the generation of six PWM outputs per EV.

As shown in Fig. 6.10 the Compare Units Include:

- Three 16-bit compare registers (CMPR1, CMPR2, and CMPR3 for EVA; and CMPR4, CMPR5, and CMPR6 for EVB), all double-buffered
- One 16-bit compare control register (COMCONA for EVA, and COM-CONB for EVB)
- One 16-bit action control register (ACTRA for EVA, and ACTRB for EVB), with an associated buffer register
- Six PWM (3-state; Low, High, High Z) output (compare output) pins (PWMy, y = 1, 2, 3, 4, 5, 6 for EVA and PWMz, z = 7, 8, 9, 10, 11, 12 for EVB)

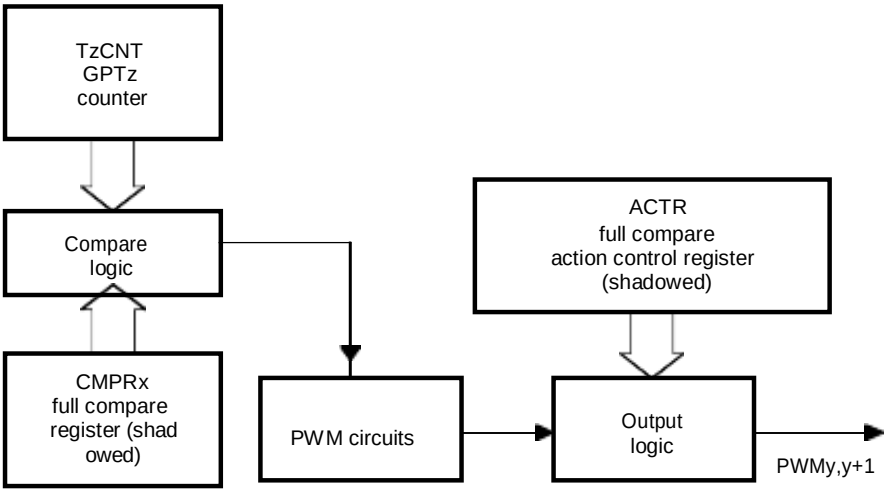


Figure 6.10 Compare unit block diagram.

For EVA: x = 1, 2, 3; y = 1, 3, 5; z = 1

For EVB: x = 4, 5, 6; y = 7, 9, 11; z = 3

Inputs and Outputs of the Compare Units

The inputs to a compare unit include:

- Control signals from compare control registers
- GP Timer 1/3 (T1CNT/T3CNT) count value, underflow, and period match signals
- System RESET
- The time base (counter value) for the compare units in EVA (CMPR1,2 ,3) is GP Timer 1, and for EVB (CMPR4, 5, 6) is GP Timer3.

The solution to this problem is to make sure that only one transistor in each leg is on at a time. In theory, this is accomplished by feeding complementary PWM gating signals to each of the two transistors in a leg. So when one transistor is on, the other is off. In reality, all transistors turn on faster than they turn off. Therefore, it is necessary to add a time delay (dead-band) between the PWM signals to allow for the first transistor to fully turn off before the second one is turned on.

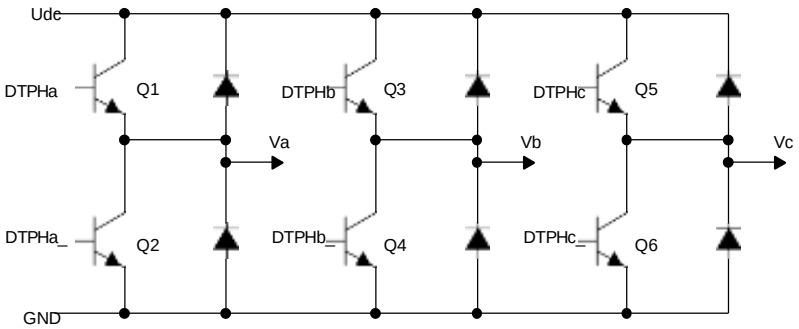


Figure 6.11 Basic three-phase inverter circuit.

6.4.3 Dead Band Generation

Unlike the GP Timer Compare PWM generation, the compare unit PWM outputs allow for a programmable dead band. Each EV on the LF2407 has its own programmable dead-band unit. The dead-band generators generate the dead-band delay between the toggling of the independent and dependent PWM outputs. Dead band solves the problem of inverter leg shoot through (short circuits). Figure 6.12 shows the interconnection between the dead band units and the compare units.

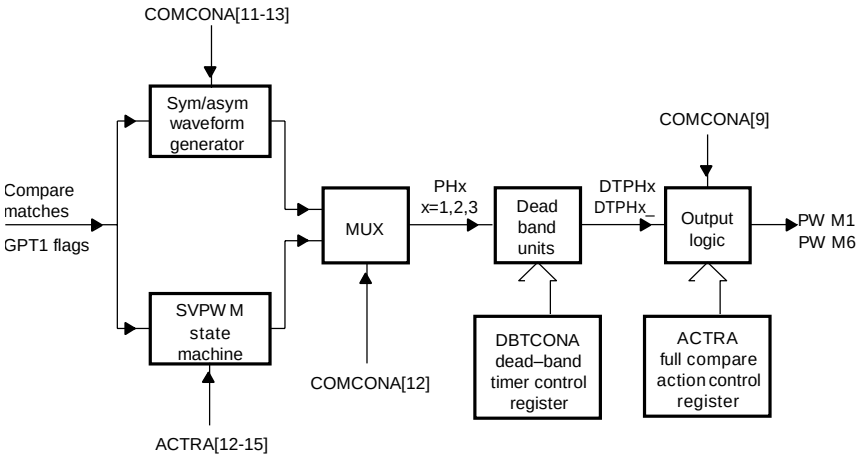


Figure 6.12 Block diagram of PWM outputs showing dead-band units.

Compare Control Register A (COMCONA) — Address 7411h

15	14	13	12	11	10	9	8
CENABLE	CLD1	CLD0	SVENABLE	ACTRLD1	ACTRLD0	FCOMPOE	*PDPINTA* STATUS
RW –0	RW –0	RW –0	RW –0	RW –0	RW –0	RW –0	R– $\overline{PDPINTA}$ PIN
7 - 0							
Reserved							
R– 0							

Note: *R = read access, W = write access, -0 = value after reset.*

Bit 15 CENABLE. Compare enable.

0 Disables compare operation. All shadowed registers (CMPRx, ACTRA) become transparent

1 Enables compare operation

Bits14–13 CLD1, CLD0. Compare register CMPRx reload condition.

00 When T1CNT = 0 (that is, on underflow)

01 When T1CNT = 0 or T1CNT = T1PR (that is, on underflow or period match)

10 Immediately

11 Reserved; result is unpredictable

Bit 12 SVENABLE. Space vector PWM mode enable.

0 Disables space vector PWM mode

1 Enables space vector PWM mode

Bits 11–10 ACTRLD1, ACTRLD0. Action control register reload condition.

00 When T1CNT = 0 (on underflow)

01 When T1CNT = 0 or T1CNT = T1PR (on underflow or period match)

10 Immediately

11 Reserved

Bit 9 FCOMPOE. Compare output enable. Active PDPINTA clears this bit to zero.

0 PWM output pins are in high-impedance state; that is, they are disabled

1 PWM output pins are not in high-impedance state; that is, they are enabled

Bit 8 PDPINTA STATUS. This bit reflects the current status of the PDPINTA pin. (This bit is applicable to 240xA devices only — it is reserved on 240x devices and returns a zero when read.)

Bits 7–0 **Reserved**. Read returns zero; writes have no effect.

Compare Control Register B (COMCONB) — Address 7511h

15	14	13	12	11	10	9	8
CENABLE	CLD1	CLD0	SVENABLE	ACTRLD1	ACTRLD0	FCOMPOE	$\overline{PDPINTB}$ STATUS
RW –0	RW –0	RW –0	RW –0	RW –0	RW –0	RW –0	R– $\overline{PDPINTB}$ PIN
7-0							
Reserved							
R–0							

Note: *R = read access, W = write access, -0 = value after reset.*

Bit 15 **CENABLE**. Compare enable.

0 Disable compare operation. All shadowed registers (CMPRx, ACTRB) become transparent

1 Enable compare operation

Bits14–13 **CLD1, CLD0**. Compare register CMPRx reload condition.

00 When T3CNT = 0 (that is, on underflow)

01 When T3CNT = 0 or T3CNT = T3PR (that is, on underflow or period match)

10 Immediately

11 Reserved; result is unpredictable

Bit 12 **SVENABLE**. Space vector PWM mode enable.

0 Disables space vector PWM mode

1 Enables space vector PWM mode

Bits 11–10 **ACTRLD1, ACTRLD0**. Action control register reload condition.

00 When T3CNT = 0 (on underflow)

01 When T3CNT = 0 or T3CNT = T3PR (on underflow or period match)

10 Immediately

11 Reserved

Bit 9 **FCOMPOE**. Compare output enable. Active PDPINTB clears this bit to zero.

0 PWM output pins are in high-impedance state; that is, they are disabled

1 PWM output pins are not in high-impedance state; that is, they are enabled

Bits 7–6 CMP4ACT1–0. Action on compare output pin 4, CMP4.

00 Forced low
01 Active low
10 Active high
11 Forced high

Bits 5–4 CMP3ACT1–0. Action on compare output pin 3, CMP3.

00 Forced low
01 Active low
10 Active high

Bits 3–2 CMP2ACT1–0. Action on compare output pin 2, CMP2.

00 Forced low
01 Active low
10 Active high
11 Forced high

Bits 1–0 CMP1ACT1–0. Action on compare output pin 1, CMP1.

00 Forced low
01 Active low
10 Active high
11 Forced high

Compare Action Control Register B (ACTRB) — Address 7513h

15	14	13	12	11	10	9	8
SVRDIR	D2	D1	D0	CMP12ACT1	CMP12ACT0	CMP11ACT1	CMP11ACT0
RW –0	RW –0	RW –0	RW –0	RW –0	RW –0	RW –0	RW –0
7	6	5	4	3	2	1	0
CMP10ACT1	CMP10ACT0	CMP9ACT1	CMP9ACT0	CMP8ACT1	CMP8ACT0	CMP7ACT1	CMP7ACT0
RW –0	RW –0	RW –0	RW –0	RW –0	RW –0	RW –0	RW –0

Note: *R = read access, W = write access, -0 = value after reset.*

Bit 15 SVRDIR. Space vector PWM rotation direction. Used only in space vector PWM output generation.

0 Positive (CCW)
1 Negative (CW)

Bits 14–12 D2–D0. Basic space vector bits. Used only in space vector PWM output generation.

Bits 11–10 CMP12ACT1–0. Action on compare output pin 12, CMP12.

00 Forced low
01 Active low

10 Active high
11 Forced high

Bits 9–8 CMP11ACT1–0. Action on compare output pin 11, CMP11.
00 Forced low
01 Active low
10 Active high
11 Forced high

Bits 7–6 CMP10ACT1–0. Action on compare output pin 10, CMP10.
00 Forced low
01 Active low
10 Active high
11 Forced high

Bits 5–4 CMP9ACT1–0. Action on compare output pin 9, CMP9.
00 Forced low
01 Active low
10 Active high
11 Forced high

Bits 3–2 CMP8ACT1–0. Action on compare output pin 8, CMP8.
00 Forced low
01 Active low
10 Active high
11 Forced high

Bits 1–0 CMP7ACT1–0. Action on compare output pin 7, CMP7.
00 Forced low
01 Active low
10 Active high
11 Forced high

Dead-Band Timer Control Register A (DBTCONA) — Address 7415h

15-12				11	10	9	8
Reserved				DBT3	DBT2	DBT1	DBT0
R–0				RW –0	RW –0	RW –0	RW –0
7	6	5	4	3	2	1-0	
EDBT3	EDBT2	EDBT1	DBTPS2	DBTPS1	DBTPS0	Reserved	
RW –0	RW –0	RW –0	RW –0	RW –0	RW –0	R–0	

Note: *R = read access, W = write access, -0 = value after reset.*

Bits 15–12 Reserved. Reads return zero; writes have no effect.

Bits 11–8 DBT3 (MSB)–DBT0 (LSB). Dead-band timer period. These bits define the period value of the three 4-bit dead-band timers.

Bit 7 **EDBT3**. Dead-band timer 3 enable (for pins PWM5 and PWM6 of Compare Unit 3).
0 Disable
1 Enable

Bit 6 **EDBT2**. Dead-band timer 2 enable (for pins PWM3 and PWM4 of Compare Unit 2).
0 Disable
1 Enable

Bit 5 **EDBT1**. Dead-band timer 1 enable (for pins PWM1 and PWM2 of Compare Unit 1).
0 Disable
1 Enable

Bits 4–2 DBTPS2 to DBTPS0. Dead-band timer prescaler.
000 x/1
001 x/2
010 x/4
011 x/8
100 x/16
101 x/32
110 x/32
111 x/32
x = Device (CPU) clock frequency

Bits 1–0 Reserved. Reads return zero; writes have no effect.

Dead-Band Timer Control Register B (DBTCONB) — Address 7515h

15-12				11	10	9	8
Reserved				DBT3	DBT2	DBT1	DBT0
R–0				RW –0	RW –0	RW –0	RW –0
7	6	5	4	3	2	1-0	
EDBT3	EDBT2	EDBT1	DBTPS2	DBTPS1	DBTPS0	Reserved	
RW –0	RW –0	RW –0	RW –0	RW –0	RW –0	R–0	

Note: *R = read access, W = write access, -0 = value after reset.*

Bits 15–12 Reserved. Reads return zero; writes have no effect.

Bits 11–8 **DBT3 (MSB)–DBT0 (LSB)**. Dead-band timer period. These bits define the period value of the three 4-bit dead-band timers.

Bit 7 **EDBT3**. Dead-band timer 3 enable (for pins PWM11 and PWM12 of Compare 6).
0 Disable
1 Enable

Bit 6 **EDBT2**. Dead-band timer 2 enable (for pins PWM9 and PWM10 of Compare 5).
0 Disable
1 Enable

Bit 5 **EDBT1**. Dead-band timer 1 enable (for pins PWM7 and PWM8 of Compare 4).
0 Disable
1 Enable

Bits 4–2 DBTPS2 to DBTPS0. Dead-band timer prescaler.
000 x/1
001 x/2
010 x/4
011 x/8
100 x/16
101 x/32
110 x/32
111 x/32
x = Device (CPU) clock frequency

Bits 1–0 Reserved. Reads return zero; writes have no effect.

Capture Units and Quadrature Encoded Pulse (QEP) Circuitry

The Capture Units on the LF2407 allow an event (rising/falling edge) on the capture pin to be time stamped by a selected GP Timer. There are three Capture Units in each EV, each with its own capture input pin (CAPx). Capture Units 1, 2, and 3 are associated with EVA while Capture Units 4, 5, and 6 are associated with EVB. Each EV module contains the following (shown in Figs. 6.14 and 6.15):

- One 16 bit capture control register per EV (CAPCOMA for EVA, CAPCOMB for EVB) is used for configuring the Capture Unit functionality.
- Three 16-bit, 2-level-deep First-In-First-Out (FIFO) stacks per EV (CAPxFIFO); one FIFO stack for each Capture Unit; the “captured” timer count value is stored here.
- One 16-bit capture status register (CAPFIFOA for EVA, CAPFIFOB for EVB); provides information on the number of timer captures in each capture FIFO.

- Inputs of either GP Timer 1 or 2 (for EVA) and GP Timer 3 or 4 (for EVB) as the time base.
- One capture pin per Capture Unit with user-specified transition detection (rising edge, falling edge, or both edges). CAP1 through CAP3 for EVA, and CAP4 through CAP6 for EVB.
- Six maskable interrupt flags, one for each Capture Unit.

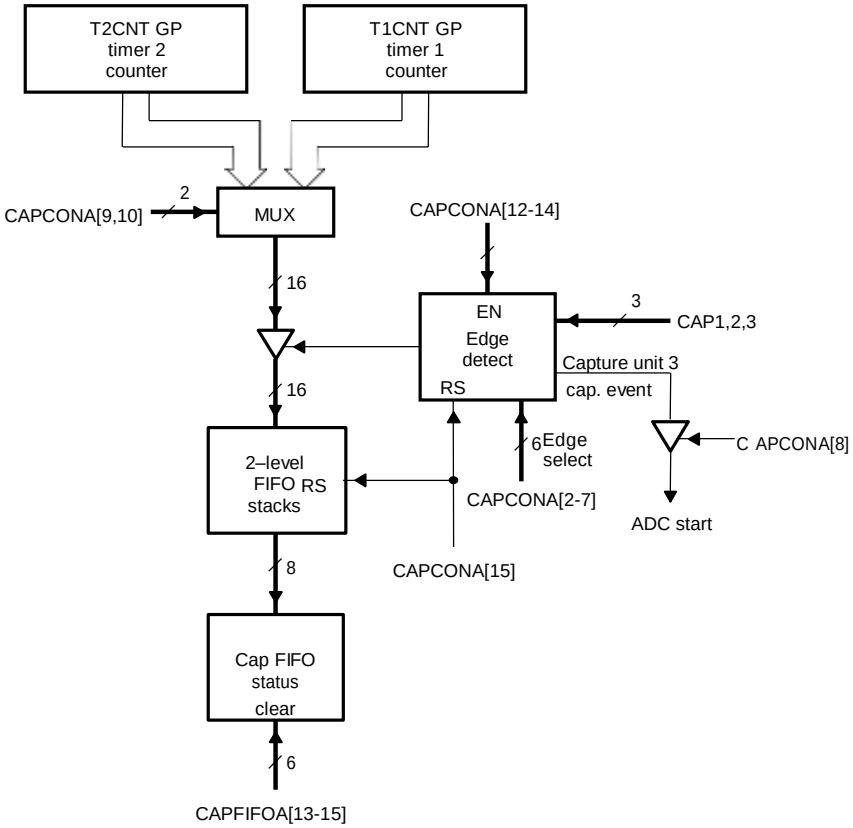


Figure 6.15 EVA capture unit diagram. (Courtesy of Texas Instruments)

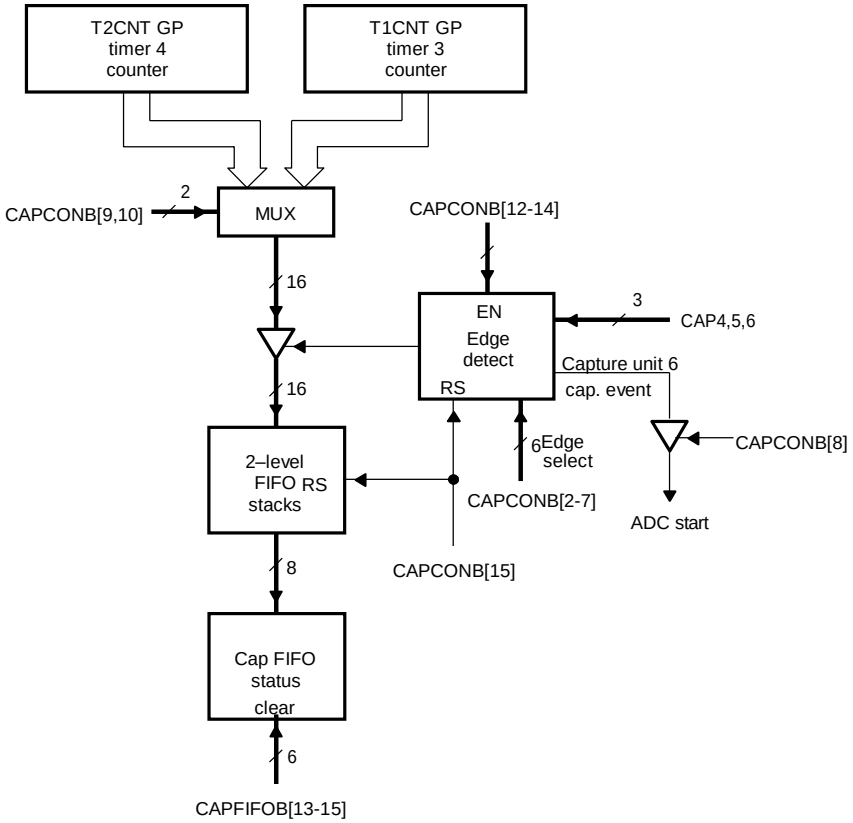


Figure 6.16 EVB Capture unit diagram. (Courtesy of Texas Instruments)

The Capture Units are useful in applications where the time of an external trigger needs to be "captured". For example, if we want to measure the time between the rising edges of two pulses, we would configure the appropriate registers for capture operation on a specific capture pin. At each rising edge, the Capture Unit would then store the corresponding timer values. The user program could then subtract the second capture value from the first value and determine the time between the pulses.

The Capture Units are accompanied by the Quadrature Encoded Pulse (QEP) circuitry which uses the GP Timers to "decode" a QEP signal. When the QEP mode is selected, pins CAP1 and CAP2 (CAP4 and CAP5 in case of EVB) are used as QEP inputs. More on the QEP circuitry will be discussed shortly.

EVB:

1. Load GP Timer 4's counter, period, and compare registers with desired values; for simple QEP decoding, this is not required.
2. Configure T4CON to set GP Timer 4 in directional-up/down mode with the QEP circuits as clock source, and enable the selected timer.
3. Configure CAPCONB to enable the QEP circuit.

Interrupt flags normally associated with the timer operation are still operational with the QEP. Period, underflow, overflow, and compare interrupt flags for a GP Timer with a QEP circuit clock are generated on respective matches. If the respective interrupt flags are unmasked, timer interrupt requests will be generated.

6.5.4 Capture Unit / QEP ControlRegisters

Upon a RESET, all capture registers are cleared to zero. There are four 16-bit registers that control the functionality of the Capture Units. These registers are CAPCONA, CAPCONB, CAPFIFOA, and CAPFIFOB. In addition to these four registers the individual timer control registers (TxCON, x = 1, 2, 3, or 4) control the selected timer which acts as the time base for the Capture Unit. CAPCONA and CAPCONB also control the QEP functionality.

Capture Control Register A (CAPCONA) — Address 7420h

15	14-13	12	11	10	9	8
CAPRES	CAPQEPN	CAP3EN	Reserved	CAP3TSEL	CAP12TSEL	CAP3TOADC
RW –0	RW –0	RW –0	R–0	RW –0	RW –0	RW –0

7-6	5-4	3-2	1-0
CAP1EDGE	CAP2EDGE	CAP3EDGE	Reserved
RW –0	RW –0	RW –0	R–0

Note: *R = read access, W = write access, -0 = value after reset.*

Bit 15 **CAPRES**. Capture reset. Always reads zero.
Note: This bit is not implemented as a register bit. Writing a 0 simply clears the capture registers.
0 Clear all registers of Capture Units and QEP circuit to 0
1 No action

Bits 14–13 CAPQEPN. Capture Units 1 and 2 control.
00 Disables Capture Units 1 and 2; FIFO stacks retain their contents
01 Enables Capture Units 1 and 2
10 Reserved
11 Reserved

Bit 12 **CAP3EN**. Capture Unit 3 control.

0	Disables Capture Unit 3; FIFO stack of Capture Unit 3 retains its contents
1	Enable Capture Unit 3

Bit 11 **Reserved**. Reads return zero; writes have no effect.

Bit 10 **CAP3TSEL**. GP Timer selection for Capture Unit 3.

0	Selects GP Timer 2
1	Selects GP Timer 1

Bit 9 **CAP12TSEL**. GP Timer selection for Capture Units 1 and 2.

0	Selects GP Timer 2
1	Selects GP Timer 1

Bit 8 **CAP3TOADC**. Capture Unit 3 event starts ADC.

0	No action
1	Starts ADC when the CAP3INT flag is set

Bits 7–6 **CAP1EDGE**. Edge detection control for Capture Unit 1.

00	No detection
01	Detects rising edge
10	Detects falling edge
11	Detects both edges

Bits 5–4 **CAP2EDGE**. Edge detection control for Capture Unit 2.

00	No detection
01	Detects rising edge
10	Detects falling edge
11	Detects both edges

Bits 3–2 **CAP3EDGE**. Edge detection control for Capture Unit 3.

00	No detection
01	Detects rising edge
10	Detects falling edge
11	Detects both edges

Bits 1–0 **Reserved**. Reads return zero; writes have no effect.

Capture Control Register B (CAPCONB) — Address 7520h

15	14-13	12	11	10	9	8
CAPRES	CAPQEPN	CAP6EN	Reserved	CAP6TSEL	CAP45TSEL	CAP6TOADC
RW –0	RW –0	RW –0	R–0	RW –0	RW –0	RW –0

7-6	5-4	3-2	1-0
CAP4EDGE	CAP5EDGE	CAP6EDGE	Reserved
RW –0	RW –0	RW –0	RW –0

Note: *R = read access, W = write access, -0 = value after reset.*

Bit 15 **CAPRES**. Capture reset. Always reads zero.
Note: This bit is not implemented as a register bit. Writing a 0 simply clears the capture registers.
0 Clears all registers of Capture Units and QEP circuit to 0
1 No action

Bits 14–13 **CAPQEPN**. Capture Units 4 and 5 and QEP circuit control.
00 Disables Capture Units 4 and 5 and QEP circuit. FIFO stacks retain their contents
01 Enables Capture Units 4 and 5, disable QEP circuit
10 Reserved
11 Enables QEP circuit. Disable Capture Units 4 and 5; bits 4–7 and 9 are ignored

Bit 12 **CAP6EN**. Capture Unit 6 control.
0 Disables Capture Unit 6; FIFO stack of Capture Unit 6 retains its contents
1 Enables Capture Unit6

Bit 11 **Reserved**. Reads return zero; writes have no effect.

Bit 10 **CAP6TSEL**. GP Timer selection for Capture Unit6.
0 Selects GP Timer 4
1 Selects GP Timer 3

Bit 9 **CAP45TSEL**. GP Timer selection for Capture Units 4 and 5.
0 Selects GP Timer 4
1 Selects GP Timer 3

Bit 8 **CAP6TOADC**. Capture Unit 6 event starts ADC.
0 No action
1 Starts ADC when the CAP6INT flag is set

Bits 7–6 CAP4EDGE. Edge detection control for Capture Unit 4.

00	No detection
01	Detects rising edge
10	Detects falling edge
11	Detects both edges

Bits 5–4 CAP5EDGE. Edge detection control for Capture Unit 5.

00	No detection
01	Detects rising edge
10	Detects falling edge
11	Detects both edges

Bits 3–2 CAP6EDGE. Edge detection control for Capture Unit 6.

00	No detection
01	Detects rising edge
10	Detects falling edge
11	Detects both edges

Bits 1–0 Reserved. Reads return zero; writes have no effect.

Capture Status Registers

The ability to write to the CAPFIFOx registers can be used as a programming advantage. For example, if a "01" is written to the CAPnFIFO bits by user code, the EV module is led to believe that there is already an entry in the FIFO. Subsequently, every time the FIFO gets a new value, a capture interrupt will be generated. If a write occurs to the CAPnFIFOA status bits at the same time as they are being updated by hardware (because of a capture event), the user written data takes precedence.

Capture FIFO Status Register A (CAPFIFOA) — Address 7422h

15 - 14	13 -1 2	11 - 10	9 - 8
R eserved	C AP 3 FIFO	C AP 2 FIFO	C AP 1 FIFO
R – 0	RW – 0	RW – 0	RW – 0
7 - 0			
R e s erved			
R– 0			

Note: *R = read access, W = write access, -0 = value after reset.*

Bits 15–14 Reserved. Reads return zero; writes have no effect.

Bits 13–12 CAP3FIFO. CAP3FIFO Status

1.	Empty
2.	Has one entry
10	Has two entries

11 Had two entries and captured another one; first entry has been lost

Bits 11–10 CAP2FIFO. CAP2FIFO Status

1. Empty
2. Has one entry
10 Has two entries
11 Had two entries and captured another one; first entry has been lost

Bits 9–8 CAP1FIFO. CAP1FIFO Status

00 Empty
01 Has one entry
10 Has two entries
11 Had two entries and captured another one; first entry has been lost

Bits 7–0 Reserved. Reads return zero; writes have no effect.

Capture FIFO Status Register B (CAPFIFOB) — Address 7522h

15 - 14	13 -1 2	11 - 10	9 - 8
R eserved	C AP 6 FIFO	C AP 5 FIFO	C AP 4 FIFO
R – 0	RW – 0	RW – 0	RW – 0
7 - 0			
R e s erved			
R– 0			

Note: *R = read access, W = write access, -0 = value after reset.*

Bits 15–14 Reserved. Reads return zero; writes have no effect.

Bits 13–12 CAP6FIFO. CAP6FIFO Status

1. Empty
2. Has one entry
10 Has two entries
11 Had two entries and captured another one; first entry has been lost

Bits 11–10 CAP5FIFO. CAP5FIFO Status

1. Empty
2. Has one entry
10 Has two entries
11 Had two entries and captured another one; first entry has been lost

Bits 9–8 CAP4FIFO. CAP4FIFO Status

00 Empty
01 Has one entry
10 Has two entries
11 Had two entries and captured another one; first entry has been lost

Bits 7–0 **Reserved**. Reads return zero; writes have no effect.

General Event Manager Information

Table 6.5 Event Manager A (EVA) Pins

Pin Name	Description
CAP1/QEP1	Capture Unit 1 input, QEP circuit input 1
CAP2/QEP2	Capture Unit 2 input, QEP circuit input 2
CAP3	Capture Unit 3 input
PWM1	Compare Unit 1 output 1
PWM2	Compare Unit 1 output 2
PWM3	Compare Unit 2 output 1
PWM4	Compare Unit 2 output 2
PWM5	Compare Unit 3 output 1
PWM6	Compare Unit 3 output 2
T1CMP/T1PWM	Timer 1 compare/PWM output
T2CMP/T2PWM	Timer 2 compare/PWM output
TCLKINA	External clock-in for timers in EVA (*when configured to operate from external clock*)
TDIRA	External timer direction input in EVA (*when timer is in directional up/down mode*)

Table 6.6 Event Manager B (EVB) Pins

Pin Name	Description
CAP4/QEP3	Capture Unit 4 input, QEP circuit input 3
CAP5/QEP4	Capture Unit 5 input, QEP circuit input 4
CAP6	Capture Unit 6 input
PWM7	Compare Unit 4 output 1
PWM8	Compare Unit 4 output 2
PWM9	Compare Unit 5 output 1
PWM10	Compare Unit 5 output 2
PWM11	Compare Unit 6 output 1
PWM12	Compare Unit 6 output 2
T3CMP/T3PWM	Timer 3 compare/PWM output
T4CMP/T4PWM	Timer 4 compare/PWM output
TCLKINB	External clock-in for timers in EVB(*when configured to operate from external clock*)
TDIRB	External timer direction input in EVB (*when timer is in directional up/down mode*)

NOTE: *Most of the EV pins are mapped with a second function. In order to use the EV, you must configure the appropriate pins to their EV function. For more information on how pin sharing works and how to configure pins refer to Chapter 4.*

Event Manager (EV) Register Addresses

Table 6.7 Addresses of EVA Timer Registers

Address	Register	Name	
7400h	GPTCONA	GP Timer control register A	
7401h	T1CNT	Timer 1 counter register	Timer 1
7402h	T1CMPR	Timer 1 compare register	
7403h	T1PR	Timer 1 period register	
7404h	T1CON	Timer 1 control register	
7405h	T2CNT	Timer 2 counter register	
7406h	T2CMPR	Timer 2 compare register	Timer 2
7407h	T2PR	Timer 2 period register	
7408h	T2CON	Timer 2 control register	

Table 6.8 Addresses of EVB Timer Registers

Address	Register	Name	
7500h	GPTCONB	GP Timer control register B	
7501h	T3CNT	Timer 3 counter register	
7502h	T3CMPR	Timer 3 compare register	Timer 3
7503h	T3PR	Timer 3 period register	
7504h	T3CON	Timer 3 control register	
7505h	T4CNT	Timer 4 counter register	
7506h	T4CMPR	Timer 4 compare register	Timer 4
7507h	T4PR	Timer 4 period register	
7508h	T4CON	Timer 4 control register	

Table 6.9 Addresses of EVA Compare Control Registers

Address	Register	Name
7411h	COMCONA	Compare control register
7413h	ACTRA	Compare action control register
7415h	DBTCONA	Dead-band timer control register
7417h	CMPR1	Compare register 1
7418h	CMPR2	Compare register 2
7419h	CMPR3	Compare register 3

Table 6.10 Addresses of EVB Compare Control Registers

Address	Register	Name
7511h	COMCONB	Compare control register
7513h	ACTRB	Compare action control register
7515h	DBTCONB	Dead-band timer control register
7517h	CMPR4	Compare register 4
7518h	CMPR5	Compare register 5
7519h	CMPR6	Compare register 6

Table 6.11 Addresses of EVA Capture Registers

Address	Register	Name
7420h	CAPCONA	Capture control register
7422h	CAPFIFOA	Capture FIFO status register
7423h	CAP1FIFO	Two-level-deep capture FIFO stack 1
7424h	CAP2FIFO	Two-level-deep capture FIFO stack 2
7425h	CAP3FIFO	Two-level-deep capture FIFO stack 3
7427h	CAP1FBOT	Bottom registers of FIFO stacks; allows most recent CAPTURE value to be read
7428h	CAP2FBOT	
7429h	CAP3FBOT	

Table 6.12 Addresses of EVB Capture Registers

Address	Register	Name
7520h	CAPCONB	Capture control register
7522h	CAPFIFOB	Capture FIFO status register
7523h	CAP4FIFO	Two-level-deep capture FIFO stack 4
7524h	CAP5FIFO	Two-level-deep capture FIFO stack 5
7525h	CAP6FIFO	Two-level-deep capture FIFO stack 6
7527h	CAP4FBOT	Bottom registers of FIFO stacks, allows most recent CAPTURE value to be read
7528h	CAP5FBOT	
7529h	CAP6FBOT	

Table 6.13 Addresses of EVA Interrupt Registers

Address	Register	Name
742Ch	EVAIMRA	Interrupt mask register A
742Dh	EVAIMRB	Interrupt mask register B
742Eh	EVAIMRC	Interrupt mask register C
742Fh	EVAIFRA	Interrupt flag register A
7430h	EVAIFRB	Interrupt flag register B
7431h	EVAIFRC	Interrupt flag register C

Table 6.14 Addresses of EVB Interrupt Registers

Address	Register	Name
752Ch	EVBIMRA	Interrupt mask register A
752Dh	EVBIMRB	Interrupt mask register B
752Eh	EVBIMRC	Interrupt mask register C
752Fh	EVBIFRA	Interrupt flag register A
7530h	EVBIFRB	Interrupt flag register B
7531h	EVBIFRC	Interrupt flag register C

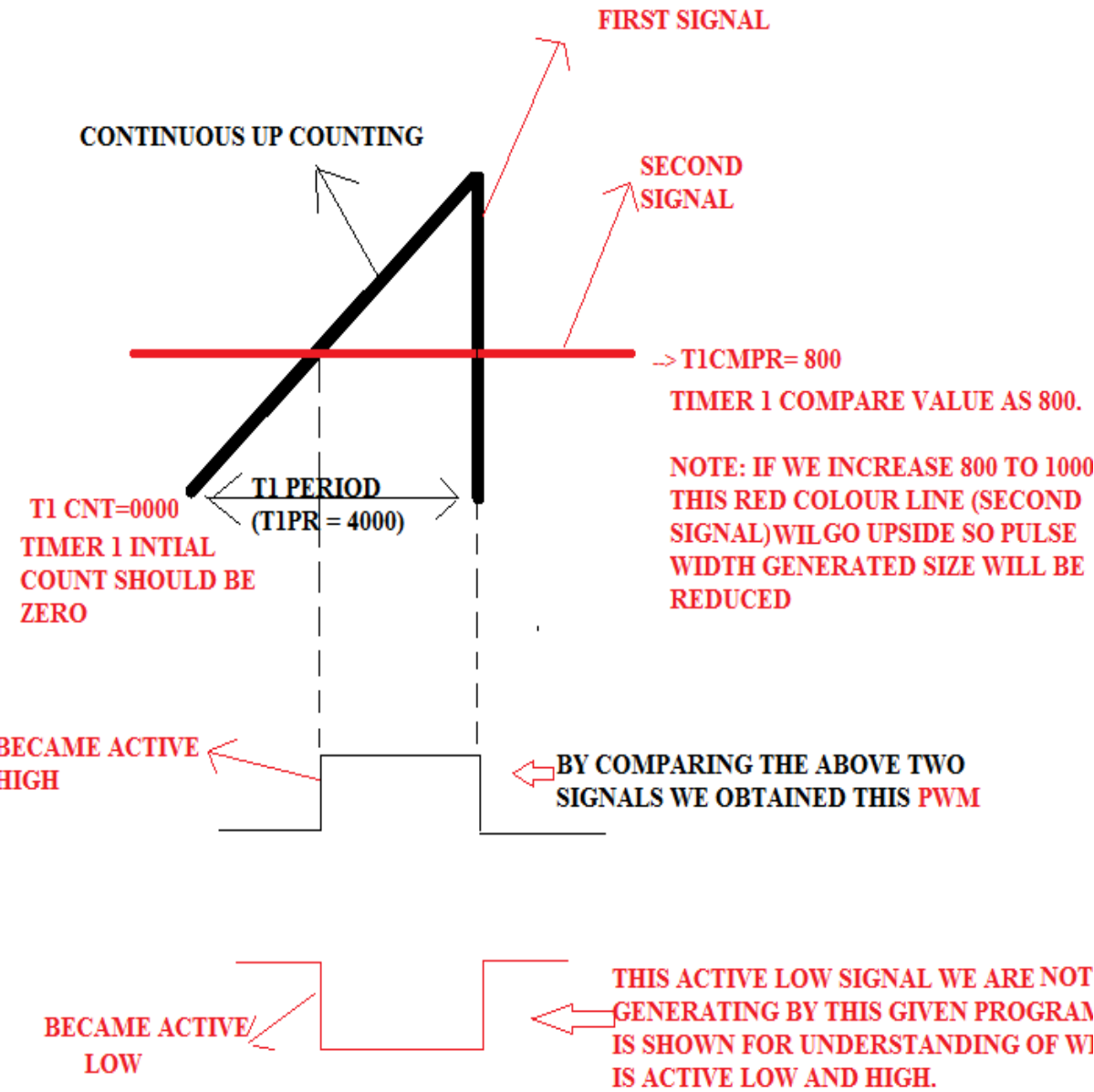
7. Exercise: PWM Signal Generation

As discussed in the previous sections, there are two ways to generate a PWM signal on the LF2407: through the GP Timer compare operation, or the Compare Units. This exercise will allow you to use your knowledge of the LF2407 DSP to write code that will generate PWM signals on both the GP Timer and Compare Unit outputs.

Procedure:

1. Write a program that outputs a fixed duty cycle "PWM" on a GP Timer 2 compare pin. Create the program so that the period of the PWM signal is 1 kHz and the duty cycle (on time/period) is fixed at 75%. The information on the GP Timer compare operation in the previous section will be very useful in writing this code.
2. View the output (1 kHz fixed duty cycle signal) on the T1PWM/T1CMP/IOPB4 pin. The Spectrum Digital LF2407 EVM schematic will be helpful in determining the location of this pin connection on the EVM.
3. If available, connect this fixed duty cycle signal to a dc voltage converter and use it to control the speed of a dc motor by varying the duty cycle of the waveform.
4. Modify the above program to now create a sinusoidally modulated PWM signal on the GP Timer Compare pin. To do this, a sinusoidal look-up

EXAMPLE TO GENERATE PWM



PROGRAM FOR GENERATION OF SINGLE PULSE PWM USING TIMER 1

```
LDP #0E1
SPLK #1000H,MCRA
LDP # 0E8
SPLK #6042H,GPTCONA
SPLK #0000H,T1CNT
SPLK #800H,T1CMPR
SPLK #4000H,T1PR
SPLK #9042H,T1CON
H:B:H
```

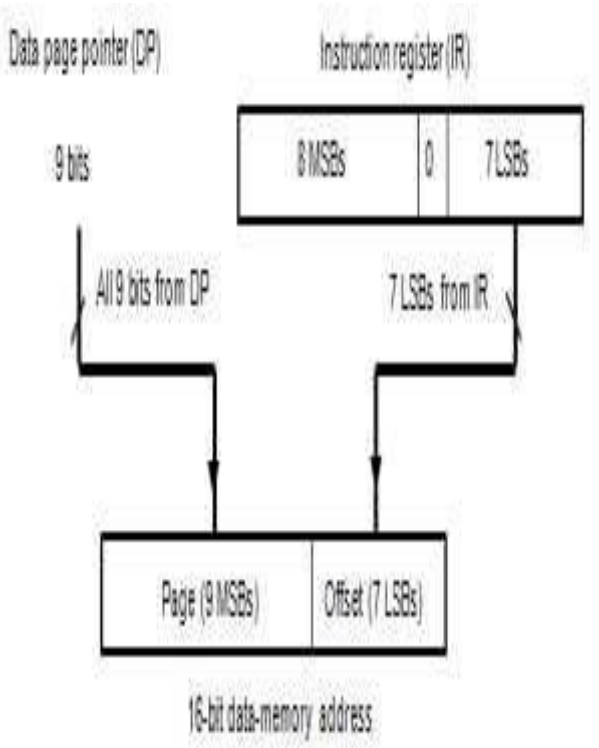
EXPLANATION OF FIRST LINE

LDP #0E1 –LOAD DATA PAGE WITH 0E1 VALUE.

THIS IS DIRECT ADDRESSING

0,1110,0001 →0E1 (DATA PAGE IS LOADED WITH THIS 9 BITS)

DP Value	Offset	Data Memory
0000 0000 0 ⋮ 0000 0000 0	000 0000 ⋮ 111 1111	Page 0: 0000h-007Fh
0000 0000 1 ⋮ 0000 0000 1	000 0000 ⋮ 111 1111	Page 1: 0080h-00FFh
0000 0001 0 ⋮ 0000 0001 0	000 0000 ⋮ 111 1111	Page 2: 0100h-017Fh
⋮	⋮	⋮
1111 1111 1 ⋮ 1111 1111 1	000 0000 ⋮ 111 1111	Page 511: FF80h-FFFFh



EXPLANATION OF SECOND LINE

- SPLK #1000H,MCRA → store long constant 1000 in MUX CONTROL REGISTER A.

- VALUE IS 1000 BUT WE DON'T KNOW THE ADDRESS OF MCRA REGISTER TO LOAD 1000 ON MCRA. SO TO FRAME THE ADDRESS OF MCRA USE THE PREVIOUS SLIDE.
- 9+7 =16 BIT ADDRESS
- IN PREVIOUS SLIDE 16 BIT ADDRESS IS SPLITED AS 9 BITS(0E1→0,1110,0001) FROM DATA PAGE POINTER.AND REMAINING 7 FROM INSTRUCTION REGISTER 7 BIT LSB.
- AFTER READING THE SECOND LINE OF THE PROGRAM, THE INSTRUCTION REGISTER LAST 7 LSB BITS AUTOMATICALLY LOADED BY THE MACHINE WITH 0010000.
- THEREFORE JOIN THIS ABOVE 9 BITS AND 7 BITS
- 0111000010010000
- SPLIT THE ABOVE NUMBER WITH 4 NUMBERS EQUALLY

0111,0000,1001,0000→7090 →REFER NEXT SLIDE FOR MCR A REGISTER ADDRESS. I HOPE U UNDERSTOOD THE FIRST LINE OF THE PROGRAM.

LDP,#0E1

USEFUL REGISTERS WITH ADDRESS FOR PULSE GENERATION PROGRAM

Data Memory Address	Register Name	Description
7090h	MCRA	I/O MUX Control Register A
7092h	MCRB	I/O MUX Control Register B
7094h	MCRC	I/O MUX Control Register C
7098h	PADATDIR	I/O Port A Data and Direction Register
709Ah	PBDATDIR	I/O Port B Data and Direction Register
709Ch	PCDATDIR	I/O Port C Data and Direction Register
709Eh	PDDATDIR	I/O Port D Data and Direction Register
7095h	PEDATDIR	I/O Port E Data and Direction Register
7096h	PFDATDIR	I/O Port F Data and Direction Register

Address	Register	Name	
7400h	GPTCONA	GP Timer control register A	
7401h	T1CNT	Timer 1 counter register	Timer 1
7402h	T1CMPR	Timer 1 compare register	
7403h	T1PR	Timer 1 period register	
7404h	T1CON	Timer 1 control register	
7405h	T2CNT	Timer 2 counter register	
7406h	T2CMPR	Timer 2 compare register	Timer 2
7407h	T2PR	Timer 2 period register	
7408h	T2CON	Timer 2 control register	

Address	Register	Name	
7500h	GPTCONB	GP Timer control register B	
7501h	T3CNT	Timer 3 counter register	
7502h	T3CMPR	Timer 3 compare register	Timer 3
7503h	T3PR	Timer 3 period register	
7504h	T3CON	Timer 3 control register	
7505h	T4CNT	Timer 4 counter register	
7506h	T4CMPR	Timer 4 compare register	Timer 4
7507h	T4PR	Timer 4 period register	
7508h	T4CON	Timer 4 control register	

Address	Register	Name
7411h	COMCONA	Compare control register
7413h	ACTRA	Compare action control register
7415h	DBTCONA	Dead-band timer control register
7417h	CMPR1	Compare register 1
7418h	CMPR2	Compare register 2
7419h	CMPR3	Compare register 3

USEFUL REGISTERS WITH ADDRESS FOR PULSE GENERATION PROGRAM

Address	Register	Name
7511h	COMCONB	Compare control register
7513h	ACTRB	Compare action control register
7515h	DBTCONB	Dead-band timer control register
7517h	CMPR4	Compare register 4
7518h	CMPR5	Compare register 5
7519h	CMPR6	Compare register 6

Address	Register	Name
7420h	CAPCONA	Capture control register
7422h	CAPFIFOA	Capture FIFO status register
7423h	CAP1FIFO	Two-level-deep capture FIFO stack 1
7424h	CAP2FIFO	Two-level-deep capture FIFO stack 2
7425h	CAP3FIFO	Two-level-deep capture FIFO stack 3
7427h	CAP1FBOT	Bottom registers of FIFO stacks; allows most recent CAPTURE value to be read
7428h	CAP2FBOT	
7429h	CAP3FBOT	

Address	Register	Name
7520h	CAPCONB	Capture control register
7522h	CAPFIFOB	Capture FIFO status register
7523h	CAP4FIFO	Two-level-deep capture FIFO stack 4
7524h	CAP5FIFO	Two-level-deep capture FIFO stack 5
7525h	CAP6FIFO	Two-level-deep capture FIFO stack 6
7527h	CAP4FBOT	Bottom registers of FIFO stacks, allows most recent CAPTURE value to be read
7528h	CAP5FBOT	

AFTER FRAMING THE 16 BIT ADDRESS 7090 OF MCRA REGISTER

- THE VALUE 1000 IS LOADED IN THE MCRA REGISTER.
- WE WILL SEE WHAT HAPPENS IF WE LOAD 1000 IN MCRA REGISTER.

I/O MUX Control Register A (MCRA) Configuration

15	14	13	12	11	10	9	8
MCRA.15	MCRA.14	MCRA.13	MCRA.12	MCRA.11	MCRA.10	MCRA.9	MCRA.8
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0
7	6	5	4	3	2	1	0
MCRA.7	MCRA.6	MCRA.5	MCRA.4	MCRA.3	MCRA.2	MCRA.1	MCRA.0
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

Bit #	Name.bit #	Pin Function Selected (MCA.n = 1) (Primary)	Pin Function Selected (MCA.n = 0) (Secondary)
0	MCRA.0	SCITXD	IOPA0
1	MCRA.1	SCIRXD	IOPA1
2	MCRA.2	XINT1	IOPA2
3	MCRA.3	CAP1/QEP1	IOPA3
4	MCRA.4	CAP2/QEP2	IOPA4
5	MCRA.5	CAP3	IOPA5
6	MCRA.6	PWM1	IOPA6
7	MCRA.7	PWM2	IOPA7

Bit #	Name.bit #	Pin Function Selected (MCA.n = 1) (Primary)	Pin Function Selected (MCA.n = 0) (Secondary)
8	MCRA.8	PWM3	IOPB0
9	MCRA.9	PWM4	IOPB1
10	MCRA.10	PWM5	IOPB2
11	MCRA.11	PWM6	IOPB3
12	MCRA.12	T1PWM/T1CMP	IOPB4
13	MCRA.13	T2PWM/T2CMP	IOPB5
14	MCRA.14	TDIRA	IOPB6
15	MCRA.15	TCLKINA	IOPB7

AFTER LOADING 1000 IN MCRA REGISTER

I/O MUX Control Register A (MCRA) Configuration

0	0	0	1	0	0	0	0
15	14	13	12	11	10	9	8
MCRA.15	MCRA.14	MCRA.13	MCRA.12	MCRA.11	MCRA.10	MCRA.9	MCRA.8
MCRA.7	MCRA.6	MCRA.5	MCRA.4	MCRA.3	MCRA.2	MCRA.1	MCRA.0
0	0	0	0	0	0	0	0

LOADING 1000 IN MCRA

Bit #	Name.bit#	Pin Function Selected (MCA.n = 1) (Primary)	Pin Function Selected (MCA.n = 0) (Secondary)
0	MCRA.0	SCITXD	IOPA0
1	MCRA.1	SCIRXD	IOPA1
2	MCRA.2	XINT1	IOPA2
3	MCRA.3	CAP1/QEP1	IOPA3
4	MCRA.4	CAP2/QEP2	IOPA4
5	MCRA.5	CAP3	IOPA5
6	MCRA.6	PWM1	IOPA6
7	MCRA.7	PWM2	IOPA7

Bit #	Name.bit#	Pin Function Selected (MCA.n = 1) (Primary)	Pin Function Selected (MCA.n = 0) (Secondary)
8	MCRA.8	PWM3	IOPB0
9	MCRA.9	PWM4	IOPB1
10	MCRA.10	PWM5	IOPB2
11	MCRA.11	PWM6	IOPB3
12	MCRA.12	T1PWM/T1CMP	IOPB4
13	MCRA.13	T2PWM/T2CMP	IOPB5
14	MCRA.14	TDIRA	IOPB6
15	MCRA.15	TCLKINA	IOPB7

- FROM THE PREVIOUS SLIDE WHAT WE UNDERSTOOD IS THAT EXCEPT BIT 12 REMAINING BITS ARE SET AS “0”.

THAT MEANS IAM GOING TO USE **TIMER 1 FOR PWM** GENEARTION AS WELL AS **COMPARISION.**

REMAINING AND ALL “0” MEANS **SECONDARY FUCTION** OF THE PIN IS SELECTED SO THOSE PINS ARE GOING WORK AS **GENERAL I/O PORT PINS.**

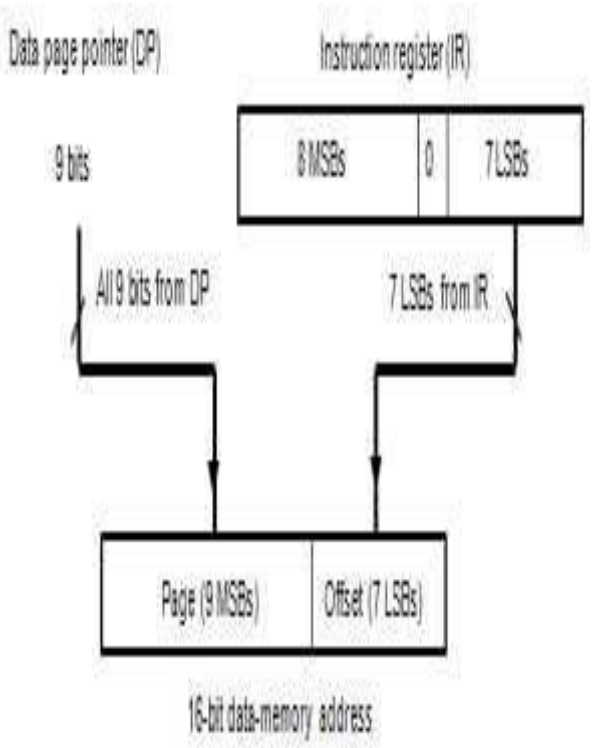
EXPLANATION OF THIRD LINE

LDP #0E8 –LOAD DATA PAGE WITH 0E8 VALUE.

THIS IS DIRECT ADDRESSING

0,1110,1000 →0E8 (DATA PAGE IS LOADED WITH THIS 9 BITS)

DP Value	Offset	Data Memory
0000 0000 0 ⋮ 0000 0000 0	000 0000 ⋮ 111 1111	Page 0: 0000h-007Fh
0000 0000 1 ⋮ 0000 0000 1	000 0000 ⋮ 111 1111	Page 1: 0080h-00FFh
0000 0001 0 ⋮ 0000 0001 0	000 0000 ⋮ 111 1111	Page 2: 0100h-017Fh
⋮	⋮	⋮
1111 1111 1 ⋮ 1111 1111 1	000 0000 ⋮ 111 1111	Page 511: FF80h-FFFFh



EXPLANATION OF FOURTH LINE

- SPLK #6042H,GPTCONA → store long constant 6042 in GENERAL PURPOSE TIMER CONTROL REGISTER A.

- VALUE IS 6042 BUT WE DON'T KNOW THE ADDRESS OF GPTCONA REGISTER TO LOAD 6042 ON GPTCONA. SO TO FRAME THE ADDRESS OF GPTCONA USE THE PREVIOUS SLIDE.
- 9+7 =16 BIT ADDRESS
- IN PREVIOUS SLIDE 16 BIT ADDRESS IS SPLITED AS 9 BITS(0E8→0,1110,1000) FROM DATA PAGE POINTER.AND REMAINING 7 FROM INSTRUCTION REGISTER 7 BIT LSB.
- AFTER READING THEFOURTH LINE OF THE PROGRAM, THE INSTRUCTION REGISTER LAST 7 LSB BITS AUTOMATICALLY LOADED BY THE MACHINE WITH 0000000.
- THEREFORE JOIN THIS ABOVE 9 BITS AND 7 BITS
- 0111010000000000
- SPLIT THE ABOVE NUMBER WITH 4 NUMBERS EQUALLY 0111,0100,0000,0000

→7400 →REFER **USEFUL REGISTERS WITH ADDRESS FOR PULSE GENERATION PROGRAM SLIDE**. I HOPE U UNDERSTOOD THE THIRD LINE OF THE PROGRAM. LDP,#0E8

AFTER FRAMING THE 16 BIT ADDRESS 7400 OF GPTCONA REGISTER

- THE VALUE 6042 IS LOADED IN THE GPTCONA REGISTER.
- WE WILL SEE WHAT HAPPENS IF WE LOAD 6042 IN GPTCONA REGISTER.

GP Timer Control Register A (GPTCONA) Bit Descriptions — Address 7400h

15	14	13	12-11	10-9	8-7
0	1	1	0 0	0 0	0 0
Reserved	T2STAT	T1STAT	Reserved	T2TOADC	T1TOADC

6	5-4	3-2	1-0
TCOMPOE	Reserved	T2PIN	T1PIN
1	0 0	0 0	1 0

LOADING 6042 IN GPTCON A REGISTER

Bit 15 **Reserved.** Reads return zero; writes have no effect.

Bit 14 **T2STAT.** GP Timer 2 Status. Read only.
- 0 Counting downward
- 1 Counting upward

Bit 13 **T1STAT.** GP Timer 1 Status. Read only.
- 0 Counting downward
- 1 Counting upward

Bits 12–11 Reserved. Reads return zero; writes have no effect.

Bits 10–9 T2TOADC. Start ADC with timer 2 event.
- 00 No event starts ADC
- 01 Setting of underflow interrupt flag starts ADC
- 10 Setting of period interrupt flag starts ADC
- 11 Setting of compare interrupt flag starts ADC

Bits 8–7 T1TOADC. Start ADC with timer 1 event.
- 00 No event starts ADC
- 01 Setting of underflow interrupt flag starts ADC
- 10 Setting of period interrupt flag starts ADC
- 11 Setting of compare interrupt flag starts ADC

Bit 6 **TCOMPOE.** Compare output enable. If PDPINTx is active this bit is set to zero.
- 0 Disable all GP Timer compare outputs (all compare outputs are put in the high-impedance state)
- 1 Enable all GP Timer compare outputs

Bits 5–4 Reserved. Reads return zero; writes have no effect.

Bits 3–2 T2PIN. Polarity of GP Timer 2 compare output.
- 00 Forced low
- 01 Active low

AFTER LOADING 6042 IN GPTCON A

- WE KNOW THAT THIS IS TO GENERATE SINGLE PWM USING TIMER 1 SO **BIT 13** IS "1"
- AND IF IN CASE TIMER 1 FAILURE I MAY USE TIMER 2 SO **BIT 14** ALSO MADE AS "1".MOREOVER THAT "**1" MEANS UPCOUNTING SELCTION.**
- **WHY UPCOUNTING KINDLY CHECK THE "EXAMPLE TO GENERATE PWM SLIDE".**
- I DON'T REQUIRED ADC FOR PWM GENERATION SO **BIT 7 TO 10** IS "0".
- **BIT 6 IMPORTANT TO MAKE AS "1"**. BECAUSE PWM IS GENERATED BY COMAPRING TWO SIGNALS (IN THIS PROGRAM COMPARED AT 800). **KINDLY CHECK THE "EXAMPLE TO GENERATE PWM SLIDE".**
- I AM USING TIMER 1 AFTER COMPARISON I NEED PWM WITH ACTIVE HIGH **SO BIT 1 AND 0** MADE AS **"1" "0". CHECK THE "EXAMPLE TO GENERATE PWM SLIDE".**
- I HOPE U UNDERSTOOD THE FOURTH LINE OF THE PROGRAM. SPLK #6042H,GPTCONA

LINE 5,6,7 EXPLANATION

SPLK #0000H,T1CNT
SPLK #800H,T1CMPR
SPLK #4000H,T1PR

- **FIRST SIGNAL STARTS AT 0000 THAT'S WHY 0000H,T1CNT TIMER 1 COUNT REGISTER IS LOADED WITH VALUE 0000.**

- **SECOND SIGNAL AND FIRCT SIGNAL ARE COMPARED AT VALUE 800 . THAT'S WHY 800H,T1CMPR**
  TIMER 1 COMPARE REGISTER IS LOADED WITH VALUE 0000.

- **PERIOD (OR) WIDTH OF THE FIRST SIGNAL IS DECIDED BY THE VALUE LOADED IN THE TIMER 1 PERIOD REGISTER. CHECK THE "EXAMPLE TO GENERATE PWM SLIDE".**

- **ADDRESS FOR VARIOUS REGISTERS GIVEN BELOW.**

- **TRY BY YOURSELF TO GET THESE 16 BIT ADDRESS BY TAKING LDP,#0E8.**

- **T1CNT→7401**
- **T1CMPR →7402**
- **T1PR →7403**

LINE 8,9 EXPLANATION

SPLK #9042H,T1CON

H:B:H

- BIT 15,14 → "1""0" →PWM GENERATION OPERATION WILL NOT BE STOPPED WHILE SOMEOTHER PROGRAM INTERRUPTION HAPPENS.
- BIT 12,11→ "0""1"→ UPCOUNTING REQUIRED TO GENERATE PWM SIGNAL.
- BIT 10 TO 8 → DECIDE THE VARIOUS CLOCK FREQUENCY(DECIDES THE OPERATING SPEED OF TIMER 1)
- BIT 7 → USING TIMER 1 OWN BIT TO START TIMER.
- **FOR EXAMPLE:** IF WE SET BIT 7 AS "1" AND IF WE USED TIMER 2 FOR PWM GENERATION. THEN **TIMER 2** CAN BE START BY USING **TIMER 1 BIT .**
- **NOTE: TIMER 2 CANNOT START TIMER 1.**
- **BIT 6 → FOR TIMER ENABLE/DISABLE.**
- **BIT 5,4 →TIMER 1 GOING TO USE INTERNAL CLOCK FREQUENCY SO "0""0".**
- **BIT 3,2→ "0""0". COUNTER REGISTER AS TO RELOAD TO 0000 (ONCE IT REACHES 4000 – T1PR VALUE AS PER THIS EXAMPLE).TO GENERATE CONTINUOUS PWM SIGNAL.**
- **BIT 1→ "1" TIMER1 COMPARE UNIT HAS TO BE ENABLED FOR PWM GENERATION.**
- **BIT 0 → SIMILAR TO BIT 7.**

H:B:H → BRANCH TO H , WHERE IS H , ACTUALLY IT IS STARTING OF THIS LINE .SO IT JUMP WITHIN THIS LINE ITSELF. IT IS LIKE HALT THE PROGRAM.