COPYRIGHT IS NOT RESERVED BY AUTHORS. AUTHORS ARE NOT RESPONSIBLE FOR ANY LEGAL ISSUES ARISING OUT OF ANY COPYRIGHT DEMANDS AND/OR REPRINT ISSUES CONTAINED IN THIS MATERIALS. THIS IS NOT MEANT FOR ANY COMMERCIAL PURPOSE & ONLY MEANT FOR PERSONAL USE OF STUDENTS FOLLOWING SYLLABUS PRINTED NEXT PAGE. READERS ARE REQUESTED TOSEND ANY TYPING ERRORS CONTAINED, HEREIN.

Digital Signal Processing (3-1-0)

Module-I (10 Hours)

Discrete time signals and systems, The Convolution Sum and its properties, Difference Equation,
Implementation of DT System, Correlation, LTI systems as Frequency-Selective Filters, Inverse Systems and
Deconvolution.

Module-II (10 Hours)

Analysis of LTI system in z-Domain, One-sided z-Transform, The DFT as a linear transformation, Circular
Convolution, Circular Correlation, Linear Filtering Methods Based on the DFT, The Discrete Cosine
Transform(Brief Idea only).

Module-III (10 Hours)

Fast Fourier Transform Algorithms: Radix –2 FFT algorithm – Decimation – in Time (DIT) and Decimation
– in Frequency (DIF) algorithm, Applications of FFT Algorithms, The Chirp-z Transform Algorithm.

Module-IV (10 Hours)

Structures for FIR and IIR Systems - Direct and Cascaded form, Design of Digital Filters: Causality and its
Implications, Design of Linear Phase FIR filters using different windows, Design of IIR Filters - Impulse
Invariance Method and Bilinear transformation method.

Text Books:

1. Digital Signal Processing – Principles, Algorithms and Applications - J.G.Proakis and D.G.Manolakis,
4th Edition, PHI Learning Pvt. Ltd. (Selected Portions from Chapters 1, 2, 3, 4, 5, 7, 8, 9, 10, 11,
13, 14)
2. Digital Signal Processing - S.Salivahanan, A.Vallavaraj, C. Gnanapriya, 2nd Edition The McGraw-Hill
Companies. (Selected Portions from Chapters 1, 2, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14)

Reference Books:

1. Introduction to Digital Signal Processing –J.R.Johnson, PHI learning Pvt. Ltd.
2. Discrete Time Signal Processing- A.V. Oppenheim and Schafer, PHI Learning Pvt.
3. Digital Signal Processing: A computer based Approach - Sanjit K. Mitra, The McGraw-Hill Companies.

MODULE-1

Introduction

Signal:

A *signal* is defined as any physical quantity th a t varies with time, space, or any other in dependent variable or variables. Mathematically, we describe a signal as a function of one or mo re independent variables. For example, the functions

$$s(t)= 5t$$

describe a signal, one that varies linearly with the in d e p e n d e n t variable *t* (time).

$$s(x, y) = 3x + 2xy + 10y^2$$

This function describes a signal of two in dependent variables *x* and y that could represent the two spatial coordinates in a p lane.

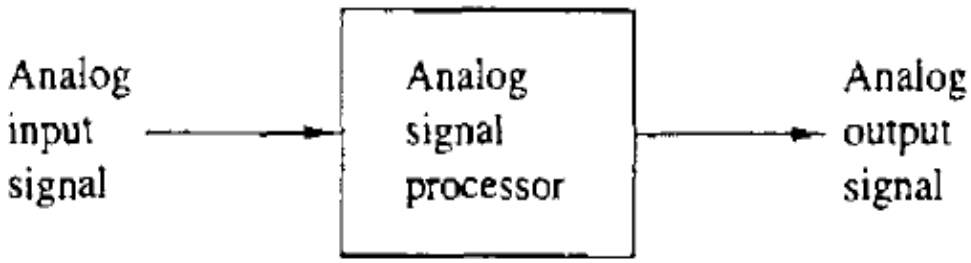
System:

A *system* may also be defined as a physical device th a t performs an operatioon a signal. For ex ample, a filter used to reduce the noise and interference corrupting desired in formation bearing signal is called a system .

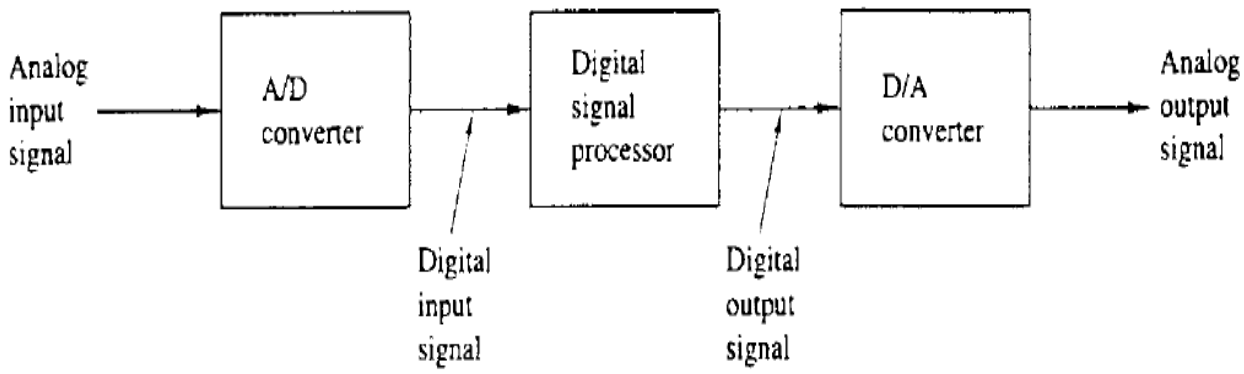
signal processing:

W h en we pass a signal thrugh a system , as in filtering, we say that we have processed the signal. In this case the processing of the signal involves filtering the noise and interference from the desired signal. If the operation on the signal is n o n linear, the system is said to be non linear, and so forth . Such operations are usually referred to as *signal rocessing.*

Analog signal processing:



Digital signal processing:



Advantages of Digital over Analog Signal Processing :

1- a digital programmable system allow s flexibility in re configuring the digital signal processing operations simply by changing the program .

2- a digital system provides much better control of accuracy .

3- Digital signals are easily stored on magnetic media (tape or disk) without deterioration or loss of signal fidelity beyond that introduced in the A /D conversion.

4- digital implementation of the signal processing system is cheaper than analog signal processing.

Limitations:

One practical limitation is the speed of operation of A /D converters and digital signal processors. We shall see that signals having extremely wide band widths require fast-sampling -rate A /D converters and fast digital signal processors. Hence there are analog signals with large bandwidths for which a digital processing approach is beyond the state of the art of digital hardware.

Discrete time signals and systems

CLASSIFICATION OF SIGNALS : There are 3 types of signals

Continuous-time signals: *Continuous-time signals* or *analog signals* are defined for every value of time.

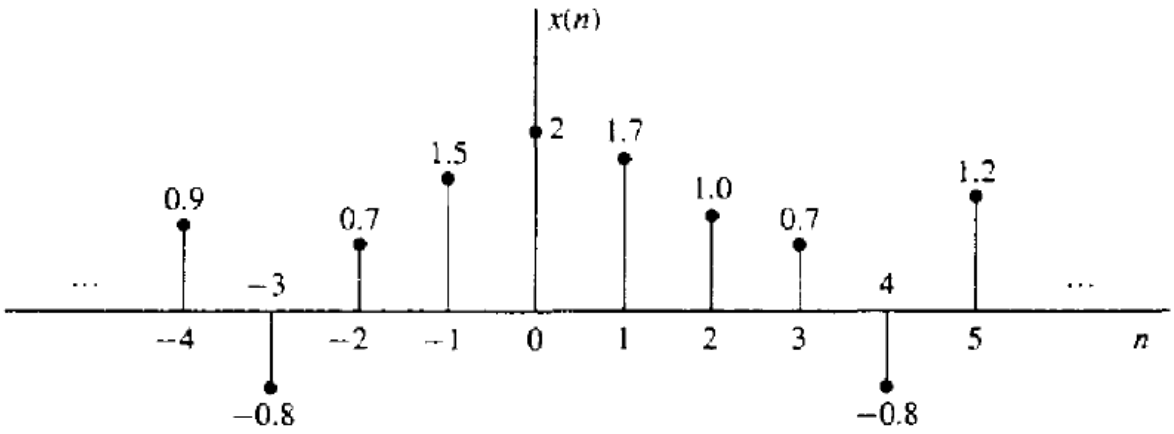
Discrete-time signals : Discrete-time signals are defined only at certain specific values of time.

Digital Signals: digital signal is defined as a function of an integer independent variable and its values are taken from a finite set of possible values, which are represented by a string of 0's and l's .

DISCRETE-TIME SIGNALS : A disc rete-time signal *x{n)* is a function of an in dependent variable that is an integer. discrete-time signal is *n o t defined* at instants between two successive samples. Simply, the signal *x (n)* is n o t defined for n o n integer values o f *n*. So x(n) was obtained from sampling an analog signal x a(t), then .i(n) = x a(nT) , where T is the sampling period (i.e., the time between successive samples).

Representation of discrete-time signal :

A discrete-time signal can be represented in various way**.** But all can be represented graphically.



Graphical representation of a discrete-time signal.

Besides the graphical representation of a discrete-time signal or sequence as illustrated in above Fig. there are some alternative representations that are often more convenient to use. These are:

1. **Functional representation** :

$$x(n) = \begin{cases} 1, & \text{for } n = 1, 3 \\ 4, & \text{for } n = 2 \\ 0, & \text{elsewhere} \end{cases}$$

2. **Tabular representation :**

n	...	−2	−1	0	1	2	3	4	5	...
$x(n)$	...	0	0	0	1	4	1	0	0	...

3.Sequence representation : An infinite-duration signal or sequence with the time origin *(n* = 0) indicated by the symbol ↑ is represented as

$$x(n) = \{\ldots 0, \underset{\uparrow}{0}, 1, 4, 1, 0, 0, \ldots\}$$

A finite-duration sequence can be represented as

$$x(n) = \{3, -1, \underset{\uparrow}{-2}, 5, 0, 4, -1\}$$

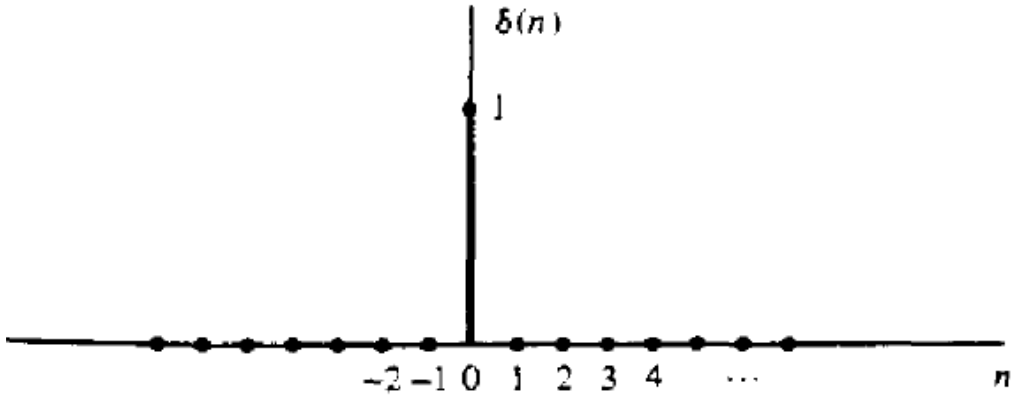
Some Elementary Discrete-Time Signals :

In discrete-time signals and systems there are a number of basic signals that appear often and play an important role. These signals are defined below .

1. Unit sample sequence/ unit impulse : It is denote d as δ(n) and is defined as

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

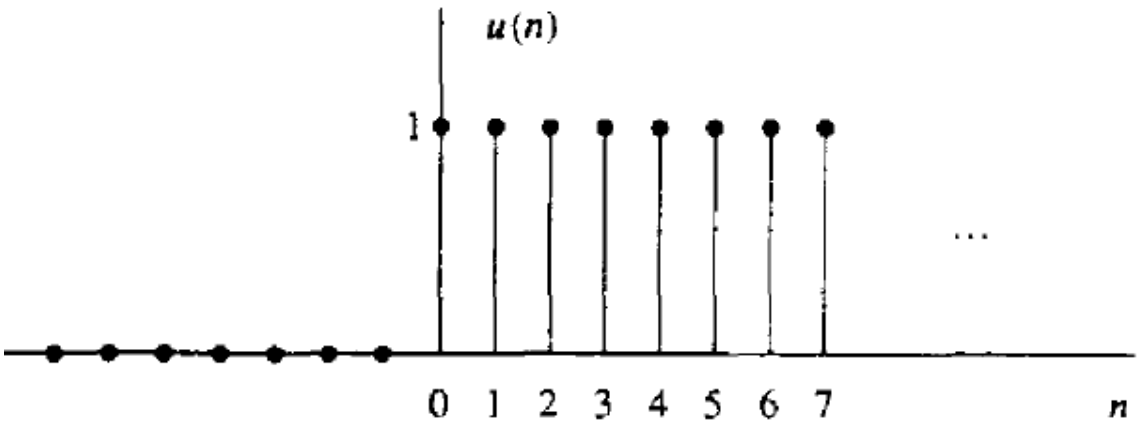
the unit impulse sequence is a signal that is zero every where, except at n =0 where its value is unity. The graphical representation of *δ(n)* is



2. **Unit step signal:** It is denoted as u(n) and is defined as

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

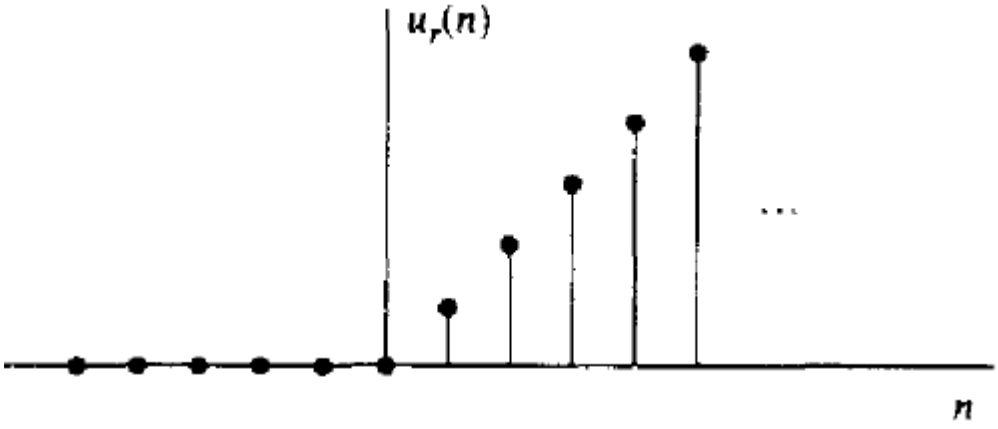
The graphical re presentation of *u(n)* is



3. **Unit ramp signal** *:* It is denoted as $u_r(n)$ and is defined as

$$u_r(n) \equiv \begin{cases} n, & \text{for } n \geq 0 \\ 0, & \text{for } n < 0 \end{cases}$$

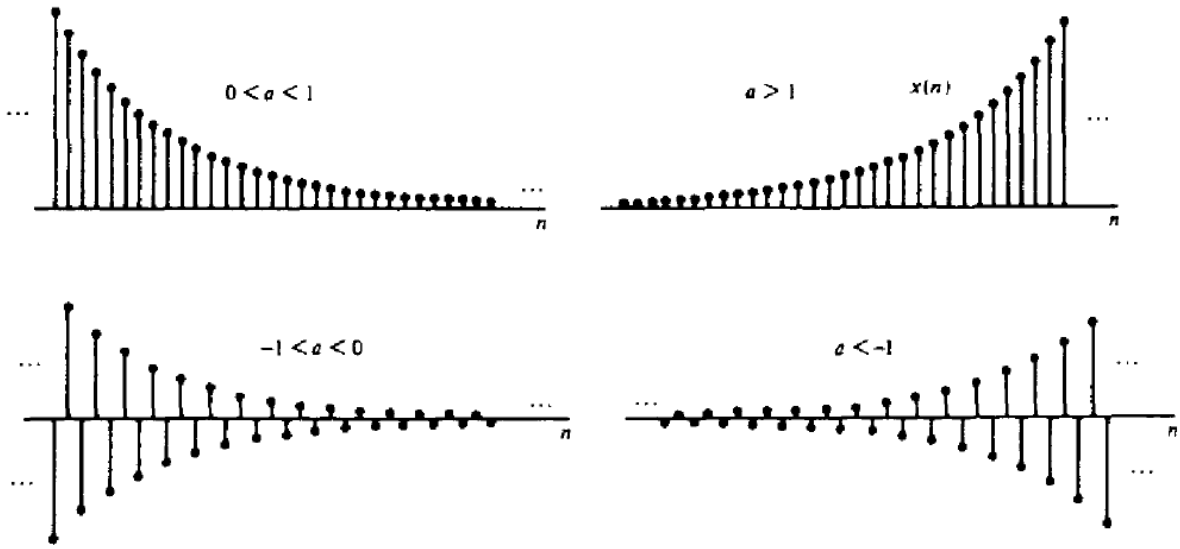
The graphical representation of $u_r(n)$ is



4-Exponential signal : It is a sequence of the form

$$x(n) = a^n \qquad \text{for all } n$$

If the parameter *a* is real, then x(n) is a real signal. illustratation of *x(n)* for various values of the parameter *a is*



When the parameter *a* is complex valued , it can be expressed as

$$a \equiv re^{j\theta}$$

where *r* and *Θ* are now the parameters. Hence we can express *x(n)* as

$$\begin{aligned} x(n) &= r^n e^{j\theta n} \\ &= r^n(\cos\theta n + j\sin\theta n) \end{aligned}$$

Classification of Discrete-Time Signals:

1-**Energy signals and power signals:** The energy E of a signal *x(n)* is defined as

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

If *E* is finite (i.e., 0 <*E* <∞), if *E* is finite , *P* = 0. then x(n) is called an *energy signal.*

Many signals that possess infinite energy, have a finite average power. The average power of a d iscrete-time signal *x(n)* is defined as

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

If we define the signal energy of *x(n)* over the finite interval —*N* < *n* < *N* as

$$E_N \equiv \sum_{n=-N}^{N} |x(n)|^2$$

the average power of the signal *x(n)* as

$$P \equiv \lim_{N\to\infty} \frac{1}{2N+1} E_N$$

if *E* is infinite and *P* is finite. the signal is called a *power signal.*

2-Periodic signals and aperiodic signals:

signal *x(n)* is periodic with period *N (N* > 0) if an d only if

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

the sinusoidal signal of the form

$$x(n) = A \sin 2\pi f_0 n$$

is periodic when f_0, is a rational number, that is, if f_0 can be expressed as

$$f_0 = \frac{k}{N}$$

where *k* and *N* are integers.

3-Symmetric (even) and antisymmetric (odd) signals :

A real valued signal *x (n*) is called symmetric (even) if

$$x(-n) = x(n)$$

O n the other hand , a signal *x(n)* is called antisymmetric (odd) if

$$x(-n) = -x(n)$$

We can illustrate that any arbitrary signal can be expressed as the sum of two signal components, one of which is even and the other odd. The even signal component is formed by adding x(n) to *x (—n)* and dividing by 2. that is.

$$x_e(n) = \tfrac{1}{2}[x(n) + x(-n)]$$

Similarly, w e form an odd signal component $x_0(n)$ according to the relation

$$x_o(n) = \frac{1}{2}[x(n) - x(-n)]$$

So we obtain x(n),that is,

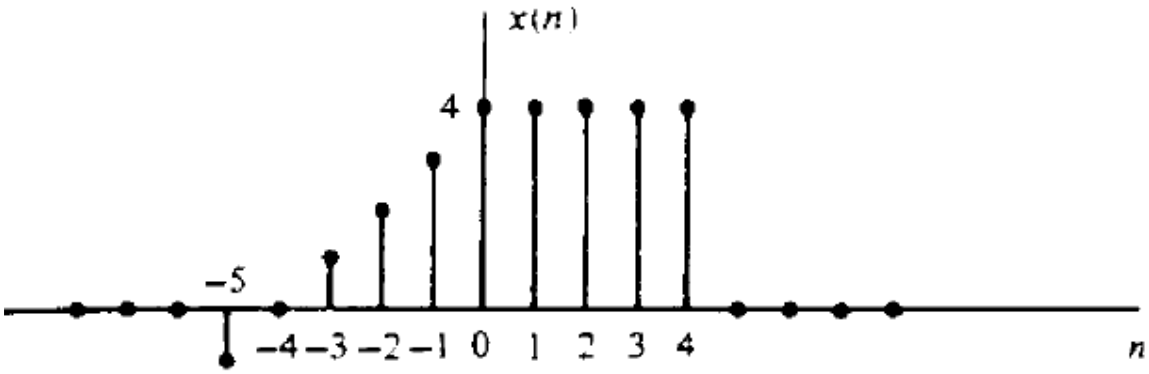
$$x(n) = x_e(n) + x_o(n)$$

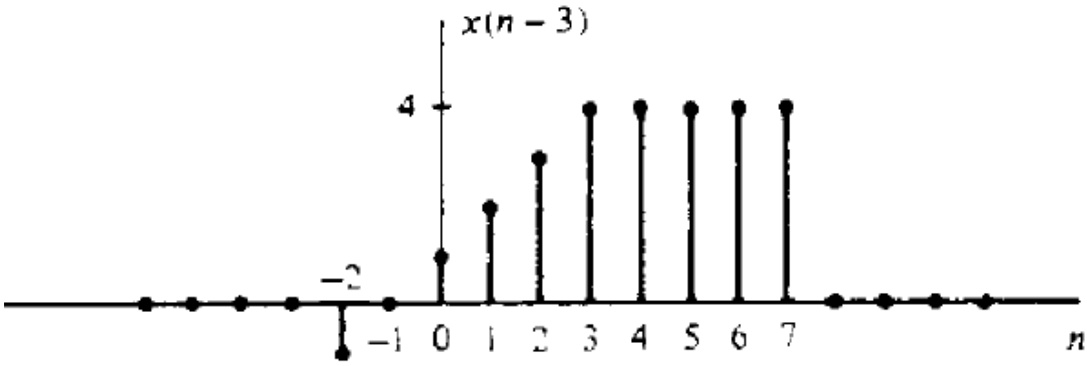
Simple Manipulations of Discrete-Time Signals :

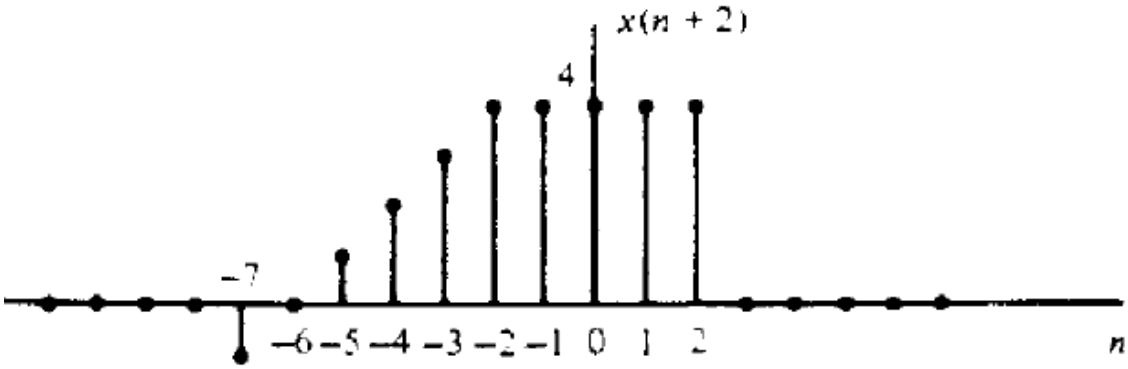
Time shifting :

A signal *x (n)* may be shifted in time by replacing the independent variable *n* by *n* — *k*, w here *k* is an integer. If k is a positive integer, the time shift results in a delay of the signal by *k* units o f time. If *k* is a negative integer, the time shift results in an advance of the signal by *k*\ units in time.

Ex- A signal *x(n)* is graphically illustrated in Fig. below. Show a graphical representation of the signals *x(n* — 3) and *x(n* + 2).





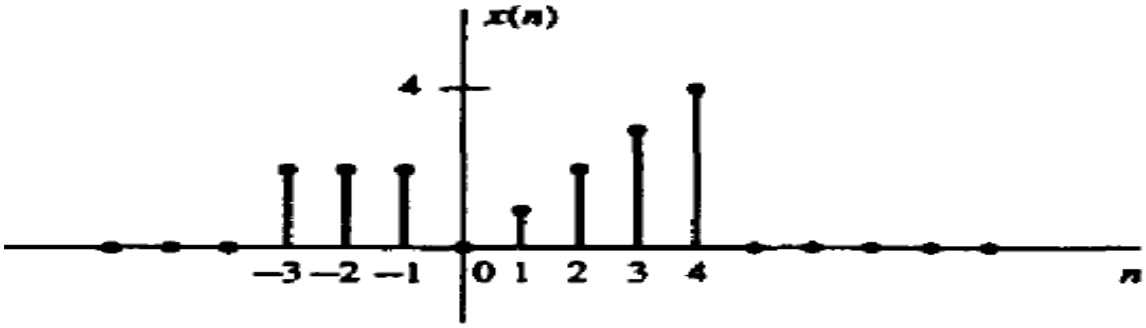


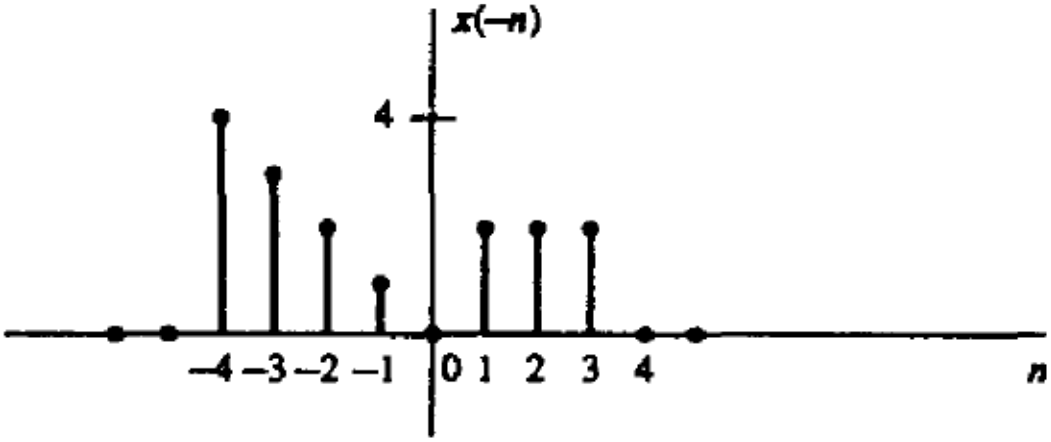
The signal *x* (n — 3) is obtained by delaying x(n) by three units in time. On the other hand, the signal *x(n* + 2) is obtained by advancing *x (n)* by two units in time. Note that delay corresponds to shifting a signal to the right, whereas advance implies shifting the signal to the left on the time axis.

Time Folding : The operations of folding is defined by

FD[x(n)] = *x (— n)*

Example:





Addition, multiplication, and scaling of sequences:

Amplitude modifications include *addition, multiplication,* and *scaling* o f discrete-time signals. *Amplitude scaling* o f a signal by a constant *A* is accomplished by multiplying the value o f every signal sample by *A.*

$$y(n) = Ax(n) \qquad -\infty < n < \infty$$

The sum of two signals x1(n) an d x2(n) is a signal y(n), whose value at any instant is equal to the sum of the values of these two signals at that instant, that is.

$$y(n) = x_1(n) + x_2(n) \qquad -\infty < n < \infty$$

The product of two signals is similarly defined on a sample -to -sample basis as

$$y(n) = x_1(n)x_2(n) \qquad -\infty < n < \infty$$

DISCRETE-TIME SYSTEMS :

A *discrete-time system* is a device or algorithm that operates on a discrete -time signal, called the *input* o r *excitation,* according to some w ell-defined rule, to produce another discrete-time signal called the *output* or *response* of the system .

We say that the input signal *x(n)* is *Transformed* by the system in to a signal y(n), and the general relationship Between x(n) and *y(n*) as

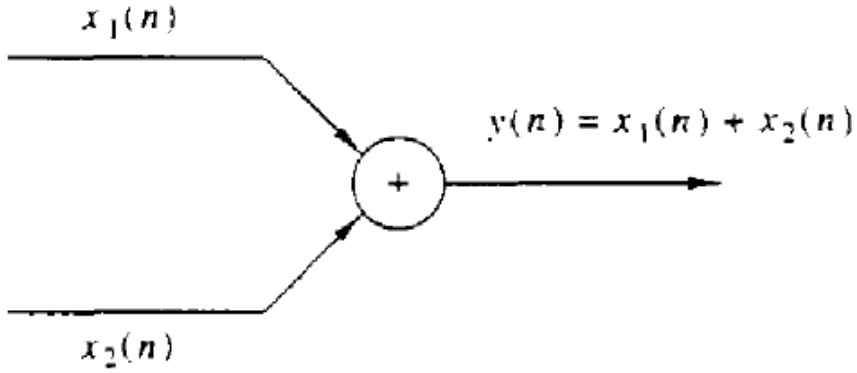
$$y(n) \equiv \mathcal{T}[x(n)]$$

where the symbol *T* denotes the transformation (also called an operator), or processing performed by the system on x(n) to produce y(n).

Representation of Discrete-Time Systems :

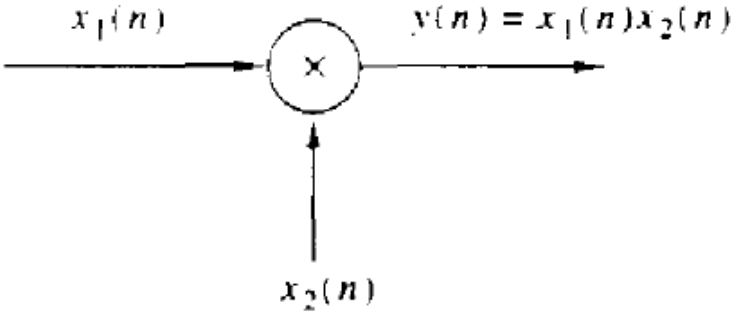
It is useful at this point to introduce a block diagram representation of discrete time systems. For this purpose we need to define some basic building blocks that can be interconnected to form complex systems.

An adder: Figure below illustrates a system (adder) that perform s the addition o f two signal sequences to form another (the sum) sequence, which we denote as y(n).



A constant multiplier: This operation is depicted by below Fig., and simply represents applying a scale factor on the input *x (n).*

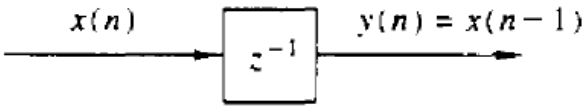
A signal multiplier: Figure below illustrates the multiplication of two signal sequences to form another (the product) sequence, denoted in the figure as y(n). we can view the multiplication operation as memory less.



A unit delay element: The unit delay is a special system that simply delays the signal passing th rough it by one sample. Fig. below illustrates such a system .If the input signal is *x(n),* the output is *x(n* — 1). In fact, the sample *x{n* — 1) is stored in memory at time *n* — 1 an d it is recalled fro m memory at time *n* to form y(n),

$$y(n) = x(n-1)$$

T h e use o f the symbol z^{-1} to denote the unit of delay



A unit advance element: In contrast to the unit delay, a unit advance moves the input *x (n*) ahead by one sample in time to yield *x(n* + 1). Fig. below illustrates this operation , with the operator z being used to denote the unit advance.

Classification of Discrete-Time Systems :
There are various types of Discrete-Time Systems such as

1-Static versus dynamic systems:
A discrete-tim e system is called *static* or memory less if its output at any instant n depends at most on the input sample at the same time, but not on past or future samples of the input. In any other case, the system is said to be *dynamic* or to have memory .T h e systems described by the following input-output equations are both static or memory less

y(n) = *a x {n)*
y (n) = *nx(n)* + *b x* ^{3}*(n)*

On the other hand , the systems described by the following input-output relations are dynamic systems or systems with memory.

$$y(n) = x(n) + 3x(n-1)$$

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

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

Time-invariant versus time-variant systems: We can subdivide the general class of systems in to the two broad categories, time -invariant systems and time -variant systems. A system is called time-in variant if its input-output characteristics do not change with time.
A relaxed system *T* is *time invariant* o r *shift invariant* if and
only if

$$x(n) \xrightarrow{\mathcal{T}} y(n)$$

implies that for every in p u t signal x(n) a n d every time shift *k.*

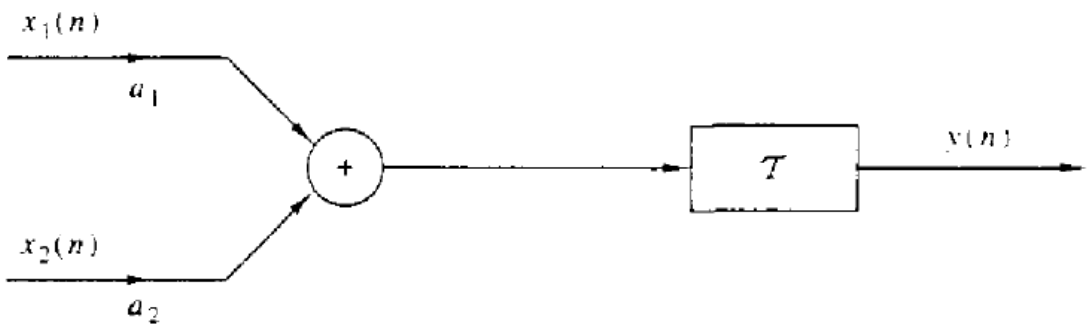
$$x(n-k) \xrightarrow{\mathcal{T}} y(n-k)$$

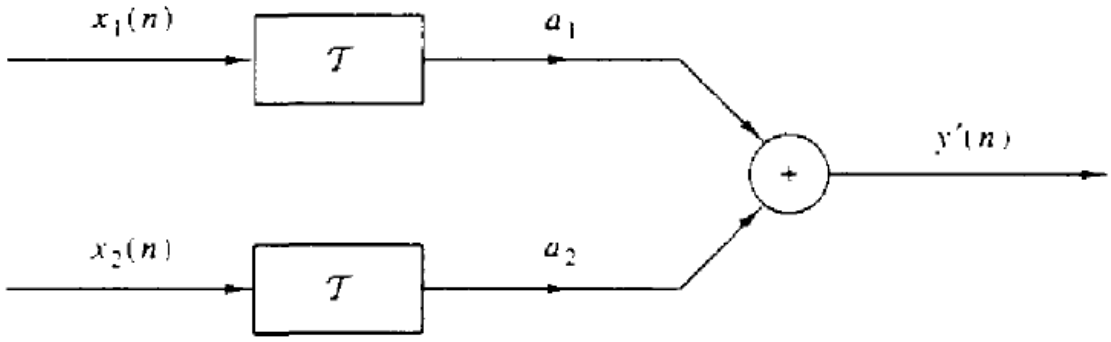
Now if this output y{n, k) = y{n — k), for all possible values o f k, the system is time invariant. O n the other hand , if the output y(n, k) ≠ y(n — k), even for one value o f k, the system is time variant.

Linear versus nonlinear systems: The general class o f system s can also be subdivided into linear system s and nonlinear system s. A linear system is one that satisfies the *superposition principle.* Simply stated, the principle o f superposition requires that the response o f the system to a weighted sum o f signals be equal to the corresponding weighted sum of the responses (outputs) of the system to each of the individual input signals**.**
A relaxed T system is linear if and only if

$$\mathcal{T}[a_1x_1(n) + a_2x_2(n)] = a_1\mathcal{T}[x_1(n)] + a_2\mathcal{T}[x_2(n)]$$

for any arbitrary input sequences $x_1(n)$ and $x_2(n)$, and any arbitrary constants a_1 and a_2.





Causal versus noncausal systems:
A system is said to be *causal* if the output of the system at any time *n* [i.e., *y(n)]* depends only on present and past inputs [i.e., *x { n*), *x(n* - 1),*x(n* — 2) , . . .] , but does not depend on future inputs [i.e., *x(n* + 1), *x(n* + 2) , . . .] . In mathematical terms, the output of a causal system satisfies an equation of the form

$$y(n) = F[x(n), x(n-1), x(n-2), \ldots]$$

If a system does not satisfy this definition, it is called *noncausal.* Such a system has an output tha t depends not only on present and past inputs but also on future inputs.

Stable versus unstable systems:
An arbitrary relaxed system is said to be stable if an d only if every bounded input produces a bounded output (i:e; BIBO).

The conditions that the input sequence *x{n)* and the output sequence *y(n)* are bounded is transla ted mathematically to mean that there exist some finite numbers, say *M x* and *M y.* such that

$$|x(n)| \leq M_x < \infty \qquad |y(n)| \leq M_y < \infty$$

for all *n.* If. for some bounded input sequence ,x(n), the output is unbounded (infinite), the system is classified as unstable .

DISCRETE-TIME LINEAR TIME-INVARIANT SYSTEMS:
The linearity and time-invariance properties of the system , the response of the system to any arb itrary input signal can be expressed in terms of the unit sample response of the system . The gen eral form of the expression that relates the unit sample response of the system and the arbitrary input signal to the output signal, called the convolution sum or the convolution formula, is also derived. Thus we are able to determine the output of any linear, time-invariant system to any arbitrary input signal.

Response of LTI Systems to Arbitrary Inputs:

The Convolution Sum :
An arbitrary input signal *x(n)* in to a weighted sum of impulses, We are now ready to determine the response of any relaxed linear system to any Input signal. First, we denote the response y(n, k) of the system to the input unit Sample sequence at *n* = k by the special symbol *h(n, k),* $-\infty < k < \infty$. T h a t is,

$$y(n, k) \equiv h(n, k) = \mathcal{T}[\delta(n-k)]$$

if the input is the arbitrary signal x(n) that is expressed as a sum of weighted impulses, that is.

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

then the response of the system to *x(n)* is the corresponding sum of weighted outputs, that is,

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

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

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

Clearly, the above equation follows from the superposition property of linear systems, and is know n as the *superposition summation.*th en by the time-invariance property , the response of the system to the delayed unit sample sequence δ(n - *k)* is

$$h(n-k) = \mathcal{T}[\delta(n-k)]$$

Consequently , the *superposition summation* formula in reduces to

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

The above formula gives the response *y(n)* of the LTI system as a function of the input signal *x (n*) and the unit sample (impulse) response *h(n)* is called a *convolution sum.*

To summarize, the process of computing the convolution between *x (k)* and *h(k)* involves the following four steps**.**

1. *Folding.* **Fold** *h(k)* **about** *k* **= 0 to obtain** *h (- k)* .

2. *Shifting,* Shift *h (—k)* by n_0 to the right (left) if n_0 is positive (negative), to obtain *h*(n_0— k).

3. *Multiplication.* Multiply *x (k)* by *h*(n_0— k) to obtain the product sequence$v_{n0}(k) = x(k)\, h(n_0 - k)$.

4. *Summation.* Sum all the values o f the product sequence $v_{n0}(k)$ to obtain the value of the output at time $n = n_0$.

Example:

The impulse response of a linear time-invariant system is

$$h(n) = \{\underset{\uparrow}{1}, 2, 1, -1\}$$

Determine the response of the system to the input signal

$$x(n) = \{\underset{\uparrow}{1}, 2, 3, 1\}$$

Solution : We shall compute the convolution according to its formula. But we shall use graphs of the sequences to aid us in the computation. In Fig. below we illustrate the input signal sequence *x(k)* and the impulse response *h{k)* of the system, using *k* as the time index. The first step in the computation of the convolution sum is to fold *h(k).* The folded sequence *h(-k)* is illustrated inconsequent figs . Now we can compute the output at *n* = 0. according to the convolution formula which is

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

Since the shift *n* = 0, we use *h(—k)* directly without shifting it. The product sequence

$$v_0(k) \equiv x(k)h(-k)$$

We continue the computation by evaluating the response of the system at *n* = 1.

$$y(1) = \sum_{h=-\infty}^{\infty} x(k)h(1-k)$$

Finally, the sum of all the values in the product sequence yields

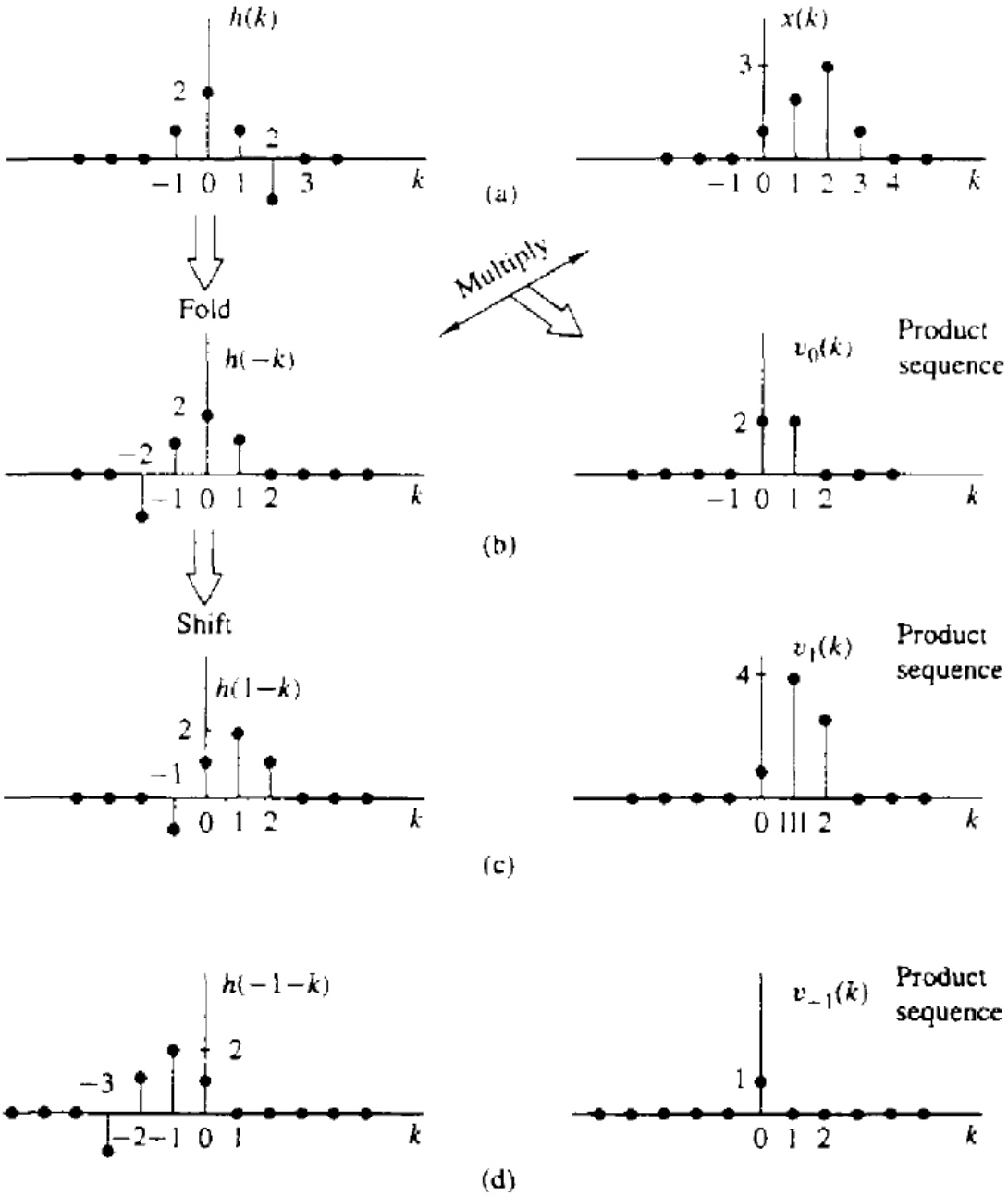
$$y(1) = \sum_{k=-\infty}^{\infty} v_1(k) = 8$$

In a similar manner, we can obtain y(2) by shifting *h (- k)* two units to the right. And y(2) = 8.

Then y(3) = 3. y(4) = - 2 , y(5) = -1 .For *n* >5, we find that y(n) = 0 because the product sequences contain all zeros.

Next we wish to evaluate y(n) for *n* < 0. We begin with *n =-1.Then*

$$y(-1) = \sum_{k=-\infty}^{\infty} x(k)h(-1-k)$$



$$y(0) = \sum_{k=-\infty}^{\infty} v_0(k) = 4$$

Finally, summing over the values of the product sequence, we obtain

$$y(-1) = 1$$

then

$$y(n) = 0 \qquad \text{for } n \leq -2$$

Now we have the entire response of the system for $-\infty < n < \infty$. which we summarize below as

$$y(n) = \{\ldots, 0, 0, 1, \underset{\uparrow}{4}, 8, 8, 3, -2, -1, 0, 0, \ldots\}$$

Properties of Convolution:

1- **Commutative law :**

$$x(n) * h(n) = h(n) * x(n)$$

2- **Associative law :**

$$[x(n) * h_1(n)] * h_2(n) = x(n) * [h_1(n) * h_2(n)]$$

3-Distributive law :

$$x(n) * [h_1(n) + h_2(n)] = x(n) * h_1(n) + x(n) * h_2(n)$$

Finite-Duration and Infinite-Duration Impulse Response system:

Linear time-invariant system s into two types, those that have a finite-duration Impulse response (FIR) and those that have an infinite-duration impulse response(IIR). Thus an fir system has an impulse response that is zero outside o f some Finite time interval.

Stability and unstable Linear Time-Invariant Systems :

We defined an arbitrary relaxed system as BIBO stable if and only if its output sequence y(n) is bounded for every bounded input *x(n).*

The output is bounded if the impulse response of the system satisfies the condition

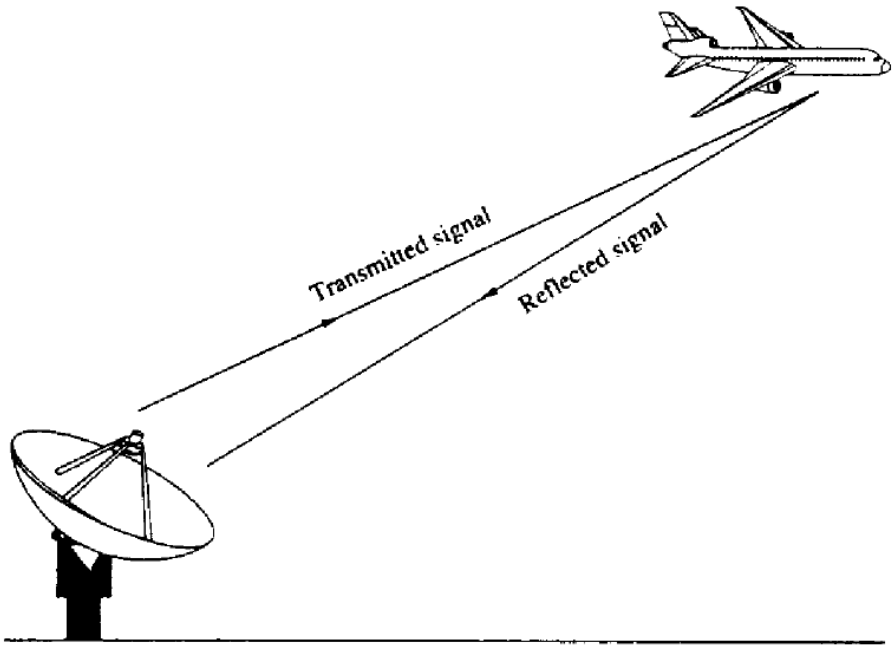
$$S_h \equiv \sum_{k=-\infty}^{\infty} |h(k)| < \infty$$

T hat is, *a linear time-invariant system is stable if its impulse response is absolutely summable* .

CORRELATION OF DISCRETE-TIME SIGNALS:

A mathematical operation that closely resembles convolution is correlation .Just as in the case of convolution , two signal sequences are involved in correlation. correlation between the two signals is to measure the degree to which the two signals are similar and thus to extract some in formation that depends to a large extent on the application. Correlation o f signals is often encountered in radar, sonar, digital communications, geology, an do the rare as in science and en gineering .

Let us suppose that we have two signal sequences x(n) and y(n) that we wish to compare. In radar and active sonar applications. x(n) can represent the sampled version of the transmitted signal and y{n) can represent the sampled version of the received signal at the output of the analog -to -digital (A /D) converter. If a target is p resent in the space being searched by the radar or sonar, the received signal y(n) consists of a delayed version of the transmitted signal, reflected from the target.



This comparison process is performed by means of the correlation operation of 2 different types.

Cross-correlation and Autocorrelation Sequences :

Suppose that we have two real signal sequences *x(n)* and *y(n)* each of which has finite energy. T he *cross-correlation* o f *x(n)* and y(n) is a sequence $r_{xy}(l)$, which is defined as

$$r_{xy}(l) = \sum_{n=-\infty}^{\infty} x(n)y(n-l) \qquad l = 0, \pm 1, \pm 2, \ldots$$

or, equivalently , as

$$r_{xy}(l) = \sum_{n=-\infty}^{\infty} x(n+l)y(n) \qquad l = 0, \pm 1, \pm 2, \ldots$$

The index l is the (time) shift (or *lag)* parameter and the subscripts *x y* on the cross-correlation se quence $r_{xy}(l)$, indicate the sequences being correlated .If we reverse the roles of x(n) an d y(n) and there fore reverse the order of the indices xy. we obtain the cross-correlation sequence

$$r_{yx}(l) = \sum_{n=-\infty}^{\infty} y(n)x(n-l)$$

or, equivalently ,

$$r_{yx}(l) = \sum_{n=-\infty}^{\infty} y(n+l)x(n)$$

By comparing the above 4 equations we conclude that

$$r_{xy}(l) = r_{yx}(-l)$$

Hence , $r_{yx}(l)$ provides exactly the same information as $r_{xy}(l)$,with respect to the similarity of $x(n)$ to y(n).

Example:

Determine the cross-correlation sequence $r_{xy}(l)$ of the sequences

$$x(n) = \{\ldots, 0, 0, 2, -1, 3, 7, \underset{\uparrow}{1}, 2, -3, 0, 0, \ldots\}$$

$$y(n) = \{\ldots, 0, 0, 1, -1, 2, -2, \underset{\uparrow}{4}, 1, -2, 5, 0, 0, \ldots\}$$

Solution : Let us use the definition of cross-correlation to compute $r_{xy}(l)$. For $I = 0$ w e have

$$r_{xy}(0) = \sum_{n=-\infty}^{\infty} x(n)y(n)$$

The product sequence $v_0(n) = x(n)\, y(n)$ is

$$v_0(n) = \{\ldots, 0, 0, 2, 1, 6, -14, \underset{\uparrow}{4}, 2, 6, 0, 0, \ldots\}$$

and hence the sum over all values of n is

$$r_{xy}(0) = 7$$

For $I > 0$, we simply shift y(n) to the right relative to x(n) hy l units, compute the product sequence $v_l(n)$ = x(n)y(n — I), and finally, sum over all values o f the product sequence. Thus we obtain

$$r_{xy}(1) = 13, \quad r_{xy}(2) = -18, \quad r_{xy}(3) = 16, \quad r_{xy}(4) = -7$$

$$r_{xy}(5) = 5, \quad r_{xy}(6) = -3, \quad r_{xy}(l) = 0, \quad l \geq 7$$

For l< 0, we shift $y(n)$ to the left relative to x(n) by l units, compute the product sequence $v_l(n)$ = x(n)y(n — I), and sum over all values of the product sequence. Thus we obtain the values of the cross-correlation sequence

$$r_{xy}(-1) = 0, \quad r_{xy}(-2) = 33, \quad r_{xy}(-3) = -14, \quad r_{xy}(-4) = 36$$

$$r_{xy}(-5) = 19, \quad r_{xy}(-6) = -9, \quad r_{xy}(-7) = 10, \quad r_{xy}(l) = 0, \ l \leq -8$$

Therefore, the cross-correlation sequence of $x(n)$ and y(n) is

$$r_{xy}(l) = \{10, -9, 19, 36, -14, 33, 0, \underset{\uparrow}{7}, 13, -18, 16, -7, 5, -3\}$$

Then the convolution o f $x(n)$ with y (—n) yields the cross-correlation $r_{xy}(l)$ that is,

$$r_{xy}(l) = x(l) * y(-l)$$

Autocorrelation*:*
when *y(n) = x(n),* we have the *autocorrelation* of x(n),which is defined as the sequence

$$r_{xx}(l) = \sum_{n=-\infty}^{\infty} x(n)x(n-l)$$

or, equivalently, as

$$r_{xx}(l) = \sum_{n=-\infty}^{\infty} x(n+l)x(n)$$

For finite-duration sequences,

$$r_{xy}(l) = \sum_{n=i}^{N-|k|-1} x(n)y(n-l)$$

and

$$r_{xx}(l) = \sum_{n=i}^{N-|k|-1} x(n)x(n-l)$$

where $i = l$, $k = 0$ for l> 0, and i = 0, $k = l$ for l < 0.

Properties of the Autocorrelation and Crosscorrelation Sequences :

1-The cross-correlation sequence satisfies the condition that

$$|r_{xy}(l)| \le \sqrt{r_{xx}(0)r_{yy}(0)} = \sqrt{E_x E_y}$$

when y(n) = *x (n*), reduces to

$$|r_{xx}(l)| \le r_{xx}(0) = E_x$$

2-Th e normalized auto correlation sequence is defined as

$$\rho_{xx}(l) = \frac{r_{xx}(l)}{r_{xx}(0)}$$

Similarly, we define the normalized cross-correlation sequence

$$\rho_{xy}(l) = \frac{r_{xy}(l)}{\sqrt{r_{xx}(0)r_{yy}(0)}}$$

Now $|\rho_{xx}(l)| < 1$ and $|\rho_{xy}(l)| < 1$, and hence these sequences are independent of signal scaling.

3-the cross-correlation sequence satisfies the property

$$r_{xy}(l) = r_{yx}(-l)$$

the autocorrelation sequence satisfies the property

$$r_{xx}(l) = r_{xx}(-l)$$

Hence the auto correlation function is an even function.

MODULE-2

The One-sided z-Transform:

The one-sided or unilateral z-transform of a signal x(n) is defined by

$$X^{+}(z) \equiv \sum_{n=0}^{\infty} x(n) z^{-n} \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots..(1.1)$$

Properties:

1. It does not contain information about the signal x(n) for negative values of time.
2. It is unique only for causal signals.
3. The one-sided z-transform $X^{+}(z)$ of x(n) is identical to the two-sided z-transform of the signal x(n)u(n).

Shifting Property:

- Time delay:

If $\quad x(n) \overset{z^{+}}{\leftrightarrow} X^{+}(z)$

then $\quad x(n-k) \overset{z^{+}}{\leftrightarrow} z^{-k}[X^{+}(z) + \sum_{n=1}^{k} x(-n) z^{n}] \quad$ k > 0 $\quad \ldots\ldots(1.2)$

In case x(n) is a causal signal

then $\quad x(n-k) \overset{z^{+}}{\leftrightarrow} z^{-k} X^{+}(z) \quad$ k > 0 $\quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(1.3)$

- Time advance:

$$x(n+k) \overset{z^{+}}{\leftrightarrow} z^{k}[X^{+}(z) - \sum_{n=0}^{k-1} x(n) z^{-n}] \qquad k > 0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots(1.4)$$

Final Value Theorem:

If $\quad x(n) \overset{z^{+}}{\leftrightarrow} X^{+}(z)$

then $\quad \lim_{n\to\infty} x(n) = \lim_{z\to 1} (z-1)\, X^{+}(z) \quad \ldots\ldots\ldots\ldots\ldots\ldots....(1.5)$

The limit exists if the ROC of $(z-1)X^{+}(z)$ includes the unit circle.

<u>Analysis of LTI System in z-domain:</u>

Response of Systems with Rational System:

We consider a linear constant coefficient difference equation:

$$y(n) = -\sum_{k=1}^{N} a_k\, y(n-k) + \sum_{k=0}^{M} b_k x(n-k) \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.1)$$

corresponding system function H(z) is given by

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1+\sum_{k=1}^{N} a_k z^{-k}} \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.2)$$

we apply an input signal x(n) whose z-transform is X(z). For $X(z) = \frac{N(z)}{Q(z)}$, $H(z) = \frac{B(z)}{A(z)}$ and zero initial conditions, the z-transform of the output of the system has the form

$$Y(z) = H(z)X(z) = \frac{B(z)N(z)}{A(z)Q(z)} \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.3)$$

Suppose the system contains simple poles p_1 , p_2 , … … … , p_N and X(z) contains poles q_1 , q_2 , … … … , q_L , where $p_k \neq q_m$ for all k = 1, 2 , …… , N and m = 1, 2 , ……. , L. Assuming no pole-zero cancellation the partial fraction expansion of Y(z) yields

$$Y(z) = \sum_{k=1}^{N} \frac{A_k}{1-p_k z^{-1}} + \sum_{k=1}^{L} \frac{Q_k}{1-q_k z^{-1}} \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.4)$$

The inverse transform of Y(z) is the output signal y(n) from the system:

$$y(n) = \sum_{k=1}^{N} A_k (p_k)^n u(n) + \sum_{k=1}^{L} Q_k (q_k)^n u(n) \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.5)$$

where scale factors $\{A_k\}$and $\{Q_k\}$are functions of both sets of poles $\{p_k\}$and $\{q_k\}$.

Response of Pole-Zero Systems with Non-zero Initial Conditions:

We consider the input signal x(n) to be a causal signal applied at n=0. The effects of all previous input signals to the system are reflected in the initial conditions y(-1), y(-2), ... , y(-N). We are interested in determining the output y(n) for $n \geq 0$.

$$Y^{+}(z) = -\sum_{k=1}^{N} a_k z^{-k} \left[Y^{+}(z) + \sum_{n=1}^{k} y(-n) z^{n} \right] + \sum_{k=0}^{M} b_k z^{-k} X^{+}(z)$$

Causality and Stability:

A causal linear time invariant system is one whose unit sample response h(n) satisfies the condition

$$h(n) = 0 \qquad n < 0$$

An LTI system is causal if and only if the ROC of the system function is the exterior of a circle of radius $r < \infty$, including the point $z = \infty$.

A necessary and sufficient condition for an LTI system to be BIBO stable is

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

An LTI system is BIBO stable if and only if the ROC of the system function includes the unit circle.

Consequently, a causal and stable system must have a system function that converges for $|z| > r < 1$. Since the ROC cannot contain any poles of H(z) , it follows that ***a causal linear time-invariant system is BIBO stable if and only if all the poles of H(z) are inside the unit circle***.

<u>The DFT as a Linear Transformation:</u>

The formulas for the DFT and IDFT may be expressed as

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn} \quad , \qquad k = 0,1, \dots, N-1 \quad \dots\dots\dots\dots\dots (3.1)$$

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k)\, W_N^{-kn} \quad , \qquad n = 0,1, \dots, N-1 \quad \dots\dots\dots\dots\dots\dots (3.2)$$

where $\qquad W_N = e^{\frac{-j2\pi}{N}}$

which is an Nth root of unity.

The computation of each point of the DFT can be accomplished by N complex multiplications and (N-1) complex additions. Hence the N-point DFT values can be computed in a total of N^2 complex multiplications and N(N-1) complex additions.

Let us define an N-point vector $\mathbf{x}_N$ of the signal sequence x(n), n=0,1,…,N-1, an N-point vector $\mathbf{X}_N$ of frequency samples, and an $N \times N$ matrix $\boldsymbol{W}_N$ as

$$\mathbf{x}_N = \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ x(N-1) \end{bmatrix}, \quad \mathbf{X}_N = \begin{bmatrix} X(0) \\ X(1) \\ \vdots \\ X(N-1) \end{bmatrix}$$

$$\boldsymbol{W}_N = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_N & W_N^2 & \cdots & W_N^{N-1} \\ \vdots & W_N^2 & W_N^4 & \cdots & W_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & W_N^{N-1} & W_N^{2(N-1)} & \cdots & W_N^{(N-1)(N-1)} \end{bmatrix} \quad \ldots\ldots\ldots\ldots\ldots\ldots (3.3)$$

With these definitions, the N-point DFT may be expressed in the matrix form as

$$\mathbf{X}_N = \mathbf{W}_N \mathbf{x}_N \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3.4)$$

where $\mathbf{W}_N$ is the matrix of the linear transformation. $\mathbf{W}_N$ is a symmetric matrix. If we assume that the inverse of $\mathbf{W}_N$ exists, then we also write

$$\mathbf{x}_N = \mathbf{W}_N{}^{-1}\mathbf{X}_N \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3.5)$$

IDFT can also be expressed as

$$\mathbf{x}_N = \frac{1}{N}\mathbf{W}_N^*\mathbf{X}_N \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3.6)$$

where $\mathbf{W}_N^*$ denotes the complex conjugate of the matrix $\mathbf{W}_N$. Comparison of equations 3.5 and 3.6 leads us to conclude that

$$\mathbf{W}_N{}^{-1} = \frac{1}{N}\mathbf{W}_N^* \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3.7)$$

which in turn implies

$$\mathbf{W}_N\mathbf{W}_N^* = N\mathbf{I}_N \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3.8)$$

where $\mathbf{I}_N$ is a $N \times N$ identity matrix.

<u>Circular Convolution:</u>

Suppose that we have two finite-duration sequences of length N, $x_1(n)$ and $x_2(n)$. Their respective N point DFTs are

$$X_1(k) = \sum_{n=0}^{N-1} x_1(n)e^{-j2\pi nk/N} \quad , \quad k = 0,1,\ldots,N-1 \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (4.1)$$

$$X_2(k) = \sum_{n=0}^{N-1} x_2(n) e^{-j2\pi nk/N} \quad , \quad k = 0,1,\dots,N-1 \quad \dots\dots\dots\dots\dots\dots\dots (4.2)$$

Multiplying the above two DFTs we get:

$$X_3(k) = X_1(k)X_2(k) \ , \qquad k = 0,1,\dots,N-1 \quad \dots\dots\dots\dots\dots\dots\dots (4.2)$$

IDFT of $\{X_3(k)\}$ is

$$x_3(m) = \frac{1}{N}\sum_{n=0}^{N-1} X_3(k) e^{\frac{j2\pi km}{N}}$$

$$= \frac{1}{N}\sum_{n=0}^{N-1} X_1(k)X_2(k) e^{\frac{j2\pi km}{N}} \quad \dots\dots\dots\dots\dots\dots\dots (4.3)$$

Substituting for $X_1(k)$ and $X_2(k)$ in (4.3) using DFTs given in (4.1) and (4.2), we obtain

$$x_3(m) = \frac{1}{N}\sum_{n=0}^{N-1}\left[\sum_{n=0}^{N-1} x_1(n) e^{-\frac{j2\pi nk}{N}}\right]\left[\sum_{n=0}^{N-1} x_2(n) e^{-\frac{j2\pi nk}{N}}\right] e^{\frac{j2\pi km}{N}} \quad \dots\dots\dots\dots (4.4)$$

The inner sum in the brackets in (4.4) has the form

$$\sum_{k=0}^{N-1} a^k = \begin{cases} N\,, & a = 1 \\ \dfrac{1-a^N}{1-a}, & a \neq 1 \end{cases} \quad \dots\dots\dots\dots\dots (4.5)$$

where a is defined as

$$a = e^{j2\pi(m-n-l)/N}$$

Consequently,

$$\sum_{k=0}^{N-1} a^k = \begin{cases} N\,, & l = m-n+pN = ((m-n))_N\,, \quad p\ an\ integer \\ 0\,, & otherwise \end{cases} \quad \dots\dots\dots (4.6)$$

If we substitute the result in (4.6) into (4.4) , we obtain

$$\boldsymbol{x_3(m) = \sum_{n=0}^{N-1} x_1(n) x_2\left((m-n)\right)_N} \quad , \qquad m = 0,1,\dots,N-1 \quad \dots\dots\dots (4.7)$$

The above convolution sum is called ***circular convolution***. Thus we conclude that ***multiplication of the DFTs of two sequences is equivalent to the circular convolution of the two sequences in the time domain.***

<u>Linear Filtering Methods Based on the DFT</u>:

Use of the DFT in Linear Filtering:

Suppose we have a finite-duration sequence x(n) of length L which excites an FIR filter of length M. Let

$$x(n) = 0, \quad n < 0 \; and \; n \geq L$$

$$h(n) = 0, \quad n < 0 \; and \; n \geq M$$

where $h(n)$ is the impulse response of the FIR filter.

The output sequence $y(n)$ of the FIR filter:

$$y(n) = \sum_{k=0}^{M-1} h(k)x(n-k) \qquad \dots\dots\dots\dots\dots\dots (5.1)$$

The duration of $y(n)$ is $L + M - 1$.

The frequency-domain equivalent to (5.1) is

$$Y(\omega) = X(\omega)H(\omega) \quad \dots\dots\dots\dots\dots\dots\dots\dots (5.2)$$

If the sequence $y(n)$ is to be represented uniquely in the frequency domain by samples of its spectrum $Y(\omega)$ at a set of discrete frequencies, the number of distinct samples must equal or exceed $L + M - 1$. Therefore, a DFT of size $N \geq L + M - 1$ is required to represent {y(n)} in the frequency domain.

Now if

$$Y(k) \equiv Y(\omega)\big|_{\omega=\frac{2\pi k}{N}}, \qquad k = 0,1,\dots,N-1$$

$$= X(\omega)H(\omega)\big|_{\omega=\frac{2\pi k}{N}}, \qquad k = 0,1,\dots,N-1$$

then

$$Y(k) = X(k)H(k), \qquad k = 0,1,\dots,N-1 \qquad \dots\dots\dots\dots\dots..(5.3)$$

where $\{X(k)\}$ and $\{H(k)\}$ are the N-point DFTs of the corresponding sequences x(n) and h(n), respectively. Since the sequences x(n) and h(n) have a duration less than N, we simply pad these sequences with zeros to increase their length to N.

Since the $(N = L + M - 1)$-point DFT of the output sequence y(n) is sufficient to represent y(n) in the frequency domain, it follows that the multiplication of the N-point DFTs X(k) and H(k) followed by the computation of the N-point IDFT, must yield sequence {y(n)}.

Thus, ***the N-point circular convolution of x(n) with h(n) must be equivalent to the linear convolution of x(n) with h(n)***. *Thus with zero padding, the DFT can be used to perform linear filtering.*

<u>Filtering of Long Data Sequences:</u>

Let the FIR filter has duration M. The input data sequence is segmented into blocks of L points, where , by assumption, $L \gg M$.

Overlap-save method:

Size of input data blocks, $N = L + N - 1$

DFTs and IDFTs are of length N.

Each data block consists of the last $M - 1$ data points of the previous data block followed by L new data points to form a data sequence of length $N = L + N - 1$. An N-point DFT is computed for each data block.

The impulse response of the FIR filter is increased in length by appending $L - 1$ zeros and an N-point DFT of the sequence is computed once and stored. The multiplication of the two N-point DFTs $\{H(k)\}$ and $\{X_m(k)\}$ for the mth block of data yields

$$\hat{Y}_m(k) = H(k)X_m(k), \qquad k = 0,1, \dots, N-1 \qquad \dots\dots\dots\dots (5.4.1)$$

Then the N-point IDFT yields the result

$$\hat{Y}_m(n) = \{\hat{y}_m(0)\hat{y}_m(1) \dots \hat{y}_m(M-1)\hat{y}_m(M) \dots \hat{y}_m(N-1)\} \qquad \dots\dots\dots (5.4.2)$$

Since the data record is of length N, the first $M - 1$ points of $y_m(n)$ are corrupted by aliasing and must be discarded. The last L points of $y_m(n)$ are exactly same as the result from linear convolution and, as a consequence,

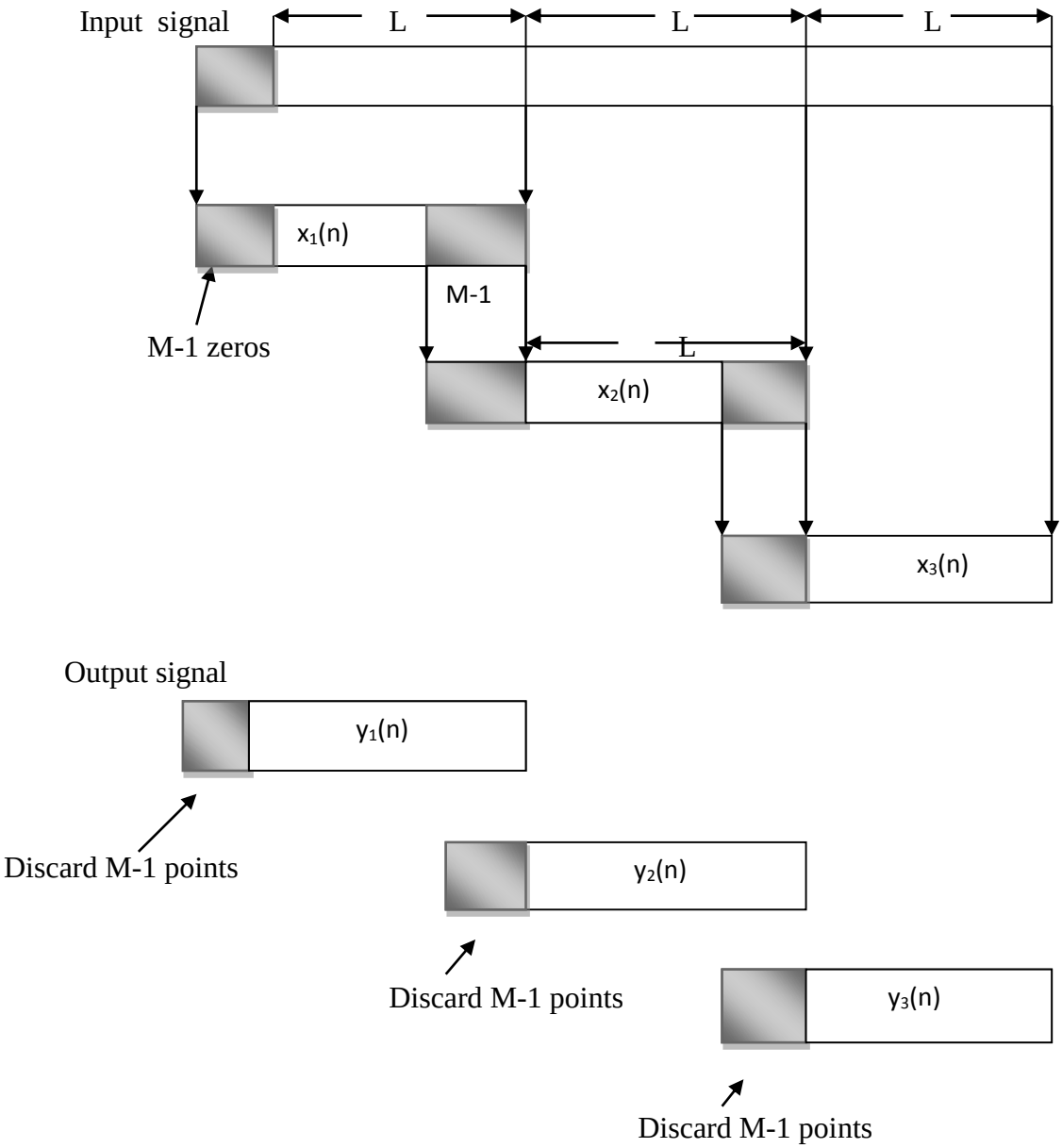
$$\hat{y}_m(n) = y_m(n)\,, \qquad n = M, M+1, \dots, N-1 \qquad \dots\dots\dots\dots (5.4.3)$$

To avoid loss of data due to aliasing, the last $M-1$ points of each data record are saved and these points become the first $M-1$ points of the subsequent record. To begin the processing, the first $M-1$ points of the first record are set to zero. Thus blocks of data sequences are:

$$x_1(n) = \left\{ \underbrace{0, 0, \dots, 0}_{M-1\ points}, x(0), x(1), \dots, x(L-1) \right\} \qquad \dots\dots\dots (5.4.4)$$

$$x_2(n) = \left\{ x\underbrace{(L-M+1), \dots, x(L-1)}_{M-1\ data\ points\ from\ x_1(n)}, \underbrace{x(L), \dots, x(2L-1)}_{L\ new\ data\ points} \right\} \qquad \dots\dots\dots (5.4.5)$$

$$\square_3(n) = \left\{ x\underbrace{(2L-M+1), \dots, x(2L-1)}_{M-1\ data\ points\ from\ x_2(n)}, \underbrace{x(2L), \dots, x(3L-1)}_{L\ new\ data\ points} \right\} \qquad \dots\dots\dots (5.4.6)$$



(Linear FIR filtering by the overlap-save method)

Overlap-add method:

Size of input block = L

Size of the DFTs and IDFT is $N = L + M - 1$.

To each data block we append $M - 1$ zeros and compute the N-point DFT. The data blocks may be represented as

$$x_1(n) = \left\{x(0), x(1), \dots, x(L-1), \underbrace{0,0,\dots,0}_{M-1\ zeros}\right\} \quad \dots\dots\dots\dots (5.5.1)$$

$$x_2(n) = \left\{x(L), x(L-1), \dots, x(2L-1), \underbrace{0,0,\dots,0}_{M-1\ zeros}\right\} \quad \dots\dots\dots\dots (5.5.2)$$

$$x_3(n) = \left\{x(2L), \dots, x(3L-1), \underbrace{0,0,\dots,0}_{M-1\ zeros}\right\} \quad \dots\dots\dots\dots (5.5.3)$$

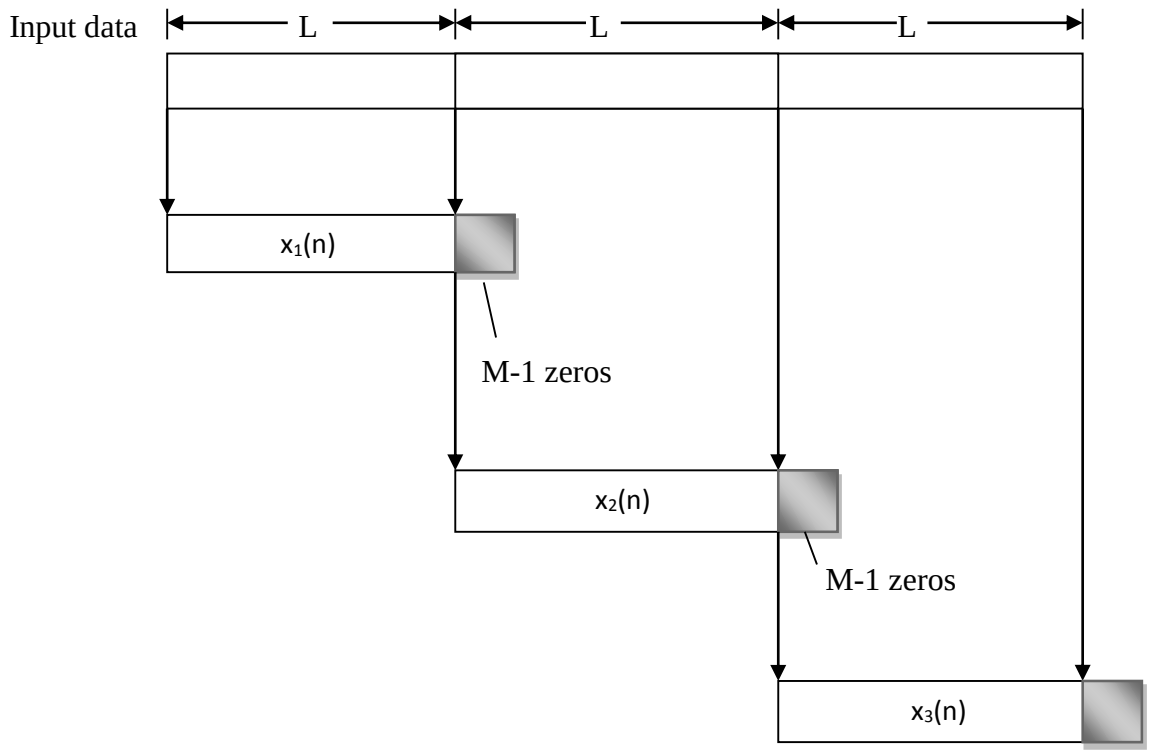
and so on. The two N-point DFTs are multiplied together to form

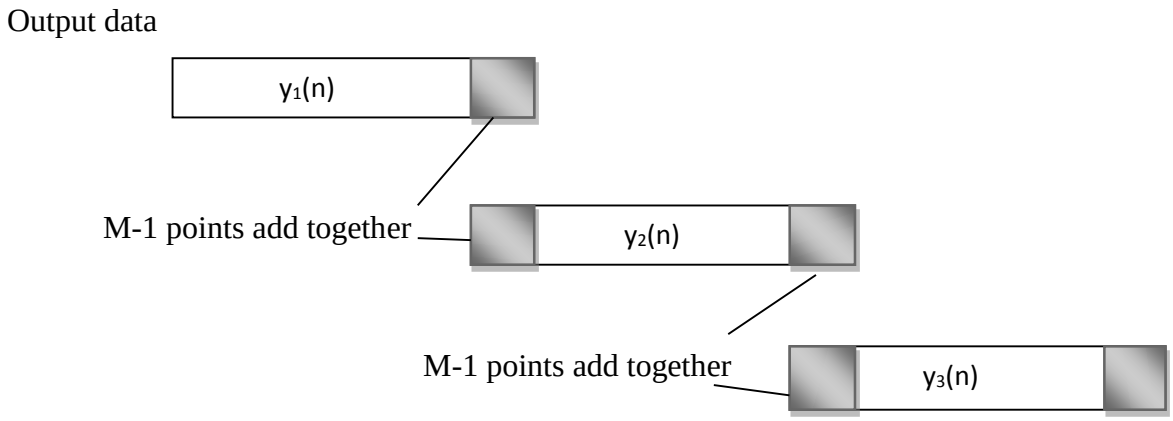
$$Y_m(k) = H(k)X_m(k), \qquad k = 0,1,\dots,N-1 \qquad \dots\dots.. (5.5.4)$$

The IDFT yields data blocks of length N that are free of aliasing, since the size of the DFTs and IDFT is $N = L + M - 1$ and the sequences are increased to N-points by appending zeros to each block.

Since each data block is terminated with M-1 zeros , the last M-1 points from each output block must be overlapped and added to the first M-1 points of the succeeding block. Hence this method is called the overlap-add method. The output sequence is:

$$y(n) = \{y_1(0), y_1(1), \dots, y_1(L-1), y_1(L)+y_2(0), y_1(L+1) + y_2(1), \dots, y_1(N-1) + y_2(M-1), y_2(M), \dots\} \qquad \dots\dots\dots\dots\dots.. (5.5.5)$$





(Linear FIR filtering by the overlap-add method)

<u>The Discrete Cosine Transform</u> :

Forward DCT:

Let an N-point sequence x(n) which is real and even, that is,

$x(n) = x(N-n), 0 \leq n \leq N-1$

Let s(n) be a 2N-point even symmetric extension of x(n) defined by

$$s(n) = \begin{cases} x(n), & 0 \le n \le N-1 \\ x(2N-n-1), & N \le n \le 2N-1 \end{cases} \quad \dots\dots\dots\dots(6.1)$$

The DCT of x(n) can be computed by taking the 2N-point DFT of s(n) and multiplying the result by $W_{2N}^{k/2}$. The forward DCT is defined by

$$V(k) = 2\sum_{n=0}^{N-1} x(n)\cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right], \quad 0 \le k \le N-1 \quad \dots\dots\dots\dots\dots\dots(6.2)$$

Inverse DCT

$$x(n) = \frac{1}{N}\left\{\frac{V(0)}{2} + \sum_{k=1}^{N-1} V(k)\cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right]\right\}, \quad 0 \le n \le N-1 \quad \dots\dots(6.3)$$

DCT as an Orthogonal Transform

The $N \times N$ DCT matrix C_N of the sequence $x(n),\ 0 \le n \le N-1$ is a real orthogonal matrix, that is, it satisfies

$$C_N^{-1} = C_N^T \quad \dots\dots\dots(6.4)$$

Orthogonality simplifies the computation of the inverse transform because it replaces matrix inversion by matrix transposition.

<u>Circular Correlation</u> :

If x(n) and y(n) are two periodic sequences, each with period N, then their cross correlation sequence is defined as

$$r_{xy}(l) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)y(n-l) \quad \dots\dots\dots\dots(7.1)$$

Module-III

Fast Fourier Transform Algorithms:

1. Introduction

For a finite-duration sequence *x(n)* of length *N*, the DFT sum may be written as

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, k = 0,1, \dots, N-1$$

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 $(N\text{-}1)^2$ 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 $W_N^{k+N/2} = -W_N^k$
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:

$W_8^0 = -W_8^4 = 1 \quad W_8^1 = -W_8^5 = \frac{1-j}{\sqrt{2}}$

$W_8^2 = -W_8^6 = -j \quad W_8^3 = -W_8^7 = -\frac{1+j}{\sqrt{2}}$

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

Number of real multiplications = $4\left(\frac{N}{2}\log_2 N\right) = 2N \log_2 N$

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 82 = 64 complex multiplications, but the radix-2 FFT requires only $12\left(=\frac{8}{2}\log_2 8 = 4 \times 3\right)$.

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

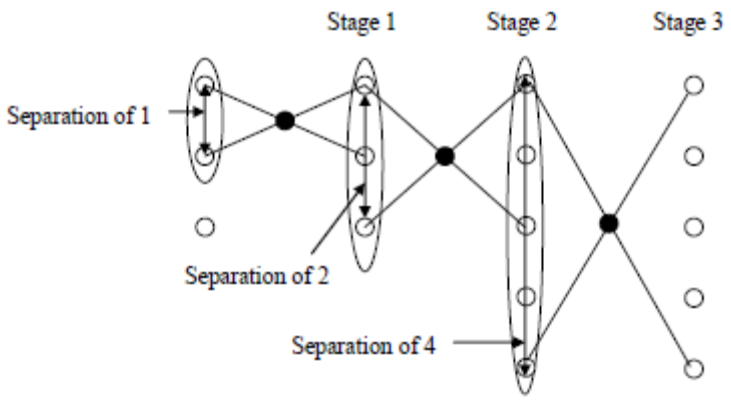
$N = N1\ N2 \ldots Nm$

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.

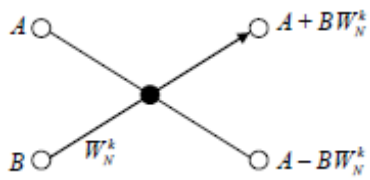
2. Radix-2 decimation-in-time FFT (Cooley-Tukey)

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 $\frac{N}{2} \log_2 N$.
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 ($A + B\,W_N^k$) and ($A - B\,W_N^k$) can be computed and stored in the same locations as A and B.



Example 1 Radix-2, 8-point, decimation-in-time FFT for the sequence

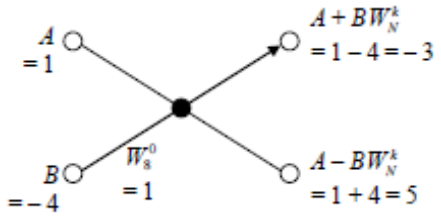
$n \rightarrow$ 0 1 2 3 4 5 6 7
$x(n)$ = {1, 2 3 4 –4 –3 –2 –1}

Solution The twiddle factors are

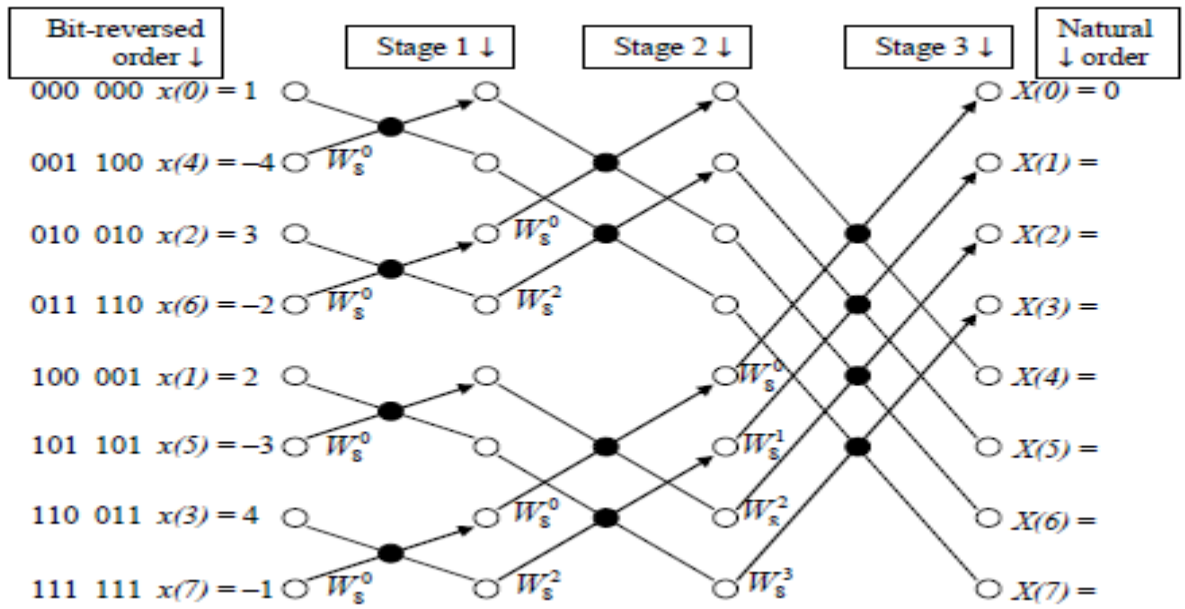
$W_8^0 = 1$ $\qquad$ $W_8^1 = e^{-j2\pi/8} = e^{-j\pi/4} = \frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}}$

$W_8^2 = \left(e^{-j2\pi/8}\right)^2 = e^{-j\pi/2} = -j$ $\qquad$ $W_8^3 = \left(e^{-j2\pi/8}\right)^3 = e^{-j3\pi/4} = -\frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}}$

One of the elementary computations is shown below:



The signal flow graph follows:



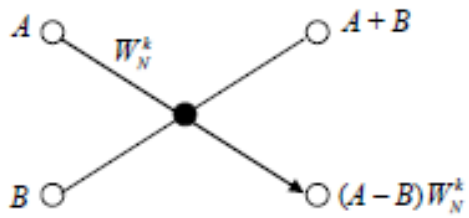
The DFT is

$X(k)$ = {0, (5 – j12.07), (–4 + j4), (5 – j2.07), –4, (5 + j2.07), (–4 – j4), (5 + j12.07)}

3. 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: Radix-2, 8-point, decimation-in-frequency FFT for the sequence

$n \rightarrow$ 0 1 2 3 4 5 6 7

$x(n) = \{1, 2\ 3\ 4\ -4\ -3\ -2\ -1\}$

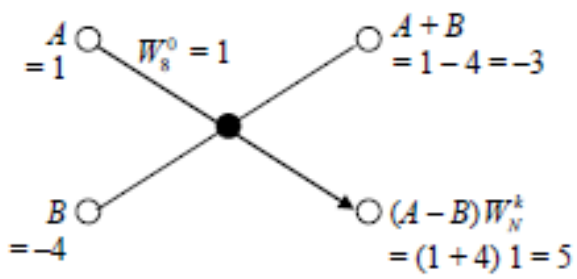
Solution :

The twiddle factors are the same as in the DIT FFT done earlier (both being 8-point DFTs):

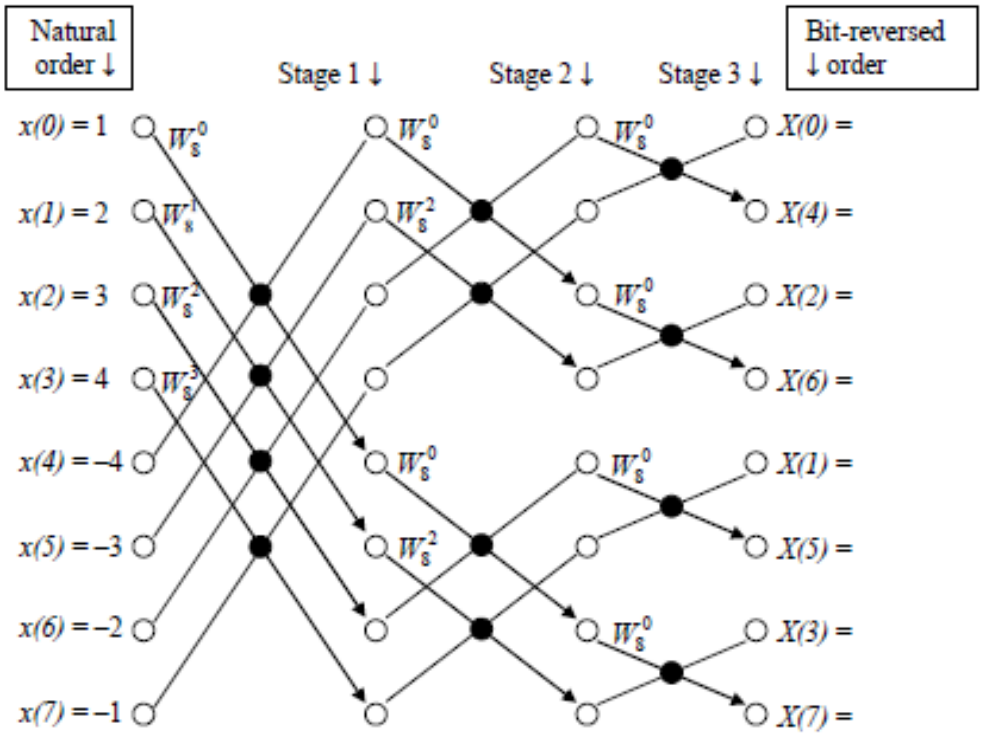
$W_8^0 = 1$ $\qquad$ $W_8^1 = e^{-j2\pi/8} = e^{-j\pi/4} = \frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}}$

$W_8^2 = \left(e^{-j2\pi/8}\right)^2 = e^{-j\pi/2} = -j$ $\qquad$ $W_8^3 = \left(e^{-j2\pi/8}\right)^3 = e^{-j3\pi/4} = -\frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}}$

One of the elementary computations is shown below:



The signal flow graph follows:

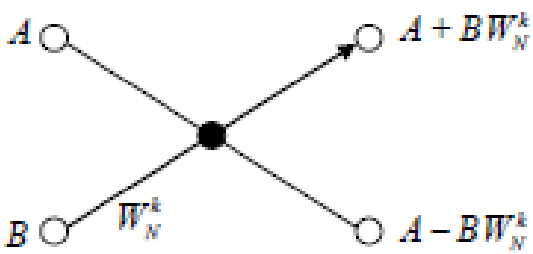


The DFT is

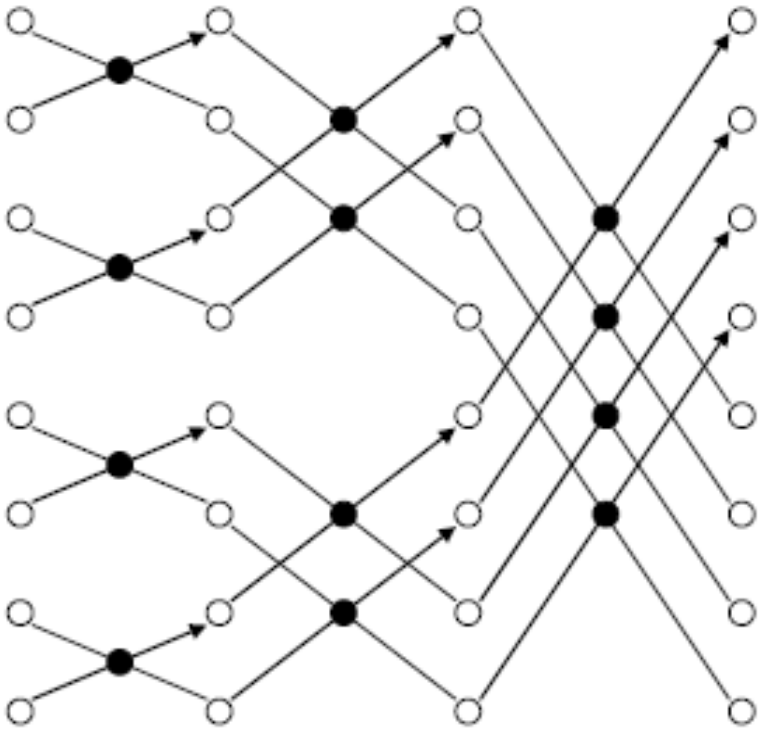
X(k) = {0, (5 – j12.07), (–4 + j4), (5 – j2.07), –4, (5 + j2.07), (–4 – j4), (5 + j12.07)}

(DIT Template)

The elementary computation (Butterfly):

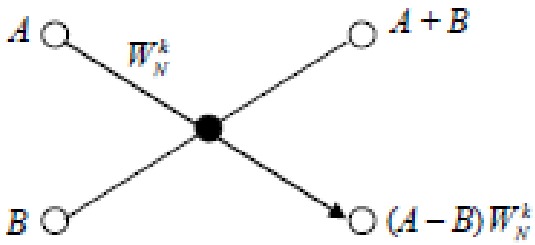


The signal flow graph:

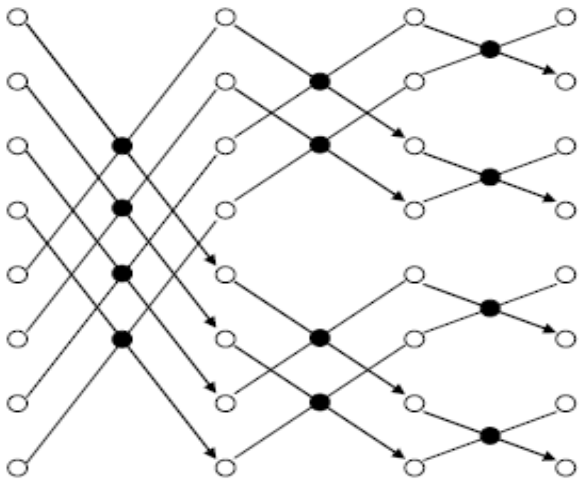


(DIF Template)

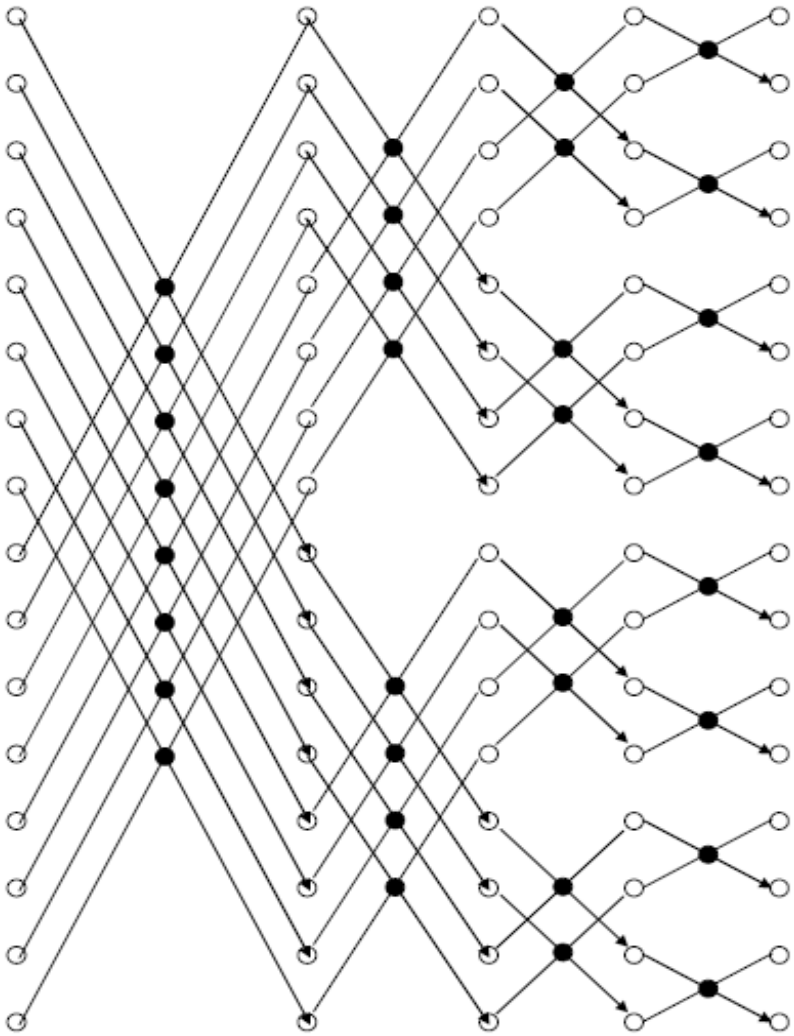
The elementary computation (Butterfly):



The signal flow graph:



16-point DIF FFT



4. 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

$$Nx^*(n) = \sum_{k=0}^{N-1} X^*(k)W_N^{kn}$$

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 3: 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.

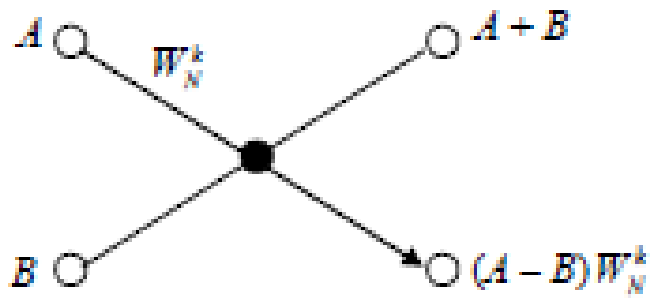
Solution:

This is an 8-point IDFT. The 8-point twiddle factors are, as calculated earlier,

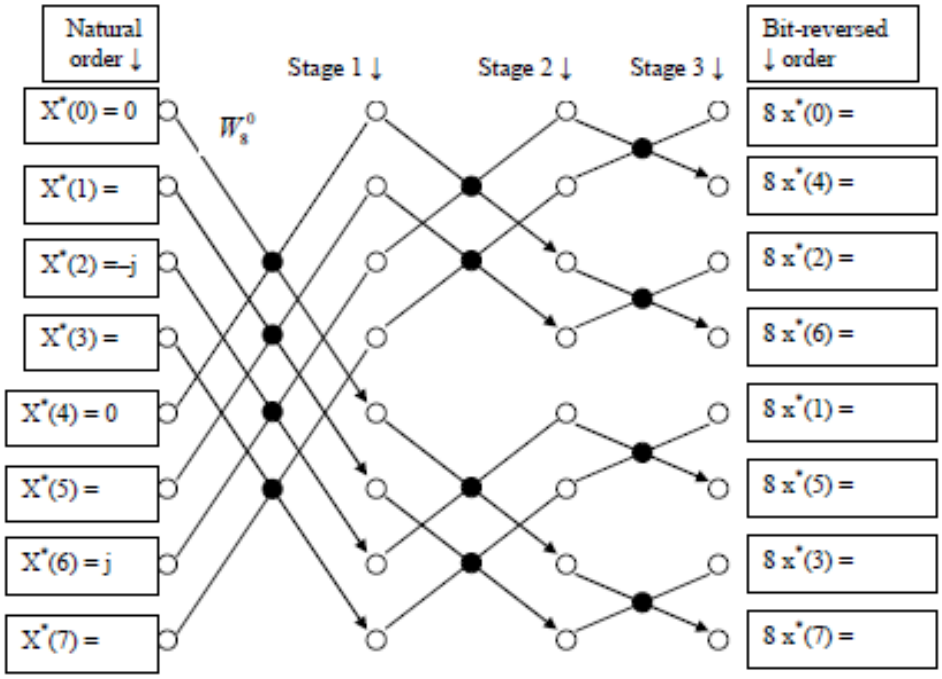
$$W_8^0 = 1 \qquad W_8^1 = e^{-j2\pi/8} = e^{-j\pi/4} = \frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}}$$

$$W_8^2 = \left(e^{-j2\pi/8}\right)^2 = e^{-j\pi/2} = -j \qquad W_8^3 = \left(e^{-j2\pi/8}\right)^3 = e^{-j3\pi/4} = -\frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}}$$

The elementary computation (Butterfly) is shown below:



The signal flow graph follows:



The output at stage 3 gives us the values $\{8x^*(n)\}$ 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:

$\{8x^*(n)\}_{normalorder}$ = {2, –6.24, 4, 6.24, –2, 2.24, –4, –2.24}

$$x(n) = \left\{\frac{1}{4}, \frac{-6.24}{8}, \frac{1}{2}, \frac{6.24}{8}, \frac{1}{4}, \frac{2.24}{8}, -\frac{1}{2}, \frac{-2.24}{8}\right\}$$

$$x(n) = \{0.25, -0.78, 0.5, 0.78, -0.25, 0.28, -0.5, -0.28\}$$

Example 4: 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)}

5. APPLICATIONS OF FFT ALGORITHMS:

1. Efficient Computation of the DFT of Two Real Sequences

The FFT algorithm is designed to perform complex multiplications and additions, even though the input data may be real valued. The basic reason for this situation is that the phase factors are complex and hence, after the first stage of the algorithm, all variables are basically complex-valued. In view of the fact that the algorithm can handle complex -valued input sequences, we can exploit this capability in the computation of the DFT of two real-valued sequences. Suppose that $x_1(n)$ and $x_2(n)$ are two real-valued sequences of length N, and let x(n) be a complex-valued sequence defined as

$$x(n) = x_1(n) + jx_2(n) \qquad 0 \le n \le N-1$$

The DFT operation is linear and hence the DFT of x(n) can be expressed as

$$X(k) = X_1(k) + jX_2(k)$$

The sequences $x_1(n)$ and $x_2(n)$can be expressed in terms of x(n) as follows:

$$x_1(n) = \frac{x(n) + x^*(n)}{2}$$

$$x_2(n) = \frac{x(n) - x^*(n)}{2j}$$

Hence the DFTs of $x_1(n)$ and $x_2(n)$ are

$$X_1(k) = \frac{1}{2}\{DFT[x(n)] + DFT[x^*(n)]\}$$

$$X_2(k) = \frac{1}{2j}\{DFT[x(n)] - DFT[x^*(n)]\}$$

Recall that the DFT of $x^*(n)$ is $X^*(N-k)$. Therefore

$$X_1(k) = \frac{1}{2}[X(k) + X^*(N-k)]$$

$$X_2(k) = \frac{1}{2j}[X(k) - X^*(N-k)]$$

Thus, by performing a single DFT on the complex-valued sequence x(n), we have obtained the DFT of the two real sequences with only a small amount of additional computation that is involved in computing $X_1(k)$ and $X_2(k)$from *X(k).*

2. Efficient Computation of the DFT of a 2N-Point Real Sequence

Suppose that g(n) is a real-valued sequence of 2N points. We now demonstrate how to obtain the 2N-point DFT of g(n) from computation of one N-point DFT involving complex-valued data. First, we define

$$x_1(n) = g(2n)$$

$$x_2(n) = g(2n+1)$$

Thus we have subdivided the 2N-point real sequence into two N-point real sequences. Now we can apply the method described in the preceding section.
Let x(n) be the N-point complex-valued sequence

$$x(n) = x_1(n) + jx_2(n)$$

From the results of the preceding section, we have

$$X_1(k) = \frac{1}{2}[X(k) + X^*(N-k)]$$

$$X_2(k) = \frac{1}{2j}[X(k) - X^*(N-k)]$$

Finally, we must express the 2N-point DFT in terms of the two N-point DFTs, $X_1(k)$ and $X_2(k)$. To accomplish this, we proceed as in the decimation-in-time FFT algorithm, namely,

$$G(k) = \sum_{n=0}^{N-1} g(2n)W_{2N}^{2nk} + \sum_{n=0}^{N-1} g(2n+1)W_{2N}^{(2n+1)k}$$

$$= \sum_{n=0}^{N-1} x_1(n)W_N^{nk} + W_{2N}^{k}\sum_{n=0}^{N-1} x_2(n)W_N^{nk}$$

Consequently,

$$G(k) = X_1(k) + W_2^k N X_2(k) \qquad k = 0,1,\ldots.,N-1$$

$$G(k+N) = X_1(k) - W_2^k N X_2(k) \qquad k = 0,1,\ldots.,N-1$$

Thus we have computed the DFT of a 2N-point real sequence from one N-point DFT and some additional computation.

6. The Chirp-z Transform Algorithm:

The DFT of an N-point data sequence *x(n)* has been viewed as the z-transform of $x_1(n)$ evaluated at N equally spaced points on the unit circle. It has also been viewed as N equally spaced samples of the Fourier transform of the data sequence *x(n).* In this section we consider the evaluation of *X(z)* on other contours in the z-plane, including the unit circle.

Suppose that we wish to compute the values of the z-transform of *x(n)* at a set of points $\{z_k\}$. Then,

$$X(z_k) = \sum_{n=0}^{N-1} x(n) z_k^{-n} \qquad k = 0,1,\dots,L-1$$

For example, if the contour is a circle of radius r and the z_k are N equally spaced points, then

$$z_k = re^{j2\pi kn/N} \qquad k = 0,1,2,\dots,N-1$$

$$X(z_k) = \sum_{n=0}^{N-1} [x(n) r^{-n}] e^{-j2\pi kn/N} \qquad k = 0,1,2,\dots,N-1$$

In this case the FFT algorithm can be applied on the modified sequence $x(n)r^{-n}$.

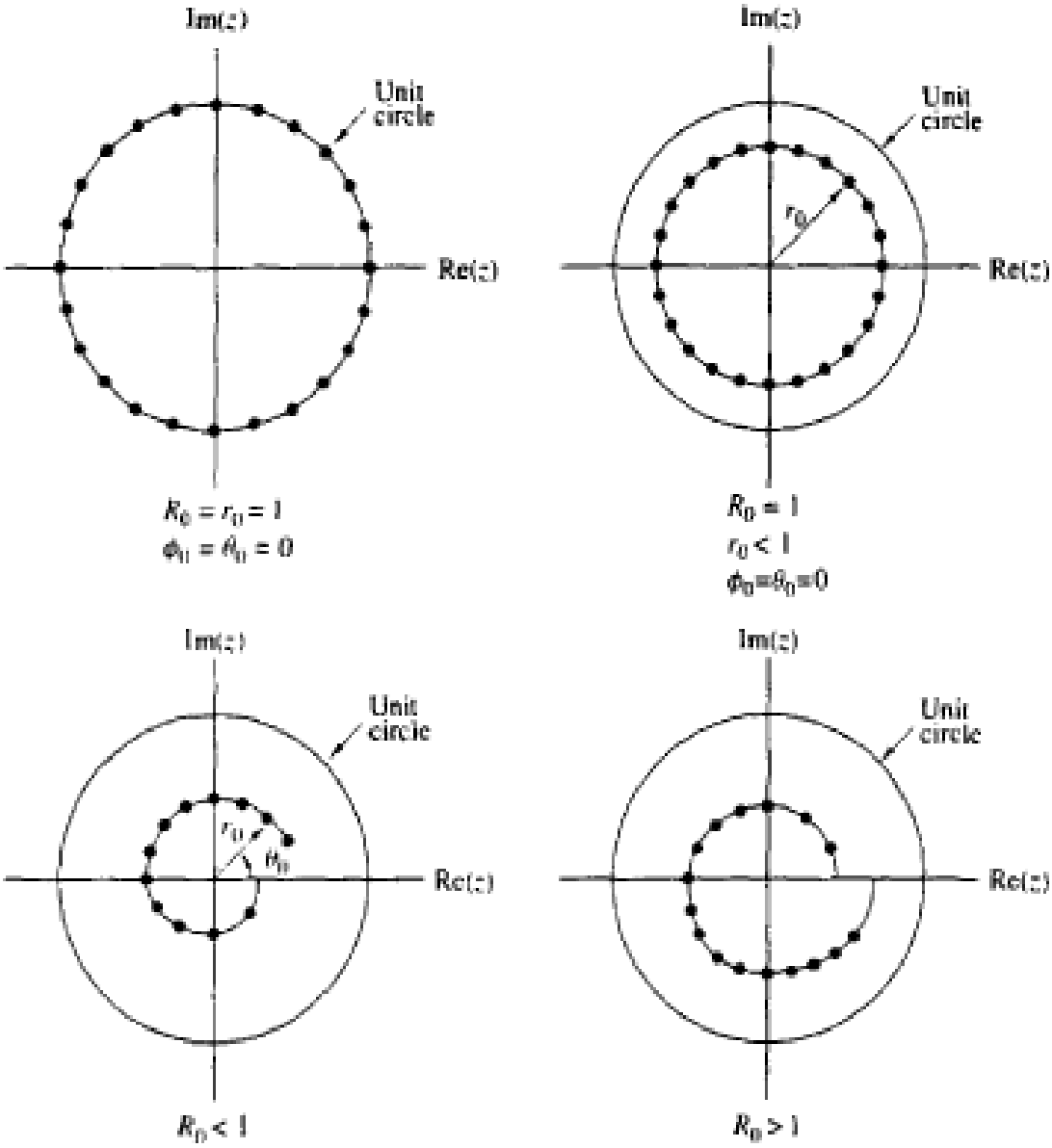
More generally, suppose that the points z_k in the z-plane fall on an arc which begins at some point

$$z_0 = r_0 e^{j\Theta_0}$$

and spirals either in toward the origin or out away from the origin such that the points z_k are defined as

$$z_k = r_0 e^{j\Theta_0} \left(R_0 e^{j\phi_0}\right)^k \qquad k = 0,1,\dots,L-1$$

Note that if $R_0 < 1$, the points fall on a contour that spirals toward the origin and if $R_0 > 1$, the contour spirals away from the origin. If Ro = 1, the contour is a circular arc of radius r_0. If $r_0 = 1$ and $R_0 = l$, the contour is an arc of the unit circle. The latter contour would allow us to compute the frequency content of the sequence x(n) at a dense set of L frequencies in the range covered by the arc without having to compute a large DFT, that is, a DFT of the sequence x(n) padded with many zeros to obtain the desired resolution in frequency. Finally, if $r_0 = R_0 = 1$, = 0, $\Theta_0 = 0$, $\phi_0 = 2n / N$, and L = N, the contour is the entire unit circle and the frequencies are those of the DFT.



When points $\{z_k\}$ are substituted into the expression for the z transform, we obtain

$$X(z_k) = \sum_{n=0}^{N-1} x(n) z_k^{-n}$$

$$= \sum_{n=0}^{N-1} x(n) \left(r_0 e^{j\theta_0}\right)^{-n} V^{-nk}$$

where, by definition, $V = R_0 e^{j\phi_0}$

We can express the above equation in the form of a convolution, by noting that

$$nk = \frac{1}{2}[n^2 + k^2 - (k-n)^2]$$

$$X(\mathrm{z_k}) = V^{-k^2/2} \sum_{n=0}^{N-1} \left[x(n)\left(\mathrm{r}_0 e^{j\Theta_0}\right)^{-n} V^{-n^2/2}\right] V^{(k-n)^2/2}$$

Let us define a new sequence g(n) as

$$g(n) = x(n)\left(\mathrm{r}_0 e^{j\Theta_0}\right)^{-n} V^{-n^2/2}$$

Then,

$$X(\mathrm{z_k}) = V^{-k^2/2} \sum_{n=0}^{N-1} g(n) V^{(k-n)^2/2}$$

The summation in the above expression can be interpreted as the convolution of the sequence g(n) with the impulse response h(n) of a filter, where

$$h(n) = V^{n^2/2}$$

Hence,

$$X(\mathrm{z_k}) = V^{-k^2/2} y(k) = \frac{y(k)}{h(k)} \qquad k = 0,1, \dots, L-1$$

Where y(k) is the output of the filter

$$y(k) = \sum_{n=0}^{N-1} g(n) h(k-n) \qquad k = 0,1, \dots, L-1$$

We observe that both h(n) and g(n) are complex-valued sequences. The sequence h(n) with $R_0 = 1$ has the form of a complex exponential with argument $wn = n^2\phi_0/2 = (n\phi_0/2)n$. The quantity $n\phi_0/2$ represents the frequency of the complex exponential signal, which increases linearly with time. Such signals are used in radar systems and are called chirp signals. Hence the z-transform evaluated is called the chirp-z transform.

MODULE 4:

Structures for FIR and IIR Systems:

Structure for FIR Systems:

In general a FIR system is described by the difference equation

$$y(n) = \sum_{k=0}^{M-1} b_k x(n-k)$$

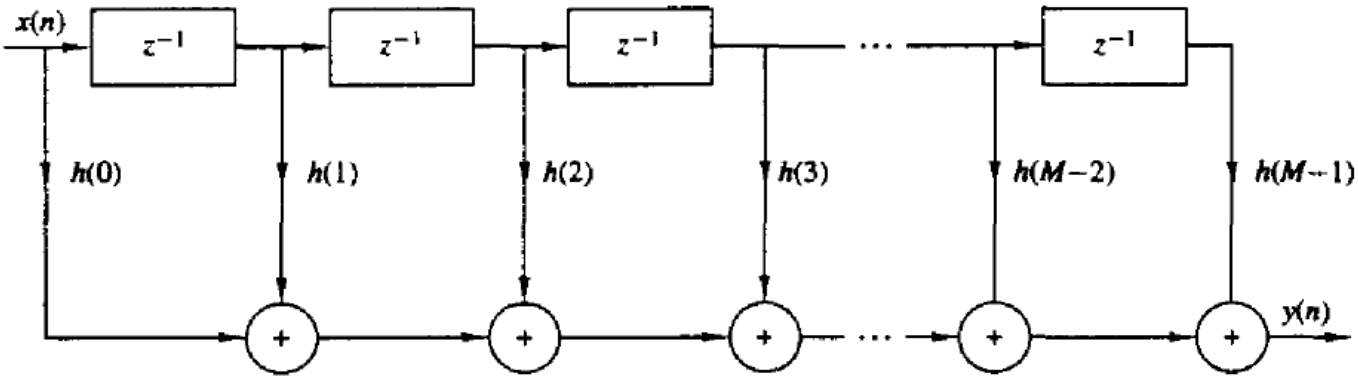
Or equivalently, by the system function

$$H(z) = \sum_{k=0}^{M-1} b_k z^{-k}$$

1. Direct-Form Structure:

The direct-form realization follows the convolution summation

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



Direct form realisation of FIR system

We observe that this structure requires M-1 memory locations for storing the M-1 previous inputs, and has a complexity of M multiplications and M-1 additions per output point. Since the output consists of a weighted linear combination of M-1 past values of the input and the weighted current value of the input, the structure in above figure, resembles a tapped delay line or a transversal system consequently, the direct-form realization is often called a transversal or tapped-delay-line filter.

2. <u>Cascade-Form Structures:</u>

The cascade realization follows naturally from the system function given by

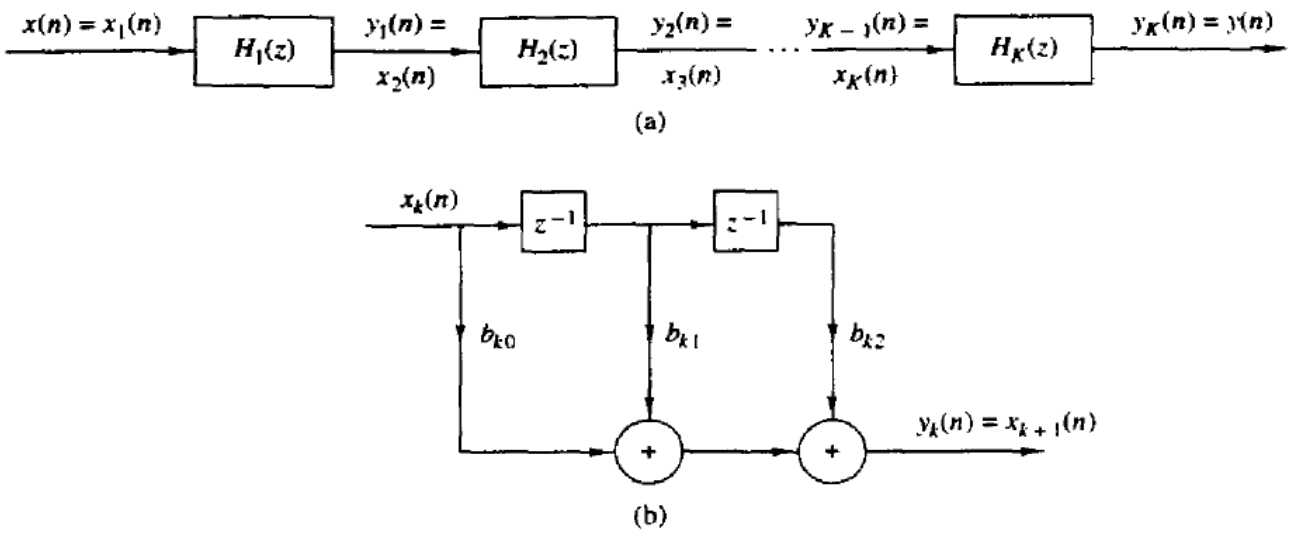
$$H(z) = \sum_{k=0}^{M-1} b_k z^{-k}$$

It is simple matter to factor H(z) into second order FIR system so that

$$H(z) = \prod_{k=1}^{M} H_k(z)$$

Where $H_k(z)=b_{k0}+b_{k1}z^{-1}+b_{k2}z^{-2}$, k=1,2,3……k

And K is the integer part of (M + l) /2. The filter parameter b_0 may be equally distributed among the K filter sections, such that $b_0 = b_{10}b_{20}\cdots b_{K0}$ or it may be assigned to a single filter section. The zeros of H (z) are grouped in pairs to produce the second-order FIR systems. It is always desirable to form pairs of complex-conjugate roots so that the coefficients $\{b_{ki}\}$ are real valued. On the other hand, real-valued roots can be paired in any arbitrary manner. The cascade-form realization along with the basic second-order section is shown below.



Cascade Realisation of a FIR system

Design of Digital Filters:

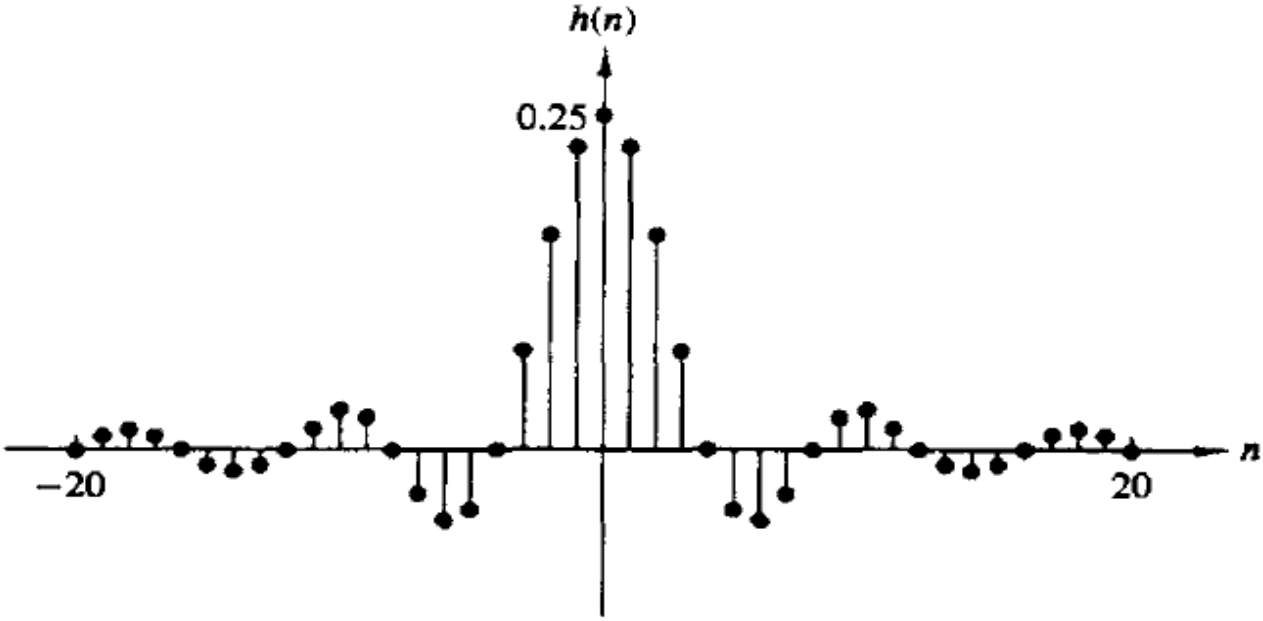
Causality and Its Implications:

Let us consider the issue of causality in more detail by examining the impulse response h(n) of an ideal low pass filter with frequency response characteristic

$$H(w)=\{ \begin{matrix} 1 & |\omega| \le \omega_c \\ 0 & \omega_c < \omega < \pi \end{matrix}$$

The impulse response of the filter is

$$h(n)=\{ \begin{matrix} \frac{\omega_c}{\pi}, n=0 \\ \frac{\omega_c}{\pi}\frac{\sin \omega_c n}{\omega_c n}, n \ne 0 \end{matrix}$$



Unit sample response of an ideal low pass filter

A plot of h{n) for $w_c = \pi/4$ is illustrated in the above figure. It is clear that the ideal low pass filter is noncausal and hence it cannot be realized in practice.

One possible solution is to introduce a large delay n_0 in h(n) and arbitrarily to set h(n)=0 for $n < n_0$. However, the resulting system no longer has an ideal frequency response characteristic. Indeed, if we set h(n) = 0 for $n < n_0$, the Fourier series expansion of H(w) results in the Gibbs phenomenon.

Paley-Wiener Theorem:

If h(n) has finite energy and h(n) = 0 for n < 0, then

$$\int_{-\pi}^{\pi} |\ln|H(\omega)||d\omega < \infty$$

Conversely, if |H(ω)| is square integrable and if the integral in the above equation is finite, then we can associate with |H(ω)| a phase response $\Theta(\omega)$, so that the resulting filter with frequency response $H(\omega)= |H(\omega)| e^{j\theta(\omega)}$ is causal.

One important conclusion that we draw from the Paley-Wiener theorem is that the magnitude function |H(ω)| can be zero at some frequencies, but it can't be zero over any finite band of frequencies, since the integral then becomes infinite. Consequently any ideal filter is noncausal.

Apparently causalty imposes some tight constraints on a linear time invariant system. In addition to the Paley-Wiener condition causalty also implies a strong relation between $H_R(\omega)$ and $H_I(\omega)$, the real and imaginary components of the frequency response H(ω).To illustrate this dependence we decompose h(n).That iseven and an odd sequence, that is

$$H(n)=h_e(n)+h_o(n)$$

Where $h_e(n)= \frac{1}{2}[h(n)+h(-n)]$ and $\frac{1}{2}[h(n)-h(-n)]$

Now, if h(n) is causal ,it is possible to recover h(n) from its even part $h_e(n)$ for $0\leq n\leq\infty$ or from its odd component $h_o(n)$ for $1\leq n\leq\infty$.

Indeed, it can be easily seen that

$$h(n)=2h_e(n)u(n)-h_e(0)\delta(n) \qquad n\geq 0$$

and

$$h(n)=2h_o(n)u(n)-h_o(0)\delta(n) \qquad n\geq 1$$

Since h_0 (n) = 0 for n = 0, we cannot recover h(0) from h_0 (n) and hence we also must know h(0). In any case, it is apparent that h_0 (n) = h_e(n) for n > 1, so there is a strong relationship between h_0 (n) and h_e(n).

If h (n) is absolutely summable (i.e., BIBO stable), the frequency response H(w) exists, and

$$H(\omega) = H_R(\omega) + jH_I(\omega)$$

In addition, if h(n) is real valued and causal, the symmetry properties of the Fourier transform imply that

$$h_e(n) \overset{F}{\longleftrightarrow} H_R(\omega)$$

$$h_o(n) \overset{F}{\longleftrightarrow} H_I(\omega)$$

Since h(n) is completely specified by $h_e(n)$, it follows that H(ω) is completely determined if we know $H_R(\omega)$.alternatively H(ω) is completely determined from $H_I(\omega)$ and h(0).In short $H_R(\omega)$ and $H_I(\omega)$ are independent and cannot be specified independently if the system is causal. Equivalently the magnitude and phase responses of a causal filter are interdependent and hence cannot be specified independently.

<u>Design of Linear Phase FIR filters using different windows:</u>

In many cases a linear phase characteristics is required through the passband of the filter. It can be shown that causal IIR filter cannot produce a linear phase characteristics and only special forms of causal FIR filters can give linear phase. If {h[n]} represents the impulse response of a discrete time linear system a necessary and sufficient condition for linear phase is that {h[n]} have finite duration N, that it be symmetric about its midpoint, i.e.

$$h[n] = h[N-1-n], \quad n = 0, 1, 2, \ldots (N-1)$$

$$\begin{aligned} H(e^{jw}) &= \sum_{n=0}^{N-1} h[n]e^{-j\omega n} \\ &= \sum_{n=0}^{\frac{N}{2}-1} h[n]^{-j\omega n} + \sum_{n=N/2}^{N-1} h[n]e^{-j\omega n} \\ &= \sum_{n=0}^{N/2-1} h[n]e^{-j\omega n} + \sum_{m=0}^{N/2-1} h[m]e^{-j\omega}(N-1-m) \end{aligned}$$

For N even, we get

$$H(e^{j\omega}) = e^{-j\omega(N-1)/2} \sum_{n=0}^{N/2-1} 2h[N]\cos(\omega(n-(N-1)/2))$$

For N odd

$$H(e^{j\omega}) = e^{-j\omega(N-1)/2} \left\{ h[\frac{N-1}{2}] + \sum_{n=0}^{\frac{N-3}{2}} 2h[n]\cos[\omega(n-\frac{N-1}{2})] \right\}$$

For N even we get a non-integer delay, which will cause the value of the sequence to change.

One approach to design FIR filters linear phase is to use windows. The easiest way to obtain FIR filter is to simply truncate the impulse response of an IIR filter. If $\{h_d[n]\}$is the impulse response of the designed FIR filter then the fir filter with impulse response {h[n]} can be obtained as follows.

$$H[n]=\{ \begin{matrix} hd[n], N_1 \le n \le N_2 \\ 0, otherwise \end{matrix}$$

This can be thought of as being formed by a product of $\{h_d[n]\}$ and a window function {w[n]}

$\{h[n]\}= \{h_d[n]\} \{w[n]\}$

where {w[n]} is the window function.

Using modulation property of fourier transform

$$H(e^{j\omega})= \frac{1}{2\pi} [H_d(e^{j\omega}) \otimes -w(e^{j\omega})]$$

In general for smaller N values spreading of main lobe more, and for larger N narrower thr main lobe and $\left| H(e^{j\omega}) \right|$ comes closer to $\left| H_d(e^{j\omega}) \right|$.Much work has been done on adjusting {w[n]} to satisfy certain main lobe and side lobe requirements .Some of the commonly used windows are given below-

(a) Rectangular Window

$$W_R(n)=\{ \begin{matrix} 1, 0 \le n \le N-1 \\ 0, otherwise \end{matrix}$$

(b) Bartlett (Triangular)

$$W_B(n)=\{ \begin{matrix} \frac{2n}{N-1}, 0 \le n \le (N-1)/2 \\ 2-\frac{2n}{N-1}, (N-1)/2 \le n \le N-1 \\ 0, elsewhere \end{matrix}$$

(c) Hanning Window

$$W_{Han}(n)=\begin{cases} \frac{1-\cos[2\pi n/(N-1)]}{2}, & 0 \le n \le N-1 \\ 0, & otherwise \end{cases}$$

(d)Blackman Window

$$W_{Bl}(n)=\begin{cases} .42-.5\cos\left[2\pi n/N-1\right]+.08\cos\left[4\pi n/N-1\right], & 0 \le n \le N-1 \\ 0, & otherwise \end{cases}$$

(e)Kaiser Window

$$W_K(n)=\begin{cases} \frac{I_0\omega_a\left[\left(\frac{N-1}{2}\right)^2-\left(n-\frac{N-1}{2}\right)^2\right]^{\frac{1}{2}}}{I_0\left\{w_a(\frac{N-1}{2})\right\}}, & 0 \le n \le N-1 \\ 0, & otherwise \end{cases}$$

Where $I_0(x)$ is the modified Zero Order Bessel Function of the first kind.

The Transition width and the minimum stopped attenuation for different windows are listed below-

Window	Transition Width	Minimum stopband attenuation
Rectangular	$4\pi/N$	-21db
Bartlett	$8\pi/N$	-25dB
Hanning	$8\pi/N$	-44dB
Hamming	$8\pi/N$	-53dB
Blackman	$12\pi/N$	-74 dB
Kaiser	variable	variable

We first choose a window that satisfies the minimum attenuation and the bandwidth that allows us to choose the appropriate value of N. Actual frequency response characteristics are then calculated and we check the requirments are met or not

<u>Design of IIR Filters:</u>

There are two methods for design the IIR filter.

1. Impulse Invariant Method

2. Bilinear Transformation Method

1. <u>Filter design by impulse invariance:</u>

Here the impulse response h[n] of the desire discrete time system is proportional to equally spaces samples of the continuous time filter i.e,

$H[n]=T_d h_a(nT_d)$

Where T_d represents a sample interval.Since the specification of the filter are given in discrete time domain it turns out that T_d has no role to play in design of the filter. From the sampling theorem the frequency response of the discrete time filter is given by

$$H(e^{j\omega})= \sum_{k=-\infty}^{\infty} H_a(j\frac{\omega}{T_d}+j\frac{2\pi k}{T_d})$$

Since any practical continuous time filter is not strictly band limited there is some aliasing. However if the continuous time filter approaches zero at high frequency the aliasing may be negligible. Then the frequency response of the discrete time filter is

$$H(e^{j\omega})\approx \sum_{k=-\infty}^{\infty} H_a(j\frac{\omega}{T_d}), |\omega| \le \pi$$ Type equation here.

We first convert digital filter specifications to continuous time filter specifications. Neglecting aliasing we get $H_a(j\Omega)$ specification by applying the relation $\Omega= \omega/T_d$. Where $H_a(j\Omega)$ is transferred to the designed filter H(z).

Let us assume that the poles of the continuous time filter are simple, then

$$H_a(s)= \sum_{k=1}^{N} \frac{A_k}{s-s_k}$$

The corresponding Impulse response is $h_a(t)=\{ \sum_{k=1}^{N} A_k e^{s_k t}, t \ge 0$
$0, t < 0$

Then $h[n]=T_d h_a(nT_d)= \sum_{k=1}^{N} T_d A_k e^{s_k n T_d} u[n]$

The system function function for this is $H(z)= \sum_{k=1}^{N} \frac{T_d A_k}{1-e^{s_k T_d} z^{-1}}$

We see that a pole at $s= s_k$ in the s-plane is transferred to a pole at $z= e^{s_k T_d}$ in the z-plane. If the continuous time filter is stable i.e $Re\{s_k\}<0$, then the magnitude of $e^{s_k T_d}$ will be less than 1.So the pole will be inside the unit circle. Thus the causal discrete filter is stable. The mapping of zero is not so straight forward.

Bilinear Transformation:

This technique avoids the problem of aliasing by mapping jΩ axis in the s-plane to one revolution of unit circle in the z-plane. If $H_a(s)$ is the continuous time transfer function the discrete time transfer function is detained by replacing s with

$$S= \frac{2}{T_d}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)$$

From which we get $z= \frac{1+(T_d/2)s}{1-(T_d/2)s}$

Substituting s= σ+jΩ , we get $z=\frac{1+\sigma\frac{T_d}{2}+j\frac{\Omega T_d}{2}}{1-\sigma\frac{T_d}{2}-j\frac{\Omega T_d}{2}}$

If σ<0, it is then magnitude of the real part in the denominator is more than that of the numerator and so $|z|$<1. Similarly if σ>0 then $|z|$>1 for all Ω.Thus pole in the left half of the s-plane will get mapped to the poles inside the unit circle in z-plane. If σ=0 then

$$z=\frac{1+j\frac{\Omega T_d}{2}}{1-j\frac{\Omega T_d}{2}}$$

so $|z|$=1,writing z=$e^{j\omega}$ we get

$$e^{j\omega}=\frac{1+j\frac{\Omega T_d}{2}}{1-j\frac{\Omega T_d}{2}}$$

Rearranging we get $j\frac{\Omega T_d}{2}=\frac{e^{j\omega}-1}{e^{j\omega}-1}=\frac{e^{j\omega/2}(e^{+j\omega/2}-e^{-j\omega/2})}{e^{j\omega/2}(e^{+j\omega/2}+e^{-j\omega/2})}=j\frac{\sin\omega/2}{\cos\omega/2}$

Or $\Omega=\frac{2}{T_d}\tan\omega/2$ or $\omega=2\tan^{-1}\frac{\Omega T_d}{2}$.